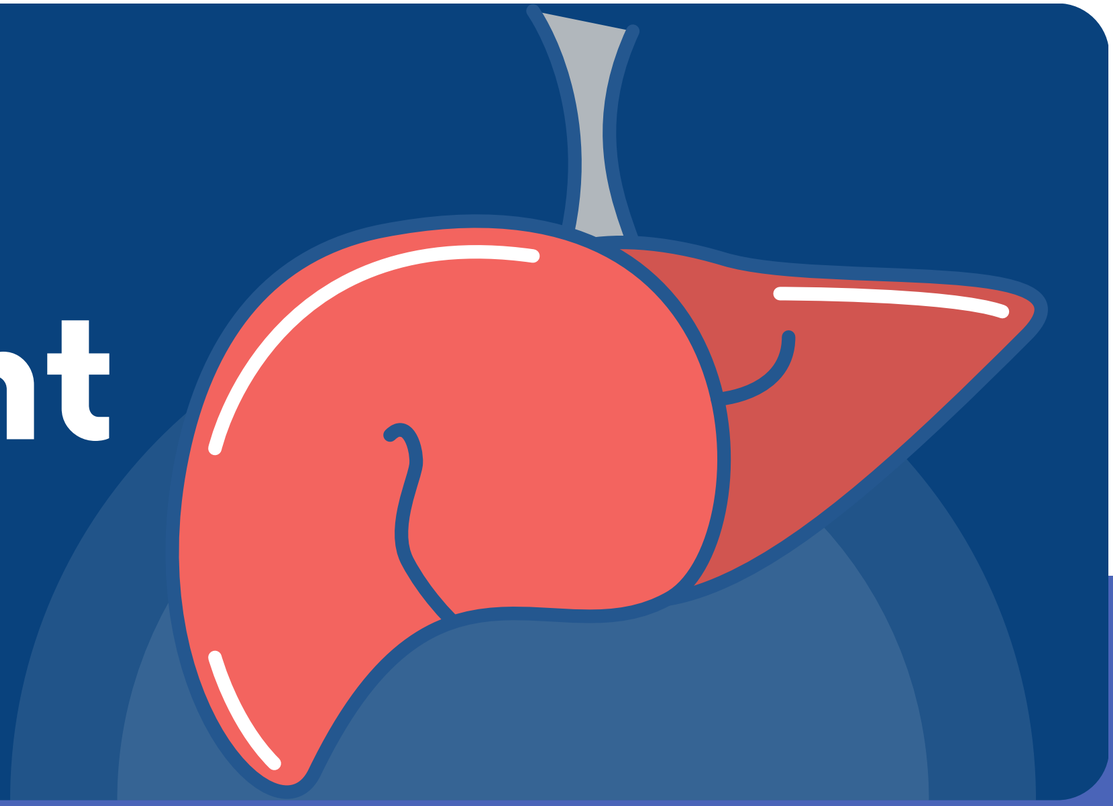
DATA MINING (IT326)

Liver patient



RAGHAD FARES

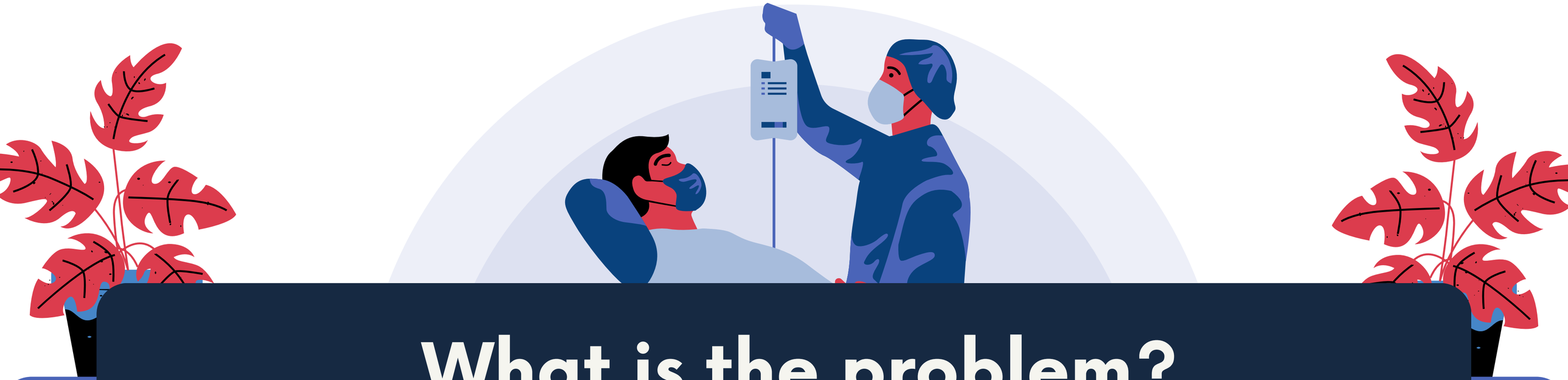
SEREEN AL-HMOUD

LULUH ALYAHYA

AESHAH ALMAKHLIFI

Content Outline

Topics for discussion



What is the problem?

Recently, the prevalence of liver diseases has been increasing, becoming more common among people. This trend leads to numerous serious issues in individuals' lives, potentially resulting in fatal outcomes. In our project, we aim to study and analyse patient data, which will greatly assist in identifying possible factors and risks associated with liver diseases. By predicting the likelihood of developing liver disease, we can help many individuals take preventive measures to safeguard their health.

DATA

About the **Dataset**

No. of Attribute

11

No. of Object

583

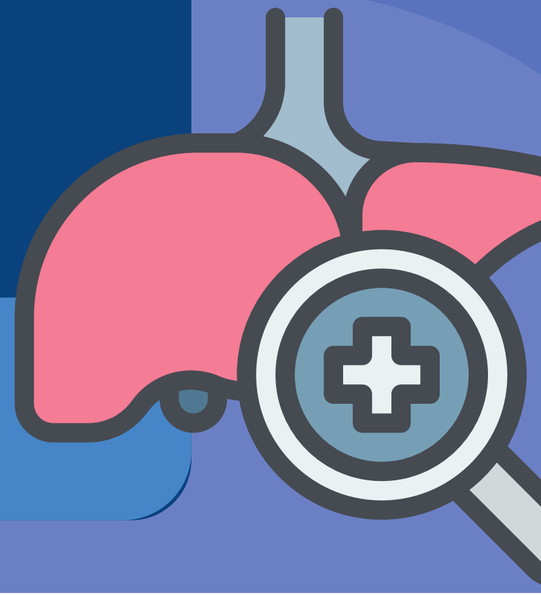
Class label

Selector

Has disease or not!

Source:

Click here to view



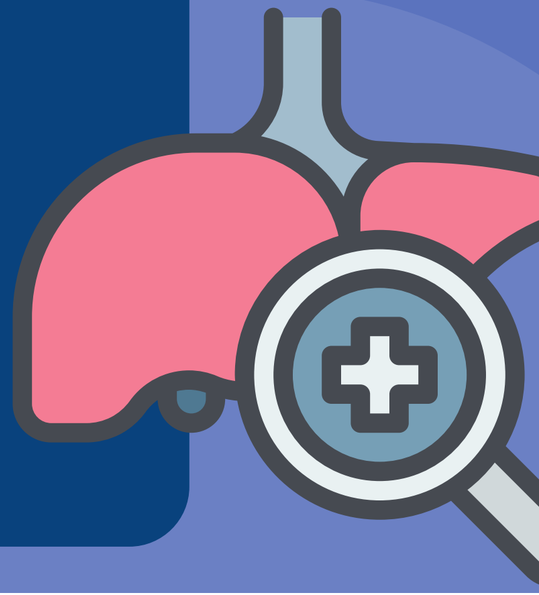
About the **Attribute**

Numeric **Age**	female or male **Gender**	Numeric **TB**	Numeric **DB**
Numeric **Alkphos**	Numeric **Sgpt**	Numeric **Sgot**	Numeric **TP**
Numeric **ALB**	Numeric **A/G Ratio**		

Summary Statistical Measures

	Age	TB	DB	Alkphos	Sgpt
count	583.000000	583.000000	583.000000	583.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551
std	16.189833	6.209522	2.808498	242.937989	182.620356
min	4.000000	0.400000	0.100000	63.000000	10.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000
75%	58.000000	2.600000	1.300000	298.000000	60.500000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000

	Sgot	TP	ALB	A/G Ratio	Selector
count	583.000000	583.000000	583.000000	579.000000	583.000000
mean	109.910806	6.483190	3.141852	0.947064	1.286449
std	288.918529	1.085451	0.795519	0.319592	0.452490
min	10.000000	2.700000	0.900000	0.300000	1.000000
25%	25.000000	5.800000	2.600000	0.700000	1.000000
50%	42.000000	6.600000	3.100000	0.930000	1.000000
75%	87.000000	7.200000	3.800000	1.100000	2.000000
max	4929.000000	9.600000	5.500000	2.800000	2.000000



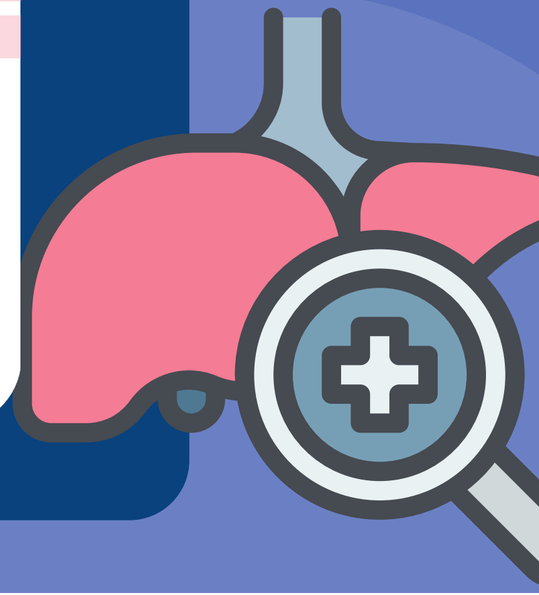
About the **Dataset**

Missing value

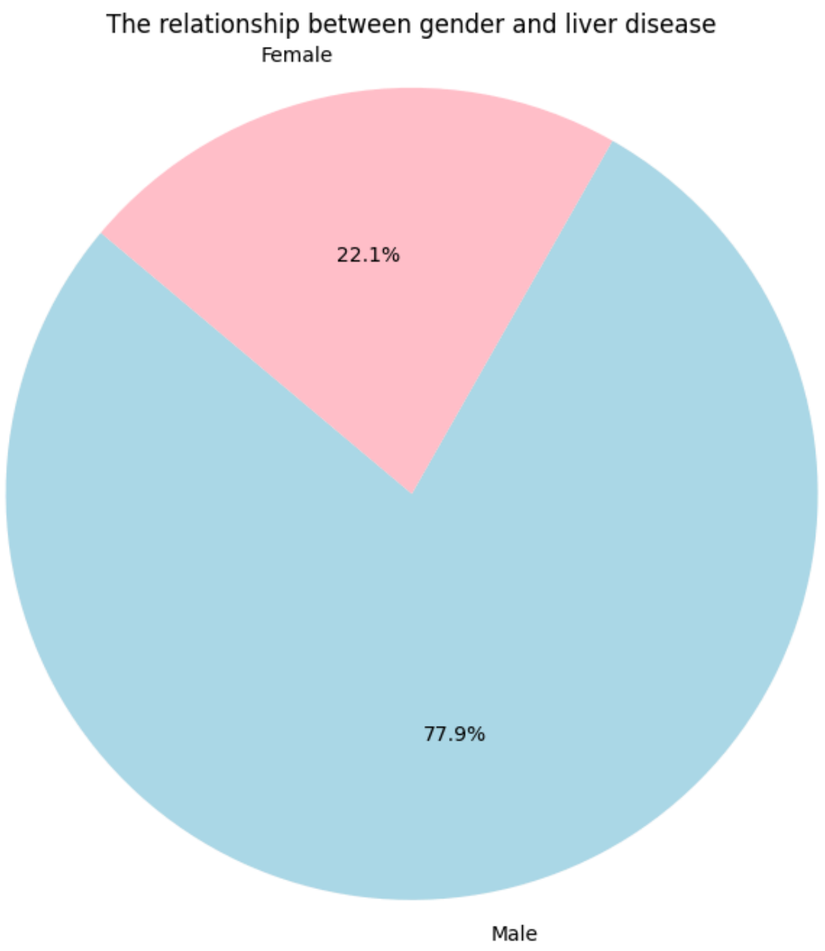
```
Missing values in each column:
Age           0
Gender        0
TB            0
DB            0
Alkphos       0
Sgpt          0
Sgot          0
TP            0
ALB           0
A/G Ratio     4
Selector      0
dtype: int64
```

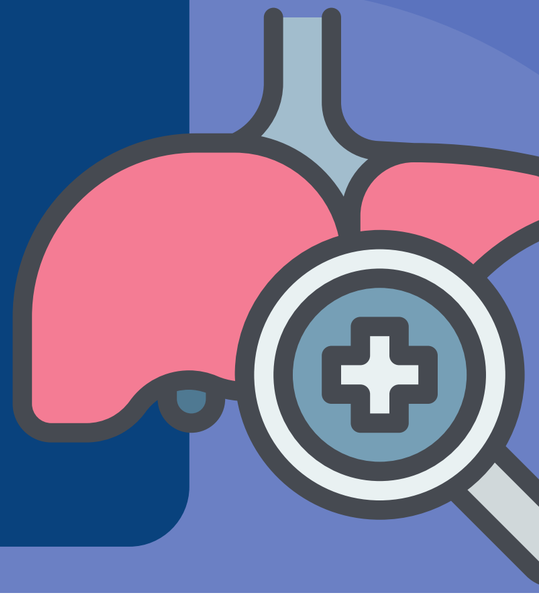
Variance

```
Age             262.110702
TB               38.558160
DB                7.887659
Alkphos       59018.866587
Sgpt          33350.194438
Sgot          83473.916429
TP                1.178205
ALB               0.632850
A/G Ratio         0.102139
Selector          0.204747
dtype: float64
```

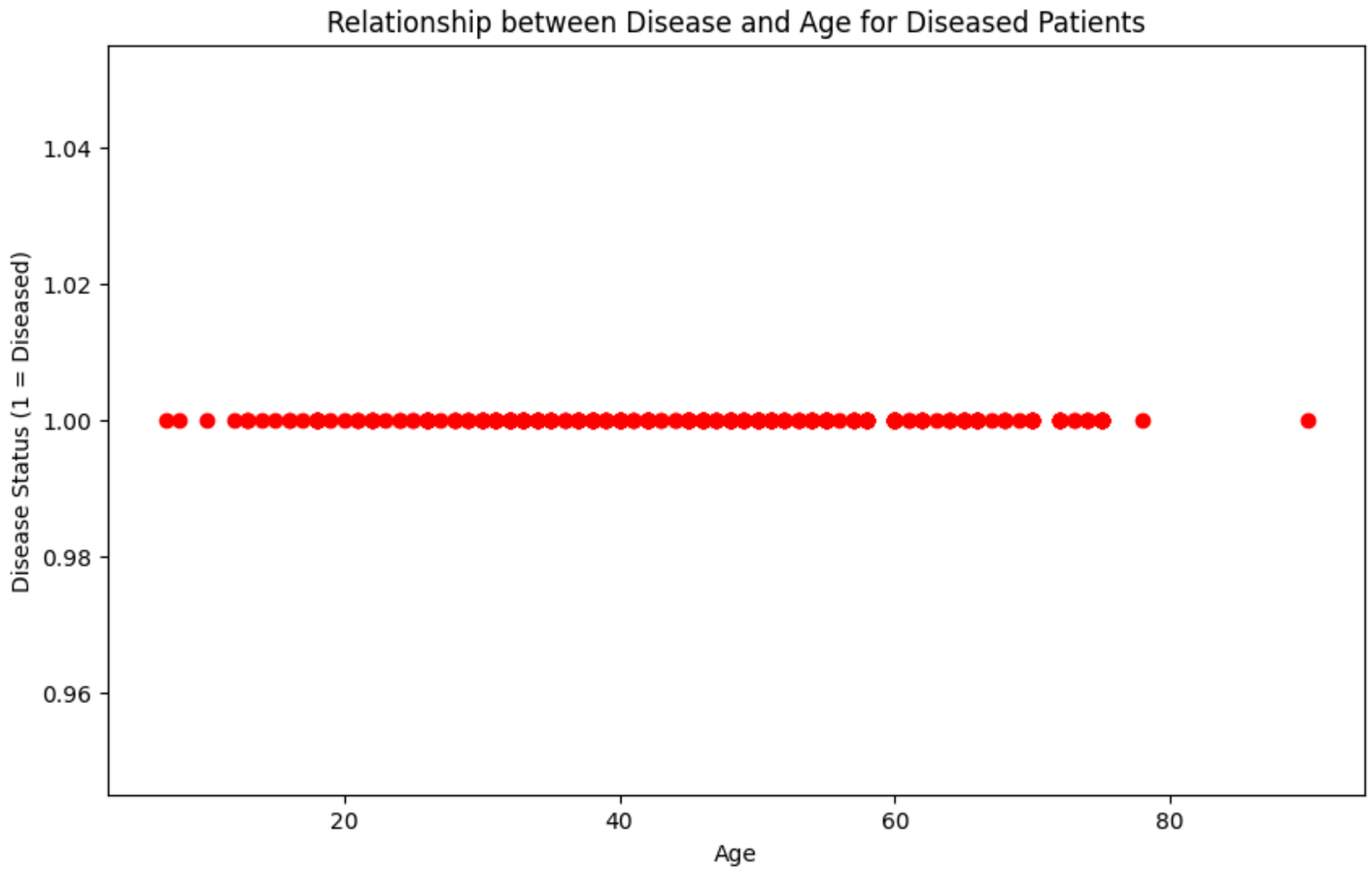


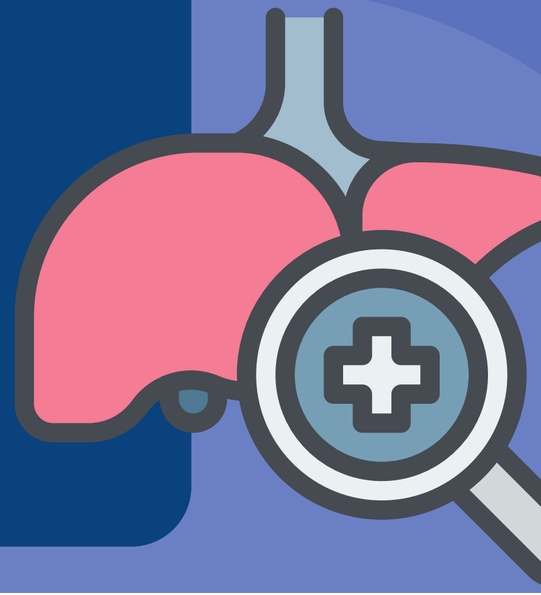
Data Visualisation



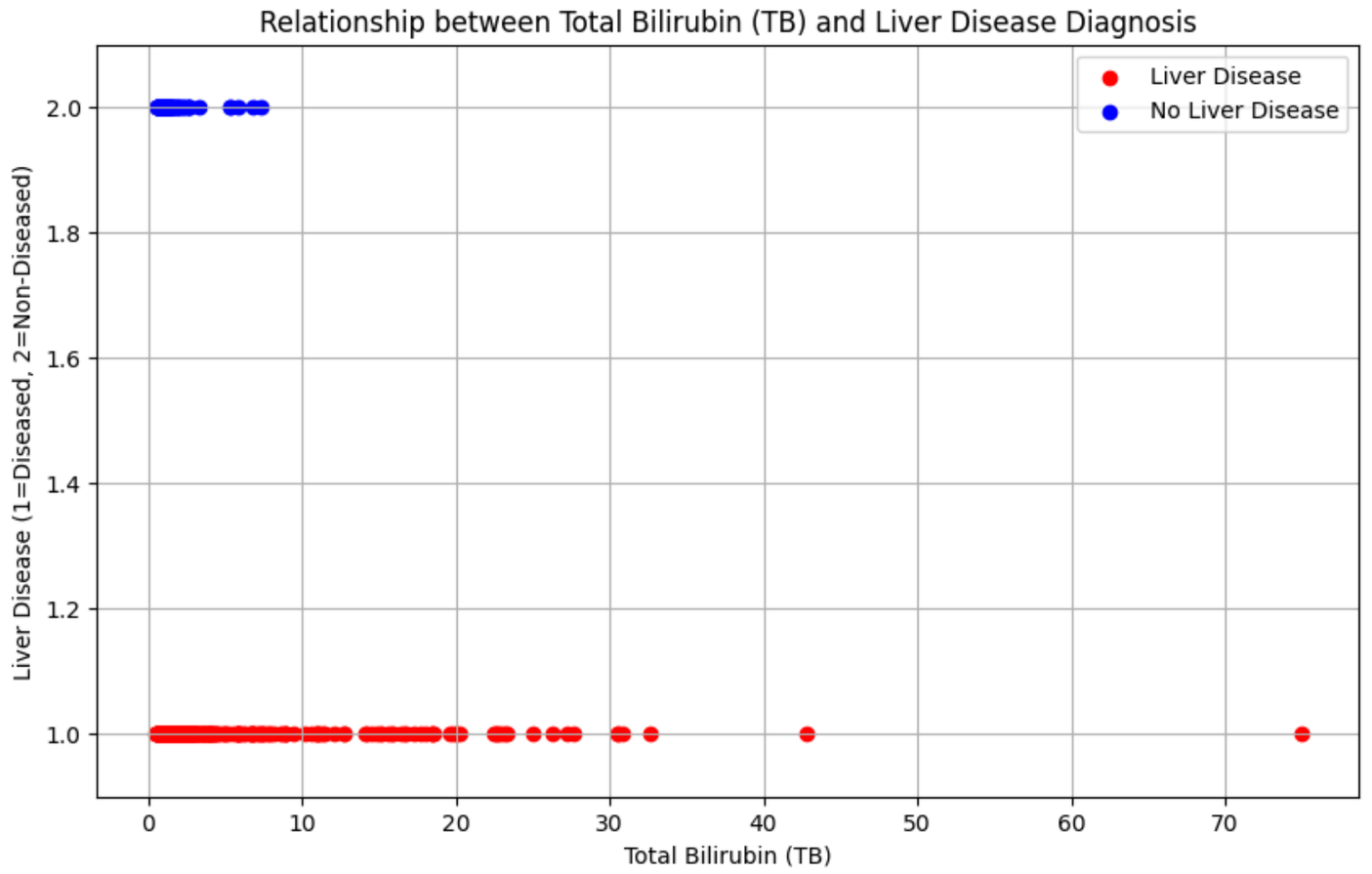


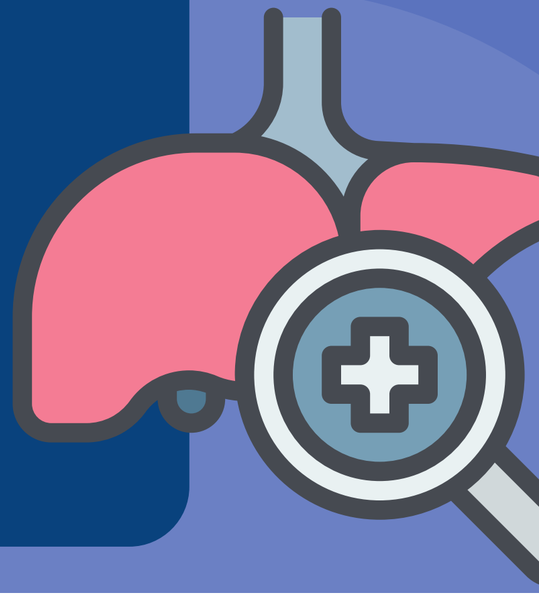
Data Visualisation



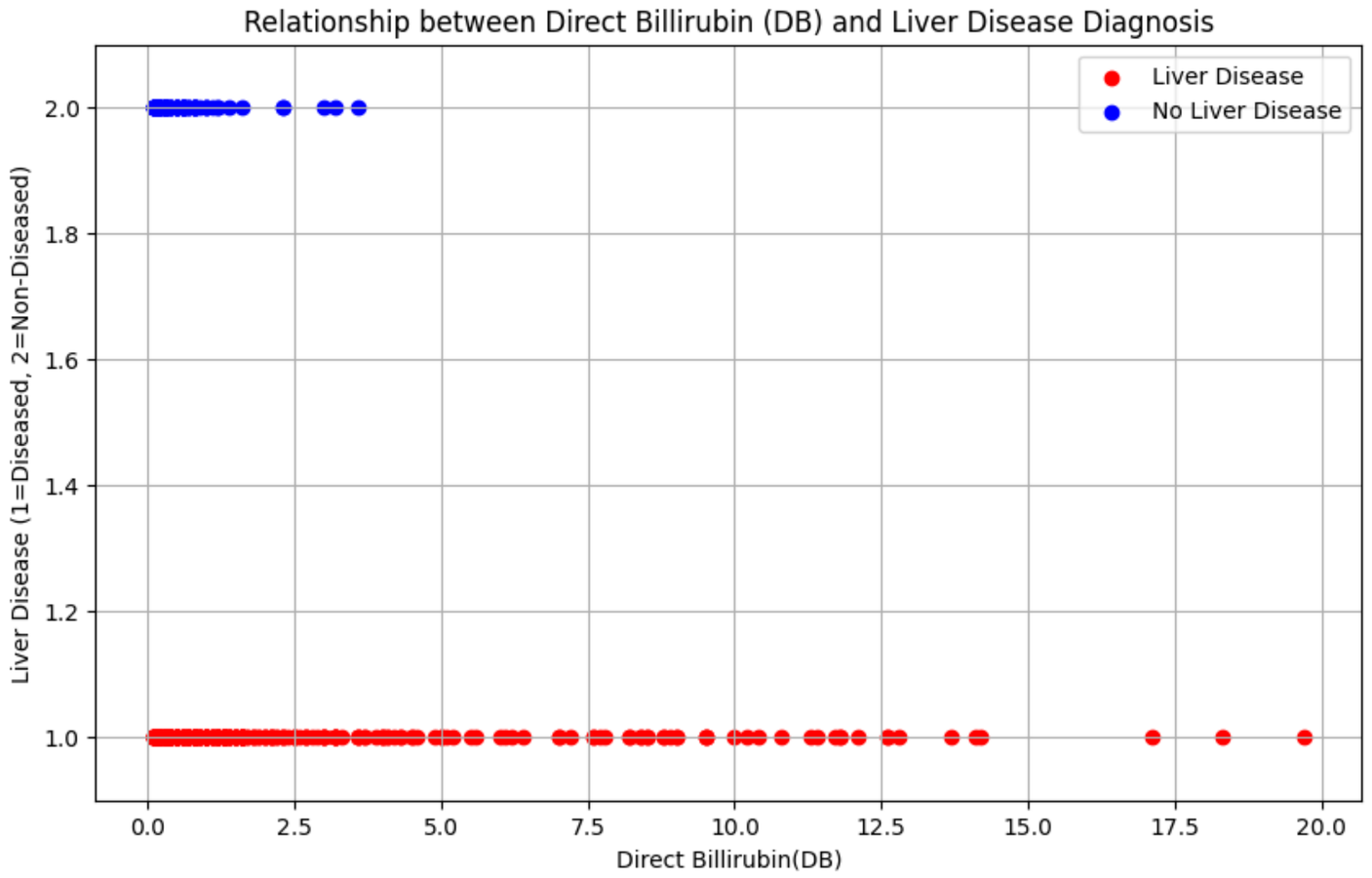


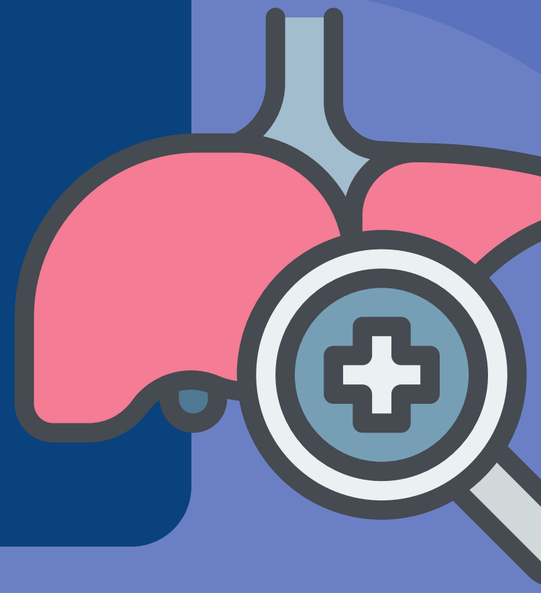
Data Visualisation



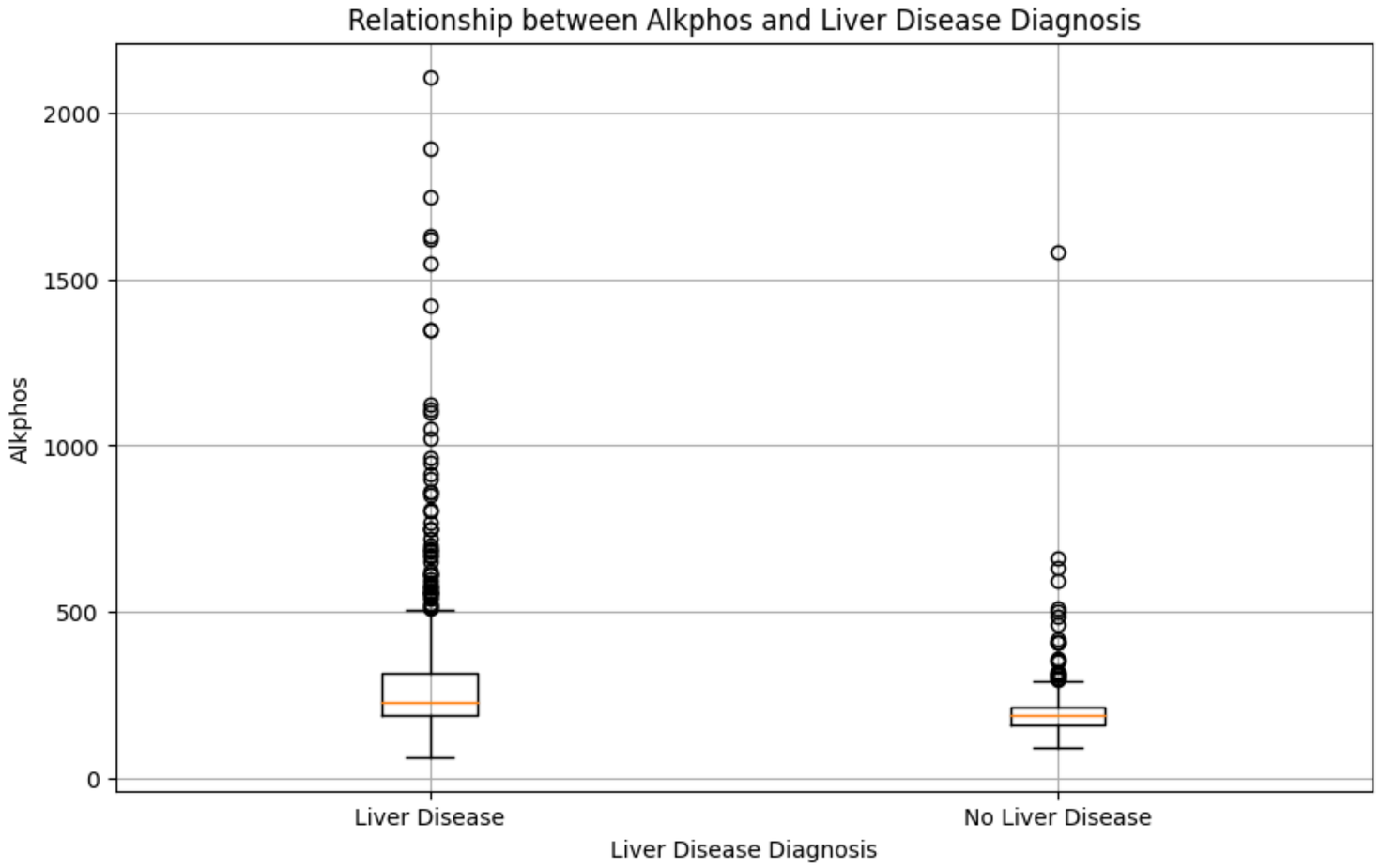


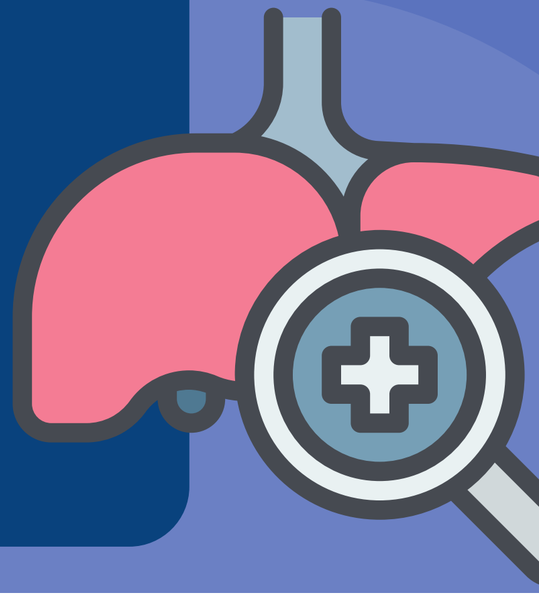
Data Visualisation



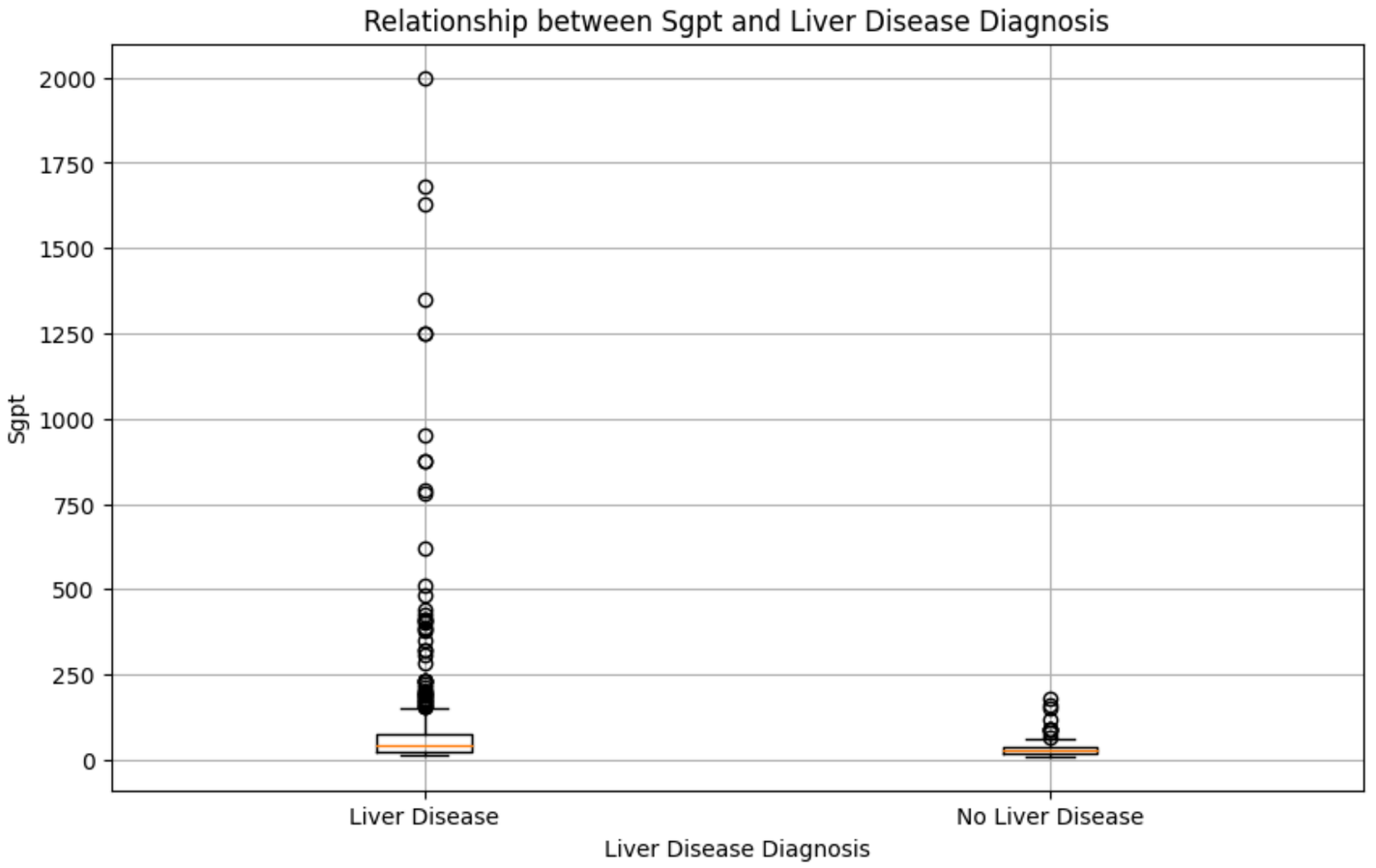


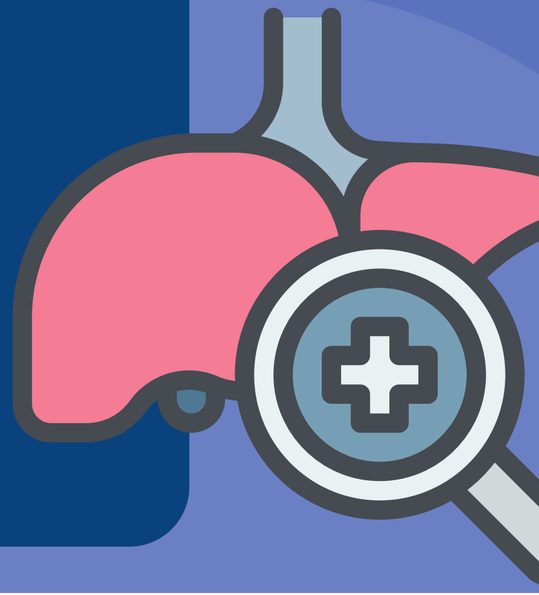
Data Visualisation



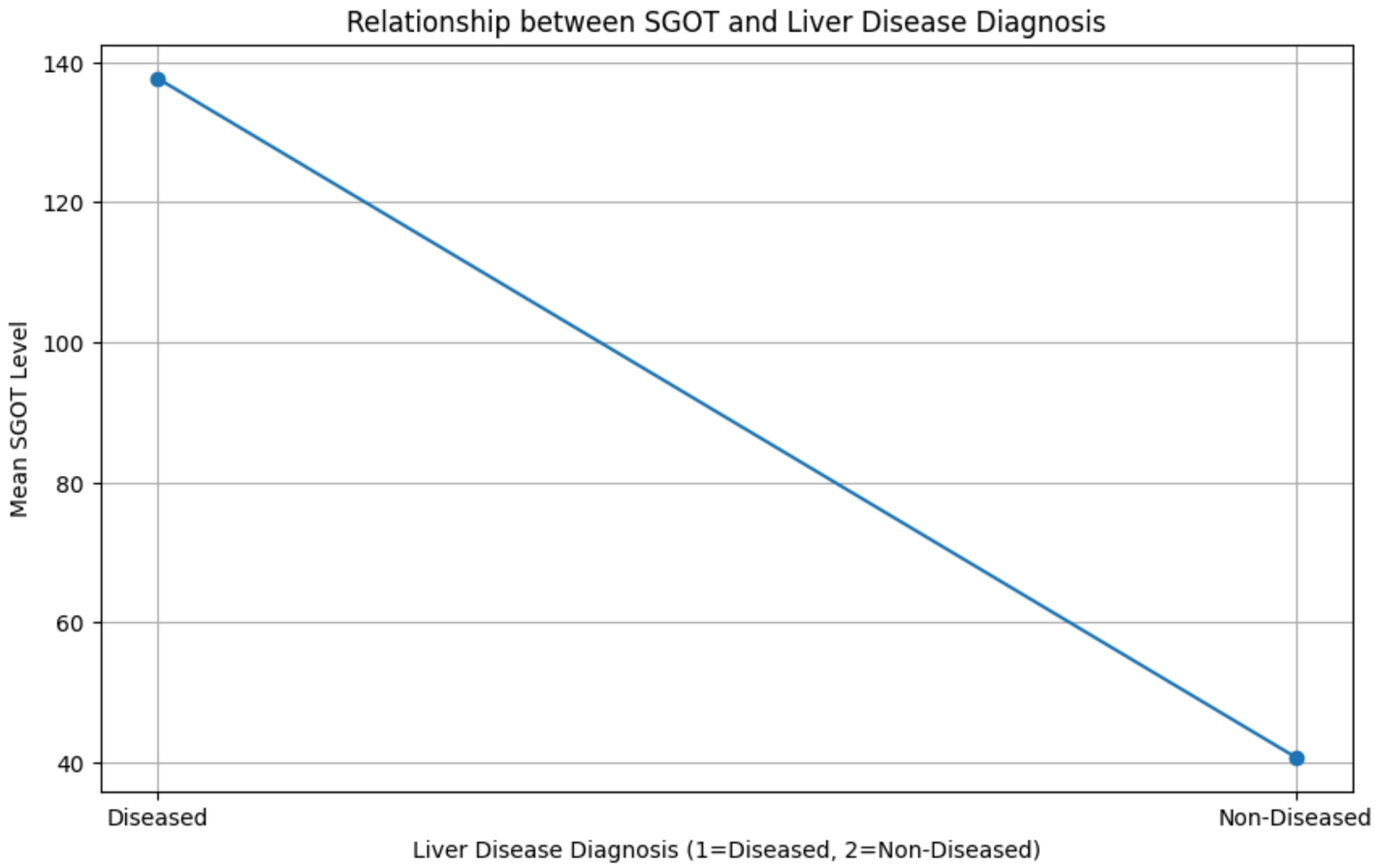


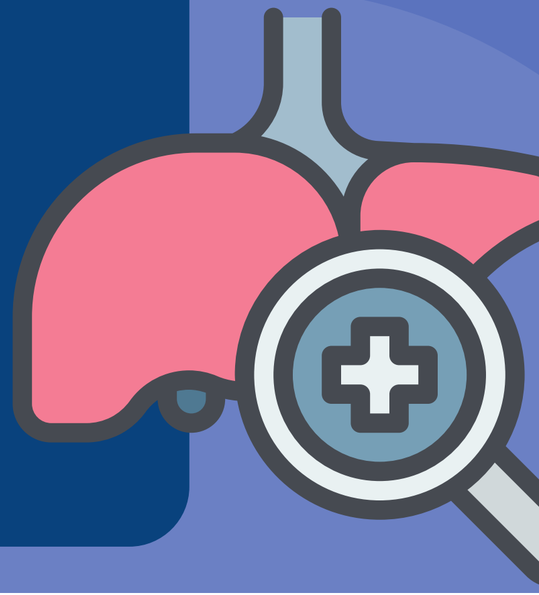
Data Visualisation



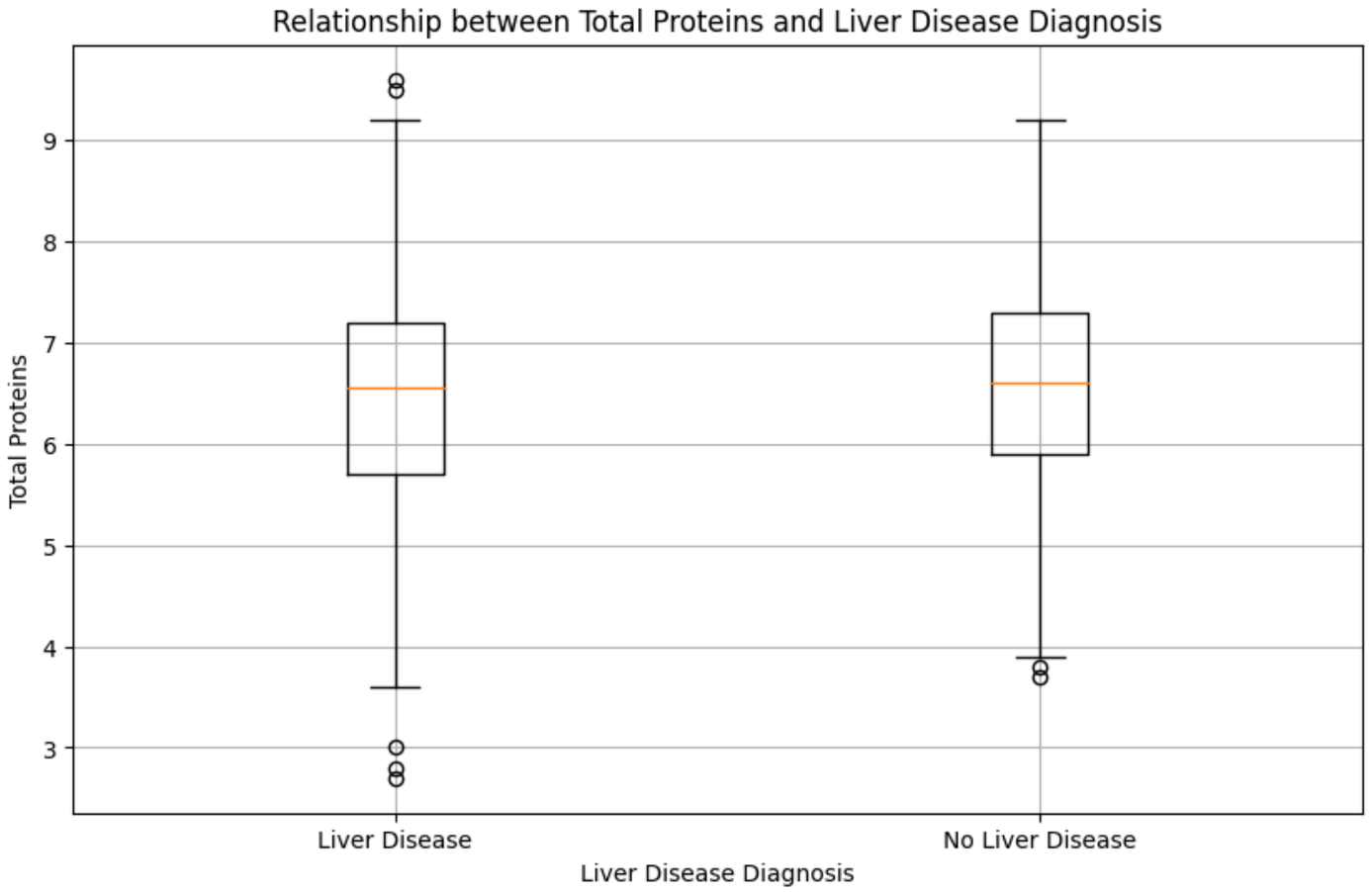


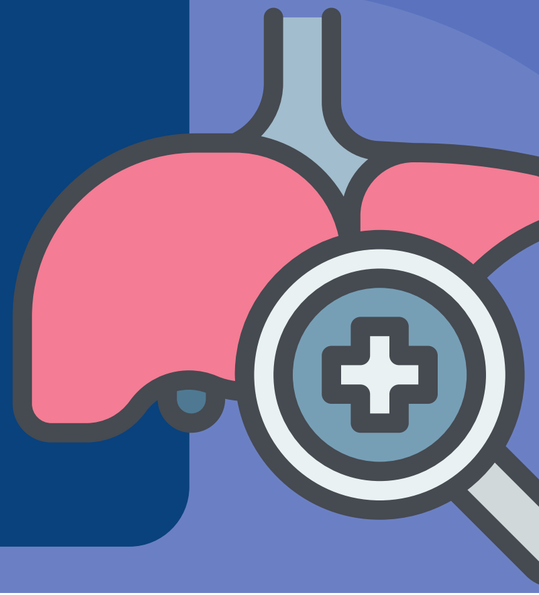
Data Visualisation



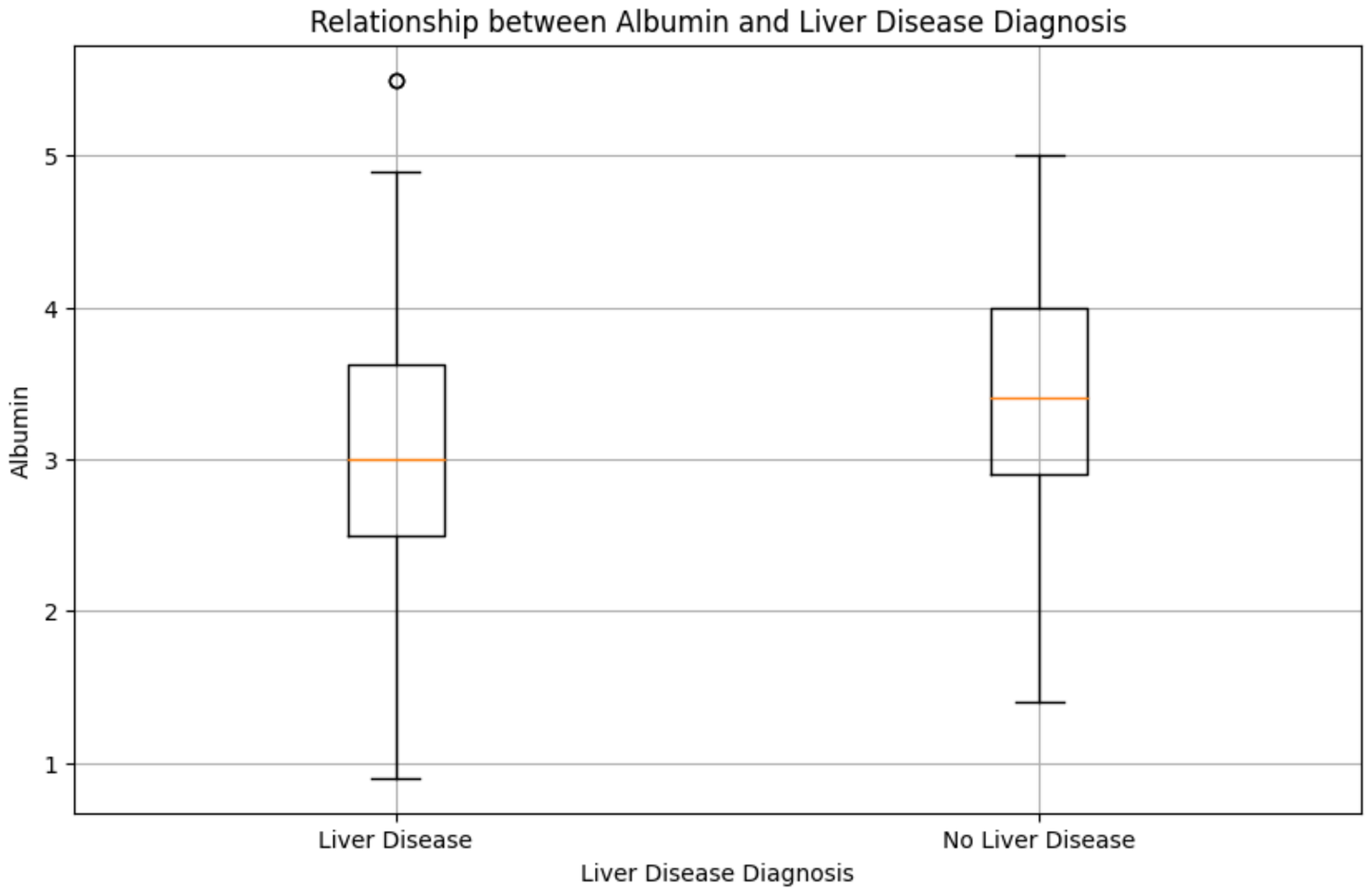


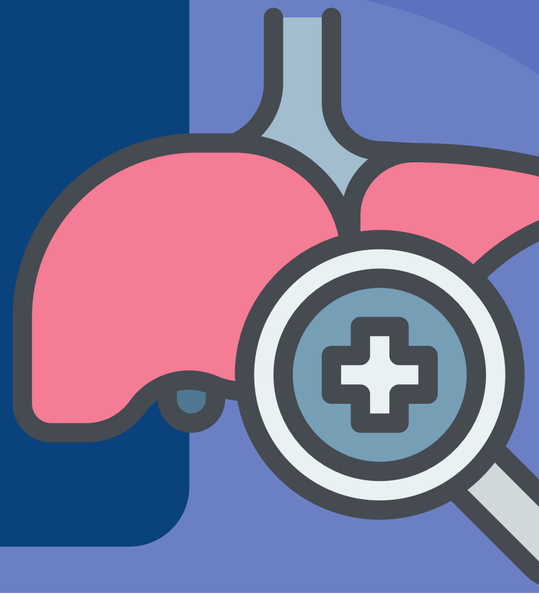
Data Visualisation



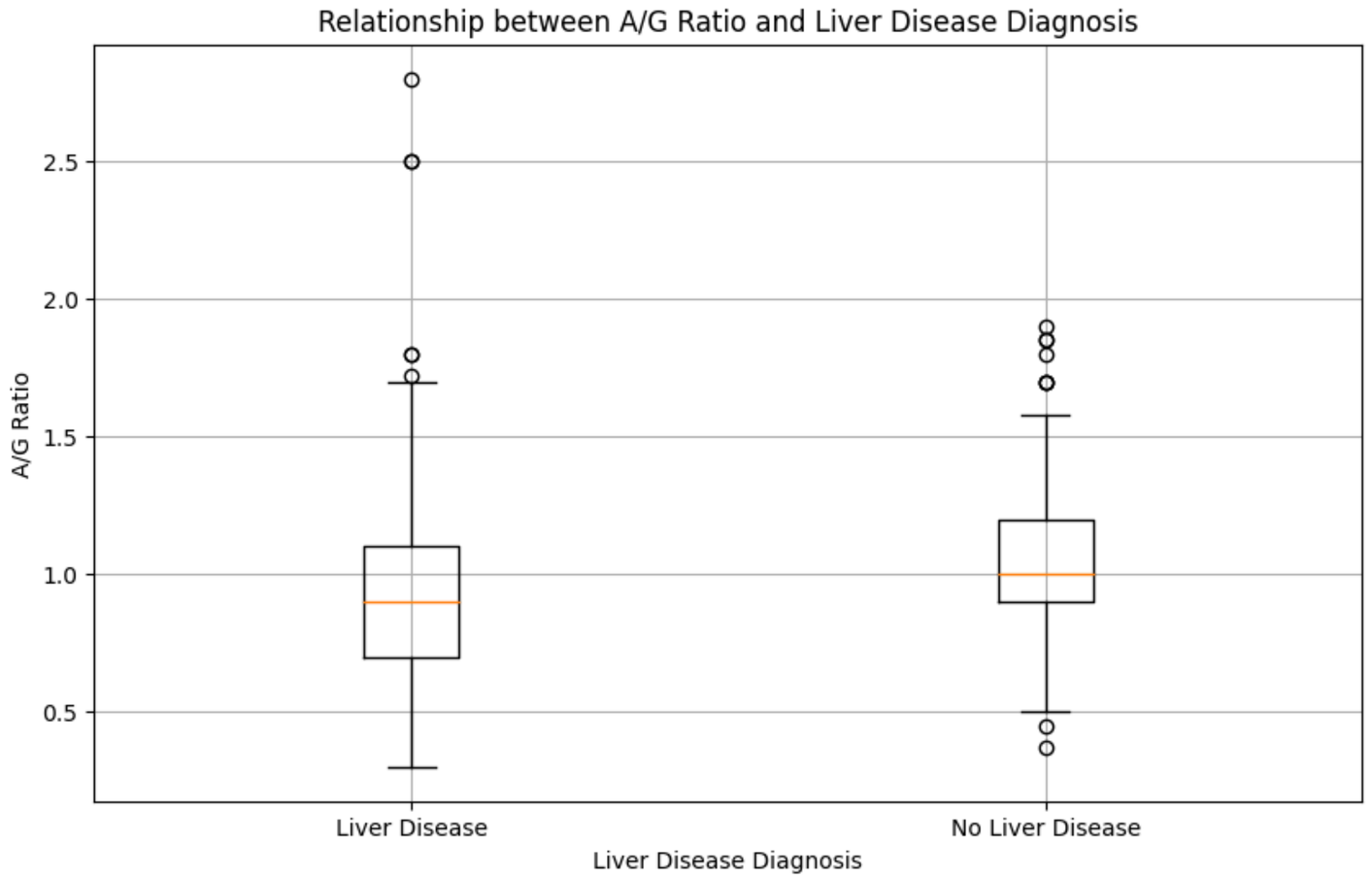


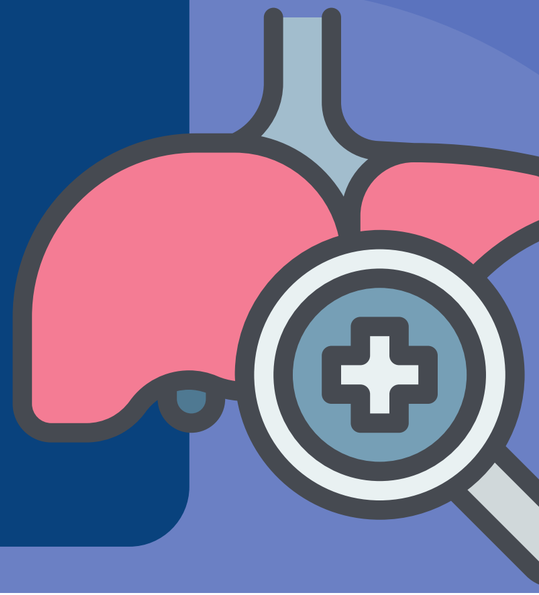
Data Visualisation





Data Visualisation





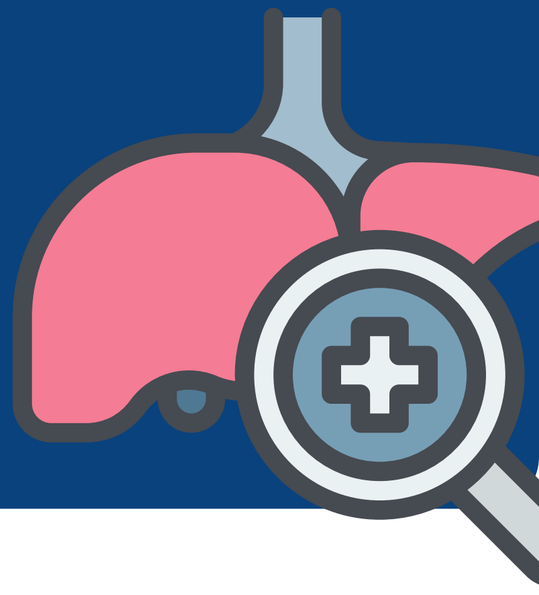
Data preprocessing

DATA CLEANING:
DUPLICATED ROWS

```python
import pandas as pd
data = pd.read_csv('Indian Liver Patient Dataset (ILPD).csv')
num_duplicates = data.duplicated().sum()
data_cleaned = data.drop_duplicates()
print("Number of duplicate rows:", num_duplicates)
print("DataFrame after dropping all duplicate rows:")
print(data_cleaned)
data_cleaned.to_csv('Cleaned_dataset.csv', index=False)
```

```
Number of duplicate rows: 13
DataFrame after dropping all duplicate rows:
     Age  Gender    TB   DB  Alkphos  Sgpt  Sgot   TP  ALB  A/G Ratio  \
0     65  Female   0.7  0.1      187    16    18  6.8  3.3       0.90
1     62    Male  10.9  5.5      699    64   100  7.5  3.2       0.74
2     62    Male   7.3  4.1      490    60    68  7.0  3.3       0.89
3     58    Male   1.0  0.4      182    14    20  6.8  3.4       1.00
4     72    Male   3.9  2.0      195    27    59  7.3  2.4       0.40
..   ...     ...   ...  ...      ...   ...   ...  ...  ...        ...
578   60    Male   0.5  0.1      500    20    34  5.9  1.6       0.37
579   40    Male   0.6  0.1       98    35    31  6.0  3.2       1.10
580   52    Male   0.8  0.2      245    48    49  6.4  3.2       1.00
581   31    Male   1.3  0.5      184    29    32  6.8  3.4       1.00
582   38    Male   1.0  0.3      216    21    24  7.3  4.4       1.50
```

```
Number of rows: 570
Dataset:
     Age  Gender    TB   DB  Alkphos  Sgpt  Sgot   TP  ALB  A/G Ratio  \
0     65  Female   0.7  0.1      187    16    18  6.8  3.3       0.90
1     62    Male  10.9  5.5      699    64   100  7.5  3.2       0.74
2     62    Male   7.3  4.1      490    60    68  7.0  3.3       0.89
3     58    Male   1.0  0.4      182    14    20  6.8  3.4       1.00
4     72    Male   3.9  2.0      195    27    59  7.3  2.4       0.40
..   ...     ...   ...  ...      ...   ...   ...  ...  ...        ...
565   60    Male   0.5  0.1      500    20    34  5.9  1.6       0.37
566   40    Male   0.6  0.1       98    35    31  6.0  3.2       1.10
567   52    Male   0.8  0.2      245    48    49  6.4  3.2       1.00
568   31    Male   1.3  0.5      184    29    32  6.8  3.4       1.00
569   38    Male   1.0  0.3      216    21    24  7.3  4.4       1.50
```



DATA CLEANING:
HANDLING MISSING VALUES.

```
Missing values per column:
Age           0
Gender        0
TB            0
DB            0
Alkphos       0
Sgpt          0
Sgot          0
TP            0
ALB           0
A/G Ratio     0
Selector      0
dtype: int64
```

```python
import pandas as pd
data = pd.read_csv('Cleaned_dataset.csv');
sample=data.sample(n=20);
print(sample);
missing_values = data.isna()
missing_counts = missing_values.sum();
rows_with_missing = data[data.isna().any(axis=1)];

print("Missing values in each column:");
print(missing_counts);
print("\nRows with missing values:");
print(rows_with_missing);
```

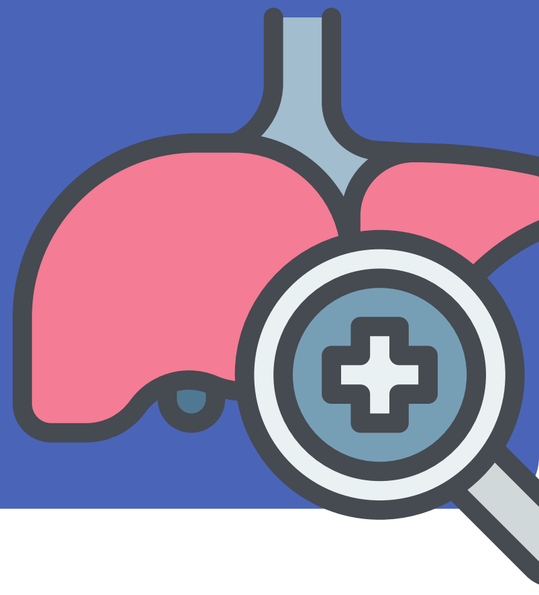
```
Missing values in each column:
Age           0
Gender        0
TB            0
DB            0
Alkphos       0
Sgpt          0
Sgot          0
TP            0
ALB           0
A/G Ratio     4
Selector      0
dtype: int64
```

```python
mean_value = data['A/G Ratio'].mean()
data['A/G Ratio'].fillna(value=mean_value, inplace=True)

missing_values = data.isnull().sum()
print('Missing values per column:')
print(missing_values)
data.to_csv('Cleaned_dataset.csv', index=False)
```

DATA CLEANING:
HANDLING OUTLIERS .

```python
import pandas as pd
data = pd.read_csv('Cleaned_dataset.csv')
import numpy as np
outlier_threshold = 1.5

def count_outliers(column_data):
    q1 = np.percentile(column_data, 25)
    q3 = np.percentile(column_data, 75)
    iqr = q3 - q1
    upper_bound = q3 + outlier_threshold * iqr
    lower_bound = q1 - outlier_threshold * iqr
    outliers = (column_data > upper_bound) | (column_data < lower_bound)
    return sum(outliers)

numeric_columns = data.select_dtypes(include=[np.number]).columns

outlier_counts = {}
total_rows_with_outliers = 0

for column in numeric_columns:
    outliers = count_outliers(data[column])
    outlier_counts[column] = outliers
    total_rows_with_outliers += outliers

total_rows = len(data)

print("Outlier Counts:")
for column, count in outlier_counts.items():
    print(f"{column}: {count} rows with outliers")

print(f"Total Rows with Outliers: {total_rows_with_outliers}")
```

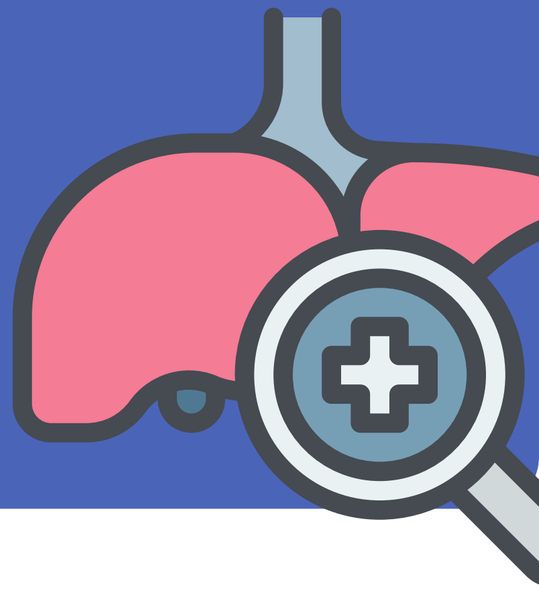
```
Outlier Counts:
Age: 0 rows with outliers
TB: 83 rows with outliers
DB: 80 rows with outliers
Alkphos: 69 rows with outliers
Sgpt: 72 rows with outliers
Sgot: 66 rows with outliers
TP: 8 rows with outliers
ALB: 0 rows with outliers
A/G Ratio: 10 rows with outliers
Selector: 0 rows with outliers
Total Rows with Outliers: 388
```



DATA CLEANING:
HANDLING OUTLIERS .

```python
import numpy as np

outlier_threshold = 1.5

def count_outliers(column_data):
    q1 = np.percentile(column_data, 25)
    q3 = np.percentile(column_data, 75)
    iqr = q3 - q1
    upper_bound = q3 + outlier_threshold * iqr
    lower_bound = q1 - outlier_threshold * iqr
    outliers = (column_data > upper_bound) | (column_data < lower_bound)
    return sum(outliers)

numeric_columns = data.select_dtypes(include=[np.number]).columns

outlier_counts = {}
total_rows = len(data)

for column in numeric_columns:
    outliers = count_outliers(data[column])
    outlier_counts[column] = outliers

    # Cap outliers by setting them to the nearest non-outlier value
    q1 = np.percentile(data[column], 25)
    q3 = np.percentile(data[column], 75)
    iqr = q3 - q1
    upper_bound = q3 + outlier_threshold * iqr
    lower_bound = q1 - outlier_threshold * iqr
    data[column] = np.clip(data[column], lower_bound, upper_bound)

data.to_csv('Cleaned_dataset.csv', index=False)
```

```
Outlier Counts:
Age: 0 rows with outliers
TB: 0 rows with outliers
DB: 0 rows with outliers
Alkphos: 0 rows with outliers
Sgpt: 0 rows with outliers
Sgot: 0 rows with outliers
TP: 0 rows with outliers
ALB: 0 rows with outliers
A/G Ratio: 0 rows with outliers
Selector: 0 rows with outliers
Total Rows with Outliers: 0
```



DATA TRANSMATION:

Data transformation involves modifying raw data into a more usable format through processes , making it suitable for analysis and decision-making.

THE PROCESSES WE APPLY :

- ENCODING
- NORMALIZTION
- AGGREGATION
- DISCRETIZATION

DATA ENCODING:

applying encoding

We applied encoding to convert non-numeric data into a format that can be easily processed .

so, we represent the gender with values 0 and 1 .

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
from scipy import stats

data1= pd.read_csv('Cleaned_dataset.csv')
le = LabelEncoder()
data1['Gender'] = le.fit_transform(data1['Gender'])
print(data1)
data1.to_csv('encoding_normaliztion.csv', index=False)
```

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	A/G Ratio	\
0	65	0	0.7	0.10	187	16.0	18.0	6.8	3.3	0.90	
1	62	1	5.3	2.95	481	64.0	100.0	7.5	3.2	0.74	
2	62	1	5.3	2.95	481	60.0	68.0	7.0	3.3	0.89	
3	58	1	1.0	0.40	182	14.0	20.0	6.8	3.4	1.00	
4	72	1	3.9	2.00	195	27.0	59.0	7.3	2.4	0.40	
..	...	...	...	...	...	...	...	...	...	...	
565	60	1	0.5	0.10	481	20.0	34.0	5.9	1.6	0.37	
566	40	1	0.6	0.10	98	35.0	31.0	6.0	3.2	1.10	
567	52	1	0.8	0.20	245	48.0	49.0	6.4	3.2	1.00	
568	31	1	1.3	0.50	184	29.0	32.0	6.8	3.4	1.00	
569	38	1	1.0	0.30	216	21.0	24.0	7.3	4.4	1.50	

DATA NORMALIZTION:

applying Normaliztion

we apply normaliztion to normalize the attributes and unify their scale since the range for each attribute is quite different.

This method helps us to format all the values in dataset and facilitates the analysis process.

```python
import pandas as pd

data1 = pd.read_csv('encoding_normaliztion.csv')
data1 = pd.DataFrame(data1)

# Columns to normalize
columns_to_normalize = ['TP', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB', 'A/G Ratio']

# Decimal scaling normalization
for column in columns_to_normalize:
    max_abs_value = data1[column].abs().max()
    data1[column] = data1[column] / (10 ** len(str(int(max_abs_value))))

print("DataFrame after Decimal Scaling Normalization:")
print(data1)
data1.to_csv('encoding_normaliztion.csv', index=False)
```

```
DataFrame after Decimal Scaling Normalization:
     Age  Gender   TB     DB  Alkphos   Sgpt   Sgot     TP   ALB  A/G Ratio  \
0     65       0  0.7  0.010    0.187  0.016  0.018  0.068  0.33      0.090
1     62       1  5.3  0.295    0.481  0.064  0.100  0.075  0.32      0.074
2     62       1  5.3  0.295    0.481  0.060  0.068  0.070  0.33      0.089
3     58       1  1.0  0.040    0.182  0.014  0.020  0.068  0.34      0.100
4     72       1  3.9  0.200    0.195  0.027  0.059  0.073  0.24      0.040
..   ...     ...  ...    ...      ...    ...    ...    ...   ...        ...
565   60       1  0.5  0.010    0.481  0.020  0.034  0.059  0.16      0.037
566   40       1  0.6  0.010    0.098  0.035  0.031  0.060  0.32      0.110
567   52       1  0.8  0.020    0.245  0.048  0.049  0.064  0.32      0.100
568   31       1  1.3  0.050    0.184  0.029  0.032  0.068  0.34      0.100
569   38       1  1.0  0.030    0.216  0.021  0.024  0.073  0.44      0.150

     Selector
0           1
1           1
2           1
3           1
4           1
```

DATA AGGREGATION:

applying Aggregation

Aggregation is used to simplify data by grouping the "Gender" and "Selector" columns and applying the "mean" function, yielding average attribute values for selected and non-selected male and female patients.

```
In [3]:
import pandas as pd

data1 = pd.read_csv('encoding_normaliztion.csv')
data1.groupby('Gender').agg('max')
data1.groupby(['Gender', 'Selector']).agg('mean')
```

Gender	Selector	Age	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	A/G Ratio
0	1	43.384615	1.687912	0.074286	0.264253	0.043676	0.058323	0.066934	0.323297	0.091701
	2	42.836735	0.871429	0.024898	0.198490	0.028235	0.031327	0.066000	0.335714	0.100833
1	1	47.107937	2.420000	0.118238	0.269467	0.056414	0.078999	0.064133	0.301587	0.090323
	2	40.678261	1.177391	0.042913	0.214513	0.034491	0.042737	0.065443	0.335826	0.103885

DATA DISCRETIZATION:

applying Discretization

we apply Discretization to transform continuous data into discrete categories.

we apply it in " Age " attribute to making it easier to handle and analyze.

```python
import pandas as pd

data1 = pd.read_csv('encoding_normaliztion.csv')

median_age = data1["Age"].median()
data1["Age"].fillna(median_age, inplace=True)

# Discretize the "Age" column into ranked age groups
age_labels = ["Children", "Adults", "Seniors"]
data1["Age"] = pd.qcut(data1["Age"], q=3, labels=age_labels)

print(data1)
data1.to_csv('after_discretiztion.csv', index=False)
```

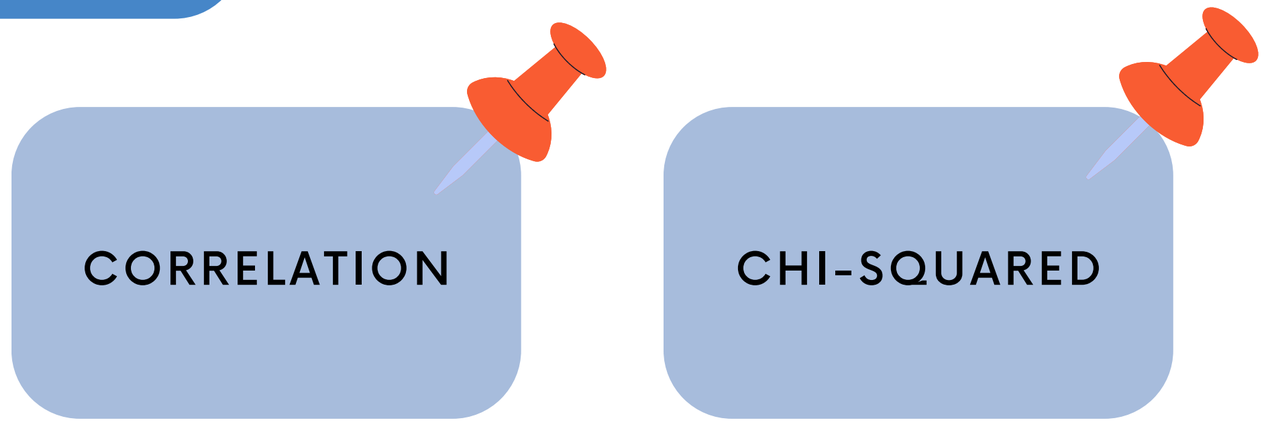
	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	\
0	Seniors	0	0.7	0.010	0.187	0.016	0.018	0.068	0.33	
1	Seniors	1	5.3	0.295	0.481	0.064	0.100	0.075	0.32	
2	Seniors	1	5.3	0.295	0.481	0.060	0.068	0.070	0.33	
3	Seniors	1	1.0	0.040	0.182	0.014	0.020	0.068	0.34	
4	Seniors	1	3.9	0.200	0.195	0.027	0.059	0.073	0.24	
..	...	...	...	...	...	...	...	...	...	
565	Seniors	1	0.5	0.010	0.481	0.020	0.034	0.059	0.16	
566	Adults	1	0.6	0.010	0.098	0.035	0.031	0.060	0.32	
567	Adults	1	0.8	0.020	0.245	0.048	0.049	0.064	0.32	
568	Children	1	1.3	0.050	0.184	0.029	0.032	0.068	0.34	
569	Adults	1	1.0	0.030	0.216	0.021	0.024	0.073	0.44	

	A/G Ratio	Selector
0	0.090	1
1	0.074	1

SELECTION FEASTUR:

Feature selection is a process in data preprocessing used to select the most relevant features for use in model construction. It reduces the complexity of models, improves their performance.

THE PROCESSES WE APPLY :



CORRELATION COEFFICIENT:

applying Correlation

we used the correlation coefficients between each numeric attributes and the 'Selector' , we use it to measures the strength and direction of the linear relationship between two variables .

```python
import pandas as pd
from scipy.stats import chi2_contingency

data1 = pd.read_csv("after_discretiztion.csv")

# Correlation calculations
cor_gender = data1['Gender'].corr(data1['Selector'])
cor_TB = data1['TB'].corr(data1['Selector'])
cor_DB = data1['DB'].corr(data1['Selector'])
cor_Alkphos = data1['Alkphos'].corr(data1['Selector'])
cor_Sgpt = data1['Sgpt'].corr(data1['Selector'])
cor_sgot = data1['Sgot'].corr(data1['Selector'])
cor_TP = data1['TP'].corr(data1['Selector'])
cor_ALB = data1['ALB'].corr(data1['Selector'])
cor_A_G_Ratio = data1['A/G Ratio'].corr(data1['Selector'])

print("Correlation coefficients:")
print("Gender:", cor_gender)
print("TB:", cor_TB)
print("DB:", cor_DB)
print("Alkphos:", cor_Alkphos)
print("Sgpt:", cor_Sgpt)
print("Sgot:", cor_sgot)
print("TP:", cor_TP)
print("ALB:", cor_ALB)
print("A/G Ratio:", cor_A_G_Ratio)
```

```
Correlation coefficients:
Gender: -0.07850064087057382
TB: -0.32065060827128267
DB: -0.32238401191602156
Alkphos: -0.2415042278254347
Sgpt: -0.287635527288642
Sgot: -0.29710060539822697
TP: 0.035910434434097444
ALB: 0.16683508037277275
A/G Ratio: 0.18900587094752194
```

CHI-SQUARE:

applying Chi-square

we used Chi-square for "Age" attribute , this test is used to determine if there is a statistically significant relationship between categorical attributes.

```python
import pandas as pd
from scipy.stats import chi2_contingency
data1 = pd.read_csv('after_discretiztion.csv')
data1 = pd.DataFrame(data1)

# Create a contingency table
contingency_table = pd.crosstab(data1['Age'], data1['Selector'])
print("Contingency Table:")
print(contingency_table)

#Perform the chi-square test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
print("\nChi-Square Statistic:", chi2_stat)
print("Degrees of Freedom:", dof)
print("Expected Frequencies:")
print(expected)
```

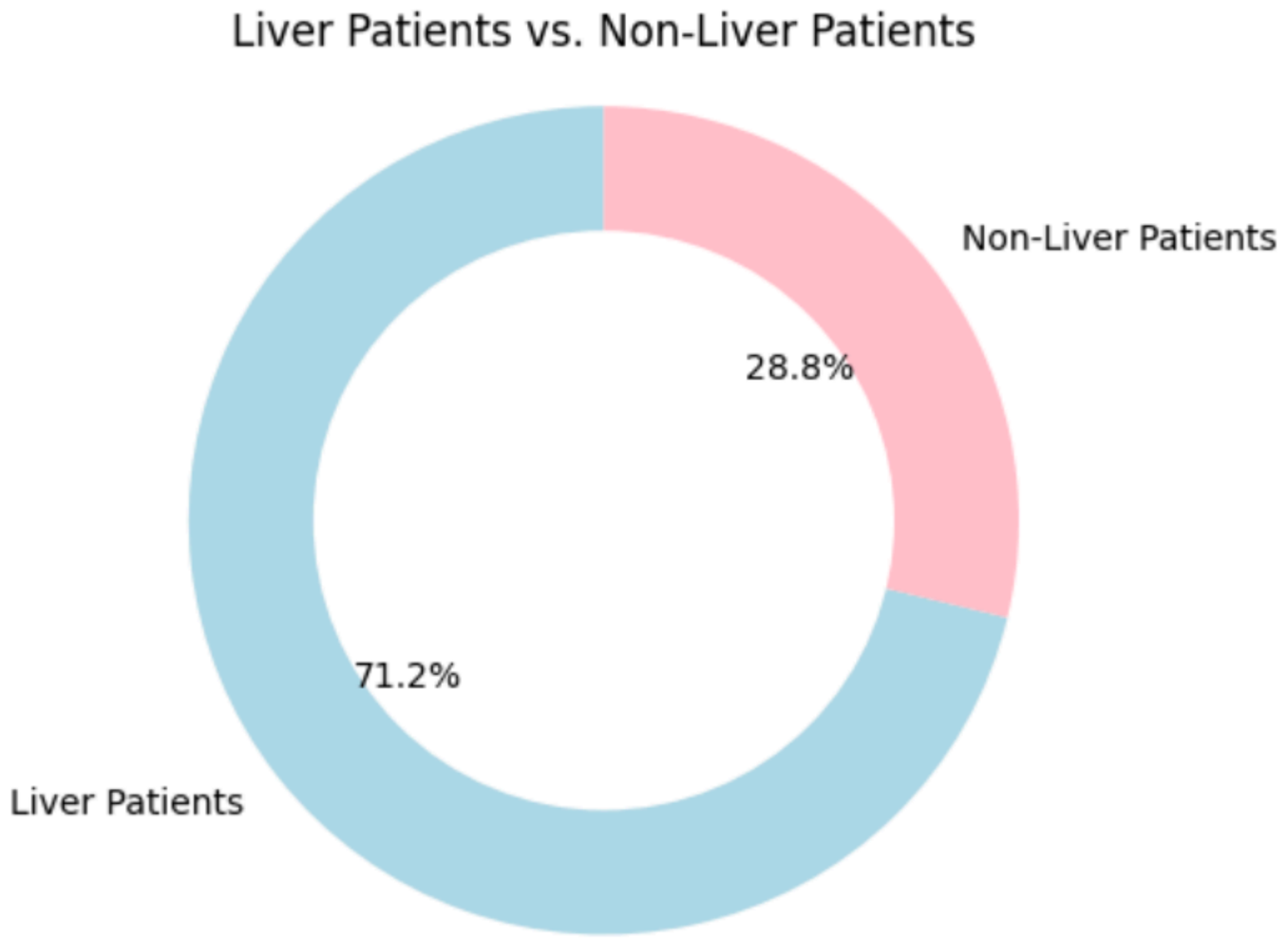
```
Contingency Table:
Selector    1   2
Age
Adults    143  47
Children  122  70
Seniors   141  47

Chi-Square Statistic: 8.349810014417876
Degrees of Freedom: 2
Expected Frequencies:
[[135.33333333  54.66666667]
 [136.75789474  55.24210526]
 [133.90877193  54.09122807]]
```

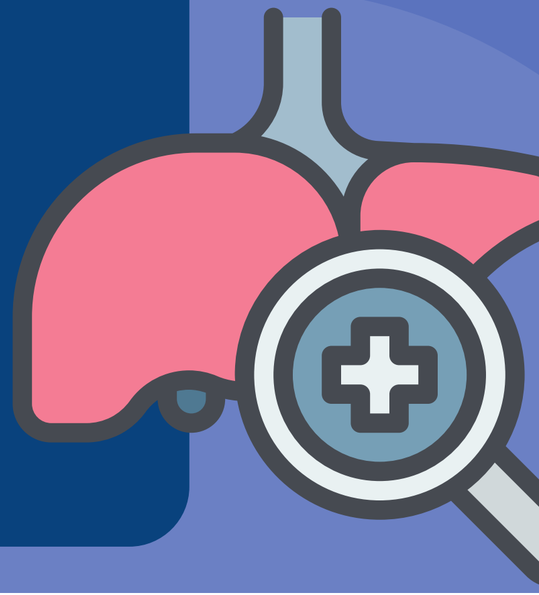
Data mining techniques:

We applied both supervised and unsupervised learning techniques on our data set.

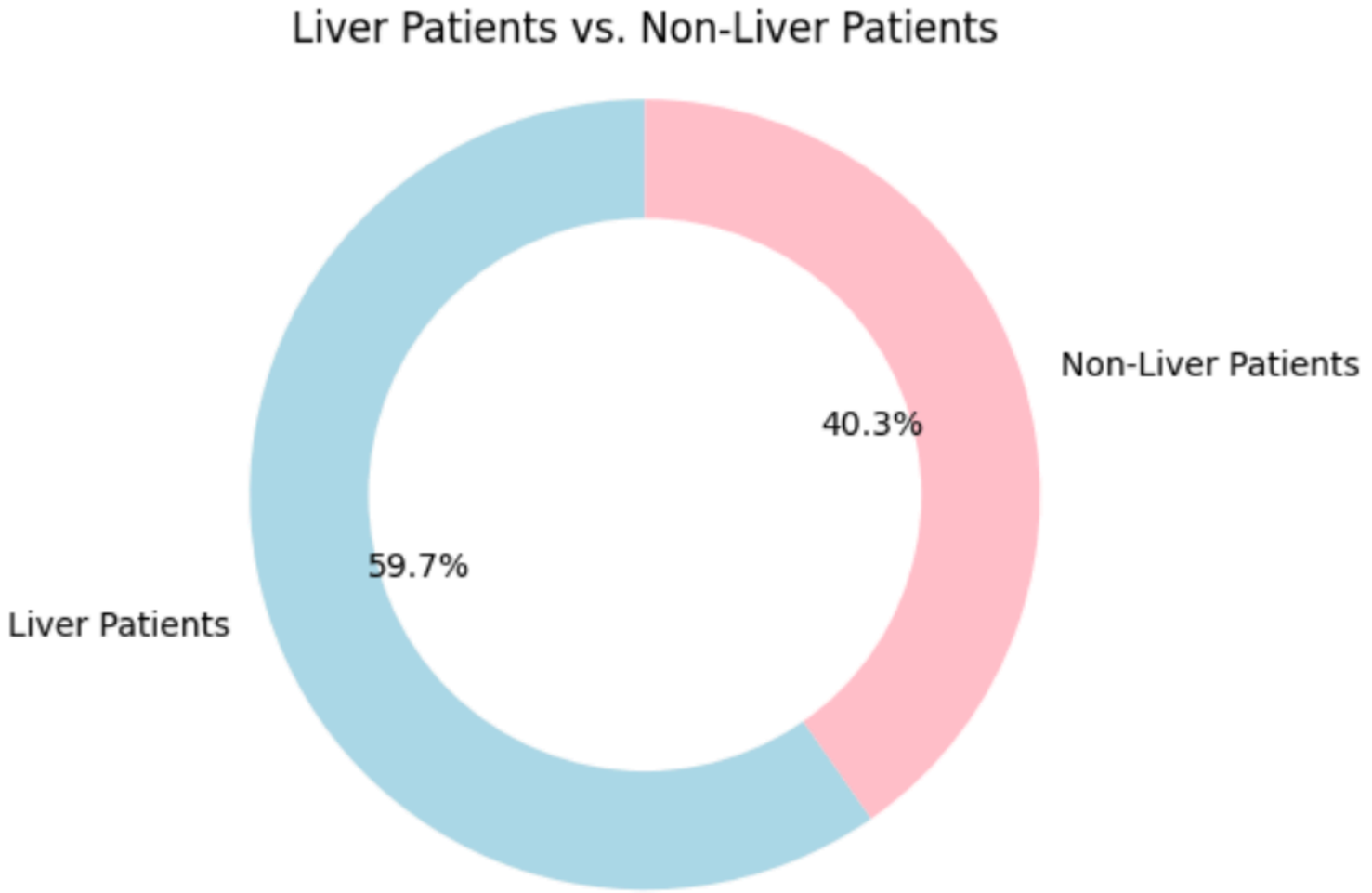
- Classification
- Clustering

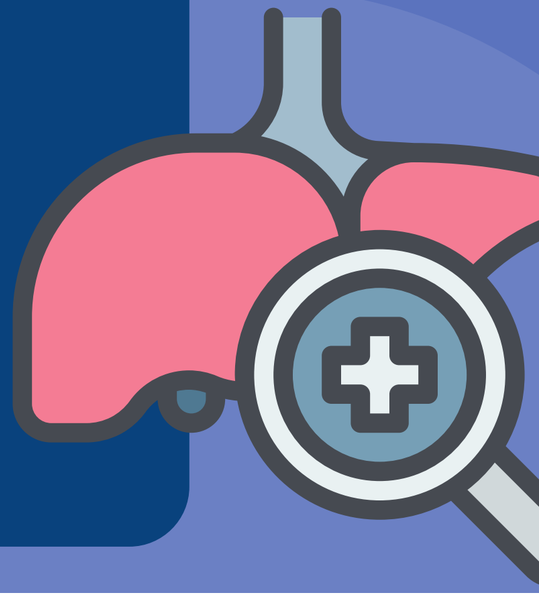
Before Balance Data





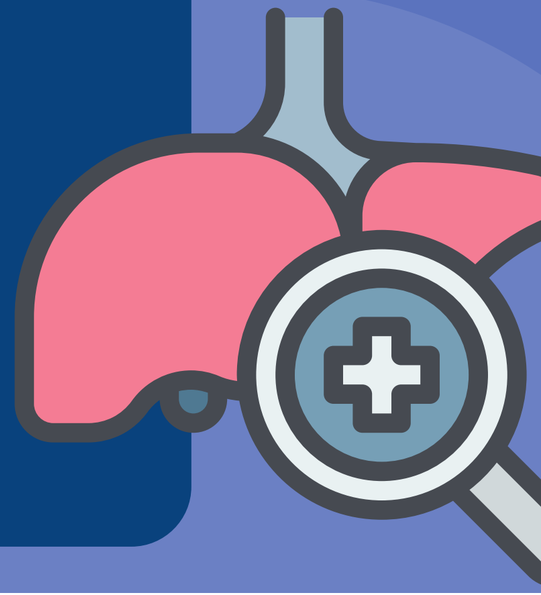
After Balance Data





Classification

classification is a type of supervised learning where the goal is to predict the categorical class labels of new instances, based on past observations.



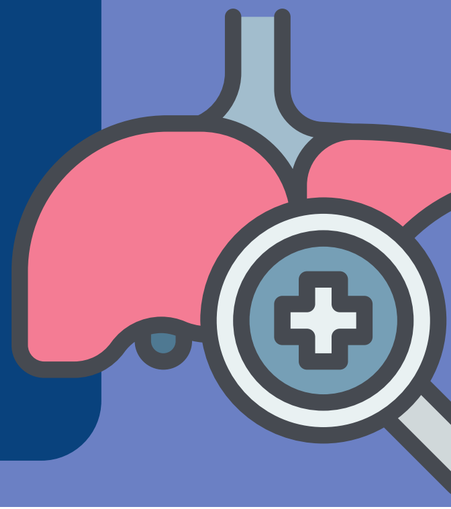
Classification

Dividing our dataset into training and testing sets:

- **70% TRAINING**
  30% TESTING
- **60% TRAINING**
  40% TESTING
- **80% TRAINING**
  20% TESTING

Appled different attribute selection measures:

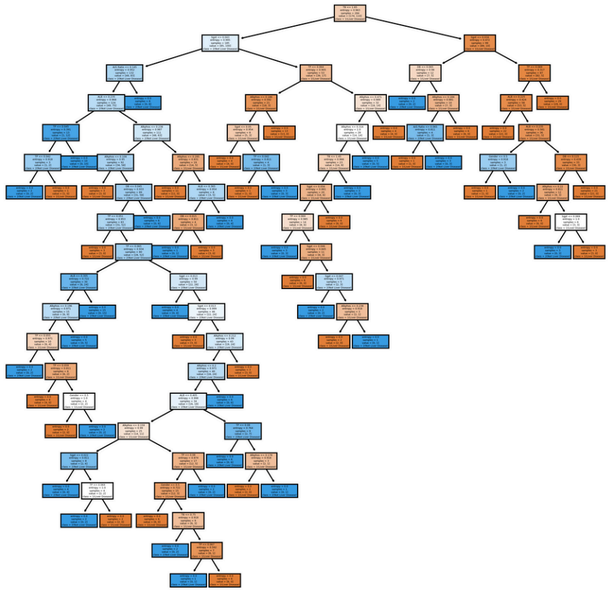
- INFORMATION GAIN
- GINI INDEX



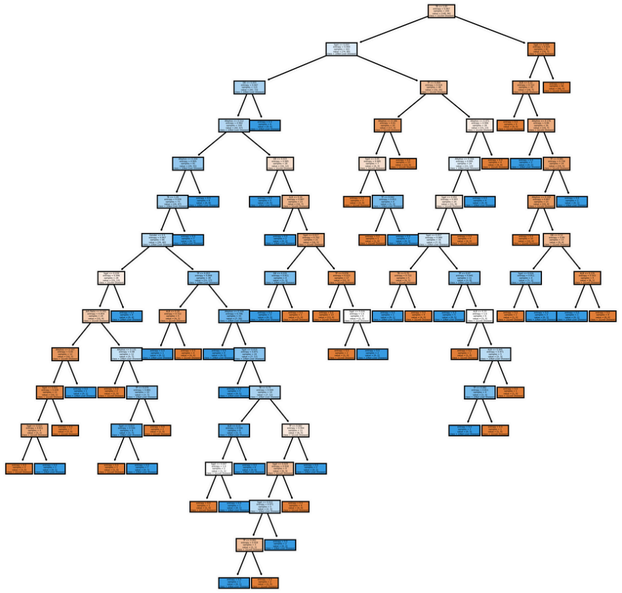
Information Gain

Decision Tree :

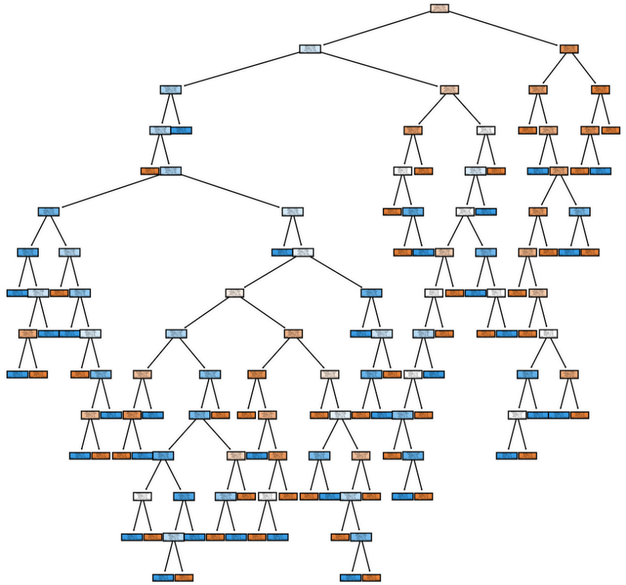
70% Training and 30% testing



60% Training and 40% testing



80% Training and 20% testing



Information Gain

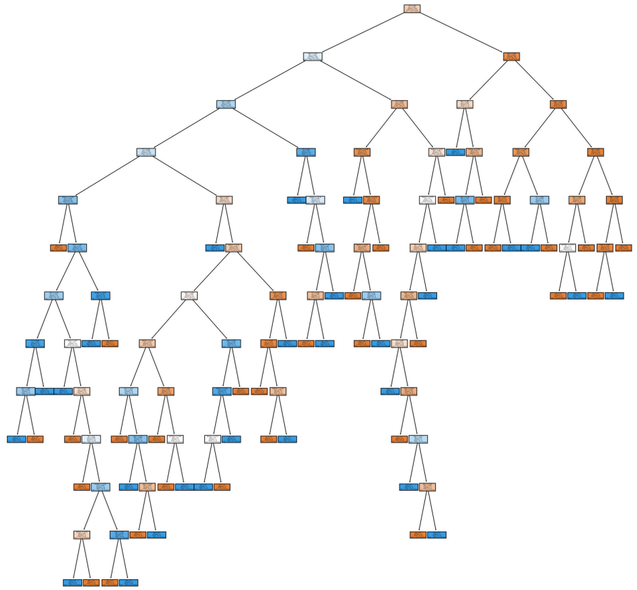
Evaluating the model :

	70% training, 30% testing	60% training, 40% testing	80% training, 20% testing
Accuracy	0.6504065040650406	0.6257668711656442	0.6219512195121951
Error Rate	0.3495934959349593	0.3742331288343558	0.3780487804878049
Sensitivity	0.48148148148148145	0.47058823529411764	0.3870967741935484
Specificity	0.782608695652174	0.7368421052631579	0.7647058823529411
Precision	0.6341463414634146	0.5614035087719298	0.5

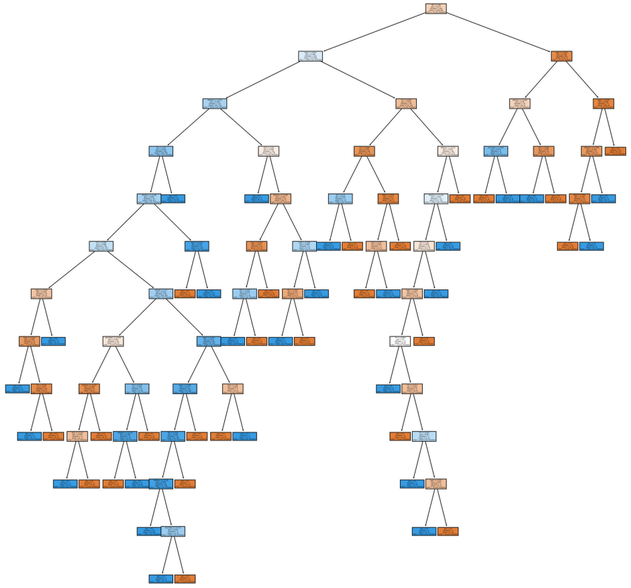
Gini Index

Decision Tree :

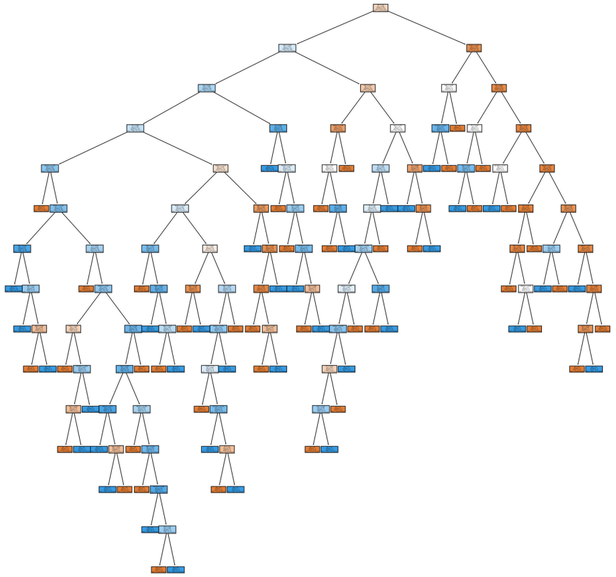
70% Training and 30% testing



60% Training and 40% testing



80% Training and 20% testing



Gini Index

Evaluating the model :

	70% training, 30% testing	60% training, 40% testing	80% training, 20% testing
Accuracy	0.6422764227642277	0.6422764227642277	0.6707317073170732
Error Rate	0.3577235772357723	0.3619631901840491	0.3292682926829268
Sensitivity	0.48148148148148145	0.5147058823529411	0.4838709677419355
Specificity	0.7681159420289855	0.7263157894736842	0.7843137254901961
Precision	0.6190476190476191	0.5737704918032787	0.5769230769230769

Gini Index

Evaluating the model :

After we compare betwwen three different sizes of partitions and two attribute selection measures (IG (entropy) Gini index) we found :

the 80-20 split model is considered the best based on the presented results.

HIGHEST ACCURACY

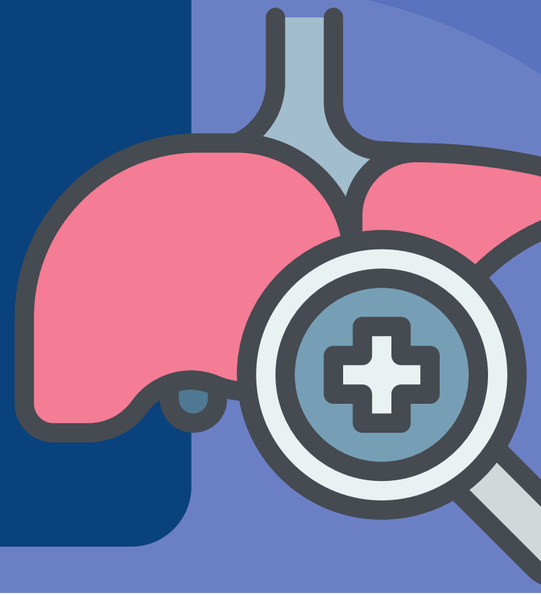
HIGHEST SPECIFICITY

LOWEST ERROR RATE

BALANCED BETWEEN SENSITIVITY AND PRECISION

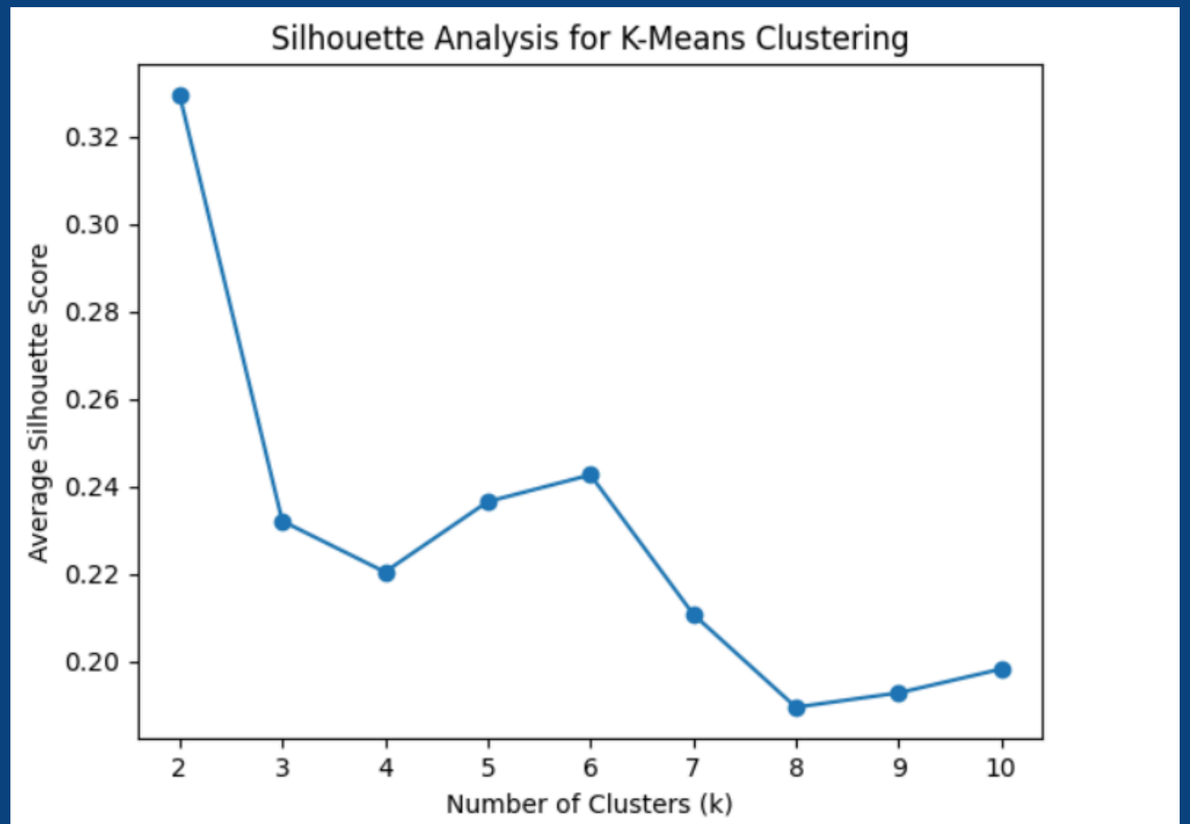
CLUSTERING

CLUSTERING GROUPS SIMILAR DATA POINTS INTO CLUSTERS BASED ON INTRINSIC CHARACTERISTICS WITHOUT PREDEFINED LABELS.

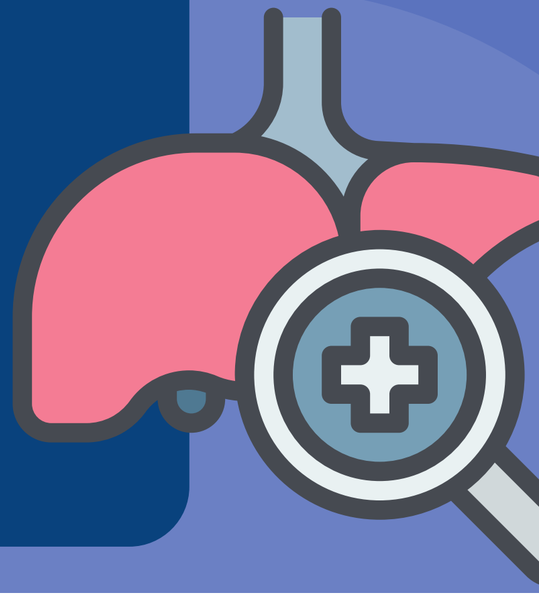


K-means Clustering

THE CHOICE FOR THE THREE DIFFERENT SIZES OF K-MEANS CLUSTERING :

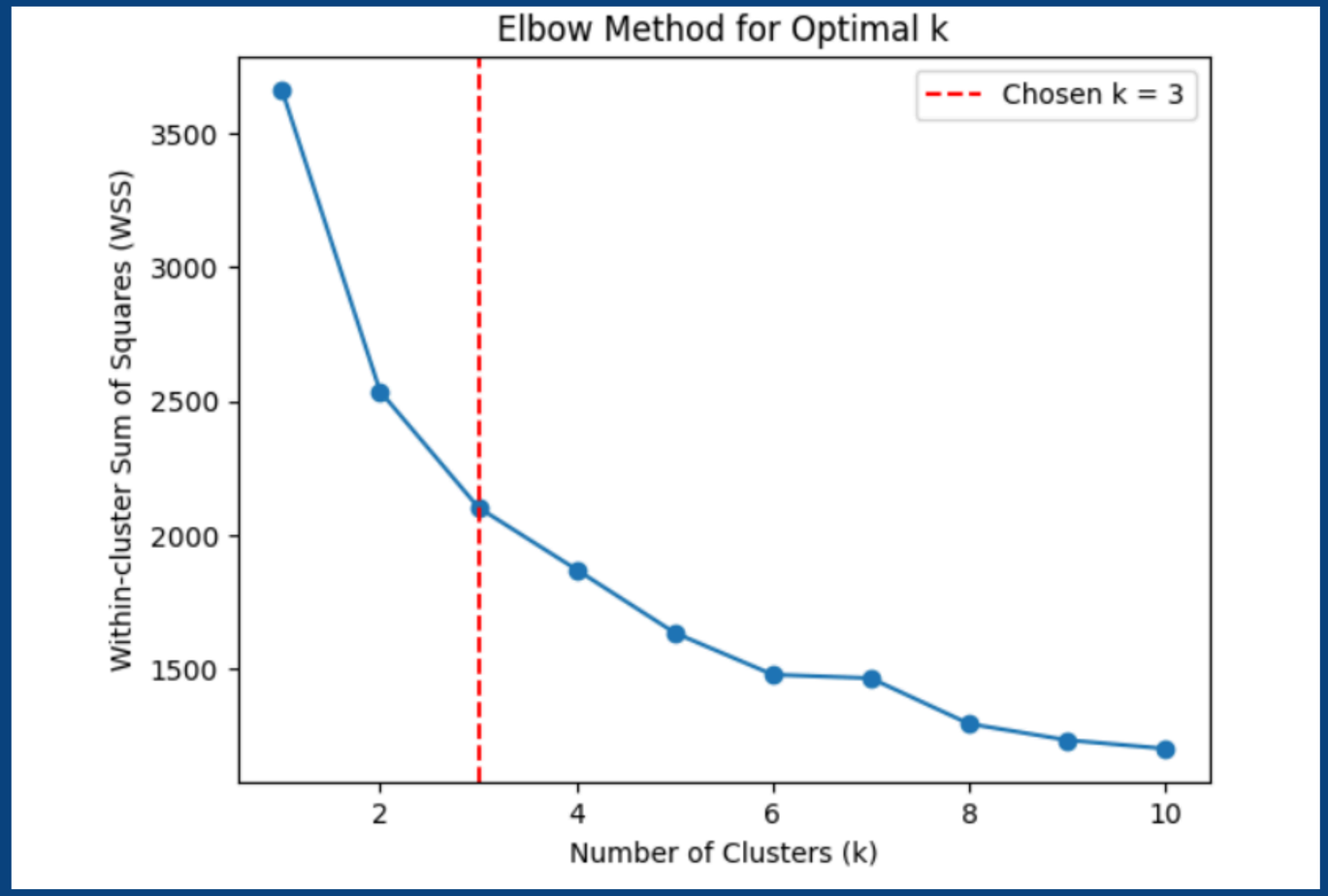



```
The highest average Silhouette score is 0.32946210901516015 with k=2.
The second highest average Silhouette score is 0.24255393014329987 with k=6.
```



K-means Clustering

THE CHOICE FOR THE THREE DIFFERENT SIZES OF K-MEANS CLUSTERING :

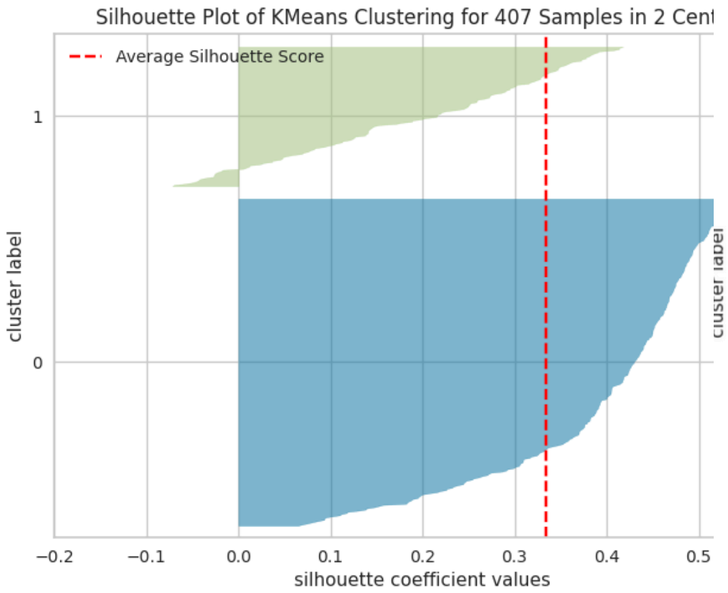


THE RESULTS

K	K=2	K=3	K=6
WSS	2537.0	2125.6	1536.79
Average Silhouette Score	0.329	0.232	0.2427

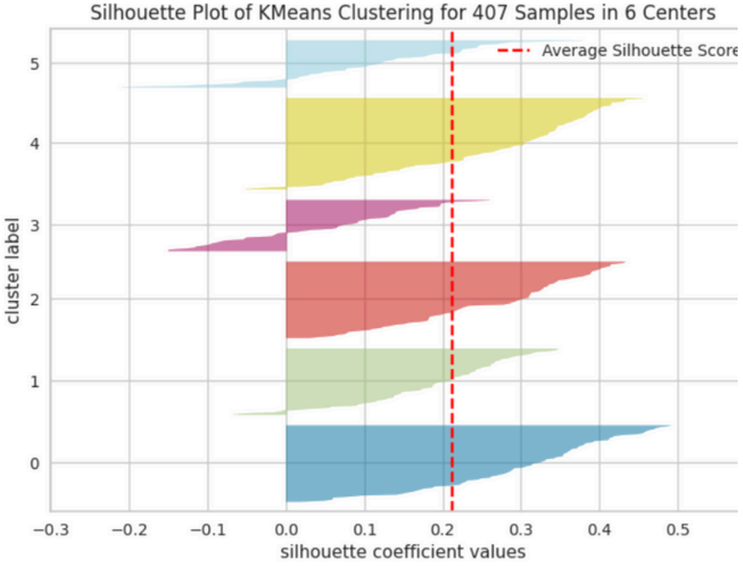
COMPARING BETWEEN VISUALIZATION RESULTS

K=2



K=3

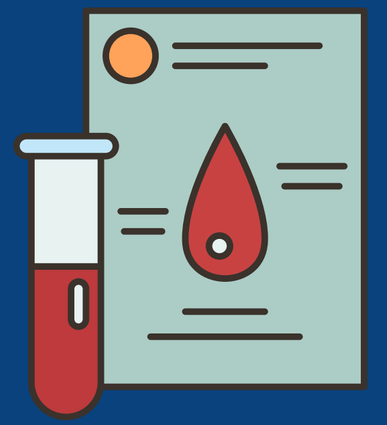
K=6



FINDINGS AND RESULTS

FROM THE PREVIOUS INFORMATIONS CLASSIFICATION WAS CONSIDERED THE BEST OPTION TO PREDICT THE POSSIBILITIES OF HAVING LIVER DISEASEBASE ON THE ATTRIBUTES, SINCE:

- DATA SET INCLUDES CLASS LABEL WHICH IS OUR TARGET WHICH GIVE US MORE ACCURATE PREDICTIONS .
- CLASSIFICATION MODELS OFFER INTERPRETABLE RESULTS, REVEALING THE MOST INFLUENTIAL FEATURES FOR PREDICTING SUCH AS: TB,SGOT.



REFERENCE

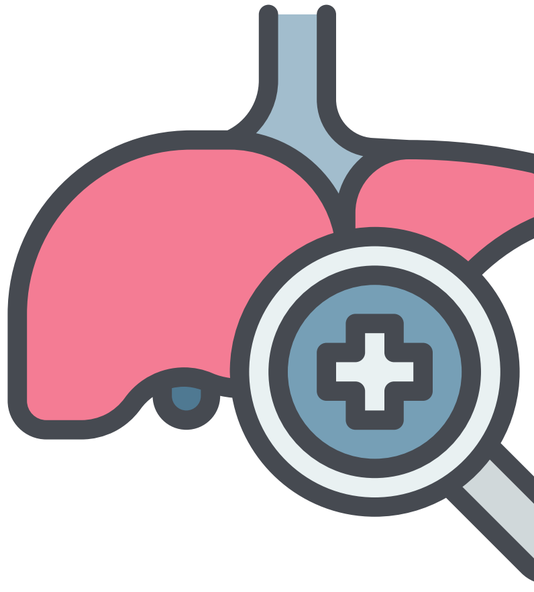
- FATEMEH MEHRPARVAR, "LIVER DISORDERS DATASET", KAGGLE, AVAILABLE: HTTPS://WWW.KAGGLE.COM/DATASETS/FATEMEHMEHRPARVAR/LIVER-DISORDERS

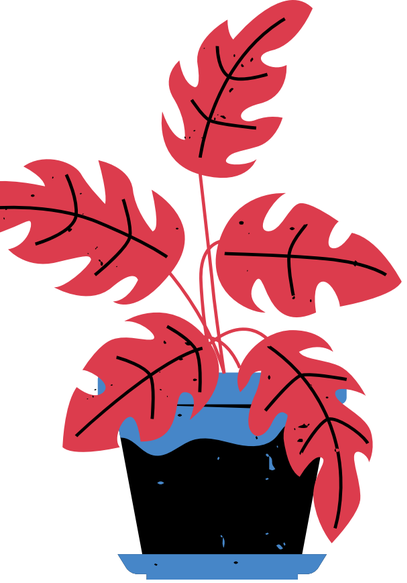
- "LIVER DISEASE ACCOUNTS FOR TWO MILLION ANNUAL DEATHS GLOBALLY: THE NEED FOR JOINT AND ROBUST POLICY AND PRACTICES ACROSS LIVER DISEASES." PUBMED, AVAILABLE: HTTPS://PUBMED.NCBI.NLM.NIH.GOV/36990226/#:~:TEXT=LIVER%20DISEASE%20ACCOUNTS%20 FOR%20TWO,RELATED%20DEATHS%20OCCUR%20IN%20MEN.

- AMERICAN LIVER FOUNDATION, "HOW MANY PEOPLE HAVE LIVER DISEASE?" AVAILABLE: HTTPS://LIVERFOUNDATION.ORG/ABOUT-YOUR-LIVER/FACTS-ABOUT-LIVER-DISEASE/HOW-MANY-PEOPLEHAVE-LIVER-DISEASE/

- “LABS AND LECTURE SLIDES," COLLEGE OF COMPUTER SCIENCE, DEPARTMENT OF INFORMATION TECHNOLOGY, KING SAUD UNIVERSITY

- WORLD HEALTH ORGANIZATION, "HEPATITIS B," WORLD HEALTH ORGANIZATION, AVAILABLE: HTTPS://WWW.WHO.INT/NEWS-ROOM/FACT-SHEETS/DETAIL/HEPATITIS-B. ACCESSED: MAY 7, 2024.





Any questions?