

## **Predict Defaults on Credit Card Payments**

### **Definition**

#### **Project Overview**

For this project, I aim to identify the leading cause for default payments for money loans. Being able to identify what characteristics cause borrowers/investors to default in their payments will benefit both parties since the borrower/investor won't end up in a pile of debt and the lender won't invest money into a place of no return.

#### **Problem Statement**

The goal of this project is to create a model that can accurately depict whether a person will default or not based on certain features. The tasks involved are the following:

1. Preprocess the data and remove missing data and extra features
2. Perform unsupervised learning to determine the important features
3. Handpick a few supervised algorithms that I deem accurate for this problem.
4. Pick the best supervised algorithm out of the handful.

The final model shall be able to predict whether an investor/borrower will default based on the features provided. Hopefully, the final model is able to predict the output within a few seconds to at most minutes.

#### **Metrics**

The metrics I am using to measure the performance of the model is the F1-score and the variance to measure the importance of features. I chose to use the F1-score for this case because the accuracy measure does not work well for this dataset. There are 6636 defaults and 23364 non-defaults which comes out to be defaults happening only 22.12% of the time. If the model predicted non-defaults for every guess, the accuracy would come out to be 77.88% which is sufficiently high. Although, we do not want the model to randomly guess but rather attempt to find relationships between the features and defaulting payments. The possible options to measure the performance of the model are either the F1-score or the ROC/AUC metric.

Since I am using PCA to reduce the dimensionality of this problem, variance is the built in metric for PCA. Variance is used to determine which features have the largest impact on which

principal components. We then use the explained variance to determine which principal components have the largest impact on the result.

## **Analysis**

### **Data Exploration**

The dataset I am using is from Kaggle. The dataset is a sample of credit card clients in Taiwan from 2005 with one of the features being a binary feature; either 0 or 1 to indicate a default. Upon evaluation of the dataset, there are 6,635 out of 30,000 users in this dataset that have defaulted on their credit card payments; which means approximately 22.12% of credit card users default on their payments. The dataset's features are as follows: maximum credit line, gender, education level, marriage status, age, and history of payments. By eyeball, it's most likely that the maximum credit line and the education level are features which have impact on determining whether a credit card default will happen.

The maximum credit line feature determines the amount of given credit to a consumer in NT dollars. Gender is an input feature in case there exists a pattern for whether males or females are more likely to default on credit card payments. Education level has 6 inputs, 1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown. In theory, credit card user with a higher education level is more likely to have self-control when it comes to spending in comparison to another user with pre-university education. Marriage status is another feature because there is a possibility that whether the credit card user is single or married has an impact on the outcome of unpaid credit card payments. Age is also of importance since young adults are more likely to spend lavishly despite lacking money. The final important feature is history of payments; attempting to determine whether a credit card user will default early based on the history of payments is a goal. An example of the importance for payment histories happens when there is a pattern for defaults where the credit card user only pays the minimum amount for the bill a few months prior to the default.

I plan to first evaluate the features provided to me by determining which sets of features provide the strongest relationship to defaulting. After determining the strongest features, I will then proceed to use those features to make a model that can predict whether a credit card user will default based on the features provided.

