2016

B659 – Applied Machine Learning
Final Project Report
Credit Card Defaulter and Fraud Transaction Analysis

PRASHANTH BALASUBRAMANI(PRASBALA)
SIDDHARTH JAYASANKAR(SIDJAYAS)

APPLIED MACHINE LEARNING

Abstract: -

The goal of this project is to apply machine learning techniques to solve problems related to credit card usage. We also have analyzed the performance of different machine learning algorithms in the domain. In this project, we have used machine Learning techniques to 2 different problems related to credit cards.

1) To predict if a credit card transaction is a fraudulent transaction or not.
2) To predict if a credit card user would be a defaulter (miss to pay his credit card bills on time) or not.

We have also tried to analyze how under sampling (having a balanced training set) and feature reduction has an impact on the performance of the machine learning algorithms.

Motivation: -

Almost every developing and developed country of world are moving towards digital currency. This change could potentially lead to increase in fraudulent transactions. Added to this, people would also exploit the banks by taking large loans / credits and defaulting with their payments. If machine learning could be used in this domain to help detect fraudulent transactions and people who would be more likely to default their payments, then the process of moving towards digital currency could gather speed.

Thus, through this project we would like to explore how effective machine learning techniques could help detect credit card frauds and credit card payment defaults.

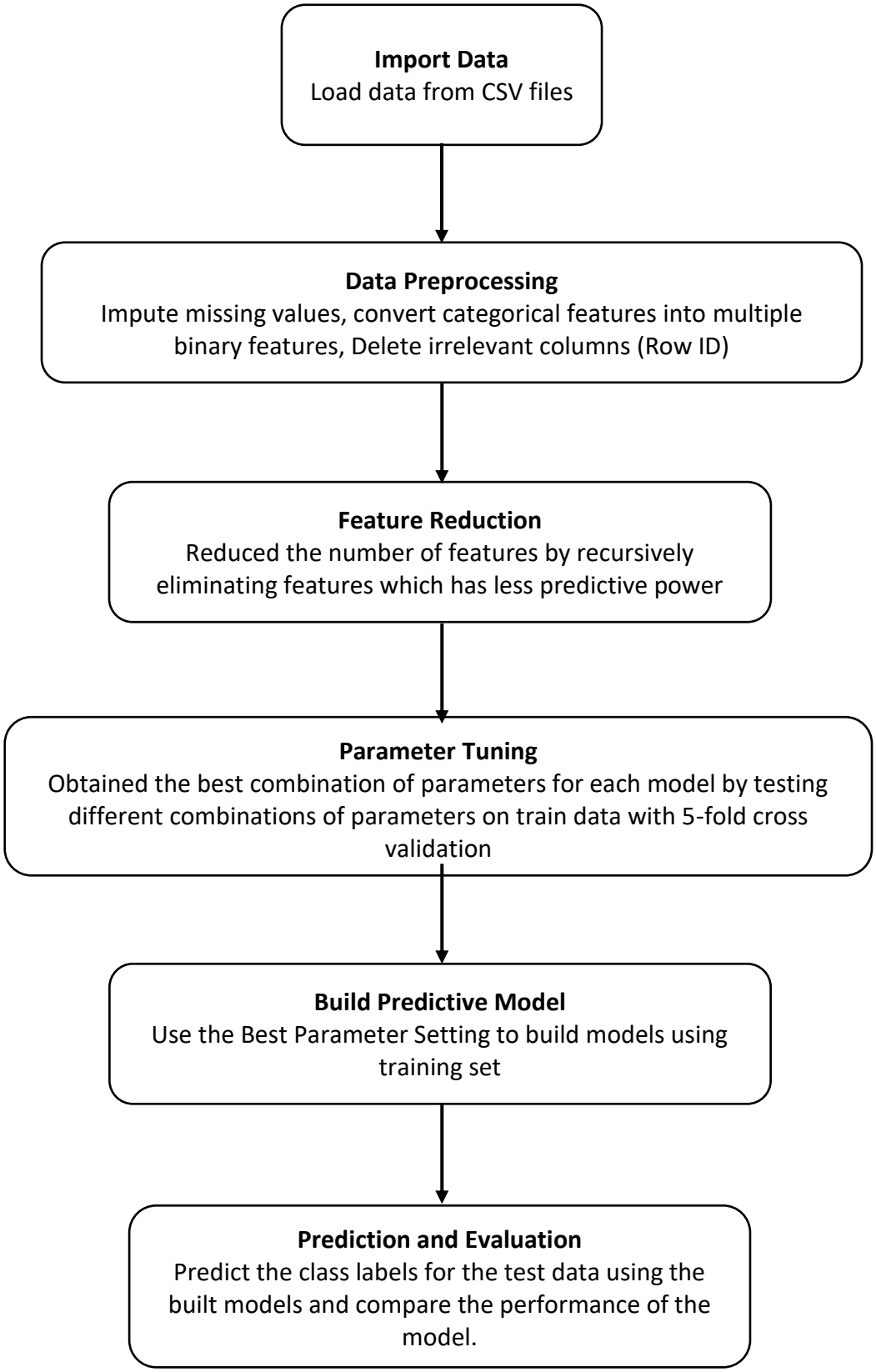
Data Set: -

Fraud Detection: **-** We have used a data set from Kaggle for this task. The dataset presents transactions that occurred in 2013 by European card holders. The dataset contains 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. The dataset contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, original features were not provided.

