Loyola College (Autonomous), Chennai-600034

UST 6708 - PROJECT

SUBMITTED TO THE

DEPARTMENT OF STATISTICS

Loyola College (Autonomous)

In
Partial fulfillment for the requirement of
Bachelor of Science degree
In Statistics

By

DEBAYAN DATTA

(20-UST-042)

Under the Supervision of

Dr. S Amala Revathy

BONAFIDE CERTIFICATE

This is to certify that the report '***Statistical Analysis on Hepatic Data***' is a Bonafide record completed by **DEBAYAN DATTA (20-UST-042)** under the guidance of **Dr. S Amala Revathy**, Assistant Professor (Loyola College), during the academic year 2022-23.

Dr. T EDWIN PRANHAKARAN	**DR S AMALA REVATHY**
Head of the Department	***Assistant Professor***
Department of Statistics	Department of Statistics
Loyola College, Chennai -34	Loyola College, Chennai -34

CONTENTS

ACKNOWLEDGEMENT

It gives me immense pleasure in presenting the Internship Report on “Statistical Analysis on Hepatic Data”. I wish to place on record my sincere thanks and obligations to all those who have contributed directly or indirectly to making this internship a success. Firstly, I would like to thank the god almighty for his kind blessings upon me for the opportunity to have given me to pursue my project on a dataset posted on UCI Machine Learning Repository.

I express my gratitude to Loyola College, Chennai and Principal Rev Dr A Thomas, for including the Application Oriented Paper – Project Work as part of our curriculum, as it provides students with a first-hand opportunity to understand the application of their skill-sets in the a real time basis. It also helps students to develop an interest in Independent Research.

I cannot express enough thanks to the Department of Statistics and Dr T Edwin Prabhakaran (Head of the Department) for their continued support and encouragement. I am thankful to my project guide Dr S Amala Revathy (Assistant Professor) for assisting me through this report.

DECLARATION

I hereby declare that the internship titled “Statistical Analysis on Hepatic” is based on the original work carried out by me under the guidance of Dr S. Amala Revathy, Assistant Professor, Loyola College, Chennai *(Project Guide),* submitted in partial fulfilment of the requirement of the course of study.

Debayan Datta

20-UST-042

KNOW MORE ABOUT THE DATA

The dataset is collected from **UC Irvine Machine Learning Repository** forum.

The link is given below:

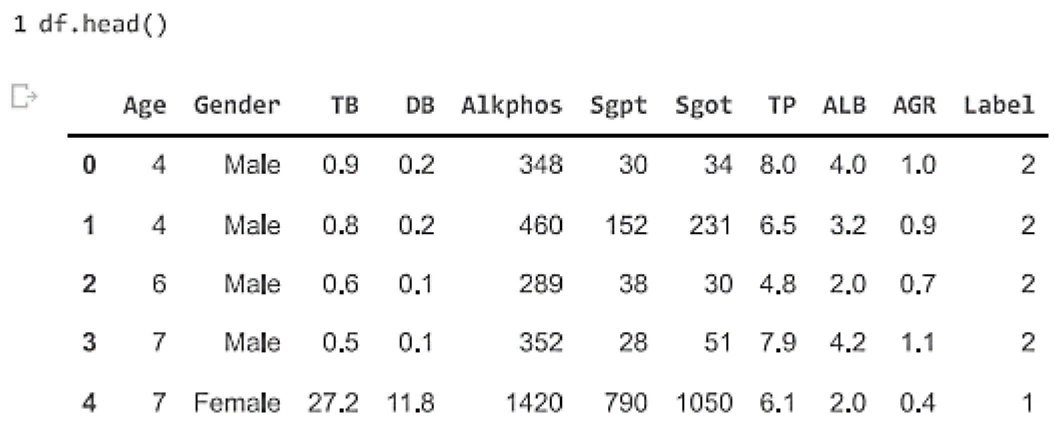
https://archive.ics.uci.edu/ml/datasets/ILPD+%28Indian+Liver+Patient+Dataset%29#

About UC Irvine Machine Learning Repository:

The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms. The archive was created as an ftp archive in 1987 by David Aha and fellow graduate students at UC Irvine. Since that time, it has been widely used by students, educators, and researchers all over the world as a primary source of machine learning data sets.

As an indication of the impact of the archive, it has been cited over 1000 times, making it one of the top 100 most cited "papers" in all of computer science. The current version of the web site was designed in 2007 by Arthur Asuncion and David Newman, and this project is in collaboration with Rexa.info at the University of Massachusetts Amherst. Funding support from the National Science Foundation is gratefully acknowledged.

DATA to be worked on:



```
df.head()
```

```
   Age  Gender    TB    DB  Alkphos  Sgpt  Sgot   TP  ALB  AGR  Label
0    4    Male   0.9   0.2      348    30    34  8.0  4.0  1.0      2
1    4    Male   0.8   0.2      460   152   231  6.5  3.2  0.9      2
2    6    Male   0.6   0.1      289    38    30  4.8  2.0  0.7      2
3    7    Male   0.5   0.1      352    28    51  7.9  4.2  1.1      2
4    7  Female  27.2  11.8     1420   790  1050  6.1  2.0  0.4      1
```

META DATA:

HEADING	DESCRIPTION	TYPE
Age	Age of the subject	Continuous
Gender	Gender of the subject	Categorical
TB	Total Bilirubin	Continuous
DB	Direct Bilirubin	Continuous
Alkphos	Alkaline Phosphotase	Continuous
Sgpt	Alamine Aminotransferase	Continuous
Sgot	Aspartate Aminotransferase	Continuous
TP	Total Protiens	Continuous
ALB	Albumin	Continuous
AGR	Albumin and Globulin Ratio	Continuous

DESCRIPTION:

1. **AGE**: Age of the subjects that are considered

2. **GENDER**: Gender of the subjects. (Male/Female)

3. **TB**: Bilirubin is a substance made when your body breaks down red blood cells. This is a normal process. Bilirubin is also part of bile, which your liver makes to help digest the food you eat. Testing of Bilirubin is used to find out how well your liver is working. A small amount of bilirubin in your blood is normal, but a high level may be a sign of liver disease.The liver makes bile to help you digest food, and bile contains bilirubin. Most bilirubin comes from the body's normal process of breaking down old red blood cells. A healthy liver can normally get rid of bilirubin. But when you have liver problems, bilirubin can build up in your body to unhealthy levels. Bilirubin results depend on your age, gender, and health.

4. **DB**: In the liver, bilirubin is changed into a form that your body can get rid of. This is called conjugated bilirubin or direct bilirubin. This bilirubin travels from the liver into the small intestine. A very small amount passes into your kidneys and is excreted in your urine.

NOTE: Bilirubin attached by the liver to glucuronic acid, a glucose-derived acid, is called direct, or conjugated, bilirubin. Bilirubin not attached to glucuronic acid is called indirect, or unconjugated, bilirubin. All the bilirubin in your blood together is called total bilirubin.

5. **ALKPHOS**: Alkaline Phosphotase (ALP) is an enzyme found in many parts of your body. Each part of your body produces a different type of ALP. Most ALP is found in your liver, bones, kidneys, and digestive system. Abnormal levels of ALP in your blood may

be a sign of a wide range of health conditions, including liver disease, bone disorders, and chronic kidney disease. But an alkaline phosphatase test alone can't identify the source of ALP in your blood, so other tests are usually needed to make a diagnosis.

6. **SGPT**: A high level of SGPT released into the blood may be a sign of liver damage, cancer, or other diseases. Also called alanine transferase and serum glutamate pyruvate transaminase. Your body uses Sgpt to break down food into energy. Normally, Sgpt levels in the blood are low. If your liver is damaged, it will release more Sgpt into your blood and levels will rise.

7. **SGOT**: Serum glutamic oxaloacetic transaminase, an enzyme that is normally present in liver and heart cells. SGOT is released into blood when the liver or heart is damaged. The blood SGOT levels are thus elevated with liver damage (for example, from viral hepatitis) or with an insult to the heart (for example, from a heart attack). Some medications can also raise SGOT levels. SGOT is also called aspartate aminotransferase

8. **TP**: Serum total protein, also known as total protein, is a clinical chemistry parameter representing the concentration of protein in serum. Serum contains many proteins including serum albumin, a variety of globulins. A total protein test measures the amount of protein in your blood. Proteins are important for the health and growth of the body's cells and tissues. The test can help diagnose a number of health conditions, including: kidney disease, liver disease.

9. **ALB**: Albumin is a protein made by your liver. Albumin enters your bloodstream and helps keep fluid from leaking out of your blood vessels into other tissues. It is also carries hormones, vitamins, and enzymes throughout your body.

10. **AGR**: Albumin is produced in the liver, and globulins are produced majorly from the immune system and the liver. They help in metabolism as well as make the immune system stronger. The albumin-to-globulin concentration is called the AG ratio. This ratio is tested to determine the amount or concentration of proteins in the blood. It compares the amount of albumin in your blood to that of globulins. The AG ratio test is also known as the total serum protein test.

RESPONSE VARIABLE:

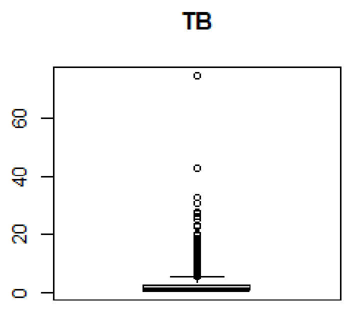
❖ **LABEL**: Binary classified data taking 1 or 2 where
    - 1 means subject is a liver patient
    - 2 means subject is not a liver patient

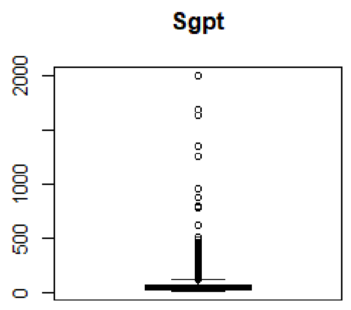
There are 579 data points in total in which there are 414 points having (1) as the Label value and 165 points having (2) as the Label value.

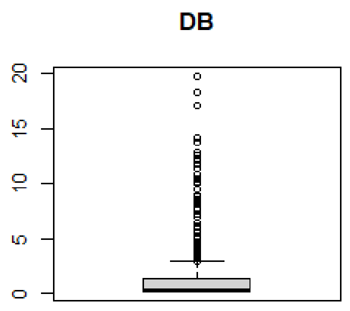
Describing the data with the different explanatory variables:

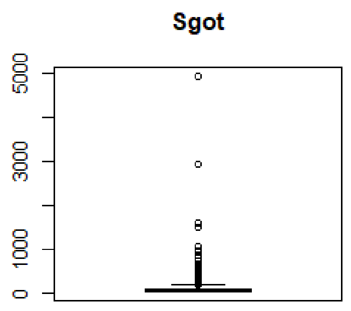
```
               TB          DB      Alkphos         Sgpt
count  579.000000  579.000000   579.000000   579.000000
mean     3.315371    1.494128   291.366149    81.126079
std      6.227716    2.816499   243.561863   183.182845
min      0.400000    0.100000    63.000000    10.000000
25%      0.800000    0.200000   175.500000    23.000000
50%      1.000000    0.300000   208.000000    35.000000
75%      2.600000    1.300000   298.000000    61.000000
max     75.000000   19.700000  2110.000000  2000.000000

              Sgot          TP         ALB         AGR
count   579.000000  579.000000  579.000000  579.000000
mean    110.414508    6.481693    3.138515    0.947064
std     289.850034    1.084641    0.794435    0.319592
min      10.000000    2.700000    0.900000    0.300000
25%      25.000000    5.800000    2.600000    0.700000
50%      42.000000    6.600000    3.100000    0.930000
75%      87.000000    7.200000    3.800000    1.100000
max    4929.000000    9.600000    5.500000    2.800000
```

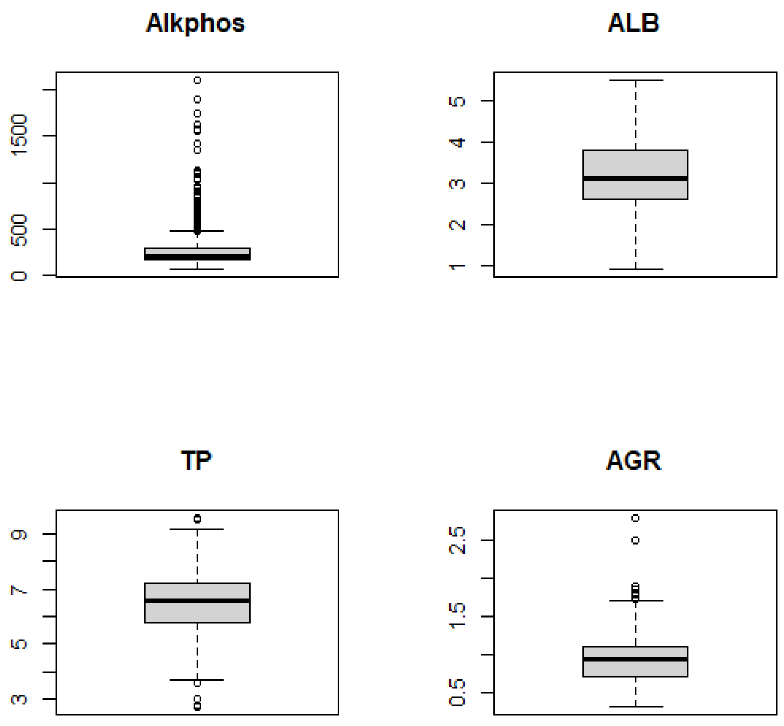
BOX PLOTS:



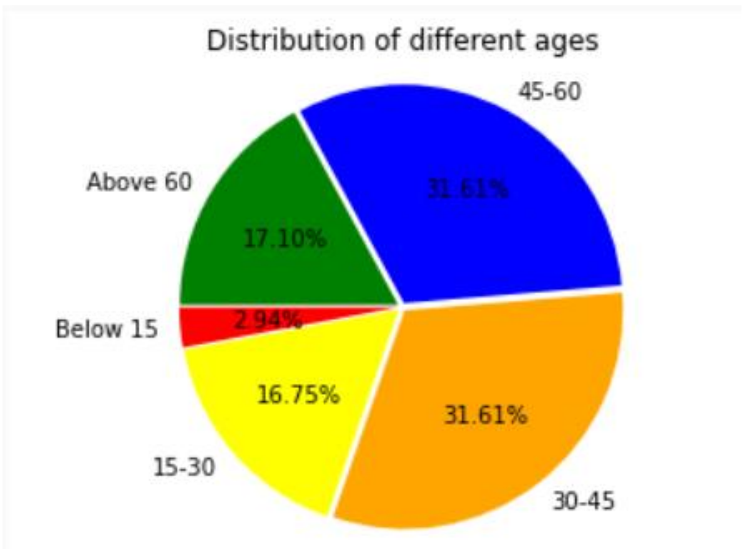




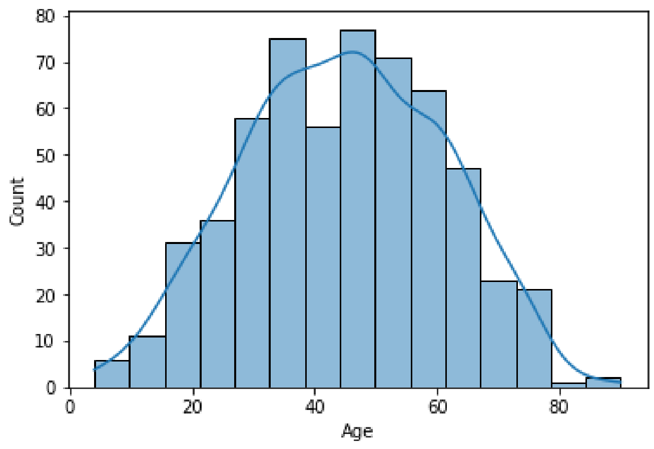




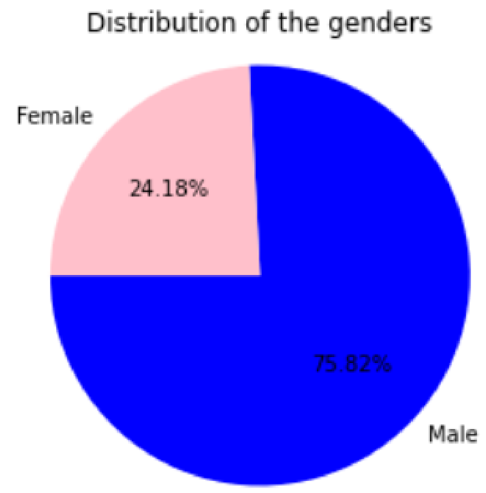
The Age distribution in the dataset:

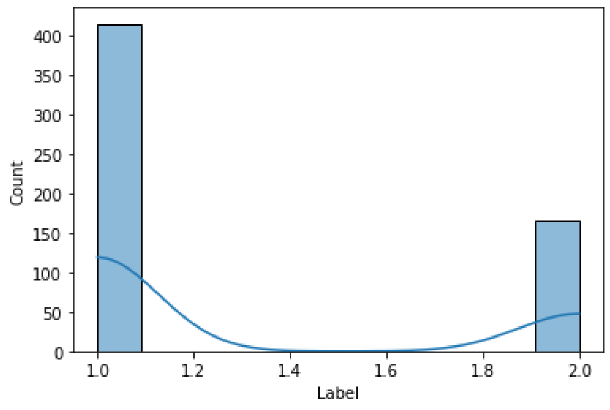


```
<Axes: xlabel='Age', ylabel='Count'>
```



The Gender distribution in the dataset:

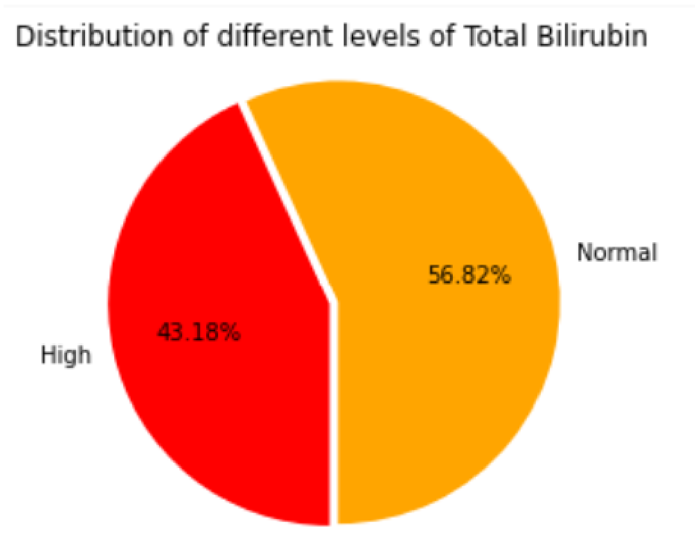




414 points having (1) as the Label value and 165 points having (2) as the Label value.

From this we can conclude that the data is highly imbalanced.

