Loan Default Prediction: A Machine Learning approach

Vasu Gupta, Xavier Kidston

MacEwan University

1. Introduction

In today's world, there is a huge increase in the financial risks with a rapid growth in the financial industry [1]. Out of all the money banks lose, research shows that 80% of the loss occurs from the customer credit risk [1]. Due to customer credit risk being the biggest factor of losses, banks use various systems to identify the credit default. However, most systems do not include machine learning and based on financial calculations. Also, a common man does not have access to such systems and goes through a long process of loan approval from the bank.

This project aims to create a machine learning system which a common man or an organization can use to determine the result of a loan application. This system can also be used by the financial institutions to run preliminary tests to determine the loan application results instantaneously. The loan result can be calculated by having the user enter their personal information like annual income, loan amount, education status etc. The use of machine learning for accomplishing the complex task of loan prediction makes it easier for users to discover the results while saving time.

The loan default prediction is a type of supervised classification problem which uses 0 and 1 as labels to indicate the loan default or non-default respectively. The data includes over 300,000 entries of loan applications with its corresponding success or failures. As shown in Fig 1.1, the dataset contains 122 columns which include the features like income, age, number of children etc. leading to the prediction.

SK_ID_CUI	TARGET	NAME_CO	CODE_GEI	FLAG_OW	FLAG_OW	CNT_CHIL	AMT_INCC	AMT_CRE	AMT_ANN	AMT_GOC	NAME_TYI	NAME_IN	NAME_ED	NAME_FAI	NAME_HC	REGION_P	DAYS_BIR	DAYS_EMI	DAYS_REG	DAYS_ID_I	OWN_CAF	FLAG_MOI	FL
100002	1	Cash loans	M	N	Y	0	202500	406597.5	24700.5	351000	Unaccomp	Working	Secondary	Single / no	House / ap	0.018801	-9461	-637	-3648	-2120		1	
100003	0	Cash loans	F	N	N	0	270000	1293503	35698.5	1129500	Family	State serv	Higher edu	Married	House / ap	0.003541	-16765	-1188	-1186	-291		1	
100004	0	Revolving	M	Y	Y	0	67500	135000	6750	135000	Unaccomp	Working	Secondary	Single / no	House / ap	0.010032	-19046	-225	-4260	-2531	26	1	
100006	0	Cash loans	F	N	Y	0	135000	312682.5	29686.5	297000	Unaccomp	Working	Secondary	Civil marri	House / ap	0.008019	-19005	-3039	-9833	-2437		1	
100007	0	Cash loans	M	N	Y	0	121500	513000	21865.5	513000	Unaccomp	Working	Secondary	Single / no	House / ap	0.028663	-19932	-3038	-4311	-3458		1	
100008	0	Cash loans	M	N	Y	0	99000	490495.5	27517.5	454500	Spouse, pa	State serv	Secondary	Married	House / ap	0.035792	-16941	-1588	-4970	-477		1	
100009	0	Cash loans	F	Y	Y	1	171000	1560726	41301	1395000	Unaccomp	Commerci	Higher edu	Married	House / ap	0.035792	-13778	-3130	-1213	-619	17	1	
100010	0	Cash loans	M	Y	Y	0	360000	1530000	42075	1530000	Unaccomp	State serv	Higher edu	Married	House / ap	0.003122	-18850	-449	-4597	-2379	8	1	
100011	0	Cash loans	F	N	Y	0	112500	1019610	33826.5	913500	Children	Pensioner	Secondary	Married	House / ap	0.018634	-20099	365243	-7427	-3514		1	
100012	0	Revolving	M	N	Y	0	135000	405000	20250	405000	Unaccomp	Working	Secondary	Single / no	House / ap	0.019689	-14469	-2019	-14437	-3992		1	
100014	0	Cash loans	F	N	Y	1	112500	652500	21177	652500	Unaccomp	Working	Higher edu	Married	House / ap	0.0228	-10197	-679	-4427	-738		1	
100015	0	Cash loans	F	N	Y	0	38419.16	148365	10678.5	135000	Children	Pensioner	Secondary	Married	House / ap	0.015221	-20417	365243	-5246	-2512		1	
100016	0	Cash loans	F	N	Y	0	67500	80865	5881.5	67500	Unaccomp	Working	Secondary	Married	House / ap	0.031329	-13439	-2717	-311	-3227		1	
100017	0	Cash loans	M	Y	N	1	225000	918468	28966.5	697500	Unaccomp	Working	Secondary	Married	House / ap	0.016612	-14086	-3028	-643	-4911	23	1	
100018	0	Cash loans	F	N	Y	0	189000	773680.5	32778	679500	Unaccomp	Working	Secondary	Married	House / ap	0.010006	-14583	-203	-615	-2056		1	

