S

Machine Learning Engineer Nanodegree

Capstone Project

Mahmoud Helal

May 4th, 2019

Definition

1. Project Overview

Mobile application market considered one of the fastest growing markets in the last few years, according to (Gandhewar N and el.) [1] in 2010 Mobile devices approximately have usage 3.5 times more than PCs, that not only leads most of the companies with web-based systems like Facebook and Twitter, and companies that develop desktop applications such as Acrobat, Office, Photoshop to provide their services on mobile devices but also recently make companies and startups boost their business by only target smartphones platforms such as telegram, WhatsApp, Uber and TrueCaller or developing mobile games like PUBG, Clash of Clans and Candy Crush Saga.

2. Problem Statement

Most new applications and startups tend to fail; 90% of startups fail and the number one for failure is the lack of market need of their product [2] , so any startup or company intend to develop a new mobile app needs to make certain their App fulfill a market need;

A model can predict application success by classifying the estimated number of downloads will occur based on previous similar applications published on the App store with similar features can help most of startups and mobile development companies to take business decisions to proceed or alter their plans from earlier stages, save money and reduce risks and increase customer satisfaction.

We will try to construct a model using known supervised learning classifiers such as logistic regression, Naïve Bayes, Decision Trees, SVM. Gradient Boosting against real dataset crawled from Google play store, also will try cross-validation to test reliability and PCA to enhance performance.

3. Metrics

The main evaluation method that will be used to measure kernel performance is f beta score with beta = 1; F1 score is considered a good evaluation metric for unbalanced datasets, the optimal value at 1 for perfect classifier and worst value at 0 for random classifier, the F1 score equation as follows:

$$F1\ score\ = 2 * \frac{precision\ *\ recall}{precision + recall}$$

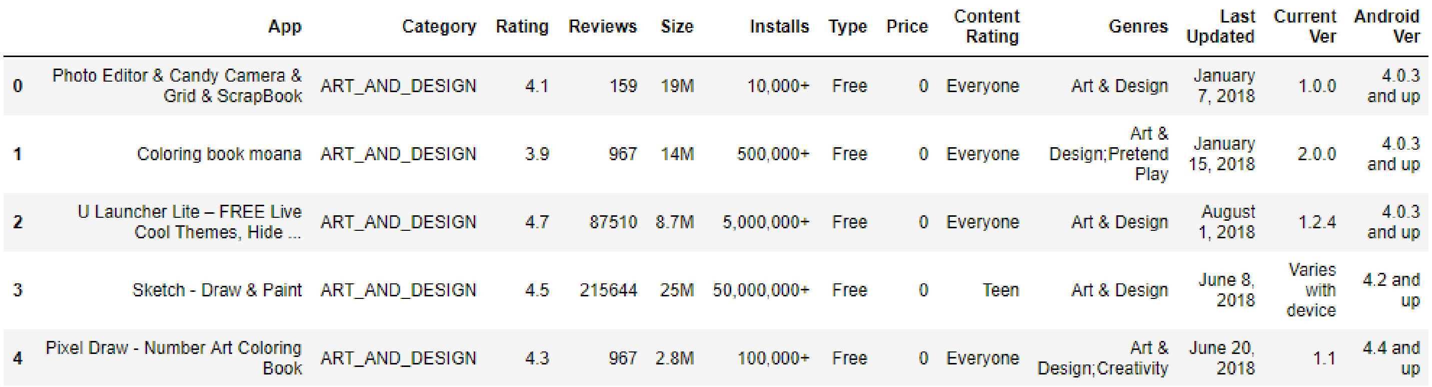
Where $percision = \frac{true\ positive}{true\ positive + false\ positive}$

And $recall = \frac{true\ positive}{true\ positive + false\ negative}$

Analysis

1. Data Exploration

The dataset that will used is "Google Play Store Apps" [3] from Kaggle, the dataset structured in CSV format in one file named “googleplaystore” with 10.8k rows and 13 columns each row indicates real mobile application in play store and columns represent features as follows : Application name, Category the app belongs to, Overall user rating, Number of user reviews, Size of the app, Number of user downloads/installs, Type (Paid or Free), Price, Rating, Age group the app is targeted at (Children, Mature 21+, Adult), Genres, Last Updated, Current Version, Min required Android version, the table below shows sample of dataset.



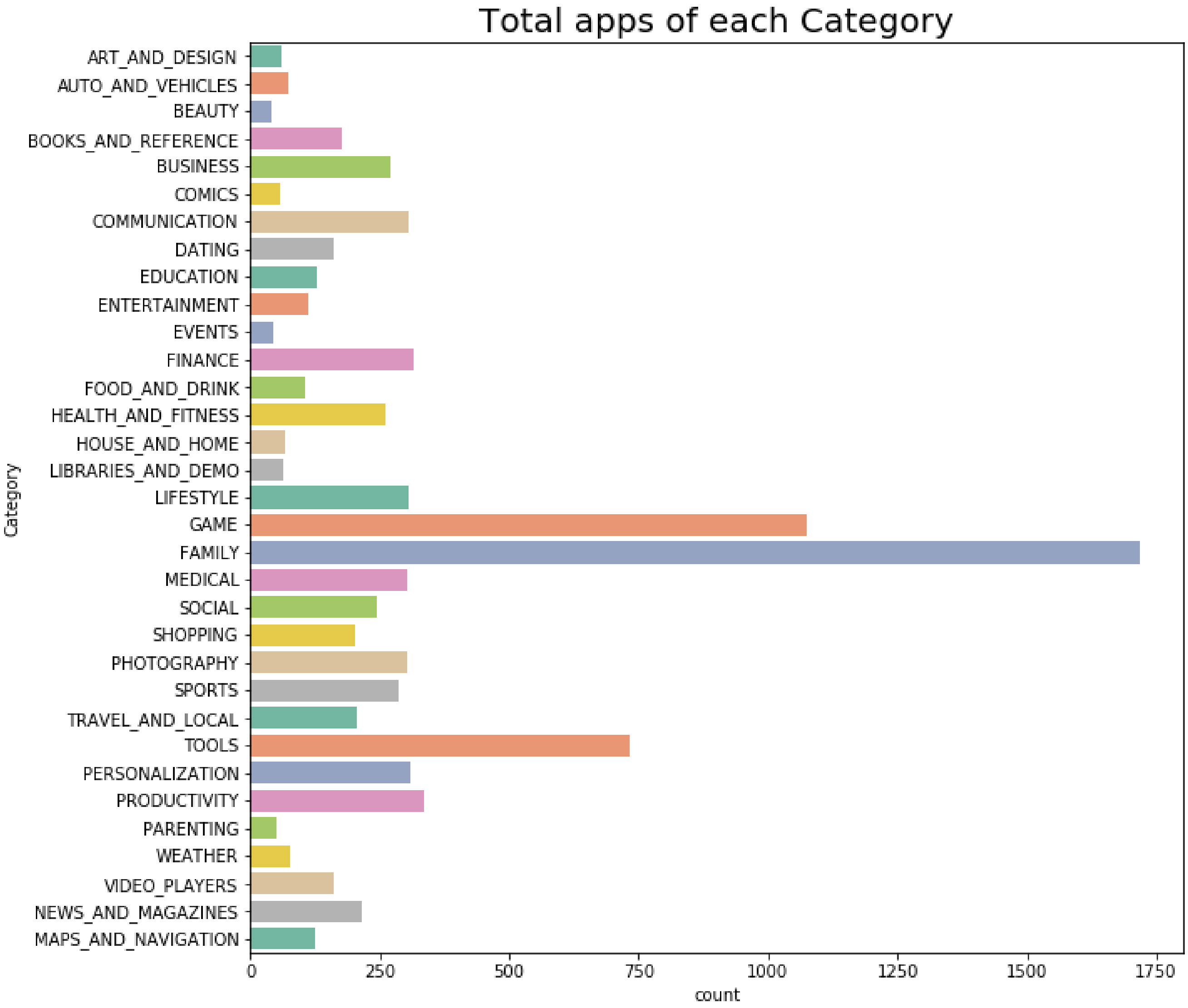
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up

The dataset contains NAN values and most exist in Rating column with 13.6%, sadly we think Rating column is an important feature that affect application success as more rating can leads to a greater number of installs unlike other columns/features like Current version, Android Version, Content Rating and Type. Also, the dataset contains 474 duplicate row that represent 5.06% of overall dataset

