ml-lab-02-aie23114

February 16, 2025

[2]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Lab Session Data.xlsx to Lab Session Data.xlsx
```

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

[12]:
```
import pandas as pd
import numpy as np
df = pd.read_excel(r'/content/Lab Session Data.xlsx', sheet_name="Purchase␣
↪data")
df = df.iloc[:, :5]
print(df)

A = df.iloc[:10, 1:4].values
print(f"A:{A}")
C = df.iloc[:10, 4].values.reshape(-1, 1)
print(f"C:{C}")

print(f"Dimensionality of the vector space:{df.shape}")

print(f"Number of vectors:{df.shape[0]}")

rank_A = np.linalg.matrix_rank(A)
print(f"Rank of matrix A: {rank_A}")

A_pinv = np.linalg.pinv(A)
print(f"pinv of A is {A_pinv}")
```

```python
X = np.dot(A_pinv, C)
print(f"model vector X is {X}")
print(f" Cost of each candy is Rs. {X[0]}")
print(f" Cost of each mango is Rs. {X[1]}")
print(f" Cost of each milk packet is Rs. {X[2]}")

category = []
for payment in df['Payment (Rs)']:
  if payment>=200 :
    category.append('Rich')
  else:
    category.append('Poor')
df['category']= category
print(df)
```

```
  Customer  Candies (#)  Mangoes (Kg)  Milk Packets (#)  Payment (Rs)
0      C_1           20             6                 2           386
1      C_2           16             3                 6           289
2      C_3           27             6                 2           393
3      C_4           19             1                 2           110
4      C_5           24             4                 2           280
5      C_6           22             1                 5           167
6      C_7           15             4                 2           271
7      C_8           18             4                 2           274
8      C_9           21             1                 4           148
9     C_10           16             2                 4           198
A:[[20  6  2]
 [16  3  6]
 [27  6  2]
 [19  1  2]
 [24  4  2]
 [22  1  5]
 [15  4  2]
 [18  4  2]
 [21  1  4]
 [16  2  4]]
C:[[386]
 [289]
 [393]
 [110]
 [280]
 [167]
 [271]
 [274]
 [148]
 [198]]
```

