Instituto Superior Técnico - UL

Multivariate Analysis

Project

Authors:	*IST ID:*
Diogo Moura	86976
Moustafa Khalil	85660
Rustam Zayanov	98443
Silvestre Pires	85349

Professor Maria do Rosário De Oliveira Silva

3rd of January of 2021

Contents

1 Introduction

1.1 Problem Description

The proportion of mobile devices is rising every year. Android accounts for about 53% of the smartphone market, and iPhone accounts for 43%. Business owners and mobile app developers are the two potential target groups who can benefit from an app market analysis. For example, a business owner might want to expand their business by digitally transforming it, but they need to assure the success of this transition. To achieve this, they should be aware of the factors that determine the popularity of an app. To help the targeted groups, we analysed a dataset [1], which contains more than 7000 app entries from Apple Store.

1.2 Objectives

By this work we aim to discover the variables that influence the app ratings, and use our results to answer the question: "How can we predict the rating of an app given its size, price, genre, and other characteristics?". This answer will help app developers and business owners choose what to take into consideration while designing and developing a mobile app.

2 Preliminary Analysis

The dataset is represented as a single table with 7,197 rows and 17 columns.

2.1 Column analysis

The dataset columns are:

- *X*: An id column with ascending values starting from 1.
- *id*: The app ID from the Apple Store.
- *track_name*: The app name.
- *size_bytes*: App size in bytes.
- *currency*: Currency of the app price. All values are equal to "USD".
- *price*: App price in the Store. E.g. 2.99.
- *rating_count_tot*: Total number of user ratings, for the current and all previous app versions.
- *rating_count_ver*: Number of ratings for the current app version.
- *user_rating*: Average user rating for all versions. Can take values from 0 to 5.
- *user_rating_ver*: Average user rating for the current version. Can take values from 0 to 5.
- *ver*: The current version number of an app. E.g. "4.0.4".
- *cont_rating*: Content rating. I.e. the minimum age of a person eligible to access the app. E.g. "12+".
- *prime_genre*: Primary genre. E.g. "Weather".
- *sup_devices.num*: Number of Apple devices supported by the app.
- *ipadSc_urls.num*: Number of screenshots visible on the app store listing page.
- *lang.num*: Number of languages supported by the app.
- *vpp_lic*: A logical (zero-one) field indicating whether the app supports the Apple VPP (Volume Purchase Program) licensing scheme.

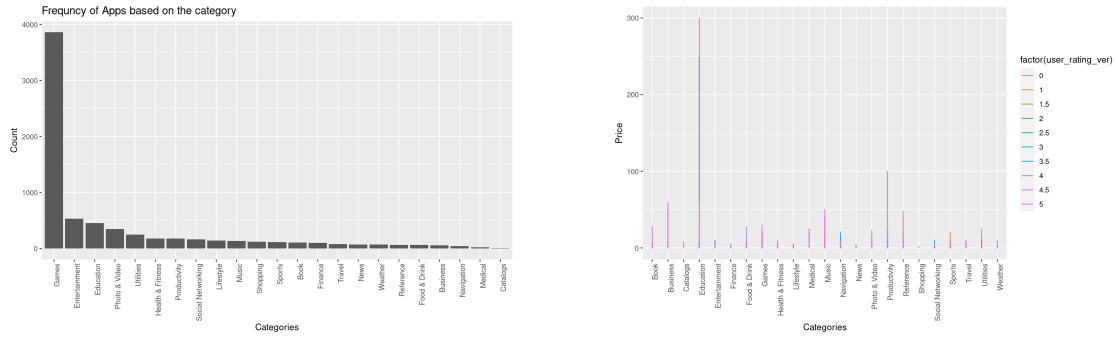


Figure 1: Genre frequency and genre/price/rating distributions

Variable	Mean	SD	Variable	Mean	SD
price	1.80	5.02	rating_count_tot	764.98	5035.08
user_rating	4.15	0.69	cont_rating	6.83	3.94
ipadSc_urls.num	37.39	4.00	lang.num	4.06	1.73

Table 1: Statistical Description of some variables

The variables *user_rating_tot* and *user_rating_ver* both represent the success of an app. However, we consider *user_rating_ver* to be the main label, because it represents the popularity of an app in its present state, and all the other fields describe the app in its present state as well.

