ARYABHATTA KNOWLEDGE UNIVERSITY

(Patna, Bihar – 800001)

DEPARTMENT OF SCIENCE AND TECHNOLOGY

(Government of Bihar)

A Project on

"CREDIT CARD FRAUD DETECTION"

Submitted in Partial fulfillment of requirements for the award of degree of

Bachelor of Technology in

Computer Science and engineering

By

"**Rahul Sharma**"	Reg. No. 16105108026
"**Nitesh Kumar**"	Reg. no. 16105108033
"**Shivam Kumar**"	Reg. No. 16105108018
"**Kundan Kumar**"	Reg. No. 16105108010

Under the esteemed guidance of

Prof. Mintu Singh

(Assistant Professor)

Department of Computer Science and Engineering

BHAGALPUR COLLEGE OF ENGINEERING

(Sabour, Bhagalpur, Bihar – 813210)

ARYABHATTA KNOWLEDGE UNIVERSITY
(Patna, Bihar – 800001)

BHAGALPUR COLLEGE OF ENGINEERING
(Sabour, Bhagalpur, Bihar – 813210)

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

This is to certify that this project report entitled "**Credit Card Fraud Detection**" submitted to **Bhagalpur College of Engineering, Bhagalpur** is a bonafide record of work done by

"**Rahul Sharma**"	Reg. No. 16105108026
"**Nitesh Kumar**"	Reg. no. 16105108033
"**Shivam Kumar**"	Reg. No. 16105108018
"**Kundan Kumar**"	Reg. No. 16105108010

under my supervision during 8th semester (2016 batch).

Prof. Mintu Singh
(Assistant Professor)
Department of Computer Science and Engineering
Bhagalpur college of Engineering, Bhagalpur

Prof. Mani Kant Mandal
(Head of Department)
Department of Computer Science and Engineering
Bhagalpur college of Engineering, Bhagalpur

Name of Examiner	Signature with date
1. _____	_____
2. _____	_____

DECLARATION BY AUTHOR

This is to declare that this report has been written by us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be plagiarized, we shall take full responsibility for it.

"**Rahul Sharma**"	Reg. No. 16105108026
"**Nitesh Kumar**"	Reg. no. 16105108033
"**Shivam Kumar**"	Reg. No. 16105108018
"**Kundan Kumar**"	Reg. No. 16105108010

ACKNOWLEDGEMENTS

It is a pleasant duty to express our sincere gratitude to Aryabhatta Knowledge University, Patna for giving us such a wonderful opportunity to complete our project.

We ascribe countless number of thanks to Hon'ble principal, Dr. Nirmal Kumar and TEQIP team, Bhagalpur College of Engineering, Bhagalpur.

We would like to acknowledge our guide mentor Prof. Mintu Singh, who helped us and guided us throughout the course of completion of this project on "Credit Card Fraud Detection".

Lastly, and most importantly, we thank our parents for their love, support and motivation throughout our life. We also place on record, our sense of gratitude to one and all who, directly or indirectly, have lent their helping hand in this venture.

"**Rahul Sharma**" Reg. No. 16105108026

"**Nitesh Kumar**" Reg. no. 16105108033

"**Shivam Kumar**" Reg. No. 16105108018

"**Kundan Kumar**" Reg. No. 16105108010

TABLE OF CONTENTS

1. CREDIT CARD FRAUD

As we are heading towards a digitally connected world the thieves and scammers have also adapted to cheat us digitally. Most of these scams are done through fraudulent credit card transaction. Globally each year a total of $27.85 billion fraud payment card or credit card transactions are done. This is a big hit to digital security and a hurdle in connecting people digitally. To stop these kind of scams or frauds it is necessary to identify the fraudulent transactions in real time identify the regions of scams and take action. Since a legitimate person who owns the credit card has a particular way of spending, which may not be identified by the humans but these trends can be identified by the computer using proper algorithms.

2. OUTLIERS

Statistically, outliers are the data points that do not fit in the obvious trends in collection of many data points (dataset).

