



GOOGLE PLAY STORE DATA ANALYSIS AND APP RATING PREDICTION

MINI PROJECT REPORT

Submitted by

AJAY B (111420243001)

YUVARAJ P (111420243026)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PRATHYUSHA ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

THIRUVALLUR – 602 025

NOVEMBER 2023

PRATHYUSHA ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

THIRUVALLUR – 602 025

BONAFIDE CERTIFICATE

Certified that this project report “**GOOGLE PLAY STORE DATA ANALYSIS AND APP RATING PREDICTION**” is the bonafide work of "**AJAY B (111420243001), YUVARAJ P (111420243026)**" who carried out the project work under my supervision.

SIGNATURE

Ms. R. KANNAMMA,
ASSOCIATE PROFESSOR,
SUPERVISOR,
Department of AI & DS,
Prathyusha Engineering College,
Thiruvallur – 602 025

SIGNATURE

Ms. R. KANNAMMA,
ASSOCIATE PROFESSOR,
HEAD OF THE DEPARTMENT,
Department of AI & DS,
Prathyusha Engineering College,
Thiruvallur – 602 025

Place: Thiruvallur

Date: 28-11-2023

Submitted for the Project Viva-Voce help on At
PRATHYUSHA ENGINEERING COLLEGE, Thiruvallur – 602 025

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Our sincere thanks to **Shri. P. RAJARAO**, Chairman, Prathyusha Engineering College for facilitating us to do this project.

We are grateful to our CEO, **Smt. P. PRATHYUSHA**, for being a source of Inspiration.

We are sincerely thanking **Dr. B. R. RAMESH BAPU**, Principal, Prathyusha Engineering College for encouraging our endeavors for this project.

We are sincerely thanking our HOD and Project Coordinator **Ms. S KANNAMA**, Associate Professor, Head of Department, for supporting us in all stages of project conduction.

We also wish to extend our sincere thanks to all the committee members for their constant support throughout the review.

We wish to express our sincere gratitude to **PARENTS AND FRIENDS** for valuable help, co-operation and encouragement during this project work.

Last but not least, we wish to express our sincere thanks to the entire teaching faculty and non-teaching staff for their support.

ABSTRACT

In the ever-expanding Google Play Store, characterized by a relentless influx of new apps, achieving success has become a formidable challenge for developers. This research employs Exploratory Data Analysis (EDA) to scrutinize the characteristics of high-rated applications, focusing on crucial features such as size, installs, reviews, type (free/paid), rating, category, content rating, and price. Leveraging a comprehensive dataset sourced from the Google Play Store, this study aims to uncover hidden patterns and relationships contributing to app success. Complementing the EDA, this research incorporates data visualization using PowerBI to provide a dynamic and insightful representation of the findings. Through visually compelling dashboards, the study enhances the understanding of trends and correlations within the Google Play Store app data. Furthermore, the research extends beyond visualization to encompass predictive modeling using Python. Employing machine learning techniques, including Linear Regression, Random Forest, Decision Tree, Gradient Boosting, and K-Means, the study aims to discern intricate relationships among app features. While evaluating the model, Random Forest performs better than other algorithm with an r^2_Score value of 0.95. The amalgamation of Exploratory Data Analysis, PowerBI visualization, and Python-based modeling offers a holistic and nuanced perspective on the dynamics of the Google Play Store. The research outcomes are poised to empower developers, marketers, and stakeholders with actionable insights, facilitating informed decision-making in the competitive realm of Android application development.

TABLE OF CONTENTS

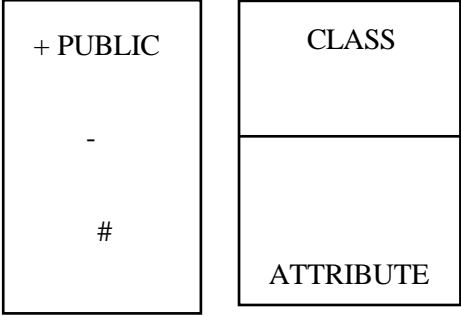
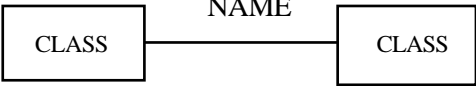

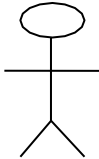
CH.NO	TITLE	PAGE NO
	ABSTRACT	4
	LIST OF FIGURES	7
	LIST OF SYMBOLS	8
1	INTRODUCTION	10
	1.1 OVERVIEW	11
	1.2 OBJECTIVE	12
	1.3 LITERATURE SURVVEY	13
2	SYSTEM ANALYSIS	15
	2.1 EXISTING SYSTEM	16
	2.1.1 DISADVANTAGES	16
	2.2 PROPOSED SYSTEM	17
	2.2.1 ADVANTAGES	17
3	SYSTEM REQUIREMENTS	18
	3.1 HARDWARE REQUIREMENTS	19
	3.2 SOFTWARE REQUIREMENTS	19
	3.3 SOFTWARE DESCRIPTION	20
4	SYSTEM DESIGN	22
	4.1 SYSTEM ARCHITECTURE	23
	4.2 UML DIAGRAM	23
	4.2.1 USECASE DIAGRAM	23
	4.2.2 ACTIVITY DIAGRAM	24
	4.2.3 SEQUENCE DIAGRAM	25



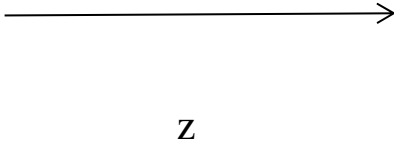
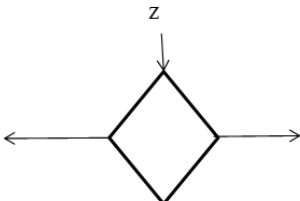
5	SYSTEM IMPLEMENTATION	26
	5.1 LIST OF MODULES	27
	5.2 MODULE DESCRIPTION	27
	5.2.1 IMPORTING LIBRARIES & DATASET	27
	5.2.2 PRE-PROCESSING	27
	5.2.3 PERFORMING EDA	28
	5.2.4 DATA MODELLING	28
	5.2.5 VISUALIZATION WITH POWERBI	28
6	TESTING	29
	6.1 UNIT TESTING	30
	6.2 INTEGRATION TESTING	30
	6.3 SYSTEM TESTING	31
7	RESULTS AND DISCUSSION	32
	7.1 RESULTS	33
	7.2 DISCUSSION	33
8	CONCLUSION AND FUTURE ENHANCEMENT	35
	8.1 CONCLUSION	36
	8.2 FUTURE ENHANCEMENT	36
	ANNEXURE	38
	ANNEXURE 1 SOURCE CODE	39
	ANNEXURE 2 SCREENSHOT	49
	REFERENCES	52

LIST OF FIGURES

FIG. NO.	FIGURE NAME	PAGE NO.
4.1	System architecture for Google App Store Anlaysis and Rating prediction	12
4.2.1	Use case diagram for Google App Store Anlaysis and Rating prediction	12
4.2.2	Activity diagram for Google App Store Anlaysis and Rating prediction	14
4.2.3	Sequence diagram for Google App Store Anlaysis and Rating prediction	15

LIST OF SYMBOLS

S.No	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represent Static relationships between classes.
			Roles represent the way the two classes see each Other.
3.	Actor		Specifies a role played by a user that interacts with the subject.
4.	Communication		Communication between various use cases

S.No	NOTATION NAME	NOTATION	DESCRIPTION
5.	Initial State		Initial state of the object
6.	Final state		Final state of the object
7.	Control flow		Represents various control flow between the states.
8.	Decision box		Communication between various use cases

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Currently the number of Android applications with rapidly growing users affects the rapidly developing economy . (The 2017-2022 App Economy Forecast: 6 Billion Devices, \$ 157 Billion in Spend & More, nd). The rapidly growing mobile phone application market is attracting the attention of many developers with more than 11 million developers already submitting their applications on the Google Play Store. Many Android app developers are still struggling to find ways to profit from their apps by including them on the Play Store. Knowing the characteristics of a successful application can be a way for developers to create successful applications.