Variable *ver* is parsed as a *factor*. We considered converting it into numerical, but concluded that version numbers of different apps can not be compared, because different developers use incompatible numbering schemes.

By default, due to the presence of the "+" sign in the *cont_rating* field, it is parsed by R as *factor*. However, since it is essentially an age, we converted it into a numeric column.

2.2 Row analysis

The dataset has no missing values. However, there are 1,443 apps in the dataset that have no ratings for the current version. Among them, 929 apps have no ratings at all.

2.3 Association between variables

Figure 2 shows the correlations between the variables. There are no significant correlations, however, *price* and *cont_rating* are slightly, but more significantly, correlated with *size_mb*.

2.4 Variable Normality Assessment

As can be seen in figures 2 and 3, the distributions are extremely skewed and don't visually follow the normal distribution. The Shapiro-Wilk test further proved this hypothesis, with P-values being around 1e-90. The box plots also suggest a lot of univariate outliers.

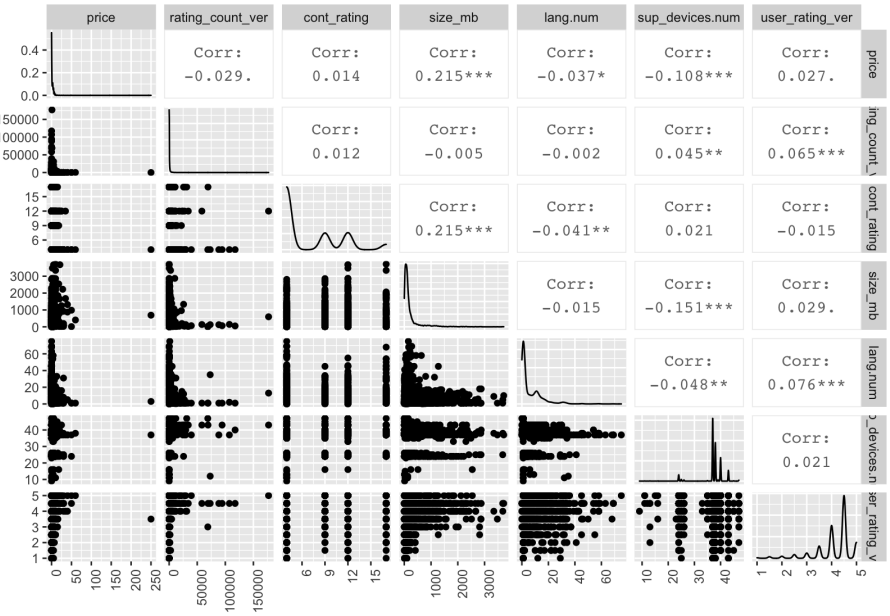


Figure 2: Variable distribution, histograms, and correlation.

