Transaction Fraudulence Analysis

Jamie, Martin (n10212361)  
CAB220 Project Report

# Executive Summary

This report details the exploration of fraudulent and non-fraudulent credit card transaction data; investigating various techniques to build a model. This was investigation was conducted to shed light on the ever-growing market that is fraud analysis. It investigates utilising logistic regression and unsupervised learning models to accurately determine the effect various fields have on detecting transactional fraud.

After conducting exploratory analysis into correlations between fraudulent transactions and various fields, several models were investigated. Initially logistic regression was investigated, however due to the complex nature of both the dataset and fraudulent transactions it was deemed not suitable. Supervised machine learning techniques were then explored. Classifiers were developed using several common classification models. These models were then compared to one another, giving insights into accuracy and how well they perform given test data. Lastly, an unsupervised learning model, the Isolation Forest, was investigated. This model is common in anomaly prediction and fraud analysis.

Conclusions were drawn from this to deduce potential classifiers to use for a predictive fraud analysis model, giving insight into the variables that could correlate to a fraudulent transaction. An Isolation Forest model was finally chosen. This model had a fraudulence true positive rate of 84.55%, predicting false negatives 5% of the time. This model could be utilised for anomaly prediction in transactions to detect fraudulence.

# 1. Introduction

Since the downturn of COVID-19, fraud has been on the rise across Australia. In Australia alone, an estimated $1.26 million has been lost across roughly 8000 fraudulent reports, just between January and July of this year. [1]

These are all transactions that could have been prevented with proper technological measures in place for fraud prevention. Financial institutions and banks are quickly adopting artificial intelligence to examine transactions on a global scale. Detection systems driven by artificial intelligence can offer a variety of benefits for fraud prevention, giving real-time analysis more accurate than that of a human. Algorithms can pick up anomalies in transactional data and act upon in effectively without the need for human interaction. [2]

# 2. Project Definition

This report investigates transactional data of fraudulent and non-fraudulent transactions, experimenting with the accuracy of regression, supervised learning, and unsupervised learning classifiers in determining fraudulent and non-fraudulent transactions.

A variety of predictive models should be investigated to predict fraudulent transactions with relative accuracy, as well as have insight into what variables may be a sign of fraudulent transactions. It is important that credit card companies can recognise fraudulent transactions so that customers are not charged for products they did not purchase, whilst also minimising false positives which may affect end-user experience by halting transactions that would otherwise be non-fraudulent.

This analysis utilised common Python packages generally used in Data Science to:

* Clean and structure data
* Graph
* Develop several predictive models
* Analyse developed models

(Refer to Appendix 2.1)

# 3. Data Preparation

The dataset contained transactional data over a period of two days. In this time 492 fraudulent transactions were made from a total 284,807. Most of the data fields have been transformed using PCA - prior to receiving the dataset - for confidentiality reasons. The table below summarises the fields in the dataset.

|  |  |  |
| --- | --- | --- |
| Name | Description | Remarks |
| Time | Seconds elapsed between each transaction and the first |  |
| V1 | PCA transformed variables |  |
| … |  |
| V28 |  |
| Amount | The amount charged on the transaction |  |
| Class | Whether the transaction was fraudulent or not | 0 = non-fraudulent  1 = fraudulent |

Table 1. Summarisation of transactional dataset variables

The dataset did not need any cleaning for null or mistyped values. However, the ‘Time’ and ‘Amount’ were transformed to PCA variables to explore and demonstrate an understanding towards principal component analysis (PCA).

(Refer to Appendix 3.1)

The idea behind PCA is to reduce the number of variables in the dataset, whilst preserving as much information as possible. It is a form of variable standardisation. Naturally, it is commonly used to reduce the dimensionality of larger datasets, converting variables of all types to a ranged variable. [3]

# 4. Data Exploration

Whilst most of the data can be seen to have little conceivable meaning, each variable still holds insights into correlations between one another.