```
Dimensionality of the vector space:(10, 5)
Number of vectors:10
Rank of matrix A: 3
pinv of A is [[-0.01008596 -0.03124505  0.01013951  0.0290728   0.0182907
0.01161794
  -0.00771348  0.00095458  0.01743623 -0.00542016]
 [ 0.09059668  0.07263726  0.03172933 -0.09071908 -0.01893196 -0.06926996
   0.05675464  0.03152577 -0.07641966  0.00357352]
 [ 0.00299878  0.15874243 -0.05795468 -0.06609024 -0.06295043  0.03348017
   0.01541831 -0.01070461  0.00029003  0.05938755]]
model vector X is [[ 1.]
 [55.]
 [18.]]
 Cost of each candy is Rs. [1.]
 Cost of each mango is Rs. [55.]
 Cost of each milk packet is Rs. [18.]
  Customer  Candies (#)  Mangoes (Kg)  Milk Packets (#)  Payment (Rs) category
0      C_1           20             6                 2           386     Rich
1      C_2           16             3                 6           289     Rich
2      C_3           27             6                 2           393     Rich
3      C_4           19             1                 2           110     Poor
4      C_5           24             4                 2           280     Rich
5      C_6           22             1                 5           167     Poor
6      C_7           15             4                 2           271     Rich
7      C_8           18             4                 2           274     Rich
8      C_9           21             1                 4           148     Poor
9     C_10           16             2                 4           198     Poor
```

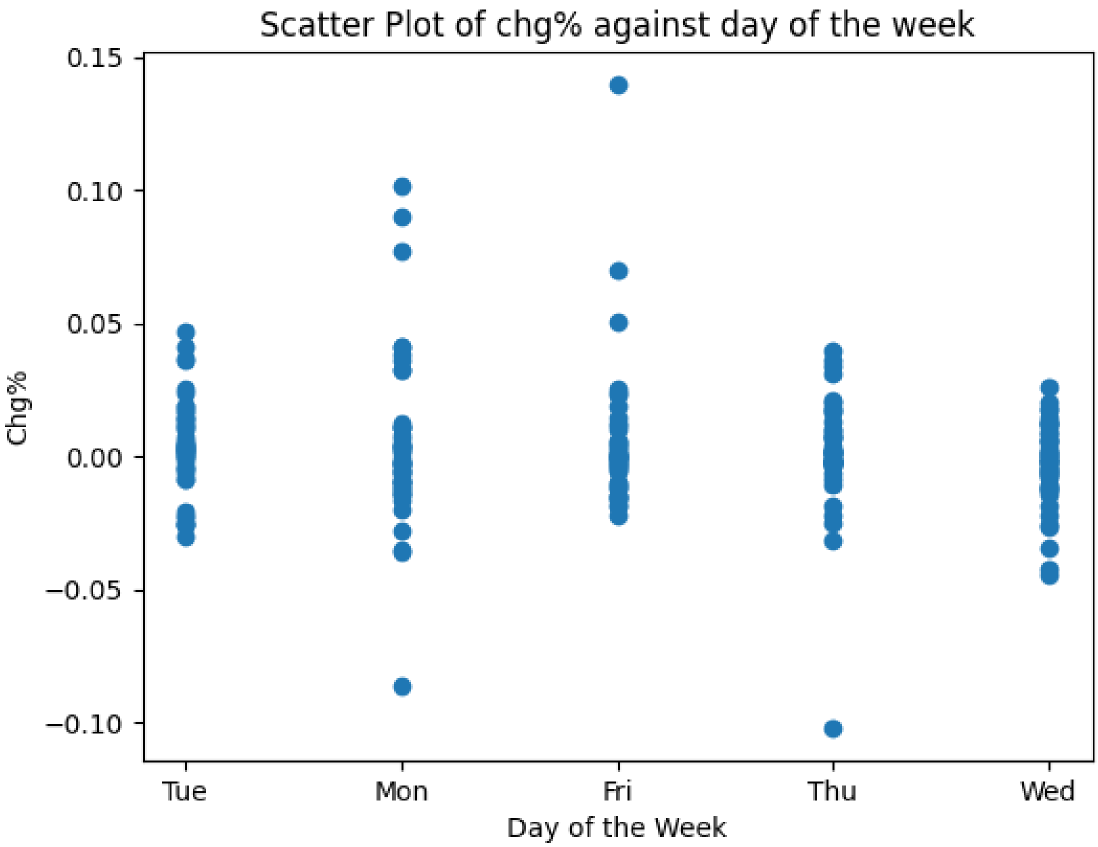
A4. Please refer to the data present in "IRCTC Stock Price" data sheet of the above excel file. Do the following after loading the data to your programming platform. • Calculate the mean and variance of the Price data present in column D.
(Suggestion: if you use Python, you may use statistics.mean() & statistics.variance() methods). • Select the price data for all Wednesdays and calculate the sample mean. Compare the mean with the population mean and note your observations. • Select the price data for the month of Apr and calculate the sample mean. Compare the mean with the population mean and note your observations. • From the Chg% (available in column I) find the probability of making a loss over the stock. (Suggestion: use lambda function to find negative values) • Calculate the probability of making a profit on Wednesday. • Calculate the conditional probability of making profit, given that today is Wednesday. • Make a scatter plot of Chg% data against the day of the week

[13]:
```python
import pandas as pd
import numpy as np
import statistics
import matplotlib.pyplot as plt

df = pd.read_excel(r'/content/Lab Session Data.xlsx', sheet_name="IRCTC Stock␣
 ↪Price")
df = df.iloc[:, :9]
```

```
print(df)

mean = statistics.mean(df["Price"])
var = statistics.variance(df["Price"])
print(f"Price Mean: {mean}")
print(f"Price Variance: {var}")

wed_mean = df.loc[df['Day'] == 'Wed', 'Price'].mean()
print(f"Mean Price on Wednesdays: {wed_mean}")
print("Observation: Sales at IRCTC are lower on Wednesdays compared to the␣
↪overall average.")

apr_mean = df.loc[df['Month'] == 'Apr', 'Price'].mean()
print(f"Mean Price in April: {apr_mean}")
print("Observation: Sales at IRCTC are higher in April compared to the overall␣
↪average.")

loss_probability = sum(df['Chg%'] < 0) / len(df)
print(f"Probability of making a loss: {loss_probability}")

wed_df = df[df['Day'] == 'Wed']
wed_profit = sum(wed_df['Chg%'] > 0) / len(wed_df)
print(f"Probability of making a profit on Wednesday: {wed_profit}")

profit_given_wed = (sum((df['Day'] == 'Wed') & (df['Chg%'] > 0)) /␣
↪sum(df['Day'] == 'Wed'))
print(f"Conditional Probability (Profit | Wednesday): {profit_given_wed}")

days = df['Day']
chg = df['Chg%']
plt.scatter(days,chg)
plt.xlabel("Day of the Week")
plt.ylabel("Chg%")
plt.title("Scatter Plot of chg% against day of the week")
plt.show()
```

```
             Date Month  Day    Price     Open     High      Low   Volume  \
0    Jun 29, 2021   Jun  Tue  2081.85  2092.00  2126.90  2065.05    1.67M
1    Jun 28, 2021   Jun  Mon  2077.75  2084.00  2112.45  2068.40  707.73K
2    Jun 25, 2021   Jun  Fri  2068.85  2084.35  2088.50  2053.10  475.82K
3    Jun 24, 2021   Jun  Thu  2072.95  2098.00  2098.00  2066.00  541.51K
4    Jun 23, 2021   Jun  Wed  2078.25  2102.00  2111.40  2072.00  809.62K
..            …    …   …      …       …       …       …       …
244  Jul 07, 2020   Jul  Tue  1397.40  1410.00  1411.00  1390.05  480.21K
245  Jul 06, 2020   Jul  Mon  1400.75  1405.50  1415.50  1394.00  614.93K
246  Jul 03, 2020   Jul  Fri  1405.10  1415.00  1425.00  1398.00  599.49K
247  Jul 02, 2020   Jul  Thu  1412.35  1440.00  1467.80  1395.30    2.16M
```

```
248  Jul 01, 2020   Jul  Wed  1363.05  1363.65  1377.00  1356.00  383.00K

       Chg%
0    0.0020
1    0.0043
2   -0.0020
3   -0.0026
4   -0.0023
..      …
244 -0.0024
245 -0.0031
246 -0.0051
247  0.0362
248  0.0032

