*Al-alamein international university*

*ML course*

**Credit card fraud detection**



***Introduction:***

***Procedures have taken to implement the project:***

1)Define Your Problem: Clearly understand what you want to achieve with your machine learning model.

2)Gather Data: Collect relevant data for your problem. Ensure the data is clean, consistent, and properly formatted.

3)Preprocess the Data: Prepare your data for modeling by handling missing values, encoding categorical variables, scaling numerical features, and splitting it into training and testing sets.

4)Choose a Model: Select the appropriate machine learning algorithm for your problem. Consider factors like the size of your dataset, the nature of your data, and the desired outcome

5)Train the Model: Fit your chosen model to the training data.

6) Compare : compare the train model and test model

To ensure that is neither overfiting or underfiting in the model.

7)Evaluate the Model: Assess the performance of your model using

evaluation metrics,metrics like mean squared error or

R- squared are common.

8)Deploy the Model: Once you're satisfied with your model's

performance, deploy it into production.

#Importing the Dependencies

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import f1\_score ,accuracy\_score, confusion\_matrix, classification\_report

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

import math

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

from sklearn.neighbors import KNeighborsClassifier

We must import libraries to to utilize the functionality and 1 features in python like (numpy, pandas,math) to pre- implemented functions and modules that can be reused across the project.

# loading the dataset to a Pandas DataFrame

credit\_card\_data = pd.read\_csv('creditcard.csv'

# first 10 rows of the dataset

credit\_card\_data.head(10)

# In[294]:

credit\_card\_data.tail(10)

# In[295]:

credit\_card\_data.drop(["Time"],axis=1,inplace=True)

we must import the dataset with all features and select what will be the suitable feature and the output feature.

# checking the number of missing values in each column

credit\_card\_data.isnull().sum()

# In[300]:

credit\_card\_data.drop\_duplicates(inplace=True)

we must check that the dataset is clean to start, so we ensure that our dataset doesn’t have empty or duplicate cells.

# distribution of legit transactions & fraudulent transactions

credit\_card\_data['Class'].value\_counts()

#This Dataset is highly unblanced

#0 --> Normal Transaction

#1 --> fraudulent transaction

# separating the data for analysis

legit = credit\_card\_data[credit\_card\_data.Class == 0]

fraud = credit\_card\_data[credit\_card\_data.Class == 1]

#legit.drop(["Time"],axis=1,inplace=True)

# statistical measures of the data

legit.Amount.describe()

# In[305]:

fraud.Amount.describe()

#Under-Sampling

#Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

#Number of Fraudulent Transactions --> 473

legit\_sample = legit.sample(n=473)

#Concatenating two DataFrames

new\_dataset = pd.concat([legit\_sample, fraud], axis=0)

#Splitting the data into Features & Targets

X = new\_dataset.drop(columns='Class', axis=1)

Y = new\_dataset['Class']

# Split the data into Training data & Testing Data

# In[318]:

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, stratify=Y, random\_state=2)

Logistic regression

#Model Training

#training the Logistics regression for binary classificationLogistics

model = LogisticRegression()

model.fit(X\_train, Y\_train)

# accuracy on training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train)

# accuracy on test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score( X\_test\_prediction, Y\_test)

print('Accuracy on Training data : ', round(training\_data\_accuracy\*100,2))

print('Accuracy score on Test Data : ', round(test\_data\_accuracy\*100,2))

Decision tree

#max\_depth = int(math.ceil(math.sqrt(X\_train.size)))

dtree = DecisionTreeClassifier(max\_depth=5)

dtree.fit(X\_train, Y\_train)

# In[359]:

# accuracy on training data

X\_train\_predictiontree = dtree.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_predictiontree, Y\_train)

# accuracy on test data

X\_test\_predictiontree = dtree.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_predictiontree, Y\_test)

print('Accuracy on Training data : ', round(training\_data\_accuracy\*100,2))

print('Accuracy score on Test Data : ', round(test\_data\_accuracy\*100,2))

# In[331]:

print("DecisionTree Performance:")

print("Accuracy:", round(accuracy\_score(X\_test\_predictiontree,Y\_test )\*100,2))

print("Confusion Matrix:\n", confusion\_matrix(X\_test\_predictiontree,Y\_test))

print("Classification Report:\n", classification\_report( X\_test\_predictiontree,Y\_test))

Linear regression

regressor=LinearRegression()

regressor.fit(X\_train, Y\_train)

# In[338]:

# accuracy on training data

X\_train\_predictiontr = regressor.predict(X\_train)

X\_test\_predictiontr = regressor.predict(X\_test)

print(metrics.mean\_absolute\_error(Y\_test,X\_test\_predictiontr))

r2\_score1=regressor.score(X\_train,Y\_train)

r2\_score = regressor.score(X\_test,Y\_test)

# In[364]:

# accuracy on training data

X\_train\_predictiontr = regressor.predict(X\_train)

# accuracy on test data

X\_test\_predictiontr = regressor.predict(X\_test)

r2\_score = regressor.score(X\_test,Y\_test)

print ("error of linear reggresion")

print(metrics.mean\_absolute\_error(Y\_test,X\_test\_predictiontr))

print("LinearRegression Performance:")

print("test accuary:")

print(r2\_score\*100,'%')

print("train accuary")

r2=regressor.score(X\_train,Y\_train)

x= print(r2\*100,'%')