**CREDIT CARD FRAUD DETECTION**

**PHASE III PROJECT: DEVELOPMENT PART 2**

**TABLE OF CONTENTS:**

1. **INTRODUCTION**
2. **PROBLEM DEFINITION**
3. **FEATURE ENGINEERING**
4. **MODEL TRAINING**
5. **EVALUTION**
6. **CONCLUSION**
7. **REFERENCES**

**INTRODUCTION:**

The problem is to develop a machine learning-based system for real-time credit card fraud detection. The goal is to create a solution that can accurately identify fraudulent transactions while minimizing false positives. This project involves data preprocessing, feature engineering, model selection, training, and evaluation to create a robust fraud detection system.

**PROBLEM DEFINITION:**

The mission of our project is to prevent real-time credit card fraud and take measures for it to stop.

**FEATURE ENGINEERING:**

**Scale the numerical features:**

To scale the numerical features, such as Time and Amount, you can use the StandardScaler from Scikit-learn. This will transform the features to have zero mean and unit variance.

from sklearn.preprocessing import StandardScaler

scaled\_data = dataset.copy()

scaler = StandardScaler()

scaled\_data["Time"]=scaler.fit\_transform(scaled\_data["Time"].values.reshape(-1,1))

scaled\_data["Amount"]=scaler.fit\_transform(scaled\_data["Amount"].values.reshape(-1, 1))

**Reduce the dimensionality of the data:**

To reduce the dimensionality of the data, you can use Principal Component Analysis (PCA) from Scikit-learn. This will project the data onto a lower-dimensional subspace that preserves most of the variance.

from sklearn.decomposition import PCA

pca\_data = scaled\_data.copy()

X = pca\_data.drop("Class", axis=1)

y = pca\_data["Class"]

pca = PCA(n\_components=10)

X\_pca = pca.fit\_transform(X)

**Create new features from existing ones:**

To create new features from existing ones, you can use feature engineering techniques such as polynomial features, interaction terms, or domain knowledge.

from sklearn.preprocessing import PolynomialFeatures

poly\_data = pca\_data.copy()

poly = PolynomialFeatures(degree=2)

X\_poly = poly.fit\_transform(X\_pca)

**MODEL TRAINING:**

y = dataset["Class"] # target

X = dataset.iloc[:,0:30]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size = 0.2, random\_state = 42)

X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import GridSearchCV, cross\_val\_score, RandomizedSearchCV

kf = StratifiedKFold(n\_splits=5, random\_state = None, shuffle = False)

from imblearn.pipeline import make\_pipeline ## Create a Pipeline using the provided estimators .

from imblearn.under\_sampling import NearMiss ## perform Under-sampling based on NearMiss methods.

from imblearn.over\_sampling import SMOTE ## PerformOver-sampling class that uses SMOTE.

from sklearn.metrics import roc\_curve, roc\_auc\_score, accuracy\_score, recall\_score, precision\_score, f1\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

rfc.fit(X\_train, y\_train)

y\_pred = rfc.predict(X\_test)

data = pd.read\_csv('credit\_card.csv')

X = data.drop(columns=["V1"])

y = data["V2"]

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

print("The accuracy is", accuracy\_score(y\_test, y\_pred))

print("The precision is", precision\_score(y\_test, y\_pred))

print("The recall is", recall\_score(y\_test, y\_pred))

print("The F1 score is", f1\_score(y\_test, y\_pred))

**EVALUATION:**

from sklearn.metrics import precision\_score,

recall\_score, f1\_score, accuracy\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=20, test\_size=0.20)

tree = DecisionTreeClassifier()

tree.fit(X\_train, y\_train)

y\_pred = tree.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision:",precision\_score(y\_test,y\_pred, average="weighted"))

print('Recall:'recall\_score(y\_test,y\_pred average="weighted"))

print('F1 score:', f1\_score(y\_test, y\_pred, average="weighted"))

confusion\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

cm\_display=metrics.ConfusionMatrixDisplay( confusion\_matrix=confusion\_matrix, display\_labels=[0, 1, 2])

cm\_display.plot()

plt.show()

import numpy as np

from sklearn .metrics import roc\_auc\_score

y\_true = [1, 0, 0, 1]

y\_pred = [1, 0, 0.9, 0.2]

auc = np.round(roc\_auc\_score(y\_true, y\_pred), 3)

print("Auc", (auc))

# importing the libraries

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error,

mean\_squared\_error, mean\_absolute\_percentage\_error

df = pd.read\_csv('weather.csv')

X = df.iloc[:, 2].values

Y = df.iloc[:, 3].values

X\_train, X\_test,

Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.20, random\_state=0)

X\_train = X\_train.reshape(-1, 1)

X\_test = X\_test.reshape(-1, 1)

regression = LinearRegression()

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

Y\_pred = regression.predict(X\_test)

mae = mean\_absolute\_error(y\_true=Y\_test, y\_pred=Y\_pred)

print("Mean Absolute Error", mae)

mse = mean\_squared\_error(y\_true=Y\_test, y\_pred=Y\_pred)

print("Mean Square Error", mse)

rmse = mean\_squared\_error(y\_true=Y\_test, y\_pred=Y\_pred,

squared=False)

print("Root Mean Square Error", rmse)

mape = mean\_absolute\_percentage\_error(Y\_test, Y\_pred,

sample\_weight=None, multioutput='uniform\_average')

print("Mean Absolute Percentage Error", mape)

**CONCLUSION:**

Thus we have developed a model with feature engineering, trained a model and evaluated it.

**REFERNCES:**

[**Credit Card Fraud Detection in Python - Python Code (thepythoncode.com)**](https://thepythoncode.com/article/credit-card-fraud-detection-using-sklearn-in-python)

[**Detecting Credit Card Fraud with Autoencoders in Python | by Dimitris Panagopoulos | Towards Data Science**](https://towardsdatascience.com/detecting-credit-card-fraud-with-autoencoders-in-python-98391cace8a3)

[**Credit Card Fraud Detection with Python & Machine Learning - DataFlair (data-flair.training)**](https://data-flair.training/blogs/credit-card-fraud-detection-python-machine-learning/)

<https://thepythoncode.com/article/credit-card-fraud-detection-using-sklearn-in-python/>