R.V.RaghuNandan

BL.EN.U4AIE23147

LAB-2 ASSIGNMENT

import numpy as np

import pandas as pd

import statistics

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler, StandardScaler

data\_file = "Lab Session Data.xlsx"

Question A1: Please refer to the “Purchase Data” worksheet of Lab Session Data.xlsx. Please load the data and segregate them into 2 matrices A & C (following the nomenclature of AX = C). Do the following activities. • What is the dimensionality of the vector space for this data? • How many vectors exist in this vector space? • What is the rank of Matrix A? • Using Pseudo-Inverse find the cost of each product available for sale. (Suggestion: If you use Python, you can use numpy.linalg.pinv() function to get a pseudo-inverse.)

df\_purchase = pd.read\_excel(data\_file, sheet\_name="Purchase Data")

A = df\_purchase.iloc[:, :-1].values

C = df\_purchase.iloc[:, -1].values

vector\_space\_dim = A.shape[1]

num\_vectors = A.shape[0]

rank\_A = np.linalg.matrix\_rank(A)

pseudo\_inv\_A = np.linalg.pinv(A)

cost\_vector = np.dot(pseudo\_inv\_A, C)

Question A2: . Use the Pseudo-inverse to calculate the model vector X for predicting the cost of the products available with the vendor.

Soln:

X = np.dot(pseudo\_inv\_A, C)

Question A3: Mark all customers (in “Purchase Data” table) with payments above Rs. 200 as RICH and others as POOR. Develop a classifier model to categorize customers into RICH or POOR class based on purchase behavior.

Soln:

df\_purchase["Customer\_Type"] = ["RICH" if x > 200 else "POOR" for x in C]

Question A4: Please refer to the data present in “IRCTC Stock Price” data sheet of the above excel file. Do the following after loading the data to your programming platform.

Soln:

df\_stock = pd.read\_excel(data\_file, sheet\_name="IRCTC Stock Price")

price\_data = df\_stock["Price Column D"]

mean\_price = statistics.mean(price\_data)

variance\_price = statistics.variance(price\_data)

wednesday\_data = df\_stock[df\_stock["Day"] == "Wednesday"]["Price Column D"]

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

df\_stock["Loss\_Prob"] = df\_stock["Chg% Column I"].apply(lambda x: x < 0)

prob\_loss = df\_stock["Loss\_Prob"].mean()

plt.figure(figsize=(8, 6))

sns.scatterplot(x=df\_stock["Day"], y=df\_stock["Chg% Column I"])

plt.xlabel("Day of the Week")

plt.ylabel("Chg%")

plt.title("Stock Change Percentage vs Day of the Week")

plt.show()

Question A5: Data Exploration: Load the data available in “thyroid0387\_UCI” worksheet. Perform the following tasks:

• Study each attribute and associated values present. Identify the datatype (nominal etc.) for the attribute.

• For categorical attributes, identify the encoding scheme to be employed. (Guidance: employ label encoding for ordinal variables while One-Hot encoding may be employed for nominal variables).

• Study the data range for numeric variables.

• Study the presence of missing values in each attribute.

• Study presence of outliers in data.

• For numeric variables, calculate the mean and variance (or standard deviation).

Soln:

df\_thyroid = pd.read\_excel(data\_file, sheet\_name="thyroid0387\_UCI")

attribute\_types = df\_thyroid.dtypes

missing\_values = df\_thyroid.isnull().sum()

outlier\_detection = df\_thyroid.describe()

label\_enc = LabelEncoder()

one\_hot\_enc = OneHotEncoder()

for col in df\_thyroid.select\_dtypes(include=["object"]).columns:

df\_thyroid[col] = label\_enc.fit\_transform(df\_thyroid[col])

Question A6: Data Imputation: employ appropriate central tendencies to fill the missing values in the data variables. Employ following guidance.

• Mean may be used when the attribute is numeric with no outliers

• Median may be employed for attributes which are numeric and contain outliers

• Mode may be employed for categorical attributes

Soln:

df\_thyroid.fillna(df\_thyroid.mean(), inplace=True)

Question A7: Data Normalization / Scaling: from the data study, identify the attributes which may need normalization. Employ appropriate normalization techniques to create normalized set of data.

Soln:

scaler = MinMaxScaler()

df\_thyroid\_normalized = scaler.fit\_transform(df\_thyroid)

Question A8: Similarity Measure: Take the first 2 observation vectors from the dataset. Consider only the attributes (direct or derived) with binary values for these vectors (ignore other attributes). Calculate the Jaccard Coefficient (JC) and Simple Matching Coefficient (SMC) between the document vectors. Use first vector for each document for this. Compare the values for JC and SMC and judge the appropriateness of each of them.

JC = (f11) / (f01+ f10+ f11)

SMC = (f11 + f00) / (f00 + f01 + f10 + f11)

f11= number of attributes where the attribute carries value of 1 in both the vectors.

Soln:

vec1, vec2 = df\_thyroid.iloc[0, :], df\_thyroid.iloc[1, :]

f11 = np.sum((vec1 == 1) & (vec2 == 1))

f00 = np.sum((vec1 == 0) & (vec2 == 0))

f01 = np.sum((vec1 == 0) & (vec2 == 1))

f10 = np.sum((vec1 == 1) & (vec2 == 0))

JC = f11 / (f01 + f10 + f11)

SMC = (f11 + f00) / (f00 + f01 + f10 + f11)

Question A9: Cosine Similarity Measure: Now take the complete vectors for these two observations (including all the attributes). Calculate the Cosine similarity between the documents by using the second feature vector for each document.

Soln:

cos\_sim = np.dot(vec1, vec2) / (np.linalg.norm(vec1) \* np.linalg.norm(vec2))

Question A10:

Heatmap Plot: Consider the first 20 observation vectors. Calculate the JC, SMC and COS between the pairs of vectors for these 20 vectors. Employ similar strategies for coefficient calculation as in A4 & A5. Employ a heatmap plot to visualize the similarities. Suggestion to Python users

→ import seaborn as sns

sns.heatmap(data, annot = True)

Soln:

similarity\_matrix = np.zeros((20, 20))

for i in range(20):

for j in range(20):

vec1, vec2 = df\_thyroid.iloc[i, :], df\_thyroid.iloc[j, :]

similarity\_matrix[i, j] = np.dot(vec1, vec2) / (np.linalg.norm(vec1) \* np.linalg.norm(vec2))

plt.figure(figsize=(10, 8))

sns.heatmap(similarity\_matrix, annot=True, cmap="coolwarm")

plt.title("Heatmap of Similarity Measures")

plt.show()