

# Digital Medicine (OE)

## Case Study on Effects of Excessive Alcohol on Liver

Name: Aman Singh Thakur

Registration Number: 150905138

# INDEX

|    |   |    |
|----|---|----|
| 1. | <b>Problem Statement</b>                                | 3  |
| 2. | <b>Theory</b>   | 3  |
| 3. | <b>Key Features</b>                                     | 3  |
| 4. | <b>Dataset</b>  | 4  |
| 5. | <b>Methodology</b>                                      | 6  |
|    | a) Data Pre-Processing                                  |    |
|    | b) Check if attributes are independent                  |    |
|    | c) Realizing Trends of Features                         |    |
|    | d) Accuracy of Machine Learning Models                  |    |
|    | e) Use of Receiver Operating Characteristic Curve (ROC) |    |
|    | f) User Friendly Console based application              |    |
|    | <b>g) Liver Care Operations: Web Based Application</b>  |    |
| 6. | <b>Code</b>   | 22 |
| 7. | <b>Conclusion</b>                                       | 27 |
| 8. | <b>References</b>                                       | 28 |

# Problem Statement

Building a Python based Machine Learning application that takes in account blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption and compare with number of alcoholic beverages consumed per day.

## Theory

In the age of free access to cheap alcohol to anyone above 21 and lenient rules for younger children consuming alcohol, we are coming across cases of liver damage, liver failure or other liver disorders among the masses at very young age. It's the need of the hour to address people what liver disorders excessive consumption of alcohol can lead to as well as how much alcohol consumption is excessive and can lead to serious repercussions.

## Key Features

1. Working with UCI Machine Learning Repository datasets.
2. Complex medical problem statement solved using machine learning.
3. 4 supervised learning techniques used with extensive mathematical and graph comparison to yield the best solution.
4. **Average Accuracy = 78.611%**
5. User friendly console based application for users to predict whether they might be suffering from liver disease or not.
6. "LIVER CARE OPERATIONS" a web based application that will help normal people with required blood test results self-diagnose themselves with Liver diseases.

# Dataset

Liver disorders dataset is one of the most popular datasets on UCI Machine Learning Repository. It is created by BUPA Medical Research Ltd. by donor Richard S. Forsyth. Here are all the details:

Table 1: Liver Disorder Dataset

|                                  |                                |
|----------------------------------|--------------------------------|
| <b>Dataset Characteristics</b>   | Multivariate                   |
| <b>Attribute Characteristics</b> | Categorical, Integer, Real     |
| <b>Number of Instances</b>       | 345                            |
| <b>Number of Attributes</b>      | 7                              |
| <b>Area</b>                      | Life Science, Machine Learning |
| <b>Data Donated</b>              | 05/15/1990                     |

## Attribute Information

There are 7 attributes. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the bupa.data file constitutes the record of a single male individual. They are broadly defined as follows:

Table 2: Attribute Information

| No | Name    | Description   |
|----|---------|---|
| 1  | McV     | Mean corpuscular Volume   |
| 2  | Alkphos | Alkaline phosphatase  |
| 3  | Sgpt    | Alamine aminotransferase  |
| 4  | Sgot    | Aspartate aminotransferase  |
| 5  | Gammagt | Gamma-glutamyl transpeptidase                                     |
| 6  | Drinks  | No of half pints equivalents of alcoholic beverages drunk per day |

|   |          |                                      |
|---|----------|--------------------------------------|
| 7 | Selector | Field used to split data into 2 sets |
|---|----------|--------------------------------------|

Detailed description of attributes is as follows

Table 3: Detailed Description of Blood Tests

|   |   |
|---|---|
| <b>Mean corpuscular Volume (mcv)</b>          | A MCV blood test measures the size of your red blood cells. If blood cells are too small or too large, it may indicate a blood disorder.  |
| <b>Alkaline phosphatase (Alkphos)</b>         | A ALP test measures the amount of ALP in your blood. ALP is an enzyme found throughout the body, but it is mostly found in the liver, bones, kidneys, and digestive system.                         |
| <b>Alanine aminotransferase(Sgpt)</b>         | An alanine aminotransferase (ALT) test measures the level of ALT in your blood. ALT is an enzyme made by cells in your liver. High levels of ALT help in diagnose liver problems.                   |
| <b>Aspartate aminotransferase(Sgot)</b>       | The AST test measures the amount of AST in your blood. AST levels increase when there's damage to the tissues and cells where the enzyme is found.  |
| <b>Gamma-glutamyl transpeptidase(Gammagt)</b> | The gamma-glutamyl transpeptidase (GGT) test measures the amount of the enzyme GGT in your blood. GGT is concentrated in the liver and GGT blood levels are usually high when the liver is damaged. |

# Methodology

We are predicting whether drinking affects liver diseases (0) or not (1), so we'll be using machine learning algorithms to predict this value. The language used to code the problem statement is Python 2 and Python libraries used for machine learning are sklearn, matplotlib, pandas and numpy. The Step-wise breakdown of solution is given as follows:

## 1. Data Pre-Processing:

First, we will read “bupa.data” dataset file into a data frame. A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Since, the 7<sup>th</sup> attribute is a selector attribute we drop it in data pre processing and put labels on each column. We also attach an id to each row as part of data pre-processing. We do this by calling “data\_preprocessing ()” function in the code.

**Figure 1: Observation of Dataset as Data Frame**

```

Project — bash — 127x71
[Amans-MacBook-Pro: amansinghthakur$ cd Artificial\ Intelligence\Digital\ Medicine\Project/
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
   0   1   2   3   4   5   6   id
0   85  92  45  27  31  0.0  0   1
1   85  64  59  32  23  0.0  1   2
2   86  54  33  16  54  0.0  1   3
3   91  78  34  24  36  0.0  1   4
4   87  70  12  28  10  0.0  1   5
5   98  55  13  17  17  0.0  1   6
6   88  62  20  17  9   0.5  0   7
7   88  67  21  11  11  0.5  0   8
8   92  54  22  20  7   0.5  0   9
9   90  68  25  19  5   0.5  0   10
10  89  52  13  24  15  0.5  0   11
11  82  62  17  17  15  0.5  0   12
12  96  64  61  32  13  0.5  0   13
13  86  77  25  19  18  0.5  0   14
14  96  67  29  20  11  0.5  0   15
15  91  78  20  31  18  0.5  0   16
16  89  67  23  16  10  0.5  0   17
17  89  79  17  17  16  0.5  0   18
18  91  107 20  20  56  0.5  0   19
19  94  116 11  33  11  0.5  0   20
20  92  59  35  13  19  0.5  0   21
21  93  23  35  20  20  0.5  0   22
22  96  60  23  27  5   0.5  0   23
23  96  68  18  19  19  0.5  0   24
24  84  86  47  33  97  0.5  0   25
25  92  70  24  13  26  0.5  0   26
26  98  47  28  15  18  0.5  0   27
27  88  66  20  21  10  0.5  0   28
28  91  102 17  13  19  0.5  0   29
29  87  41  31  19  16  0.5  0   30
...
315 99  86  58  42  283 6.0  0   316
316 98  66  103 57  114 6.0  0   317
317 92  80  10  26  20  6.0  0   318
318 96  74  27  25  43  6.0  1   319
319 95  93  21  27  47  6.0  1   320
320 86  109 16  22  28  6.0  1   321
321 91  46  30  24  39  7.0  1   322
322 102 82  34  78  283 7.0  1   323
323 85  50  12  18  14  7.0  1   324
324 91  57  33  23  12  8.0  0   325
325 91  52  76  32  24  8.0  0   326
326 93  70  46  30  33  8.0  0   327
327 87  55  36  19  25  8.0  0   328
328 98  123 28  24  31  8.0  0   329
329 82  55  18  23  44  8.0  1   330
330 95  73  20  25  225 8.0  1   331
331 97  80  17  20  53  8.0  1   332
332 100 83  25  24  28  8.0  1   333
333 88  91  56  35  126 9.0  1   334
334 91  138 45  21  48  10.0 0   335
335 92  41  37  22  37  10.0 0   336
336 86  123 20  25  23  10.0 1   337
337 91  93  35  34  37  10.0 1   338
338 87  87  15  23  11  10.0 1   339
339 87  56  52  43  55  10.0 1   340
340 99  75  26  24  41  12.0 0   341
341 96  69  53  43  283 12.0 1   342
342 98  77  55  35  89  15.0 0   343
343 91  68  27  26  14  16.0 0   344
344 98  99  57  45  65  20.0 0   345
[345 rows x 8 columns]
End of program

```

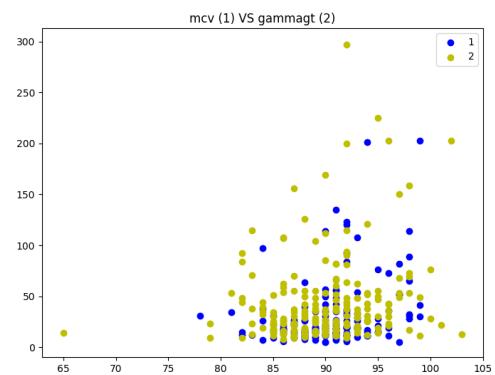
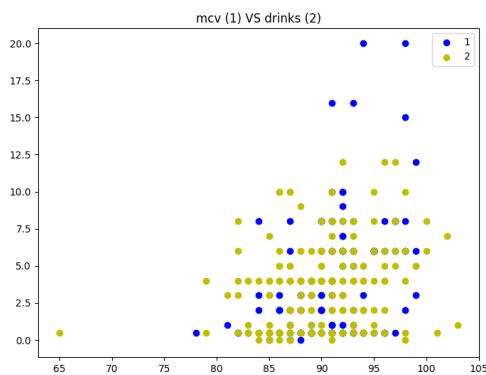
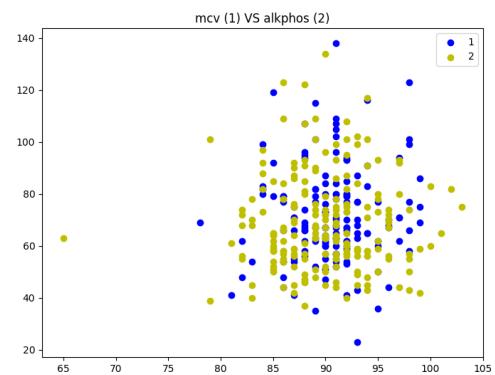
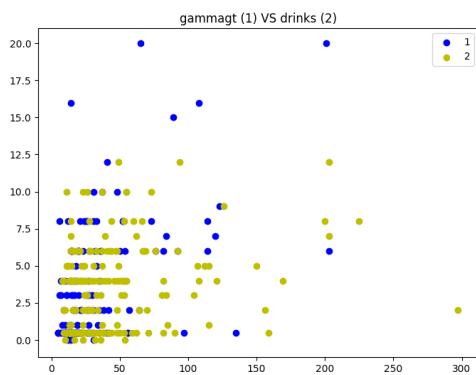
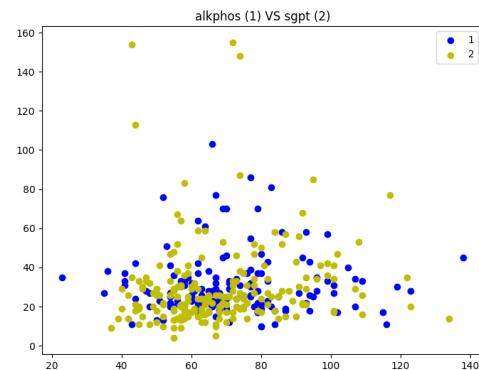
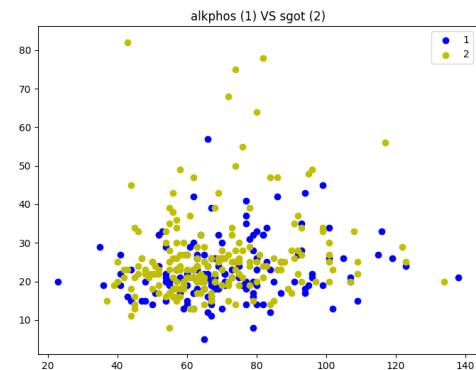
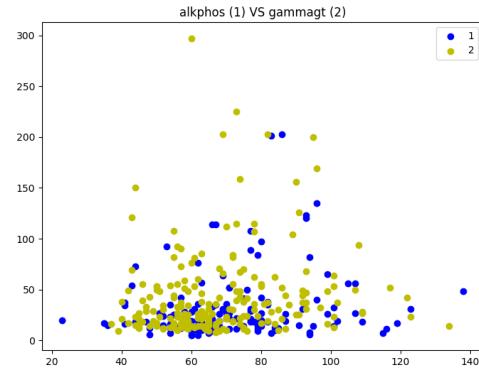
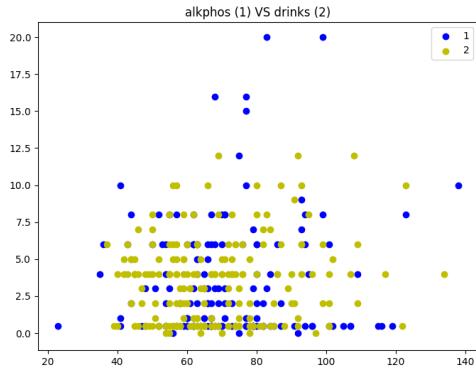
## 2. Check if attributes are independent:

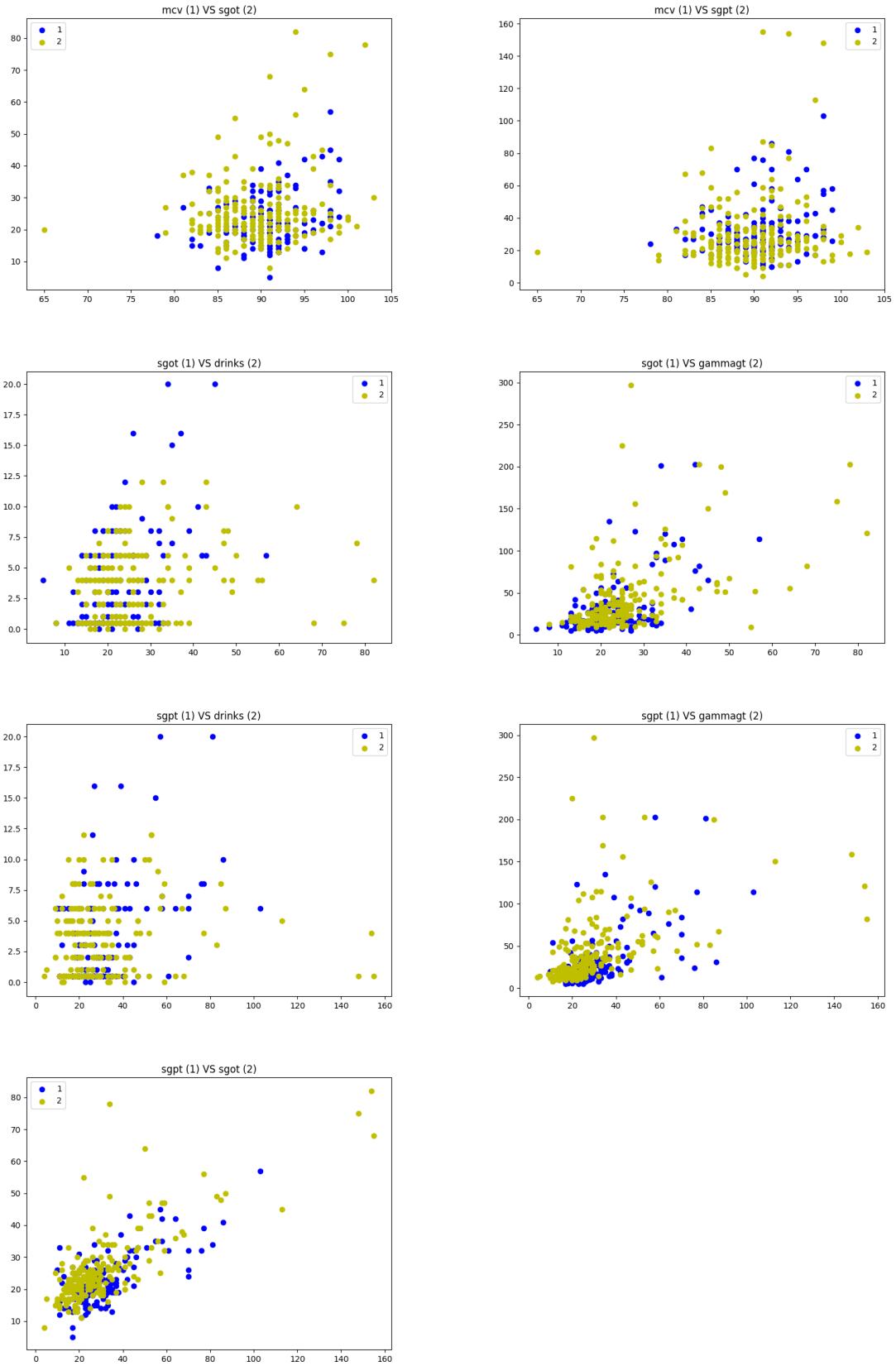
It's very crucial to realize that all features selected for prediction must be IID (Independent and Identically Distributed).

For e.g. Features such as height of a man and weight of a man might not be always independent but features such as height and color of a man will always be independent.

We draw Scatter plots for all possible features compared with each other. All the graphs are shown below. We use GRAPH\_CLASS () to draw all graphs in code.

Figure 2: Scatter Plots of all features with each other to determine whether they are independent or not





### 3. Realizing Trends of Features:

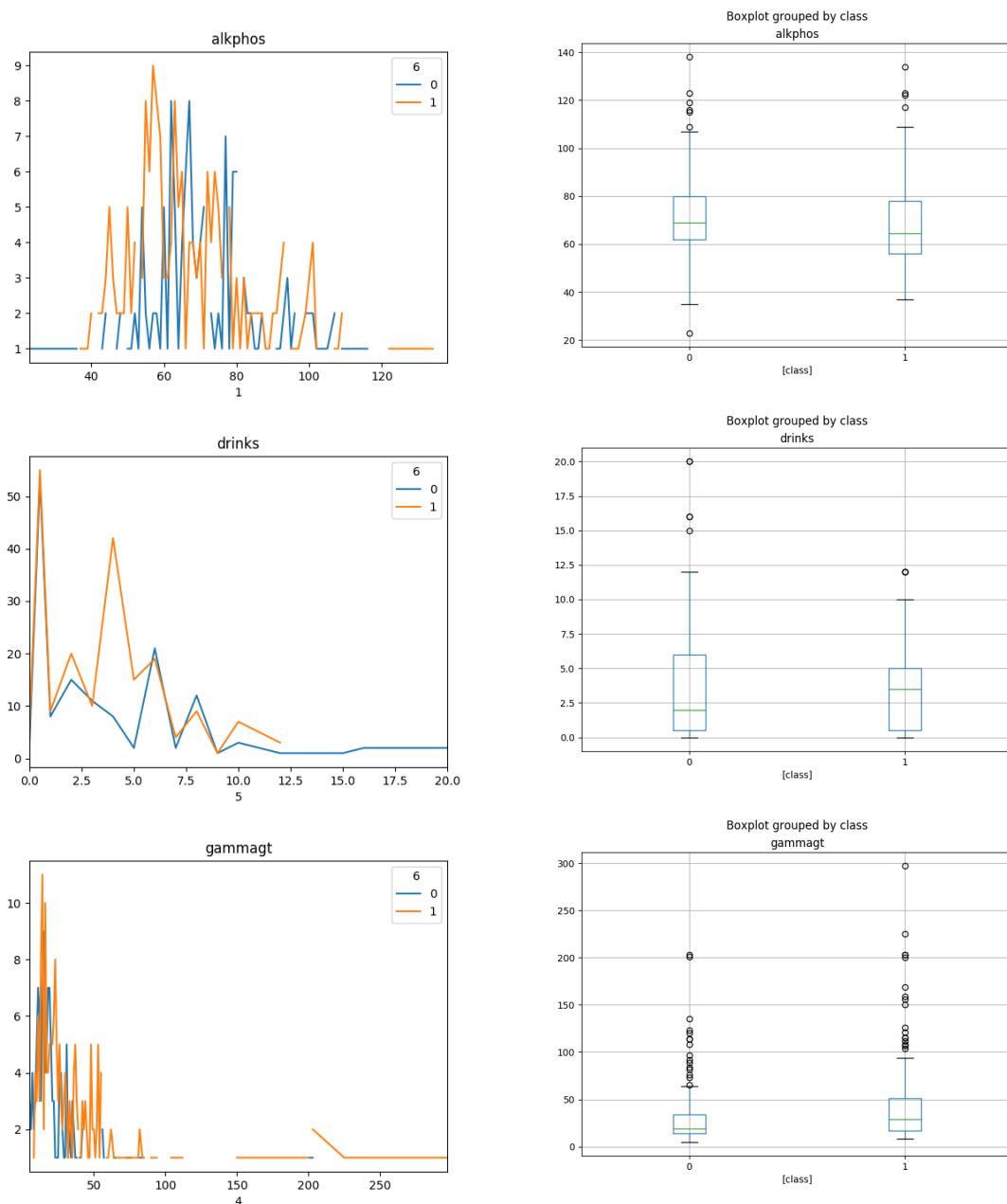
As observed from Fig 2, We can see that all the features are independent of each other and form an independent and identically distributed space.

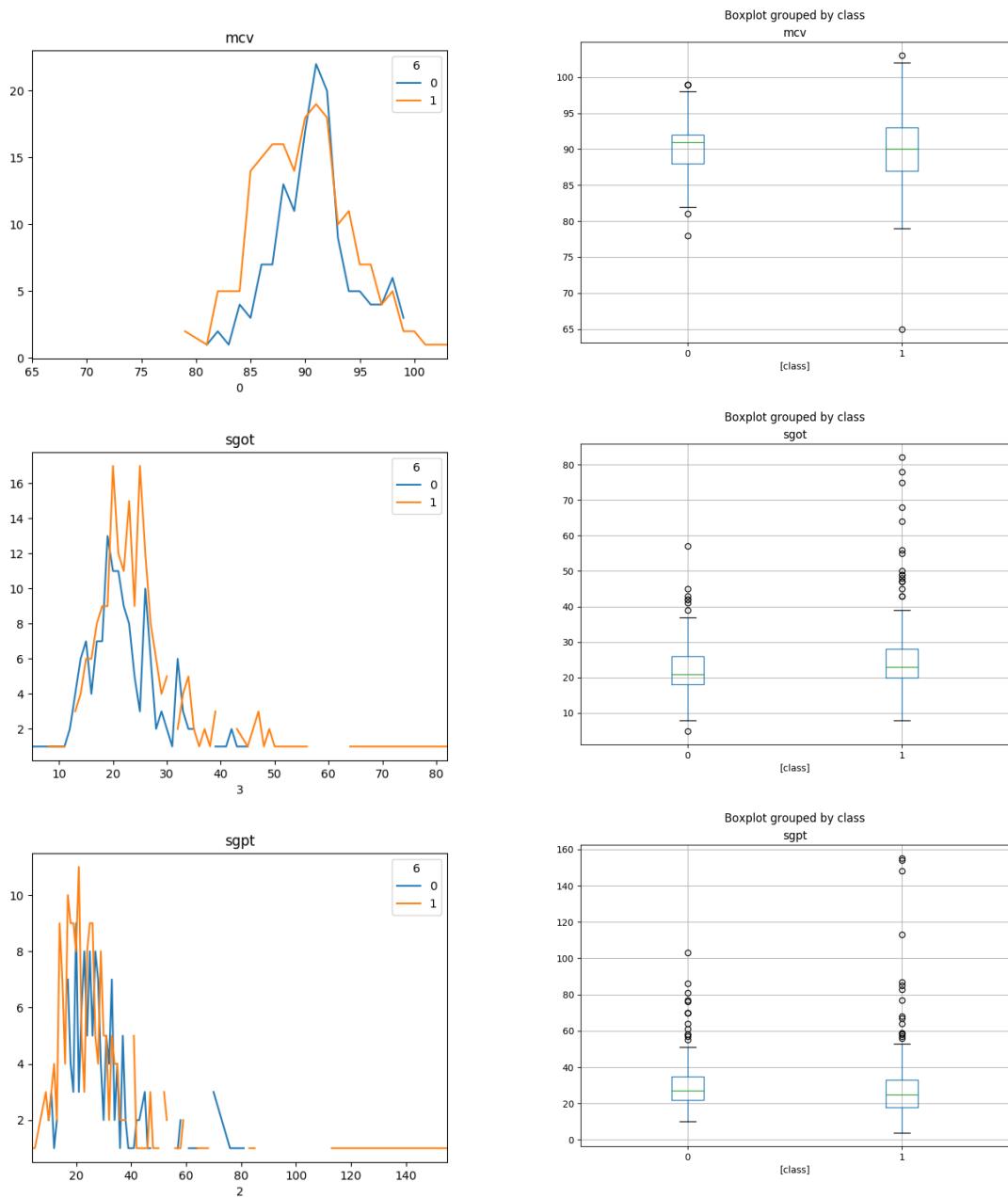
The most important question encountered while building a machine learning application is the choice of the algorithm used. Different algorithms have different behavior and accuracy. It also largely depends on the datasets as well.

For e.g. Support Vector Machines (SVM) comes in handy for small datasets while sometimes a simple linear regression program beats other complex neural networks.

Since This is such an integral part of this program, we will draw the graph and box plots of each class to realize their concentration, distribution, behavior, etc. All the graphs are plotted using CLASS\_PLOT () and all the box plots are plotted using BOX\_PLOT (). All the graphs (class wise) are shown as below:

Figure 3: Observation of Graphs(Left) and Box plots(Right) of different classes with titles as mentioned.





The observations from graph analysis is:

- It can be observed that all the features are sparsely populated throughout the feature space. Hence, we need machine learning algorithms that take in account the entire range.
- At the same time, every features attains a non-smooth bell curve. This bell curve has a maximum value where the data is most concentrated.
- Also, most of these test results when high results in positive diagnose of liver diseases and disorders.
- All the features are categorical i.e. binary in nature (0,1).
- Since, numeric input and output are given, we need a supervised machine learning technique.

Taking in account their distribution and their behaviour, the types of machine learning algorithm we are going to use are:

### a) Logistic Regression:

Logistic regression or logit regression is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

$$\text{logit}(p) = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + \dots + b_k X_k$$

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation. Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data. It's a supervised machine learning algorithm. It's the only ML algorithm used with linear model. Logistic regression is implemented by importing `sklearn.linear_model` package and using `LogisticRegression` class.

### b) Gradient Boosting Classifier:

Gradient descent is a ML technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The math of this ML algorithm is highly complicated and is skipped to keep the report shorter and simpler. It's a supervised learning technique. It's implemented by importing `sklearn.ensemble` package and using `GradientBoostingClassifier` class.

### c) Random Forest Classifier:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The math of this ML algorithm is beyond the scope of this project.

It's a supervised classification problem. It's implemented by importing sklearn.ensemble package and using RandomForestClassifier class.

#### d) K-Nearest Neighbor Classifier:

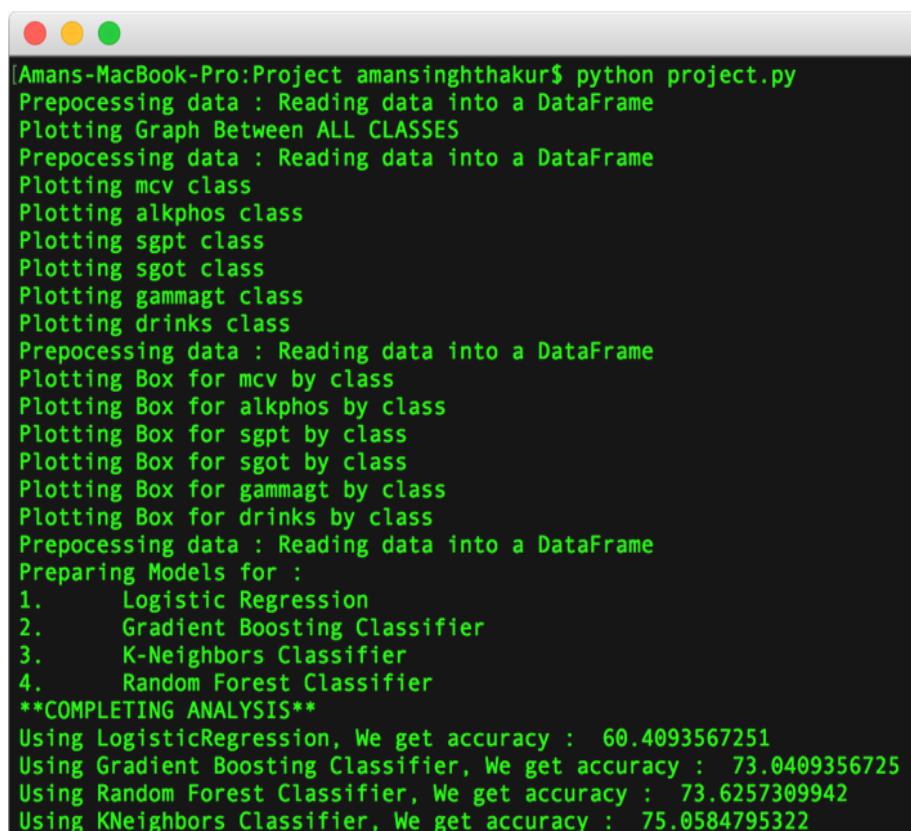
KNN is a non-parametric, supervised lazy learning algorithm. KNN can be used for classification—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest neighbours. When used for regression the output is the value for the object (predicts continuous values). This value is the average (or median) of the values of its k nearest neighbours. It's implemented by importing sklearn.ensemble package and using KNeighborsClassifier class.

### 4. Accuracy of Machine Learning Models:

Any Machine learning model is linked to it's accuracy. More accurate the model, the more appropriate it is for the dataset and the better results we will get. We use the function ROC () to prepare the models and train it.

Note: Here the 7th field selector is used for training and splitting the data.

Figure 4: Different Accuracies Observed for all Algorithms every time.



```
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
Plotting Graph Between ALL CLASSES
Preprocessing data : Reading data into a DataFrame
Plotting mcv class
Plotting alkphos class
Plotting sgpt class
Plotting sgot class
Plotting gammagt class
Plotting drinks class
Preprocessing data : Reading data into a DataFrame
Plotting Box for mcv by class
Plotting Box for alkphos by class
Plotting Box for sgpt by class
Plotting Box for sgot by class
Plotting Box for gammagt by class
Plotting Box for drinks by class
Preprocessing data : Reading data into a DataFrame
Preparing Models for :
1.      Logistic Regression
2.      Gradient Boosting Classifier
3.      K-Neighbors Classifier
4.      Random Forest Classifier
**COMPLETING ANALYSIS**
Using LogisticRegression, We get accuracy : 60.4093567251
Using Gradient Boosting Classifier, We get accuracy : 73.0409356725
Using Random Forest Classifier, We get accuracy : 73.6257309942
Using KNeighbors Classifier, We get accuracy : 75.0584795322
```

```
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
Plotting Graph Between ALL CLASSES
Preprocessing data : Reading data into a DataFrame
Plotting mcv class
Plotting alkphos class
Plotting sgpt class
Plotting sgot class
Plotting gammagt class
Plotting drinks class
Preprocessing data : Reading data into a DataFrame
Plotting Box for mcv by class
Plotting Box for alkphos by class
Plotting Box for sgpt by class
Plotting Box for sgot by class
Plotting Box for gammagt by class
Plotting Box for drinks by class
Preprocessing data : Reading data into a DataFrame
Preparing Models for :
1. Logistic Regression
2. Gradient Boosting Classifier
3. K-Neighbors Classifier
4. Random Forest Classifier
**COMPLETING ANALYSIS**
Using LogisticRegression, We get accuracy : 67.8378378378
Using Gradient Boosting Classifier, We get accuracy : 73.2972972973
Using Random Forest Classifier, We get accuracy : 78.2972972973
Using KNeighbors Classifier, We get accuracy : 70.2702702703
```

```
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
Plotting Graph Between ALL CLASSES
Preprocessing data : Reading data into a DataFrame
Plotting mcv class
Plotting alkphos class
Plotting sgpt class
Plotting sgot class
Plotting gammagt class
Plotting drinks class
Preprocessing data : Reading data into a DataFrame
Plotting Box for mcv by class
Plotting Box for alkphos by class
Plotting Box for sgpt by class
Plotting Box for sgot by class
Plotting Box for gammagt by class
Plotting Box for drinks by class
Preprocessing data : Reading data into a DataFrame
Preparing Models for :
1. Logistic Regression
2. Gradient Boosting Classifier
3. K-Neighbors Classifier
4. Random Forest Classifier
**COMPLETING ANALYSIS**
Using LogisticRegression, We get accuracy : 78.0769230769
Using Gradient Boosting Classifier, We get accuracy : 78.021978022
Using Random Forest Classifier, We get accuracy : 82.4725274725
Using KNeighbors Classifier, We get accuracy : 67.7197802198
```

Every time the model runs, the previous functions are called. In the end, we call ROC () which prepares models for all the machine learning algorithm. The highlight observations are:

- Different accuracies every time program run.

- Most of the times, Random Forest Classifier and Gradient Boosting yield highest accuracy.
- No Conclusive proof as to what is the best algorithm to use.

## **5. Use of Receiver operating characteristic curve (ROC):**

ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-out or probability of false alarm and can be calculated as  $(1 - \text{specificity})$ .

We modify the ROC () function to add the ROC functionality by importing python package `sklearn.metrics` and using `roc_curve` class. We add this functionality to all the models to get a graph for all models comparing their performances. To differentiate between this type of graph and the rest of the graphs, we create a new folder called `algorithm_analysis` and keep all the new graph in it and put the old graph in `graph_analysis`.

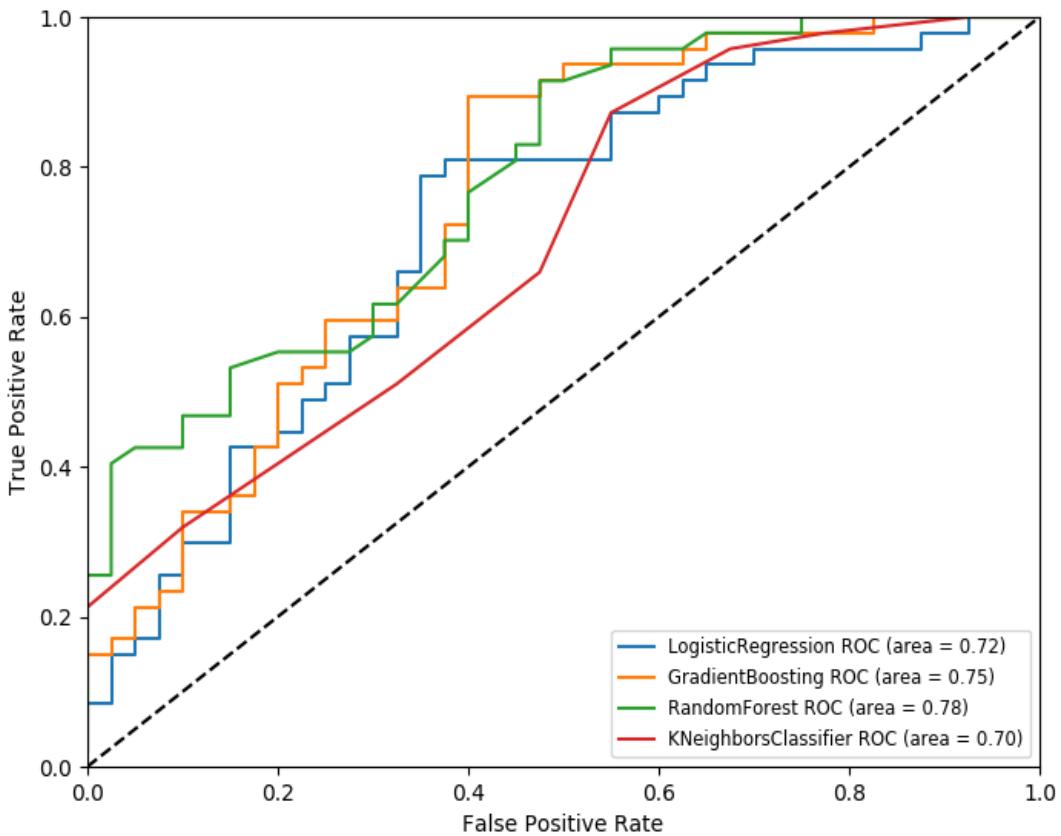


Figure 5.1: ROC curve for all ML algorithms

Looking at the ROC curve we realize that the two best algorithms that can be used are Random forest and Gradient Boosting as they yield to True positive state as fast as possible as compared to others. To compare these two, we create another function called COMPARE () in which re-process the data and plot the graph of ROC for Random forest vs Gradient descent.

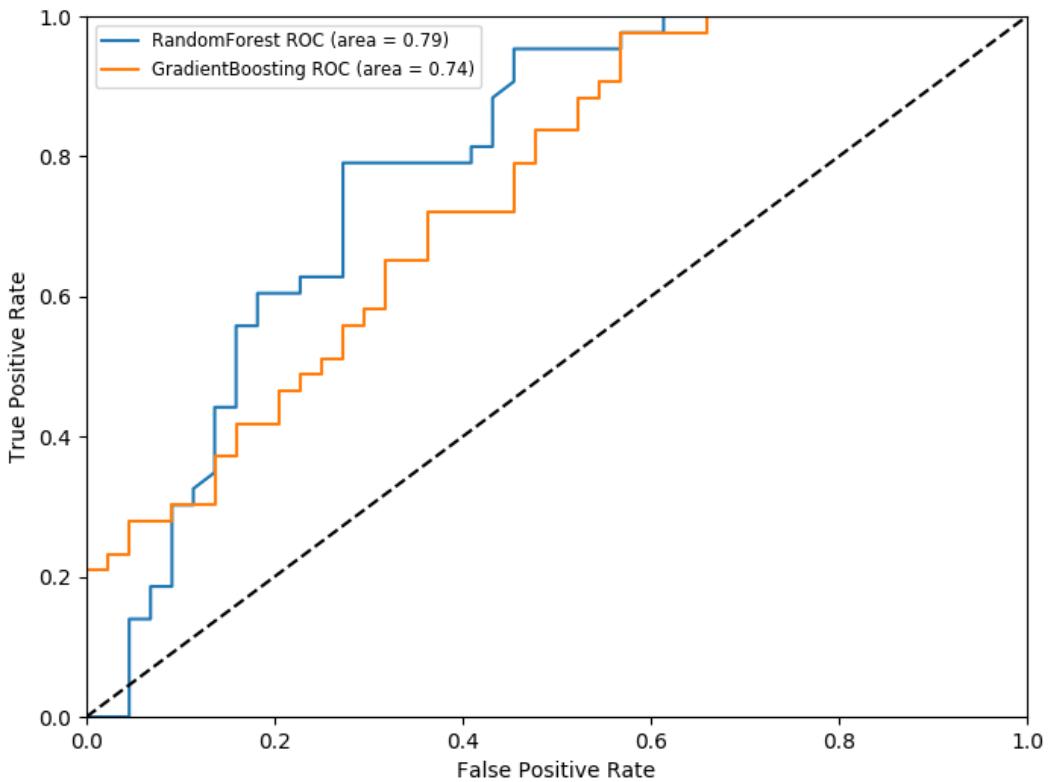


Figure 5.2: ROC curve for Random Forest Classifier Vs Gradient Boosting

Observing Fig. 6, we realize that the Random forest classifier is better than Gradient Boosting as it reaches True positive state before than Gradient Boosting. Also, From the graph it can be seen that Random Forest is almost always ahead of Gradient Boosting Classifier. Hence, we can conclude that Random Forest Classifier is best suited for this dataset.

Random Forest Classifier Average Accuracy = 78.611%

## 6. User Friendly Console Based Application:

Seeing the positive results from the trained model, we decide to make it a commercial model and open to users so that they can take these blood tests and use this software to enter data.

We create another function called TEST () which will only train the random forest classifier. After training the model, we will first look at all the resultant coefficients. Then, we will input the test results in the console as shown in below figure. After entering all 6 features, it's going to use RandomForestClassifier class built-in function ‘predict’ to predict whether it’s 1 or 0. 1 will mean that the patient is suffering from liver diseases or disorders due to excessive drinking. Hence, the software would

prompt the output to consult a Hepatologist in hospital. If the output is 0, the patient is safe and need not worry about any diseases.

Let's consider 2 situations:

1. Person who takes 0.5 half-pints daily and has normal test results

```
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
Plotting Graph Between ALL CLASSES
Preprocessing data : Reading data into a DataFrame
Plotting mcv class
Plotting alkphos class
Plotting sgpt class
Plotting sgot class
Plotting gammagt class
Plotting drinks class
Preprocessing data : Reading data into a DataFrame
Plotting Box for mcv by class
Plotting Box for alkphos by class
Plotting Box for sgpt by class
Plotting Box for sgot by class
Plotting Box for gammagt by class
Plotting Box for drinks by class

Preprocessing data : Reading data into a DataFrame
Preparing Models for :
1. Logistic Regression
2. Gradient Boosting Classifier
3. K-Neighbors Classifier
4. Random Forest Classifier

**COMPLETING ANALYSIS**

Using LogisticRegression, We get accuracy : 67.0810810811
Using Gradient Boosting Classifier, We get accuracy : 70.4324324324
Using Random Forest Classifier, We get accuracy : 77.4864864865
Using KNeighbors Classifier, We get accuracy : 64.5135135135
Preprocessing data : Reading data into a DataFrame

Plotting ROC for Random Forest vs Gradient Descent
Preprocessing data : Reading data into a DataFrame

Inside the TEST function, Training Random Forest Classifier
All the coefficients after training are [ 0.13493546  0.17997885  0.20822725  0.15106742  0.21401909  0.11177193]

Please Input the following Test Results :
mcv mean corpuscular volume : 80
alkphos alkaline phosphatase : 90
sgpt alamine aminotransferase: 40
sgot aspartate aminotransferase: 20
gammagt gamma-glutamyl transpeptidase : 10
number of half-pint equivalents of alcoholic beverages drunk per day : 0.5

You are not suffering from any Liver disease or disorder. Predicted with an accuracy of 74.0740740741
End of program
Amans-MacBook-Pro:Project amansinghthakur$ ]
```

Figure 6.1: Full Program example 1

2. Person who takes 5 half-pints daily and has high values in test results.

```
[Amans-MacBook-Pro:Project amansinghthakur$ python project.py
Preprocessing data : Reading data into a DataFrame
Plotting Graph Between ALL CLASSES
Preprocessing data : Reading data into a DataFrame
Plotting mcv class
Plotting alkphos class
Plotting sgpt class
Plotting sgot class
Plotting gammagt class
Plotting drinks class
Preprocessing data : Reading data into a DataFrame
Plotting Box for mcv by class
Plotting Box for alkphos by class
Plotting Box for sgpt by class
Plotting Box for sgot by class
Plotting Box for gammagt by class
Plotting Box for drinks by class

Preprocessing data : Reading data into a DataFrame
Preparing Models for :
1. Logistic Regression
2. Gradient Boosting Classifier
3. K-Neighbors Classifier
4. Random Forest Classifier

**COMPLETING ANALYSIS**

Using LogisticRegression, We get accuracy : 74.2919389978
Using Gradient Boosting Classifier, We get accuracy : 73.8562091503
Using Random Forest Classifier, We get accuracy : 74.7004357298
Using KNeighbors Classifier, We get accuracy : 67.6470588235
Preprocessing data : Reading data into a DataFrame

Plotting ROC for Random Forest vs Gradient Descent
Preprocessing data : Reading data into a DataFrame

Inside the TEST function, Training Random Forest Classifier
All the coefficients after training are [ 0.13265095  0.17427394  0.2041972   0.16133053  0.219622   0.10792538]

Please Input the following Test Results :
mcv mean corpuscular volume : 100
alkphos alkaline phosphatase : 110
sgpt alamine aminotransferase: 80
sgot aspartate aminotransferase: 60
gammagt gamma-glutamyl transpeptidase : 40
number of half-pint equivalents of alcoholic beverages drunk per day : 5

You are suffering from Liver disease or disorder because of your excessive drinking. Predicted with an accuracy of 72.6329787234
PLEASE CONSULT A HEPATOLOGIST !
End of program
Amans-MacBook-Pro:Project amansinghthakur$
```

Figure 6.2: Full program example 2

## 7. Liver Care Operations: The Web Based Application:

As part of commercializing this software, it's important to allow people with minimum resources to access it. So, we will be using a website called 'Liver Care Operations' where anybody can check results after taking tests.

The web based application is created using Python based web framework called flask in which we will integrate our algorithm and create the user interface even more simpler.

Later, after completing this demo application and entering more functionalities, we can host it and spread the links to all medical clinics and hospitals so that they can use it as a FOSS (Free Open Source Software).

Here are some glimpses of the software:

Figure 7.1: Home Page of Liver Care Operations website.



127.0.0.1

# Please Enter Your Medical Test Results

Enter Mean corpuscular Volume Test Results

Enter Alkaline Phosphatase Test Results

Fill out this field

Enter Alanine aminotransferase Test Results

Enter Aspartate aminotransferase Test Results

Enter Gamma-glutamyl transpeptidase Test Results

Enter the no of half pints equivalents of alcoholic beverages drunk per day

**Sign Up**

Figure 7.2: Taking Inputs with Validators in Website

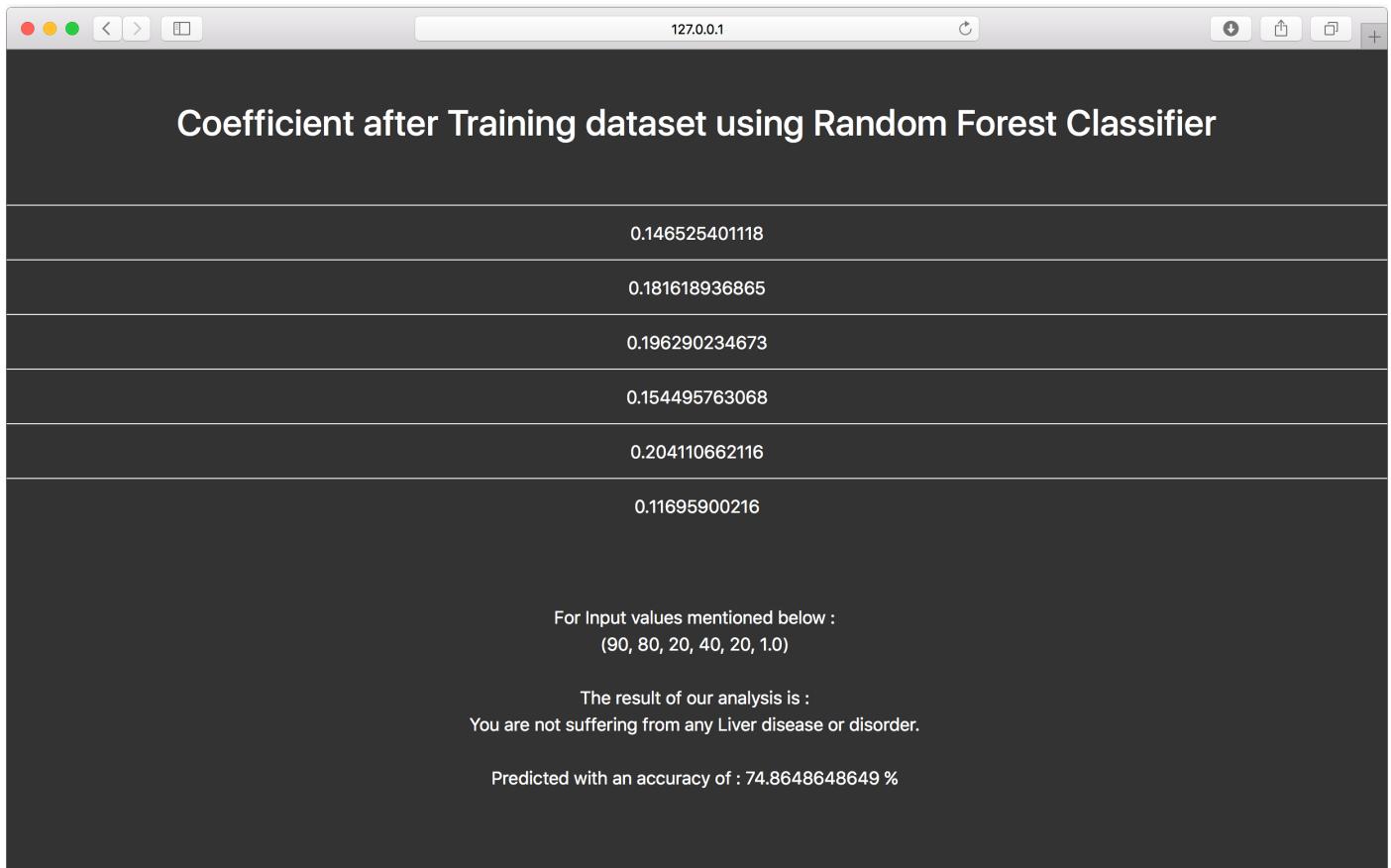


Figure 7.3: Results shown on final page of Liver Care Operations with coefficients with predicted accuracy.

# Code

```
***  
*****  
 Date Created: 10th April, 2018  
 Project Title: Case Study on Effects of Alcohol on Liver  
 Author: Aman Singh Thakur  
*****  
***  
  
##### IMPORT NECESSARY PYTHON LIBRARIES #####  
  
import numpy  
import pylab as plt  
  
from sklearn.model_selection import train_test_split  
  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
from sklearn.linear_model import LogisticRegression  
  
import time  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import roc_curve, auc  
from sklearn.metrics import accuracy_score, roc_auc_score  
  
from pandas import DataFrame, read_csv, concat  
import matplotlib.pyplot as plt  
import os  
  
***  
*****  
 START OF PROGRAM  
*****  
***  
  
##### FETCHING AND PREPROCESSING DATA #####  
  
def data_preprocessing():  
    print "Preprocessing data : Reading data into a DataFrame"  
    dataset = read_csv("./bupa.data", sep=',', header=None)  
    dataset['id'] = range(1, len(dataset)+1)  
    lab = LabelEncoder()  
    lab.fit(dataset[6].drop_duplicates())  
    dataset[6] = lab.transform(dataset[6])  
    class_names = list(lab.classes_)  
    return (dataset, class_names)  
  
##### Plot Graph Between ALL CLASSES #####  
  
def GRAPH_CLASS():
```

```

dataset, class_names = data_preprocessing()
print "Plotting Graph Between ALL CLASSES"
title = ['mcv', 'alkphos', 'sgpt', 'sgot', 'gammagt', 'drinks']
for i in range(0,6):
    for j in range(i,6):
        if i is not j :
            f = plot.figure(figsize=(8, 6))
            colors = ['b', 'y']
            for k in range(0, 2):
                plot.scatter(dataset[dataset[6] == k][i], dataset[dataset[6] == k][j], c=colors[k], label="%s" %
class_names[k])
            plot.legend()
            plot.title('%s (1) VS %s (2)' % (title[i],title[j]))
            f.savefig('./%s/%s_%s.png' % (directories[0],title[i],title[j]))
            plot.close()

```

##### Plot Graph for all classes #####

```

def CLASS_PLOT():
    dataset, class_names = data_preprocessing()
    title = ['mcv', 'alkphos', 'sgpt', 'sgot', 'gammagt', 'drinks']
    for k in range(0,6):
        print "Plotting %s class" % title[k]
        fig, axes = plot.subplots(ncols=1)
        e = dataset.pivot_table('id', [k], 6, 'count').plot(ax=axes, title='%s' % title[k])
        f = e.get_figure()
        f.savefig('./%s/%s.png' % (directories[0], title[k]))
        plot.close()

```

##### Plot BOX PLOTS for all classes #####

```

def BOX_PLOT():
    dataset, class_names = data_preprocessing()
    title = ['mcv', 'alkphos', 'sgpt', 'sgot', 'gammagt', 'drinks']
    dataset['class'] = dataset[6]
    for k in range(0, 6):
        print "Plotting Box for %s by class" % title[k]
        df = concat([dataset[k], dataset['class']], axis=1, keys=[title[k], 'class'])
        f = plot.figure(figsize=(8, 6))
        p = df.boxplot(by='class', ax = f.gca())
        f.savefig('./%s/%s_box.png' % (directories[0], title[k]))
        plot.close()

```

##### ANALYZING CLASSES AND PLOTTING ROC #####

```

def ROC():
    print "\n"
    dataset, class_names = data_preprocessing()
    target = dataset[6]
    train = dataset.drop(['id', 6], axis = 1)

    print "Preparing Models for :\n1.\tLogistic Regression\n2.\tGradient Boosting Classifier\n3.\tK-Neighbors
Classifier\n4.\tRandom Forest Classifier"

    model_rfc = RandomForestClassifier(n_estimators = 100, criterion = "entropy", n_jobs = -1)
    model_knc = KNeighborsClassifier(n_neighbors = 10, algorithm = 'brute')

```

```

model_gbc = GradientBoostingClassifier(n_estimators = 100)
model_lr = LogisticRegression(penalty='l1', tol=0.01)

ROCtrainTRN, ROCtestTRN, ROCtrainTRG, ROCtestTRG = train_test_split(train, target, test_size=0.25)

print "\n**COMPLETING ANALYSIS**\n"

ply.clf()
plot.figure(figsize=(8,6))

# LogisticRegression
probas = model_lr.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
print "Using LogisticRegression, We get accuracy : ",float(roc_acc)*100
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('LogisticRegression',roc_acc))

# GradientBoostingClassifier
probas = model_gbc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
print "Using Gradient Boosting Classifier, We get accuracy : ",float(roc_acc)*100
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('GradientBoosting',roc_acc))

# RandomForestClassifier
probas = model_rfc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
print "Using Random Forest Classifier, We get accuracy : ",float(roc_acc)*100
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('RandomForest',roc_acc))

# KNeighborsClassifier
probas = model_knc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
print "Using KNeighbors Classifier, We get accuracy : ",float(roc_acc)*100
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('KNeighborsClassifier',roc_acc))

##### Plotting ROC graph for all classes in algorithm_analysis folder #####
ply.plot([0, 1], [0, 1], 'k--')
ply.xlim([0.0, 1.0])
ply.ylim([0.0, 1.0])
ply.xlabel('False Positive Rate')
ply.ylabel('True Positive Rate')
ply.legend(loc=0, fontsize='small')
ply.savefig('./%s/ROC_ALL.png' % directories[1])

##### Comparing ROC between Random forest and Gradient Boosting #####
def COMPARE():
    dataset, class_names = data_preprocessing()

    print "\nPlotting ROC for Random Forest vs Gradient Descent"

```

```

target = dataset[6]
train = dataset.drop(['id', 6], axis = 1)

model_rfc = RandomForestClassifier(n_estimators = 625, criterion = "entropy", n_jobs = -1)
model_gbc = GradientBoostingClassifier(n_estimators = 625)

ROCtrainTRN, ROCtestTRN, ROCtrainTRG, ROCtestTRG = train_test_split(train, target, test_size=0.25)

ply.clf()
plot.figure(figsize=(8,6))

# RandomForestClassifier
probas = model_rfc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('RandomForest',roc_acc))

# GradientBoostingClassifier
probas = model_gbc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_acc = auc(fpr, tpr)
ply.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('GradientBoosting',roc_acc))

ply.plot([0, 1], [0, 1], 'k--')
ply.xlim([0.0, 1.0])
ply.ylim([0.0, 1.0])
ply.xlabel('False Positive Rate')
ply.ylabel('True Positive Rate')
ply.legend(loc=0, fontsize='small')
ply.savefig('./%s/compare.png' % directories[1])

#####
##### TEST FUNCTION TO MAKE IT USER FRIENDLY #####
#####

def TEST():
    dataset, class_names = data_preprocessing()

    print "\nInside the TEST function, Training Random Forest Classifier"

    target = dataset[6]
    train = dataset.drop(['id', 6], axis = 1)

    model_rfc = RandomForestClassifier(n_estimators = 625, criterion = "entropy", n_jobs = -1)

    ROCtrainTRN, ROCtestTRN, ROCtrainTRG, ROCtestTRG = train_test_split(train, target, test_size=0.25)
    probas = model_rfc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
    fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
    roc_acc = auc(fpr, tpr)

    print "All the coefficients after training are ",model_rfc.feature_importances_

    print "\nPlease Input the following Test Results :"

    test1 = int(raw_input("mcv mean corpuscular volume : "))
    test2 = int(raw_input("alkphos alkaline phosphotase : "))
    test3 = int(raw_input("sgpt alamine aminotransferase: "))
    test4 = int(raw_input("sgot aspartate aminotransferase: "))

```

```

test5 = int(raw_input("gammagt gamma-glutamyl transpeptidase : "))
test6 = float(raw_input("number of half-pint equivalents of alcoholic beverages drunk per day : "))

result = model_rfc.predict([[test1,test2,test3,test4,test5,test6]])

if result == 0:
    print "\nYou are not suffering from any Liver disease or disorder. Predicted with an accuracy of
",float(roc_acc)*100
else :
    print "\nYou are suffering from Liver disease or disorder because of your excessive drinking. Predicted with an
accuracy of ",float(roc_acc)*100
    print "PLEASE CONSULT A HEPATOLOGIST !"

```

##### CREATE DIRECTORIES FOR GRAPH AND BOX ANALYSIS & ALGORITHM ANALYSIS #####

```
directories = ['graph_analysis', 'algorithm_analysis']
```

```
for check in directories:
```

```
    if not os.path.exists(check):
        os.makedirs(check)
```

##### FUNCTION CALL #####

```
GRAPH_CLASS()
CLASS_PLOT()
BOX_PLOT()
ROC()
COMPARE()
TEST()
print "End of program"
```

\*\*\*

\*\*\*\*\*  
END OF PROGRAM  
\*\*\*\*\*

\*\*\*\*\*  
\*\*\*

# Conclusion

This whole experiment demonstrates how severe the effects of alcohol is on liver by using various blood tests. It also allows us to check whether we might be having a liver disease or disorder if we have all the required information.

According to “National Council on Alcoholism and Drug Dependence”, excessive alcohol use led to approximately 88,000 deaths and 2.5 million years of potential life lost (YPLL) each year in the United States from 2006 – 2010. Alcohol induced problems include:

- Alcohol use, both short- and long-term affect the brain.
- Alcoholism can contribute to severe, chronic diseases.
- Drinking alcohol can cause sickness the next day – hangovers.
- Drinking alcohol while pregnant can result in birth defects.
- You could be injured while drinking alcohol.
- Using alcohol can cause dependence.
- Drinking alcohol can make you gain weight.
- Drinking alcohol can kill you, quickly, not just slowly.

# References

1. <https://justbelievereccovery.com/reasons-not-to-drink-alcohol/>
2. <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
3. <https://archive.ics.uci.edu/ml/datasets/Liver+Disorders>
4. <https://data.england.nhs.uk/dataset/ccgois-1-8-emergency-admissions-for-alcohol-related-liver-disease>