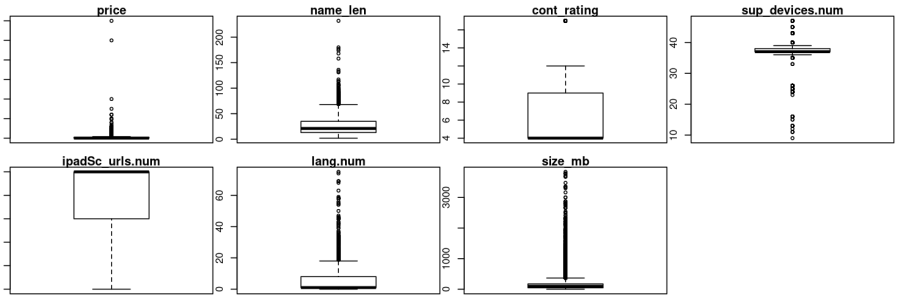


Figure 3: Box plots

3 Preprocessing and Classification

3.1 Data Preprocessing

3.1.1 Variable removal/transformation

We have removed columns *X* and *id* because they contain unique values that would not help in the analysis. We have replaced *track_name* with the computed length of the app name, *name_len*. We also have removed *currency* because it has only a single value. Column *ver* was removed because version numbers can't be compared between different apps. Columns *rating_count_tot* and *rating_count_ver* were removed because they might cause a bias in classification, as high *rating_count* suggests high rating, and we are rather interesting in classifying an app based on its intrinsic properties. Column *user_rating* was removed in order to focus our analysis on its current counterpart, *user_rating_ver*. The column *size_bytes* was converted into *size_mb* to make it more visually appealing. Such rounding is likely to preserve important information. We have converted column *cont_rating* from categorical to quantitative, because it follows a natural order and can be represented numerically. Finally, in order to perform binary classification, we have added a new column *user_rating_is_good*. This variable equals 1 when the user rating is greater

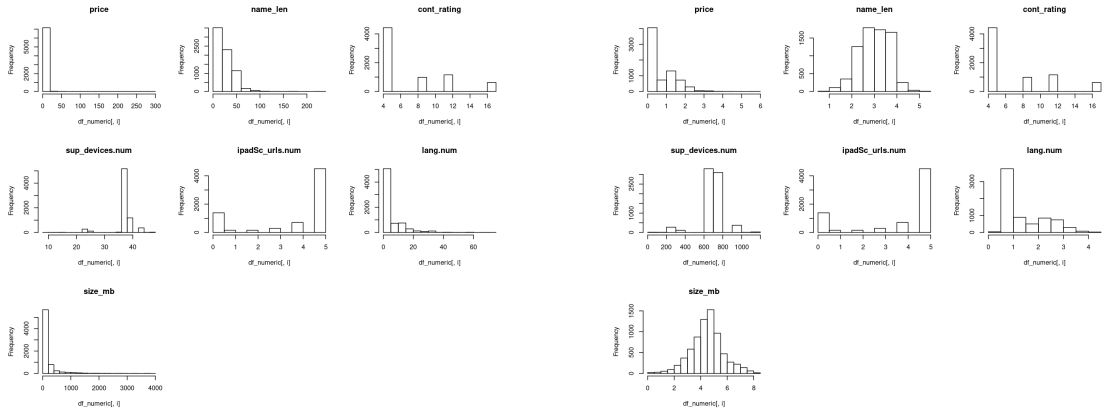


Figure 4: Histograms before (left) and after (right) the Box-Cox transform

or equal to 4, and 0 otherwise.

3.1.2 Marking apps with no ratings as "bad"

There are apps that have no ratings ($rating\_count == 0$). These apps are attributed a *rating* of 0 in the dataset, by default. These apps have no ratings most likely because they are of low quality, and not because they are recent. Our decision was not to remove them, but to associate them with the "bad" class. Our assumption was backed by the data, since this decision increases both the classifier PPV and NPV, showing that this data contains patterns particularly associated to bad apps, which allow the classifier to improve both its accuracy and balanced accuracy. Another relevant factor is that these observations constitute a relevant portion of the dataset, around 13% (929 out of a total of 7197). Keeping the observations allows the classifier to generalize better and less prone to overfitting.

3.2 Variable Space Transformation

In order to make variable distributions normal, we applied Box-Cox transformation on every numerical variable. Variables *content_rating* and *ipadSc_urls.num* did not change shape, and we decided to keep them intact. Log transformation was applied on *price*, *name_len*, *lang.num*, and *size_mb*. The variable *sup_devices_num* was squared. The resulting variables have the histograms showing in figure 4.

The distributions began to look more normal, but the Shapiro-Wilk test still returns P-values of magnitudes around 1e-20. We used these transformed variables for the rest of the analysis.

