FINAL PROJECT REPORT

GOOGLE PLAY STORE APPS: ANALYSIS & RATING PREDICTION

1. OBJECTIVE:

The goal of this project is to conduct a comprehensive analysis of Google Play Store apps and develop a predictive model for rating prediction. The project aims to uncover valuable insights into the factors influencing app ratings and provide a reliable model that can predict the ratings of new or existing apps based on various features. Develops a predictive model that considers both app-specific features from the Google Play Store dataset and broader market indicators from the Google Stock Market dataset to forecast the app ratings accurately.

2. ABSTRACT:

In the rapidly evolving landscape of mobile applications, understanding the factors that contribute to the success of an app is crucial for developers and stakeholders. This project aims to delve into the Google Play Store ecosystem, analyzing a vast dataset of over 2.3 million applications. The primary focus is on uncovering patterns, trends, and correlations within the dataset to gain valuable insights into the factors influencing app ratings. Additionally, the development of a robust predictive model aims to assist app developers in anticipating and improving the ratings of their applications. The project is motivated by the need for actionable insights in the highly competitive app market, where user satisfaction and app ratings play a pivotal role in an app's success. Ultimately, the findings and predictive model derived from this analysis will empower developers and stakeholders with valuable information for making informed decisions to enhance the overall quality and performance of Google Play Store applications.

The success of this project will be evaluated based on achieving the criterias like effective exploratory data analysis, development of a robust predictive model, feature analysis, visualizations, summaries, and interpretations should be accessible to both technical and non-technical stakeholders, providing actionable insights for decision-makingThe exploration and analysis of the Google Play Store Apps dataset, with a primary focus on understanding the factors influencing app ratings and developing a predictive model.

The project is constrained by the availability and quality of the Google Play Store Apps dataset. Inaccuracies, missing data, or inconsistencies within the dataset may impact the accuracy of analyses and model predictions. limited processing power and memory, may influence the scalability of analyses and the training of complex machine learning models. Optimal resource utilization will be a consideration. The project is subject to time limitations, and the depth of analysis and model complexity must align with the available time frame. Balancing thorough exploration with timely delivery is crucial. In an era dominated by mobile applications, understanding the factors that influence app ratings is crucial for developers and stakeholders alike. This project delves into the realm of Google Play Store apps, aiming to conduct an in-depth analysis and develop a predictive model for app rating prediction. Leveraging a diverse dataset encompassing app-specific features from the Google Play Store and broader market indicators from the Google Stock Market

Through exploratory data analysis, feature engineering, and advanced modeling techniques, we seek to uncover the intricate relationships between app attributes, market dynamics, and user ratings. By identifying key determinants of app success and failure, we aim to provide valuable guidance for

developers looking to optimize their app performance and user satisfaction. Our predictive model, equipped with machine learning algorithms such as Linear Regression, Random Forest, Gradient boosting, SVR, MLP and XGBRegressor promises to offer reliable predictions for app ratings based on the input variables. By blending domain-specific knowledge with cutting-edge analytical tools, this project endeavors to empower app developers and stakeholders with actionable insights for enhancing app quality, driving user engagement, and ultimately, achieving success in the competitive landscape of the Google Play Store.

3.DESIGN & IMPLEMENTATION:

3.1.Dataset Overview:

For this project, two primary datasets were utilized, procured from Kaggle

Google Play Store Apps Dataset: This dataset provides comprehensive information about various mobile applications available on the Google Play Store. It includes attributes such as app name, category, ratings, rating count, number of installs, price, and more.

Google Stock Market Dataset: This dataset offers market indicators and stock-related information from the Google stock market. It encompasses attributes such as stock prices, trading volumes, and other market metrics.

3.2.Data Wrangling:

Merging Datasets:

The first step in our project is to merge the two datasets effectively. We have two primary datasets: one containing information about app characteristics from the Google Play Store and the other containing Google's stock market performance data. Merging these datasets will create a unified dataset that combines both sets of information, allowing us to analyze the relationship between app ratings and stock market indicators.

Data Cleaning and Explore Correlations:

- **Handling Missing Values:** Checked for missing values in both datasets and decided on appropriate strategies for dealing with them, such as filling null values or removal.
- **Handling Duplicates:** Identified and removed duplicate rows or entries in the datasets to ensure data integrity.
- **Data Type Conversion:** Ensured that the data types of variables are appropriate for analysis and modeling. Converted categorical variables to the categorical data type, changed date into proper format and other numerical values to int or float.

Exploring Correlations:

- **Correlation Analysis:** Computed correlation coefficients between relevant variables in the merged dataset to understand the strength and direction of their relationships.
- **Visualization:** Created visualizations such as scatter plots, heatmaps, or correlation matrices to

visualize the relationships between app ratings and stock market indicators.

- Before proceeding with the analysis, we performed data cleaning and preparation tasks to ensure that the datasets were ready for exploration. This involved handling missing values, removing duplicates, converting data types as needed, and addressing any inconsistencies in the data.

3.3. Feature Engineering:

Feature engineering is a crucial step in preparing the data for modeling. We created new features and modified existing ones to improve the performance of our predictive models. This may involve transforming variables, creating interaction terms, and adding new features like, Time Gap Between App Release and Stock Market Data, Interaction Features, Duration Since Last Update, App Name Length & Install range, Ratio between Rating and average stock price.

1) Time Gap Between App Release and Stock Market Data:

Calculated the time difference (in days or months) between the release date of each app and the corresponding date in the stock market dataset. This feature captures the temporal relationship between app launches and stock market data availability.

Purpose: Assess whether there is any discernible pattern or correlation between the timing of app releases and fluctuations in the stock market, which could potentially impact app ratings. We found that based on our current dataset, there are no instances where the app release date and the stock market date differ. In other words, for all the rows in our dataFrame, the app was released on the same day as the corresponding stock market date. Since, it does not have any meaningful insights to draw in apps rating prediction, removed this feature.

2) Interaction Features:

Created new features by combining existing features through mathematical operations

Fig.3.3.1

such as multiplication, division, or addition. For example, interaction features can be derived by multiplying the rating count with the interaction rating volume.