Various characteristics can be used as benchmarks for making a successful application. Just like previous studies in this study the authors chose ranking ratings as a benchmark for successful applications, therefore the factors or features that influence application success, namely the characteristics of high-ranking applications, need to be known. Previous research on the prediction of success from applications on the google play store, researchers obtained random forest model results that predict the success of applications on the google play store with an good performance .

To classify a successful application it can be seen from the number of downloads, but many users download the application without using it. The Google Play Store displays the number of downloads based on a range, not the actual value (example: 10M + Downloads) which makes it difficult to distinguish between apps that are in the same range. Another way to identify successful apps is to analyze the comments that users enter when rating them, but many users rate

them without including comments.

In previous research on the analysis of factors that affect application ratings, it was found that applications with high ratings had lower API bug fix values compared to applications with low ratings after the data was tested with the Mann-Whitney test ($p\text{-value} < 0.0001$) and Cliff's Delta (0.37). . This shows that there is a relationship between application rankings with other factors such as changes in Android APIs and the complexity of the user interface .

The Google Play Store stands as a dynamic marketplace, hosting an ever-expanding array of mobile applications that cater to diverse user needs. Since its inception, the platform has witnessed an explosive growth in the number of apps, presenting both opportunities and challenges for developers.. The Google Play Store, a cornerstone of the Android ecosystem, has become synonymous with the proliferation of mobile applications.

OBJECTIVE

The primary objective of this research project is to conduct a comprehensive analysis of the Google Play Store ecosystem, with a focus on understanding the factors that contribute to the success of mobile applications. The project aims to leverage Exploratory Data Analysis (EDA) techniques and machine learning models to extract hidden insights, patterns, and correlations within the vast dataset of Google Play Store app data. The exploration will delve into key features such as app size, number of installs, reviews, type (free/paid), rating, category, content rating, and price, aiming to discern their impact on app success. PowerBI will be employed for effective data visualization, creating dynamic dashboards that provide a visually compelling representation of trends and correlations within the Google Play Store app data. Predictive modeling will play a crucial role, with the implementation of machine learning models,

including Linear Regression, Random Forest, Decision Tree, Gradient Boosting, and K-Means. The goal is to predict and understand the factors that contribute to the success of an app, providing valuable insights for app developers.

1.2 LITERATURE SURVEY

[1] O. Lengkong and R. Maringka, "Apps Rating Classification on Play Store Using Gradient Boost Algorithm," 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), Manado, Indonesia, 2020, pp. 1-5, doi: 10.1109/ICORIS50180.2020.9320756..

Prior studies have laid the foundation for understanding app success factors within app marketplaces. Research by O. Lengkong and R. Maringka delves into the critical features influencing the success of mobile applications, emphasizing the importance of factors such as user ratings, reviews, and download counts. O. Lengkong and R. Maringka work specifically on machine learning applications in app success prediction has provided valuable insights into the application of algorithms in understanding user preferences and app quality.

[2] I. Gunaratnam and D. N. Wickramarachchi, "Computational Model for Rating Mobile Applications based on Feature Extraction," 2020 2nd International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka, 2020, pp. 180-185, doi: 10.1109/ICAC51239.2020.9357270.

The main objective of this research is to build a platform that rate apps by feature extraction and sentiment analysis to calculate the functionality index of apps based on metrics obtained by surveying 204 mobile phone users. The 5 topmost metrics obtained from them among the 16 metrics obtained from the literature review are usability, price, and frequency of updates, ad-freeness and battery consuming level. This research focuses on selected apps in music and audio category.

[3] E. Noei, F. Zhang and Y. Zou, "Too Many User-Reviews! What Should App Developers Look at First?," in IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 367-378, 1 Feb. 2021, doi: 10.1109/TSE.2019.2893171.

This paper studied 4,193,549 user-reviews of 623 Android apps that were collected from Google Play Store in ten different categories. The results show that few key topics commonly exist across categories, and each category has a specific set of key topics. We also evaluated the identified key topics with respect to the changes that are made to each version of the apps for 19 months. We observed, for 77 percent of the apps, considering the key topics in the next versions shares a significant relationship with increases in star-ratings

[4] I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst and A. E. Hassan, "Examining the Rating System Used in Mobile-App Stores," in IEEE Software, vol. 33, no. 6, pp. 86-92, Nov.-Dec. 2016, doi: 10.1109/MS.2015.56.

Mobile apps are continuously evolving, with new versions rapidly replacing the old ones. Nevertheless, many app stores still use an Amazon-style rating system, which aggregates every rating ever assigned to an app into one store rating. To examine whether the store rating captures the changing user satisfaction levels regarding new app versions, researchers mined the store ratings of more than 10,000 mobile apps in Google Play, every day for a year. Even though many apps' version ratings rose or fell, their store rating was resilient to fluctuations once they had gathered a substantial number of raters. The conclusion is that current store ratings aren't dynamic enough to capture changing user satisfaction levels. This resilience is a major problem that can discourage developers from improving app quality.

CHAPTER 2

SYSTEM ANALYSIS

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The current landscape of app development within the Google Play Store lacks a comprehensive and systematic approach to understanding the characteristics that define high-rated applications. Developers often rely on anecdotal evidence or generic insights, leading to a lack of precision and efficiency in app development strategies. The absence of a robust system for evaluating and predicting the success factors of Android apps hinders the potential for developers to create applications that resonate with users and perform well in the competitive app market.

Furthermore, existing studies and tools may not leverage the full potential of advanced machine learning algorithms to uncover intricate patterns and correlations within the Google Play Store data. This limitation results in a less accurate and insightful understanding of the features that contribute to high app ratings. As a consequence, developers may face challenges in optimizing their apps for success, relying on trial-and-error methods rather than data-driven decision-making.

2.1.1 DISADVANTAGES

- Lacks a precise and systematic method for understanding the characteristics of high-rated Android applications.
- The existing system not fully leverage the potential of advanced ML algorithms to uncover intricate patterns within the Google Play Store data.
- Results in a less accurate understanding of the features that contribute to app success.

2.2 PROPOSED SYSTEM

The proposed system aims to revolutionize the way developers approach app development within the Google Play Store by introducing a sophisticated framework that systematically analyzes and predicts the success factors of Android applications. Leveraging advanced machine learning algorithms, specifically the random-forest classifier and the Gradient Boost Algorithm, the system will comprehensively explore features such as Size, installs, reviews, types (free/paid), rating, Category, content rating, and Price.

The system will provide developers with a user-friendly interface to input their app details, which will then be subjected to a rigorous analysis. Through the integration of a random-forest classifier, the system will identify the most significant features associated with high-rated apps, offering developers unprecedented insights into the elements crucial for success. The adoption of the Gradient Boost Algorithm will further enhance the accuracy of these predictions, ensuring a robust and reliable system for app success evaluation.

Additionally, the proposed system will present results in a visually compelling manner, incorporating data visualization techniques to help developers easily interpret and understand the identified success factors.

2.2.1 ADVANTAGES

- ensures a more nuanced understanding of the factors contributing to success.
- allows for the identification of the most significant features associated with high-rated apps, facilitating proactive decision-making.
- enhances accessibility for developers, enabling them to utilize the system effectively without requiring extensive technical expertise.

CHAPTER 3

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. This describes the projects target audience and its user interface, hardware and software requirements.