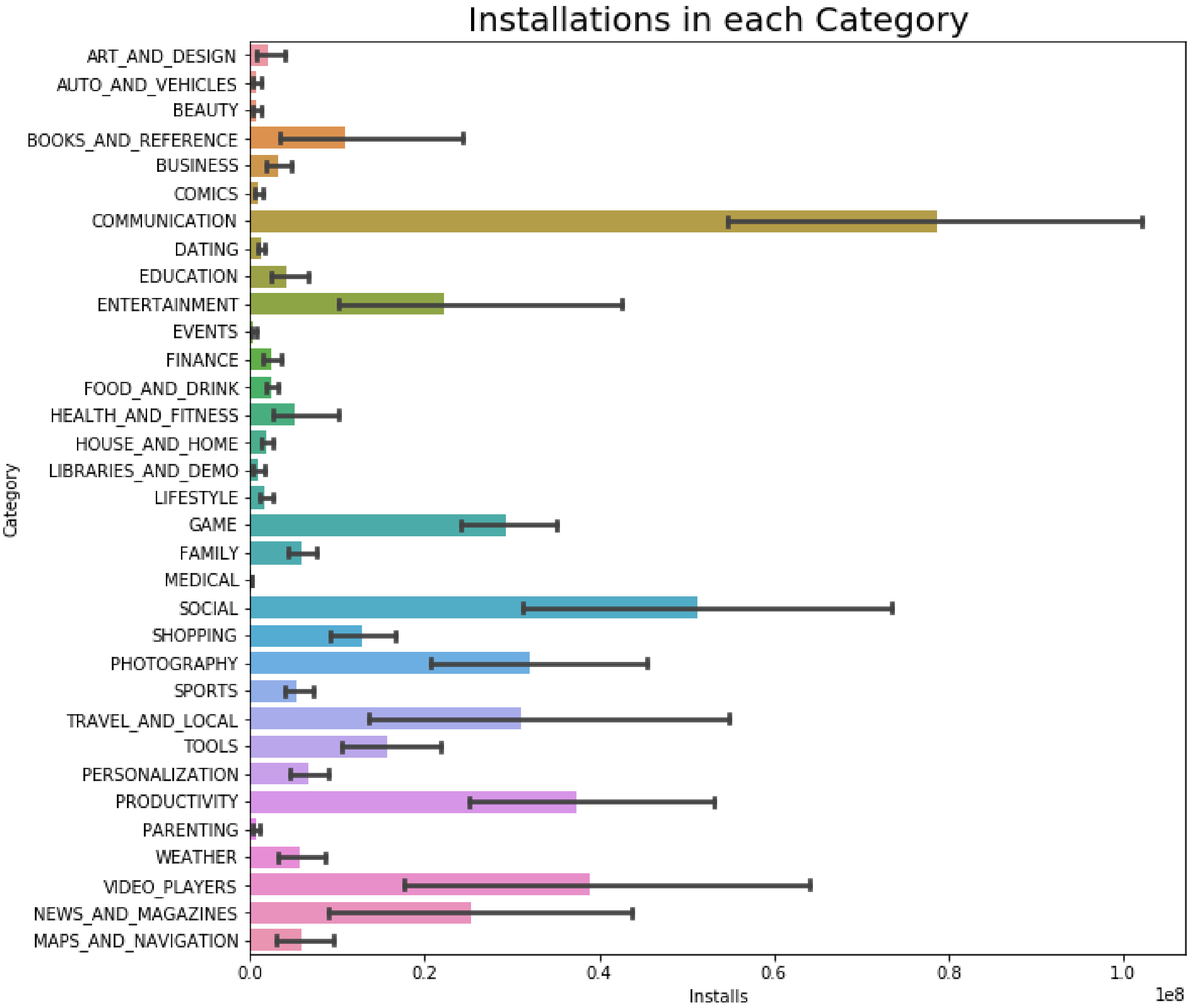
2. Exploratory Visualization

We explore and visualize dataset as the graphical representation of data will allow us to better understand the dataset, data patterns and outliers can be detected using visual elements like charts and graphs.

The following graph will represent the number of applications per category that will indicate the category most applications belongs to in this dataset and as shown the largest number of apps under Family category (above 1600 app) as mentioned earlier then Games (above 1000) and Tools (above 650), then other apps distributed among other categories.



