# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Credit\_Card\_Applications.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature\_range = (0, 1))

X = sc.fit\_transform(X)

from minisom import MiniSom

som = MiniSom(x = 13, y = 13, input\_len = 15, sigma = 1.0, learning\_rate = 0.5, random\_seed=42)

# randomly initialize the weight vectors to small numbers close to 0

som.random\_weights\_init(X)

# train som on X, matrix of features and patterns recognized

som.train\_random(data = X, num\_iteration = 100)

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

bone()

pcolor(som.distance\_map().T)

colorbar()

markers = ['o', 's']

colors = ['r', 'g']

for i, x in enumerate(X):

w = som.winner(x)

plot(w[0] + 0.5,

w[1] + 0.5,

markers[y[i]],

markeredgecolor = colors[y[i]],

markerfacecolor = 'None',

markersize = 10,

markeredgewidth = 2)

show()

# Finding the frauds

mappings = som.win\_map(X)

frauds = np.concatenate((mappings[(5,10)], mappings[(6,4)]), axis = 0)

frauds = sc.inverse\_transform(frauds)

# Creating the matrix of features

customers = dataset.iloc[:, 1:].values

# Creating the dependent variable

is\_fraud = np.zeros(len(dataset))

for i in range(len(dataset)):

if dataset.iloc[i,0] in frauds:

is\_fraud[i] = 1

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

customers = sc.fit\_transform(customers)

# Creating the dependent variable

is\_fraud = np.zeros(len(dataset))

for i in range(len(dataset)):

if dataset.iloc[i,0] in frauds:

is\_fraud[i] = 1

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

customers = sc.fit\_transform(customers)

# Part 2 - Now let's make the ANN!

# Importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Dense

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

customers\_train, customers\_test, is\_fraud\_train, is\_fraud\_test = train\_test\_split(customers, is\_fraud, test\_size =.30,random\_state=10)

# Initialising the ANN

classifier = Sequential()

# Adding the input layer and the first hidden layer

classifier.add(Dense(units = 8, 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\_train, is\_fraud\_train, batch\_size = 2, epochs = 2)

# Predicting the probabilities of frauds

y\_pred = classifier.predict(customers\_test)

y\_pred = (y\_pred > 0.5)

from sklearn.metrics import classification\_report, accuracy\_score,confusion\_matrix

cm = confusion\_matrix(is\_fraud\_test, y\_pred)

#print accuracy

print(accuracy\_score(is\_fraud\_test,y\_pred))

from sklearn import feature\_selection

from sklearn import model\_selection

from sklearn import metrics

#logistic

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 42)

# Fit the regressor object into the training set

classifier.fit(customers\_train, is\_fraud\_train)

# regressor is the machine that learns the corelation of the training set to make some future predictions

y\_pred = classifier.predict(customers\_test)

print(accuracy\_score(is\_fraud\_test,y\_pred))

#RandomForest

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(random\_state = 42)

# Fit the regressor object into the training set

classifier.fit(customers\_train, is\_fraud\_train)

# regressor is the machine that learns the corelation of the training set to make some future predictions

y\_pred = classifier.predict(customers\_test)

print(accuracy\_score(is\_fraud\_test,y\_pred))

#KNeighbor

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier()

# Fit the regressor object into the training set

classifier.fit(customers\_train, is\_fraud\_train)

# regressor is the machine that learns the corelation of the training set to make some future predictions

y\_pred = classifier.predict(customers\_test)

print(accuracy\_score(is\_fraud\_test,y\_pred))