[249 rows x 9 columns]
Price Mean: 1560.663453815261
Price Variance: 58732.365352539186
Mean Price on Wednesdays: 1550.7060000000001
Observation: Sales at IRCTC are lower on Wednesdays compared to the overall
average.
Mean Price in April: 1698.9526315789474
Observation: Sales at IRCTC are higher in April compared to the overall average.
Probability of making a loss: 0.4979919678714859
Probability of making a profit on Wednesday: 0.42
Conditional Probability (Profit | Wednesday): 0.42
```



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). 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 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. 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. 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. 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.

[20]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics.pairwise import cosine_similarity

df = pd.read_excel(r"/content/Lab Session Data.xlsx",␣
↪sheet_name="thyroid0387_UCI")
# Replace '?' with NaN and infer proper datatypes
df.replace('?', np.nan, inplace=True)
df = df.infer_objects()

data_types = {}
# Identify and classify data types: Nominal, Ordinal, Ratio, Interval
for col in df.columns:
    if df[col].dtype == 'object':
        if df[col].nunique() <= 10:
            data_types[col] = 'Ordinal'
        else:
            data_types[col] = 'Nominal'
    elif df[col].dtype in ['int64', 'float64']:
        if df[col].min() >= 0:
            data_types[col] = 'Ratio'
        else:
            data_types[col] = 'Interval'
print("Attribute Data Types:")
for col, dtype in data_types.items():
    print(f"{col}: {dtype}")

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Encoding Scheme
print("Encoding Scheme:")
for col in categorical_cols:
    if data_types[col] == 'Ordinal':
        print(f"{col}: Label Encoding (Ordinal)")
    else:
        print(f"{col}: One-Hot Encoding (Nominal)")

numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
```

```
print("Range for Numeric Variables:")
for col in numerical_cols:
    min_value = df[col].min()
    max_value = df[col].max()
    data_range = max_value - min_value
    print(f"{col}: Min = {min_value}, Max = {max_value}, Range = {data_range}")
print("Mean and Standard Deviation (or Variance) for Numeric Variables:")
for col in numerical_cols:
    mean_value = df[col].mean()
    std_dev = df[col].std()
    variance = std_dev ** 2
    print(f"{col}: Mean = {mean_value:.2f}, Standard Deviation = {std_dev}, 
 ↪Variance = {variance}")

# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:", missing_values)

# Apply Label Encoding to categorical features
for col in categorical_cols:
    df[col] = df[col].astype(str)
    df[col] = LabelEncoder().fit_transform(df[col])

# Identify outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
outliers = ((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum()
print("Outliers per column:", outliers)

# Data Imputation
for col in df.columns:
    if df[col].dtype in ['float64', 'int64']:
        if outliers[col] > 0:
            df[col] = df[col].fillna(df[col].median())
        else:
            df[col] = df[col].fillna(df[col].mean())
    else:
        df[col] = df[col].fillna(df[col].mode()[0])
print("Missing values after imputation:", df.isnull().sum())

#Data Normalization
scaler = MinMaxScaler()
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
print("Normalized Data Sample:", df.head())
```

```
# jc,smc
vector1 = df.iloc[0, :].values
vector2 = df.iloc[1, :].values
f11 = np.sum((vector1 == 1) & (vector2 == 1))
f00 = np.sum((vector1 == 0) & (vector2 == 0))
f10 = np.sum((vector1 == 1) & (vector2 == 0))
f01 = np.sum((vector1 == 0) & (vector2 == 1))
if (f01 + f10 + f11) != 0:
    jc = f11 / (f01 + f10 + f11)
else:
    jc = 0
if (f00 + f01 + f10 + f11) != 0:
    smc = (f11 + f00) / (f00 + f01 + f10 + f11)
else:
    smc = 0
print(f"Jaccard Coefficient: {jc}, SMC: {smc}")

# Cosine Similarity
vector1 = df.iloc[0, :].values.reshape(1, -1)
vector2 = df.iloc[1, :].values.reshape(1, -1)
cosine_sim = cosine_similarity(vector1, vector2)[0][0]
print(f"Cosine Similarity: {cosine_sim}")

# Heatmap plot
df_subset = df.iloc[:20, :]
similarity_matrix = np.zeros((20, 20))
for i in range(20):
    for j in range(20):
        if i != j:
            similarity_matrix[i, j] = np.linalg.norm(df_subset.iloc[i] - 
 ↪df_subset.iloc[j])
sns.heatmap(similarity_matrix)
plt.show()
```

```
<ipython-input-20-cadb9acfc739>:10: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df.replace('?', np.nan, inplace=True)
```

```
Attribute Data Types:
Record ID: Ratio
age: Ratio
sex: Ordinal
on thyroxine: Ordinal
query on thyroxine: Ordinal
on antithyroid medication: Ordinal
sick: Ordinal
```

```
pregnant: Ordinal
thyroid surgery: Ordinal
I131 treatment: Ordinal
query hypothyroid: Ordinal
query hyperthyroid: Ordinal
lithium: Ordinal
goitre: Ordinal
tumor: Ordinal
hypopituitary: Ordinal
psych: Ordinal
TSH measured: Ordinal
TSH: Ratio
T3 measured: Ordinal
T3: Ratio
TT4 measured: Ordinal
TT4: Ratio
T4U measured: Ordinal
T4U: Ratio
FTI measured: Ordinal
FTI: Ratio
TBG measured: Ordinal
TBG: Ratio
referral source: Ordinal
Condition: Nominal
Encoding Scheme:
sex: Label Encoding (Ordinal)
on thyroxine: Label Encoding (Ordinal)
query on thyroxine: Label Encoding (Ordinal)
on antithyroid medication: Label Encoding (Ordinal)
sick: Label Encoding (Ordinal)
pregnant: Label Encoding (Ordinal)
thyroid surgery: Label Encoding (Ordinal)
I131 treatment: Label Encoding (Ordinal)
query hypothyroid: Label Encoding (Ordinal)
query hyperthyroid: Label Encoding (Ordinal)
lithium: Label Encoding (Ordinal)
goitre: Label Encoding (Ordinal)
tumor: Label Encoding (Ordinal)
hypopituitary: Label Encoding (Ordinal)
psych: Label Encoding (Ordinal)
TSH measured: Label Encoding (Ordinal)
T3 measured: Label Encoding (Ordinal)
TT4 measured: Label Encoding (Ordinal)
T4U measured: Label Encoding (Ordinal)
FTI measured: Label Encoding (Ordinal)
TBG measured: Label Encoding (Ordinal)
referral source: Label Encoding (Ordinal)
Condition: One-Hot Encoding (Nominal)
```

```
Range for Numeric Variables:
Record ID: Min = 840801013, Max = 870119035, Range = 29318022
age: Min = 1, Max = 65526, Range = 65525
TSH: Min = 0.005, Max = 530.0, Range = 529.995
T3: Min = 0.05, Max = 18.0, Range = 17.95
TT4: Min = 2.0, Max = 600.0, Range = 598.0
T4U: Min = 0.17, Max = 2.33, Range = 2.16
FTI: Min = 1.4, Max = 881.0, Range = 879.6
TBG: Min = 0.1, Max = 200.0, Range = 199.9
Mean and Standard Deviation (or Variance) for Numeric Variables:
Record ID: Mean = 852947346.61, Standard Deviation = 7581968.780346589, Variance
= 57486250586150.34
age: Mean = 73.56, Standard Deviation = 1183.9767180444667, Variance =
1401800.8688713466
TSH: Mean = 5.22, Standard Deviation = 24.184006144749777, Variance =
584.8661532092949
T3: Mean = 1.97, Standard Deviation = 0.8875788237425206, Variance =
0.7877961683561565
TT4: Mean = 108.70, Standard Deviation = 37.52267036706598, Variance =
1407.9507914754913
T4U: Mean = 0.98, Standard Deviation = 0.2003604411805482, Variance =
0.04014430639006391
FTI: Mean = 113.64, Standard Deviation = 41.551649606979, Variance =
1726.5395850611578
TBG: Mean = 29.87, Standard Deviation = 21.080503860189545, Variance =
444.3876429994663
Missing Values: Record ID                       0
age                             0
sex                           307
on thyroxine                    0
query on thyroxine              0
on antithyroid medication       0
sick                            0
pregnant                        0
thyroid surgery                 0
I131 treatment                  0
query hypothyroid               0
query hyperthyroid              0
lithium                         0
goitre                          0
tumor                           0
hypopituitary                   0
psych                           0
TSH measured                    0
TSH                           842
T3 measured                     0
T3                           2604
TT4 measured                    0
```

```
TT4                            442
T4U measured                     0
T4U                            809
FTI measured                     0
FTI                            802
TBG measured                     0
TBG                           8823
referral source                  0
Condition                        0
dtype: int64
Outliers per column: Record ID                        0
age                              4
sex                              0
on thyroxine                  1240
query on thyroxine             153
on antithyroid medication      116
sick                           344
pregnant                       107
thyroid surgery                134
I131 treatment                 169
query hypothyroid              630
query hyperthyroid             651
lithium                         93
goitre                          84
tumor                          241
hypopituitary                    2
psych                          418
TSH measured                   842
TSH                            884
T3 measured                      0
T3                             360
TT4 measured                   442
TT4                            422
T4U measured                   809
T4U                            420
FTI measured                   802
FTI                            501
TBG measured                   349
TBG                             29
referral source                  0
Condition                     2401
dtype: int64
Missing values after imputation: Record ID                        0
age                         0
sex                         0
on thyroxine                0
query on thyroxine          0
on antithyroid medication   0
```

```
sick                        0
pregnant                    0
thyroid surgery             0
I131 treatment              0
query hypothyroid           0
query hyperthyroid          0
lithium                     0
goitre                      0
tumor                       0
hypopituitary               0
psych                       0
TSH measured                0
TSH                         0
T3 measured                 0
T3                          0
TT4 measured                0
TT4                         0
T4U measured                0
T4U                         0
FTI measured                0
FTI                         0
TBG measured                0
TBG                         0
referral source             0
Condition                   0
dtype: int64
Normalized Data Sample:       Record ID       age  sex  on thyroxine  query on
thyroxine  \
0  0.000000e+00  0.000427  0.0           0.0                  0.0
1  3.410871e-08  0.000427  0.0           0.0                  0.0
2  9.891527e-07  0.000610  0.0           0.0                  0.0
3  6.934301e-05  0.000534  0.0           0.0                  0.0
4  6.937712e-05  0.000473  0.0           0.0                  0.0

   on antithyroid medication  sick  pregnant  thyroid surgery  I131 treatment  \
0                        0.0   0.0       0.0              0.0             0.0
1                        0.0   0.0       0.0              0.0             0.0
2                        0.0   0.0       0.0              0.0             0.0
3                        0.0   0.0       0.0              0.0             0.0
4                        0.0   0.0       0.0              0.0             0.0

   …  TT4 measured       TT4  T4U measured       T4U  FTI measured  \
0  …           0.0  0.170569           0.0  0.365741           0.0
1  …           1.0  0.210702           0.0  0.365741           0.0
2  …           0.0  0.170569           0.0  0.365741           0.0
3  …           0.0  0.170569           0.0  0.365741           0.0
4  …           0.0  0.170569           0.0  0.365741           0.0
```

```
        FTI  TBG measured       TBG  referral source  Condition
0  0.122328           0.0  0.129565              1.0   0.806452
1  0.122328           0.0  0.129565              1.0   0.806452
2  0.122328           1.0  0.054527              1.0   0.806452
3  0.122328           1.0  0.129565              1.0   0.806452
4  0.122328           1.0  0.179590              1.0   1.000000

[5 rows x 31 columns]
Jaccard Coefficient: 0.4, SMC: 0.8636363636363636
Cosine Similarity: 0.6605204689109685
```