Google Play Store Rating Prediction

Milestone: Project Proposal

Group 5

Niraj Sai Prasad
Sindhu Swaroop

857-269-4546
857-269-4536

saiprasad.n@northeastern.edu
swaroop.s@northeastern.edu

Percentage of Effort Contributed by Student 1: 100%
Percentage of Effort Contributed by Student 2: 100%

Signature of Student 1: NIRAJ SAI PRASAD
Signature of Student 2: SINDHU SWAROOP

Submission Date: 04/23/23

Problem Setting:

Google Play Store, formerly known as the Android Market, has come a long way since its birth in 2008. With over 3.5 million apps and various other content such as books, movies and games listed on the platform, it is the most popular digital distribution service amongst android app developers as well as android cell-phone users. Google Play has won consumers' trust because of the security protocols it has in place.
While Google Play provides developers ease of app distribution, apps do require to be marketed for better reach and more downloads. Success of an app does not solely depend on how well optimized it is. It depends on user experience, ratings, and reviews. Google Play has a star-based rating system in place with 5 stars being the highest on the scale. Every decision we take in today's world is in the least, based on a quick glance at the rating and the number of ratings. A user's experience in using a particular app and how they would rate the app helps other consumers determine if they would want to install the app.

Problem Definition:

User ratings can be considered as an accurate representation of consumer impression on the application. Lower-rated applications are often at the bottom of search results, have no reach and may even be flagged and removed from the platform. Consequently, every developer's main aim is to maximize the rating of the app.

The goal of this project is to predict the overall rating of an app as ratings play the most important role in gaining users' confidence. Higher-rated apps and apps with a higher number of ratings are more likely to be recommended by the search algorithm on Google Play and more likely to be trusted by users that find the app while browsing the app store. This project aims to employ supervised machine learning, data mining, and visualization concepts to gain insights into what leads to higher number of installations and high user ratings - which loosely translates to the success of applications.

Few of the business questions we are looking to answer through this analysis are:

1. What category of apps has the greatest number of downloads? Group downloads by category to gain insight into what kind of apps are most popular among consumers.
2. When a user installs an app is it likely he will rate it? Is there any relation between number of ratings and number of downloads?
3. What is the mean and median rating of apps on play store? Plot a frequency distribution of ratings to visualize this.
4. Are paid apps more likely to be rated and reviewed by users on the platform? Does paying a price for apps mean that users feel more obligated to give feedback?
5. Does size of the app influence the number of installs? Do users prefer apps which occupy lesser space on their phones?

Some machine learning models that we plan to use for rating prediction are:

1. KNN Regressor
2. Random Forest Regressor

Data Source:

https://www.kaggle.com/code/taghredsalah199/google-playstore-regression-model/data

Data Description:

The data has been selected from Kaggle. The raw dataset comprises 10,841 records and 13 attributes. The different data fields in the dataset are as follows:

Sr no.	Attribute	Description
1.	App	Name of the application
2.	Category	Category of the application
3.	Rating	Rating provided by the users to the application
4	Reviews	Number of reviews provided by the users to the application
5.	Size	Size of the application in kilobytes
6.	Install	Number of users who have installed the application
7.	Type	Free or Paid

8.	Price	Price of the application to download in USD
9.	Content Rating	Age of the target audience
10.	Genres	Subcategory
11.	Last Updated	Date when the application was last updated on Play Store
12.	Current Version	Current version of the application available on Play Store
13.	Android Version	Android version required to install the application

Dataset understanding:

The raw dataset contains 10,841 records and 13 attributes. Our target variable is the attribute 'Rating', which varies from 0 to 5, 5 being the highest possible rating. The dataset is a combination of numerical and string attributes.

Dataset cleaning and pre-processing:

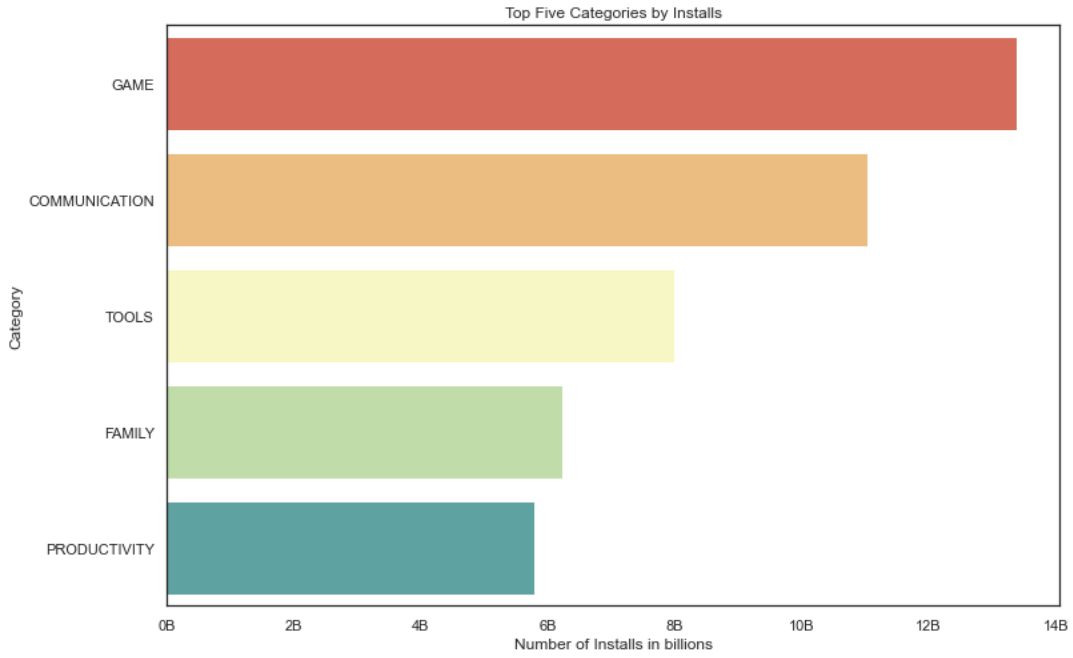
The steps we took to clean and pre-process data:

1. Listed out all column names and renamed some of them (added underscores) to enhance readability and for ease of use.
2. Found the number of nulls in each column. 'Rating' column had the greatest number of nulls (1474) followed by a few other columns like 'Current_Version', 'Android_Version', 'Type', etc. Since this is unwanted data, we dropped all records with null values. The number of records dropped to 9360.
3. The 'Price' attribute had a '$' sign and comma separators in between the digits. This makes it difficult to treat price as a numerical value and perform calculations and analysis. So, we removed these characters from the 'Price' feature.
4. Similarly, the 'Installs' feature also had unnecessary characters like '+' and ',' which we removed.
5. Next, we checked the dtype of each feature and found that all but 'Rating' was listed as an object. We changed 'Price' to 'float' and 'Installs' to 'int' data type to make them easier to work with.

