Machine Learning Engineer Nanodegree

Capstone Project Proposal

Classifying Liver Disease dataset

MEKALA NAGA HARISH YADAV

February 6th,2019

Proposal:

Classifying Liver Disease dataset Domain Background:

History:

Problems with liver patients are not easily discovered in an early stage as it will be functioning normally even when it is partially damaged. An early diagnosis of liver problems will increase patient's survival rate. Liver failures are at high rate of risk among Indians. It is expected that by 2025 India may become the World Capital for Liver Diseases. The widespread occurrence of liver infection in India is contributed due to deskbound lifestyle, increased alcohol consumption and smoking. There are about 100 types of liver infections.

1. A patient going to a doctor with certain symptoms.

2. The doctor recommending certain tests like blood test, urine test etc depending on the symptoms.

3. The patient taking the aforementioned tests in an analysis lab.

4. The patient taking the reports back to the reports back to the hospital, where they are examined the disease is identified

Reference Link:

https://www.researchgate.net/publication/319983998_Analysis_of_classification_algorithms_for_liver_disease_diagnosis

Applications:

This project Classifying Liver Disease Data set can be applied in any medical hospital to check the person is infected by liver disease or not.

To serve the medicinal community for the diagnosis of liver disease among patients, a graphical user interface will be developed using python.

The GUI can be readily utilized by doctors and medical practitioners as a screening tool for the liver disease.

Problem Statement:

Given a dataset containing various attributes of 583 Indian patients, define a classification algorithms. To apply different classification algorithms on the Indian patient liver disease dataset than choose the best algorithms based on the accuracy which can identify whether a person is suffering from liver disease or not.

Datasets and Inputs:

The dataset for this problem is the ILPD (Indian Liver Patient Dataset) taken from the UCI Machine Learning Repository

The number of instances are 583. It is a multivariate data set, contain 10 variables that are age, gender, total Bilirubin, direct Bilirubin, total proteins, albumin, A/G ratio, SGPT, SGOT and Alkphos. All values are real integers

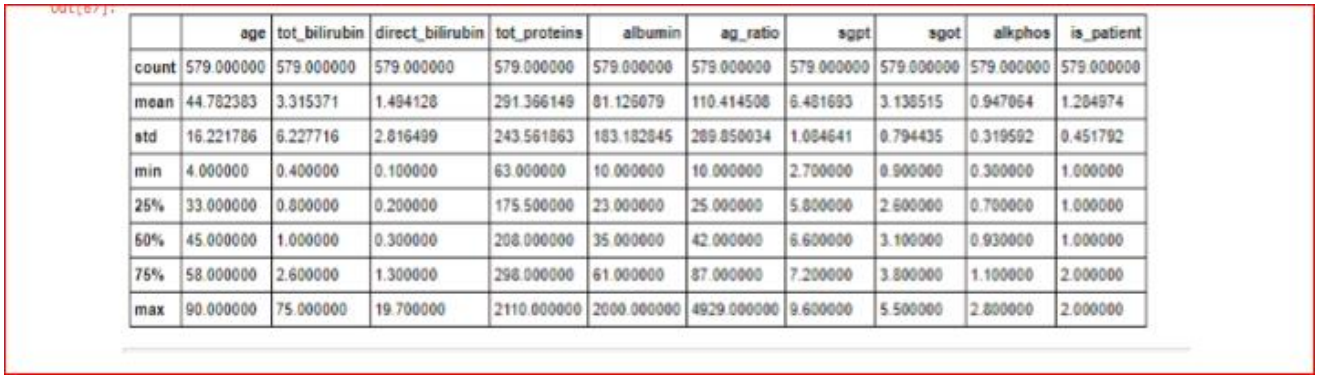
Based on chemical compounds(bilrubin,albumin,protiens,alkaline phosphatase) present in human body and tests like SGOT , SGPT the outcome mentioned whether person is patient ie needs to be diagnosed or not.

.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. ‘dataset‘ label '1' representing presence of disease and '2' representing absence of disease.

Here this data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India.

