DSC532 Statistical Learning

Course Professor: Dr Xenia Miscouridou

Playstore Downloads - Classification Problem

April 2024

Team members:
Fivos Lympouras
Constantinos Constantinou
Andreas Papadopoulos
Iosif Pintirishis

Contents

1 Introduction

In todays fast-moving smartphone world, Androids are the mainstream norm, with over 2 billion people using it every month. This success comes from not just its easy-to-use design and high-tech features, but also from the huge number of apps available, both free and paid. Our project is all about understanding how popular an Android app might become. We plan to use machine learning to predict how many downloads an app will have based on different factors.

2 Our Data Set

The data comes from a machine hack's competition. Here you can find the link for the competition, Playstore Dataset.

Attributes description:
Our data set has 16516 rows and 11 columns. We have 10 columns for predictors and one column for the response variable, which is the `Downloads` column. The columns of the dataset are:

- `Offered_By`: The publisher/Organization/Company that develops the application
- `Category`: The category/Genre of the application
- `Rating`: The total ratings received from consumers
- `Reviews`: The total reviews received from consumers
- `Size`: The size of the application with unit
- `Price`: The total price of the application or cost of the in-app purchases
- `Content_Rating`: The content rating for the application
- `Last_Updated_On`: The date at which the application was last updated
- `Release_Version`: The version of the application that is currently being served
- `OS_Version_Required`: The minimum Android OS version required to run the application
- `Downloads`: The approximated range of downloads for the application

3 Problem Formulation

In this project, we formulate the problem as a multiclass classification task aimed at predicting the download frequency categories for Android applications within the Play Store. Leveraging a dataset rich in features such as `Rating` , `Reviews`, `Size`, and more, we plan to employ encoding strategies to handle categorical variables effectively. Our dual approach will initially segment the apps into multiple download ranges before simplifying the classification into a binary framework to identify apps as `Popular` or `Not Popular`. This predictive model endeavors to unravel the complexities influencing an app's popularity, providing valuable insights for app developers and marketers.

4 Data Pre-processing

First things first, we check about any missing values in our dataset and there are not. Then we proceed with preprocessing in some columns.

4.1 Release Version Imputation

In the `Release_Version` column, the value `Varies with Device` is treated as `NA`. Since this value occurs in 9.3% of the cases, which is below the 10% threshold, we have decided to retain this feature. As a result, these values will be replaced with the median. Here is the approach we followed: First, we split the dataset into an 80% training set and a 20% test set. This is done initially because we aim to impute the columns based solely on the training set. Next, we create a new column named `Major_Version`, where we extract the first digit from the `Release_Version`. If it is not numeric, we convert it to `NA`. Then, we fill the `NA` values with the median of the `Major_Version` values, utilizing only the data points from the training set.

4.2 OS Version Required Imputation

Also, the value `Varies with Device` in the column `OS_Version_Required` is treated as NA. This value appeared 8.8% of the times so we decide to keep that feature and replace it with the median. We are following the same method as before by taking the first two digits from the `OS_Version_Required`, to create a new column, `OS_Version_Numeric`.

5 Exploratory Data Analysis

5.1 Summary:

```
 Offered_By         Category            Rating         Reviews              Size
Length:16516       Length:16516       Min.   :1.00   Min.   :       1   Length:16516
Class :character   Class :character   1st Qu.:4.09   1st Qu.:     147   Class :character
Mode  :character   Mode  :character   Median :4.36   Median :    1890   Mode  :character
                                      Mean   :4.26   Mean   :  193197
                                      3rd Qu.:4.58   3rd Qu.:   22669
                                      Max.   :5.00   Max.   :85766433
    Price           Content_Rating     Last_Updated_On    Release_Version    OS_Version_Required
Length:16516       Length:16516       Length:16516       Length:16516       Length:16516
Class :character   Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character


 Downloads
Length:16516
Class :character
Mode  :character
```

The data type of the response variable is character. There are 8 predictors which are also characters and there are 2 predictors (`Rating`, `Reviews`), which are numeric.

5.2 Rating

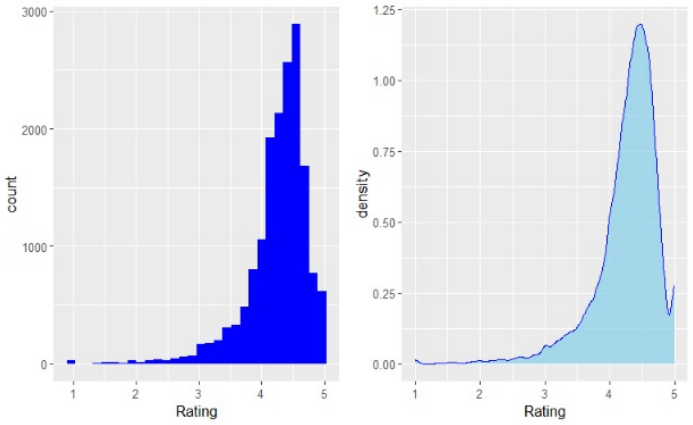

Figure 1: Histogram for Rating and Density plot for Rating.

In figure 1, we have two plots related to Ratings column, which range from 1 to 5. The plot on the left is a histogram that shows the frequency distribution of ratings, with most ratings clustering at the high end near 5. The plot on the right is a kernel density plot that estimates the probability density function of the rating variable. Both plots indicate that the majority of the ratings are high, with a peak at 5, suggesting that either users tend to rate apps favorably or that there is a selection of highly rated apps.

