Final Project - Statistical Modeling and Inference
Predicting Apps Popularity with image Data

Barcelona School of Economics
Dec - 2021

Ferreira Pose, Agustin
Gil Vergara, Richard

Prof. Mesters, Geert
Prof. Rossell David

1. Introduction

With the boom of online marketplaces customers face an overwhelming amount of information when choosing the best products for them, several companies are analyzing these customer choices and which are the main drivers for these decisions.

This project aims to develop an algorithm that takes as input the information from the product in the way it is presented online and tries to forecast its popularity.

Particularly, this project will be focused on predicting Apps popularity. For this purpose, features from different mobile applications will be extracted from Google Play Store including its price, size, category, release date, update date, content rating, in-app purchases and presence of advertising inside the app. More importantly, the main focus of the project is to determine whether the main shown image is relevant to explain the popularity of the application measured as number of Installs. An app will be considered popular when its number of installations is in the top 50 percentile.

Specifically, a PCA algorithm and an advanced Neural Network will be used to preprocess and reduce the dimensionality of the images, a K-means algorithm will be used to cluster similar images and a logistic regression penalized with both L1 and L2 methods will be trained to determine the variables that 'truly matter' and to classify the popularity of the apps.

This approach could be very useful to help determine the design of both the marketplace itself but mainly deciding how a product should be presented to maximize its popularity.

2. Related Work

Even though no papers or related publications could be found addressing the specific problem of predicting app popularity based on its image, there is a huge relevant literature approaching the problem of estimating image popularity in general and, particularly, such examples are commonly developed under the Social Networks environment. Interesting approaches on this topic can be seen at Almgren and Lee (2016) or at a recently published article by Dutta and Barbhuiya (2021) where a very similar approach to the one handled in this present work is done in order to predict image popularity in a 30-days time frame. It is important to note that some ideas of the following approach to be stated in this project are derived from this article.

Many examples about applying PCA over images, and particularly, applying it over vectors and matrices, as done in the present paper are available where Yang and Yang (2001) propose an interesting approach for handling images in that sense.

Alternatively, not following a strictly academic approach, one could consider Kaggle where many relevant code examples, analysis and datasets used to predict app popularity based on structured data are available.

3. Dataset

This project uses both structured and unstructured data in the form of images to predict the popularity of an app. Since treatments for the two types of data are different the processing stages were handled independently and therefore will be explained separately.

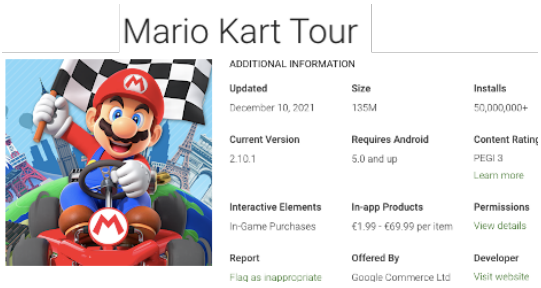
a. Structured Data

The structured data was obtained from a Kaggle competition https://www.kaggle.com/gauthamp10/google-playstore-apps, this dataset describe features such as app name, category, rating, number of installs, price, size, developer, time released, last update, content rating, ad supported and in app purchases. For the purpose of this analysis and taking into account the computational constraints to process several models but also for web scraping the images, 1.425 apps were randomly chosen taking into account in first place filters of extremely bad ratings to reduce the noise of test and death applications.

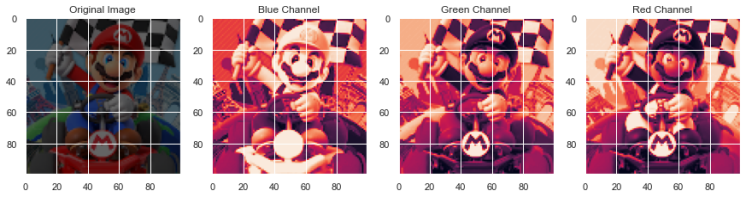
The preprocessing used was one hot encoding over the categorical variables to obtain a dummy for each category, calculating time features such as months since the last update or months since the release from the dates features, imputing missing with the media and scaling the variables with the media and the standard deviation. Since the approach to test the models is cross validation and Bayesian information criteria the data set was not splitted in train and test.

b. Image Data

The image of each application was obtained through a web scraping script built in python that goes to the Google Play Store and with the name of the app download the image and which was lately processed following two different approaches.



Initially, (Pixel Method) images were imported with a resolution of 100x100 pixels, coded into RGB format and transformed into a 100x100 matrix per image, where each position is a 3 elements RGB array representing each pixel taking values from 0 to 100 representing the proportion of red,green and blue at each pixel. Each image was then decomposed into its 3 channels (red, green and blue) resulting in three 100x100 matrices per image. As a result, a 1425x3 matrix was obtained where 1425 and 3 represent the number of observations and each RGB channel respectively.



Alternatively, (Context Method) instead of processing the images into individual pixels, a VGG16 convolutional neural network without its last two layers was used to extract features from each image to obtain more meaningful representations that will, ideally, add a bit of context for each partition. As a result, a 1425x4096 matrix was obtained where 1425 represents the amount of images and 4096 the number of obtained features.

Finally, a PCA to first reduce the dimensionality and a clustering approach with the K-Means algorithm was used to group the data obtained from the two approaches. This resulted in 18 clusters

selected with the Elbow Method Heuristic for the first method and only 10 for the second one. Respectively a 1425x18 and 1425x10 matrix was obtained where 18 and 10 represent the dummy variables for each cluster where 1 means the image was assigned to the cluster and 0 it was not.

4. Unsupervised

In this project unsupervised learning techniques were used to preprocess the data and reduce its dimensionality. Firstly, a PCA algorithm that maximizes the explained variance was used to reduce the dimensionality for the image features and, after that, a K-means clustering algorithm was applied to cluster similar images into K clusters that were used as input features for the classification algorithms.

a. Methods

PCA-Maximum Expected Variance

To reduce the dimensionality of the images both reducing the storage space needed and the processing time, a PCA that maximizes the Explained Variance as implemented in the Python Package Scikit-Learn was used which is basically the simplest method for this algorithm where the principal components are chosen as the eigenvector associated with the nth relevant eigenvalue of the treated image.
In this project, since the objective was to reduce the dimensionality of the data, the PCA was applied differently for each of the preprocessing approaches previously described (Pixel and Context Method). For the Pixel Method, a PCA was applied independently image by image for each one of its three channels (RGB) totalling 3 PCAs per image while for the Context Method only one PCA was applied to the hole dataset obtained from the VGG16 Neural Network.

Further describing the applied method, the additional principal components are defined such that they maximize the variance in a direction “that is orthogonal to the directions already considered”. In the case of the Pixel Method, sigma is the covariance matrix associated with 20 largest eigenvalues according to the selected parameters.

$$\frac{1}{n}\sum_{i=1}^{n}\{\boldsymbol{x}_i'\boldsymbol{u}_1 - \bar{\boldsymbol{x}}'\boldsymbol{u}_1\}^2 = \boldsymbol{u}_1'\widehat{\boldsymbol{\Sigma}}\boldsymbol{u}_1 \qquad \text{with} \qquad \widehat{\boldsymbol{\Sigma}} \equiv \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})'$$

For example: in the case of a PCA with 1 component, the selected eigenvector is the one associated with the largest eigenvalue of the covariance matrix:

$$\boldsymbol{u}_1'\widehat{\boldsymbol{\Sigma}}\boldsymbol{u}_1 = \lambda_1$$

Subsequently, if one uses a 20 components PCA, as in the case of this project in its first preprocessing approach, the 20th eigenvector is that one associated with the 20th largest eigenvalue.
This method was applied in its basic form because neither the dimensionality nor the sparsity of our data were relevant enough to recur to more specific methods such as Sparse PCA or other optimization methods to perform the Singular Value Decomposition (SVD).

The number of eigenvalues selected for the Pixel Method were decided based on the average proportion of explained variance for all the images, specifically considering that for the majority of observations at least 90% of the variance was explained.