Payment Defaulter Detection: **-** We have used a data set from UCI machine learning repository for this task. This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. This data set is also unbalanced, with the positive class (defaulters) accounting for 22.22% of all transactions.

Overview of Methdology: -

All the below mentioned processes were performed in python using pandas, sklearn and numpy packages.



Task 1: - Fraud Detection

Data Preprocessing: - Since all the features were numeric and there were no missing data, no data preprocessing was performed.

Feature Selection: - The data set already contained PCA transformed features. So, no additional feature selection and feature engineering techniques were applied to the data.

Sampling: - Since the data set is highly imbalanced, we split the data into train and test sets using 2 different techniques and compared the performance of the models on these 2 sets of data

Random Sampling: The data was split into train and test sets by randomly sampling data. Due to this the class imbalance was still preserved in the training set. (we had chosen to work with smaller train set because if we split the data in 60-40 ratio, the different processes were taking long time and caused performance issues on our personal computers)

	Negatives	Positives
Train set	30000	52
Test set	254751	434

Under Sampling: The data was split into train and test sets in such a way that the classes were made balanced in the training set.

	Negatives	Positives
Train set	200	200
Test set	165907	291

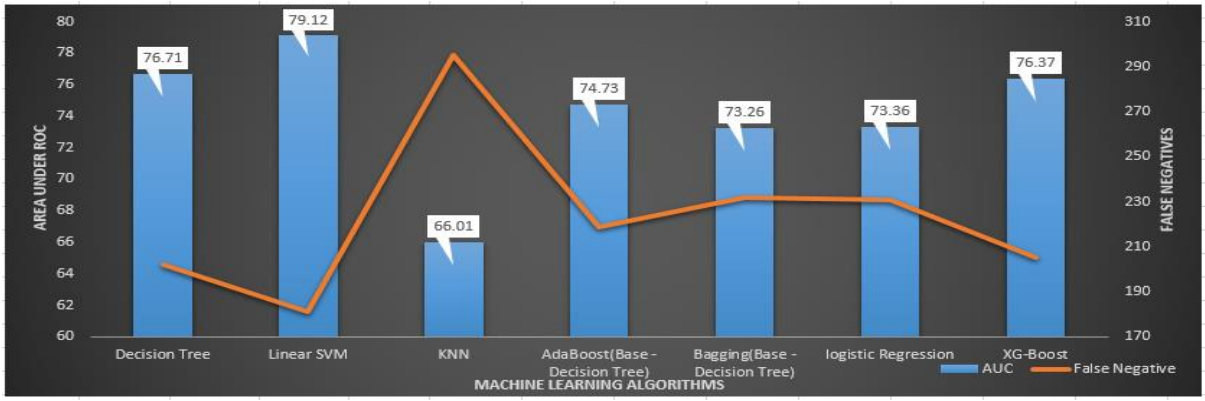
Parameter Tuning: - For each model, the parameters, and the values to consider were decided. Then models were built using all possible combinations of parameters and the model was evaluated using 5-fold cross validation of the train set. Then the best set of parameters were chosen based on the result of the 5-fold cross validation.

Model Building: - The machine learning models were built using the best combination of parameters obtained from the previous step. Decision Tree, KNN, Linear SVM, Logistic Regression, Adaboost(Base Classifier-Decision Tree), Bagging(Base Classifier -Decision Tree) and XGBoost models were built.

