Risk Management Final Project

Final Paper

Ensembled modelling for Credit card fraud detection system

Arun Mishra-am3464

Acknowledgments

I would like to extend my deepest gratitude to Professor Elliot Noma for his invaluable guidance and expert advice throughout the course of this project. His profound knowledge and insightful feedback significantly enhanced the quality and depth of this subject. I am immensely thankful for his patience and the time he invested in mentoring us, which has been vital in achieving the outcomes of this project. This project could not have reached its fruition without his exemplary guidance.

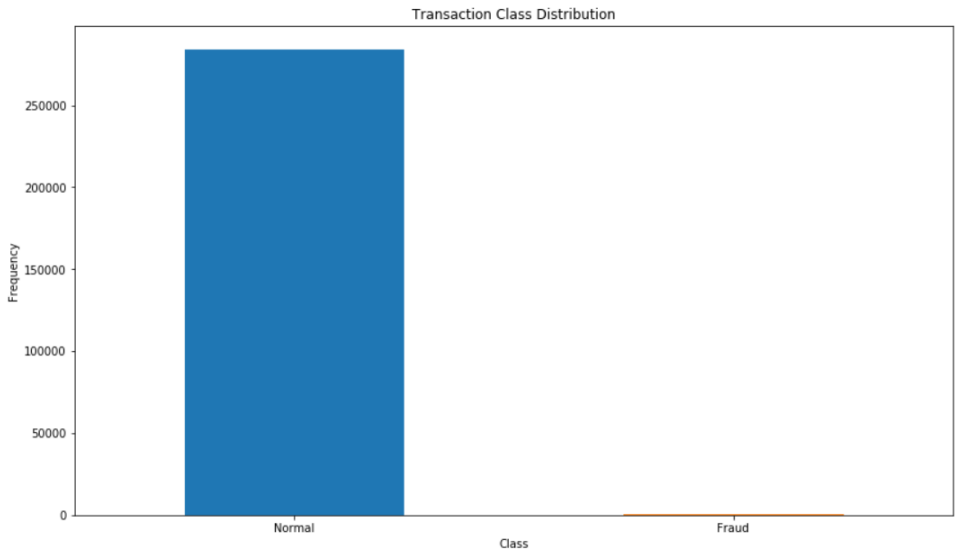
Introduction

Credit card fraud represents a significant challenge within the financial sector, necessitating sophisticated detection systems to safeguard consumer transactions. Traditional methods often struggle with the high volume and complex nature of modern credit transactions, which are compounded by the highly unbalanced nature of fraud occurrences. This research introduces an ensembled modeling approach, to enhance the detection and classification of fraudulent transactions.

Method

Dataset

The dataset utilized comprises transactions made by European cardholders in September 2013, recorded over two days, featuring 284,807 transactions of which 492 were fraudulent. This dataset is characterized by a significant class imbalance, with fraudulent transactions comprising only 0.172% of the total dataset. Principal Component Analysis (PCA) is employed to transform the features, maintaining confidentiality, and focusing analysis on transformed features and non-PCA-transformed 'Time' and 'Amount'.



Background Knowledge

One-class classification: One-class classification, also referred to as unary classification or class-modelling, is a specialized machine learning algorithm designed to determine whether new instances belong to a specific, predefined "normal" class. Unlike traditional binary or multi-class classification methods that learn from multiple classes, this approach focuses exclusively on data from a single class during training.

This methodology builds a model based on the common patterns of the normal class, using it as a benchmark to assess new instances. The primary task is to detect deviations from this norm, making it especially useful in scenarios like fraud detection, where anomalies are rare compared to normal occurrences.

By learning what constitutes normal behaviour, one-class classification effectively identifies significant anomalies, proving valuable in fields such as surveillance for unusual activities or maintenance for signs of system failure.

Support Vector Machine (SVM): Support Vector Machine (SVM) is a robust and versatile machine learning algorithm predominantly utilized for classification, though it is also capable of performing regression tasks. This method is particularly effective in complex domains where the distinctions between classes are not clearly defined.

SVM operates by identifying an optimal hyperplane that maximizes the margin between different classes. This characteristic makes it highly suitable for scenarios where the classification boundary needs to be as distinct as possible, even in highly dimensional spaces. As a result, SVM is widely applied in fields such as image recognition, bioinformatics, and text classification, where precise separation of classes is crucial for accurate prediction and analysis.

Fundamental concept of SVM: The fundamental concept of Support Vector Machine (SVM) centers on identifying the optimal hyperplane that

effectively separates the classes within the feature space. In a two-dimensional setting, this hyperplane manifests as a line. When extended to three dimensions, it becomes a plane, and in higher dimensions, it is referred to as a hyperplane. Although hyperplanes in higher-dimensional spaces are not visually representable, they can be precisely defined and computed using mathematical methods.

This capacity to define separation in multi-dimensional spaces is what makes SVM exceptionally useful for a variety of complex classification tasks. By maximizing the margin between differing class boundaries, SVM ensures a clear and robust classification, enhancing both the accuracy and generalizability of the model.

Random Forest: Random Forest is an advanced ensemble learning method used for tasks such as classification, regression, and more. It works by building numerous decision trees during the training phase and aggregates their outcomes to make predictions. Specifically, for classification tasks, it determines the class outcome based on the mode of the classes predicted by the individual trees, and for regression tasks, it calculates the mean prediction of these trees.

This method is highly regarded in the field of machine learning for its robustness, simplicity, and wide applicability across different types of data and predictions. Random Forest not only improves prediction accuracy but also helps in handling overfitting by averaging multiple deep decision trees, each trained on different parts of the same training set. This diversity of application makes it a powerful and versatile tool in the arsenal of machine learning techniques.

Fundamental concept of SVM: Random Forest operates through an ensemble approach, employing multiple decision trees to improve the robustness and accuracy of predictions. Here's how it functions:

Ensemble of Decision Trees: Random Forest generates a multitude of decision trees at training time. Each tree is independently constructed using a bootstrapped sample of the data, which means each sample is drawn with replacement from the original training set. This method ensures that each tree learns from a slightly different subset of the data, enhancing the diversity of the model.

Feature Randomness: During the construction of each tree, when a node is split, the selection of the split is not based on the best split among all features. Instead, it is the best split from a randomly selected subset of features. This randomness prevents the trees from being too similar and reduces the model's variance, typically leading to better performance.

Aggregation of Results: In classification tasks, Random Forest determines the final class based on the mode of the classes predicted by the individual trees—the most common output class among all trees is chosen as the final prediction. For regression tasks, it calculates the mean of the predictions made by all trees, thereby producing a more stable and accurate prediction by averaging multiple estimates.

This ensemble technique effectively combines the predictions of numerous models, mitigating the risk of overfitting associated with single decision trees and leveraging the strength of multiple learners to achieve superior predictive performance. outputs by the different trees.

Exploratory Data Analysis

The Exploratory Data Analysis (EDA) of the transaction amounts for both fraudulent and normal transactions reveals distinct patterns and differences in the behavior of the two classes:

In [9]:
```
## Get the Fraud and the normal dataset

fraud = data[data['Class']==1]

normal = data[data['Class']==0]
```

