Data Mining
Exercise 3

Name: Chethan Kashyap Bangalore Muralidhara

Date: 16/11/2023

Completed Exercises: 1,2,3,4,5,6(All)

Solutions:

Question 1: The exercises are done in python and the code fil is attached as well.

Question 1

In [1]:
```
import pandas as pd

df = pd.read_excel("D:/MS/Data Mining/3/bloodp.xlsx")

#Replace Zero and Missing Values with Mean:
df['sbp'].replace(0, df['sbp'].mean(), inplace=True)
df['dbp'].replace(0, df['dbp'].mean(), inplace=True)
df['sbp'].fillna(df['sbp'].mean(), inplace=True)
df['dbp'].fillna(df['dbp'].mean(), inplace=True)
```

In [2]:
```
df.head()
```

Out[2]:

	sbp	dbp
0	132.426204	75.140528
1	132.426204	75.140528
2	132.426204	75.140528
3	132.426204	75.140528
4	132.426204	75.140528

In [3]:
```
#Correct Erroneous Values:
df['sbp'] = df['sbp'].apply(lambda x: x * 10 if x < 80 else x)
df['dbp'] = df['dbp'].apply(lambda x: x * 10 if x < 40 else x)
```

In [5]:
```
#Remove Impossible Values:
df = df[(df['sbp'] <= 300) & (df['dbp'] <= 160)]
```

In [6]:
```
#Save the Cleaned Data:
df.to_excel("D:/MS/Data Mining/3/cleaned_bloodp.xlsx", index=False)
```

Question 2:

Question 2

In [7]:
```
import numpy as np

# Assuming df is your DataFrame from Task 1
O = np.column_stack((np.ones_like(df['sbp']), df['sbp'], df['dbp']))
```

In [8]:
```
y = df['dbp'].values
X = np.column_stack((np.ones_like(df['sbp']), df['sbp']))
```

In [9]:
```
# Using the formula b = (X^T * X)^(-1) * X^T * y
coefficients = np.linalg.inv(X.T @ X) @ X.T @ y
```

In [10]:
```
print(coefficients)
```

```
[38.43171163  0.32123727]
```

Question 3:

Question 3

In [11]:
```
# Reference document
Fo = np.array([15, 7, 6, 11, 4])
Nw = 500
normalized_reference = Fo / Nw

# Document 1
Fo1 = np.array([1, 4, 3, 3, 6])
Nw1 = 200
normalized_doc1 = Fo1 / Nw1

# Document 2
Fo2 = np.array([20, 1, 5, 16, 9])
Nw2 = 210
normalized_doc2 = Fo2 / Nw2

# Calculate dot products
dot_product_ref_doc1 = np.dot(normalized_reference, normalized_doc1)
dot_product_ref_doc2 = np.dot(normalized_reference, normalized_doc2)

# Calculate cosine distances
cosine_distance_doc1 = 1 - dot_product_ref_doc1 / (np.linalg.norm(normalized_reference) * np.linalg.norm(normalized_doc1))
cosine_distance_doc2 = 1 - dot_product_ref_doc2 / (np.linalg.norm(normalized_reference) * np.linalg.norm(normalized_doc2))

print("Cosine Distance between Reference Document and Document 1:", cosine_distance_doc1)
print("Cosine Distance between Reference Document and Document 2:", cosine_distance_doc2)
```

```
Cosine Distance between Reference Document and Document 1: 0.33763241404943267
Cosine Distance between Reference Document and Document 2: 0.05993839763966191
```

Question 4:

Question 4

In [19]:
```python
file_path = 'D:/MS/Data Mining/3/tcpc.csv'

# Load the CSV file into a DataFrame
df = pd.read_csv(file_path)
# Print the columns to identify the correct column name
#print("Columns:", df.columns)


# Drop the "DateTime" column
df = df.drop(columns=['DateTime'])

# Handle missing values
df.fillna(df.mean(), inplace=True)

# Convert non-numeric values to numeric
df = df.apply(pd.to_numeric, errors='coerce')

# Binarize the variables based on mean
mean_values = df.mean()
binary_df = (df < mean_values).astype(int)

# Define the sample
s = [0, 1, 0, 0, 0, 0, 0, 0]

# Calculate Hamming distance
hamming_distances = (binary_df != s).sum(axis=1)

# Find the index of the nearest neighbor
nearest_neighbor_index = hamming_distances.idxmin()

# Display the nearest neighbor
nearest_neighbor = df.loc[nearest_neighbor_index]
print("Nearest Neighbor:\n", nearest_neighbor)
```

```
Nearest Neighbor:
 Temperature                  25.27000
Humidity                     23.52000
Wind Speed                    4.92000
general diffuse flows       207.30000
diffuse flows               230.70000
Zone 1 Power Consumption    34169.70492
Zone 2  Power Consumption   21299.07121
Zone 3  Power Consumption   18037.89474
Name: 17968, dtype: float64
```

Question 5:

Question 5

In [23]:
```
from sklearn.metrics import jaccard_score

# Calculate Jaccard similarity matrix
jaccard_matrix = pd.DataFrame(index=binary_df.columns, columns=binary_df.columns)

for col1 in binary_df.columns:
    for col2 in binary_df.columns:
        jaccard_matrix.loc[col1, col2] = jaccard_score(binary_df[col1], binary_df[col2])

# Convert the Jaccard matrix to numeric format
jaccard_matrix_numeric = pd.to_numeric(jaccard_matrix.stack())

# Find the variable with the highest average correlation
max_correlation_variable = jaccard_matrix_numeric.idxmax()
max_correlation_value = jaccard_matrix_numeric.max()

print("\nVariable with the Highest Average Correlation:")
print(f"Variable: {max_correlation_variable[1]}")
print(f"Average Correlation: {max_correlation_value}")
```

```
Variable with the Highest Average Correlation:
Variable: Temperature
Average Correlation: 1.0
```

Question 6:

The triangle inequality states that for any three vectors a, b, and c, the following must hold:

$$d(a, c) \leq d(a, b) + d(b, c)$$

where d is the distance (or dissimilarity) measure.

Now, let's consider three vectors a, b, and c in a vector space:

$$a = [1, 0]$$
$$b = [0, 1]$$
$$c = [-1, 0]$$

Let's calculate the cosine similarities and distances:

$$\text{cosine similarity}(a, b) = \frac{a \cdot b}{\|a\| \cdot \|b\|} = 0$$
$$\text{cosine similarity}(b, c) = \frac{b \cdot c}{\|b\| \cdot \|c\|} = 0$$
$$\text{cosine similarity}(a, c) = \frac{a \cdot c}{\|a\| \cdot \|c\|} = 1$$

Now, let's check the triangle inequality:

$$\text{cosine similarity}(a, c) \leq \text{cosine similarity}(a, b) + \text{cosine similarity}(b, c)$$

$$1 \leq 0 + 0$$

This inequality does not hold, violating the triangle inequality property. Therefore, cosine similarity is not a metric.