

Credit Card Fraud Detection

The above mini project is submitted in partial fulfilment of the requirements
Of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

By

Harsh Harwani

Chinmay Jadhav

Jenil Jain

Guide:

Mr. Vaibhav Ambhire

(Assistant Professor, Department of Computer Engineering, TSEC)



**Computer Engineering Department
Thadomal Shahani Engineering College
University of Mumbai**

2019-20

CERTIFICATE

This is to certify that the mini project titled **“Credit Card Fraud Detection”** is
a bonafide work of

Roll Number	Semester	Name
51	VI	Harsh Harwani
53	VI	Chinmay Jadhav
55	VI	Jenil Jain

(Name and sign)

Guide

Internal Examiner

External Examiner

Table of Contents

List of Figures		iii
Chapter 1	Introduction	1
	1.1 General Introduction	
	1.2 Problem Definition	
	1.3 Domain	
	1.4 Need	
	1.5 Scope	
	1.6 Application	
Chapter 2	Design	5
	2.1 Literature Survey	
	2.2 Technologies Used	
	2.3 Requirements	
Chapter 3	Implementation	9
	3.1 Algorithm Used	
	3.2 Output	
	3.3 Result	
Chapter 4	Conclusion and Future Scope	19

List of Figures

Fig No.	Description	Page Nos.
1.1	Increasing credit Card frauds over the years	2
1.2	Graphic depiction of Data analysis using SOM and ANN	3
2.1	Self-Organizing Maps	6
3.1	Dataset	10
3.2	SOM Heatmap	12
3.3	Detected Frauds	13
3.4	Output after SOM And ANN	15
3.5	Output 1	16
3.6	Output 2	16
3.7	Output 3	17
3.8	Output 4	17

Chapter 1: Introduction

1.1 General Introduction:

Today use of Credit Card even in developing countries has become a common scenario. People use it to shop, pay bills and for online transactions. But with increase in number of Credit Card users, the cases of fraud in Credit Card have also been on rise. Credit Card related frauds cause globally a loss of billions of dollars. Fraud can be classified as any activity with the intent of deception to obtain financial gain by any manner without the knowledge of the cardholder and the issuer bank. Credit Card fraud can be done in numerous ways. By lost or stolen cards, by producing fake or counterfeit cards, by cloning the original site, by erasing or modifying the magnetic strip present at the card which contains the user's information, by phishing, by skimming or by stealing data from a merchant's side.

1.2 Problem Definition:

The Credit Card Fraud Detection Problem includes modeling 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.

The credit card fraud detection features use user behaviour and location scanning to check for unusual patterns. These patterns include user characteristics such as user spending patterns as well as usual user geographic locations to verify his identity. If any unusual pattern is detected, the system requires revivification. The system analyses user credit card data for various characteristics. These characteristics include user country, usual spending procedures. Based upon previous data of that user the system recognizes unusual patterns in the payment

procedure. So now the system may require the user to login again or even block the user for more than 3 invalid attempts.