5.3 Reviews

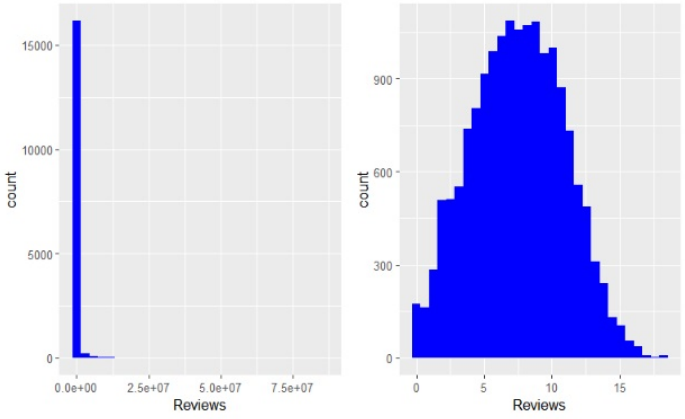

Figure 2: Histogram for Reviews and Log(Reviews)

In the first histogram of figure 2 Almost all data points fall into the first bin, which is around zero.This suggests that a vast majority of the items have a relatively small number of reviews. Then, we plot it with the log(Reviews), in the second histogram, and this histogram provides a more detailed view of the data. There's a long tail to the right, which shows that while most items have fewer reviews, a small number have a much higher count, stretching out the distribution.

5.4 Price

The feature `Price` at first was a character type. The reason is that it included the value `Free`. In order to convert it to numeric, we replaced the value `Free` to 0.

0	73.8342	74.58	78.309	88.7502	93.9708	96.2082	111.1242	111.87
15450	163	1	1	6	1	2	51	1
118.5822	120.0738	126.786	131.2608	148.4142	149.16	164.076	178.992	185.7042
1	1	1	1	153	4	1	1	46
186.45	190.9248	208.0782	216.282	222.9942	223.74	225.2316	229.7064	240.1476
1	1	1	1	184	1	1	1	1
260.2842	264.759	279.675	284.1498	287.133	289.3704	297.5742	298.32	319.9482
20	1	1	1	1	1	85	2	1
334.8642	335.61	342.3222	355.7466	360.9672	369.171	372.1542	409.4442	428.0892
13	1	1	1	1	1	120	11	1
429.5808	446.7342	447.48	469.1082	484.0242	521.3142	558.6042	577.2492	595.8942
1	32	1	1	2	18	2	2	14
633.1842	656.304	670.4742	707.7642	730.1382	742.071	745.0542	801.735	894.2142
1	1	9	2	1	1	29	1	8
968.7942	1043.3742	1098.5634	1117.9542	1192.5342	1267.1142	1341.6942	1416.2742	1490.8542
5	1	1	8	5	1	1	1	3
1789.1742	1863.7542	2087.4942	2236.6542	2385.8142	2534.9742	2833.2942	5592.7542	28339.6542
1	5	1	7	1	1	1	1	1
29085.4542	29831.2542							
1	4							

Table 1: Frequency Table of Price.

In the table 1 we can observe that the most frequent value is 0, meaning that the app is free.

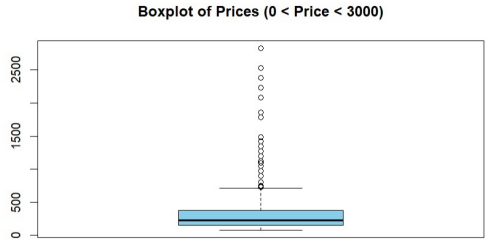

Figure 3: Boxplot of Prices between 0 and 3000

For visualization purposes we filter the prices that are greater than 0 and less than 3000, because there are so many zeros and some extremely large values. From figure 3, we can see that the median line is near the bottom of the box. We can infer that the median price is relatively low compared to the maximum price. There are lot of point beyond the top whisker which are consider as outliers.

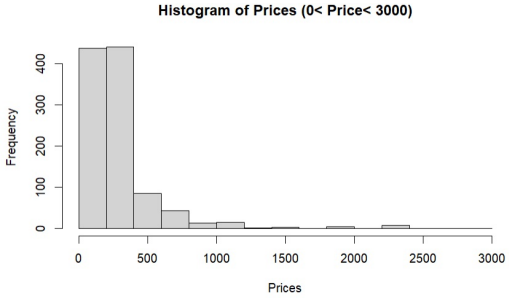

Figure 4: Histogram of Prices between 0 and 3000

Again, for visualization purposes we filter the prices that are greater than 0 and less than 3000. The histogram in figure 4 shows a large concentration of items at the lower price range. The distribution is right skewed.

5.5 Category

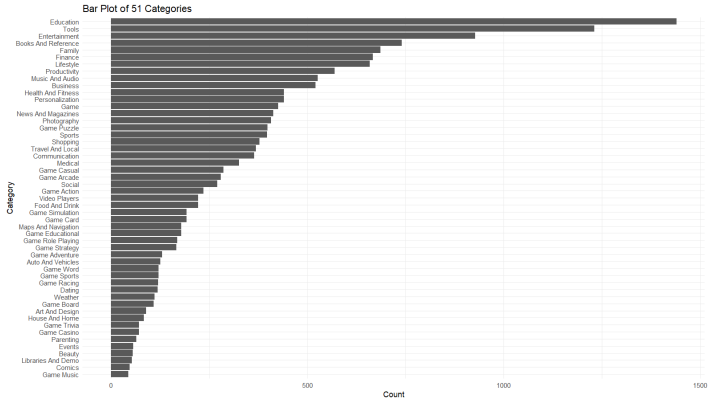

Figure 5: Many different application categories.

From the bar plot in figure 5, we can see we have a lot of distinct categories, 51 in total unique classifications that cater to a wide range of user interests and needs. Among these, notable categories include Music & Audio, Finance, and Business, which serve specific professional and personal functions. Additionally, categories like Medical, Books and Reference, and Tools provide educational and practical utilities. The dataset also covers a variety of entertainment options, with categories such as Game Casual, Game Puzzle, and Game Strategy, highlighting the prevalence of gaming apps. Lifestyle, Health and Fitness, and Shopping represent areas focused on daily activities and personal well-being. The most frequent category being `Education` and the least frequent `Game Music`. We knew that the large number of categories would hinder our progress and cause issues during classification. Our first step to reduce the number of categories was to see which of them could be combined together having reasonable relevance with each other.

In this step we 'handpicked' which categories to merge, for example if the name of the category began with the word `Game` then they were all merged into the `Game` category. Similarly, `Lifestyle`,`Health And Fitness` and `Beauty` were merged into the category `Lifestyle and Personal Care`. We followed the same logic until we were left with 30 categories. Our next step was was to run a K-means Clustering algorithm on our data and create a new column to represent the categories hopefully, successfully reducing the categories even further. The K-means algorithm however requires only numeric features so we made two attempts.

First we run the algorithm on only the numeric data of our dataset, classify our data in clusters, find which category was most frequent in each cluster and finally give all data points of each cluster its relative category.

Our second attempt was to somehow include the `Category` column into the clustering process. We decided to take the reduced categories (30) and transform them as factor so that we could run another clustering algorithm (one that would allow non-numeric features). We made several attempts using DBscan, K-Prototypes clustering, Gower distance and Hierarchical clustering. All of the aforementioned algorithms would return drastically imbalanced categories.

So we reverted to only using the K-means algorithm on the numeric data using taking into consideration the initial 51 categories. We decided to use K-means with $K = 5$, which that means that we create a maximum 5 clusters. Below in figure 6, you can see that we have `Education`, `Finance`, `Lifestyle` and `Tools` and their frequencies.

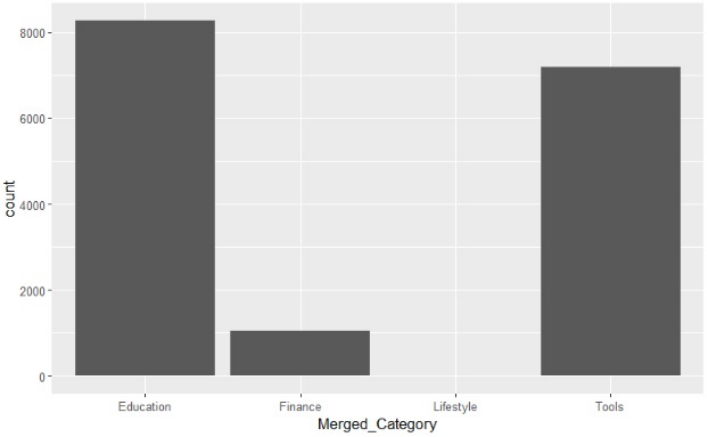

Figure 6: Reduced different application categories

5.6 Offered By

This column pertains to the publisher of the app and contains 320 duplicates. The publisher `"ps_id-1"` is the most frequent, appearing 20 times. Most publishers are listed only once, which suggests that this column may not be useful for fitting a model due to its high cardinality.

5.7 Size

Again, like `OS_Version` and `Release_Version`, in the `Size` column the value `Varies with device` is the most frequent. We calculate then the percentage of the apps whose size varies with the device and after indicating 11.6% of the entries have `Varies with device` as their size, which is more than the cutoff 10%, we consider may excluding this variable from later model building. We convert size to bytes in order to have a numeric format. In the figure 7 the left histogram overlaid with a theoretical exponential density curve in red, illustrates the distribution of app sizes, with the density indicating a decline as size increases, suggesting that a larger proportion of apps have smaller sizes, with few apps reaching larger sizes. The exponential curve fit highlights that the size distribution closely follows an exponential pattern, but using the Kolmogorov-Smirnov test to check if the `Size` follows exponential with rate parameter the $\frac{1}{\mu_{\text{Size}}}$, we conclude that this statement is not correct after occurring a $p - value < 0.05$, so our null hypothesis is rejected for $\alpha = 0.05$ the significance level. On the right, the density plot provides a smoother visualization of the size distribution with a clear peak at the lower end and a long tail extending towards larger sizes.

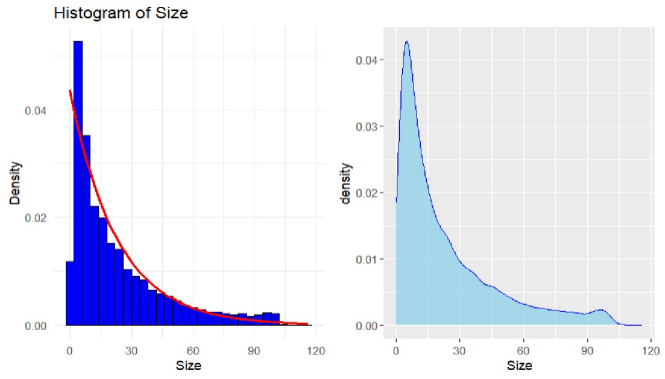

Figure 7: Distribution of App Sizes in the Play Store

5.8 Content Rating

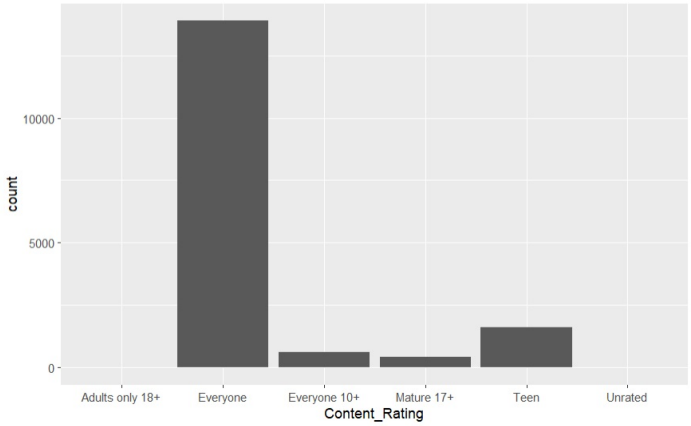

Figure 8: Content Ratings Frequency Bar plot

From the figure 8 we can see that the 'Everyone' category of content rating is the most popular by far.

5.9 Last Updated On

At this column, we change the format of the date in order to extract the year only. Then, we create a bar plot for the occurrences of each year. From the bar plot in figure 9 we can see an increasing trend over the years with the highest counts at the last 2 years.

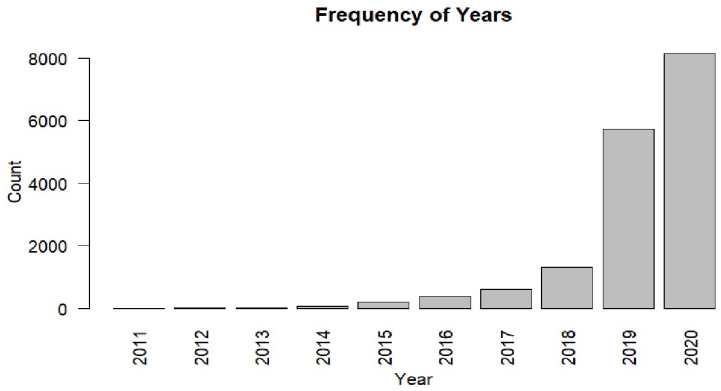

Figure 9: Barplot of frequency of each year.

Finally, we convert this column to data object and we calculate the difference in days between each date in the `Last_Updated_On` column and the oldest date in that column.

5.10 Downloads

We use this feature as our response variable. First we clean the column by removing commas and plus signs, then we convert it to a numeric data type for further analysis. The most popular value is 100000 which indicated approximately 3000 times. Plotting the bar plot (figure 10), we observe that our values may follow a Normal distribution and by performing the lilliefors test we were right. The p-value is relatively big, so we don't have enough evidence to reject the null hypothesis. Thus, the frequency table follows a Normal distribution.

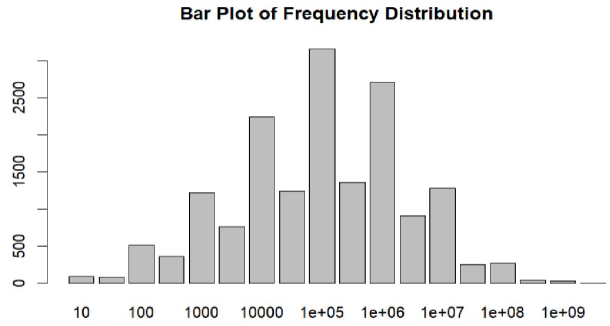

Figure 10: Bar Plot of Frequency Distribution.

In figure 11, on the left we plot the box plot of the original values of `Downloads`. The median is very close to the bottom of the box, which means that at least half of the download counts are relatively low. There are several outliers above the upper whisker. At this point, we logarithm transform the distribution of download counts. Now, the median is closer to the center of the box, which suggest that the transformed data has a symmetric distribution. The range of the data is more compact and the box appears more proportionally spread, indicating less skewness in the log-transformed data. Finally, the outliers are still present but appear less extreme on the logarithmic scale.

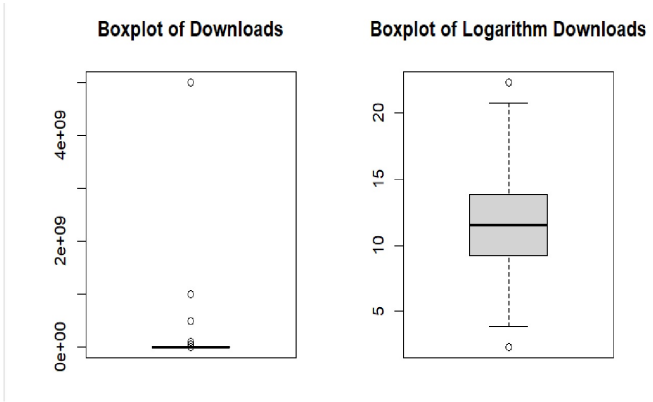

Figure 11: Boxplot of Downloads

In the end, we divide the `Download` data to 8 distinct classes for categorical analysis. After we ensure the numeric format, we compute the quantiles to define class boundaries. Follow, each value is classified into one of these 8 bins, creating a new factor in the dataset, named `Class`. Apps with the lowest downloads are assigned to class 0 and the highest downloads to class 7. The `Class` variable is converted to a factor, suitable for categorical analysis and visualization.

Class	Minimum Downloads	Maximum Downloads
0	10+	1000+
1	5000+	10000+
2	50000+	50000+
3	100000+	100000+
4	500000+	500000+
5	1000000+	1000000+
6	5000000+	5000000+
7	10000000+	5000000000+

Table 2: The separation of Downloads in 8 classes

In the table 2 we can see the separation of the downloads into 8 classes. Class 0 contains the downloads that are between 10 and 4999. Class 1 contains the downloads the downloads from 5000 to 49 999. Class 2 from 50000 to 99 999. Class 3 from 100 000 to 499 999. Class 4 from 500 000 to 999 999. Class 5 from 1 000 000 to 4 999 999. Class 6 from 5 000 000 to 9 999 999 and class 7 from 10 000 000 and above. So now we explained each number in the new column `Class` what means.

5.11 Correlation and Covariation

First of all, we start with a correlation matrix of the numeric columns. Given the figure 11, we can see that `Rating` has no correlation with `Downloads` and also it's not correlated with the other predictors, except `Cluster`. `Reviews` is only correlated with `Downloads`. Lastly, `Last_Updated_On` seems to be slightly correlated with `OS_Version_Numeric`. There are no other correlations between the features. The predictor with the highest correlation with the `Downloads` is the `Reviews`.

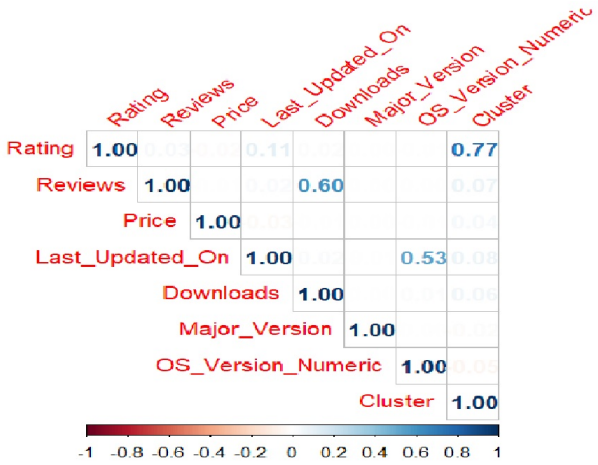

Figure 12: Correlation Matrix.

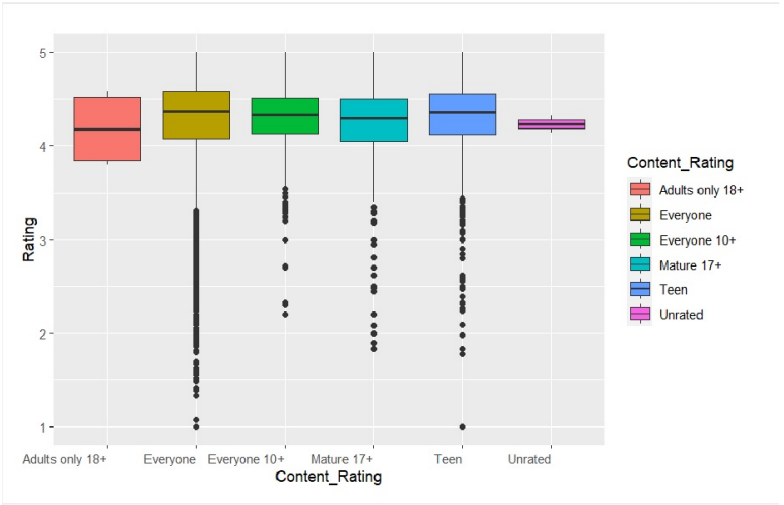

Figure 13: Boxplots of Content Rating - Rating

In the boxplots with the distribution of ratings across different content ratings (figure 13), we can see that the medians are above 4 for all categories, suggesting that apps tend to be rated favorably. All contents except of 'Adults only 18+' and 'unrated' has outliers. All the outliers are below the lower whisker. Categories like 'Everyone' and 'Teen' have more outliers.

In the upcoming phase of our analysis, we conduct a detailed examination to ascertain the degree of correlation between our response variable `Class` and selected attributes within our dataset.

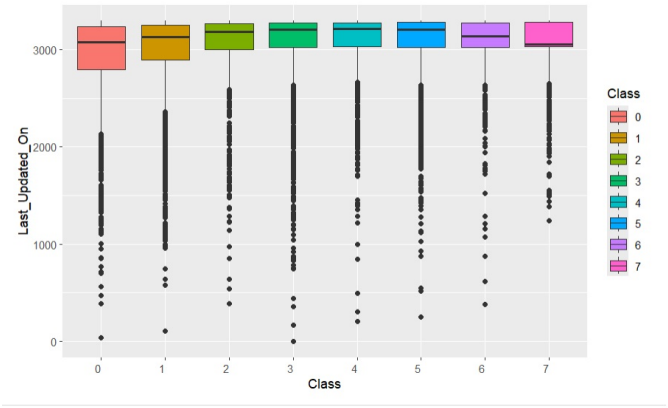

Figure 14: Boxplots of Last Update On with Classes

We examine the association of `Class` with `Last Update On`, plotting the boxplots of each Class (figure 14). Class 0 exhibits the largest interquartile range (IQR), indicating it has the greatest variability in update dates. Additionally, Class 0 has the second lowest median, as well as the lowest extents for both the lower and upper whiskers. This suggests that, generally, entries in this class are associated with the oldest Last Update On dates when compared to other classes, indicating that updates in Class 0 tend to be less recent.

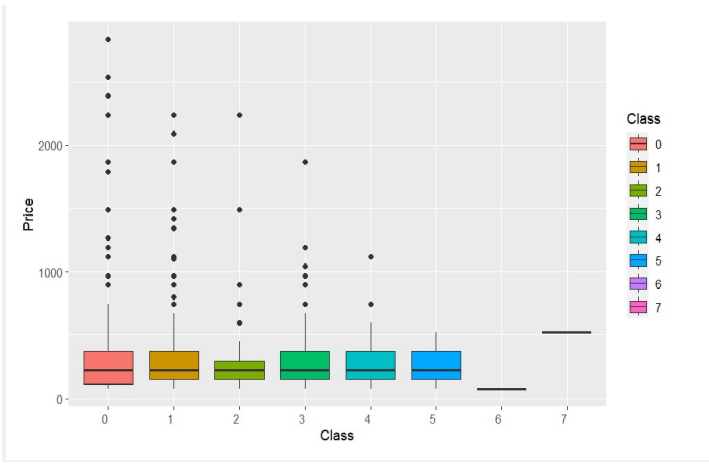

Figure 15: Boxplots of Price with Classes

Furthermore, we investigate the connection between `Price` with `Class`. Initially, we excluded prices that were equal to 0 and those exceeding 3000. We noted that classes 6 and 7 exhibit no variability, as there is only one data point remaining in each of these classes after the filtering. The medians of the other classes are quite similar to one another.

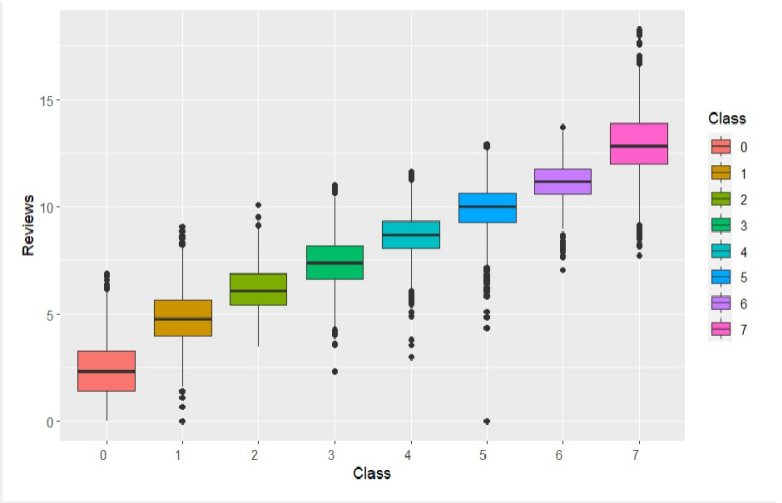

Figure 16: Boxplots of Reviews with Classes

The next feature we will examine (figure 16) for its correlation with `Class` is `Reviews`. It is apparent that classes with higher numbers have more reviews, suggesting a correlation between the number of downloads and the quantity of reviews. This is likely because classes with higher numbers include apps with a greater number of downloads. Class 0 and class 7 display wider boxes, indicating a broader distribution of data relative to the other classes.

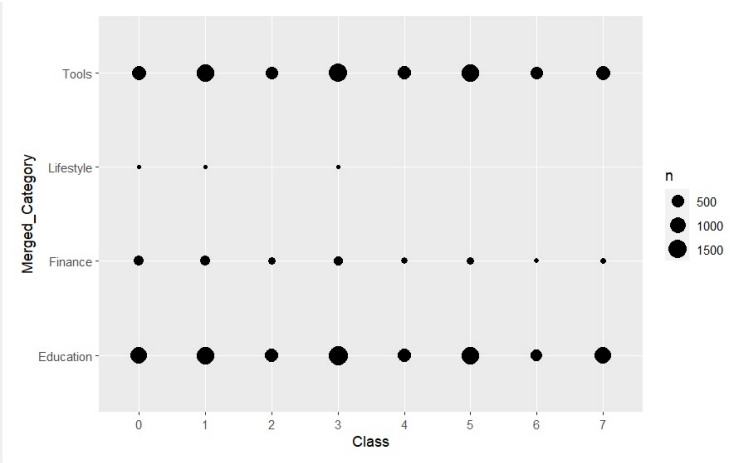

Figure 17: Dot Plot of Merged Category with Classes

At last, we visualize a dot plot for the distribution of app categories from the new feature we created across different classes. It seems that certain classes within specific categories have a larger value, indicated by bigger bubbles. For example, the 'Tools' category has larger bubbles in classes 1,3 and 5, which means that there are more values classified in these classes.

5.12 Encoding after EDA

After doing our Exploratory Data Analysis we realised it is better to proceed as follows. For the column `Content_Rating` feature we saw fit to encode, (one hot encoding), so we can give the column a numeric representation. As we mentioned before the column had 6 distinct values ($k = 6$) so we added

5 ($k-1=5$) columns. We use as a baseline the value `Adults_only_18plus`. Thus, the 5 dummy variables are the rest values as you can see in table 3.

Content_Rating_Everyone	Content_Rating_Everyone_10plus	Content_Rating_Mature_17plus	Content_Rating_Teen	Content_Rating_Unrated
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	1	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	1	0

Table 3: The five new Content Rating columns.

6 Feature Selection

LASSO regression is a technique that combines variable selection and regularization to enhance model interpretability and prevent overfitting. In our approach, LASSO regression is employed as a feature selection method to identify the most significant predictors by applying regularization pressure.

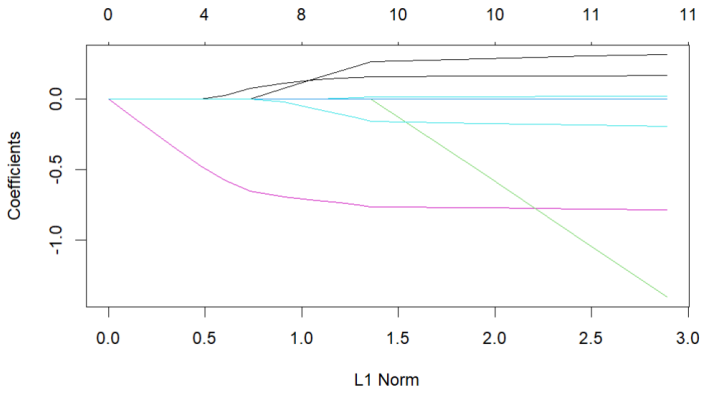

Figure 18: LASSO regression

We can see from the coefficient plot (figure 18) that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. We now perform cross-validation and compute the associated test error (figure **??**).

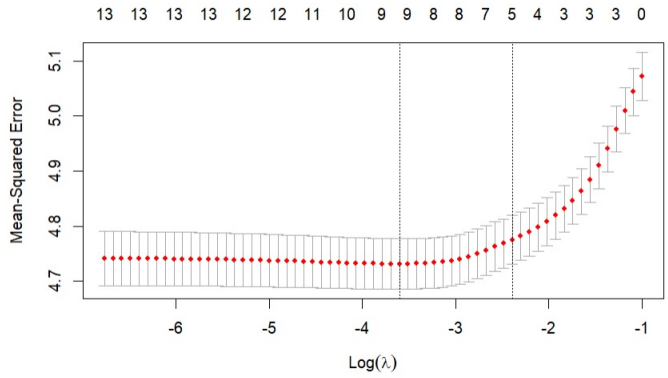

Figure 19: LASSO regression associated test error

After conducting LASSO regression analysis, we have identified several key predictors that are instrumental in determining the outcome. The important features that the LASSO model selected include `Reviews`, `Price`, `Rating`, `Last_Updated_On`, `Cluster`, `Content_Rating_Everyone`, `Content_Rating_Everyone_10plus`, `Major_Version` and `OS_Version_Numeric`. Thus, we use all the predictors since the ones that are not important are dummy variables of the `Content_Rating`.

We did a different feature selection for Random Forest by selecting the 3 most important features that Random Forest suggested. More details about that in the Models section.

7 Models

We implemented a range of machine learning algorithms to construct models, selectively applying these algorithms based on the nature of the response required. Some algorithms were exclusively used for binary responses, others solely for multi-class responses, and a some used to both contexts.
We used as metrics the accuracy, macro precision, macro recall and macro f1 score. By using Macro-averaged metrics, the metrics are calculated independently for each class and then take the average. This means that each class is given equal weight, regardless of its prevalence. This is particularly important in imbalanced datasets. We calculate these metrics as follows: Precision measures the fraction of correct positive predictions among all positive predictions for a class. Recall assesses the fraction of correct positive predictions out of all actual positives for that class. The F1 score is the harmonic mean of precision and recall, balancing the two metrics. These calculations are performed for each class, treating each in turn as the `positive` class while grouping all others as `negative`. The macro-average versions of precision, recall, and the F1 score are then computed by taking the simple average of these metrics across all classes, providing an overall performance measure that treats each class equally.

7.1 Multi class Response

From the frequency plot (table 4), we can observe that our data set is not so balanced. Thus, our best metric to decide which model is better is the F1 score because it is more robust for imbalanced datasets. Accuracy sometimes is not trustworthy when the response variable is not balanced.

Class	Frequency
0	2268
1	3002
2	1243
3	3158
4	1360
5	2707
6	912
7	1866

Table 4: Frequency table of Classes

7.1.1 KNN

Our initial strategy involves applying the K-Nearest Neighbors (KNN) algorithm, testing different values of K. We employ all available predictors for each variation of K.

K = 1

We fit KNN algortihm with K=1 we get these scores:

Accuracy: 0.5266344
Macro Precision: 0.4907112
Macro Recall: 0.4841722
Macro F1 Score: 0.4869367

K = 5

By using K=5 we get higher scores in performance metrics.

Accuracy: 0.5699153
Macro Precision: 0.5097386
Macro Recall: 0.5102703
Macro F1 Score: 0.5073149

K = 10

Employing a K value of 10 we observe even higher scores than before. Actually, there is a slight improvement in all performance metrics compared to K=5.

Accuracy: 0.5892857
Macro Precision: 0.5164124
Macro Recall: 0.5249605
Macro F1 Score: 0.5133999

In general, we observe that as K is increased, the model became simpler so we avoid the overfitting.

7.1.2 Feature Selection using Random Forest

We used Random Forest to calculate the importance of each feature. By checking the Mean Decrease Accuracy, the 5 most important predictors are `Reviews`, `Price`, `Rating`, `Cluster` and `Last_Updated_On`. For the Mean Decrease Gini, the 5 most important predictors are `Reviews`, `Rating`, `Last_Updated_On`, `Major_Version` and `OS_Version_Numeric`.

Reviews seems to be the most important in both metrics and it confirms our correlation matrix's indication that it is highly correlated with `Downloads`. By using all the features the model has the following scores:

Accuracy: 0.6540557
Macro Precision: 0.5370832
Macro Recall: 0.6267137
Macro F1 Score: 0.5133392

7.1.3 Random Forest using the 3 common predictors

We fit again Random Forest with the 3 common important predictors (`Reviews`, `Rating`, `Last_Updated_On`). The scores are:

Accuracy: 0.6026029
Macro Precision: 0.5312718
Macro Recall: 0.5472337
Macro F1 Score: 0.5299987

We observe that F1 increased slightly but Accuracy and Recall decreased significantly. This means that with all predictors captures the actual positives better.

7.2 Binary Response

We converted our target variable into a binary format, labeling it as `Popular` and `Not Popular`. As a result, the dataset now consists of 9,671 entries categorized as `Not Popular` and 6,845 entries classified as `Popular`. From the 500 000 downloads and above are classified as Popular and the rest as Not Popular. From now on, all models will target this binary variable. The threshold that we use is 0.5. It is the default for many classification algorithms, making it suitable for evaluating and comparing different models.

7.2.1 Logistic Regression

Now we fit a logistic regression model with response variable the binary one. As predictors we use the predictors of the Lasso Regression (we use all the predictors since the ones that are not important are dummy variables of the `Content_Rating`).

GLM Prediction	**Actual Test Class**	
	Not Popular	**Popular**
Not Popular	1917	181
Popular	63	1143

Table 5: Confusion Matrix of Logistic regression

We observe really high scores. The metrics for the Confusion Matrix above (table 5) are:

Accuracy: 0.9261501
Macro Precision: 0.9157374
Macro Recall: 0.9307443
Macro F1 Score: 0.921862

7.2.2 Logistic Regression with interaction term

We fitted a logistic regression model using the same predictors as before, but this time we added an interaction term between `Reviews` and `Last_Updated_On`.