# Exploratory Visualization

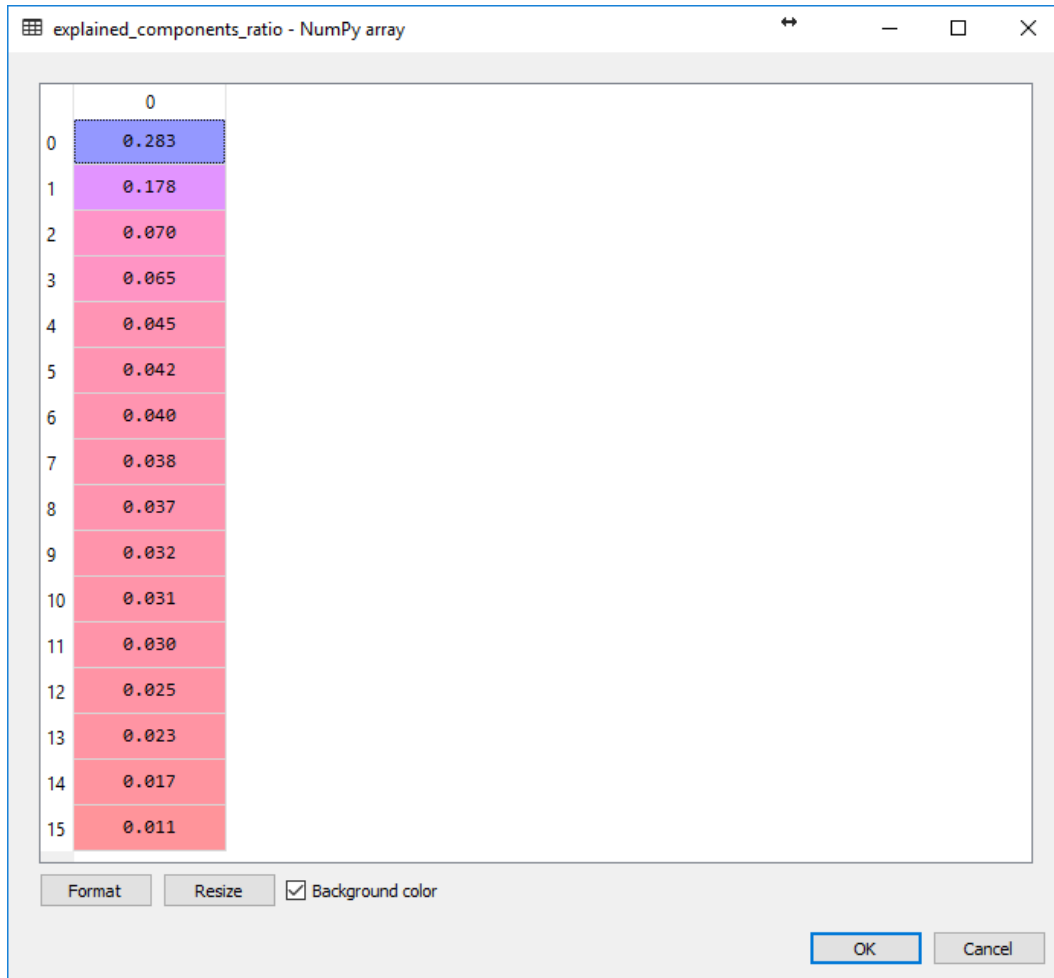
- Below, I have attached an image of a brief overview for what the dataset will look like.

The screenshot shows an Excel spreadsheet with the following columns: ID, LIMIT\_BAL, SEX, EDUCATION, MARRIAGE, AGE, RAY\_0, RAY\_1, RAY\_2, RAY\_3, RAY\_4, RAY\_5, BILL\_AMT0, BILL\_AMT1, BILL\_AMT2, BILL\_AMT3, BILL\_AMT4, BILL\_AMT5, RAY\_AMT0, RAY\_AMT1, RAY\_AMT2, RAY\_AMT3, RAY\_AMT4, RAY\_AMT5, and default.payment.next.month. The data is organized into rows, with the first row being a header and subsequent rows containing numerical values for each feature.

- Below, I show the features ranging from 1 to 23 with the feature name and which principal components they influence most.

Untitled - Notepad
File Edit Format View Help
feature 1 (Limit_Balance) - Principal Component 13
feature 2 (Sex) - Principal Component 4
feature 3 (Education) - Principal Component 6
feature 4 (Marriage) - Principal Component 12
feature 5 (Age) - Principal Component 3
feature 6 (Repayment Status in September 05) - Principal Component 11
feature 7 (Repayment Status in August 05) - Principal Component 15
feature 8 (Repayment Status in July 05) - Principal Component 14
feature 9 (Repayment Status in June 05) - Principal Component 15
feature 10 (Repayment Status in May 05) - Principal Component 11
feature 11 (Repayment Status in April 05) - Principal Component 15
feature 12 (Amount of bill statement in September 05) - Principal Component 0
feature 13 (Amount of bill statement in August 05) - Principal Component 0
feature 14 (Amount of bill statement in July 05) - Principal Component 0
feature 15 (Amount of bill statement in June 05) - Principal Component 0
feature 16 (Amount of bill statement in May) - Principal Component 0
feature 17 (Amount of bill statement in April 05) - Principal Component 0
feature 18 (Amount of previous payment in September 05) - Principal Component 10
feature 19 (Amount of previous payment in August 05) - Principal Component 10
feature 20 (Amount of previous payment in July 05) - Principal Component 9
feature 21 (Amount of previous payment in June 05) - Principal Component 7
feature 22 (Amount of previous payment in May 05) - Principal Component 6
feature 23 (Amount of previous payment in April 05) - Principal Component 7

- The picture below shows the explained variance ratio of each principal component, describing to us which principal component has the most impact ratio-wise.