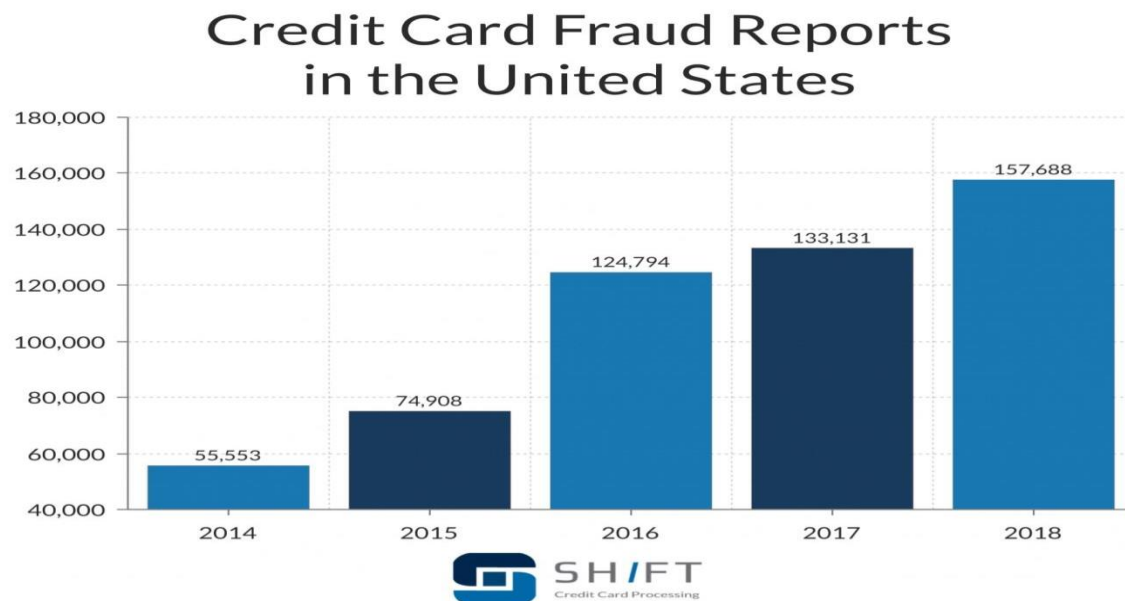


Fig 1.1 Increasing credit card frauds over the years

1.3 Domain:

- Deep Learning:

Deep learning is this generation's solution which replaces such methodologies and can work on large datasets which is not easily possible for human beings. Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. Deep learning algorithms are employed to analyse all the authorized transactions and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent.

- Web-Development:

Web development includes many types of web content creation. A web developer may be involved in designing a website, but may also write web scripts in languages such as PHP and frameworks such as Django and Flask. Additionally, a web developer may help maintain and update a database used by a dynamic website.

1.4 Need:

Fraud has been increasing drastically with the progression of state-of-art technology and worldwide communication. With continued advancement in fraudulent strategies it is important to develop effective models to combat these frauds in their initial stage only, before they can take to completion. This is a very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated.

1.5 Scope:

In this project we have used a hybrid learning approach for fraud detection. In the first step we have used the Self Organizing Maps (SOM) mechanism and the data points which point toward fraudulent transactions are fetched to an Artificial Neural Networks (ANN) model. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result.

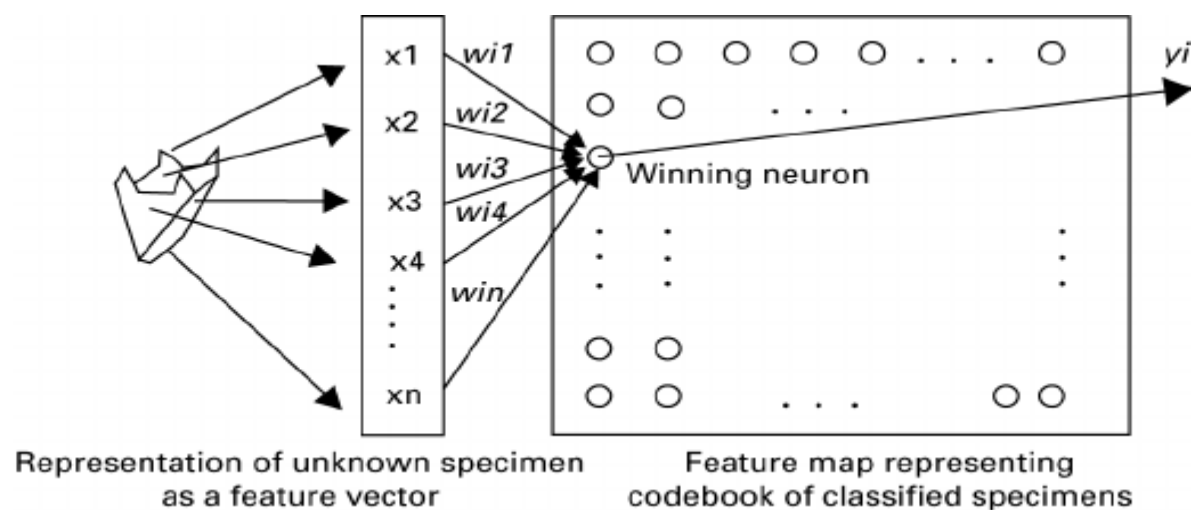


Fig 1.2 Graphic depiction of data analysis using an ANN and a SOM

1.6 Application:

Credit card fraud detection has been a keen area of research for the researchers for years and will be an intriguing area of research in the coming future. The main goal of this project is to help merchants, financial services consultancies and payment service providers distinguish fraudsters from customers. It also gathers information and analyses highly-detailed behavioural patterns such as browsing patterns, keyboard preferences and screen tilt.