Something that may be of concern is the disparity between the number of fraudulent and non-fraudulent transactions. The dataset is highly unbalanced, with fraudulent transactions accounting for 0.172% of all transactions. This is something to keep in mind when moving forward, as it could produce models with a ‘naïve’ behaviour that are otherwise useless. Precision, recall and F-1 score of each model will be used to provide a basic evaluation of these classifiers. [4]

# 5. Data Analysis

Initially the density of each variable was compared, splitting the results into fraudulent and non-fraudulent transactions to see it there were some commonality between the two. This could indicate insignificant variables that we could leave out of the models as they have little effect on the result.

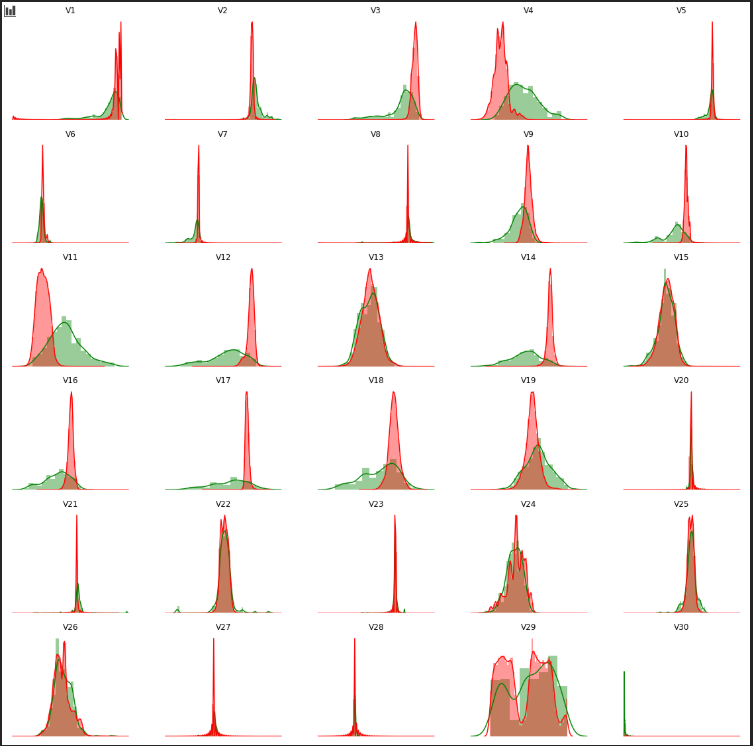


Figure 1: Transaction Density of Fraudulence per each variable

(Refer to Appendix 5.0.1)

Looking at Figure 1., there are a few variables that could potentially be insignificant. We can see these at V13, V15, V20, V22 and V25; just to name a few.

Testing their z-score against a critical value will determine if they are statistically significant or not. We can then use this information to determine whether we should drop them from the training and testing data, so as not to interfere with the various models results.

For this test, a p-value of 0.01 was chosen to ensure accuracy. This corresponds to a 1% chance the result is random. Due to the nature of the dataset, ensuring accuracy is vital. Any variables that may have similar density between the two should be dropped as they clearly do not correlate towards a fraudulent transaction. With a chosen p-value of 0.01, this gave a critical value of 2.5758. This means that any value below the critical value is of little significance when determining the fraudulence of a transaction. [5][6]

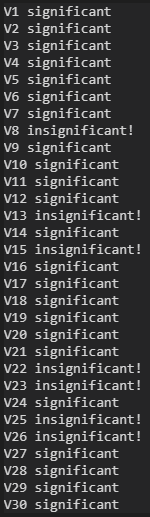


Figure 2: z-score results of significance

(Refer to Appendix 5.0.2)

In Figure 2., we can see that from this test we were able to drop V8, V13, V15, V22, V23, V25, and V26 as they can be dropped as they do not contribute significance in whether the transaction is fraudulent or not.

