CREDIT CARD FRAUD DETECTION

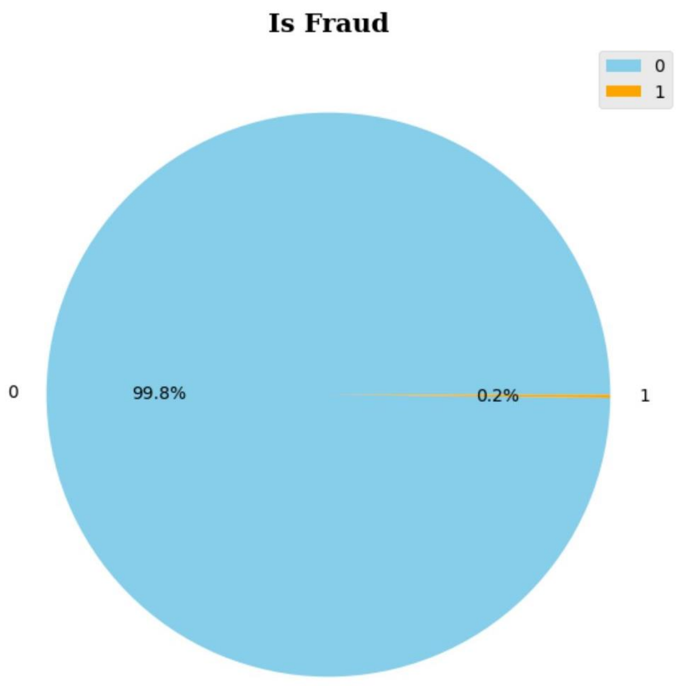
INTRODUCTION

Financial fraud is a major worry in the current digital era since technological improvements have made transactions easier while creating new criminal activity opportunities. To reduce losses, preserve confidence, and protect the general stability of financial institutions, fraud detection in real-time is essential. We will be able to examine enormous volumes of transaction data to find trends and anomalies connected to fraudulent conduct by utilizing the power of machine learning algorithms such as Logistic Regression, SVM, SGD classifier, KNN classifier, Naïve Bayes, Decision tree, Extra tree classifier, random forest, Extra trees classifier, MLP classifier and stack based model(DT, RF, Extra trees classifier).

DATASET

The dataset contains transactions made by credit cards. We assume the data set contains only numerical input variables which are the result of Principal Component Analysis(PCA) due to confidentiality.

Features v1,v2,……v28 are the principal components obtained with PCA. The dataset is highly unbalanced as it contains 99.8% non-fraud transactions and about 0.2% fraudulent transactions.



CLASS IMBALANCE HANDLING

Address the issue of class imbalance in the dataset, as fraudulent transactions are usually a small proportion of the total data. Techniques like oversampling, undersampling, or generating synthetic data (e.g., SMOTE) can be used.

When machine learning is used to detect credit card fraud, handling class imbalances is important to ensure the model's effectiveness in detecting fraudulent transactions and the best approach depends on a particular dataset and machine learning algorithm chosen.

Under-sampling seeks to balance the class distribution by randomly removing samples from the majority group (a non-deceptive task).

Over-sampling is the replication of samples from a subclass (the use of fraud) to balance the class distribution.

SMOTE (Synthetic Minority Over-Sampling Method) creates artificial samples for minorities by interpolating between existing minority class samples.

In practice, there is no one-size-fits-all solution, and the best approach may vary depending on the specific dataset and machine learning.

WITHOUT CLASS IMBALANCE HANDLING

Model	Y train accuracy	Y test accuracy	Precision	Recall	F1 score
Logistic regression	99.90	99.914	0.91	0.80	0.84
SVM	99.83	99.847	0.82	0.55	0.58
SGD	99.81	99.836	0.50	0.50	0.50
KNN	99.834	99.846	1.00	0.52	0.53
Naive bayes (bernouli nb)	99.912	99.920	0.90	0.83	0.86
DT	100	99.91	0.84	0.90	0.87
Extra tree classifier	100	99.913	0.86	0.88	0.87
Random forest	99.99	99.95	0.98	0.89	0.93
Extra trees classifier	100	99.96	0.98	0.90	0.93
MLP classifier	99.832	99.826	0.74	0.90	0.80
Stack model	100	99.96	0.97	0.91	0.94

