Credit Card Fraud Detection

Name:	**Manideep Pendyala**
Registration No./Roll No.:	19219
Institute/University Name:	IISER Bhopal
Program/Stream:	e.g., PHY
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

1 Introduction

Credit card fraud detection is like looking for needles in a haystack which is almost impossible for a human. For this reason, the use of machine learning techniques is now widespread in the field of fraud detection. Given a set of real bank transactions made by users, the goal is to identify fraudulent transactions which have not been made by the users.The given dataset contains transactions made by credit card users in September 2013 by European cardholders.

1.1 Data Exploration

The given problem is supervised classification problem, so we were provided a training data set it's class labels and a test data set. The training set has 57116 rows and 30 columns. The test data set has 14280 rows and 30 columns. Unfortunately, due to confidentiality issues, the original features weren't provided.
The data contains only numerical input variables which are the result of a PCA transformation i.e they all are already mutually independent.Features V1, V2, . . . V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. This doesn't help in training our model so this along with time will be omitted.

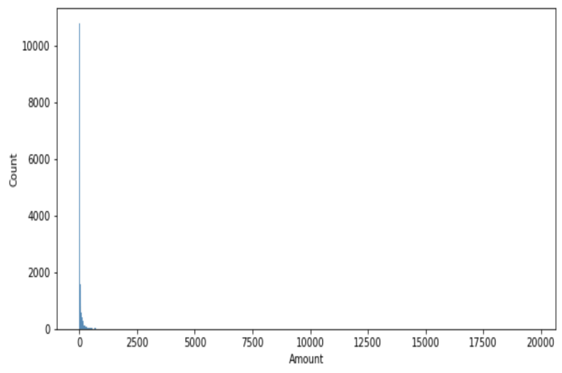

Figure 1: Distribution of transaction amount

The training data set doesn't have any null values, whereas the test data set has four null values one in each V26, V27, V28, Amount respectively. All the features of test and training data has all float data types other than the time feature.

The class label set also have same number of rows with one column with feature 'Class' which is the response variable and it takes value 1 in case of fraud and 0 otherwise.

1.2 Handling Imbalance

Given class labels are highly imbalanced. Out of 56974 there are only 142 fraudulent transactions and rest are genuine. It can be seen in the plot below.

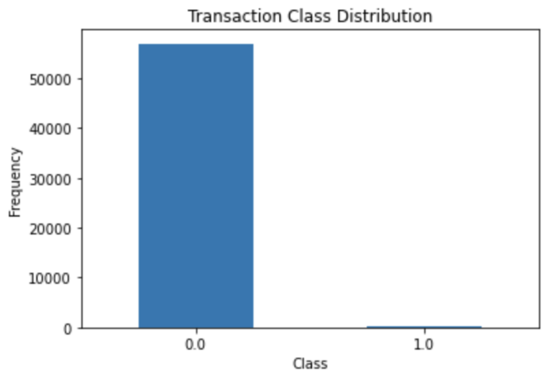

Figure 2: Overview of transaction distribution

We employ three methods on data to deal with imbalance and use the resultant datsets along with the original dataset for our predictions. They are taking a stratified sample, oversampling with SMOTE and undersampling.

1.2.1 Oversampling with SMOTE

Oversampling is imported from imblearn library. SMOTE (Synthetic Minority Oversampling Technique) is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.
After OverSampling, counts of label '1': 45582
After OverSampling, counts of label '0': 45582

1.2.2 Undersampling

Undersampling methods delete or select a subset of examples from the majority class. We undersample the imbalanced dataset with the Edited Nearest Neighbor rule here. Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class. Oversampling is preferred over this because undersampling tend to remove instances from data that may be carrying important information. The Edited Nearest Neighbors rule, or ENN for short, is another method for selecting examples for deletion. This rule involves using k=3 nearest neighbors to locate those examples in a dataset that are mis-classified and deleting them. We are going with 3 as it gave the best result after implementing classification methods. The ENN procedure is repeated multiple times to increase the refinement of selection.

1.2.3 Stratified sample

We take a stratified sample using command random_state and frac which will randomly select given fraction of data from the total dataset.The problem with this is though it is the number of fraud cases is way less and this leads to selecting all genuine transactions most of the time. Which leads to full performance metrics for all classifiers and it won't even work for Knn as it needs different classes. So, this is omitted.

1.3 Feature Extraction

Though all the features are PCA transformed, I performed feature extraction to see if I can get better results while reducing dimensionality and also computation time. I will apply the methods on all features and then on selected features based on how they score and then analysed the results. We will be using SelectKBest function from sklearn.feature_selection library and we will use f_classif as score function from the same library. So, these will evaluate and gives a score to each feature.