When it comes to the Context Method, the number of components was set as the maximum possible, limited only by the number of observations (1425) to allow SVD to be feasible. This decision was taken because the Neural Network already supposed a strong dimensionality reduction from the original data and, a result with approx. 3000 variables still seemed reasonable for the application of the K-Means Algorithm

K-Means Algorithm

To further reduce the dimensionality of the observations but mainly with the goal of clustering similar images, a K-means algorithm was applied and the optimal number of clusters decided based on the Heuristic of the Elbow Method.

The K-means algorithm assigns each observation to a cluster such that the distance to that cluster is minimized and the distance to other clusters maximized. Each cluster is characterized by a randomly initialized point and, iteratively, different observations are assigned according to the previously mentioned distance metric. The metric to optimize is the following that calculates the sum of the squares of the distances of each image to its assigned vector μ_k. When $r_{ik} = 1$ the image is assigned to the cluster k, when $r_{ik} = 0$ the image is not assigned to cluster k.

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

The elbow method plots the number of clusters and a distortion measure. The idea is to identify the number of clusters associated with an inflection point in the distortion measure. The idea behind this heuristic is to select the smallest quantity of clusters, therefore, after an inflection point where the distortion measure is stabilized, it makes no sense to keep adding additional clusters.

In the case of this project, the decision was made taking into account not only the previously mentioned method, but also, considering the fact that the dimensionality previous to clustering the images was huge and it was possible to select some extra clusters than those suggested by the method as will be showed in the results.

b. Results

Results for the unsupervised section are going to be shown in two separate sections, one for each preprocessing approach (Pixel and Context Methods). It is important to state that all of the presented results in the upcoming section are going to be used as input features for the models stated under Section 6 hence the used techniques must be placed under the preprocessing stage of the project.

i. Pixel Method

The PCA algorithm in this instance was applied independently at each channel (RGB), each one consisting of a 100x100 matrix representing the image in the corresponding color. Results were evaluated for 10, 20 and 30 principal components for each one of the channels and the number of components was decided based on the proportion of explained variance provided across the three colors. The reference for this analysis was to choose a number of components that granted, at least, 90% explained variance on average across every image for the three channels.

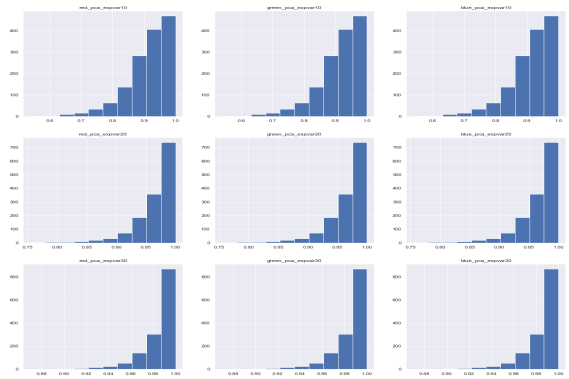


Figure 4.1 Explained variable across images & channels

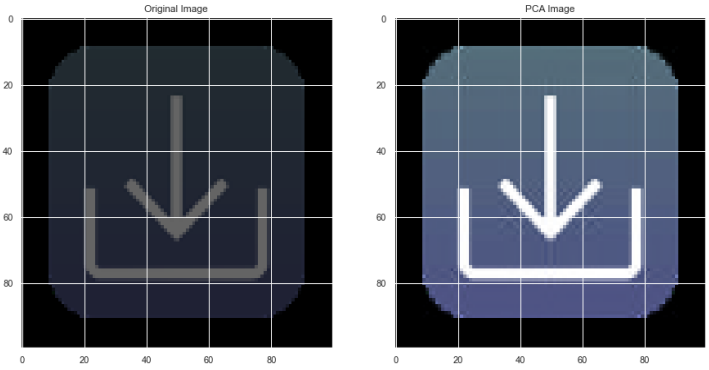


Figure 4.2 Original image vs PCA 20 components image

It is important to note in this section that the original matrix dimension for each image was 100x100 and that for prediction purposes under the Pixel approach the data has to be transformed into a 1x(Number of Rows*Number of Columns) array representing each element a pixel for a specific color. This implies three 1x10000 arrays per image and per color, clearly exemplifying the need for dimensionality reduction. Also one must consider that in this particular project the dataset consisted only of 1425 observations but, in practice, image processing is done at much larger scales and image size becomes relevant.

Analyzing the results from the PCA (Fig 4.1), one can conclude that selecting the 20 first components seems reasonable and in line with the premise previously stated where the explained variance had to be, at least, 90% for the great majority of images. The number of components selected implies the creation of three 100x20 matrices, one for each color. As a result, after decomposing the three matrices into an array, 6000 columns, per image, were obtained. (2000 pixels per color times 3 colors).

Figure 4.2 shows the comparison between the original image and the one after PCA decomposition was applied and evidences that the goal of retaining the maximum possible explained variable was achieved. It is clear that the resolution dropped substantially but the image can still be clearly devised.

The PCA decomposed data was then fed into a K-Means clustering algorithm allowing a similarity analysis between images while also providing a way of comparing the distribution of different apps categories across the different clusters. The following results were obtained:

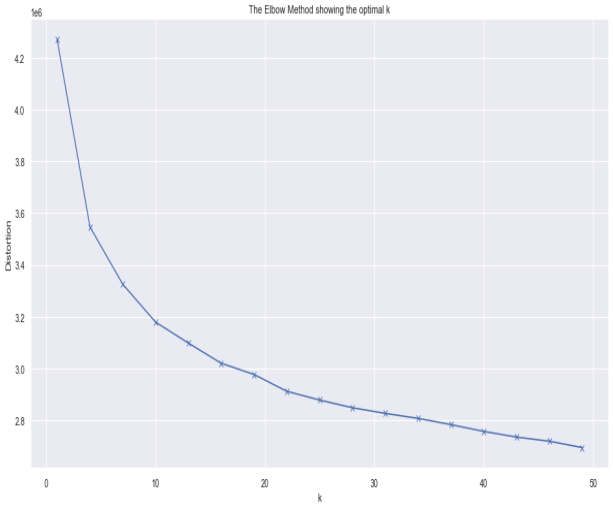


Figure 4.3 Elbow method for 1-20 clusters

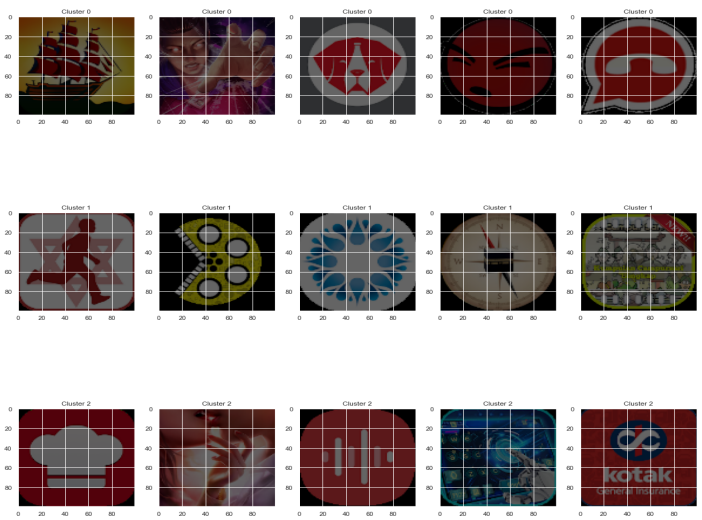


Figure 4.4 Random selected images from three clusters

The elbow method presented in Figure 4.3 suggests the number of clusters should be set around 18 (where one could devise what seems to be an inflection point). Even though the distortion measure kept decreasing proportionally as the number of clusters increased, its slope is evidentially smaller after that point. Since the clustering is considered as a preprocessing step and the objective is to obtain as much information as possible, 18 clusters were selected and no further methods for decision were analyzed while it may seem reasonable to explore the Gap-statistic method for precision purposes.