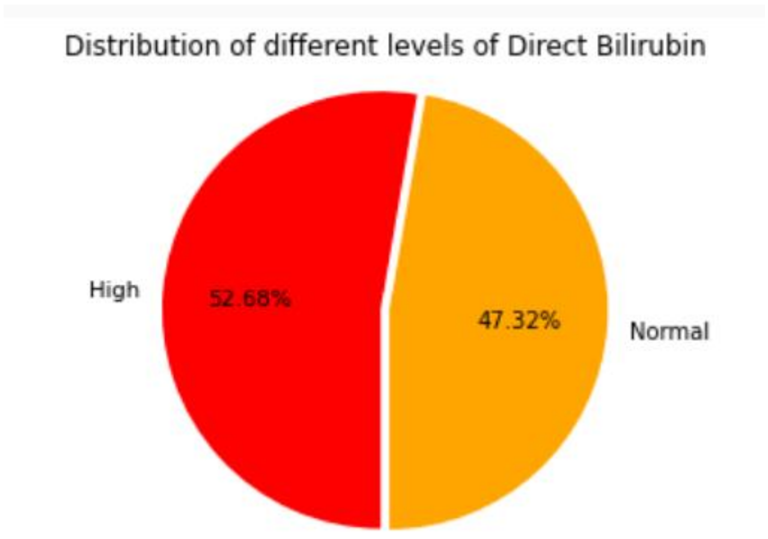
We separate the observations of the different explanatory variables with respect to their normal range. Their normal range is given beside the data visualizations.



Total Bilirubin

NORMAL RANGE:

0.1 to 1.2 mg/dL

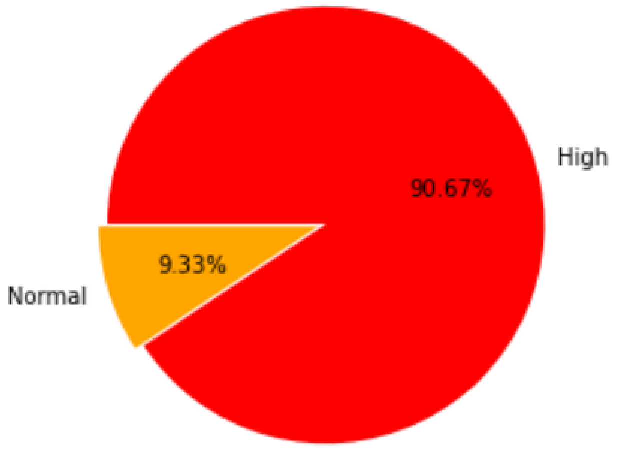


Direct Bilirubin

NORMAL RANGE:

less than 0.3 mg/dL

Distribution of different levels of Alkaline Phosphatase

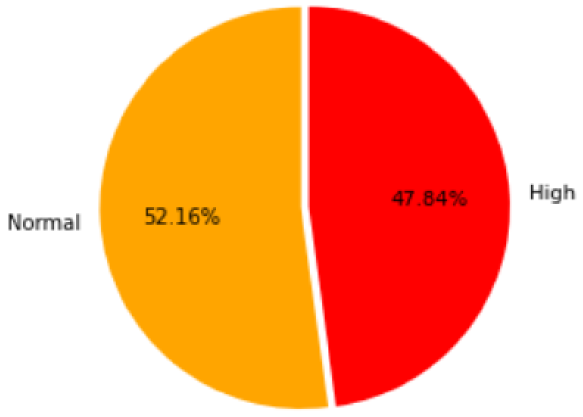


Alkaline Phosphatase

NORMAL RANGE:

44 to 147 units/litre

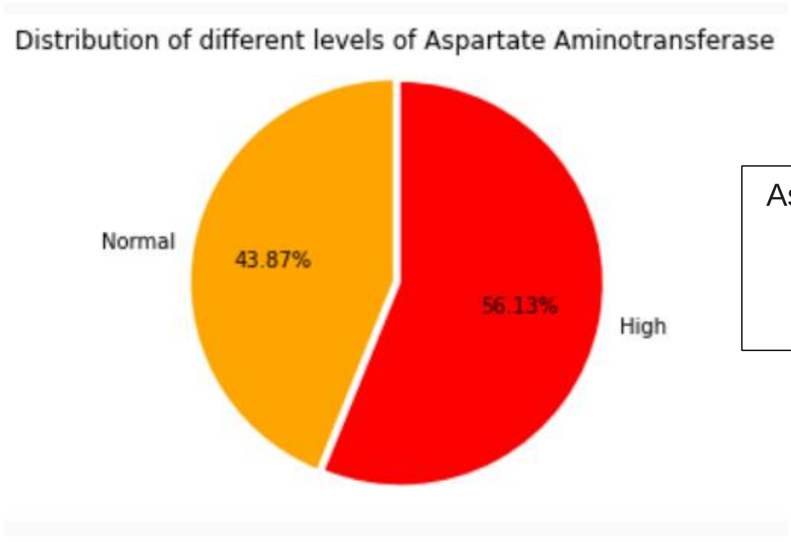
Distribution of different levels of Alamine Aminotransferase



Alamine Aminotransferase

NORMAL RANGE:

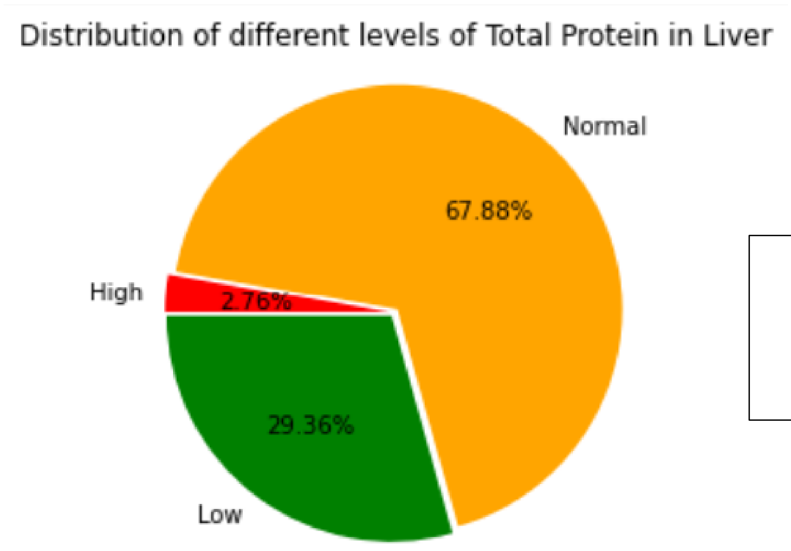
7 to 56 UI/litre



Asparate Aminotransferase

NORMAL RANGE:

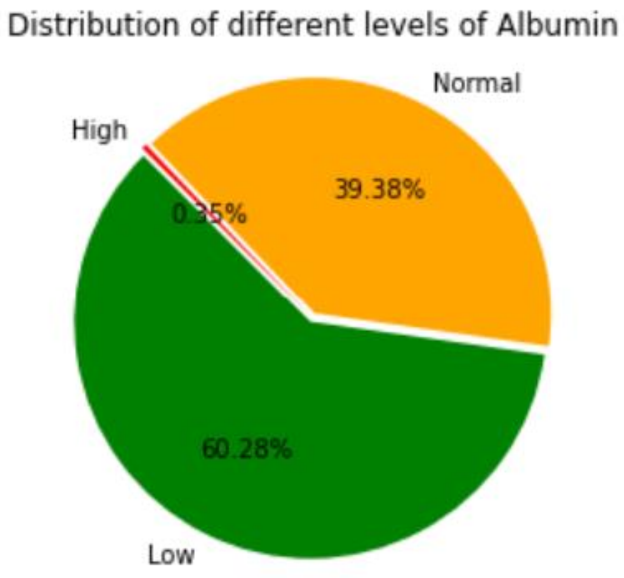
8 to 45 UI/litre



Total Protein

NORMAL RANGE:

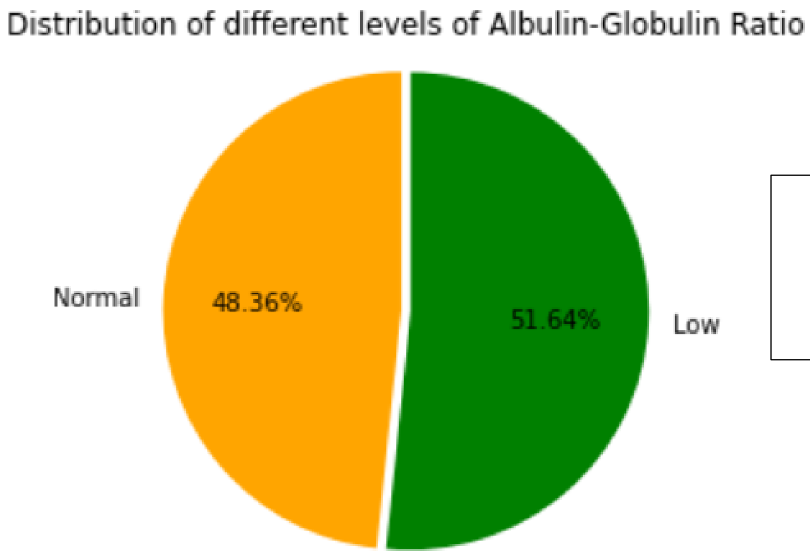
6.0 to 8.3 g/dL



Albumin

NORMAL RANGE:

3.4 to 5.4 g/dL



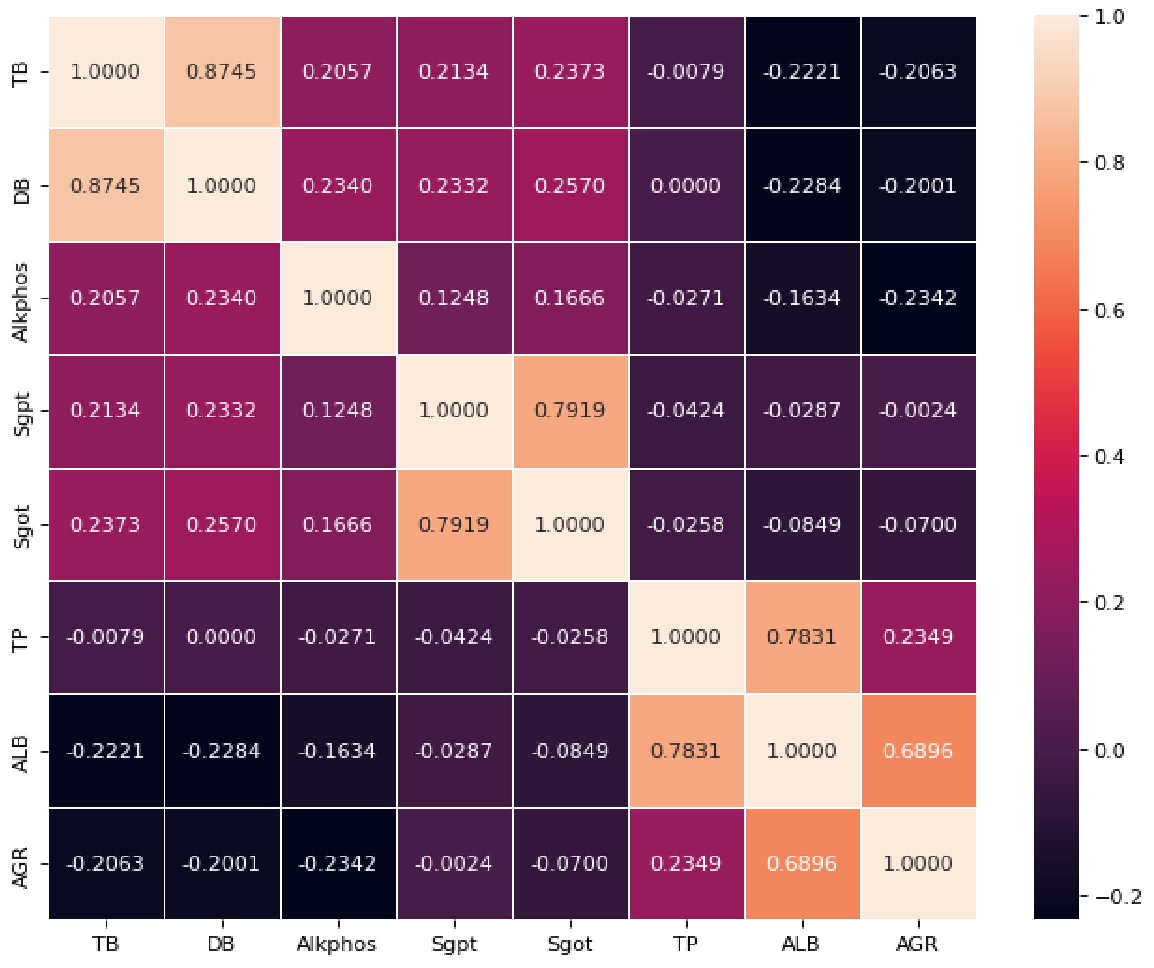
Albumin Globulin

NORMAL RANGE:

1.1 to 2.5

BIVARIATE ANALYSIS

Correlation Matrix:



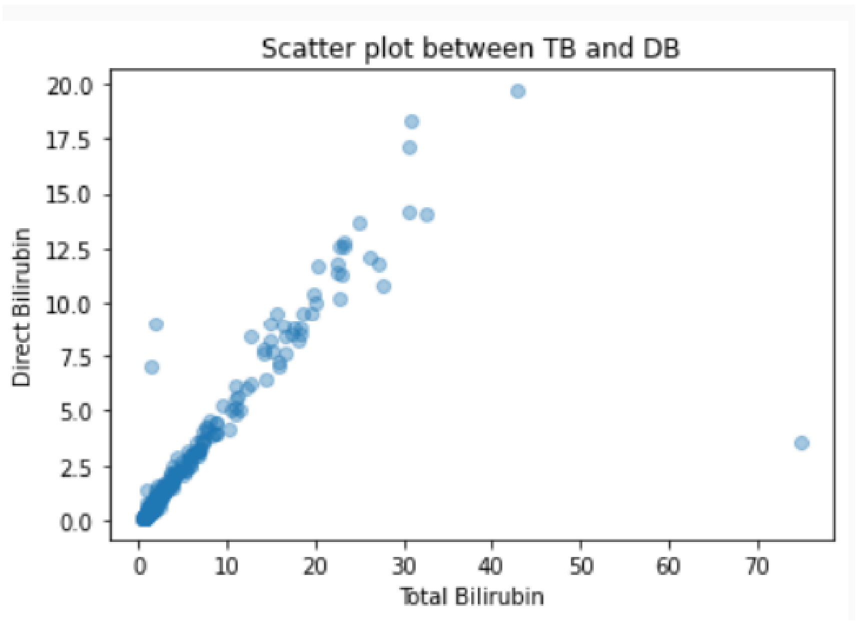
	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGR
TB	1.0000	0.8745	0.2057	0.2134	0.2373	-0.0079	-0.2221	-0.2063
DB	0.8745	1.0000	0.2340	0.2332	0.2570	0.0000	-0.2284	-0.2001
Alkphos	0.2057	0.2340	1.0000	0.1248	0.1666	-0.0271	-0.1634	-0.2342
Sgpt	0.2134	0.2332	0.1248	1.0000	0.7919	-0.0424	-0.0287	-0.0024
Sgot	0.2373	0.2570	0.1666	0.7919	1.0000	-0.0258	-0.0849	-0.0700
TP	-0.0079	0.0000	-0.0271	-0.0424	-0.0258	1.0000	0.7831	0.2349
ALB	-0.2221	-0.2284	-0.1634	-0.0287	-0.0849	0.7831	1.0000	0.6896
AGR	-0.2063	-0.2001	-0.2342	-0.0024	-0.0700	0.2349	0.6896	1.0000

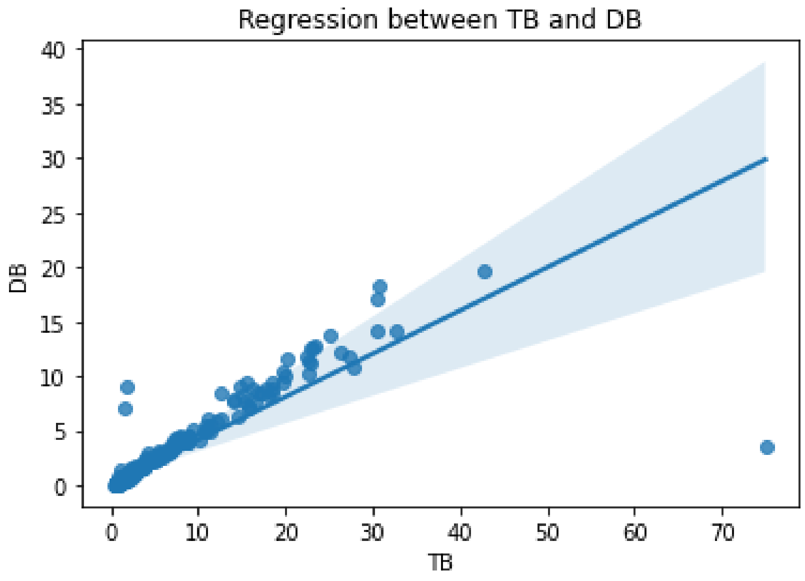
INFERENCE: We observe that (TB,DB), (Sgpt,Sgot), (TP,ALB) and (ALB,AGR) pairs show a high correlation. Let us see their scatter plots to verify it.

REMARK:

- The p-value for a hypothesis test whose null hypothesis is that the slope is zero, using Wald Test with t-distribution of the test statistic.
- Standard error of the estimated slope (gradient), under the assumption of residual normality

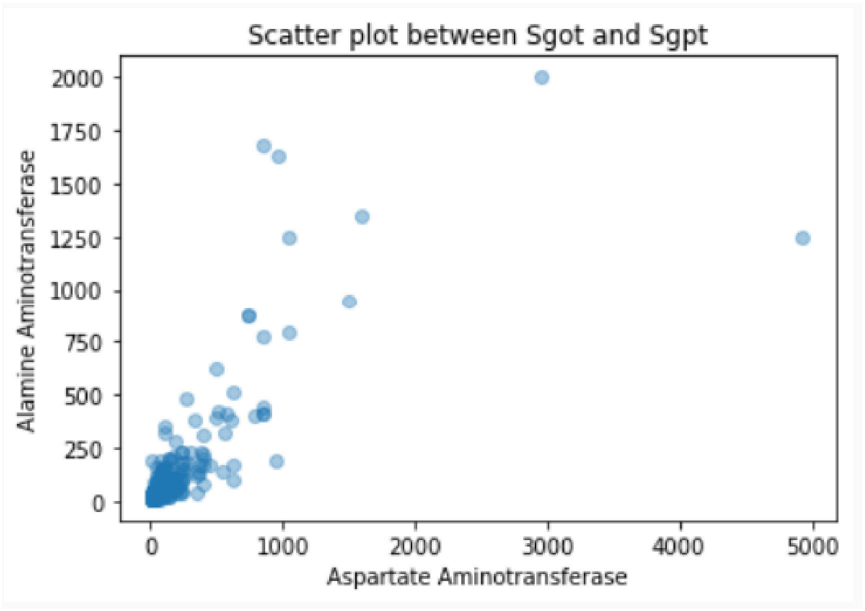
Relationship between Total Bilirubin and Direct Bilirubin

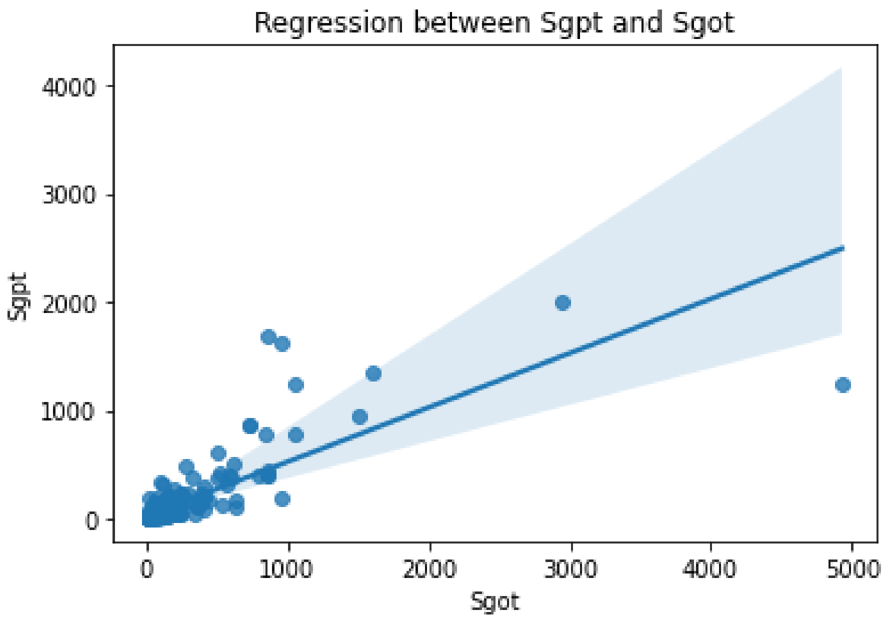




```
Slope of the Regression Line:       0.39548596376115897
Intercept of the Regression Line:   0.18294498093968792
Coefficient of Determination:       0.7647169648763394
Standard error of estimated slope:  0.009132467017835813
p-value:                            1.919001236357252e-183
```

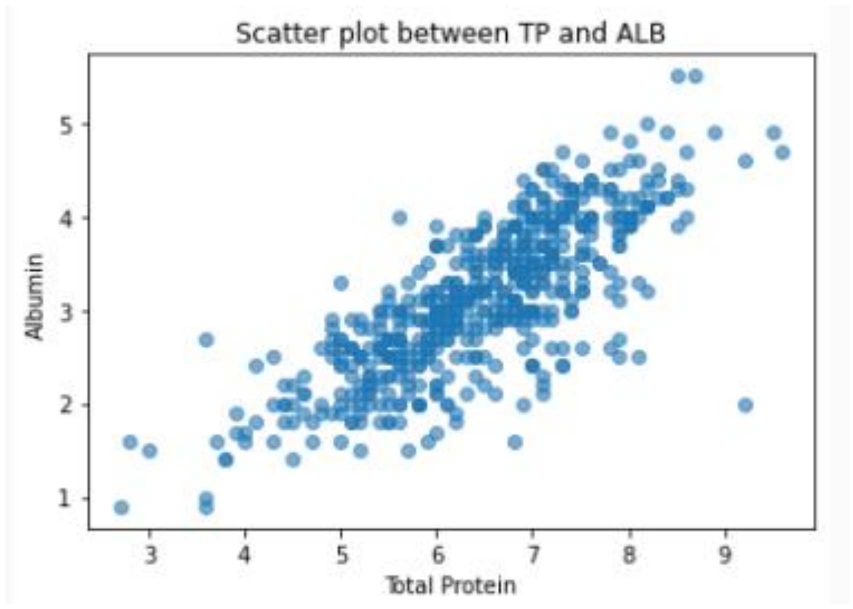
Relationship between Alamine and Asparate Aminotransferase

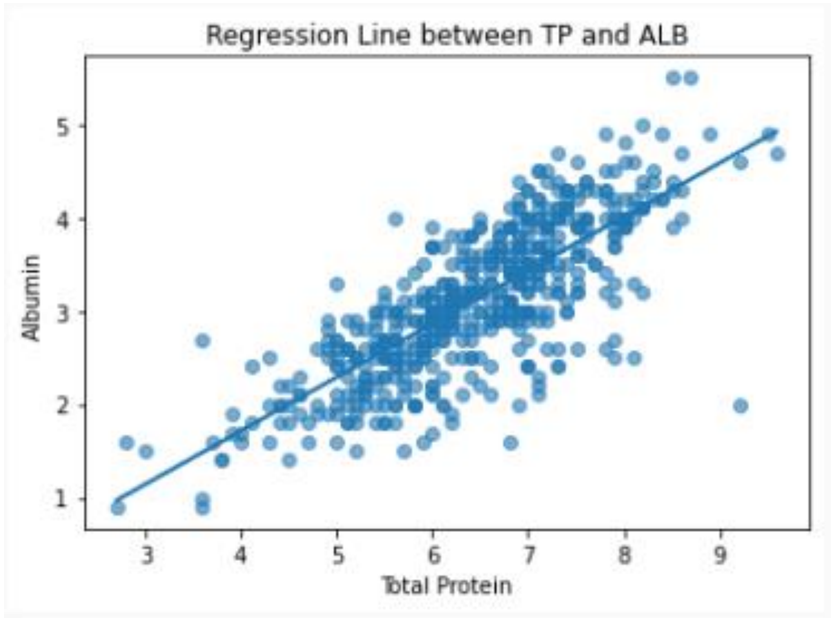




```
Slope of the Regression Line:       0.5004503848787011
Intercept of the Regression Line:   25.869096536622862
Coefficient of Determination:       0.6270456633002384
Standard error of estimated slope:  0.016067613169001554
p-value:                            1.1084796956358684e-125
```

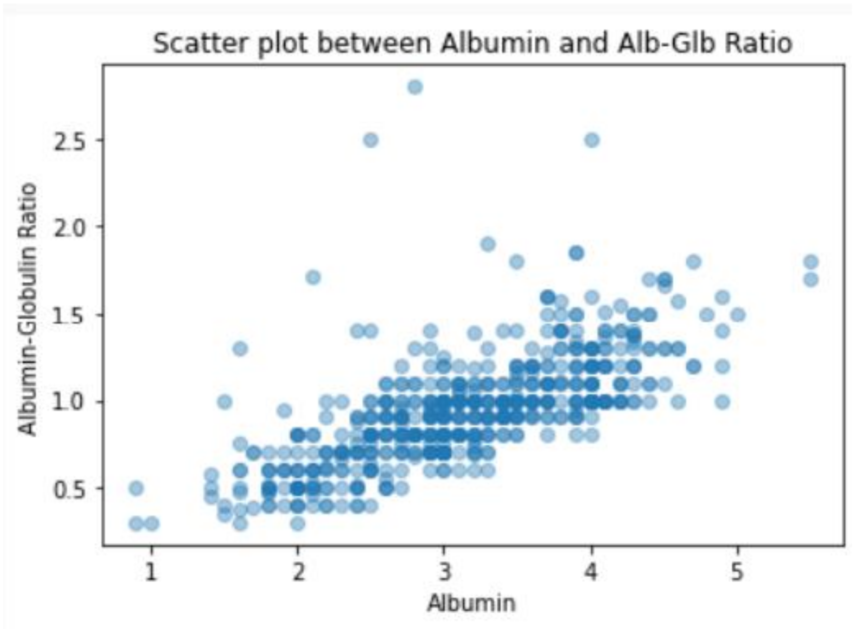
Relationship between Albumin and Total Proteins

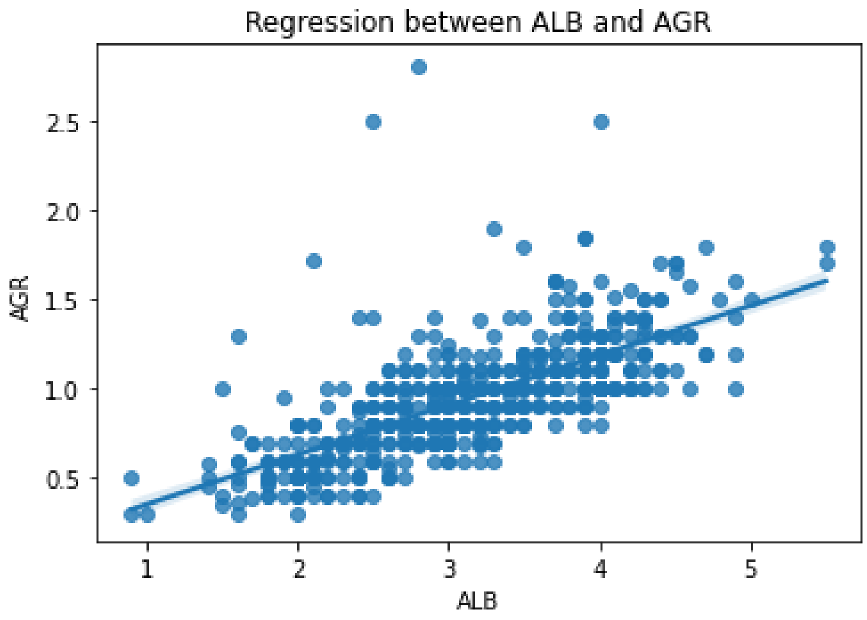




```
Slope of the Regression Line:       0.57358282272726
Intercept of the Regression Line:   -0.5792728418188839
Coefficient of Determination:       0.613264670993127
Standard error of estimated slope:  0.018962307317707067
p-value:                            3.942423581783356e-121
```

Relationship between Albumin and Albumin-Globulin Ratio



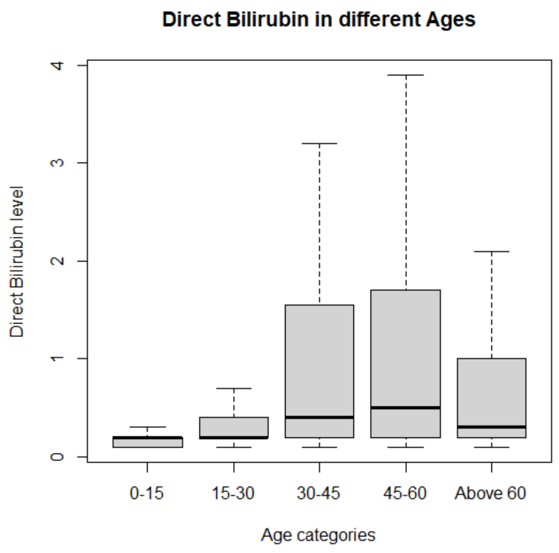


```
Slope of the Regression Line:       0.27743129884440476
Intercept of the Regression Line:   0.07634169903272481
Coefficient of Determination:       0.47559276698983055
Standard error of estimated slope:  0.012127862502597649
p-value:                            6.399343100418436e-83
```

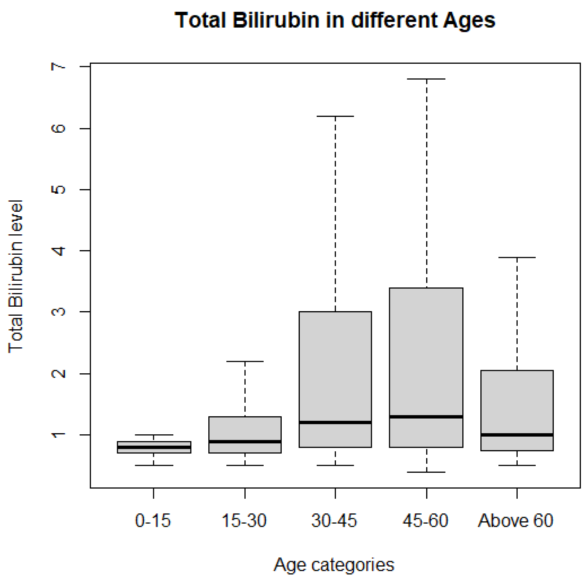
Now we divide the data into 5 age categories namely:

- 0 to 15 years
- 15 to 30 years
- 30 to 45 years
- 45 to 60 years
- Above 60 years

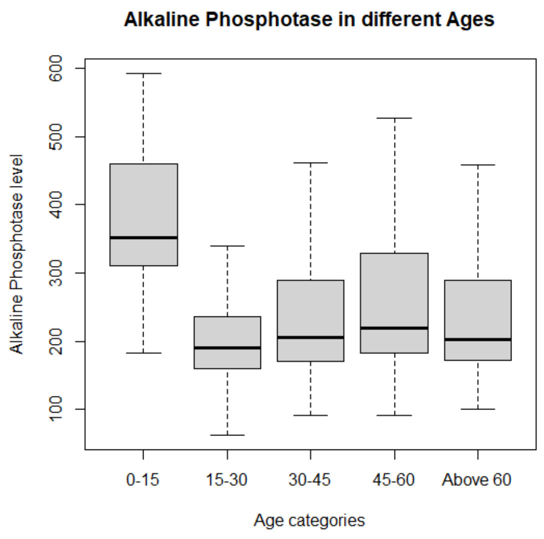
Now let us study the different explanatory variables with respect to these age categories



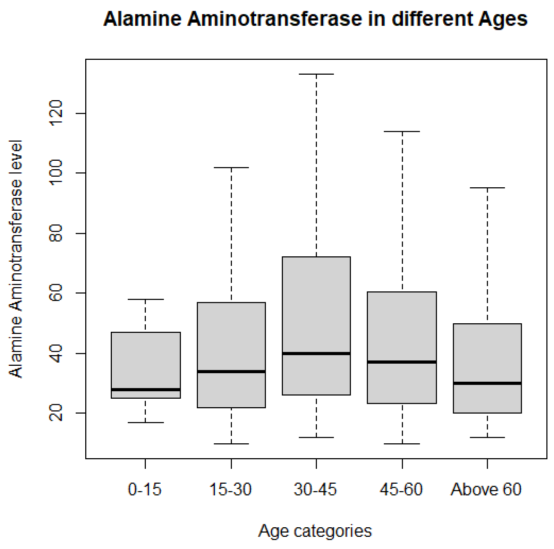
The least DB count can be seen in the ages 0-15 and most wide range of values can be found in 45-60 ages.



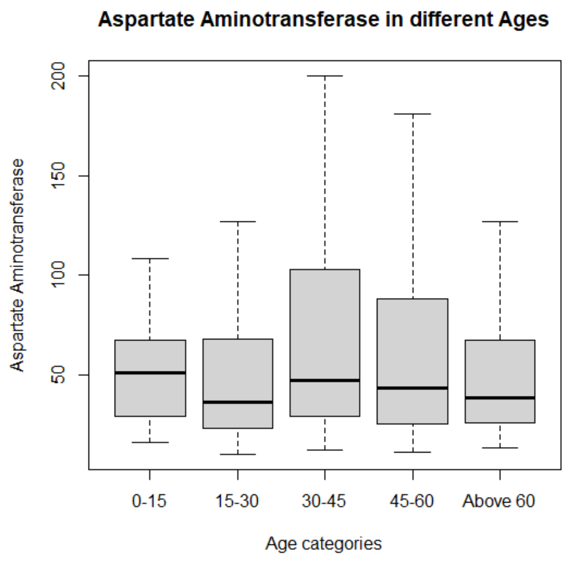
The least TB count can be seen in the ages 0-15 and most wide range of values can be found in 45-60 ages. Again for more than 60, it falls.



Ages 0-15 show high values in Alkphos but the rest show similar kind of values.

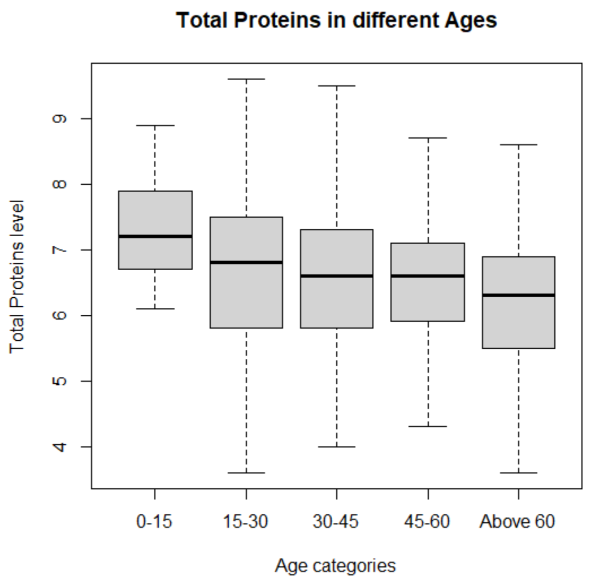


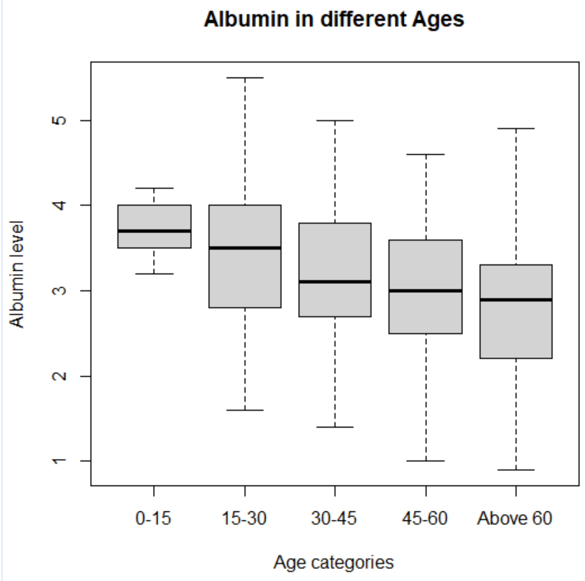
At the age of 30-45, Sgpt has the widest range of values. From 0-15 to more than 60, the range increases and then decreases.



At 30-45 Sgot has the widest range of values. 0-15 and Above 60 has the same spread for Sgot

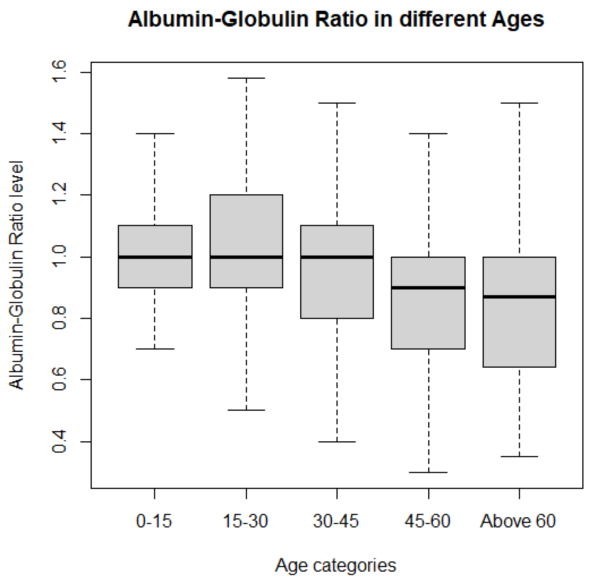
With increase in age, the mean value of Total Proteins decrease. In childhood the TP levels are maximum.





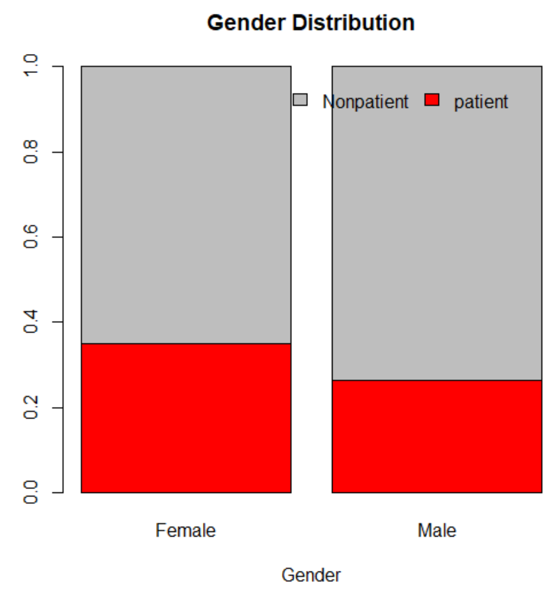
The Albumin range is less but the value is high in childhood. It again decreases gradually with the increase in age

We notice AG Ratio increases till 15-30 and after that it falls or keeps constant.