Evaluation: - Metrics such has AUC, Accuracy, Precision, Recall and F1-Score were calculated. However due to the high class imbalance present in the data we compared models based on AUC. Also in this task, it is very important to identify the fraud transactions correctly. Thus, the cost of false negatives is very high. Hence, we have considered the number of false negatives as a major evaluation metric.

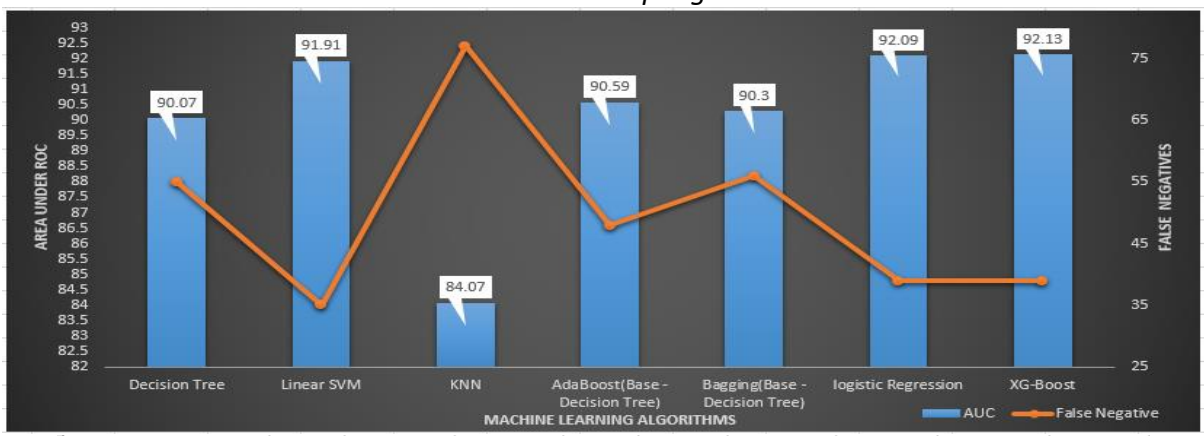
Results: -

Random Sampling



ML Algorithm	AUC	Accuracy	Precision	Recall	F1 - Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	76.71	99.89	100	100	100	254245	71	202	232
Linear SVM	79.12	99.89	100	100	100	254218	98	181	253
KNN	66.01	99.88	100	100	100	254309	7	295	139
AdaBoost(Base - Decision Tree)	74.73	99.84	100	100	100	254135	181	219	215
Bagging(Base - Decision Tree)	73.26	99.89	100	100	100	254290	26	232	202
logistic Regression	73.36	99.86	100	100	100	254214	102	231	203
XG-Boost	76.37	99.9	100	100	100	254286	30	205	229

Under Sampling



ML Algorithm	AUC	Accuracy	Precision	Recall	F1 - Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	90.07	98.95	100	99	99	164228	1678	55	237
Linear SVM	91.91	95.79	100	96	98	158959	6947	35	257
KNN	84.07	94.48	100	94	97	156820	9086	77	215
AdaBoost(Base - Decision Tree)	90.59	97.59	100	98	99	161963	3943	48	244
Bagging(Base - Decision Tree)	90.3	99.53	100	100	100	165552	354	56	236
logistic Regression	92.09	97.53	100	98	99	161840	4066	39	253
XG-Boost	92.13	97.61	100	98	99	161983	3923	39	253

Analysis and Inference: -

From the results, we can see that under sampling the data increases the performance considerably on all the algorithms. Even the performance of simple nonlinear classifiers like KNN and Decision tree increases considerably. This clearly shows that in case of imbalanced datasets, under sampling the data helps build better models.

Of all the algorithms, KNN performed the worst. This is also due to the high-class imbalance in the data. Each test data point often found itself surrounded by negative class data points from the train data. This is probably the reason why even a reasonable value of K=10 yielded bad performance of 66% AUC.

Another interesting observation is that, for random sampled data both ensemble algorithms Adaboost and Bagging decreased the AUC value when compared to decision tree. This gives us an interesting insight that ensemble algorithms are a bad choice for highly imbalanced data sets. The ensemble algorithms do well in predicting the major class better than the minor class when dealing with imbalanced data.

For random sampled train data, Linear SVM performed the best. This gives us insight into the data that despite the huge imbalance in the data the data is mostly linearly separable. The reasonable performance of Decision tree also shows that the fraud transactions can be fairly detected by a set of rules which can be easily interpreted.