We gave k=4 as we will be selecting the best four features. The reason for choosing f_classif is that the data has negative values and statistical scoring function like chi2 can't work on such datasets. Based on scores of 28 features, I used selected four features with the best scores.Based on the scores the descending order for V's is 3,22,12,10,27,26,15,.. But we are selecting four best features. We will be omitting all features other than V3, V22, V12 and V10. I used the pairplot of these four features to understand their relation and see what type of models I can implement. Clearly, when we look at

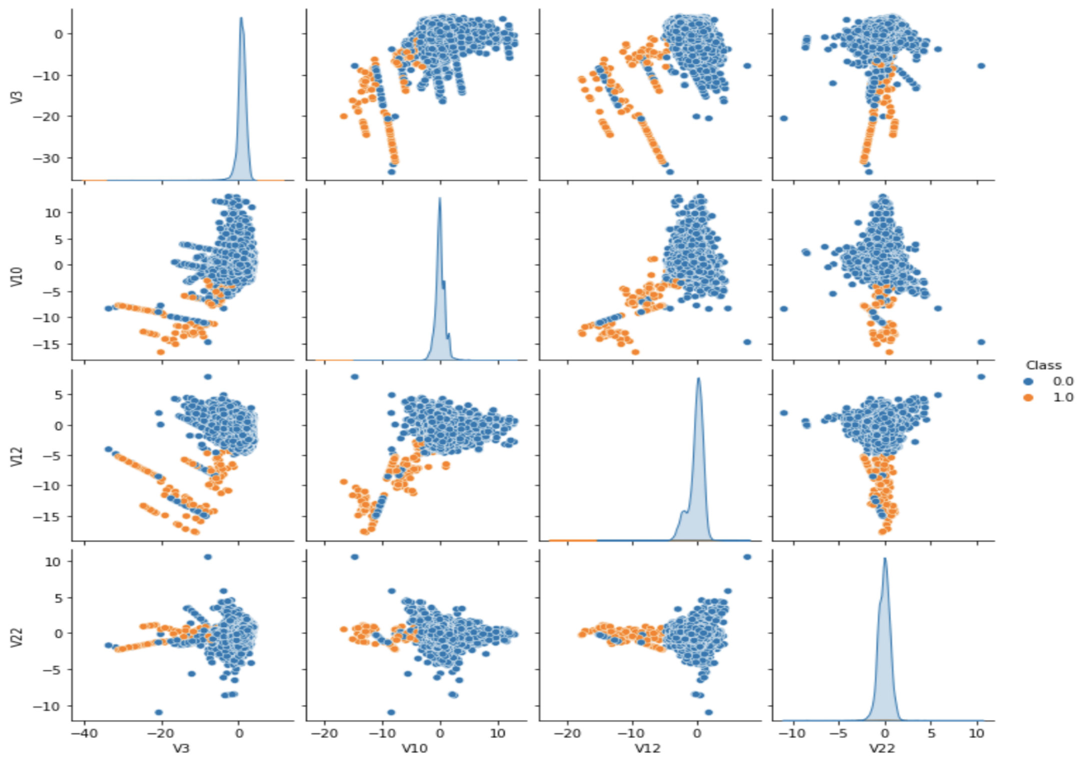

Figure 3: Pairplot of the selected features

the diagonal plot the data distribution of each features is Gaussian with a little error, So we chose Gaussian naive Bayes as in that the function we use is Gaussian and it might give better predictions. Also, the data looks linearly separable with minimal error, so we are selecting logistic regression. We will also be using K-nearest neighbours method. So, we will be using Gaussian naive Bayes, linear regression and knn as our classifiers.

2 Methods

The reason for choosing these classifiers is mentioned in the last section. We have a binary classification problem, so all our classifiers are based on supervised learning.

2.1 Gaussian Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Here, we will be implementing Gaussian Naive Bayes algorithm for classification. Here, the parameters σ_y and μ_y are estimated using maximum likelihood.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

2.2 K Nearest Neighbours

The KNN rule puts a test data point into a particular class, if the class has the maximum number of members among the, k-nearest neighbors of the set of training samples. The method finds the k-nearest neighbors of the test data point from the training samples using a distance function. Therefore it assigns the test point to a particular class by taking a majority vote among the k-nearest neighbors.

2.3 Logistic regression

It is used for classification of data samples. It is also known as hyperplane classifier, since the aim of this method is to find a hyperplane to segregate samples of two different classes. We selected this as the data seems to be separable by a hyperplane with minimal error.