Our final cleaned dataset consists of 9360 records and 13 columns.

Dataset Exploration and Visualization:

1. Top 5 App Categories based on Number of Installations



Observations:

- From the chart we can see that **most apps downloaded** on Playstore are either **Games or Communication** based.
- This is a great representation because entertainment is top priority and communication services like Whatsapp and Messenger are essential in today's world.

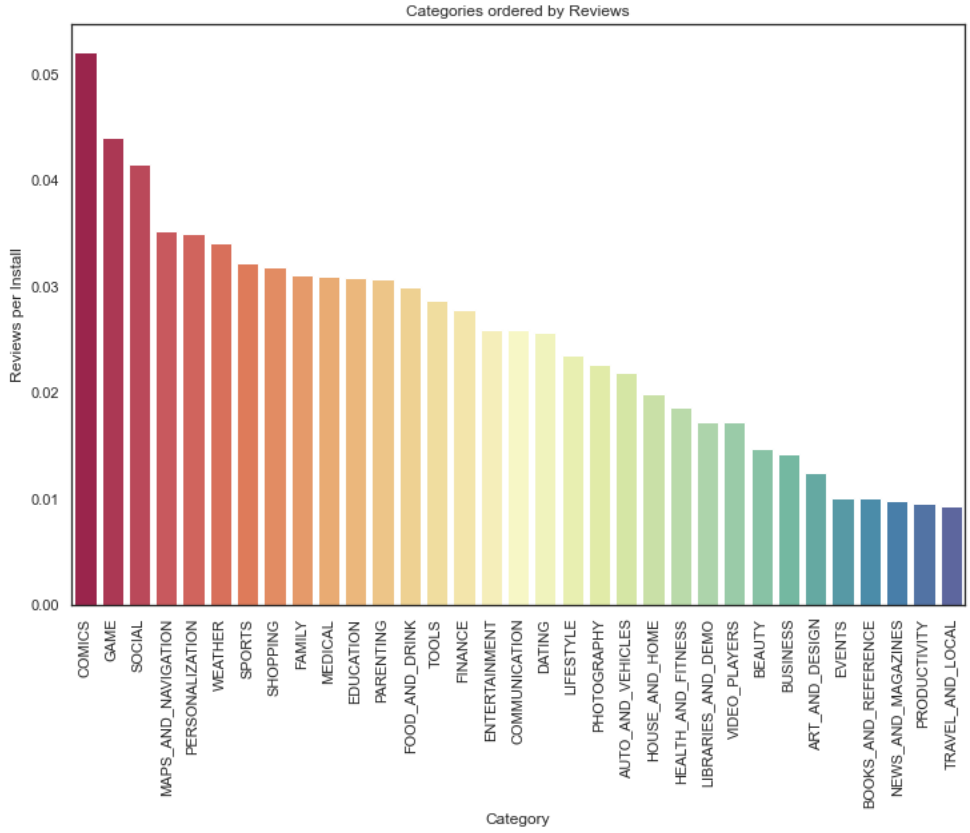
2. Top 5 Highest Rated Apps based on Number of Reviews

	App	Category	Rating	Reviews	Installs	Type	Content_Rating
0	Facebook	SOCIAL	4.1	78158306	1000000000	Free	Teen
1	WhatsApp Messenger	COMMUNICATION	4.4	69119316	1000000000	Free	Everyone
2	Instagram	SOCIAL	4.5	66577446	1000000000	Free	Teen
3	Messenger – Text and Video Chat for Free	COMMUNICATION	4.0	56646578	1000000000	Free	Everyone
4	Clash of Clans	GAME	4.6	44893888	100000000	Free	Everyone 10+

Observations:

- The top 4 apps with **most number of reviews and highest rating** all belong to Meta - they are all **social media apps or communication services**.
- **Clash of Clans** looks to be one of the **most popular games** on Playstore.

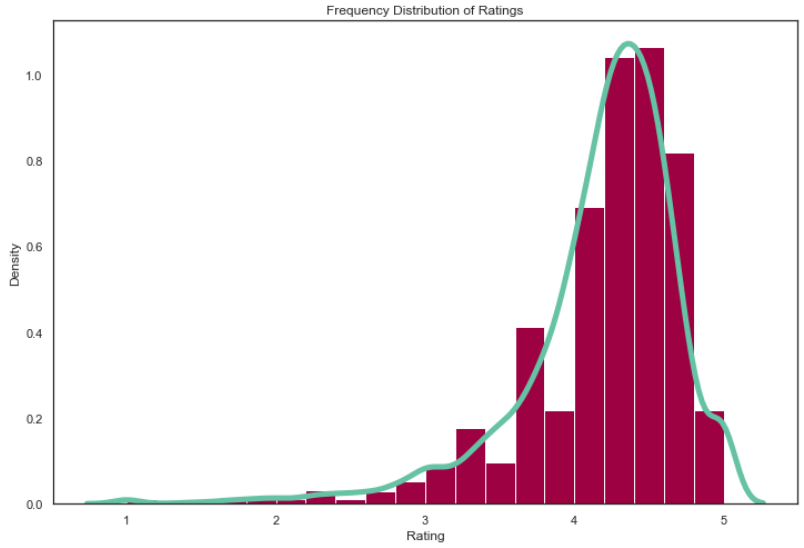
3. App Categories ordered by Number of Reviews



Observations:

- We calculated a new field '**Reviews per install**' to use for this chart so as to not skew the results (some records have high number of installs and hence more reviews).
- **Comics** is the **most reviewed category** followed by **games and social media**.
- The **least reviewed categories** are **books, news, productivity and travel**.

