Name: Kanishka Jingar	**Roll No.: 22/AI/023**
Branch : CSE(AI)	**Reg. no.: PCE22CA024**

ASSIGNMENT

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
III B.TECH. (VI Sem.) SEC- D
Code: 6CAI6-02
Subject Name–Machine Learning
(BRANCH: ADVANCE COMPUTING (AI))

Max. Time: 2 hrs. **Max. Marks: 20 Marks**

INSTRUCTIONS: UPLOAD THE SOLUTION ON YOUR GITHUB REPOSITORY and MENTIONED THE URL OF THE REPOSITORY ON TCSION

ASSIGNMENT QUESTION 1: CO3

Fault Prediction Using Supervised Machine Learning

Problem Context

You are an engineer working for a power distribution company responsible for maintaining and ensuring the reliability of the electrical grid. Your task is to develop a system for detecting and classifying electrical faults in the grid. Electrical faults can lead to disruptions, damage equipment, and pose safety hazards. The company is interested in a predictive maintenance system that can identify and classify different types of electrical faults to facilitate timely intervention.

Fault Prediction Dataset: https://www.kaggle.com/code/pythonafroz/fault-prediction-usingdecision-tree-algorithm

S/r No.	Question	Marks
Q1.	Name any 4 libraries required for the implementation of the problem statement using python	2
Ans1.	The given problem have used these libraries:- 1) import pandas as pd – Pandas enables robust data manipulation and analysis through easy-to-use data structures. 2) import numpy as np – Numpy provides efficient numerical computation with powerful array and matrix operations.	

	3) from sklearn.linear model import LogisticRegression – Scikit-learn Offers comprehensive tools for machine learning, including algorithms for prediction and analysis. 4) import matplotlib.pyplot as plt - Facilitates data visualization by creating diverse and customizable plots and charts.	
Q2.	Go through the above Kaggle link and answer the following: About this dataset file: https://www.kaggle.com/code/pythonafroz/fault-prediction-using-decisiontree-algorithm a. Total no. of columns in the dataset: 10 columns b. Write and count input columns: Inputs - [Ia,Ib,Ic,Va,Vb,Vc] – 6 columns c. Write and count the output column: class['Fault_Type'] with output [G,C,B,A] – 4 columns	3
Q3.	What is the purpose this library used for the given problem statement: `from sklearn.preprocessing import LabelEncoder`	1
Ans3.	It is used for encoding categorical variables into numerical values.	
Q4.	What is the purpose this library used for the given problem statement: `from sklearn.model_selection import train_test_split`	1
Ans4.	It is used to divide a dataset into distinct subsets for training and testing a model.	
Q4.	List all the algorithms through which you can able to find Electrical Faults Detection and Classification	3
Ans4.	1). **Logistic Regression:** predicts binary fault classifications based on a logistic function. 2). **Decision Trees:** Creates a tree of decisions based on feature values to classify faults**.** 3). **Random Forests:** Builds multiple decision trees and averages their predictions for better accuracy. 4). **Support Vector Machines (SVMs):** Finds the optimal hyperplane to separate fault classes in high-dimensional space.	
Q5.	How to read the Classification_Report generated by several models in the given problem statement: https://www.kaggle.com/code/pythonafroz/fault-prediction-using-decisiontree-algorithm	5
Ans5.	**1). Precision (Positive Predictive Value):** • Precision measures the proportion of predicted positive instances (predicted faults) that were actually correct. • A high precision indicates that the model minimizes false positives, or false alarms.	

	2). Recall (Sensitivity/True Positive Rate): • Recall measures the proportion of actual positive instances (faults) that were correctly identified by the model. • A high recall indicates that the model is effective at minimizing false negatives, or missed faults. **3). F1-Score (Harmonic Mean of Precision and Recall):** • The F1-score provides a balanced measure of a model's performance by considering both precision and recall. • It is particularly useful when dealing with imbalanced datasets, where either false positives or false negatives are particularly costly. • A higher F1-score signifies a better balance between correctly identifying faults and minimizing both false alarms and missed faults. **4). Support (Class Distribution):** • Support represents the number of actual occurrences of each class (fault type) within the test dataset. • It indicates the prevalence of each fault type, providing context for the other metrics. **5). Accuracy (Overall Correctness):** • It gives an overall sense of how often the model makes correct predictions. • It is calculated by dividing the number of correct predictions by the total number of predictions.	
Q6.	From the mentioned link: https://www.kaggle.com/code/pythonafroz/faultprediction-using-decision-tree-algorithm Do one sight analysis and figure out which algorithms work well on the given dataset. And on what basis are Model comparisons done over there?	5
Ans6.	By checking how well the models did on both the training data and new, unseen data, one can say that Random Forest, Decision Tree, and XGB Classifier all got 100% accuracy as these models perfectly predicted the faults in the dataset. **Basis of Models Comparisons:** • **Training Accuracy:** How well the models did on the data they were trained on. • **Model Accuracy Score:** How the models performed on data they hadn't seen before (test data). This shows how well the model can generalize. • **Classification Report:** This uses a detailed report that includes things like precision, recall, and F1-score to see how well the models did for each type of fault.	

Name: Kanishka Jingar	**Roll No.: 22/AI/023**

ASSIGNMENT QUESTION 2: CO4

Customer Segmentation using Unsupervised Problem

Context:

You are a data scientist working for a retail company that wants to improve its marketing strategies by better understanding customer behaviour. One approach is to segment customers into distinct groups based on their purchasing habits. This will allow the company to tailor marketing campaigns to specific groups, ultimately increasing sales and customer satisfaction.

Task:

Design and explain the customer segmentation model using any one of the unsupervised algorithms.

1. Data Collection: Obtain a dataset containing customer purchase history, including details such as purchase frequency, amount spent, types of products purchased, etc. You may use publicly available datasets or simulate data for this assignment Include the first 10 rows of the dataset that you are going to consider. 5 marks

2. Data Preprocessing: Clean the dataset and perform necessary preprocessing steps such as normalization, handling missing values, and feature engineering. 5 marks

3. Unsupervised Learning (Clustering): Apply an unsupervised learning algorithm (e.g., Kmeans clustering, hierarchical clustering) to segment customers into distinct groups based on their purchasing behaviour. 5 marks

4. Evaluate the clustering results using appropriate evaluation metrics. 5 marks

Github Link:
https://github.com/Kanishka-Jingar/Machine_Learning-Lab/blob/main/Assignment/Assignment2.ipynb

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

[9]:
```
# 1. Data Collection
data = {
    'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'PurchaseFrequency': [7, 4, 8, 7, 6, 4, 8, 1, 5, 5],
    'AmountSpent': [20, 300, 250, 50, 130, 180, 40, 450, 280, 500],
    'Types of product purchased': [2, 2, 4, 3, 4, 2, 5, 3, 3, 1]
}
```

[10]:
```
df = pd.DataFrame(data)

print(df.head(10))
```

```
   CustomerID  PurchaseFrequency  AmountSpent  Types of product purchased
0           1                  7           20                           2
1           2                  4          300                           2
2           3                  8          250                           4
3           4                  7           50                           3
4           5                  6          130                           4
5           6                  4          180                           2
6           7                  8           40                           5
7           8                  1          450                           3
8           9                  5          280                           3
9          10                  5          500                           1
```

[36]:
```
# 2. Data Preprocessing
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df[['PurchaseFrequency', 'AmountSpent',␣
 ↪'Types of product purchased']])
```

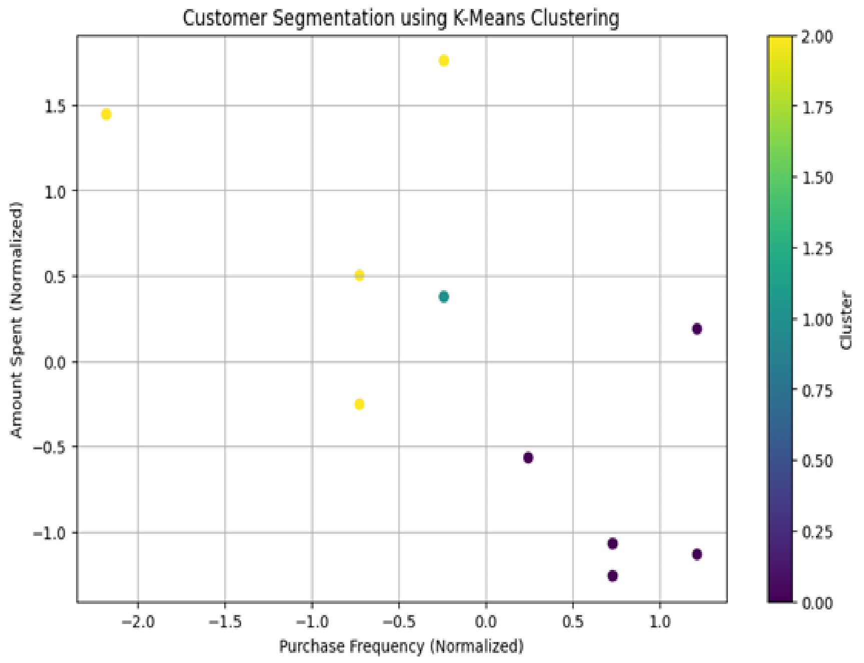
```
scaled_df = pd.DataFrame(scaled_features, columns=['PurchaseFrequency',␣
↪'AvgSpending', 'Types of product purchased'])
scaled_df['CustomerID'] = df['CustomerID']
```

[49]:

```
# 3. Unsupervised Learning (K-means Clustering)
def segment_customers(df, n_clusters=3): # Defining a function to segment␣
↪customers and setting default clusters to 5
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    df['Cluster'] = kmeans.fit_predict(df[['PurchaseFrequency', 'AmountSpent',␣
↪'Types of product purchased']])

# Visualization of Clusters
    plt.figure(figsize=(10, 6))
    plt.scatter(df['PurchaseFrequency'], df['AmountSpent'], c=df['Cluster'],␣
↪cmap='viridis', marker='o')
    plt.xlabel('Purchase Frequency (Normalized)')
    plt.ylabel('Amount Spent (Normalized)')
    plt.title('Customer Segmentation using K-Means Clustering')
    plt.colorbar(label='Cluster')
    plt.grid()
    plt.show()
    return df

segmented_df = segment_customers(df, n_clusters=3)
print(segmented_df.head())
```



```
   CustomerID  PurchaseFrequency  AmountSpent  Types of product purchased  \
0           1           0.727607    -1.256893                   -0.792406
1           2          -0.727607     0.502757                   -0.792406
2           3           1.212678     0.188534                    0.968496
3           4           0.727607    -1.068359                    0.088045
4           5           0.242536    -0.565602                    0.968496

   Cluster  ProductTypes
0        0     -0.792406
1        2     -0.792406
2        0      0.968496
3        0      0.088045
4        0      0.968496
```

[50]:
```
# Calculate and print inertia
inertia = kmeans.inertia_
print(f'Inertia: {inertia}')

# Calculate and print silhouette score
silhouette_avg = silhouette_score(df[['PurchaseFrequency', 'AmountSpent',␣
 ↪'Types of product purchased']], df['Cluster'])
print(f"Silhouette Score: {silhouette_avg}")

# Printing the Cluster Analysis for dataframe
print(df)
```

```
Inertia: 12.289737974987196
Silhouette Score: -0.008930242493551094
   CustomerID  PurchaseFrequency  AmountSpent  Types of product purchased  \
0           1           0.727607    -1.256893                   -0.792406
1           2          -0.727607     0.502757                   -0.792406
2           3           1.212678     0.188534                    0.968496
3           4           0.727607    -1.068359                    0.088045
4           5           0.242536    -0.565602                    0.968496
5           6          -0.727607    -0.251379                   -0.792406
6           7           1.212678    -1.131203                    1.848947
7           8          -2.182821     1.445426                    0.088045
8           9          -0.242536     0.377068                    0.088045
9          10          -0.242536     1.759650                   -1.672857

   Cluster  ProductTypes
0        0     -0.792406
1        2     -0.792406
2        0      0.968496
3        0      0.088045
4        0      0.968496
5        2     -0.792406
6        0      1.848947
7        2      0.088045
8        1      0.088045
9        2     -1.672857
```