Analyzing Figure 4.4 it is evident that the clustering method worked as expected and similar images were grouped into the same clusters. Images tagged under 'Cluster 1' evidentially share the same rounded shapes while images under 'Cluster 2' present a clear predominance of red tones. When it comes to 'Cluster 0' one could argue those images could be easily related to different types of games. Appendix Figure number XX shows how original game categories are spread across the different selected clusters.

ii. Context Method

The PCA algorithm in this instance was applied generally at the 4096 features obtained from the VGG16 Neural Network. Each observation consisted of a 1x4096 array representing the image with its corresponding features. Results were evaluated for principal components considering that 95% of the variance was explained which resulted in 661 Principal Components selected.

As a result, a 1424x661 image dataset including only the principal components for each observation was fed into the K-Means algorithm that provided the following results:

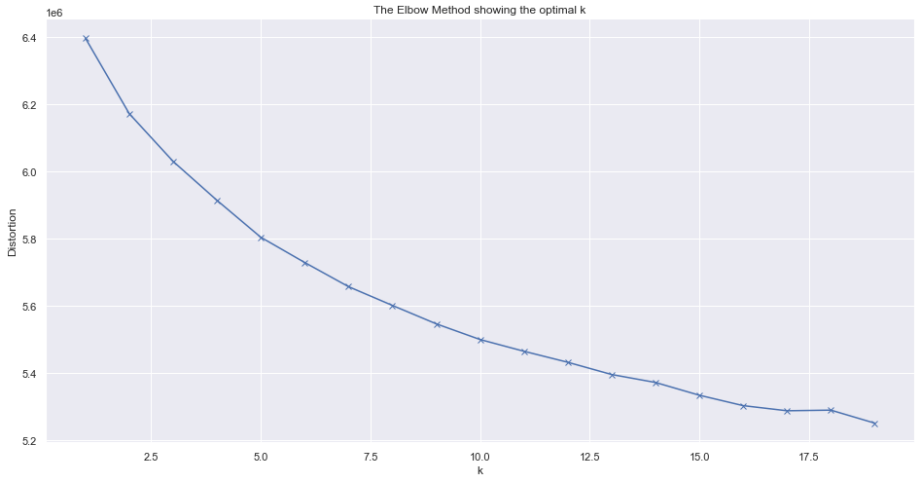


Figure 4.5 Elbow Method for Context Method Preprocessing

In this case, according to the elbow method shown in Figure 4.5, selecting 14 clusters seems reasonable since there is, apparently, an inflection point occurring for that number of clusters. The previously selected K-Means algorithm clustered the images into 14 small clusters. A small example of two relevant clusters can be seen in Figure 4.6 where one can easily devise car shapes for 'Cluster 1', and images with centered rounded shapes for 'Cluster 0'. It is important to note that the clustering method done in this preprocessing section is substantially more clear than the one presented under the pixels approach.

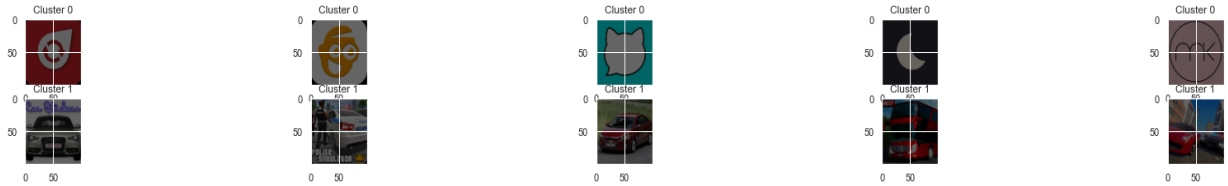


Figure 4.6 Relevant Clustered Images

5. Regression analysis

Since the main objective of this project is to find the features that truly explain the popularity of an application but also predict with new data how popular an application would be,the methods chosen for this purpose are Lasso logistic regression and Ridge logistic regression. Also the criteria to determine the best models between several techniques and hyperparameters are cross validation for AUC and the bayesian information criteria.

a. Methods:

Logistic regression

Is a statistical analysis method used to explain and predict a binary outcome based on prior observations of a dataframe. The log likelihood of a logistic regression could be expressed as:

$$l(\alpha_0, \alpha) = \sum_{i=0}^{n} -log1 + e^{\alpha_0 + \alpha} + \sum_{i=0}^{n} y_i(\alpha_0 + \alpha x_i)$$

where :

$$Y_i: \begin{cases} 1 \; if \; installs \; of \; app \; i \; is \; in \; the \; top \; 50 \; percentile \\ 0 \; otherwise \end{cases}$$

And X depends on the models to try:

$$X_i: \begin{cases} Covariates + k \; means \; pixel \; preprocessing \; method \\ Covariates + \; k \; means \; preprocessing \; context \; method \end{cases}$$

Model 1: the first model takes into account the clusters developed over the pixel preprocessing method, the aim of splitting models is to determine which type of preprocessing throws better results.

Model 2: the second model takes into account the clusters developed over the context preprocessing method.

The MLE of this model can be found by differentiating the above expression with respect to its parameters and setting them to be 0. This logistic model was chosen for this analysis since the output is a binary variable describing if the app is popular or not given features of the app.

Lasso Penalty

To manage the problem of overfitting but also as a way to select the main features explaining the popularity of an app lasso penalty gives an elegant approach penalizing the size of the parameters by adding an L1 norm penalty weighted by a hyperparameter.

$$\hat{\alpha} = argmin_{\alpha} - \log p(y|\alpha) + \lambda|\alpha|_1$$

Lasso fits very well the purpose of feature selection since it allows to set parameters exactly to 0 and in this way giving a method to choose the features that truly explains apps popularity, feature selection was one of top goals of this project since the dimensionality and complexity of the dataset increases when working with images.

Ridge Penalty

To manage the problem of overfitting but also as a way to improve the logistic prediction ridge penalty gives an approach that could be solved computationally fast since the log likelihood of a logistic regression is concave and the L2 penalty is strictly convex, therefore this optimization results in a unique solution.

$$\hat{\alpha} = argmin_{\alpha} - \log p(y|\alpha) + \lambda|\alpha|_2^2$$

The velocity of solving this optimization problem is important especially in cases where the sizes of the databases are considerably high, for this particular problem of Google applications analysis where the data set is a random sample this could be not important but for the generalization with approximately 2 million applications this could be a key determinant to choose across penalizations.

Cross validation

Another way to tackle the overfitting problem is to apply cross validation over the dataset and observe how a particular metric behaves in the test subset across different folds. For the different models and penalizations we follow the next steps:

1. Divide the 1.425 observations into 10 subsets.
2. For each subset fit a model using all data except that subset. Obtain a prediction for the subset.
3. Estimate ROC-AUC.

Bayesian Information Criteria (BIC)

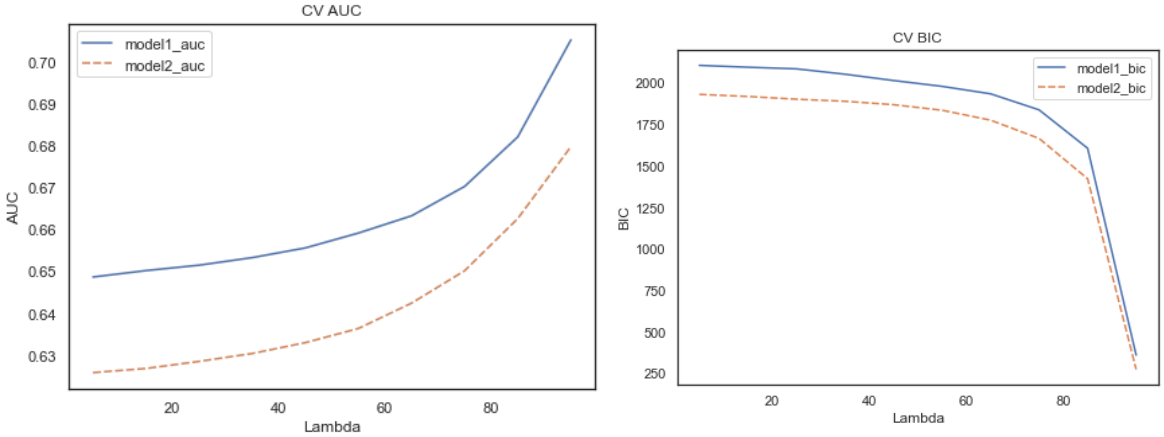
Although this project does not uses a L0 penalized regression since for this one must solve a discrete optimization problem that may take too long, it uses bayesian information criterion that is strongly related to L0 penalty to evaluate the performance of differents models and select the best, BIC criteria is defined as:

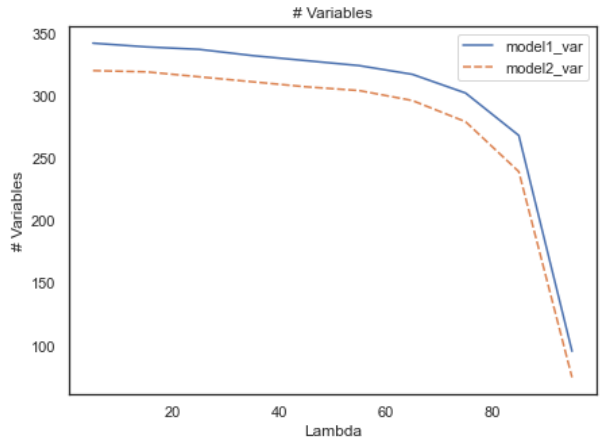
$$BIC = -2logp(y|\alpha) + \log(n)\,|\alpha|_0$$

This criteria has a really convenient property for the problem we are tackling that is model selection consistency as n growths to infinity.

b. Results

A Lasso-Penalized Logistic Regression was fitted for each model (Model 1 and Model 2) with different penalization weights ranging from 0.5 to 1. The following graphs present the results for cross validated AUC, BIC and the number of coefficients different to zero.





As a result of different simulations of lambda for both models under Pixel Preprocessing Method and Context Preprocessing Method the best metrics for CV AUC and BIC were obtained under the most penalized logistic regressions, suggesting that the most parsimonious models, the ones with less number of variables are the best to predict popularity but also the most consistent ones. Comparing the metrics between the two models it is possible to infer that the model 1 "The Pixel Method" behaves better than the Context one, this is clear for the CV-AUC metric where the difference is almost 3 percentual points. As expected, according to the theory, as penalization increases, both the BIC and the number of variables tend to similar values for both models. It is important to state that models are being selected according to its AUC and not other criterions.

Regarding the best model, chosen by CV AUC criteria, the quantity of non-zero features to explain popularity were 96 variables. The following table shows the top 10 important variable ordered by its contribution according to the mutual information classifier criterion:

Most_Important_var	**scores**
Big_Category2_Games cluster_NN_17	4,73%
time_realise_update Ad Supported_True	4,59%
Category_Communication cluster_NN_16	4,05%
size_norm time_realise_update	3,86%
size_norm In App Purchases_True	3,52%
version Category_Simulation	3,39%

Most_Important_var	scores
time_realise_update Category_Racing	3,27%
version cluster_NN_16	3,19%
Category_Adventure cluster_NN_2	3,19%

This table suggest images may have certain relevance when explaining application popularity since 4 of the 10 most important variables contains an interaction between clusters and other covariates such as category or version, in particular the combination between category and images seems very relevant especially in those categories where customers has a free will to decide across applications, games and communications are good examples of this, in contrast others categories as business or tools where the choices are reduced to particular necessities the images are not relevant.

