Financial Crisis Prediction
Assignment 1

Rik Harng Loong
Student ID: 18061593

February 12, 2020

1 Introduction

Financial crises are costly and have a long-term impact on the real economy. Hence, we aim to utilize the historical data about financial crash provided and then make predictions on financial crash in the succeeding years. We will optimize and compare various models such as Logistic Regression, Random Forests, Support Vector Machines and Gradient Boosting in terms of performance based on the Area Under Curve (AUC) of the standard ROC curve and also confusion matrix. Our data are then drilled down more deeply with the help of PCA at the end of the report. All the codes are written in Python 3.

2 Data Exploration

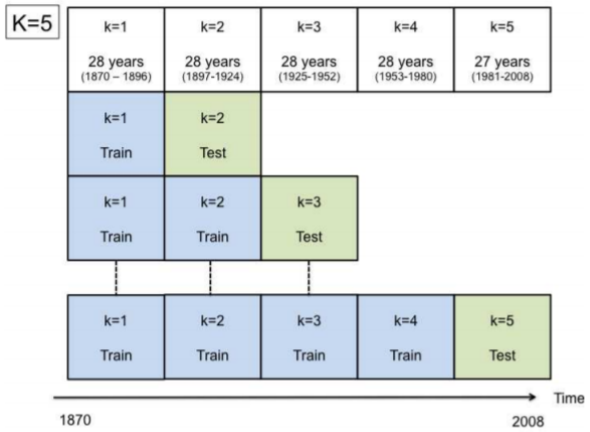
Let us begin by looking at our data. We first load our data into a data frame by using Python module Pandas alongside Numpy to identify the number of observations and features we have. Our data consists of 1946 observations and 17 features, after we excluded years and countries, but included real credit growth and two of its lags for reasons discussed later. Note also that our dataset is highly imbalanced, as only around 4% of the full sample are marked as crisis. We will see later how to deal with this problem by using under-sampling technique. Next, we determine the number of missing values for our features:

Table 1: No of missing values for features

	loans1	bassets2	govass	narrowm	money	gdp	iy	cpi
No of NaNs	243	360	1223	155	204	175	202	70

	stir	ltrate	stocks	CreditGrowth	1yrlag	2yrlag
No of NaNs	433	48	251	298	321	344

Having NaN values in real-world data is pretty typical due to observations that were not recorded or data corruption. The obvious solution is to remove rows with missing values from our data. In our case, this means dropping 1351 observations from our data which is around 69% of our entire dataset. However, doing so can cause a high bias for any models that we use and potentially lead to wrong prediction because of the information loss. We solve the issue by imputing missing values with the mean of respective columns. We then followed the same approach adopted by the project description in the referenced papers that considered real credit growth and its lags as an additional features. Since we are dealing with real-time prediction, we will use time-series split validation which splits our data into K equally-sized blocks, but trains each model using information on previous blocks only (see Figure below for an illustration with K = 5).

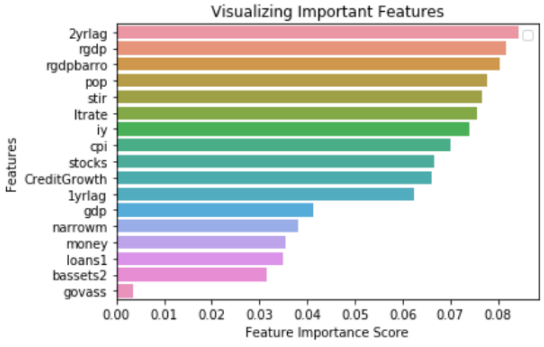


3 Feature Engineering

Firstly, we standardize our input features to obtain the same range of values for all our features before feeding into our model. This is to ensure stability of our model and good accuracy. Following this, we will perform feature selection on our training set to select features to be considered by our model. We will do so via multiple decision trees (random forests). The idea is that we compute how much each feature decreases the Gini impurity when training a tree. The more a feature decreases the impurity, the more important the feature is. In random forests, the impurity decrease from each feature can be averaged across trees (10000 trees in our case) to determine the final importance of the variable. The Gini impurity index is defined as

$$G = \sum_{i=1}^{n_c} p_i(1 - p_i)$$

where n_c is the number of classes in the target variable ($n_c = 2$ in our case) and p_i is the ratio of this class. Random forests are used here because they aggregate many decision trees to limit overfitting that a single decision tree is usually prone to, so they tend to provide a more accurate result and generalize better. In addition, they are easy to interpret. The plot below shows our results:



Our results seem to support the claim in the referenced papers that real credit growth and its second lag features are good predictors of financial crash. The mean of the scores is 0.0588. Consequently, we will neglect all features with scores lower than this amount, and only the remaining features are included in our training dataset. Next, we will compare the resulting performances of different machine learning models before and after feature selection process. The table below shows our results:

4 Machine Learning Models

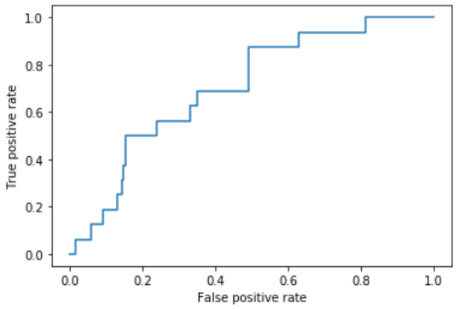
Table 2: Out-of-Sample Performance (AUC) for K = 5

	AUC (Before FS)	AUC (After FS)
Gradient Boosting	0.664	0.734
Logistic Regression	0.493	0.518
Random Forests (RF)	0.488	0.502
Support Vector Machines (SVM)	0.441	0.500

Our results show that there is a small improvement in out-of-sample performance of all our models after feature selection. The traditional approach for classification problem (Logistic regression) can be seen more superior than other complex ML methods such as RF and SVM, so it supports the claims made on the referenced papers. However, the results that we obtained are not very promising as indicated by our models accuracy only slightly better than a coin flip. In fact, Gradient boosting is the only method that stands out among others. We use XGBoost (Extreme Gradient Boosting) classifier here. The principle behind XGBoost is tree ensembles. In our case, the tree ensemble model is a set of classification trees. Trees are grown one after another, and attempts to reduce the misclassification rate are made in subsequent iterations to build a new, stronger model. It has been found that the optimal parameter max depth of the tree is 4 after tuning the XGBoost model. The plot below shows the performance of our XGBoost classifier:

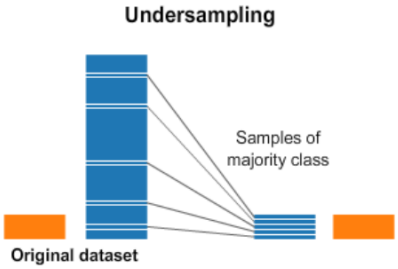
		Predicted	
		0	1
Actual	0	308	0
	1	16	0

Table 3: Confusion Matrix



(Area = 0.704)

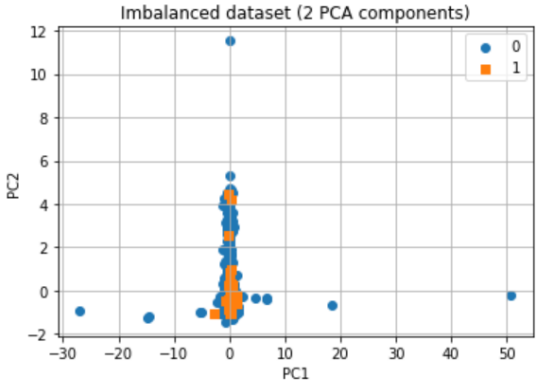
We get quite a good performance on our XGBoost classifier. However this result can be rather misleading because our model simply predicts 0 in all cases. In fact, in a dataset with highly unbalanced classes, if the classifier always "predicts" the most common class ('0' No Crisis) without performing any analysis of the features, it will still have a high accuracy rate, obviously illusory. In our case, we want to be able to accurately predict the minority class ('1' Crisis) as its effect is often more significant. So, we need to think of ways to handle our highly imbalanced data. Fortunately, resampling technique can be adopted to deal with this problem. It consists of removing samples from the majority class (under-sampling) which we will use here.