In [10]:
```
print(fraud.shape,normal.shape)
```

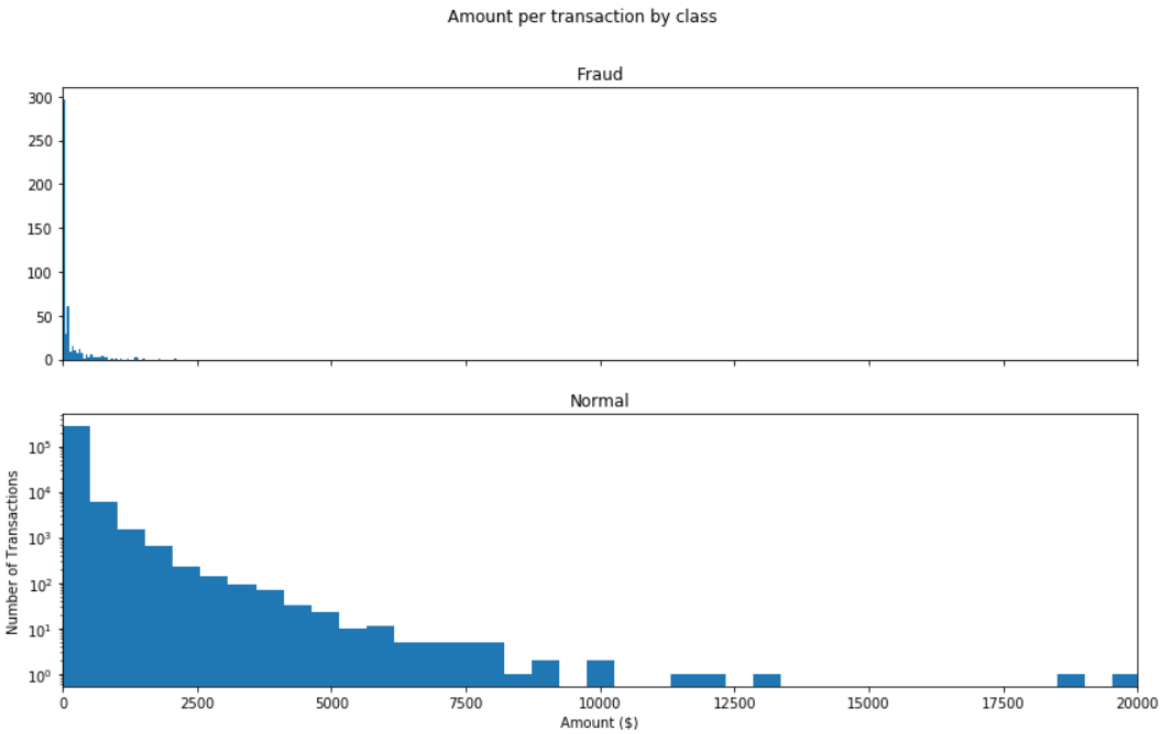
```
(492, 31) (284315, 31)
```

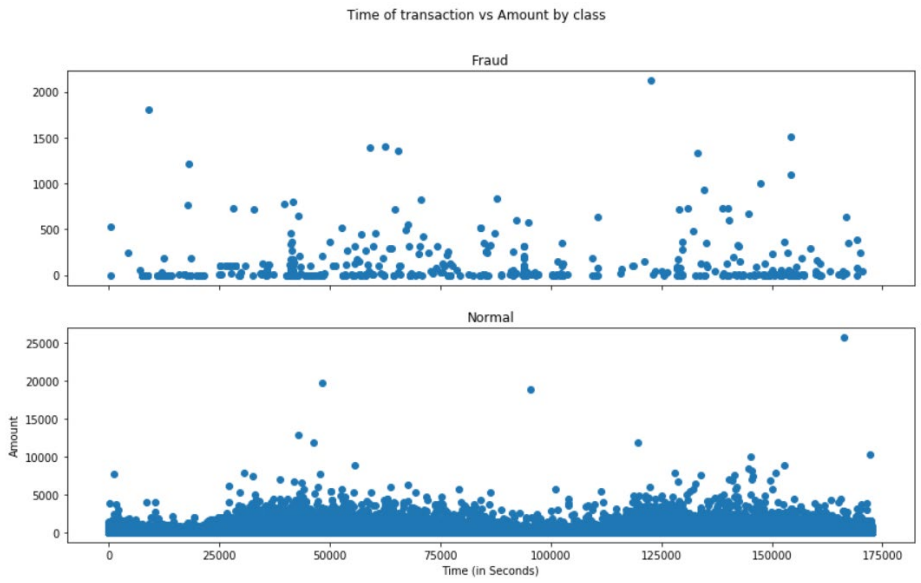
In [12]:
```
## We need to analyze more amount of information from the transaction data
#How different are the amount of money used in different transaction classes?
fraud.Amount.describe()
```

Out[12]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [13]:
```
normal.Amount.describe()
```

Out[13]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```





Fraudulent Transactions:

Count: There are 492 fraudulent transactions in the dataset.

Mean Amount: The average amount involved in fraudulent transactions is approximately $122.21, which is higher than the average for normal transactions.

Standard Deviation: A high standard deviation of $256.68 indicates significant variability in the amount of money involved in fraudulent transactions.

Minimum and Maximum Amount: The minimum transaction amount is $0.10, and the maximum is $2125.87, suggesting that fraud can occur at any transaction level, from very small to relatively large amounts.

Quartiles:

25%: At least 25% of fraudulent transactions involve $1.00 or less, indicating that a quarter of fraudulent activities involve very low transaction amounts.

50% (Median): The median amount is $9.25, much lower than the mean, reflecting a right-skewed distribution where most fraudulent transactions are of lower value, but a few high-value transactions raise the average.

75%: 75% of fraudulent transactions are $105.89 or less, further highlighting that while most fraudulent transactions are low in value, there are occasional higher value frauds.

Normal/legit/non-fraudulent Transactions:

Count: There are 284,315 normal transactions.

Mean Amount: The average amount for normal transactions is $88.29, which is lower than the average amount in fraudulent transactions.

Standard Deviation: The standard deviation is $250.11, similar to fraudulent transactions, indicating a wide range in transaction amounts in normal activities as well.

Minimum and Maximum Amount: Normal transactions range from $0.00 to $25,691.16, showing a broader range of transaction amounts compared to fraudulent transactions.

Quartiles:

25%: The first quartile is $5.65, suggesting that 25% of normal transactions are of this amount or less.

50% (Median): The median amount is $22.00, which is significantly lower than the average, indicating a right-skewed distribution where most transactions are of lower value.

75%: The third quartile is $77.05, confirming that most normal transactions do not involve high amounts.

The analysis of transaction amounts between fraudulent and normal transactions shows that, on average, fraudulent transactions involve higher amounts than normal ones, although the majority of both types of

transactions involve relatively small amounts. The distributions for both are right-skewed, with a smaller number of high-value transactions pushing the average above the median. This suggests that while fraudsters often target lower-value transactions to possibly avoid detection, they also occasionally attempt high-value frauds. The wide range in both classes indicates the complexity of patterns that a predictive model must learn to effectively differentiate between normal and fraudulent activities.

Main Framework

Ensembled model: The ensembled model designed for credit card fraud detection adopts a strategic approach by prioritizing the accurate identification of legitimate transactions, thus minimizing the volume of doubtful transactions that need further analysis. This approach is rooted in enhancing customer experience by reducing the likelihood of false positives, which can inconvenience genuine customers.

Context and Methodology: The dataset utilized for this project is extensive, containing over 280,000 transactions. Due to the computational limitations and time constraints associated with training models on such a large dataset, a strategy was adopted to sample 10% of the data randomly. This sampling was critical in managing resource utilization effectively while ensuring diverse transaction coverage. The sampled dataset included one randomly chosen fraudulent transaction alongside legitimate transactions, mirroring a one-class classification setup.

Algorithm Description:

Stage 1: Non-Fraudulent Transaction Identification

An SVM model was trained exclusively on legitimate transactions. This model's purpose was to sift through the data and segregate clear-cut legitimate transactions, relegating any uncertain cases to a "Grey-Area-Transactions" category. This effectively reduced the dataset's complexity by isolating straightforward cases.