The dataset was then split into test (30%) and train (70%) data. This ensures the models are not overfit or underfit. Overfitting refers to the act of training a model “too well”, such that the dataset the model has been trained on is inferring correlation from noise in the dataset, potentially mean that feed other datasets, performs poorly. In contrast to this, underfitting means our model is potentially missing important trends that are not considered in the model’s predictions due to lack of data. [7]

(Refer to Appendix 5.0.3)

Due to the highly imbalanced nature of this dataset, the potential for overfitting and underfitting is high. There may not be enough fraudulent transactional data for the models to accurately interpret relationships between the classifications or consider trends that are of importance.

To interpret the validity and accuracy of the developed models, classification reports were used to score them. A classification report gathers three main statistics of how the classification model did at predicting fraudulent and non-fraudulent transactions. These three statistics include the precision, recall and F-1 score. The precision tells us the percentage of the predictions that were classified correctly for each class. These classes are ‘0’ (non-fraudulent) and ‘1’ (fraudulent). The recall is the percentage of true positives predicted for each class. The F-1 is the harmonic mean between precision and recall. [4][8]

For this experimentation, the F-1 score is focused on, as this gives us a balanced result that considers both the precision and recall. [8]

## 5.1 Regression

### Logistic Regression

The classification analysis initially began with investigating the use of a logistic regression model on the dataset. Classification models suit best as fraudulence is non-linear.

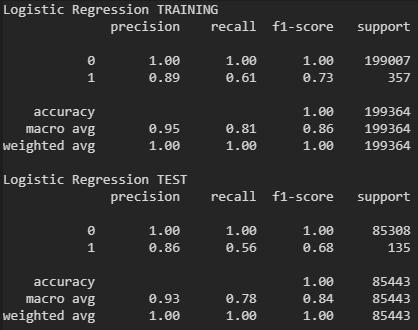


Figure 3: Logistic Regression classification report

(Refer to Appendix 5.1.1)

Testing the model with the test data split (Figure 3.), the fraudulence predictor has a high precision and low recall. This infers that the mode cannot detect fraudulence well, detecting fraudulent transactions 56% of the time, but when it does it does so with an accuracy of 86%.

After further research into the use of logistic regression in it was clear that a logistic regression model was not useful. It is highly likely that there is not a simple relationship for classifying fraudulent transaction. A variety of the variables could potentially interact with one another and the model is a risk of being overfitted. Thus, supervised, and unsupervised learning techniques were investigated. [9][10]

## 5.2 Supervised Techniques

A few supervised machine learning classification models were experimented with to give insight and comparison into the accuracy of their predictions. Supervised learning involves providing the training input with the classification. The model then uses this classification to evaluate the accuracy on its own, adjusting where necessary. [11]

K-folds cross validation is a model scoring technique that is primarily used in machine learning to estimate the skill of a model. K being a positive numerical value, is the number of groups the dataset gets split into. A k value of ten was chosen as this is commonly used as it generally results in an estimate with low bias and modest variance. [7][12]

Often a multitude of hyper-parameters are used in a cross-validation method. However – for the sake of simplicity – only a single hyper-parameter was used per classifier.

### Decision Tree

A decision tree classifier is a simple classification model used to iteratively split data according to a particular parameter. These uses a tree structure creating subsets of data at each node based on cut-off values that the model decides provides the most value. In this case we are using a classification decision tree to attempt to classify a ‘yes’ or ‘no’ answer of whether a transaction is fraudulent or not. [13]

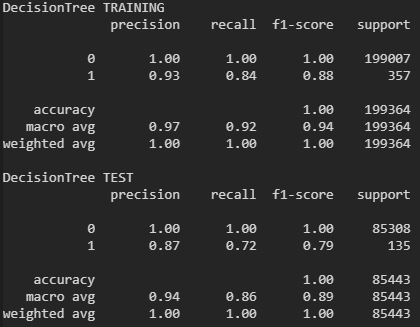


Figure 4: Decision Tree classification report

(Refer to Appendix 5.2.1)

In Figure 4., we can see how the decision tree model has performed on the test data. Whilst accurate at making predictions on non-fraudulent transactions, the model still has issues predicting fraudulent transactions to a high degree with an F-1 score of 79%. This could potentially be too low for detecting anomalies of this nature.

