Machine Learning Final Report

Project Title:

Credit Card Fraud Detection Using Unsupervised anomaly algorithms

Group Members:

Name	Enrolment Number
Mano Vishnu A	U101115FEC012
Sai Chaitanya Kathiri	U101115FCS137

Project Idea:

Financial fraud is an ever growing menace with far consequences in the financial industry. Data mining had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons - first, the profiles of normal and fraudulent behaviours change constantly and secondly, credit card fraud data sets are highly skewed.

The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used.

The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications. We will be using a local outlier factor from ski kit learn package to calculate anomaly scores and Isolation force algorithm.

Dataset Link:

https://www.kaggle.com/mlg-ulb/creditcardfraud (Source: Kaggle)

Size of Dataset: 68MB

Tools and Technologies Used:

- **Platform:** Jupyter Notebook, Python 3.7.1
- **Packages:** Numpy, Scipy, Pandas, Seaborn, MatPlotLib

Algorithms Used:

- Isolation Forest Algorithm
- Local Outlier Factor (LOF)

Results:

```
import  sys
import numpy
import pandas
import matplotlib
import seaborn
import scipy
import sklearn

print ('Python: {}'.format(sys.version))
print ('Numpy: {}'.format(numpy.__version__))
print ('Pandas: {}'.format(pandas.__version__))
print ('MatplotLib: {}'.format(matplotlib.__version__))
print ('Seaborn; {}'.format(seaborn.__version__))
print ('Scipy; {}'.format(scipy.__version__))
print ('Sklearn; {}'.format(sklearn.__version__))
```

```
Python: 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)]
Numpy: 1.15.4
Pandas: 0.23.4
MatplotLib: 3.0.2
Seaborn; 0.9.0
Scipy; 1.1.0
Sklearn; 0.20.1
```

```
# Import the Necessary Packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the Dataset from the CSV file using Pandas
data = pd.read_csv('creditcard.csv')
```

```
# Explore the dataset
print(data.columns)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```


```
# Load the Dataset from the CSV file using Pandas
data = pd.read_csv('creditcard.csv')
```

```
# Explore the dataset
print(data.columns)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```


```
print(data.shape)
```

```
(284807, 31)
```

