Introduction

In this paper we will apply these methods using the sci-kit learn and keras machine learning libraries. We will work on a credit card fraud dataset where the goal will be to create an effective model for predicting cases of fraud.

Credit Card Fraud Dataset

The dataset:

This dataset contains information on credit card transactions over two days during the month of September 2013 by European cardholders. This is a noticeably larger dataset than the previous one. It contains 284,807 total transactions with a total of 30 possible input features and 1 binary target class. The target class indicates whether or not fraud occurred. The 30 input features contain 28 features (V1-V28) that have been obtained by principal component analysis (PCA); these features have already been transformed and not much is known about background info due to confidentiality issues. The remaining two input features are Time and Amount, which have not been transformed. All input feature data types in this dataset are quantitative. For our purposes we will only be using the 28 already transformed features because they have already been transformed and it’s not clear whether it would be appropriate to try to put all of them through another transformation with the other two input features. Also because they were obtained from PCA it stands to reason that the principal component features are likely to be able to create a good model on their own. Lastly, another important aspect of this dataset is that it is highly imbalanced. Out of the 284,807 total transactions there were only 492 cases of fraud. The approach will be to test out several classification models. The priority is to have a model which detects fraud over the possibility of getting false positives because the consequences for the former are far greater than the latter. So because of that we will consider maximizing the recall score, True Positives / (True Positives + False Negatives), to be one of our most important goals. We will also consider other metrics like Area Under Receiver Operating Curve, F1 Score, Precision, and Accuracy important although to a lesser degree.

Data Pre-Processing:

Before our classification models can be created it is necessary to prepare the data so the models can be trained and evaluated properly. This involved three main steps. First, as alluded to before we will only be using the 28 principal components as input features so the Time and Amount columns were dropped from the dataset. Second, the data was split into training, validation, and testing sets. This was done using the ‘train_test_split’ function in sklearn twice (one extra for the validation portion) using a testing size of 20%. Third, due to the high imbalance among the class data (fraud or not fraud) it was necessary to use a technique called Synthetic Minority Oversampling Technique (SMOTE), from the imbalanced-learn library, on the training data which transformed them into balanced datasets by generating synthetic samples using minority class observations as points of reference. Since the remaining input features, the principal components from the original dataset, had already been transformed it was not necessary to use any scaling as had been done for the previous models.

Classification Models:

For the first model a simple logistic regression model from sklearn was fit to the training data using the Limited-memory Broyden Fletcher Goldfarb Shanno as its solver. It also used a L2 (ridge) regularization to prevent overfitting, and the maximum number of iterations was increased (from the default) to 100,000 as the model was failing to converge on lower iterations. The performance metrics for the model are recorded in the table below.

TABLE 4

AUROC	0.941
Accuracy	0.973
Recall	0.908
Precision	0.056
F1	0.105

The next classification model to be trained and tested was a random forest classifier from the sklearn library. This model was fit to the training data using the default 100 estimators, and a max depth of 8. The performance metrics for the model are recorded in the table below.

TABLE 5

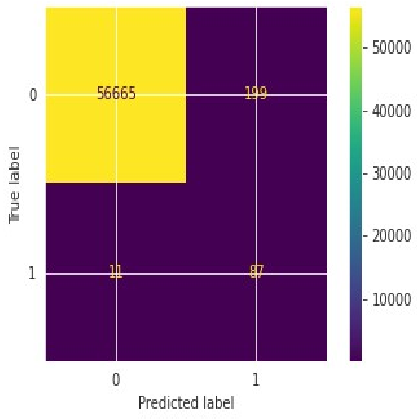
AUROC	0.941
Accuracy	0.996
Recall	0.888
Precision	0.275
F1	0.420

The final classification model to be trained and tested is a neural network built from the Keras API which is itself built upon TensorFlow. The sequential model was used so that adjustments at each layer could be made easily. The model.add() function was used to add two hidden layers both of which consisted of 14 nodes. Each hidden layer was followed by a dropout rate of 0.2. The output layer consisted of one node that represented the model's prediction. For the hidden layers the kernel initializer was set to orthogonal and activation function was set to rectified linear unit (RELU). For the output layer the kernel initializer was set to uniform and the activation function was set to sigmoid. The model was compiled using the ADAM optimizer and binary cross entropy loss function. Then the model was fit to the training data (with validation data also passed in) using a batch size of 5 with 20 epochs. After all this the resulting neural network was used to test predictions and the performance metrics are recorded in the table below.

TABLE 6

AUROC	0.942
Accuracy	0.996
Recall	0.888
Precision	0.304
F1	0.453

Also a confusion matrix was plotted for the neural network's results and is displayed below.



Given these results it seems all three of the models were able to obtain reasonably high recall values between 0.888 to 0.908. The difference is just 0.02. They also had almost the exact same Areas Under Receiver Operating Curve (AUROC). This indicates that they perform similarly at various threshold settings and all have good high ability to distinguish classes. Looking at the other metrics it seems they generally had noticeably lower precision values and thus lower F1 values. As alluded to before this was an expected trade off because we decided to prioritize the maximizing recall values since it's likely more important to detect as many fraud cases as reasonably possible while also not casting too wide of a net in false positives. In the case of credit card fraud it can be expected that it would be relatively simple to resolve a false positive from the cardholders perspective by verifying their transactions while that may not be the case with false negatives.

Taking into account the higher precision and F1 values and small difference in recall values it seems that the random forest and neural network models would be better overall models. Between those two the higher precision and F1 values seem to favor the neural network model slightly considering the rest of their performance metrics are either the same or nearly the exact same.

Future Directions

Some possibilities for future work on either the same or similar datasets may likely include greater hyperparameter tuning for both datasets. Although they were adjusted here to improve performance because constructing the models often took a while to train, especially for the neural network, it's likely some models had better possible parameters which were not explored. It may also be a good idea to try to incorporate the two excluded input features from the second dataset after experimenting with how to best handle the data transformation. It's also a bit unclear if the method of handling the imbalanced dataset so some further experimentation with the imbalanced-learn library may prove useful for building better classification models.