So,the data set is an unbalanced data set

Example of the data set:



	age	tot_bilirubin	direct_bilirubin	tot_proteins	albumin	ag_ratio	sgpt	sgot	alkphos	is_patient
count	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000
mean	44.782383	3.315371	1.494128	291.366149	81.126079	110.414508	6.481693	3.138515	0.947064	1.284974
std	16.221786	6.227716	2.816499	243.561863	183.182845	289.850034	1.084641	0.794435	0.319592	0.451792
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.000000
75%	58.000000	2.600000	1.300000	298.000000	61.000000	87.000000	7.200000	3.800000	1.100000	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.000000

Solution Statement:

To solve this problem, I will be using one or more classification algorithms covered in the udacity Machine Learning . First explore the data set and using visualizations which helps me to better understand the solution.Then we will find the accuracy score for each classification model then find best classification algorithm for liver disease.

I will be trying out Logistic Regression, knearest neighbours and one ensemble method. (Adaboost) for this project

Different combinations of hyperparameters for individual algorithms , like kernel, degree and C for SVM and weights, n_neighbours and algorithms for k-Nearest Neighbours will be tried across the training sets. Depending on their respective performances on the cross-validation sets, the best algorithm with appropriate hyperparameter tuning will be finalised as the solution.

Benchmark Model:

However the problem lies in finding a dataset where the results are given in such a fashion which is easily comparable with our classification values. In datasets it is intrinsically difficult to compare the scores given with our outputs. Therefore, we will use a simple algorithm like Naïve bayes as our benchmark model and try to improve upon its performance by using other algorithms like Naïve bayes,SVM, ensemble methods etc. If i classifies the data applying on different algorithms we got the accuracy_score and f-score with minimum 50% accuracy.

Evaluation Metric:

I want to use accuracy score as evaluation metric for prediction of liver disease.The performance of a model cannot be assessed by considering only the accuracy, because there is a possibility for misleading. Therefore this experiment considers the F1 score along with the accuracy for evaluation.. This is because depending on the context like severity of disease, sometimes it is more important that an algorithm does not wrongly predict a disease as a nondisease.

Thus, here we will use F-beta score as a performance metric, which is basically the weighted harmonic mean of precision and recall. Precision and Recall are defined as:

Precision=TP/ (TP+FP), Recall=TP/ (TP+FN), where

TP=True Positive

FP=False Positive

FN=False Negative

In the same vein, F-beta score is:

F-beta score = (1+β^2)*precision*recall/((β^ 2*precision)+recall)

We can use F-beta score as a metric that considers both precision and recall

Additionally, one more metric called as Receiver Operating Characteristics (ROC) curve will be used. It plots the curve of True Positive Rate and the False Positive Rate for a given algorithm, with a greater area under the curve indicating a better True Positive Rate for the same False Positive Rate, indicating the usefulness of the classifier.

Analysis

Exploring the Data :

The Indian liver patient 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

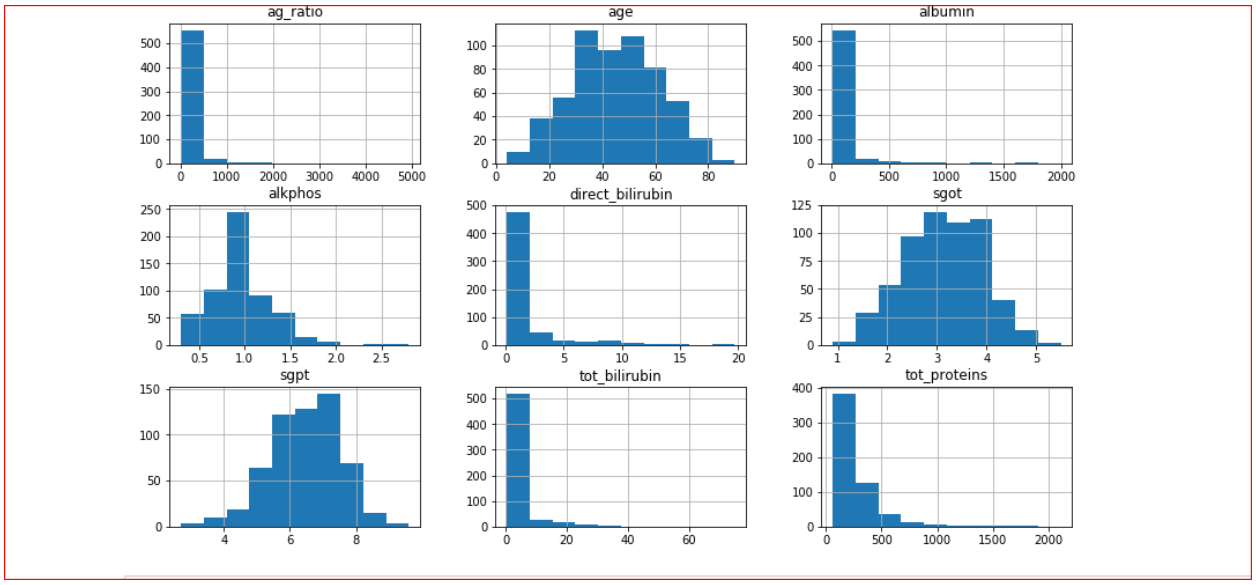
All features, except Gender are real valued integers. The last column, is_patient, is the label with '1' representing presence of disease and '2' representing absence of disease. Total number of data points is 583, with 416 liver patient records and 167 non liver patient records. A brief description of dataset, including parameters like mean, min, max for each column is given below:

Out[17]:

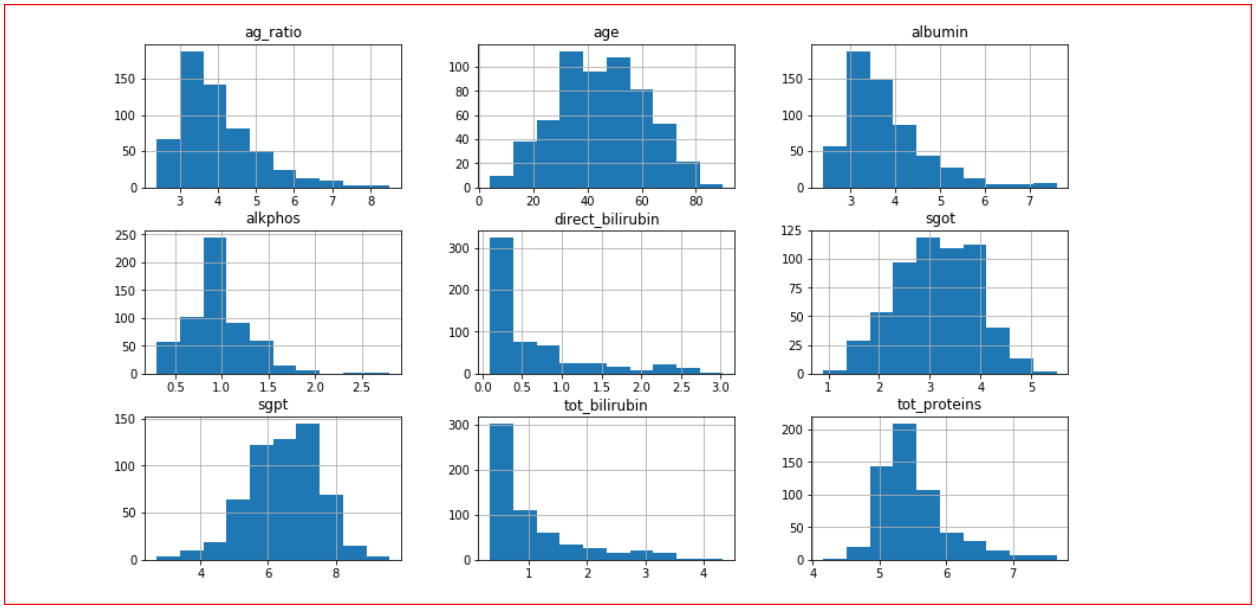
	age	tot_bilirubin	direct_bilirubin	tot_proteins	albumin	ag_ratio	sgpt	sgot	alkphos	is_patient
count	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000	579.000000
mean	44.782383	3.315371	1.494128	291.366149	81.126079	110.414508	6.481693	3.138515	0.947064	1.284974
std	16.221786	6.227716	2.816499	243.561863	183.182845	289.850034	1.084641	0.794435	0.319592	0.451792
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	1.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	1.000000
75%	58.000000	2.600000	1.300000	298.000000	61.000000	87.000000	7.200000	3.800000	1.100000	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.000000

Exploratory Visualization :

After removing the column' is_patient 'from the dataset as it is the label, we display all features in a histogram format to check if any feature is skewed (contains a small number of outlier values).



