Title:

“ANALYSIS OF LIVER PATIENT DATASET A PROJECT REPORT.”

Project Review-2.

DATA VISULIZATION.

Team Members:

1. Fenil Jain-18BCE0435.
2. D.Jeevan Kishore- 18BCE0160.
3. S.S.ShreyasRaj-18BCE2019.
4. I.Prashanth-18BCE0161.

Submitted To:

Dr. Dilip Kumar Choubey Assistant Professor (Senior)

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Content:

Abstract:

Data Mining technologies have been widely used in the process of medical diagnosis and prognosis, extensively. These data mining techniques have been used to analysed a colossal amount of medical data. The steep increase in the rate of obesity and an unhealthy lifestyle eventually reflects the likelihood and the frequent occurrence of liver-related diseases in the mass. In this project, the patient data sets are analysed for the predictability of the subject to have a liver disease based purely on a widely analysed classification model. Since there are pre-existing processes to analysed the patient data and the classifier data, the more important facet here is to predict the same the conclusive result with a higher rate of accuracy. The data mining techniques are very useful to make medicinal decisions in curing diseases. The healthcare industry collects huge amount of healthcare data which, unfortunately, are not "mined" to discover hidden information for effective decision making. The discovered knowledge can be used by the healthcare administrators to improve the quality of service.

Keywords – SVM, KNN, Random Forest.

INTRODUCTION:

The aim of this project is to somewhat reduce the time delay caused due to the unnecessary back and forth shuttling between the hospital and the pathology lab. Given a dataset containing various attributes of 583 Indian patients, use the features available in the dataset and define a supervised classification algorithm which can identify whether a person is suffering from liver disease or not. The dataset for this problem is the ILPD (Indian Liver Patient Dataset) taken from the UCI Machine Learning Repository Number of instances are 583. It is a multivariate data set. The data set was collected from north east of Andhra Pradesh, India. and our aim will be to train a variety of Supervised Learning algorithms on this data, so that, when a new data point arises, our best performing classifier can be used to categorize the data point as a positive example or negative. And here we used 3 types of algorithm. We are going to compare the results of those three algorithms and going to conclude which algorithm has better performance.

BACKGROUND:

The basic knowledge of database management systems should be known and also data pre -processing should be done properly and effectively. And knowledge of supervised learning should be there. And coming to the implementation part these algorithms are applied on the dataset using python language. This project is implemented in the 5 software “PyCharm”. Basic knowledge of how to operate PyCharm is required. Various machine algorithms are used like SVM, KNN and Random Forest. These algorithms should be properly analysed before applying on the dataset.

Motivation:

There are many disorders of the liver that require clinical care by a physician or other healthcare professional. The study of liver development has significantly contributed to developmental concepts about morphogenesis and differentiation of other organs. Knowledge of the the understanding of human congenital diseases. Significantly, much of understanding of organ development has arisen from analyses of patients with liver deficiencies. In this paper the data classification is based on liver disorder the training data set is developed by collecting data from UCI repository consists of 345 instances with 7 different attributes.

Contributions of the Project:

Registration Number	**Name**	**Work Assigned**
18BCE0435.	Fenil Jain	Code and Analysis of Data.
18BCE2019.	S.S.ShreyasRaj.	Literature Survey and Documentation.
18BCE0160.	D. Jeevan Kishore.	Found Existing Techniques.
18BCE0161.	I.Prashanth.	Data Methods and Introduction.

Literature Survey:

Title	**Method**	**Purpose**	**Advantages**	**Disadvantages**
1.Analysis of Liver Disorder Using Data mining Algorithm.	FT Tree algorithm and K Star algorithm.	Analysis of Liver Disorder Using Data mining Algorithm : In this paper authors analysed the liver dataset which is taken from the UCI repository.	It shows the enhanced performance according to its attribute. Attributes are fully classified by this algorithm and it gives 97.10% of accurate result.	Based on the experimental results the classification accuracy is found to be better using FT Tree algorithm compare to other algorithms.
2.Mining Medical Data to Identify Frequent Diseases using Apriori Algorithm.	Apriori data mining technique.	This research work proposes a association rule based apriori data mining technique that finds the frequency of diseases affecting patients. The study is made on patients from various geographical locations and at various time periods.	Totally 1216 patientrecords affected by 29 different diseases during the year 2012 are analysed. The analysis revealed the fact that 4 different diseases affected the patients frequently at various geographical locations during the year 2012 and to grow.	WEKA data mining tool is employed to identify the frequency of the diseases that are recurring in people living in various geographical locations during different time periods.
3.Estimating the Surveillance of Liver Disorder using Classification Algorithms.	Proposed algorithms are Naive Bayesian Classification, C4.5 Decision Tree.	In this paper authors analysed the liver dataset which is taken from public charitable hospital in Chennai. According to this methodology, it may useful for experts to	In developing countries the average time for a patient to reach into a hospital in any emergency situation is more than one hour.	So medical practitioners are demonstrating awareness of evidence based treatment and therefore this application will give more support to such society for their future work and assessments.

		identify the chances of disease and conscious prescription of further medical examinations and treatment.		
4.A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis.	selected Accuracy, Precision, Sensitivity and Specificity are better for the AP Liver Dataset compared to UCLA liver datasets with all the selected algorithms.	In this study, popular Classification Algorithms were considered for evaluating their classification performance in terms of Accuracy, Precision, Sensitivity and Specificity in classifying liver patient’s dataset.	This can be attributed to more number of useful attributes like Total bilirubin, Direct bilirubin, Indirect bilirubin, Albumin, Gender,Age and Total proteins are available in the AP liver dataset 6 compared to the UCLA dataset.	With the selected dataset, KNN, Back propagation and SVM are giving better results with all the feature set combinations.
5.Liver disease prediction using machine learning.	J48 algorithm.	The performance classification of liver-based diseases is further improved. Time complexity and accuracy can be measured by various machine learning models.	So that we can measures different parameters, owing to the needs of the user. Risk factors can be predicted early by machine learning models.	Future work we can utilize the blend and increasingly Hybrid way to deal with improve execution exactness for liver issue maladies forecast with their reasonable informational.
6.Prediction of liver disease in patients whose liver function tests have been checked in primary care.	ALFI model for prediction of liver disease diagnosis.	This study has developed and externally validated the ALFI model for prediction of liver disease diagnosis in patients with no clinically obvious liver disease having their LFTs taken	From this model, a simple scoring tool was developed to facilitate GP decision-making with regard to retesting or referring their patients.	From this model, a GP decisions regarding referral of patients to secondary care should be based on probability thresholds where benefits outweigh harm.

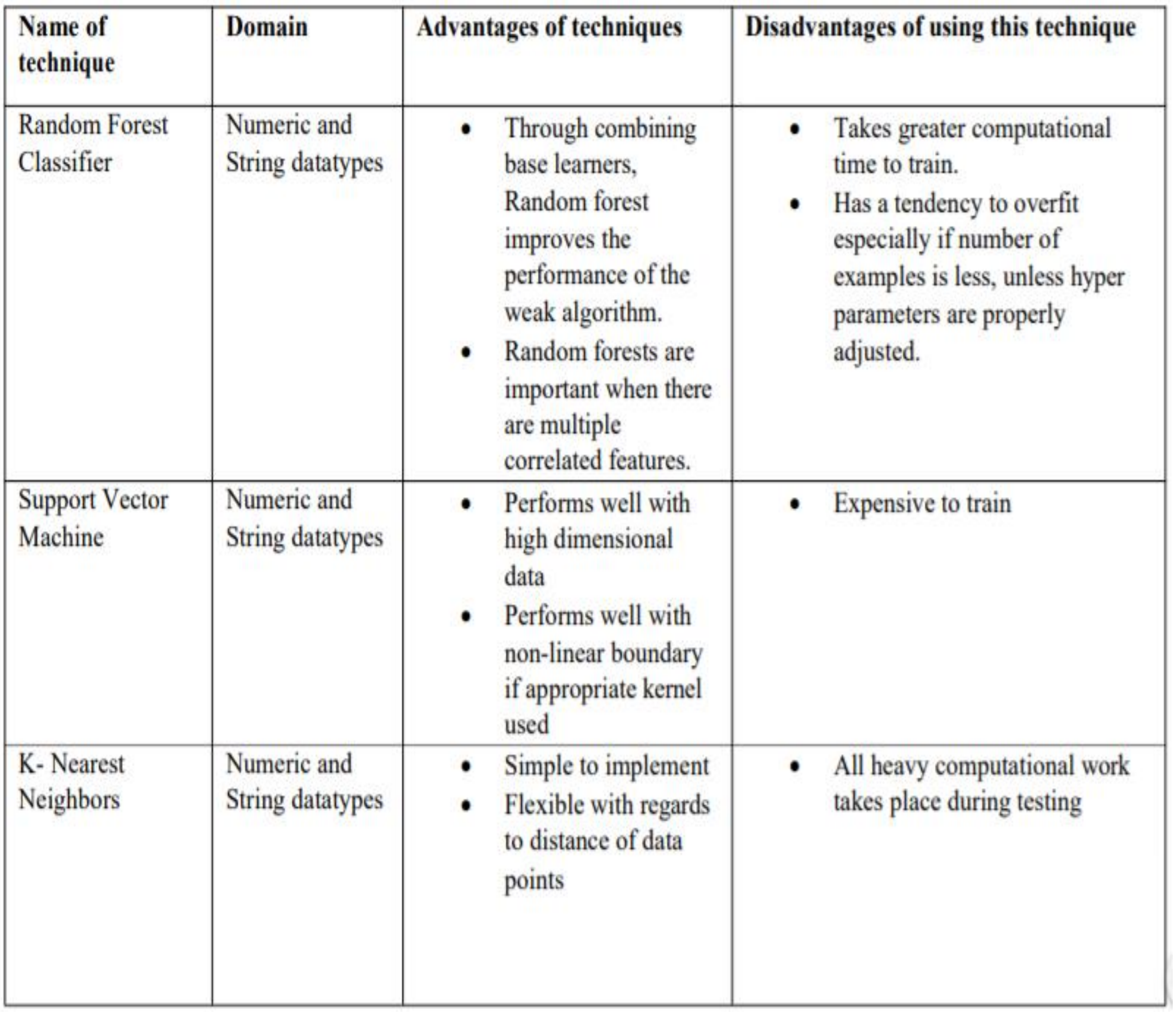
		within primary care.		
7.Data Mining Framework for Fatty Liver Disease Classification in Ultrasound: AHybrid Feature Extraction Paradigm.	CAD based technique.	Several features that were based on the image texture, Higher Order Spectra, and Wavelet Packet Decomposition were extracted from the speckle images of the liver ultrasound images.	Using only three features, the KNN, and PNN classifiers presented a high accuracy of 93.3%. They recorded almost similar values for both sensitivity and specificity (94.4% and 91.7%, respectively). All these classifiers have given significant performance measures using a small dataset.	We believe that with the inclusion of more representative features, and study of other classifiers using a larger dataset, it should be possible to improve the current accuracy of the technique.
8.Biochemical Markers of Fibrosis for Chronic Liver Disease: Data mining-based Approach.	Decision trees provide explicit rules to relate the range of values of the biomarkers with fibrosis scores.	Serum markers are of great value not only in patients at risk for LB, but also as a 7 part of the assessment of patients with chronic liver disease avoiding the invasive methods.	One of the most widely used non-invasive markers to stage liver fibrosis is the FT which involves the measurement of a set of surrogate markers that, in combination, have a high predictive value for the diagnosis of significant fibrosis.	. The results indicated that Decision trees model were useful and effective to predict liver fibrosis stage at least similar to biopsy and provide a qualitative and quantitative overview for physician to find the relations between FT and LB.
9.A Survey and Compare the Performance of IBM SPSS Modeller .	Two data mining software, IBM SPSS Modeler and Rapid Miner.	For this purpose, the data for Indian patients which were 583 records and were available in University of California Irvine	After implementation and running of the algorithms, the C5.0 algorithm showed a better performance in	The results of this study showed that in IBM SPSS Modeler software are better than Rapid Miner software. In Rapid Miner software

