Capstone Project

Predicting Liver disease from data

Machine Learning Engineer Nanodegree

Dinkar Juyal

09/03/2017

Definition

Project Overview

In India, delayed diagnosis of diseases is a fundamental problem due to a shortage of medical professionals. A typical scenario, prevalent mostly in rural and somewhat in urban areas is:

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 and the disease is identified.

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. Historically, work has been done in identifying the onset of diseases like heart disease, Parkinson's from various features, for example in this paper https://link.springer.com/chapter/10.1007/978-3-319-11933-5_17.In this case, a machine learning algorithm will be trained to predict a liver disease in patients.

Problem Statement

The problem statement is formally defined as:

‘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, 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. 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. 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".

Relevant sources: Bendi Venkata Ramana, Prof. M. S. Prasad Babu and Prof. N. B. Venkateswarlu, â€œA Critical Comparative Study of Liver Patients from USA and INDIA: An Exploratory Analysisâ€�, International Journal of Computer Science Issues, ISSN :1694-0784, May 2012.

Metrics

In problems of disease classification like this one, simply comparing the accuracy, that is, the ratio of correct predictions to total predictions is not enough. 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 non-disease, while predicting a healthy person as diseased will attract a comparatively less severe penalty.

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 = $(1+\beta^{2})$*precision*recall/((β^{2}*precision)+recall)

β = A number that decides relative weightage of precision and recall. In this case, a disease being classified as a non-disease will incur a high penalty. So, more emphasis is placed on recall.

Additionally, one more metric called as Receiver Operating Characteristics (ROC) curve will be used. It plots the curve of True Positive Rate vs 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.

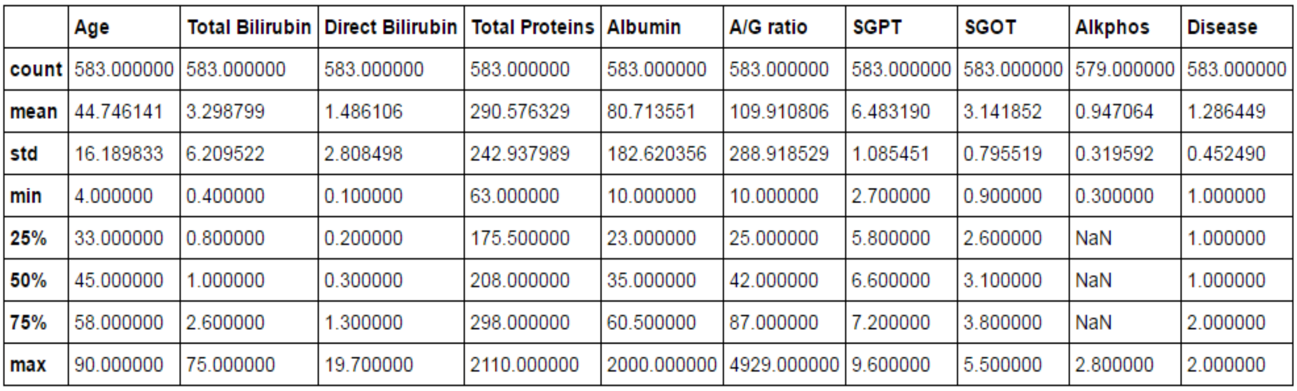
Analysis

Exploring the Data

The ILPD dataset contains ten features as listed below:

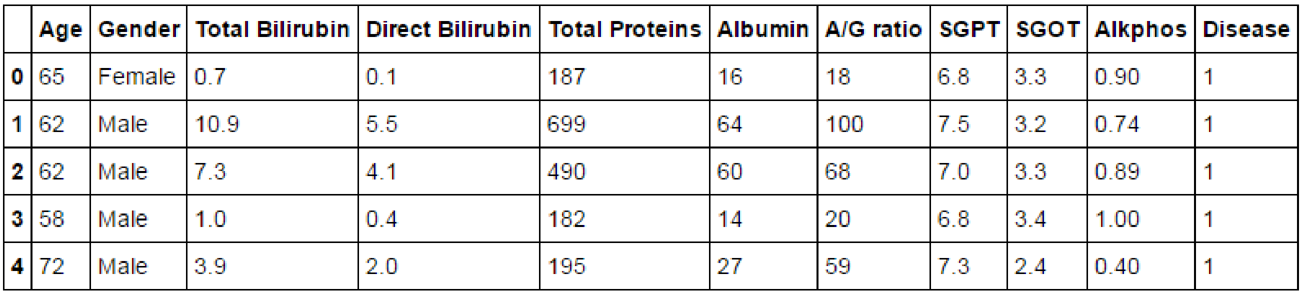
1. Age
2. Gender
3. Total bilirubin
4. Direct bilirubin
5. Total proteins
6. Albumin
7. A/G ratio
8. SGPT
9. SGOT
10. Alkphos

