ANDROID APPS DOWNLOAD PREDICTION

Uyyala Aishwarya(U00349057)

Jillella Sai Shreya(U00349056)

Pendyala Thanvi Lahari(U00353645)

Abstract: In today's digital era, the Google Play Store hosts millions of applications catering to diverse user needs. For app developers and marketers, predicting the download count of an application prior to its launch is crucial for strategizing marketing campaigns, estimating revenue potential, and optimizing resource allocation. This project is to develop a predictive model for estimating the download count of Android applications based on various features available in the dataset.

Table of Contents

1. Introduction to Project

Introduction

This project focuses on predicting download counts for Android apps, crucial for developers, marketers, and investors. Leveraging machine learning techniques, we aim to provide accurate predictions to optimize app performance and marketing strategies. The mobile app industry is experiencing exponential growth, with millions of apps available on platforms like Google Play Store. Predicting app download counts is vital for developers, marketers, and investors to make informed decisions.

Motivation

Predictive analytics in the mobile app industry is essential for informed decision-making, resource allocation, and maximizing ROI. Accurate download predictions help developers understand user preferences, optimize app features, and allocate marketing budgets effectively.

Contribution

This project on Android app download prediction offers a robust framework for app developers and marketers to leverage predictive analytics, enabling informed decision-making. By analyzing key factors like category, size, ratings, and reviews, the model facilitates precise estimations of app download counts, guiding marketing strategies and resource allocation. Through actionable insights and trained machine learning models, this project empowers stakeholders to optimize app development and marketing endeavors, fostering enhanced user engagement and revenue generation in the competitive app ecosystem.

Research Questions and Rationale

Question: How do different machine learning algorithms compare in terms of performance for predicting app download counts?

Rationale: Comparing the performance of various algorithms such as linear regression and random forest regression can help identify the most effective approach for download count prediction in the Android app ecosystem.

Question: What are the key factors influencing app downloads, and how do they contribute to the prediction model?

Rationale: Understanding the significant factors such as app category, size, ratings, and reviews can provide valuable insights into user preferences and behaviors, leading to better predictive models.

2. Dataset

Overview of Dataset

A dataset is a permanent set of data that is stored as a Catalog item. Datasets contain data that can be viewed and altered in a grid, where columns and rows form intersecting cells that store data. Datasets can be used for storing static data and will often represent data not stored in records that exist in the Meridium database. While questions return the most current information from the database, a dataset shows the same data every time you view the dataset. For example, you might use a dataset to store data from an external system so that the data can be used to make reports and graphs in the Meridium APM Framework application.

Data Preprocessing

Data preprocessing involves cleaning the data by removing duplicates, handling missing values, and filtering out irrelevant or noisy tweets. Textual data is tokenized, and techniques such as stemming, and lemmatization are applied to normalize the text. Feature extraction techniques, such as bag-of-words or word embeddings, are employed to represent text in a numerical format suitable for analysis.

Dataset Description

Source and Size: The dataset is sourced from Kaggle and comprises information on over 600,000 applications from the Google Play Store.

Data Set Link: https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps

Attributes: It includes 23 attributes such as App Name, App ID, Category, Rating, Rating Count, Installs, Minimum and Maximum Installs, Free, Price, Currency, Size, Minimum Android version, Developer ID, Developer Website, Developer Email, Release Date, Last Updated, Content Rating, Ad Supported, In-App Purchases, and Editor's Choice.

App Information: Each entry provides details about an application, including its name, category, ratings, number of installs, pricing, size, developer information, release date, content rating, and more.

Data Completeness: The dataset contains various types of data, including numerical, categorical, and textual, offering comprehensive insights into the characteristics of Google Play Store apps.