		(UCI) archive were used. Among all the algorithms, the C5.0 algorithm had the best performance with having 87.91% of Accuracy.	comparison to C4.5 algorithm in IBM SPSS Modeler software. CHAID, QUEST, Random Forest algorithms had almost a similar performance in both software.	was that all the medial and output layers were developed in details but, these details were not available in the obtained diagram by IBM SPSS Modeler software.
10.Diagnosis of Liver Tumor from CT Images using Digital Image Processing.	Tumour detection using CT image has been done using the digital image processing.	We have developed an automated method for the detection of tumours in liver CT images using Mask, colour map, colour segmentation, modified K-means clustering and image processing techniques.	The segmentation accuracy is obtained using the modified KMeans clustering. Our system has been successfully tested on a large number of tumour images, liver tumour implemented for the discrimination.	Our system has been successfully tested on a large number of tumour images, liver tumour implemented for the discrimination of the normal and pathological tissues. The liver regions related to a tumour can be exactly separated from the liver image.

<u>**Conclusion for Literature Survey:**</u>

We are using classification, one of the major data mining models, which is used to predict previously unknown class of objects. Unlike other diseases, liver disorder prediction from common symptoms is typically difficult job for medical practitioners. Most of the features or symptoms are seen in many other fever related diseases and so it is not free from false assumptions. In most cases, the opportunity of liver disease will not identified because of the domination of other diseases.

Some of the Techniques:



Name of technique	Domain	Advantages of techniques	Disadvantages of using this technique
Random Forest Classifier	Numeric and String datatypes	• Through combining base learners, Random forest improves the performance of the weak algorithm. • Random forests are important when there are multiple correlated features.	• Takes greater computational time to train. • Has a tendency to overfit especially if number of examples is less, unless hyper parameters are properly adjusted.
Support Vector Machine	Numeric and String datatypes	• Performs well with high dimensional data • Performs well with non-linear boundary if appropriate kernel used	• Expensive to train
K- Nearest Neighbors	Numeric and String datatypes	• Simple to implement • Flexible with regards to distance of data points	• All heavy computational work takes place during testing

DATASET DESCRIPTION & SAMPLE DATA:

The ILPD dataset contains ten features as listed below:

1. Age
2. Gender
3. Total bilirubin
4. Direct bilirubin

5. Total proteins

6. Albumin

7. A/G ratio

8. SGPT

9. SGOT

10. Alkphos.

All features, except Gender are real valued integers. The last column, Disease, is the label (with '1' representing presence of disease and '2' representing absence of disease). Total number of data points is 583, with 416 liver patient records and 167 non liver patient records.

S.NO	AGE	GENDER	TOTAL BILIRUIN	DIRECT BILIRUIN	TOTAL PROTIENS	ALBUMIN	A/G RATIO	SGPT	SGOT	CONDITI ON
1	32	Male	12.1	6	515	48	92	6.6	2.4	Person is Normal
2	32	Male	25	13.7	560	41	88	7.9	2.5	Person is Normal
3	32	Male	15	8.2	289	58	80	5.3	2.2	Person is Normal
4	32	Male	12.7	8.4	190	28	47	5.4	2.6	Person is Normal
5	60	Male	0.5	0.1	500	20	34	5.9	1.6	Person has Liver Problem
6	40	Male	0.6	0.1	98	35	31	6	3.2	Person is Normal
7	52	Male	0.8	0.2	245	48	49	6.4	3.2	Person is Normal
8	31	Male	1.3	0.5	184	29	32	6.8	3.4	Person is Normal

PROPOSED ALGORITHM WITH FLOWCHART:

1. Random Forest Classifier:

It comes under the category of ensemble methods. It employs ‘bagging’ and ‘boosting’ methods to draw a random subset from the data, and train a Decision Tree on that (hence, the name Random Forest). In this case, we don’t know the relative importance of each feature while deciding the output, so a

Random Forest can be successful as it will ensure training on different randomized subsets. Hyperparameters to be manipulated:

- n_estimators (number of trees in a forest)
- max_depth (maximum depth of one single tree)
- max_features (decides how many features are to be used)
- oob_score (decides whether to include out-of-bag or prediction error)

2. Support Vector Machine:

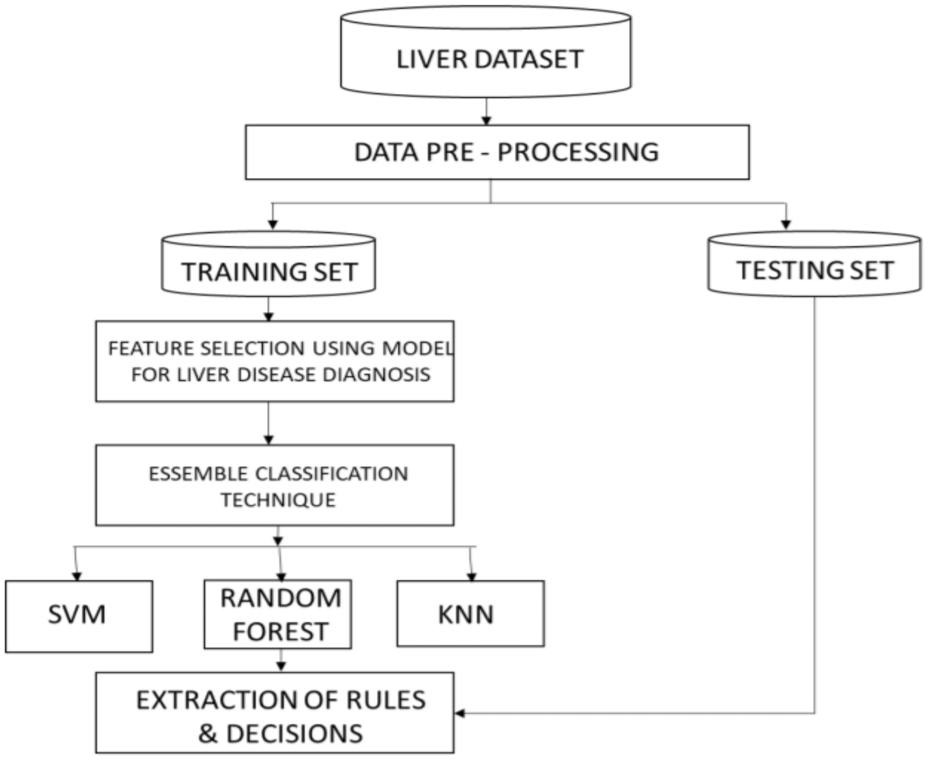
SVM aims to find an optimal hyperplane that separates the data into different classes, using a method called as kernel to project data points belonging to a particular class into different dimensions, so that a hyperplane can easily pass through and maintain the largest possible distance between itself and these data points.