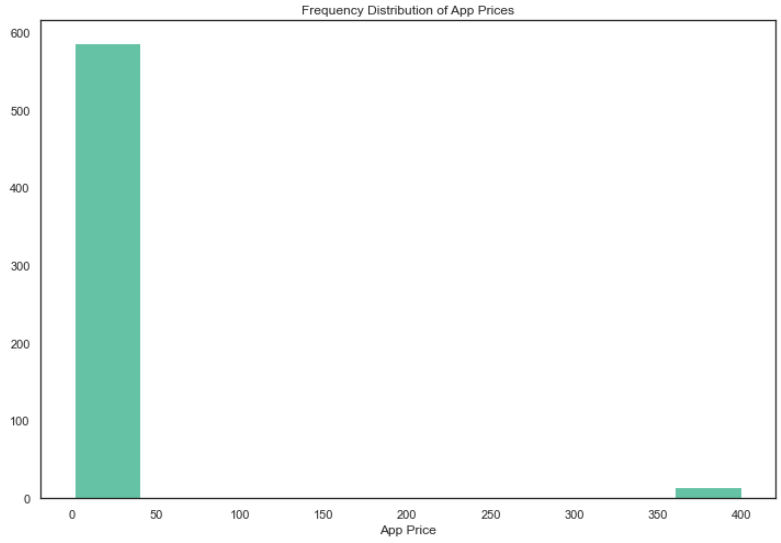
4. **a) Frequency Distribution of Ratings on Playstore**



Observations:

- The **most common rating** on playstore is between **4.5 and 4.6** as per this dataset.
- The distribution tells us that most ratings **range between 4 and 5**.
- **Ratings below 3 are low** in number which implies that people are generally content with apps.
- **Ratings around 1.5** are practically non-existent.

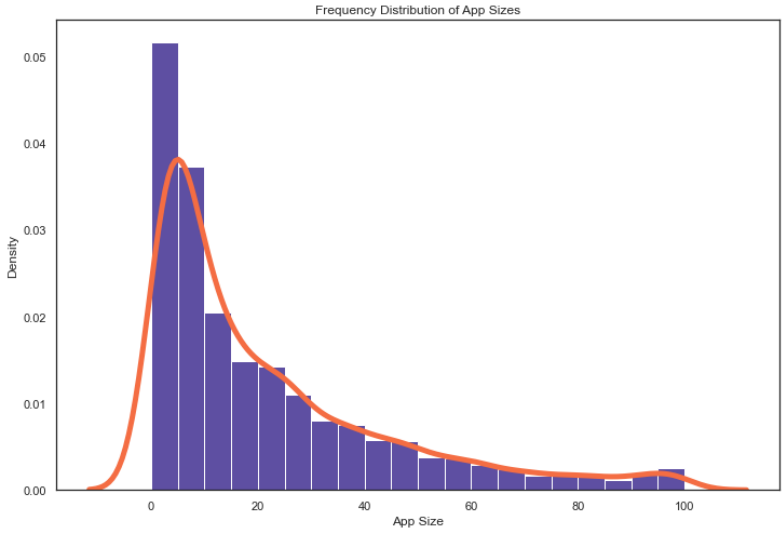
b) Frequency Distribution of App Prices



Observations:

- Of the paid apps, most apps have their prices in the range **1-45 USD**.
- About **14 apps** have a price of **>350 USD and <400 USD**.

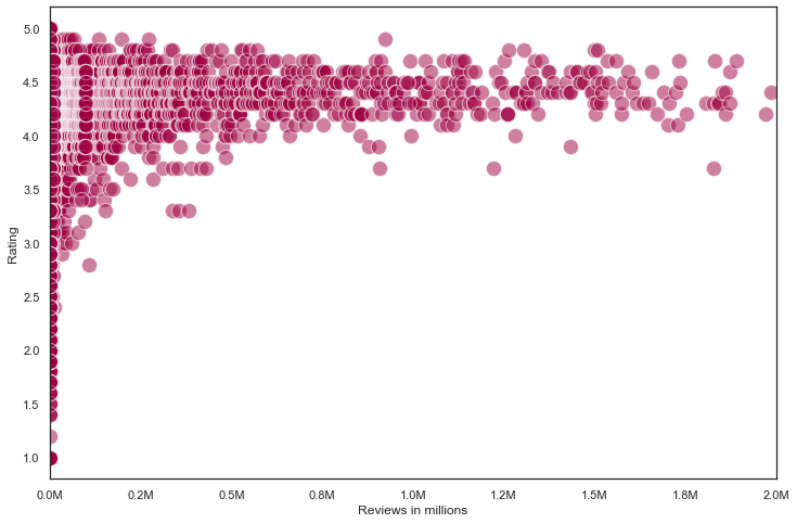
c) Frequency Distribution of App Sizes



Observations:

- Of the 8000+ apps in the dataset, **~4263 apps** have a size of **<20 MB**.
- About **14 apps** have a size of **100 MB** which is the maximum in this dataset.