2.4 Parameter Tuning

2.4.1 GridSearchCV

GridSearchCV is imported from sklearn.model_selection library. It performs an exhaustive search over specified parameter values for an estimator. GridSearchCV implements a "fit" and a "score" method. All parameters to be cross-validated are given before hand. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid. The parameters selected are those that maximize the score of the left out data, unless an explicit score is passed in which case it is used instead.
For Gaussian naive Bayes, 'var_smoothing' is the only parameter which doesn't change anything and gives same result. So, GNB doesn't change.
For Logistic regression, the parameters used are 'C' (hyperparameter) which was given a range (-3,3,7), 'penalty' is other parameter which can take 'l1' or 'l2'. GridSearchCV gives the tuned hyper parameters that gives the maximum accuracy. 'C'=1.0, 'penalty'='l2'.We took the parameter 'cv' of GriSearchCV as '10' as it gave good results.
For K nearest neighbour, the only parameter is 'n_neighbours' which was given a range of (1,31). 'n_neighbours=3' gives the best accuracy. We took the parameter 'cv' of GriSearchCV as '3' due to optimal results.
Parameter tuning is performed after oversampling, undersampling and feature selection the best tuned hyper parameters reamined same.

3 Evaluation Criteria

The evaluation criteria used are macro averaged precision, recall, f-measure.Let us assume there are two classes in the data set, say, positive and negative. For a binary classification problem, A classifier makes error when a positive sample is predicted as negative and a negative sample is predicted as positive. The confusion matrix looks like this,

Table 1: Confusion matrix

	positive	negative
positive	TP	FN
negative	FP	TN

Here TP stands for true positive and it counts the number of data points correctly predicted to the positive class. FP stands for false positive and it counts the number of data points that actually belong to the negative class, but predicted as positive (i.e., falsely predicted as positive). FN stands for false negative and it counts the number of data points that actually belong to the positive class, but predicted as negative (i.e., falsely predicted as negative). TN stands for true negative and it counts the number of data points correctly predicted to the negative class.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

f-measure is harmonic mean of precision and recall.

$$f - measure = \frac{2 * precision * recall}{precision + recall}$$

There are two conventional methods to evaluate the performance of a classifier aggregated over all classes, namely macro-averaging and micro-averaging.The macro averaged measure finds the precision and recall score for each class from the confusion matrix and then the these scores for all the classes are averaged. The micro averaged measure individually aggregates the true positives, false positives and false negatives over all the classes and then finds the precision and recall. We will be considering macro averaged for this classification problem.

$$\begin{aligned} \text{Macro Averaged Precision} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \\ \text{Macro Averaged Recall} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \end{aligned}$$

$$\begin{aligned} \text{Micro Averaged Precision} &= \frac{\sum_{i=1}^{m}\text{TP}_i}{\sum_{i=1}^{m}\text{TP}_i + \sum_{i=1}^{m}\text{FP}_i} \\ \text{Micro Averaged Recall} &= \frac{\sum_{i=1}^{m}\text{TP}_i}{\sum_{i=1}^{m}\text{TP}_i + \sum_{i=1}^{m}\text{FN}_i} \end{aligned}$$

Table 2: macro-averaged performance metrics without parameter tuning for whole data

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.60
LR	0.83	0.78	0.80
KNN	0.92	0.94	0.93

Table 3: macro-averaged performance metrics with parameter tuning for whole data

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.60
LR	0.84	0.80	0.82
KNN	0.98	0.94	0.96

Table 4: macro-averaged performance metrics without parameter tuning for data after oversampling

Classifier	Precision	Recall	F-measure
GNB	0.55	0.93	0.59
LR	0.55	0.94	0.59
KNN	0.79	0.94	0.85

Table 5: macro-averaged performance metrics with parameter tuning for data after oversampling

Classifier	Precision	Recall	F-measure
GNB	0.55	0.93	0.59
LR	0.55	0.94	0.59
KNN	0.79	0.94	0.85

Table 6: macro-averaged performance metrics without parameter tuning for data after undersampling

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.60
LR	0.84	0.80	0.82
KNN	0.88	0.94	0.91

Table 7: macro-averaged performance metrics with parameter tuning for data after undersampling

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.60
LR	0.84	0.80	0.82
KNN	0.88	0.94	0.91

Table 8: macro-averaged performance metrics without parameter tuning for selected features of data

Classifier	Precision	Recall	F-measure
GNB	0.60	0.89	0.65
LR	0.83	0.80	0.81
KNN	0.87	0.89	0.88

Table 9: macro-averaged performance metrics with parameter tuning for selected features of data

Classifier	Precision	Recall	F-measure
GNB	0.60	0.89	0.65
LR	0.83	0.80	0.81
KNN	0.92	0.89	0.90

Table 10: macro-averaged performance metrics without parameter tuning for selected features of data after oversampling

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.61
LR	0.56	0.93	0.60
KNN	0.57	0.91	0.62

Table 11: macro-averaged performance metrics with parameter tuning for selected features of data after oversampling

Classifier	Precision	Recall	F-measure
GNB	0.56	0.93	0.61
LR	0.58	0.93	0.63
KNN	0.57	0.98	0.63