Chapter 2: Design

2.1 Literature Survey:

In earlier studies, many approaches have been proposed to bring solutions to detect fraud from supervised approaches, unsupervised approaches to hybrid ones; which makes it a must to learn the technologies associated in credit card frauds detection and to have a clear understanding of the types of credit card fraud. As time progressed fraud patterns evolved introducing new forms of fraud making it a keen area of interest for researchers. During the project we went through many research papers of which we primarily focused on ResearchGate published conference paper “Real-time Credit Card Fraud Detection Using Machine Learning “. We also went through popular computer vision websites such as www.towardsdatascience.com, www.rubiksgcode.net and read documents related to Self-Organizing Maps (SOM) and watched various videos so as to understand the approaches to solve the problem. The main aim in referring the websites was to understand the techniques and select an appropriate and viable method for the project.

While searching for a dataset we went through many websites such as www.archive.ics.uci.edu and other dataset websites. We choose Credit Approval Dataset due to its contents i.e. it contains about 15 features divided into 3 classes (integer, real and categorical) which would surely help in training the model to achieve a high accuracy. This dataset was very interesting because there was a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There were also a few missing values. For building the neural network used keras with TensorFlow as a backend to build the network.

The network is a 3-layer network with 11 input nodes 3 nodes in middle layer and one node in Output layer. Visualization of the results is done using Matplotlib.

2.2 Technologies Used:

2.2.1 Self-Organizing Feature Maps (SOM):

SOM was introduced by Finnish professor Teuvo Kohonen in the 1980s is sometimes called a **Kohonen map**. Each data point in the data set recognizes themselves by competing for representation. SOM mapping steps starts from initializing the weight vectors. From there a sample vector is selected randomly and the map of weight vectors is searched to find which weight best represents that sample. Each weight vector has neighbouring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. The neighbours of that weight are also rewarded by being able to become more like the chosen sample vector. This allows the map to grow and form different shapes. Most generally, they form square/rectangular/hexagonal/L shapes in 2D feature space.

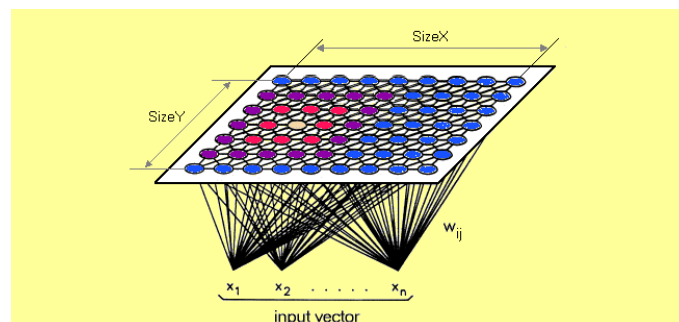


Fig 2.1 Self-Organizing Feature Maps

2.2.2 Artificial Neural Network (ANN):

It combines the thinking power of human brain with computational power of machine. It makes use of neurons as the deciding sites and the edges between neurons to calculate the contribution of each neuron in the previous layer in the decision and result at the current neuron. It is based on pattern recognition. Previous year's data is fed into the network and then based upon that

data it recognises a new incoming transaction to be a fraud or genuine one. Its training can either be supervised i.e. the outcome is already known for a given transaction and the expected output is compared with actual to train the system or it can be unsupervised where we have no actual results to compare it with and thus are not sure about the results.

2.2.3 Django:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Django also helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically.

2.3 Requirements:

2.3.1 Functional Requirements:

- The system should train the parameters based on the testing dataset provided.
- System shall recognize the type of transaction (legitimate or fraud) from the testing dataset and place them in the appropriate class.
- The system shall display the predicted result and the actual result for the unknown sample provided.

2.2.2 Non-Functional Requirements:

- Performance: The designed system will recognize and classify the transaction in its appropriate class with an accuracy close to 94%.
- Reliability: The system should work for all type of values in the dataset.
- Functionality: The system will deliver the functional requirements mentioned in the documents.

2.3.3 Software Requirements:

1. Python 3.x
2. Sklearn
3. Matplotlib
4. Pandas
5. NumPy
6. PyCharm

2.3.4 Hardware Requirements:

1. Intel core i3 +
2. GPU like Nvidia GeForce
3. 4 GB + RAM

Chapter 3: Implementation

3.1 Data Pre-processing:

3.1.1 Data Preparation:

We will use the UCI Machine Learning Repository for credit card fraud dataset. After extensive research, we have selected the Credit approval dataset.

