Sheet

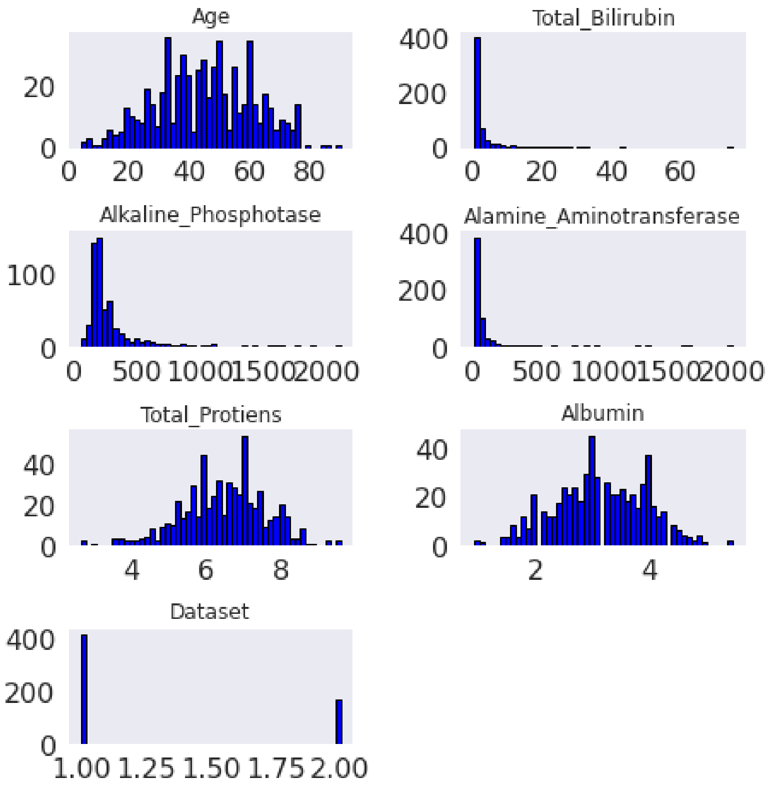
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn')
```

```
dataset = pd.read_csv("./Liver.csv")
```

```
dataset.info()
```

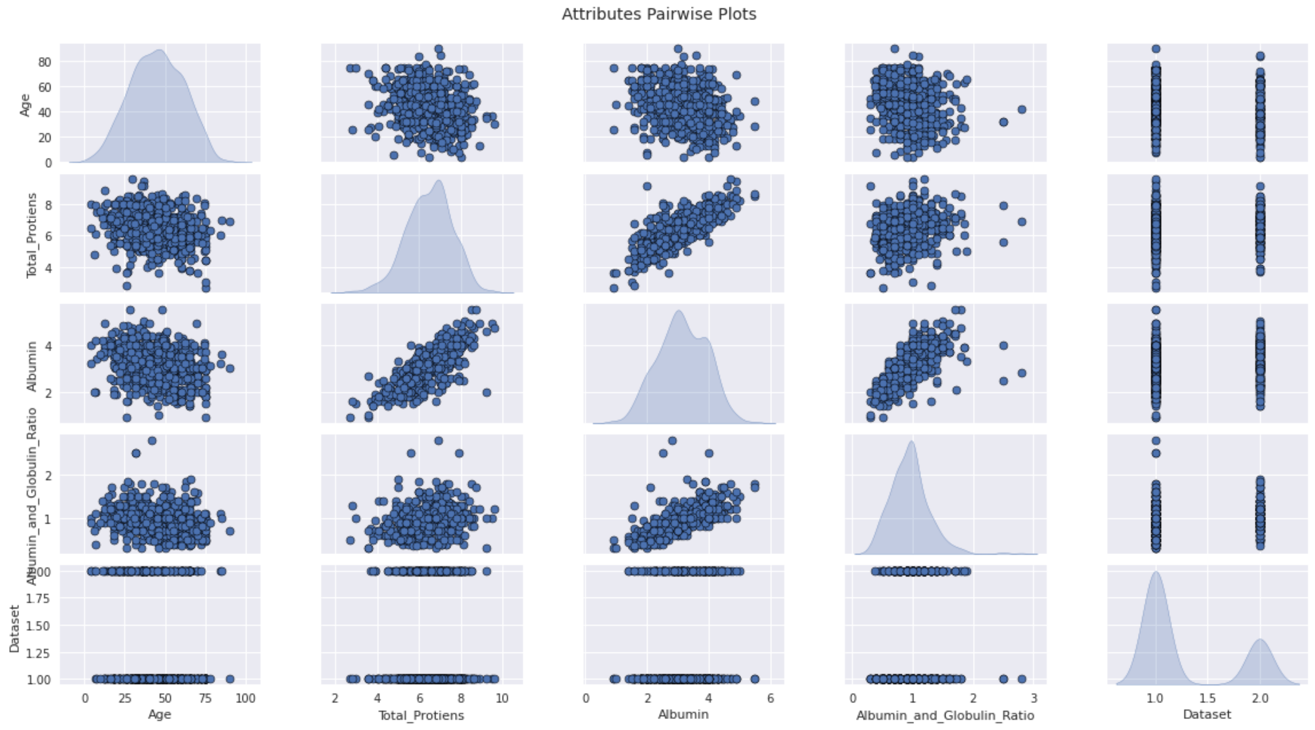
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         583 non-null    int64  
 1   Gender                      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
```

```
dataset.hist(bins=50, color='blue', edgecolor='black', linewidth=1.0,
           xlabelsize=16, ylabelsize=16, grid=False)
plt.tight_layout(rect=(0, 0, 1.2, 1.2))
```



```
# Pair-wise Scatter Plots
cols = ['Age', 'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio',
pp = sns.pairplot(dataset[cols], height=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

fig = pp.fig
fig.subplots_adjust(top=0.93, wspace=0.3)
t = fig.suptitle('Attributes Pairwise Plots', fontsize=14)
```



```
dataset.isnull().sum()
```

```
# Filling Null values
```

```
dataset['Albumin_and_Globulin_Ratio'].fillna(dataset['Albumin_and_Globuli
```

```
# Changing Male and Female to class 0/1
```

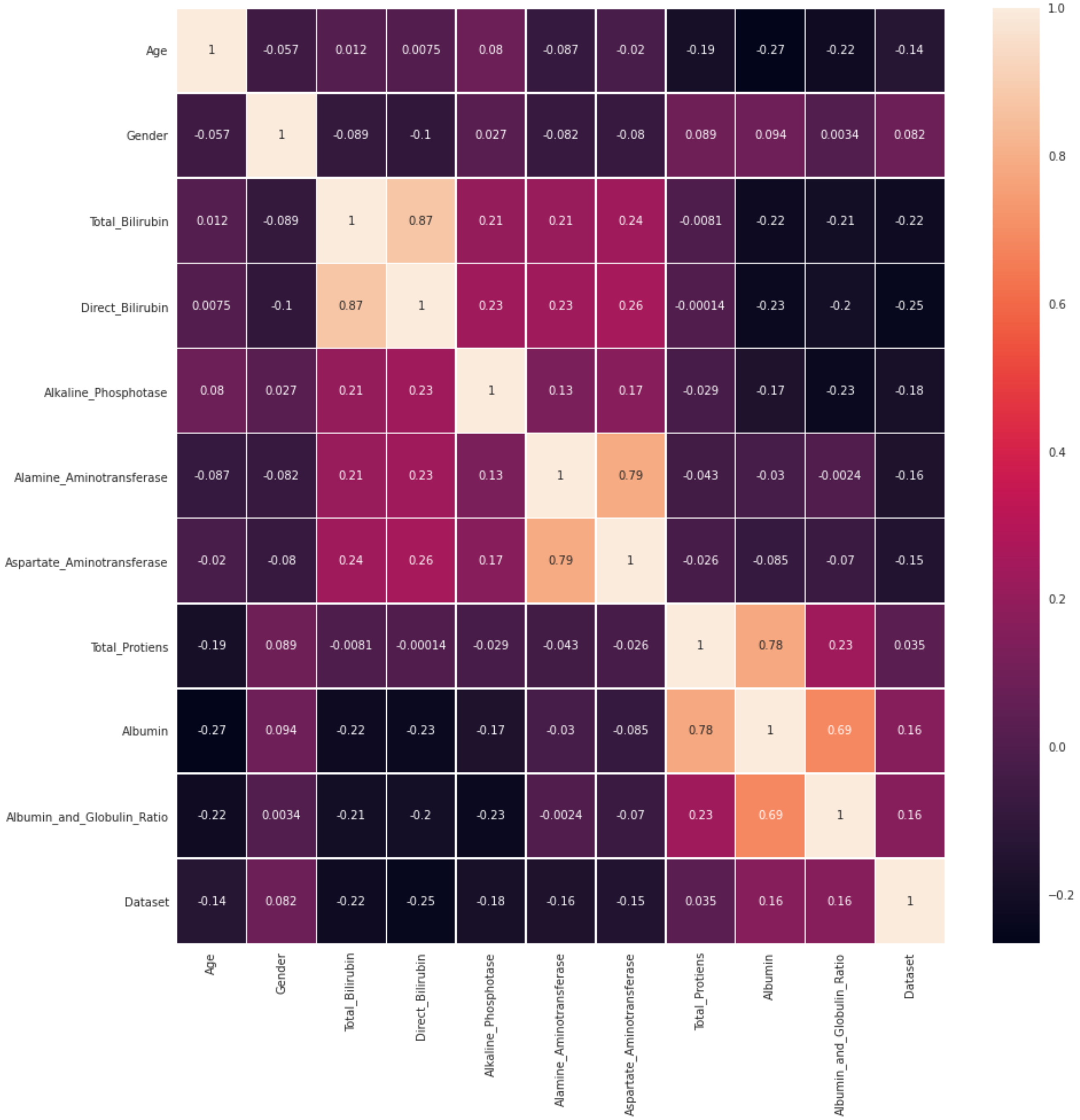
```
dataset['Gender'].replace('Female',1,inplace=True)
dataset['Gender'].replace('Male',0,inplace=True)
dataset.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferas
0	65	1	0.7	0.1	187	16
1	62	0	10.9	5.5	699	64
2	62	0	7.3	4.1	490	60
3	58	0	1.0	0.4	182	14
4	72	0	3.9	2.0	195	27

```
dataset.describe()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.0000
mean	44.746141	0.243568	3.298799	1.486106	290.576329	80.71355
std	16.189833	0.429603	6.209522	2.808498	242.937989	182.6203
min	4.000000	0.000000	0.400000	0.100000	63.000000	10.00000
25%	33.000000	0.000000	0.800000	0.200000	175.500000	23.00000
50%	45.000000	0.000000	1.000000	0.300000	208.000000	35.00000
75%	58.000000	0.000000	2.600000	1.300000	298.000000	60.50000
max	90.000000	1.000000	75.000000	19.700000	2110.000000	2000.000

```
import seaborn as sns
df = dataset.corr()
plt.figure(figsize=(15, 15))
sns.heatmap(df, vmax=1, annot=True, linewidths=.5)
plt.show()
```



Actually we can remove more than 2 features from this dataset Since they don't have higher impact on our y value.

```
dataset.shape,type(dataset)
```

```
((583, 11), pandas.core.frame.DataFrame)
```

```
x = dataset.iloc[:,:-1]
y = dataset.iloc[:,-1]
```

```
x.shape,y.shape
```

```
((583, 10), (583,))
```

```
# Spliting the data
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25)
```

Using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
Lr = LogisticRegression(max_iter=1000)
Lr.fit(x_train,y_train)
```

```
LogisticRegression(max_iter=1000)
```

```
y_pred = Lr.predict(x_test)

from sklearn.metrics import accuracy_score
LR_accuracy = accuracy_score(y_test,y_pred)
LR_accuracy
```

```
0.7328767123287672
```

Using Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dT = DecisionTreeClassifier(max_depth=10)
```

```
dT.fit(x_train,y_train)
```

```
DecisionTreeClassifier(max_depth=10)
```

```
dT_pred = dT.predict(x_test)
```

```
dT_accuracy = accuracy_score(y_test,dT_pred)
dT_accuracy
```

```
0.636986301369863
```

support vector machine

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(x_train,y_train)
```

```
SVC()
```

```
svc_pred = svc.predict(x_test)
```

```
svc_accuracy = accuracy_score(y_test,svc_pred)
svc_accuracy
```

```
0.7397260273972602
```