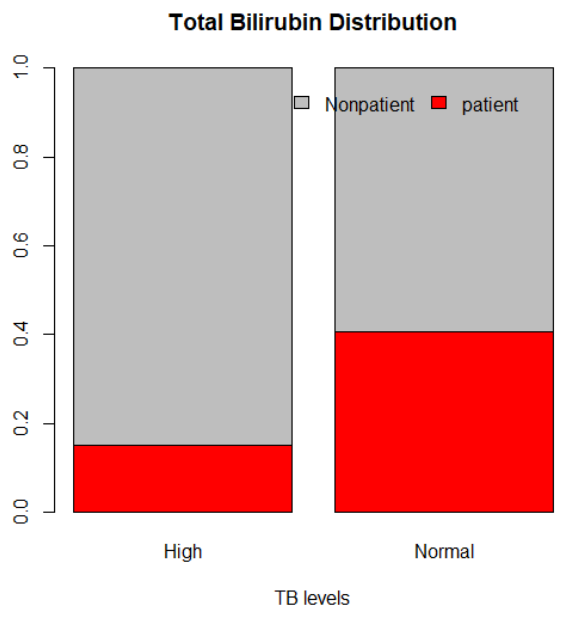


Study of the explanatory variable with respect to the Response variable:

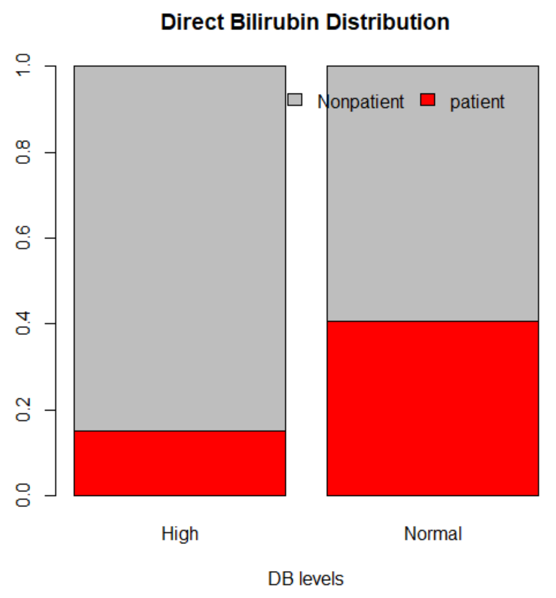
We display how many subjects (Patient/ Non-patient) fall under the different levels of the explanatory variable.



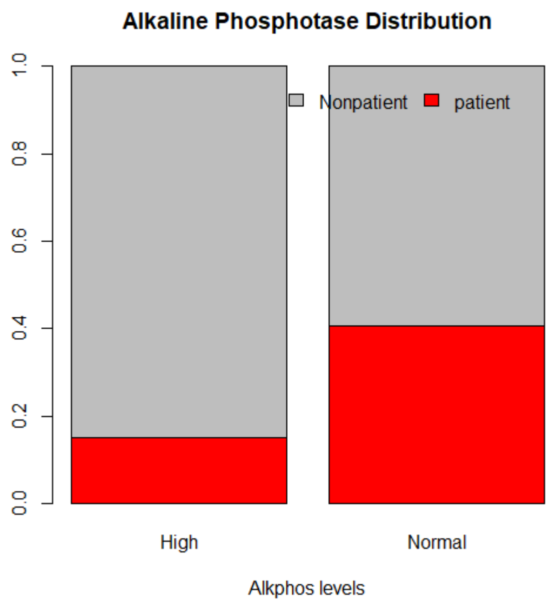
Being a Liver Patient is not biased on Gender of the subject since both the genders have more or less equal chances of being a patient



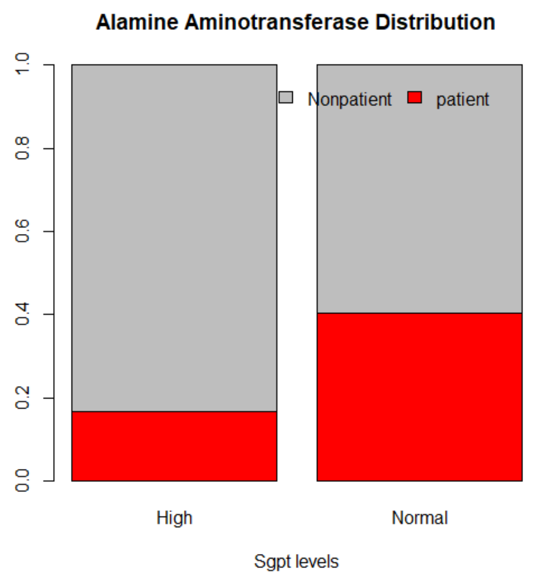
Number of patients who have normal TB levels are more than the patients having high level by a factor of 2



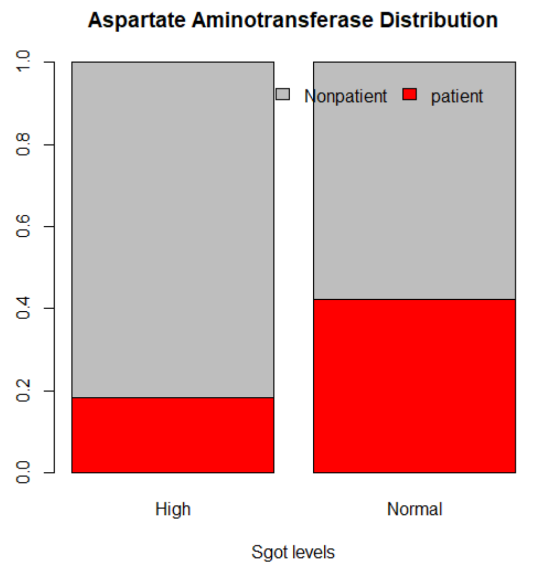
Number of patients who have normal DB levels are more than the patients having high level by a factor of 2



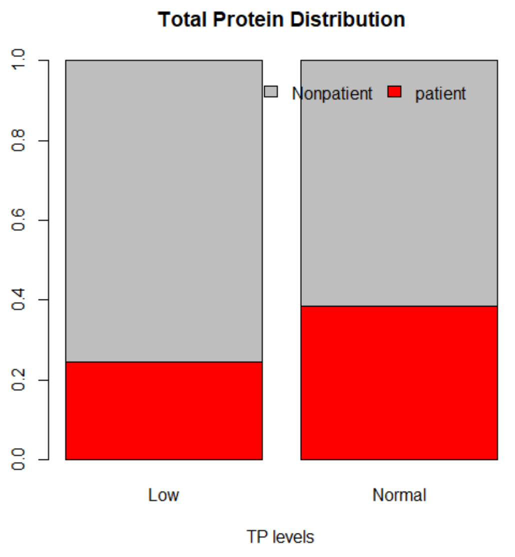
Number of patients who have normal Alkphos levels are more than the patients having high level by a factor of 2



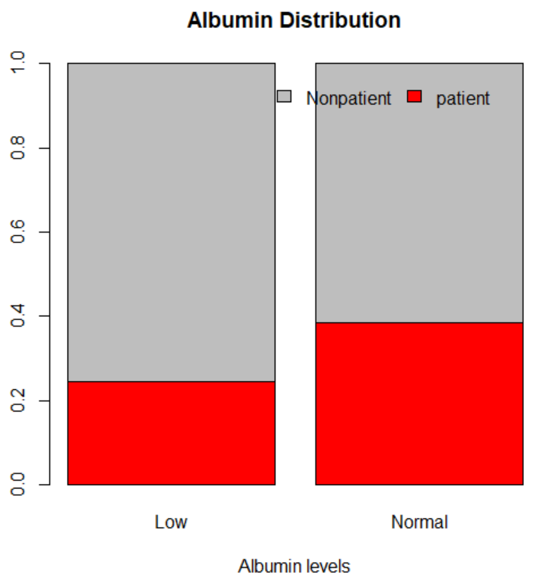
Number of patients who have normal Sgpt levels are more than the patients having high level by a factor of 2



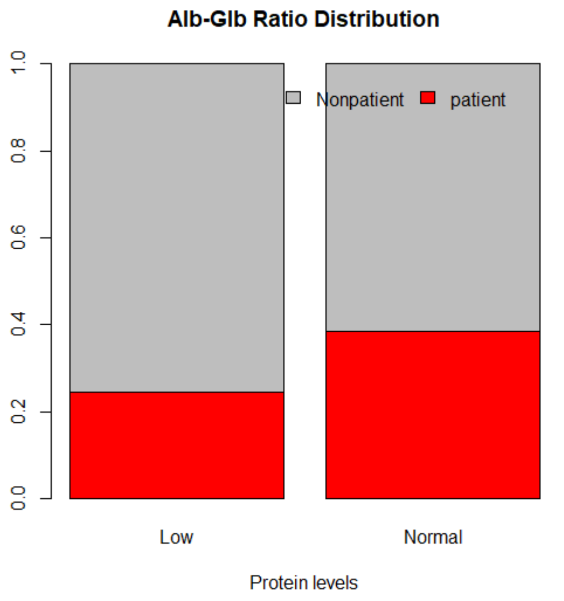
Number of patients who have normal Sgot levels are more than the patients having high level by a factor of 2



Number of patients having Normal Total proteins level and low protein level is more or less the similar



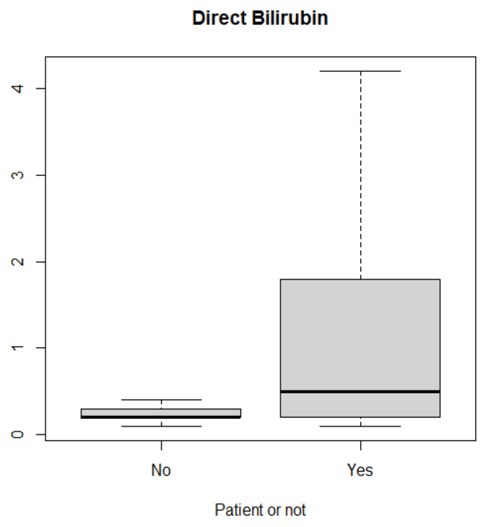
Number of patients having Normal Albumin level and low protein level is more or less the similar

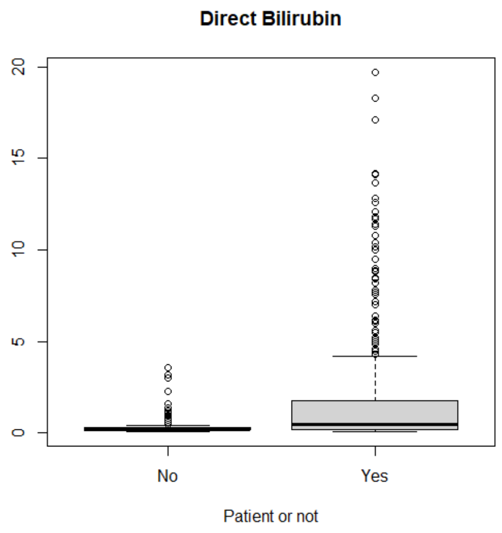


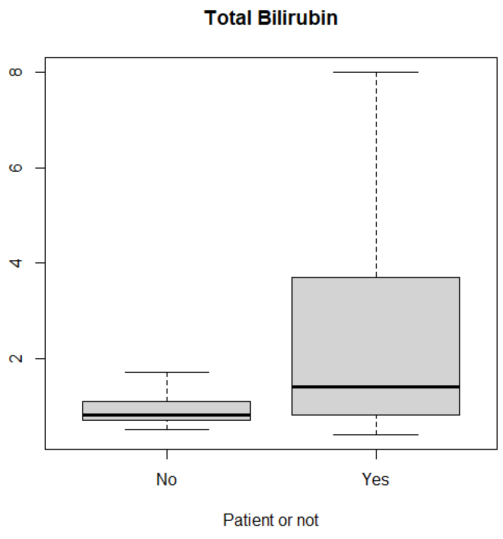
Number of patients having Normal AG Ratio level and low protein level is more or less the similar

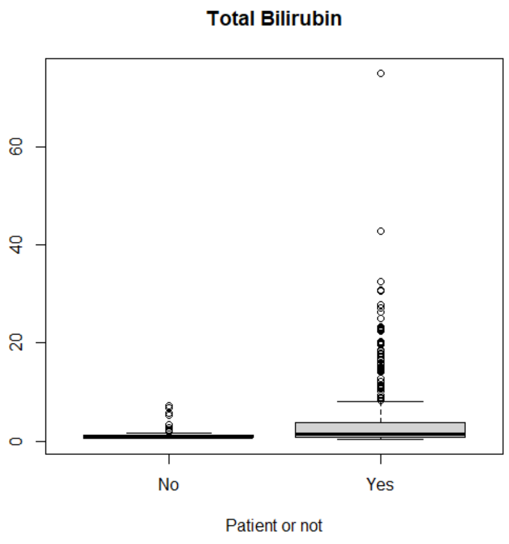
We saw previously that our data contains many outliers. Let's check whether the outliers are only present for a particular type of the response variable.