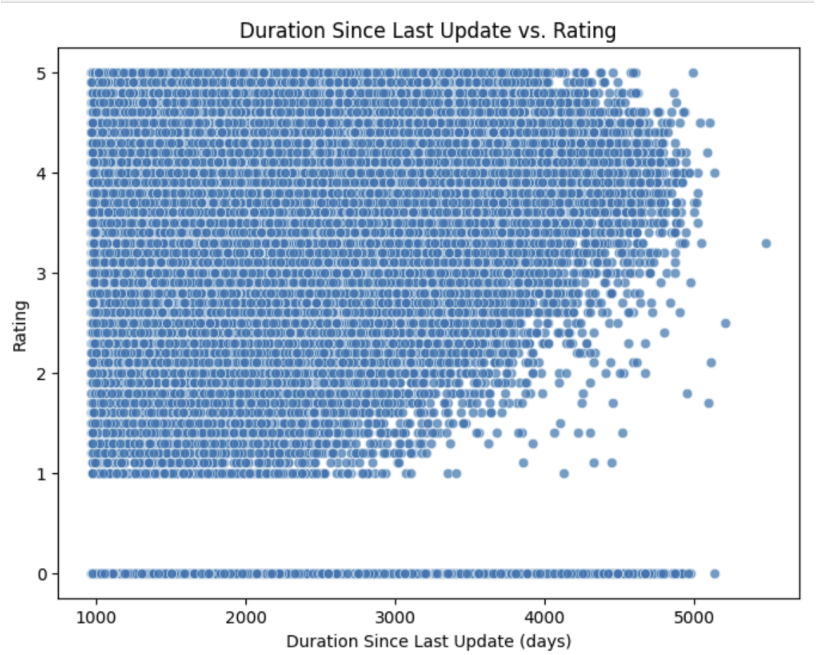
Purpose: Capture synergistic effects or interactions between different app characteristics and stock market indicators, which may have a collective influence on app ratings. Creating interaction features involves combining information from different columns to capture potential dependencies. The resulting 'Interaction_Rating_Volume' feature reflects the interaction or joint influence of the number of ratings and the trading volume. This interaction feature might be useful if we suspect that the impact of ratings on the app's performance in the stock market is related to the trading volume. The pattern in the scatter plot suggests a potential relationship between the interaction feature and the app ratings (Fig.3.3.1). Positive trends indicate that higher values of the interaction feature are associated with higher ratings, and vice versa.

3. Duration Since Last Update:

Calculated the duration (in days or months) since the last update of each app, based on the 'Last Updated' column in the Google Play Store dataset. This feature represents the recency of app updates.

Purpose: Investigate whether more recently updated apps tend to have higher ratings, as

Fig.3.3.2



frequent updates may indicate active maintenance and responsiveness to user feedback. The scatter plot visualizes the relationship between the duration since the last update and the app ratings. (Fig.3.3.2) Each point on the plot represents an app, where the x-axis shows the number

of days since the last update, and the y-axis represents the app's rating. Pattern in the plot can provide insights into whether there is any correlation between the recency of updates and the app ratings. Here we can say that if the updates are more frequent or if the duration is less then there is a chance for higher rating.

4. App Name Length:

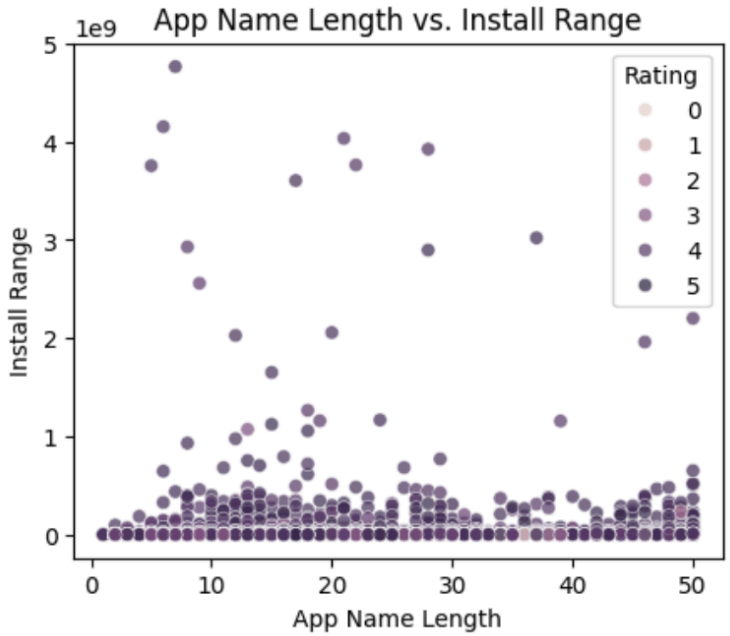
Determined the length of each app's name (e.g., number of characters or words) as a feature. This feature quantifies the complexity or informativeness of app names.

Purpose: Explore whether the length of app names correlates with app ratings, as shorter or more concise names may be more memorable or appealing to users. The length of the app name might influence user perception and ratings. Also check how the install range varies with the length of app names. Checked whether rating have any relation with app name length, but no significant change observed.

5. Install Range:

Calculated the range (difference between maximum and minimum values) of the 'Installs' feature for each category of apps. This feature captures the variability in installation counts within each category.

Fig.3.3.3



Purpose: Assess whether apps with wider install ranges exhibit different rating patterns compared to those with narrower install ranges, indicating potential user preferences or market dynamics. Install range feature represents the range of installs, providing additional information about the popularity of the app. Install range is almost same or a lighter effect for most of the apps with different app name lengths and a very few of them are with high install range and rating with short app name length (Fig.3.3.3). From this we can say that Install range and rating is higher for apps with short app name length. This would help us in predicting rating.

6. Ratio between Rating and Average Stock Price:

Computed the ratio of each app's rating to the average stock price of Google (or other relevant stock) on the date of app release or last update.

Purpose: Investigate whether there is any relationship between app ratings and the financial performance of the parent company, as reflected by stock prices. A higher ratio may indicate

Fig.3.3.4

better alignment between app quality and company performance. The 'Rating_to_Stock_Price_Ratio' reflects the relationship between the app's rating and the avg stock price. This ratio is calculated by dividing the app's rating by the avg stock price. It represents the relationship between the app rating and avg stock price. It indicates how the quality or satisfaction level of an app, as reflected in its rating, relates to its stock market performance. The ratio provides unique insights into how app popularity and user perception influence stock market dynamics. Calculate the average stock price for each app and then create a feature representing the ratio of app ratings to the average stock price.

From Fig.3.3.4, A higher ratio may suggest that the app is relatively more popular compared to its stock market valuation, while a lower ratio may suggest the opposite. Changes in the 'Rating_to_Stock_Price_Ratio' over time can help identify trends in how app popularity correlates with stock market performance. Positive trends indicate that highly rated and popular apps are experiencing positive trends in stock prices, and vice versa. We can say that the 'Rating_to_Stock_Price_Ratio' can serve as a relevant feature that combines information from both domains. The plot shows a positive correlation between the ratio and app rating, indicating that as the ratio increases (indicating higher app ratings relative to stock price), the app rating tends to increase as well.

