```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from scipy import stats
```

```
data = pd.read_csv("/content/indian_liver_patient.csv")
```

```
data.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotı
0	65	Female	0.7	0.1	187	
1	62	Male	10.9	5.5	699	
2	62	Male	7.3	4.1	490	
3	58	Male	1.0	0.4	182	
4	72	Male	3.9	2.0	195	

```
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
```

```
data.isnull().any()
```

```
Age                           False
Gender                        False
Total_Bilirubin               False
Direct_Bilirubin              False
Alkaline_Phosphotase          False
Alamine_Aminotransferase      False
Aspartate_Aminotransferase    False
Total_Protiens                False
Albumin                       False
Albumin_and_Globulin_Ratio     True
Dataset                       False
dtype: bool
```

```
data.isnull().sum()
```

```
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
```

```
from sklearn.preprocessing import LabelEncoder
lc= LabelEncoder()
data['Gender']= lc.fit_transform(data['Gender'])
```

```
data['Albumin_and_Globulin_Ratio'].fillna(data['Albumin_and_Globulin_Ratio'].mode()[0], inplace=True)
data.isnull().sum()
```

```
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
```

```
data.rename(columns={ 'Dataset': 'outcome' } , inplace=True)
```

```
data.head()
```

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

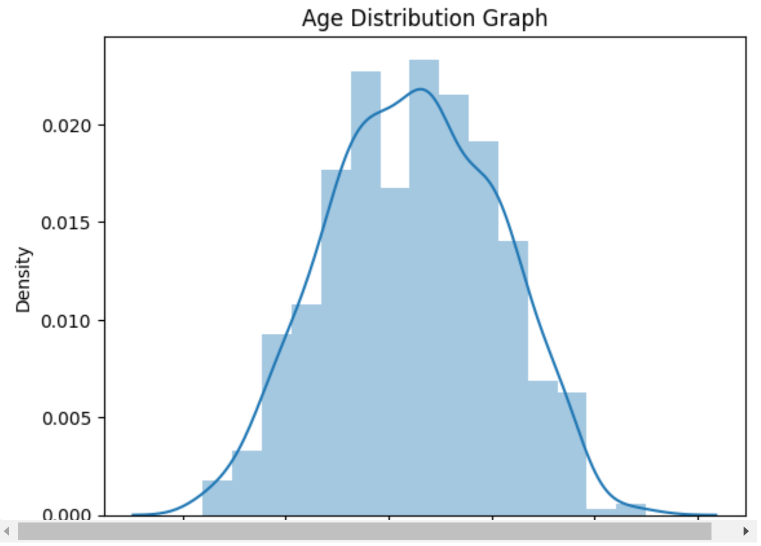
Exploratory Data Analysis

```
data.describe()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin
count	583.000000	583.000000	583.000000	583.000000
mean	44.746141	0.756432	3.298799	1.486106
std	16.189833	0.429603	6.209522	2.808498
min	4.000000	0.000000	0.400000	0.100000
25%	33.000000	1.000000	0.800000	0.200000
50%	45.000000	1.000000	1.000000	0.300000
75%	58.000000	1.000000	2.600000	1.300000

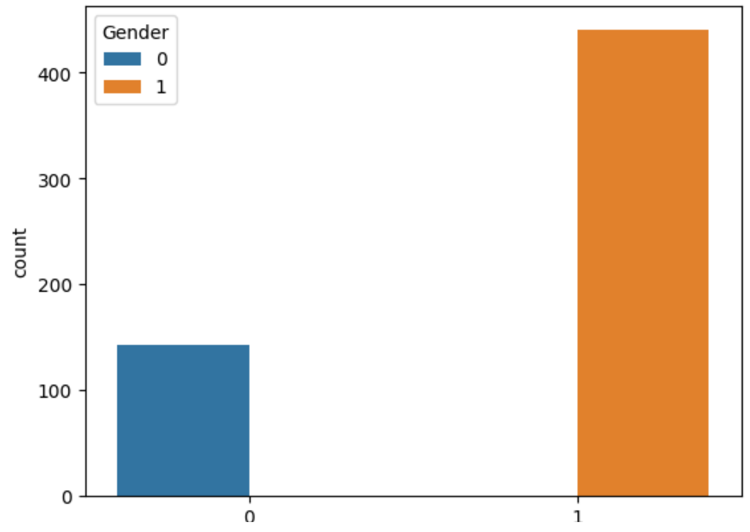
```
sns.distplot(data['Age'])
plt.title('Age Distribution Graph')
plt.show()
```

```
<ipython-input-190-a9533a3b6a8d>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.(

Please adapt your code to use either `displot` (a figure-level function wi
similar flexibility) or `histplot` (an axes-level function for histograms)

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Age'])
```



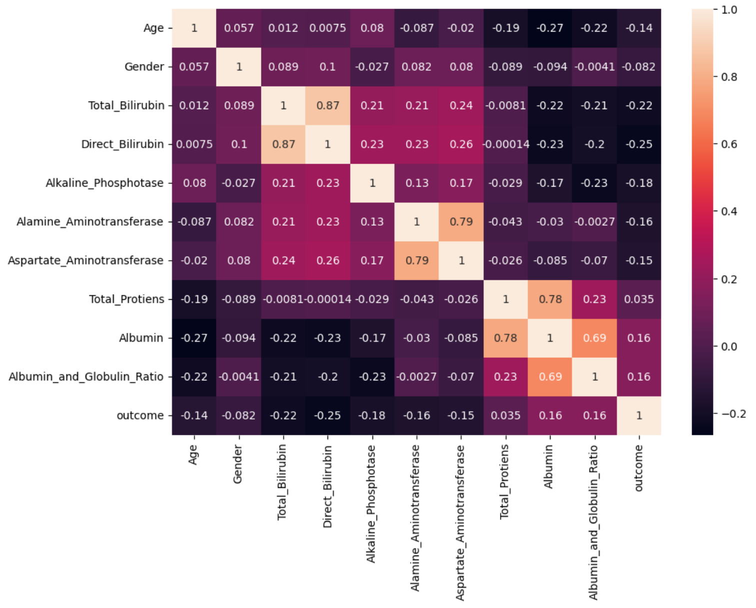
```
sns.countplot(x="Gender", hue="Gender", data=data)
```

```
<Axes: xlabel='Gender', ylabel='count'>
```



```
plt.figure(figsize=(10,7))
sns.heatmap(data.corr(),annot=True)
```

```
<Axes: >
```



```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame (scale(X), columns= X.columns)
```

```
X_scaled.head()
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phospho
0	1.252098	-1.762281	-0.418878	-0.493964	-0.42
1	1.066637	0.567446	1.225171	1.430423	1.68
2	1.066637	0.567446	0.644919	0.931508	0.82
3	0.819356	0.567446	-0.370523	-0.387054	-0.44
4	1.684839	0.567446	0.096902	0.183135	-0.39

```
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
578    2
579    1
580    1
581    1
582    2
Name: outcome, Length: 583, dtype: int64
```

```
y=data.outcome
X=data.iloc[:,:-1]
```

```
X
```

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase
0	65	0	0.7	0.1	187
1	62	1	10.9	5.5	699
2	62	1	7.3	4.1	490
3	58	1	1.0	0.4	182
4	72	1	3.9	2.0	195
...	...	...	...	...	...
578	60	1	0.5	0.1	500
579	40	1	0.6	0.1	98
580	52	1	0.8	0.2	245
581	31	1	1.3	0.5	184

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X ,y, test_size=0.2, random_state=42)
```

```
pip install imblearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: imblearn in /usr/local/lib/python3.9/dist-packages (0.0)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.9/dist-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.10.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.1.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.9/dist-packages (from imbalanced-learn->imblearn) (1.2.2)
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
```

```
y_train.value_counts()
```

```
1    329
2    137
Name: outcome, dtype: int64
```

```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

```
y_train_smote.value_counts()
```

```
1    329
2    329
Name: outcome, dtype: int64
```

Model Building

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import pandas as pd
model1=RandomForestClassifier()
model1.fit(X_train_smote, y_train_smote)
y_predict=model1.predict(X_test)
rfc1=accuracy_score(y_test,y_predict)
rfc1
pd.crosstab(y_test, y_predict)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.80      0.77      0.78        87
           2       0.39      0.43      0.41        30

    accuracy                           0.68       117
   macro avg       0.60      0.60      0.60       117
weighted avg       0.69      0.68      0.69       117
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call
last)
<ipython-input-193-8c4e61c2efdb> in <cell line: 1>()
----> 1 frrr

NameError: name 'frrr' is not defined
```

```
from sklearn.tree import DecisionTreeClassifier
model4=DecisionTreeClassifier ()
model4.fit(X_train_smote, y_train_smote)
y_predict=model4.predict(X_test)
dtc1=accuracy_score(y_test,y_predict)
dtc1
pd.crosstab(y_test,y_predict)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.80      0.68      0.73        87
           2       0.35      0.50      0.41        30

    accuracy                           0.63       117
   macro avg       0.57      0.59      0.57       117
weighted avg       0.68      0.63      0.65       117
```

```
from sklearn.neighbors import KNeighborsClassifier
model2=KNeighborsClassifier()
model2.fit (X_train_smote, y_train_smote)
y_predict = model2 . predict (X_test)
knn1=(accuracy_score (y_test, y_predict) )
knn1
pd.crosstab (y_test,y_predict)
print(classification_report (y_test, y_predict) )
```

```
              precision    recall  f1-score   support

           1       0.80      0.63      0.71        87
           2       0.33      0.53      0.41        30

    accuracy                           0.61       117
   macro avg       0.57      0.58      0.56       117
weighted avg       0.68      0.61      0.63       117
```

```
from sklearn.linear_model import LogisticRegression
model5=LogisticRegression()
model5.fit(X_train_smote, y_train_smote)
y_predict=model5.predict(X_test)
logil=accuracy_score (y_test, y_predict)
logil
pd.crosstab (y_test,y_predict)
print (classification_report (y_test, y_predict))
```

```
              precision    recall  f1-score   support

           1       0.96      0.63      0.76        87
           2       0.47      0.93      0.62        30

    accuracy                           0.71       117
   macro avg       0.72      0.78      0.69       117
weighted avg       0.84      0.71      0.73       117

/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      n_iter_i = _check_optimize_result(
```

```
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
classifier = Sequential()
classifier.add(Dense(units=100, activation='relu',input_dim=10))
classifier.add(Dense(units=50,activation='relu'))
classifier.add(Dense(units=1, activation='sigmoid' ))
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
model_history = classifier.fit(X_train, y_train, batch_size=100, validation_split=0.2, epochs=100)
```

```
    Epoch 1/100
    4/4 [==============================] - 3s 406ms/step - loss: -9.7616 - accuracy: 0.7016 - val_loss: -13.5747 - val_accuracy: 0.7234
    Epoch 2/100
    4/4 [==============================] - 0s 39ms/step - loss: -18.1570 - accuracy: 0.7016 - val_loss: -20.8907 - val_accuracy: 0.7234
    Epoch 3/100
    4/4 [==============================] - 0s 62ms/step - loss: -26.6301 - accuracy: 0.7016 - val_loss: -28.3389 - val_accuracy: 0.7234
    Epoch 4/100
    4/4 [==============================] - 0s 88ms/step - loss: -34.9400 - accuracy: 0.7016 - val_loss: -36.1598 - val_accuracy: 0.7234
    Epoch 5/100
    4/4 [==============================] - 0s 50ms/step - loss: -44.0940 - accuracy: 0.7016 - val_loss: -44.2829 - val_accuracy: 0.7234
    Epoch 6/100
    4/4 [==============================] - 0s 57ms/step - loss: -53.5357 - accuracy: 0.7016 - val_loss: -53.3573 - val_accuracy: 0.7234
    Epoch 7/100
    4/4 [==============================] - 0s 46ms/step - loss: -64.2457 - accuracy: 0.7016 - val_loss: -63.7904 - val_accuracy: 0.7234
    Epoch 8/100
    4/4 [==============================] - 0s 39ms/step - loss: -76.9120 - accuracy: 0.7016 - val_loss: -75.6600 - val_accuracy: 0.7234
    Epoch 9/100
    4/4 [==============================] - 0s 62ms/step - loss: -90.1789 - accuracy: 0.7016 - val_loss: -89.4181 - val_accuracy: 0.7234
    Epoch 10/100
    4/4 [==============================] - 0s 36ms/step - loss: -106.6882 - accuracy: 0.7016 - val_loss: -105.0504 - val_accuracy: 0.7234
    Epoch 11/100
    4/4 [==============================] - 0s 20ms/step - loss: -125.9354 - accuracy: 0.7016 - val_loss: -122.8025 - val_accuracy: 0.7234
    Epoch 12/100
    4/4 [==============================] - 0s 30ms/step - loss: -146.2602 - accuracy: 0.7016 - val_loss: -143.3049 - val_accuracy: 0.7234
    Epoch 13/100
    4/4 [==============================] - 0s 24ms/step - loss: -170.9753 - accuracy: 0.7016 - val_loss: -166.5192 - val_accuracy: 0.7234
    Epoch 14/100
    4/4 [==============================] - 0s 36ms/step - loss: -197.2021 - accuracy: 0.7016 - val_loss: -193.0482 - val_accuracy: 0.7234
    Epoch 15/100
    4/4 [==============================] - 0s 58ms/step - loss: -228.1044 - accuracy: 0.7016 - val_loss: -222.8955 - val_accuracy: 0.7234
    Epoch 16/100
    4/4 [==============================] - 0s 48ms/step - loss: -262.4264 - accuracy: 0.7016 - val_loss: -256.4129 - val_accuracy: 0.7234
    Epoch 17/100
    4/4 [==============================] - 0s 43ms/step - loss: -301.1494 - accuracy: 0.7016 - val_loss: -293.9031 - val_accuracy: 0.7234
    Epoch 18/100
    4/4 [==============================] - 0s 40ms/step - loss: -346.2448 - accuracy: 0.7016 - val_loss: -335.6077 - val_accuracy: 0.7234
    Epoch 19/100
    4/4 [==============================] - 0s 25ms/step - loss: -395.8590 - accuracy: 0.7016 - val_loss: -382.1749 - val_accuracy: 0.7234
    Epoch 20/100
    4/4 [==============================] - 0s 28ms/step - loss: -449.8561 - accuracy: 0.7016 - val_loss: -434.5131 - val_accuracy: 0.7234
    Epoch 21/100
    4/4 [==============================] - 0s 29ms/step - loss: -507.7209 - accuracy: 0.7016 - val_loss: -493.0289 - val_accuracy: 0.7234
    Epoch 22/100
    4/4 [==============================] - 0s 85ms/step - loss: -577.1080 - accuracy: 0.7016 - val_loss: -557.2144 - val_accuracy: 0.7234
    Epoch 23/100
    4/4 [==============================] - 0s 42ms/step - loss: -651.3359 - accuracy: 0.7016 - val_loss: -628.1809 - val_accuracy: 0.7234
    Epoch 24/100
    4/4 [==============================] - 0s 31ms/step - loss: -736.5449 - accuracy: 0.7016 - val_loss: -705.5308 - val_accuracy: 0.7234
    Epoch 25/100
    4/4 [==============================] - 0s 53ms/step - loss: -823.1683 - accuracy: 0.7016 - val_loss: -791.1340 - val_accuracy: 0.7234
    Epoch 26/100
    4/4 [==============================] - 0s 26ms/step - loss: -923.3344 - accuracy: 0.7016 - val_loss: -884.0878 - val_accuracy: 0.7234
    Epoch 27/100
    4/4 [==============================] - 0s 40ms/step - loss: -1035.2036 - accuracy: 0.7016 - val_loss: -984.5273 - val_accuracy: 0.723
    Epoch 28/100
    4/4 [==============================] - 0s 50ms/step - loss: -1146.5388 - accuracy: 0.7016 - val_loss: -1095.0234 - val_accuracy: 0.72
    Epoch 29/100
```

```
model4.predict([[50, 1,1.2,0.8, 150, 70, 80,7.2,3.4, 0.8]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier
  warnings.warn(
array([1])
```

```
model1.predict([[50, 1, 1.2, 0.8, 150, 70, 80, 7.2,3.4,0.8]])
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier
  warnings.warn(
array([1])
```

```
classifier.save("liver.h5")
```

```
y_pred = classifier.predict(X_test)
```

```
4/4 [==============================] - 0s 5ms/step
```

```
y_pred
```

```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
y_pred =(y_pred > 0.5)
y_pred
```

```
    array([[ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
           [ True],
```

```
 y
```

```
    0      1
    1      1
    2      1
    3      1
    4      1
          ..
    578    2
    579    1
    580    1
    581    1
    582    2
    Name: outcome, Length: 583, dtype: int64
```

```
def predict_exit(sample_value):
 sample_value = np.array(sample_value)
 sample_value = sample_value.reshape(1, -1)
```

```
    sample_value = scale(sample_value)
    return classifier.predict(sample_value)


    sample_value = [[50,1,1.2,0.8,150, 70, 80, 7.2,3.4,0.8]]
    if predict_exit(sample_value) >0.5:
     print('Prediction: Liver Patient')
    else:
     print('Prediction : Healthy ')
```

```
1/1 [==============================] - 0s 67ms/step
Prediction: Liver Patient
```

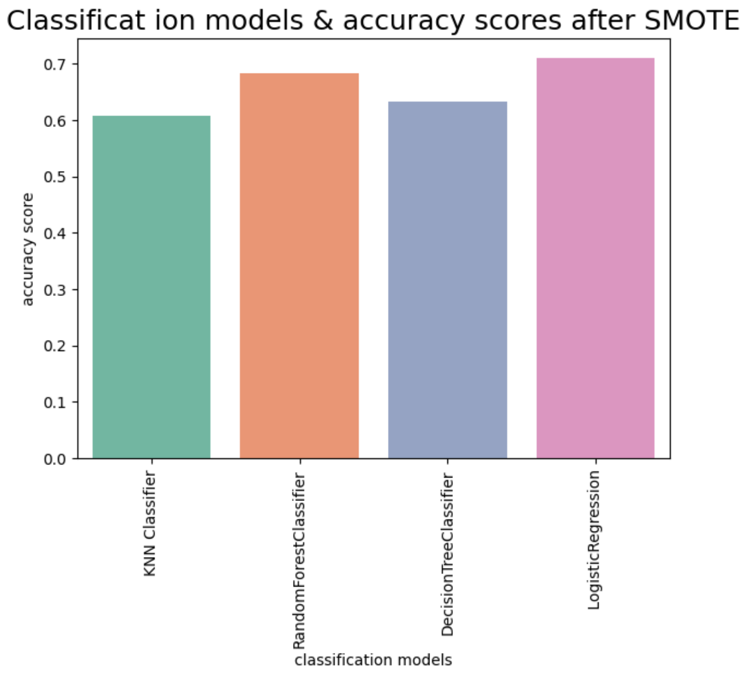
Performance Testing & Hyperparmeter tuning

```
acc_smote= [['KNN Classifier', knn1], ['RandomForestClassifier', rfc1], ['DecisionTreeClassifier', dtc1], [' LogisticRegression' , logil]]
Liverpatient_pred= pd.DataFrame(acc_smote, columns = ['classification models', 'accuracy score'])
Liverpatient_pred
```

	classification models	accuracy score
0	KNN Classifier	0.606838
1	RandomForestClassifier	0.683761
2	DecisionTreeClassifier	0.632479
3	LogisticRegression	0.709402

```
plt.figure(figsize=(7, 5))
plt.xticks(rotation=90)
plt.title('Classificat ion models & accuracy scores after SMOTE' , fontsize=18)
sns.barplot(x="classification models", y="accuracy score", data=Liverpatient_pred, palette ="Set2")
```

```
<Axes: title={'center': 'Classificat ion models & accuracy scores after
SMOTE'}, xlabel='classification models', ylabel='accuracy score'>
```



```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier()
model.fit(X, y)
```

```
▾ ExtraTreesClassifier
ExtraTreesClassifier()
```

```
model.feature_importances_
```

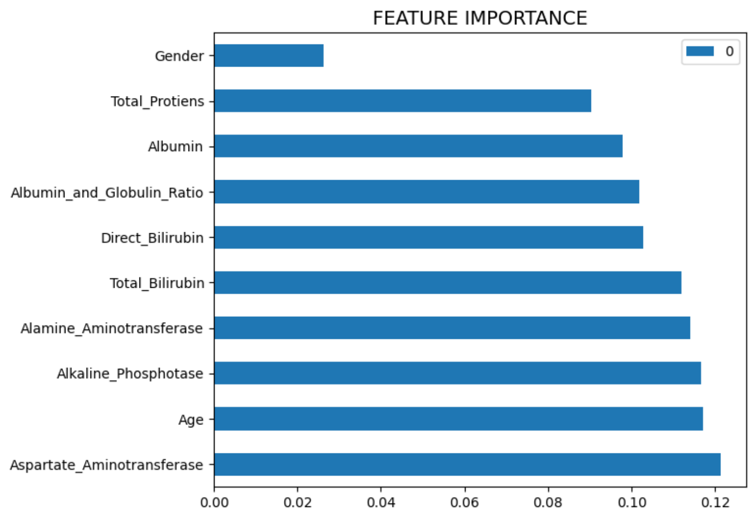
```
array([0.11702872, 0.02622561, 0.11202284, 0.10274433, 0.11658481,
       0.11404011, 0.12130022, 0.09025552, 0.09784651, 0.10195132])
```

```
dd=pd.DataFrame (model.feature_importances_, index=X.columns).sort_values(0, ascending=False)
dd
```

	0
Aspartate_Aminotransferase	0.121300
Age	0.117029
Alkaline_Phosphotase	0.116585
Alamine_Aminotransferase	0.114040
Total_Bilirubin	0.112023
Direct_Bilirubin	0.102744
Albumin_and_Globulin_Ratio	0.101951
Albumin	0.097847
Total_Protiens	0.090256

```
dd.plot (kind='barh', figsize=(7,6))
plt.title("FEATURE IMPORTANCE",fontsize=14)
```

```
Text(0.5, 1.0, 'FEATURE IMPORTANCE')
```



▾ model Devlopment

```
import joblib
joblib.dump(model1, 'ETC.pkl')

    ['ETC.pkl']
```