ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A SUPPORT DECISION SYSTEM FOR PREDICTING RATING VALUES OF PREPRODUCTION TV CONTENT: AN EXPLAINABLE MACHINE LEARNING APPROACH

M.Sc. THESIS

Burak BATIBAY

Department of Mathematical Engineering

Mathematical Engineering Programme

APRIL 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A SUPPORT DECISION SYSTEM FOR PREDICTING RATING VALUES OF PREPRODUCTION TV CONTENT: AN EXPLAINABLE MACHINE LEARNING APPROACH

M.Sc. THESIS

Burak BATIBAY
(509211203)

Department of Mathematical Engineering

Mathematical Engineering Programme

Thesis Advisor: Prof. Dr. Atabey KAYGUN

APRIL 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAYINLANMAMIŞ TV İÇERİĞİNİN REYTİNG DEĞERİNİN TAHMİN EDİLEBİLMESİ İÇİN KARAR DESTEK SİSTEMİ: BİR AÇIKLANABİLİR MAKİNE ÖĞRENİMİ YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Burak BATIBAY
(509211203)

Matematik Mühendisliği

Matematik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Atabey KAYGUN

NİSAN 2024

Burak BATIBAY, a M.Sc. student of ITU Graduate School student ID 509211203 successfully defended the thesis entitled "A SUPPORT DECISION SYSTEM FOR PREDICTING RATING VALUES OF PREPRODUCTION TV CONTENT: AN EXPLAINABLE MACHINE LEARNING APPROACH", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Atabey KAYGUN** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Özgür MARTİN** Mimar Sinan Fine Arts University	.....
	Assoc. Prof. Gül İNAN Istanbul Technical University	.....

Date of Submission : 2 April 2024
Date of Defense : 22 April 2024

Dedicated to my family

FOREWORD

I would like to express my sincere gratitude to my esteemed supervisor, Prof. Dr. Atabey KAYGUN, who shared valuable knowledge throughout this study and patiently and attentively assisted me whenever I needed it. His guidance and contributions to my academic life, motivation, work energy, continuous encouragement, trust, and patience are highly appreciated. He consistently supported me in allowing this thesis to truly reflect my work, while also guiding me in the right direction.

Finally, I would like to express my gratitude to my family, who have provided me with all kinds of material and spiritual support throughout my life.

April 2024

Burak BATIBAY
(M.Sc Student)

TABLE OF CONTENTS

ABBREVIATIONS

TV	**:** Television
IMDB	**:** Internet Movie Database
TRT	**:** Turkish Radio and Television Corporatio
RTÜK	**:** Radio and Television Supreme Council
TVR	**:** Total Views Rate
ATS	**:** Average Television Viewer Time
GRP	**:** Gross Viewing Rate
TİAK	**:** Television Viewing Research Committee
MGM	**:** Turkish State General Directorate of Meteorology
SVR	**:** Support Vector Regressior
KNN	**:** K-Nearest Neighbors
LGBM	**:** Light Gradient Boosting Machines
XGBoost	**:** eXtreme Gradient Boosting
MAE	**:** Mean Absolute Error
MAPE	**:** Mean Absolute percentage Error
MSE	**:** Mean Squared Error
RMSE	**:** Root Mean Squared Error
AWS	**:** Amazon Web Services
SHAP	**:** SHapley Additive exPlanations
LASSO	**:** Least Absolute Shrinkage and Selection Operator
SQL	**:** Structured Query Language
ML	**:** Machine Learning
AI	**:** Artificial Intelligent
ODBC	**:** Open Database Connectivity
ETL	**:** Extract Transform Load

LIST OF TABLES

LIST OF FIGURES

A SUPPORT DECISION SYSTEM FOR PREDICTING RATING VALUES OF PREPRODUCTION TV CONTENT: AN EXPLAINABLE MACHINE LEARNING APPROACH

SUMMARY

The motivation behind this study is to create a decision support system based on machine learning, resulting from the rapid digitization of the media industry, which has led to numerous content options for publication. The aim is to forecast the rating values that publication content will receive under different components and conditions before they are even published. Strategic project planning during specific publication seasons requires predicting the audience's value through various options. Presenting the expected outcomes based on rating metrics facilitates comprehensive evaluations addressing budgetary concerns and strategic considerations. Consequently, evaluating the predicted success of independently planned scenarios using rating metrics provides a concrete framework for various action plans. This study aims to assist publishers and producers in predicting the success probability of their content using machine learning techniques. Specifically, predicting the expected rating of a particular content publication involves understanding how this prediction is influenced by various conditions or circumstances. Ensuring consistency and comparability of relationships between different units supports the creation of budget action plans by internally comparing different options while scenarios are still on the table.

This study aims to provide publishers and producers with accurate tools to forecast the success of their content. Leveraging machine learning techniques, it seeks to reliably predict the expected ratings of broadcast content. This predictive capability is crucial for stakeholders, enabling them to understand viewer preferences and make informed decisions about content creation, scheduling, and resource allocation. Ultimately, the goal is to empower industry professionals to optimize strategies, improve content quality, and stay competitive in a dynamic media environment

The methods employed in this study initially involve examining relevant ratings, alongside analyzing series names from mainstream rating reports in the broadcast industry over the past seven years. Additionally, open-source data and social media data are utilized to enrich the dataset, which is then stored in a relational database for ongoing analysis. In the study, the models to be used were selected based on progressive-order logic and the metric values obtained from the model results. Subsequently, steps were taken to enhance and develop the selected ensemble boosting models. Finally, the developed model is integrated into a user-friendly interface, allowing for the interactive exploration of predicted outcomes based on specific parameters or user inputs. Using interpretable machine learning techniques, reports visualizing the predicted values obtained by the end user have been created, making the model outputs easier to interpret and understand.

YAYINLANMAMIŞ TV İÇERİĞİNİN REYTİNG DEĞERİNİN TAHMİN EDİLEBİLMESİ İÇİN KARAR DESTEK SİSTEMİ: BİR AÇIKLANABİLİR MAKİNE ÖĞRENİMİ YAKLAŞIMI

ÖZET

Günümüzde son yıllarda hızla artan dijitalleşmenin de etkisiyle yapay zekâ çağında makine öğrenimi ve otonom sistemler her alanda olduğu gibi yayın sektöründe de etkisini ve kapsam alanını artırmaktadır. Rekabete dayalı sistemlerde, gerçekleşmesi beklenen durumların önceden gerçeğe en yakın şekilde simultane edilmesi veya öngörülerde bulunulması stratejik ve bütçesel planlamaların yapılması adına oldukça önemlidir. Yayın sektöründeki alışkanlıkların giderek dijitalleşmeye uygun şekilde değişmesi, yayınlanacak içeriklerin de farklılaşmasına ve alternatiflerinin çoğalmasına yol açmaktadır. Özellikle yayın sezonu olarak adlandırılan dönemlerde proje olarak konulmuş bir senaryonun planlanmasının yapılması için farklı seçimler üzerinden alacağı reyting değerinin öngörülerek bu sonucun açıklanabilir düzeyde yapımcıya sunulması hem bütçesel olarak hem de stratejik olarak çok yönlü incelenmesine olanak sağlamaktadır. Dolayısıyla yayına girmesi beklenen bir senaryonun öngörülen başarı durumu, reyting metriği üzerinden değerlendirilerek çeşitli eylem planlarının oluşmasına somut bir ortam hazırlar.

Bir yayın içeriğinin yayın sırasında ve ölçüm zamanında rassal ve gizli olarak dağıtılmış ölçüm cihazlarından aldığı paya reyting değeri denir. Söz konusu ölçüm cihazlarının sadece açık olan televizyon başına oranına TVR değeri, TVR değerinden aldığı paya ise paylaşım (share) değeri denir. Dolayısıyla başarı metriği olarak esas alınan reyting tek başına değil, TVR ve share değerleri ile ilişkilendirilerek incelenmektedir. Ancak günümüzdeki reyting ölçümleri sadece reyting yani içeriklerin belirli bir süre yayınlanması sırasında izleyici reaksiyonları ile ölçülebilmektedir. Bu sebeple günümüzdeki çalışmalar ve eksiklikler göz önünde bulundurulduğunda yayın içeriğinin sahip olacağı reyting değerinin, içerik yayınlanmadan tahmin edilmesini, tahmin edilen değerin hangi durumlardan veya koşullardan ne kadar ve nasıl etkilendiğinin belirlenmesini, söz konusu durumun sonraki aşamada rakip analizleriyle gerçeğe uygun şekilde yeniden değerlendirilmesini ve raporlanmasını sağlayan bir sisteme ihtiyaç vardır. Bu tez kapsamında gerçekleştirilen çalışmanın ele aldığı problemler, yayın içeriklerinin geçmiş reyting değerleri ile hem yayın kanalının hem de izleyici kitlesinin karakteristik özellikleri baz alınarak bir yayın içeriğinin ortalama izlenme oranlarının yani reyting değerlerinin içerik yayınlanmadan tahmin edilmesini, tahmin edilen değerin hangi durumlardan veya koşullardan ne kadar ve nasıl etkilendiğinin belirlenmesini, rakip analizi ile gerçeğe daha yakınsayan bir simülasyon ortamında değerlendirilmesini ve raporlanmasını sağlayan makine öğrenmesi temeline dayalı bir karar destek sistemi oluşturma ihtiyacını gidermektir.

Çalışma sırasında geçmiş yedi yıl boyunca kullanıcı alışkanlıklarını dikkate alarak, masadaki içeriğin yayına girmeden gelecekte alacağı reyting değerini tahmin

etmek, geçmiş derecelendirme değerlerini ve hem yayın kanalının hem de izleyici kitlesinin karakteristik özelliklerini göz önünde bulundurmayı gerektirir. Çalışmamız, temel olarak beklenen derecelendirme öngörüsünün yapılmasına ve tahmin edilen değerin çeşitli durumlar veya koşullar tarafından nasıl etkilendiğinin anlaşılabilir bir şekilde açıklanmasına odaklanmaktadır. Farklı birimler arasındaki ilişkilerin tutarlılığını ve karşılaştırılabilirliğini sağlamak, yayına girmeye hazırlanan içerik için beklenen derecelendirme değerinin tahminini destekler. Ayrıca, yapım şirketlerinin farklı seçenekleri içten içe karşılaştırarak bütçe eylem planlarını formüle etmesine olanak tanır.Masadaki senaryo değerini karşılaştırılabilir bir yapıda bilme, mevcut kaynakların optimal yönetimini sağlar ve performansı artırmak veya azaltmak için hareket ederek kaynak yönetiminin standardizasyonunu sağlar.

Yayın içeriğinin temel unsurları arasında yayın türü, oyuncu seçimi, yönetmen, yapım şirketi ve senarist seçimleri, yayın saati ve süresi, yayın dönemine girişin planlanan tarihi, izleyici oranlarını etkileyebilecek yayın günündeki beklenen hava koşulları ve farklı etkinlikler (milli maçlar vb.) yer almaktadır. Reyting tahmin etmek, ilk adımda, geçmiş yıllara dayalı olarak her kanal için ilişkiler ve hipotezler oluşturmayı içerir. Kullanıcı alışkanlıklarına ve her kanal için verilere dikkat edilir ve her kanal ve içerik türü için aylık, günlük veya saatlik aralıklara göre ortalama, toplam ve değişkenler gibi nicel özellikler göz önünde bulundurularak işlenir. Her oyuncunun sisteme giren geçmiş derecelendirme değerleri de bu süreçlere dahil edilir. Benzer bir süreç, her yapım şirketi, yönetmen veya senarist için de uygulanır. Her kullanıcı girdisi için değişen değerler, daha önce belirtilen temel hesaplanmış değerlere eklenir. Yüksek bir derecelendirme sonucunun tahmini, öne sürülen hipotezlerin doğruluğunu sağlamak için iç denetimlere ve istatistiksel testlere tabi tutulur; çünkü tahmin edilen ve gerçek değerler arasındaki sapma oranına orantılı olarak orantısız bir yüksek sonuç, önemli bütçe kayıplarına neden olabilir. İç denetimler ve istatistiksel testler aracılığıyla hipotezlerin doğruluğuna dikkat edilir, çünkü gerçek değerden sapmanın orantısına göre yüksek bir derecelendirme tahminine ulaşmak, önemli bütçe kayıplarına yol açabilir.

Çalışmamız üç ana bileşenden oluşmaktadır: veri toplama ve detaylı analiz, toplanan verilerden ve bu verilerden elde edilen bağlamsal bilgilerden yararlanarak model oluşturma ve son kullanıcılar için bir arayüz aracılığıyla modelin etkileşimli olarak dağıtılması. Analiz, son yedi yılda yayın endüstrisindeki ana akım derecelendirme raporlarından gelen ilgili derecelendirmeler, TVR ve Pay değerlerinin yanı sıra dizi adlarının incelenmesiyle başlar. Açık kaynaklı veri ve sosyal medya verileri, veri setini zenginleştirmek için kullanılmıştır. Son yedi senelik periyotta alınan yayın raporlarındaki günlük derecelendirme verileri diğer kaynaklarla uyumlu hale getirilmiştir. Farklı derecelendirme segmentlerindeki derecelendirme verilerinin analizi yapılmıştır.

Günlük reyting raporlarından alınan veriler, IMDB'nin verileriyle eşleştirilerek içeriğin başlangıç ve bitiş yılları, süresi, türü, IMDB derecesi, oyuncu kadrosu, senarist, yönetmen, yapımcı ve müzik yapımcısı gibi bilgiler elde edildi. Oyuncular ve dizilerle ilgili trendleri toplamak için Google Trends'ten oyuncuların arama sıklığı, Instagram'daki takipçi sayısı ve oyuncular hakkındaki yorumlar da dahil olmak üzere Ekşi Sözlük sitesinden alınan veriler toplandı, bu verilere duygusal analiz

de dahil edildi. Toplanan her veri parçası bir araya getirilerek ilişkisel veritabanı oluşturulmuştur. Toplanan bu veriler yapısal formlarda depolanır, birbirleriyle eşleştirilir ve her yeni içerik tanıtıldıktan sonra haftalık olarak güncellenir. Bu nedenle, bu ilişkisel veritabanını oluşturmak, güncel tutmak için otomasyon tanımlamak ve veriler arasındaki ilişkilere dayalı olarak kurulan ilişkiler ve hipotezler aracılığıyla doğruluğu ve tutarlılığı sürekli olarak incelemek hayati öneme sahiptir.

İçeriğin tahmini reyting değeri, farklı özelliklere sahip birçok bağımsız değişkene ve farklı koşullara bağlı olduğundan, kmodelin arka uç kısmında Ensemble Öğrenme temelli bir güçlendirme algoritmasının kullanılmasının gerekliliğini çıkardık. Ancak, model seçim aşaması, deneysel bazda hangi modelin neden uygulanıp uygulanmayacağını çıktılarından hareketle belirleyerek aşamalı bir şekilde model arama mantığı kullanılarak gerçekleştirildi. Dolayısıyla geleneksel regresyon modelleri ve birliktelik kurallarına bağlı modeller denendi ve neden uygun olmayabileceklerini analiz ederek, bu analiz sonuçlarına dayanarak en uygun model seçimi takip edildi. Uygun model seçiminin ardından seçilen modelin geliştirilmesi ve verilerden daha uygun ve anlamlı sonuçlar çıkartaracak şekilde eğitilmesi için iyileştirme yöntemleri gerçekleştirildi. Böylece en uygun sonuçları sağlayan model kurgusu oluşturuldu. Hata oranını minimize etmek için geliştirilmiş algoritmalar, eXtreme Gradient Boosting (XGBOOST) ve Light Gradient Boosting Machines (LGBM) hiperparametrelerle test edilmiş ve rafine edilmiştir. Bu süreçte, kullanılan verilerin iyileştirilmesi, yeni özelliklerin oluşturulması ve yeni veri kaynaklarının eklenmesi, modelin performansını artırmaya katkıda bulunmuştur. Benzer süreçler, rakip analizi simülasyonu için seçilen Catboost algoritması olan ikinci modelin rafine edilmesinde de uygulanmıştır.

Yapılan çalışmaların ardından, toplanan verilerin detaylı bir analizini içeren bir arayüz tasarlanmıştır. Arayüzde, her bir ayrıntı için, oyuncu, yapım şirketi, yönetmen, senarist veya kanal bazında, günlük derecelendirme raporları ve sosyal medya verileri, aylık veya yıllık görünümler de dahil olmak üzere çeşitli istenen zaman aralıkları için kapsamlı bir şekilde sunulmuştur. Denenen modeller aynı arayüze entegre edildi ve kullanıcıdan girdi almak ve modelin sonuçlarını buna göre yansıtmak için ikinci bir sayfa tasarlandı. Model çıktısının yanı sıra, arayüz, tahmin edilen değerin belirli parametreler veya kullanıcı girdileri tarafından ne kadar ve hangi ölçüde etkilendiğini açıklayan grafik temsilleri içermektedir. Bu grafikler, tahmin edilen sonucu etkileyen faktörlerin açıklayıcı bir görselleştirmesini sağlamaktadır. Oluşturulan arayüz sayesinde son kullanıcının arayüz üzerinden farklı kombinasyonları input olarak seçip farklı tahmin değerleri elde etmesi ve elde ettiği tahmin değerlerinin daha kolay yorumlanabilir ve açıklanabilir şekilde karşılaştırma ve analiz yapma imkanına sahip olacaktır.

1. INTRODUCTION

It is critical to produce content that reaches the highest viewing rate at the lowest cost in the media industry. While TV content is analyzed and planned months before broadcast, the question "Which cast, with what kind of content, under what conditions can get the best ratings?" Finding the answer to the question is a difficult process.

This study is a machine learning-based forecasting system that aims to predict the average viewership rates of broadcast content before it is aired by considering the historical ratings of television channels and the characteristic features of both the production companies and the audience. We seek to understand how much and in what way various circumstances and conditions influence the predicted value. The system also facilitates the determination and reporting of the extent to which the anticipated rating of broadcast content changes within a realistic simulation environment, through competitor analyses.

This study consists of two parts. The first part aims to predict viewership ratings based on attributes such as the producer, director, writer, cast, genre, and broadcast day of TV series developed for television channels. The second part of the study focuses on predicting the ratings that rival content is expected to achieve in a simulated environment created based on anticipated rating values. Machine learning techniques were employed using a dataset created with daily rating reports, IMDB data related to the content, and external data sources such as social media engagements, weather conditions, and event information that could impact ratings. Reports were generated using explainable machine learning techniques to facilitate easier interpretation of the obtained prediction results.

1.1 Purpose of Thesis

The main purpose of the study conducted within the scope of this thesis is to forecast the rating of the first episode of TV content that has not yet been broadcast. This

forecast is intended to support the decision-making process of the business unit of the production company by shedding light on potential revenues and expenses associated with the production. Therefore, this study should be considered as a decision support system.

Our work is primarily focused on predicting the expected rating and explaining in an answerable way how the predicted value is affected by various circumstances or conditions. Ensuring consistency and comparability of relationships between different entities supports predicting the expected rating value for future content. It also allows the production company to create budget action plans by internally comparing different options. Knowing the value of a comparable scenario while it is still on the table allows the optimal management of existing resources and the standardization of resource management by making moves that increase or decrease performance.

Another aim of this study is to enable production companies to assess content performance and shape future strategies effectively using machine learning techniques. These techniques analyze metrics like engagement rates, share counts, and comments to gauge the potential success of content. By comprehensively understanding these indicators, production companies can identify strengths, weaknesses, and patterns, enabling them to make informed decisions to enhance content, address vulnerabilities, and develop more effective strategies.

1.2 Data In Context

Television ratings bear significant importance within the media industry, serving as pivotal metrics aiding television stations and broadcasters in ascertaining the degree of viewership different programs receive. These ratings distinguish programs that garner considerable attention from those with comparatively lesser viewership. Moreover, television ratings stand as a crucial criterion for advertisers.

Within the advertising industry, television ratings hold immense value. Advertisers meticulously assess high-rated programs when allocating their advertising budgets. This strategic approach signifies that promoting products or services during a specific program represents a more efficient method of reaching a broader audience. So,

ratings are pivotal in shaping marketing strategies, facilitating advertisers in effectively engaging the intended viewership.

Ratings additionally contribute significantly to shaping the prospective programming agendas of television channels. Through an understanding of programs with superior ratings, channels are better equipped to make informed decisions regarding their future program assortments. Consequently, ratings function as a critical data source within the television industry, pivotal for formulating new programs and strategic program planning. Furthermore, ratings hold substantial significance for companies sponsoring television shows. Elevated ratings associated with a program significantly enhance the visibility of sponsors' products or services, capturing the attention of a broader audience. As a result, sponsors strategically invest in programs by meticulously considering the associated rating.

Ratings serve a pivotal role in dictating the content and formats of television programs. Television channels scrutinize high-rated programs to comprehend audience preferences, subsequently tailoring new content in alignment with these insights. Within this iterative process, ratings function as a compass for creators, producers, and screenwriters within the television industry. Programs achieving high ratings bolster the reputation of channels, while those with lower ratings may undergo review or potentially face replacement.

The preparation of television programs is carried out by production companies. While the types of programs on television have diversified significantly, there has also been a gradual merging of genres as well as the emergence of new program formats over time. Television broadcasting in Turkey has historically existed as a form of public service broadcasting for a long time. However, amendments to the radio-television legislation and constitutional reforms paved the way for the start of commercial broadcasting, allowing capital groups to own radio and television. These legislative changes led to a significant transformation in the range of programs offered on commercial television networks. In particular, significant budgets were allocated to the production of television series that reached the highest viewing rates, thus increasing the overall quality of television series [1].

Television programs encompass a broad array of content, such as films, serials, news segments, entertainment features, competitive shows, sports coverage, panel discussions, documentaries, children's programming, magazine-style shows, and various other program formats. These varied types of programs are produced by production companies, with a predominant association in Turkey being the creation of TV series. Publishers undertake the production of these diverse programs either internally or by outsourcing to specialized production companies. These production entities, focusing on specific program genres, market the content they produce to the television channels with whom they have contractual agreements.

The export of TV series has gained prominence in Turkey, delineating a shift from the export of physical goods to the export of serialized content. Serial export involves licensing the broadcasting rights of a series to foreign broadcasters for a specified duration in exchange for a fee. Consequently, the foreign broadcaster airs the serialized program and compensates the rights holders in Turkey for these broadcasting privileges. Intermediary entities, serving publishers and producers, actively participate in the sale of these broadcasting rights to foreign markets.

These developments have enabled Turkish TV series to attain international acclaim, significantly impacting the Turkish television industry on a global scale. The compelling narratives, remarkable performances, and superior production quality of Turkish serials have captured the interest of international audiences, fostering the successful export of Turkish television serials.

1.2.1 The significance of television ratings in Turkey

The inception of the Turkish TV series industry can be traced back to the 1970s, marked by initial endeavors spearheaded by the Turkish Radio and Television Corporation (TRT). This period saw TRT initiating studies to address technical equipment needs and the emerging public interest, which laid the foundation for the burgeoning TV series landscape. Collaborations formed between production teams, supported by professionals from the cinema and advertising sectors, played a vital role in shaping content for TRT. As time progressed, TRT expanded its reach by distributing some of its productions beyond the institution, leading to the establishment

of numerous production companies focused on creating diverse content in line with mainstream media in Turkey. This shift gradually reduced TRT's predominant influence as the industry began to emerge in the private sector. The introduction of the first private television companies in 1989 acted as a catalyst, significantly accelerating the industry's development and establishing it as a distinct financial resource recognized under the label of the TV series sector [2].

Today, the broadcasting tradition, sustained across various genres by numerous private broadcasting organizations, remains a cornerstone within the entertainment industry due to its substantial cost and potential. Research conducted by the Radio and Television Supreme Council (RTÜK) in 2018 revealed that Turkey's average daily television viewing time stands at 3 hours and 34 minutes [2]. This finding significantly underscores the pervasive role of television broadcasts in our daily lives, presenting a tangible illustration of their significance. Hence, within the mainstream media, which boasts significant customer potential, many production companies allocate their financial resources to television broadcasts. They seek to align their investments based on the success of their content, recognizing the pivotal role that television plays in reaching and engaging audiences.

In today's broadcasting landscape, the success of content is often measured by its achieved rating value among the public. This rating value reflects the average viewership rate of the broadcast content. Notably, this metric significantly influences the financial and strategic plans of production companies and serves as a vital criterion for advertising organizations and various entities associated with digital broadcasting.

Over time, the growing emphasis on the value of content in the public eye, rather than its inherent merits and quality, is often associated with economic considerations. Economic factors have progressively led to a shift where content's commercial success and financial implications often overshadow its intrinsic qualities and artistic value in public perception.

In Turkey, rating values are assessed through specific contracted private companies with official authorization. The designated company responsible for this assessment considers diverse cultural and socioeconomic structures across various provinces.