Skewed features found are Albumin, Direct Bilirubin, A/G ratio, Total Bilirubin, Total Protein. On these, a log transformation is applied to reduce their range. Again, all the transformed features are shown in a histogram format:



Algorithms and Techniques:

Three supervised learning approaches are selected for this problem are:

1.Logistic Regreesion

2.Adaboost

3.Support vector machine

For each algorithm, we will try out different values of a few hyper parameters to arrive at the best possible classifier. This will be carried out with the help of grid search cross validation technique. For these dataset we apply the different supervised learning algorithms there are:

AdaBoostClassifier:

Adaboost is a boosting type ensemble learner. This method works by combining multiple individual "weak" learning hypotheses to create one strong model. Each weak hypothesis used is better at classifying the data than random chance. However, it's the combination of all of these independent weak learning hypotheses what makes the model more capable of predicting accurately on unseen data than each of the individual hypothesis would.

n_estimators: The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

learning_rate: Learning rate shrinks the contribution of each classifier by learning_rate. There is a trade-off between learning_rate and n_estimators.

Advantages:

- AdaBoost is the of the ensemble method. The AdaBoost is more robust than single estimators, have improved generalizability.
- Simple models can be combined to build a complex model, which is computationally fast

Disadvantages:

- If we have a biased underlying classifier, it will lead to a biased boosted model.
- AdaBoost can be sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms.

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.

Advantages:

- Performs well with high dimensional data. SVM's are very good when we have no idea on the data.
- Works well with even unstructured and semi structure data like text, Images. The kernel trick is strength of SVM. By using the kernel function to solve the complex problem.
- The SVM model have generalization in practice, the risk of overfitting is less in SVM.

Disadvantages:

- Choosing the good kernel function is not easy and it take long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights and individual impact.

Logistic Regression:

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

Adavantages:

It is a widely used technique because it is very efficient, does not require too many computational resources, it's highly interpretable, it doesn't require input features to be scaled, it

doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities.

Like linear regression, logistic regression does work better when you remove attributes that are unrelated to the output variable as well as attributes that are very similar (correlated) to each other. Therefore Feature Engineering plays an important role in regards to the performance of Logistic and also Linear Regression. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train. I typically start with a Logistic Regression model as a benchmark and try using more complex algorithms from there on.

Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is also a good baseline that you can use to measure the performance of other more complex Algorithms.

Disadvantages:

A disadvantage of it is that we can’t solve non-linear problems with logistic regression since it’s decision surface is linear.

Benchmark:

However the problem lies in finding a dataset where the results are given in such a fashion which is easily comparable with our classification values. In datasets it is intrinsically difficult to compare the scores given with our outputs. Therefore, we will use a simple algorithm like Logistic Regression as our benchmark model and try to improve upon its performance by using other algorithms like SVM, ensemble methods etc. If i classifies the data applying on different algorithms we got the accuracy_score with minimum 50% accuracy.

Data Preprocessing :

Some datasets contain irrelevant information, noise, missing values, and so on. These datasets should be handled properly to get a better result for the data mining process. Data preprocessing includes data cleaning, preparation, transformation, and dimensionality reduction, which convert the raw data into a form that is suitable for further processing.

As explained in the section 'Exploring the data', rows having 'Null' values were removed from the dataset. Thereafter, log transformation was applied to features which were showing a skewed pattern (Albumin, Direct Bilirubin, A/G ratio, Total Bilirubin, Total Protein).

Thereafter, all columns in the dataset except ‘Gender’ are normalized. We use MinMaxScaler here as StandardScaler gives very low values here, with some in the order of 10^-16, which might be difficult to relate to and visualize.

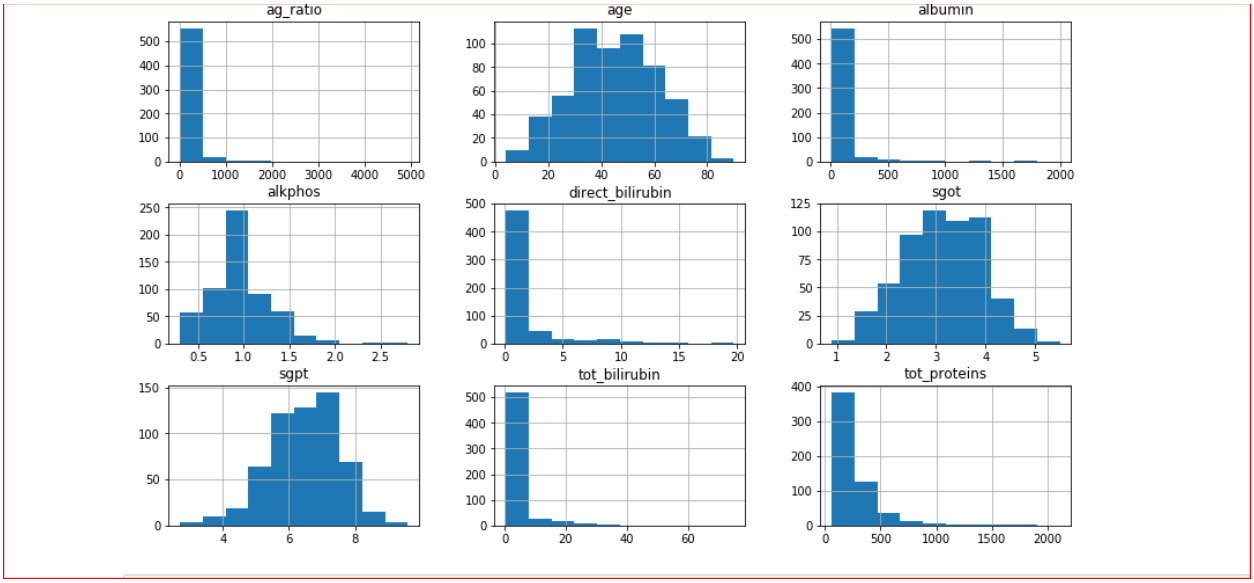
Then we use pd.get_dummies() method to one-hot encode the feature ‘gender’ as well as the label ‘is_patient’ with the integer ‘1’ representing the presence of disease and ‘2’ representing the not presence of disease.

Data Visualization:

The visualization of the data, identifying the outliers, and skewed predictors. These tasks help to inspect the data and thereby spot the missing values and irrelevant information in the

dataset. A data cleanup process is performed to handle these issues and to ensure data quality. Gaining a better understanding of the dataset helps to identify useful information and supports decision making.

The Indian Liver Patient dataset consists of 579 records in which 416 are records of people with liver disease, and the remaining are records of people without any liver disease. The dataset has 10 features in which there is only one categorical data . The endmost column of the dataset represent the class in which each sample falls liver patient or not. A value of 1 indicates the person has liver disease and a 2 indicates the person does not have the disease. There is no missing value in the dataset.



Implementation :

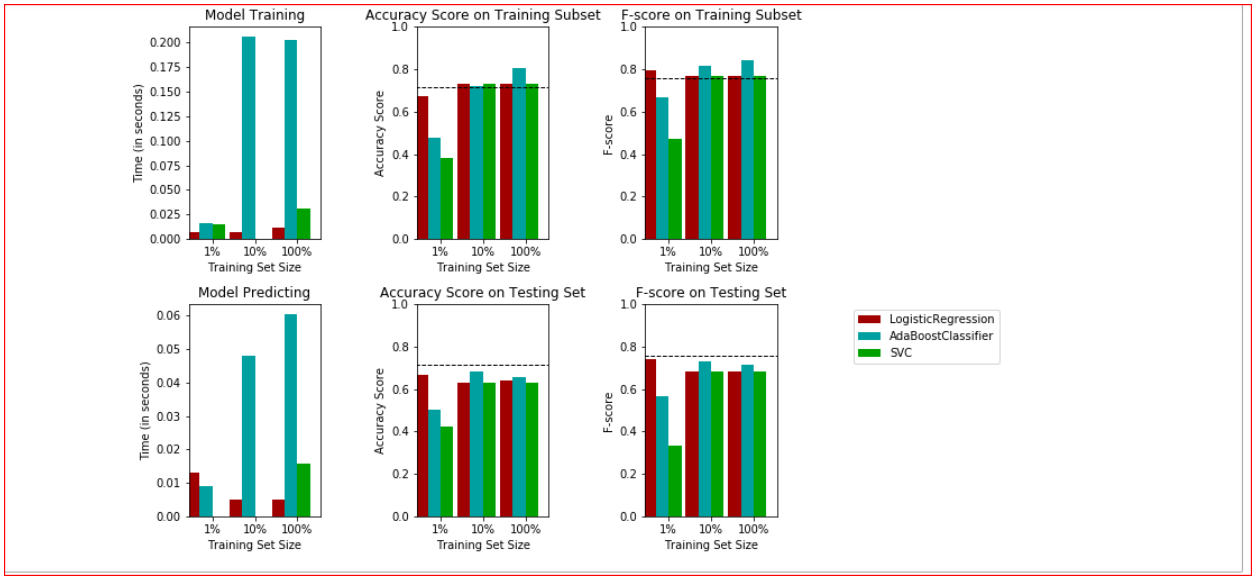
The dataset will be split into training and testing set as a 80% of training data and 20% of testing data can be split using train_test_split method from sklearn. Random state will be specified as a particular number .