With under sampling

Training data before under-sampling {0: 190477, 1: 343}

Training data after under-sampling {0: 343, 1: 343}

Model	Y train accuracy	Y test accuracy	Precision	Recall	F1 score
Logistic regression	95.189	78.00	0.50	0.86	0.44
SVM	69.679	84.47	0.50	0.63	0.46
SGD	51.311	98.397	0.50	0.50	0.50
KNN	93.731	7.206	0.50	0.50	0.07
Naive bayes (bernouli nb)	89.7959	99.124	0.57	0.93	0.62
DT	100	37.47	0.50	0.67	0.27
Extra tree classifier	100	82.091	0.50	0.86	0.46
Random forest	99.56268	70.919	0.50	0.82	0.42
Extra trees classifier	100	90.7157	0.51	0.93	0.49
MLP classifier	70.408	89.703	0.50	0.65	0.48
Stack model	100	69.935	0.50	0.82	0.42

With oversampling

Training data before under-sampling {0: 190477, 1: 343}

Training data after under-sampling {0: 190477, 1: 190477}

Model	Y train accuracy	Y test accuracy	Precision	Recall	F1 score
Logistic regression	93.910	96.314	0.52	0.93	0.53
SVM	72.9494	99.566	0.59	0.74	0.63
SGD	50.261	0.7086	0.50	0.50	0.01
KNN	99.9448	99.731	0.60	0.61	0.60
Naive bayes (bernouli nb)	90.07	99.02	0.56	0.92	0.61
DT	100	99.91	0.85	0.88	0.87

Extra tree classifier	100	99.9127	0.87	0.86	0.86
Random forest	99.999	99.96	0.98	0.90	0.93
Extra trees classifier	100	99.95	0.97	0.90	0.93
MLP classifier	95.105	95.95	0.52	0.93	0.52
Stack model	100	99.89	0.99	0.67	0.75

With SMOTETomek

Training data before under-sampling {0: 190477, 1: 343}

Training data after under-sampling {0: 189879, 1: 189879}

Model	Y train accuracy	Y test accuracy	Precision	Recall	F1 score
Logistic regression	97.027	97.97	0.53	0.94	0.56
SVM	91.16	98.35	0.54	0.90	0.56
SGD	95.62	95.77	0.52	0.94	0.52
KNN	97.633	94.533	0.51	0.74	0.50
Naive bayes (bernouli nb)	94.054	99.64	0.65	0.92	0.72
DT	100	99.79	0.71	0.89	0.78
Extra tree classifier	100	99.707	0.66	0.90	0.73
Random forest	99.999	99.946	0.91	0.93	0.92
Extra trees classifier	100	99.9553	0.93	0.93	0.93
MLP classifier	98.3189	98.5285	0.55	0.95	0.58
Stack model	100	99.96	0.97	0.90	0.93

For credit card fraud detection, you likely want to achieve a high recall to catch as many fraudulent transactions as possible while keeping precision reasonable to avoid excessive false positives. Consequently, the F1 score may be a good overall metric to evaluate your model's effectiveness.

CONCLUSION

Without class imbalance handling:

Stack model	100	99.96	0.97	0.91	0.94

The stack model consists of a Decision tree, random forest, and Extra trees classifier. The final estimator is the Extra trees classifier.

The stack model performs well.

With undersampling:

Naive bayes (bernouli nb)	89.7959	99.124	0.57	0.93	0.62

Naive Bayes perform well

With oversampling:

Random forest	99.999	99.96	0.98	0.90	0.93

Random Forest performs well

With SMOTETomek:

Extra trees classifier	100	99.9553	0.93	0.93	0.93

Extra trees classifier performs well

Among all these, for the given dataset, it is preferred to use SMOTETomek for handling Class imbalance along with the Extra trees classifier model.

Although handling class imbalance is important, If you neglect class imbalance it is preferable to use a stack model consisting of DT, RF, Extra trees classifier.