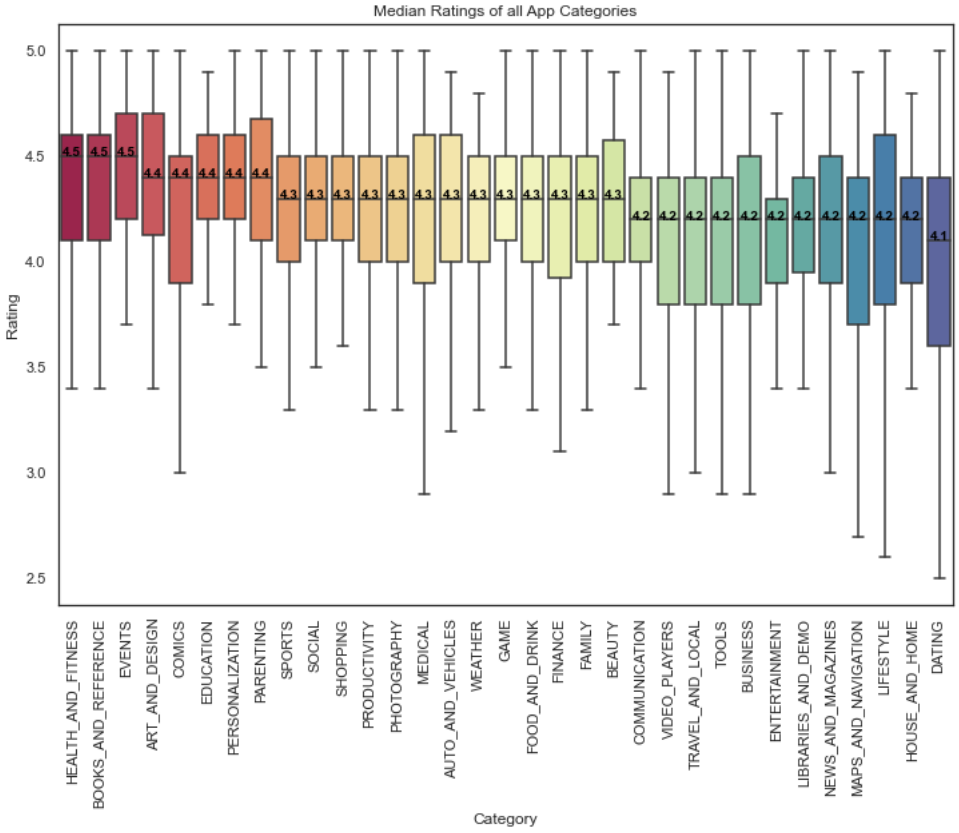
5. Ratings vs. Number of Reviews



Observations:

- Most app ratings are in the range **3.5 and 5** as can be seen from the above graph.
- The apps which have **most number of reviews** are the apps rated in the approximate range of **4.25-4.75**.
- For most apps, the number of reviews are **less than 0.5 million**.
- There are a few apps which have more than 4 million reviews and even close to 80 million reviews which we have treated as outliers in this chart.
- Not many apps have low ratings and less reviews.

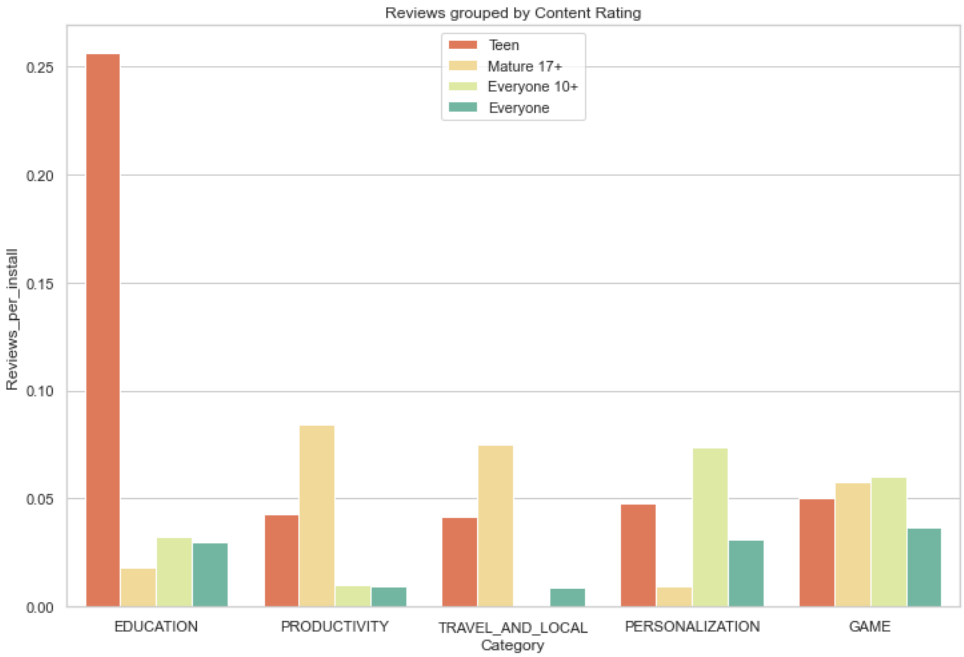
6. Median Ratings of all App Categories with Lower and Upper Quartiles



Observations:

- The categories with the **highest median rating of 4.5** are **Health & Fitness, Books and Events**.
- The **lowest median rating** of a category in the dataset is **4.1 for Dating**.
- Barring outliers which have been removed from this plot, the **lowest rated app** also belongs to the **Dating** category.
- Some categories like **Education, Weather, Beauty and Maps** do not have a **single app rated 5 stars**.
- The **Entertainment** category has the **lowest upper quartile and the lowest maximum rating**.

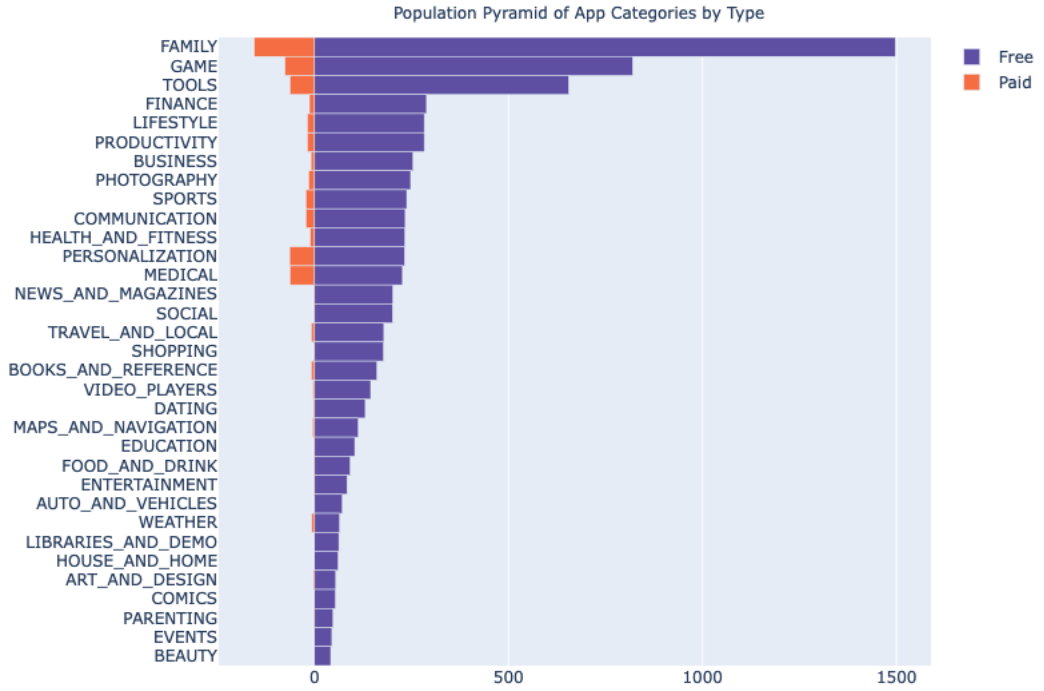
7. Top 5 Most Reviewed App Categories based on Content Rating