Each box plot represents the distribution of the ratio within a specific category, allowing us to compare how the ratio varies across different categories in Fig.3.3.5. From the boxplot we can analyze that the 'News & Magazines' category exhibits a wider range of the rating-to-stock price ratio compared to other categories, and that categories like 'Weather' and 'Casino' have relatively higher ratios. The wider range of the rating-to-stock price ratio in the 'News & Magazines' category indicates greater variability in how app ratings relate to stock prices within this category. This variability could be due to several factors such as the diversity of apps within the category, varying levels of user engagement, different monetization strategies, or fluctuations in market demand for news and magazine apps. The higher ratio observed in the 'Weather' category suggests that apps in this category may have relatively higher app ratings compared to their average stock prices. This could be because weather apps often provide essential or highly valued services to users, leading to higher user satisfaction and ratings despite potentially lower stock prices. Weather related events or phenomena may influence user engagement with these apps, contributing to their perceived value and ratings. The higher ratio observed in the 'Casino' category indicates that apps in this category may have higher app ratings relative to their average stock prices. Casino apps may attract a dedicated user base and generate significant

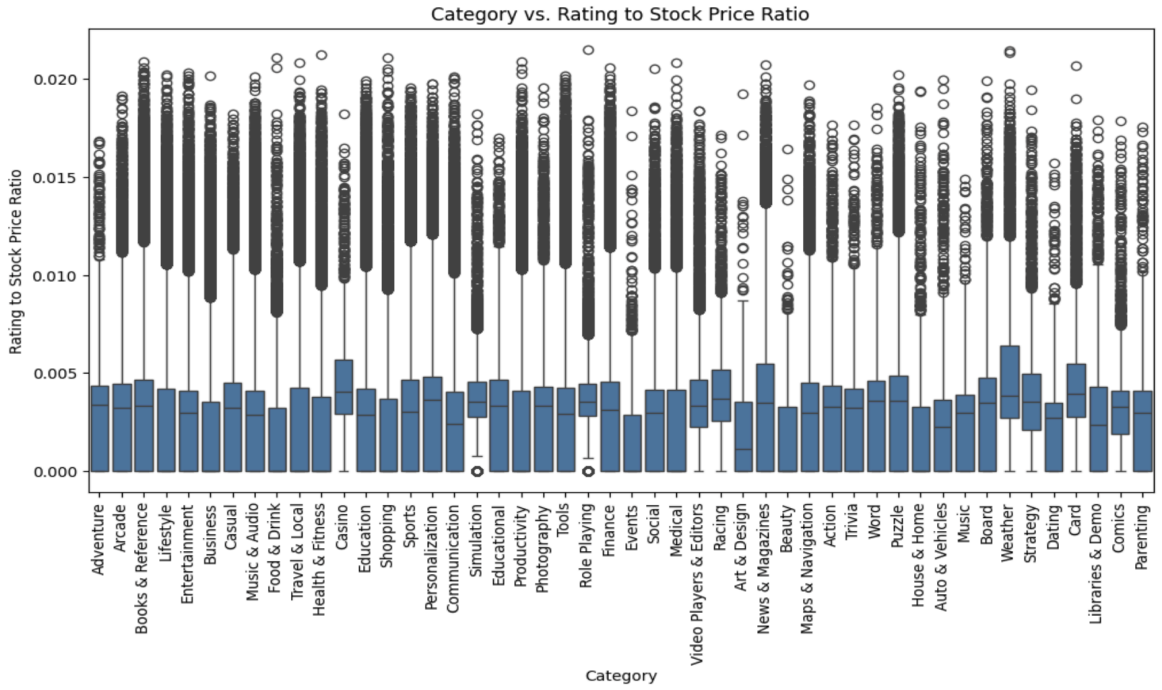


Fig.3.3.5

revenue through in-app purchases or subscriptions, contributing to their higher ratings despite potentially lower stock prices. Additionally, the entertainment value and social aspects of casino games may lead to higher user engagement and satisfaction, driving up app ratings.

These observations provide insights into how different app categories may vary in terms of the relationship between app ratings and stock prices. Factors such as app functionality, user engagement, market demand, and monetization strategies can influence this relationship and contribute to the observed patterns in the box plot analysis.

3.4. Exploratory Data Analysis:

During the exploratory data analysis phase, we delved deep into the Google Play Store apps dataset and the Google Stock Market dataset to gain valuable insights into the characteristics of the data. The primary objective of this analysis was to uncover patterns, trends, and relationships within the data that could inform our app rating prediction model.We examined the distributions of key variables, and visualized the relationships between variables using statistical plots and charts. This process will help us gain insights into the data and understand its underlying structure. We plotted scatter plots, histograms, box plots, and correlation matrices to uncover patterns and trends within the data. These visualizations helped us identify potential correlations between app characteristics and app ratings.

One of the key visualizations used during the EDA process was the heat map, which provided insights into the correlation structure of the dataset. The heat map displayed the correlation coefficients between pairs of variables, with values ranging from -1 to 1. A correlation coefficient close to 1 indicated a strong positive correlation, while a coefficient close to -1 indicated a strong negative correlation. A coefficient close to 0 suggested little to no correlation. Upon examining the heat map, (Fig.3.4.1) we observed that certain features exhibited positive correlations with each other. For instance, the "Rating_to_Stock_Price_Ratio" feature showed a positive correlation with "Rating Count" and "Interaction Rating Volume." This suggested that as the rating-to-stock-price ratio increased, the number of rating counts and interaction rating volume also tended to increase. These insights provided valuable information about the relationships between different variables and their potential impact on app ratings.