## Algorithms and Techniques

The most important thing I wish to accomplish from this dataset are to determine which features provide the most impact on whether a credit card user will default on their payment or not and to create a model which will help predict future defaults. To accomplish this, I plan to use the unsupervised learning technique PCA to reduce the dimensionality of the problem and to find out which features have the largest variance. By finding the largest variance and the most impactful principal component, I am able to determine which features have the most impact on the classification problem. The end model that I wish to use for future predictions are determined from the following: SVC, PCA into SVC, PCA into Random Forest, and PCA into ADABOOST.

PCA will crunch the feature size down into a smaller size. This will reduce the dimensionality of the problem and hopefully increase our F-score to a higher amount. We will then proceed to use the new dimension space and transform the dataset to fit into the new PCA dimension space.

Using the dataset in the PCA dimension space, I plan to model the data using the three mentioned algorithms--SVC, Random Forest, and ADABOOST—and obtain the best model using the F-score metric.

## **Benchmark**

The results obtained from the model we make will be evaluated against the benchmark model. Since this is a classification problem, the benchmark model that will be used is SVM. I chose SVM to be the benchmark model for this problem because SVM is very flexible for classification problems. SVM can be used with many kernels such as linear, polynomial, rbf, and sigmoid. Given such flexibility, I will obtain a brief idea for how the customer data is distributed.

## **Methodology**

### **Data Preprocessing**

The dataset contained numbers ranging from zero to hundreds of thousands. Since the numbers have such a large range, it was necessary to perform data preprocessing to scale the data. Instead of using the logarithm approach to scale the data, I chose to use the `sklearn.preprocessing.StandardScaler` library. `StandardScaler` preprocesses data by removing the mean and scaling to unit variance.

Another preprocessing step was used, PCA. PCA is a form of feature transformation in that PCA reduces the dimensionality of the data and forming a new dimension space usually lower than the original dataset's dimensionality. By reducing the number of dimensions, we essentially allow the data to cover more space and reduce the need to acquire more data to fill open space. PCA helps us figure out which features are the most important and hopefully helps to obtain a higher f-score.

### **Implementation**

To tackle this problem, I intend to first evaluate the dataset. Downloading the dataset and briefly scanning through the dataset to ensure there is no missing data. Once the dataset is okay by the eye, I plan to do data preprocessing by removing unnecessary columns such as the customer ID. Since there are no categorical variables I don't need to do one hot encoding. I will then proceed to scale the data so that the range for all the features are evenly distributed. For example gender, education, marriage, and age are all values that don't exceed 100 whereas credit limit and bill statements can reach the thousands. I then proceed to split the dataset into a training and testing set where the testing set is only 25% of the data.

I plan to make two different codes. One is for the benchmark model and the other is for the other models. Both models will require the portion of code for data preprocessing to scale the data from -1 to 1 since there exists a large number range for all the data; some features range from 0 to 20 while other features can range from 0 to 300,000. By scaling the data, we overcome these large differences and make the model more accurate, easier to model, and faster to compute. The

benchmark model is going to be a Support Vector Machine Classifier since SVM's are very flexible given the different kernels we can use. In my code, I plan to use the RBF, Linear, Polynomial, and Sigmoid kernels. I have four different sections of code each pertaining to a different kernel but everything else is the same. The goal of each section is to fit the training data into the SVC with their respective kernels and to predict the results using the testing dataset. We then compute the f1-score of the data by using the `sklearn.metrics.f1_score` library. And finally, we print out the f1-score of the respective SVM kernel and attempt to figure out which one has the highest f1-score.