Observations:

- Taking reviews per install into consideration for this chart as well, **Education** seems to be **most reviewed by Teens** which seems accurate.
- **Games** category is almost equally reviewed by **all age groups**.
- **Productivity and Travel** are reviewed mostly by the **Mature 17+ category** followed by **Teens**.

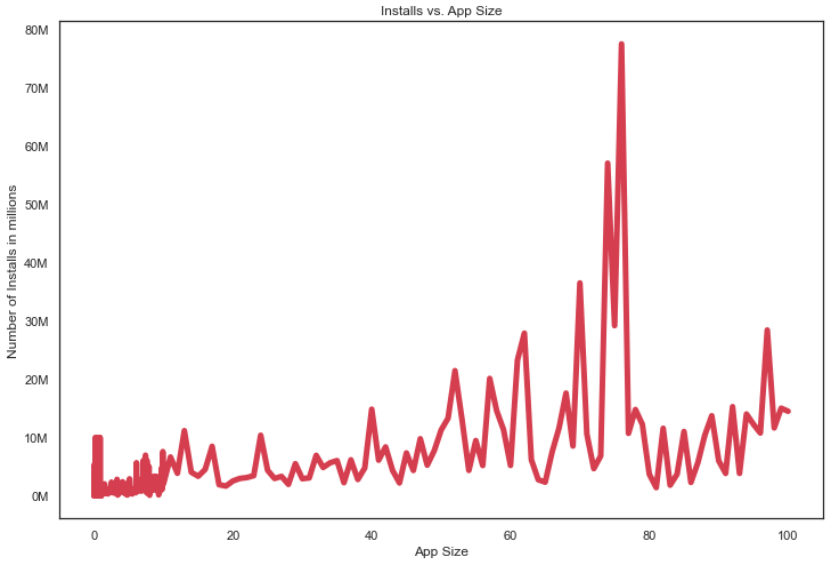
8. Population pyramid of App Categories by Type



Observations:

- The population pyramid of app count clearly tells us that the **majority of apps** on the Playstore are **free**.
- Looking at the ratio of Free to Paid apps in all categories on Playstore, most number of apps as well as the **most number of paid apps** belong to the **Family** category.
- There are aboout **76 paid apps** in the **Game** category.
- **Tools, Personalization and Medical** also have about 60 paid apps.
- **Comics, Libraries, Home, Vehicles, Events and Beauty** categories only have **free apps**.

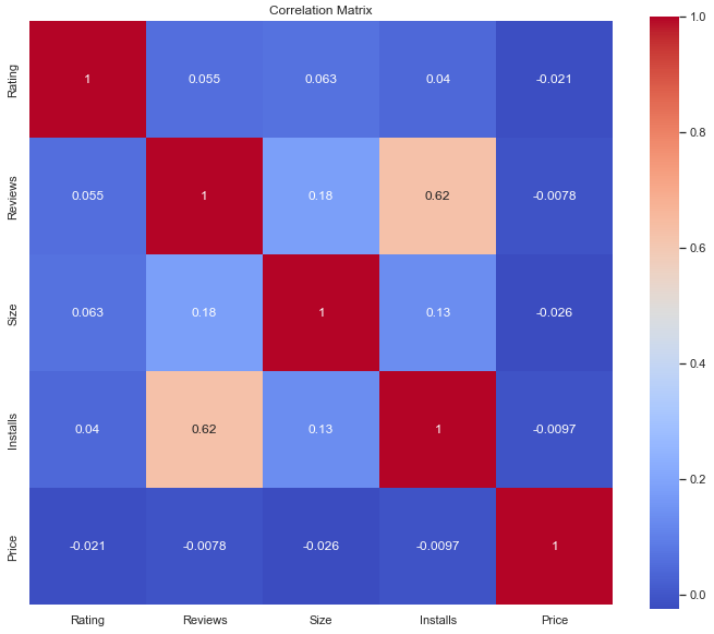
9. App Sizes vs. Installs



Observations:

- Apps of around **70-80 MB** size have the **highest** number of installs **(~80M)**.
- Apps of **close to 100 MB** size have around **30M installs**.
- Apps of size **<10MB** are mostly downloaded lesser **(~<10M installs)**.

10. Correlation Matrix



Observations:

- Looks like all numerical columns do not have a very high correlation, meaning we can use all these as features for our ML models
- Reviews vs Size has a relatively high correlation, but that makes sense because higher reviews result in higher number of installs. Hence, both these features are important.

Data Cleaning and Mining Tasks:

- Removing nulls
- Dropping duplicates
- Data imputation and handling special texts
- Converting columns to numeric/float as required
- Removing special characters that to avoid confusion:
  - Replacing '$' in price ['$200' to 200]
  - Replacing 'M' and 'K' in the app size and using appropriate multiplier [10M to 10000000]
- Creating dummies for categorical variables
- Finding correlation between columns
- Dimensionality Reduction:
  - Genres & Category : There are many unique categories (~10) and genres (>20). Making dummies and training the model with these features resulted in a very low difference of metrics. The training time was more, and cost was more. Hence, we did not use it as predictor
  - Features like 'Last_Updated', 'Current_Version' and 'Android_Version' are not relevant to our analysis, hence we've removed these.
  - Type and Price convey the same information, hence we removed Type for our analysis.