In [29]:
```
## Take some sample of the data

data= data1.sample(frac = 0.1,random_state=1)

data.shape
```

Out[29]: (28481, 31)

In [ ]:
```

```

In [30]:
```
# Filter out non-fraudulent transactions
non_fraud = data[data['Class'] == 0]
non_fraud.shape
```

Out[30]: (28432, 31)

In [31]:
```
# Feature selection: Exclude 'Time' and 'Class' for training the model
X_non_fraud = non_fraud.drop(['Time', 'Class'], axis=1)
print('done1')

# Initialize One-Class SVM model
# Note: You'll need to adjust 'nu' parameter which is an upper bound on the fraction of training errors
# and a lower bound of the fraction of support vectors.
one_class_svm_non_fraud = OneClassSVM(kernel='rbf', gamma='auto', nu=0.01)
print('done2')

# Train the model
one_class_svm_non_fraud.fit(X_non_fraud)
print('done3')

# Predict using the trained model (1 for inliers, -1 for outliers)
# Here, we're using the entire dataset for prediction to find "grey area transactions"
predictions = one_class_svm_non_fraud.predict(data.drop(['Time', 'Class'], axis=1))
data['Grey_Non_Fraud'] = predictions
print('done4')
```

```
done1
done2
done3
done4
```

In [59]:
```
# Count the number of "grey area transactions"
grey_area_count = (data['Grey_Non_Fraud'] == -1).sum()

# Print the number of "grey area transactions"
print(f'Number of "grey area transactions": {grey_area_count}')
```

```
Number of "grey area transactions": 1734
```

Stage 2: Fraudulent Transaction Detection

A parallel SVM model was trained, this time focusing solely on the characteristics of fraudulent transactions. Similar to the first model, this stage aimed to filter out evident fraudulent activities, with all ambiguous transactions being added to the "Grey-Area-Transactions."

In [36]:

```
# Pseudocode:
# Filter the dataset to include only fraudulent transactions
# Train a One-Class SVM on this filtered dataset
# Predict "grey area transactions" as those where the One-Class SVM predicts an anomaly

# Python code:
# Filter out fraudulent transactions
fraud = data[data['Class'] == 1]
print('done1')

# Feature selection: Exclude 'Time' and 'Class' for training the model
X_fraud = fraud.drop(['Time', 'Class'], axis=1)
print('done2')

# Initialize One-Class SVM model
one_class_svm_fraud = OneClassSVM(kernel='rbf', gamma='auto', nu=0.01)
print('done3')

# Train the model on fraudulent data
one_class_svm_fraud.fit(X_fraud)
print('done4')

# Predict using the trained model (1 for inliers, -1 for outliers)
# Here, we're using the entire dataset for prediction to find "grey area transactions"
fraud_predictions = one_class_svm_fraud.predict(data.drop(['Time', 'Class'], axis=1))
data['Grey_Fraud'] = fraud_predictions
print('done5')
```

```
done1
done2
done3
done4
done5
```

Stage 3: Enhanced Classification on Grey Areas

The transactions categorized as "Grey-Area-Transactions" from both previous stages were then used as the training set for a Random Forest model. This model was tasked with making the final determinations, working to

accurately classify these uncertain transactions by drawing on the decision-making strengths of the Random Forest algorithm.

This ensembled approach leverages the distinct advantages of SVM for initial filtering based on clear characteristics and Random Forest for handling more complex, uncertain cases, thereby enhancing the overall predictive accuracy.

In [51]:
```python
# Pseudocode:
# Combine the "grey area transactions" from non-fraudulent and fraudulent predictions
# Label them correctly using the original dataset
# Train a Random Forest classifier to distinguish between legitimate and fraudulent transactions

# Python code:
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score




# Labels for "grey area transactions"
Y_grey = grey_area['Class']
X_grey = grey_area.drop(['Time', 'Class', 'Grey_Non_Fraud', 'SVM_Predicted_Class', 'Grey_Fraud', 'SVM_Fraud_Pred'], axis=1)
print('done2')
# Split the "grey area transactions" into training and test sets
X_train, X_test, Y_train, Y_test = train_test_split(X_grey, Y_grey, test_size=0.2, random_state=42)

# Initialize the Random Forest classifier
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
print('done3')
# Train the model
random_forest.fit(X_train, Y_train)
print('done4')
# Predict on the test set
rf_predictions = random_forest.predict(X_test)
print('done5')
# Evaluate the model
accuracy = accuracy_score(Y_test, rf_predictions)
print(f'Accuracy Score: {accuracy:.4f}')
print(classification_report(Y_test, rf_predictions))
```

Stage 4: Running the model on entire dataset.