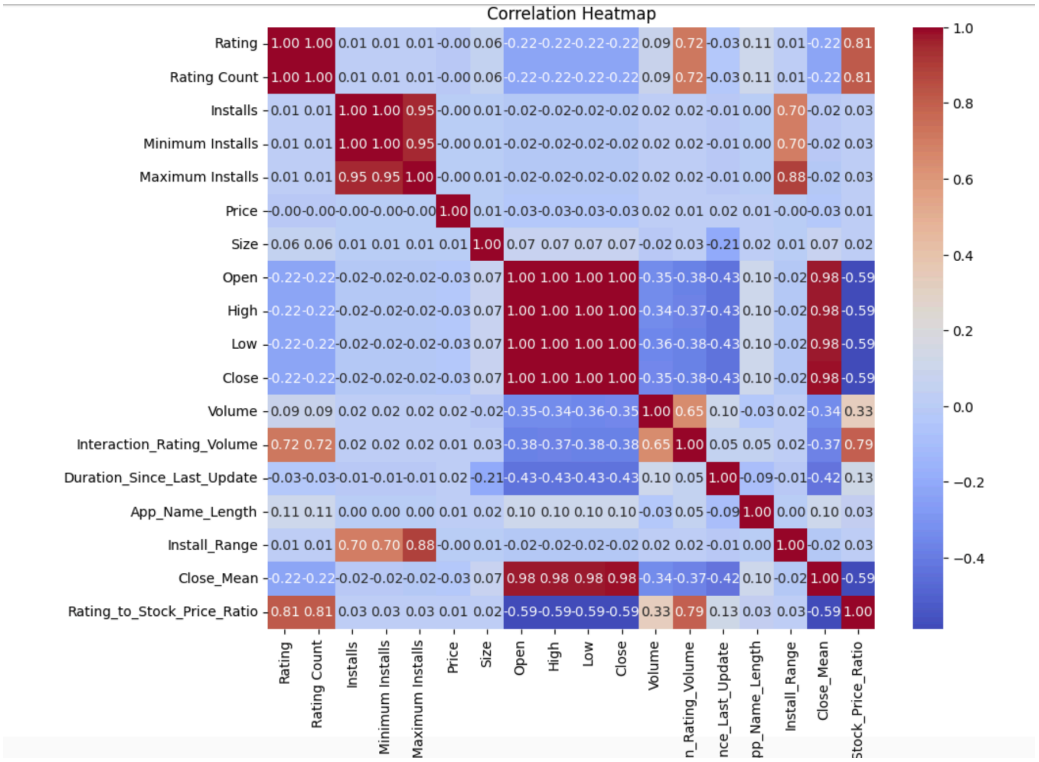


Fig.3.4.1

Plotted Rating Count vs. Rating_to_stock_price_ratio, (Fig.3.4.2) higher values of 'Rating Count' correspond to higher values of 'Rating_to_stock_price_ratio'. Each point on the scatter plot represents an app in the dataset. This figure suggests a positive correlation between the two variables. A higher number of ratings (more popular apps) may be associated with a higher ratio of app ratings to the average stock price, indicating potentially greater user satisfaction or perceived value relative to the stock price.

Fig.3.4.2

We found that, higher values of 'Interaction_Rating_Volume' correspond to higher values of

'Rating_to_Stock_Price_Ratio', suggests a positive correlation between the two variables, when plotted Interaction_Rating_Volume vs. Rating_to_stock_price_ratio (Fig.3.4.3). Apps with higher interaction volumes (e.g., more downloads, more active users) may have a higher ratio of app ratings to the

average stock price, indicating a stronger relationship between user engagement and perceived value relative to the stock price.

Fig.3.4.3

Created pair plots to visualize relationships between Rating, Rating Count, Close, Rating_to_Stock_Price_Ratio.

- Rating vs. Rating Count: This subplot shows the relationship between the actual app ratings and the number of ratings. A positive correlation between these two variables indicates that apps with higher ratings tend to have more ratings.

- Rating vs. Rating_to_Stock_Price_Ratio: This subplot compares the actual app ratings with the calculated ratio of app ratings to the average stock price. They have a positive correlation.

- This pair plot reveals an inversely proportional relationship between 'Close' (stock price) and 'Rating_to_Stock_Price_Ratio', it suggests that as the stock price increases, the ratio of app ratings to the average stock price decreases, and vice versa. This finding could imply a potential relationship between app popularity (as indicated by ratings) and the stock price, where higher stock prices may not necessarily correlate with higher user satisfaction or perceived value relative to the stock price.

- Rating vs Stock Price(Close) : No significant pattern or relation observed. App ratings and stock prices may be influenced by different sets of factors that do not directly correlate with each other.

Figure (Fig.3.4.4) shows that teens contribute to content rating more than that of adults. Apps rated for teens may often include high-quality content, such as educational resources, creative tools, or entertainment options, which receive positive feedback from users. The perceived value and enjoyment derived from these apps contribute to higher ratings.

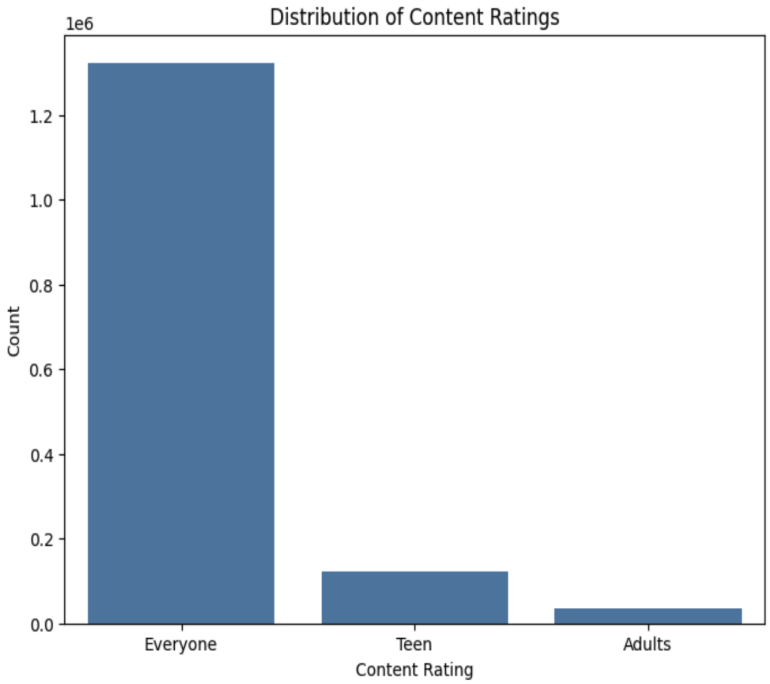


Fig.3.4.4

Teenagers may be more active and engaged users of mobile apps compared to adults, leading to increased usage, interactions, and ratings. Higher user engagement levels can positively influence app ratings. The line plot visualizes the historical trend of stock prices over time. Each point on the plot represents the closing price of the stock (Fig.3.4.5). In this plot, the line generally slopes upwards, indicating that the stock prices tend to increase over the observed period. This upward trend suggests positive growth or appreciation in the value of the stocks.

Fig.3.4.5

Time series Analysis of app ratings were done. Checked how App Ratings changes over time (Fig.3.4.6). In this plot, the line generally slopes downwards, indicating that the app ratings tend to decrease over the observed period. This downward trend suggests a decline in the overall user satisfaction or perceived quality of the apps released over time. We can see fluctuations or variability in app ratings over time, these represented by the movement of the line around the trend, reflect changes in user feedback, app updates, or other factors influencing app ratings. The decreasing trend in app ratings suggests that the apps released over time have received lower ratings from users compared to earlier releases. This may be due to various factors such as declining app quality, increased competition, changes in user preferences, or shifts in market dynamics.

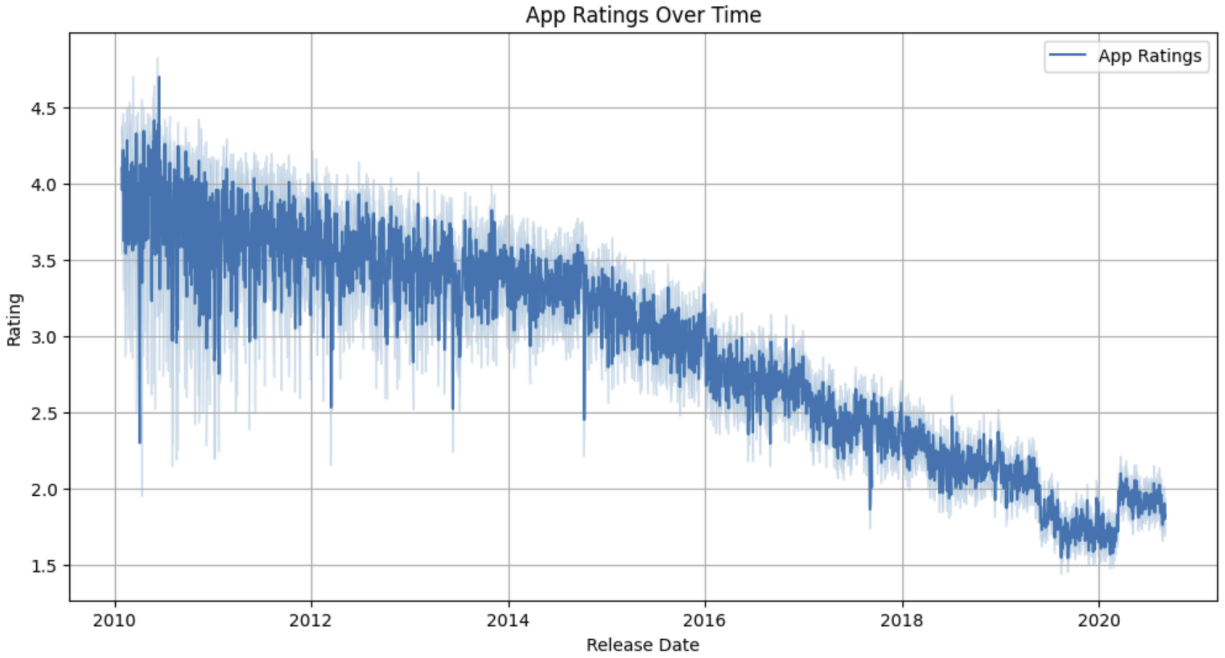


Fig.3.4.6

The decreasing trend in app ratings and the increasing trend in stock prices over time may not necessarily be directly related. App ratings and stock prices are influenced by different factors and represent different aspects of performance and value. However, there could be indirect relationships or common underlying factors that affect both.

Category of app with the highest rating was plotted using a bar chart (Fig.3.4.7). The plot shows the highest-rated app in each category, with the bars representing the rating of each app. Each category is represented by a different color (blue and pink). This visualization helps in understanding which app in each category has the highest rating. Also, plotted Price, Size, and Rating to see their relationship, (Fig.3.4.8) from this plot we can say that, when size increases the price and rating decreases. As the size of an app increases, there may be a tendency for developers to offer it at a lower price or even for free. This could be due to various factors, such as increased competition in the market, users' reluctance to pay higher prices for larger apps, or developers' strategies to attract more downloads by pricing larger apps lower. Larger apps may suffer from performance issues, longer loading times, or higher resource consumption, which can negatively impact user experience and satisfaction, leading to lower ratings. Also, Rating is higher for lower priced apps comparatively. It seems like, if the app size increases the rating is decreasing. Larger app sizes might indicate that the app requires more storage

space on the device. Users may perceive larger apps resource-intensive, potentially leading to lower ratings due to concerns about performance, storage usage, or overall user experience.

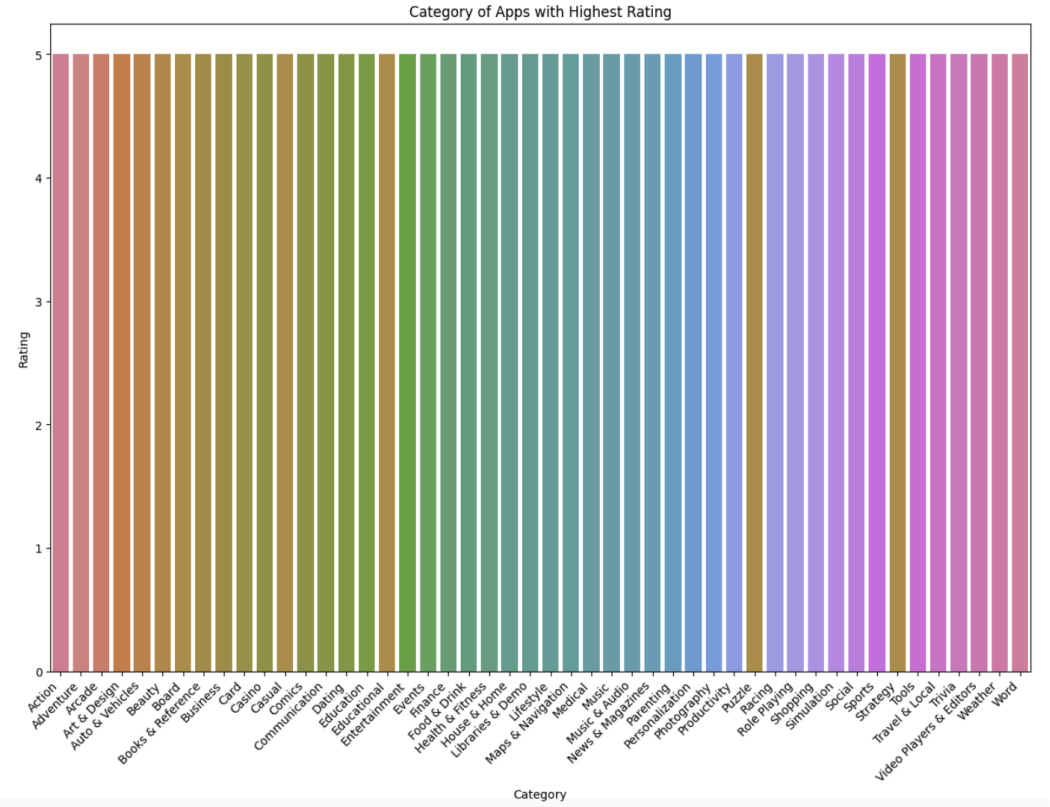


Fig.3.4.7

Users may be less satisfied with apps that exhibit such performance issues, resulting in lower ratings. They may prioritize smaller apps that take up less space on their device, leading to a preference for smaller apps and potentially influencing ratings for larger apps negatively.