Use Cases: It can be utilized for various purposes such as market analysis, trend identification, app performance evaluation, and machine learning model training for predicting app ratings or popularity**.**

Evaluation Metrics

Employed mean squared error (MSE) as the evaluation metric for regression tasks to measure the average squared differences between the predicted and actual ratings. For classification tasks, precision, recall, and F1-score were used to assess the performance of the models in predicting the app rating categories.

- **Precision:** It is the ratio of acceptably predicted positive observations to the total predicted positive observations. It is also identified as the positive predictive value.
- **Recall:** It measures the ratio of acceptably predicted positive observations to all actual positives. These metrics are mostly useful when dealing with imbalanced datasets.
- **F1 Score:** It is the mean of both precision and recall. It is a balance between the two metrics and is mainly useful when you need to take both false positives and negatives into account.
- **MSE** (Mean Squared Error): It measures the average of the squares of the errors that is, the average squared difference between the estimated values and the actual value.

3. Software Details:

Programming Language: Python

Data Source: Employed Kaggle's "Google Play Store Apps" dataset for analysis.

Software: Jupyter Notebook or any Python IDE for code development and experimentation.

Tools:

- Pandas for data manipulation and analysis.
- NumPy for numerical computations.
- Seaborn is an incredible visualization library for statistical graphics plotting in Python.
- Matplotlib is a 2D plotting library for Python. It provides a wide variety of charts and plots for visualizing data
- Applied machine learning techniques from scikit-learn library for regression and classification tasks.

Technology used:

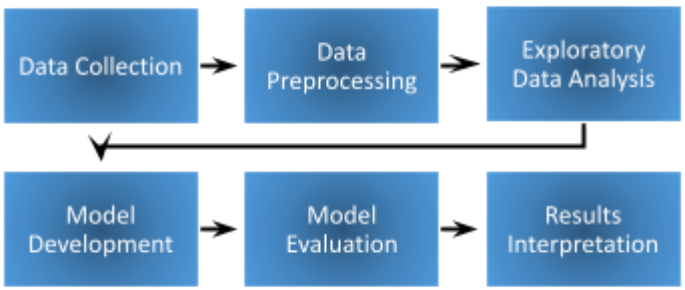
Random Forest Regression: Employed to enhance prediction accuracy through ensemble learning.

- **Random Forest:** A Random Forest is like a collection of decision-making team in machine learning (ML). It associations the opinions of many trees to make better predictions, making an extra robust and accurate overall model. Random Forest Algorithm extensive popularity stems from its user-friendly nature and adaptability, enabling it to tackle both classification and regression problems efficiently. The algorithm's strength lies in its capability to handle composite datasets and mitigate overfitting, making it a valuable tool for various predictive tasks in machine learning.

Linear Regression: Utilized for predicting download counts based on various features.

- **Linear regression**: It is used to predict the value of a variable based on the value of another variable. The variable you need to predict is called the dependent variable. The variable you are utilising to predict the other variable's value is called the independent variable.

Overview Diagram



Technical Steps:

Data Collection: Obtain the Google Play Store Apps dataset from Kaggle.

Data Preprocessing:

- Handle missing values.
- Convert categorical variables into numerical format.
- Normalize numerical features.

Exploratory Data Analysis (EDA):

- Visualize relationships between features and download counts.
- Classify pattern and correlations in the dataset.

Model Development: Train machine learning models (e.g., Linear Regression, Random Forest Regression) using preprocessed data.

Model Evaluation:

- Assess model performance using metrics like Mean Squared Error.
- Compare different algorithms to select the most suitable one.

Results Interpretation: Analyze insights gained from the model.

4. Implementation

```
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
```

The above Snippet represents what are the libraries we used and import.

```
df=pd.read_csv('Google-Playstore.csv')
```

The above code represents adding the .csv file from the database.