They establish a sample by randomly selecting households. People meters, and devices installed on the televisions of selected households, gather daily data at specified intervals during the day. These devices also reach different audiences by changing the recorded information at specific intervals.

Households that do not have regular television viewing habits or undergo significant changes in their social and economic structure are excluded from the system. Peoplemeter devices, designed to collect rating data, are equipped with a remote featuring distinct buttons for each family member. When a specific audience watches a broadcast, the corresponding individual momentarily presses the relevant button. Moreover, a frequency detection device installed in the television monitors channel transitions, while a timer device records data by detecting the times when the television is turned on and off. Rating measurements in Turkey commenced in 1989 through AGB Nielsen, a company that provides such data across several European countries. Presently, TNS Market Research and Consulting Ticaret A.Ş. conducts these rating measurements.

1.2.2 TV audience profile and watching habits

Due to the amplified interest in audience profiles and the broadcasting sector in Turkey, there was a noted decrease in media viewing habits, with a 10% decline in average daily television viewing time from 2008 to 2012 [2]. Nevertheless, in comparison to countries within the Organization for Economic Co-operation and Development (OECD), the audience profile in Turkey appears to surpass the OECD average based on the findings of this research. Several factors contribute to this decline in viewing habits. The increased ease and wider accessibility of internet connectivity over time have facilitated a shift among the general audience towards digital broadcasts or their content. This shift is pivotal in understanding the diminishing television viewership, as audiences explore and engage with digital alternatives due to the convenience and accessibility provided by the internet.

RTÜK's routine opinion polls have revealed a growing trend in television viewers' inclination towards watching domestic serials. This rise in viewership can be attributed to the expansion and diversification of television channels' series portfolios, which

have significantly influenced this pattern. The observed trend is a result of the reciprocal relationship between audience demands and the programs broadcasted by the channels. Over the past 3-4 years, domestic serials have garnered increased attention compared to other program categories, establishing a discernible trend evident across all major competing television channels. Conversely, RTÜK's research indicates a noteworthy surge in the viewership of entertainment programs. These shows, typically incurring lower production costs compared to TV series, are projected to assume a more significant role in the future. Analysis of ratings data, as presented by TNS Market Research and Consultancy Trade Inc., reveals that the six primary channels, upon evaluating the distribution of prime-time broadcasts, collectively hold a share ranging between approximately 60% to 65%. This estimation encompasses both reruns and condensed versions of the series, underlining the considerable viewership engagement with these programs.

The data unequivocally depict a rise in television viewers' interest in domestic serials, alongside a concurrent increase in popularity for entertainment programs that include reruns and condensed versions of these serials. Television channels are strategically diversifying their program offerings to cater to audience demands and gain a competitive edge. This trend notably highlights the considerable influence of television viewers' preferences in shaping program content. Consequently, it is anticipated that entertainment programs will gain prominence, aligning with the escalating interest in domestic TV series [2].

Through an analysis utilizing TNS's daily data, an examination was conducted on the five programs with the highest ratings. The findings highlight that serials emerge as the audience's most favored program type. Interestingly, it's been observed that even reruns and summaries of these series hold significant positions among the top five programs. This observation leads to the conclusion that approximately 50-55% of the top five programs comprise serials [3].

Programs lasting between 150-180 minutes, in conjunction with series reruns, form a substantial segment of television channels' prime-time schedules. Influential factors such as ratings and competition drive up series production costs, leading to a preference

for renowned actors, screenwriters, and producers. However, RTÜK regulations restrict the quantity and duration of advertisements during series broadcasts, thereby incentivizing broadcasters to prolong the duration of the series. Despite facing criticism from production companies and actors, this circumstance is unlikely to improve shortly due to the existing supply-demand dynamics in the industry.

Television channels have devised a distinct approach to counter the escalating costs of TV shows by allocating a significant portion of their weekly program schedules to series replays and summaries. While reruns and condensed versions do not entail extra production expenses for TV channels and production companies, they represent a significant supplementary revenue source for broadcasting organizations due to the advertisements they encompass. In many cases, broadcasters do not compensate producers for reruns, as production companies typically serve as subcontractors to broadcasters. Since the copyrights are often owned by the broadcasters, reruns can be incorporated into the channel's broadcast lineup even years after a series has been discontinued. This practice enables channels to leverage existing content, maximizing its profitability without incurring additional production costs.

1.2.3 Rating types

The aspiration to engage the potential audience within the advertising sector and to drive more product-oriented sales has been present since the proliferation of televisions in the 20th century. Specifically, marketing strategies developed based on the periods the general audience spent watching television and their preferred programs played a crucial role in the inception of the rating concept. The concept of rating, aimed at delineating the audience profile through a specific measurement system, has progressively laid the groundwork for producers to undertake pivotal actions within competitive environments over time. This system has become fundamental in understanding viewers' preferences and behaviors, guiding decision-making processes within the industry. The measurement, analysis, and research of broadcast audiences trace their origins back to the 1920s, coinciding with the emergence of commercial radio. As television came into existence in the 1930s, the methods developed for radio audience measurement were seamlessly adopted for television viewership analysis.

The early pioneers of these measurement systems acknowledged that their endeavors were shaping the fundamentals of a nascent media-centric democracy. Ratings assigned to broadcasts played a pivotal role in guiding consumers' (listeners/viewers) choices in station selections. However, audience measurements became controversial, particularly in the 1930s, primarily due to the use of mass media for propaganda. The potential manipulation of audiences through media content raised significant ethical concerns regarding the objectivity and influence of audience measurement techniques [4].

We can list the terminological terms used in audience measurement as follows:

i) **Rating (Viewing Rate)**: It is the percentage of the average number of spectators per minute in front of the television during the broadcast period within different socioeconomic and demographic audiences in certain periods. In other words, it is a technical term that expresses how many people watch television content from a random sample of 100 people. We can summarize it as the success rate of the content, which is paid attention to at the primary level of importance by the budgetary and advertising organizations, in return for the public.

ii) **Share**: It is the share of any channel during the total television broadcast time in certain periods. In other words, it is the viewing rate of the relevant channel in the competitive environment with the programs of other channels. While the rating is a ratio that all audiences receive regardless of the television duration at that moment, the Share value shows the share received from the audience who was in front of the television during the said broadcast time.

iii) **Total Views Rate (TVR)**: It refers to the average number of viewers per minute for all channels, which indicates how many people watch television as a percentage within a given time and socioeconomic or demographic audience.

iv) **Reach**: It is the rate of the audience watching at least 1 minute of a channel in a certain period. In other words, it is a term that indicates what percentage of the content's target audience has been reached at least once.

v) **Average Television Viewer Time (ATS)**: It is a ratio that shows how many percent of the related channel is watched during the broadcast period.

vi) **Gross Viewing Rate (GRP)**: It is the sum of the ratings received by each advertisement on television channels within a certain period.

Rating groups are systems employed to gauge the program preferences of television viewers and ascertain the popularity levels of broadcasts. These groups are frequently segmented by target audience demographics, encompassing factors such as age, gender, and socioeconomic status. For instance, rating groups categorized by letters like A/B/C1/C2/D/E are often delineated based on socioeconomic status. Ranging from A to E, Group A represents the highest-income viewers, while Group E represents the lowest-income viewers. However, rating groups can vary from one country to another, and different measurement methods might be employed. Each country may possess a distinct rating system and grouping structure. These rating groups serve as vital tools in identifying the audience for television channel programs, estimating advertising revenues, and assessing program success. Nevertheless, with the increasing popularity of digital broadcasting platforms and online video content, traditional television ratings might sometimes prove inadequate. In response, new measurement methods and metrics may be utilized to accommodate the evolving landscape of viewer habits and preferences.

If we look at the general rating groups;

- Group AB: Represents the highest income audience. This group is usually reviewed separately from the A/B/C1/C2 rating group.
- Group A: Represents high-income viewers.
- Group B: Represents the middle-upper-income audience.
- Group C1: Represents the middle-income audience.
- C2 Group: Represents the lower middle-income audience.
- Group D: Represents low-income viewers.

- Group E: Represents the lowest income viewers.
- TOTAL: Represents the audience covering the entire D and E groups.

In age-based rating groups;

- 4-15 years old (Kids)
- 16-24 years old (Youth)
- 25-34 years old (Young adults)
- Ages 35-49 (Adults)
- Ages 50+ (Middle-aged and older audiences).

In addition, the rating groups by gender are:

- female audience
- male audiences.

In the study covered in this thesis, the rating data is derived from the daily incoming rating dataset provided by Kantar Media, a company affiliated with the TNS group. The dataset encompasses rating values for the content of each channel within rating groups such as AB, ABC1, F20, M20, and TOTAL. Throughout the study, a specific focus was placed on the estimation of values associated with the ABC1 rating group. This choice was made because the ABC1 rating group is not only suitable for the general audience but is also a measurement group that encompasses a breakdown of the lower audience segment, a factor deemed crucial for producers in determining the success of content directly within that segment. Therefore, our choice of the ABC1 socio-economic segment as the target group for rating estimation is grounded on our analysis of ratings, reflecting our anticipation of their value within this particular demographic.

1.3 Literature Review

Various methods have been used in the literature to solve the problem of TV rating estimates. The most commonly used techniques are classification and regression analysis. Some studies apply machine learning models that make many classifications such as neural networks and decision trees in [5]. The article primarily focuses on analyzing how accurate television network rating forecasts are, considering factors like data aggregation methods and different models.

In [6], Nielsen aims to enhance new rating measurement methods and establish an infrastructure for regular rating predictions. The study finds that gradient-boosting methods offer more consistent results than traditional machine-learning approaches. Similar to our study in scope and methodology, various regression models, including penalized and linear regression, support vector machines, gradient boosting machines, random decision forests, and neural network models, are tested on previous datasets. Model results reveal the effectiveness of ensemble learning models based on boosting. However, the observed inconsistency and variability between actual and estimated ratings emphasize the need for context-specific actions in practical applications.

In [7] created a model incorporating features like genre, staff, and network. This article played a crucial role in shaping the methodological framework of our study and determining the characteristics of the dataset. Similarly, a framework was developed by incorporating visual features such as trailers and posters, resulting in a notable enhancement in prediction accuracy in [8] on a diverse collection of data.

In [9], a more comprehensive nested method is used for predicting the ratings of the content currently broadcast. The authors developed a two-level machine learning framework to estimate the ratings of Turkish TV series. While the first level uses a decision tree model, the second level uses a Support Vector Regression model (SVR) to categorize viewer ratings as either *high* or *low*.

Some studies used genetic algorithms for their regression analysis [5,9]–[13]. These studies allowed us to get a preliminary idea of what kind of data analysis can be done during the model preparation process, and our dataset has tried studies.

One of the most significant advancements in this field has been the identification of factors influencing the ratings of yet-to-be-aired TV series in [14]. What sets this study apart from previous research is its attempt to formulate a theory based on the analysis of obtained reports. The chosen samples for developing this theory consist of TV series from the most popular American channels. The underlying hypothesis revolves around whether upcoming series are adaptations or spin-offs of previously aired ones. Considering that this study addresses a distinct facet of the problem, the analyzed data and the exploration of relationships between them provide valuable foresight. The study reveals a correlation between these factors and the rating values of the series' inaugural episodes. Consequently, it represents a tangible stride toward crafting robust rating estimation models tailored for TV series.

In [12], a notable advancement is seen in rating estimation through a six-hour interval system. The study employs a forward-stepping feature selection regression model, incorporating data on TV content airings, audience evaluations, actor count, and TV viewership. Thus, the study explores the correlations between this data and the success rate of television content in predicting ratings. The results of these analyses suggest potential valuable insights from the developed study. Furthermore, a related study in [13] tested an estimation model encompassing attributes similar to ours, including content type, broadcast duration, program name, TV channel, and the three main actors of the content.

In a broad review of literature similar to our research, valuable insights have been gained regarding the technical aspects of our study. The findings have helped identify crucial features required for the complexity and content of our dataset and have guided the construction of a suitable model from a technical perspective. These insights have been validated through various statistical analysis studies.

1.4 Hypothesis

We put forth that variables such as actors, directors, screenwriters, content types, duration, broadcast days and times, and external factors such as national sporting events and weather conditions at the broadcast times affect viewer ratings in a

predictable way. Our machine-learning model for the content is going to provide such a reliable forecast for these ratings.

1.5 Thesis Overview

We designed a model and a user interface ready to be used as a product, capable of providing end-to-end service. The fact that we used a rich and large dataset, our focus on higher performance, the applicability of our models both before and after the broadcasts, and our comprehensive industry analysis, all contribute to the study's readiness for practical application and its potential to offer comprehensive services across the entire spectrum of the media industry.

The study utilizes two sequential models to examine content production and broadcast. The first model focuses on our hypothesis involving key elements such as cast selection and external factors such as weather conditions affect viewer ratings. Our second model predicts anticipated rating values by simulating expected ratings of competitor channels on the broadcast day. This enhances the study's ability to interpret variations in a competitive broadcasting landscape.

Our thesis distinguishes itself from similar works because of the following features:

i) **Rich and a large dataset**: The study benefits from a rich and extensive dataset derived from various sources, including social media data. This diverse dataset is used to test the model in live environments.

ii) **Higher performance**: Compared to other similar prediction systems, our work exhibits not only a higher accuracy but also a smaller error variance.

iii) **Ability to back-test the model**: The approach is applicable both before and after the content is broadcast, ensuring consistency. This feature contributes to the work's practicality and adaptability to real-world scenarios.

iv) **Comprehensive industry analysis**: We did a detailed back-test analysis on the entire available data. We used state-of-the-art machine learning techniques to explain the time-dependent variations in different choices of players,

screenwriters, directors, production companies, and channels. Our user interface provides a replicable and explainable forecast in available scenarios.

Chapter-2 explores the study's approach, focusing on understanding machine learning algorithms. It begins with an overview of the machine learning process, followed by explanations of supervised and unsupervised learning. The rationale for choosing regression models in supervised learning is discussed, along with a detailed examination of ensemble learning, particularly boosting. The chapter also covers the bias-variance tradeoff, emphasizing the importance of cross-validation techniques like k-fold cross-validation and the hold-out method. Specific machine learning algorithms such as KNN, gradient-boosting algorithms, Random Forest Regressor, and Bagging Regressor are explored, along with discussions on regularization parameters and evaluation metrics like MAE, MAPE, MSE, and RMSE. Hyperparameter optimization methods are compared, and the importance of model interpretability is highlighted through additive explanations.

Chapter-3 three delves into the machine learning pipeline, explaining its components and methods concisely. It starts by highlighting the system infrastructure's crucial role, including Apache Airflow integration and AWS components utilization. Dataset preparation, focusing on relational database establishment and feature engineering, follows. The chapter emphasizes optimizing the dataset for machine learning processes. It then outlines the modeling process and evaluates model performance metrics, introducing the Greedy Heuristic Method for auto-feature selection. Additive explainability is discussed for model interpretation and transparency. The chapter concludes with the creation of an interface page, detailing scenario-based prediction screens and competitor analysis for a user-friendly presentation of machine learning insights.

Chapter-4 the study's results, focusing on analyzing and interpreting the obtained data. It explores the outcomes of the applied machine-learning pipeline and the impact of various strategies. The chapter begins with evaluating model selection and comparing performance metrics across algorithms. It then discusses enhancements to the scenario-based model, detailing iterative improvements for

predictive accuracy. Hyperparameter tuning and its effects on model performance are examined. The chapter also covers enhancements to the competitor simulation model and analyzes SHAP charts for model interpretability. The user interface page is discussed, highlighting its intuitive design for end-users and showcasing scenario-based prediction screens and competitor analysis.

The final chapter of chapter-5 offers a concise conclusion to the study, summarizing experimental results, and practical applications, and suggesting future directions in machine learning. It provides an overview of the experimental outcomes from the applied machine-learning pipeline, including performance metrics and strategy effectiveness. Practical implications are discussed, illustrating how the study's insights can be applied in real-world scenarios. Potential future research avenues are outlined, addressing limitations and opportunities for further refinement of methodologies. This chapter encourages ongoing exploration and innovation in the field.

2. METHODOLOGY

Machine Learning (ML) is a subfield of artificial intelligence (AI) that enables computers to learn on their own by working with data. Machine learning algorithms are used to predict or classify new data by utilizing past data. In other words, machine learning is a discipline that allows computer systems to learn from data through algorithms. The primary goal in this field is to enable systems to perform specific tasks without human intervention or with minimal intervention. Machine learning typically involves training models using experiential data and applying these models to new data.

2.1 Machine Learning Process

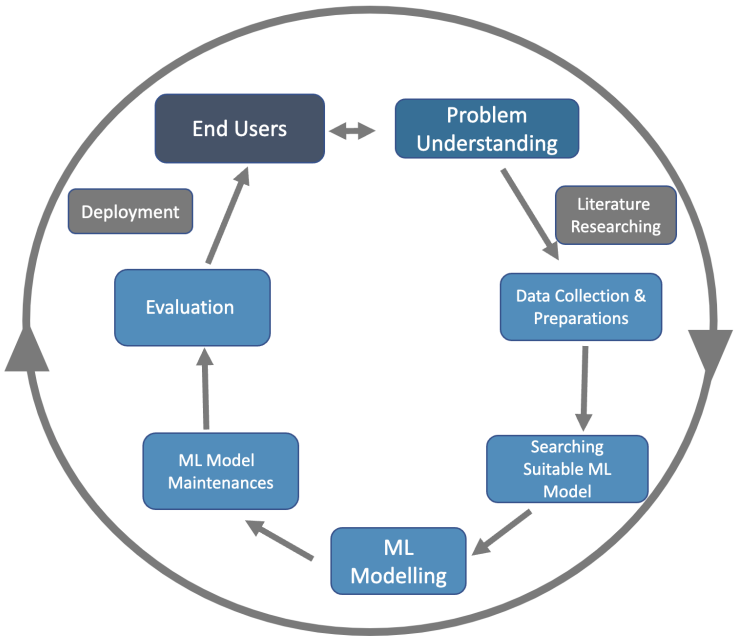

Figure 2.1 : ML Pipeline

The foundational steps of the machine learning process are crucial for the successful completion of a project and the model's effective utilization in real-world applications within an academic context. These steps encompass problem identification, data

collection, model training, and evaluation, commencing with the identification of the problem. The process is structured upon a repeatable and improvable loop, signifying the continuous integration of new information into the model. The fundamental operational steps of the study pipeline can be summarized as follows in Figure 2.1.

- **Problem Understanding**: Determine the purpose of your machine learning project and identify the problem you intend to address. Clearly define the objectives of your project, such as creating a predictive model, conducting data classification, or performing a regression analysis, for instance.
- **Data collection**: Data collection is the first and most important step in the machine learning process. Data is the input to machine learning algorithms. Data can be numerical, textual, or image-based. Data should be accurate and complete, free of errors and inconsistencies. Data should be in a format that can be processed by machine learning algorithms and data should be of sufficient quantity. Model performance is directly proportional to data quantity.
- **Data Preprocessing**: Once data is collected, it needs to be preprocessed before it can be used by machine learning algorithms. This process includes cleaning, converting to a numerical format, and filling in missing values. Data cleaning is the process of removing errors and inconsistencies from data. This ensures that the data is accurate and that the model makes accurate predictions. Data conversion to a numerical format makes it easier for machine learning algorithms to understand the data. Filling in missing values ensures that the data is complete and that the model can be trained using all of the data.
- **Model Selection**: A selection of machine learning algorithms is made. This selection is based on the type of problem and the available data.
- **Model Training**: The selected algorithm is trained on labeled data. The training process involves the algorithm learning from the data and creating a model that can be used to make predictions or classifications. The training process involves adjusting the algorithm's parameters. These parameters determine the algorithm's

performance. The quality of the training process directly affects the model's performance. Therefore, it is important to perform the training process correctly.

- **Model Evaluation**: The model is evaluated on test data. Test data is different from training data. Test data is used to evaluate how well the model will perform on real-world data. The evaluation process involves evaluating the model's performance metrics, such as accuracy, precision, and recall. The results of the evaluation process are used to evaluate the model's performance and to make necessary improvements to the model if needed.

- **Model Deployment**: The model is deployed in a production environment. A production environment is the environment where the model will be used in the real world. The deployment process involves installing and running the model in the production environment. For the deployment process to be successful, the model must meet the requirements of the production environment and must be reliable.

2.2 Types of Machine Learning

Machine learning can be summarized into two main categories: supervised and unsupervised learning. Supervised learning is a process where models are trained using labeled data. In this approach, the algorithm learns input data (features) and their corresponding output labels. Unsupervised learning, on the other hand, aims to teach patterns to the model without labeled data. Besides these two approaches, there are numerous machine learning methods, with regression models specifically employed to predict an output variable [15].

2.2.1 Unsupervised learning

Unsupervised learning, a fundamental branch of machine learning, is a powerful method to explore datasets' patterns and relationships. This approach is particularly effective for understanding and predicting structures within the data.

Also known as unsupervised learning, this methodology operates without labeled data during the learning process. This means the model autonomously identifies internal structures and similarities within the dataset. Unsupervised learning algorithms are

typically categorized into two main types: clustering and dimensionality reduction [16].

- **Clustering**: Clustering is utilized to group data points with similar features. The algorithm analyzes similarities within the dataset and aggregates data points into specific clusters. For instance, in customer behavior analysis, clustering can be employed to group customers with similar shopping patterns into the same cluster. [17].
- **Dimensionality Reduction**: Dimensionality reduction aims to transform complex structures within the dataset into a more understandable form. This can be achieved by reducing the number of features or selecting representative features. Dimensionality reduction plays a crucial role in data visualization and comprehension. [17].

Unsupervised learning finds applications in various fields. It is commonly used in market segmentation, genetic data analysis, image processing, and pattern recognition. Data scientists and researchers gain valuable insights through these applications by discovering hidden patterns within datasets.

2.2.2 Supervised learning

Supervised learning, a cornerstone of machine learning, is a method where models are trained using labeled data to make predictions and classifications. In this approach, the algorithm learns from input data (features) and their corresponding output labels, aiming to generalize patterns and relationships.

During the supervised learning process, the model is provided with a training dataset where each example is paired with the correct output. The algorithm optimizes its parameters through this labeled data, making accurate predictions on new, unseen data.

- **Regression**: Regression is a branch of supervised learning primarily used to predict a continuous output variable. The model is trained on a dataset that combines input features with corresponding output labels, aiming to generate a

continuous numerical value associated with a specific input combination. For example, regression can be employed to predict continuous variables such as house prices, temperature forecasts, or future stock prices. Algorithms in this context focus on identifying and optimizing patterns in the dataset to predict output values accurately [18].

- **Classification**: Classification, another crucial aspect of supervised learning, is specifically used to predict or categorize data into predefined classes. The model is trained on a dataset that combines input features with specific output labels. Classification attempts to associate a specific input combination with a particular category. For instance, classifying whether an email is spam or not, diagnosing the presence of a disease, or categorizing situations like these are examples of classification problems. During the training process, algorithms learn patterns in the dataset and draw decision boundaries for specific classes, enabling them to make classifications for new inputs [18].

2.3 Ensemble Learning

Ensemble learning stands as a powerful method in the field of machine learning, aiming to create a more robust model by combining different learning models. This method enables a group of weak models to come together and form a powerful model, often contributing to achieving better performance than a single model.

The fundamental philosophy of ensemble learning is based on the principle of "collective intelligence." According to this principle, the collective knowledge of a group of individuals can be stronger and more accurate than the knowledge of each individual. Similarly, ensemble learning enhances overall performance by reducing the probability of error for each model through the combination of multiple models.

Ensemble learning is generally categorized into three main types: Bagging (Bootstrap Aggregating), Boosting, and Stacking.

- **Bagging (Bootstrap Aggregating)**: In this method, the same learning algorithm is trained multiple times on different subsets of data. Each subset contains random samples from the main dataset. Then, the predictions of these models are combined

by taking their average or majority vote. Random Forest is a popular ensemble learning algorithm that operates using the bagging method.

- **Boosting**: In boosting, weak models are trained sequentially, and each model is trained to focus on the errors of the previous model. This way, each model attempts to correct the errors focused on by its predecessor. Gradient Boosting and AdaBoost are examples of ensemble learning algorithms that operate using the boosting method.

- **Stacking**: In stacking, algorithms are trained on different data sets. Each algorithm is trained using the outputs of other algorithms.

Advantages of ensemble learning include increased overall model stability, reduced risk of overfitting, and better performance in complex problems. However, disadvantages of this method may include increased training time and model complexity.

2.4 Bias-Variance Tradeoff

One of the key metrics in evaluating the performance of machine learning models lies in their ability to generalize their successes in training data to unseen data. Overfitting and underfitting are critical phenomena influencing this generalization capacity. Overfitting occurs when a model excessively conforms to the training data while underfitting is associated with the model inadequately fitting the training data. These phenomena often present challenging problems that must be addressed in various machine-learning applications.