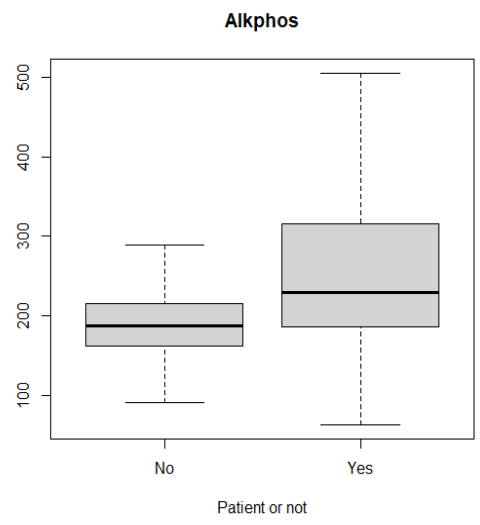
Without Outliers ***With Outliers***

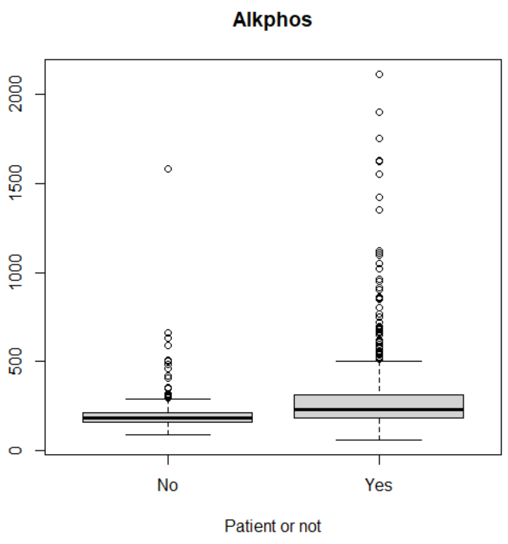


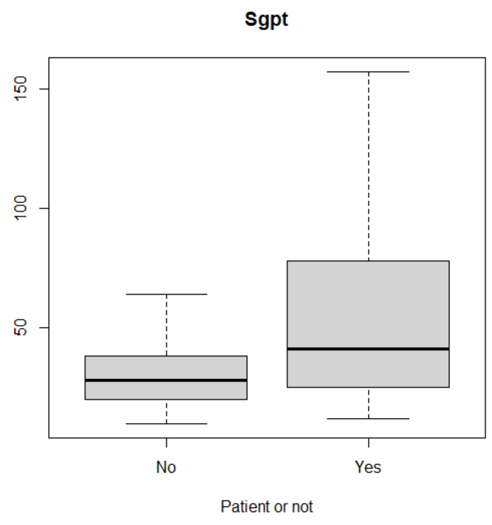


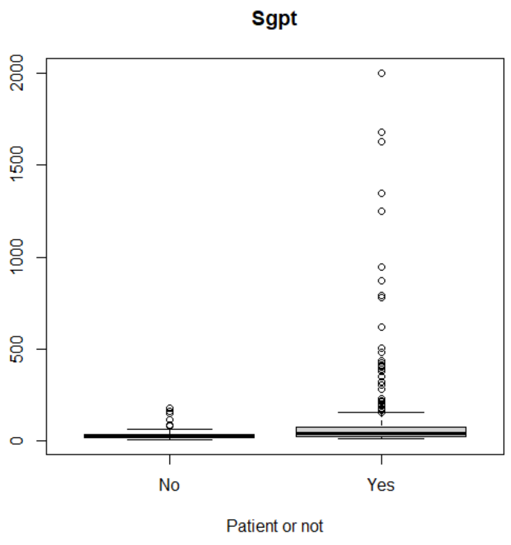


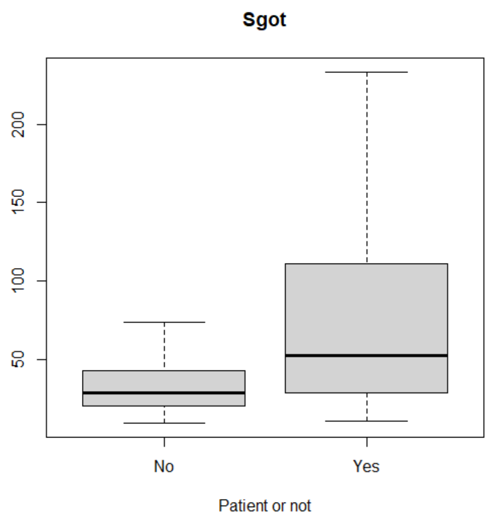


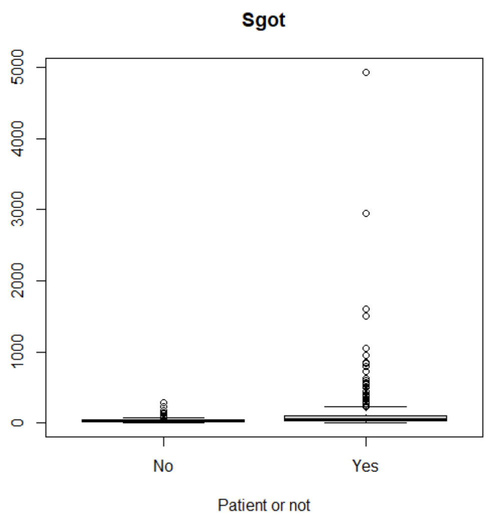

Alkphos
500
400
300
200
100
No
Yes
Patient or not

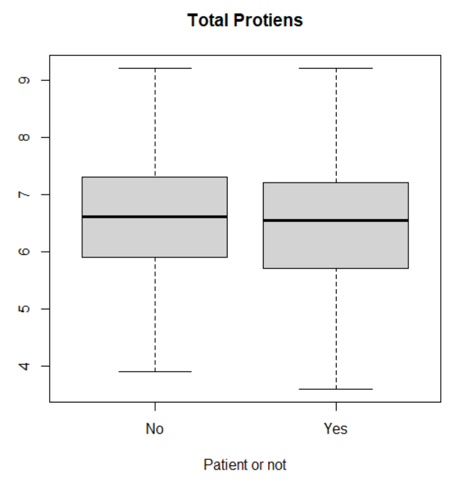

Alkphos
2000
1500
1000
500
0
No
Yes
Patient or not

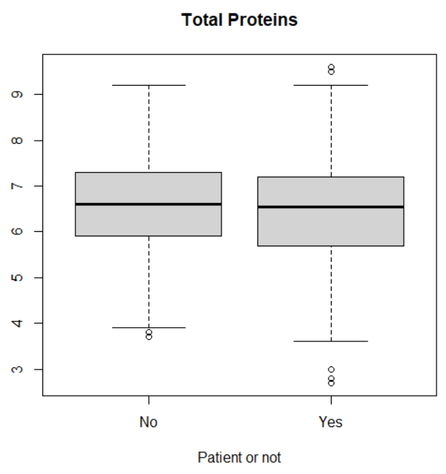

Sgpt
150
100
50
No
Yes
Patient or not

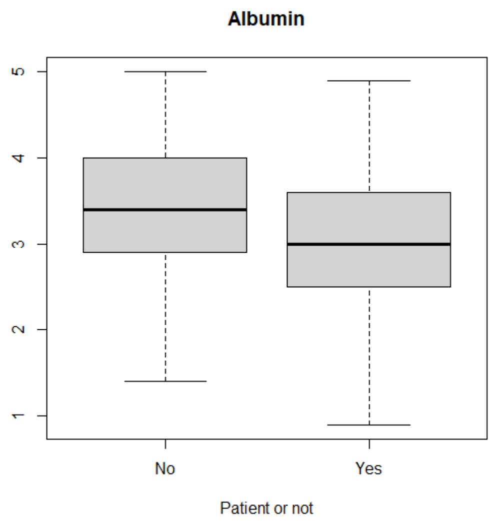

Sgpt
2000
1500
1000
500
0
No
Yes
Patient or not

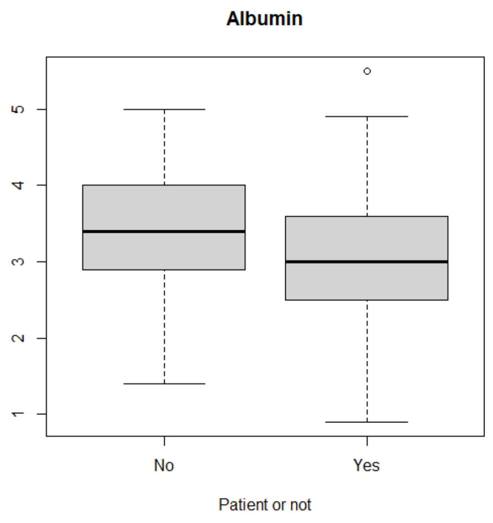

Sgot
200
150
100
50
No
Yes
Patient or not

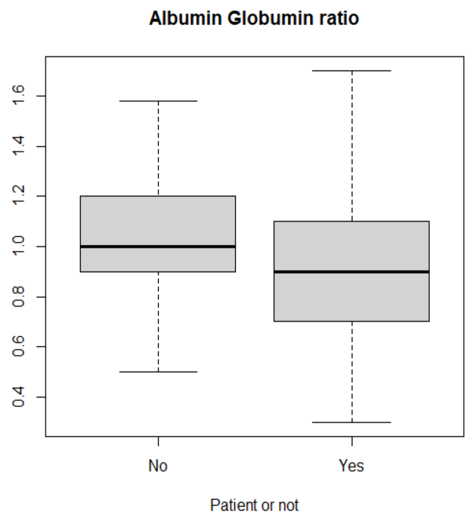

Sgot
5000
4000
3000
2000
1000
0
No
Yes
Patient or not

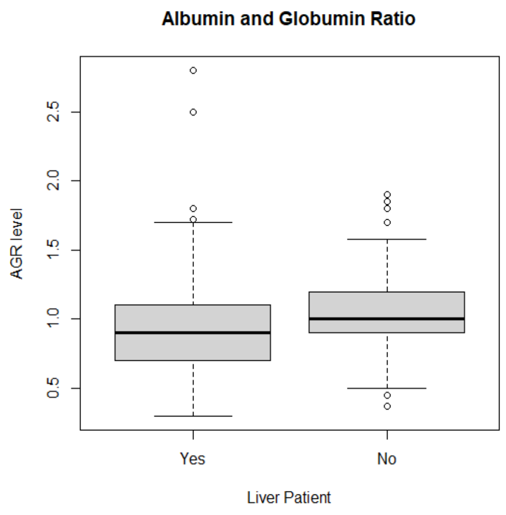

Total Protiens
9
8
7
6
5
4
No
Yes
Patient or not


Total Proteins
9
8
7
6
5
4
3
No
Yes
Patient or not

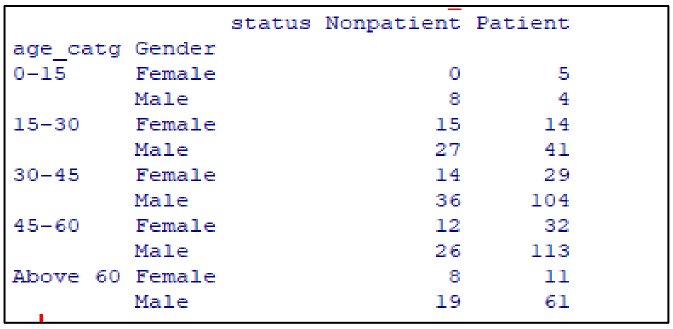

Albumin
5
4
3
2
1
No
Yes
Patient or not


Albumin
5
4
3
2
1
No
Yes
Patient or not

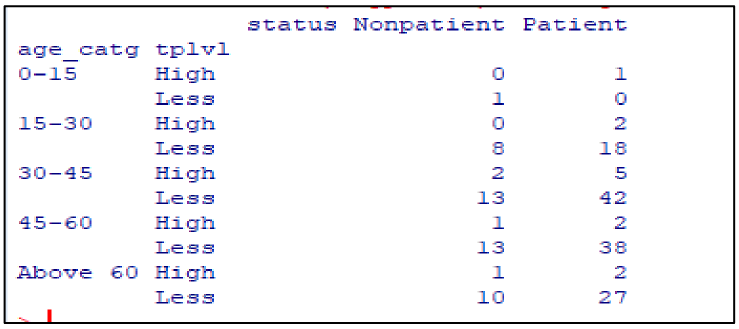

Albumin Globumin ratio
1.6
1.4
1.2
1.0
0.8
0.6
0.4
No
Yes
Patient or not


Albumin and Globumin Ratio
AGR level
2.5
2.0
1.5
1.0
0.5
Yes
No
Liver Patient

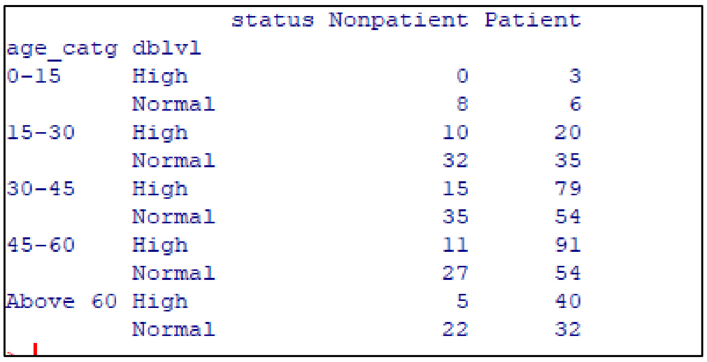
CROSS TABULATIONS:



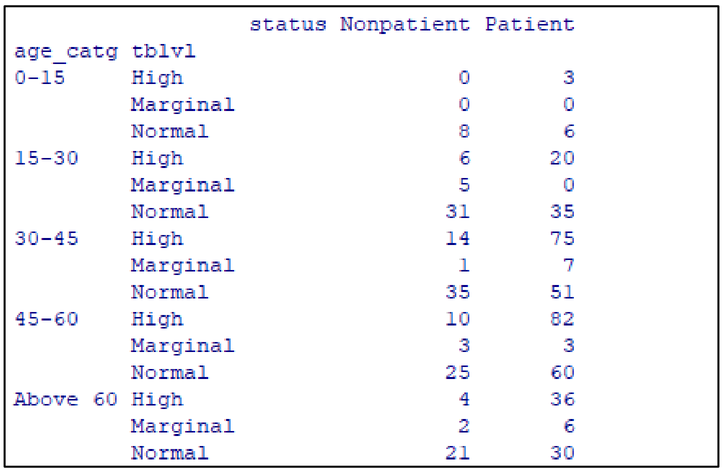
```
                status Nonpatient Patient
age_catg Gender
0-15     Female              0       5
         Male                8       4
15-30    Female             15      14
         Male               27      41
30-45    Female             14      29
         Male               36     104
45-60    Female             12      32
         Male               26     113
Above 60 Female              8      11
         Male               19      61
```



```
                status Nonpatient Patient
age_catg tplvl
0-15     High                0       1
         Less                1       0
15-30    High                0       2
         Less                8      18
30-45    High                2       5
         Less               13      42
45-60    High                1       2
         Less               13      38
Above 60 High                1       2
         Less               10      27
```



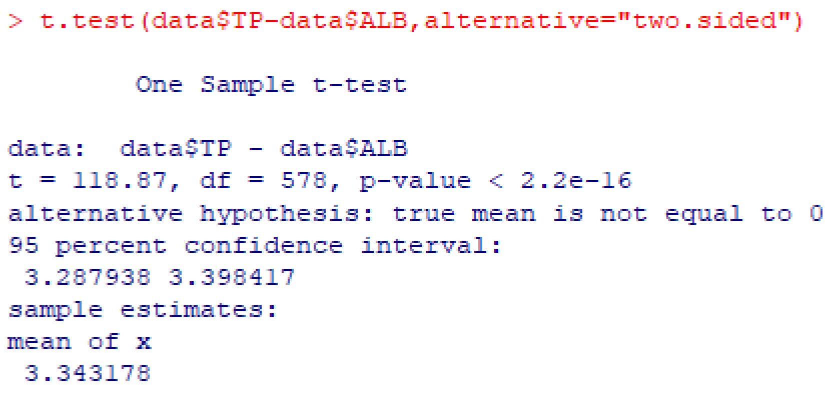
```
                status Nonpatient Patient
age_catg dblvl
0-15     High                0       3
         Normal              8       6
15-30    High               10      20
         Normal             32      35
30-45    High               15      79
         Normal             35      54
45-60    High               11      91
         Normal             27      54
Above 60 High                5      40
         Normal             22      32
```



```
                status Nonpatient Patient
age_catg tblvl
0-15     High                0       3
         Marginal            0       0
         Normal              8       6
15-30    High                6      20
         Marginal            5       0
         Normal             31      35
30-45    High               14      75
         Marginal            1       7
         Normal             35      51
45-60    High               10      82
         Marginal            3       3
         Normal             25      60
Above 60 High                4      36
         Marginal            2       6
         Normal             21      30
```

Since we observe some relation between some particular explanatory variables, below a comparative study of the means of 2groups are shown:

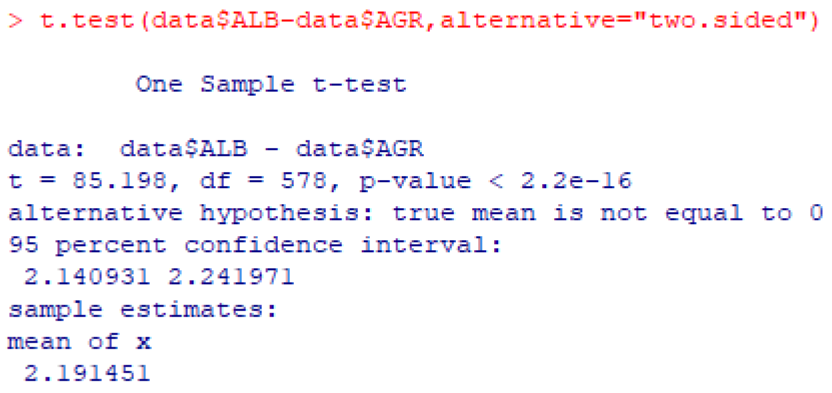
- **t-test on Total Proteins and Albumin**



```
> t.test(data$TP-data$ALB,alternative="two.sided")

        One Sample t-test

data:  data$TP - data$ALB
t = 118.87, df = 578, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 3.287938 3.398417
sample estimates:
mean of x
 3.343178
```

Since p-value<0.01, reject the null hypothesis. The means of the entities are not equal

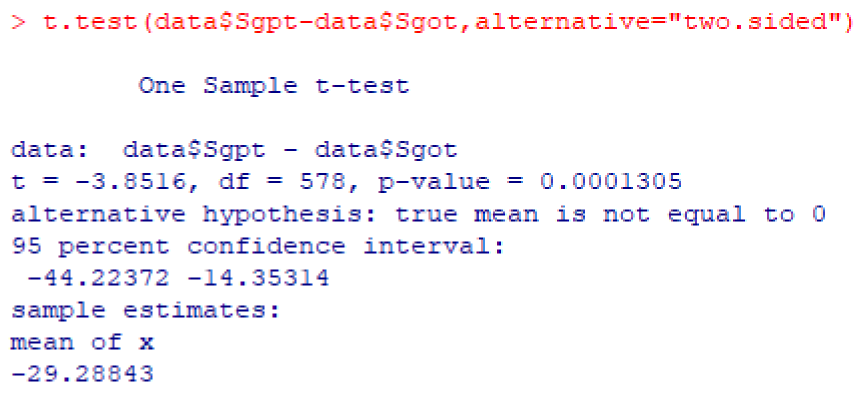
- **t-test on Albumin and Albumin-Globulin Ratio**



```
> t.test(data$ALB-data$AGR,alternative="two.sided")

        One Sample t-test

data:  data$ALB - data$AGR
t = 85.198, df = 578, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.140931 2.241971
sample estimates:
mean of x
 2.191451
```

Since p-value<0.01, reject the null hypothesis. The means of the entities are not equal

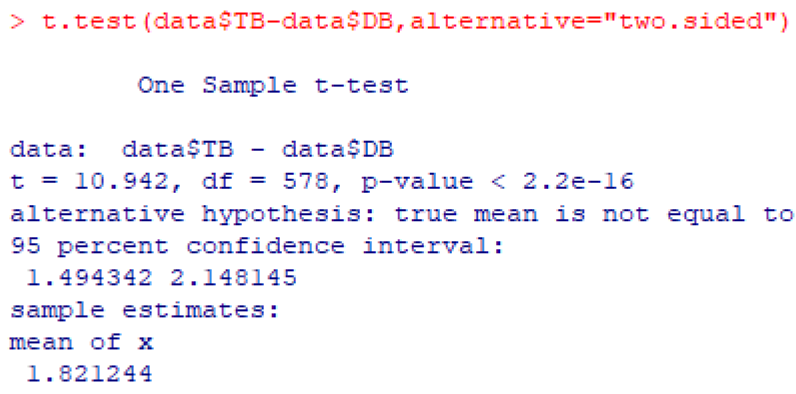
- **t-test on Alamine Aminotransferase and Asparate Aminotransferase**



```
> t.test(data$Sgpt-data$Sgot,alternative="two.sided")

        One Sample t-test

data:  data$Sgpt - data$Sgot
t = -3.8516, df = 578, p-value = 0.0001305
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -44.22372 -14.35314
sample estimates:
mean of x
-29.28843
```

Since p-value<0.01, reject the null hypothesis. The means of the entities are not equal

- **t-test on Total Bilirubin and Direct Bilirubin**

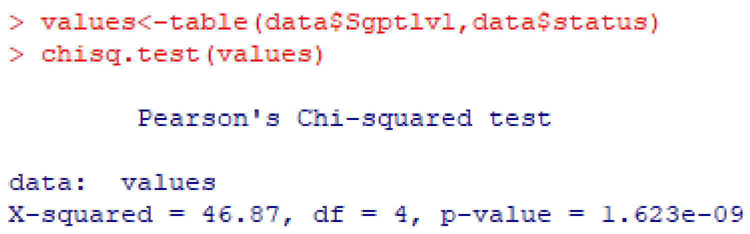


```
> t.test(data$TB-data$DB,alternative="two.sided")

        One Sample t-test

data:  data$TB - data$DB
t = 10.942, df = 578, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 1.494342 2.148145
sample estimates:
mean of x
 1.821244
```

Since p-value<0.01, reject the null hypothesis. The means of the entities are not equal

Now converting the continuous variables into Categorical variables according to the normal medical range and then comparing it the with the dependent variable

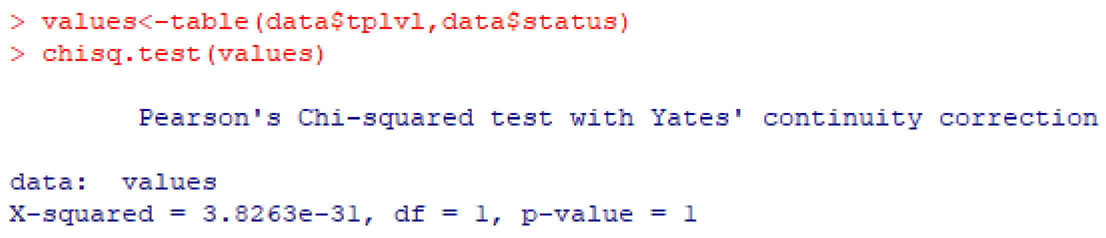
- **Chi-square test on Sgpt level and Patient Status**



```
> values<-table(data$Sgptlvl,data$status)
> chisq.test(values)

        Pearson's Chi-squared test

data:  values
X-squared = 46.87, df = 4, p-value = 1.623e-09
```

Since p-value<0.05, there is a significant difference between the observed and expected frequencies. There is an association between Sqpt levels and Patient status

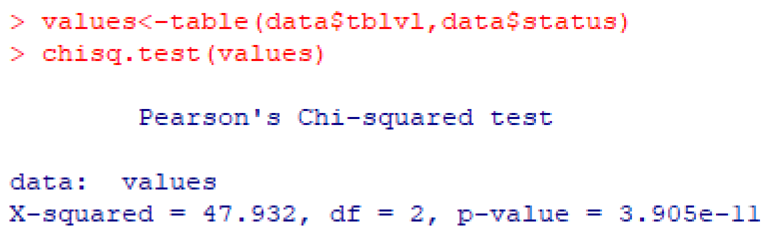
- **Chi-square test on Total Protien level and Patient Status**



```
> values<-table(data$tplvl,data$status)
> chisq.test(values)

        Pearson's Chi-squared test with Yates' continuity correction

data:  values
X-squared = 3.8263e-31, df = 1, p-value = 1
```

Since p-value>0.05, there is no significant difference between the observed and expected frequencies. No association between Total protein levels and Patient status

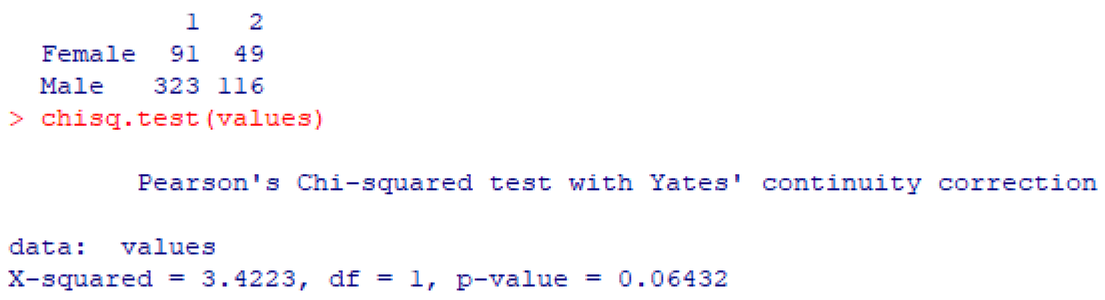
- **Chi-square test on Total Bilirubin level and Patient Status**



```
> values<-table(data$tblvl,data$status)
> chisq.test(values)

        Pearson's Chi-squared test

data:  values
X-squared = 47.932, df = 2, p-value = 3.905e-11
```

Since p-value<0.05, there is a significant difference between the observed and expected frequencies. There is an association between Total Bilirubin levels and Patient status

- **Chi-square test on Gender and Patient Status**



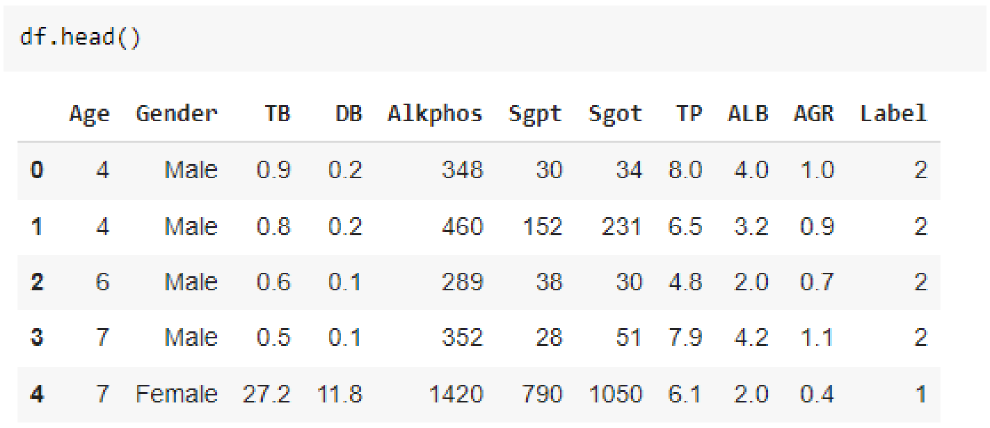
```
            1   2
  Female   91  49
  Male    323 116
> chisq.test(values)

        Pearson's Chi-squared test with Yates' continuity correction

data:  values
X-squared = 3.4223, df = 1, p-value = 0.06432
```

Since p-value>0.05, there is no significant difference between the observed and expected frequencies. No association between Total protein levels and Patient status

MODEL BUILDING

Here we will be building a few models and see which if them is the best. Model metrics will be used to say conclude the ‘Best Model’.