Data Mining Models/Methods:

Since this is a prediction problem where we need to predict the rating of an application based on various factors (or predictors), a regression model would be an ideal approach.
We plan to split the data in the ratio of ***80:20*** for training and testing the model respectively.
Some of the regression models that we plan to use are:

1. **Linear Regression:** Our baseline model would be linear regression. LR compares input and output variables based on linear/labeled data. Since our data are mostly numeric (we can convert categories into binary using one-hot encoding), LR models fit well. We can find the association between two variables using the correlation coefficient the value of which ranges from -1 to 1.
   Some advantages of the LR model are:
   - The model is easy to use and understand, it is of low complexity
   - Training time and efficiency is great
   - If the data is truly linear, then fitting the data is perfect

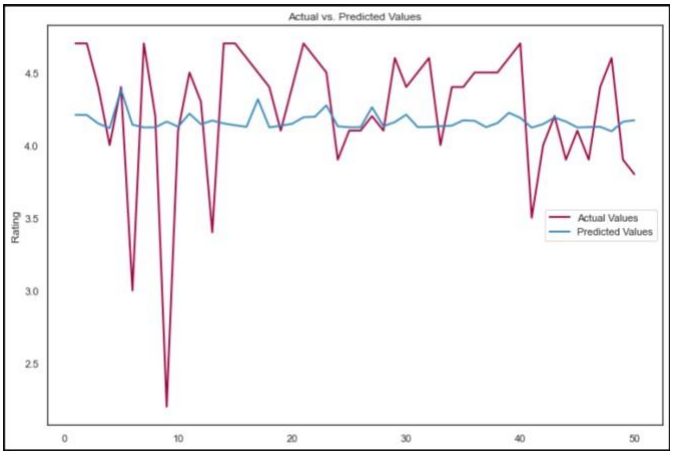
Some disadvantages of the LR model are:

- It assumes that all predictors are linear
- It usually tends to under-fit the data
- If outliers exist, it's performance dips

Implementation: This is our baseline prediction model, and we are using the default parameters and settings to find the score and accuracy. Default parameters include fit_intercept = True, normalize = False and copy_x = True. We got the following results:

- Score – 0.016
- MSE – 0.28
- RMSE – 0.53

Plot for actual vs predicted in Linear Regression:



2. **KNN Regressor:** The K-Nearest-Neighbor algorithm predicts the value based on how close it is to the nearest neighbor values. This distance can be measured using the following methods:
    - Euclidean
    - Manhattan
    - Statistical

    The 'K' is the number of neighbors the training model considers. KNN generally is used for classification problems but adding a regressor to it makes it predict numerical values. The model then predicts the value based on the mean of predictor neighbor values.
    Advantages of using the KNN Regressor:
    - Training time is very less
    - It can be used for both classification and regression
    - Does not require parametric assumptions

    Disadvantages of using the KNN Regressor:
    - If the number of predictor variables is large, it tends to under-perform

- Prediction will be skewed if range of one predictor dominates the other when using raw data. Standardization is recommended.

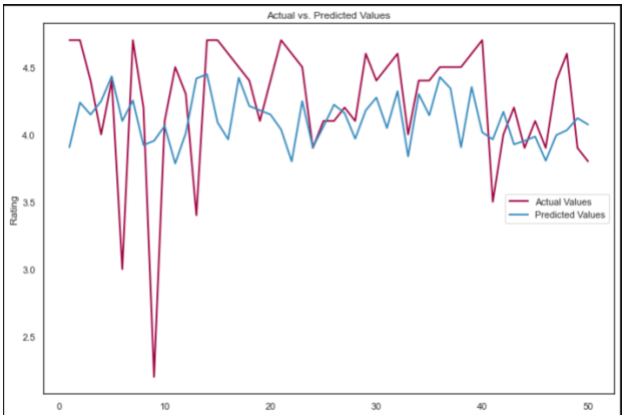
Implementation: We used GridSearchCV to find the best optimal parameters to fine tune this model. We also used Standard Scaler to scale the data because it is recommended for KNN to have all parameters under one scale for better scores. The optimal parameters for this model are:

- Number of Neighbors: 19
- Metric: Manhattan

We got the following results:

- Score: -0.0019
- MSE: 0.28
- RMSE: 0.53

Plot for actual vs predicted in KNN:



3. **Random Forest Regressor:** Large number of decision trees work as estimators that predict values based on rules and parameters. The outcome of all these trees is then combined and averaged to make an accurate prediction.
   Advantages of RF Regressor:
   - Handles missing data automatically
   - Scaling data is not needed, it is a rule-based approach
   - Since many estimator trees make the prediction, it increases accuracy and reduces variance Disadvantages of RF Regressor:
   - Training time is generally high
   - Tends to over-fit the data

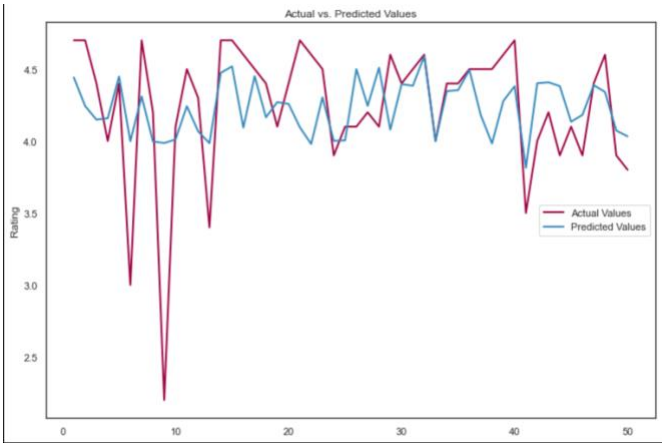
   Implementation: We used GridSearchCV to find the best optimal parameters to fine tune this model. The optimal parameters for this model are:
   - n_estimators=44
   - max_features='auto'

- max_depth=9
- random_state=300

We got the following results:

- Score: 0.146
- MSE: 0.24
- RMSE: 0.49

Plot for actual vs predicted in RF:



4. **Support Vector Machine Regressor:** It sorts the data into two categories that it most likely belongs to. The SVM model then predicts which class the new data point for validation/testing belongs to based on predictors. The idea is to fit the error within a certain threshold – this approximates the best value without a given margin.