3.2.1 PCA

We have applied both Classical and Robust approaches in finding principal components. In both approaches, the scaling was necessary due to *sup_devices.num* having variance hundred times higher than the rest of the fields.

For the classic PCA (figure 5), the first principal component tells us how "heavy-weight" an app is. That is, having a high score on PC1 means that the app is expensive, big in size, and has many screenshots. The PC2 is mostly a combination of content rating and size, which is hard to explain from the real world point of view. The rest of the PCs are not very interpretable either.

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7
price	0.385	0.135	0.254	0.631	0.110	0.593	0.082
name_len	0.235	0.375	0.468	-0.228	0.654	-0.322	0.036
cont_rating	-0.113	-0.636	0.556	-0.274	-0.002	0.217	0.390
sup_devices.num	-0.283	-0.287	-0.450	0.018	0.740	0.291	-0.054
ipadSc_urls.num	0.544	-0.219	-0.398	0.095	0.053	-0.328	0.615
lang.num	0.343	0.261	-0.215	-0.680	-0.092	0.543	0.053
size_mb	0.539	-0.488	0.033	-0.062	0.029	-0.099	-0.675

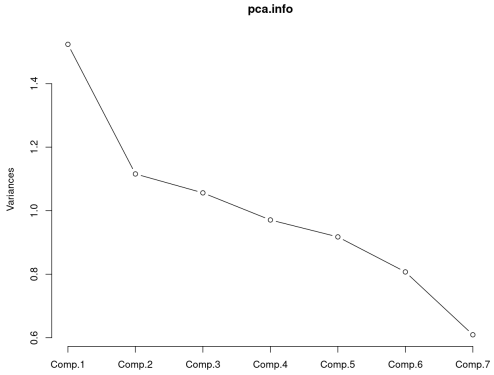


Figure 5: Classic PCA Results

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
price	-0.246	0.157	0.066	0.441	-0.845	-0.029	-0.008
name_len	-0.158	0.497	-0.699	0.392	0.289	0.000	-0.036
cont_rating	0.295	-0.595	-0.632	-0.059	-0.267	-0.291	-0.011
sup_devices.num	-0.013	-0.052	0.051	0.039	0.028	-0.012	-0.996
ipadSc_urls.num	-0.659	-0.341	0.135	0.215	0.270	-0.556	0.056
lang.num	-0.466	0.251	-0.244	-0.771	-0.236	-0.079	-0.056
size_mb	-0.420	-0.438	-0.163	0.071	0.038	0.773	0.015

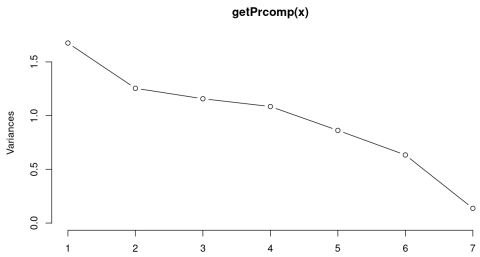


Figure 6: Robust PCA Results

In the resulting decomposition only 3 eigenvalues are higher than the average. The scree plot has 2 knees, suggesting either 1 or 6 components. We decided to keep the first k components that preserve 90% of variance. In our case it is 6 PCs.

For the Robust PCA (figure 6), we used the function *PcaHubert* that looks for the data subset that has the minimum covariance determinant (MCD) and builds PCs based on that subset. Among the resulting eigenvalues, the first 4 are above average. The scree plot is similar to that of the classic PCA, and suggests either 1 or 6 PCs to retain. The first 5 PCs retain 88% percent of the variance, so we retained 5. The first PC describes how much of content the app has. A low score signifies an app of big size, with many screenshots an supported languages. The second PC is a contrast between name length, content rating, and size, which is hard to interpret, just as the rest of the PCs.

The rest of the analysis was performed with robust PCs.