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. A brief description of dataset, including parameters like mean, min, max for each column is given below:



	Age	Total Bilirubin	Direct Bilirubin	Total Proteins	Albumin	A/G ratio	SGPT	SGOT	Alkphos	Disease
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	1.286449
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.452490
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	NaN	1.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	NaN	1.000000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	NaN	2.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	2.000000

A snapshot of the dataset containing first 5 rows is as follows:

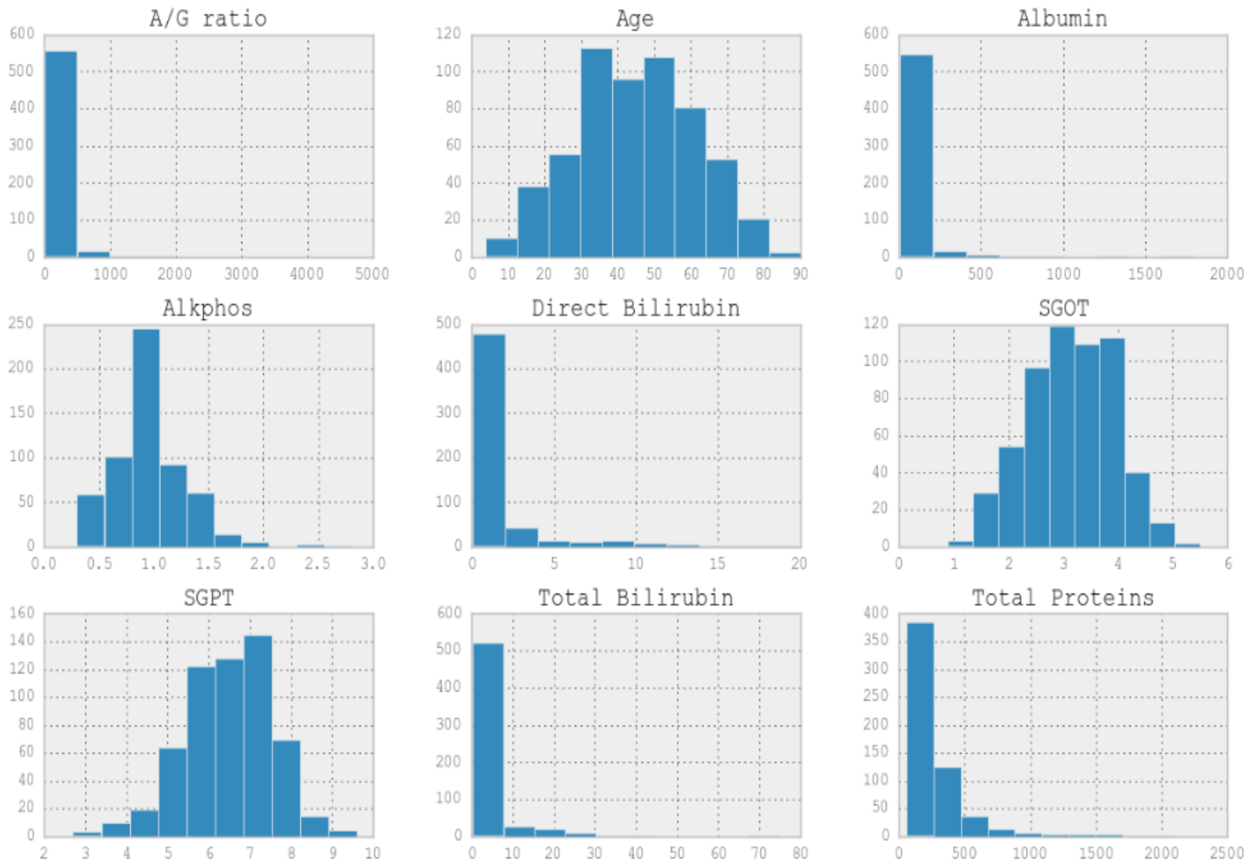


	Age	Gender	Total Bilirubin	Direct Bilirubin	Total Proteins	Albumin	A/G ratio	SGPT	SGOT	Alkphos	Disease
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1