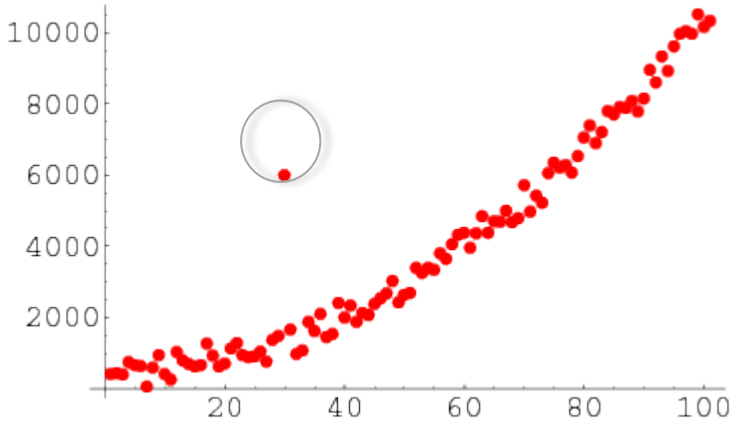

Figure 1: Outlier

In the above figure we can see that the point circled stands apart from rest of the dataset, as it does not follow their trend. Hence it is consider as outliers. Outliers' detection is very important as fraud transactions are observed to not follow the trend of expenditure by a legitimate owner of the card. Hence, identifying the outliers may lead to identification of fraud transactions.

Some more examples of outliers in real world are –

- When one student averages over 90% while the rest of the class is at 70% – a clear outlier
- While analyzing a certain customer's purchase patterns, it turns out there's suddenly an entry for a very high value. While most of his/her transactions fall below Rs.10, 000, this entry is for Rs.1, 00,000. It could be an electronic item purchase – whatever the reason, it's an outlier in the overall data

- How about Usain Bolt? Those record breaking sprints are definitely outliers when you factor in the majority of athletes

Outliers are of two types: **Univariate and Multivariate**. A univariate outlier is a data point that consists of extreme values in one variable only, whereas a multivariate outlier is a combined unusual score on at least two variables. Suppose you have three different variables – X, Y, and Z. If you plot a graph of these in a 3-D space, they should form a sort of cloud. All the data points that lie outside this cloud will be the multivariate outliers.

3. PROBLEM STATEMENT

Given the dataset of previous transactions, we have to develop a model that identifies the unseen transactions whether fraud or legitimate with very high accuracy.

The dataset used for training the model has principal components of the original dataset and also are also normalized. This is done to protect the privacy of the customer while maintaining the integrity of the dataset. However, the dataset is highly skewed i.e. there far more non-fraudulent transactions than the fraud transactions. The dataset may have following attributes

- Amount of transaction
- Time of transaction
- Gateway for payment
- Gender of legitimate owner
- Region of transaction, etc.

4. WHAT IS CLASSIFICATION

Given an object with a set of independent attributes, such that the object can be sought out of any of the available category (or class) then the process is known as classification. Classification is a supervised learning because we know the expected outcome before actually running the simulations on training data. Classification has a number of application in real world, from predicting the churn of customers to identifying the cancer type (benign or malignant). One such application can be identifying the fraudulent transactions out of given dataset. This is so because the outcome of independent attributes is binary (composed of two classes), whether a given transaction is legitimate or not. Hence we will explore some algorithms of machine learning and deep learning which uses classification and try to apply those to our problem.

5. ML BASED CLASSIFICATION

In machine learning there are numerous classification algorithm such as logistic regression, SVM, KNN classification, etc. Out of these ***Random Forest Classification*** is one of the best method. But before we go through random forest algorithm, we must look the decision tree classifier which is at the heart of random forest algorithm.

5.1 Decision Tree

Decision Tree is a **supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.** In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm** described as follows –

1. Choose an attribute from your dataset
2. Calculate the significance of attribute in splitting of data
3. Split the data based on the value of the best (most predictive) attribute
4. Go to step 1 and repeat for other attributes

To calculate impurity of a node in decision tree, we use entropy formula given by

Entropy = -p (A) log (p (A)) – p (B) log (p (B))

Where,

p (A) = proportion or ratio of the category A and

p (B) = proportion or ratio of the category B

An attribute is selected for the split only if it has highest information gain after the split as compare to available attributes, information gain is given by the formula

Information Gain = Entropy before the split – Entropy after the split

5.2 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

5.3 eXtreme Gradient Boosting (XGBoost)