Before applying any supervised learning technique, we will implement a naïve predictor, that will simply return that every data point has ‘Disease’= True. We will check our accuracy on that predictor. Note that in this case naive predictor will perform artificially well unlike in real world, a large proportion of patients has around 70% do have the disease.

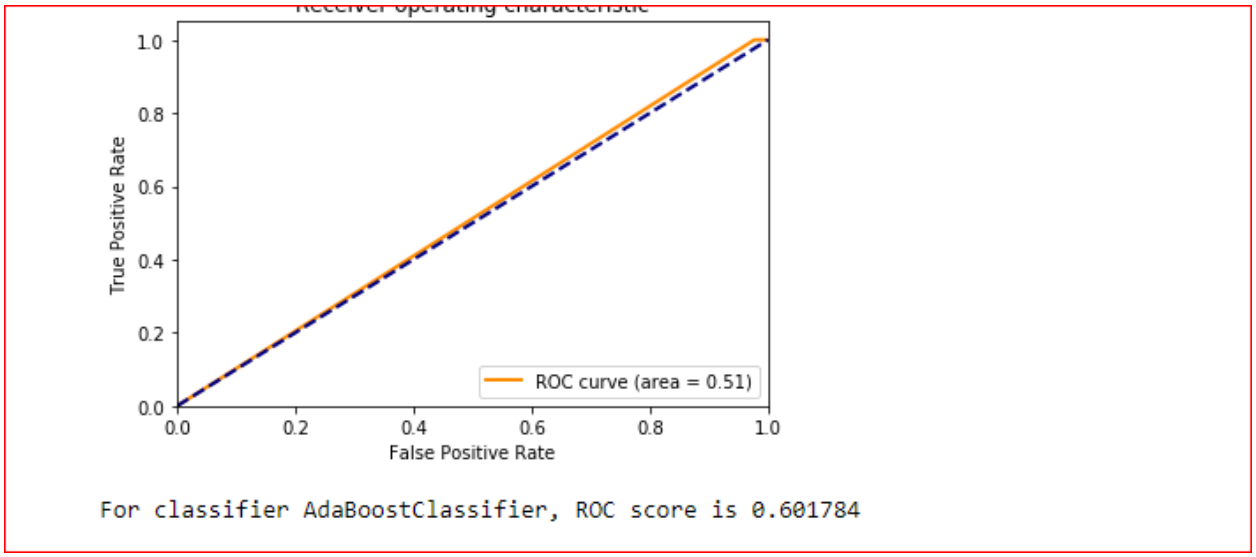
Then, a method called as 'train_predict' is defined that takes as input the following: learner, sample_size, X_train, y_train, X_test, y_test. It returns the accuracy and F-beta score on training and testing set respectively.

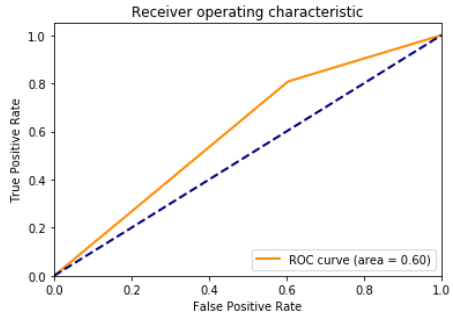
The three classifiers are sent to the 'train_predict' method, with 1%, 10% and 100% of training data respectively so that it can be seen how performance varies with sample size.

Initially, some difficulty was observed in outputting the results in the desired manner. The remedy was found in the form of a python file called as 'visuals.py' , that returns all the output variable in a bar graph format. This file has been derived from the 'finding_donors' project with some changes implemented. The output from this file is as follows:

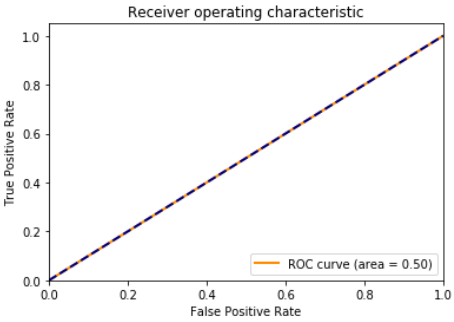


An additional metric called as Receiver Operator Characteristics(ROC) curve will be used. It plots the curve of True Positive Rate and False positive Rate, with a greater area under the curve indicating a better True Positive Rate for the same False Positive Rate. This can be helpful in this case as simply knowing the number of correct predictions may not suffice.






```
For classifier SVC, ROC score is 0.500000
```



Refinement:

As mentioned in the benchmark section, comparing the f-score of the three models Here for the improving the results I am using Grid search technique

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. And the code is

Note: Depending on the algorithm chosen and the parameter list, the following implementation may take some time to run!

In [12]:
```
# TODO: Import 'GridSearchCV', 'make_scorer', and any other necessary libraries
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV
# TODO: Initialize the classifier
from sklearn.ensemble import AdaBoostClassifier
# TODO: Initialize the classifier
clf = AdaBoostClassifier(random_state=0)
# TODO: Create the parameters list you wish to tune, using a dictionary if needed.
# HINT: parameters = {'parameter_1': [value1, value2], 'parameter_2': [value1, value2]}
parameters = {'n_estimators':[50,75,100],'learning_rate':[2.0,3.0,1.0]}
# TODO: Make an fbeta_score scoring object using make_scorer()
scorer = make_scorer(fbeta_score,beta=0.5)
# TODO: Perform grid search on the classifier using 'scorer' as the scoring method using GridSearchCV()
grid_obj = GridSearchCV(clf,param_grid=parameters,scoring=scorer)
# TODO: Fit the grid search object to the training data and find the optimal parameters using fit()
grid_fit = grid_obj.fit(X_train,y_train)
# Get the estimator
best_clf = grid_fit.best_estimator_
# Make predictions using the unoptimized and model
predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)
# Report the before-and-afterscores
print("Unoptimized model\n------")
print("Accuracy score on testing data: {:.4f}".format(accuracy_score(y_test, predictions)))
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, predictions, beta = 0.5)))
print("\nOptimized Model\n------")
print("Final accuracy score on the testing data: {:.4f}".format(accuracy_score(y_test, best_predictions)))
print("Final F-score on the testing data: {:.4f}".format(fbeta_score(y_test, best_predictions, beta = 0.5)))
```

And the f-score scores is as follows

Here I am predicting the accuracy score ,F_score and tesing, traing time for the selected models. Despite the ambiguous results, it was decided to select Adaboost Classifier as the final classifier and Final accuracy score on the testing data: 0.6724

Final F-score on the testing data: 0.7284

Here the parameters I used for Adaboost is:

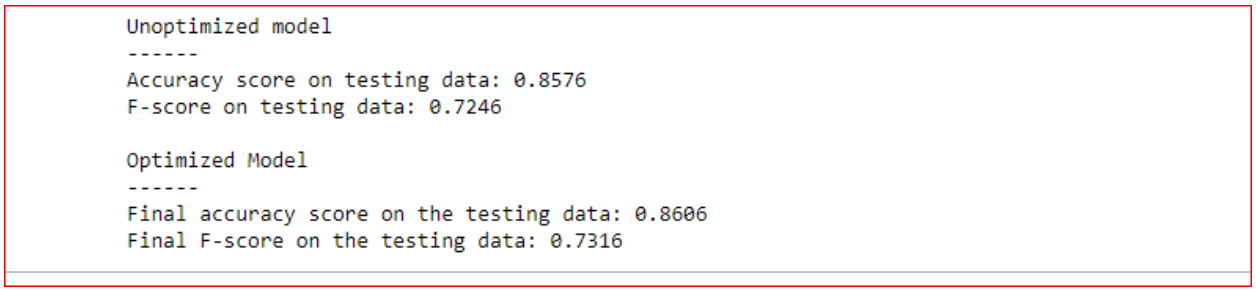
n_estimators : integer, optional (default=50)

The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early.

learning_rate : float, optional (default=1.)