This subsection delves into the intricate phenomenon of overfitting, with the primary objectives of delineating its conceptual boundaries, laying down a robust mathematical groundwork, and delving into the efficacy of various prevention strategies. To initiate this exploration, we will first offer a comprehensive overview of overfitting, elucidating its essence, before delving deeper into elucidating the underlying mathematical constructs that underpin its manifestation, as outlined by [19].

Several factors can contribute to overfitting and underfitting, including:

i) **Model Complexity**: Overfitting is more likely to occur when a model has a high degree of complexity. This is because complex models are more likely to learn the noise in the training data, rather than the true underlying patterns. Underfitting is more likely to occur when a model has a low degree of complexity. This is because simple models are less likely to be able to learn the true underlying patterns in the data.

ii) **Lack of Data**: Overfitting and underfitting are also more likely to occur when a model is trained on a small dataset. This is because a small dataset does not provide the model with enough information to learn the true underlying patterns.

iii) **Non-Representative Data**: Overfitting can also occur when a model is trained on data that is not representative of the data that the model will be used on. This is because the model may learn patterns that are not generalizable to new data. In underfitting the model may not be able to learn the patterns that are present in the new data.

2.4.1 Overfitting

Overfitting is a common problem in machine learning that occurs when a model learns the training data too well and is unable to generalize to new data. This can lead to inaccurate predictions of new data, which can be a serious problem in real-world applications.

Overfitting can have several negative effects on a machine-learning model, including:

i) **Inaccurate Predictions**: Overfitted models are more likely to make inaccurate predictions on new data. This can lead to missed opportunities, financial losses, or even safety hazards.

ii) **Reduced Generalization Ability**: Overfitted models are less likely to be able to generalize to new data. This means that they may not be able to perform well on data that they have not seen before.

iii) **Increased Variance**: Overfitted models have higher variance, which means that their predictions are more likely to vary from one data point to another. This can make it difficult to trust the predictions of an overfitted model.

Many methods can be used to prevent overfitting:

i) **Data Augmentation**: Data augmentation is a technique for artificially increasing the amount of training data by creating new data points from existing data points. This can help to reduce overfitting by providing the model with more data to learn from.

ii) **Regularization**: Regularization is a technique for adding a penalty to the model's complexity. This can help to reduce the model's variance and prevent it from learning the noise in the training data.

iii) **Cross-validation**: Cross-validation is a technique for evaluating a model on data that it has not seen before. This can help to identify overfitting by comparing the model's performance on the training data to its performance on the test data.

Overfitting is a serious problem that can have a significant impact on the performance of machine learning models. By understanding the causes of overfitting and the methods for preventing it, machine learning practitioners can develop models that are more accurate and generalizable.

2.4.2 Underfitting

Underfitting refers to the condition where a model inadequately fits the training data. In such instances, the model fails to capture essential patterns or relationships within the training data, resulting in poor performance on new, unseen data samples.

Underfitting typically arises when a model is overly simplistic or possesses insufficient learning capacity. The model may lack the flexibility or complexity required to comprehend the intricacies of patterns present in the data.

Mathematically, underfitting is characterized by a high generalization error, indicating that the model performs poorly on both the training set and the test set. This suboptimal

performance often occurs when the model's complexity is too low to adequately represent the underlying structure of the data.

To address underfitting, strategies such as increasing the model's complexity, conducting more learning iterations, or incorporating additional data can be employed. These approaches aim to enhance the model's capacity to better understand crucial patterns within the training data and, consequently, improve overall performance on new data.

Several factors can contribute to the emergence of underfitting:

i) **Inadequate Model Complexity**: An overly simplistic model cannot discern the intricate relationships within the data, resulting in underfitting.

ii) **Data Scarcity**: When the training data is insufficient, the model lacks ample opportunity to learn the inherent structure of the problem, leading to underfitting.

iii) **Improper Feature Selection**: If the chosen features fail to provide informative insights into the target variable, the model struggles to learn meaningful patterns, culminating in underfitting.

A multifaceted approach is crucial to effectively combat underfitting:

i) **Enhancing Model Complexity**: Employing a more sophisticated model architecture can expand the model's representational capacity, potentially mitigating underfitting. However, a delicate balance must be struck, as excessive complexity can invite overfitting.

ii) **Data Augmentation**: Artificially increasing the size and diversity of the training data can provide the model with richer learning opportunities, potentially alleviating underfitting.

iii) **Feature Engineering**: Carefully crafting informative features can illuminate the underlying structure of the data, enhancing the model's capacity to learn and reducing underfitting.

iv) **Regularization Techniques**: Incorporating regularization penalties into the learning process can constrain the model's complexity, preventing it from overfitting to the training data while fostering its ability to generalize to unseen instances.

Underfitting remains a significant hurdle in the pursuit of robust and reliable machine learning models. By comprehending its mathematical underpinnings, identifying its potential causes, and implementing effective mitigation strategies, we can navigate this intricate landscape and foster models capable of insightful prediction in the face of novel challenges.

2.4.3 Bias-Variance tradeoff

The Bias-Variance Tradeoff refers to the need to balance a model's complexity with its ability to generalize. It highlights the tradeoff between a model's bias (error due to overly simplistic assumptions) and variance (error due to too much complexity). Striking the right balance is crucial to achieving good predictive performance on both the training and unseen data.

1. **Variance**: Variance is a term that measures how far the values in a dataset are spread out from the mean. High variance indicates that the data points are more scattered around the mean, implying that the model or prediction is not precise. The formula for variance is as follows:

$$\text{Variance}(\sigma^2) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{2.1}$$

where:

- σ^2: Represents the variance. This symbol represents the variance, which measures how spread out the data points are from the mean. A high variance indicates that the data points are far from the mean, while a low variance indicates that they are closer together.
- n: Represents the number of data points.
- x_i: Represents each data point.

- $\bar{x}$: Represents the mean of the data points.

2. **Bias**: Bias is the tendency for a model's predictions to systematically deviate from true values. High bias indicates that the model has not learned sufficiently from the training data and has limited generalization ability.

   The formula for Bias is as follows:

$$\text{Bias} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{2.2}$$

   where:

   - $\frac{1}{n}$: Takes the average of the deviations of each data point.
   - $\sum_{i=1}^{n}$: The summation symbol denotes a sum over all data points.
   - $(\hat{y}_i - y_i)$: Measures the deviation by subtracting the actual value from the predicted value for each data point.
   - $\hat{y}_i$: predicted value
   - y_i: each data point.

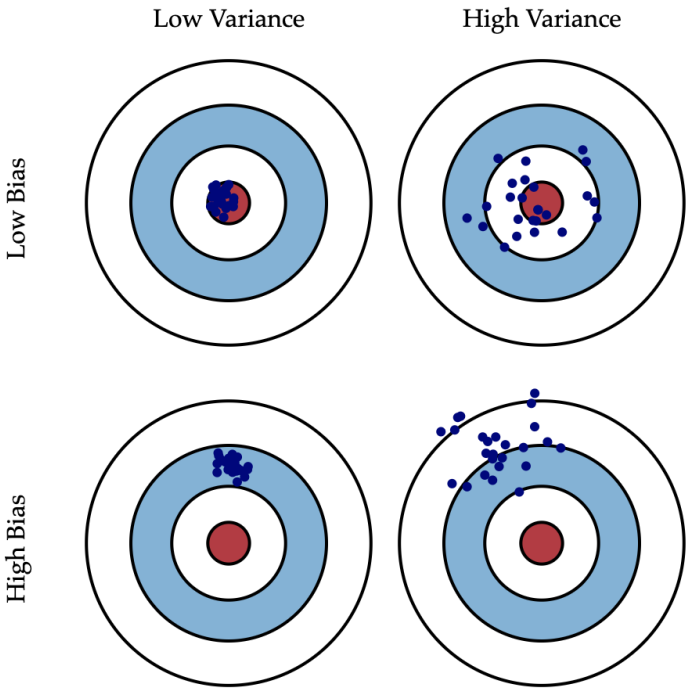

Figure 2.2 : Bias-Variance Tradeoff [20].

The overfitting situation is characterized by low bias and high variance. The model fits the training data very well (low bias), but this often leads to poor generalization of new data (high variance). The underfitting situation is characterized by high bias and low variance. The model has not learned the training data well enough (high bias), which typically results in poor generalization to new data (low variance). In an ideal scenario as we can see in Figure 2.2, a model should have both low bias and low variance. However, there should be a balance between these two terms. To achieve this balance, the concept of "bias-variance tradeoff" is often used. Increasing the complexity of a model can reduce bias but may increase variance; reducing model complexity can decrease variance but may increase bias. The goal is to strike a balance between these two terms as much as possible.

2.5 Cross-Validation Techniques

In the domain of machine learning models, after the selection of an appropriate model, diverse methodologies exist to attain optimal results, enhance the model, and mitigate potential issues related to overfitting or underfitting.

During the evaluation of model outcomes, unwanted scenarios such as overfitting or underfitting may arise due to various factors. This encompasses measures taken to sustain the continued effectiveness of the machine learning model after its development. Within this process, pivotal roles are assumed by cross-validation, hyperparameter optimization, and regularization techniques. The selection of specific parameters or methodologies may vary contingent on whether the problem at hand pertains to regression or classification.

Initially, the division of the dataset into appropriate segments enables the creation of subsets aimed at assessing the accuracy of our model's training with the designated dataset. Upon reviewing the literature, methodologies such as Bootstrap, Leave-one-out, and Hold-out are among the commonly utilized approaches in this domain. In our study, the principal dataset, extensively analyzed throughout the research, was partitioned using the Hold-out method.

Cross-validation, known as stratified validation or fold cross-validation, is an essential method applied in machine learning and statistical modeling. It serves the

purpose of impartially evaluating a model's performance and gauging its capability to generalize.

The fundamental principle behind cross-validation lies in optimizing dataset utility. Typically, datasets are partitioned into training and testing subsets, where the former aids in model learning and the latter assesses model performance. However, inherent variability in results stemming from the manner of dataset division and randomness prompted the development of the cross-validation method to mitigate this issue.

2.5.1 *k*-Fold cross-validation

In cross-validation, the dataset is segmented into distinct parts. Subsequently, each part undergoes independent training and testing procedures. One part is tested against the other parts while being trained on the remaining segments. This iterative process allows for multiple rounds of testing across the dataset, ensuring a more robust and reliable performance evaluation [21].

The preeminent approach to cross-validation is the k-fold cross-validation method, widely employed in various analyses. In this technique, the dataset is divided into k equivalent parts. Subsequently, each portion is sequentially allocated as a test set while the remaining sections serve as the training set. This iterative process continues until each segment has been utilized as a test set at least once. Consequently, k distinct performance measures are acquired, and the model's overall performance is gauged by averaging these measures. Throughout the research conducted in this study, the k-fold cross-validation method was implemented.

Cross-validation is applied across various scenarios, encompassing the validation of a model's generalization capabilities, fine-tuning hyperparameters, and the comparative assessment of different models. This method optimizes dataset utilization, ensuring more robust and reliable results are obtained.

2.5.2 The hold-out method

The hold-out method is employed to appraise the model's performance by partitioning a given dataset into two distinct segments: one for training and the other for testing. In this approach, a portion of the dataset, typically ranging between 70% to 80%, is allocated for training purposes, while the remainder is reserved for testing. Although the hold-out method is generally straightforward and expeditious, its efficacy might diminish when applied to smaller datasets. Additionally, the outcomes obtained may exhibit variability contingent upon the random division of the dataset. Consequently, the Hold-out method finds utility in larger datasets or for a rapid overall performance assessment [21].

In implementing the hold-out method, several considerations merit attention when segregating the dataset into training and testing subsets. It is essential to randomly divide the dataset into these segments, enabling the model to undergo training and evaluation on distinct samples, and facilitating improved generalization. Moreover, should the dataset exhibit an imbalanced class distribution, it becomes pivotal to partition the training and test sets to mirror this imbalance. Employing the stratified sampling method permits a division ensuring the representation of each class in the correct proportion. The dataset's size is a crucial factor when determining the proportions allocated for training and testing. Typically, a larger training set aids the model in acquiring comprehensive learning and generalization capabilities. Nonetheless, when the training set is notably small, the model might encounter challenges in adequately grasping the dataset's complexity.

In numerous academic endeavors, particularly in scenarios with a substantial volume of observations, the dataset is often partitioned into a training set, a test set, and additionally a validation set. At this stage, the validation set is further segregated from the training dataset, employing the Hold-out method, akin to the division between the training and test sets, corresponding to the test subset. Model enhancements and evaluations of success are conducted through the validation set, while the test set operates as entirely unseen data, mirroring the scenario of testing

with data external to the dataset. Consequently, this approach prevents potential data leaks and ensures that the model is trained without prior exposure to the test dataset, establishing a more robust model setup. The dataset utilized in the research conducted for this study was similarly structured by being divided into three distinct subsets: training, testing, and validation.

2.6 Machine Learning Techniques

The primary goal of regression algorithms is to conduct statistical measurements that assess the correlation strength between various independent variables concerning a designated objective variable within a dataset. These algorithms aim to solve problems by establishing a rule-based set of relationships based on these associations and subsequently conducting studies on the dataset. The fundamental objective within regression problems is the capacity to generate predictions. This capacity stems from formulating assumptions, devoid of a repetitive structure, derived from an algorithm learned through these analyses. In essence, regression algorithms present assumptions following an examination of the relationships between the dependent variable and other independent variables in the dataset analysis process.

While the primary aim of most regression models revolves around generating predictions from numerical values, it's erroneous to construe the purpose of regression solely as an estimation. Though central to its function is assumption generation, regression models serve a broader scope encompassing various other functions. Apart from estimation, regression is commonly employed for tasks such as making specific inferences, establishing new relationship rules, capturing fresh insights about a problem based on these relationships, and priming the dataset for utilization by other algorithms like clustering and deep learning. The multifaceted utility of regression extends beyond mere estimation, offering a spectrum of analytical applications within diverse problem-solving contexts.

Regression problems serve as the foundational underpinning for various machine learning algorithms. Consequently, they can be employed in tandem with other algorithms, either to generate output values for them or directly for optimizing a

project. For instance, despite logistic regression being deemed one of the most straightforward algorithms in the realm of machine learning, it forms the bedrock for complex systems like deep learning and artificial neural networks, showcasing a sophisticated mathematical underpinning.

During the execution of regression analysis, the correlation among parameters is translated into a mathematical model. These parameters can be selected either independently or interdependently, yielding linearity, multilinear aspects, or polynomial curvatures based on the dataset's distribution. The objective behind constructing this mathematical function is to encapsulate the dataset within this function and derive predictions through machine inferences trained using this model. The process aims to distill the dataset's information into a functional form, enabling accurate predictions through machine learning inferences based on this established function.

Within the scope of this thesis, a machine-learning model was constructed by examining the correlations between variables and their impacts on one another. The analysis incorporated multivariate nonlinear regression algorithms such as K-Nearest Neighbors (KNN), Random Forest, Bagging Regressor, and boosting algorithms including eXtreme Gradient Boosting (XGBoost), Light Gradient Boosting Machines (LGBM), and CatBoost models. The selection of these models was guided by a logical progression framework. Based on analyses and inferences drawn from the dataset, models from each potential regression model family were chosen, aiming to guide the completion of model research steps by determining which model was suitable and the rationale behind its selection.

2.6.1 KNN algorithm

The K-Nearest Neighbors (KNN) algorithm utilizes the influence of the closest neighbors within the training dataset to classify or predict observation points. As a result, the algorithm is notably straightforward to comprehend and implement, operating as a sample-based learning algorithm. This approach allows the algorithm to classify or predict new samples directly by leveraging the properties of samples within the training dataset. Notably, the KNN algorithm does not assume a specific

distribution for the classification or regression process. The KNN Regressor is a simple yet effective algorithm used for regression tasks. It predicts continuous values based on the average of the labels of its nearest neighbors. Let's delve into the workings of the KNN Regressor through pseudocode.

Algorithm 1 k-NN Regressor Pseudocode

Input: Training data X_{train}, labels y_{train}, test data X_{test}, number of neighbors k
Output: Predicted values y_{pred}
procedure KNN_REGRESSOR($X_{train}, y_{train}, X_{test}, k$)
  for $i \leftarrow 1$ to $|X_{test}|$ **do**
    $dists \leftarrow$ CalculateDistances($X_{test}[i]$, X_{train})
    $k\_neighbors \leftarrow$ FindTopKNeighbors($dists, k$)
    $y_{pred}[i] \leftarrow$ Predict($y_{train}[k\_neighbors]$)
  end for
  return y_{pred}
end procedure
function CALCULATEDISTANCES(X_{test}, X_{train})
  $dists \leftarrow []$
  for $j \leftarrow 1$ to $|X_{train}|$ **do**
    $dist \leftarrow$ CalculateDistance(X_{test}, $X_{train}[j]$)
    $dists$.append($dist$)
  end for
  return $dists$
end function
function FINDTOPKNEIGHBORS($dists, k$)
  $indices \leftarrow$ SortAscending($dists$)
  $k\_neighbors \leftarrow indices[:k]$
  return $k\_neighbors$
end function
function PREDICT($y_{neighbors}$)
  $y_{pred} \leftarrow$ Average($y_{neighbors}$)
  return y_{pred}
end function

Figure 2.3 : KNN Regressor

The KNN Regressor procedure is the central algorithm of the pseudocode in Figure 2.3, and for each test instance in the dataset X_test, it undergoes the following steps:

i) The algorithm calculates the distances between the current test instance and all training instances using the CALCULATEDISTANCES function. This function takes the test instance X_{test}, and the training data X_{train}, as input, and its output is a list of distances representing the dissimilarity between the test instance and each training instance.

ii) The algorithm proceeds to find the indices of the top k neighbors with the smallest distances using the FINDTOPKNEIGHBORS function. This function

takes the list of distances (*dists*) and the parameter k as input and returns the indices of the k neighbors with the smallest distances.

iii) The algorithm predicts the output for the current test instance by averaging the labels of its k neighbors with the Predict function. The Predict function takes the labels of the k neighbors $y_{\text{neighbors}}$, as input and outputs the predicted value by calculating the average. Overall, this process is repeated for each test instance, providing a set of predicted values.

In the context of the functions, CALCULATEDISTANCES computes the distances between a given test instance and all training instances, forming a list of dissimilarities. FINDTOPKNEIGHBORS takes this list of distances and the parameter k sorts the distances in ascending order, and returns the indices of the k neighbors with the smallest distances. Lastly, the Predict function utilizes the labels of the k neighbors to compute the average, which serves as the predicted value for the current test instance. These functions collectively enable the KNN Regressor algorithm to make predictions based on the similarity of instances in the feature space, providing a flexible and intuitive approach for regression tasks.

In the K-Nearest Neighbors (KNN) algorithm, the critical parameter is the "K" value, determined by the user, which dictates the number of nearest neighbors considered to classify or predict the new data point. The selection of the "K" value significantly influences both the algorithm's performance and its generalizability. A small "K" value might render the model more susceptible to noise, leading to potential overfitting by considering the idiosyncrasies of the dataset. Conversely, a large "K" value might decrease the model's ability to generalize, potentially oversimplifying the classification by ignoring subtle differences within the dataset.

Another crucial parameter is the similarity criterion, which determines the measure used to compute the distance between data points. Popular distance metrics include the Euclidean distance and the Manhattan distance. However, KNN allows for the flexibility of choosing various distance measures, and the selection of the appropriate similarity criterion greatly impacts the algorithm's performance and suitability for the given dataset [22].

2.6.2 Gradient-Boosting algorithms

Algorithm 2 Gradient Boosting Algorithm

Input: Training data X_{train}, labels y_{train}, number of boosting rounds T
Output: Boosted model $F(x)$
Initialize the model with a constant value: $F_0(x) = argmin_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
for $t \leftarrow 1$ to T **do**
Calculate the negative gradient of the loss function: $g_t = -\left[\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}\right]_{i=1}^{N}$
Train a weak learner h_t on the negative gradient g_t.
Update the model: $F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$ where η is the learning rate.
end for
return Boosted model $F(x) = F_T(x)$
function CalculateNegativeGradient($F_{t-1}(x_i)$, y_i)
Input: Previous model predictions $F_{t-1}(x_i)$, true labels y_i
Output: Negative gradient of the loss function g_t
Description: Calculates the negative gradient of the loss function with respect to the previous model predictions.
end function
function TrainWeakLearner(X_{train}, g_t)
Input: Training data X_{train}, negative gradient g_t
Output: Weak learner h_t
Description: Trains a weak learner on the negative gradient to make predictions.
end function

Figure 2.4 : Boosting Algorithms

Gradient Boosting serves as an ensemble learning technique extensively employed in machine learning and statistical regression tasks. Essentially, it aims to amalgamate multiple weak learners, signifying simple learning algorithms with relatively inferior predictive capabilities, to construct a more robust model. The process involves successive training of these weak learner models, focusing on rectifying the errors introduced by prior learners to enhance the overall model's predictive performance.

The operational sequence of the gradient boosting algorithm in the context of regression can be delineated as follows in Figure 2.4.

The boosting algorithm pseudocode involves a series of essential steps:

i) The algorithm takes input parameters such as training data X_{train}, labels y_{train}, and the number of boosting rounds, T.

ii) The algorithm starts by calculating the negative gradient of the loss function for each round. The calculation of the negative gradient determines the sensitivity

of the loss function concerning the previous model's predictions and the actual labels. This step is a key factor in boosting algorithms' scalability and speed because it performs calculations on a per-sample basis rather than the entire dataset.

iii) The model is updated using a weak learner h_t trained on the negative gradient. The weak learner is added to the model with a specific learning rate , weighted to iteratively correct the model's previous errors and make better predictions.

iv) These steps are repeated until the specified number of boosting rounds is reached, and the resulting model $F_T(x)$ is returned as a boosted model. This model can make more accurate predictions by leveraging the features of the training data.

The functions within the pseudocode assist in performing these steps. The CALCULATENEGATIVEGRADIENT function computes the negative gradient of the loss function concerning the previous predictions, indicating how much the model deviates from the correct predictions. The TRAINWEAKLEARNER function trains a weak learner on this negative gradient, optimizing it to correct the model's errors.

The gradient-boosting algorithm builds a powerful model by aggregating weak learners. This model efficiently operates, especially on large datasets, and provides notable improvements at each step of the learning process.

In the investigation conducted as part of this thesis, the study employed three gradient-boosting algorithms: XGBoost, LGBM, and Catboost.

2.6.2.1 XGBoost algorithm

Recognized as a Gradient Boosting method, XGBoost distinguishes itself among other boosting algorithms by its notable attributes. Unlike its counterparts, XGBoost is acknowledged for its swifter processing, low memory utilization, and elevated predictive performance. Employing a gradient descent algorithm to rectify errors, it leverages a tree-based structure to optimize each estimator. Additionally, it integrates features such as regularization techniques and the capacity to assess attribute importance, effectively mitigating overfitting and

assessing feature contributions to estimations. Widely lauded for its adeptness in managing large datasets and its propensity to yield superior outcomes compared to other boosting algorithms, XGBoost, albeit demanding in parameter tuning and computational resources, is broadly applied as an effective tool in diverse machine learning problems [23].

Distinctive features of XGBoost compared to other gradient boosting algorithms:

- **Utilization of Balanced Error Function**: XGBoost employs a specialized error function to fortify its resilience against overfitting. This unique error function not only enhances the precision of predictions but also serves as an effective deterrent against overfitting.
- **Optimization of Learning Rate**: XGBoost incorporates an adaptive learning rate algorithm, allowing for dynamic adjustments. This adaptive mechanism automatically fine-tunes the weights utilized in determining each tree's contribution, thereby expediting the learning process and enhancing overall efficacy.
- **Parallel Processing Capabilities**: XGBoost is equipped with robust parallel processing capabilities, enabling swift training on substantial datasets. This feature significantly accelerates the model training process and augments overall computational efficiency.
- **Tree-Based Learning Strategies (Tree Pruning)**: Embracing a tree-based modeling approach, XGBoost focuses on constructing trees through meticulous pruning of extraneous branches (leaves) and nodes. This strategic pruning ensures the development of a streamlined and more efficient model.
- **Effective Handling of Missing Data**: XGBoost distinguishes itself by its adept handling of missing data. The algorithm seamlessly processes observations with missing values, fostering a seamless integration of incomplete data and optimizing dataset utilization.

These features position XGBoost as a formidable machine learning algorithm, renowned for its exceptional performance in classification and regression tasks across a diverse spectrum of applications.

