Slip 1

Q1.Write a R program to add, multiply and divide two vectors of integer type. (Vector length should be minimum 4) [10 Marks]

```
vector1 <- c(4, 8, 12, 16)
vector2 <- c(2, 4, 6, 8)
add_vectors <- function(v1, v2) {
  return(v1 + v2)
}
multiply_vectors <- function(v1, v2) {
  return(v1 * v2)
}
divide_vectors <- function(v1, v2) {
  return(ifelse(v2 != 0, v1 / v2, NA))
}
sum_result <- add_vectors(vector1, vector2)
product_result <- multiply_vectors(vector1, vector2)
division_result <- divide_vectors(vector1, vector2)
cat("Vector 1: ", vector1, "\n")
cat("Vector 2: ", vector2, "\n")
cat("Sum: ", sum_result, "\n")
cat("Product: ", product_result, "\n")
cat("Division: ", division_result, "\n")
```

Q2.Consider the student data set. It can be downloaded from: https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO 5_6dIOw . Write a programme in python to apply simple linear regression and find out mean absolute error, mean squared error and root mean squared error. [20 Marks]

```
pip install pandas numpy scikit-learn // terminal command
```

jupyter notebook

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
url = 'https://drive.google.com/uc?id=1oakZCv7g3mlmCSdv9J8kdSaqO5_6dIOw'
data = pd.read_csv(url)
print("Dataset preview:")
print(data.head())
X = data[['Hours']]  # Predictor variable
y = data['Scores']   # Response variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
```

Slip 2

Q1. Write an R program to calculate the multiplication table using a function.

[10 Marks]

```
generate_multiplication_table <- function(number, max_multiplier) {
  for (i in 1:max_multiplier) {
    result <- number * i
```

```
   cat(number, "x", i, "=", result, "\n")
  }
}
number <- as.integer(readline(prompt = "Enter a number to generate its multiplication table: "))
max_multiplier <- as.integer(readline(prompt = "Enter the maximum multiplier: "))
cat("Multiplication Table for", number, ":\n")
generate_multiplication_table(number, max_multiplier)
```

Q2. Write a python program to implement k-means algorithms on a synthetic dataset. [20 Marks]

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
n_samples = 300
n_clusters = 4
random_state = 42
X, y_true = make_blobs(n_samples=n_samples, centers=n_clusters, cluster_std=0.60, random_state=random_state)
kmeans = KMeans(n_clusters=n_clusters, random_state=random_state)
y_kmeans = kmeans.fit_predict(X)
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title('K-Means Clustering on Synthetic Dataset')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid()
plt.show()
```

Slip 3

Q1. Write a R program to reverse a number and also calculate the sum of digits of that number. [10 Marks]

```
reverse_and_sum <- function(number) {
  original_number <- number
  reversed_number <- 0
  sum_of_digits <- 0
  while (number > 0) {
    digit <- number %% 10
    reversed_number <- reversed_number * 10 + digit
    sum_of_digits <- sum_of_digits + digit
    number <- number %/% 10
  }
  return(list(reversed = reversed_number, sum = sum_of_digits))
}
number <- as.integer(readline(prompt = "Enter a number: "))
result <- reverse_and_sum(number)
cat("Reversed Number:", result$reversed, "\n")
cat("Sum of Digits:", result$sum, "\n")
```

Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b0 and b1.(use numpy package)

x=[0,1,2,3,4,5,6,7,8,9,11,13]

y = ([1, 3, 2, 5, 7, 8, 8, 9, 10, 12,16, 18] [20 Marks]

```
import numpy as np
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12, 16, 18])
n = len(x)
```

```
x_mean = np.mean(x)
y_mean = np.mean(y)
b1_numerator = np.sum((x - x_mean) * (y - y_mean))
b1_denominator = np.sum((x - x_mean) ** 2)
b1 = b1_numerator / b1_denominator
b0 = y_mean - b1 * x_mean
print(f"Estimated coefficients:\n b0 (intercept): {b0}\n b1 (slope): {b1}")
```

Slip 4

Q1. Write a R program to calculate the sum of two matrices of given size. [10 Marks]

```
input_matrix <- function(rows, cols, name) {
  cat(paste("Enter the elements of matrix", name, "row-wise:\n"))
  matrix_data <- numeric(rows * cols)  # Create a numeric vector to hold the matrix elements
  for (i in 1:(rows * cols)) {
    matrix_data[i] <- as.numeric(readline(prompt = paste("Element", i, ": ")))
  }
  return(matrix(matrix_data, nrow = rows, ncol = cols))  # Convert vector to matrix
}
rows <- as.integer(readline(prompt = "Enter the number of rows: "))
cols <- as.integer(readline(prompt = "Enter the number of columns: "))
matrix1 <- input_matrix(rows, cols, "A")
matrix2 <- input_matrix(rows, cols, "B")
matrix_sum <- matrix1 + matrix2
cat("Matrix A:\n")
print(matrix1)
cat("Matrix B:\n")
print(matrix2)
cat("Sum of Matrix A and B:\n")
print(matrix_sum)
```

Q2. Consider following dataset

weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny','Rainy','Sunn y','Overcast','Overcast','Rainy']

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild']

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No'].

Use Naïve Bayes algorithm to predict [0: Overcast, 2: Mild] tuple belongs to which class whether to play the sports or not.

[20 Marks]

```
import numpy as np
import pandas as pd
weather = ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy',
          'Overcast', 'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcast',
          'Overcast', 'Rainy']
temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool',
        'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild',
        'Hot', 'Mild']
play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No',
        'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes',
        'Yes', 'No']
data = pd.DataFrame({
    'Weather': weather,
    'Temperature': temp,
    'Play': play
})
prior_play = data['Play'].value_counts(normalize=True)
print("Prior Probabilities:\n", prior_play)
likelihood = {}
for feature in ['Weather', 'Temperature']:
    likelihood[feature] = {}
```

```
    for class_value in prior_play.index:
        likelihood[feature][class_value] = data[data['Play'] ==
class_value][feature].value_counts(normalize=True)
input_tuple = ('Overcast', 'Mild')
posterior = {}
for class_value in prior_play.index:
    posterior[class_value] = prior_play[class_value]  # Start with prior probability
    for i, feature in enumerate(['Weather', 'Temperature']):
        posterior[class_value] *= likelihood[feature][class_value].get(input_tuple[i], 0)
print("\nPosterior Probabilities:\n", posterior)
predicted_class = max(posterior, key=posterior.get)
print("\nPredicted Class for the input tuple (Weather: 'Overcast', Temperature: 'Mild'):", predicted_class)
```

Slip 5

Q1. Write a R program to concatenate two given factors. [10 Marks]

```
concatenate_factors <- function(factor1, factor2) {
  combined_factor <- c(factor1, factor2)
  result <- factor(combined_factor)
  return(result)
}
factor1 <- factor(c("Apple", "Banana", "Cherry"))
factor2 <- factor(c("Date", "Elderberry", "Fig"))
result_factor <- concatenate_factors(factor1, factor2)
print(result_factor)
```

Q2. Write a Python program build Decision Tree Classifier using Scikit- learn package for

diabetes data set (download database from https://www.kaggle.com/uciml/pimaindians-diabetes-database)

[20 Marks]

```
import pandas as pd
```

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
from sklearn import tree
data = pd.read_csv('diabetes.csv')
print(data.head())
print(data.info())
print(data.describe())
X = data.drop('Outcome', axis=1)  # Features
y = data['Outcome']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print('Confusion Matrix:\n', conf_matrix)
print('Classification Report:\n', class_report)
plt.figure(figsize=(12, 8))
tree.plot_tree(model, feature_names=X.columns, class_names=['No Diabetes', 'Diabetes'], filled=True)
plt.title("Decision Tree for Diabetes Classification")
plt.show()
```

Slip 6

Q1. Write a R program to create a data frame using two given vectors and display the duplicate

elements. [10 Marks]

```
find_duplicates <- function(vec1, vec2) {
  df <- data.frame(Vector1 = vec1, Vector2 = vec2)
  cat("Data Frame:\n")
  print(df)
  duplicates_vec1 <- vec1[duplicated(vec1) | duplicated(vec1, fromLast = TRUE)]
  duplicates_vec2 <- vec2[duplicated(vec2) | duplicated(vec2, fromLast = TRUE)]
  cat("\nDuplicate elements in Vector 1:\n")
  print(unique(duplicates_vec1))
    cat("\nDuplicate elements in Vector 2:\n")
  print(unique(duplicates_vec2))
}
vector1 <- c(1, 2, 3, 2, 4, 5, 3, 6)
vector2 <- c("A", "B", "C", "A", "D", "B", "E")
find_duplicates(vector1, vector2)
```

Q2. Write a python program to implement hierarchical Agglomerative clustering algorithm. (Download Customer.csv dataset from github.com).

[20 Marks]

```
pip install pandas matplotlib seaborn scipy scikit-learn // install terminal
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler
url = 'https://raw.githubusercontent.com/yourusername/yourrepo/main/Customer.csv'
data = pd.read_csv(url)
data.drop(['CustomerID'], axis=1, inplace=True, errors='ignore')
```

```
scaler = StandardScaler()

data_scaled = scaler.fit_transform(data)

linked = linkage(data_scaled, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', labels=data.index, distance_sort='descending',
show_leaf_counts=True)

plt.title('Dendrogram for Hierarchical Clustering')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.show()

num_clusters = 3

clusters = fcluster(linked, num_clusters, criterion='maxclust')

data['Cluster'] = clusters

plt.figure(figsize=(10, 6))

sns.scatterplot(x=data[data.columns[0]], y=data[data.columns[1]], hue=data['Cluster'], palette='deep')

plt.title('Hierarchical Agglomerative Clustering Results')

plt.xlabel(data.columns[0])

plt.ylabel(data.columns[1])

plt.legend(title='Cluster')

plt.show()
```

Slip 7

Q1. Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

[10 Marks]

```
sequence_20_to_50 <- 20:50

print("Sequence from 20 to 50:")

print(sequence_20_to_50)

mean_20_to_60 <- mean(20:60)

cat("Mean of numbers from 20 to 60:", mean_20_to_60, "\n")
```

```
sum_51_to_91 <- sum(51:91)

cat("Sum of numbers from 51 to 91:", sum_51_to_91, "\n")
```

Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse the performance of the model

(Use sklearn package)

x = np.array([1,2,3,4,5,6,7,8])

y = np.array([7,14,15,18,19,21,26,23]) [20 Marks]

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)  # Reshape for sklearn

y = np.array([7, 14, 15, 18, 19, 21, 26, 23])

model = LinearRegression()

model.fit(x, y)

b0 = model.intercept_  # y-intercept

b1 = model.coef_[0]    # slope

print(f"Estimated coefficients:\nb0 (intercept) = {b0}\nb1 (slope) = {b1}")

y_pred = model.predict(x)

mae = mean_absolute_error(y, y_pred)

mse = mean_squared_error(y, y_pred)

r2 = r2_score(y, y_pred)

print(f"\nModel Performance:\nMean Absolute Error (MAE) = {mae}\nMean Squared Error (MSE) = {mse}\nR² Score = {r2}")

plt.scatter(x, y, color='blue', label='Actual data')

plt.plot(x, y_pred, color='red', label='Fitted line')

plt.xlabel('X values')

plt.ylabel('Y values')

plt.title('Simple Linear Regression')
```

```
plt.legend()
plt.show()
```

Slip 8

Q1. Write a R program to get the first 10 Fibonacci numbers. [10 Marks]

```
fibonacci <- function(n) {
  fib_sequence <- numeric(n)
  fib_sequence[1] <- 0  # First Fibonacci number
  fib_sequence[2] <- 1  # Second Fibonacci number
  for (i in 3:n) {
    fib_sequence[i] <- fib_sequence[i - 1] + fib_sequence[i - 2]  # Calculate next Fibonacci number
  }
  return(fib_sequence)
}
first_10_fib <- fibonacci(10)
cat("The first 10 Fibonacci numbers are:\n")
print(first_10_fib)
```

Q2. Write a python program to implement k-means algorithm to build prediction model (Use Credit Card Dataset CC GENERAL.csv Download from kaggle.com) [20 Marks]

```
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
url = 'https://raw.githubusercontent.com/your-repo/CC-GENERAL.csv'  # Update with actual path if needed
data = pd.read_csv(url)
data.drop(columns=['CUST_ID'], inplace=True)
data.fillna(0, inplace=True)  # Replace missing values with 0
```

```
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
inertia = []
silhouette_scores = []
K = range(2, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(data_scaled, kmeans.labels_))
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(K, inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.subplot(1, 2, 2)
plt.plot(K, silhouette_scores, marker='o')
plt.title('Silhouette Scores for k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.show()
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['Cluster'] = kmeans.fit_predict(data_scaled)
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
print("Cluster Centers (original scale):")
print(cluster_centers)
```

```
print("\nNumber of points in each cluster:")
print(data['Cluster'].value_counts())
plt.scatter(data_scaled[:, 0], data_scaled[:, 1], c=data['Cluster'], cmap='viridis')
plt.title('Clusters Visualization')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

Slip 9

Q1. Write an R program to create a Data frames which contain details of 5 employees and display summary of the data. [10 Marks]

```
employee_data <- data.frame(
  EmployeeID = 1:5,
  Name = c("John Doe", "Jane Smith", "Alice Johnson", "Bob Brown", "Charlie Davis"),
  Age = c(28, 34, 29, 40, 37),
  Department = c("HR", "Finance", "IT", "Marketing", "Sales"),
  Salary = c(50000, 60000, 55000, 62000, 58000)
)
print("Employee Data:")
print(employee_data)
print("Summary of Employee Data:")
summary(employee_data)
```

Q2. Write a Python program to build an SVM model to Cancer dataset. The dataset is available in the scikit-learn library. Check the accuracy of model with precision and recall. [20 Marks]

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
```

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report
data = load_breast_cancer()
X = data.data  # Features
y = data.target  # Target labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = SVC(kernel='linear')  # You can choose different kernels like 'rbf', 'poly', etc.
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Slip 10

Q1. Write a R program to find the maximum and the minimum value of a given vector [10 Marks]

```
numbers <- c(10, 25, 5, 40, 15, 30, 2)
max_value <- max(numbers)
min_value <- min(numbers)
cat("The maximum value in the vector is:", max_value, "\n")
cat("The minimum value in the vector is:", min_value, "\n")
```

Q2. Write a Python Programme to read the dataset (“Iris.csv”). dataset download from (https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm. [20 Marks]

```
pip install pandas mlxtend // terminal command
import pandas as pd
```

```
from mlxtend.frequent_patterns import apriori, association_rules
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris_data = pd.read_csv(url, header=None, names=columns)
print("Iris Dataset:")
print(iris_data.head())
iris_data['species'] = iris_data['species'].astype('category')
one_hot = pd.get_dummies(iris_data, columns=['species'])
frequent_itemsets = apriori(one_hot, min_support=0.1, use_colnames=True)
print("\nFrequent Itemsets:")
print(frequent_itemsets)
rules = association_rules(frequent_itemsets, metric="support", min_threshold=0.1)
print("\nAssociation Rules:")
print(rules)
```

Slip 11

Q1. Write a R program to find all elements of a given list that are not in another given list.

= list("x", "y", "z")

= list("X", "Y", "Z", "x", "y", "z") [10 Marks]

```
list1 <- list("x", "y", "z")
list2 <- list("X", "Y", "Z", "x", "y", "z")
vector1 <- unlist(list1)
vector2 <- unlist(list2)
not_in_list2 <- vector1[!(vector1 %in% vector2)]
cat("Elements in list1 that are not in list2:", not_in_list2, "\n")
```

Q2. Write a python program to implement hierarchical clustering algorithm. (Download Wholesale customers data dataset from github.com).

```
pip install pandas scipy matplotlib seaborn // terminal command
import pandas as pd
```

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.preprocessing import StandardScaler

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/Wholesale%20customers.csv"

data = pd.read_csv(url)

print("Dataset Overview:")

print(data.head())

features = data.drop(['Channel', 'Region'], axis=1)

scaler = StandardScaler()

scaled_features = scaler.fit_transform(features)

linked = linkage(scaled_features, method='ward')

plt.figure(figsize=(10, 7))

dendrogram(linked, orientation='top', labels=data.index, distance_sort='descending',
show_leaf_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Customer Index')

plt.ylabel('Distance')

plt.show()
```

Slip 12

Q1. Write a R program to create a Dataframes which contain details of 5 employees and display the details.

Employee contain (empno,empname,gender,age,designation)

[10 Marks]

```
employees <- data.frame(
  empno = c(101, 102, 103, 104, 105),
  empname = c("Alice", "Bob", "Charlie", "David", "Eva"),
  gender = c("Female", "Male", "Male", "Male", "Female"),
```

```
  age = c(25, 30, 28, 35, 27),
  designation = c("Manager", "Developer", "Designer", "Analyst", "Tester"),
  stringsAsFactors = FALSE  # Avoid converting strings to factors
)
print("Employee Details:")
print(employees)
```

Q2. Write a python program to implement multiple Linear Regression model for a car dataset.

Dataset can be downloaded from:

https://www.w3schools.com/python/python_ml_multiple_regression.asp

[20 Marks]

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
url = "https://www.w3schools.com/python/pandas/data/carprices.csv"
data = pd.read_csv(url)
print("Dataset Head:")
print(data.head())
X = data[['Age', 'Mileage']]
y = data['Price']
X = sm.add_constant(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = sm.OLS(y_train, X_train).fit()
print("\nModel Summary:")
print(model.summary())
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print("\nPerformance Metrics:")
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

Slip 13

Q1. Draw a pie chart using R programming for the following data distribution:

Digits on

Dice

1 2 3 4 5 6

Frequency of

getting each

number

7 2 6 3 4 8

[10 Marks]

```
digits_on_dice <- c(1, 2, 3, 4, 5, 6)
frequency <- c(7, 2, 6, 3, 4, 8)
labels <- paste("Digit:", digits_on_dice, "\nFrequency:", frequency)
pie(frequency,
   labels = labels,
   main = "Frequency of Getting Each Number on Dice",
   col = rainbow(length(digits_on_dice)))
legend("topright",
      legend = labels,
      fill = rainbow(length(digits_on_dice)),
      cex = 0.8)
```

Q2. Write a Python program to read "StudentsPerformance.csv" file. Solve following:

- To display the shape of dataset.

- To display the top rows of the dataset with their columns. Note: Download

dataset from following link :

(https://www.kaggle.com/spscientist/students-performance-inexams?

select=StudentsPerformance.csv) [20 Marks]

```
import pandas as pd
url = "https://raw.githubusercontent.com/your_username/your_repo/main/StudentsPerformance.csv"
data = pd.read_csv(url)
print("Shape of the dataset:", data.shape)
print("\nTop rows of the dataset:")
print(data.head())
```

Slip 14

Q1. Write a script in R to create a list of employees (name) and perform the following:

a. Display names of employees in the list.

b. Add an employee at the end of the list

c. Remove the third element of the list. [10 Marks]

```
employees <- list("Alice", "Bob", "Charlie", "David", "Eve")
cat("Employees in the list:\n")
print(employees)
new_employee <- "Frank"
employees <- append(employees, new_employee)
cat("\nEmployees after adding a new employee:\n")
print(employees)
employees <- employees[-3]  # Removing the third element
cat("\nEmployees after removing the third element:\n")
print(employees)
```

Q2. Write a Python Programme to apply Apriori algorithm on Groceries dataset. Dataset

can be downloaded from

(https://github.com/amankharwal/Websitedata/blob/master/Groceries

_dataset.csv).

Also display support and confidence for each rule.

[20 Marks]

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
url = 'https://github.com/amankharwal/Websitedata/raw/master/Groceries_dataset.csv'
data = pd.read_csv(url)
print("Data Sample:")
print(data.head())
transactions = data.groupby(['Member_number', 'Date'])['Item_description'].apply(list).tolist()
from mlxtend.preprocessing import TransactionEncoder
encoder = TransactionEncoder()
onehot = encoder.fit(transactions).transform(transactions)
onehot_df = pd.DataFrame(onehot, columns=encoder.columns_)
frequent_itemsets = apriori(onehot_df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.1)
print("\nAssociation Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence']])
```

Slip 15

Q1.Write a R program to add, multiply and divide two vectors of integer type. (vector length should be minimum 4) [10 Marks]

```
vector1 <- c(10, 20, 30, 40)
vector2 <- c(2, 4, 5, 10)
addition <- vector1 + vector2
multiplication <- vector1 * vector2
division <- vector1 / vector2
cat("Vector 1:", vector1, "\n")
cat("Vector 2:", vector2, "\n")
cat("Addition Result:", addition, "\n")
cat("Multiplication Result:", multiplication, "\n")
```

```
cat("Division Result:", division, "\n")
```

Q2. Write a Python program build Decision Tree Classifier for shows.csv from pandas and predict class label for show starring a 40 years old American comedian, with 10 years of experience, and a comedy ranking of 7? Create a csv file as shown in https://www.w3schools.com/python/python_ml_decision_tree.asp

[20 Marks]

Create CSV file :-

```
Age,Experience,ComedyRanking,ClassLabel
25,2,5,Not Popular
30,5,6,Popular
35,8,8,Popular
40,10,7,Popular
45,12,5,Not Popular
50,15,4,Not Popular
```

Code :-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
data = pd.read_csv('shows.csv')
print("Dataset:\n", data)
X = data[['Age', 'Experience', 'ComedyRanking']]
y = data['ClassLabel']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
new_show = [[40, 10, 7]]  # 40 years old, 10 years of experience, comedy ranking of 7
predicted_class = clf.predict(new_show)
print(f"The predicted class label for the show is: {predicted_class[0]}")
```

Slip 16

Q1. Write a R program to create a simple bar plot of given data

Year Export Import

2001 26 35

2002 32 40

2003 35 50

[10 Marks]

```
data <- data.frame(
  Year = c(2001, 2002, 2003),
  Export = c(26, 32, 35),
  Import = c(35, 40, 50)
)
library(ggplot2)
data_long <- reshape2::melt(data, id.vars = "Year")
ggplot(data_long, aes(x = Year, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Export and Import Data (2001-2003)",
       x = "Year",
       y = "Value",
       fill = "Category") +
  theme_minimal()
```

Q2. Write a Python program build Decision Tree Classifier using Scikit-learn package for

diabetes data set (download database from https://www.kaggle.com/uciml/pima-indiansdiabetes-database)

[20 Marks]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
column_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
                'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
data = pd.read_csv(url, names=column_names)
print("First few rows of the dataset:")
print(data.head())
X = data.drop('Outcome', axis=1)  # Features
y = data['Outcome']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print("\nAccuracy of the model: {:.2f}%".format(accuracy * 100))
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

Slip 17

Q1. Write a R program to get the first 20 Fibonacci numbers.

[10 Marks]

```
fibonacci <- function(n) {
  fib_sequence <- numeric(n)  # Create a numeric vector of length n
  fib_sequence[1] <- 0       # First Fibonacci number
  fib_sequence[2] <- 1       # Second Fibonacci number
```

```
  for (i in 3:n) {
    fib_sequence[i] <- fib_sequence[i - 1] + fib_sequence[i - 2]  # Calculate next Fibonacci number
  }
  return(fib_sequence)
}
first_20_fib <- fibonacci(20)
print("The first 20 Fibonacci numbers are:")
print(first_20_fib)
```

Q2. Write a python programme to implement multiple linear regression model for stock market data frame as follows:

```
Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2
016,20,16,2016,2016,2016,2016,2016,2016,2016,2016,2016],
'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1
.75,1.75,1.75,1.75,1.75,1.75,1.75],
'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5
.9,6.2,6.2,6.1],
'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,
965,943,958,971,949,884,866,876,822,704,719] }
```

And draw a graph of stock market price verses interest rate.

[20 Marks]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
Stock_Market = {
    'Year': [2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017,
             2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016, 2016],
    'Month': [12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
              12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1],
    'Interest_Rate': [2.75, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.25, 2.25, 2.25,
                      2, 2, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75, 1.75,
                      1.75, 1.75, 1.75],
    'Unemployment_Rate': [5.3, 5.3, 5.3, 5.3, 5.4, 5.6, 5.5, 5.5, 5.5, 5.6,
                          5.7, 5.9, 6, 5.9, 5.8, 6.1, 6.2, 6.1, 6.1, 6.1,
                          5.9, 6.2, 6.2, 6.1],
    'Stock_Index_Price': [1464, 1394, 1357, 1293, 1256, 1254, 1234, 1195,
                          1159, 1167, 1130, 1075, 1047, 965, 943, 958,
                          971, 949, 884, 866, 876, 822, 704, 719]
}
df = pd.DataFrame(Stock_Market)
X = df[['Interest_Rate', 'Unemployment_Rate']]
y = df['Stock_Index_Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
print("Coefficients:")
print(f"Intercept (b0): {model.intercept_}")
print(f"Interest Rate (b1): {model.coef_[0]}")
print(f"Unemployment Rate (b2): {model.coef_[1]}")
plt.figure(figsize=(10, 6))
plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'], color='blue', label='Data Points')
plt.title('Stock Index Price vs Interest Rate')
plt.xlabel('Interest Rate (%)')
```

```
plt.ylabel('Stock Index Price')
plt.grid()
plt.legend()
plt.show()
```

Slip 18

Q1. Write a R program to find the maximum and the minimum value of a given vector [10 Marks]

```
numbers <- c(12, 5, 8, 19, 3, 27, 15)
max_value <- max(numbers)
min_value <- min(numbers)
cat("The maximum value is:", max_value, "\n")
cat("The minimum value is:", min_value, "\n")
```

Q2. Consider the following observations/data. And apply simple linear regression and find out estimated coefficients b1 and b1 Also analyse the performance of the model

(Use sklearn package)

x = np.array([1,2,3,4,5,6,7,8])

y = np.array([7,14,15,18,19,21,26,23]) [20 Marks]

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
x = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)  # Reshape for sklearn
y = np.array([7, 14, 15, 18, 19, 21, 26, 23])
model = LinearRegression()
model.fit(x, y)
b0 = model.intercept_
b1 = model.coef_[0]
```

```
y_pred = model.predict(x)
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)
print(f"Estimated coefficients:")
print(f"b0 (Intercept): {b0}")
print(f"b1 (Slope): {b1}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

Slip 19

Q1. Write a R program to create a Dataframes which contain details of 5 Students and display the details.

Students contain (Rollno,Studname,Address,Marks) [10 Marks]

```
Rollno <- c(1, 2, 3, 4, 5)
Studname <- c("Alice", "Bob", "Charlie", "David", "Eve")
Address <- c("123 Maple St", "456 Oak St", "789 Pine St", "135 Birch St", "246 Cedar St")
Marks <- c(85, 90, 78, 88, 92)
students_df <- data.frame(Rollno, Studname, Address, Marks)
print(students_df)
```

Q2. Write a python program to implement multiple Linear Regression model for a car dataset.

Dataset can be downloaded from:

https://www.w3schools.com/python/python_ml_multiple_regression.asp

[20 Marks]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
url = "https://www.w3schools.com/python/pandas/data/cardata.csv"
data = pd.read_csv(url)
```

```
print("Dataset head:")
print(data.head())
X = data[['Mileage', 'Age', 'Horsepower']]
y = data['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("\nModel Coefficients:")
print(f"Intercept: {model.intercept_}")
print(f"Coefficients: {model.coef_}")
print("\nModel Performance:")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
new_car = pd.DataFrame({'Mileage': [20000], 'Age': [3], 'Horsepower': [130]})
predicted_price = model.predict(new_car)
print(f"\nPredicted price for the new car: {predicted_price[0]}")
```

Slip 20

Q1. Write a R program to create a data frame from four given vectors.

[10 Marks]

```
names <- c("Alice", "Bob", "Charlie", "Diana")
ages <- c(25, 30, 22, 28)
scores <- c(85.5, 90.0, 78.5, 88.0)
cities <- c("New York", "Los Angeles", "Chicago", "Houston")
students_df <- data.frame(Name = names, Age = ages, Score = scores, City = cities)
print(students_df)
```

Q2. Write a python program to implement hierarchical Agglomerative clustering algorithm. (Download Customer.csv dataset from github.com).

[20 Marks]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
url = "https://raw.githubusercontent.com/amankharwal/WebsiteData/master/Customer.csv"
data = pd.read_csv(url)
print("First few rows of the dataset:")
print(data.head())
features = data.select_dtypes(include=[np.number])
linked = linkage(features, method='ward')
plt.figure(figsize=(10, 7))
dendrogram(linked,
        orientation='top',
        distance_sort='descending',
        show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
plt.show()
n_clusters = 4  # Example: change as required
agglo = AgglomerativeClustering(n_clusters=n_clusters)
clusters = agglo.fit_predict(features)
data['Cluster'] = clusters
print("\nData with cluster labels:")
print(data.head())
```