	App Name	App Id	Category	Rating	Rating Count	Installs	Minimum Installs	Maximum Installs	Free	Price	...	
0	Gakondo	com.ishakwe.gakondo	Adventure	0.0	0.0	10+	10.0	15	True	0.0	...	
1	Ampere Battery Info	com.webserveis.batteryinfo	Tools	4.4	64.0	5,000+	5000.0	7662	True	0.0	...	http
2	Vibook	com.doantiepvien.crm	Productivity	0.0	0.0	50+	50.0	58	True	0.0	...	
3	Smart City Trichy Public Service Vehicles 17UC...	cst.stJoseph.ug17ucs548	Communication	5.0	5.0	10+	10.0	19	True	0.0	...	http://w

The above data represents what information available in csv file

```
print(df.isna().sum())
```

```
App Name                22883
App Id                      0
Category                    0
Rating                  22883
Rating Count            22883
Installs                  107
Minimum Installs          107
Maximum Installs            0
Free                        0
Price                       0
```

The above Snippet represents the sum of the individual items present in the database file.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2312944 entries, 0 to 2312943
Data columns (total 24 columns):
 #   Column              Dtype
---  ------              -----
 0   App Name            object
 1   App Id              object
 2   Category            object
 3   Rating              float64
 4   Rating Count        float64
 5   Installs            object
 6   Minimum Installs    float64
 7   Maximum Installs    int64
 8   Free                bool
```

The above snippet and data represents the attributes and type attributes from the file.

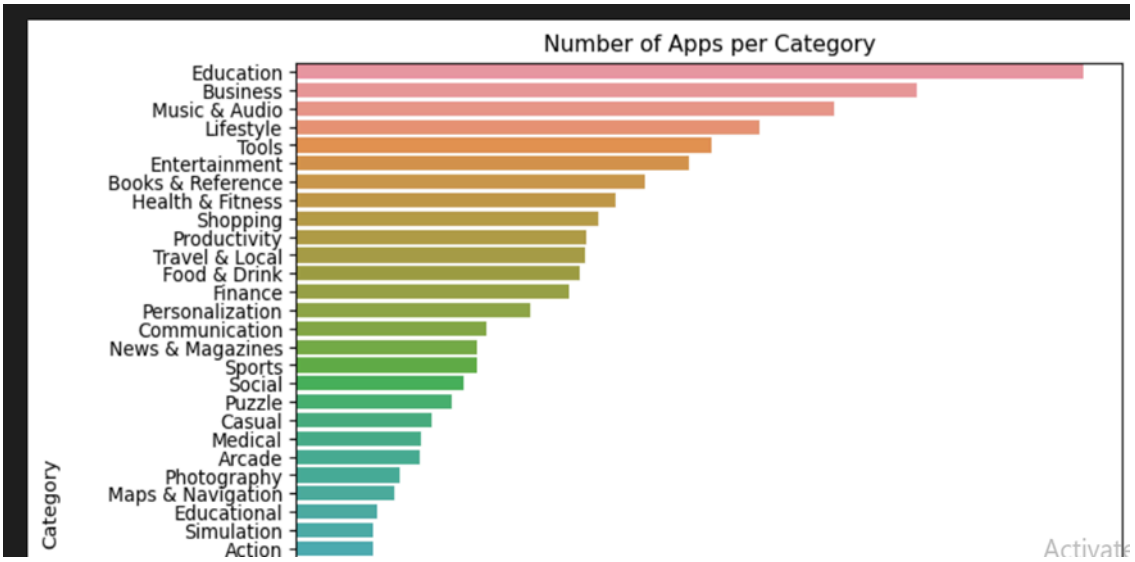
```
print('Number of Apps per category')
display(df.Category.value_counts())
fig=plt.figure(figsize=(8, 8))
a = sns.countplot(y = df.Category,order
```