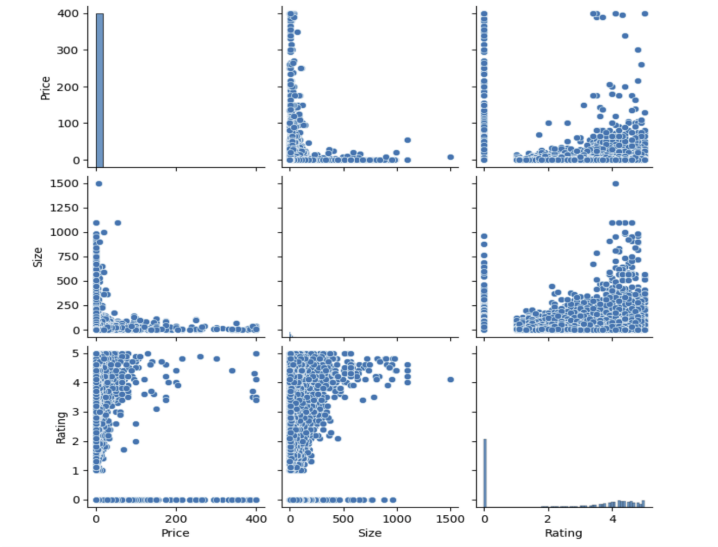


Fig.3.4.8

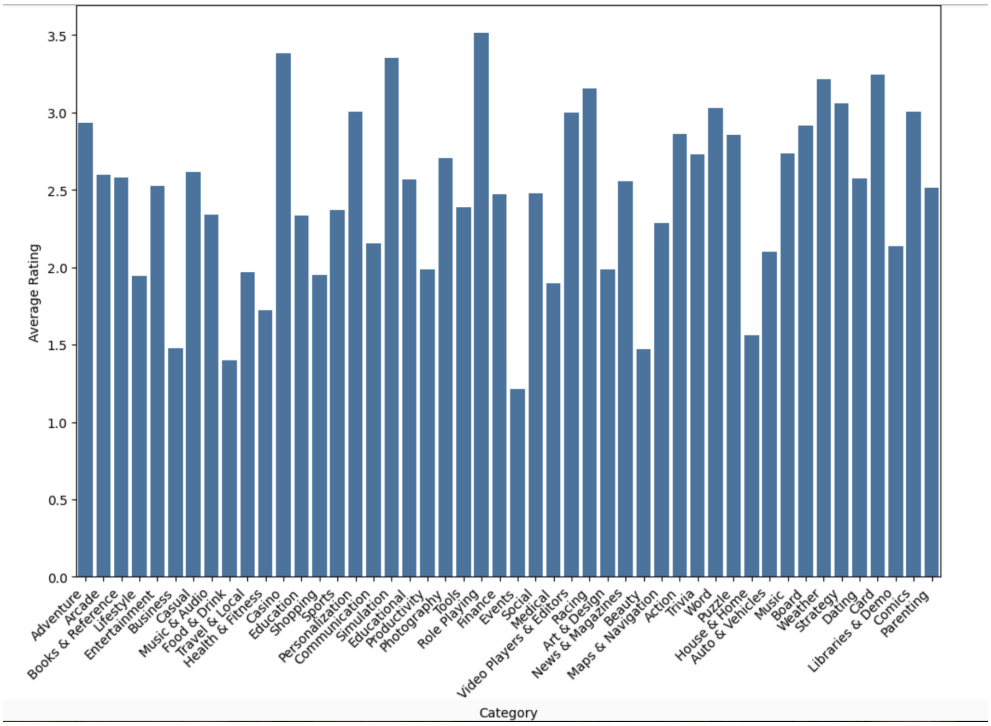


Fig.3.4.9

The plot for rating by Ad Supported and App Type illustrates that Ad-supported free apps tend to have higher ratings compared to other categories. This may indicate that users are more inclined to rate apps positively if they are provided for free and supported by ads. On the other hand, apps that are not ad-supported and require payment (Paid) have slightly higher ratings compared to free apps without ads. This could imply that users who pay for apps have higher expectations and may be more critical in their ratings. Overall, the plot provides insights into how the ad-supported status and app type can influence user ratings.

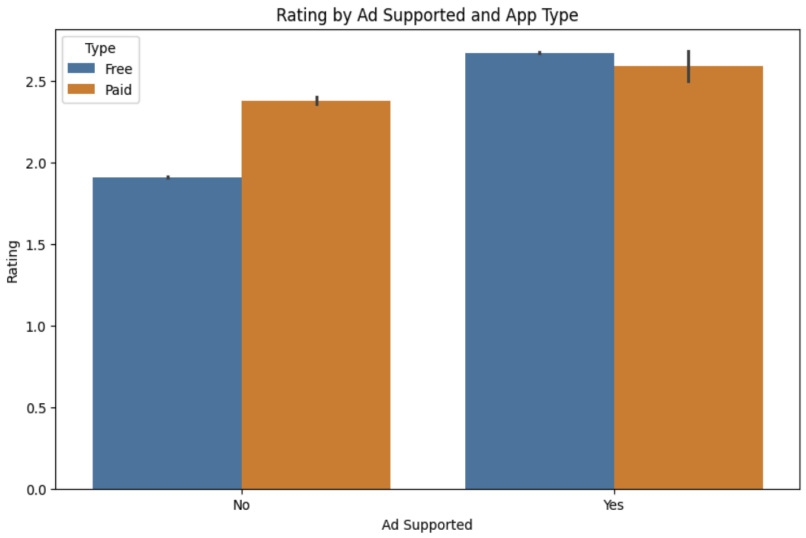


Fig.3.4.10

Paid apps have a slightly higher share of average ratings compared to free apps, with approximately 51.3% of the total ratings attributed to paid apps and 48.7% to free apps. About 58.1% of the total average ratings belong to ad-supported apps and rest for no ad supported. The majority of the average ratings (58.1%) belong to ad-supported apps, indicating that a significant portion of users rate ad-supported apps. This suggests that ad-supported apps may attract more user engagement or attention, potentially influencing the average ratings.

Based on the analysis, the app size, category, ad-supported status, and interaction features with stock market data are important factors influencing app ratings. To improve app ratings, developers could focus on optimizing app size, targeting popular categories, considering ad-supported models, and leveraging insights from stock market data. EDA provides us with valuable insights into the dataset, helping to understand the factors affecting app ratings and informing strategies for improving app performance and user satisfaction. This analysis phase allowed us to gain a deeper understanding of the data and uncover important insights that informed our subsequent modeling efforts. By visualizing distributions, relationships, and correlations within the datasets, we were able to identify key factors that may influence app ratings and develop a more informed approach to predicting app ratings.

3.5.Preprocessing:

The preprocessing steps of one-hot encoding categorical values, standardizing numerical values, and performing a train-test split are crucial for preparing the data for machine learning modeling. These steps ensure that the data is in a suitable format for training and evaluating models, ultimately improving the model's performance and generalization to unseen data. In our preprocessing pipeline, we used the get_dummies function from pandas to perform one-hot encoding on categorical features. This transformed each categorical feature into multiple binary columns, making them suitable for modeling. We utilized the StandardScaler class from scikit-learn to standardize numerical features. This scaler computes the mean and standard deviation of each feature in the training data and applies the transformation to both the training and testing data.Standardizing the numerical features prepares them for modeling, ensuring that they contribute equally to the model's predictions.

3.5.1.Train-Test Split

After preprocessing the data, we split it into training and testing sets to evaluate the performance of our models. We used an 80-20 ratio for the train-test split, meaning that 80% of the data was used for training the model, while the remaining 20% was used for testing. The train_test_split function from scikit-learn facilitated this splitting process, randomly shuffling and partitioning the data based on the specified test size and random state. By splitting the data into separate training and testing sets, we ensure that the model is evaluated on unseen data, providing a more accurate estimate of its performance in real-world scenarios.

3.6.Modeling:

In the Data Modeling section, we aim to train and evaluate several machine learning models on our preprocessed and standardized dataset. The models we chose to explore include Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, XGB Regressor, SVR, and MLP Regressor.

3.6.1.Model Selection & Hyperparameter tuning

Focused on identifying the optimal machine learning model and fine-tuning its hyperparameters to improve performance. After selecting a set of candidate models, we proceed to fine-tune their hyperparameters to improve their predictive performance. Hyperparameters are parameters that control the learning process of the machine learning algorithm and can significantly impact model performance. We used techniques such as GridSearchCV to systematically search through a range of hyperparameters and find the combination that yields the best results. We began by exploring a variety of machine learning models suitable for regression tasks, such as Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, XGB Regressor, SVR, and MLP Regressor. Each model has its own strengths and weaknesses, and by comparing their performance, we can determine which one is best suited for our specific prediction task.

3.6.1.1.Linear Regression

Linear Regression provides a simple and interpretable model that can capture linear relationships between features and the target variable. It serves as a baseline model to compare the performance of more complex algorithms. Also, it is computationally efficient and can handle large datasets with many features.

3.6.1.2.Random Forest

Random Forest can capture nonlinear relationships between features and the target variable, which may exist in the dataset. It is robust to overfitting and less sensitive to outliers in the data. Also, provides feature importance scores, which can help identify the most relevant features for predicting app ratings.

3.6.1.3.Gradient Boosting Regressor:

Gradient Boosting builds trees sequentially, focusing on correcting errors made by previous models. This iterative process often leads to improved performance. It can handle various types of data and is less prone to overfitting compared to other models. Gradient Boosting tends to achieve high predictive accuracy and is suitable for complex regression tasks.

3.6.1.4.XGB Regressor:

Extreme Gradient Boosting: XGB Regressor is an implementation of gradient boosting that is highly optimized for speed and performance. It includes built-in regularization techniques to prevent overfitting and improve generalization. XGB Regressor can handle large datasets and is widely used in industry for its efficiency and scalability.

3.6.1.5. SVR (Support Vector Regressor):

SVR can capture non-linear relationships between features and the target variable using kernel functions. It is robust to outliers in the data and can handle high-dimensional feature spaces effectively. SVR offers flexibility in choosing different kernel functions to model complex relationships.

3.6.1.6. MLP Regressor (Multi-layer Perceptron Regressor):

MLP Regressor is a type of artificial neural network that can learn complex patterns and relationships in the data. It automatically learns hierarchical representations of features, making it suitable for tasks with high-dimensional data. MLP Regressor can scale to large datasets and is capable of capturing intricate patterns in the data.

3.6.3.Cross-validation

Cross-validation assesses the model's generalization performance by splitting the training data into multiple folds. Each model is cross-validated using 5-fold cross-validation. It involves partitioning the training data into folds, training the models on subsets, and validating on the remaining data to ensure robustness. To ensure the robustness of our model selection and hyperparameter tuning process, we utilize cross-validation. This helps us assess how well the model generalizes to unseen data and provides more reliable estimates of its performance.

3.6.4.Training and Evaluation

Each model is trained using the standardized data obtained from the preprocessing stage. We then evaluate the performance of each model using various metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), and R-squared (R2). These metrics provide insights into how well the models are able to predict app ratings based on the input features.

Evaluation Metrics:

- **Mean Squared Error (MSE):** The average of the squared differences between the predicted values and the actual values.
- **Mean Absolute Error (MAE):** The average absolute difference between the predicted values and the actual values.
- **Median Absolute Error (MedAE):** The median of the absolute differences between the predicted values and the actual values.
- **R-squared (R2)**: The proportion of the variance in the dependent variable (target) that is explained by the independent variables (features) in the model.

Throughout the model selection and hyperparameter tuning process, we evaluate the performance of each model using appropriate metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), and R-squared (R2). These metrics provide insights into how well the model is able to predict app ratings and help us compare different models and hyperparameter configurations.

Exploring a variety of machine learning models, fine-tuning their hyperparameters, and selecting the best-performing model based on cross-validated performance metrics. This process ensures that we choose a robust and reliable model for predicting app ratings effectively.

Model Metrics

Models	MSE	MAE	MedAE	R2
Linear Regression	0.538	0.527	0.390	0.457
Random Forest Regressor	0.300	0.367	0.245	0.697
Gradient Boosting Regressor	0.307	0.392	0.275	0.690
XGB Regressor,	0.363	0.399	0.224	0.633
SVR	0.477	0.446	0.278	0.519
MLP Regressor	0.597	0.510	0.336	0.398

Table 3.6.4.1

The chosen models offer a diverse set of strengths, including interpretability, non-linearity, robustness, scalability, and flexibility, making them suitable for tackling the complex task of predicting app ratings in this project. By evaluating multiple models, we can leverage their unique characteristics to achieve the best predictive performance.

From the comparison (Table 3.6.4.1), it's evident that the Random Forest model model outperforms the other models based on the given metrics. It has the lowest MSE, MAE, and MedAE values, indicating better accuracy and precision in predicting app ratings. Additionally, it has the highest R2 score, suggesting a better fit of the model to the data compared to the other models. Therefore, based on these metrics, Random Forest model appears to be the best-performing model for this specific task of app rating prediction.

3.6.5. Feature Importance

We conducted a feature importance analysis on the best-performing model, Random Forest. This analysis allows us to identify the features that have the most significant impact on the model's predictions. The feature importances were calculated and visualized to provide insights into the relative importance of each feature. The analysis identified the top features that exert the most influence on the model's predictions. These features include

- **Install_Range:** This feature likely represents a range of installation counts for the app. It could indicate the popularity or demand for the app. Apps with higher installation ranges might have more visibility or user engagement, potentially leading to higher ratings.

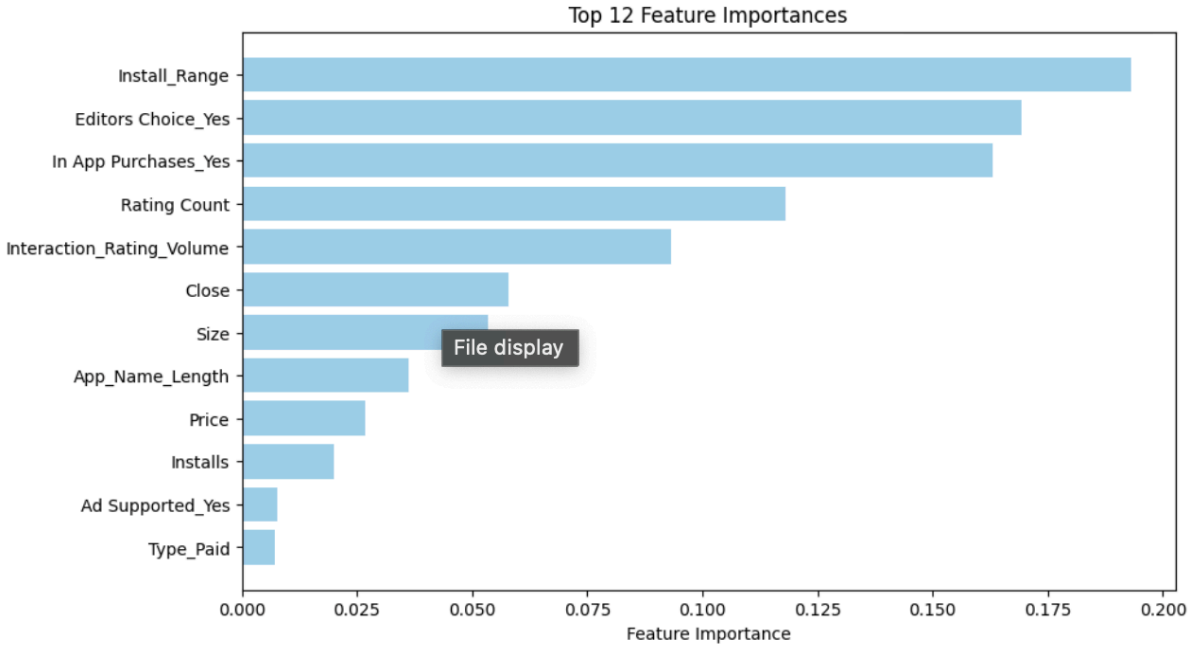


Fig.3.6.5.1

- **Editors Choice_Yes**: This indicates whether the app has been selected as an Editor's Choice. Being chosen as an Editor's Choice can provide additional exposure and credibility for the app, which may positively influence user perception and ratings.
- **In App Purchases_Yes**: This binary feature signifies whether the app offers in-app purchases. Apps with in-app purchases might provide additional content, features, or benefits to users, enhancing their overall experience and potentially leading to higher ratings.
- **Rating Count**: This feature represents the total number of ratings received by the app. Higher rating counts indicate greater user feedback and engagement. Apps with a large number of ratings may be perceived as more established or trustworthy, influencing potential users and their ratings.
- **Interaction_Rating_Volume**: This feature could represent the volume or intensity of interactions (e.g., user reviews, comments, likes) related to the app. Higher interaction volumes might indicate greater user engagement or community activity, which could correlate with higher app ratings.
- **Size**: This feature represents the size of the app in terms of storage space required for installation. While not directly related to app quality, smaller app sizes are generally preferred by users, as they consume less device storage and are quicker to download.
- **App_Name_Length**: The length of the app name might indirectly influence ratings. Short, catchy names might be more memorable and easier to find in app stores, potentially leading to more downloads and higher ratings.
- **Price**: The price of the app, if it's a paid app, can influence user perceptions of value. Users may have higher expectations for paid apps and thus provide more critical ratings compared to free apps.

- **Installs**: This feature likely represents the total number of installations for the app. Higher installation numbers can signal popularity and trustworthiness, positively impacting user perceptions and ratings.
- **Ad Supported_Yes**: This feature indicates whether the app contains advertisements. Ad-supported apps might offer free content in exchange for displaying ads, which could affect user experience and ratings depending on the implementation.
- **Type_Paid**: This feature indicates whether the app is paid. Paid apps may have different user expectations compared to free apps, potentially affecting ratings.

The analysis revealed several top features that exert the most influence on the model's predictions. These features represent various characteristics of the app, including its popularity, category, and genre, which can significantly influence user perceptions and ratings. Overall, the feature importance analysis provides valuable insights into the factors driving app ratings and helps prioritize features for further investigation or optimization. By understanding which features have the most significant impact on app ratings, developers and stakeholders can make informed decisions to improve app quality, enhance user experience, and ultimately drive higher ratings and user satisfaction. A bar (Fig.3.6.5.1) chart was generated to visually represent the importance scores of the top features. This visualization aids in the quick interpretation of which features play a pivotal role in the model's decision-making process.The feature importances offers valuable insights into the underlying patterns within the dataset. Features with higher importances suggest a stronger association with the target variable and contribute significantly to the model's ability in apps rating prediction.

4.CONCLUSION:

In conclusion, this project aimed to develop predictive models for app rating prediction by training and evaluating various machine learning algorithms on a dataset combining Google Play Store app characteristics and Google's stock market performance data. It was found that the Random Forest Regressor model outperformed the other models, achieving an R-squared value of 0.697. This indicates that the Random Forest model explains a significant proportion of the variance in app ratings and provides the best fit to the data among the models tested.

Additionally, a feature importance analysis was conducted to identify the key features that contribute most to predicting app ratings. The analysis revealed important features such as, Interaction_Rating_Volume,Install_Range, Editors Choice_Yes, In App Purchases_Yes, Rating Count, Interaction_Rating_Volume, etc. These features provide valuable insights into the factors influencing app ratings and can guide developers and stakeholders in optimizing app performance and user satisfaction. This project demonstrates the effectiveness of machine learning models in predicting app ratings and highlights the importance of feature selection and analysis in understanding the underlying factors driving app success.

5.FUTURE WORK:

In future, researchers and practitioners could explore additional features that may influence app ratings, such as user reviews sentiment analysis, app update frequency, developer reputation, or app

category-specific characteristics. Increasing the dataset size can potentially enhance the robustness and generalization ability of models. Experimenting with model ensembling techniques could also be beneficial to combine the predictions of multiple models and potentially improve predictive performance. Moreover, conducting time-series analysis to capture temporal patterns and trends in app ratings over time could provide deeper insights into user behavior and preferences. Developing dynamic models that can adapt to changing market conditions and user preferences in real-time would enable more accurate and timely predictions of app ratings. Additionally, performing localized analysis to understand how app ratings vary across different geographical regions or demographic segments could help tailor app development and marketing strategies. Integrating data from external sources such as social media platforms, app store analytics, or competitor performance metrics would enrich the analysis and enhance predictive accuracy.