In boosting, the trees are built sequentially such that each subsequent tree aims to reduce the errors of the previous tree. Each tree learns from its predecessors and updates the residual errors. Hence, the tree that grows next in the sequence will learn from an updated version of the residuals. The base learners in boosting are weak learners in which the bias is high, and the predictive power is just a tad better than random guessing. Each of these weak learners contributes some vital information for prediction, enabling the boosting technique to produce a strong learner by effectively combining these weak learners. The final strong learner brings down both the bias and the variance. **Having a large number of trees might lead to overfitting. So, it is necessary to carefully choose the stopping criteria for boosting.**

The boosting ensemble technique consists of three simple steps:

- An initial model F_0 is defined to predict the target variable y. This model will be associated with a residual $(y – F_0)$
- A new model h_1 is fit to the residuals from the previous step
- Now, F_0 and h_1 are combined to give F_1, the boosted version of F_0. The mean squared error from F_1 will be lower than that from F_0:

$$F_1(x) <- F_0(x) + h_1(x)$$

To improve the performance of F_0, we could model after the residuals of F_2 and create a new model F_2:

$$F_2(x) <- F_1(x) + h_2(x)$$

This can be done for *'m'* iterations, until residuals have been minimized as much as possible:

$$F_m(x) <- F_{m-1}(x) + h_m(x)$$

Here, the additive learners do not disturb the functions created in the previous steps. Instead, they impart information of their own to bring down the errors.

Gradient boosting helps in predicting the optimal gradient for the additive model, unlike classical gradient descent techniques which reduce error in the output at each iteration since in gradient boosting, the average gradient component would be computed.

The following steps are involved in gradient boosting:

- $F_0(x)$ – with which we initialize the boosting algorithm – is to be defined:

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$

- The gradient of the loss function is computed iteratively:

$$r_{im} = -\alpha \left[\frac{\partial (L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}, \text{ where } \alpha \text{ is the learning rate}$$

- Each $h_m(x)$ is fit on the gradient obtained at each step
- The multiplicative factor y_m for each terminal node is derived and the boosted model $F_m(x)$ is defined:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

5.4 Experimental Results

Using the **Random Forest** classifier on the dataset we got an accuracy of 0.999625 , but to check overfitting we ran the simulations on k-fold cross validation with 10 folds and calculated the average of accuracies to get more reliable results. With 10-fold cross validation we got following results

Mean Accuracy	**Mean Standard Deviation**
0.9994432288655493	0.0002180114792249493