```
Number of Apps per category

Category
Education                   127219
Business                    100311
Music & Audio                87008
Lifestyle                    75017
Tools                        67215
Entertainment                63581
Books & Reference            56515
Health & Fitness             51770
Shopping                     48981
```

The above snippet and out represents the number of apps belongs to particular category.

Results and Analysis:



The above output represents the number of Apps belongs to particular category in bar graph.

Graphical Representations:

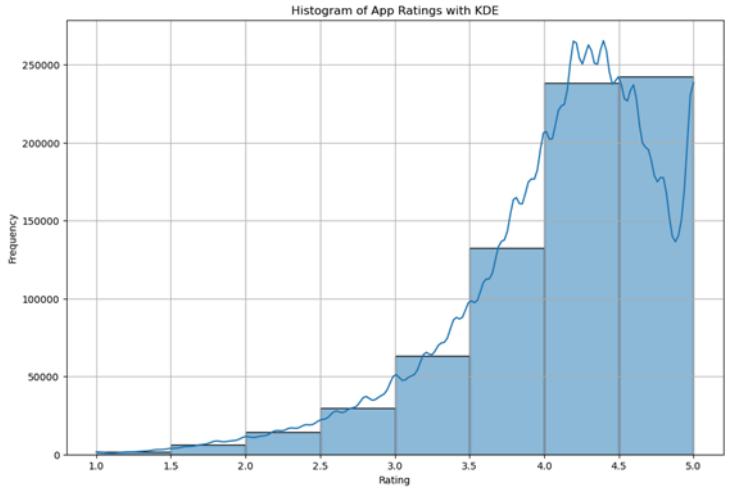


Fig: Histogram of App Ratings with KDE

Graph Axis: X-axis: Rating, Y-axis: Frequency

Explanation: This histogram displays the frequency distribution of app ratings, with Kernel Density Estimation (KDE) for smoothness.

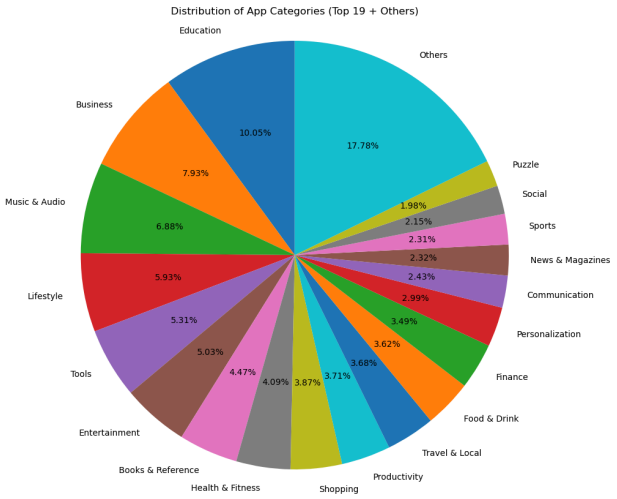


Fig: Distribution of App Categories (Top 19 + Others).

Explanation: The pie chart illustrates the distribution of app categories, showcasing the top 19 categories and grouping the rest as 'Others.'

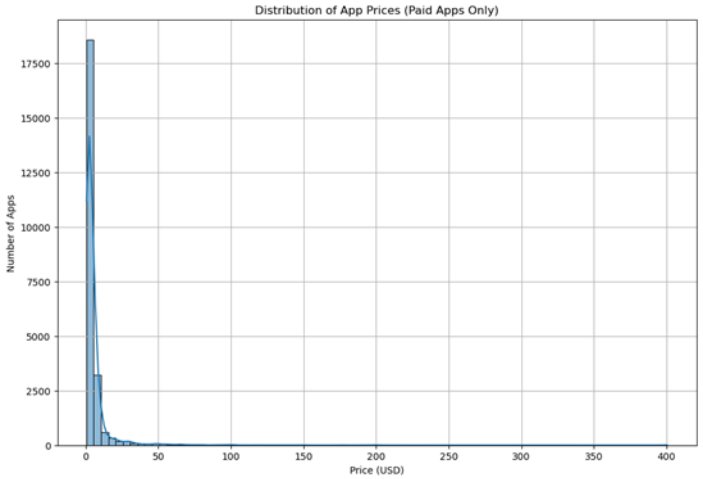


Fig: Distribution of App Prices (Paid Apps Only).

Graph Axis: X-axis: Price (USD), Y-axis: Number of Apps

Explanation: The histogram displays the distribution of prices for paid apps, with Kernel Density Estimation (KDE) for smoothness.

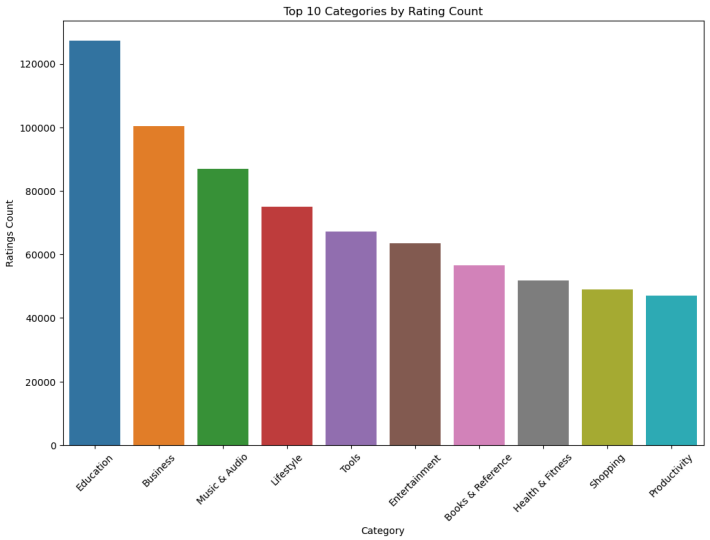


Fig: Top 10 Categories by Rating Count.

Graph Axis: X-axis: Category, Y-axis: Ratings Count**.**

Explanation: This bar plot highlights the top 10 categories with the highest rating counts, providing insights into popular app categories.

5. Conclusion, Future Scope

Findings and Conclusions:

- **Research Findings:** Regression models outperform classification models in predicting app ratings based on features like category, rating count, and size.