Diagnostic: <http://archive.ics.uci.edu/ml/datasets/Credit+Approval>

This data concerns credit card applications; good mix of attributes.

Dataset Information:

This file concerns credit card applications. This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

Attribute Information:

ATTRIBUTE NAMES HAVE BEEN MASKED FOR DATA CONFIDENTIALITY

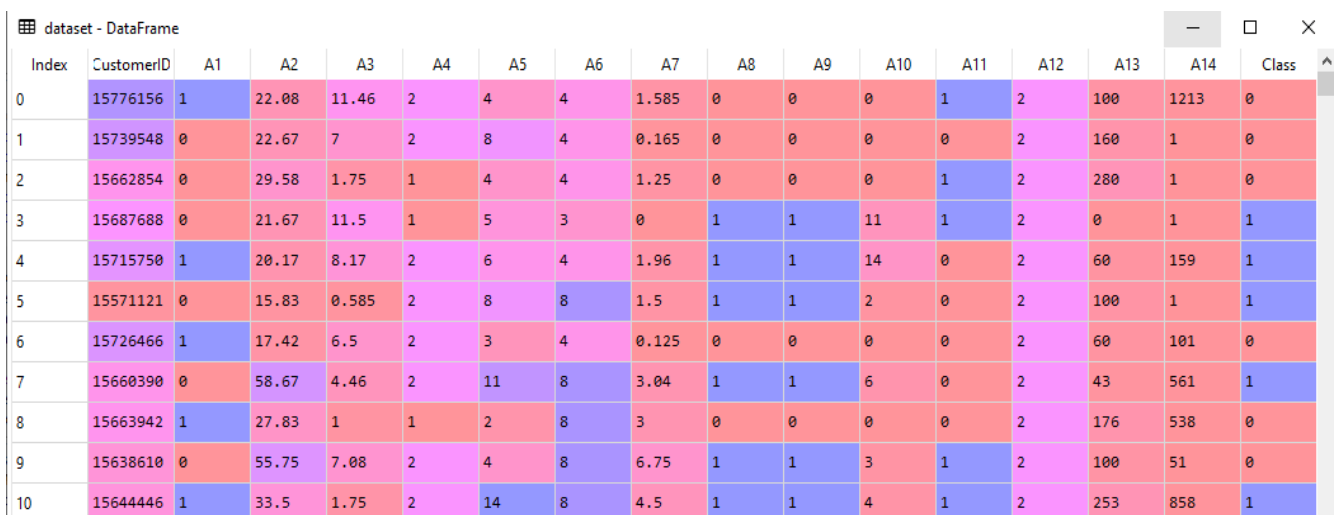
A1	b, a
A2	continuous
A3	continuous
A4	u,y,l,t
A5	g,p,gg
A6	c,d,cc,i,j,k,m,r,q,w,x,e,aa,ff
A7	v,h,bb,j,n,z,dd,ff,o
A8	continuous
A9	t,f
A10	t,f

A11	continuous
A12	t,f
A13	g,p,s
A14	continuous
A15	continuous
A16	+, - (class attribute)

3.1.2 Data Exploration:

We will be using *Spyder* to work on this dataset. We will first go with importing the necessary libraries and our dataset to PyCharm:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Credit_Card_Applications.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
print(dataset.head())
```



Index	CustomerID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
0	15776156	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	0
1	15739548	0	22.67	7	2	8	4	0.165	0	0	0	0	2	160	1	0
2	15662854	0	29.58	1.75	1	4	4	1.25	0	0	0	1	2	280	1	0
3	15687688	0	21.67	11.5	1	5	3	0	1	1	11	1	2	0	1	1
4	15715750	1	20.17	8.17	2	6	4	1.96	1	1	14	0	2	60	159	1
5	15571121	0	15.83	0.585	2	8	8	1.5	1	1	2	0	2	100	1	1
6	15726466	1	17.42	6.5	2	3	4	0.125	0	0	0	0	2	60	101	0
7	15660390	0	58.67	4.46	2	11	8	3.04	1	1	6	0	2	43	561	1
8	15663942	1	27.83	1	1	2	8	3	0	0	0	0	2	176	538	0
9	15638610	0	55.75	7.08	2	4	8	6.75	1	1	3	1	2	100	51	0
10	15644446	1	33.5	1.75	2	14	8	4.5	1	1	4	1	2	253	858	1

Fig 3.1 Dataset

3.1.3 Feature Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you're transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

For this, we use `MinMaxScaler`, to bring all values between 0 and 1