In [53]:
```python
# Import the necessary functions
from sklearn.metrics import classification_report, accuracy_score

# Predict on the entire dataset (excluding 'Time', 'Class', and the 'Grey' columns)
full_data_predictions = random_forest.predict(data1.drop(['Time', 'Class'], axis=1))

# Actual class labels of the entire dataset
Y_full = data1['Class']

# Calculate the accuracy score for the entire dataset
full_data_accuracy = accuracy_score(Y_full, full_data_predictions)

# Generate a classification report for the entire dataset
full_data_class_report = classification_report(Y_full, full_data_predictions)

# Print the accuracy score and classification report
print(f'Accuracy Score for the Entire Dataset: {full_data_accuracy:.4f}')
print('Classification Report for the Entire Dataset:')
print(full_data_class_report)
```

Results

Results were generated using other stand-alone models, such as Isolation Forest (which is considered superior to Random Forest), Support Vector Machine (SVM), and Local Outlier Factor, on the same dataset for comparison.

```
Isolation Forest: 73
Accuracy Score :
0.9974368877497279
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.26      0.27      0.26        49

   micro avg       1.00      1.00      1.00     28481
   macro avg       0.63      0.63      0.63     28481
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
Accuracy Score :
0.9965942207085425
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.02      0.02      0.02        49

   micro avg       1.00      1.00      1.00     28481
   macro avg       0.51      0.51      0.51     28481
weighted avg       1.00      1.00      1.00     28481



Support Vector Machine: 8516
Accuracy Score :
0.7009936448860644
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.70      0.82     28432
           1       0.00      0.37      0.00        49

   micro avg       0.70      0.70      0.70     28481
   macro avg       0.50      0.53      0.41     28481
weighted avg       1.00      0.70      0.82     28481
```

```
Accuracy Score: 0.9994
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    284315
           1       0.90      0.71      0.79       492

    accuracy                           1.00    284807
   macro avg       0.95      0.85      0.90    284807
weighted avg       1.00      1.00      1.00    284807
```

Results in comparison of isolation forest model:

1. Comparing the accuracies, the ensembled model again outperforms the Isolation Forest, with an accuracy that is 0.2% higher. While this might seem like a small difference, in the context of a large number of transactions, even a 0.2% improvement can translate to correctly identifying a significant number of additional fraudulent transactions.
2. The precision of 0.90 for the ensembled model means that when it predicts a transaction as fraudulent, it is correct 90% of the time. In contrast, the Isolation Forest model has a precision of just 0.26, meaning it is correct only 26% of the time, which would lead to a large number of false positives.
3. The recall of 0.71 for the ensembled model is substantially higher than the 0.27 for the Isolation Forest. This means the ensembled model is more capable of finding and correctly classifying fraudulent transactions.
4. The F1-score, which balances precision and recall, is significantly higher for the ensembled model (0.79) compared to the Isolation Forest (0.26). This suggests a better overall performance for the ensembled model.
5. The factor by which the ensembled model is better can be quantified by comparing the F1-scores. Dividing the F1-score of the ensembled model by that of the Isolation Forest gives us an improvement factor of

about 3.04 (0.79/0.26), indicating that the ensembled model is approximately three times as good at identifying fraudulent transactions according to the F1 measure.

Discussion and Conclusions

The proposed ensembled model leverages the strengths of SVM and Random Forest algorithms to effectively address the challenges presented by highly unbalanced datasets in fraud detection. The model's ability to significantly reduce false positives and improve the detection of fraudulent transactions highlights its potential applicability in real-world scenarios. Future work will focus on comparing this model against other advanced machine learning techniques and exploring the integration of additional data sources to further enhance its predictive power. **Selecting more advanced model and tuning of these models will further be explored.**

The novel approach of focusing on filtering the legitimate transactions represents a strategic shift that could redefine best practices in financial fraud detection or unbalanced datasets in general.

Reference: Dataset