3.1 HARDWARE REQUIREMENTS

3.1.1 MINIMUM REQUIREMENTS

CPU : Intel i3 7th Gen / Ryzen 5 1600
RAM : 8GB
OS : Windows / Linux / Mac

3.1.2 RECOMMENDED REQUIREMENTS

CPU : Intel i5 12th Gen / AMD Ryzen 5600X
RAM : 32GB
OS : Windows / Linux / Mac

3.2 SOFTWARE REQUIREMENTS

Languages : Python 3.7+ , PowerBI
Modules : NumPy , Pandas , matplotlib , sklearn , seaborn
IDE : Visual Code Editor

3.3 SOFTWARE DESCRIPTION

Software Description is a technical specification of requirement of software product. This specifies the environment for development, operation and maintenance of the product.

3.3.1 Python3.7

Python 3.7 is a version of the Python programming language, released on June 27th, 2018. It is the latest stable release in the 3.x series of Python. Python is a popular high-level

3.3.2 Python3.7

Python 3.7 is a version of the Python programming language, released on June 27th, 2018. It is the latest stable release in the 3.x series of Python. Python is a popular high-level, general-purpose programming language that is widely used for web development, scientific computing, data analysis, artificial intelligence, machine learning, and more

3.4 Modules Description

NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Pandas

Pandas is a powerful open-source data manipulation and analysis library for Python. It provides data structures like Series and DataFrame, which are highly efficient for working with structured data. Widely used in data science and analytics to handle and analyze tabular data, making it an essential tool for tasks ranging from exploratory data analysis to data preprocessing.

Matplotlib

Matplotlib is a comprehensive 2D plotting library for Python that produces high-quality visualizations. It offers a wide range of static, animated, and interactive plots, including line plots, bar charts, scatter plots, histograms, and more. Matplotlib is highly customizable, allowing users to control every aspect of the plot, from colors and labels to annotations.

Scikit-learn

Scikit-learn is an open-source machine learning library for Python that provides simple and efficient tools for data mining and data analysis. It includes various algorithms for classification, regression, clustering, dimensionality reduction, and model selection.

Seaborn

Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn comes with several built-in themes and color palettes to make it easy to create aesthetically pleasing visualizations. It simplifies the process of creating complex statistical plots, such as heatmaps, violin plots, and pair plots. Seaborn is particularly useful for exploring relationships in datasets and presenting findings in a visually compelling manner.

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

A System architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

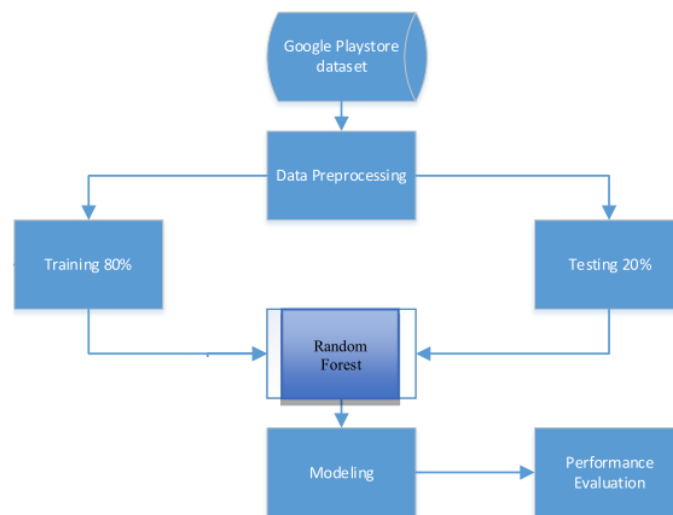


Fig 4.1 System architecture of data modelling

4.2 UML DIAGRAMS

4.2.1 USE CASE DIAGRAM

Use Cases are used to describe the visible interactions that the system will have with users and external systems.

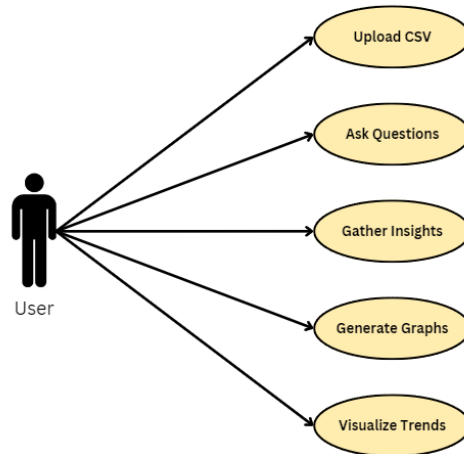


Fig 4.2.1 Use case diagram

4.2.2 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The user can use the platform to gather insights of the data and plot graphs with utmost ease.

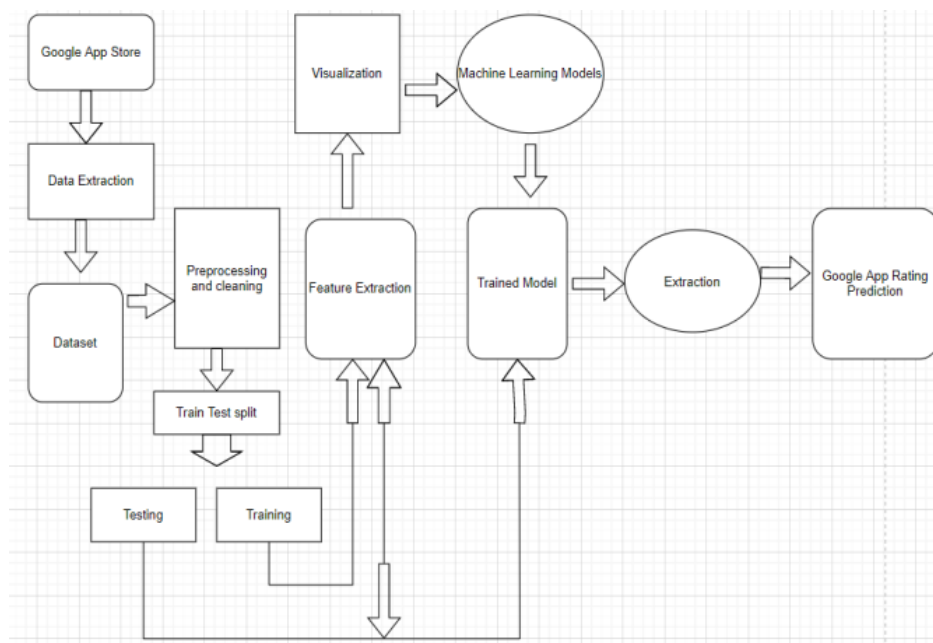


Fig 4.2.2 Activity diagram for Google Play Store Analysis and rating prediction

4.2.4 SEQUENCE DIAGRAM

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence.

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

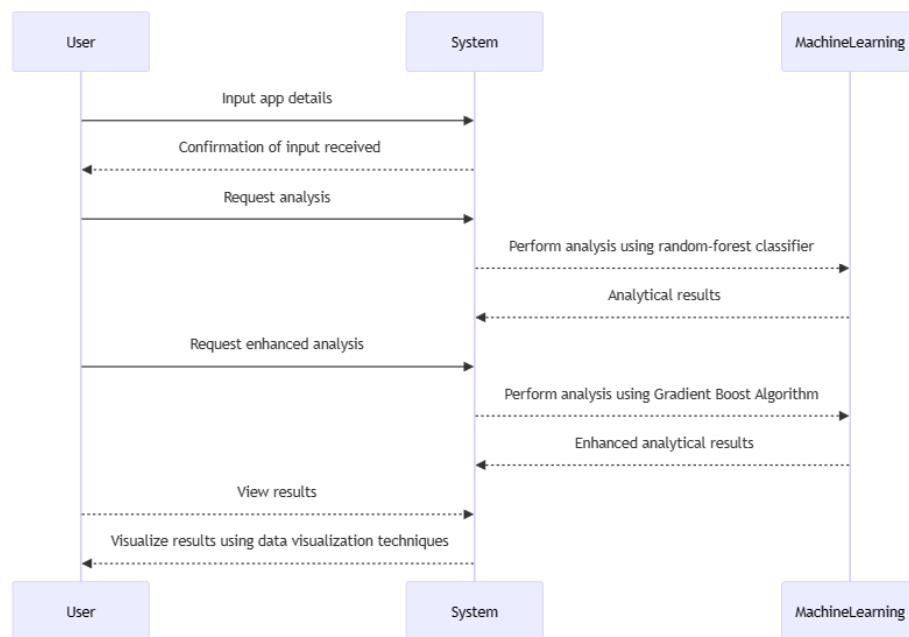


Fig 4.2.3 Sequence diagram for Google Play Store Analysis and rating prediction

CHAPTER 5

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 LIST OF MODULES

1. IMPORTING LIBRARIES & DATASET
2. PRE-PROCESSING
3. PERFORMING EDA
4. DATA MODELLING
5. VISUALIZATION WITH POWERBI

5.2 MODULE DESCRIPTION

These module descriptions outline the key tasks and objectives within each stage of the analysis process, providing a comprehensive overview of the data analysis workflow.

5.2.1 IMPORTING LIBRARIES & DATASET

This module involves sourcing the dataset required for analysis. The dataset may be obtained from various sources such as online repositories, databases, or APIs. Once the dataset is downloaded, it needs to be imported into a Python environment for further analysis. This module focuses on using tools like pandas to efficiently load the dataset and prepare it for subsequent processing steps

5.2.2 PRE-PROCESSING

This module addresses the cleaning and preparation of the dataset to ensure it is suitable for analysis. It involves handling missing values, outliers, and transforming data as necessary. The goal is to enhance data quality and eliminate potential biases.

5.2.3 PERFORMING EDA (Exploratory Data Analysis)

EDA is a crucial step in understanding the dataset's characteristics and uncovering patterns or trends. This module utilizes statistical analysis and visualizations to gain insights into the data distribution, correlations, and potential relationships.

5.2.4 DATA MODELLING

In this module, machine learning or statistical models are applied to the preprocessed dataset. The objective is to extract meaningful patterns, make predictions, or perform other analytical tasks based on the data.

- Select appropriate models based on analysis goals.
- Split the dataset into training and testing sets.
- Train and optimize models using the training set.
- Evaluate model performance using the testing set.

5.2.5 VISUALIZATION WITH POWERBI

This module focuses on leveraging PowerBI for creating interactive and dynamic visualizations that communicate the results of the analysis effectively. PowerBI allows for the development of dashboards and reports that facilitate exploration and interpretation of findings.

- Design and build visualizations based on key insights.
- Create interactive dashboards for user-friendly exploration.
- Incorporate relevant filters and slicers for dynamic analysis.

CHAPTER 6

TESTING

CHAPTER 6

TESTING

Testing is the process of executing a program or application with the intent of finding software bugs, and to verify that the software product is fit for use.

6.1 UNIT TESTING

Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately.

This testing is carried out during programming stage itself. It is done by the developer itself in every stage of the project and finding the bug and module predicated additionally done by the developer only here we are going to solve all the runtime errors.

6.1.1 TEST OBJECTIVES

- Ensure that each unit of code performs its intended functionality correctly.
- Ensure that each unit operates independently and does not rely on the state of other units.
- Assess the unit's execution time and resource usage.
- Evaluate how well the unit performs as the input size or load increases.

6.2 INTEGRATION TESTING

Integration testing is a crucial phase in the software development life cycle where individual modules or components are combined and tested as a group to ensure their seamless collaboration. In the context of the app success prediction project, integration testing plays a pivotal role in verifying that the various

modules, such as downloading the dataset, importing data in Python, preprocessing, performing exploratory data analysis (EDA), and data modeling, integrate effectively and produce the expected results when combined. The primary objectives of integration testing in this project are to identify and address any issues related to the interactions between modules, ensure data consistency and flow throughout the pipeline, and validate that the integrated system functions as intended.

6.3 SYSTEM TESTING

System system testing is a critical phase in the project lifecycle, ensuring that all components and modules work harmoniously as an integrated system. For our project focused on app success prediction in the Google Play Store, system testing involves evaluating the entire application's functionality, from data acquisition to visualization. The key objectives are to validate the overall coherence of the system, assess its adherence to requirements, and identify and resolve any potential issues.

System testing for the app success prediction project involves a holistic evaluation of the entire system, covering data flow, modeling accuracy, visualization integration, user acceptance, security, performance, and documentation. The goal is to deliver a reliable and user-friendly application that fulfills its objectives in the competitive landscape of the Google Play Store.

CHAPTER 7

RESULTS AND DISCUSSION

CHAPTER 7

RESULTS AND DISCUSSION

7.1 RESULTS

In the results of the Google Play Store data analytics and app prediction project, the implemented methodology showcased notable outcomes. The analysis of app-related data revealed patterns and trends crucial for understanding the factors influencing app success on the platform. Exploratory Data Analysis (EDA) brought forth insights into the distribution of app ratings, the impact of user reviews on app installs, and the correlation between app size and user preferences. Data modeling, utilizing machine learning algorithms such as Random Forest and Support Vector Machine, yielded predictive models for app success. The models demonstrated accuracy in forecasting app performance based on diverse features, contributing to informed decision-making for developers. The visualization phase, leveraging PowerBI, provided a compelling visual representation of the analysis results, offering an intuitive understanding of the relationships between app characteristics and success metrics. The findings highlighted the importance of factors like user engagement, app category, and pricing strategies in determining app success. These results contribute to a nuanced understanding of the dynamics within the Google Play Store ecosystem, aiding developers and stakeholders in optimizing their app development and marketing strategies.

7.2 DISCUSSION

The Google Play Store data analytics and app prediction project revealed a profound depth of insights into the dynamics of the mobile app market. The comprehensive exploration of factors such as app size, number of installs, reviews, type (free/paid), rating, category, content rating, and price elucidated

intricate patterns that contribute to the success of applications on the platform. The efficacy of the implemented exploratory data analysis (EDA) in uncovering relationships between these variables was evident, shedding light on the nuanced preferences and behaviors of users. The subsequent data modeling phase, employing techniques like Random Forest, Support Vector Machine, and Linear Regression, underscored the predictive capabilities of the models. The visualization component, facilitated by PowerBI, not only enhanced the interpretability of the results but also offered a dynamic interface for users to interact with and comprehend the analysis effortlessly. The amalgamation of these analytical stages resulted in a predictive framework capable of offering valuable insights for both developers and users navigating the vast expanse of the Google Play Store. The project not only contributes to the understanding of app success determinants but also lays a foundation for future endeavors in refining predictions and adapting to the evolving landscape of the mobile app market.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In conclusion, the Google Play Store data analytics and app prediction project culminate in a comprehensive exploration of the intricate dynamics within the mobile app ecosystem. The analysis of diverse attributes, including app size, installs, reviews, type, rating, category, content rating, and price, has illuminated nuanced patterns and trends. The predictive models derived through machine learning algorithms offer valuable insights into the factors influencing app success. The integration of PowerBI for visualization enhances the interpretability of findings, providing a user-friendly interface for stakeholders. The project not only unravels the relationships between various features but also emphasizes the significance of strategic considerations in app development. The predictive capabilities achieved open avenues for developers to make informed decisions, contributing to the ongoing evolution of the mobile app market. As the digital landscape continues to evolve, this project lays a foundation for understanding and navigating the complexities inherent in app success prediction, contributing to the broader discourse on data-driven decision-making in the realm of mobile applications.

8.2 FUTURE ENHANCEMENT

Future enhancements for the Google Play Store data analytics and app prediction project could involve the integration of advanced machine learning models and algorithms to refine prediction accuracy. Exploring ensemble learning techniques, such as stacking or boosting, may enhance the robustness of the predictive models, allowing for a more nuanced understanding of app success factors. Real-time data streaming and analytics could be implemented to ensure

that the prediction models adapt swiftly to evolving trends and user behavior. Enhanced data visualization techniques, including 3D visualizations or immersive analytics, might be considered to provide a more engaging and interactive user experience. Furthermore, the project could benefit from the integration of external data sources, such as social media trends or economic indicators, to enrich the dataset and capture broader market influences. Continuous refinement and optimization of the predictive models based on ongoing feedback and emerging technological advancements will be crucial for keeping the Google Play Store data analytics and app prediction project at the forefront of the dynamic mobile app landscape.

ANNEXURE

ANNEXURE

APPENDIX 1

SOURCE CODE

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import re
from datetime import datetime
from IPython.display import HTML
plt.rcParams['figure.figsize'] = [15, 10]

# Loading the dataset
df=pd.read_csv('/kaggle/input/google-playstore-apps/Google-Playstore.csv')
df.head()
df.columns
df.shape
df.info()
df.describe()

temp_size=[]
for x in df.Size:
    if pd.isna(x):
        temp_size.append(np.nan)
    elif 'G' in x:
```

```

        temp_size.append(float(re.sub('G',"",x))*1000)
elif 'M' in x:
        temp_size.append(float(re.sub('M|',"",x)))
elif 'K' in x or 'k' in x:
t        emp_size.append(float(re.sub('K|k|',"",x))/1000.0)
else:
        temp_size.append(0)
df.Size = temp_size
Released = []
for x in df.Released:
        if pd.isna(x):
                Released.append(np.nan)
        else:
                Released.append(datetime.strptime(x, '%b %d, %Y'))
df.Released = Released
df["Last Updated"]=pd.to_datetime(df["Last Updated"])
df.info()

```

The first 20 Category App

```

pie_data = df.Category.value_counts()
graph_data = pie_data[:19]
graph_data["Others"] = pie_data[20:].sum()
plt.pie(graph_data.values, labels=graph_data.keys(), autopct='%1.2f%%
%')
plt.show()

```

Free vs Paid

```

paid_free_data = {
"Paid": df.Price[df.Price > 0].count(),

```



```

"Free": df.Price[df.Price == 0].count()
}

plt.figure(figsize=(6,6))
plt.pie(paid_free_data.values(), labels=paid_free_data.keys(),
autopct='%1.2f%%')
plt.show()

# Distribution of Ratings for Free and Paid Apps
#Visualize the distribution of ratings for free and paid apps
plt.figure(figsize=(10, 5))
sns.kdeplot(data=df[df['Free'] == True]['Rating'], label='Free Apps',
fill=True)
sns.kdeplot(data=df[df['Free'] == False]['Rating'], label='Paid Apps',
fill=True)
plt.title('Distribution of Ratings for Free and Paid Apps')
plt.xlabel('Rating')
plt.ylabel('Density')
plt.legend()
plt.show()

# Add Supported App
ad_supported=df["Ad Supported"].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ad_supported.values, labels=ad_supported.keys(),
autopct='%1.2f%%')
plt.show()

# Purchase Option in App
app_purchase = df["In App Purchases"].value_counts()

```

```

plt.figure(figsize=(5,5))
plt.pie(app_purchase.values, labels=app_purchase.keys(),
autopct='% 1.2f%% ')
plt.show()

# Top 10 Minimum Android version
android_version = df["Minimum Android"].value_counts()
graph_data = android_version[:9]
graph_data["Others"] = android_version[10:].sum()
plt.figure(figsize=(10,10))
plt.pie(graph_data.values, labels=graph_data.keys(), autopct='% 1.2f%
%')
plt.show()

# Top 10 most expensive App
sorted_df = df.sort_values(by=['Price'], ascending=False)
sorted_df2=sorted_df.set_index('App Name')
sorted_df2[["Category","Rating","Maximum Installs","Price"]][0:10]

# Top 10 Maximum Installs App Table
sorted_Install= df.sort_values(by=['Maximum Installs'],
ascending=False)
HTML(sorted_Install[["App Name","Category","Rating","Price","Maximum
Installs"]][0:10].to_html(index=False))

# The most Rated top 10 Category
a = df.groupby(["Category"])
["Rating"].count().sort_values(ascending=False)[:10]
ax = sns.barplot(x=a.keys(), y=a.values)

```

```

ax.tick_params(axis='x', labelrotation = 45)
ax = plt.ylabel('Ratings (count)')
# Top 10 category with Ad Supported¶
topTenCategoriesNames=df["Category"].value_counts()[0:10].keys()
a = df[df.Category.isin(topTenCategoriesNames)]
ax = sns.catplot(data=a,kind="count", x="Category", hue="Ad
Supported",height=8.27, aspect=11.7/8.27)
ax.set_xticklabels(rotation= 45)

```

```

# Top 10 Developer
topTenDeveloperID= df["Developer Id"].value_counts()[0:10]
ax = sns.barplot(x=topTenDeveloperID.keys() ,
y=topTenDeveloperID.values)
ax.tick_params(axis='x', labelrotation = 45)
sns.scatterplot( data=df, x="Price", y="Rating", size=df["Size"],)

```

```

# Abondened & Non-Abondened App before 2019
app = {
"Abondened":(df[df["Last Updated"]<"2019"]).shape[0] ,
"Non-Abondened":(df[df["Last Updated"]>="2019"]).shape[0]
}
plt.figure(figsize=(6,6))
ax = plt.pie(app.values(), labels=app.keys(), autopct='%1.2f%%')

```

```

# Free & Paid in Abondened App
abondened = df[df["Last Updated"]<"2019"]
abondened_paid_free = {
"Paid": abondened.Price[abondened.Price > 0].count(),
"Free": abondened.Price[abondened.Price == 0].count()
}

```

```

}
plt.figure(figsize=(6,6))
ax = plt.pie(abondened_paid_free.values(),
labels=abondened_paid_free.keys(), autopct='% 1.2f%% ')

# Ad Supported in Abondened App
abondened = df[df["Last Updated"]<"2019"]
abondened_ad_supported = {
"Yes": abondened["Ad Supported"][abondened["Ad Supported"] ==
True].count(),
"No" : abondened["Ad Supported"][abondened["Ad Supported"] ==
False].count()
}
plt.figure(figsize=(6,6))
ax = plt.pie(abondened_ad_supported.values(),
labels=abondened_ad_supported.keys(), autopct='% 1.2f%% ')

# Number of Apps per Content Rating
#Visualize the distribution of content ratings
plt.figure(figsize=(8, 4))
sns.countplot(x='Content Rating', data=df, order=df['Content
Rating'].value_counts().index)
plt.title('Number of Apps per Content Rating')
plt.xlabel('Content Rating')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Data Modelling

```

LinearRegression

```
data=df.copy()
```

```
# Clean up the 'Installs' column
```

```
data['Installs_clean'] = data['Installs'].str.replace('[^0-9]', '',  
regex=True).fillna(0).astype(int)
```

```
# Import necessary libraries
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error, r2_score
```

```
from sklearn.pipeline import Pipeline
```

```
from sklearn.compose import ColumnTransformer
```

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
# Select features and preprocess the dataset
```

```
selected_features = ['Category', 'Size', 'Installs_clean', 'Free',  
'Price', 'Content Rating']
```

```
data_sample = data.sample(frac=0.1, random_state=42)
```

```
data_filtered = data_sample.dropna(subset=selected_features +  
['Rating']).copy()
```

```
# One-hot encode categorical features
```

```
categorical_features = ['Category', 'Content Rating']
```

```
numerical_features = ['Size', 'Installs_clean', 'Price']
```

```
preprocessor = ColumnTransformer(  
transformers=[
```

```
( 'num', StandardScaler(), numerical_features),  
( 'cat', OneHotEncoder(), categorical_features)
```

```
])
```

```
from sklearn.linear_model import LinearRegression
```

```
# Create a pipeline with preprocessing and the Linear Regression model
```

```
model_lr = Pipeline([
```

```
( 'preprocessor', preprocessor),
```

```

('linear_regression', LinearRegression())
)
# Split the data into training and testing sets
X = data_filtered[selected_features]
y = data_filtered['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Train the Linear Regression model
model_lr.fit(X_train, y_train)
# Make predictions and evaluate the model
y_pred_lr = model_lr.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
print(f"Linear Regression - MSE: {mse_lr:.2f}, R2: {r2_lr:.2f}")

```

```

# DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor
data_features_encoded =
pd.get_dummies(data_filtered[selected_features],
columns=categorical_features)
# Split the dataset into training and testing sets
X = data_features_encoded
y = data_filtered['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.1, random_state=42)
# Create and train the RandomForest model
model_rf = DecisionTreeRegressor(random_state=42)
model_rf.fit(X_train, y_train)
# Make predictions and evaluate the model

```

```

y_pred_rf = model_rf.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f"DecisionTreeRegressor - MSE: {mse_rf:.2f}, R2: {r2_rf:.2f}")

```

Gradient Boosting Regressor

```

from sklearn.ensemble import GradientBoostingRegressor
model_gb = GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, max_depth=3, random_state=42)
model_gb.fit(X_train, y_train)
# Make predictions and evaluate the model
y_pred_gb = model_gb.predict(X_test)
mse_gb = mean_squared_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)
print(f"Gradient Boosting - MSE: {mse_gb:.2f}, R2: {r2_gb:.2f}")

```

K-Nearest Neighbors

```

from sklearn.neighbors import KNeighborsRegressor
preprocessor = ColumnTransformer(
transformers=[
('num', StandardScaler(), numerical_features),
('cat', OneHotEncoder(), categorical_features)
])
# Split the dataset
X = data_filtered[selected_features]
y = data_filtered['Rating']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Create a pipeline with preprocessing and the KNeighborsRegressor

```

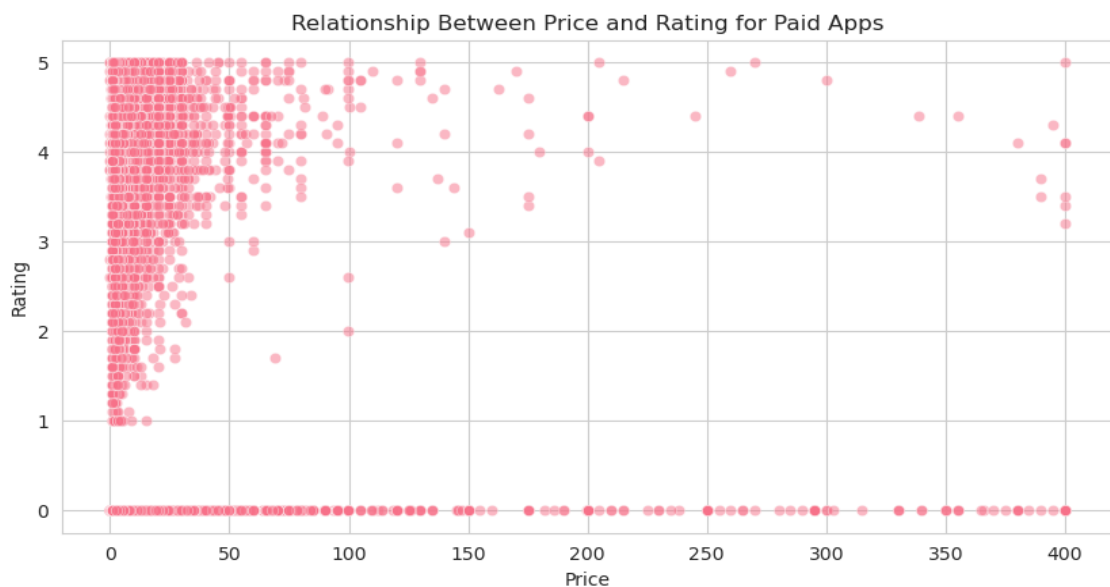
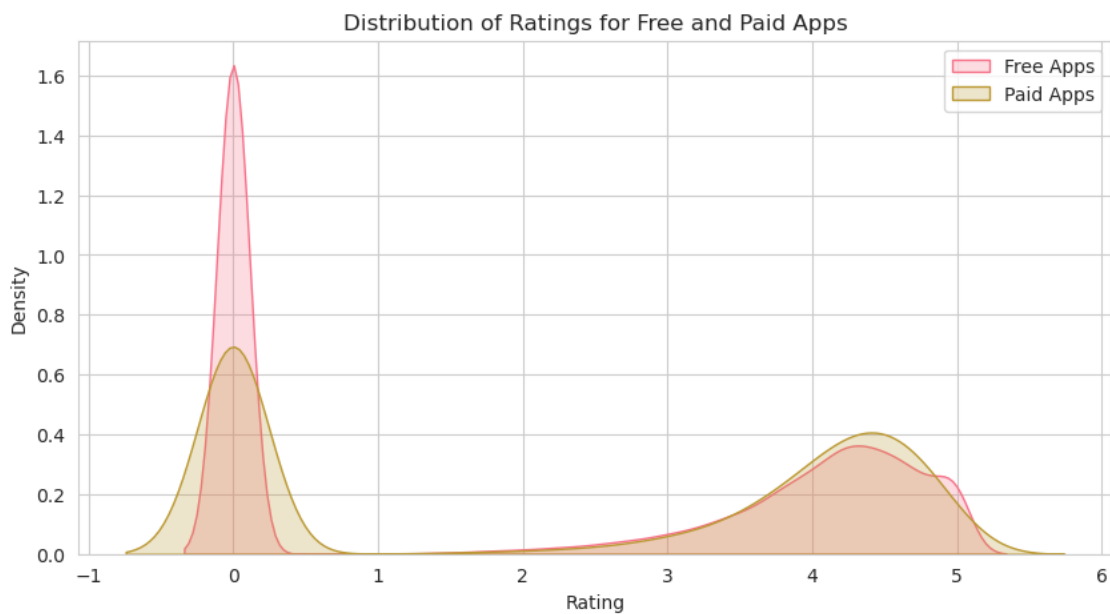
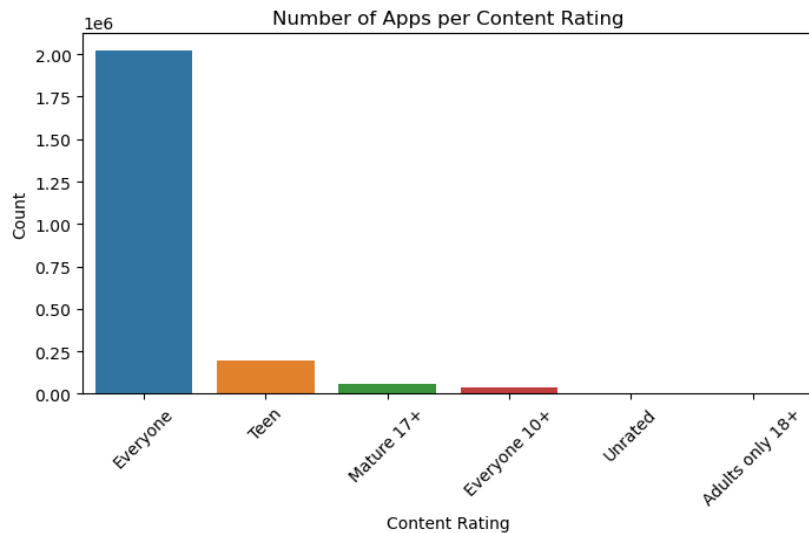
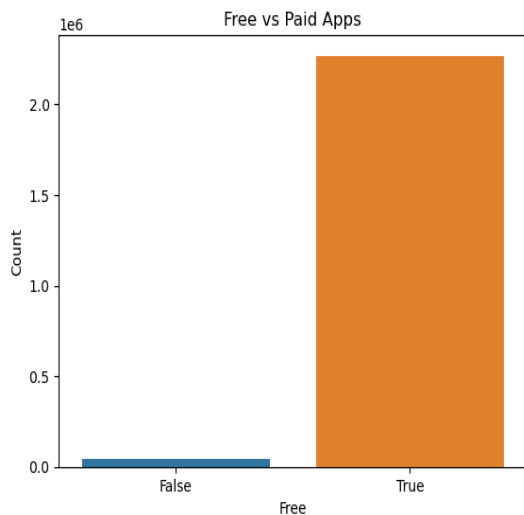
```

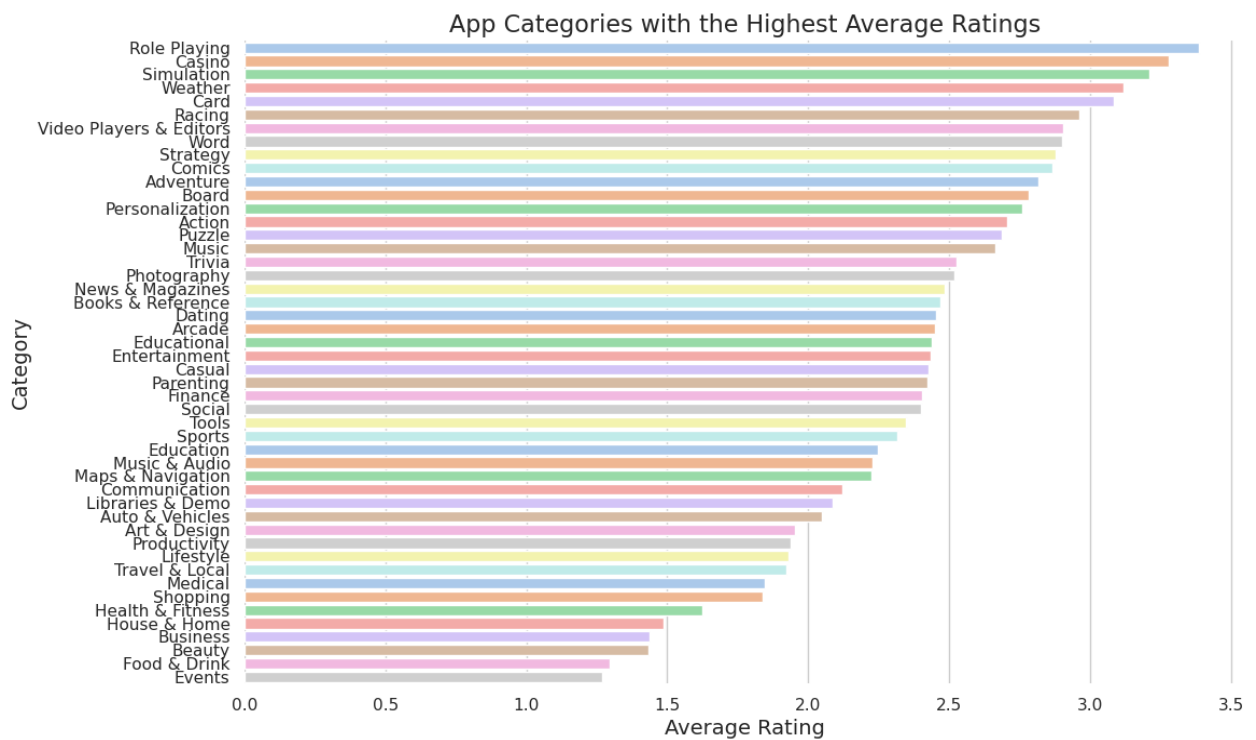
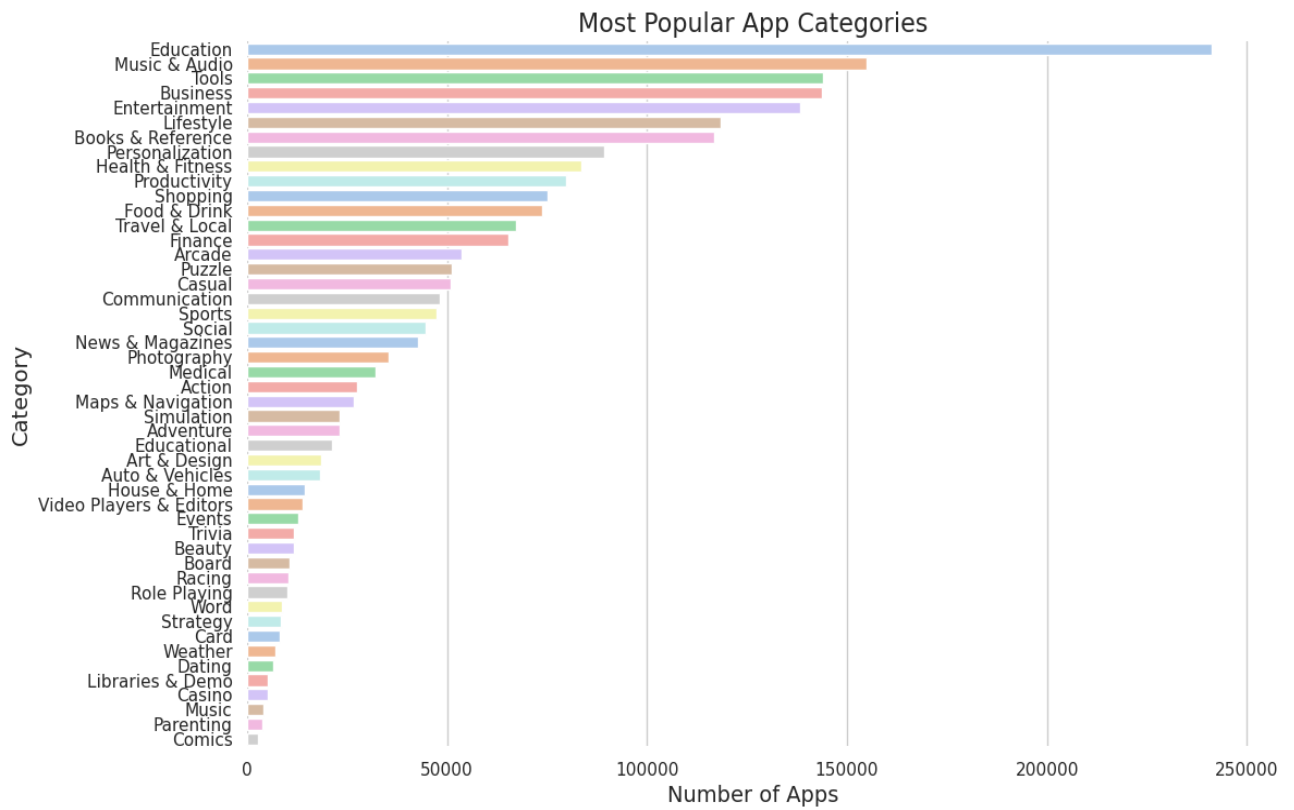
model
model_knn = Pipeline([
('preprocessor', preprocessor),
('kneighbors', KNeighborsRegressor(n_neighbors=5))
])
# Train the KNeighborsRegressor model
model_knn.fit(X_train, y_train)
# Make predictions and evaluate the model
y_pred_knn = model_knn.predict(X_test)
mse_knn = mean_squared_error(y_test, y_pred_knn)
r2_knn = r2_score(y_test, y_pred_knn)
print(f'K-Nearest Neighbors - MSE: {mse_knn:.2f}, R2: {r2_knn:.2f}')

# RandomForest model

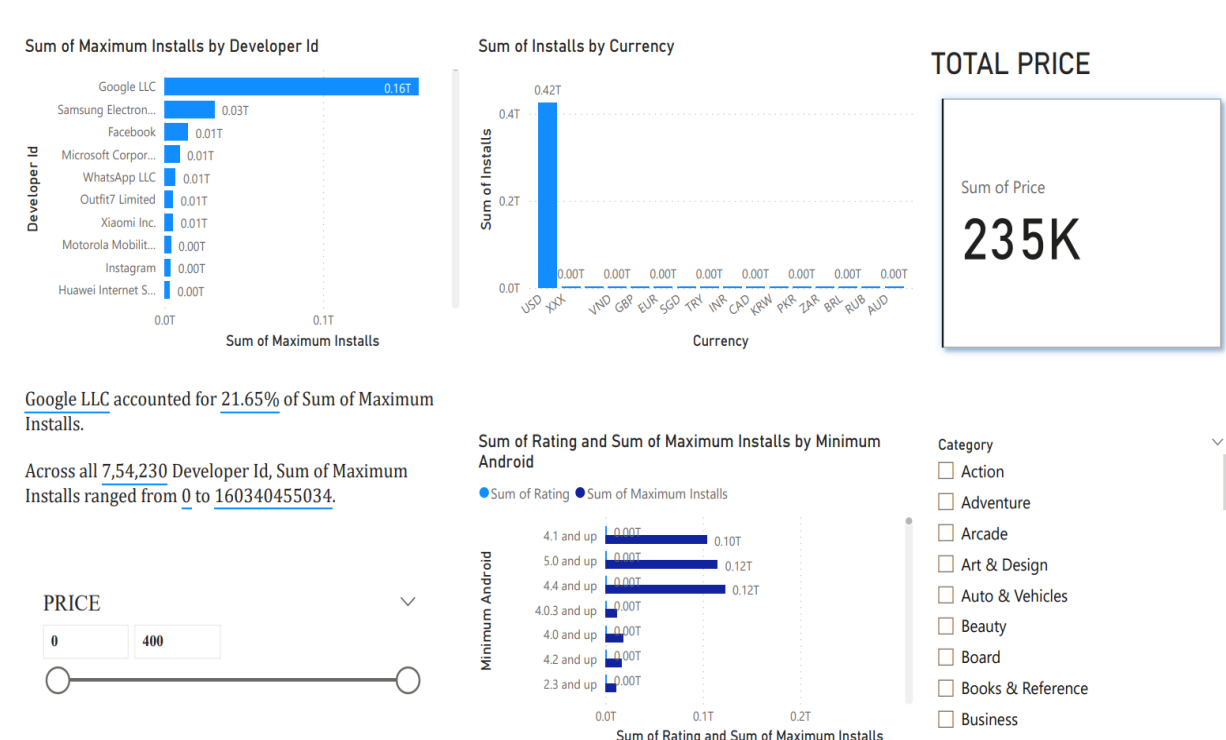
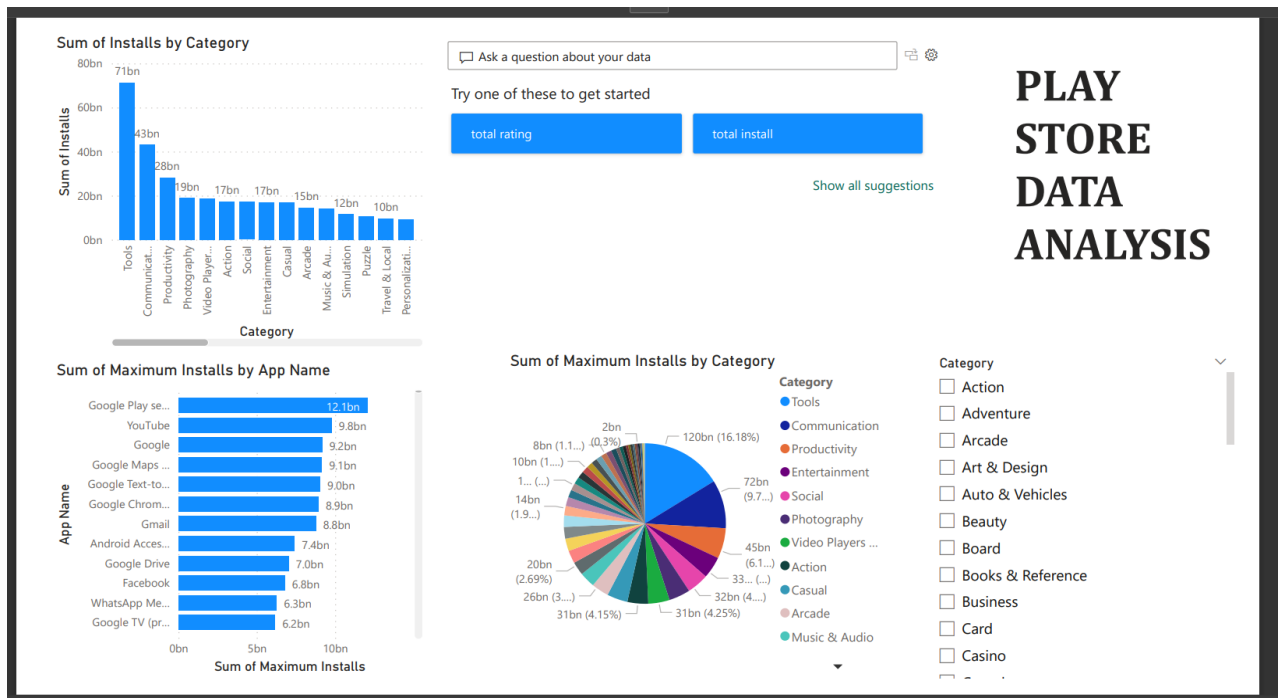
from sklearn.ensemble import RandomForestRegressor
model_rf = RandomForestRegressor(n_estimators=10, random_state=42)
model_rf.fit(X_train, y_train)
# Make predictions and evaluate the model
y_pred_rf = model_rf.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f"Random Forest - MSE: {mse_rf:.2f}, R2: {r2_rf:.2f}")