Hyperparameters to be manipulated:

- kernel (type of kernel used like ‘linear’, ‘rbf’ etc for separating data points)
- C (the penalty assigned to the error term)

3.Logistic Regression:

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets. Logistic regression is mainly used for classification problems. For example, classifying whether an email is a spam or not spam. A simplest logistic regression is one that is implemented using python NumPy and pandas without learn. Logistic regression is a classification algorithm in the realm of machine learning. logistic regression outputs a probability value that can be mapped to two or more classes. It used logit function and optimizer.


LIVER DATASET
DATA PRE - PROCESSING
TRAINING SET
TESTING SET
FEATURE SELECTION USING MODEL FOR LIVER DISEASE DIAGNOSIS
ESSEMBLE CLASSIFICATION TECHNIQUE
SVM
RANDOM FOREST
KNN
EXTRACTION OF RULES & DECISIONS

Appendix:

1. Import:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import copy
from sklearn.metrics import accuracy_score
```

2. Dataset:

```
df = pd.read_csv('Indian Liver Patient Dataset (ILPD).csv')
df.shape
```

3. Data Processing:

3.1 Removing Duplicates:

```
df_duplicate = df[df.duplicated(keep = False)]
```

```
df_duplicate.head(10)
print(df.isnull().sum())
```

3.2 Dropping Null Values:

```
copyDF = copy.deepcopy(df)
copyDF = copyDF.dropna(axis=0)
npdf[0]
```

3.3 Label Encoding Categorical Variables:

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(copyDF['Gender'])
#print(type(le.transform(copyDF['Gender'])))
le.fit(copyDF['Result'])
#print(le.transform(copyDF['Result']))
le.fit(copyDF['Gender'])
copyDF['Gender'] = le.transform(copyDF['Gender'])
le.fit(copyDF['Result'])
copyDF['Result'] = le.transform(copyDF['Result'])
copyDF
```

3.4 Test Train Spirit:

```
X = copyDF[['Age', 'Gender', 'TB', 'DB', 'AAP', 'SAA', 'TP', 'ALB', 'A/G', 'S']]
y = copyDF[['Result']]
from sklearn.model_selection import train_test_split
#X_train, x_test, y_train, y_test = train_test_split(df2, test_size = 0.2, random_
state = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
tate = 1
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

4.Isolation Forests:

```
from sklearn.ensemble import IsolationForest
```

```
clf = IsolationForest(max_samples=100, contamination=0.0
df_drop = df.drop(axis=1, index=1)
df_drop.head()
df_drop_na = df_drop.dropna()
#clf.fit(np.vstack((df_drop_na.iloc[:, 8].values, df_drop_na.iloc[:, 9].values)).T)
clf.fit(X_train, y_train)
#clf_pred = clf.predict(np.vstack((df_drop_na.iloc[:, 8].values, df_drop_na.iloc[:
, 9].values)).T)
clf_pred = clf.predict(X_test)
clf_pred
```

5.Decision Trees:

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_pred
accuracy_score(y_test, y_pred)
```

6.Logistic Regression:

```
rom sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=1, max_iter = 10000).fit(X_train, y_train
.to_numpy().reshape(463, ))
y_pred = clf.predict(X_test)
y_pred
accuracy_score(y_test, y_pred)
y_train.to_numpy().reshape(463, 1)
```

7.Random Forest:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X_train, y_train.to_numpy().reshape(463, ))
y_pred = clf.predict(X_test)
accuracy_score(y_test, y_pred)
```

8.SVM:

```
from sklearn import svm
model = svm.SVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test) accuracy_score(y_test, y_pred)
```

9.Visualizations:

1.

```
plt.scatter(df.iloc[:, 8].values, df.iloc[:, 9].values, color = ['blue'])
```

2.

```
plt.hist(df.iloc[:, [3]])
plt.show()
```

3.

```
plt.figure(figsize=(33, 33))
sns.heatmap(copyDF, annot=True, fmt=".3f", linewidths=.5, square = True)
```

4.

```
copyDF
```

5.

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
age = copyDF['Age']
result = copyDF['Result']
ax.bar(age,result)
plt.show()
```

6.

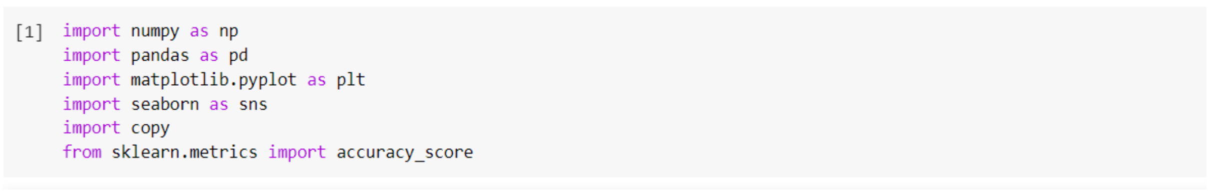
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = copyDF['Gender']
result = copyDF['Result']
ax.bar(gender,result)
plt.show()
```

7.

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
gender = copyDF['Gender']
result = copyDF['Result']
ax.bar(gender,result) plt.show()
```

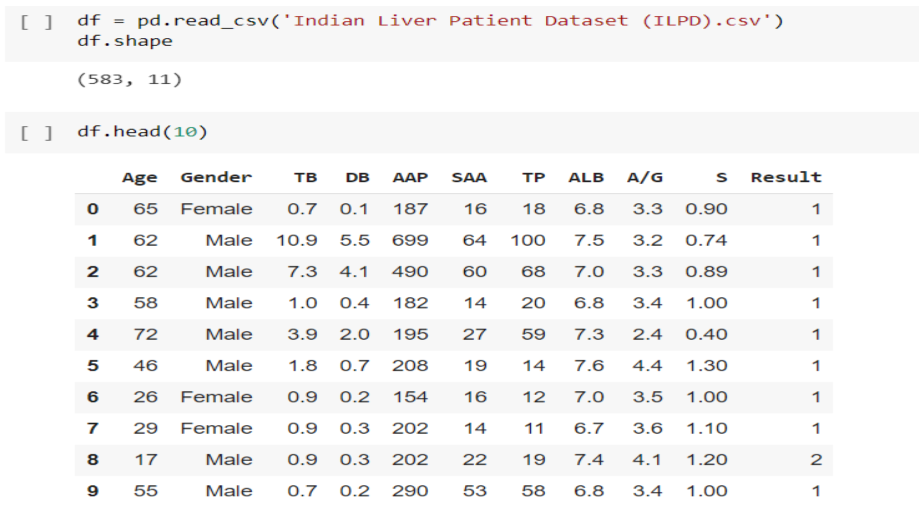
Result and Analysis:

1.Import:



```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import copy
     from sklearn.metrics import accuracy_score
```

2.Dataset:



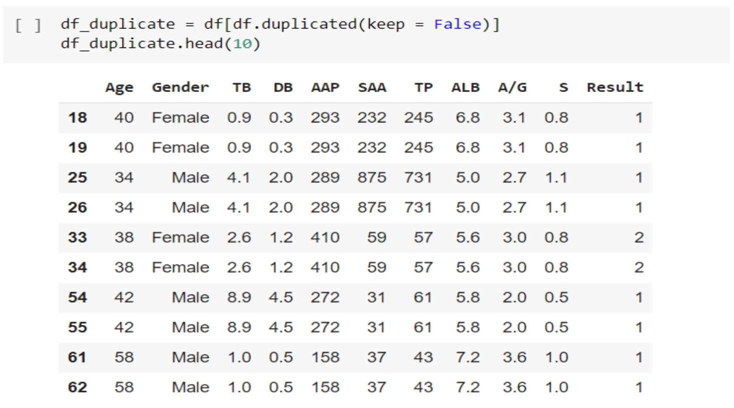
```
[ ]  df = pd.read_csv('Indian Liver Patient Dataset (ILPD).csv')
     df.shape

     (583, 11)
```

```
[ ]  df.head(10)
```

	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S	Result
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
5	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1
6	26	Female	0.9	0.2	154	16	12	7.0	3.5	1.00	1
7	29	Female	0.9	0.3	202	14	11	6.7	3.6	1.10	1
8	17	Male	0.9	0.3	202	22	19	7.4	4.1	1.20	2
9	55	Male	0.7	0.2	290	53	58	6.8	3.4	1.00	1

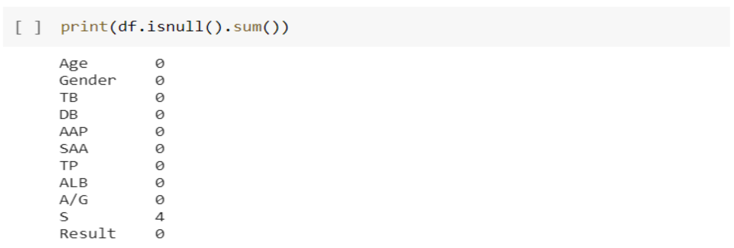
3.Data Processing:

3.1 Removing Duplicates:



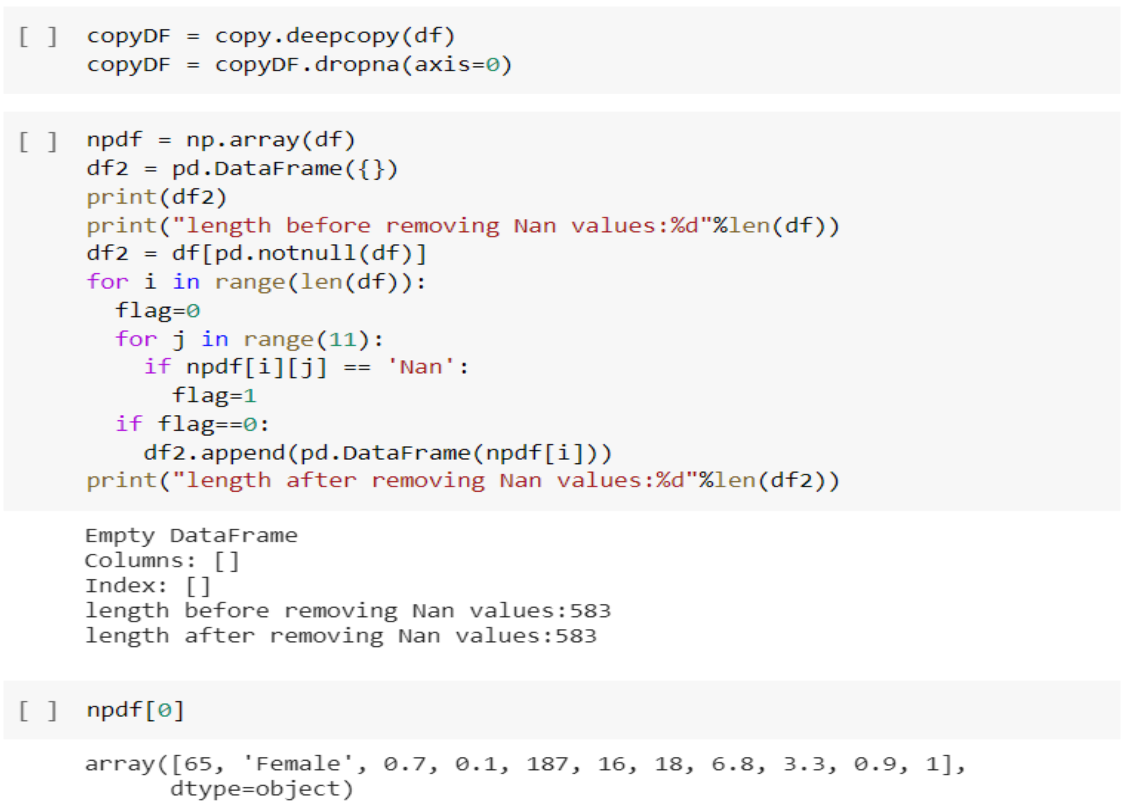
```
[ ]  df_duplicate = df[df.duplicated(keep = False)]
     df_duplicate.head(10)
```

	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S	Result
18	40	Female	0.9	0.3	293	232	245	6.8	3.1	0.8	1
19	40	Female	0.9	0.3	293	232	245	6.8	3.1	0.8	1
25	34	Male	4.1	2.0	289	875	731	5.0	2.7	1.1	1
26	34	Male	4.1	2.0	289	875	731	5.0	2.7	1.1	1
33	38	Female	2.6	1.2	410	59	57	5.6	3.0	0.8	2
34	38	Female	2.6	1.2	410	59	57	5.6	3.0	0.8	2
54	42	Male	8.9	4.5	272	31	61	5.8	2.0	0.5	1
55	42	Male	8.9	4.5	272	31	61	5.8	2.0	0.5	1
61	58	Male	1.0	0.5	158	37	43	7.2	3.6	1.0	1
62	58	Male	1.0	0.5	158	37	43	7.2	3.6	1.0	1



```
[ ]  print(df.isnull().sum())

     Age       0
     Gender    0
     TB        0
     DB        0
     AAP       0
     SAA       0
     TP        0
     ALB       0
     A/G       0
     S         4
     Result    0
```

3.2 Dropping Null Values:



```
[ ]  copyDF = copy.deepcopy(df)
     copyDF = copyDF.dropna(axis=0)
```

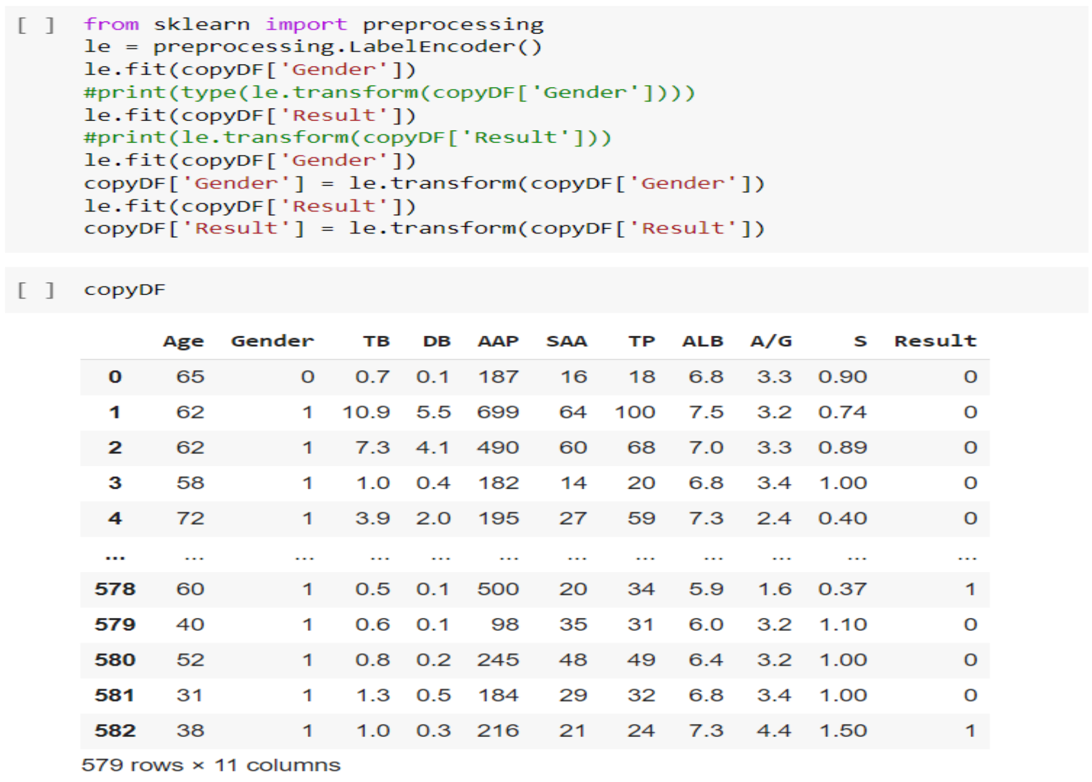
```
[ ]  npdf = np.array(df)
     df2 = pd.DataFrame({})
     print(df2)
     print("length before removing Nan values:%d"%len(df))
     df2 = df[pd.notnull(df)]
     for i in range(len(df)):
       flag=0
       for j in range(11):
         if npdf[i][j] == 'Nan':
           flag=1
       if flag==0:
         df2.append(pd.DataFrame(npdf[i]))
     print("length after removing Nan values:%d"%len(df2))
```

```
     Empty DataFrame
     Columns: []
     Index: []
     length before removing Nan values:583
     length after removing Nan values:583
```

```
[ ]  npdf[0]
```

```
     array([65, 'Female', 0.7, 0.1, 187, 16, 18, 6.8, 3.3, 0.9, 1],
           dtype=object)
```

3.3 Label Encoding Categorical Variables:



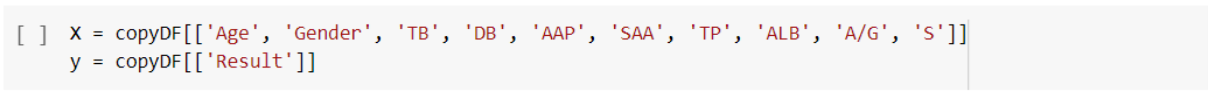
```
[ ]  from sklearn import preprocessing
     le = preprocessing.LabelEncoder()
     le.fit(copyDF['Gender'])
     #print(type(le.transform(copyDF['Gender'])))
     le.fit(copyDF['Result'])
     #print(le.transform(copyDF['Result']))
     le.fit(copyDF['Gender'])
     copyDF['Gender'] = le.transform(copyDF['Gender'])
     le.fit(copyDF['Result'])
     copyDF['Result'] = le.transform(copyDF['Result'])
```

```
[ ]  copyDF
```

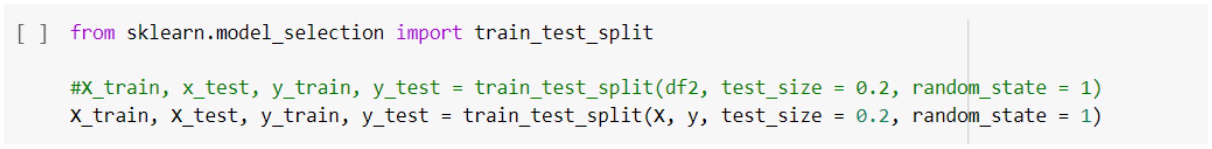
	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S	Result
0	65	0	0.7	0.1	187	16	18	6.8	3.3	0.90	0
1	62	1	10.9	5.5	699	64	100	7.5	3.2	0.74	0
2	62	1	7.3	4.1	490	60	68	7.0	3.3	0.89	0
3	58	1	1.0	0.4	182	14	20	6.8	3.4	1.00	0
4	72	1	3.9	2.0	195	27	59	7.3	2.4	0.40	0
...	...	...	...	...	...	...	...	...	...	...	...
578	60	1	0.5	0.1	500	20	34	5.9	1.6	0.37	1
579	40	1	0.6	0.1	98	35	31	6.0	3.2	1.10	0
580	52	1	0.8	0.2	245	48	49	6.4	3.2	1.00	0
581	31	1	1.3	0.5	184	29	32	6.8	3.4	1.00	0
582	38	1	1.0	0.3	216	21	24	7.3	4.4	1.50	1