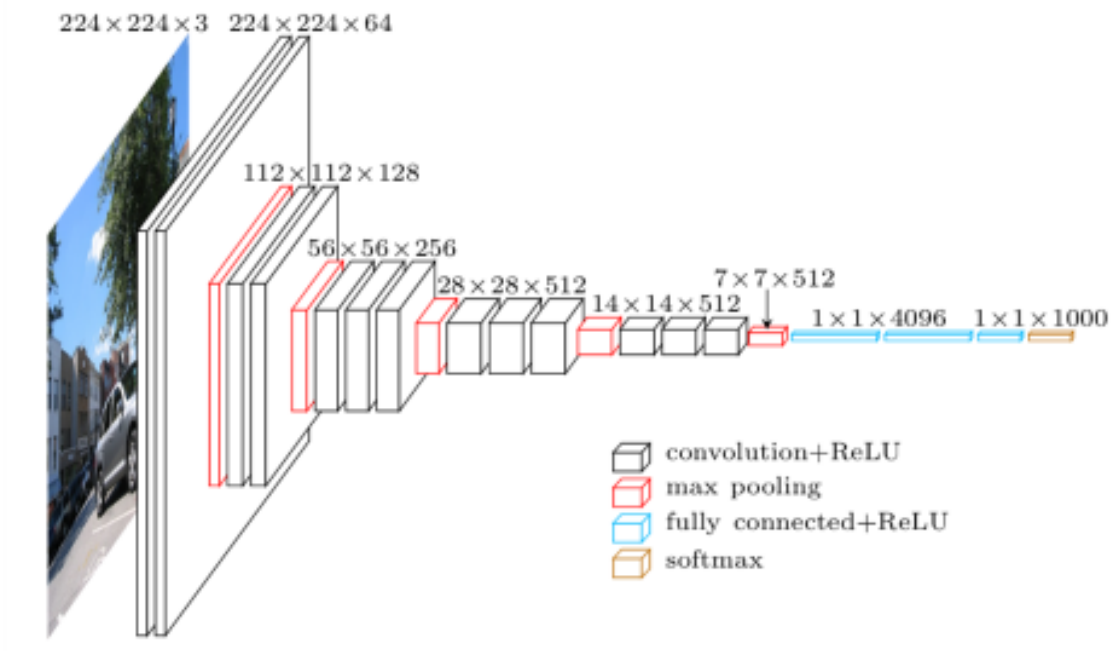
6. Discussion

The general purpose of this project was to try to determine whether images helped by any mean to predict the popularity of an app measured in its number of downloads. Even though the primary idea was to perform the analysis under a regression framework, a more conservative approach was proposed: transform the problem into a classification one that, basically, aims to differentiate those promising apps from regular or just mediocre ones. This evidentially reduces the level of complexity because now the goal is no longer to estimate those rare happening events with the same accuracy than those average events (considering the basic estimate for regression error that equally penalizes both errors), but to differentiate the two extremes, good from bad applications.

Under this new framework and analyzing several models combining multiple image preprocessing strategies the found results are encouraging. Far from being able to create a sound conclusion, it is possible to state that the obtained results suggest that images, for certain categories and under certain circumstances, do matter. Specifically in this document were presented two Logistic Regression with L1 regularization models, one trained with the data preprocessed under the Pixel Method and the other one under the Context Method. According to the results obtained under L1 (Lasso) penalization for the Pixel Method preprocessed data (the one that provided better CV-AUC metrics), 96 variables were found relevant (non-penalized to zero) while in the Top-10 most important variables one could devise a significant presence of Image-related variables.

7. Appendices

VGG16 Neural Network Architecture where the used 1x4096 can be exemplified:



References:

VGG16:
Karen, Simonyan and Andrew, Zisserman "Very Deep Convolutional Networks for Large-Scale Image Recognition"

Dutta, Amartya and Barbhuiya,Ferdous Ahmed. 2021. "Predicting Popularity of Images Over 30 Days"
Ashwini, Tonge and Cornelia, Caragea. 2019. "Image Privacy Prediction Using Deep Neural Networks"
Khaled, Almgre and Jeongkyu. Lee. 2016."Prediction of Image Popularity Over Time on Social Media Networks"
Yang, Jian and Yang, Jing-yu. 2001. "From image vector to matrix: a straightforward image projection technique—IMPCA vs. PCA"

Git-Hub Repository: https://github.com/aferreirap/finalproyect-stats