```
from sklearn.preprocessing import MinMaxScaler
```

```
sc = MinMaxScaler(feature_range = (0, 1))
```

```
X = sc.fit_transform(X)
```

3.1.4 Model Selection

We have used a Hybrid Deep Learning approach, in which we first used Unsupervised learning to find potential frauds using Self Organizing Maps and then we use the Supervised Learning approach using Artificial Neural Network with O/p from SOM as target variable of the network. Thus, improving the accuracy.

1. Self-Organizing Maps:

We have chosen a 10x10 grid to visualize the results. We have used `MiniSom` which a library used to build an SOM.

```
from minisom import MiniSom
```

```
som = MiniSom(x = 10, y = 10, input_len = 15, sigma = 1.0, learning_rate = 0.5)
```

```
som.random_weights_init(X)
```

```
som.train_random(data = X, num_iteration = 100)
```

Visualization:

```
from pylab import bone, pcolor, colorbar, plot, show
```

```
bone()
```

```
pcolor(som.distance_map().T)
```

```
colorbar()
```

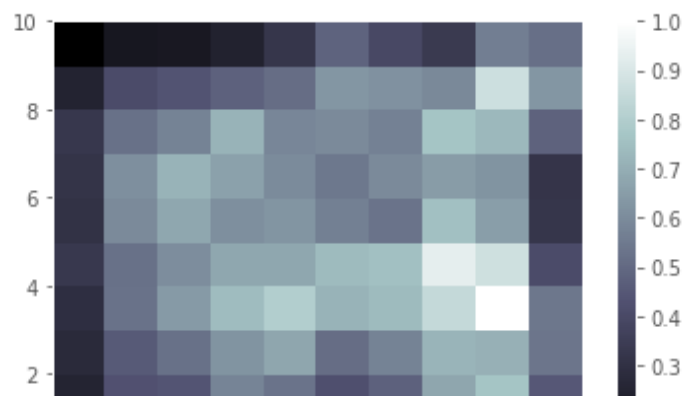


Fig 3.2 SOM Heatmap

The outlier blocks with co-ordinate (8,3) and (7,4) and s represents a node with customers who do not follow the general rules. We can find the frauds as follows:

```
# Finding the frauds
```

```
mappings = som.win_map(X)
```

```
frauds = np.concatenate((mappings[(8,3)], mappings[(7,4)]), axis = 0)
```

```
frauds = sc.inverse_transform(frauds)
```

```
print(frauds)
```


frauds - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	15711742	0	23	1.835	2	5	3	0	0	1	1	0	2	200	54
1	15770995	0	40.33	8.125	1	4	4	0.165	0	1	2	0	2	184	19
2	15776545	0	25	11	1	6	4	4.5	1	0	0	0	2	120	1
3	15651868	0	38.75	1.5	2	1	1	0	0	0	0	0	2	76	1
4	15672637	0	36.75	4.71	2	1	1	0	0	0	0	0	2	160	1
5	15769548	0	32	6	2	2	4	1.25	0	0	0	0	2	272	1
6	15736533	0	31.57	11.25	2	1	1	0	0	0	0	0	2	184	5201
7	15779207	0	27.33	1.665	2	1	1	0	0	0	0	0	2	340	2
8	15805261	0	48.17	1.335	2	3	7	0.335	0	0	0	0	2	0	121

Fig 3.3 Detected frauds

2. Artificial Neural Network(ANN):

Now, we create a Network with three layers.

First layer with 15 input nodes and apply **Relu** (Rectified Linear Unit) activation function to the nodes. The second layer has 2 nodes with Relu as activation function and third layer with one node tells the probability of the customer being fraud. Sigmoid activation function is applied which is best for output layer.

The model is build using Keras with Tensorflow as a backend.

Implementation:

```
from keras.models import Sequential
```

```
from keras.layers import Dense
```

```
# Initialising the ANN
```

```
classifier = Sequential()
```

```

# Adding the input layer and the first hidden layer

classifier.add(Dense(units = 2, kernel_initializer = 'uniform', activation = 'relu',
input_dim = 15))