The other python file will be used to compute the f1-score of the models that will be as used as comparisons to the benchmark model. Just like the benchmark models code, there will be a portion of code for data preprocessing to scale the data from -1 to 1 since there exists a large number range for all the data; some features range from 0 to 20 while other features can range from 0 to 300,000. We then proceed to use one of the unsupervised learning methods, PCA. We use a for loop to find all the f1-scores for `n_components=2` to `n_components=23`. We find the f1-scores for the RBF SVC model, Random Forest model, and ADABOOST model. Later on, I will mention why we chose the RBF kernel for the SVC model specifically. After traversing through all the possible PCA components possible and creating a printout of all the f1-scores, we manually view the f1-scores to find the one we deem fit; in this case it will be the largest f1-score corresponding to the model we want. However, in my case I ended up choosing the model with the highest f1-score. After finding out the highest f1-score, we match it with the number of principal components and attempt to recreate the model with the given number of components to further evaluate the details inside of the PCA dataset. The explained variance ratio is outputted to show what percentage each principal component contributes to the final result. Using this data, we know which features will have the most impact on the data. This is so because the highest explained variance ratio principal component has the most impact on the outcome and the features that affects that specific principal component the most will be the ones we are looking for.

## **Refinement**

Initially, I was planning to only apply the linear kernel to the SVM. However after more thought was put into the project, I realized that it was important to test all possible kernels onto the dataset. I then proceeded to use not only the linear kernel but the RBF, polynomial, and sigmoid kernel as well.

Originally, I was planning to use the accuracy metric to evaluate the models. However, the Udacity reviewer taught me a better metric to evaluate the models. The accuracy metric is a better choice in this scenario in comparison to the accuracy metric. This is so because the accuracy metric is guaranteed to provide us with a satisfactory percentage since there is an imbalance of the classes. Only 22.12% of the data is credit card defaults whereas the other 77.88% of the data isn't defaults. This means that guessing no defaults for everything will render us a 77.88% accuracy. We don't want the model to guess all no defaults but rather we want the model to find relationships between features and the output. Therefore, the accuracy model isn't

good in this scenario. The f1-score is a better metric for this dataset since the number outputted takes into account the precision and recall.

For PCA, there are a lot of possible components that can be used for evaluation. Since there are 23 features, it's possible to have 22 different principal components; we can have one principal component, two principal components, three principal components, etc. Initially, I was thinking of only using two principal components. But, after applying further thoughts into the project I came to realize that two was a ballpark number and the most optimal number of principal components could be any of the 22. I then decided to modify my code and use a for loop to find the f1-score of all the models using all 22 different principal components. The output is the f1-score of all the models with their respective principal components.

## **Results**

### **Model Evaluation and Validation**

In the end, the best model is using PCA first with 15 principal components and following with the RBF kernel for SVC. The f1-score obtained is 0.452.

This model was obtained through various testing and comparisons. The RBF kernel is chosen out of all the other possible SVC kernels because of the benchmark model. The benchmark model tested all four other kernels for SVC and RBF came out to have the highest f1-score; scoring 0.446. We then proceeded to see if PCA would help further boost our SVC model with an RBF kernel. However, there are 23 features and 22 possible principal components for PCA. I then had to use a for loop to compute the f1-score of all the models when using every different possible number of principal components. The SVC model with a RBF kernel was tested against the Random Forest model and the ADABOOST model and came out to have the highest f1-score.

The final model generalizes well to unseen data. This is so because I made sure in the beginning to split the test data from the training data. PCA was also created on the training data and not the whole data. I did this because I wanted to make sure that PCA did not provide the model with a sneak peek of the future data. Since the test data hasn't been shown to the model in any phase of the stage, I feel that it will generalize well to unseen data.

### **Justification**

The final results found after completion shows that applying PCA before applying SVC with a RBF kernel is better than applying SVC with a RBF kernel without PCA. As shown, the difference in the f1-score is approximately 0.01.