GLM Prediction	Actual Test Class	
	Not Popular	Popular
Not Popular	1919	184
Popular	61	1140

Table 6: Confusion Matrix of Logistic regression with interaction term

We can observe from the confusion matrices 5 and 6 that they are similar. Without interaction terms predicts a little bit better the actual Popular class, but with interaction term a little bit better the Not Popular class. We have almost the same scores when we use logistic regression with and without the interaction term between `Reviews` and `Last_Updated_On`.

7.2.3 LDA

By fitting an LDA model, we observed significantly lower accuracy compared to Logistic Regression. While the latter achieved over 0.90 accuracy, the LDA model only reached around 0.64.

LDA Prediction	Actual Test Class	
	Not Popular	Popular
Not Popular	1777	977
Popular	203	347

Table 7: Confusion Matrix of LDA model

From the table 7, we can observe that the majority of the actual Popular class predicted as Not Popular. This is an important reason of the low accuracy compared to Logistic Regression.

7.2.4 Naive Bayes

Using the Naive Bayes algorithm, we achieved higher accuracy than with LDA, but still lower scores than those of Logistic Regression. Specifically, we obtained these scores:

Accuracy: 0.7587772
Macro Precision: 0.7245423
Macro Recall: 0.7657508
Macro F1 Score: 0.7319643

Naive Bayes Prediction	Actual Test Class	
	Not Popular	Popular
Not Popular	1776	593
Popular	204	731

Table 8: Confusion Matrix of the Naive Bayes model

The confusion matrix (table 8) indicates that the Naive Bayes classifier predicts the Not Popular class with a higher success rate than Popular class.

7.2.5 Random Forest- Binary Response - All features

Now, we try to use again Random Forest using all the features as predictors, but with a binary response. It has similar scores with Logistic Regression. All performance metrics are approximately 0.92. Below in table 9 is the confusion matrix.

Random Forest Prediction	Actual Test Class	
	Not Popular	Popular
Not Popular	1862	130
Popular	118	1194

Table 9: Confusion Matrix of the Random Forest model

By checking the Mean Decrease Accuracy, the five most important predictors are `Reviews`, `Price`, `Rating`, `Cluster`, and `Last_Updated_On`. For the Mean Decrease Gini, the five most important predictors are `Reviews`, `Rating`, `Last_Updated_On`, `Major_Version`, and `OS_Version_Numeric`.

Reviews seems to be the most important in both metrics and it confirms our correlation matrix's indication that it is highly correlated with `Downloads`.

7.2.6 Random Forest- Binary Response - 3 common features

We fit again Random Forest with the 3 common important predictors. The 3 features are exactly same as before when we used Random Forest to predict the multiclass response variable `Downloads`. We see that the results are slightly worse than with using all the predictors. All metrics here are approximately 0.91 and the confusion matrix is the following:

Random Forest Prediction	Actual Test Class	
	Not Popular	Popular
Not Popular	1837	131
Popular	143	1193

Table 10: Confusion Matrix of the Random Forest model with 3 predictors

From the confusion matrices in 9 and 10 we can see that they have similar results, but the Random forest with all the predictors tends to predict the Not Popular class a bit better.

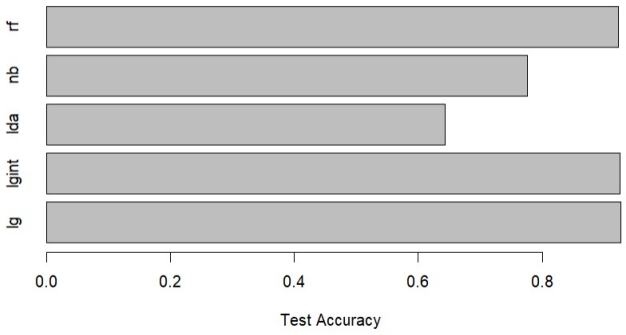

Figure 20: Models' Accuracy

From the Barplot in figure 20 we can conclude that when dealing with a binary target variable, Logistic Regression (both with and without interaction terms) and Random Forest achieve similar, high Accuracy. These methods are followed by Naive Bayes, with LDA performing the lowest.

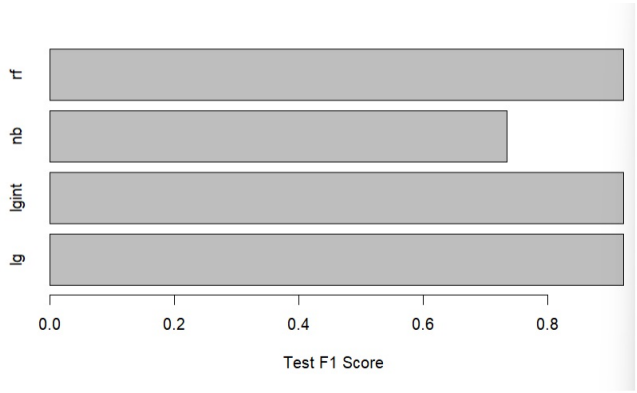

Figure 21: Models' F1 Score

From the Barplot in figure 21 we can conclude that when dealing with a binary target variable, Logistic Regression (both with and without interaction terms) and Random Forest achieve similar, high F1 Score. These methods are followed by Naive Bayes.

8 Conclusion

This project leveraged machine learning to predict Android app popularity on the Playstore, with a focus on multiclass and binary classifications. Key findings revealed `Reviews` as a crucial predictor of app downloads, highlighting the importance of user engagement in determining app success. The logistic regression and Random Forest models outperformed others, demonstrating high accuracy and F1 scores in binary classification scenarios. These results suggest app developers should prioritize enhancing user engagement and feedback mechanisms, to ensure user satisfaction, which in turn will boost app visibility and downloads. Overall, this project underscores the potential of machine learning in identifying factors that contribute to app popularity, offering valuable insights for developers and digital marketers.