### Nearest Neighbours

The K-Nearest Neighbours classifier assumes that similar things exist near one another. This classifier attempts to groups predictors by closeness, and predicts outcomes based on this. K is how many neighbours near the model should draw its conclusions from. When K is equal to one, the model would only look to its first neighbours to develop a prediction. As K grows, the model looks further away from its closest neighbours. [14]

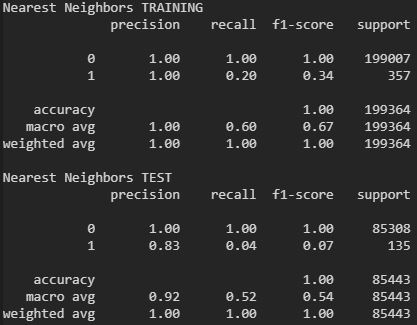


Figure 5: K-Nearest Neighbours classification report

(Refer to Appendix 5.2.2)

In Figure 5., it can be seen that the nearest neighbours implementation performs poorly when given test data. Of the 135 fraudulent transactions tested, it only predicted 4%. With this classifier hyper-parameters of 1-10 were given for k to the cross-validation method. Increasing k over 10 could potentially give better results. However, naturally these predictions would become noisier as the model looks at neighbours further away.

### Support Vector Machine (SVM)

Support vector machines are another common classification learning method. The idea is to find a hyperplane that divides the dataset, resulting in a best-fit segregation of the two classes. SVM can show failure on larger datasets. This is due to potential amounts of noise that larger datasets generally hold. As well as this, a highly imbalanced dataset may mean the SVM cannot draw conclusions properly. [15]

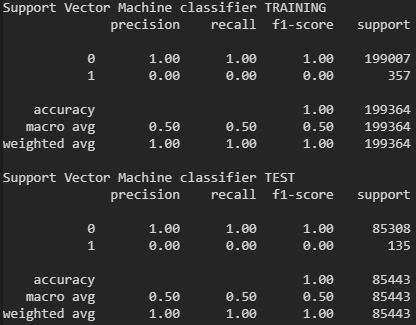


Figure 6: Support Vector Machine classification report

(Refer to Appendix 5.2.3)

From the experimentation (see Figure 6), we can clearly see failure from the SVM model. Whilst the classification report shows it is 100% accurate at predicting normal cases, the model has a 0% success rate at predicting fraud. This is likely due to the aforementioned issues with using SVM on large imbalanced datasets, as well as the potential complexity that fraudulence holds. SVM may lack the necessary complexity to deduce complex relationships of this nature.

### Neural Network

An artificial neural network seeks to interpret relationships in the given dataset. This is done through a process that is likened to how the human brain operates. Just as neurons create a network in the brain, so do nodes in an artificial neural network. The algorithm takes input through multiple interconnected layers. Each layer is a column of nodes, where each n column is connected to columns n-1 and n+1. A key advantage of using neural networks is the ability for it to model complex, non-linear patterns. [16]

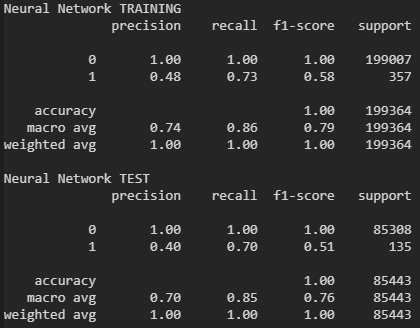


Figure 7: Neural Network classification report

(Refer to Appendix 5.2.4)

From the neural network experimentation (see Figure 7.), the precision and recall of this model are quite poor. Considering precision and recall, the harmonic mean of this model’s fraudulence prediction is 51%. Whilst 100% accurate at deducing normal transactions, this model struggles to find patterns with fraudulence. Neural networks follow a generalisation method. Whilst useful this could be the downfall as to why this model has performed poorly. Tuning the hyper-parameters on this classifier could potentially give more a more accurate prediction model.