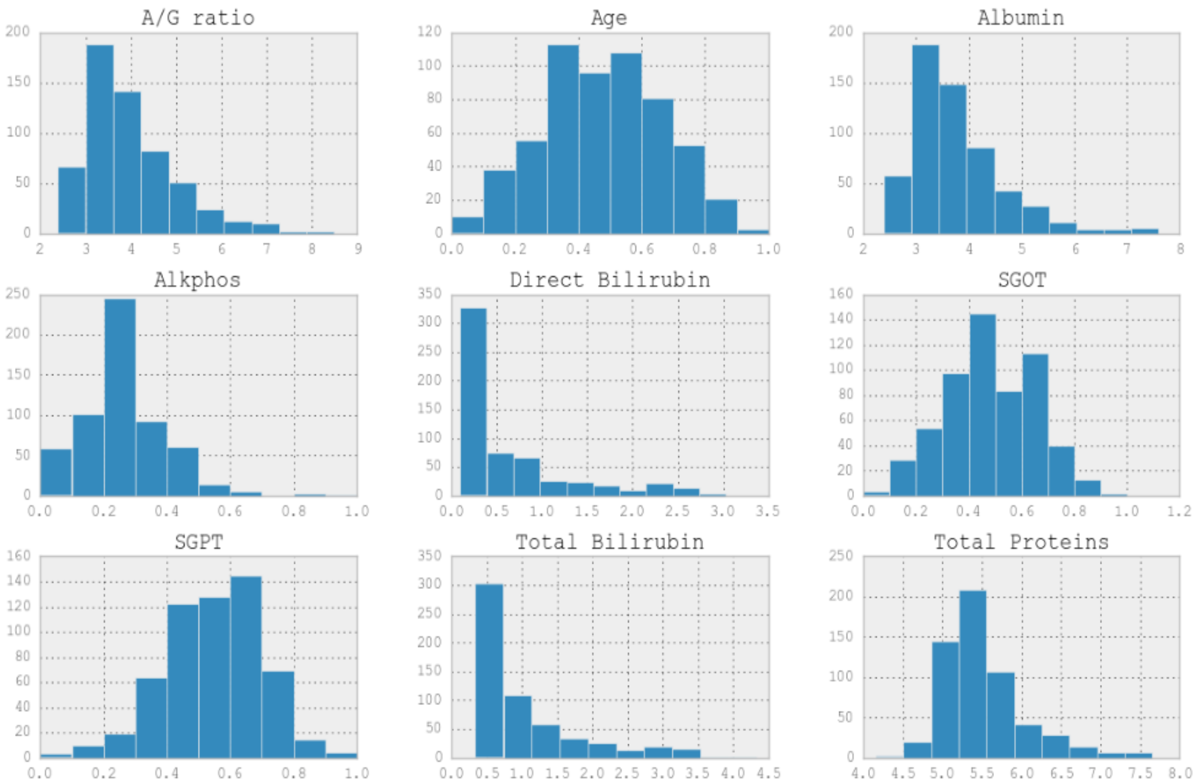
In the description of this dataset, it is observed that some values are ‘Null’ for the ‘Álkphos’ column. Accordingly, 4 rows containing those values are removed. Remaining number of rows in the dataset is 579.

Exploratory Visualization

After removing the column ‘Disease’ 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. Care is taken that all these approaches are fundamentally different from each other, so that we can cover as wide an umbrella as possible in term of possible approaches. For example- We will not select Random Forest and Ada Boost together as they come from the same family of 'ensemble' approaches. The choice of algorithms was influenced from these two sources:
http://scikit-learn.org/stable/tutorial/machine_learning_map

http://stackoverflow.com/questions/2595176/when-to-choose-which-machine-learning-classifier

For each algorithm, we will try out different values of a few hyperparameters to arrive at the best possible classifier. This will be carried out with the help of grid search cross validation technique. The algorithms are described below:

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)

Pros:

- Through combining base learners, Random forest improves the performance of the weak algorithm.
- Random forests are important when there are multiple correlated features.

Cons:

- Takes greater computational time to train.
- Has a tendency to overfit especially if number of examples is less, unless hyperparametersare properly adjusted.

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)

Pros:

- Performs well with high dimensional data
- Performs well with non-linear boundary if appropriate kernel used

Cons:

- Expensive to train

3. **K- Nearest Neighbors:** KNN classifies a given data point by looking at its neighbors and assigning weights to them in such a way that the closest neighbours have a greater say in determining the class. Here, distance can be Euclidean Distance, Minkowski Distance etc.

Hyperparameters to be manipulated:

- n_neighbors(number of neighbors)
- weights(the degree of influence various data points possess)

Pros:

- Simple to implement
- Flexible with regards to distance of data points

Cons:

- All heavy computational work takes place during testing

Benchmark

Some models have been created to analyze the chances of liver injury in critically ill patients. The following paper gives one such model: https://www.ncbi.nlm.nih.gov/pubmed/26820880. It assigns a LiFe score from 0 to 10 to patients: with 0 denoting low risk and >8 denoting very high risk. The authors of this paper have stated that: 'a significant positive correlation exists between LiFe score and acute-on-chronic liver failure grade, (r = 0.478, P < 0.001)'.

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 like the one mentioned above, 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.

Logistic Regression: Since the outcome is binary and we have a reasonable number of examples at our disposal compared to number of features, this approach seems suitable. At the core of this method is a logistic or sigmoid function that quantifies the difference between each prediction and its corresponding true value. When presented with a number of inputs , it assigns different weights to features(based on their relative importance).Since for this data it already knows the output beforehand, it continuously adjusts the weights such that when these weights summed up with their features are introduced in the logistic function, the results are as near as possible to the actual ones. Once presented with a test value, it again inserts the value into our logistic function and returns the output as a number between 0 and 1, which represents the probability of that test value being in a particular class.

Beating this benchmark model means that our method is suitable to be applied in the real world, as the problem dataset inherently favours Logistic Regression in terms of limited sample size and large number of positive examples (people having the disease). In real world, a much greater dataset size can be created due to the large population, and the percentage of positive cases will also be quite less. In such cases, the algorithms chosen are known to perform better, and our assumption of placing more emphasis on recall will also be better placed. So, if any one of these models manages to beat or even have comparable performance metrics to Logistic regression, it will have a high probability of giving a better performance in a real world scenario.

Methodology

Data Preprocessing

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. Transformation is given by:

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min

Then we use pd.get_dummies() method to one-hot encode the feature ‘Gender’ as well as the label ‘Disease’ (with the integer ‘1’ representing presence of disease).

Implementation

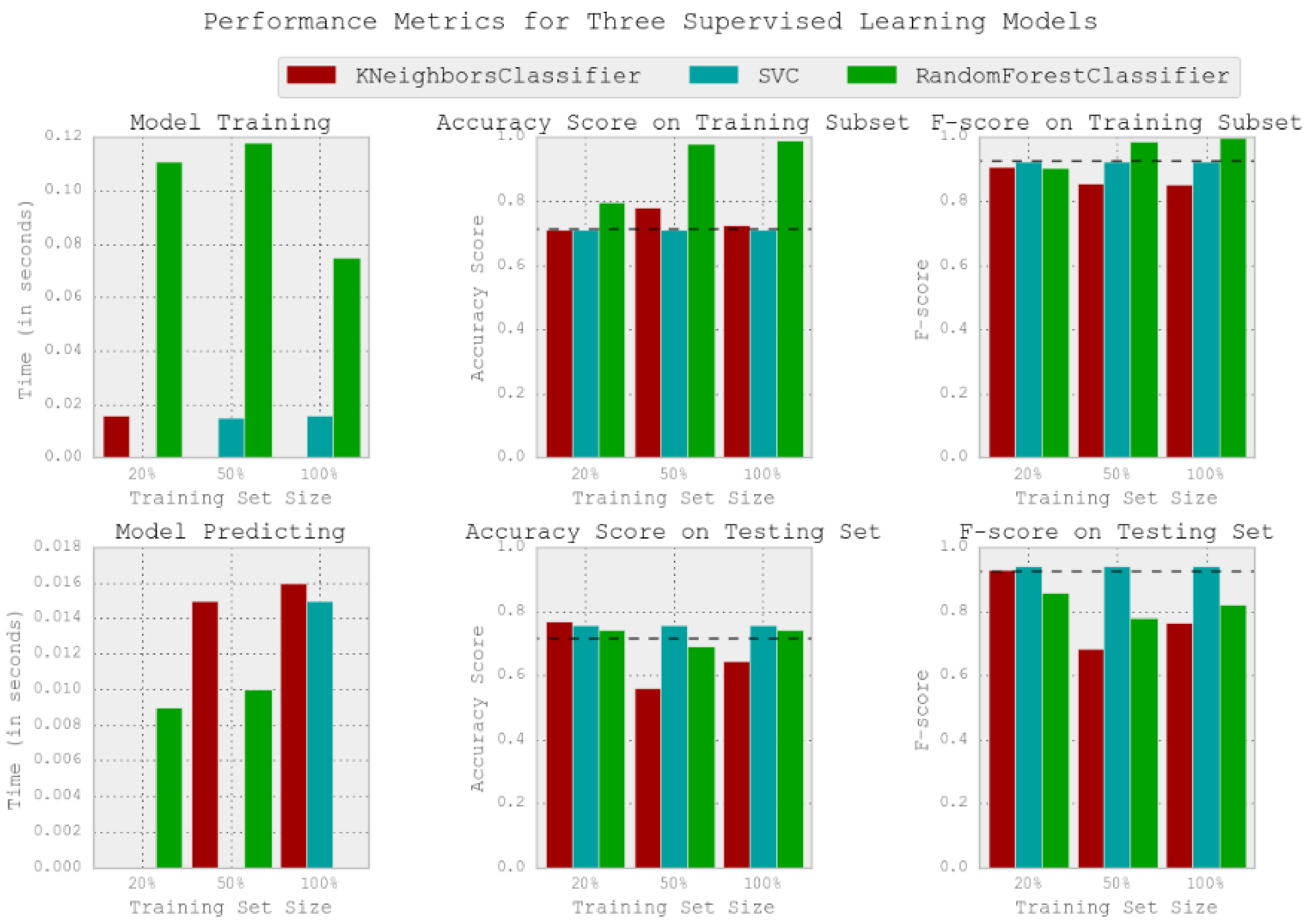
The dataset will be split into training and testing set as a 80-20 split using train_test_split method from sklearn. Random state will be specified as a particular number so that we have a means for comparison later, by specifying the same random state.

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 (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.

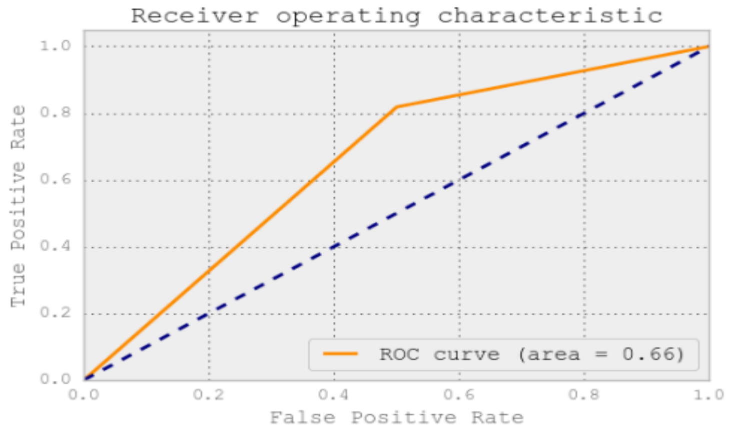
The three classifiers are sent to the 'train_predict' method, with 20%, 50% and 100% of training data respectively so that it can be seen how performance varies with sample size.

