Credit Card Fraud Detection

Name:	**Yash Dhiman**
Registration No./Roll No.:	21317
Institute/University Name:	IISER Bhopal
Program/Stream:	Electrical Engineering and Computer Science
Problem Release date:	August 17, 2023
Date of Submission:	November 19, 2023

1 Introduction

The aim of this project is to develop a suitable supervised machine learning classification model that predicts whether a transaction has been made by the user(i.e, genuine) or not(i.e, fraud) based on a given data point(i.e, instance). The training data set contains 30 different features and 57116 data points(which are transactions in real life) out of which 142 transactions are fraud and 56974 are genuine. In the test data we have 14280 data points whose class labels are to be predicted. Test data has 3 three missing values in three distinct columns which were filled with mean of values in those columns. Since the model just has to classify a transaction as fraud or not fraud it is clearly a two class classification problem.

2 Methods

In this project, seven different supervised machine learning classification models(AdaBoost, K-nearest-neighbours, Decision Tree, Logistic Regression, Multinomial Naive Bayes, Random Forest, Support Vector Machine) were used on training data and their performances were evaluated using pre-existing evaluation criterions following which the model with best performance(Random Forest in this case) was used to classify a transaction into one of the two classes in test data. The training data contains 30 features out of which 2 were dropped out in accordance to heuristics. Now, on this dataset preprocessing techniques like feature selection and scaling were implemented. Feature selection techniques,which find relation between features and classes and assign scores to features accordingly, like chi-square and mutual information gain were used to select k top scoring features(value of k being chosen randomly). Scaling techniques like MinMax scaling which scales down all values in dataset between [0,1] and Standard scaler which standardizes the given dataset were used while training the model and their effect on the performance was noted.

GitHub link

3 Experimental Setup and Result

The following classification models were used from scikit-learn[8]:
AdaBoost Classifier[1]
Parameters used are:

- estimator: It defines the base estimator from which the boosted ensemble is built. Eg DecisionTreeClassifier

- random_state: Controls the random seed given at each estimator at each boosting iteration.

K-Nearest-Neighbours Classifier [5]
Parameters used are:

- n_neighbours:Number of neighbors to use for k-neighbors weights:Weight function used in prediction. Possible values:

  'uniform' : uniform weights. All points in each neighborhood are weighted equally.

  'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

- algorithms: Algorithm used to compute the nearest neighbors:

Support Vector Machine Classifier [6]
Parameters used are:

- C: It is the cost parameter whose value is chosen so as to minimize misclassification rate and maximize margin.

- kernel: Specifies the functions used to find the maximum-margin hyperplane

Decision Tree Classifier [7]
Parameters used are:

- criterion:The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain.
- max_features:It specifies the number of features to consider when looking for the best split:

- max_depth:The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split(default value = 2) samples.

- ccp_alpha:Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed.

Random Forest Classifier [2]
Parameters used are:

- criterion:The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain.
- max_depth:The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split(default value = 2) samples.

- max_features:The number of features to consider when looking for the best split,the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features.

Table 1: Performance Of Different Classifiers Using All Features

Classifier	Precision	Recall	F-measure
Adaptive Boosting	0.881677	0.943319	0.910820
Decision Tree	0.872115	0.879956	0.879950
K-Nearest Neighbor	0.959763	0.904841	0.930549
Logistic Regression	0.569093	0.957574	0.616760
Random Forest	0.974969	0.904876	0.937117
Support Vector Machine	0.864810	0.946770	0.901626
Multinomial Naive Bayes	0.498756	0.500000	0.499377

Table 2: Confusion Matrices of Different Classifiers

Actual Class	Predicted Class	
	True	Fraud
True	56935	39
Fraud	16	126

Adaptive Boosting

Actual Class	Predicted Class	
	True	Fraud
True	56974	0
Fraud	142	0

Naive Bayes

Actual Class	Predicted Class	
	True	Fraud
True	56937	37
Farud	34	108

Decision Tree

Actual Class	Predicted Class	
	true	fraud
True	56964	10
Fraud	27	115

K-Nearest Neighbor

Actual Class	Predicted Class	
	True	Fraud
True	56152	822
Fraud	10	132

Logistic Regression

Actual Class	Predicted Class	
	True	Fraud
True	56968	6
Fraud	27	15

Random Forest

Actual Class	Predicted Class	
	True	Fraud
True	56927	47
Fraud	15	127

SVM

- n_estimators:The number of trees in the forest.

Multiomial Naive Bayes Classifier [4]
Parameters used are:

- aplha:Laplace smoothing parameter, as alpha increases, the likelihood probability moves towards uniform distribution (0.5). Most of the time, alpha = 1 is being used to remove the problem of zero probability.

Logistic Regression Classifier [3]
Parameters used are:

- solver:Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, the following aspects may help in deciding a solver:

  For small datasets, 'liblinear' is a good choice. For multiclass problems, only 'newton-cg', 'lbfgs' handle multinomial loss; 'liblinear' is limited to one-versus-rest schemes.

4 Conclusion

On rigorous implementation of several classification models Random Forest was found to have best score and hence was used in predicting the labels of test data.

References

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.adaboostclassifier.html.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.randomforestclassifier.html.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear$_m$*odel.logisticregression.html*.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.naive$_b$*ayes.multinomialnb.htmlsklearn.naive*$_b$*ayes.m*

[5] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.kneighborsclassifier.html.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.svc.html.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.tree.decisiontreeclassifier.html.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.