## 5.3 Unsupervised Techniques

As opposed to supervised learning, unsupervised learning models are fed unclassified data. In doing so, the algorithm attempts to make sense of the data by interpreting trends of its own finding. [11]

### Isolation Forest

The isolation forest classifier is used for anomaly detection and is a common method for fraud analysis. An isolation forest model is fit by feeding normal (non-fraudulent) points to it. The model will then draw its prediction by identifying instances that are averse to its known normal, classifying these as anomalies. [17][18]

In this case, the scikit Isolation Forest was used. All inliers (normal transactions) from the dataset were fitted to the classifier. From this, a model was gained that can detect outlier (fraudulent transactions) anomalies. For non-fraudulent transactions, the model was 95.74% accurate in classifying these as non-fraudulent. On the other hand, for fraudulent transactions, the model was 84.55% accurate in classifying these as fraudulent (See Figure 8).



Figure 8: Isolation Forest accuracy output

(Refer to Appendix 5.3.1)

# Conclusion

Through exploring various classification models, investigating both supervised and unsupervised learning techniques, several prediction models for analysing fraudulent transactions were made. With this, one can detect fraudulent transactions 84.55% of the time.

Throughout research, a better understanding of classification models was gained. Giving insight into how reliable models can be developed for accurate predictions. With regards to all models there is some potential that overfitting and underfitting occurred. With most of the supervised classifiers 100% accuracy was attained for normal non-fraudulent transactions, however on the other hand, the models could not make sense of the fraudulent transactions. This could indicate there is minimal correlation in the data, or at least not enough fraudulent transactions for the model to accurately determine trends in fraudulence.

Given more time, one could potentially investigate tuning hyper-parameters to see if different results are gained. Largely, more unsupervised methods could also be explored as they seem to have the most success and are commonly used in fraud analysis and anomaly detection.

Speed is also a consideration for real-world usage, something that is largely out of scope for this investigation but nevertheless, potentially critical for analysing transactions in real-time - on a global scale - in an efficient and timely manner. Whilst these times were not noted, the Isolation Forest seemed to have the most efficiency, with some of the supervised models taking upwards of five minutes to train.

Another consideration could be training models per user. Training a model on only the users’ data might give better insight into fraudulent, as well as being efficient in terms of model training and usage. There are potentially better trends to be found in training models on a specific user and utilising that model for that user when transactions are made by them.

Whilst the dataset was highly imbalanced, high accuracy was able to be seen on the Isolation Forest. Hopefully with more balanced data, further accuracy would be possible. 15% of fraudulent transactions going potentially unnoticed is still rather high; being roughly 1 in 7 fraudulent transactions.

# References

[1] ‘Government impersonation scams on the rise | Scamwatch’. <https://www.scamwatch.gov.au/news-alerts/government-impersonation-scams-on-the-rise> (accessed Oct. 19, 2020).  
[2] ‘AI’s Expanding Role in Fraud Detection | Roman’. <https://www.romansearch.com/blog/artificial-intellegence-ai-fraud-detection> (accessed Oct. 19, 2020).  
[3] ‘A One-Stop Shop for Principal Component Analysis | by Matt Brems | Towards Data Science’. <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c> (accessed Oct. 19, 2020)  
[4] ‘Handling imbalanced datasets in machine learning | by Baptiste Rocca | Towards Data Science’. <https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28> (accessed Oct. 20, 2020)  
[5] ‘How to Calculate Critical Values for Statistical Hypothesis Testing with Python’. <https://machinelearningmastery.com/critical-values-for-statistical-hypothesis-testing/> (accessed Oct. 20, 2020)  
[6] ‘Statistical Significance Explained | by Will Koehrsen | Towards Data Science’. <https://towardsdatascience.com/statistical-significance-hypothesis-testing-the-normal-curve-and-p-values-93274fa32687> (accessed Oct. 20, 2020)  
[7] ‘Train/Test Split and Cross Validation in Python | by Adi Bronshtein | Towards Data Science’. <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6> (accessed Oct. 20, 2020)  
[8] ‘Accuracy, Precision, Recall or F1? | by Koo Ping Shung | Towards Data Science’. <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9> (accessed Oct. 20, 2020)  
[9] ‘4.2 Logistic Regression | Interpretable Machine Learning’. <https://christophm.github.io/interpretable-ml-book/logistic.html> (accessed Oct. 20, 2020)  
[10] ‘(Tutorial) Understanding Logistic REGRESSION in PYTHON - DataCamp’. <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python> (accessed Oct. 20, 2020)   
[11] ‘Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog’. <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/> (accessed Oct. 20, 2020)  
[12] ‘A Gentle Introduction to k-fold Cross-Validation’. <https://machinelearningmastery.com/k-fold-cross-validation/> (accessed Oct. 20, 2020)  
[13] ‘4.4 Decision Tree | Interpretable Machine Learning’. <https://christophm.github.io/interpretable-ml-book/tree.html> (accessed Oct. 20, 2020)  
[14] ‘Machine Learning Basics with the K-Nearest Neighbors Algorithm | by Onel Harrison | Towards Data Science’. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761> (accessed Oct. 20, 2020)  
[15] ‘Support Vector Machines: A Simple Explanation’. <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html> (accessed Oct. 20, 2020)  
[16] ‘First neural network for beginners explained (with code) | by Arthur Arnx | Towards Data Science’. <https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf> (accessed Oct. 20, 2020)  
[17] ‘Outlier Detection with Isolation Forest | by Eryk Lewinson | Towards Data Science’. <https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e> (accessed Oct. 20, 2020)  
[18] ‘Anomaly Detection Using Isolation Forest in Python | Paperspace Blog’. <https://blog.paperspace.com/anomaly-detection-isolation-forest/> (accessed Oct. 20, 2020)

# Appendix

## 2

### 2.1

from IPython import get\_ipython

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from sklearn.decomposition import PCA

from sklearn import metrics

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.neural\_network import MLPClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import IsolationForest

import warnings

warnings.filterwarnings('ignore')

get\_ipython().run\_line\_magic('matplotlib', 'inline')

## 3

### 3.1

# %%

df = pd.read\_csv("./creditcard.csv")

cols = ['Time', 'Amount']

pca = PCA()

pca.fit(df[cols])

X\_pca = pca.transform(df[cols])

df['V29'] = X\_pca[:, 0]

df['V30'] = X\_pca[:, 1]

df.drop(cols, axis=1, inplace=True)

## 5.0

### 5.0.1

columns = df.drop('Class', axis=1).columns

grid = plt.GridSpec(6, 5)

plt.figure(figsize=(20, 10\*2))

for n, col in enumerate(df[columns]):

    ax = plt.subplot(grid[n])

    sns.distplot(df[df.Class == 1][col], color='g')

    sns.distplot(df[df.Class == 0][col], color='r')

    ax.axes.xaxis.set\_visible(False)

    ax.axes.yaxis.set\_visible(False)

    ax.set\_frame\_on(False)

    ax.set\_title(str(col))

    ax.set\_xlabel('')

plt.show()

### 5.0.2

# https://www.kaggle.com/sabanasimbutt/anomaly-detection-using-unsupervised-techniques

columns = df.drop('Class', axis=1).columns

non = df[df.Class == 0]

fraud = df[df.Class == 1]

size = len(fraud)

significant = ['Class']

critical\_value = 2.5758

for i in columns:

    mean = non[i].mean()

    f\_mean = fraud[i].mean()

    f\_std = fraud[i].std()

    z = (f\_mean - mean) / (f\_std/np.sqrt(size))

    if(abs(z) >= critical\_value):

        print(i, "significant")

        significant.append(i)

    else:

        print(i, "insignificant!")