- **Model Performance:** Random Forest Regressor achieved the lowest mean squared error of 3.26 compared to Linear Regression's 4.35.
- **Classification Outcome:** Random Forest Classifier demonstrated better classification results, particularly for medium and high-rated apps.
- **Recommendation:** Random Forest models are recommended for predicting app ratings due to their superior performance over other models.

Limitations:

- **Data Quality:** Incomplete or inaccurate data in the dataset may affect the performance of the predictive model.
- **Feature Selection:** The selection of features may not fully capture all relevant factors influencing app download counts, leading to potential bias.
- **Generalization:** The predictive model's performance may vary across different app categories or user demographics, limiting its generalizability.
- **External Factors:** External factors such as changes in market trends or competitor strategies could impact app download counts, beyond the scope of the model.

Future Scope:

- **Advanced Feature Expansion:** Include user demographics, sentiment analysis, and engagement metrics for richer insights.
- **Dynamic Pricing Strategy:** Adjust prices based on demand and competition for optimized revenue.
- **Refined User Segmentation:** Group users by preferences for personalized experiences and targeted marketing.

References

1. Gautham Prakash, "Google Play Store Apps", https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps
2. Sabreen Abulhaija, Shyma Hattab, "Predicting Mobile Apps Performance using Machine Learning ", https://www.aasmr.org/jsms/Vol12/JSMS%20DEC%202022/Vol.12.No.06.19.pdf
3. Shiva Prakash S, "App Review Prediction Using Machine Learning", https://www.researchgate.net/publication/366714819_App_Review_Prediction_Using_Machine_Learning
4. Zhenghuan Zhong1 , Yuhang Bao2, "Predict New App Quality By Using Machine Learning", https://iopscience.iop.org/article/10.1088/1742-6596/1693/1/012111/pdf