With confusion matrix we observed that there are

- 85301 True Positive
- 109 True Negative
- 5 False Positive
- 27 False Negative

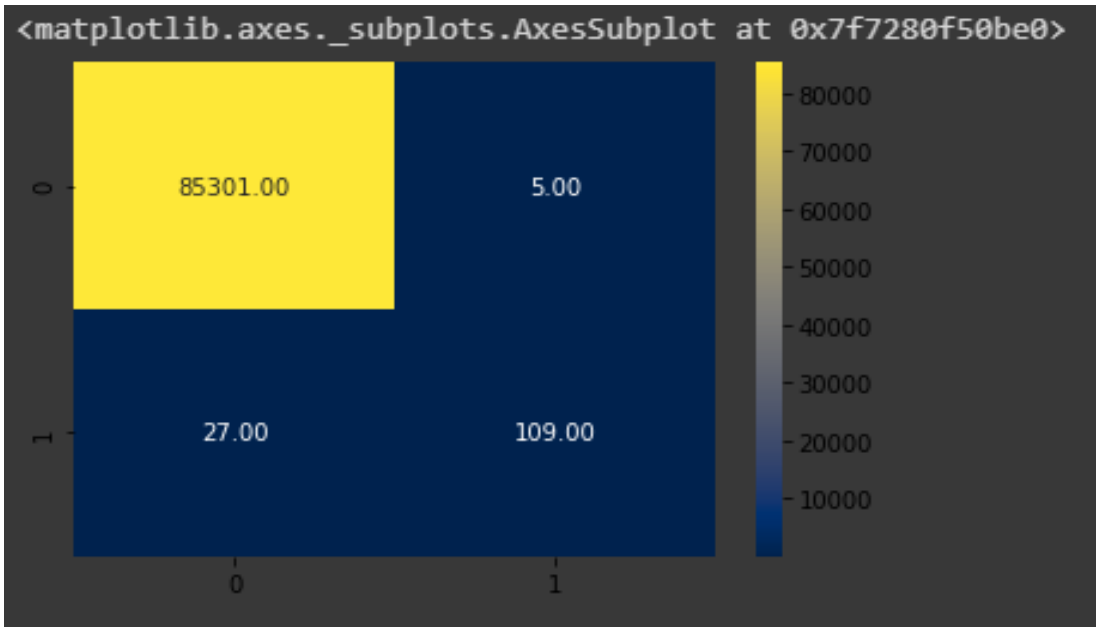

Figure 2: Confusion Matrix describing the results of Random Forest Classifier.

6. Artificial Neural Network Based Classification

Artificial Neural Network is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its **activation** or **node value.** The code implementing the ANN classifier for predicting the same is in appendix B. The implementation is done in Google Collaboratory Jupiter notebook.

6.1 Architecture

The ANN is composed of nodes, nodes can be of various types i.e. input nodes, hidden nodes and output nodes. Combination of nodes is described as layers, hence the name input layer, hidden layer and output layer. The learning happens in hidden layers. Since we are constructing a classifier so, there will be a single output node.