3.2.2 One-hot Encoding and MDS

In order to apply numeric methods to the 2 categorical variables that we have, we used one-hot encoding, then computed Hamming Distance, and then applied *Multidimensional Scaling* (MDS). The number of above-average eigenvalues is 23. The knee on the scree plot (figure 7) is at 4. 80% of variance is described by 9 PCs. We chose to keep 9.

3.2.3 Outlier detection

We have tried 3 techniques for outlier detection:

1. Detection based on Score and Orthogonal distances over the 6 classic PCs: With the help of the *pcaDiagplot* function, we computed SD and OD based on the classic PCA

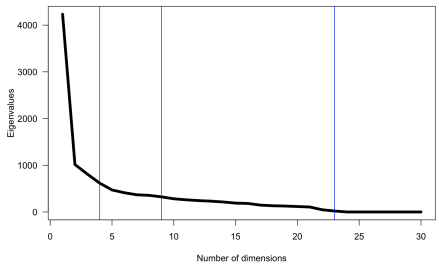


Figure 7: MDS eigenvalues

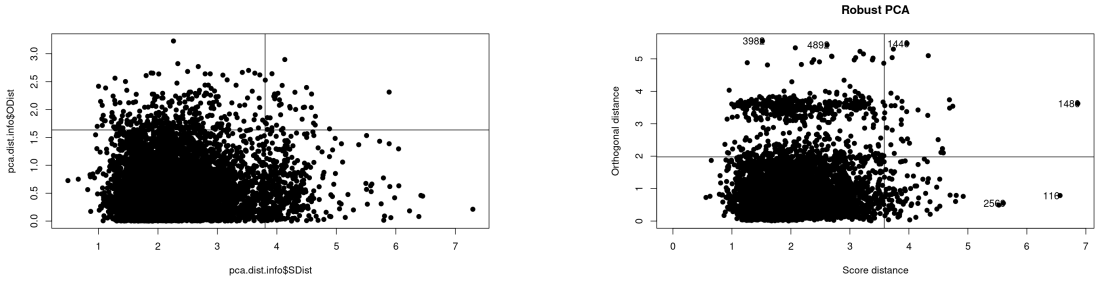


Figure 8: Outlier plots for classic (left) and robust PCA (right)

components. This method marked 741 observations as outliers (figure 8). However, since SD is the Mahalanobis distance, and our variables are not normal, we did not find these indications very reliable.

2. SD and OD over the 5 robust PCs. The horizontal group in the OD zone (figure 8) consists of apps with low or high number of supported devices. Since *sup_ devices.num* is not represented in the first 5 Robust PCs, any deviation from the mean will result in a high OD distance. This method has marked 726 apps as outliers, with 552 apps being outliers both for classic and robust approaches.

3. Univariate detection: for every variable we have obtained a list of outliers identified by the *boxplot* function with the standard *range* value of 1.5. After checking all variables separately, we got 278 observations that have outliers in 2 or more columns.

3.3 Classification

First the data classification with no dimensionality reduction was used with one-hot encoding for the categorical variables(no PCA and no MDS). The results are presented in table 2.

Then the dataset with robust PCA and MDS variable transformation was used. Only a portion of the original variability is represented. The results are presented on table 3. As expected, classification isn't affected very much by the use of PCA and MDS, since the dimensionality is reduced but still +80% of the variability in the original data is still explained by the transformed data. Despite this fact some methods perform worse with MDS and PCA, such as KNN.

Ensemble decisions methods were used on the Robust PCA + MDS dataset. The used methods are majority voting, average and weighted average. Even though Random forest

	Accuracy	Balanced Acc	Sensitivity	Specificity	Avg F1-score	PPV	NPV
LDA	0.701	0.664	0.837	0.491	0.668	0.718	0.660
RDA	0.597	0.501	0.945	0.057	0.420	0.608	0.400
SVM	0.699	0.659	0.842	0.479	0.663	0.714	0.661
Neural Network	0.694	0.659	0.819	0.499	0.663	0.717	0.640
Random Forest	0.689	0.661	0.789	0.534	0.664	0.724	0.620
KNN	0.675	0.638	0.809	0.468	0.641	0.702	0.612