Figure 1.1 Representing the 24/122 features from the original dataset

The proposed system will help the banks generate more revenue, and lowers the loan processing time for the individuals and organizations. Above all, this project will also give us an insight on the factors responsible for making a major change in the loan decisions. A similar system created by Equifax, a United States based credit bureau which assesses the loan validity, and found a 15% increase in the predictive ability of their machine learning models [2]. Moreover, companies like Upstart have also started looking into variables like SAT scores, GPA, or educational attainment of the applicant for a better prediction of the results [2].

The major challenge with the data is the imbalancement in classes, where the minority class of the loan defaulters is about 8% of the entire dataset. Therefore, multiple techniques will be used to tackle the issue of data imbalancement.

2. Methods

2.1 Dataset

The dataset that was utilized for default prediction consisted of over 300,000 entries, with only around 8% being people who didn't pay successfully for their loans. The dataset includes 122 features like: loan amount, annual income, age, number of children etc. which help to predict the output. As shown in Fig 1.1, the source of the data is the Credit Card Fraud Detection dataset found on kaggle.

2.2 Data Preprocessing

The data preprocessing involved the removal of features having greater than 56% of missing values in the columns. The cutoff for removing the excess missing data was arbitrarily chosen. The rest of the missing entries in the dataset were replaced with the mean of their respective columns. As outliers can affect the training of algorithms, they were removed from the dataset to make valid predictions. At the end of data preprocessing, label encoding was performed to convert the text data to model understandable numerical data. The label encoding was performed using scikit-learn's preprocessing library. The data extracted after the preprocessing stage is called Filtered Data. The Filtered Data contains 99 features out of the total 122 features.

2.3 Data Imbalancement Handling

Most Machine Learning algorithms perform the best when the classes in the dataset are about the equal ratio. However, the Default Prediction dataset just had 8% minority class data and 92% majority class data.

There were three data imbalancement approaches followed in this project.

2.3.1 Undersampling

Undersampling is a method of removing entries from the majority class to make it equal to the size of the minority class. As a result of removal of data entries, the resultant data set contains equal entries from majority and minority classes. This method is usually a good choice when we have a huge dataset [3]. However, there is a potential of removing valuable data which might lead to poor generalization to the test dataset [3].

In this technique, undersampling is performed on the entire dataset and then further divided into test and train dataset. Test dataset size was chosen to be 33% and the train dataset was 67%.

2.3.2 Oversampling- Duplication

Oversampling is a method of increasing the size of a minority class by making it equal to the majority class size. In this technique, we create duplicates of the minority class to make it bigger [3]. This technique is usually performed when the dataset is not huge, but it does not hurt to experiment and analyse the results.

When performing oversampling, the first step we took was to split the dataset into train and test data in a 3:1 ratio. The initial data split is to avoid the memorization of data points in the test set which could lead to an overfit. To keep the test data generalized, we split the data before. Next, the data points of the minority class are duplicated to make them equal to the size of the majority class.