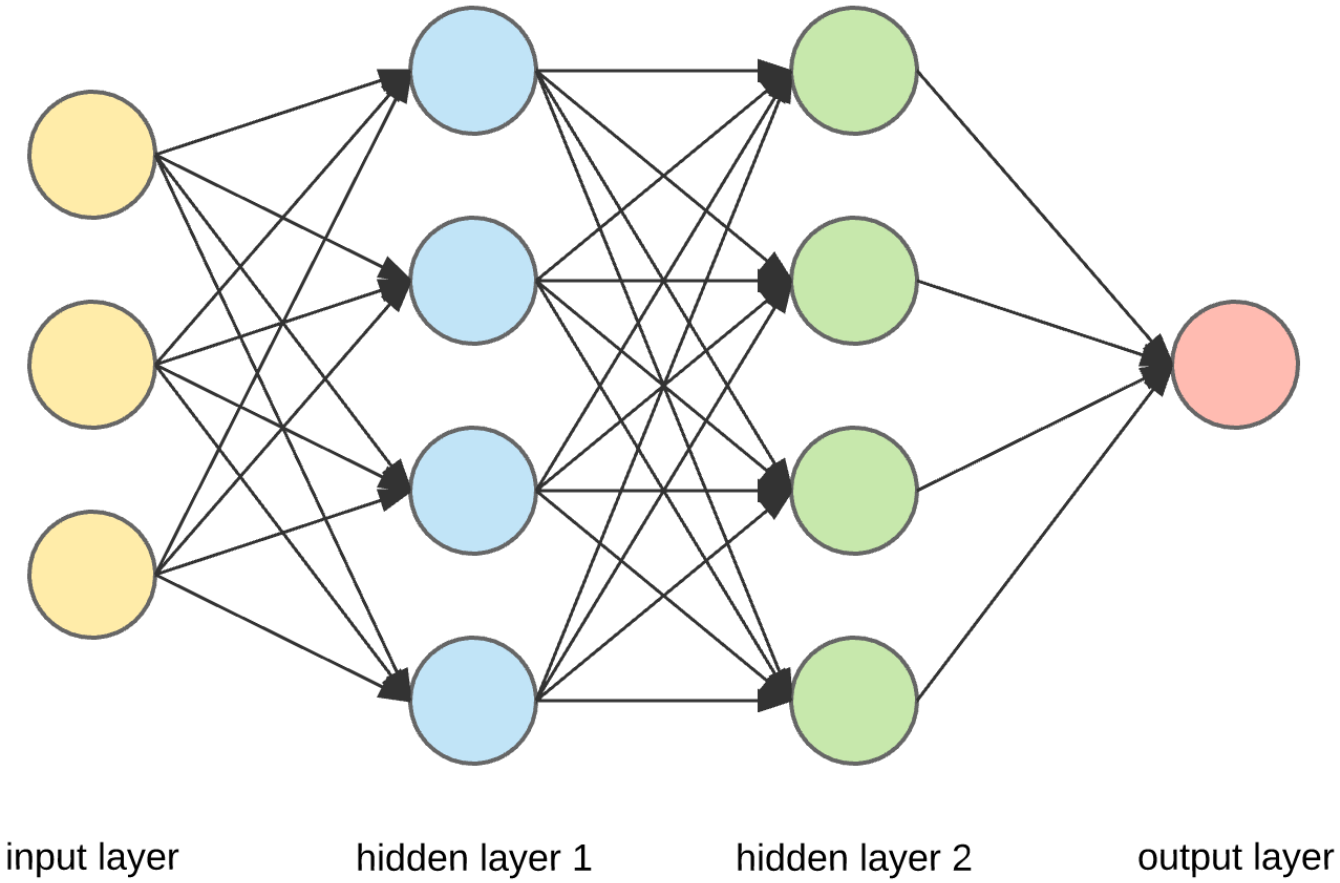

Figure 3: Architecture of an ANN Classifier

By noting i the i^{th} layer of the network and j the j^{th} hidden unit of the layer, we have:

$$z_j^{[i]} = {w_j^{[i]}}^T x + b_j^{[i]}$$

Where we note w, b, z the weight, bias and output respectively.

Activation function – Activation functions are used at the end of a hidden unit to introduce non-linear complexities to the model. In our model we have used sigmoid activation function, which also facilitates with confidence value of categorical output.

$$g(z) = \frac{1}{1 + e^{-z}}$$

Figure 4: Sigmoid Function

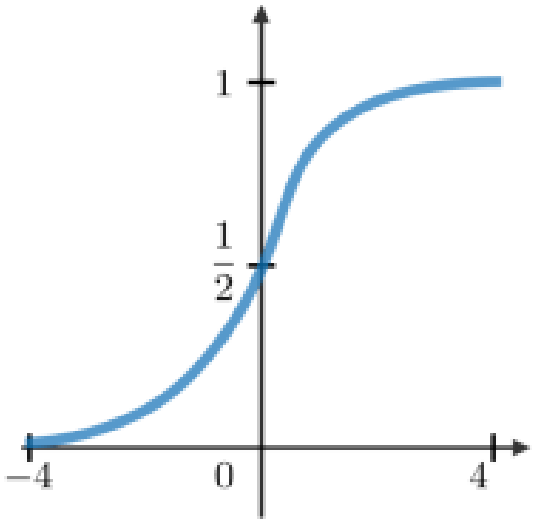

Figure 5: Graphical representation of sigmoid function

In context of artificial neural network, the cross entropy loss is used to calculate the loss in accuracy which the artificial neural network try to minimize. The cross entropy loss function is given by the following formula

$$L(z,y) = -\Big[y \log(z) + (1 - y) \log(1 - z)\Big]$$

Figure 6: Binary Cross Entropy Function

Where, y is vector of predicted values and z is vector of target values (or true values).

Learning rate – The learning rate, often noted η, indicates at which pace the weights get updated. This can be fixed or adaptively changed. The current most popular method is called Adam, which is a method that adapts the learning rate.

Backpropagation – Backpropagation is a method to update the weights in the neural network by taking into account the actual output and the desired output. The derivative with respect to weight w is computed using chain rule and is of the following form:

$$\frac{\partial L(z,y)}{\partial w} = \frac{\partial L(z,y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Figure 7: Backpropagation Formula

As a result, the weight is updated as follows:

$$w \longleftarrow w - \eta \frac{\partial L(z,y)}{\partial w}$$

Figure 8: Updating weight using learning rate and gradient descent

Updating weights – In a neural network, weights are updated as follows:

- Step 1: Take a batch of training data.
- Step 2: Perform forward propagation to obtain the corresponding loss.
- Step 3: Back propagate the loss to get the gradients.
- Step 4: Use the gradients to update the weights of the network.