579 rows × 11 columns

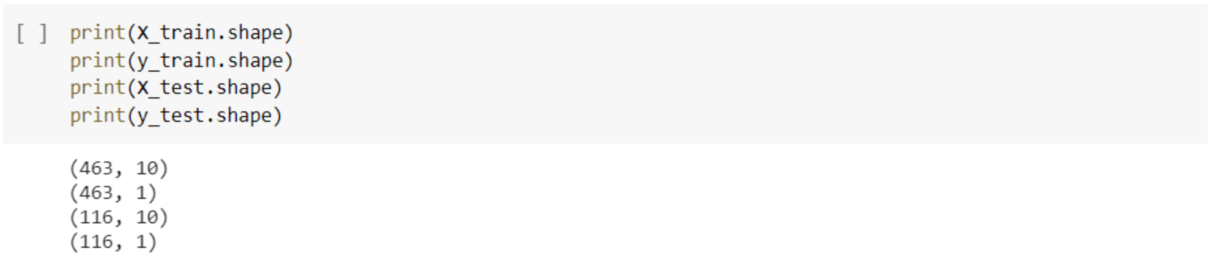
3.4 Test Train Spirit:



```
[ ]  X = copyDF[['Age', 'Gender', 'TB', 'DB', 'AAP', 'SAA', 'TP', 'ALB', 'A/G', 'S']]
     y = copyDF[['Result']]
```

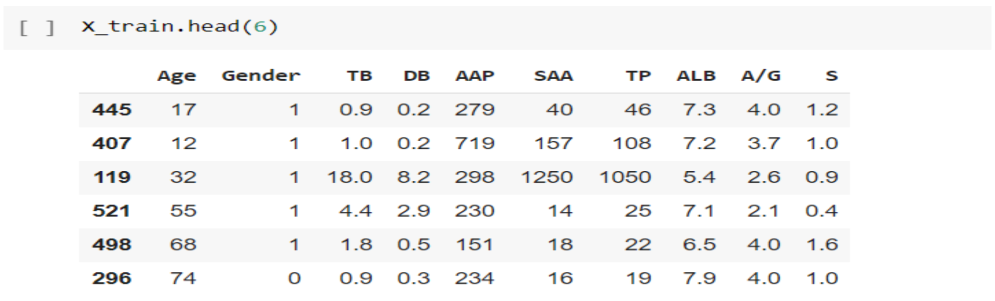


```
[ ]  from sklearn.model_selection import train_test_split

     #X_train, x_test, y_train, y_test = train_test_split(df2, test_size = 0.2, random_state = 1)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```



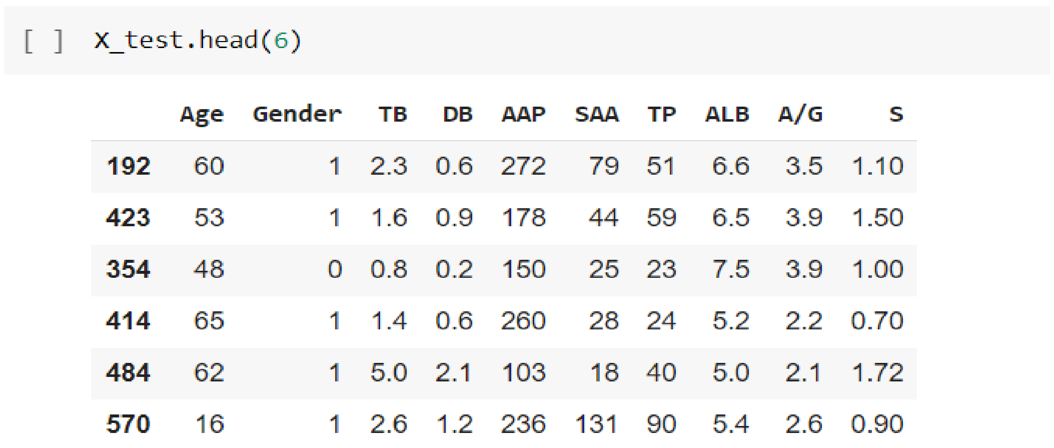
```
[ ]  print(X_train.shape)
     print(y_train.shape)
     print(X_test.shape)
     print(y_test.shape)
```

```
     (463, 10)
     (463, 1)
     (116, 10)
     (116, 1)
```



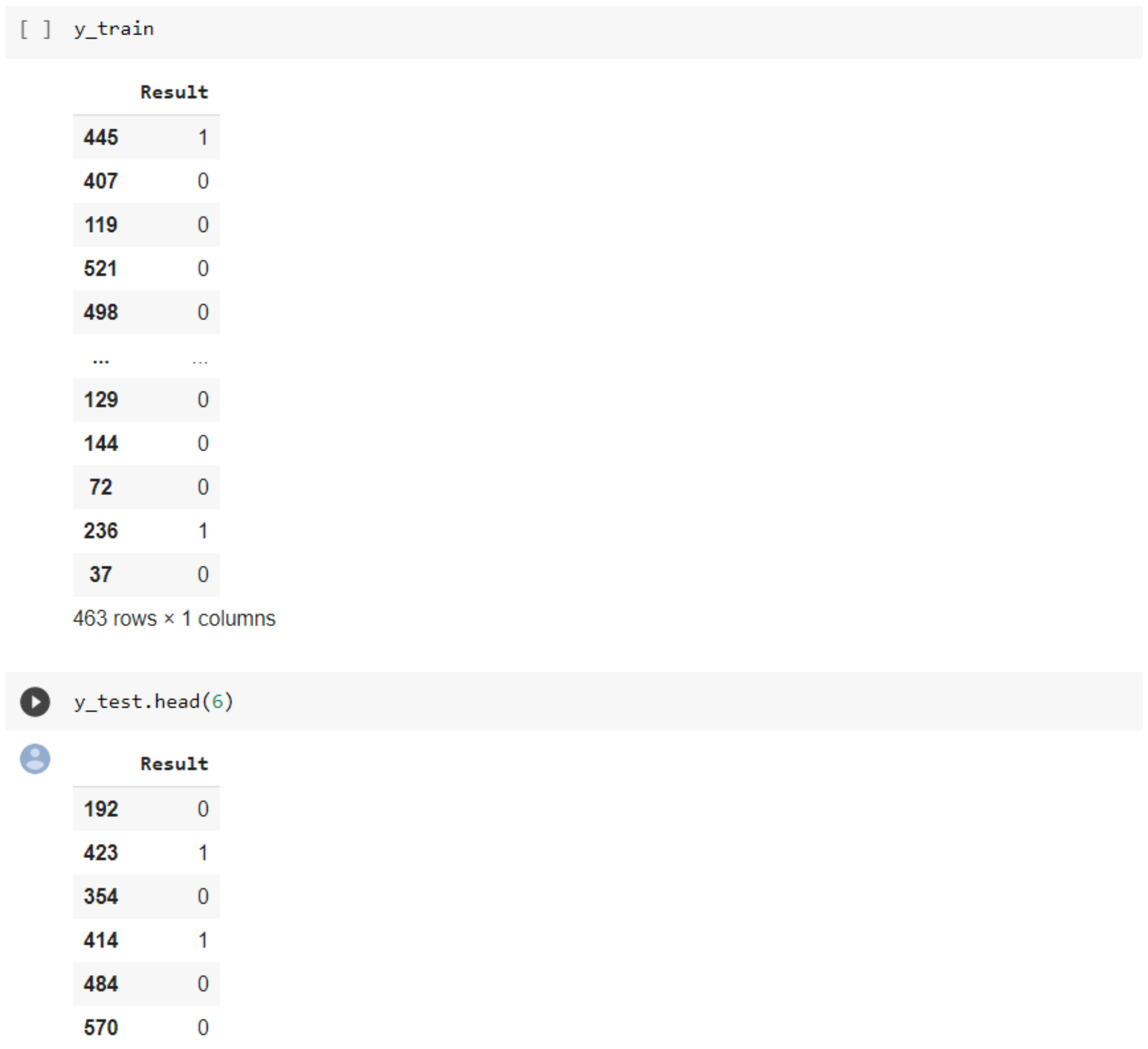
```
[ ]  X_train.head(6)
```

	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S
445	17	1	0.9	0.2	279	40	46	7.3	4.0	1.2
407	12	1	1.0	0.2	719	157	108	7.2	3.7	1.0
119	32	1	18.0	8.2	298	1250	1050	5.4	2.6	0.9
521	55	1	4.4	2.9	230	14	25	7.1	2.1	0.4
498	68	1	1.8	0.5	151	18	22	6.5	4.0	1.6
296	74	0	0.9	0.3	234	16	19	7.9	4.0	1.0



```
[ ]  X_test.head(6)
```

	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S
192	60	1	2.3	0.6	272	79	51	6.6	3.5	1.10
423	53	1	1.6	0.9	178	44	59	6.5	3.9	1.50
354	48	0	0.8	0.2	150	25	23	7.5	3.9	1.00
414	65	1	1.4	0.6	260	28	24	5.2	2.2	0.70
484	62	1	5.0	2.1	103	18	40	5.0	2.1	1.72
570	16	1	2.6	1.2	236	131	90	5.4	2.6	0.90



```
[ ] y_train
```

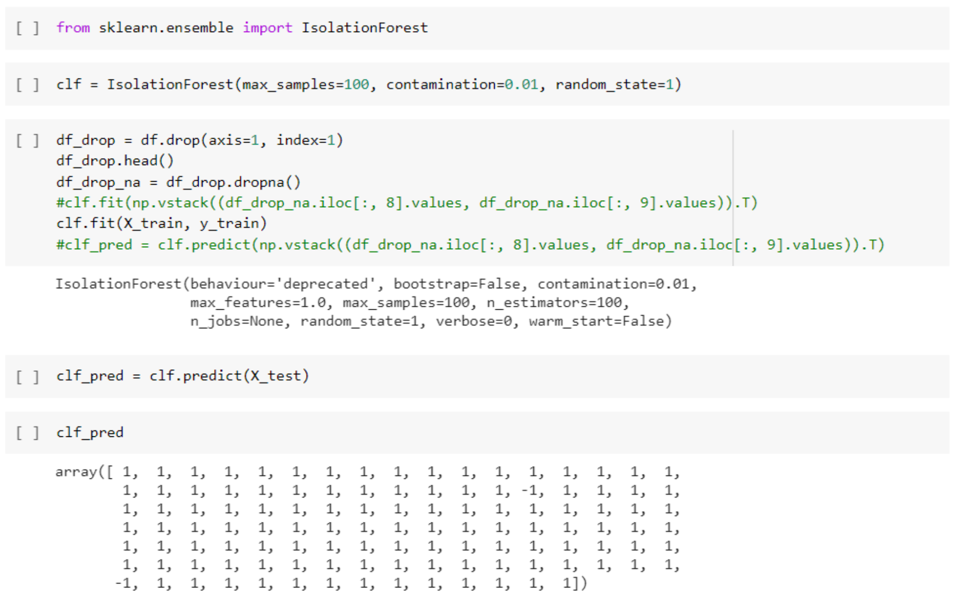
	Result
445	1
407	0
119	0
521	0
498	0
...	...
129	0
144	0
72	0
236	1
37	0

463 rows × 1 columns

```
y_test.head(6)
```

	Result
192	0
423	1
354	0
414	1
484	0
570	0

4.Isolation Forests:



```
[ ] from sklearn.ensemble import IsolationForest
```

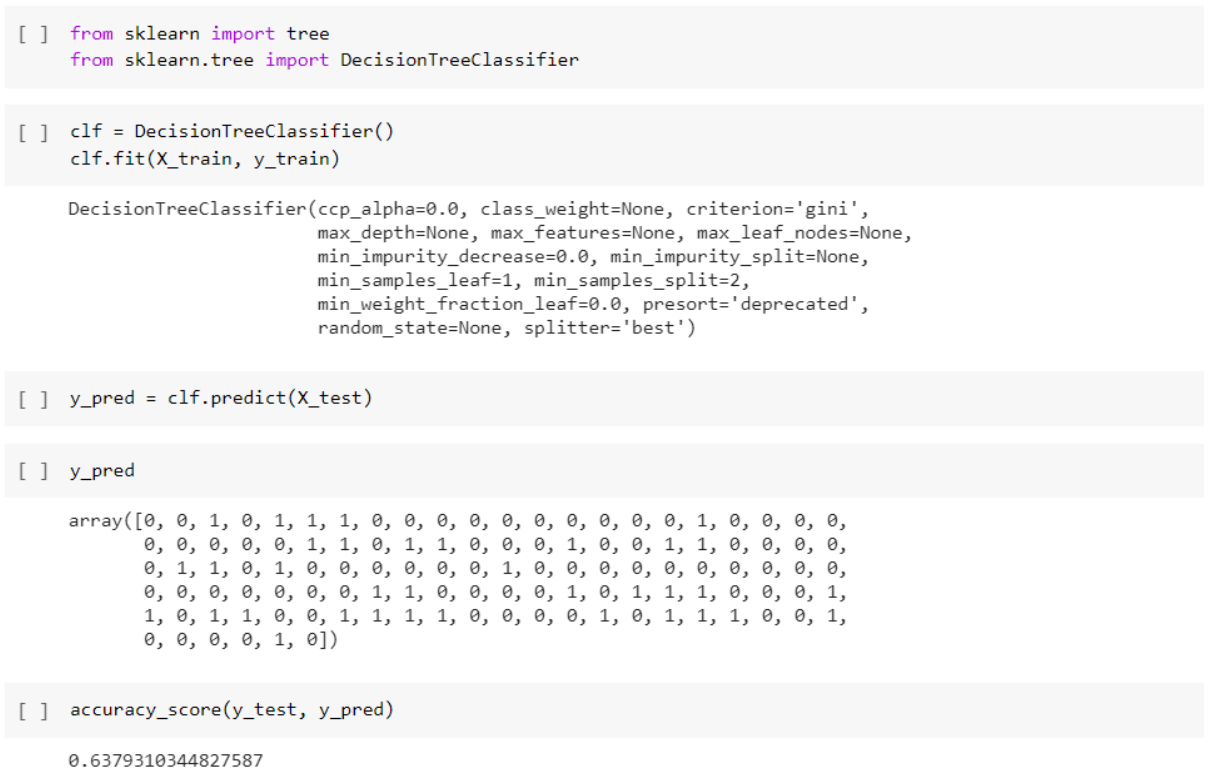
```
[ ] clf = IsolationForest(max_samples=100, contamination=0.01, random_state=1)
```

```
[ ] df_drop = df.drop(axis=1, index=1)
    df_drop.head()
    df_drop_na = df_drop.dropna()
    #clf.fit(np.vstack((df_drop_na.iloc[:, 8].values, df_drop_na.iloc[:, 9].values)).T)
    clf.fit(X_train, y_train)
    #clf_pred = clf.predict(np.vstack((df_drop_na.iloc[:, 8].values, df_drop_na.iloc[:, 9].values)).T)

    IsolationForest(behaviour='deprecated', bootstrap=False, contamination=0.01,
                    max_features=1.0, max_samples=100, n_estimators=100,
                    n_jobs=None, random_state=1, verbose=0, warm_start=False)
```

```
[ ] clf_pred = clf.predict(X_test)
```

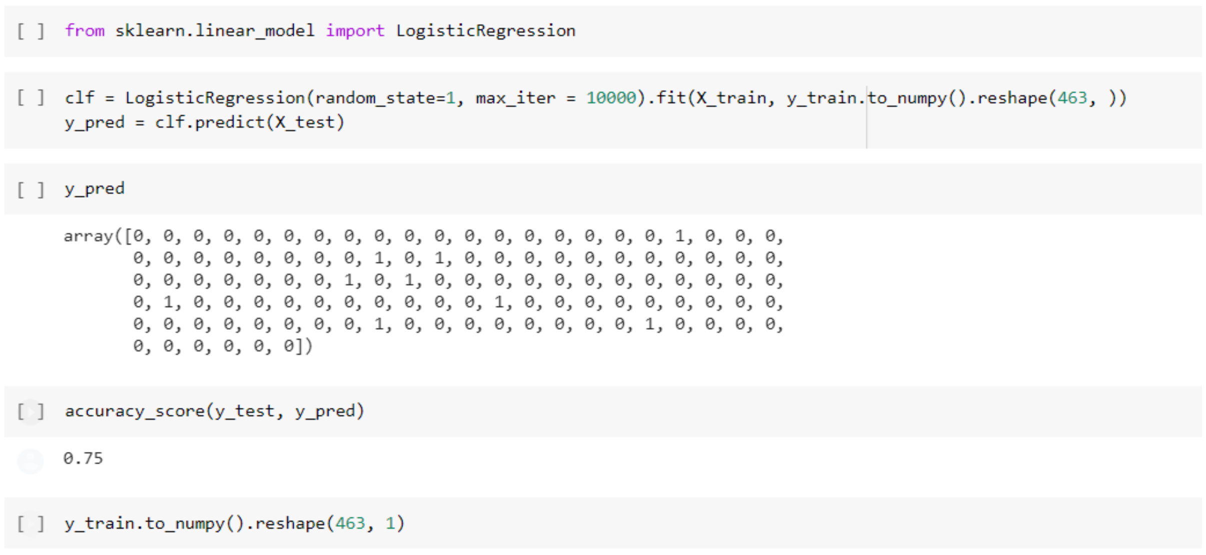
```
[ ] clf_pred

    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,  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])
```

5.Decision Trees:



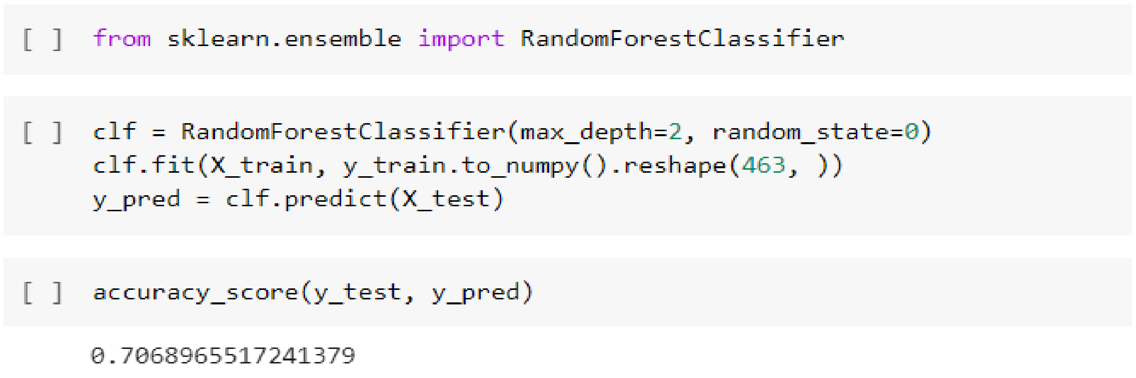
```
[ ]  from sklearn import tree
     from sklearn.tree import DecisionTreeClassifier

[ ]  clf = DecisionTreeClassifier()
     clf.fit(X_train, y_train)

     DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                            max_depth=None, max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=None, splitter='best')

[ ]  y_pred = clf.predict(X_test)

[ ]  y_pred

     array([0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
            0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
            1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
            0, 0, 0, 0, 1, 0])

[ ]  accuracy_score(y_test, y_pred)

     0.6379310344827587
```

6.Logistic Regression:



```
[ ]  from sklearn.linear_model import LogisticRegression

[ ]  clf = LogisticRegression(random_state=1, max_iter = 10000).fit(X_train, y_train.to_numpy().reshape(463, ))
     y_pred = clf.predict(X_test)

[ ]  y_pred

     array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0])

[ ]  accuracy_score(y_test, y_pred)

     0.75

[ ]  y_train.to_numpy().reshape(463, 1)
```

7.Random Forest:

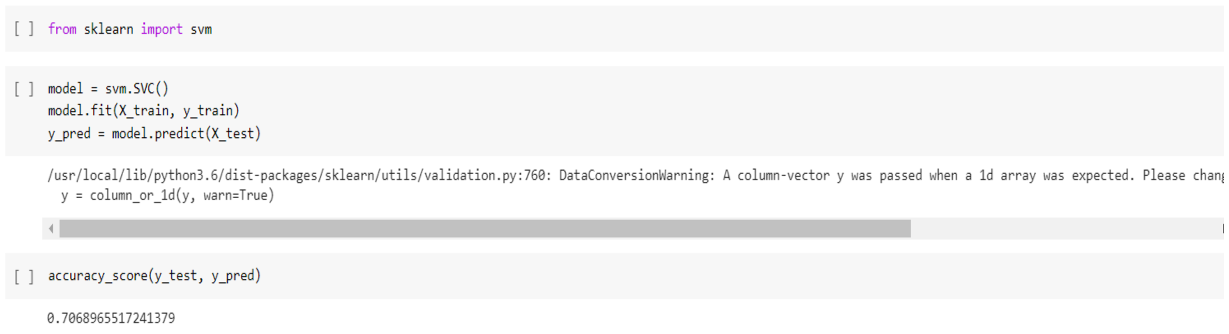


```
[ ]  from sklearn.ensemble import RandomForestClassifier
```

```
[ ]  clf = RandomForestClassifier(max_depth=2, random_state=0)
     clf.fit(X_train, y_train.to_numpy().reshape(463, ))
     y_pred = clf.predict(X_test)
```

```
[ ]  accuracy_score(y_test, y_pred)

     0.7068965517241379
```

8.SVM:



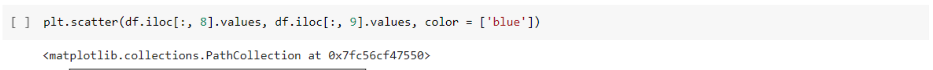
```
[ ]  from sklearn import svm
```

```
[ ]  model = svm.SVC()
     model.fit(X_train, y_train)
     y_pred = model.predict(X_test)

     /usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please chang
       y = column_or_1d(y, warn=True)
```

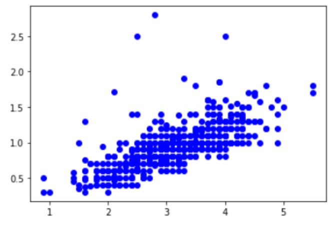
```
[ ]  accuracy_score(y_test, y_pred)

     0.7068965517241379
```

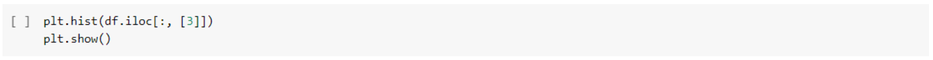
9.Visualizations:

1,2:

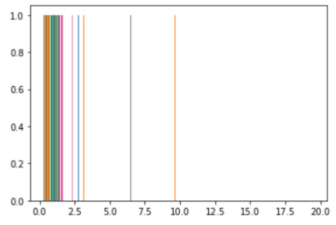


```
[ ]  plt.scatter(df.iloc[:, 8].values, df.iloc[:, 9].values, color = ['blue'])

     <matplotlib.collections.PathCollection at 0x7fc56cf47550>
```





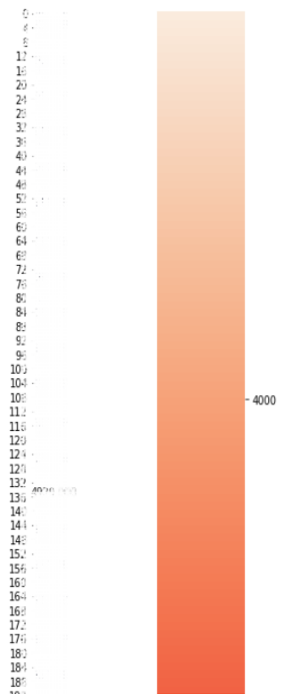
```
[ ]  plt.hist(df.iloc[:, [3]])
     plt.show()
```



3:

```
[ ] plt.figure(figsize=(33, 33))
    sns.heatmap(copyDF, annot=True, fmt=".3f", linewidths=.5, square = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdee906b940>
```



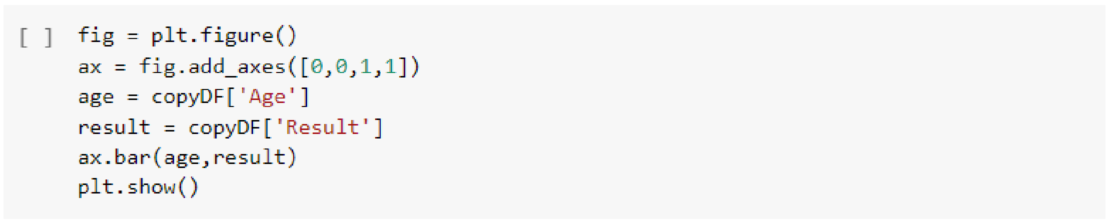
4:

```
[ ] copyDF
```

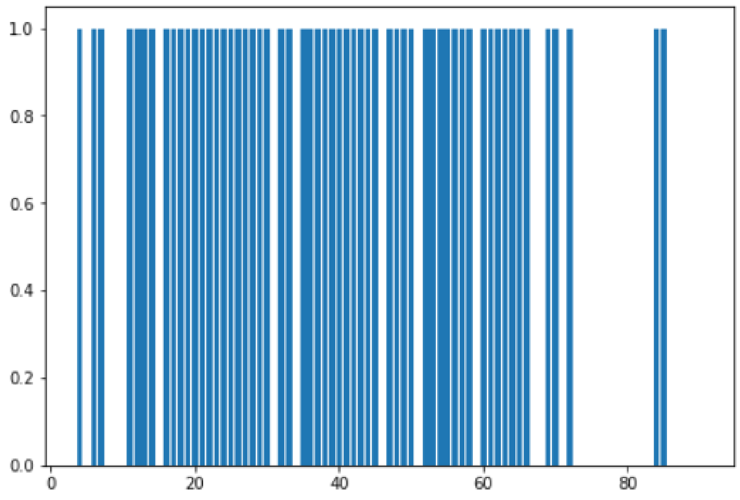
	Age	Gender	TB	DB	AAP	SAA	TP	ALB	A/G	S	Result
0	65	0	0.7	0.1	187	16	18	6.8	3.3	0.90	0
1	62	1	10.9	5.5	699	64	100	7.5	3.2	0.74	0
2	62	1	7.3	4.1	490	60	68	7.0	3.3	0.89	0
3	58	1	1.0	0.4	182	14	20	6.8	3.4	1.00	0
4	72	1	3.9	2.0	195	27	59	7.3	2.4	0.40	0
...	...	...	...	...	...	...	...	...	...	...	...
578	60	1	0.5	0.1	500	20	34	5.9	1.6	0.37	1
579	40	1	0.6	0.1	98	35	31	6.0	3.2	1.10	0
580	52	1	0.8	0.2	245	48	49	6.4	3.2	1.00	0
581	31	1	1.3	0.5	184	29	32	6.8	3.4	1.00	0
582	38	1	1.0	0.3	216	21	24	7.3	4.4	1.50	1

579 rows × 11 columns

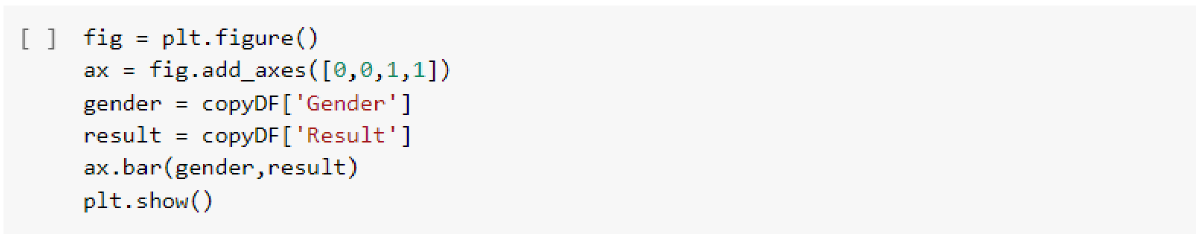
5:



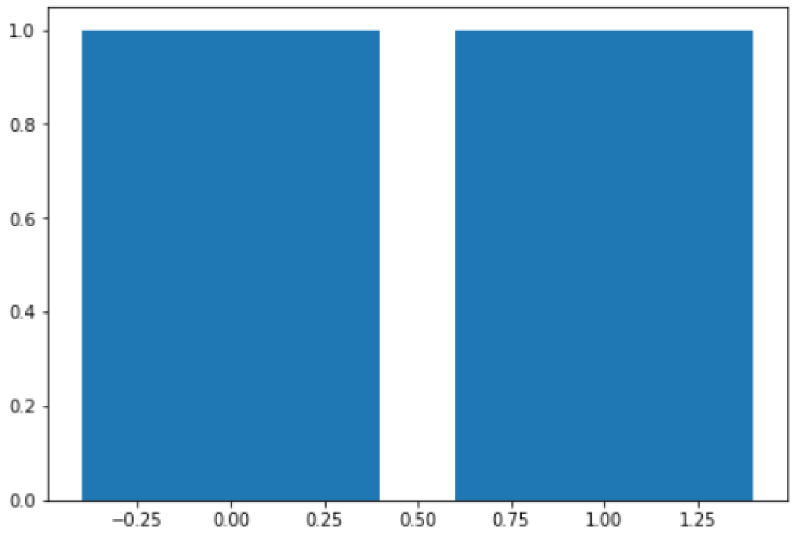
```
[ ]  fig = plt.figure()
     ax = fig.add_axes([0,0,1,1])
     age = copyDF['Age']
     result = copyDF['Result']
     ax.bar(age,result)
     plt.show()
```



6:

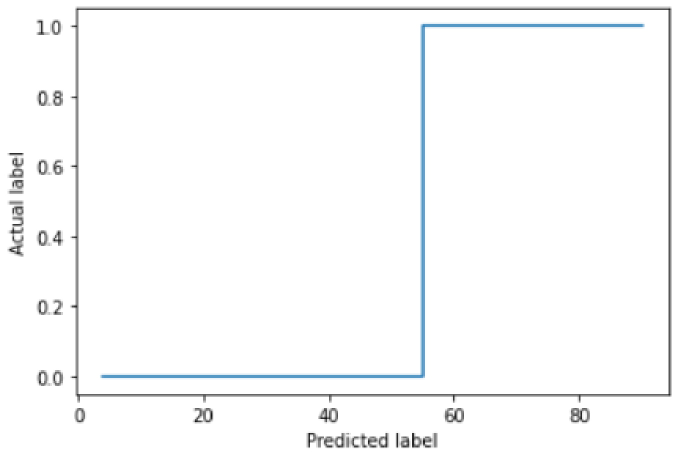


```
[ ]  fig = plt.figure()
     ax = fig.add_axes([0,0,1,1])
     gender = copyDF['Gender']
     result = copyDF['Result']
     ax.bar(gender,result)
     plt.show()
```



7:

```
[ ]  result = list(copyDF['Result'])
     age = list(copyDF['Age'])
     result.sort()
     age.sort()
     plt.plot(age, result)
     plt.ylabel('Age')
     plt.xlabel('Result')
     plt.show()
```



ROC curves that show the performance of the respective algorithms before and after applying Grid Search CV have been displayed in the preceding sections. Initially, the dataset was explored and made ready to be fed into the classifiers. This was achieved by removing some rows containing null values, transforming some columns which were showing skewness and using appropriate methods (one-hot encoding) to convert the labels so that they can be useful for classification purposes. Performance metrics on which the models would be evaluated were decided. The dataset was then split into a training and testing set. Firstly, a naive predictor and a benchmark model ('Logistic Regression') were run on the dataset to determine the benchmark value of F-beta scores. Then, three classifiers were trained on the dataset and tested on the testing set. Finally, all three models were refined to a certain extent by choosing different values of their hyperparameters using Grid Search CV. The models were compared on their F- beta as well as their ROC scores. The results turned out to be ambiguous, with SVM and Random Forest running neck-to neck for best performance. The performance with respect to F-beta score was also marginally better than Logistic Regression for SVM, and slightly worse for Random Forest.

Comparitive Study/ Results and Discussion:

So, we see that 'Total Proteins' is the most important feature, although not by a significant margin. Using only 5 most important features leads to a reduction in accuracy by around 5% and F-score by around 3%. Maybe in future if number of data pointsis very large and we are willing to save some training time at the cost of accuracy, we can use this method. As the results show, it is difficult to arrive at a clear-cut answer for the best performing model among SVM and Random Forest. While SVM shows better result in terms of F- beta score (0.94), its area under the ROC curve is surprisingly low. Random Forest shows a decent enough F-beta score (0.87) and its area under ROC curve is 0.6.

Conclusion and Future -work.

It was decided to select Random Forest as the final classifier, even though its F-score (0.8686) was less than that of SVM. This is because its area under the ROC curve (0.60 and 0.65 for two configurations) is greater than that of SVM (0.50). we are considering ROC score as an important metric, and Random Forest trumps in that area. KNN was the worst performer with a F-score of 0.796 after refinement. Having said that, only one model (SVM) managed to surpass our benchmark classifier of Logistic Regression, which had an F-score of 0.91, with Random Forest coming close. this is probably due to the small size of the dataset and the very high number of positive examples. Once the size of the dataset increases beyond a limit, the algorithm selected by us should be able to surpass Logistic Regression. Thus, in the real world, if the size of dataset increases by a large amount in the future and we are willing to slightly compromise on the accuracy, we can apply this method to save some precious training time, which was one of the objectives we started with.

References:

[1] P.Rajeswari and G.Sophia Reena, "Analysis of Liver Disorder Using data mining Algorithms", Global Journal of Computer Science and Technology, Volume.10 issue 14(version 1.00 November 2010) PP 48.

[2] Hoon Jin, Seoungchon Kim, Jinhong Kim,"Decision Factors on effective liver patient Data Prediction", International journal of Bio-Science and Bio-Technology, Volume6 No 4(2014) PP 167-168.

[3] Cassangnou M, Boruchowicz A, Guillemot F "Hepatic Steatosis revealing celiac disease: A case complicated by transistory liver failure.AMJ Gastroenterol 1996;91: PP 1291-1292.

[4] Bal MS,Singh SP, Bodal VK, Oberoi SS, Surinder K, "Pathological findings in liver autopsy", Jounal of Indian Academy of Forensic Medicine 2004; 26(2) PP 971-973.

[5] Yao,H.,Hamilton, H.J., Buzz, C.J.,"A foundational Approach to mining itemset utilities from databases", In 4th SIAM International Conference on Data Mining, Florida USA(2004).

[6] Bendi Venkata Ramana, M.rendra Prasad Babu and N.B. Venkaeswarlu, "A Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis", International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011.

[7] A.S.Aneeshkumar and C.Jothi Venkateswaran, "Estimating the Surveillance of Liver Disorder using Classification Algorithms", International Journal of Computer Applications (095-8887), Volume 57-No.6, November 2012.

[8] Bendi Venkata Ramanaland Prof.M.Surendra Prasad Babu, "Liver Classification Using Modified Rotation Forest", Internationa Journal of Engineering Research and Development ISSN: 2278-067X, Volume 1, Issue6, (June 2012), PP.17-24.

[9] H.Ratnamala Kiruba and Dr G. Tholkappia arasu, "An Intelligent Agent based Framework for Liver Disorder Diagnosis Using Artificial Intelligence Techniques", Journal of Theoretical and Applied Information Technology, Vol.69 No.1, 10th November 2014.

[10] S.Dhamodharan, "Liver Disease Prediction Using Bayesian Classification", 4th National Conference on Advanced Computing, Applications & Technologies, Special Issue, May 2014.

- https://ieeexplore.ieee.org/document/6496471
- https://www.ijeat.org/wp-content/uploads/papers/v8i6/F8365088619.pdf 6
- https://pdfs.semanticscholar.org/d463/51b7f02cef84e62f9d9677a03252348a4be7.pdf
- https://pdfs.semanticscholar.org/5f4e/eb74843e501d7ec89eef9e7af9fc0f8c9c1a.pdf
- https://pdfs.semanticscholar.org/3c8c/920d2e8ac4485940faf317e59e7b1eb6d441.pdf
- https://www.irjet.net/archives/V5/i1/IRJET-V5I142.pdf
- https://acadpubl.eu/jsi/2018-118-7-9/articles/9/72.pdf 12
- http://aircconline.com/ijdkp/V8N2/8218ijdkp01.pdf
- https://pdfs.semanticscholar.org/c92d/38a7a76c20a317de63fb9278bb10102c758b.pdf
- https://www.ijcsmc.com/docs/papers/August2017/V6I8201712.pdf

<Thank You Sir...!>