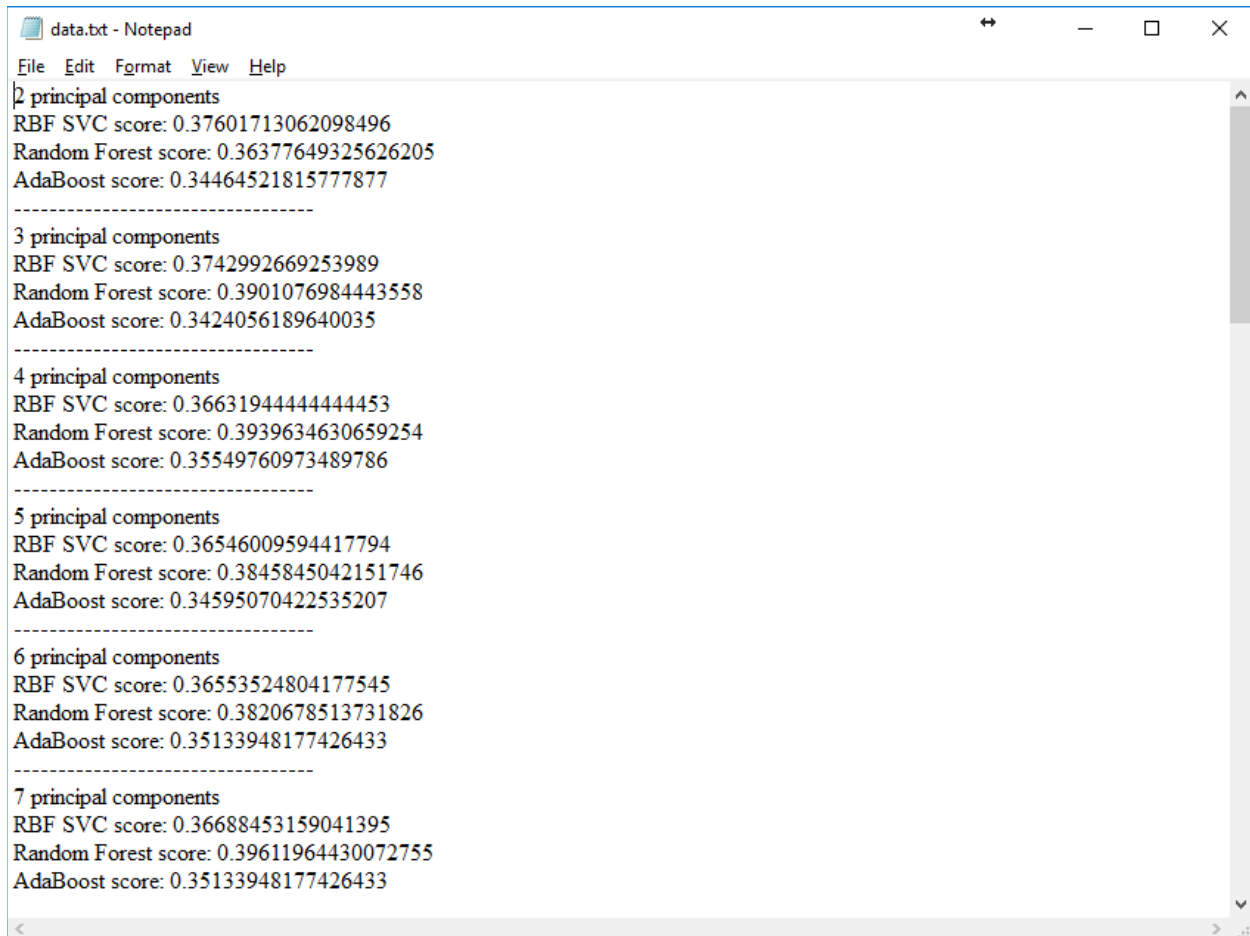
The final solution was obtained through rigorous comparisons of all possible principal components and comparing the f1-scores between all models. In total, 63 different models were made and 63 total f1-scores were compared.

The final solution isn't significant enough to solve this problem since there exists many other algorithms to model this data that I have not touched on. Examples are neural networks, Gradient Boosting, KNN, K-means, and other forms of unsupervised learning for example LDA.

However, the final solution is good enough to be used for predicting future default payments but not the best solution since everything algorithm wasn't tested.