Table 2: Classification metrics into bad/good app for the dataset with no PCA for numerical variables and with One-Hot encoding for categorical

	Accuracy	Balanced Acc	Sensitivity	Specificity	Avg F1-score	PPV	NPV
LDA	0.693	0.652	0.842	0.462	0.655	0.708	0.654
RDA	0.701	0.659	0.852	0.465	0.662	0.712	0.670
SVM	0.701	0.660	0.852	0.469	0.664	0.713	0.671
Neural Network	0.693	0.651	0.845	0.457	0.654	0.707	0.656
Random Forest	0.683	0.654	0.788	0.521	0.657	0.718	0.613
KNN	0.693	0.662	0.807	0.516	0.665	0.721	0.633

Table 3: Classification metrics into bad/good app for the dataset with Robust PCA for numerical variables and with MDS for categorical

is also an ensemble method, it is an ensemble of only decision trees, so we decided not to include it on table 4, which includes ensembles of different types of classsifiers.

It can be verified that the ensemble methods have generally much better performance than other types of methods. This can be justified by the high number of classifiers used in the ensemble (7), which minimizes the errors committed by each classifier. This introduces robustness.

Even tough the consensus might hypothetically not always yield better results the ensemble provides a way of averaging the classifier scores, so we will arrive to a more robust score The number of situations where the classifier performs in a very bad way is reduced, the performance becomes more independent of the new data that might need to be classified. As a downside is the fact that the ensemblers take slightly longer to train/classify due to the computation of several models/scores.

4 Clustering

4.1 Choosing data and techniques

We have performed 19 experiments in total. We tried clustering on either the numeric variables or their robust PCA scores. We also tried adding categorical variables or their MDS components. We also tried removing the outliers identified by the Robust SD-OD

	Accuracy	Balanced Acc	Sensitivity	Specificity	Avg F1-score	PPV	NPV
Average	0.713	0.672	0.865	0.478	0.676	0.720	0.696
Weighted Average	0.711	0.673	0.852	0.494	0.677	0.723	0.682
Majority	0.707	0.665	0.861	0.468	0.669	0.715	0.685

Table 4: Ensemble decisions metrics with the dataset with no PCA and with One-hot Encoding

#	Dataset	Outliers	Method	Bal.Avg	k	Silh	CH
1	Numeric+Cat	No	Single	.5	2	.33	3
2	Numeric+Cat	No	Complete	.5	2	.32	58
3	Numeric+Cat	No	PAM	.6	2	.34	2524
4	Numeric	No	Complete	.5	3	.7	562
5	Numeric	No	PAM	.6	8	.17	977
6	Rob.PCA+Cat	No	Complete	.5	2	.32	58
7	Rob.PCA+Cat	No	Average	.5	2	.32	58
8	Rob.PCA+Cat	No	Ward	.5	2	.06	393
9	Rob.PCA+Cat	No	PAM	.59	4	.03	353
10	Rob.PCA+MDS	No	Complete	.59	10	.1	488
11	Rob.PCA+MDS	No	PAM	.63	4	.18	1132
12	Rob.PCA	No	Complete	.6	4	.13	969
13	Rob.PCA	No	PAM	.61	10	.24	1354
14	Rob.PCA+Cat	Yes	Complete	.59	10	.41	794
15	Rob.PCA+Cat	Yes	PAM	.58	4	.33	1344
16	Rob.PCA+MDS	Yes	Complete	.5	2	.19	140
17	Rob.PCA+MDS	Yes	PAM	.62	6	.17	1130
18	Rob.PCA	Yes	Complete	.5	2	.32	94
19	Rob.PCA	Yes	PAM	.6	9	.23	1497

Table 5: Performance of various clustering techniques

scores.

We used Gower distance in the experiments that involved categorical variables. Otherwise, we used Euclidean distance. When Euclidean distance was used on the numeric variables only, the variables were scaled before clustering.

For the clustering techniques, we focused mostly on complete linkage and partitioning around medoids. We chose Silhouette coefficient as a criterion for choosing the number of clusters in each experiment. We also computed CH score and balanced accuracy (by treating each cluster as either "cluster of bad apps" or "cluster of good apps" and comparing with *user_rating_is_good*). The obtained results are in table 5.

Our first attempt was using Gower distance over all 9 fields (7 numeric and 2 categorical). We realized that the resulting distances and groups are dominated by the categorical variables. For example, this can be seen in the dendrogram of the single linkage in figure 9. The top horizontal line is the division into categories. The vertical lines above are the outliers.

This dominance can be explained by the fact that numeric variables are not evenly distributed, resulting in smaller pairwise distances compared to the distances generated by the categorical variables. In order to mitigate that, we considered omitting categorical variables or using MDS instead.

We chose clustering number 3 (numeric and categorical variables without outliers, Gower distance, clustered by PAM) for the further analysis. This clustering has a good degree of separating good and bad apps (as measured by the balanced average of .60), a good average Silhouette score of .34, and the highest CH index of 2524 among all experiments.

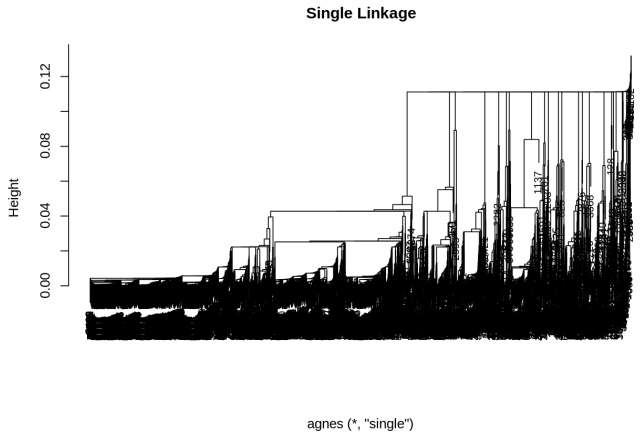


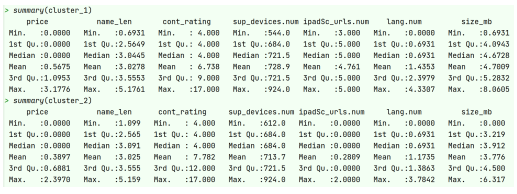
Figure 9: Single Linkage over Gower distance is dominated by the *prime_genre* category

4.2 Cluster interpretation

Table 6 is the contingency table for the chosen clustering. Cluster 1 has more good apps (ratio 2.14), and cluster 2 has more bad apps (ratio 1.5). As we can see in the scatterplot grid (figure 10), the clusters are perfectly divided by *ipadSc_urls.num*. Cluster summaries additionally show that cluster 1 has significantly higher app size.