2.3.3 Oversampling- Synthetic Data

Generation of synthetic data is a type of oversampling technique which allows to create new data from the minority class to make it equal to the size of the majority class. In theory, this approach is better than creating duplicates of the minority class which will avoid the memorization of specific data points by the model.

Similar to the previous oversampling technique, firstly, the data is divided into train and test sets in a 3:1 ratio. To perform oversampling, we use SMOTE, an oversampling tool to generate new data points of minority class. The new data is created by using k-nearest algorithm.

2.4 Machine Learning Models

There were three machine learning models used for the prediction of loan defaults. Before training any models, feature scaling was always performed to normalize the data. The feature scaling was done using scikit-learn's StandardScaler implementation. The training time for each model is calculated and displayed. All models are evaluated by using metrics like accuracy, precision, recall, f1 score, confusion matrix and learning curves.

2.4.1 Random Forest (RF)

The Random Forest model was created by using the scikit-learn implementation of RandomForestClassifier. The model is originally created with the number of estimators as 1000 and the maximum tree depth as 10. As shown in Fig 2.1, the model has a depth of 5, but it was calculated by tuning the hyper parameter depth from 10 to 5 in several iterations while retaining the evaluation metrics and lowering the training time.

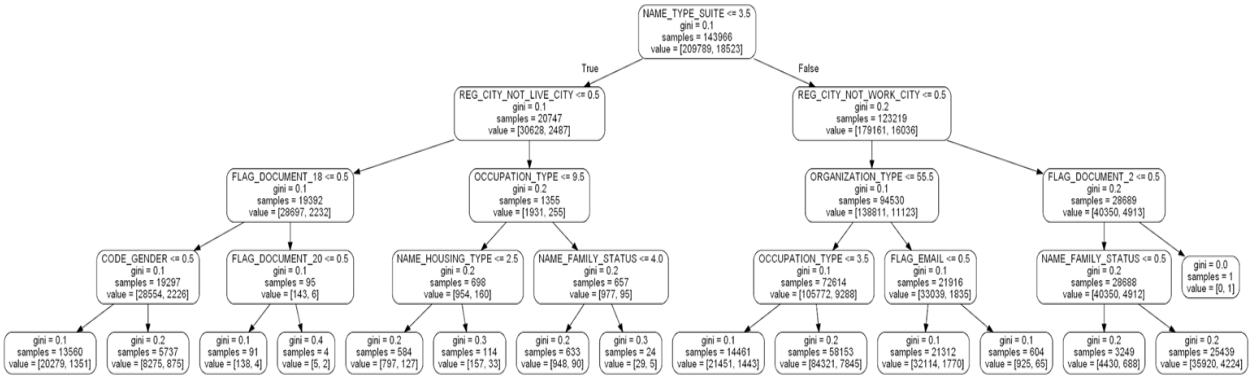

Figure 2.1 Random Forest Tree of depth 5

2.4.2 Logistic Regression (LR)

The Logistic Regression model was created by using the scikit-learn implementation of LogisticRegression. This model is trained with the default parameters with a random state of 27. For each prediction there is a sigmoid function where $g(z) = \frac{1}{1+e^{-z}}$ and then we would get $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$ using $h_\theta(x) = \frac{1}{1+e^{-x\theta}}$ to get it. This process will give the probability of a true value and assign it to either a 0 or 1 dependent on which is closest as this is a binary classification problem.

2.4.3 Support Vector Machine (SVM)

The Linear SVM model was created by using the scikit-learn implementation of svm.LinearSVC. The model was trained on the regularization parameter as 1 with 1000 maximum iterations.

2.5 Principal Component Analysis (PCA)

PCA was implemented by computing the covariance, singular value decomposition, and projecting the scaled data to a given dimensionality. To reduce the data dimensionality, PCA was originally implemented to project the number of features to 30.

2.6 Important Features

To exact the features which hold some weight in the prediction, the importance function in the scikit-learn's Random Forest implementation and the coef function in the scikit-learn's Logistic Regression were used. When a feature had no weight in prediction using RF or LR, they were removed from the dataset.