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:

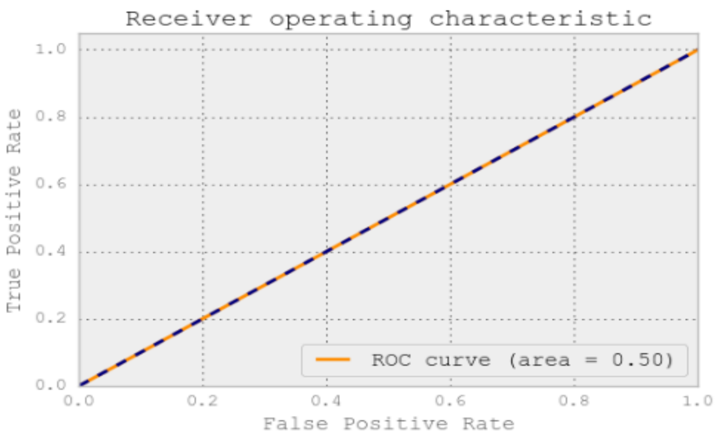


ROC curves are also plotted for the three algorithms:

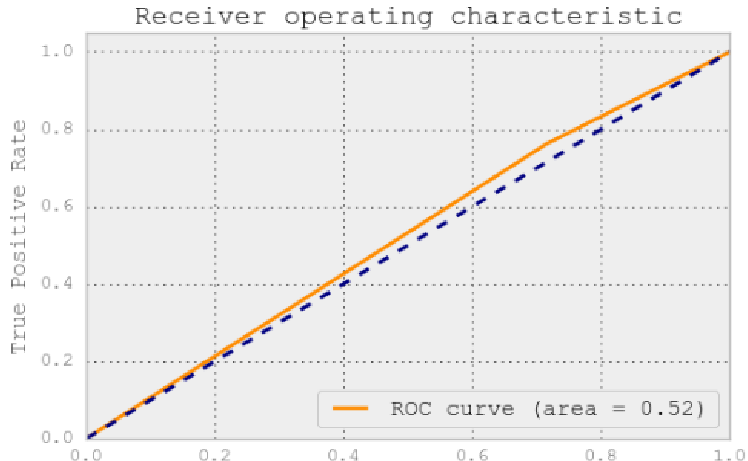
```
For classifier RandomForestClassifier, ROC score is 0.659091
```



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



```
For classifier KNeighborsClassifier, ROC score is 0.523539
```



Refining The Models

It is not very clear which model is the best performer: SVM gives the best scores on testing set, while Random Forest Classifier gives the most area under ROC curve. So, we will test both these models on a variety of parameters using the GridSearchCV technique. For the sake of comparison, we will include KNN in the fray too. The number of parameters used is constrained by the computational time taken to compute the results. Hyperparameters and their values for different classifiers are:

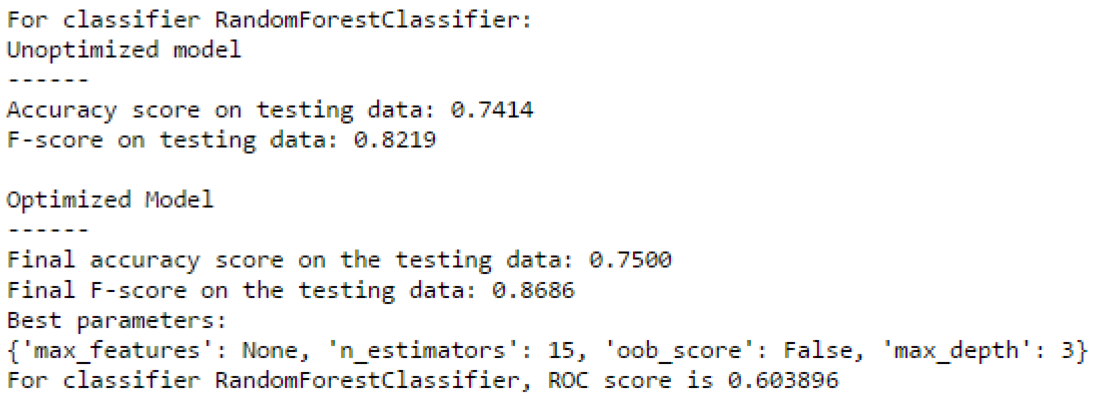
Random Forest Classifier	
Parameter	**Values**
max_features	[auto, None]
oob_score	[False,True]
max_depth	[3,10,15]
n_estimators	[3,10,15]
SVM	
kernel	[poly,rbf,linear]
c	[0.001,1,1000]
K nearest neighbors	
n_neighbors	[5,10,15]
weights	[uniform,distance]

Results

Model Evaluation and Validation

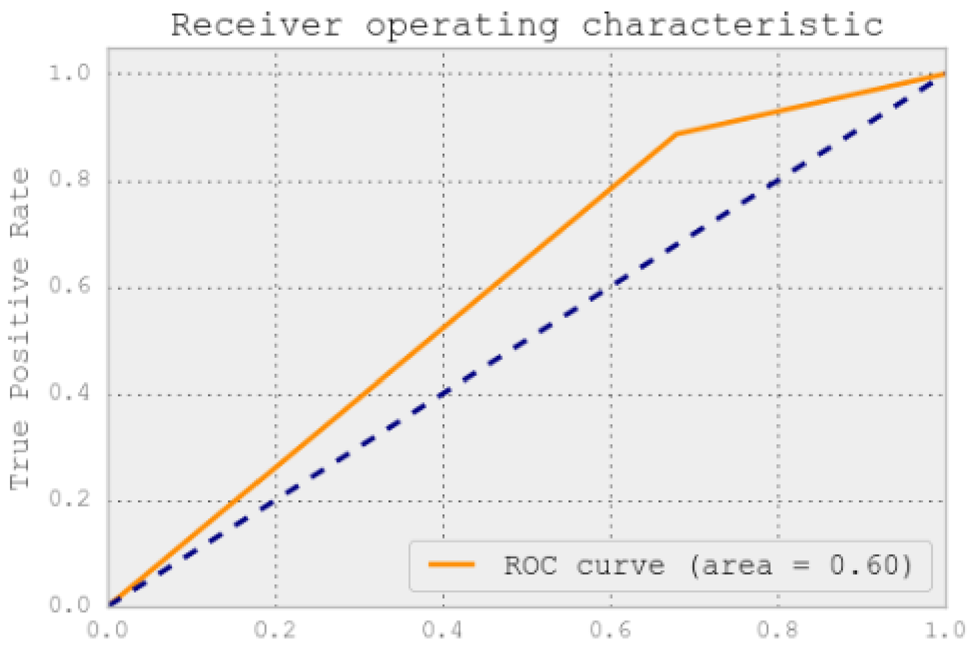
Since the size of dataset is small at present , there is not much difference between training and testing times of different algorithms. However, for the sake of comparison, these times have been displayed in the 'Implementation' sub-heading of 'Analysis' section. Random Forest Classifier consumes maximum time during training, while KNN takes the maximum time during testing.

Details of performance metrics for each classifier after implementing GridSearchCV and choosing the best combination of their parameter values is given below:



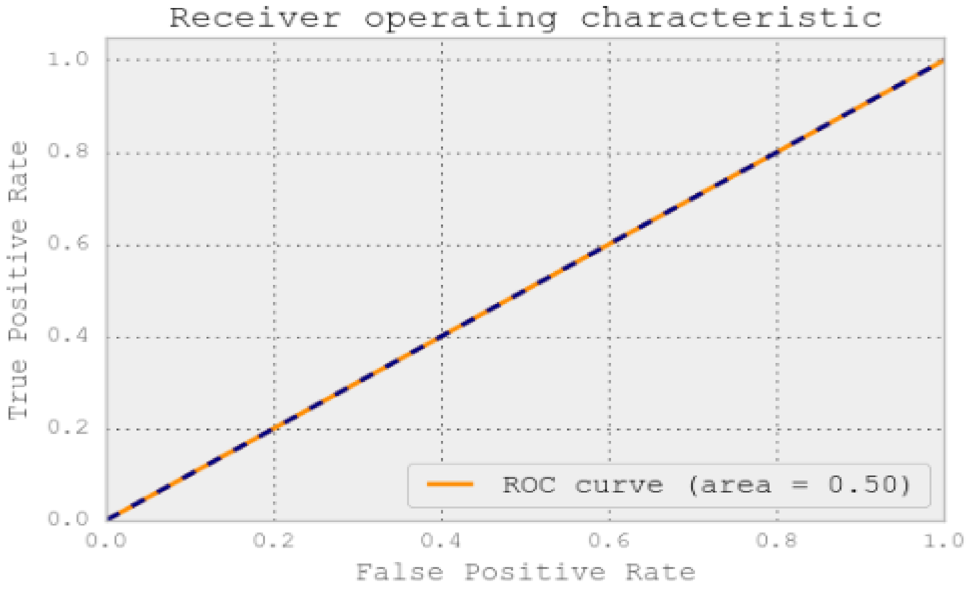
```
For classifier RandomForestClassifier:
Unoptimized model
------
Accuracy score on testing data: 0.7414
F-score on testing data: 0.8219

Optimized Model
------
Final accuracy score on the testing data: 0.7500
Final F-score on the testing data: 0.8686
Best parameters:
{'max_features': None, 'n_estimators': 15, 'oob_score': False, 'max_depth': 3}
For classifier RandomForestClassifier, ROC score is 0.603896
```