MODELLING DATA:



```
df.head()
```

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGR	Label
0	4	Male	0.9	0.2	348	30	34	8.0	4.0	1.0	2
1	4	Male	0.8	0.2	460	152	231	6.5	3.2	0.9	2
2	6	Male	0.6	0.1	289	38	30	4.8	2.0	0.7	2
3	7	Male	0.5	0.1	352	28	51	7.9	4.2	1.1	2
4	7	Female	27.2	11.8	1420	790	1050	6.1	2.0	0.4	1

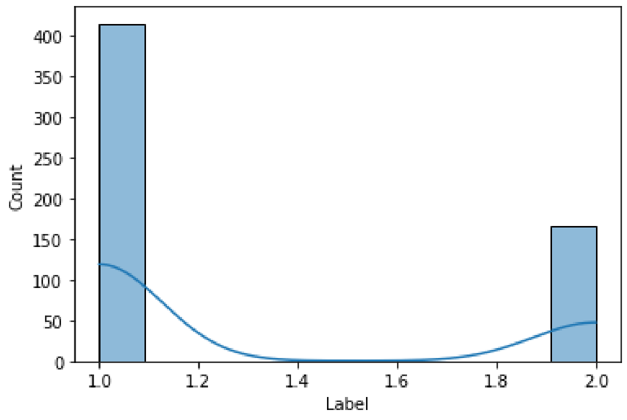
What is Model Building?

Building a machine learning model involves several steps, including data preprocessing, model selection, training, evaluation, fine-tuning, and deployment. Each of these steps is critical to building an accurate and robust model that can make reliable predictions on new data.

Data Set Information:

This data set contains 416 liver patient records and 167 non liver patient records. The data set was collected from north east of Andhra Pradesh, India. Selector is a class label used to divide into groups (liver patient or not). This data set contains 441 male patient records and 142 female patient records. Any patient whose age exceeded 89 is listed as being of age "90".

```python
import seaborn as sns

sns.histplot(df["Label"],kde=True);
```

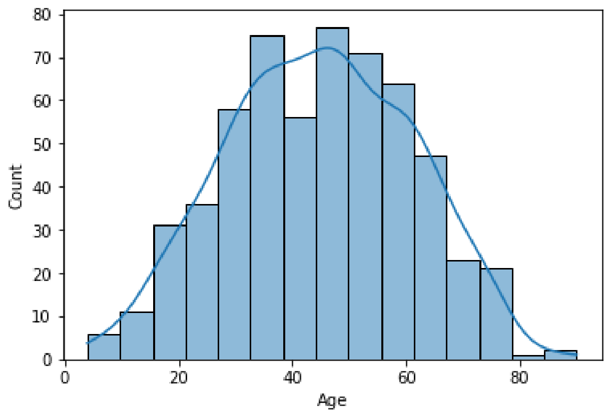


Using Seaborn Library:

Seaborn is a popular data visualization library for Python that is built on top of the matplotlib library. Seaborn provides a high-level interface for creating informative and visually appealing statistical graphics. It is particularly useful for exploring and visualizing complex data sets. Below are a few examples of how it is used.

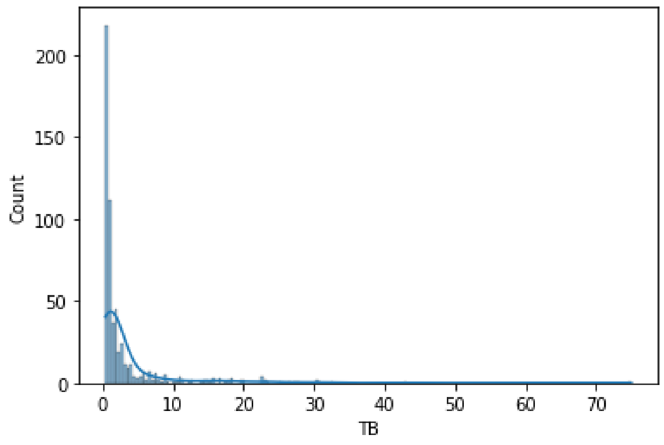
```python
sns.histplot(X["Age"],kde=True)
```

```
<Axes: xlabel='Age', ylabel='Count'>
```



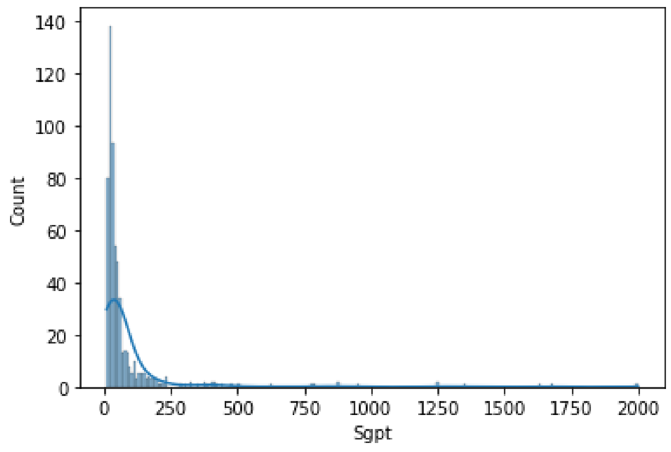
```
sns.histplot(X["TB"],kde=True)
```

```
<Axes: xlabel='TB', ylabel='Count'>
```



```
sns.histplot(X["Sgpt"],kde=True)
```

```
<Axes: xlabel='Sgpt', ylabel='Count'>
```



```
[ ]  from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.10,random_state=42)
```

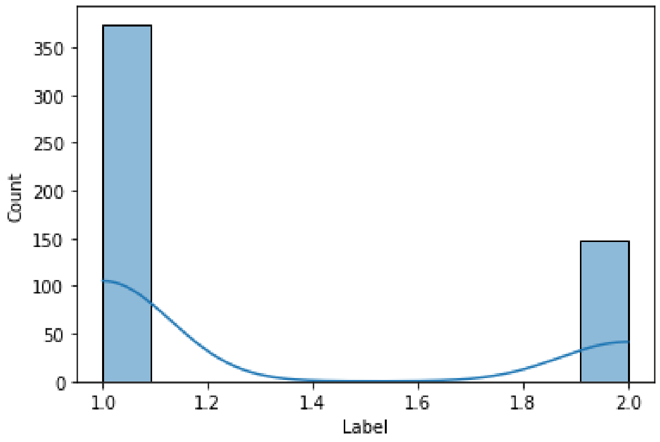
sklearn.model_selection is a module in the popular Python library scikit-learn that provides a range of tools for data splitting, cross-validation, and hyperparameter tuning for machine learning models.

The model_selection module provides several classes and functions that allow you to split data into training and testing sets, perform cross-validation, tune hyperparameters, and evaluate model performance.

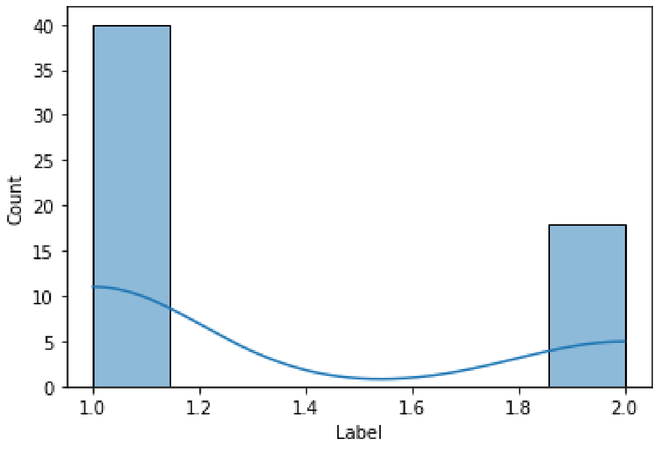
We have divided the data into 2 sets: Training and testing data

```
[ ]  from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.10,random_state=42)
```

```
[ ]  sns.histplot(y_train,kde=True);
```



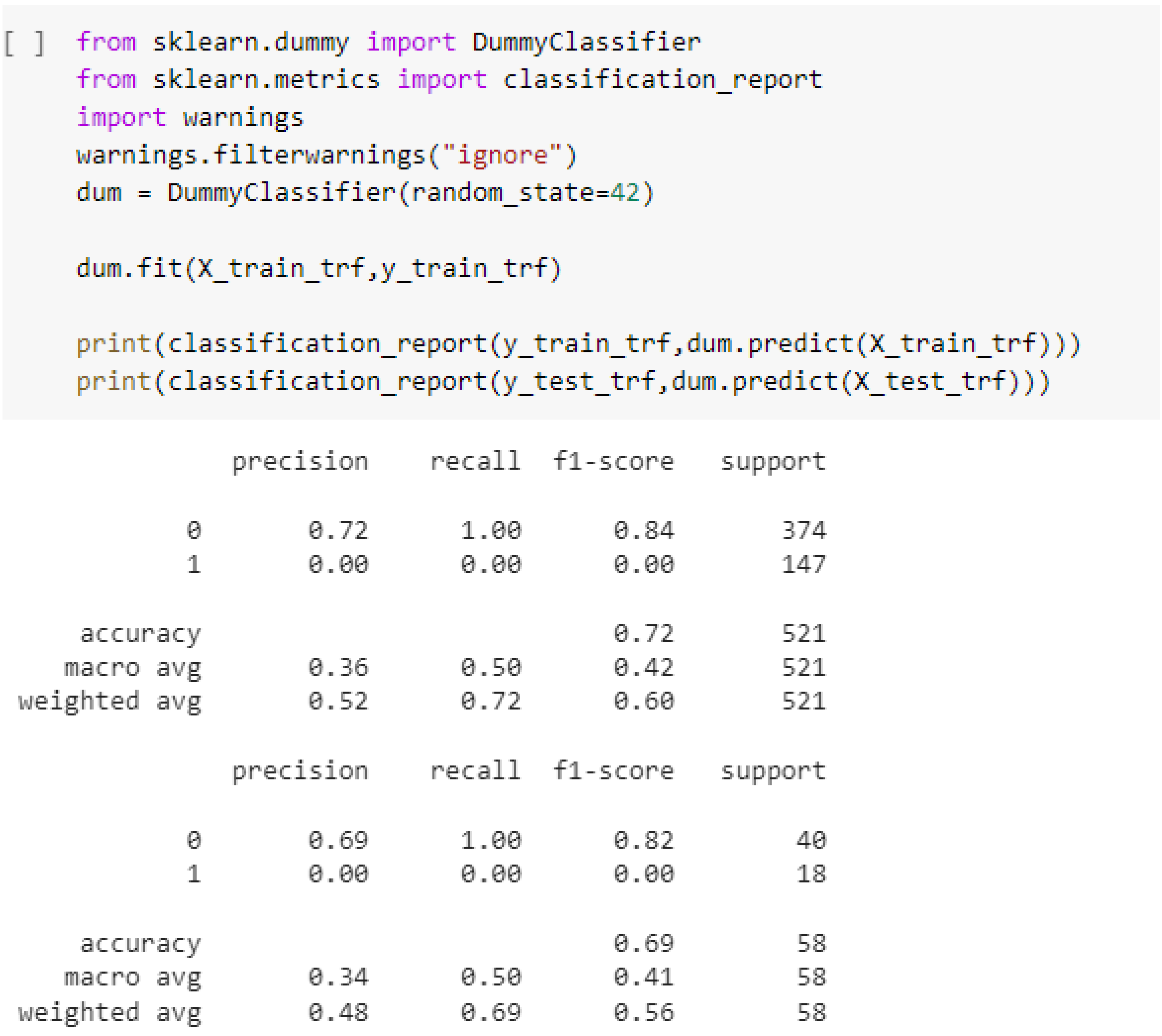
```
[ ]  sns.histplot(y_test,kde=True);
```



A ColumnTransformer is a transformer from the scikit-learn library that allows you to apply different pre-processing steps to different columns of a dataset. It is useful when you have a dataset with multiple columns and you need to pre-process each column differently before feeding it to a machine learning model.

```
[ ]  from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
     from sklearn.compose import ColumnTransformer

     ct = ColumnTransformer([('std',StandardScaler(),["Age", "TB", "DB", "Alkphos",  "Sgpt", "Sgot", "TP", "ALB",  "AGR"]),
                             ('ohe',OneHotEncoder(sparse=False),["Gender"])])
```

BASELINE MODEL



```
[ ]  from sklearn.dummy import DummyClassifier
     from sklearn.metrics import classification_report
     import warnings
     warnings.filterwarnings("ignore")
     dum = DummyClassifier(random_state=42)

     dum.fit(X_train_trf,y_train_trf)

     print(classification_report(y_train_trf,dum.predict(X_train_trf)))
     print(classification_report(y_test_trf,dum.predict(X_test_trf)))
```

```
              precision    recall  f1-score   support

           0       0.72      1.00      0.84       374
           1       0.00      0.00      0.00       147

    accuracy                           0.72       521
   macro avg       0.36      0.50      0.42       521
weighted avg       0.52      0.72      0.60       521

              precision    recall  f1-score   support

           0       0.69      1.00      0.82        40
           1       0.00      0.00      0.00        18

    accuracy                           0.69        58
   macro avg       0.34      0.50      0.41        58
weighted avg       0.48      0.69      0.56        58
```

Since this is a Baseline Model, it randomly assigns the response variable. We have to make a better model that the Baseline Model

LOGISTIC REGRESSION (Default Model)

```
[ ]  from sklearn.linear_model import LogisticRegression

     lr = LogisticRegression(random_state=42)

     lr.fit(X_train_trf,y_train_trf)

     print(classification_report(y_train_trf,lr.predict(X_train_trf)))
     print(classification_report(y_test_trf,lr.predict(X_test_trf)))
```

```
              precision    recall  f1-score   support

           0       0.76      0.93      0.84       374
           1       0.60      0.27      0.37       147

    accuracy                           0.74       521
   macro avg       0.68      0.60      0.60       521
weighted avg       0.72      0.74      0.71       521

              precision    recall  f1-score   support

           0       0.69      0.90      0.78        40
           1       0.33      0.11      0.17        18

    accuracy                           0.66        58
   macro avg       0.51      0.51      0.47        58
weighted avg       0.58      0.66      0.59        58
```

We apply Binary Logistic Regression Model here. We observe that the precision for the training and testing data is better than that of the Baseline Model, but we need to find a better model than this.

RANDOM FOREST (Default Model)

```
[ ]  from sklearn.ensemble import RandomForestClassifier

     rf = RandomForestClassifier(random_state=42)

     rf.fit(X_train_trf,y_train_trf)

     print(classification_report(y_train_trf,rf.predict(X_train_trf)))
     print(classification_report(y_test_trf,rf.predict(X_test_trf)))
```

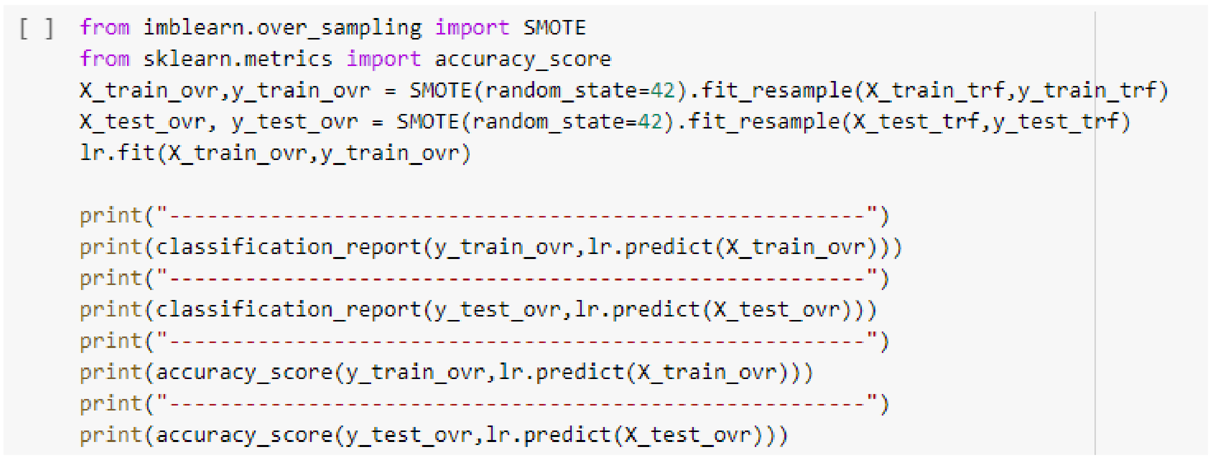
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       374
           1       1.00      1.00      1.00       147

    accuracy                           1.00       521
   macro avg       1.00      1.00      1.00       521
weighted avg       1.00      1.00      1.00       521

              precision    recall  f1-score   support

           0       0.69      0.88      0.77        40
           1       0.29      0.11      0.16        18