6.2 Experimental Results

Using the **Artificial Neural Network** classifier with K-fold cross (10 Folds) validation we got following results

Mean Accuracy	**Mean Standard Deviation**
0.9992375671863556	0.0002183970961974028

With confusion matrix we observed that there are

- 85284 True Positive
- 111 True Negative
- 22 False Positive
- 25 False Negative

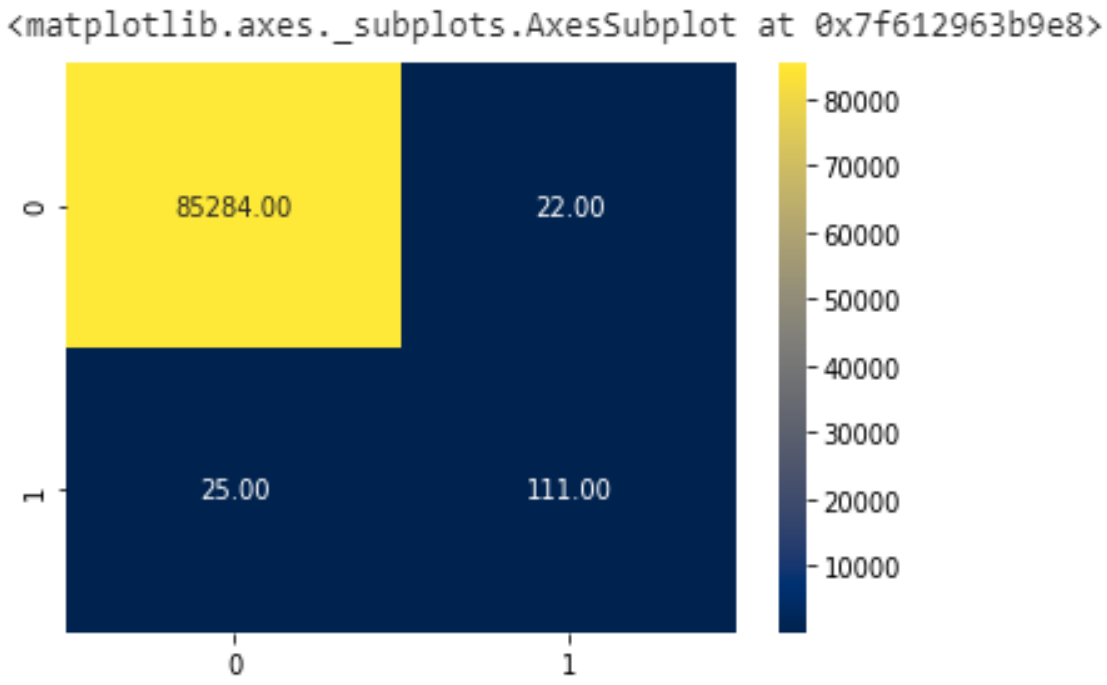

Figure 9: Confusion Matrix describing the results of ANN classifier

BIBLIOGRAPHY AND REFERENCES

Although, there are a ton of resources that helped us to complete this project, however it is not possible to mention them all here. So following are the resources that contributed significantly towards completion of this project

Video Lectures:

1. https://www.udemy.com/course/machinelearning/learn/lecture/6249412?start=1#content
2. https://www.coursera.org/learn/machine-learning-with-python/

Literature:

3. Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow by Aurelien Geron
4. https://stanford.edu/~shervine/teaching/cs-229/
5. http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf
6. https://blog.easysol.net/using-isolation-forests-anamoly-detection/
7. https://www.kaggle.com/mlg-ulb/creditcardfraud/
8. https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm

APPENDIX A

Code to implement ANN classifier

```
# -*- coding: utf-8 -*-
"""RandomForestClassifier.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1FjvD-IGuUM7VpChxfwJC4rmDk8XSh_my

# **Classifier using XGBoost to detect outliers (Fraud Transactions) in the credit Card Dataset**
"""

import numpy as np
import pandas as pd
from xgboost import XGBClassifier
import matplotlib.pyplot as plt

"""**Importing the dataset from gdrive**"""
dataset = pd.read_csv('/content/drive/My Drive/creditcard.csv')
"""**The dataset contains 30 columns out of which 27 are data related to credit card transactions. The 'amount' column is not scaled so must be scaled before XGBoost Descision tree classifier can be fitted to it. StandardScaler is used to do the job. 'Time' column have very little correlation with outcome. So 'Time' column is dropped. 'Amount' column is replaced by Scaled_Amount.**"""
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
scaled_amount = sc.fit_transform(dataset.iloc[:, 29].values.reshape(-1, 1))

dataset = dataset.drop(['Time','Amount'], axis = 1)

dataset.insert(28, "Scaled_Amount", scaled_amount, True)
"""**Data is splitted into training and testing data. Training data = 70% of original data, Test data = 30% of original data.**"""

from sklearn.model_selection import train_test_split
X = dataset.iloc[:, dataset.columns != 'Class'].values
y = dataset.iloc[:, 29]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 40)

"""**'classifier' is an object of XGBClassifier which uses Random Forest Classfier with Gradient Boosting. Object is fitted to training data. **"""

classifier = XGBClassifier()
classifier.fit(X_train, y_train)

"""**y_pred contains the predicted value of test data. Confusion Matrix is used to compare the predicted classification with actual values for test data.**"""

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)
print('Accuracy = '+ str((cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0])))

"""**The confusion matrix shows an accuracy of 0.9996254769317198 which is good . However to check whether the model is not overfitted, we have used cross_val_score to train and test model on 10 different partitions of training and test data.**"""

from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator= classifier, X = X_train, y = y_train, cv = 10)

"""**The average accuracy over 10 folds = 0.9994432288655493 and a standard deviation of 0.0002180114792249493.**"""

print(accuracies.mean())
print(accuracies.std())

"""**Seaborn heatmap to plot confusion matrix**"""

import seaborn as sns
sns.heatmap(cm, annot= True, fmt = '.2f', cmap = 'cividis')
```