Cluster	App is good	Not good
1	3371	1570
2	611	916

Table 6: Contingency table of the Partition Around Medoids clustering



```
> summary(cluster_1)
     price          name_len       cont_rating     sup_devices.num ipadSc_urls.num    lang.num         size_mb
Min.   :0.0000   Min.   :0.6931   Min.   : 4.000   Min.   :544.0   Min.   :3.000   Min.   :0.0000   Min.   :8.6931
1st Qu.:0.0000   1st Qu.:2.5649   1st Qu.: 4.000   1st Qu.:684.0   1st Qu.:5.000   1st Qu.:0.6931   1st Qu.:4.0943
Median :0.0000   Median :3.0445   Median : 4.000   Median :721.5   Median :5.000   Median :0.6931   Median :4.6728
Mean   :0.5675   Mean   :3.0278   Mean   : 6.738   Mean   :728.9   Mean   :4.761   Mean   :1.4353   Mean   :4.7009
3rd Qu.:1.0953   3rd Qu.:3.5553   3rd Qu.: 9.000   3rd Qu.:721.5   3rd Qu.:5.000   3rd Qu.:2.3979   3rd Qu.:5.2832
Max.   :3.1776   Max.   :5.1761   Max.   :17.000   Max.   :924.0   Max.   :5.000   Max.   :4.3307   Max.   :8.0605
> summary(cluster_2)
     price          name_len       cont_rating     sup_devices.num ipadSc_urls.num    lang.num         size_mb
Min.   :0.0000   Min.   :1.099   Min.   : 4.000   Min.   :612.0   Min.   :0.0000   Min.   :0.0000   Min.   :8.000
1st Qu.:0.0000   1st Qu.:2.565   1st Qu.: 4.000   1st Qu.:684.0   1st Qu.:0.0000   1st Qu.:0.6931   1st Qu.:3.219
Median :0.0000   Median :3.091   Median : 4.000   Median :684.0   Median :0.0000   Median :0.6931   Median :3.912
Mean   :0.3897   Mean   :3.025   Mean   : 7.782   Mean   :713.7   Mean   :0.2809   Mean   :1.1735   Mean   :3.776
3rd Qu.:0.6881   3rd Qu.:3.555   3rd Qu.:12.000   3rd Qu.:721.5   3rd Qu.:0.0000   3rd Qu.:1.3863   3rd Qu.:4.580
Max.   :2.3970   Max.   :5.159   Max.   :17.000   Max.   :924.0   Max.   :2.0000   Max.   :3.7842   Max.   :6.317
```

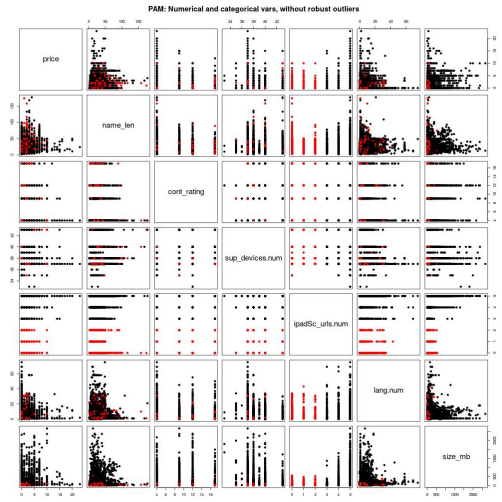


Figure 10: Cluster summaries and scatterplots

5 Classification into Clusters

The classification task is repeated but now with the clustering results as the response variable. As can be seen in table 7, the results are much better than for the normal classifi-

cation task. This is expected since the observations seem to be divided by a single variable (number of screenshots). In table 8 can be observed the classification for the classifier which obtained the best performance, the neural network. Near perfect classification is obtained.

Since clustering yields only 60% balanced accuracy, classification over the original data would be more preferable. Since the clustering is very focused on the screenshot count, it would be hard to find use cases where this clustering would be preferable.

	Accuracy	Balanced Accuracy	F1-score
LDA	0.971	0.965	0.961
RDA	0.971	0.965	0.961
SVM	0.981	0.972	0.974
Neural Networks	0.989	0.989	0.985
Random Forest	0.971	0.959	0.961
KNN	0.982	0.975	0.975

Table 7: Classification into clusters results for different classifiers

	Reference 1	Reference 2
Predicted 1	1451	6
Predicted 2	15	469

Table 8: Confusion Matrix for the Neural network classifier

6 Conclusion and Suggestions

The variable distributions of the dataset are not normal, so it was very important to apply the Box-Cox transformation to achieve more reliable results. The variables are not strongly correlated, therefore dimensionality could not be greatly reduced. The best classification results were obtained by the ensemble methods, with balanced accuracy of 67%. Among all variables, the number of screenshots seems to play an important role both in clustering and in predicting the app popularity. As a future work, we recommend considering dataset balancing techniques, like SMOT or oversampling, and to evaluate the potential impact on clustering and classification due to the high imbalance of it.

References

[1] Ramanathan. *Mobile App Store (7200 apps)*. URL: https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps.