**22AIE213 – MACHINE LEARNING**

Lab Assignment Report – 03

**Submitted by**

Nossam Sri Chakradhar - BL.EN. U4AIE22132

Gopa Pulastya - BL.EN. U4AIE22116

Rishi Anirudh K - BL.EN. U4AIE22139

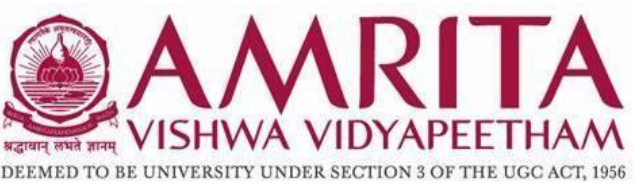
**For the fulfilment of the Course 22AIE213**

**Of**

**BACHELOR OF TECHNOLOGY**

**IN**

**“ARTIFICIAL INTELLIGENCE”**



AMRITA SCHOOL OF COMPUTING, BANGLORE

AMRITA VISHWA VIDYAPEETHAM

BENGALURU – 560035

**A-1:**

**Pseudocode:**

1. INITIALIZE an empty matrix A
2. INITIALIZE an empty vector C
3. FOR each row IN the data:

Append all elements of the row except the last one to matrix A

Append the last element of the row to vector C

1. PRINT matrix A
2. PRINT vector C
3. PRINT "The Dimensionality of the vector space is the number of features, which is the number of columns in matrix A."
4. PRINT "The number of vectors is the number of rows in matrix A."
5. Calculate the rank of matrix A using NumPy's linalg.matrix\_rank function
6. PRINT "The Rank of Matrix:"
7. COMPUTE the pseudo inverse of matrix A using NumPy's linalg.pinv() function
8. PRINT the pseudo inverse

**Explanation:** This program Segregates a dataset into two matrices, one containing the features and the other containing the target values. Calculates and prints the dimensionality of the vector space, which is the number of features in the dataset. Determines and prints the rank of the feature matrix, indicating the maximum number of linearly independent columns, crucial for understanding the dataset's structure and redundancy.

Dimensionality of the Vector Space:

Dimensionality refers to the number of features or attributes that characterize each data point in a dataset. In this case, the data consists of observations where each observation has four values in which three features representing different characteristics and one target value representing the payment details. So, the dimensionality of the vector space here is 3, as there are three features. These features form the basis of the vector space in which the data resides.

Number of Vectors in the Vector Space:

The number of vectors in a vector space corresponds to the number of data points or observations in the dataset. Each data point can be represented as a vector in the vector space. In this dataset, there are 10 rows, each representing a different observation. So, there are 10 vectors in this vector space.

Rank of Matrix A:

The rank of a matrix is the maximum number of linearly independent column vectors in the matrix. In simpler terms, it tells us the number of truly informative features in the data. We calculate it using techniques from linear algebra. In this problem we find the rank of matrix A because it contains only the features from the dataset. By finding the rank of matrix A, we can understand how many of these features are linearly independent and contribute unique information. It helps us understand the data. The rank is important because if it's less than the number of features, it indicates some level of collinearity in the data.

**A-2:**

**Pseudocode:**

1. INITIALIZE list for vector X
2. CALCULATE the model vector X:

ITERATE over each row in the pseudo-inverse matrix:

INITIALIZE a variable for the dot product, set it to 0

Iterate over each element in the row and its corresponding in the payment vector C:

Multiply the elements and add the result to the dot product variable

APPEND the dot product to the model vector X list

PRINT the model vector X

1. INITALIZE list FOR the product costs

Calculate the cost of each product:

Iterate over each row in matrix A:

Initialize a variable for the dot product, set it to 0

Multiply the elements and add the result to the dot product variable

APPEND the dot product to the product costs list

1. PRINT the cost of each product

**Explanation:** This pseudocode calculates the model vector X by multiplying the pseudo-inverse of matrix A with the payment vector C. Then, it estimates the cost of each product by taking the dot product of each row in matrix A with the model vector X. The simplified approach uses nested loops to iterate through the rows and elements of the matrices, simplifying the calculation process for understanding and implementation**.**

**A-3**

**Pseudocode:**

1. Import the numpy library as np

2. Define purchase\_data as a numpy array

3. FUNCTION euclidean\_distance

Calculate the square of the difference between each corresponding element of x1 and x2

Sum the squared differences

Take the square root of the sum

4. FUNCTION knn\_predict:

X\_train, y\_train, x\_test, k

INITIALIZE list distances

FOR x\_train in X\_train:

CALCULATE euclidean\_distance function

APPEND the tuple (distance, y\_train[i]) to the distances list

SORT the distances list based on the distances and select the first k elements

Count the occurrences of each label in the selected distances

Return the label with the highest count as the predicted class for x\_test

5. Create binary labels y\_train where 1 represents payments above 200 rupees and 0 represents payments below or equal to 200 rupees

6. Define the training data X\_train as all columns except the last one in the purchase\_data array

7. PRINT Predicted Classes

8. Iterate over each customer payment in X\_train and its index i:

Predict the class for the customer's payment using the knn\_predict function

If the predicted class is 1, print "Customer {i+1}: RICH"

Else, print "Customer {i+1}: POOR"

9. END

**Explanation:** This program imports the NumPy library and loads payment and purchase data for 10 customers. Defines functions to calculate Euclidean distance and perform K-nearest neighbours classification and predicts whether each customer is "RICH" or "POOR" based on their payment data using the KNN algorithm. The program defines functions for calculating Euclidean distance and performing K-nearest neighbours (KNN) classification. Using the loaded data, it creates binary labels where payments above 200 rupees are labelled as 1 and others as 0. Finally, it predicts the class for each customer payment based on the KNN algorithm and prints whether each customer is classified as "RICH" or "POOR" based on their payment data.

**A-4:**

**Pseudocode:**

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import statistics

path = "C:\\Users\\nossa\\Downloads\\IRCTC Stock Price.xlsx"

df = pd.read\_excel(path)

mean\_of\_price = statistics.mean(df['Price'])

variance\_of\_price = statistics.variance(df['Price'])

print("The Mean of the column Price data is:", mean\_of\_price)

df['Date'] = pd.to\_datetime(df['Date'])

wednesday = df[df['Day'] == 'Wed']['Price']

sample\_mean\_wednesday = statistics.mean(wednesday)

print("The Sample mean of Wednesday for the column prices is:", sample\_mean\_wednesday)

april = df[df['Month'] == 'Apr']['Price']

sample\_mean\_april = statistics.mean(april)

print("The Sample mean of April for the column prices is:", sample\_mean\_april)

chg\_percentages = df['Chg%'].apply(lambda x: float(x.strip('%')) if isinstance(x, str) else x)

loss\_probability = len(chg\_percentages[chg\_percentages < 0]) / len(chg\_percentages)

print("The Probability of making a loss over the stock is:", loss\_probability)

profit\_probability = len(wednesday[wednesday.diff() > 0]) / len(wednesday)

print("The Probability of making a profit on Wednesday is:", profit\_probability)

conditional\_profit\_probability = len(wednesday[wednesday.diff() > 0]) / len(wednesday)

print("The Conditional probability of making profit on Wednesday is:", conditional\_profit\_probability)

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

sns.scatterplot(data=df, x='Day', y='Chg%')

plt.title('The Scatter plot of Chg% data against the day of the week')

plt.xlabel('Day of the week')

plt.ylabel('Chg%')

plt.show()

**Explanation:** This program analyzes stock price data from an Excel file for IRCTC. It imports necessary libraries like Pandas, Matplotlib, Seaborn, and Statistics. It reads the Excel file into a DataFrame, calculates the mean and variance of the 'Price' column, and prints the results. It then converts the 'Date' column to datetime format and filters the DataFrame to select 'Price' data for Wednesdays and April separately, calculating and printing their respective sample means. The program also calculates the probability of making a loss over the stock, the probability of making a profit on Wednesdays, and the conditional probability of making a profit on Wednesdays given that today is Wednesday, printing each result. Finally, it creates a scatter plot of 'Chg%' data against the day of the week using Seaborn and Matplotlib and displays it.