By extracting the data of important features from the filtered data, the number of important features came down to 46 from 99. 53 features out of 99 features were found unnecessary and had no weight in the prediction of the results.

2.7 Flow of Data

The techniques and the models were implemented in the project in various ways. RF, LR and SVM models were created for every technique implemented. Fig 2.2 represents the preprocessing, the methods and the models used in the application. To have a fair comparison between the methods not dealing with data imbalancement, the test and train dataset was divided by scikit-learn's StratifiedShuffleSplit class. The test and train data was divided in 1:3 ratio with 3 splits for data shuffling. For all the methods like PCA, Using Important Features and No Method (using Filtered Data), the same instance of the object of StratifiedShuffleSplit class was used. By using the same instance, the test and train indices remain the same for all the methods.

However, for techniques such as Oversampling, Undersampling and Synthetic Data generation, the test and train datasets for the models cannot be the same as these techniques generate or remove data from the original dataset.

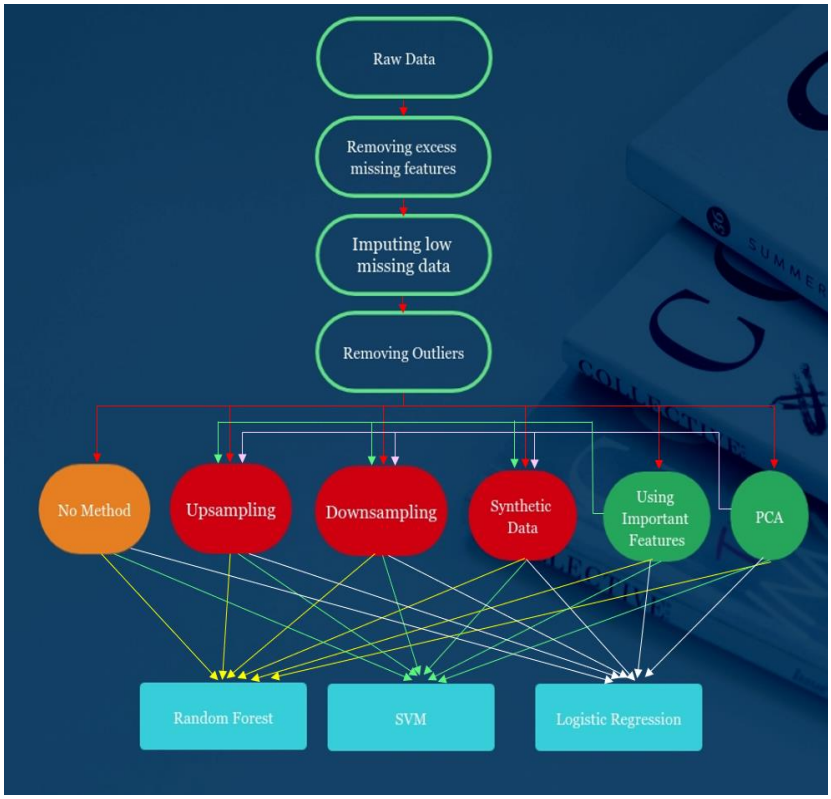

Figure 2.2 Flow chart representing all the methods implemented and models created

As Fig 2.2 shows, the filtered data after preprocessing is passed through multiple techniques and then the models are trained at the end. The approaches taken are as follows:

- Applying no techniques on the filtered data.
- Pass the filtered data to perform data oversampling by adding duplicates of the minority class.
- Pass the filtered data to perform data undersampling by removing random entries from the majority class.
- Pass the filtered data to perform data oversampling by generating new data points of the minority class.