2.6.2.2 LGBM algorithm

LGBM constitutes a rapid and scalable Gradient Boosting Framework. Its distinction lies in rapid runtime and minimal memory usage. The framework's high performance is attributed to its parallel processing capabilities and support for categorical variables. It employs optimization techniques like histogram-based feature scanning and derives the final estimate through a weighted average of predictions. Offering notable features such as swift training times, minimal memory requirements, and the ability to evaluate feature importance, LGBM stands as a powerful algorithm capable of swift and effective predictions, particularly when handling extensive datasets. While parameter tuning demands attention, LGBM is the preferred boosting algorithm, delivering successful outcomes in classification, regression, and sequencing problems [24].

LGBM distinguishes itself prominently from other gradient-boosting algorithms through a suite of impactful features:

- **Histogram-Based Learning**: LGBM embraces a sophisticated histogram-based learning methodology. This entails the strategic grouping of the dataset into predefined bins, resulting in expedited computational processes. This feature manifests in superior memory efficiency and accelerated training times compared to conventional methodologies.
- **Balanced Learning (Gradient-based One-Side Sampling - GOSS)**: LGBM incorporates Gradient-based One-Side Sampling (GOSS), an astute sample selection technique accentuating instances with substantial gradients during the training phase. This deliberate emphasis expedites the learning process of the model.

- **Direct Support for Categorical Features**: LGBM exhibits adept handling of categorical features by offering direct support, circumventing the need for feature transformations that are often requisite in alternative algorithms.
- **Parallel and Distributed Training**: LGBM boasts robust parallel and distributed computing capabilities, amplifying its prowess in expeditiously training on extensive datasets.
- **Leaf-Wise Growth Strategy**: LGBM employs a leaf-wise growth strategy for tree construction. This strategic approach facilitates the expeditious deepening of trees, empowering the model to discern intricate relationships and potentially attain heightened accuracy.

These meticulously crafted features collectively position LGBM as a superior choice in scenarios involving substantial datasets, intricate categorical features, and the imperative for expeditious training durations, thereby outshining alternative gradient-boosting algorithms.

2.6.2.3 Catboost algorithm

Catboost represents a boosting algorithm that directly processes categorical variables. This automatic handling of categorical variables reduces data preprocessing time and enhances prediction performance. Its rapid training and predictive capabilities position CatBoost as the algorithm of choice. CatBoost's distinct feature lies in its ability to generalize more effectively than other algorithms when working with discrete variables. The automatic attribute sorting enhances prediction accuracy. Renowned for its effectiveness in classification and regression problems, CatBoost offers advantages such as less stringent parameter settings and a user-friendly interface [25].

The CatBoost Regressor offers a distinct set of features and advantages compared to other gradient-boosting algorithms:

- **Automatic Encoding for Categorical Features (CatBoost Encoding)**: CatBoost stands out with its ability to automatically encode categorical features. While other algorithms often require steps for transforming or

processing categorical features, CatBoost can handle them more effectively through automatic encoding.

- **Balanced Learning (Ordered Boosting)**: CatBoost utilizes a technique called Ordered Boosting, employing the use of sequentially ordered examples at each node of the tree to enhance learning. This approach aids in making the model more resistant to overfitting.
- **Scale Sensitivity**: CatBoost can operate in a scale-sensitive manner, directly addressing different scales among features. This enables the model to make more precise and balanced predictions by handling variations in feature scales.
- **Value Assignment for Categorical Features (CatBoost Baseline**: CatBoost includes a feature known as "CatBoost Baseline," which provides a default value assignment mechanism for categorical features. This feature can efficiently assign initial values in datasets containing categorical variables.
- **Multi-Model Support**: CatBoost comes equipped with the capability to support multiple models simultaneously. This feature allows the combination of various models to achieve a more robust predictive power.

The CatBoost Regressor, distinguished by these features, can particularly excel in datasets with categorical features and in the context of balanced learning.

2.6.3 Random forest regressor algorithm

Algorithm 3 Random Forest Regressor Algorithm

Input: Training data X_{train}, labels y_{train}, number of trees T, maximum depth D
Output: Random Forest model $F(x)$
Initialize an empty forest: $forest = []$
for $t \leftarrow 1$ to T **do**
    Randomly select a subset of features: $selected\_features \leftarrow$ randomly_select_features(X_{train})
    Randomly sample data with replacement: $subset\_X, subset\_y \leftarrow$ randomly_sample_data(X_{train}, y_{train})
    Train a decision tree on the subset of data with selected features: $tree \leftarrow$ TrainDecisionTree($subset\_X, subset\_y, selected\_features, D$)
    Add the trained tree to the forest: $forest.append(tree)$
end for
return Random Forest model $F(x) = forest$

Figure 2.5 : Random Forest Algorithm

The Random Forest Regressor is a form of ensemble learning within the field of machine learning, utilized specifically for addressing regression problems. Similar to the Random Forest algorithm, the Regressor is composed of a collection of decision trees. Each tree produces its prediction, contributing collectively to the final prediction.

The operational procedure of the Random Forest Regressor algorithm involves the following sequential steps:

i) The algorithm takes input parameters such as training data X_{train}, labels y_{train} the number of trees T, and the maximum depth D for each tree.

ii) The algorithm initializes an empty forest, denoted as *forest*, which is a list that will store the individual decision trees.

iii) For each iteration from t=1 to T;

- Randomly select a subset of features from the input training data. This helps introduce diversity among the trees.
- Randomly sample the training data with replacement to create a subset for the current tree. This is known as bootstrap sampling.
- Train a decision tree on the subset of data with the selected features and a specified maximum depth D.
- Add the trained decision tree to the forest.

iv) After completing all iterations, the algorithm returns the trained random forest, which consists of T decision trees.

As we can see in Figure 2.5, we use the Randomly Select Features function is randomly select a subset of features from the input training data, ensuring diversity among the trees. Randomly Sample Data function performs bootstrap sampling by randomly selecting data points from the training data with replacement. Train Decision Tree function trains a decision tree on the subset of data with the selected features and a specified maximum depth D.

The Random Forest algorithm introduces stochastic elements both through its application of the iterative partitioning method a technique rooted in sample-based

learning and its selection of a randomized subset of features for each tree. This strategy aims to promote diversity among the constituent trees and mitigate the potential for overfitting. Simultaneously, this algorithm facilitates the assessment of the significance of individual features in regression analysis. Feature importance is gauged by assessing the contribution of the features employed during the trees' training process to the overall information [26].

2.6.4 Bagging regressor algorithm

Algorithm 4 Bagging Algorithm

Input: Training data X_{train}, labels y_{train}, number of base learners T
Output: Ensemble model $F(x)$
Initialize the ensemble model: $F(x) = 0$
for $t \leftarrow 1$ to T **do**
    Randomly sample training data with replacement: $X_t, y_t \leftarrow \text{BootstrapSample}(X_{train}, y_{train})$
    Train a base learner on the bootstrapped sample: $h_t \leftarrow \text{TrainBaseLearner}(X_t, y_t)$
    Add the base learner to the ensemble model: $F(x) \leftarrow F(x) + h_t(x)$
end for
return Ensemble model $F(x)$
function BOOTSTRAPSAMPLE(X, y)
    Input: Training data X, labels y
    Output: Bootstrapped sample X_t, y_t
    Randomly sample with replacement from X and y to create X_t and y_t
    return X_t, y_t
end function
function TRAINBASELEARNER(X_t, y_t)
    Input: Bootstrapped sample X_t, y_t
    Output: Base learner h_t
    Train a base learner (e.g., decision tree) on X_t and y_t
    return h_t
end function

Figure 2.6 : Bagging Regressor Algorithm

The Bootstrap Aggregating (Bagging Regressor) algorithm, an ensemble learning approach within machine learning, operates on the foundational concept of Bootstrap Aggregating (Bagging). It amalgamates multiple estimators, deriving an outcome by averaging these individual estimators.

The operational methodology of the Bagging Regressor algorithm unfolds in the following sequence:

i) The algorithm initially takes input parameters, including the training data X_{train}, labels y_{train}, and the desired number of base learners T.

ii) Next, the ensemble model F(x) is initialized. Typically, it starts with zero or a constant value.

iii) The Bagging algorithm, also referred to as Bootstrap Aggregating, entails training each estimator with subsets generated via bootstrap sampling from the training dataset. This technique enables each estimator to concentrate on a distinct subset of the data, fostering diversity and relying on the independent contributions of each estimator. Moreover, this method allows for the simultaneous, independent computations of estimators, thereby reducing computational time.

iv) The algorithm initiates a loop to train a specified number of base learners. In each iteration, a random sample with a replacement (Bootstrap sample) is drawn from the training data. A base learner (e.g., a decision tree) is then trained on this sample, and the trained learner is added to the ensemble model. This step, performed with random sampling, aims to increase the diversity of the base learners, enhancing generalization performance.

v) Once all base learners are trained, the ensemble model is completed and returned. The Bagging algorithm aggregates the predictions of base learners to create a more reliable and generalized model. Bootstrap sampling and random sampling are fundamental principles that contribute to the effective operation of Bagging.

As we can see in Figure 2.6, we use two functions. The CALCULATENEGATIVEGRADIENT function takes as input the previous model's predictions $F_{\text{t-1}(x\_i)}$ and the true labels y_{i}. It calculates the negative gradient of the loss function, denoted as g_{t}, which represents how the loss changes concerning the previous model's predictions. The TRAINWEAKLEARNERfunction takes the training data X_{train} and the negative gradient g_{t} as input. It outputs a weak learner h_{t}, which is trained on the negative gradient. The purpose is to create diverse weak learners that collectively contribute to improving the overall model.

These functions are crucial components of the Bagging algorithm, contributing to the calculation of negative gradients and the training of weak learners, which are then aggregated to form the ensemble model.

A pivotal advantage of the Bagging Regressor lies in its capacity to deliver superior estimation performance. This is achieved through the amalgamation of multiple estimators, leading to more robust and reliable predictions. By mitigating overfitting through the training of estimators on random sub-samples, it enhances the model's generalization ability and augments performance on new data points [27].

Bagging (Bootstrap Aggregating) and Gradient Boosting stand as distinctive paradigms within the realm of ensemble learning algorithms. Bagging orchestrates the amalgamation of multiple independent learning models, each honed on randomly sampled subsets (bootstrap samples). The ensuing predictions are then aggregated through mechanisms like averaging or the election of the most prevalent class or value. Bagging strategically addresses the amelioration of high-variance models, preserving independence among the constituent models assuming paramount significance. Conversely, Gradient Boosting takes a sequential approach to integrate weak learning models. Each model in the sequence is meticulously trained to rectify the errors of its predecessor, with the overarching objective of minimizing the ensemble's cumulative errors. Gradient Boosting emerges as a tool of choice for enhancing the performance of low-bias models, demonstrating particular efficacy in discerning intricate relationships latent within datasets. However, it demands meticulous hyperparameter tuning and may entail time-intensive training processes. So, Bagging and Gradient Boosting exemplify divergent methodologies in embracing the principles of ensemble learning. While Bagging excels in fostering independence and managing variance, Gradient Boosting distinguishes itself through its adeptness in cultivating low-bias models and unraveling complex relationships within data structures.

2.7 Regularization Parameters

Regularization techniques play a pivotal role in addressing the challenge of overfitting in machine learning models. This paper delves into the exploration of two prominent methods, Ridge and Lasso regression, which effectively enhance model generalization by strategically penalizing model complexity. The mathematical underpinnings, hyperparameter tuning, and distinct effects on model parameters are thoroughly examined to provide a comprehensive understanding of their applications.

Overfitting, a pervasive issue in machine learning, occurs when models excessively capture idiosyncrasies within the training data, ultimately compromising their performance on unseen data. Regularization techniques serve as a crucial countermeasure to this phenomenon by constraining model complexity and promoting generalizability. In the realm of machine learning, these techniques are deployed to prevent overfitting, as they introduce a penalty to the model's complexity. This penalty proves instrumental in preventing the model from learning noise or random variations present in the training data.

The detrimental impact of overfitting lies in the model's diminished ability to generalize with new data. Regularization, therefore, endeavors to curb overfitting by limiting the complexity of the model. This is achieved through the imposition of penalties or regulations on the model's parameters. The widely utilized L1 (Lasso) and L2 (Ridge) regularization methods are commonly applied in regression models to strike a balance between model complexity and performance. Additionally, the elastic net technique, a combination of L1 and L2 regularization, may be employed in certain scenarios, contingent upon the number of observations within the dataset. In essence, this paper underscores the significance of regularization techniques in maintaining the delicate equilibrium between model complexity and effective generalization in the realm of machine learning [28].

2.7.1 Lasso (L1) regularization

LASSO regularization is a technique commonly used in machine learning models to address overfitting and enhance the model's generalization ability. It achieves this by incorporating an additional term into the model's loss function. This term utilizes the L1 norm to drive the coefficients toward zero or even set them precisely to zero. This characteristic enables the model to consider fewer significant or redundant features (variables), ultimately simplifying the model.

The L1 norm represents the sum of the absolute values of a vector. LASSO's utilization of the L1 norm distinctly reduces unnecessary features in the model by pulling coefficients towards zero. The primary objective of LASSO is to prevent overfitting while simultaneously improving model performance through feature selection. Another advantage of LASSO is its tendency to simultaneously shrink closely related features within a group to zero during feature selection. This can help eliminate redundant patterns in the model, leading to a more organized and meaningful representation.

$$\text{L1 regularization term} = \lambda \sum_{i=1}^{n} |w_i| \tag{2.3}$$

Complete objective function with L1 regularization:

$$\text{Objective L1(x)} = \text{Loss} + \lambda \sum_{i=1}^{n} |w_i| \tag{2.4}$$

where:

- λ**:** This represents the regularization strength, and it is a hyperparameter that you can adjust. A higher value of λ will result in a stronger regularization, penalizing the absolute values of the weights more
- **n:** This denotes the number of weights in your model. In the context of machine learning models, "weights" refer to the parameters that the model learns during the training process. When you train a machine learning

model, it goes through a process of adjusting its internal parameters to make predictions that are as close as possible to the actual target values.

- w_i**:** This represents an individual weight in the model. The sum of all w_i computes the sum of the absolute values of all individual weights in the model. The regularization term encourages the model to have sparse weights by penalizing non-zero values.

Objective L1(x) is the overall objective function that you are trying to minimize during the training of your model. The Loss term is the standard loss term, which is the measure of the difference between the predicted values and the actual values. It depends on the specific problem you are working on (e.g., mean squared error for regression, cross-entropy for classification)

2.7.2 Ridge (L2) regularization

Ridge regularization, also known as L2 regularization, is a common technique employed in machine learning models to tackle overfitting and enhance generalization performance. The primary objective of Ridge regularization is to improve the model's generalization capability by penalizing large coefficients.

In Ridge regularization, a term proportional to the squared values of the coefficients is added to the model's loss function. This term, utilizing the L2 norm, encourages the model to reduce the magnitude of all coefficients, driving them toward zero. Unlike LASSO, which tends to eliminate some coefficients, Ridge regression typically shrinks all coefficients but rarely reduces them to exactly zero. The use of the L2 norm, which involves the sum of the squared values of a vector, ensures a smooth and continuous shrinkage of coefficients. This results in a more stable and well-behaved optimization process. Ridge regularization is particularly useful when dealing with multicollinearity, where features are highly correlated, as it tends to distribute the impact of correlated features more evenly across coefficients [29].

$$\text{L2 regularization term} = \frac{\lambda}{2} \sum_{i=1}^{n} w_i^2 \tag{2.5}$$

Complete objective function with L2 regularization:

$$\text{Total objective} = \text{Loss} + \frac{\lambda}{2}\sum_{i=1}^{n} w_i^2 \tag{2.6}$$

In L2 regularization, the penalty term is based on the squared values of the weights. The factor $\frac{\lambda}{2}$ is often included for mathematical convenience, as it simplifies calculations by canceling out when taking derivatives during optimization.

2.7.3 Optimal lambda value selection

Both Ridge and Lasso regression rely on a hyperparameter, denoted as λ (lambda), to regulate the strength of regularization. Higher λ values impose stronger regularization, leading to simpler models with a potentially reduced variance but increased bias. Lower λ values allow for more complex models, capable of capturing intricate relationships in the data but potentially susceptible to overfitting. Selecting the optimal λ value is essential for achieving the desired balance between model complexity and generalization performance. Techniques such as cross-validation are invaluable for this purpose, enabling the model to be evaluated on unseen data and identify the λ value that yields the best generalization.

In tree-based regression models such as decision trees or gradient boosting algorithms, hyperparameters like the depth of the tree or the minimum samples required in a leaf node influence the model's complexity. For instance, increasing the depth of a decision tree allows the model to capture more intricate patterns in the data. However, intense trees might lead to overfitting. Similarly, adjusting the minimum samples required in a leaf node affects decision granularity, where a smaller value can result in more detailed nodes but may also lead to overfitting [30].

2.8 Evaluation Metrics

Various metrics are employed to appraise the effectiveness of models established in regression analysis. Within the realm of both the research conducted and existing literature, multiple metric types are commonly utilized for evaluating regression-based machine learning models. These metrics include Root Mean

Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE). These error measures serve as evaluative tools for the performance of prediction models. RMSE and MAE gauge the magnitude of errors, MAPE quantifies the percentage size of errors, and MSE measures the squared magnitude of errors.

The selection of a specific error criterion is contingent upon the problem domain and the characteristics of the dataset at hand. RMSE, with its emphasis on larger errors due to squaring, might be more sensitive to outliers, while MAE tends to be more robust in the presence of outliers. MAPE offers a perspective on the magnitude of errors in terms of percentages, making it useful for interpreting errors in a relative sense. MSE, by squaring errors, amplifies the effect of larger errors, which might influence its applicability based on the context and the significance of different error sizes within the dataset. The choice of which metric to utilize is thus determined by the specific requirements and characteristics of the given problem domain and dataset.

In the realm of statistical forecasting or regression, the fundamental objective resides in the development of a predictive model aimed at minimizing various error metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), and Mean Absolute Percentage Error (MAPE). However, the selection of appropriate metrics hinges upon specific criteria. When errors maintain relatively uniform magnitudes, MSE emerges as a suitable measure. Nevertheless, the appropriateness of MSE diminishes when confronted with substantial errors surpassing one or more estimated values, as it accentuates deviations by squaring errors. In such scenarios, the utilization of Mean Absolute Error (MAE) presents itself as a more fitting alternative. Instances arise where it becomes imperative to assess the potential bias inherent in an estimation method. A consistent overestimation or underestimation of model-generated values in comparison to actual values indicates a bias in the model. To rectify this, Mean Percent Error (MPE) is employed. In cases where error values vary in units, such as when one forecasting model utilizes real values while another employs natural logarithms, the Mean Absolute

Percent Error (MAPE) serves as a significant metric. MAPE proves particularly advantageous in comparing models with differing unit values, thereby mitigating potential drawbacks.

It is imperative to acknowledge that while each performance metric adheres to distinct criteria for optimal assessment, the thresholds for acceptable success rates may fluctuate contingent upon factors such as dataset size, dataset characteristics, and problem complexity.

2.8.1 Mean absolute error (MAE)

Mean Absolute Error (MAE) stands as a pivotal metric utilized in assessing the performance of predictive models. It represents the average of the absolute differences between predicted values and actual values. This metric quantifies the absolute magnitude of errors, and lower MAE values signify a closer alignment of predictions with the true values. Notably, MAE is commensurate with the unit of the predicted value, providing a direct representation of the average magnitude of errors within the given dataset.

The Mean Absolute Error (MAE) formula is expressed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.7}$$

where:

- n, represents the number of samples.
- y_i, represents actual value.
- $\hat{y}_i$, represents the predicted value.

The calculation steps for MAE are methodical and essential in computing this error measure:

i) Determine the difference between the actual value and the predicted value for each data point.

ii) Obtain the absolute value of each difference.

iii) Calculate the average of all the absolute differences, which yields the MAE.

The significance of MAE lies in its straightforward interpretation of error magnitude, offering a clear and direct assessment of the predictive model's accuracy. Its unit-based representation aids in understanding the average error scale within the specific context of the dataset. The utilization of MAE as an evaluation metric is essential in various applications, especially in regression analysis and predictive modeling, providing a meaningful and interpretable measure of error in the context of the predictive accuracy of models.

2.8.2 Mean absolute percentage error (MAPE)

Mean Absolute Percentage Error (MAPE) is a vital error measure primarily utilized in predictions associated with ratios. It represents the average of the absolute percentage differences between predicted values and actual values. MAPE characterizes the percentage magnitude of errors and is denoted in percentage form. With values ranging between 0 and 100, MAPE is commonly represented with a percentage symbol due to its typical expression as a percentage. Lower MAPE values indicate a closer alignment of predictions with the true values.

The Mean Absolute Percentage Error (MAPE) formula is expressed as follows:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{2.8}$$

where:

- n represents the number of observations.
- y_i denotes the actual value.
- $\hat{y}_i$ denotes the predicted value.

The steps involved in computing MAPE are systematic and crucial in generating this error measure:

i) Calculate the absolute value of the difference between the actual value and the predicted value for each data point.

ii) Divide the absolute value of each difference by the absolute value of the actual value.

iii) Calculate the percentage value of the obtained ratios.

iv) Compute the average of all percentage values to obtain the MAPE.

MAPE's significance lies in its direct representation of error magnitude in percentage form, specifically catering to predictions involving ratios. This metric is essential in understanding and evaluating the accuracy of predictive models, particularly when dealing with ratio-based predictions. A lower MAPE signifies a higher accuracy in predictions relative to the true values, providing valuable insights into the predictive performance within the context of ratio-related forecasting scenarios.

2.8.3 Mean squared error (MSE)

Mean Squared Error (MSE) stands as a crucial error measure employed to assess the performance of predictive models. It represents the average of the squared differences between predicted values and actual values. MSE serves as an indicator of the magnitude of errors, with larger errors being more heavily weighted due to the squaring process. Notably, MSE's value is commensurate with the square of the unit of the predicted value, thus illustrating the error magnitude in squared unit form.

The Mean Squared Error (MSE) formula is expressed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

where:

- n represents the number of observations.
- y_i denotes the actual value.
- $\hat{y}_i$ denotes the predicted value.

The computation steps for MSE entail a systematic approach to generate this pivotal evaluation metric:

i) Determine the difference between the actual value and the predicted value for each data point.

ii) Square each of these differences.

iii) Calculate the average of all the squared differences to obtain the MSE.

The significance of MSE lies in its representation of error magnitude in squared unit form. Due to the squaring of differences, larger errors are amplified, emphasizing their impact on the overall error measure. This property of MSE makes it particularly sensitive to larger errors within the dataset, allowing for a comprehensive understanding of error variance in predictive models. MSE is extensively utilized in various domains to evaluate model accuracy, providing a quantitative measure of the average squared error within the context of predictive model assessments.

2.8.4 Root mean square error (RMSE)

Root Mean Square Error (RMSE) stands as a prominent metric, particularly employed to evaluate the performance of predictive models. It symbolizes the square root of the mean of the squared differences between predicted values and actual values. RMSE serves as an indicator of the magnitude of errors, where lower RMSE values suggest a closer alignment of predictions with the true values. Notably, RMSE is expressed in the unit of the predicted value, thereby directly reflecting the unit-based error magnitude within the dataset.

The Root Mean Squared Error (RMSE) formula is expressed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.10}$$

where:

- n represents the number of observations.
- y_i denotes the actual value.
- $\hat{y}_i$ denotes the predicted value.

The stepwise computation of RMSE involves a systematic process:

i) Determine the difference between the actual value and the predicted value for each data point.

ii) Square each of these differences.

iii) Calculate the average of all the squared differences.

iv) Finally, obtain the RMSE by computing the square root of this mean, effectively representing the error magnitude in the unit of the predicted value.

The significance of RMSE lies in its square root computation, emphasizing larger errors due to the squaring of differences. This sensitivity to larger errors and its unit-based expression render RMSE a crucial metric in assessing the predictive accuracy of models, particularly in regression analysis and predictive modeling. Its unit-specific representation facilitates a comprehensive understanding of the average error magnitude within the context of the dataset, offering a robust means of measuring predictive model accuracy.

2.9 Hyperparameter Optimization

Hyperparameters are the parameters that take many different values from the user as input while the model is being trained and finds the most appropriate value by doing various experiments. Hyperparameters play an important role in training machine learning models. Each dataset and model needs different hyperparameter values and multiple experiments are performed to determine these values. Training the model sequentially with different sets of hyperparameters is called hyperparameter tuning. This process can be performed manually or by automated methods. By monitoring the results of the experiments, the hyperparameter set that provides the best performance is determined. This determination is usually made by statistical analysis methods. Hyperparameter tuning is an important and time-consuming process and a critical step for data scientists to optimize their machine-learning models.

Hyperparameters are parameters in a machine-learning model that are set before the training process and are not learned during training. They are key to the model's

architecture and settings, influencing how the model learns patterns and generalizes from the data.