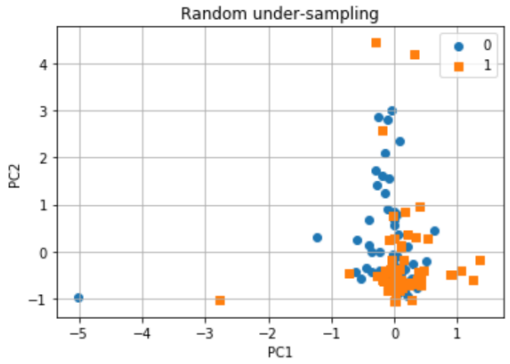
Despite the advantage of balancing classes, this technique also has its weakness (there is a trade-off). The simplest implementation of under-sampling is to remove random records from the majority class, which can cause loss of information. Therefore a more sophisticated resampling technique is needed. For example, we can cluster the records of the majority class, and do the under-sampling by removing records from each cluster, thus seeking to preserve information. Because our dataset has many dimensions (features), we will reduce the size of the dataset using PCA for ease of visualization of our data distribution in 2D.

5 Dimensionality Reduction for Analysis

In this section, we apply PCA on our standardized training dataset to reduce the dimension of our feature space by selecting the most important features that capture maximum information about the dataset. The features are selected based on the variance they account for in the output. The feature that account for highest variance is the first principal component, followed by the second principal component that is responsible for second highest variance and so on. One important note is that principal components are uncorrelated with each other. Let us first measure the variance ratio of the principal components: [0.57790999, 0.27293422, 0.14915578]. We can see that the first 2 components contributes to 85% of the total variance. So it is good enough to choose only 2 components to plot our classes. The plot is shown below:



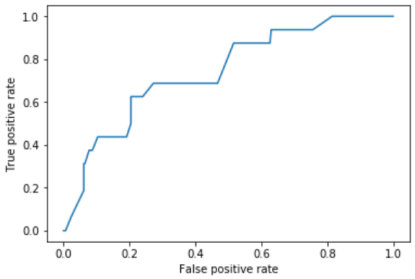
This plot reveals an ineffectiveness of all our models apart from XGBoost. It is clear from the plot that the orange squares (denoted by financial crash) are embedded in the the blue circles (denoted by non financial crash). This implies that our models are not capable of distinguishing between the classes because our data is not easily separable. Therefore, it is probably reasonable to say that this trend involves higher dimensions and requires non-linear kernels in order to solve this issue. However, the problem with PCA is that it only captures the linear relationship of features so that this plot could be an unrealistic depiction of our dataset. Nevertheless, it provides an useful insight into our dataset. Now, we will perform random under-sampling to remove records from the majority class as we said earlier. The plot below shows our results:



From the plot, we can clearly see that our data is balanced now with the same number of counts of both classes. Finally, we can fit our favourite model XGBoost classifier to the outputs from PCA in order to make predictions on our test set. Lets see if our model is doing a good job:

		Predicted	
		0	1
Actual	0	245	63
	1	6	10

Table 4: Confusion Matrix



(Area = 0.741)

We get a reasonable AUC score of 0.741 which is slightly higher than the value before random under-sampling. Note that since the sampling is done randomly, the score may deviate by a little if we repeat the same procedure again. More importantly, our classifier is now able to predict financial crash more accurately (10 correct out of 16) on our test set (from year 1981-2008). Despite the fact that the number of false positives has increased, we are not too concerned about this because it is always better to predict ('1' Crisis) in advance even if it did not happen in the end rather than failing to predict ('1' Crisis) when it actually happens.

6 Conclusion

It is found that XGBoost performed the best among other models for predicting financial crash after resampling our training dataset. In fact, most competitions like Kaggle have been won using gradient boosting trees. It was no surprise because XGBoost used a more regularized model formalization to control over-fitting, which boosts its performance. However, we can obtain even higher accuracies by using a model which can take sequences of data points (in time-series format) as an input to make a prediction rather than using individual date. An alternative or perhaps better approach is to implement a Long Short Term Memory (LSTM) recurrent neural network. In this case, we would attempt to capture the long-term dependencies by looking back more into the past data in order to make an accurate prediction about crashes in the future. Although this approach has the main drawback that networks are very expensive to train as they normally involve many parameters.