For Randomly sampled train data SVM performed better than XGBoost which is currently regarded as the best algorithm to implement in kaggle challenges. This clearly show that even the best algorithm needs good quality data to perform better. For under sampled data XGBoost emerged as the winner. This again emphases the previous statement that even good algorithms need good quality data to provide good performance.

Even though XGBoost has the best performance for under sampled data, it is SVM that has the least number of False negatives. Coupling this with the performance of SVM on random sampled data, we can conclude that SVM is the best possible algorithm to apply for this task.

Task 2: - Credit Card Payment Default Detection

Data Preprocessing: **-** Missing data were imputed with mean for numerical features and mode for categorical features. The categorical features were converted to series of binary features.

Feature Selection: - The features were ranked using the method of recursive feature elimination. The process works by eliminating the least important feature in each iteration. The features were ranked based on their importance. The number of features were reduced to 1/3rd. Thus 23 features were reduced to 8 features by taking the top 8 ranked features. The

Sampling: **-** Since this data set is also imbalanced, we split the data into train and test sets using 2 different techniques and compared the performance of the models on these 2 sets of data

Random Sampling: The data was split into train and test sets by randomly sampling data in 60-40 ratio. Due to this the class imbalance was still preserved in the training set.

Under Sampling: The data was split into train and test sets in such a way that the classes were made balanced in the training set.

	Negatives	Positives
Train set	2000	2000
Test set	21365	4635

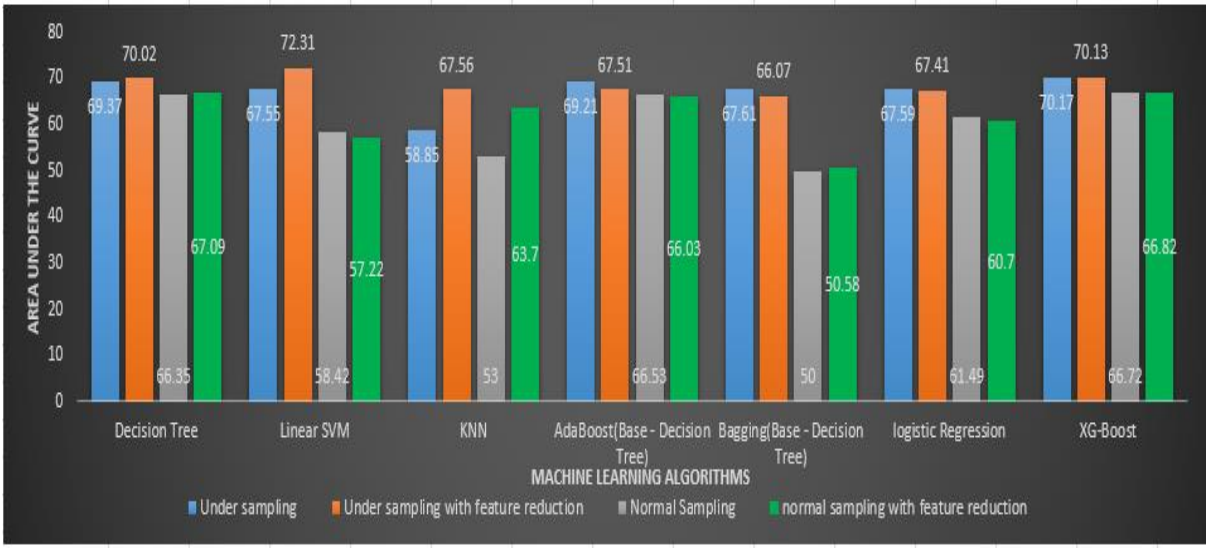
Parameter Tuning: **-** For each model, the parameters values to consider were varied. Then models were built using all possible combinations of parameters and the model was evaluated using 5-fold cross validation of the train set. Then the best set of parameters were chosen based on the result of the 5-fold cross validation.

Model Building: - The machine learning models were built using the best combination of parameters obtained from the previous step. Decision Tree, KNN, Linear SVM, Logistic Regression, Adaboost(Base Classifier-Decision Tree), Bagging(Base Classifier -Decision Tree) and XGBoost models were built.

Evaluation: **-** Metrics such has AUC, Accuracy, Precision, Recall and F1-Score were calculated. However due to the high-class imbalance present in the data we compared models based on AUC. In this task, false positives and false negatives have equal cost hence only AUC was considered as the major evaluation metrics.