The second graph will demonstrate the number of installations in each category this will represent the most (successful) category that mobile apps belong to, and as shown the most installed apps under Communication category even the number of apps wasn't high in this category (see the first graph) then the Social applications



3. Algorithms and Techniques

We tried four classifiers and compare results with benchmark kernel result to choose best classifier to our kernel, we used the default classifiers parameters that set in sklearn 0.19.1 library, the classifiers are:

1. **Naive Bayes:** NBC is based on Bayes' Theorem which describes the probability of a certain event, based on prior knowledge of conditions that might be related to this event.
2. **Decision Tree:** DTC goal is to create a model that predicts a target based on input features using Tree models, in these model leaves represent class labels and branches represent features conjunctions that lead to those class labels.
3. **Support Vector Machine:** SVM is considered a robust classifier that aims to find optimal hyperplane that divides feature space by finding the maximum distance between hyperplane and

the closest features for all classes (margin), also SVM can be a non-linear classifier if non-linear kernel function is used.

4. **Gradient Boosting:** Gradient Boosting Classifier used an ensemble trees to create more powerful prediction model for classification and regression, by building a series of trees where each tree trained so that it attempts to correct the mistake of the previous tree in the series , Gradient Boosting algorithm uses a lot of shallow trees known as a weak learners built in a nonrandom way to create a model that makes fewer mistakes as more trees are added.

Also cross-validation technique is used to test classifiers reliability, cross-validation divides the training data into k parts (k=10 in our implementation), then using 9 parts as a training dataset and test with the remaining part (validation dataset) then repeat this operation 10-times then the final result is the average score of 10 trails

4. Benchmark

We build the basic kernel with simple classifier (**Logistic Regression**) to use the output performance as the datum for other classifiers and enhancements applied to the kernel to compare how far our kernel progress.

Benchmark result was (0.4203) in 0.0939 seconds for training dataset and (0.7385) in 0.0140 seconds for testing dataset.

Methodology

1. Data Preprocessing

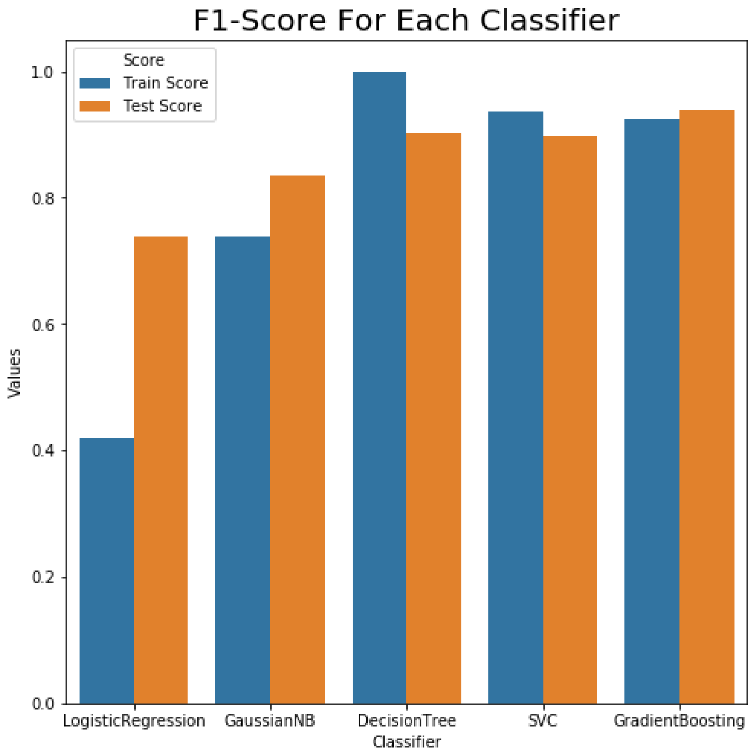
Data was preprocessed for classification to enhance classifier accuracy and our kernel performance also will work on 'Installs' column as it considered our target labels, the following actions done in preprocessing phase:

- Cast features with their datatypes (instead of an object) so columns Reviews will cast to int, Last Updated to date-time, Rating and price to float (and remove $ from Price column).
- Encode features (Type, Category, Content Rating) by label encoding method as most articles state classifiers do a better job with numeric features.
- One-hot-encoding will be applied to Genres features and handle Apps with multiple Genres.
- Replace 'Varies with device' values Size column with an estimated average size in the same category, also unified size values by converting Megabytes and Kilobytes into Bytes, then normalizing App size values.
- Sorting data by 'Last Updated' values that make data sorted from old to new, this step is necessary for the next stage (split train and test sets) to make test set contains newer data which simulate the old data used to train our kernel and the new data is used for prediction
- Remove un-important features that we think it'll affect a lot like (App, Current Ver, Android Ver, Last Updated), as Last Updated used only for sorting and more need as a feature for classifiers