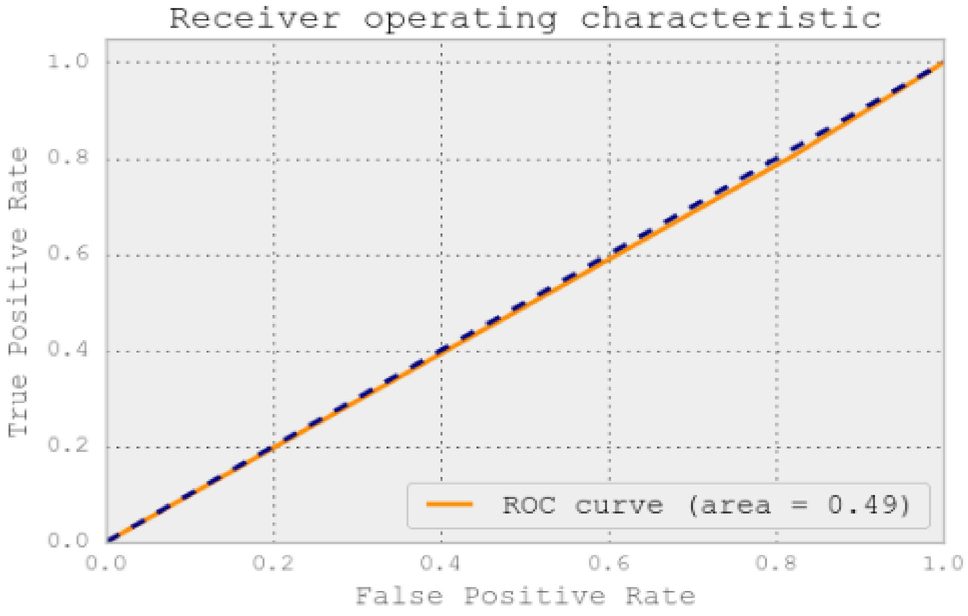
```
For classifier SVC:
Unoptimized model
------
Accuracy score on testing data: 0.7586
F-score on testing data: 0.9402

Optimized Model
------
Final accuracy score on the testing data: 0.7586
Final F-score on the testing data: 0.9402
Best parameters:
{'kernel': 'poly', 'C': 0.001}
For classifier SVC, ROC score is 0.500000
```



```
For classifier KNeighborsClassifier:
Unoptimized model
------
Accuracy score on testing data: 0.6466
F-score on testing data: 0.7631

Optimized Model
------
Final accuracy score on the testing data: 0.6552
Final F-score on the testing data: 0.7960
Best parameters:
{'n_neighbors': 15, 'weights': 'uniform'}
For classifier KNeighborsClassifier, ROC score is 0.492695
```



Justification

Despite the ambiguous results, 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). As discussed before, 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. As discussed before, 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.

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 ('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 GridSearchCV. 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. However, it is believed that these algorithms will improve once they are presented with a much larger dataset.

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, we can save some training time at the cost of accuracy by extracting the most influential features using features_importance_ attribute of the RandomForestClassifier and training the model only on those features. This is implemented in the final portion of the project, and an F-beta score of 0.80 was obtained on predicting with 5 most influential features. 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.