    accuracy                           0.64        58
   macro avg       0.49      0.49      0.46        58
weighted avg       0.56      0.64      0.58        58
```

The Random Forest Model shows a very high precision for training data but a very less accuracy for testing data. It should give a similar type of accuracy. So it is not a better model than Logistic Regression

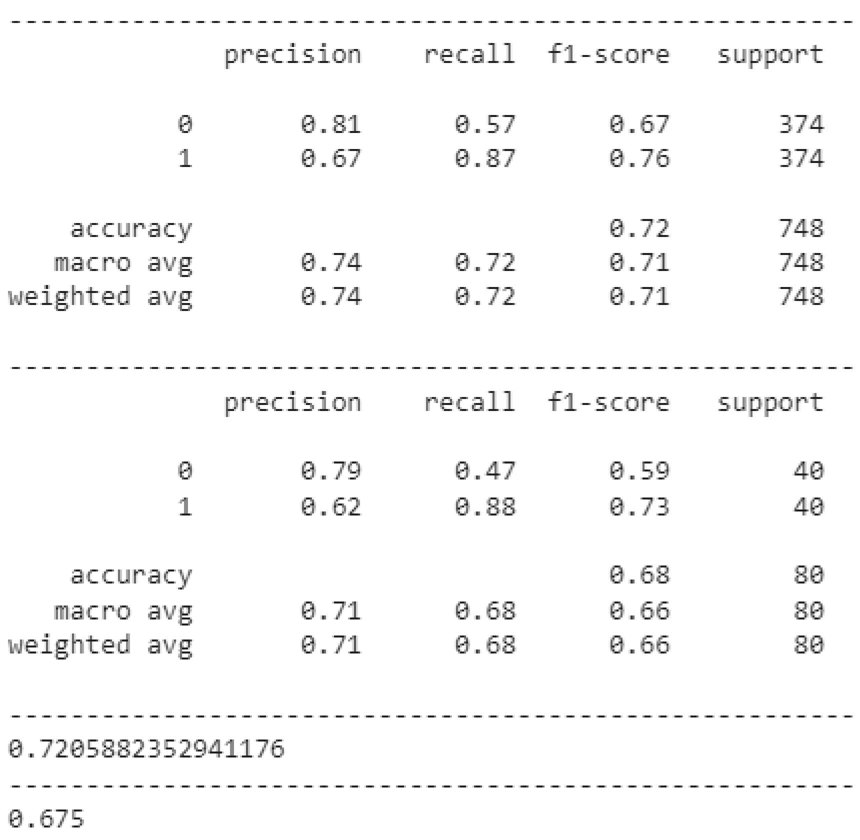
The data observation is not equal for the number of patients and number of non-patients, which might be a big problem in building a good model. The machine must have enough data points for the patients and non-patients to learn. Hence we introduce the concept of oversampling.

Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a data set.

LOGISTIC REGRESSION with Oversampling



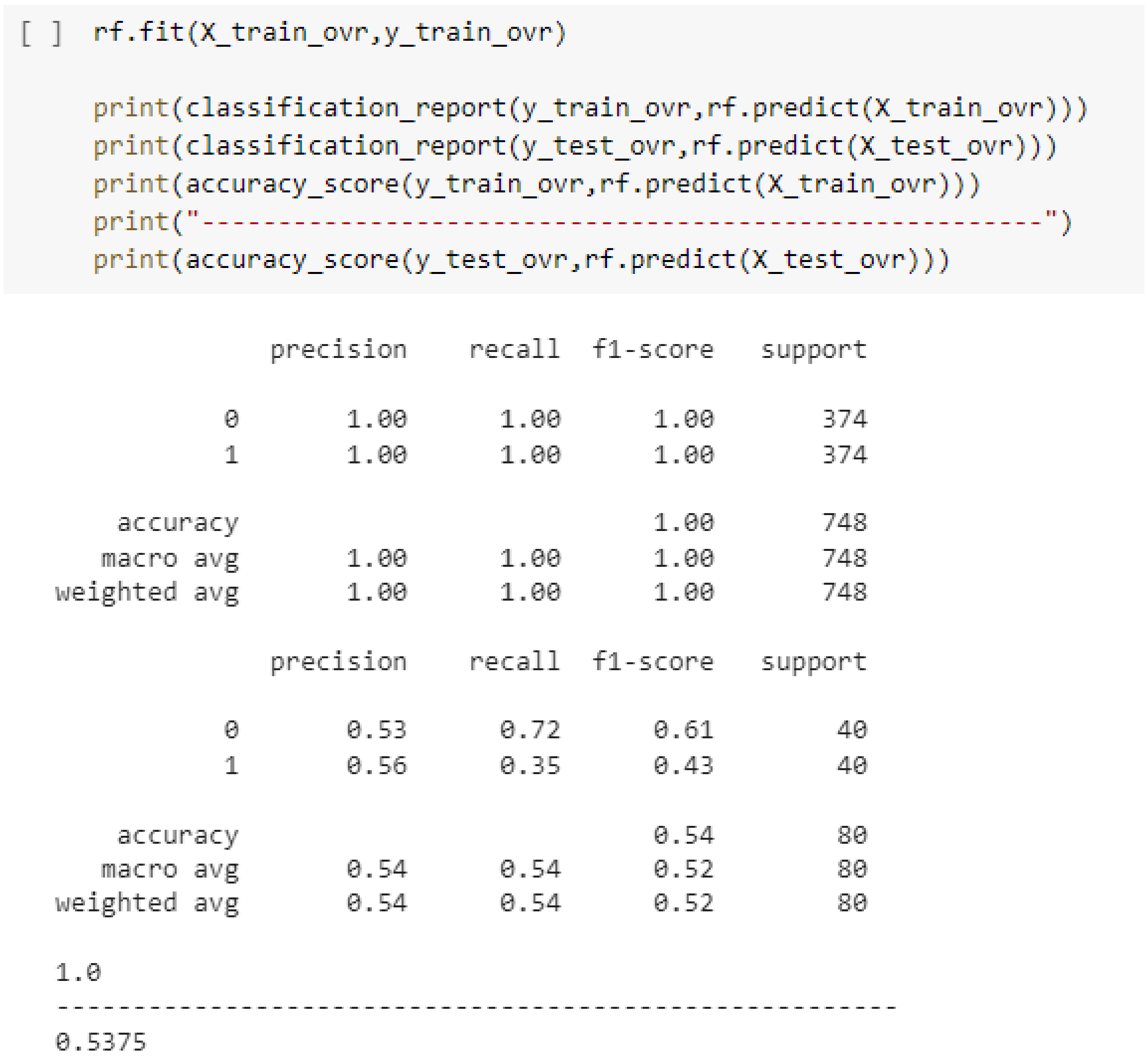
```
[ ]  from imblearn.over_sampling import SMOTE
     from sklearn.metrics import accuracy_score
     X_train_ovr,y_train_ovr = SMOTE(random_state=42).fit_resample(X_train_trf,y_train_trf)
     X_test_ovr, y_test_ovr = SMOTE(random_state=42).fit_resample(X_test_trf,y_test_trf)
     lr.fit(X_train_ovr,y_train_ovr)

     print("-------------------------------------------------------")
     print(classification_report(y_train_ovr,lr.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(classification_report(y_test_ovr,lr.predict(X_test_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_train_ovr,lr.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,lr.predict(X_test_ovr)))
```



```
-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.81      0.57      0.67       374
           1       0.67      0.87      0.76       374

    accuracy                           0.72       748
   macro avg       0.74      0.72      0.71       748
weighted avg       0.74      0.72      0.71       748

-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.79      0.47      0.59        40
           1       0.62      0.88      0.73        40

    accuracy                           0.68        80
   macro avg       0.71      0.68      0.66        80
weighted avg       0.71      0.68      0.66        80

-------------------------------------------------------
0.7205882352941176
-------------------------------------------------------
0.675
```

You can observe the number of 1’s in the training data set has increased from 147 to 374 and in testing data from 18 to 40. We get a far better model than the previous models that we have considered.

RANDOM FOREST with Oversampling



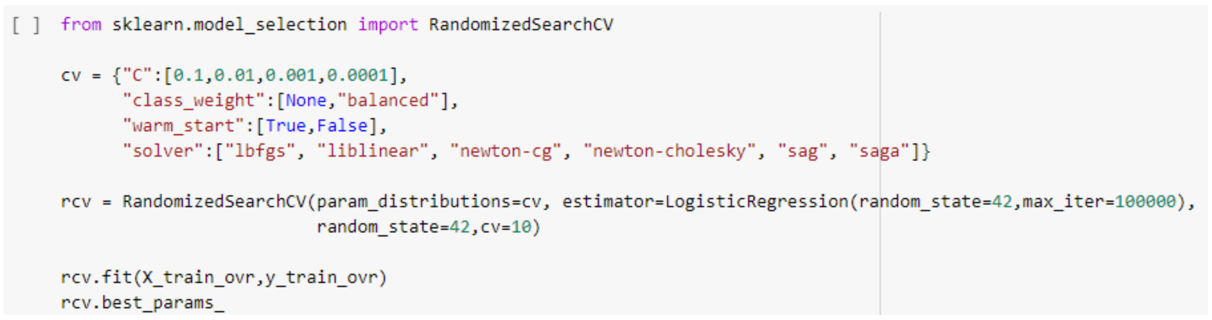
```
[ ]  rf.fit(X_train_ovr,y_train_ovr)

     print(classification_report(y_train_ovr,rf.predict(X_train_ovr)))
     print(classification_report(y_test_ovr,rf.predict(X_test_ovr)))
     print(accuracy_score(y_train_ovr,rf.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,rf.predict(X_test_ovr)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       374
           1       1.00      1.00      1.00       374

    accuracy                           1.00       748
   macro avg       1.00      1.00      1.00       748
weighted avg       1.00      1.00      1.00       748

              precision    recall  f1-score   support

           0       0.53      0.72      0.61        40
           1       0.56      0.35      0.43        40

    accuracy                           0.54        80
   macro avg       0.54      0.54      0.52        80
weighted avg       0.54      0.54      0.52        80

1.0
-------------------------------------------------------
0.5375
```

Applying Oversampling to Random Forest didn’t improve the accuracy. Hence we can conclude that Logistic Regression Model is better than Decision tree for this dataset.

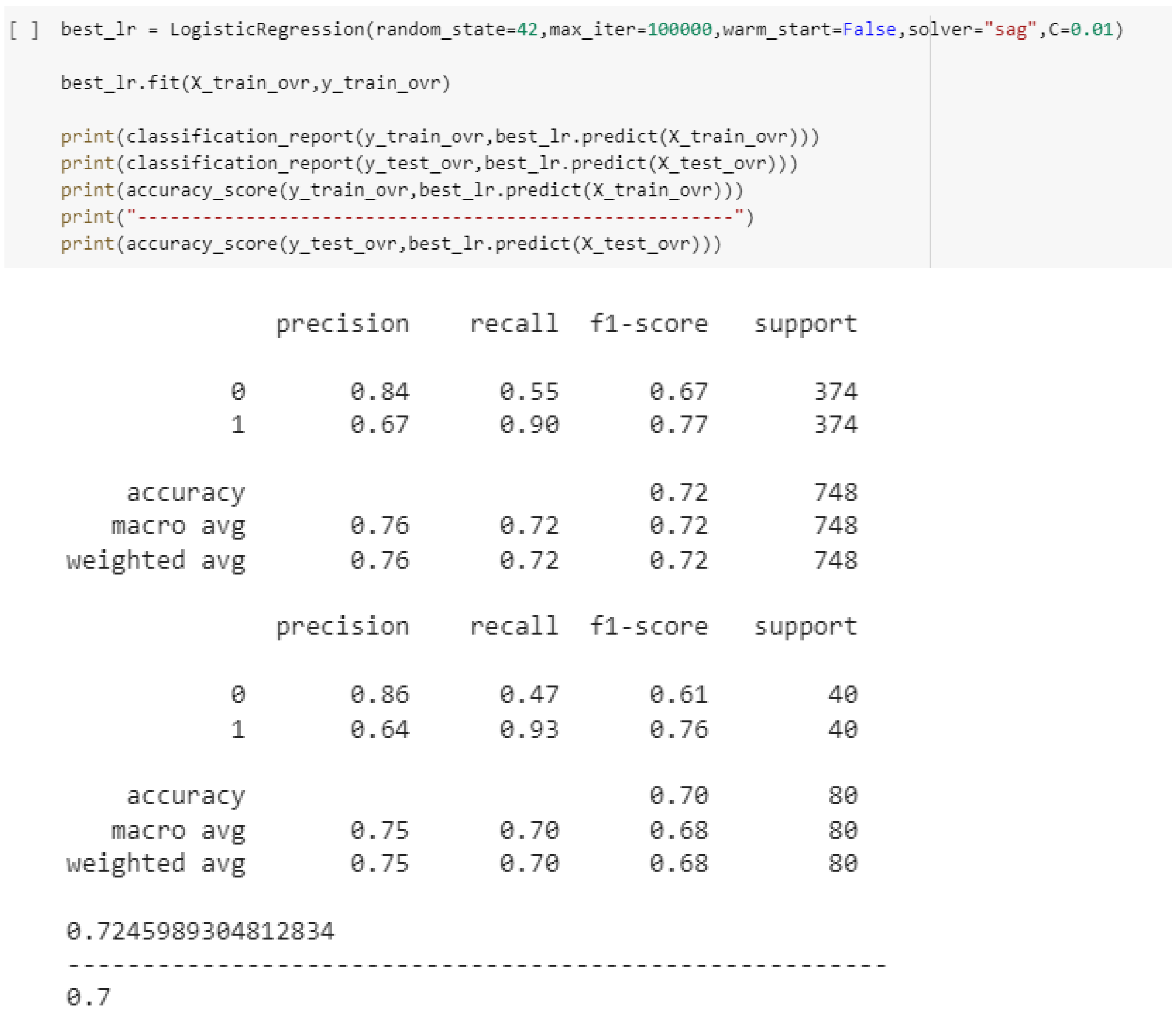
MODEL TUNING

Model tuning, also known as hyperparameter optimization, is the process of selecting the best set of hyperparameters for a machine learning algorithm in order to maximize its performance on a given task. Hyperparameters are parameters of the model that are not learned from the data, but rather set before the training process begins, such as learning rate, batch size, and regularization strength.



```
[ ] from sklearn.model_selection import RandomizedSearchCV

    cv = {"C":[0.1,0.01,0.001,0.0001],
          "class_weight":[None,"balanced"],
          "warm_start":[True,False],
          "solver":["lbfgs", "liblinear", "newton-cg", "newton-cholesky", "sag", "saga"]}

    rcv = RandomizedSearchCV(param_distributions=cv, estimator=LogisticRegression(random_state=42,max_iter=100000),
                             random_state=42,cv=10)

    rcv.fit(X_train_ovr,y_train_ovr)
    rcv.best_params_
```

Logistic Regression (Tuned) with Oversampling



```
[ ] best_lr = LogisticRegression(random_state=42,max_iter=100000,warm_start=False,solver="sag",C=0.01)

    best_lr.fit(X_train_ovr,y_train_ovr)

    print(classification_report(y_train_ovr,best_lr.predict(X_train_ovr)))
    print(classification_report(y_test_ovr,best_lr.predict(X_test_ovr)))
    print(accuracy_score(y_train_ovr,best_lr.predict(X_train_ovr)))
    print("-------------------------------------------------------")
    print(accuracy_score(y_test_ovr,best_lr.predict(X_test_ovr)))
```

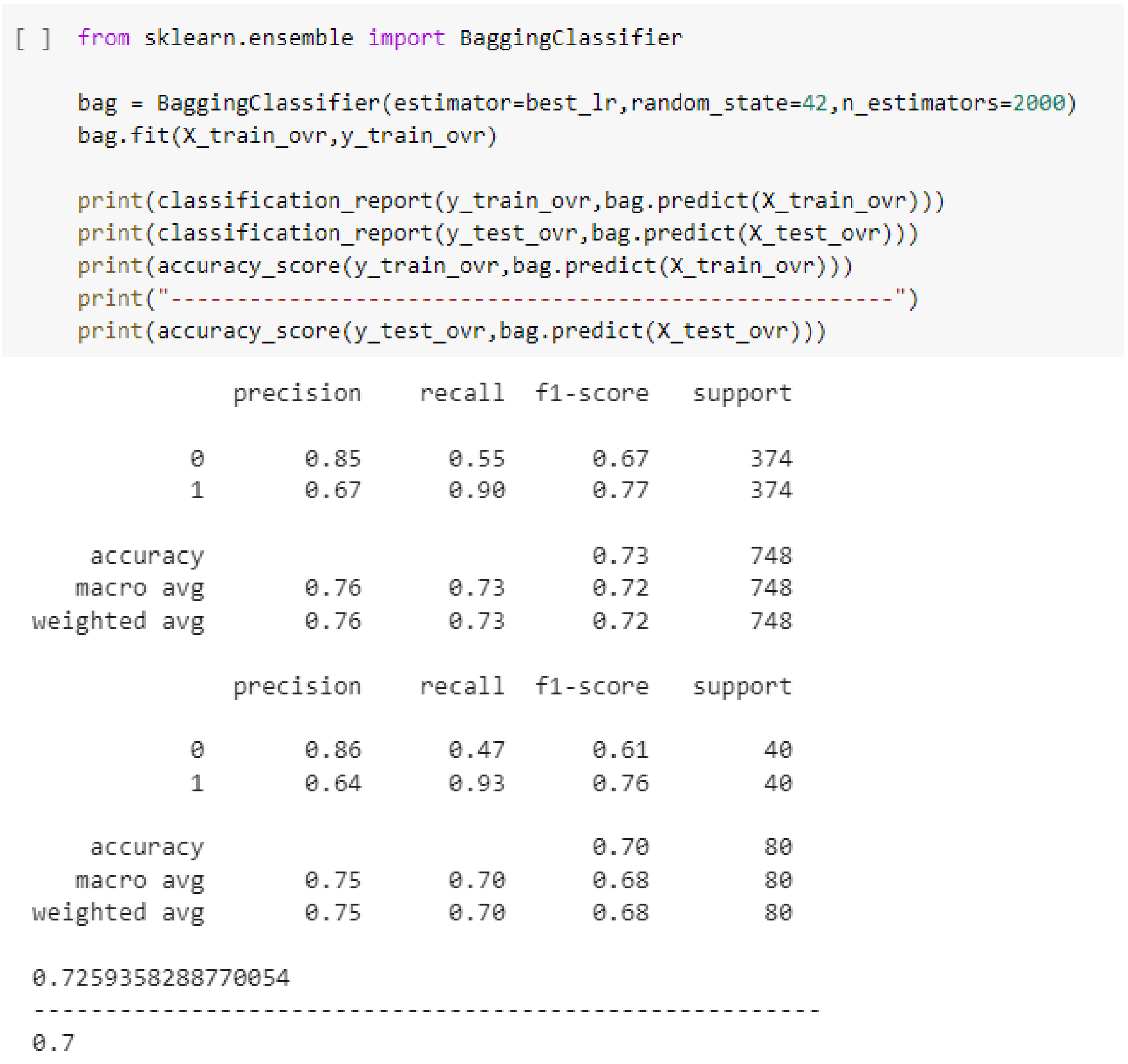
```
              precision    recall  f1-score   support

           0       0.84      0.55      0.67       374
           1       0.67      0.90      0.77       374

    accuracy                           0.72       748
   macro avg       0.76      0.72      0.72       748
weighted avg       0.76      0.72      0.72       748

              precision    recall  f1-score   support

           0       0.86      0.47      0.61        40
           1       0.64      0.93      0.76        40

    accuracy                           0.70        80
   macro avg       0.75      0.70      0.68        80
weighted avg       0.75      0.70      0.68        80

0.7245989304812834
-------------------------------------------------------
0.7
```

After tuning, we can observe the accuracy for the Training data and the testing data are both very close and are high too. So we choose this as the best model built so far.

BAGGING

A bagging classifier, or bootstrap aggregating classifier, is an ensemble learning method in which multiple base classifiers are trained on bootstrapped samples of the training data and their predictions are aggregated to make a final prediction. Bagging is a popular method for reducing the variance of a model and improving its generalization performance.

Using Bagging with Logistic Regression Estimator



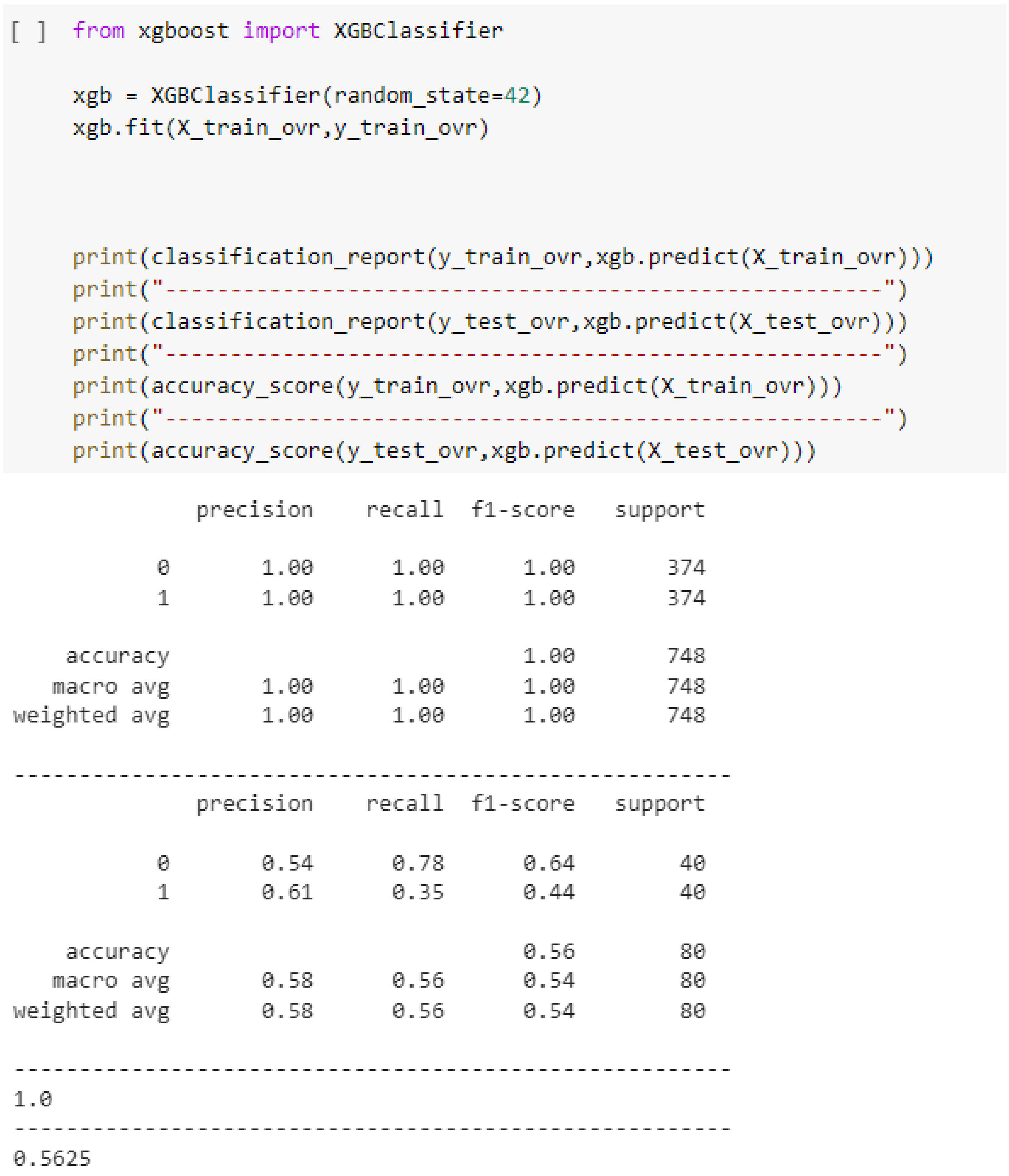
```
[ ]  from sklearn.ensemble import BaggingClassifier

     bag = BaggingClassifier(estimator=best_lr,random_state=42,n_estimators=2000)
     bag.fit(X_train_ovr,y_train_ovr)

     print(classification_report(y_train_ovr,bag.predict(X_train_ovr)))
     print(classification_report(y_test_ovr,bag.predict(X_test_ovr)))
     print(accuracy_score(y_train_ovr,bag.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,bag.predict(X_test_ovr)))
```

```
              precision    recall  f1-score   support

           0       0.85      0.55      0.67       374
           1       0.67      0.90      0.77       374

    accuracy                           0.73       748
   macro avg       0.76      0.73      0.72       748
weighted avg       0.76      0.73      0.72       748

              precision    recall  f1-score   support

           0       0.86      0.47      0.61        40
           1       0.64      0.93      0.76        40

    accuracy                           0.70        80
   macro avg       0.75      0.70      0.68        80
weighted avg       0.75      0.70      0.68        80

0.7259358288770054
-------------------------------------------------------
0.7
```

Bagging didn’t improve the accuracy of the training data much and of the testing data set at all.

XGB

XGBoost (short for eXtreme Gradient Boosting) is a powerful and popular machine learning algorithm used for supervised learning tasks such as classification and regression. It is a boosting algorithm that builds an ensemble of weak prediction models, which are typically decision trees, and combines them to create a strong model.

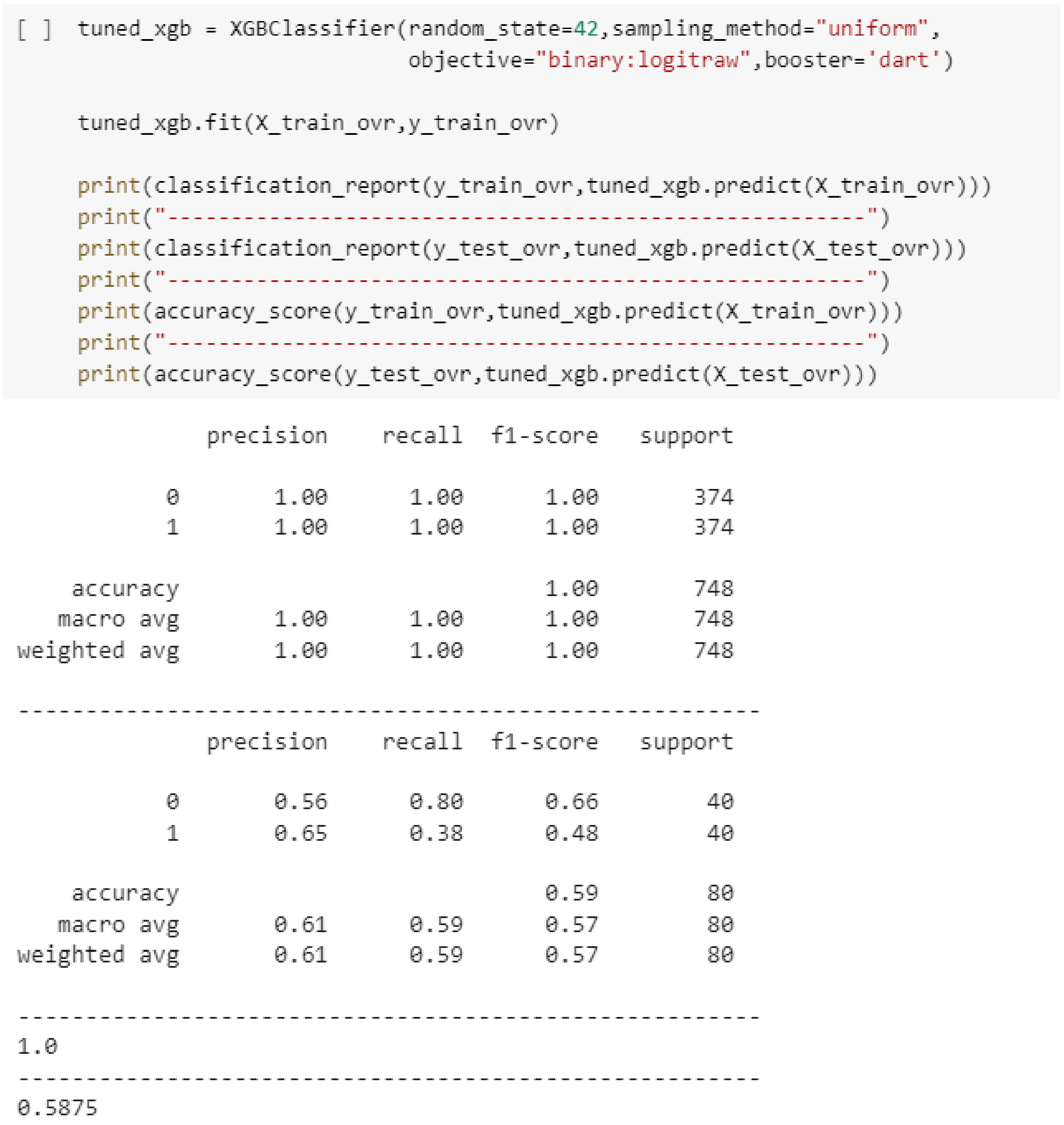


```
[ ]  from xgboost import XGBClassifier

     xgb = XGBClassifier(random_state=42)
     xgb.fit(X_train_ovr,y_train_ovr)



     print(classification_report(y_train_ovr,xgb.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(classification_report(y_test_ovr,xgb.predict(X_test_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_train_ovr,xgb.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,xgb.predict(X_test_ovr)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       374
           1       1.00      1.00      1.00       374

    accuracy                           1.00       748
   macro avg       1.00      1.00      1.00       748
weighted avg       1.00      1.00      1.00       748

-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.54      0.78      0.64        40
           1       0.61      0.35      0.44        40

    accuracy                           0.56        80
   macro avg       0.58      0.56      0.54        80
weighted avg       0.58      0.56      0.54        80

-------------------------------------------------------
1.0
-------------------------------------------------------
0.5625
```

XGB gives a model where the accuracy for Training data is very high and for that of testing data is less. There is a huge difference. Thus it is not considered to be a good model.

Tuned XGB



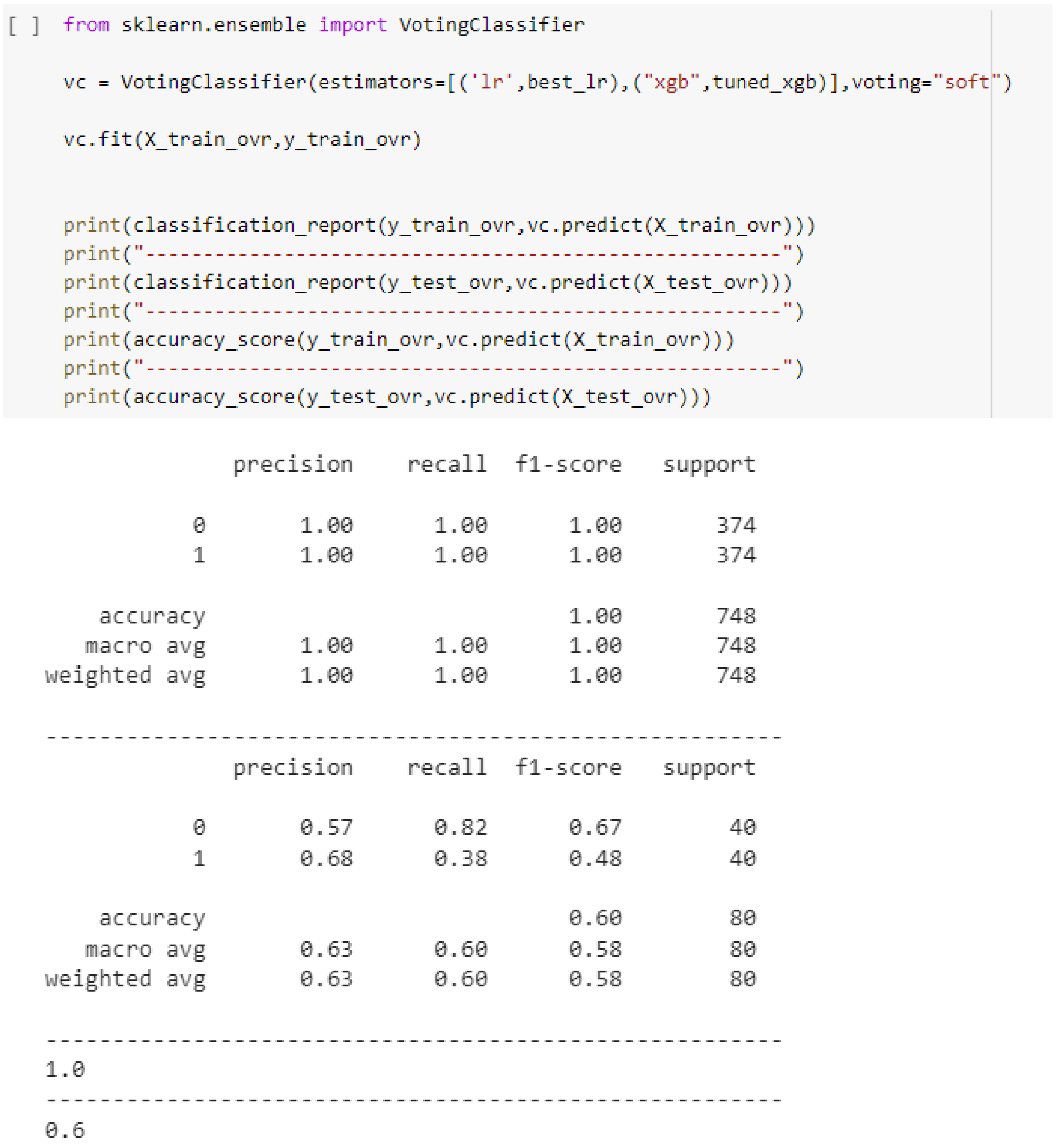
```
[ ]  tuned_xgb = XGBClassifier(random_state=42,sampling_method="uniform",
                               objective="binary:logitraw",booster='dart')

     tuned_xgb.fit(X_train_ovr,y_train_ovr)

     print(classification_report(y_train_ovr,tuned_xgb.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(classification_report(y_test_ovr,tuned_xgb.predict(X_test_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_train_ovr,tuned_xgb.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,tuned_xgb.predict(X_test_ovr)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       374
           1       1.00      1.00      1.00       374

    accuracy                           1.00       748
   macro avg       1.00      1.00      1.00       748
weighted avg       1.00      1.00      1.00       748

-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.56      0.80      0.66        40
           1       0.65      0.38      0.48        40

    accuracy                           0.59        80
   macro avg       0.61      0.59      0.57        80
weighted avg       0.61      0.59      0.57        80

-------------------------------------------------------
1.0
-------------------------------------------------------
0.5875
```

A major difference between the accuracy of the testing data and of the training data is visible again. Tuning the XGB model didn't offer much help.

VOTING CLASSIFIER

A Voting Classifier is a type of ensemble learning algorithm in machine learning that combines multiple models (classifiers) to improve the accuracy and robustness of the final prediction. The idea is to combine the predictions of multiple models and use a majority vote to determine the final prediction.



```
[ ]  from sklearn.ensemble import VotingClassifier

     vc = VotingClassifier(estimators=[('lr',best_lr),("xgb",tuned_xgb)],voting="soft")

     vc.fit(X_train_ovr,y_train_ovr)


     print(classification_report(y_train_ovr,vc.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(classification_report(y_test_ovr,vc.predict(X_test_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_train_ovr,vc.predict(X_train_ovr)))
     print("-------------------------------------------------------")
     print(accuracy_score(y_test_ovr,vc.predict(X_test_ovr)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       374
           1       1.00      1.00      1.00       374

    accuracy                           1.00       748
   macro avg       1.00      1.00      1.00       748
weighted avg       1.00      1.00      1.00       748

-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.57      0.82      0.67        40
           1       0.68      0.38      0.48        40

    accuracy                           0.60        80
   macro avg       0.63      0.60      0.58        80
weighted avg       0.63      0.60      0.58        80

-------------------------------------------------------
1.0
-------------------------------------------------------
0.6
```

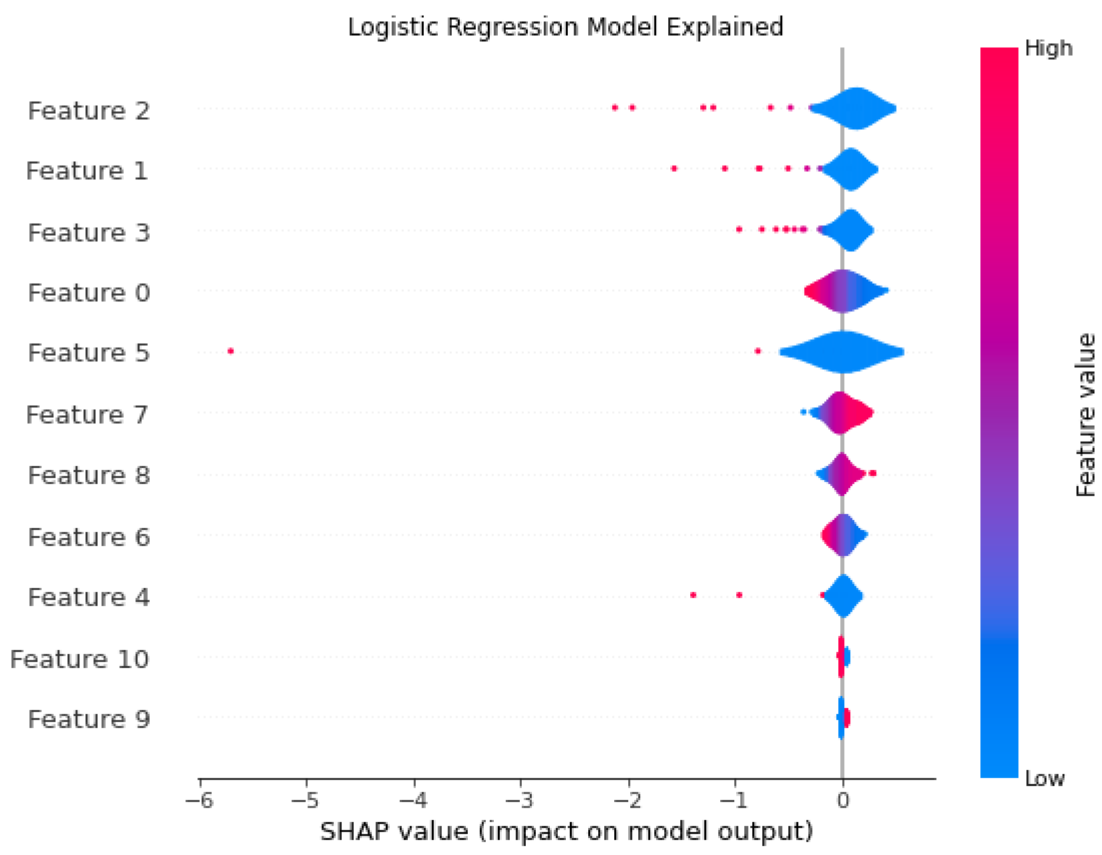
A voting classifier improved the tuned XGB model very little but still the difference between the accuracies of testing and training remains almost the same.

SHAP

SHAP (SHapley Additive exPlanations) is a framework used in machine learning for explaining the output of a model by attributing the importance of each feature in the input to the output.

SHAP provides a way to quantify the contribution of each feature to a model's prediction for a given input. It takes into account the interactions between features and provides a more accurate and realistic estimate of feature importance compared to other methods like feature importance, which only considers the contribution of individual features in isolation.

```
[33] explainer = shap.LinearExplainer(best_lr, X_train_ovr, feature_dependence="independent",
                                     feature_names=list(X.columns))
     shap_values = explainer.shap_values(X_test_ovr)
```



```
[ ]  X.columns

     Index(['Age', 'Gender', 'TB', 'DB', 'Alkphos', 'Sgpt', 'Sgot', 'TP', 'ALB',
            'AGR'],
           dtype='object')
```

CONCLUSION

CONCLUSION 1:

So, here we can see feature 7: TP & feature 8: ALB have highest positive contirbution towards theoverall model performance. Whereas feature 5: Sgpt has the highest negative contribution towards the overall model performance. Which means

TP: Total Protiens, ALB: Albumin, Sgpt: serum glutamate pyruvate transaminase are the most important factors in the determination of Liver Disease

CONCLUSION 2:

The Tuned Logistic Regression Model with oversampling is considered the best model among all the models that we have built because of its high accuracies of the training and testing data and their least difference.

Bagging applied to the above model didn’t improve the accuracy, hence we are not considering that the Tuned Logistic regression with oversampling (Bagging applied) as the best one.