Results: -

Comparison of ML algorithms with and without under sampling and feature engineering



Under sampling without feature engineering

ML Algorithm	AUC	Accuracy	Precision	Recall	F1 - Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	69.37	78.54	81	79	80	17864	3499	2080	2556
Linear SVM	67.55	70.72	80	71	74	15483	5880	1732	2904
KNN	58.85	56.32	76	56	61	11733	9630	1725	2911
AdaBoost(Base - Decision Tree)	69.21	74.12	81	74	76	16416	4947	1781	2855
Bagging(Base - Decision Tree)	67.61	72.4	80	72	75	16034	5329	1846	2790
logistic Regression	67.59	71.09	80	71	74	15604	5759	1755	2881
XG-Boost	70.17	74.77	81	75	77	16518	4845	1714	2922

Under sampling with feature engineering

ML Algorithm	AUC	Accuracy	Precision	Recall	F1 - Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	70.02	78.56	81	79	80	17796	3567	2005	2631
Linear SVM	72.31	72.31	80	72	75	16019	5344	1855	2781
KNN	67.56	75.31	80	75	77	17006	4357	2062	2062
AdaBoost(Base - Decision Tree)	67.51	74.51	80	75	76	16747	4616	2010	2626
Bagging(Base - Decision Tree)	66.07	79.54	80	80	80	18587	2776	2543	2093
logistic Regression	67.41	71.92	80	72	75	15901	5462	1836	2800
XG-Boost	70.13	77.59	81	78	80	17460	3903	1922	2714

Random sampling without feature engineering

ML Algorithm	AUC	Accuracy	Precision	Recall	F1 -Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	66.35	82.39	81	82	80	8897	500	1613	990
Linear SVM	58.42	80.98	80	81	76	9234	163	2119	484
KNN	53	77.76	72	78	72	9091	306	2362	241
AdaBoost(Base - Decision Tree)	66.53	82.3	81	82	80	8870	527	1596	1007
Bagging(Base - Decision Tree)	50	78.3	61	78	69	9397	0	2603	0
logistic Regression	61.49	81.83	81	82	78	9154	243	1937	666
XG-Boost	66.72	82.54	81	83	81	8895	502	1593	1010

Random sampling with feature engineering

ML Algorithm	AUC	Accuracy	Precision	Recall	F1 - Score	True Negative	False Positive	False Negative	True Positive
Decision Tree	67.09	82.19	81	82	81	8812	585	1551	1052
Linear SVM	57.22	80.66	80	81	75	9268	129	2191	412
KNN	63.7	81.43	79	81	79	8929	468	1760	843
AdaBoost(Base - Decision Tree)	66.03	82.02	80	82	80	8859	538	1619	984
Bagging(Base - Decision Tree)	50.58	78.48	77	78	69	9384	13	2569	34
logistic Regression	60.7	81.65	80	82	78	9182	215	1986	617
XG-Boost	66.82	82.69	81	83	81	8913	484	1593	1010

Analysis and Inference: -

From the results, it can be observed that feature engineering does not increase the performance of the algorithm considerably. However, one interesting information that was observed during feature selection is that, the credit limit and bill amount due contribute little when predicting if the user would be a defaulter in the coming month. Interestingly Age, Sex and Marital status are the highly-ranked features.

Again in this task also bagging performs bad and actually there is reduction in performance when compared to Decision tree. This again proves that bagging is a bad choice when the data is imbalanced.

The best result 72.31% AUC is achieved by Linear SVM with under sampling and feature reduction. This proves that even with the advent of new algorithms like xgboost, SVM is still one of the best performing algorithm which provides generalized predictive model.

Among the nonlinear classifiers, Decision tree constantly out performs KNN. This clearly emphasizes that decision tree can easily explain the data and distinguish positives from negatives better.

Overall linear classifiers are the best suited type of algorithms and ensemble algorithms are the worst performing algorithms for this data set.

Conclusion: -

This project clearly proves that machine learning can be useful in the credit card domain. Also, Machine learning algorithms does a good job in identifying fraud transactions. Thus we need not worry about fraudulent transactions if we decide to move to a complete digital currency system. Also one interesting information that is unearth from this project is that a person's demographic information has a more significant part in determining if a person would default in paying his credit card bills than his credit limit and amount due.

We believe that this project provides compelling evidence that machine learning can help banks and credit card companies and help them identify frauds and defaulters efficiently.

References: -

Task 1 Dataset :- https://www.kaggle.com/dalpozz/creditcardfraud
Task 2 Dataset :- https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
http://scikit-learn.org/stable/documentation.html
https://github.com/dmlc/xgboost
https://xgboost.readthedocs.io/en/latest/