df = df[significant]

df.shape

### 5.0.3

# https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

cols = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10',

        'V11', 'V12', 'V14', 'V16', 'V17', 'V18', 'V19', 'V20',

        'V21', 'V24', 'V27', 'V28', 'V29', 'V30']

X = np.array(df[cols])

y = np.array(df['Class'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.3, random\_state=1)  
5.0.3

# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification\_report.html

def classify(name, clf):

    '''function to test all models'''

    # fit the classifier with train data

    m = clf.fit(X\_train, y\_train)

    # run model prediction with training data

    print(name, "TRAINING")

    print(metrics.classification\_report(y\_train, m.predict(X\_train)))

    # run model prediction with test data

    print(name, "TEST")

    print(metrics.classification\_report(y\_test, m.predict(X\_test)))

## 5.1

### 5.1.1

# https://realpython.com/logistic-regression-python/

# https://datascience.foundation/sciencewhitepaper/understanding-logistic-regression-with-python-practical-guide-1

lr = LogisticRegression()

classify('Logistic Regression', lr)

## 5.2

### 5.2.1

# https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html

# https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

# possible values for the parameter max\_depth

max\_depths = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, None]

# parameters that will be used in cross validation

params = {"max\_depth": max\_depths}

# setup cross validation using a Decision Tree classifer,

# random\_state set to 1 for repeatability and cv set to 10 folds

classify('DecisionTree', GridSearchCV(

    DecisionTreeClassifier(random\_state=1), params, cv=10))

### 5.2.2

# https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

# possible values for the parameter n\_neighbors

n\_neighbors = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# parameters that will be used in cross validation

params = {"n\_neighbors": n\_neighbors}

# setup cross validation using a Nearest Neighbors classifer,

# random\_state set to 1 for repeatability and cv set to 10 folds

classify('Nearest Neighbors', GridSearchCV(

    KNeighborsClassifier(), params, cv=10))

### 5.2.3

# https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

# possible values for the parameter C

Cs = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# parameters that will be used in cross validation

params = {"C": Cs}

# setup cross validation using a Support Vector Machine classifer,

# random\_state set to 1 for repeatability and cv set to 10 folds

classify('Support Vector Machine classifier',

         GridSearchCV(SVC(random\_state=1), params, cv=10))

### 5.2.4

# https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html

# possible values for the parameter hidden\_layer\_sizes

hidden\_layer\_sizes = [50, 75, 100, 125, 150, 175,

                      200, 225, 250, 275, 300, 325, 350, 375, 400]

# parameters that will be used in cross validation

params = {"hidden\_layer\_sizes": hidden\_layer\_sizes}

# setup cross validation using a Neural Network classifer,

# random\_state set to 1 for repeatability

# tol set to 1e-2 so that the maximum iteration is never reached

# cv set to 10 folds

classify('Neural Network', GridSearchCV(

    MLPClassifier(random\_state=1), params, cv=10))

## 5.3

### 5.3.1

# https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

# separate inliers and outliers

non = df[df.Class == 0].drop(['Class'], axis=1)

fraud = df[df.Class == 1].drop(['Class'], axis=1)

# setup and fit IsolationForest with non-fraudulent transactions

clf = IsolationForest(random\_state=1)

clf.fit(non)

def accuracy(values, type):

    '''tests accuracy of given values'''

    accuracy = list(values).count(type)/values.shape[0]

    return '{accuracy:.2f}%'.format(accuracy=accuracy \* 100)

# test accuracy of non-fraudulent transactions

print("Non-Fraudulent Transaction Accuracy:",

      accuracy(clf.predict(non), 1))

# test accuracy of fraudulent transactions

print("Fraudulent Transaction Accuracy:",

      accuracy(clf.predict(fraud), -1))