   Advantages of SVM Regressor:

   - Can be used for both classification and regression
   - Robust to outliers
   - Data with high dimensions can be fitted

   Disadvantages of SVM Regressor:

   - Training time is high
   - Data needs to be scaled
   - Tends to over-fit the data

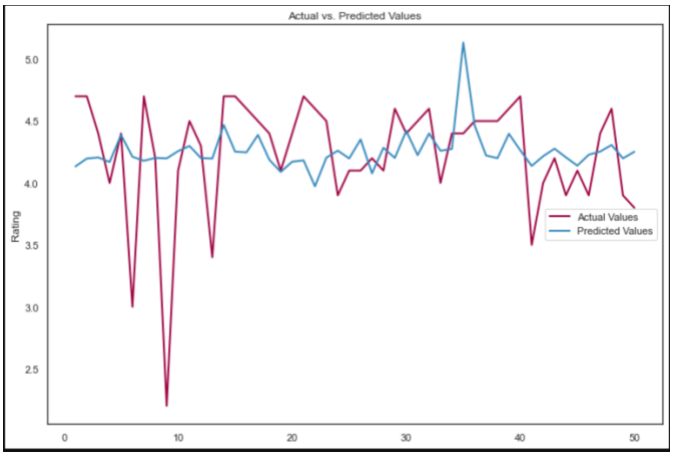
   Implementation: We used GridSearchCV to find the best optimal parameters to fine tune this model. We also used Standard Scaler to scale the data because it is recommended for SVM to have all parameters under one scale for better scores. The optimal parameters for this model are:

   - 'C': 10
   - 'gamma': 1
   - 'kernel': 'rbf'

   We got the following results:

- Score: 0.0032
- MSE: 0.28
- RMSE: 0.53

Plot for actual vs predicted in SVM:



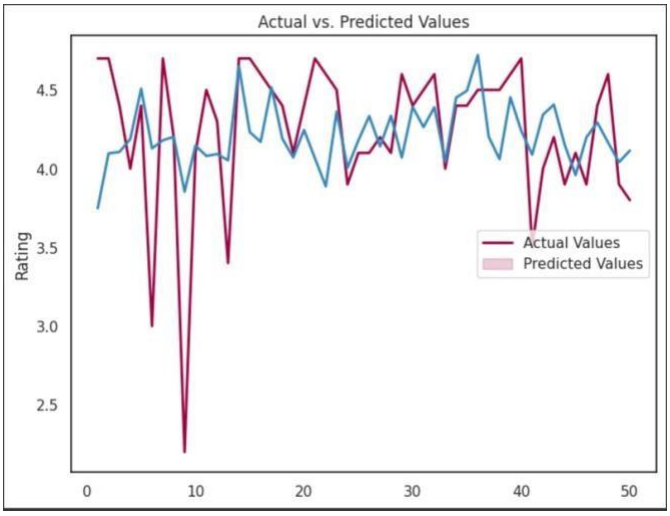
5. **Neural Nets Model Implementation:**

MLP (Multi-layer Perceptron) regression is a versatile and widely used technique for solving regression problems in NLP, healthcare industry, and retail. MLP neural net problems predict continuous numerical values based on input parameters. However, like other neural network models, MLP regression requires careful hyperparameter tuning, regularization techniques, and validation to ensure optimal performance and prevent overfitting.

Scaling data is recommended while training the model in MLP. We ran a custom function to fine tune the hyperparameters for our data. We found that the best parameters for our data are:

- activation = tanh
- hidden_layer_sizes = (50, 100, 50)
- max_iter = 5000
- n_iter_no_change = 200 Scores:
- RMSE – 0.52
- SSE – 0.27

Actual vs Predicted values graph:



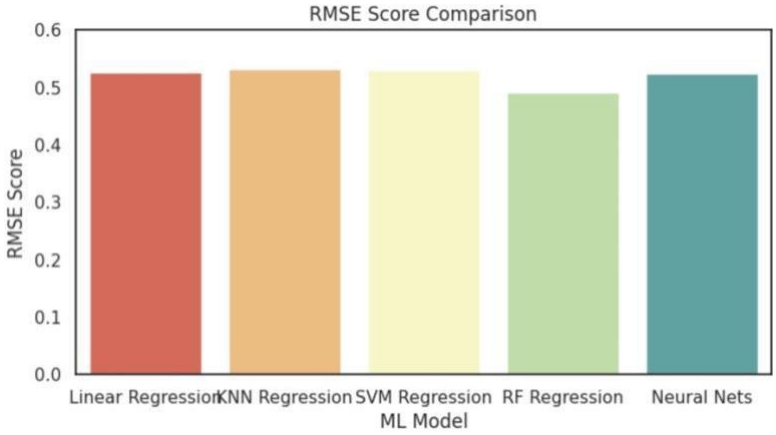
Performance Evaluation:

Since this is a prediction problem where we need to predict the rating of an application based on various factors (or predictors), a regression model would be an ideal approach. We trained our data using the following models:

- Linear Regression
- KNN Regressor
- SVM Regressor
- Random Forest Regressor
- Neural Nets Predictor

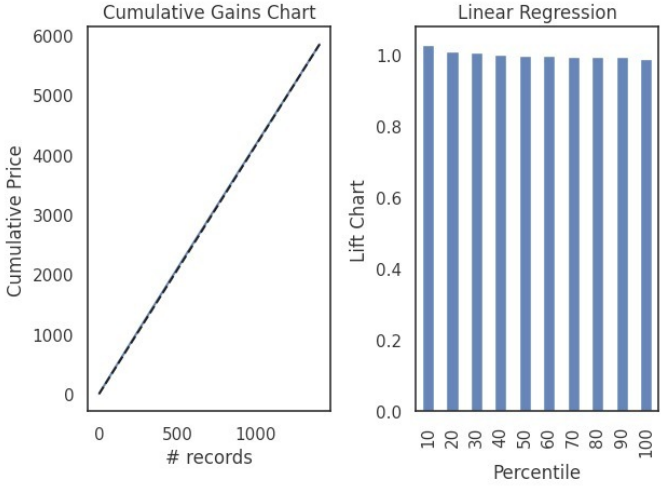
Overall Performance Comparison between all models:

Model	**RMSE**	**MSE**
Linear Regressor	0.53	0.28
KNN Regressor	0.53	0.28
SVM Regressor	0.53	0.28
Random Forest Regressor	0.49	0.24
Neural Networks MLP	0.52	0.27

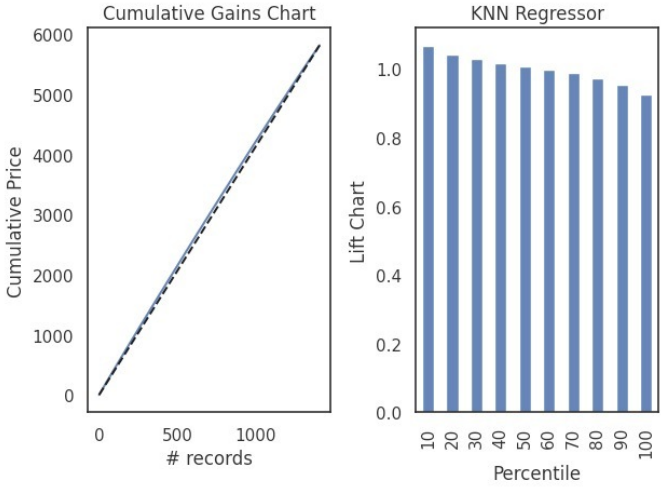


Lift and Gains Charts:

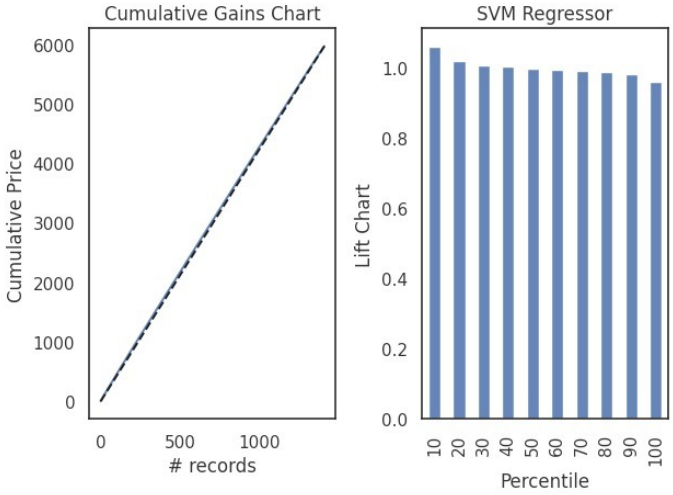
1. Linear Regression



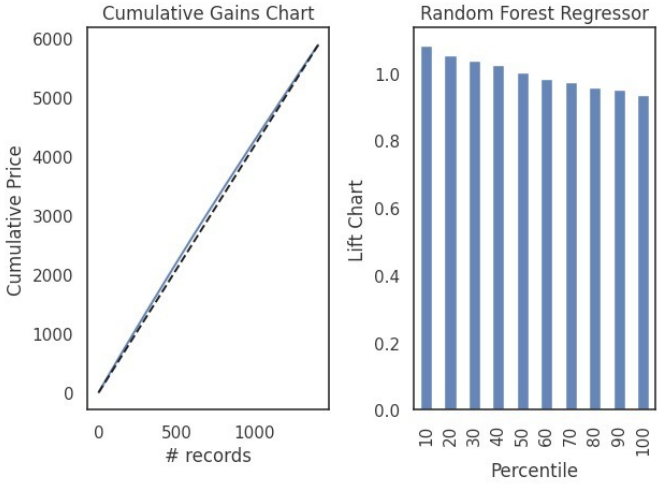
2. KNN Regressor



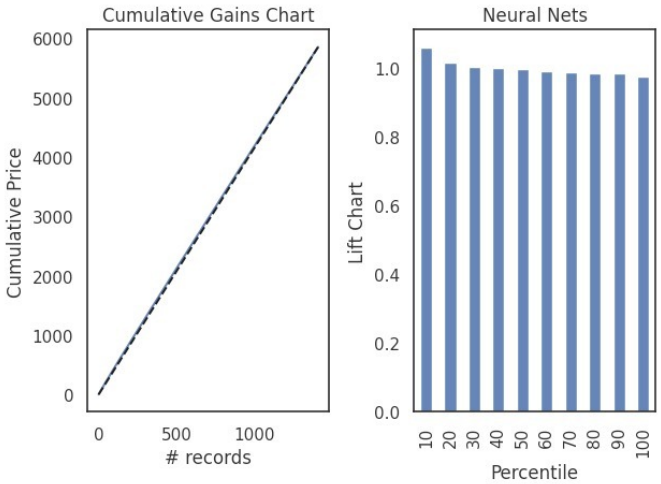
3. SVM Regressor



4. Random Forest Regressor



5. Neural Nets Regressor



Project Results:

The objective of this project was to assess the rating of applications on the Google Play Store based on various predictors such as Category, Price, Content Rating, etc. Identifying the factors that influence app ratings is crucial for app developers to enhance the incentives for highly rated apps and avoid penalties associated with low-rated apps. This knowledge can help developers focus on improving user engagement and ratings. Among the models evaluated, it was found that the ***Random Forest Regressor*** exhibited the lowest RMSE (Root Mean Squared Error) compared to other models, indicating its superior performance. Therefore, the Random Forest Regressor is recommended as the ideal model for detecting the rating of applications on the Google Play Store.

Impact of the Project Outcomes:

App developers can utilize the findings of this project to optimize their app development strategies and improve the overall rating and performance of their apps on the Google Play Store.

Future Scope:

Random Forest has been identified as the ideal model for this prediction task due to its low RMSE value. However, there is still scope for further improving the model's performance by experimenting with the inclusion or exclusion of certain predictors, based on their relevance and correlation with the output variable. Additionally, tweaking parameters such as maximum depth, number of estimators, etc., could also potentially enhance the model's efficiency and accuracy in predicting the ratings of applications.