- Pass the filtered data to run RF and LR models and identify the important features. The new important data extracted by using this technique.
- Pass the filtered data through PCA for reducing data dimensionality.
- Pass the filtered data to run RF and LR models and identify the important features. Now the important data is fed as input to perform the oversampling by duplication.
- Pass the filtered data to run RF and LR models and identify the important features. Now the important data is fed as input to perform the undersampling.
- Pass the filtered data to run RF and LR models and identify the important features. Now the important data is fed as input to perform the oversampling by synthetic data generation.
- Pass the filtered data through PCA for reducing data dimensionality. The reduced data is fed as input to perform oversampling by duplication.
- Pass the filtered data through PCA for reducing data dimensionality. The reduced data is fed as input to perform undersampling.
- Pass the filtered data through PCA for reducing data dimensionality. The reduced data is fed as input to perform oversampling by synthetic data generation.

After applying all these techniques, the final data is used to create Random Forest, Logistic Regression and Support Vector Machine models. After the model creation, each model is evaluated based on the metrics like accuracy, precision, recall, f1 score, confusion matrix and learning curves. All these evaluation metrics are implemented using scikit-learn's implementation of the libraries.

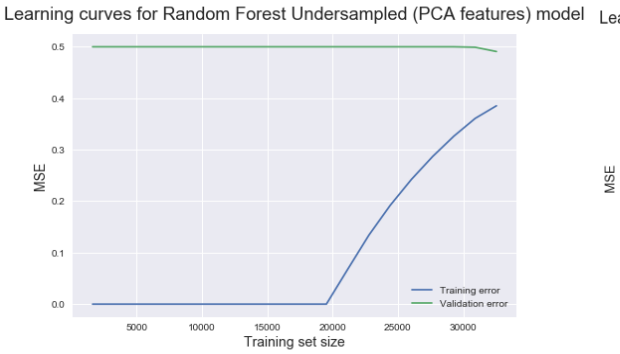

Figure 2.3

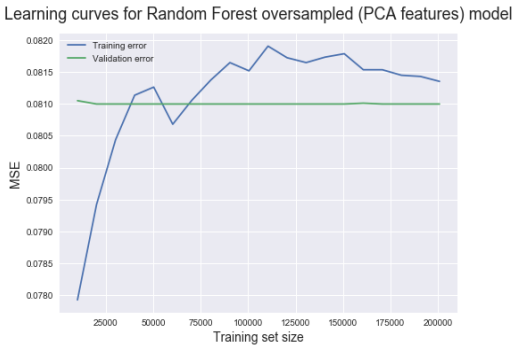

Figure 2.4

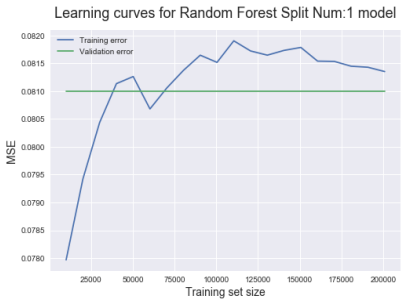

Figure 2.5

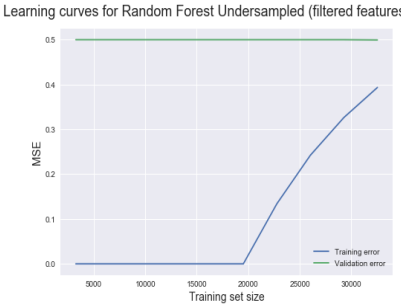

Figure 2.6

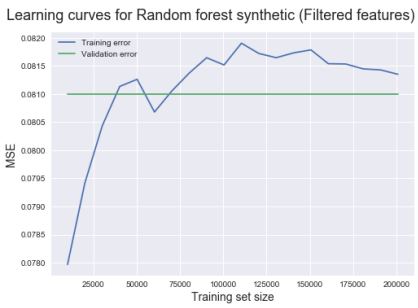

Figure 2.7

2.8 Parameter tuning and analysis

To start off with, Random Forest had a maximum tree depth as 10 and 1000 estimators as the hyperparameters. The training time was found to be very large with these hyperparameters. By following an iterative approach after applying PCA or using Important Data, the optimal numbers for tree depth and number of estimators were found as 1 and 1 respectively. When the RF model was created using PCA, and important data, training time was significantly reduced from 150 seconds to 0.09 seconds, while maintaining the evaluation metrics. By using an iterative approach, the optimal parameters for oversampling technique by data duplication were found to be 3 estimators and a tree depth of 3. The optimal number of estimators for oversampling by synthetic data were 7 and the tree depth was 3. Using the undersampling technique, the optimal tree depth was 4 and the number of estimators was 15. Each node in the tree (Fig 2.1) has its associated gini value which is $\sum_{i=1}^{c} f_i(1 - f_i)$. Where f_i is the frequency of the label at i, and c is the amount of unique labels. The learning curves in the Fig 2.3- 2.7 shows the issue of underfitting even after the process of hyperparameter tuning.

The Logistic Regression model had no hyperparameter tuning implemented. When training the Logistic Regression Model, we get an underfit through every different dataset for logistic regression as shown by Fig 2.8-2.11. The data seemed to follow a more anomaly-based pattern rather than complex disbursements.

Due to the high training time (in hours) of SVM models, the hyperparameter tuning using an iterative approach was tough to implement. When training the SVM Model, it results in an underfit for all the different datasets as shown in Fig 2.12 - 2.15. We were forced to skip learning curves for over sampling techniques for SVM as it was computationally expensive and highly time consuming. Also, due to the high training time (in hours) of SVM models for oversampling techniques, their results are not added in Fig 3.1 as their complete training did not take place.

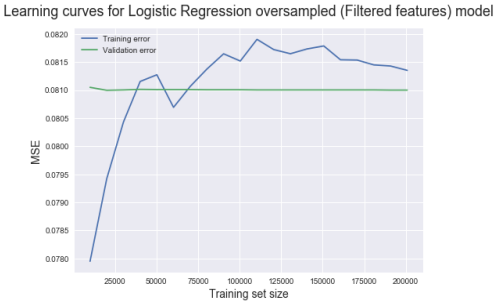

Figure 2.8

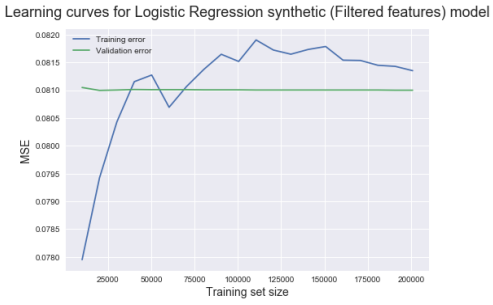

Figure 2.9

Figure 2.12

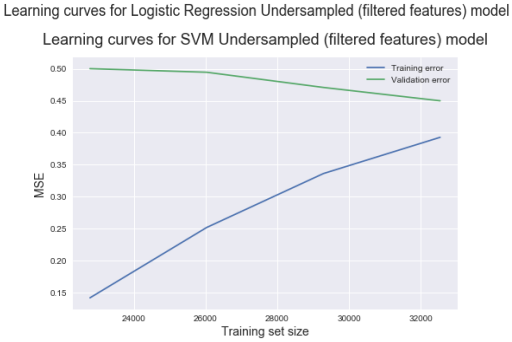

Figure 2.13

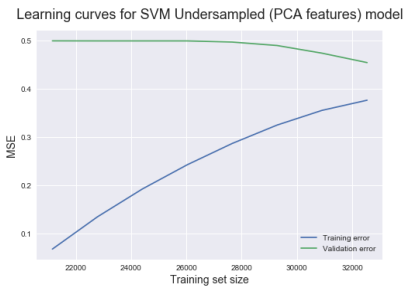

Figure 2.14

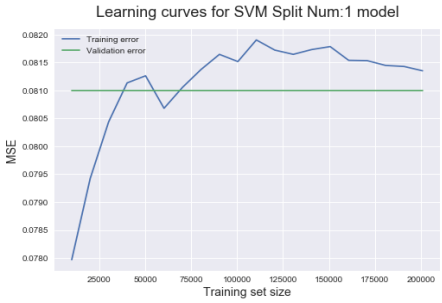

Figure 2.15

2.9 Implementation

Apart from the implementation details mentioned in the respective sections, there were a few other libraries leveraged in the project. Pandas was used for storing and formatting the dataset in an organized form. Numpy, Scipy and Matplotlib were used for computation and graphing. Imblearn was used for SMOTE. The intuition behind the dataset was gathered from the Kaggle kernels.

3. Results

Based on the results in Fig 3.1, the best result (around 92%) is given by using the Random Forest model when using Filtered Data with no techniques performed. Also, the worst performing technique was oversampling in most cases with the lowest accuracy as 46.3%.

Most of the techniques had similar results for all the models implemented. This leads us to interpret that the data fed to the models had deficiencies which sometimes led to poor results. For example: Filtered data, PCA, undersampling on filtered data, oversampling on filtered data, undersampling on important data, oversampling on important data, synthetic data generation on important data, undersampling on PCA reduced data, synthetic data generation on PCA reduced data - had similar results for all the models. Therefore, there was a need of trying more techniques or improving the existing ones.

Based on the current task of predicting loan defaults, Precision is a more suitable metric as compared to the recall. This is because a financer will prefer having low false positives in order to save money. However, the results in Fig 3.1 show that all the evaluation metrics for each case are equal. As $precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN}, f_1 = \frac{2TP}{2TP+FP+FN}$ or $f_1 = precision + recall$, all the metrics fall under the unique case of the false positives being equal to the false negatives. When the FP = FN, then all the metrics are equal.

	Random Forest	Logistic regression	SVM
Filtered Data			
Accuracy	0.919	0.918	0.918
Precision	0.919	0.918	0.918
Recall	0.919	0.918	0.918
f1	0.919	0.918	0.918
Time to complete (in sec)	0.09	1.98	82
PCA			
Accuracy	0.918	0.919	0.919
Precision	0.918	0.919	0.919
Recall	0.918	0.919	0.919
f1	0.918	0.919	0.919
Time to complete (in sec)	0.07	0.12	31
Under Sampled + Filtered Features			
Accuracy	0.58	0.58	0.58
Precision	0.58	0.58	0.58
Recall	0.58	0.58	0.58
f1	0.58	0.58	0.58
Time to complete (in sec)	0.14	0.56	9.5
Over Sampled + Filtered Features			
Accuracy	0.58	0.578	NA
Precision	0.58	0.578	NA
Recall	0.58	0.578	NA
f1	0.58	0.578	NA
Time to complete (in sec)	0.65	2.34	NA
Synthetic Data + Filtered Features			
Accuracy	0.569	0.688	NA
Precision	0.569	0.688	NA
Recall	0.569	0.688	NA
f1	0.569	0.688	NA
Time to complete (in sec)	1.123	3.588	NA

Under Sampled + Important Features			
Accuracy	0.53	0.53	0.53
Precision	0.53	0.53	0.53
Recall	0.53	0.53	0.53
f1	0.53	0.53	0.53
Time to complete (in sec)	0.02	0.009	1.04
Over Sampled + Important Features			
Accuracy	0.506	0.507	NA
Precision	0.506	0.507	NA
Recall	0.506	0.507	NA
f1	0.506	0.507	NA
Time to complete (in sec)	0.06	0.12	NA
Synthetic Data + Important Features			
Accuracy	0.507	0.507	NA
Precision	0.507	0.507	NA
Recall	0.507	0.507	NA
f1	0.507	0.507	NA
Time to complete (in sec)	0.065	0.119	NA
Under Sampled + PCA			
Accuracy	0.58	0.58	0.58
Precision	0.58	0.58	0.58
Recall	0.58	0.58	0.58
f1	0.58	0.58	0.58
Time to complete (in sec)	1.62	0.15	5.99
Over Sampled + PCA			
Accuracy	0.463	0.572	0.561
Precision	0.463	0.572	0.561
Recall	0.463	0.572	0.561
f1	0.463	0.572	0.561
Time to complete (in sec)	0.909	1.79	22.25
Synthetic Data + PCA			
Accuracy	0.558	0.571	0.57
Precision	0.558	0.571	0.57
Recall	0.558	0.571	0.57
f1	0.558	0.571	0.57
Time to complete (in sec)	10.264	2.75	26.18

