Dhruv Singhal

500075346 | R177219074

Course - B.Tech CSE AI - ML (B-3 Sem5)

Application of ML in Industries LAB Exp Number 6

Exploration, Visualisation, Preprocessing, Various Classifiers on Indian Liver Patients Dataset

In [1]:
```
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#reading Dataset
data=pd.read_csv("indian_liver_patient.csv")
data.head()
```

Out[2]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

In [3]:
```
#Data Exploration, Visualisation & Preprocessing
data.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
```

In [4]:

```
#checking null values in a column
data.isnull().sum()
```

Out[4]:

```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  4
Dataset                     0
dtype: int64
```

In [5]:

```
#converting string data to int and filling NaN of Ailbumin and Globulin_ration
data['Gender'] = data['Gender'].replace('Male', 0)
data['Gender'] = data['Gender'].replace('Female', 1)
data['Albumin_and_Globulin_Ratio'].fillna(int(data['Albumin_and_Globulin_Ratio'].mean()), inplace=True)
print(data.info())
print(data.isnull().sum())
```

```
<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    int64
 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  583 non-null    float64
 10  Dataset                     583 non-null    int64
dtypes: float64(5), int64(6)
memory usage: 50.2 KB
None
```

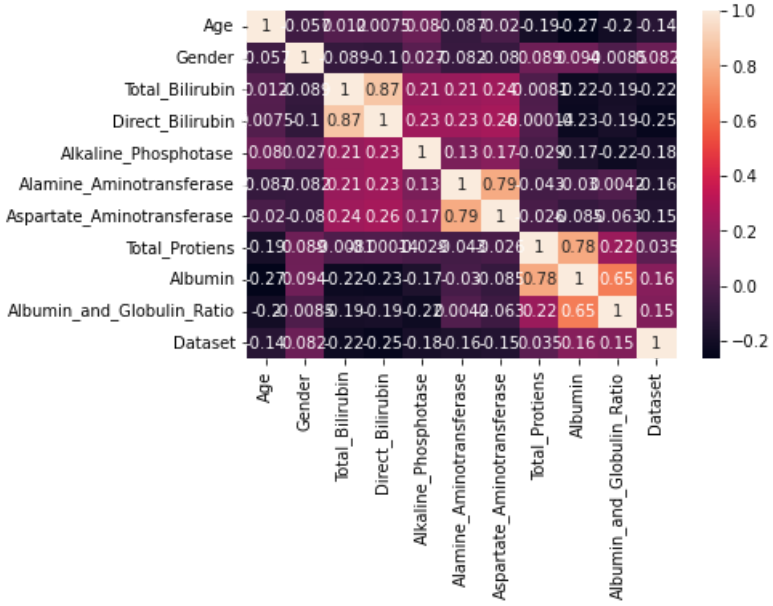
```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  0
Dataset                     0
dtype: int64
```

In [6]:

```
#making heatmap to fing correlation
sns.heatmap(data.corr(),annot=True)
```

Out[6]:

```
<AxesSubplot:>
```



In [7]:

```
print("Distribution in dataset =>")
print(data["Dataset"].value_counts())
```

```
Distribution in dataset =>
1    416
2    167
Name: Dataset, dtype: int64
```

In [8]:

```
#splitting of feature and target variable
X=data.iloc[:,0:10]
```

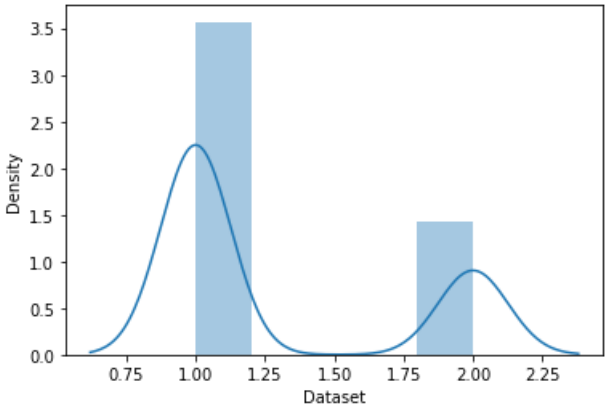
```
Y=data.iloc[:,10]
Y.tail()
```

Out[8]:
```
578    2
579    1
580    1
581    1
582    2
Name: Dataset, dtype: int64
```

In [9]:
```
#scaling the features
from sklearn.preprocessing import StandardScaler
StandSc = StandardScaler()
StandSc.fit(X)
X_new = StandSc.transform(X)
```

In [10]:
```
print(sns.distplot(Y))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [11]:
```
#splitting data in training and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X_new,Y,train_size=0.175,random_state=0)
```

The numpy module of Python provides a function called numpy.ravel, which is used to change a 2-dimensional array or a multi-dimensional array into a contiguous flattened array. The returned array has the same data type as the source array or input array. If the input array is a masked array, the returned array will also be a masked array.

In [12]:
```
#Building the KNN Model on our dataset
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,Y_train)
```

Out[12]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [13]:
```
y_pred = knn.predict(X_test)
```

In [14]:
```
from sklearn.metrics import accuracy_score
print("Training Accuracy = ",knn.score(X_train, Y_train))
print("Testing Accuracy = ", accuracy_score(Y_test, y_pred))
```

```
Training Accuracy =  0.7843137254901961
Testing Accuracy =  0.6715176715176715
```

Other Models

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

In [16]:
```
forest_descision = RandomForestClassifier(max_depth=1, random_state=2)

SVC_model = SVC(kernel='linear')
```

In [17]:
```
forest_descision.fit(X_train, Y_train)
print("random forest: ", forest_descision.score(X_test, Y_test))

SVC_model.fit(X_train, Y_train)
print("SVM: ", SVC_model.score(X_test, Y_test))
```

```
random forest:  0.7130977130977131
SVM:  0.7110187110187111
```

In [ ]:

In [ ]:

In [ ]: