```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split, KFold
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import RobustScaler, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import f1_score, confusion_matrix, ConfusionMatrixDisplay
```

```
df = pd.read_csv("/content/drive/MyDrive/HCA Project Datasets/indian_liver_patient.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         583 non-null    int64
 1   Gender                      583 non-null    object
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64
 5   Alamine_Aminotransferase    583 non-null    int64
 6   Aspartate_Aminotransferase  583 non-null    int64
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Dataset                     583 non-null    int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

```
df.describe()
```

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	To
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	

```
SS
```

```
13
```

```
df.drop_duplicates(inplace = True)
df.duplicated().sum()
```

```
0
```

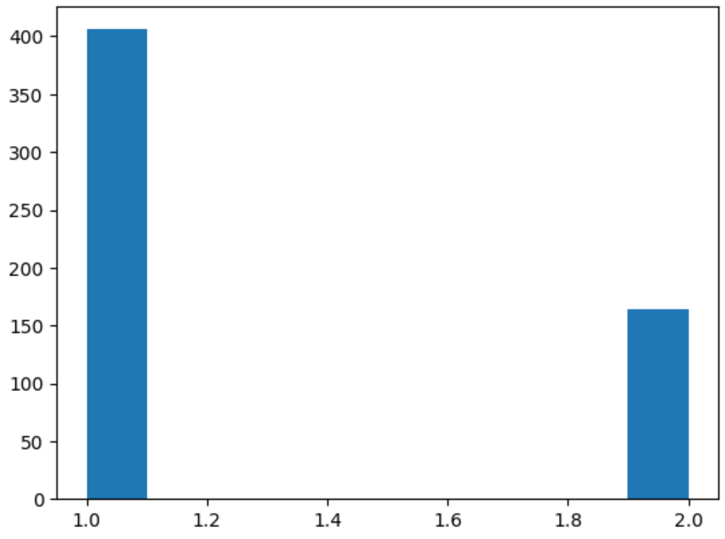
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 570 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         570 non-null    int64
 1   Gender                      570 non-null    object
 2   Total_Bilirubin             570 non-null    float64
 3   Direct_Bilirubin            570 non-null    float64
 4   Alkaline_Phosphotase        570 non-null    int64
 5   Alamine_Aminotransferase    570 non-null    int64
 6   Aspartate_Aminotransferase  570 non-null    int64
 7   Total_Protiens              570 non-null    float64
 8   Albumin                     570 non-null    float64
 9   Albumin_and_Globulin_Ratio  566 non-null    float64
 10  Dataset                     570 non-null    int64
```

```
dtypes: float64(5), int64(5), object(1)
memory usage: 53.4+ KB
```

```
plt.hist(df.Dataset)
```

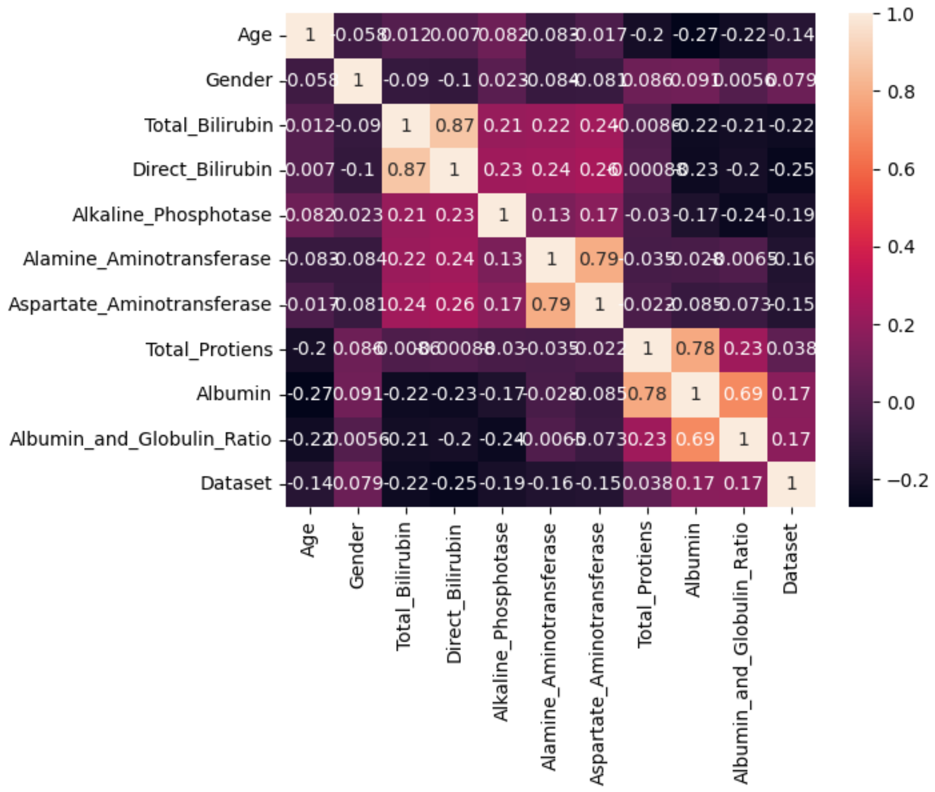
```
(array([406.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 164.]),
 array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ]),
 <BarContainer object of 10 artists>)
```



```
df.replace({"Male":0,"Female": 1}, inplace = True)
```

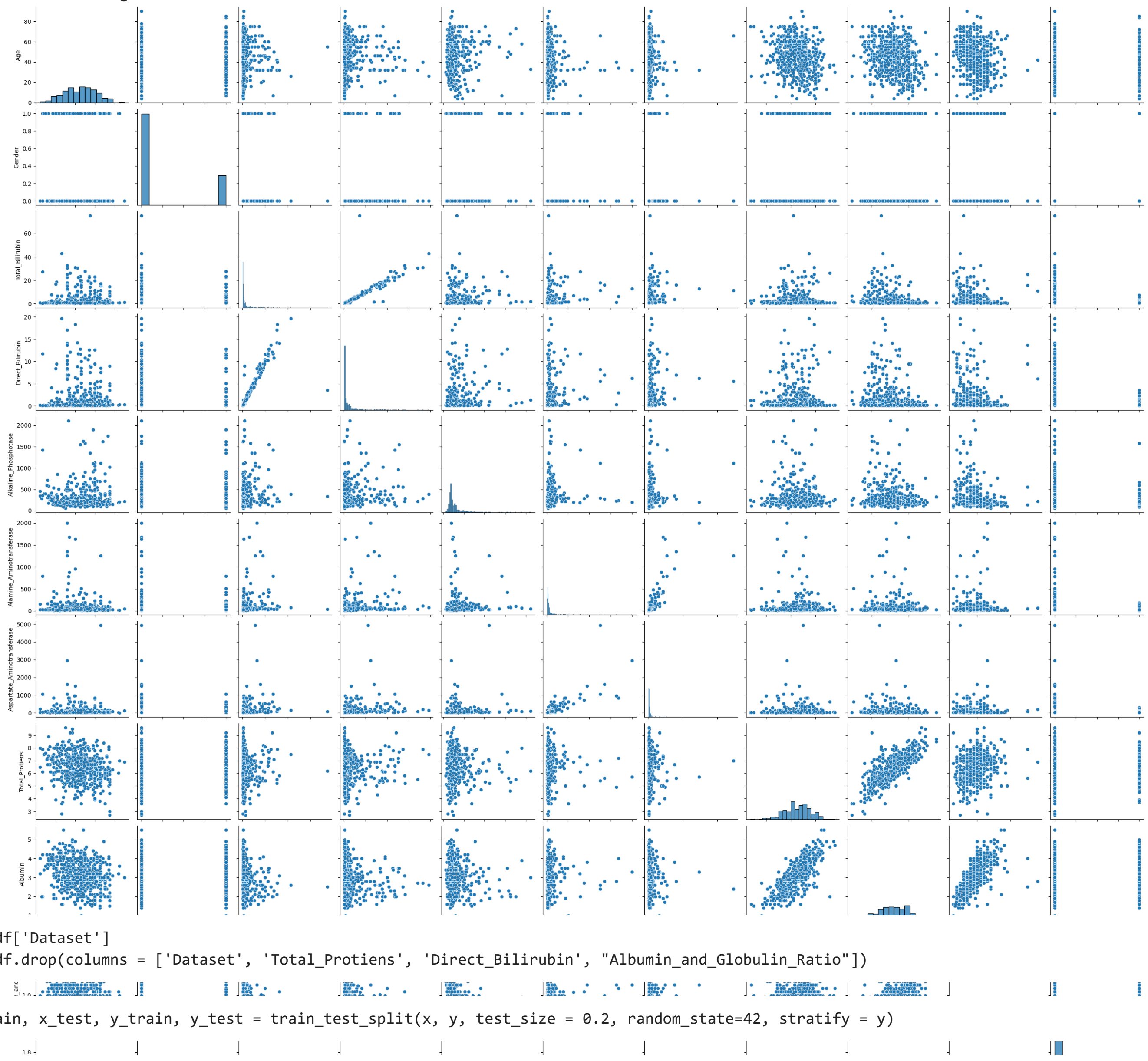
```
sns.heatmap(df.corr(), annot = True)
```

```
<Axes: >
```



```
sns.pairplot(df)
```

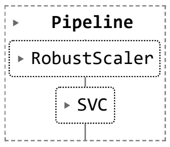
```
<seaborn.axisgrid.PairGrid at 0x7f28c2838610>
```



```
y = df['Dataset']
x = df.drop(columns = ['Dataset', 'Total_Protiens', 'Direct_Bilirubin', "Albumin_and_Globulin_Ratio"])
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=42, stratify = y)
```

```
clf = make_pipeline(RobustScaler(), SVC(gamma='auto'))
clf.fit(x_train, y_train)
```

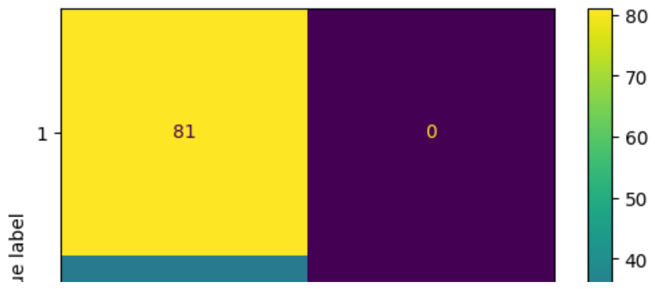


```
y_pred = clf.predict(x_test)
f1_score(y_test, y_pred)
```

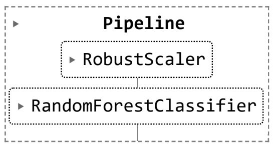
```
0.8307692307692308
```

```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=clf.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f28c2838370>
```



```
from sklearn.ensemble import RandomForestClassifier
dt_clf = make_pipeline(RobustScaler(), RandomForestClassifier(max_depth = 4, random_state = 20))
dt_clf.fit(x_train, y_train)
```



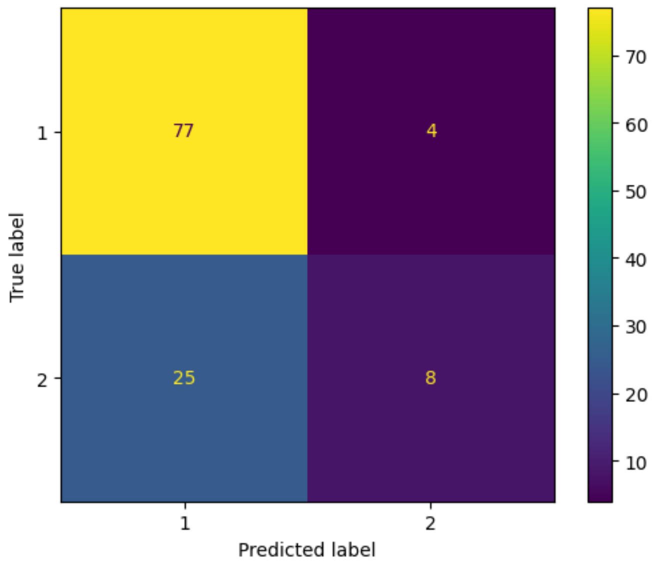
```
Pipeline
  RobustScaler
  RandomForestClassifier
```

```
y_pred_dt = dt_clf.predict(x_test)
f1_score(y_test, y_pred_dt)
```

```
0.8415300546448087
```

```
cm = confusion_matrix(y_test, y_pred_dt)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=dt_clf.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f28b7a1bf70>
```



```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X, Y = oversample.fit_resample(x, y)
```

```
Y.value_counts()
```

```
1    406
2    406
Name: Dataset, dtype: int64
```

```
x_train_smote, x_test_smote, y_train_smote, y_test_smote = train_test_split(X, Y, test_size = 0.2, random_state = 30, shuffle = True)
```

```
dt_clf_smote = make_pipeline(RobustScaler(), RandomForestClassifier(max_depth = 10, random_state = 32))
dt_clf_smote.fit(x_train_smote, y_train_smote)
```

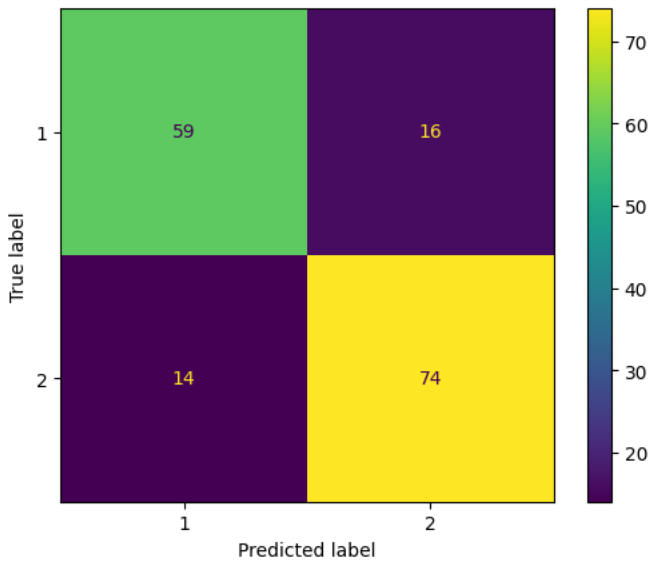
```
Pipeline
```

```
y_pred_smote = dt_clf_smote.predict(x_test_smote)
f1_score(y_test_smote, y_pred_smote, average = "micro")
```

```
0.8159509202453987
```

```
cm = confusion_matrix(y_test_smote, y_pred_smote)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=dt_clf_smote.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f28b562fe50>
```


True label
Predicted label