Figure 3.1 Results of all the techniques performed

Moreover, all the data imbalancement techniques had a few limitations which made the models perform poorly. Out of all the imbalancement techniques, the upsampling of minority class by generation of synthetic data performs the best with 68% accuracy. The data undersampling was performed to remove the entries from the majority class to make the size of both classes equal. However, the removal of entries was purely random which made us lose some valuable data. Techniques like Tomek links, Condensed Nearest Neighbour Rule and One-sided selection remove redundant or noisy data from the majority class [4]. Other undersampling techniques like EasyEnsemble trains models using different subsets of the majority class and adds the outputs of the models, are also found to be effective [5]. The oversampling technique by duplication creates redundant data in the dataset which makes the model prone to overfitting and makes poor generalization of the data. In our case, overfitting did not occur, but poor generalization of data definitely took place. The usage of SMOTE for upsampling the data does not handle the datasets with nominal features [4]. SMOTE is generalized to deal with mixed datasets of nominal and continuous features [4]. A few improvements in SMOTE have been made to account for the nominal features. These techniques are called SMOTE-NC (Synthetic Minority Over-sampling Technique Nominal Continuous) and SMOTE-N (Synthetic Minority Over-sampling Technique Nominal) [4]. Other improvements in SMOTE allow oversampling on the borderline between majority and the minority classes which are called borderline-SMOTE1 and borderline-SMOTE2 [6].

These techniques would have been a good direction to proceed, but the time constraints did not allow us to move in that direction.

4. Conclusion

In this project, there were multiple techniques implemented to create Random Forest, Logistic Regression and Support Vector Machine models for the prediction of load defaults. After going through the process of creating a machine learning system for loan default prediction, we also identified a need of major improvement in other real-life systems. There was a high imbalance in the dataset which was a major challenge to solve. There were 3 techniques implemented to solve the data imbalancement which performed poorly. There was a trend of underfitting found in all the models even after tuning the models. For most cases, the Random Forest model performed the best out of all models; however this can vary with more nuanced differences in features. Future work would include trying improved methods to tackle the data imbalancement and techniques like Anomaly Detection.

References

[1] Hsu, T. C., Liou, S. T., Wang, Y. P., & Huang, Y. S. "Enhanced Recurrent Neural Network for Combining Static and Dynamic Features for Credit Card Default Prediction" In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 2019, pp. 1572-1576. IEEE.

[2] D. Fagella, "Artificial Intelligence Applications for Lending and Loan Management", *Emerj,* April 3, 2020 [Online]. Available: emerj.com/ai-sector-overviews/artificial-intelligence-applications-lending-loan-management/ [Accessed April 13, 2020].

[3] Boyle, T, "Dealing with Imbalanced Data", *Towards Data Science,* Feb 3, 2019 [Online]. Available: towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18 [Accessed April 6, 2020].

[4] Ganganwar, V. "An overview of classification algorithms for imbalanced datasets" in *International Journal of Emerging Technology and Advanced Engineering*, 2012, *2*(4), pp.42-47.

[5] Liu, X. Y., Wu, J., & Zhou, Z. H. "Exploratory undersampling for class-imbalance learning", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 2008, *39*(2), 539-550.

[6] Han, H., Wang, W. Y., & Mao, B. H. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning" In *International conference on intelligent computing,* Springer, Berlin, Heidelberg, 2005, pp. 878-887.