Learning rate shrinks the contribution of each classifier by learning_rate. There is a trade-off between learning_rate and n_estimators.

By using this parameters the results is:



The above image define the change of f-score and accuracy score before using Grid Search technique and after using Grid Search Technqiue

Results

Model Evaluation and Validation :

Since the size of dataset is small at present , there is not much difference between training and testing times of different algorithms. However, for the sake of comparison, these times have been displayed in the 'Implementation' sub-heading of 'Analysis' section. Adaboost Classifier

consumes maximum time during training and good accuracy score ,F_score. From this dataset use three models based on the accuracy_score,F_score we decide Adaboost is best suitable for this dataset.

LogisticRegression

	1%	10%	100%
acc_test	0.663793	0.629310	0.637931
acc_train	0.675000	0.730000	0.730000
f_test	0.741840	0.679702	0.684803
f_train	0.797214	0.771670	0.771670
pred_time	0.012990	0.005004	0.004976
train_time	0.006996	0.006995	0.012014

AdaBoostClassifier

	1%	10%	100%
acc_test	0.500000	0.681034	0.655172
acc_train	0.480000	0.720000	0.805000
f_test	0.563910	0.731415	0.714286
f_train	0.666667	0.814448	0.845865
pred_time	0.009015	0.048118	0.060465
train_time	0.015995	0.205601	0.203114

SVC

	1%	10%	100%
acc_test	0.422414	0.629310	0.629310
acc_train	0.380000	0.730000	0.730000
f_test	0.333333	0.679702	0.679702
f_train	0.470085	0.771670	0.771670
pred_time	0.000000	0.000000	0.015624
train_time	0.015622	0.000000	0.031245

Here to validate the model AdaBoost I used the parameter random_state

If you don't specify the random_state in your code, then every time you run(execute) your code a new random value is generated and the train and test datasets would have different values each time.

So,using different random_State values the results are:

When random_state=0: result is f-score :0.73,accuracyscore:0.86

When random_state=5: result is f-score :0.72,accuracyscore:0.67

When random_state=10: result is f-score :0.72,accuracyscore:0.67

By doing small changes in the model will not affect the model because we are using the optimization technique is Grid Search.So by using the data set with same features the model will run correctly

Justification :

I used accuracy score and f_score as evaluation metric for prediction of liver disease. Here I am predicting the accuracy score ,F_score and tesing, traing time for the selected models. Despite the ambiguous results, it was decided to select Adaboost Classifier as the final classifier and Final accuracy score on the testing data: 0.6724

Final F-score on the testing data: 0.7284

When we compare Logistic Regression and SVM the adaboost classifier has good accuracy_score and f_score but it will take more training and testing time compare to other models.

When we see the area under the ROC curve 0.60 is greater than that of Logistic Regression(0.50) and SVM(0.51) . As discussed before, we are considering ROC score as an important metric.

Conclusion

Free Form Visualization :

ROC curves that show the performance of the respective algorithms before and after applying GridSearchCV have been displayed in the preceding sections.

Reflection :

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 were run on the dataset to determine the benchmark . 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 GridSearchCV. The models were compared on their F-beta as well as their ROC scores. The results turned out to be ambiguous, with SVM,Logistic Regression and Adaboost running for best performance. The performance with respect to F-beta score was also marginally better SVM, Logistic Regression and slightly worse for Adaboost. However, it is believed that these algorithms will improve once they are presented with a much larger dataset. I am learned the intreseted topic is visualization of the distribution function for every feature.

Secondly, fine tuning models using GridSearch was also tough, as it was computationally too expensive to include all the parameters for Adaboost. For this, I went through the definitions of hyperparameters in scikit-learn documentation, and then selected the best performing parameters and their values on the basis of a limited number of trials. This exercise made me realize that parameter tuning is not only a very interesting but also a very important part of machine learning. I think this area can warrant further improvement, if we are willing to invest a greater amount of time as well as computing power.

Improvement:

One way for improving is to conduct a more exhaustive search for a combination of parameters which might give better results using Grid Search, but that is computationally expensive. Also, can save some training time at the cost of accuracy by extracting the most influential features using features_importance_ attribute of the AdaBoost and training the model only on those features. This is implemented in the final portion of the project, and an Fbeta score of 0.72 was obtained on predicting with 5 most influential features.