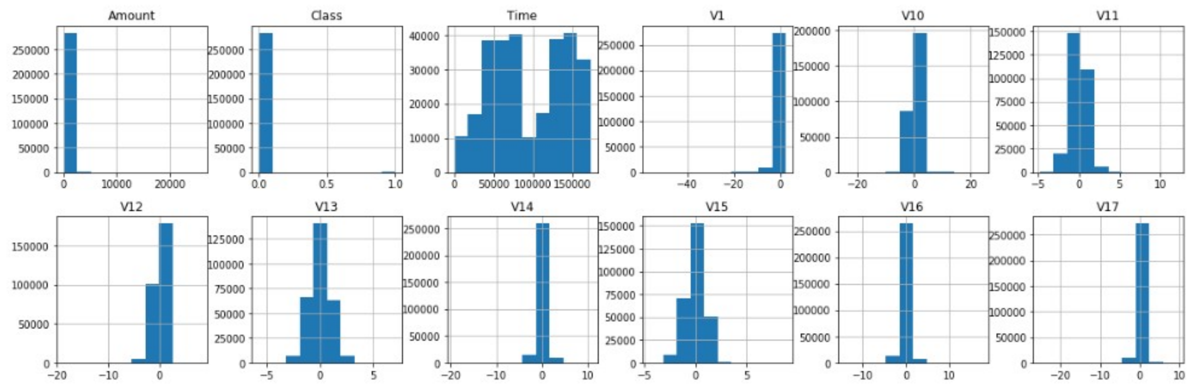
```
print(data.describe())
```

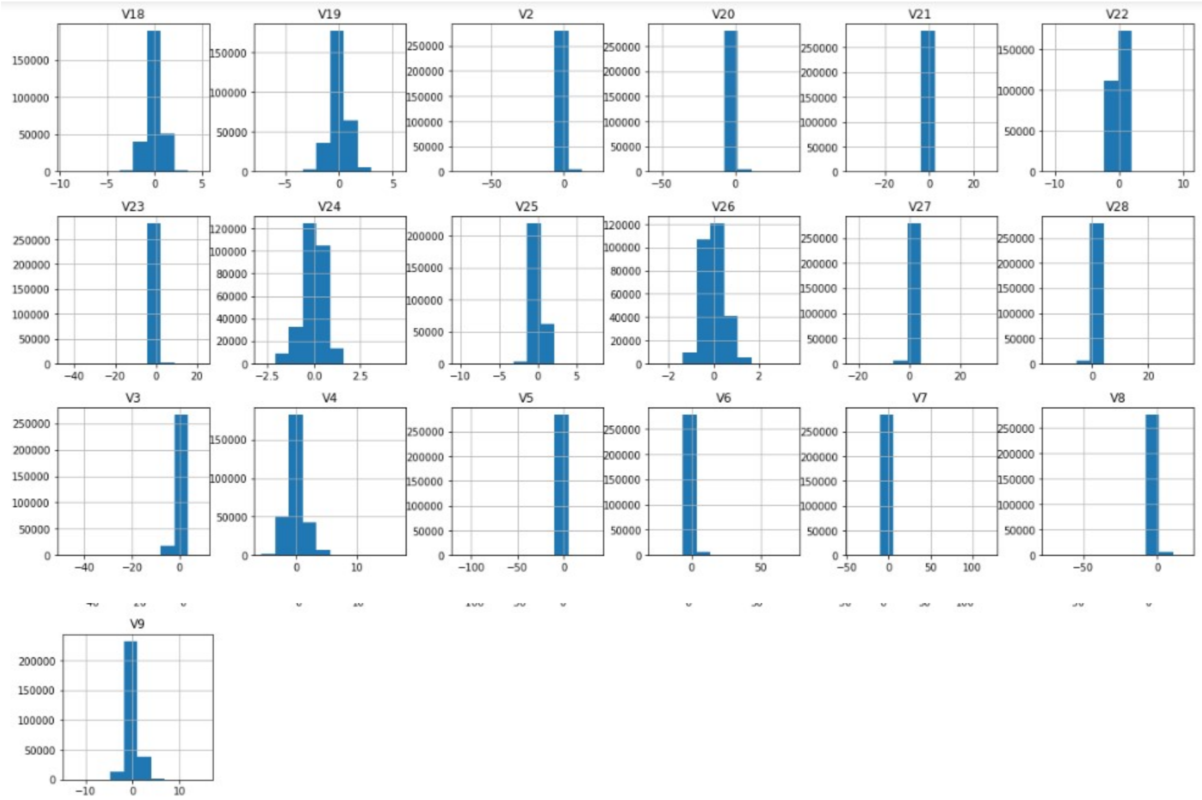
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.165980e-15  3.416908e-16 -1.373150e-15  2.086869e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.490107e-15 -5.556467e-16  1.177556e-16 -2.406455e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

           ...              V21           V22           V23           V24  \
count      ...     2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean       ...     1.656562e-16 -3.444850e-16  2.578648e-16  4.471968e-15
std        ...     7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min        ...    -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%        ...    -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%        ...    -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%        ...     1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max        ...     2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

```
data = data.sample(frac = 1.0, random_state = 1)
print (data.shape)
```

```
(284807, 31)
```

```
#plot a Histogram of each parameter
data.hist(figsize = (20,20))
plt.show()
```