The process of determining the optimal hyperparameters involves extensive experimentation. It's a critical part of model development, as different datasets and models may require specific hyperparameters to perform effectively. Examples of hyperparameters include learning rates in neural networks, depth and width of decision trees, or the regularization parameter in regression models.

Hyperparameters can significantly impact a model's performance. For instance, in neural networks, altering the learning rate can influence how quickly or slowly the model learns and converges on an optimal solution. Setting it too high might lead to overshooting the optimal solution while setting it too low might result in a model that learns too slowly. Similarly, in decision trees, changing the maximum depth or minimum number of samples per leaf can affect the model's complexity and ability to fit the data accurately.

Hyperparameter tuning involves running multiple experiments by training the model with various hyperparameter configurations. This process can be carried out manually by a data scientist or through automated methods such as grid search or random search, where the model is trained with different combinations of hyperparameters. The model's performance on a validation set or through cross-validation is then evaluated for each set of hyperparameters.

In regression tasks, hyperparameters are crucial determinants of a model's behavior. They include parameters like the regularization strength in Ridge or Lasso regression and the complexity of the model itself.

2.9.1 Hyperparametre tuning methods

The goal of hyperparameter tuning is to strike a delicate balance between two essential aspects: model complexity and the ability to generalize. Model complexity refers to how intricate or flexible the model is, while generalization is its capability to perform well on new, unseen data. Striking the right balance is vital because an overly complex model might perform exceptionally well on the training data but struggle to generalize to new data, a phenomenon known as overfitting.

The optimal configuration, or the best hyperparameter values, is determined by choosing the set of parameters that minimizes errors on the training data while still allowing the model to generalize well to unseen data. This process is typically performed using techniques like cross-validation, where the dataset is split into multiple subsets, and the model is trained and evaluated on different combinations of training and validation sets.

Hyperparameter tuning in regression involves systematic exploration of these parameters. This can be conducted manually by specifying a range of values for the hyperparameters and then training and evaluating the model for each set of values. Alternatively, automated techniques like `GridSearchCV` or `RandomizedSearchCV` can efficiently explore a defined space of hyperparameter values. We can see `GridSearchCV` and `RandomizedSearchCV` comparison in Table 2.1.

Table 2.1 : Comparison of `GridSearchCV` and `RandomizedSearchCV`

Aspect	`GridSearchCV`	`RandomizedSearchCV`
Search Strategy	Attempting all combinations in a specified hyperparameter space.	Attempting random combinations in a specified hyperparameter space.
Search Mechanism	All combinations within a specified hyperparameter space are being tried	Searching with randomly selected subset samples from the hyperparameter space instead of trying all combinations
Use Case	Suitable for small hyperparameter spaces or cases with predefined interactions between parameters	Appropriate for extensive hyperparameter spaces and scenarios where predicting interactions in advance is difficult
Advantages	Offers an exhaustive search, ensuring the discovery of the best combination	Quick and resource-efficient, especially beneficial for extensive hyperparameter spaces
Disadvantages	May incur high computational costs, particularly in large hyperparameter spaces	It does not ensure an exhaustive search for the best combination, but often performs well in practical scenarios.

`GridSearchCV` focuses on finding the best combination of hyperparameters by exhaustively trying all combinations within a specified hyperparameter range. This

method aims to achieve optimal performance by conducting a thorough search across the hyperparameter space. However, this approach can be computationally expensive, especially when dealing with large hyperparameter spaces.

On the other hand, `RandomizedSearchCV` evaluates randomly selected combinations from a specified hyperparameter distribution. This method can be more computationally efficient compared to `GridSearchCV` because it evaluates a randomly chosen subset of combinations, rather than all possible combinations. This leads to faster results, particularly in scenarios with extensive hyperparameter spaces.

The advantages of `GridSearchCV` include an increased likelihood of finding the best solution by testing all possible hyperparameter combinations. However, its disadvantage lies in the high computational cost associated with extensive hyperparameter spaces. `RandomizedSearchCV`, on the other hand, achieves faster results by reducing computational costs through random selection but comes with a lower probability of guaranteeing the absolute best combination.

In this study, due to the high dimensionality of the dataset, which means having a considerable number of features, capturing the optimal values for the parameters to be used became challenging. Therefore, to mitigate the long-term cost, reduce model complexity, and minimize the execution time, `RandomizedSearchCV` was preferred.

2.10 Additive Model Construction

The explainability of machine learning models is crucial for various reasons, especially in the context of regression models. Let's delve into the importance of model interpretability and various techniques, specifically focusing on regression models.

Model Development and Maintenances:

- **Trust and Reliability**: Understanding how a regression model arrives at its predictions builds trust. Users and stakeholders are more likely to trust a model when they can comprehend the decision-making process.

- **Fairness and Equality**: Explainable regression models facilitate the assessment of whether model decisions are fair and equal across different features or groups. This is particularly important for ensuring ethical and unbiased predictions.
- **Legal and Ethical Compliance**: In industries with legal and ethical considerations, such as healthcare and finance, the transparency of regression models is crucial for compliance. Regulatory bodies often require models to be interpretable to ensure adherence to laws and ethical standards.
- **Model Development and Maintenances**: Interpretability aids in model development and troubleshooting. If a regression model exhibits unexpected behavior, understanding the model's decisions becomes essential for identifying and addressing issues.

These models have fewer parameters and are easier to interpret.

- **Visualizing Tree-Based Models**: For tree-based models like decision trees or random forests, visualizing the trees' structures can provide insights into how each decision is made, aiding in understanding the overall model behavior.
- **SHAP (SHapley Additive exPlanations)**: SHAP values assess the contribution of each feature to the model's prediction. For regression, SHAP values help quantify the impact of each feature on the predicted outcome.
- **Partial Dependence Plots (PDP)**: PDPs illustrate the marginal effect of a particular feature on the predicted outcome while holding other features constant. They are useful for understanding the relationship between individual features and the predicted values in regression models.
- **Residual Analysis**: Examining residuals (the differences between predicted and actual values) is a simple yet effective way to diagnose a regression model. Unusual patterns in residuals can indicate areas where the model may need improvement.

In this study, we utilized the SHAP library to enhance the interpretability of the model outputs, making them more easily understandable and readily interpretable for end-users.

The proliferation of various machine learning libraries aims to enhance the interpretability of machine learning models, allowing end-users to comprehend the impact of different features on model outcomes. One such notable tool is the SHAP (Shapley Additive Explanations) library, extensively recognized in the literature for its efficacy in providing meaningful interpretations of model results [31].

The pursuit of explainable machine learning methods has witnessed significant advancement in the past decade, becoming an indispensable component of the modeling process. The Shap library, introduced in a 2017 publication by Lundberg and Lee, is grounded in principles derived from game theory [31]. Within game theory, the Shapley value is employed to estimate outcomes across alternative scenarios by quantifying the individual impact of each player on the game, elucidating how they contribute positively or negatively. Translated into the realm of machine learning, the Shap library determines the extent to which each data point and attribute influences the formation of the model's predictions. It quantifies the contribution of each feature in predicting outcomes, shedding light on the importance of different variables in the model's decision-making process.

The SHAP library finds significant utility in algorithms grounded in decision tree methods, natural language processing, and deep learning models. It serves as a pivotal tool in presenting model outcomes to end-users through various graphics, facilitating a comprehensive understanding of results from diverse perspectives.

The SHAP library uses graphics to interpret developed models through attributes, offering insights from multiple angles, both collectively and individually. These graphical representations provide a means to analyze how each feature contributes to model predictions. They offer a visual comprehension of the relative importance of different attributes, aiding in the analysis of how individual or combined features impact the model's outcomes. This multi-perspective approach significantly contributes to the interpretability and comprehensibility of machine learning

models, allowing for a more in-depth understanding of their decision-making processes.

3. MACHINE LEARNING PIPELINE

The present study adopts a qualitative methodology to conduct a comparative analysis of diverse methods and techniques utilized within the domain of machine learning algorithms, a burgeoning subdivision of artificial intelligence. This methodological choice was made to comprehensively comprehend these technologies and subject them to thorough analysis.

The applied section delineates individual studies undertaken within the research framework. Sequentially following the data collection phase, the exposition encompasses modeling and pre-modeling investigations, feature selection procedures, model comparisons, the iterative model enhancement process, and the assessment of model performance.

3.1 System Infrastructure

In data science projects, factors such as accessibility, quality, size, and diversity significantly impact the efficacy and success of the analyses conducted. The presence of these factors in datasets bolsters the reliability and depth of the analysis, while also enhancing the applicability of the results to a broader audience.

Accessibility denotes the ease of access to data sources employed in a project. Swift and hassle-free access to data significantly influences project efficiency. Ensuring easy access to data sources is an essential initial step in project planning. On the other hand, the quality of data sources relies on factors such as accuracy, timeliness, and consistency. High-quality data sources foster more precise and reliable analytical outcomes. Thus, the reliability and quality of selected data sources are paramount in influencing the accuracy and effectiveness of a project's findings.

The size of a dataset pertains to its volume and extent within a project. Larger and more comprehensive datasets facilitate in-depth analysis and provide a broader

perspective. Particularly in data-intensive methodologies like machine learning, utilizing large datasets is crucial to enhance the accuracy and robustness of analyses.

Diversity within datasets involves amalgamating various data types from different sources. Combining data obtained from diverse sources allows for analysis from multiple perspectives, resulting in more comprehensive outcomes. Working with a spectrum of data sources enriches the project by offering a broader perspective and a more holistic understanding of the analyzed information.

For machine learning models to yield accurate outcomes, thorough scrutiny and appropriate utilization of data sources related to all features directly influencing the model's results are essential. This involves multiple processes, including consolidating and merging data with distinct characteristics, identifying errors or anomalies detected through various dataset analyses, and detecting and eliminating data that might cause outliers or inaccuracies in the model's outcome. It's critical to carefully examine and integrate data featuring diverse characteristics and address any inconsistencies or anomalies within the dataset. This meticulous approach ensures the model's reliability by minimizing the impact of outliers or irregularities, thereby enhancing the accuracy and effectiveness of the machine learning model.

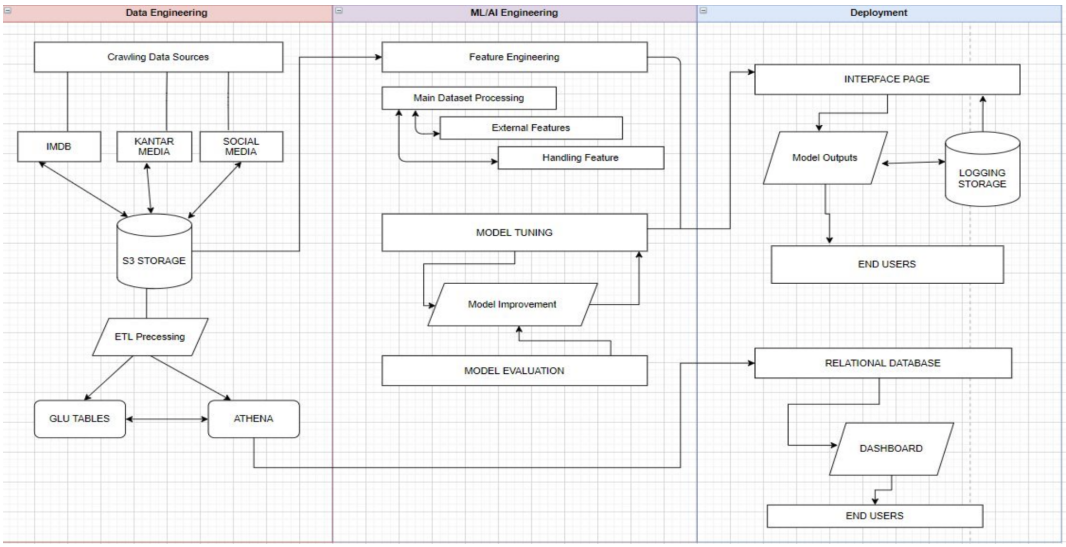

Figure 3.1 : FlowChart of the System

The main dataset alone may not always suffice to generate accurate model outcomes. In such cases, leveraging domain knowledge, external resources from diverse fields are employed to augment the dataset. These additional resources

help the model yield improved and healthier results. Furthermore, based on the model's outcomes, new features can be generated from the existing dataset to enhance the level of success, achieving more optimized and effective results. This technique, often referred to as feature engineering in the literature, involves creating meaningful and correlated internal features, enhancing the model's performance and predictive capabilities. This process aims to refine the dataset, allowing the model to operate with greater accuracy and efficiency.

For this purpose, as depicted in Figure 3.1, the system developed in the study is made accessible to the end user through specific automated procedures. This involves steps such as identifying and acquiring data sources, and storing and processing the collected data to create a meaningful dataset. Subsequently, this dataset undergoes machine learning processes, culminating in the establishment of a regression model. The model's developmental and refinement stages are then finalized. Following this, an interface is formulated for end-user access, allowing manipulation of the model. Simultaneously, user-friendly and coherent reports concerning the model results, generated using explainable machine learning techniques, are presented for enhanced comprehension and convenience.

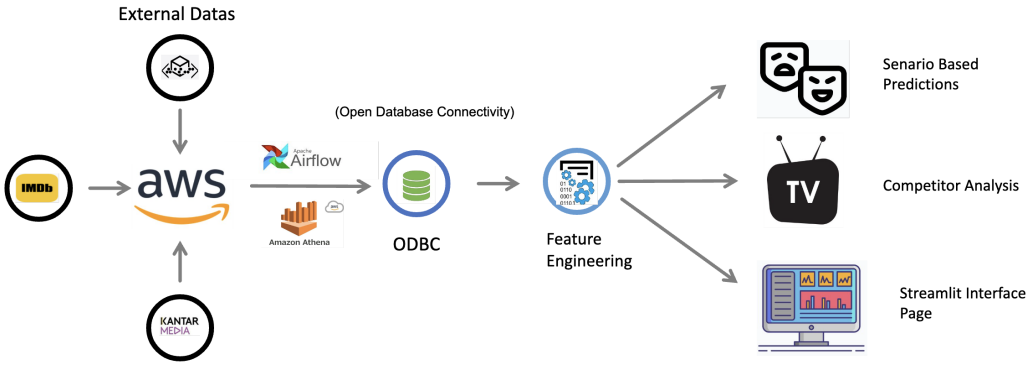

Figure 3.2 : Technical Process of the System

Maintaining the current success rate of the model relies significantly on keeping the created dataset regularly updated and enriched with new data while preserving its meaning and integrity. As depicted in Figure 3.2, this essential task is managed within the framework of cloud systems operating on Amazon Web Services (AWS). The AWS infrastructure allows for efficient and reliable management of data

updates, ensuring the dataset remains current, relevant, and robust, which is integral for sustaining the model's accuracy and effectiveness over time.

3.1.1 Apache airflow

The process of systematically updating the core dataset in the context of this research is facilitated through Apache Airflow. Apache Airflow, initially introduced in 2014 when Airbnb's workflows grew more intricate, stands as an open-source tool for scheduling, orchestrating, and monitoring workflows. The tool is preferred for its extensive and engaged user community, stemming from its open-source accessibility. It seamlessly integrates with various cloud systems such as Google, AWS, and Azure, among others, making it a popular choice. Moreover, it offers the ease of constructing adaptable workflows using the Python programming language. One of its key strengths lies in providing a visually intuitive user interface, enabling streamlined monitoring and management of workflows. This interface ensures that data streams can be easily observed and controlled, enhancing the efficiency and effectiveness of the dataset update process [32].

Workflows established within Apache Airflow are depicted as Directed Acyclic Graphs (DAGs), which represent the tasks they contain and the dependencies outlining the order of execution among these tasks. DAGs serve as a framework for managing and organizing workflow sequences. In this context, the scheduler oversees the scheduled workflows and directs tasks to the executor for execution. The executor then manages task execution by assigning and distributing these tasks among workers. Typically, Airflow's architecture consists of several key components, including a web server that utilizes a database to store metadata, enabling task and DAG audit, debugging, and state management. Additionally, a designated folder houses the DAG files, storing the configurations and definitions for the workflows and facilitating their execution within the Airflow environment [33].

3.1.2 AWS components

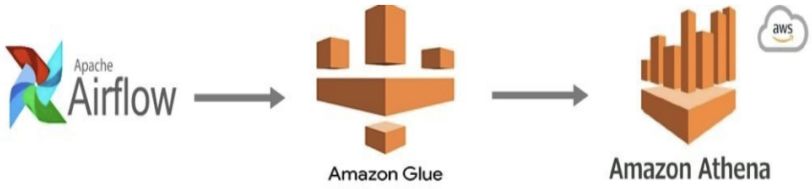

Figure 3.3 : AWS Components Workflow

The thesis's entire system infrastructure is meticulously built upon Amazon Web Services (AWS) and seamlessly operates within a sophisticated cloud-based framework. During the modeling phase, meticulous attention is given to structuring all intended data into a tabular format, a crucial step to ensure not only its meaningful representation but also to maintain semantic relationships within the dataset. As depicted in Figure 3.3, the research extensively relies on key AWS components, such as S3 for scalable storage, Glue for ETL (Extract, Transform, Load) processes, Athena for querying data seamlessly using SQL, and Sagemaker for advanced machine learning model development and deployment. This comprehensive utilization of AWS components underscores the project's commitment to leveraging cutting-edge cloud technologies for robust and efficient data processing and modeling.

- **S3**: S3 (Amazon Simple Storage Service) is used for scalable storage, hosting datasets, and allowing easy and secure access to data required for the study.
- **Glue**: AWS Glue serves as an ETL service used for data cataloging, cleaning, and transformation to ensure the data is prepared for analysis.
- **Athena**: Athena is a serverless query service enabling SQL queries on data stored in S3. It facilitates quick and cost-effective data analysis.
- **Sagemaker**: Sagemaker is an AWS service used for building, training, and deploying machine learning models. It provides a comprehensive environment for machine learning development.

These AWS components collectively form the infrastructure supporting the study, enabling efficient data processing, analysis, and machine learning model development within a cloud-based environment.

3.1.3 Relational database

A relational database is a type of database that organizes data into structures known as tables, which are related to each other based on defined relationships. It's a structured way of storing and accessing data. MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and SQLite are popular relational database management systems (RDBMS) used across various industries. Relational databases are widely used in various applications, from small-scale systems to large enterprise-level setups, due to their reliability, data integrity, and support for complex querying capabilities.

Relational Database components are:

- **Tables**: In a relational database, data is organized into tables. Each table represents one entity type or concept. For example, in a library database, there might be tables for "Series," "Authors," and "Followers."
- **Rows and Columns**: Tables consist of rows and columns. Rows, also known as records, represent individual instances of data. Columns, also called fields, store specific attributes or pieces of information about the entities. For instance, in a "Series" table, columns might include "SeriesID", "Title", "Writer", "Genre," etc.
- **Relationships**: Relationships between tables are established using keys. Primary keys uniquely identify each row in a table, while foreign keys create links between different tables. For instance, a "Series ID" in a "Genres" table could link to a "Genre ID" in a "Genres" table, creating a relationship between the two.

They ensure data accuracy and consistency through various constraints and relationships. For instance, a foreign key constraint ensures that references to data in one table exist in another table. They allow complex queries and support the

ability to extract specific information from multiple tables using SQL (Structured Query Language). Tables in a relational database are often normalized to eliminate redundancy and inconsistencies, optimizing storage and reducing the likelihood of errors. They can handle a growing amount of data and users due to their structured nature.

3.2 Preparation of Dataset

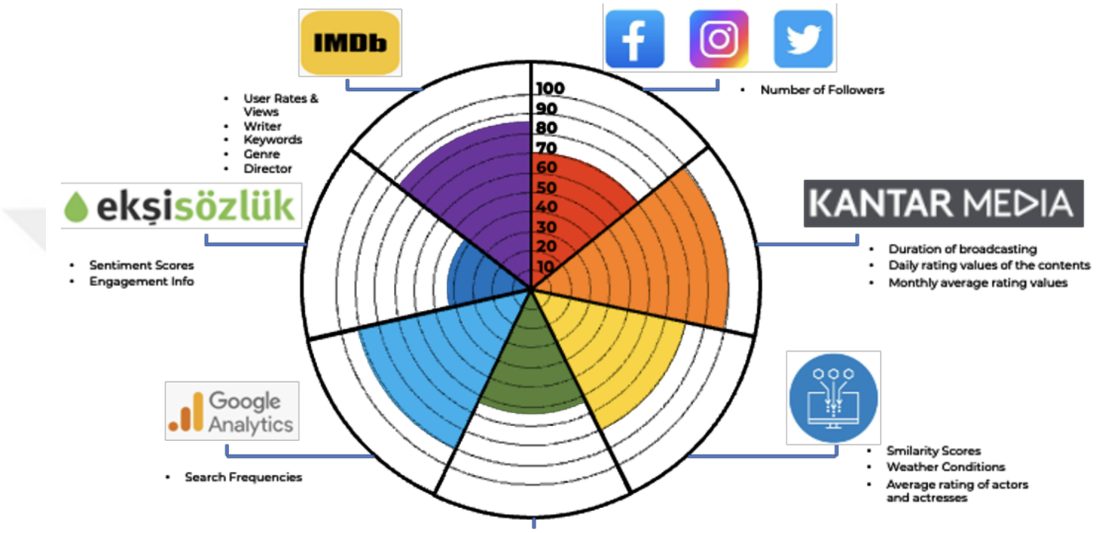

Figure 3.4 : Data Sources

Storing the data set, which is the basic point as shown in Figure 3.4, has used many data sources with different computers that will affect the rating components. Various resources in Turkey serve as repositories of rating datasets, essential for evaluating the program preferences and viewing habits of television audiences. While some of these datasets are obtainable freely and openly from their respective sources through diverse data collection methodologies, certain sources refrain from openly presenting data due to various commercial agreements within the framework of data confidentiality. Rating datasets in Turkey are primarily generated by the Television Monitoring Research Committee (TİAK), Kantar Media, and AGB Nielsen, organizations dedicated to measuring television viewership.

TİAK functions as a research institution in Turkey aimed at utilizing television viewers' preferences. Kantar Media, on the other hand, is a market research company utilized for assessing television viewing habits in Turkey. The data

sources provided by Kantar Media comprise viewership rates for television channels, the popularity levels of programs, and audience demographics. AGB Nielsen, similarly, stands as another market research company employed for evaluating television viewing habits in Turkey. The datasets furnished by AGB Nielsen encapsulate broadcasts of television channels, program engagement metrics, and audience profiles. These datasets serve as a crucial information source for stakeholders such as media companies, advertisers, and publishers. They aid in programming and content planning for television channels while facilitating advertisers in formulating strategies to reach their target audiences. In the context of this research, the primary dataset analyzed is derived from Kantar Media and TİAK at https://tiak.com.tr/tablolar, specifically, the daily data obtained, which forms the central dataset.

The daily rating values acquired for each content serve as the direct indicators for the target variable. However, critical information concerning the actors, directors, screenwriters, and producers, constituting the fundamental attributes of the target variable, was extracted via web crawling from the Internet Movie Database (IMDB) at https://www.imdb.com/, a freely accessible open-source data repository. Supplementary data, including the content's broadcast time, airing schedule, IMDB ratings, keywords, and popularity metrics for each actor, were also collected. This information was gathered during the Web Crawling phase utilizing pertinent Python programming language libraries such as BeautifulSoup.

The external contextual features on the day of content publication can significantly influence the content's rating value. The planned broadcast day's temperature and weather directly impact the viewer profile. Therefore, within the framework of traditional media, the transitional seasons of warm weather and summer are considered an interlude. At the same time, the period between September and June is termed the new broadcasting season. The primary reason for this distinction is the variation in ratings and viewer profiles due to weather conditions. Even during a typical broadcast season, a shift from a sunny to a cloudy or rainy day on the scheduled broadcast day might lead people to spend more time indoors and watch television, as they tend to be less inclined to go outside. Particularly considering

the significance of initial episode ratings, all these factors cannot be overlooked. Hence, among the most crucial data sources influencing the anticipated rating value of the content is the weather dataset. The weather data used in the study were obtained through web crawling techniques from the official website of the Turkish State Meteorological Service (MGM) at https://www.mgm.gov.tr/.

Another external factor influencing the formation of the rating value is the presence of competitive national or European league matches related to the content. The interest in a national or European league match occurring within close time intervals on the broadcast day might result in the content encountering an expected decrease in its rating value. As this situation is not consistent, it is not included in the competitor analysis simulation, allowing for a temporary "what-if" analysis for end-users. The study's dataset incorporated fixture data of national and European league matches obtained through web crawling methods from the FBRef (https://fbref.com/en/) website. The variability of interest in these matches is a significant aspect that might affect the viewership of the broadcast content. By not including it in the competitor analysis simulation, the approach enables an exploratory analysis for end-users to gauge potential scenarios or fluctuations in viewership.

Additionally, the dataset incorporated other factors that can influence the rating value, such as the social media data of each player and content, the frequency of Google searches, and sentiment analysis scores derived from the comments about them on Ekşi Sözlük (a Turkish social platform similar to Reddit). These elements are regarded as additional influencers affecting the rating value of the content.

3.2.1 Establishing relational database

The data sources and data varieties used throughout the study were determined based on the concept of the rating to be predicted, which was stated as the purpose of the study. It is aimed to create a dataset based on the elements that make up the rating concept in question and data on all situations that have the potential to affect the rating change. In the world of television, many factors that change over time, such as the constant change of viewer habits within certain time intervals,

the changing audience profile, production companies highlighting certain content suitable for the audience profile within the supply-demand balance or focusing on content with certain features, also play an important role in the change of the rating value. At the same time, situations such as weather information on the day the content is broadcast or a special event, program, or special broadcast for that day have also been determined as data sources that play an important role in changing the rating value.

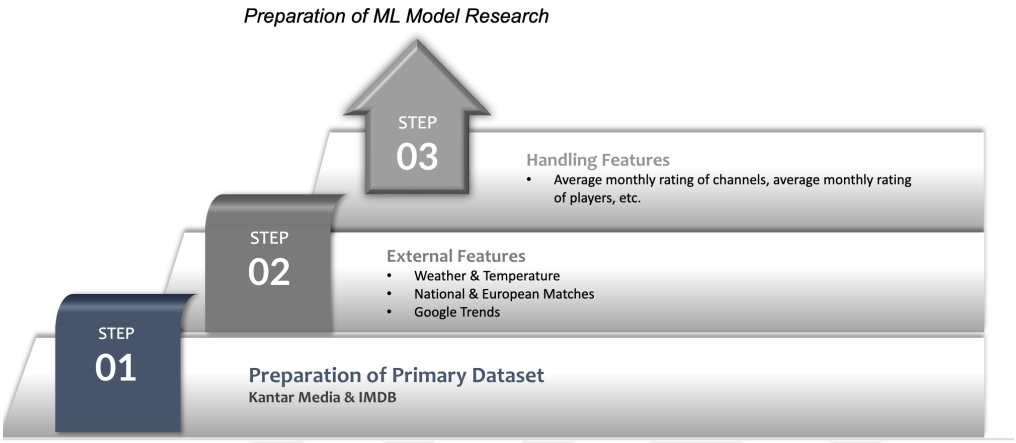

Figure 3.5 : Dataset Preparation Phases

The framework of the dataset to be created consists of two main data sources: daily rating reports and IMDB. At the same time, these two data sources are the basic components that help create the rating value. While the daily rating reports are the main data source, the IMDB data source, where detailed data for each content is collected based on these data, constitutes the other leg of the framework, even though it is an external source. In addition, it has been determined that the two most important external sources affecting the rating value are weather and event data. Sensitivity has been shown to people's need to socialize depending on the weather, to television viewing time related to spending time at home, and to changing interest situations after an important match broadcast is on another channel in parallel during the broadcast. It is aimed to improve the sensitivity of the model against rating changes that may occur in these situations. In this context, the steps to create the dataset that needs to be created for the model are summarized in Figure 3.5.

To ensure the semantic coherence, causality, and contemporaneity of the acquired data, it is imperative to organize the data collected within designated periods into coherent tables, each constituting its dataset. This segmentation aids in seamless processing. Establishing meaningful relational connections between these tables has facilitated the development of a relational database. Each table maintains its internal causal coherence and interlinks with others, adhering to semantic integrity.

The system enabling querying of each table comprises Glue structures providing suitable storage within this environment, leveraging the Athena system offering an SQL environment, and S3 buckets facilitating data storage to support each Glue structure.

One of the foundational components of the dataset is the daily rating data, encompassing reports presenting values attributed to each content across various rating types on a channel-by-channel basis on a daily frequency. These reports include details such as the content name, the channel where the content was broadcast, content typology, date and time of broadcast, and the corresponding rating value. Additionally, supplementary data such as share information and advertising rating values between the pre-and post-generation ad intervals are integrated. These rating reports are generated daily, scripted within the Airflow system, and subsequently stored in designated S3 buckets within AWS, capturing the previous day's data. Subsequently, a table is constructed from the content-specific information, encompassing content types, channel names, broadcast dates and times, and the associated rating values sourced from these reports. This table amalgamates data from daily rating reports dating back to 2017, presenting relevant content information at each stage of processing.

To maintain the comprehensiveness and currency of the data, the IMDB database serves as an important resource that enriches the rating values by providing complex metadata about the content under review. Leveraging a Python script embedded in the Airflow system, detailed metadata is collected from IMDB based on content names extracted from tables derived from ratings reports. Metadata obtained from IMDB includes IMDB name and ID, genres, user ratings, views, content summaries, plot details, popularity, casting features (actors, actresses, screenwriter,

director, producer), content keywords, similarities, etc. Contains primary content information. scores and recommendations. In addition, broadcast durations (e.g. 120 minutes, 45 minutes) and broadcast planning dates are also included in the data set. To keep data up-to-date, Python scripts are executed weekly on Mondays through the Airflow system, fetching data from the IMDB site and transferring it to the relevant storage buckets in S3.

The obtained IMDB data is divided into different tables in terms of semantic clarity and understandability. The tables created in this study include a string table containing IMDB IDs, names, and content names in rating reports; a screenwriter table containing the IMDB IDs and names/IDs of the screenwriters; an actors table consisting of gender information, IMDB IDs, and actor/actress names associated with the content; a music producer table detailing the IMDB ID of music producers and content; a director table containing screenwriters' names, IMDB IDs, and content IMDB IDs; a company table documenting content creators and IMDB content IDs; a genre table specifying content types and IMDB content IDs; a data processing table from daily rating reports; and a primary information table containing basic content metadata, IMDB IDs, release start/end years, IMDB rating values, and runtime information.

After data is extracted from IMDB, Python scripts in the Airflow system facilitate the creation of table structures that comply with semantic and integrity standards. This data is automatically transferred from S3 buckets to the Glue system through an AWS-defined browser, creating the first stages of creating a relational database. Then, in AWS's SQL environment, the Athena system verifies primary and foreign key alignments to detect the queryable form of the data, assess its concurrency, and establish cross-table connections. Any discrepancies detected initiate a backward, step-by-step correction process to fix the issues. As a result, a relational database is created that links tables using primary and foreign keys, as shown in Figure 3.6. This preliminary modeling phase establishes a structured system infrastructure that runs automatically, maintaining dataset integrity, and semantic links, and ensuring up-to-dateness before moving on to the modeling phase.

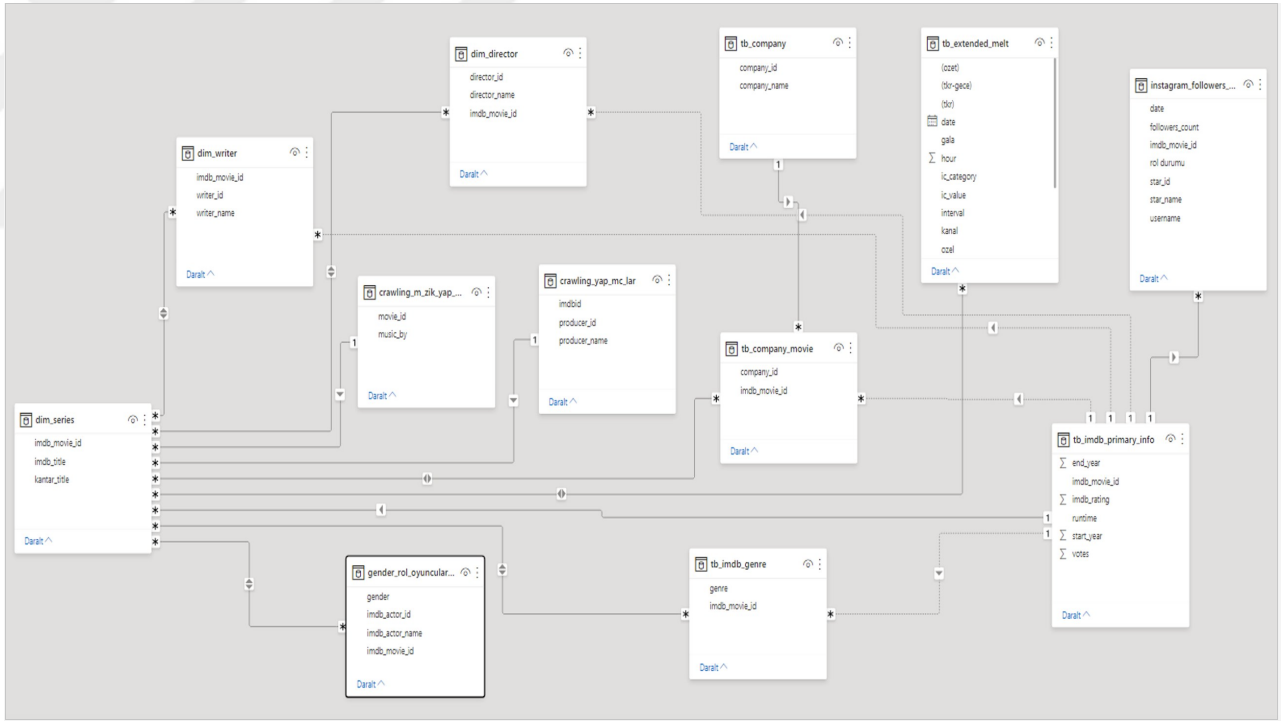

Figure 3.6 : Relational Database

The data sources comprising the primary dataset and the external data sources from which data is fetched are supervised by the respective Directed Acyclic Graphs (DAGs) in the Apache Airflow environment. Python scripts, serving as the primary source of data, are scheduled to run every week to regularly scrape data from the IMDB website. This weekly update is essential for ensuring that the dataset remains current, especially concerning potential changes in the cast teams of both new and existing series. Every Monday, following the collection of the rating dataset for that day, scripts triggered by content names in the primary dataset automatically execute. They crawl the relevant metadata from the IMDB site, and the obtained data is then structured and transferred to the data collection buckets defined within the AWS system. Similarly, the Airflow DAGs linked to scripts responsible for fetching national or European league match data from external sources also run every week. Since match data availability isn't guaranteed every week, information on its availability is recorded as dummy data on daily dates. Another DAG, which governs scripts that extract weather and temperature information from a relevant data source, runs daily to fetch weather and temperature data for the current day and stores it in the corresponding AWS data collection buckets. This approach guarantees the dataset's accuracy and relevancy by continuously updating the external data and incorporating it into the primary dataset.

The data streams, regularly fetched through Apache Airflow, are stored within S3 buckets. The AWS Glue infrastructure organizes the tables created in each script, saving them in the desired Comma-Separated Values (CSV) file format within distinct folders named accordingly. This process serves a dual purpose: establishing a data storage area while classifying the data into tables based on their specific purposes and types. These stored data files are referenced through their file extensions to identify their location. In the subsequent phases of data processing and modeling, this data is read into Amazon SageMaker. SageMaker, chosen over Python, provides a Jupyter Notebook environment, enabling efficient data processing, exploration, and model development. This environment facilitates streamlined workflows and model development, aiding in the iterative process of data analysis and machine learning model creation. It's crucial to maintain data

organization and semantic relationships within a unified database to ensure the system remains robust against potential future changes without direct impact. To accomplish this, a comprehensive dataset named "imdb" was created by employing an AWS component known as AWS Crawler, tasked with establishing a relational dataset for each table. The entire process was carried out within the Glue infrastructure.

The "imdb" dataset was formed by moving all data tables into the Glue structure, orchestrated by the AWS Crawler. This process involved collecting updated data generated from Airflow's data flows and storing it in CSV files under the relevant tables within the S3 environment. The AWS Crawler then cataloged and structured the tables, facilitating the formation of the "imdb" dataset within the Glue environment. By creating relational links between each table, the Glue environment automatically fosters a comprehensive relational database encompassing all IMDb data. This approach ensures the preservation of table structures and interconnections, contributing to a well-organized and relational database within the Glue framework.

3.2.2 Feature engineering

To enhance the predictive capability of the forthcoming model and effectively capture evolving rating trends, changing audience profiles, and shifting viewing habits over time, it became imperative to establish new features derived from the data within the relational database. The objective was to uphold semantic coherence amidst rating values that fluctuate with seasonal variations. Furthermore, the goal was to heighten the model's sensitivity towards varying rating values contingent on individual channels, production companies, screenwriters, and directors, ensuring the model's robustness across diverse scenarios. Statistical computations were utilized in crafting these novel features. In this process, various features were engendered through statistical calculations to incorporate distinctive channel characteristics into the model. Kurtosis values, signaling the predominant direction of the distribution of historical rating trends for each channel, were computed. Similarly, skewness values, indicative of the symmetry of the

distribution and its proximity to a specific distribution form, were calculated. These derived features considered internal data sources after these statistical calculations, were generated to enrich the model's understanding and prediction capabilities.

The creation of these processing functions will go through the following stages: average viewing value per channel per month, average viewing value for all players' previous content, and average viewing value per channel per day, the data set contains the average viewing value per channel per day The ratings and total ratings correspond to a three-hour time interval. The average rating values for producers and content types are calculated and treated as features. Additionally, variance values for all broadcasters, producers, genres, and actors are calculated by month, day, and interval and added as features to make the model results more consistent. Since player interactions in the model directly affect the rating values to be estimated, the average and variance rating values are calculated based on each player's average and variance rating based on the type and their combination with the producer.

The fact that each channel has its viewership trends and establishes popular habits creates its personality for each channel. To teach the characteristics of individual channels to the developed machine learning model, the alpha value of the monthly channel average (which calculates the percentage by which a channel's monthly ratings exceeds the average of all channels) and the alpha value of the monthly channel uses the channel variance, which calculates The extent to which a channel's monthly viewership exceeds the variance of all channels is characterized as a feature. Enriched data set. Parallel to these operations, the number of TV series broadcast per channel per month and the number of actors in the TV series are also important features in the dataset.

3.3 Machine Learning Modeling Process: Steps and Strategies

The study encompasses the development of two distinct models, each progressing methodically in stages.

The first model involves predicting the rating value, termed the scenario-based model. It operates based on comprehensive data factors such as director, actor,

producer, and screenwriter details of the content, along with specifics on the broadcasting channel, timing, weather conditions for broadcast, and the presence or absence of any concurrent events.

The second model revolves around calculating the rating value predicated on anticipated ratings of competitor channels that could be considered rivals for the content aired on the same day. The rationale behind this bifurcated model development approach is to obtain the scenario-based value initially and subsequently calculate the expected value after conducting an analysis involving competitor channels. This simulation environment enables the determination and interpretation of the variance between the two values.

The intention behind this dual-model approach is not only to derive separate values in these two distinct stages but also to facilitate the interpretation of the differences between these values. Additionally, the study aims to enhance the interpretability of the model outcomes by incorporating graphical representations that aid the user in understanding the results. Hence, the study unfolds in two distinct stages to effectively model and interpret the data.

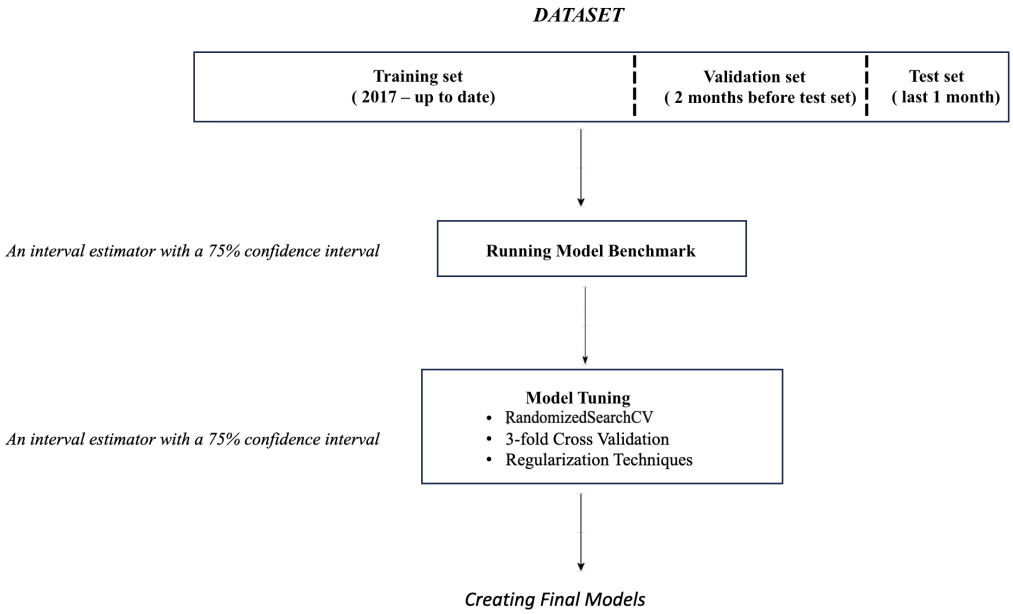

Figure 3.7 : The Stage of Preparing Dataset for the Model

To eliminate the possible data leakage problem and get more accurate prediction results during the machine learning modeling phase, as can be seen in Figure 3.7, the dataset; is divided into three sets: train, test, and validation sets. In addition, since the data in the system infrastructure is fed with new data in regular periods, to keep the final data set and the model up to date, the test set is used for 1 month before the model's working time, the validation set is for 2 months before the start of the test set, and the training set is for the part until before the beginning of the validation set. It is ensured that it is automatically created based on time. Thus, it is aimed to keep the model up to date and to train the model with sensitivities such as changing user habits and seasonality through the training set. The next step is to select the most appropriate model and follow the model improvement and development processes based on the final model.

3.3.1 Model evaluation metrics

Given that the study addresses a regression problem, the chosen models were assessed using key regression model evaluation metrics, including RMSE (Root Mean Square Error), MAE (Mean Absolute Error), MSE (Mean Squared Error), and MAPE (Mean Absolute Percentage Error). During the model research phase, the models were initially evaluated based on their benchmark scores with default parameters and without any model improvement, with a focus on the RMSE value.

The RMSE metric was preferred due to its ability to facilitate model comparability and provide a more precise basis for commentary on benchmark scores. The aim was to identify the most suitable models based on RMSE values, thus guiding the overall model strategy for the study. For instance, while there was an intuitive expectation that ensemble and boosting models might be more appropriate, the structure and inherent characteristics of the dataset required empirical validation. Comparing the RMSE values of various model types allowed for confirmation of this hypothesis.

Notably, the observation that RMSE values were higher for traditional machine learning methods, which were initially considered less suitable, as compared to ensemble models, indicated a higher error rate for the former. This empirical

evidence contributed to the selection of more suitable models and affirmed the model strategy to be adopted throughout the study.

After selecting, developing, and fine-tuning a model, we examine multiple metrics together such as MAE (Mean Absolute Error), MSE (Mean Square Error), and MAPE (Mean Absolute Percent Error) to evaluate the success of the model. The evaluation of metrics in this study does not solely rely on typical regression model outputs. Predictions of rating values consider the unique characteristic features and statistical quantities within the dataset. While the established model predicts with precision, it's essential to understand that rating values should be examined separately, considering the numerical value ranges. The difference between predicted values for higher ratings and those for lower ratings should not be solely viewed in direct numerical terms, as each rating range bears a different weight or significance.

Hence, the evaluation of model results in this study is not solely based on metric values. The assessment is extended to consider the impact of each result within different rating ranges. MAPE values indicate how well the model is established, yet MSE and MAE values detail the deviations of rating values on a unit basis. It's crucial to understand the model's performance not only in terms of these metrics but also how the model's predictions and accuracy vary across different ranges of rating values. This nuanced evaluation provides a more comprehensive understanding of the model's efficacy.

3.3.2 Auto-Feature selection: greedy heuristic method

In addition, there is an auto-feature selection process performed with a greedy heuristic before the modeling phase. This process is a method used to select the best set of features for a machine learning or data mining task. This aims to improve the performance of the model or reduce the training time by reducing the number of features. Since there are a large number of features in our dataset, it is aimed to select the features derived with some statistical values in order of importance and make automatic decisions by looking at the model performance.

The applied greedy heuristic method focuses on selecting the best features at each step and creates a feature set by proceeding step by step in this way. The derived features in question are; the monthly channel average, count of stars, monthly channel variance, average of stars' experience, average of channel day interval rating, whether is channel monthly rating distribution symmetrical, average of producers' rating, variance of stars' rating, monthly number of series aired on channel, variance of producer rating, channel monthly rating kurtosis, variance of channel day interval rating, variance of channel day interval rating, channel monthly rating frequency kurtosis, is channel monthly symmetrical.

In the Greedy heuristic method, a criterion is determined to measure the ranking of features. Our criterion in this study is to rank based on the best average MAE value. The main reason for choosing this criterion is that during the greedy heuristic, firstly, it is examined how much MAE decreases when each feature is added separately, and then the model is automatically sorted according to its improvement rates. The stopping function consists of adding the features derived from the benchmark score of the model that has not been tuned at all, respectively, based on the average MAE value in all combinations, and the features exceeding this value are automatically selected and ranked as features that improve the model. Therefore, the MAE value during the model without any tuning was calculated, and among the above-mentioned derived features, respectively, an average of producers' rating, monthly channel average rating, an average of stars' rating, channel monthly rating kurtosis, is channel monthly rating distribution symmetrical, is channel monthly symmetrical and variance of channel day interval rating features were selected.

3.3.3 Optimal model selection

The machine learning modeling process encompasses several key stages: model selection, methodologies for model enhancement and development, and evaluation of model outputs, following an extensive search for the most suitable model on the finalized dataset. To ensure a more reliable and consistent outcome within the scenario-based rating estimation model (the first model), an additional confidence

interval model referred to as interval estimation was established. This interval estimation model utilizes quantile regression methods, producing a confidence interval for the rating value, rather than a single point estimate. In contrast, for the rating value derived from the second model, focused on competitor analysis, a point estimation model is utilized. This model generates and presents a single value as its output, indicative of the anticipated rating value.

The search for a suitable model has progressed within the framework of progressive-order logic. Based on the definition of the problem, Stochastic Gradient Descent (SGD), Support Vector Regression (SVR), and K-Nearest Neighbors (KNN) algorithms were determined as the most suitable traditional machine learning methods. The KNN algorithm was the one that gave the fastest and easiest benchmark score due to the nature of the data set obtained, the correlation relationship between the features, and the presence of predictors with different features. For the same reasons, the results of the KNN model in the model research section have led us to focus on the Random Forest Regressor, which is a tree-based learning algorithm, LGBM, Catboost, and XGBoost, which are boosting methods, and the Bagging Regressor, which is a bagging algorithm. During the search for the most suitable model, interpretations were made based on the model outputs and the characteristic features of the dataset, and the most appropriate model selections were investigated.

The ensemble learning model is the most suitable machine learning method both technically and in terms of the definition of the problem we are considering as it deals with the final predictions made by creating sub-datasets from the dataset and combining the predictions of weak models, its function it is maintained in parallel and independently of each other. Moreover, they stand out during the model search phase by normalizing complexity and sophistication in the dataset to achieve higher and consistent results. To sum up, basic algorithms such as SGD, SVR, and KNN are used to evaluate the basic properties of the data set, while bagging regressors and boosting models are more suitable for handling the complexity of the data set and obtaining higher accuracy. In this context, the LGBM algorithm to be used in scenario-based rating estimation and the Catboost algorithm to be used in

competitor analysis rating estimation were chosen as the most suitable models. The success performance scores for these selected models are elaborately explained in Chapter-4 of this study.

3.3.4 Model Maintenance

Following the selection of suitable models, the subsequent steps involved model improvement and development aimed at enhancing the success values of these chosen models and delivering more robust outcomes. These steps primarily included the utilization of cross-validation methods and optimizing hyperparameters and regularization parameters.

The objective behind these steps was to mitigate potential issues such as overfitting or data leakage problems, ensuring more reliable and consistent results. Cross-validation techniques aid in validating the model's performance, while hyperparameter and regularization parameter optimization aim to fine-tune the model's settings to achieve the best possible performance without overfitting or introducing biases due to data leakage issues. The goal was to establish models that not only performed well with the current dataset but were also more adaptable and generalizable to new data.

The LGBM model utilized in the first phase involved a quantile regressor model based on a tree-based boosting algorithm, specifically working within a 75% confidence interval. In the fine-tuning phase, `RandomizedSearchCV` served as the method for hyperparameter optimization, integrated with regularization and cross-validation techniques to prevent overfitting.

This optimization process involved several steps:

(a) Employing 3-fold cross-validation for tuning regularization parameters such as maximum of depth, number of estimators, and number of leaves.