APPENDIX B

Code to implement ANN classifier

```
# -*- coding: utf-8 -*-
"""ANN_Classifier.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ygEtLVWd-RBuH0QjXhV81el4ekO1vyln

# **Classifier using ANN Classifier to detect outliers(Fraud Transactions) in the credit Card Dataset**
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

"""**Importing the dataset from gdrive**"""

dataset = pd.read_csv('/content/drive/My Drive/creditcard.csv')

"""**The dataset contains 30 columns out of which 27 are data related to credit card transactions. The 'amount' column is not scaled so must be scaled before XGBoost Descision tree classifier can be fitted to it. StandardScaler is used to do the job. 'Time' column have very little correlation with outcome. So 'Time' column is dropped. 'Amount' column is replaced by Scaled_Amount.**"""
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
scaled_amount = sc.fit_transform(dataset.iloc[:, 29].values.reshape(-1, 1))

dataset = dataset.drop(['Time','Amount'], axis = 1)

dataset.insert(28, "Scaled_Amount", scaled_amount, True)

"""**Data is splitted into training and testing data. Training data = 70% of original data, Test data = 30% of original data.**"""

from sklearn.model_selection import train_test_split
X = dataset.iloc[:, dataset.columns != 'Class'].values
y = dataset.iloc[:, 29]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 40)

"""**'classifier' is an object of ANN Model which uses Random Keras(TensorFlow Backend) with Sochastic Gradient Boosting. Object is fitted to training data. **"""

import keras
from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()
classifier.add(Dense(units = 15, kernel_initializer = 'uniform', activation = 'relu', input_dim = 29))

# Adding the second hidden layer
```

```
classifier.add(Dense(units = 15, kernel_initializer = 'uniform', activation = 'relu'))

# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 100, epochs = 100)

"""**y_pred contains the predicted value of test data. Confusion Matrix is used to compare the predicted classification with actual values for test data.**"""

y_pred = classifier.predict(X_test)

y_pred = (y_pred>0.5)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print(cm)
print('Accuracy = '+ str((cm[0][0]+cm[1][1])/(cm[0][0]+cm[1][1]+cm[0][1]+cm[1][0])))

"""**The confusion matrix shows an accuracy of 0.9994499192434634 which is good . However to check whether the model is not overfitted, we have used cross_val_score to train and test model on 10 different partitions of training and test data.**"""
```

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from keras.models import Sequential
from keras.layers import Dense

def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 15, kernel_initializer = 'uniform', activation = 'relu', input_dim = 29))
    classifier.add(Dense(units = 15, kernel_initializer = 'uniform', activation = 'relu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
    return classifier

classifier = KerasClassifier(build_fn = build_classifier, batch_size = 100, epochs = 100)
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10, n_jobs = -1)
mean = accuracies.mean()
variance = accuracies.std()

"""**The average accuracy over 10 folds = 0.9992375671863556 and a standard deviation of 0.0002183970961974028.**"""

print(mean)
print(variance)

import seaborn as sns
sns.heatmap(cm, annot= True, fmt = '.2f', cmap = 'cividis'
```