# 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(customers, is_fraud, batch_size = 1, epochs = 5)

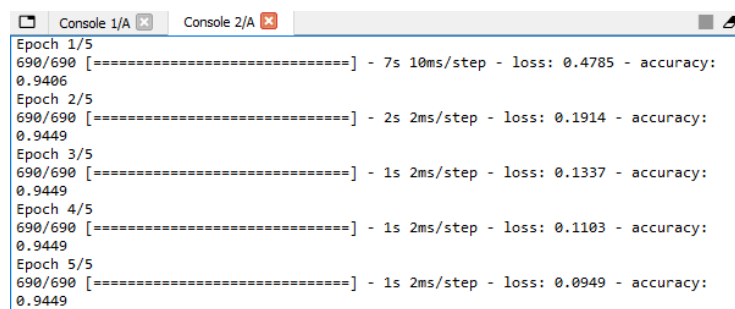
```

The Algorithm:

1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data.
3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the **Best Matching Unit** (BMU).
4. Then the neighbourhood of the BMU is calculated. The amount of neighbours decreases over time.
5. The winning weight is rewarded with becoming more like the sample vector. The neighbours also become more like the sample vector. The closer a node is to the BMU, the more its weights get altered and the farther away the neighbour is from the BMU, the less it learns.
6. Repeat step 2 for N iterations.

Best Matching Unit is a technique which calculates the distance from each weight to the sample vector, by running through all weight vectors. The weight with the shortest distance is the winner. There are numerous ways to determine the distance, however, the most commonly used method is the **Euclidean Distance**, and that's what is used in the following implementation.

3.2 Output:



```
Epoch 1/5
690/690 [=====] - 7s 10ms/step - loss: 0.4785 - accuracy:
0.9406
Epoch 2/5
690/690 [=====] - 2s 2ms/step - loss: 0.1914 - accuracy:
0.9449
Epoch 3/5
690/690 [=====] - 1s 2ms/step - loss: 0.1337 - accuracy:
0.9449
Epoch 4/5
690/690 [=====] - 1s 2ms/step - loss: 0.1103 - accuracy:
0.9449
Epoch 5/5
690/690 [=====] - 1s 2ms/step - loss: 0.0949 - accuracy:
0.9449
```

Fig 3.4 Output after SOM and ANN

Accuracy: After training the network for 5 epochs, we get an accuracy of **94.49%**.

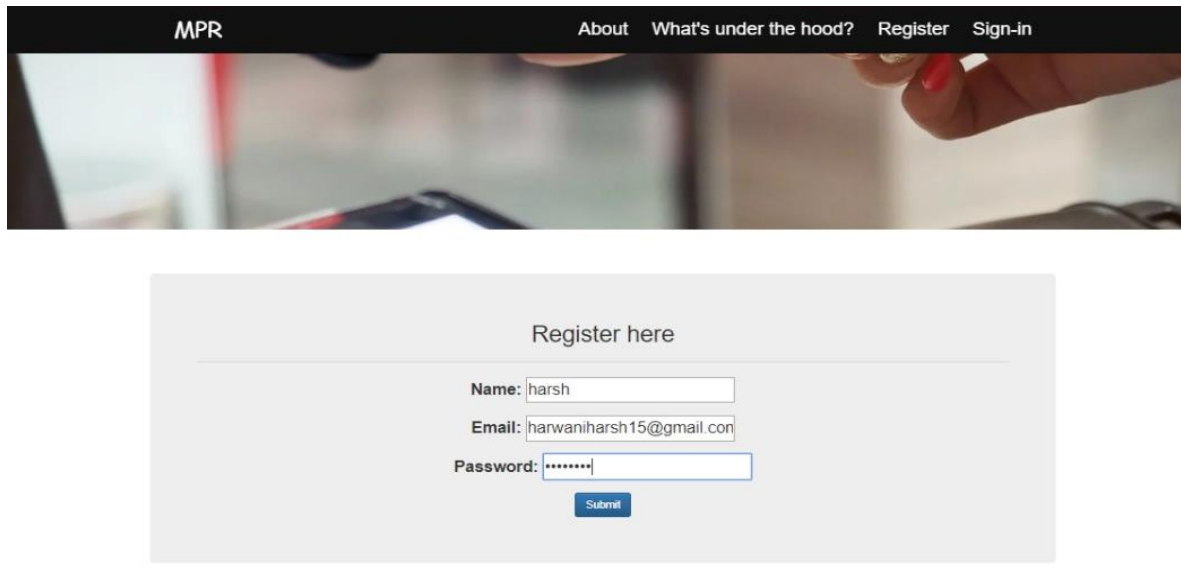


Fig 3.5 Output 1

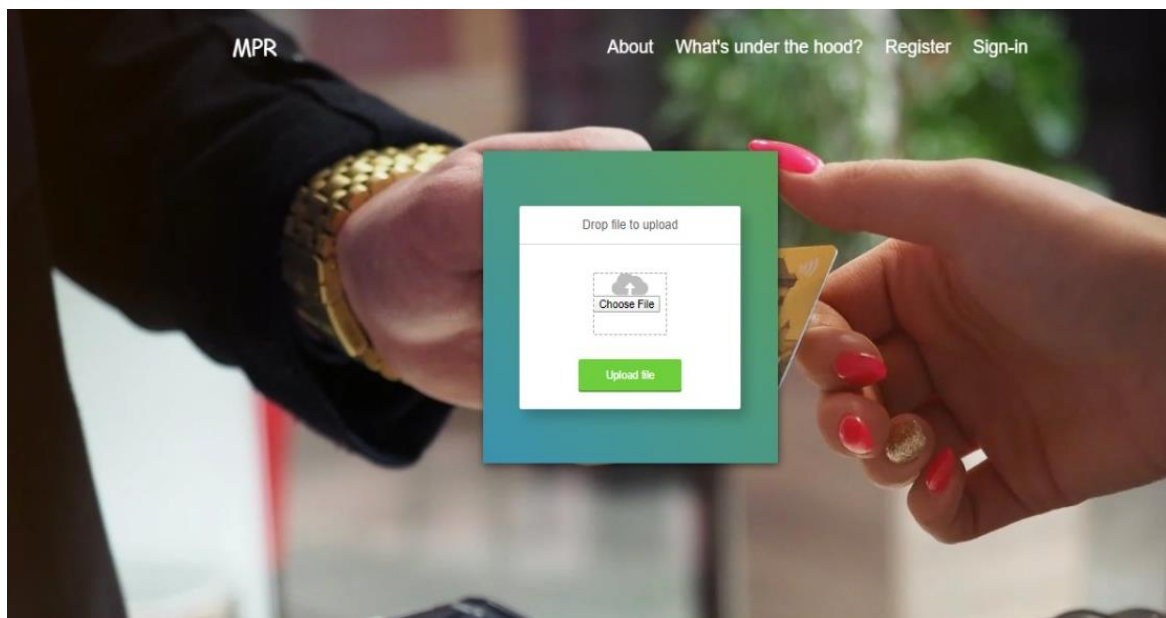


Fig 3.6 Output 2

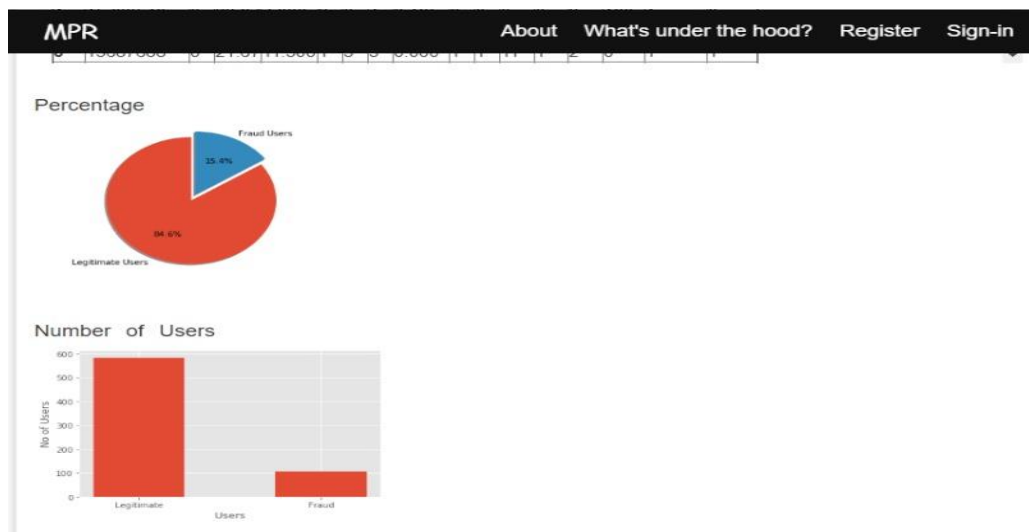


Fig 3.7 Output 3

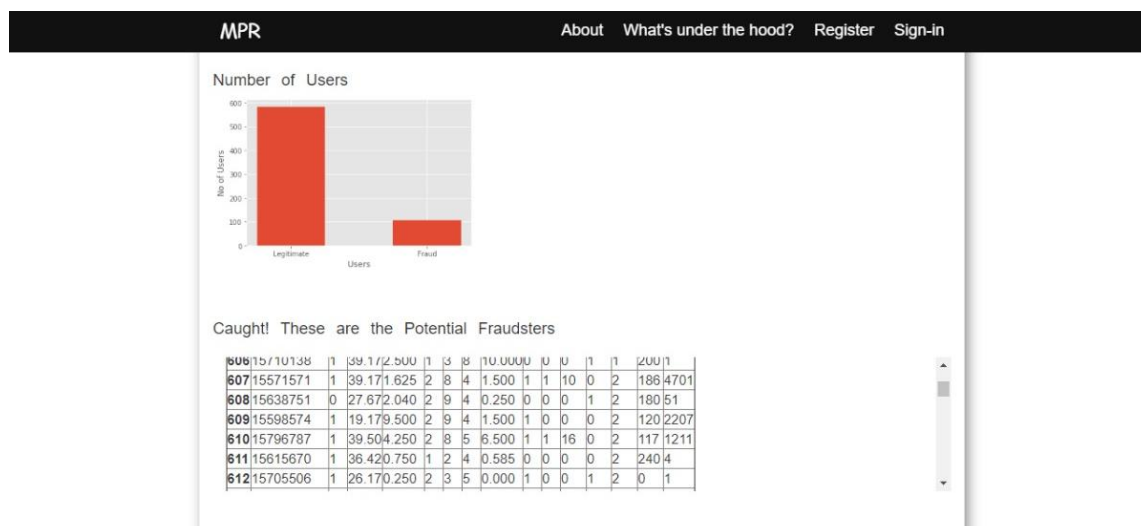


Fig 3.8 Output 4

3.3 Result:

Our model achieved a accuracy of 94.49 % for fraud detection and the affected population (which were considered fraud but were not fraud was found to be 22.60 %. The results may vary if you run the same jupyter notebook because initialization of the weights of the nodes of SOM grid is done by randomly selecting the records/ patterns from the input space i.e. randomly selecting the records from the given dataset. Since, we have done training for 100 iterations and weights are randomly initialized every time, convergence may vary. We may try with different iterations like 100, 150, 200 etc. to have better convergence. You may also store the weights of the SOM for which you achieve better accuracy.

Chapter 4: Conclusion and Future Scope

4.1 Conclusion:

From the above comparative analysis of the various credit card fraud detection techniques it is clear that Artificial Neural Networks performs best in this scenario. But the drawbacks of Artificial Neural Networks is that they are very expensive to train and can be easily