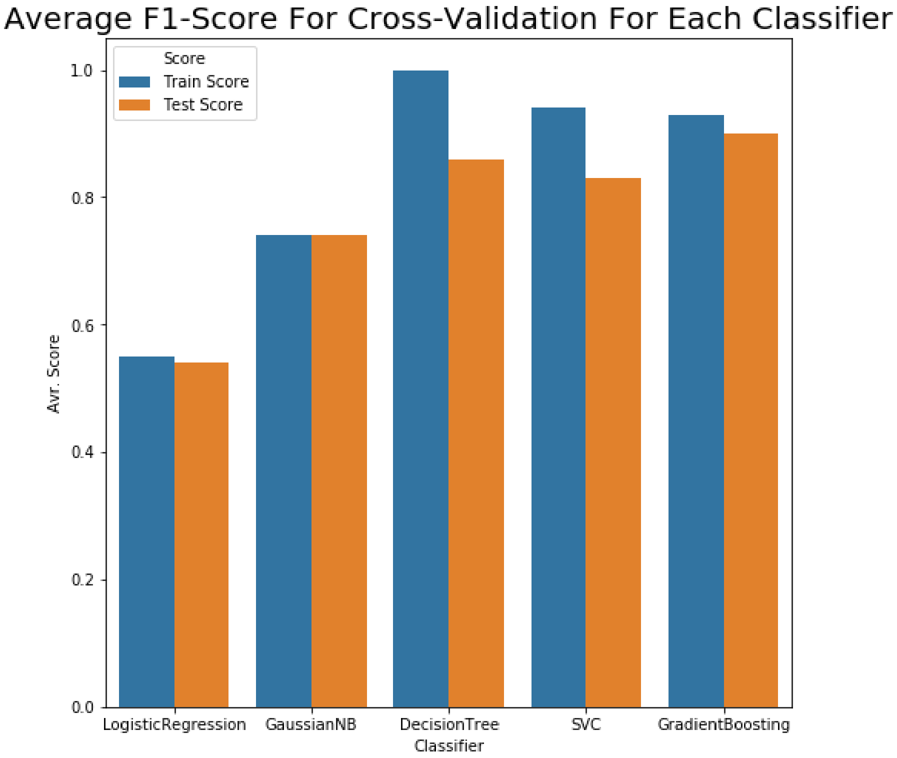
2. Implementation

We split the dataset into training and testing sets we will use 75% of data as a training set and 25% as testing set taking in consideration the newer data are in prediction which simulate the old data used to train our kernel and the new data is used for prediction

The results show the top classifiers are Gradient Boosting Classifier with score = 0.938 and Decision Tree Classifier with score = 0.90, SVM was very close with score = 0.89 and although NBC is the last score = 0.83, the graph below demonstrate F1-Score result for each mentioned classifier.