## Conclusion

### Free-Form Visualization



```
data.txt - Notepad
File Edit Format View Help
2 principal components
RBF SVC score: 0.37601713062098496
Random Forest score: 0.36377649325626205
AdaBoost score: 0.34464521815777877
-----
3 principal components
RBF SVC score: 0.3742992669253989
Random Forest score: 0.3901076984443558
AdaBoost score: 0.3424056189640035
-----
4 principal components
RBF SVC score: 0.36631944444444453
Random Forest score: 0.3939634630659254
AdaBoost score: 0.35549760973489786
-----
5 principal components
RBF SVC score: 0.36546009594417794
Random Forest score: 0.3845845042151746
AdaBoost score: 0.34595070422535207
-----
6 principal components
RBF SVC score: 0.36553524804177545
Random Forest score: 0.3820678513731826
AdaBoost score: 0.35133948177426433
-----
7 principal components
RBF SVC score: 0.36688453159041395
Random Forest score: 0.39611964430072755
AdaBoost score: 0.35133948177426433
```

In the process of determining what number of principal components is best for the dataset and the models, we output the f1-scores for the user. This picture only partially shows a few of the principal components and the alternative models f1-scores in conjunction to the principal components used. The data obtained from computing all 22 principal components in conjunction to the RBF SVC, Random Forest, and ADABOOST models determines which model is best with what number of principal components and whether it is better than the benchmark model.

### Reflection

In the beginning of the project, I was hoping to find a relationship between the features and whether a credit card user will default on their payment. I was aiming to use a feature selection or feature reduction technique to sample the data first and then perform supervised learning after



to model the data. Feature selection wasn't something I was familiar with and not strong at so I decided to not use that technique for this project and to use that technique for another project in the future. Feature reduction however was a technique I am good at since it was taught in the Udacity course. The feature reduction algorithm I chose to use was PCA. My initial goal was to perform PCA and compare a few well known models using the accuracy score to determine who the winner will be. However, my approach for determining the best model was wrong. I found out that a benchmark model was needed and for a very good reason. The benchmark model provided me with something simple, sweet, and straight to the point. A benchmark model is easy to make and provide a decent score that can be used to compare other models. An example of a benchmark model is linear/logistic regression. In my project I decided to opt into SVM since it was more well-known and flexible with datasets. Another thing I realized was that the accuracy metric wasn't the best choice for this dataset. Since there was a class imbalance between credit card users defaulting and not defaulting, I needed to use another scoring metric. The Udacity reviewer recommended using either ROC/AUC or the f-score. I decided to use the f-score since it was something I was very familiar with. While using PCA, I needed to determine the best number of principal components for obtaining the highest score. To do that, a for loop was used to loop through all possible principal components ranging from 2 to 22. All the f1-scores were outputted for all possible principal components and their respective models; RBF kernel SVC model, Random Forest, and ADABOOST. I then proceeded to compare the highest f1-score obtained via the PCA preprocessing method with the benchmark model and deemed whichever one scored the higher f1-score as the better model.

An interesting aspect of this project is that PCA cannot be used on the whole dataset. In the beginning, I didn't know that PCA shouldn't be used on the whole dataset. Upon researching, I learned that giving the whole dataset to PCA isn't good since we are essentially giving the model a sneak peek for what the testing dataset will be.

An aspect of this project that I found difficult was determining whether the f1-score was sufficiently high enough to be considered decent. In the case of the accuracy score metric. A score of 60% or higher is considered decent but scoring higher than 70% is considered really good. This wasn't the same case for the f1-score as it uses 1 as the best possible score and 0 as the worst possible score.

## **Improvement**

There are two things I could improve for the project.

The first thing is using Gridsearch cross validation. Gridsearch is better for testing parameters and obtaining the most optimal parameter. I could have applied Gridsearch to replace the for loops when I wanted to determine what number of principal components was best for this dataset. I could have also done Gridsearch for determining which kernel was best for SVM. Gridsearch could've also been used for ADABOOST and the Random Forest model. Cross validation could've been used in conjunction with Gridsearch to obtain a better f1-score. Obtaining the f1-score on different folds of the dataset is more accurate than obtaining the f1-score on the whole training dataset.

The second thing is testing more models. Testing more feature selection and feature reduction algorithms would provide more information on which features were necessary and which weren't. They also have a chance to boost the f1-score. Trying more supervised algorithms isn't a bad idea as well since there may exist a better model but it just isn't known yet. For example, neural networks could've been used in this project or gradient boosting.

Overall, there is always room for improvement for a modeling data which leads us into software engineering and the Software Development Life Cycle.