over trained. In order to minimize their expense we need to create a hybrid of neural network with some optimisation technique. Optimisation techniques that could be successfully paired with Neural Network are Genetic Algorithm, Artificial Immune System, Case Based Reasoning and any other similar optimisation technique.

4.2 Future Scope:

This model can be trained further so as to increase the accuracy of the system. It can be used for other datasets as well, with minor changes in the data pre-processing and wrangling.

We can further increase the accuracy and performance of data-sets prediction either by increasing or decreasing data feature or features selection or applying feature engineering in our machine learning model.

References

[1] Dataset:

<http://archive.ics.uci.edu/ml/datasets/Credit+Approval>

[2] IEEE Technical Paper:

https://www.researchgate.net/publication/334761474_Real-time_Credit_Card_Fraud_Detection_Using_Machine_Learning

[3]Book:

Linda Delamaire (UK), Hussein Abdou (UK), John Pointon (UK),” Credit card fraud and detection techniques: areview”, Banks and Bank Systems, Volume 4, Issue 2, 2009

[4]Website:

<https://towardsdatascience.com/credit-card-fraud-detection-using-self-organizing-featuremaps-f6e8bca707bd>

[5]Website:

<https://rubikscore.net/2018/09/24/credit-card-fraud-detection-using-self-organizing-maps-and-python/>

Acknowledgement

We would like to express special thanks of gratitude to our teacher Mr Vaibhav Ambhire of Thadomal Shahani Engineering College, Computer Department who gave us the golden opportunity to do this wonderful project on “CREDIT CARD FRAUD DETECTION” and for helping us with providing invaluable guidance along with elating encouragement throughout the course of our project development and submission