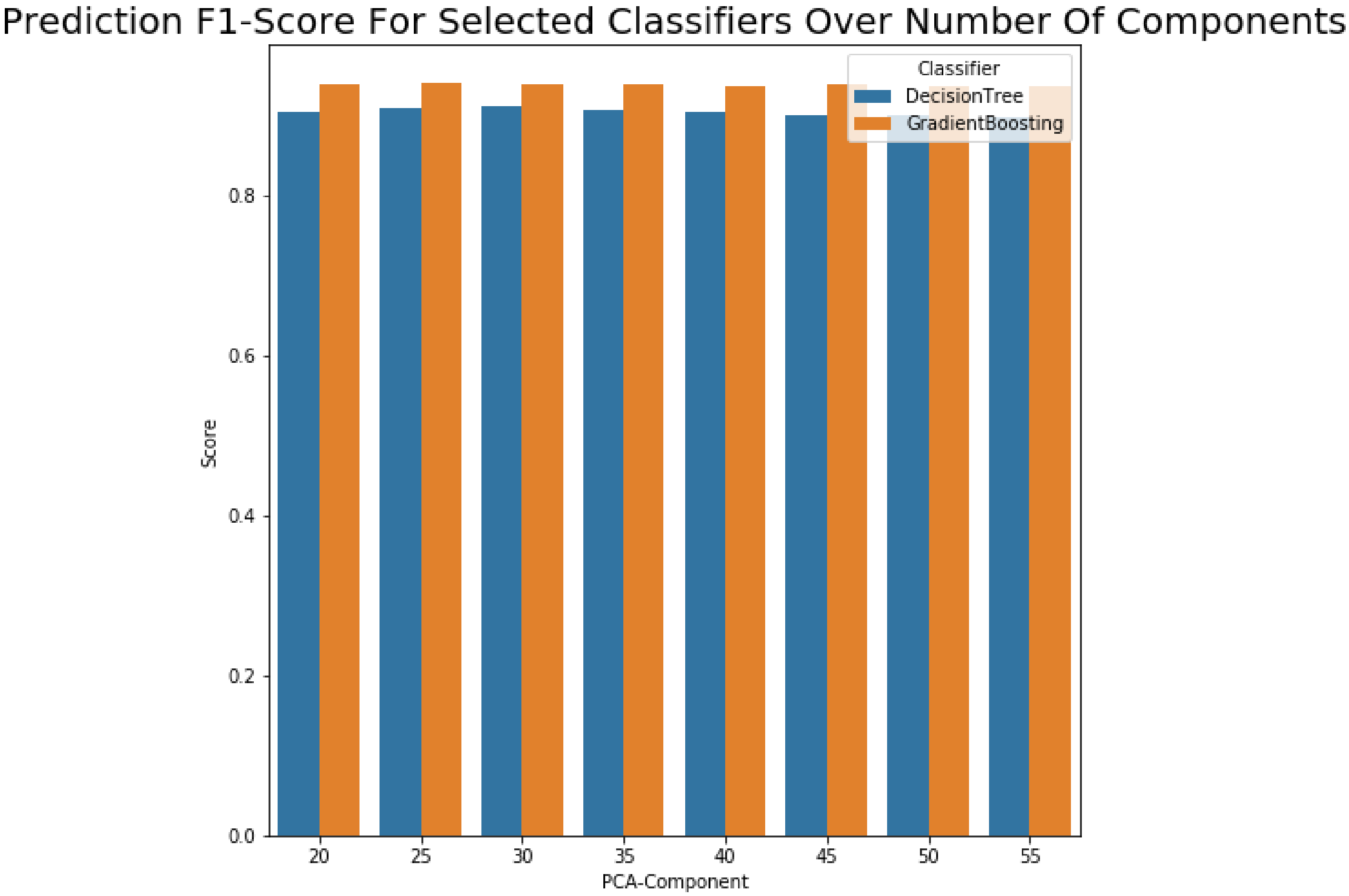


Gradient Boosting Classifier and Decision Tree Classifier proves there are most reliable classifiers in our Cross-validation technique test with mean scores 0.90 and 0.86 respectively, the graph below shows average F1-Score for each classifier in the cross-validation phase.



3. Refinement

We applied PCA before classification stage as many articles mentioned it should improve classification performance , PCA is a technique for dimensionality reduction the main goal is to project data on the minimum number of principal components that represent the maximum number of variance of data, the graph below demonstrate F1-Score for prediction data for our selected classifiers (Decision Tree and Gradient Boosting) over number of components ranges from 20 to 55, the best score for Decision Tree Classifier was 0.9122 when was number of PCA component = 30 and best score for Gradient Boosting Classifier was 0.9419 when was number of PCA component = 25, both results are slightly increased.



Result

1. Model Evaluation and Validation

We started building model as benchmark model with score (0.7385) then trying different classifiers in order to increase performance and the scores ranges from (0.8339) to (0.9383),then try enhance our kernel using PCA technique to reach our final kernel with 25 principal components and Gradient Boosting Classifier to detect Mobile App success with final f1-scores (0.9308) in 12.45 seconds for training data and (0.9415) in 0.0879 seconds for testing data, we believe our results enhanced by (20.3%) from benchmark model which is good improvement.