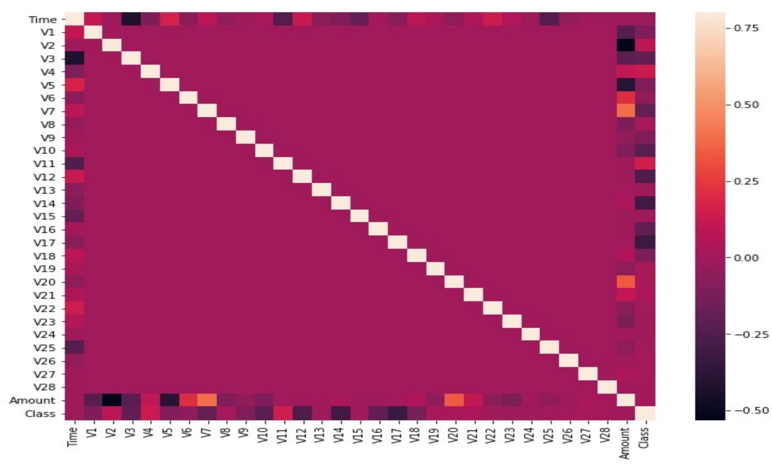





```
# determine number of fraud cases in dataset
Fraud = data[data['Class']==1]
Valid = data[data['Class']==0]

outlier_fraction = len(Fraud) / float(len(Valid))
print(outlier_fraction)

print('Fraud Cases: {}'.format(len(Fraud)))
print('Valid Cases: {}'.format(len(Valid)))
```

```
0.0017304750013189597
Fraud Cases: 492
Valid Cases: 284315
```

```
# Correlation Matrix
corrmat = data.corr()
fig = plt.figure(figsize=(12,9))

sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



```
# Get all the colums from the Dataframe
columns = data.columns.tolist()

#Filter the columsn to remove data we do not want
columns = [c for c in columns  if c not in ["Class"]]

# Store the variable we will be predicting on
target = "Class"
X = data[columns]
Y = data[target]

# print the shapes of X and Y
print(X.shape)
print(Y.shape)
```

```
(284807, 30)
(284807,)
```

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

# define a random state
state = 1

# define the outlier detection methods
classifiers = {
    "Isolation Forest" : IsolationForest(max_samples = len(X),
                                         contamination = outlier_fraction,
                                         random_state = state),
    "Local Outlier Facotr": LocalOutlierFactor(
    n_neighbors =20,
    contamination = outlier_fraction)
}
```

```
# Fit the model
plt.figure(figsize=(9, 7))
n_outliers = len(Fraud)


for i, (clf_name, clf) in enumerate(classifiers.items()):

    # fit the data and tag outliers
    if clf_name == "Local Outlier Factor":
        y_pred = clf.fit_predict(X)
        scores_pred = clf.negative_outlier_factor_
    else:
        clf.fit(X)
        scores_pred = clf.decision_function(X)
        y_pred = clf.predict(X)

    # Reshape the prediction values to 0 for valid, 1 for fraud.
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1

    n_errors = (y_pred != Y).sum()

    # Run classification metrics
    print('{}: {}'.format(clf_name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
```

```
Local Outlier Factor: 97
0.9965942207085425
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     28432
          1       0.02      0.02      0.02        49

avg / total       1.00      1.00      1.00     28481

Isolation Forest: 71
0.99750711000316
             precision    recall  f1-score   support

          0       1.00      1.00      1.00     28432
          1       0.28      0.29      0.28        49

avg / total       1.00      1.00      1.00     28481
```

Conclusion:

The Isolation Forest Algorithm performs better when compared to Local Outlier factor in Fraud Detection

Reference Papers:

1. Credit Card Fraud Detection using machine learning models and collating machine models, Navanshu Khare and Saad Yunus Sait, International Journal of pure and Applied Mathematics, Volume 118 No.20 2018, 825-838.
2. Credit Card Fraud Detection using Machine Learning Techniques: A Comparative Analysis. John O. Awoyemi, Adebayo O. Adetunmbi, Samuel A. Oluwadare, ICCNI 29- 31 Oct 2017.
3. Detecting Credit Card Fraud by ANN and Logistic Regression. Yusuf Sahin, Ekrem Duman, IEEE 28 February 2014.