```



Linear Regression - MSE: 4.26, R2: 0.04
 Gradient Boosting - MSE: 2.27, R2: 0.49
 DecisionTree Regressor - MSE: 0.41, R2: 0.91
 K-Nearest Neighbors - MSE: 3.07, R2: 0.31
 Random Forest - MSE: 0.26, R2: 0.94



REFERENCES

- [1] O. Lengkong and R. Maringka, "Apps Rating Classification on Play Store Using Gradient Boost Algorithm," 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), Manado, Indonesia, 2020, pp. 1-5, doi: 10.1109/ICORIS50180.2020.9320756..

- [2] I. Gunaratnam and D. N. Wickramarachchi, "Computational Model for Rating Mobile Applications based on Feature Extraction," 2020 2nd International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka, 2020, pp. 180-185, doi: 10.1109/ICAC51239.2020.9357270.

- [3] E. Noei, F. Zhang and Y. Zou, "Too Many User-Reviews! What Should App Developers Look at First?," in IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 367-378, 1 Feb. 2021, doi: 10.1109/TSE.2019.2893171.

- [4] I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst and A. E. Hassan, "Examining the Rating System Used in Mobile-App Stores," in IEEE Software, vol. 33, no. 6, pp. 86-92, Nov.-Dec. 2016, doi: 10.1109/MS.2015.56.

- [5] Harman, M., Jia, Y., & Zhang, Y. (2012). App store mining and analysis: MSR for app stores. IEEE International Working Conference on Mining Software Repositories. <https://doi.org/10.1109/MSR.2012.6224306>