Table 12: macro-averaged performance metrics without parameter tuning for selected features of data after undersampling

Classifier	Precision	Recall	F-measure
GNB	0.60	0.89	0.65
LR	0.84	0.83	0.83
KNN	0.84	0.89	0.86

Table 13: macro-averaged performance metrics with parameter tuning for selected features of data after undersampling

Classifier	Precision	Recall	F-measure
GNB	0.55	0.93	0.59
LR	0.84	0.83	0.83
KNN	0.86	0.89	0.87

4 Analysis of Results

For the entire data set, without parameter tuning we get best metrics from KNN followed by LR and GNB. After parameter tuning, by applying the hyper tuned parameters the performance of KNN and LR increased and after parameter tuning KNN gives really high precision, recall and thus f-measure.

Once oversampling using SMOTE is implemented on the entire data set, the performance of classifiers decreased this may be due to addition of a lot of synthetic labels to balance class labels. Surprisingly, after parameter tuning the classifiers for oversampled data generate same metrics as that of without parameter tuning. GNB is the least effected, I think it's beacuse the distribution won't be effected much leading to similar predictions where as for LR and KNN the new data points change a lot thus leading to lower metrics.

Once undersampling using ENN is implemented on the entire data set, the performance of classifiers decreased compared to entire data but this is better than that of oversampling case this may be due to addition of a lot of synthetic labels to balance class labels in oversampling. After parameter tuning the classifiers for undersampled data generate same metrics as that of without parameter tuning.

After selecting the best four features and implementing classifiers on it, it gives better result for GNB and LR than that of the whole data but the metrics of KNN are reduced I believe this is due to reduced number of features leading to less data points and lower accuracy. After parameter tuning, the metrics improved but are lower than the metrics for parameter tuned full data.

Once oversampling using SMOTE is implemented on the feature selected data set, the performance of classifiers decreased. Surprisingly, after parameter tuning GNB, KNN for oversampled data generate same(almost) metrics for as that of without parameter tuning. Scores of LR even increased.

Once undersampling using ENN is implemented on the feature selected data set, the performance of KNN decreased compared to entire data but GNB remained same as that of full data and LR performed better. After parameter tuning the classifiers for undersampled data generate same metrics as that of without parameter tuning with KNN being exception as it showed some improvement.

Oversampling is preferred over undersampling techniques. The reason being, in undersampling we tend to remove instances from data that may be carrying some important information.
Oversampling with feature selection gave decently better results.
Based on the performance metrics for classifiers over different data processes, hyper tuned K-nearest neighbours performs best when implemented on the whole data. Here, the parameters are, n_neighbours which is equal to 3.

5 Discussions and Conclusion

KNN is the most effected by any type of data handling may it be oversampling and undersampling. GNB is the least effected. I thought GNB will give best results but due to lack of possibility of tuning various parameters it didn't give much way to manipulate, KNN gave the best performance. Feature selection did help and the results are also a little similar to the full data, so it can be used to fasten the process when datasets are even larger.
The problem with stratified sample where it selects mostly class 0 needs some work to make sure it will properly distribute random selection across classes.
Further work is possible with using SMOTE and ENN together. Also, different classifiers can be used together to give better results as they have more parameters to play with. Other methods can be used like decision trees, SVM,etc..
A pipeline can be built which automatically detect fraud transactions in real time, but this needs work on big data. Further improvements can be made using deep learning and neural networks.

6 References

R. Duda, P. Hart, and D. G. Stork. Pattern Classification. Wiley, 2000.

J. Han, M. Kamber, and J. Pei. Data Mining Concepts and Techniques. Morgan Kaufmann, third edition, 2012.

R. Kohavi. A study of cross validation and bootstrap for accuracy estimation and model selection. In Proceedings of International Joint Conference on Artificial Intelligence, pages 1137–1145, 1995.

P. Refaeilzadeh, L. Tang, and H. Liu. Cross validation. In L. Liu and M. T. Ozsu, editors, Encyclopedia of Database Systems, pages 532–538. Springer, 2009.

Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. The Elements of Statistical Learn- ing. Springer, second edition, 2008.

C. M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

T. Basu and C. A. Murthy. Effective text classification by a supervised feature selection approach. In Proceedings of the IEEE International Conference on Data Mining Workshops, ICDMW'12, pages 918–925, Brussels, Belgium,, 2012.

L. Galavotti, F. Sebastiani, and M. Simi. Feature selection and negative evidence in automated text categorization. In Proceedings of the Knowledge Discovery and Data Mining Workshop on Text Mining, KDD'00, 2000.

Feature Extraction
K nearest neighbours
GNB
oversampling using SMOTE
Stratified sampling
GridSearchCV