(b) Setting evaluation metric as RMSE and using X_val, y_val sets as evaluation sets.

(c) Exploring optimal parameter combinations via `RandomizedSearchCV`, involving a minimum of children samples, a minimum of children weight,

subsample, learning rate, values of regression alpha, values of regression lambda, among other parameters.

(d) Training the model using quantile values (0.25, 0.5, and 0.75) and defining these values as alpha, while using "quantile" as the metric parameter in each iteration.

The hyperparameter optimization process involved 200 iteration parameters, 3-fold cross-validation, and a scoring parameter with a negative root mean squared error value. Ultimately, the best model emerged based on the most optimal parameter combination identified during the optimization.

Given the 75% confidence interval for these models, three separate model outputs were generated: the lowest model, middle model, and upper model, each with distinct parameter combinations. Among all hyperparameters input for these models, certain parameters stood out as most impactful:

(a) Class weight, determining the weight of each class.

(b) Regularization parameters: maximum of depth, a minimum of children samples, a minimum of split gain.

(c) Number of estimators, which dictates the preferred model iteration in the boosting structure.

(d) Number of jobs, affecting the machine's core usage.

(e) Number of leaves, providing detailed tree structure information.

(f) Optimal values for values of regression alpha and regression lambda, used for model regularization.

(g) Subsample parameter, controlling the randomness in the model within a range of 0-1.

MAE, MAPE, and MSE values were calculated for each model output to facilitate comparison and verify the control over the potential overfitting status. This comprehensive approach aimed to ensure the most effective parameter settings for the models and validate their performance against the target metrics.

The second model, the Catboost model, centered on competitor analysis and focused on point estimation, generated a single output. Similar to the approach used in the LGBM model, hyperparameter optimization and regularization parameter tuning was conducted for the Catboost model. These procedures involved utilizing the same parameters and values, incorporating 3-fold cross-validation and `RandomizedSearchCV`. The hyperparameters employed in the Catboost model and their characteristics were aligned with those used in the LGBM model. Through this process, the best model was selected based on the most optimal parameter combination that resulted in the highest-scoring performance. The performance of this selected best model was assessed by considering MAPE, MAE, and MSE values. These metrics provided a comprehensive evaluation of the model's success in terms of prediction accuracy and general performance.

The best models and best parameters in both model results were listed and the compatibility between them was examined. In addition, the consistency of the model development process was checked by not limiting it to only the best parameter or model, but by keeping the results and parameter values of all tried different parameter combinations in a data frame.

The success metrics of the developed model, the fine-tuning parameter values, and the interpretation of these values are explained in detail in Chapter 4 of this thesis.

3.4 Additive Model Construction

In the realm of applied machine learning models, the output is represented by the predicted rating value. Various factors, features, and patterns contribute to creating this predictive value. However, the effectiveness of these elements can vary depending on the composition of the training dataset used during model training. Rating values are notably influenced by viewer habits and the evolving audience profiles during specific periods referred to as broadcast seasons or when analyzed at a channel level.

It's essential to recognize that these circumstances are not static when the models are executed. The constantly evolving and dynamic nature of these factors makes it challenging to interpret and understand model outputs more easily. While feature

importance charts automatically generated from Python libraries can offer insights, they might not provide a comprehensive understanding of the model's outputs.

To overcome this challenge and gain a more detailed understanding, the Shapley Additive Explanations (SHAP) library, widely used for its capabilities, has been employed. The SHAP library provides a more nuanced and detailed commentary on the model's outputs, facilitating a deeper understanding and interpretation of the model's inner workings and predictions. This tool allows for a more comprehensive examination of individual features' impacts on the model's predictions, enhancing interpretability and insight into the model's decision-making process. The main purpose of the SHAP library analysis is to report the rating value obtained from which features and to what extent to the end user. Thus, when the model is run directly, the user will be able to examine which features are more important and effective with understandable graphics [34].

The Shap charts employed in the study encompass several essential types, namely:

(a) **Summary Plot:** This visual display isolates and illustrates the effect of each feature on the prediction result.

(b) **Cumulative Summary Plot:** This plot showcases how the collective impact of different features contributes to the prediction result. It starts from the baseline impact of each feature and demonstrates the cumulative effect of additional features.

(c) **Force Plot Graphics:** These graphs provide a summarized view of feature impacts, emphasizing the most influential features.

In these SHAP graphs, the feature rankings are organized to highlight the top 10 most impactful features. The values within these graphs represent the extent to which each feature affects the prediction result. These values are generated through the calculation algorithm implemented in the SHAP library.

The Shapley values, as calculated by this algorithm, originate from a selected prediction point, which commonly is the average of the model's output predictions. For each value of a feature, a positive Shapley value is assigned if it enhances

the model's performance and a negative value if it diminishes it. The theoretical foundation of Shapley's values relies on game theory. It involves the creation of coalitions that contain different combinations of features within the dataset. The average Shapley value for each feature is calculated by iteratively adding and removing each coalition, analyzing the rate of change in predictive outcomes. This methodology determines the influence and role of features in shaping the model's outputs and the formation of predictive values.

In the study's first model LGBM for the scenario-based model, the graphs depicted the impact of all features in the dataset and were presented to the end user. However, in the second model as a Catboost for opponent simulation, which focused on competitor analysis, a different approach was taken. Graphics were developed to display how competing channels influenced the forecasting results. These graphs, specifically designed for this analysis, allowed users, even those without technical expertise, to draw inferences and provide comments based on the Shap graphics present in both model outputs.

This methodology, equipped with a reporting feature, significantly enhances the explainability of the models used. It effectively communicates to users the extent and manner in which various features affect the forecasting results. By providing clear and easily understandable visualizations, the method enriches the interpretability of the models, enabling end users to grasp and assess the influences of features on the predictive outcomes.

3.5 Creating an Interface Page

Data scientists bear the responsibility of overseeing the comprehensive data science process, leveraging creative approaches to craft and develop intricate models. Following this progression, the dissemination of their product to stakeholders becomes essential for collaborative input and feedback. These stakeholders encompass a diverse spectrum, including clientele, colleagues, or individuals situated in various locations, spanning across cities or countries. The dissemination of the model requires its deployment on the web, representing a pivotal stage after

the successful construction of a data science model. Engagingly presenting this work holds the utmost importance.

Streamlit library is an open-source Python library and the primary utilization revolves around markdown and Python, supplemented by support for HTML and CSS. This diverse and flexible combination equips users to create visually captivating web applications. Users can personalize their applications, deploy models, and facilitate testing of the functionalities by others.

Streamlit provides a comprehensive solution for crafting an interactive environment conducive to the development of web applications, enabling the demonstration of work to colleagues or customers with compelling content. This platform facilitates the acquisition of user input values, allowing for dynamic adjustments of results in response to their inputs. Streamlit serves as an invaluable tool for translating conceptual notions into web-based platforms. Noteworthy for its user-friendly installation process, it enables the seamless creation of web applications.

The models employed in the study derive predictions based on the dataset's characteristics. However, the end user could not manipulate these models. The necessity to generate output values derived from individual inputs when different combinations were selected prompted the development of an interface as an extension to the research. The primary aim was to offer users the chance to explore variations in model results by conducting various experiments through the interface.

This interface allows users to interact with the models, enabling them to input various combinations and observe how these choices impact the model's predictive outcomes. By empowering users to experiment with different input variables, the interface serves as a tool for users to gain insights into how alterations in data or parameters influence the model's predictions.

The interface developed for the study was constructed using the Streamlit library. The primary rationale for selecting the Streamlit library is its open-source and Python-based platform. Leveraging Python as the underlying language offers the advantage of facilitating rapid page design and quick adjustments when needed.

The interface, organized into distinct pages for both models, features two tabs. The first tab serves as the page for making metadata selections related to scenario-based rating predictions. In this tab, users can configure various input parameters that impact the model's predictions. The second tab is designed as the page that furnishes prediction values by taking into account the anticipated rating values of rival channels, which are provided by the user. These tabs provide an organized and user-friendly interface for users to interact with the models and access prediction results based on their preferences and inputs.

3.5.1 Scenario-based prediction screen

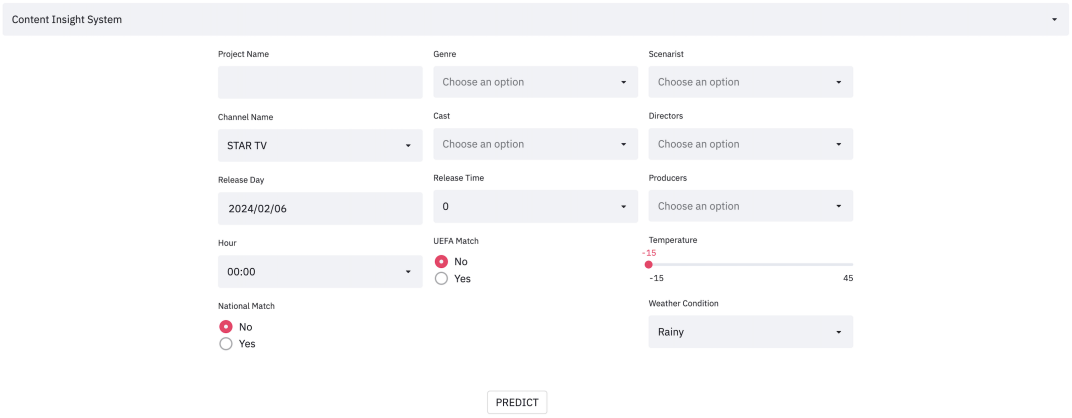

Figure 3.8 : The First Model Input Screen

In the first tab of the interface, Comma-Separated Values (CSV) files encompassing lists of actors, directors, mainstream media channels, screenwriters, prime-time hours, and producers used during the training of the scenario-based model, along with versions of the models saved in pickle format, are loaded into the interface file. Subsequently, the necessary front-end creation commands are initiated, allowing users to interact with the interface via the local prompt.

As displayed in Figure 3.8, users are presented with various options such as cast members, directors, screenwriters, and producer information, broadcast date and time, air temperature, weather conditions, the possibility of national or European league matches on a specific day, genre, and broadcast duration. Users can select various combinations for each input, forming an input list for different selections.

For each chosen input, the pre-trained model, stored in pickle format and incorporated with the relevant selections from the background CSV files, is activated. It then generates the prediction result, displaying it on the screen with a 75 percent confidence interval, based solely on the provided inputs. Furthermore, akin to model training, the interface generates graphs on the screen via the Shap library, elucidating the predicted values and providing explanations for their forecasts. This empowers users to gain insights into the factors and influences behind the model's predictions.

3.5.2 Competitor analysis and prediction screen

Moving to the second stage, the competitor analysis page aims to simulate the prediction value based on various combinations, assuming a scenario where a particular program is on air. The initial step involves obtaining expected rating values for the day from the user, particularly for rival channels in mainstream media, assuming they are competitors.

Figure 3.9 : The Second Model Input Screen

As depicted in Figure 3.9, the interface is designed to update information on competitors based on the channel details entered by the user on the previous screen. It visually represents retrospective rating values and TVR (Television Viewer Ratings) of channels, excluding the selected channels, for the specified day. This visual representation serves to guide the user in inputting the required values.

The second model is then triggered, utilizing the additional TVR value and rating values provided by the user. Following the competitor analysis simulation, the predicted value is displayed on the screen using explainable machine learning methods. This process allows users to compare and interpret the disparity between two estimated values derived from different combinations. The interface facilitates an interactive and comparative assessment, enabling users to glean insights from varying scenarios and their impact on predicted outcomes.

4. RESULTS AND ANALYSIS

Throughout the study, these processes followed the machine learning pipeline cycle, including searching for the best-fitting model and developing and refining the selected best-fitting model. The developed roadmap is created intuitively based on the analysis of the dataset, its characteristics, and the correlations between the predictor variables. These strategic steps are developed by gaining insights from success measures applied to regression problems.

Furthermore, the study focuses on providing end-users with easy-to-understand insights via Shap value-derived charts. These visual elements are critical to representing machine learning models at an interpretable level, enabling a complete understanding of the inner workings of the model. This comprehensive approach aims to ensure transparency and interpretability of model results for end users.

4.1 Appropriate Model Selection

Table 4.1 exhibits the metric outcomes of the models assessed during the quest for the most fitting model. The inclusion of various success metrics alongside RMSE in the table is attributed to the profound impact of significant variations between high and low ratings on the overall success rates. It became evident that while traditional machine learning methods such as KNN were initially tested, they inherently displayed similarities with the predisposition of the dataset towards ensemble models.

Moreover, the evaluation strongly suggested that a tree-based structure would offer more precision, whereas boosting algorithms would be more operationally effective. Analyzing the RMSE and other success metrics presented in the table distinctly indicates that the LGBM model displays the least error rate, positioning it as the most appropriate choice.

Table 4.1 : Model evaluation scores for model research process

Models	RMSE	MAE	MSE	MAPE
KNN	3.2	1.1	2.3	44.8
Random Forest	2.1	0.9	1.2	44.1
Bagging Regressor	2.0	0.9	1.1	35.8
XGBoost	1.5	1.0	2.1	30.5
LGBM	1.2	0.8	2.0	21.3
Catboost	1.2	0.9	1.8	20.8

As depicted in Table 4.1, LGBM emerges with the most favorable performance compared to CatBoost in the process of identifying the suitable model, leading to the selection of LGBM for the first model and CatBoost for the second model. The nature of the output value, which relies on various independent variables with diverse properties, indicated the necessity of employing a boosting algorithm founded on ensemble learning in the model's architecture. Thus, leveraging multiple algorithms through boosting logic aims to generate an extensive array of estimation outputs, with a focus on attaining the optimal output by averaging the results of the regression trees utilized in amalgamating these estimations.

The machine learning models utilized in this study underwent interpretation through the integration of outcomes derived from relevant success metrics. This interpretative process unfolded systematically during both the exploration phase and the development of the most suitable model. The identified success metrics played a pivotal role, serving as a foundational basis for the assessment and comprehension of the performance and efficacy exhibited by diverse models. Furthermore, these metrics guided the selection and fine-tuning procedures, leading to the identification of the most optimal model tailored to meet the intended purposes of the study.

During the stage of model selection, our evaluation hinges on a comparative analysis of root mean square error (RMSE) values, employed to discern nuanced semantic distinctions among the considered models. The scrutiny of how effectively these RMSE values align with anticipated benchmarks lends crucial support to our ongoing model exploration endeavors. Our modeling research undertook a dual focus on semantics and technology, as evidenced by the iterative refinement

of RMSE values. The identification of independent predictors within the dataset, along with their observed redundancy, advocates for the adoption of an ensemble model as the most fitting solution. This conceptual alignment is in harmony with our methodological trajectory, wherein the gradual convergence of the RMSE value towards its optimal state is systematically achieved through our successive exploration and evaluation steps.

4.2 Improvements to the Scenario-Based Model

Table 4.2 : Model evaluation scores with confidence intervals

Models	RMSE	MAE	MSE	MAPE
Lowest Model (25%)	2.3	1.5	4.9	17.1
Middle Model	1.9	1.3	3.6	12.8
Upper Model (75%)	2.2	1.7	5.0	34.3

We observe the results of LGBM, which has been chosen as a scenario-based rating prediction model in Table 4.2. After selecting a suitable model as LGBM, and performing a model optimization process using hyperparameters, a tree-based regularization process, and a triple cross-validation process, the values of the three models with a confidence interval of 75% are shown in Table 4.2. After selecting a suitable model, the values of 3 models with a confidence interval of 75% were determined using hyperparameters, a tree-based regularization process, and a 3-fold cross-validation process during model tuning.

The preference for quantile regression over traditional mean-based regression in the first model is rooted in the recognition that the latter might not fully encapsulate the intricate dynamics between variables. Given that this initial model serves as the foundation for the subsequent model, the adoption of an interval estimator, as opposed to a point estimator, was deemed more appropriate. This choice aims to yield results that are more robust and adaptable. The generated ranges are presented to the user with a 75 percent confidence interval. This interval includes both the directly estimated expected rating value of the content and the lower and upper limits. By providing the user with a range rather than a single-point estimate, this approach acknowledges the inherent variability in the data and model predictions. Users can interpret the estimated rating value within a defined confidence interval,

offering a more nuanced understanding of the potential range of outcomes. This interval estimation is particularly valuable in the context of user interaction, where different combinations of inputs may lead to slight variations in the predicted values. By conveying the estimated value within a specific confidence interval, the model accounts for the inherent uncertainty associated with user choices and input variability, ultimately providing a more accurate and informative result to the end user.

Table 4.2 emphasizes the significance of Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) values as pivotal success metrics for the models under scrutiny. These scores typically span a range from 0 to 100, although the specific range may vary contingent on the characteristics of the utilized dataset and the targeted error threshold. In examining the characteristic trend behaviors and conducting a thorough analysis of data across various mainstream media channels within a rating forecasting system, it is imperative that all gathered external and derived features maintain both complexity and semantic data integrity, aligning with the methods employed. Furthermore, given the inherent variation in the problem at hand, characterized by sequences categorized as high-rated and low-rated, optimal success thresholds for the models can be construed within a range of 20 to 50, based on the average values derived from the tested models for this particular issue. For instance, the Light Gradient Boosting Machine (LGBM) model estimates a deviation averaging between 2.5 to 3 units concerning the average rating. Notably, the MAPE values exhibit variability based on whether the rating is categorized as high or low. As we can see in Table 4.2, The LGBM model operates with an average accuracy of approximately 87.2%, within optimal bounds, with an average deviation of approximately 1.3 units. These success rates can vary based on the densities of the data in the dataset and the hyperparameter values.

4.3 Hyperparameter Tuning

As we can see in Table 4.3, many different hyperparameter distributions are suitable for the tree structure of the LGBM Regressor with quantile models for interval estimation in `RandomizedSearchCV` with 3-fold cross-validation.

Table 4.3 : Hyperparameter Scores with Quantile Regressor

Hyperparameter	Lowest Model (25%)	Middle Model	Upper Model (75%)
colsample_bytree	0.5	0.6	0.9
learning_rate	0.2	0.2	0.2
max_depth	-1	-1	-1
min_child_samples	148	151	203
min_child_weight	100	100	0.1
n_estimators	30	400	300
n_jobs	-1	-1	-1
num_leaves	25	8	30
reg_alpha	0.5	0.5	0.2
reg_lambda	0.2	0.6	0.8
subsample	0.6	0.4	0.2

The hypermeters used are:

- `colsample_bytree` **(colsample by tree)**: This parameter determines the number of randomly sampled columns of each tree. A value of 0.5 ensures that half of each tree is based on randomly selected features.
- **importance type**: This determines how importance values are calculated. The value "split" emphasizes the importance of splits in tree nodes.
- `learning_rate` **(value of learning rate**: The learning rate controls the weight of each tree before it is added. The value of the learning rate is close to 0 ensuring a slow and reliable learning process.
- `max_depth` **(maximum depth value)**: This controls the maximum depth of each tree. A value of -1 does not limit depth.
- `min_child_samples` **(minimum children value of samples)**: Determines the minimum number of instances required to split a node.
- `min_child_weight` **(minimum weight value of children)**: This determines the minimum weight of a node. The value of the learning rate provides some form of regulation by controlling node weights.
- `n_estimators` **(number of estimators)**: Determines the number of trees to be used in total. It controls the complexity of the model.

- `n_jobs` **(value of jobs)**: It determines the number of workers to be used in the training process. A value of -1 ensures that all processors are used.
- `num_leaves` **(number of leaves)**: Determines the maximum number of leaves on each tree. The value of the number of leaves ensures that each tree has a limited complexity.
- `reg_alpha` **(value of alpha)**: Alpha value used for L1 regularization.
- `reg_lambda` **(value of lambda)**: Lambda value used for L2 regularization.
- `subsample` **(value of sample)**: Sample rate used for each training round

As we can see in Table 4.3, from the quantile regressors made after this process, the optimum parameter values for the lowest model were obtained as colsample by tree value is 0.5, importance type value is split, learning rate value is 0.2, maximum of depth value is -1, minimum children of samples value is 148, minimum children of weight value is 100. Additionally, the number of estimators value is 30, the number of jobs value is -1, and the number of leaves value is 25. If we look at the regularization values of the regression, the regression alpha value is 5, the regression lambda value is 0.0, and the `subsample` value is 0.6. Additionally, the alpha value of the model is 0.25. When we look at the middle model, the colsample by tree value is 0.6, the importance type value is split, the learning rate value is 0.2, the maximum depth value is -1, the minimum children of samples value is 151, and the minimum children of weight value is 100. Additionally, the number of estimators value is 400, the number of jobs value is -1 and the number of leaves value is 8. If we look at the regularization values of the regression, the regression alpha value is 0, the regression lambda value is 0.4, and the `subsample` value is 0.4. The alpha value of the middle model is 0.5. When we look at the upper model, the colsample by tree value is 0.9, the importance type value is split, the learning rate value is 0.2, the maximum depth value is -1, the minimum children of samples value is 203, and the minimum children of weight value is 0.1. Additionally, the number of estimators value is 300, the number of jobs value is -1 and the number of leaves value is 30. If we look at the regularization values, the regression alpha value

is 0.2, the regression lambda value is 10.0, and the `subsample` value is 0.2. The alpha value of the model is 0.75.

As we can see in Table 4.3, the optimal hyperparameter outcomes of the three aforementioned models align well within the context of quantile regressor logic. Given that all three models are based on tree structures, the construction of each tree aligns with the prescribed number of classes and requisite children. The derived importance rankings and randomness values of the chosen attributes consistently yield results, highlighting the suitability of these models concerning the dataset under consideration. Consequently, the analysis of error rates can be performed through the assessment of performance metrics when employing the specified hyperparameters for predictive purposes. July 2023 serves as the reference point for testing these methodologies.

4.4 Improvements to the Competitor Simulation

Table 4.4 : Model evaluation for Catboost model

Models	RMSE	MAE	MSE	MAPE
Catboost	2.1	1.1	1.4	15.4

As depicted in Table 4.1, the Catboost model, which aligns harmoniously with the Light Gradient Boosting Machine (LGBM), has been chosen as the primary model for conducting competitor analysis within this study. This decision was made primarily to mitigate potential issues related to data leakage. Notably, even beyond the initial model, the dataset has been enriched by incorporating both Total Viewership Rating (TVR) and rating values of rival channels. This expansion enables the estimation of values based on the ratings that content from other channels, capable of concurrent competition, may garner, concerning the content against which the actual rating value will be determined.

It is worth noting that the hyperparameter results mirror those of the LGBM model, and the compatibility between both models was a key consideration. The performance of the Catboost model was rigorously evaluated utilizing the values presented in Table 4.1, resulting in superior outcomes. The choice to go with CatBoost as the second model, rather than LGBM, is motivated by two key

considerations. Firstly, it effectively mitigates the risk of data leakage during the model training process, ensuring that the model doesn't have advanced knowledge of the values it is expected to predict. This precautionary measure helps maintain the integrity of the training process. Secondly, the choice is influenced by the higher number of categorical variables in the second model compared to the initial one. CatBoost demonstrates a superior capability to handle a larger set of categorical features, making it more adept at making accurate predictions in scenarios with an increased number of such variables. By leveraging CatBoost's inherent sensitivity to categorical data, the second model aims to enhance its predictive performance, especially in situations where the complexity of categorical variables plays a significant role in the overall prediction accuracy.

These results were achieved through the application of a well-regulated 3-fold cross-validation framework and the employment of `RandomizedSearchCV` for hyperparameter tuning, as detailed in Table 4.4. The Catboost model operates with an average accuracy of approximately 84.6%, within optimal bounds, with an average deviation of approximately 1.4 units. These success rates can vary based on the densities of the data in the dataset and the hyperparameter values.

4.5 Analysis of SHAP Charts

Within this study, the SHAP Library has been selected as the primary explainable machine learning library. The fundamental objective of employing SHAP Analysis is to provide the end user with insights into the rating values acquired, specifically detailing which features contribute and to what degree. Consequently, by implementing the Shap Analysis, end users will have the ability to discern and comprehend the significance and impact of various features, presented through user-friendly and interpretable graphics. This method allows users to directly observe and comprehend the significance and influence of different features when the model is executed.

Images of SHAP analyses showing the outputs of the LGBM model, which is a scenario-based rating forecasting model, are given in Figure 4.1, Figure 4.2 and Figure 4.3.

As an example, we can give the shape values presented to the user regarding the estimation results obtained after the trained model performance results of the model. For instance, we present the SHAP values pertinent to the estimation results following the completion of the model training. Figure 4.1 showcases a graph displaying the prediction values utilized after the classical machine learning modeling process, presenting a feature importance graph that highlights the most influential features. This chart delineates the degree to which each feature impacts the model, focusing on the top 10 most effective features. Notably, in Figure 4.1, it is evident that the parameters significantly boosting the model include the episode the number, and the variance in ratings attributed to production companies. The substantial positive dominance of the episode number information further reaffirms the appropriateness of selecting the LGBM model, particularly for content tailored for inaugural episodes.

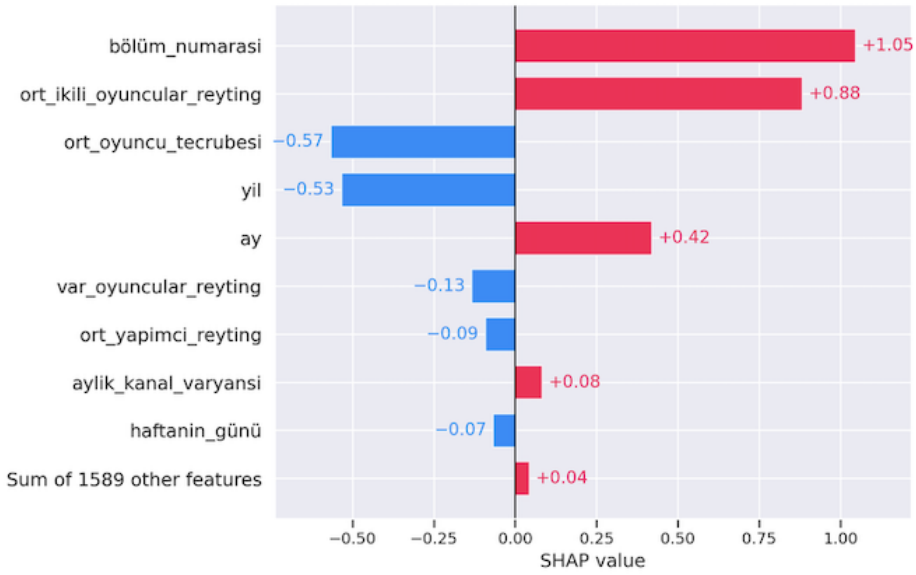

Figure 4.1 : Force Plot Graphics

Conversely, within the Shap analysis graph depicted in Figure 4.1 it is observable that parameters exerting the most adverse effects include the average rating values of doubles players, the current year and month, the average rating of production companies, the collective experience of selected players, the variance of monthly channel ratings, and kurtosis values. This analysis reveals a declining trend in rating values across all channels during the operational period of the model. The primary

factors contributing to this trend involve the broadcast season, evolving audience profiles, and shifts in channel habits.

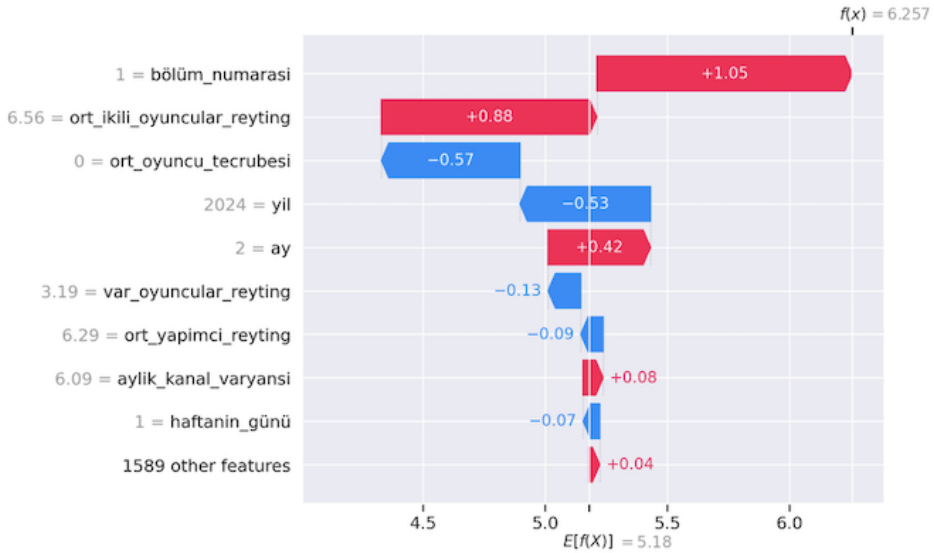

Figure 4.2 : Cumulative Summary Plot

In Figure 4.2, a comprehensive assessment was conducted to analyze the cumulative impact rates associated with the top 10 most influential features. This examination aims to delineate the extent to which the influence rate of each feature impacts other features positively or negatively, and to what degree it ultimately leads. Within the graph, the $f(x)$ value represents the estimated final rating value, whereas $E[f(x)]$ signifies the rating value that serves as the initial estimation derived directly from the raw dataset, devoid of any alterations from feature effects. Therefore, the impact value of each feature on the $E[f(x)]$ value progresses cumulatively, culminating in $f(x)$, which represents the ultimate predictive value. This cumulative process depicts how each feature's impact collectively contributes to deriving the final predictive value from the initial estimation.

Figure 4.3 : Summary Plot

As we can see in Figure 4.3, there is a graph showing how much each one of them affects based on the features in the dataset, which is considered as a summary figure.

4.6 User Interface Page

The user interface presents a screen where users can input and visualize various details. This interface includes fields for the expected broadcast content's name, genres, director and screenwriter details, channel name, scheduled day and duration of the broadcast, cast, and production company, as well as additional contextual elements like air temperature, weather conditions, and the presence of national or European league matches on the planned broadcast day.

Upon the entry of relevant information and the activation of the "predict" button, these operations facilitate the transmission of inputs to the models in pickle format. The system then automatically assigns a value of 1 to the information about the selected features and designates a value of 0 to the non-selected features. Consequently, the model is activated, generating results for the selected features, and these outcomes are displayed on the screen, akin to the depiction in Figure 4.4. This process allows users to interact with the model, enabling them to observe the model's predictions for the provided selected features through the user interface.

As illustrated in Figure 4.4 the outputs encompassing model results and graphical representations of Shap values are swiftly relayed to the end user, practically in milliseconds. This immediate transmission allows users to input various combinations on the screen and instantaneously view the results for each selection combination. Consequently, users can compare and organize their different selections, facilitating a "what-if" analysis to explore the diverse scenarios resulting from their choices.

Furthermore, the graphical depictions obtained serve as a significant aid, enabling users to interpret and comprehend the model results in a more accessible and user-friendly manner. These visuals provide a clearer understanding of the relationships between selected features and their impact on the final model outcomes, fostering a more comfortable analysis and comprehension of the predictive results.

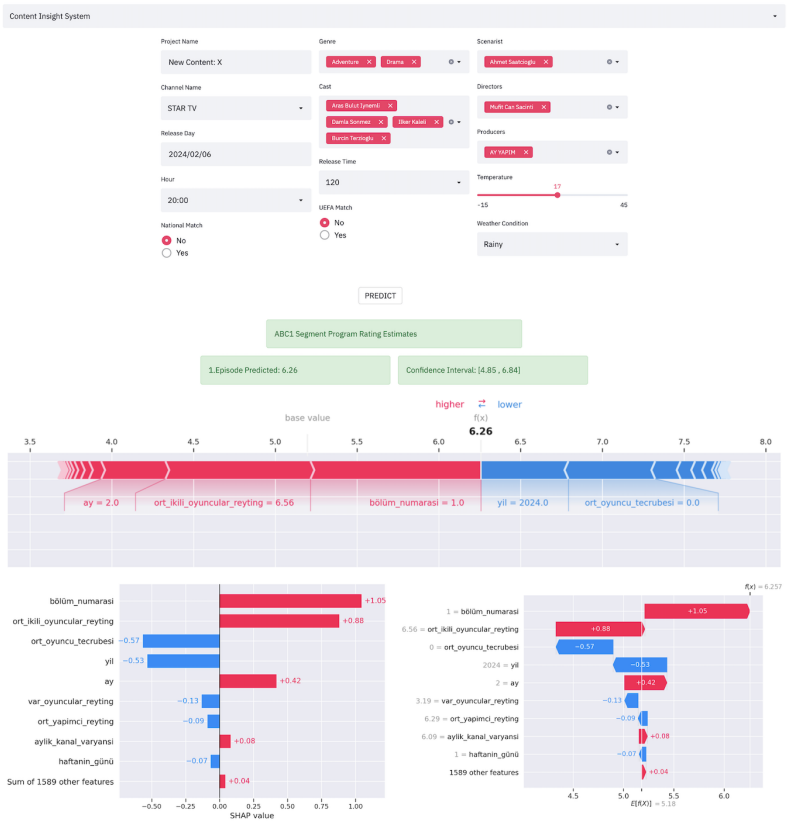

Figure 4.4 : Senerio-based Screen

Upon transitioning to the second screen, users encounter the combinations associated with the competitor analysis, following the selection of pertinent inputs, these choices are utilized as inputs for the Catboost model. Upon activating the "predict" button, these selections are processed by the model, generating an output. The resulting output screen, depicted in Figure 4.5, displays the outcomes obtained from the Catboost model based on the provided input combinations, thereby offering users insights into the competitive analysis. This interface allows users to observe the model's predictions and assessments regarding competitive dynamics based on the chosen input criteria.

In Figure 4.5, the Shap graphics were specifically generated for the additional channels selected on the screen for the competitor analysis. Among these channels, it was observed that the parameter with the most positive impact on the forecast result was the "hour," while conversely, the parameter that exhibited the most negative impact was identified as the "current year."

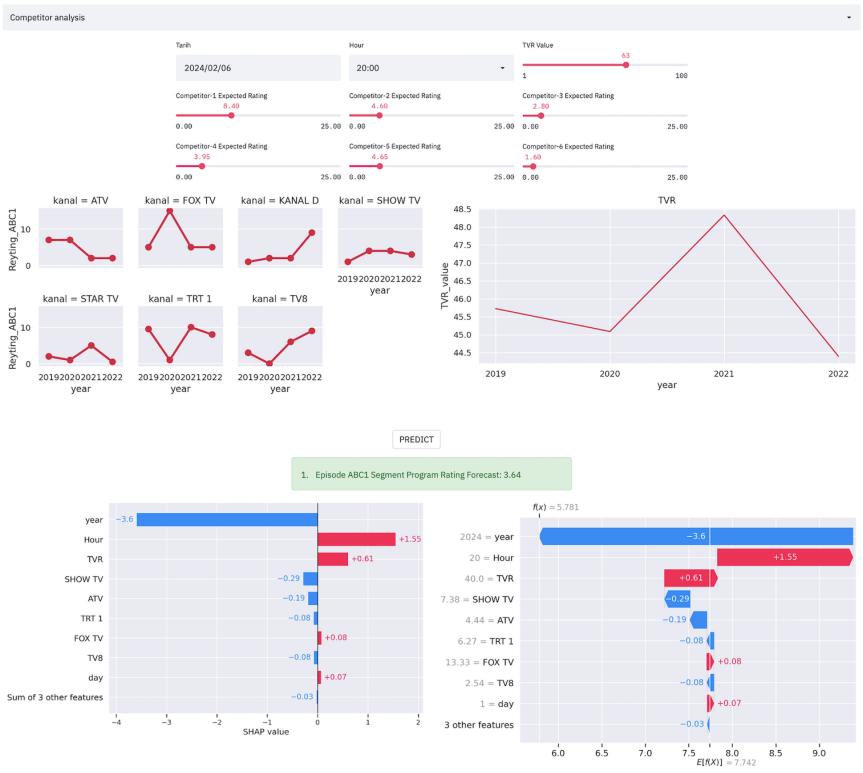

Figure 4.5 : Competitor Analysis Simulation Screen

Furthermore, the Shap graphics showcase the positive or negative effects of other channels on the estimated value, presented to the user through two distinct graphs. One set of graphs illustrates the cumulative impact, demonstrating the collective influence of multiple parameters on the predicted outcome. The other set presents isolated impacts, delineating the individual effects of each parameter on the estimated value, allowing for a more detailed understanding of each feature's contribution to the overall forecast.

Users are empowered to access various prediction outputs generated through meticulously developed models, following the phases of model preparation, model development, and model evaluation. This capability is facilitated by presenting a user-friendly screen design. By manipulating different input combinations, users can compare the model outputs for each configuration. This ability allows for a comparative analysis of the prediction results between different scenarios and enables the comparison between the outcome of the scenario-based model and

that of the competitor analysis model. Such analysis helps in assessing and understanding the variations between the models' outputs.

Moreover, this interactive approach assists in providing a more comprehensive and understandable interpretation of each model's results for the end user. By offering the ability to explore different combinations and compare various model outputs, users can gain insights into the nuances of the predictions, facilitating a clearer and more accessible comprehension of the models' outcomes.

5. CONCLUSIONS

5.1 Results of Our Experiments

As mentioned in Chapter-1, producing content in the media industry that attains maximum viewership while minimizing costs is paramount. Therefore, uncovering the solution to queries like "Which ensemble, along with what genre of content, and under what circumstances, can yield optimal ratings?" presents a formidable endeavor. So, we have devised a solution to address this issue in this study. While developing this solution, we developed two ensemble learning-based prediction models using daily rating data from 2017 to the present, along with program names and metadata from IMDB. We also integrated external data like weather conditions and European football matches.

The first model predicts scenario content values based on user choices, while the second focuses on expected ratings in a competitive analysis setting. Our system, hosted on AWS, prioritizes fresh, significant, and consistent datasets to ensure real-time accuracy.

Automated scripts connect to relevant infrastructures for regular data updates, maintaining a shared dataset. We experimented with various machine-learning models like gradient-boosting algorithms, dividing the dataset into training, validation, and testing sets to prevent overfitting.

After evaluating models based on RMSE values, we selected LGBM and Catboost for further development. We optimized parameters and conducted cross-validation for model maintenance, achieving optimal MAPE and margin of unit error. Simultaneously, these models provided a faster and more flexible solution in terms of time cost.

To ensure user understanding, we employed SHAP analysis for explainable results, presenting feature impact ratios after each model run. We also developed an

interface using the Streamlit library for easy comparison of results based on different inputs, enabling what-if analysis and competitor scenario simulations.

5.2 Practical Application of This Study

In the evolving landscape of the media industry, the increasing prevalence of digital transformation underscores the growing importance of data-driven decision-making and predictive capabilities in marketing strategies. Within this context, analytical tools like the developed rating prediction model in contemporary times hold the potential to assist media companies and broadcasting entities in crafting more effective and foresighted content strategies. The significant contributions of this study extend beyond addressing current needs, offering a promising avenue for these models to provide solutions tailored to future dynamic demands and evolving consumer behaviors.

The comprehensive scope of the present thesis allows for distilling the principal domains of inquiry, encapsulating the groundwork for prospective applications as follows:

- **Program Scheduling and Content Strategy Development**: The rating prediction models for the preproduction series play a crucial role in optimizing program scheduling and content strategies for television channels and media companies. For instance, it can be used to make strategic decisions about when and on which days certain types of programs should be aired, aligning better with the behavioral patterns of the target audience.
- **Competition Analysis and Competitor Strategies**: The model provides the ability to conduct competition analysis for a specific program or content. Understanding the strategies of rival channels or content providers and integrating this information into their content strategies can help companies gain a competitive advantage.
- **Media Buying and Advertising Strategies**: In television advertising, the rating prediction model can be employed to assess the potential impact of advertising spaces. This allows for the optimization of advertising strategies and the more effective utilization of advertising budgets.

- **Content Investment Decisions and Project Selection**: Media companies can use the rating prediction model to evaluate the potential success of future content projects before making investments. This helps them determine which types of content to invest in and focus on strategies that will make their projects more successful.
- **User Experience and Viewer Engagement**: The model can be utilized to understand how specific content resonates with a particular audience. Tailored content strategies can be created to enhance viewer experience and engagement by integrating viewer feedback.
- **Marketing and Brand Strategies**: The preproduction series' rating prediction models can be used to determine marketing and brand strategies. Data-driven strategies can be developed to reach the target audience more effectively, increase brand awareness, and optimize advertising campaigns.

These fields provide companies in the media industry with valuable insights to optimize their content and marketing strategies, ultimately gaining a competitive advantage. Additionally, similar methods can be employed in other industries looking to make data-driven decisions and better respond to customer demands.

5.3 Potential Future Work

This study highlights the potential of machine learning in media decision-making, particularly in rating construction. Augmenting the dataset meaningfully can improve model performance and analytical reliability. The aim is to aid decision-making by exploring diverse outcomes and making machine learning results clearer for users, supporting administrative decisions strategically and financially.

Expanding the foundational dataset is crucial for better predictions. Incorporating variables like emotional analysis or thematic categorization alongside existing metadata can offer deeper insights into viewer reactions.

Enriching the model with diverse factors beyond traditional data sources, like social media interactions or holiday seasons, provides a comprehensive view of viewer behavior.

Advanced feature engineering techniques, such as considering program airing time dynamics or day-of-week trends, can enhance the model's sensitivity to viewer preferences.

Exploring alternative machine learning models like Artificial Neural Networks (ANN) or Long Term Short Memory (LSTM) can optimize model performance, requiring a thorough comparative analysis.

Efficient automatic update mechanisms are vital for maintaining real-time relevance, ensuring swift integration of the latest data to sustain system vitality.

Using interpretability techniques like Local Interpretable Model-agnostic Explanations(LIME) or Partial Dependence Plots (PDP) alongside SHAP Analysis offers a finer understanding of model predictions, enhancing transparency for users.

Improving the user interface with Streamlit should focus on enhancing interactivity and user-friendliness through visual elements and filtering options for better exploration of results.

REFERENCES

[1] **Özmen, S.** (2013). *Radyo ve Televizyon Okumaları*, Derin Yayınları, İstanbul.

[2] **Radio and Television Supreme Council** (2018). *Television Viewing Trends Survey 2018*, `https://www.rtuk.gov.tr/Media/FM/Birimler/Kamuoyu/televizyonizlemeegilimleriarastirmasi2018.pdf`, accessed on: February 24, 2024.

[3] **Deloitte** (2014). *Turkish media and TV report*, `https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-telecommunications/tr-media-tv-report.pdf`, accessed on: February 24, 2024.

[4] **Bourdon, J. and Méadel, C.** (2014). *Television audiences across the world: Deconstructing the ratings machine*.

[5] **Meyer, D. and Hyndman, R.** (2006). The accuracy of television network rating forecasts: The effects of data aggregation and alternative models, *Model Assisted Statistics and Applications*, *1*, 147–155.

[6] **Sereday, S. and Cui, J.** (2017). Using machine learning to predict future TV ratings, *Nielsen Journal of Measurement*, *17*(1), 3–12.

[7] **Akula, R.**, **Wieselthier, Z.**, **Martin, L. and Garibay, I.** (2019). Forecasting the Success of Television Series using Machine Learning.

[8] **Wang, L.** (2021). Forecast Model of TV Show Rating Based on Convolutional Neural Network, *Complexity*, *2021*, 1–10.

[9] **Akgül, B. and Küçükyılmaz, T.** (2022). Forecasting TV Ratings of Turkish Television Series using a Two-level Machine Learning Framework, *Turkish Journal of Electrical Engineering and Computer Sciences*, *30*(3), 18.

[10] **Huang, H.L.**, **Lee, H.C.**, **Shu, L.S.**, **Lai, S.C.**, **Tsai, T.M.**, **Chou, S.C.**, **Liu, B.F.**, **Yin, Y.J.**, **Chen, H.A. and Ho, S.Y.** (2013). Predicting Television Ratings and Its Application to Taiwan Cable TV Channels.

[11] **Song, L.**, **Shi, Y.**, **Tso, G. and Lo, H.** (2020). Forecasting week-to-week television ratings using reduced-form and structural dynamic models, *International Journal of Forecasting*, *37*.

[12] **Nan, M.**, **Patrick, W.**, **Qin, H.**, **Wenjia, L.**, **Ying, Z.** *et al.* (2017). Prediction of Television Audience Rating Based on Fuzzy Cognitive Maps with Forward Stepwise Regression, *International Journal of Pattern Recognition and Artificial Intelligence*, *31*(7), 1–13.

[13] **Nan, M.**, **Sicheng, Z.**, **Zhen, S.**, **Xiuping, W. and Yun, Z.** (2019). An Improved Ridge Regression Algorithm and Its Application in Predicting TV Ratings, *Multimedia Tools and Applications*, *78*(1), 525–536.

[14] **Hunter, S.**, **Chinta, R.**, **Smith, S.**, **Shamim, A. and Bawazir, A.** (2016). Moneyball for TV: A Model for Forecasting the Audience of New Dramatic Television Series, *Studies in Media and Communication*, *4*.

[15] **Bell, J.** (2014). *Machine Learning: Hands-On for Developers and Technical Professionals*, John Wiley & Sons, Incorporated, Somerset, UNITED STATES.

[16] **Ghahramani, Z.** (2004). Unsupervised Learning, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.72–112.

[17] **Law, H.** (2006). Clustering, Dimensionality Reduction, and Side Information, **Technical Report**, Michigan State University, Department of Computer Science/Engineering, East Lansing, Michigan.

[18] **Ng, A.** (2000). CS229 Lecture Notes, *CS229 Lecture Notes*, *1*(1), 1–3, `https://cs229.stanford.edu/main_notes.pdf`, accessed on: February 24, 2024.

[19] **Friedman, J.**, **Hastie, T. and Tibshirani, R.** (2001). *The Elements of Statistical Learning*, volume 1 of *Springer Series in Statistics*, Springer, New York.

[20] **Cornell University** (2018). *CS 4780: Machine Learning for Intelligent Systems - Lecture Notes*, `https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html`, accessed on: February 24, 2024.

[21] **Refaeilzadeh, P.**, **Tang, L. and Liu, H.** (2009). Cross-Validation, *Encyclopedia of Database Systems*, 532–538.

[22] **Song, Y.**, **Liang, J.**, **Lu, J. and Zhao, X.** (2017). An efficient instance selection algorithm for k nearest neighbor regression, *Neurocomputing*, *251*, 26–34, `https://www.sciencedirect.com/science/article/pii/S0925231217306884`, accessed on: February 26, 2024.

[23] **XGBoost Developers**. *XGBoost Documentation*, `https://xgboost.readthedocs.io/en/latest/`, accessed on: February 26, 2024.

[24] **Microsoft**. *LightGBM Documentation*, `https://lightgbm.readthedocs.io/en/latest/`, accessed on: February 26, 2024.

[25] **CatBoost Development Team**. *CatBoost GitHub Repository*, `https://github.com/catboost/catboost`, accessed on: February 26, 2024.

[26] **scikit-learn**. *scikit-learn RandomForestRegressor Documentation*, `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`, accessed on: February 26, 2024.

[27] **scikit-learn**. *scikit-learn BaggingRegressor Documentation*, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html, accessed on: February 26, 2024.

[28] **Freund, R.J.**, **Wilson, W.J. and Sa, P.S.** (2006). *Regression Analysis*, Academic Press, second edition.

[29] **van Wieringen, W.N.** (2015). Lecture Notes on Ridge Regression, *arXiv preprint*, 1509.09169.

[30] **Tibshirani, R.** (1996). Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society*, *58*, 267–288.

[31] **Lundberg, S.M. and Lee, S.I.**, (2017). A Unified Approach to Interpreting Model Predictions, **I. Guyon**, **U.V. Luxburg**, **S. Bengio**, **H. Wallach**, **R. Fergus**, **S. Vishwanathan and R. Garnett**, editors, Advances in Neural Information Processing Systems 30, Curran Associates, Inc., pp.4765–4774, https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf, accessed on: February 24, 2024.

[32] **Apache Airflow**. *Apache Airflow Documentation*, https://airflow.apache.org/docs/apache-airflow/stable/index.html, accessed on: February 24, 2024.

[33] **Run:AI**. *Machine Learning Operations Guide - Apache Airflow*, https://www.run.ai/guides/machine-learning-operations/apache-airflow, accessed on: February 24, 2024.

[34] **Lundberg, S.M.**, **Erion, G.**, **Chen, H.**, **DeGrave, A.**, **Prutkin, J.M.**, **Nair, B.**, **Katz, R.**, **Himmelfarb, J.**, **Bansal, N. and Lee, S.I.** (2020). From local explanations to global understanding with explainable AI for trees, *Nature Machine Intelligence*, *2*(1), 2522–5839.

CURRICULUM VITAE

Name SURNAME: Burak BATIBAY

EDUCATION:

- **B.Sc.:** 2020, Yıldız Technical University, Chemistry Metallurgy Faculty, Department of Mathematical Engineering, Mathematical Engineering Program
- **M.Sc.:** 2024, Istanbul Technical University, Science and Letter Faculty, Department of Mathematical Engineering, Mathematical Engineering Program

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Batıbay, B.** and Kaygun, A. 2023. Explainable Machine Learning Methods in Forecasting TV Ratings During Preproduction. *2nd International Graduate Research Symposium (IGRS)*, May 16-18, 2023 İstanbul, Turkey.