2. Justification

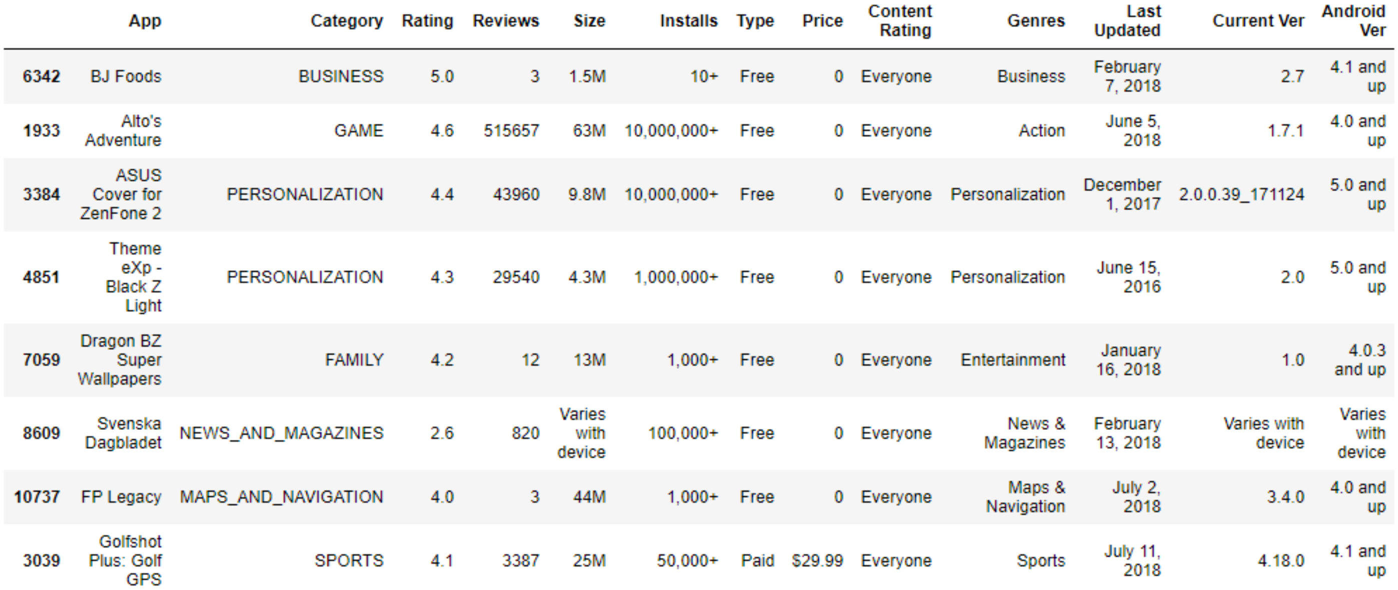
This is considered first time implement system using python and Jupiter notebook from scratch, of course faced some challenging points like drawing graphs (as it's not mentioned in Nanodegree), integrate PCA with classifiers (solving dimensions compatibility) and add the final test (make sure it working correct), classifiers results was surprising as I wasn't expect the results will be that high and, PCA result was surprising also as many articles mention a dramatic improvement after applying this algorithm but it was a small improvement in my kernel, overall the project was really useful and I learned a lot from it.

Conclusion

1. Free-Form Visualization

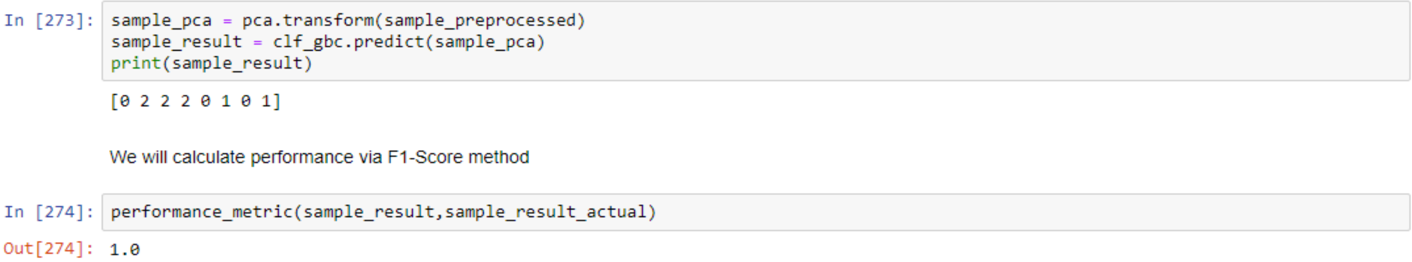
We took 8 random Apps and check their number of installs and get their actual successful rate (Fail, limited success, success), then run against our final kernel then compare predicted output with actual one and calculate the score, the table below shows random Apps fall under different installs category:

- 3 failed Apps (1000 install or below): BJ Foods, Dragon BZ Super Wallpapers, FP Legacy.
- 2 limited success App (between 1K and 1M): Svenska Dagbladet, Golfshot Plus: Golf GPS.
- 3 success Apps (1M and more): Alto's Adventure, ASUS Cover for ZenFone 2, Theme eXp - Black Z Light



	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
6342	BJ Foods	BUSINESS	5.0	3	1.5M	10+	Free	0	Everyone	Business	February 7, 2018	2.7	4.1 and up
1933	Alto's Adventure	GAME	4.6	515657	63M	10,000,000+	Free	0	Everyone	Action	June 5, 2018	1.7.1	4.0 and up
3384	ASUS Cover for ZenFone 2	PERSONALIZATION	4.4	43960	9.8M	10,000,000+	Free	0	Everyone	Personalization	December 1, 2017	2.0.0.39_171124	5.0 and up
4851	Theme eXp - Black Z Light	PERSONALIZATION	4.3	29540	4.3M	1,000,000+	Free	0	Everyone	Personalization	June 15, 2016	2.0	5.0 and up
7059	Dragon BZ Super Wallpapers	FAMILY	4.2	12	13M	1,000+	Free	0	Everyone	Entertainment	January 16, 2018	1.0	4.0.3 and up
8609	Svenska Dagbladet	NEWS_AND_MAGAZINES	2.6	820	Varies with device	100,000+	Free	0	Everyone	News & Magazines	February 13, 2018	Varies with device	Varies with device
10737	FP Legacy	MAPS_AND_NAVIGATION	4.0	3	44M	1,000+	Free	0	Everyone	Maps & Navigation	July 2, 2018	3.4.0	4.0 and up
3039	Golfshot Plus: Golf GPS	SPORTS	4.1	3387	25M	50,000+	Paid	$29.99	Everyone	Sports	July 11, 2018	4.18.0	4.1 and up

And Our kernel detects all eight random Apps successfully.



```
In [273]: sample_pca = pca.transform(sample_preprocessed)
          sample_result = clf_gbc.predict(sample_pca)
          print(sample_result)

          [0 2 2 2 0 1 0 1]
```

We will calculate performance via F1-Score method

```
In [274]: performance_metric(sample_result,sample_result_actual)
Out[274]: 1.0
```

So, we believe the final kernel implemented capable of solving our business problem as it's achieved high score (0.9415) in detecting 25% of dataset correctly, also passed our smoke test and detects all random Apps successfully.

2. Reflection

This kernel implementation consists of 8 main stages:

1. **Datasets and Inputs:** Import our dataset and remove noise (NANs, duplications, ...).

2. **Data Visualization:** Explore and represent our dataset graphically in order to have some insights.
3. **Data Pre-processing:** Prepare our dataset for classification phase (removing not useful features, make one-hot-encoding, …)
4. **Evaluation Metrics:** Identify our score method that used to evaluate our classifiers.
5. **Benchmark:** Build Linear regression classifier as a benchmark model and get baseline performance values.
6. **Algorithms and Techniques:** Explore more classifiers (Naive Bayes, Decision Trees, SVM, Gradient Boosting) and compare results with base kernel result then choose the best two classifiers to the enhancement stage.
7. **Enhancements:** Try to use PCA for feature reduction to enhance classification results.
8. **Final Kernel:** Based on previous results we will choose the final classifier and enhancement tuning to our kernel.

We believe these stages build a reliable kernel with a good score (0.9415), which able to solve our business problem.

3. Improvement

To enhance kernel accuracy, we can apply the following actions:

- Try to tune classifiers parameters instead of using default values (Grid Search technique might be using)
- Lower number PCA components intervals to 2 (for example) instead of 5 to try more numbers.
- Try Neural Network for classification.
- Can try increase number of label categories to 4 instead 3 to be (fail, limited success, success, booming)

References

[1] N. Gandhewar and R. Sheikh, "Google Android: An Emerging Software Platform For Mobile Devices," *International Journal on Computer Science and Engineering (IJCSE),* vol. 1, no. 1, pp. 12-17, 2010.

[2] E. Griffith, "Why startups fail, according to their founders," Fortune, 14 September 2014. [Online]. Available: http://fortune.com/2014/09/25/why-startups-fail-according-to-their-founders/.

[3] L. Gupta, "Google Play Store Apps," Kaggle, [Online]. Available: https://www.kaggle.com/lava18/google-play-store-apps.