Digital Medicine (OE)

Case Study on Effects of Excessive Alcohol on Liver

Name: Aman Singh Thakur

Registration Number: 150905138

INDEX

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.

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

Dataset Characteristics	Multivariate
Attribute Characteristics	Categorical, Integer, Real
Number of Instances	345
Number of Attributes	7
Area	Life Science, Machine Learning
Data Donated	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

Mean corpuscular Volume (mcv)	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.
Alkaline phosphatase (Alkphos)	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.
Alamine aminotransferase(Sgpt)	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.
Aspartate aminotransferase(Sgot)	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.
Gamma-glutamyl transpeptidase(Gammagt)	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.

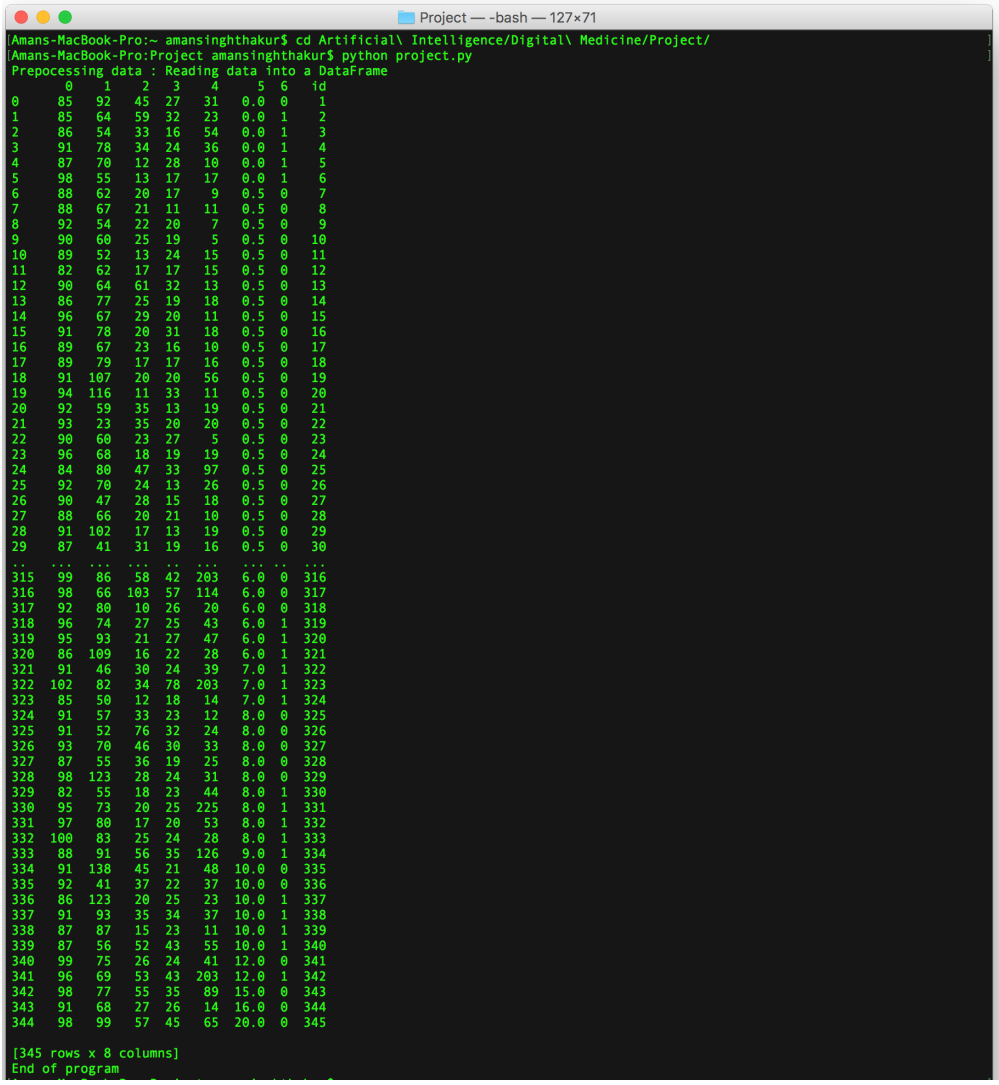
Methodology

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 7th 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.

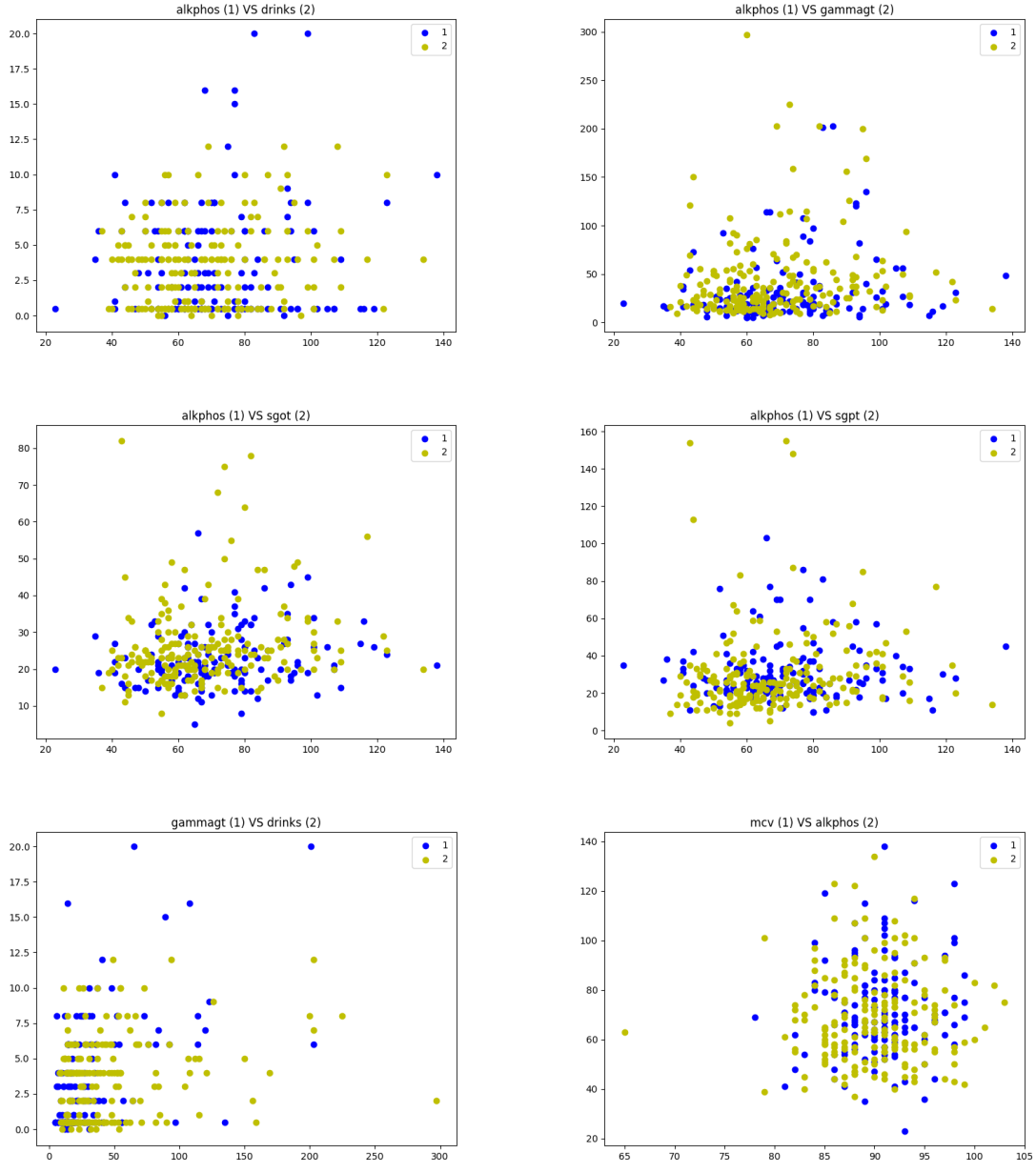
Figure 1: Observation of Dataset as Data Frame

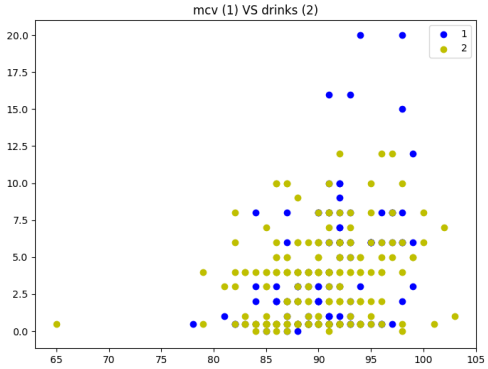


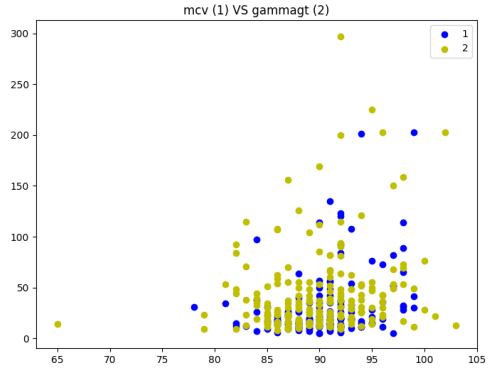
2. Check if attributes are independent:

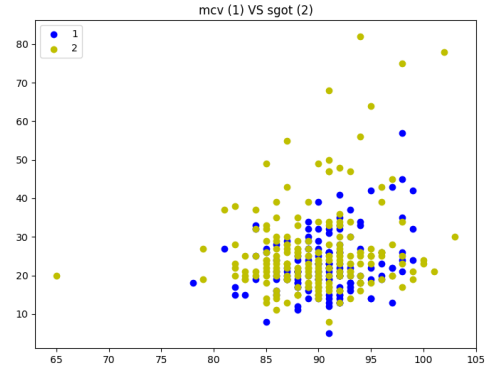
It's very crucial to realize that all features selected for prediction must be IID (Independent and Identically Distributed).
For e.g. Features such as height of a man and weight of a man might not be always independent but features such as height and color of a man will always be independent.
We draw 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.

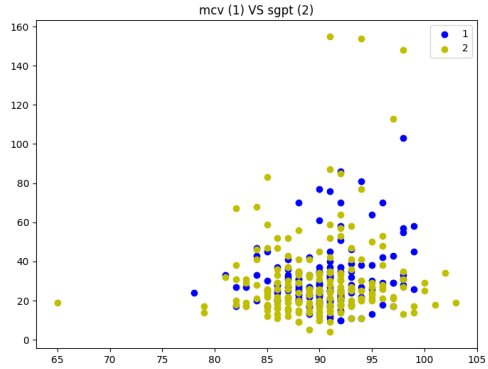
Figure 2: Scatter Plots of all features with each other to determine whether they are independent or not

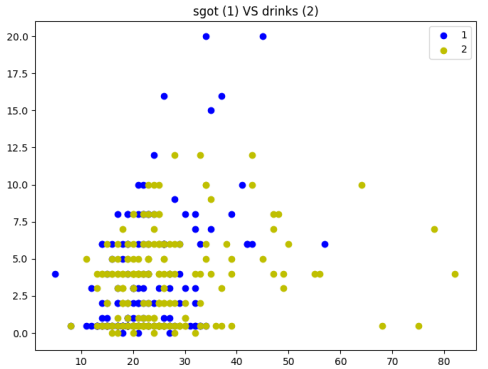


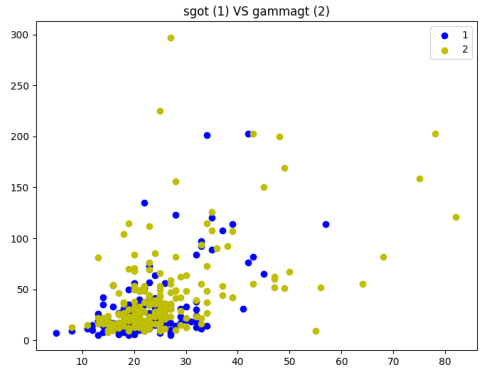

mcv (1) VS drinks (2)
1
2

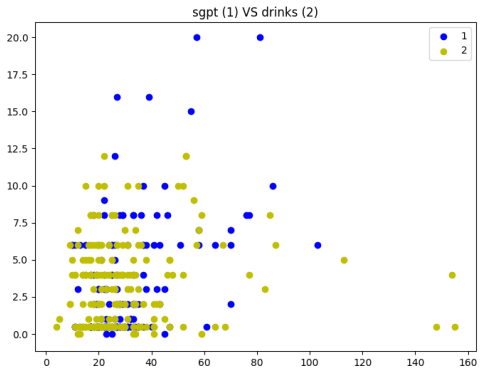

mcv (1) VS gammagt (2)
1
2

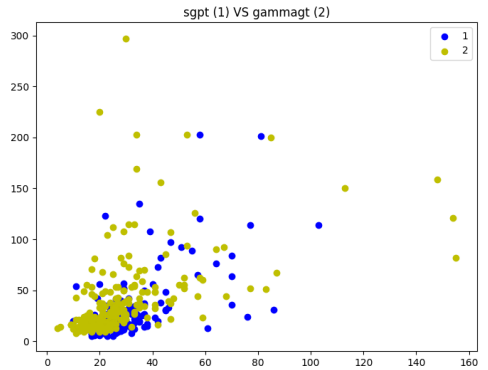

mcv (1) VS sgot (2)
1
2

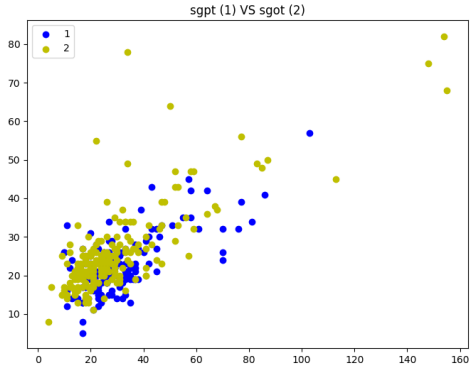

mcv (1) VS sgpt (2)
1
2


sgot (1) VS drinks (2)
1
2


sgot (1) VS gammagt (2)
1
2


sgpt (1) VS drinks (2)
1
2


sgpt (1) VS gammagt (2)
1
2



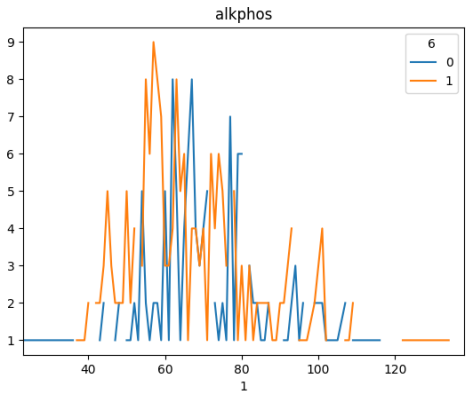
3. Realizing Trends of Features:

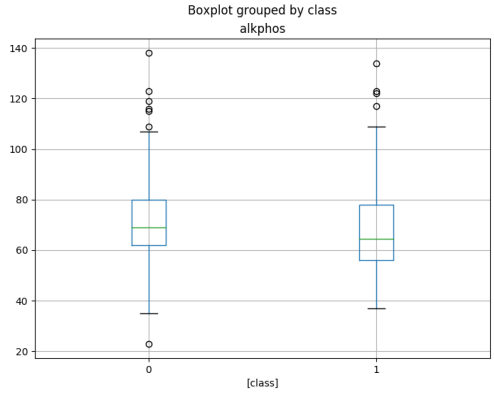
As observed from Fig 2, We can see that all the features are independent of each other and form an independent and identically distributed space.

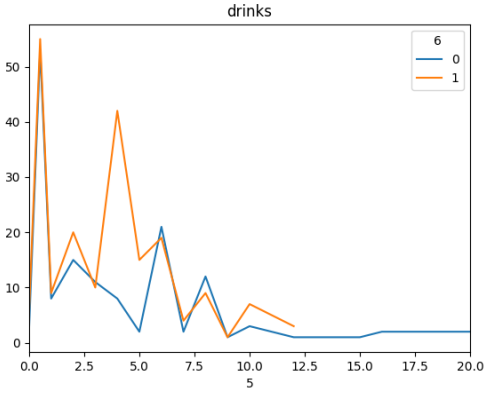
The most important question encountered while building a machine learning application is the choice of the algorithm used. Different algorithms have different behavior and accuracy. It also largely depends on the datasets as well.
For e.g. Support Vector Machines (SVM) comes in handy for small datasets while sometimes a simple linear regression program beats other complex neural networks.

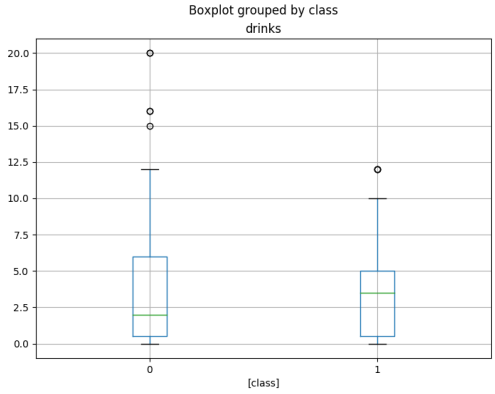
Since This is such an integral part of this program, we will draw the graph and box plots of each class to realize their concentration, distribution, behavior, etc. All the graphs are plotted using CLASS_PLOT () and all the box plots are plotted using BOX_PLOT (). All the graphs (class wise) are shown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

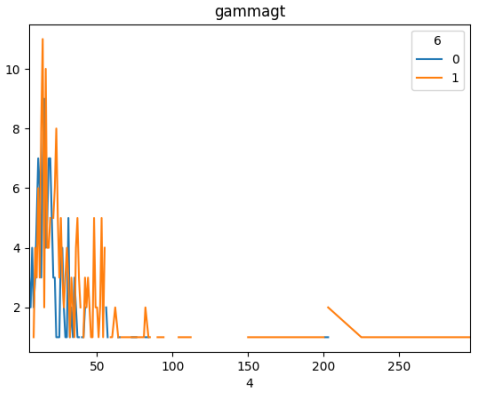
Figure 3: Observation of Graphs(Left) and Box plots(Right) of different classes with titles as mentioned.

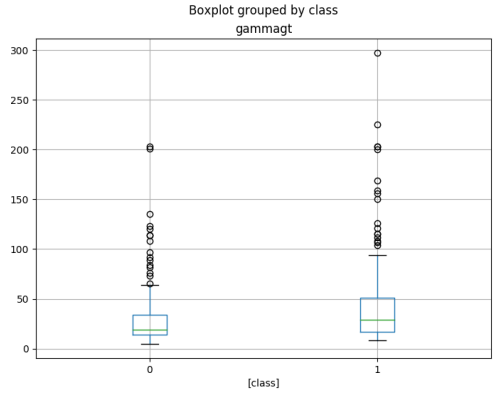


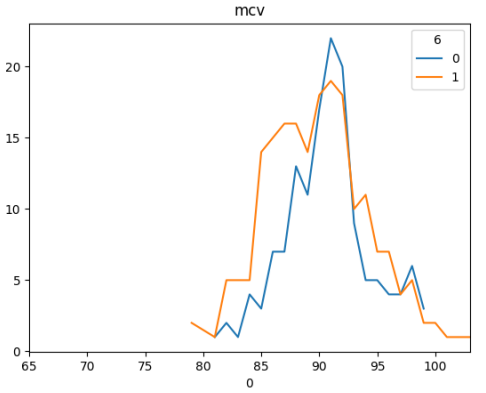


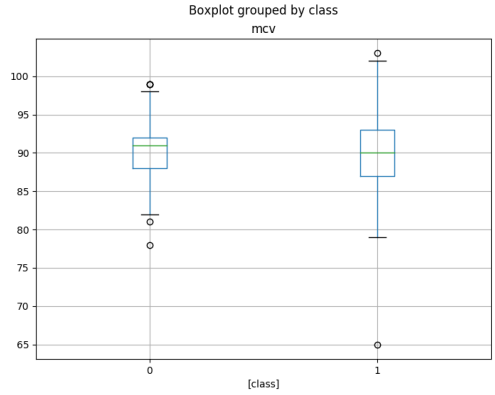

drinks
6
0
1
5

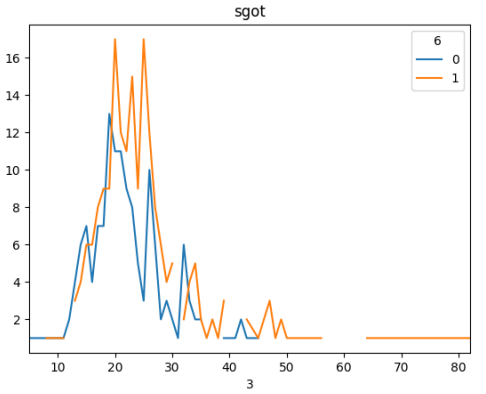

Boxplot grouped by class
drinks
[class]

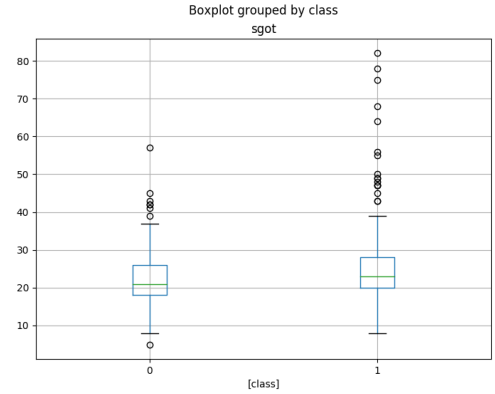

gammagt
6
0
1
4

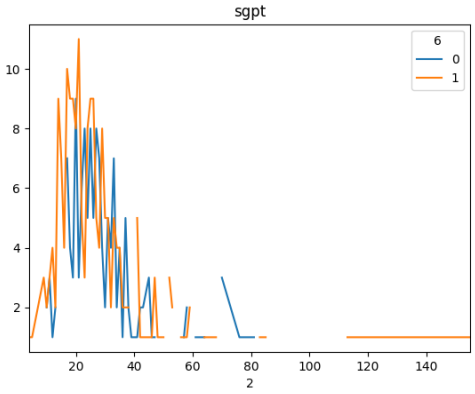

Boxplot grouped by class
gammagt
[class]

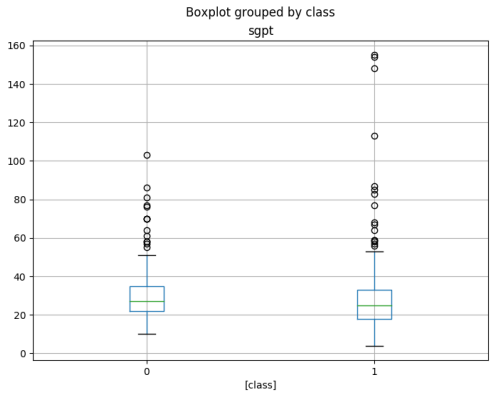

mcv
6
0
1
0


Boxplot grouped by class
mcv
[class]


sgot
6
0
1
3


Boxplot grouped by class
sgot
[class]





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).

$$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

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

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:

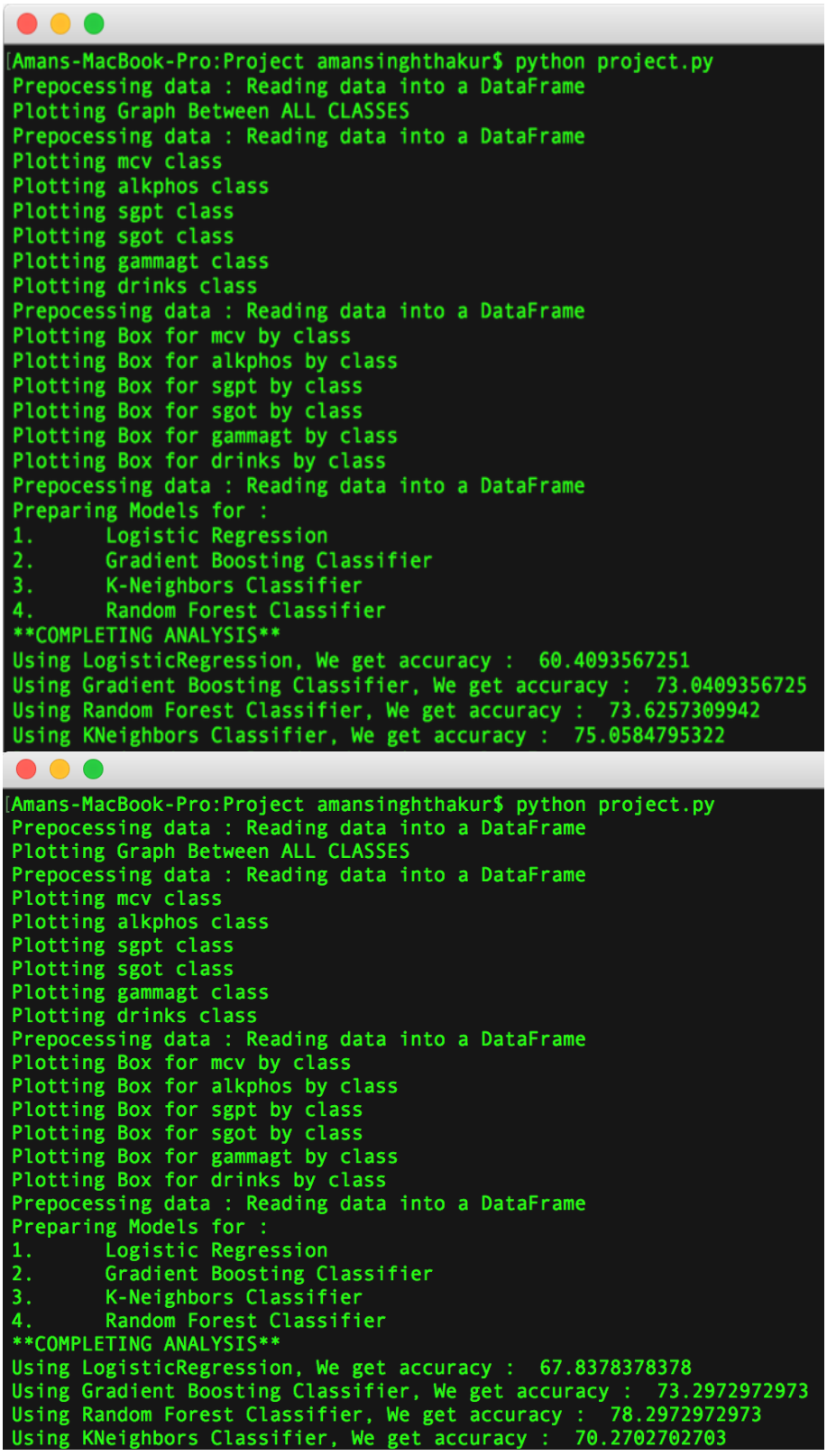
KNN is a non-parametric, supervised lazy learning algorithm. KNN can be used for classification—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its k nearest 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 KNearestNeighbourClassifier class.

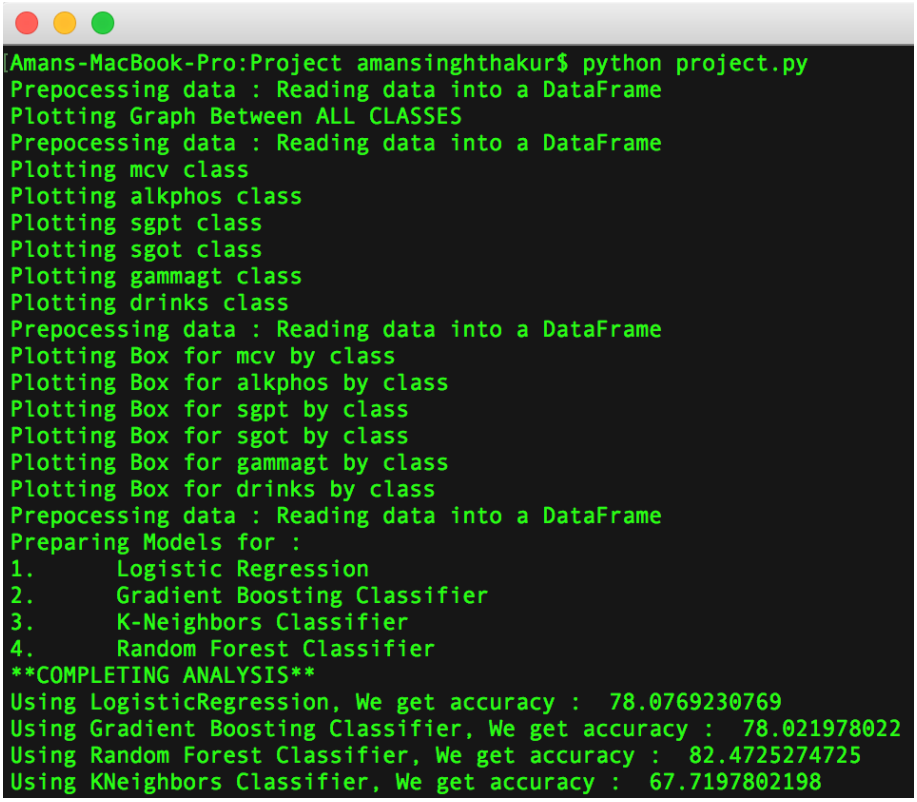
4. Accuracy of Machine Learning Models:

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.





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 – 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.

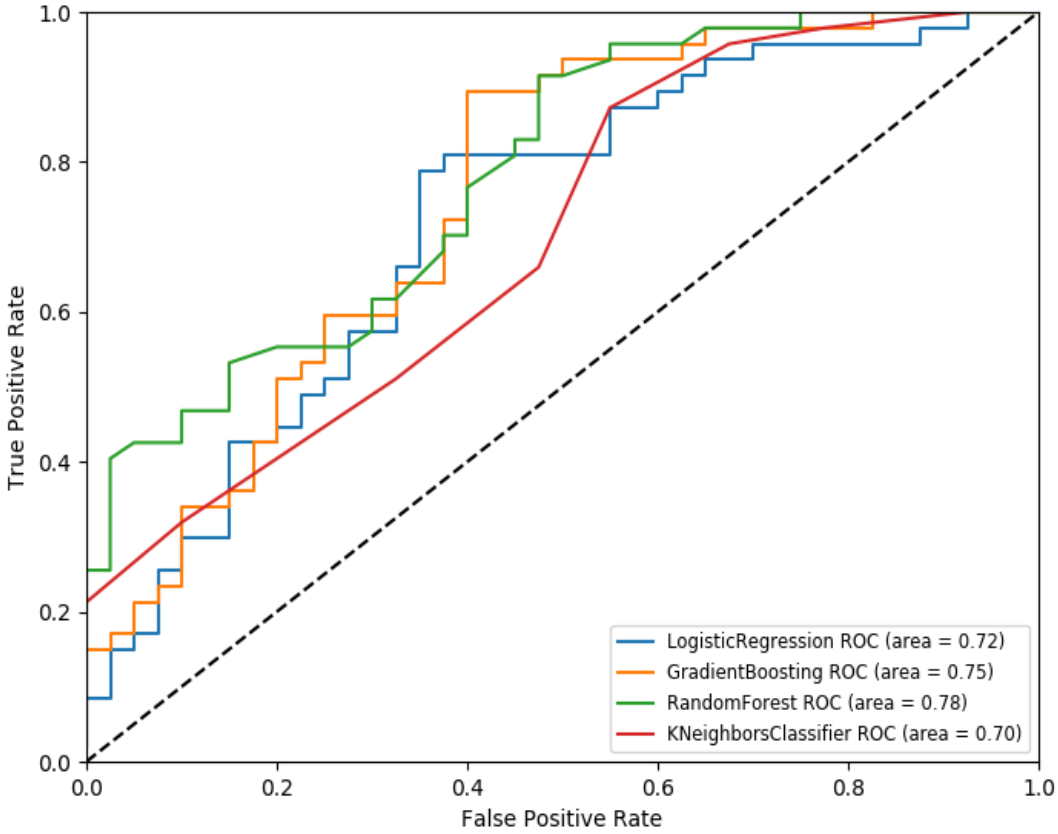


Figure 5.1: ROC curve for all ML algorithms

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.

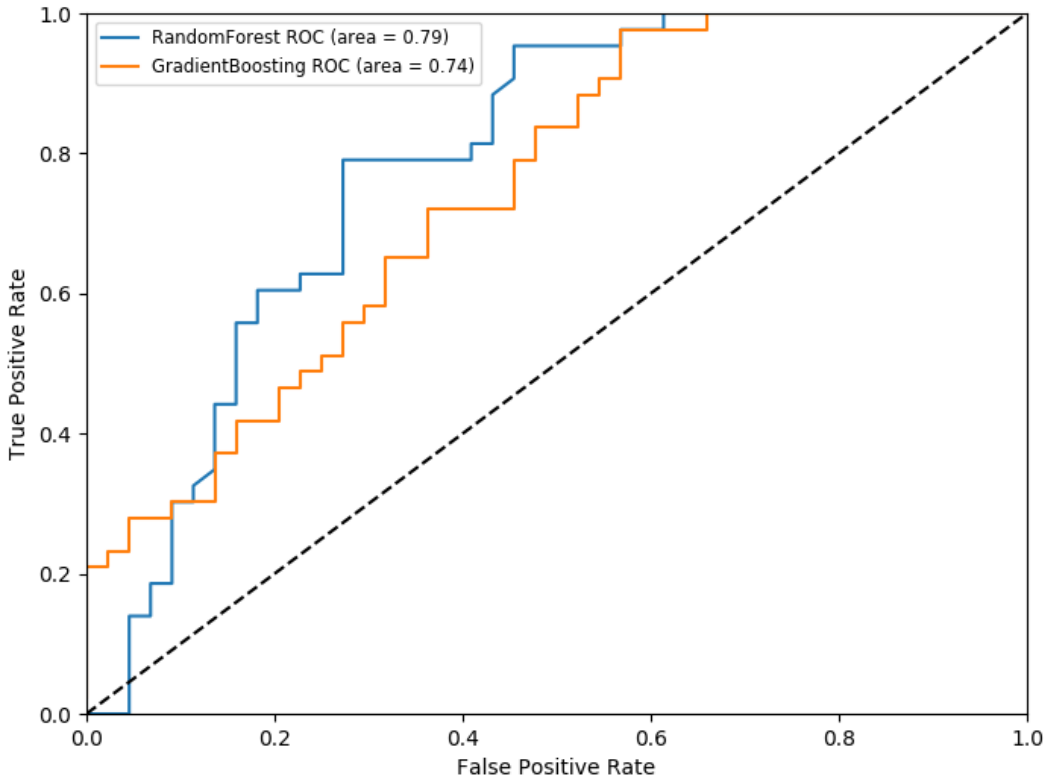


Figure 5.2: ROC curve for Random Forest Classifier Vs Gradient Boosting

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 Heptalogist 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

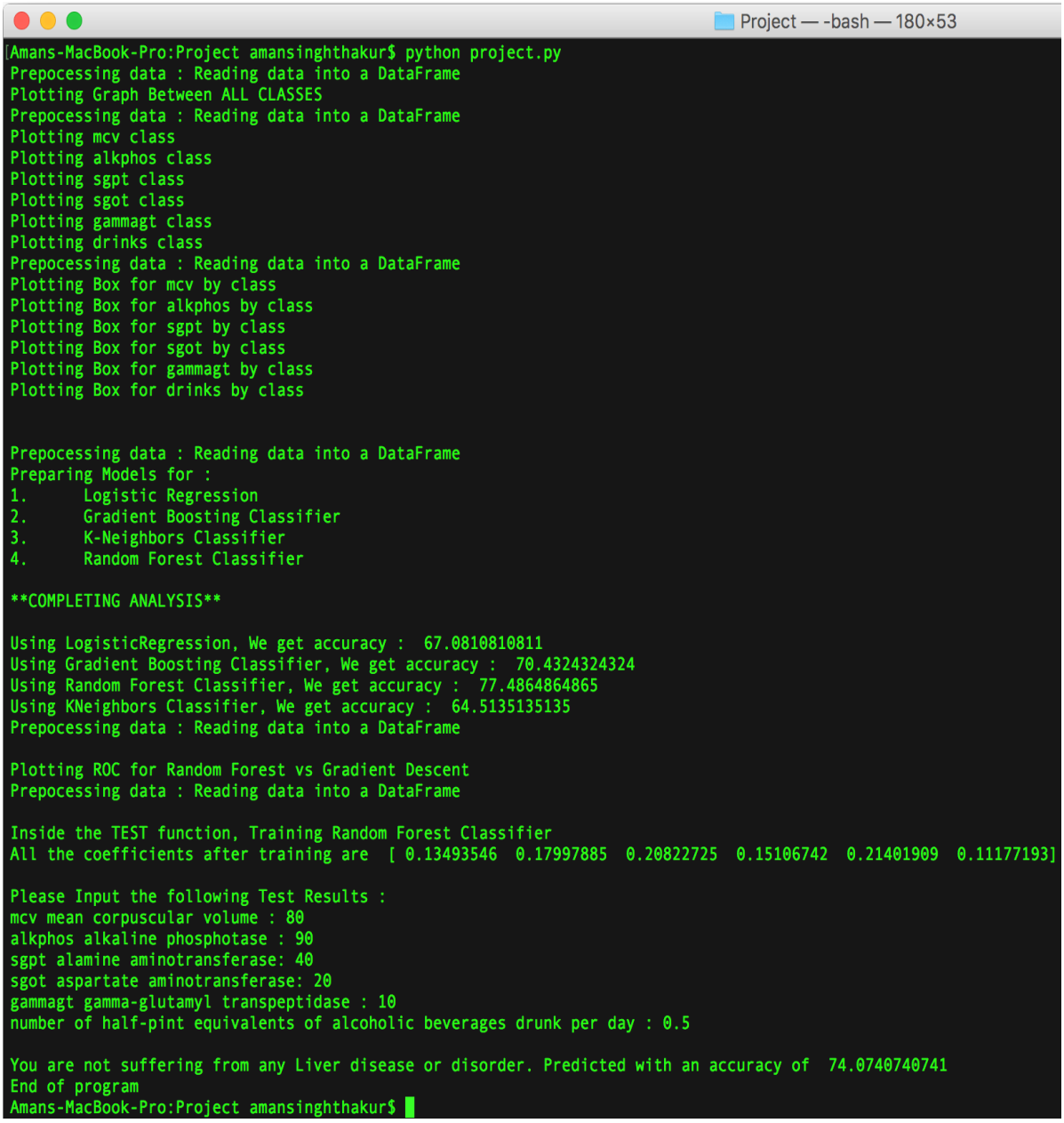


Figure 6.1: Full Program example 1

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

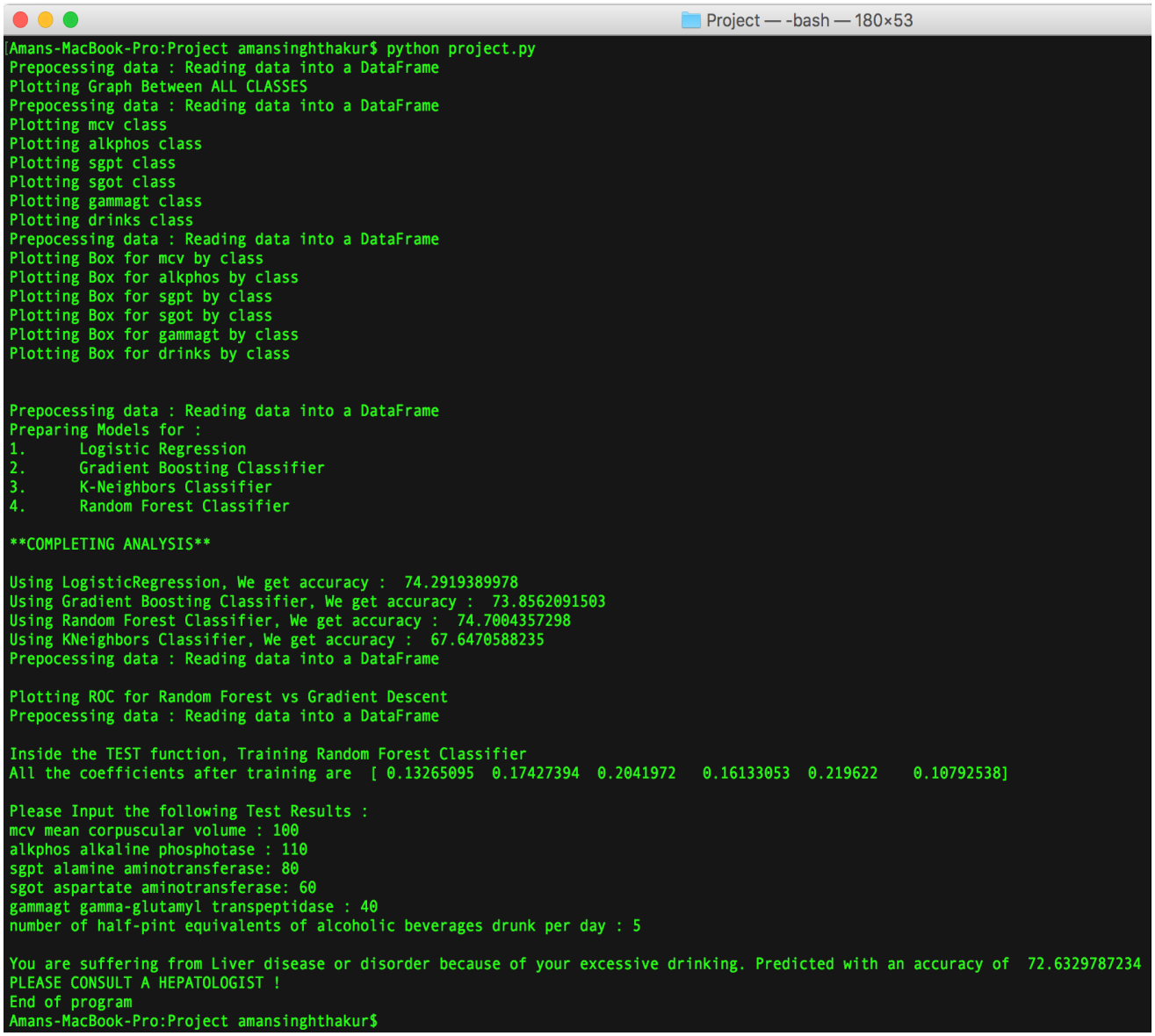


Figure 6.2: Full program example 2

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.

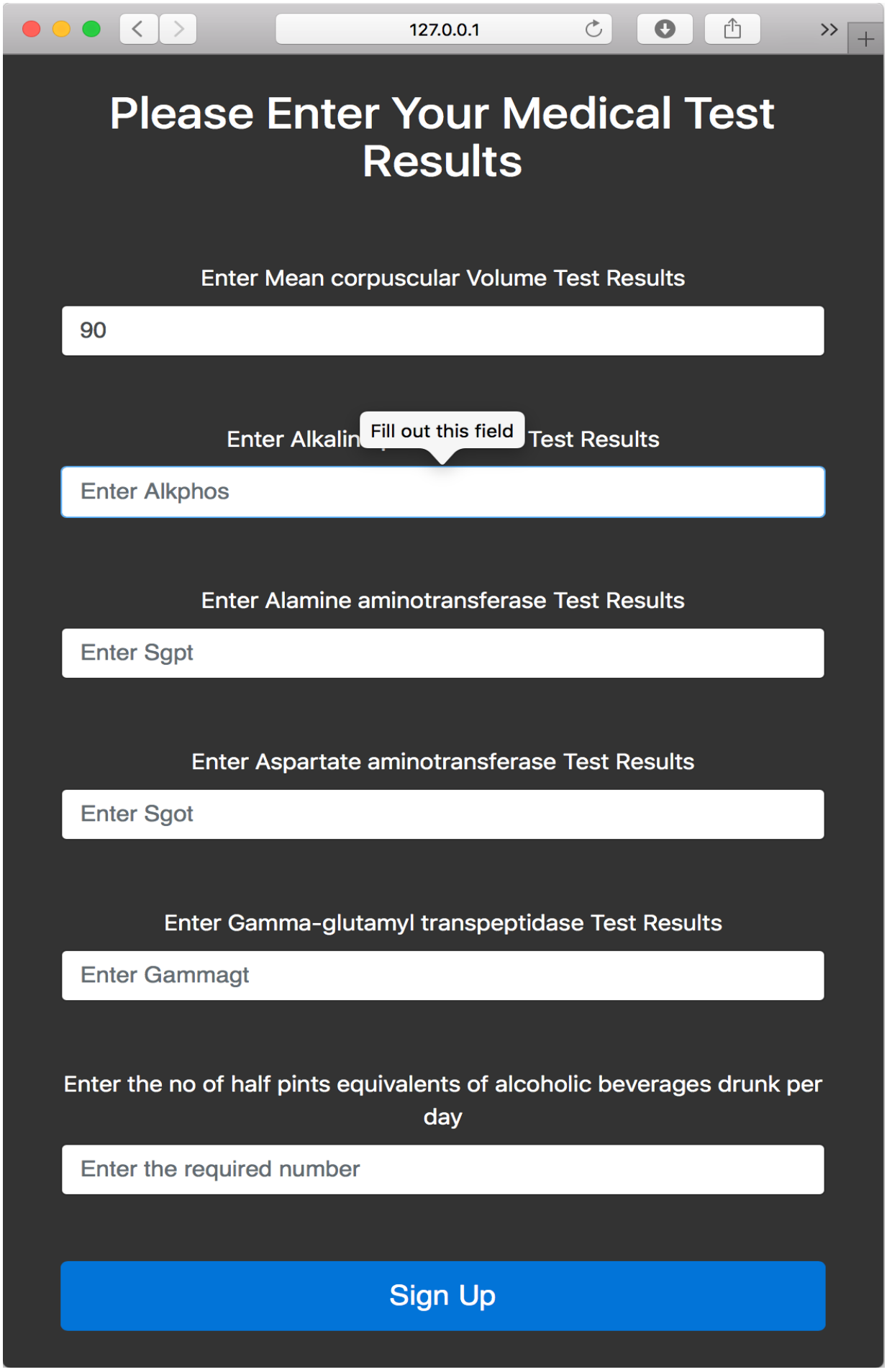


Figure 7.2: Taking Inputs with Validators in Website

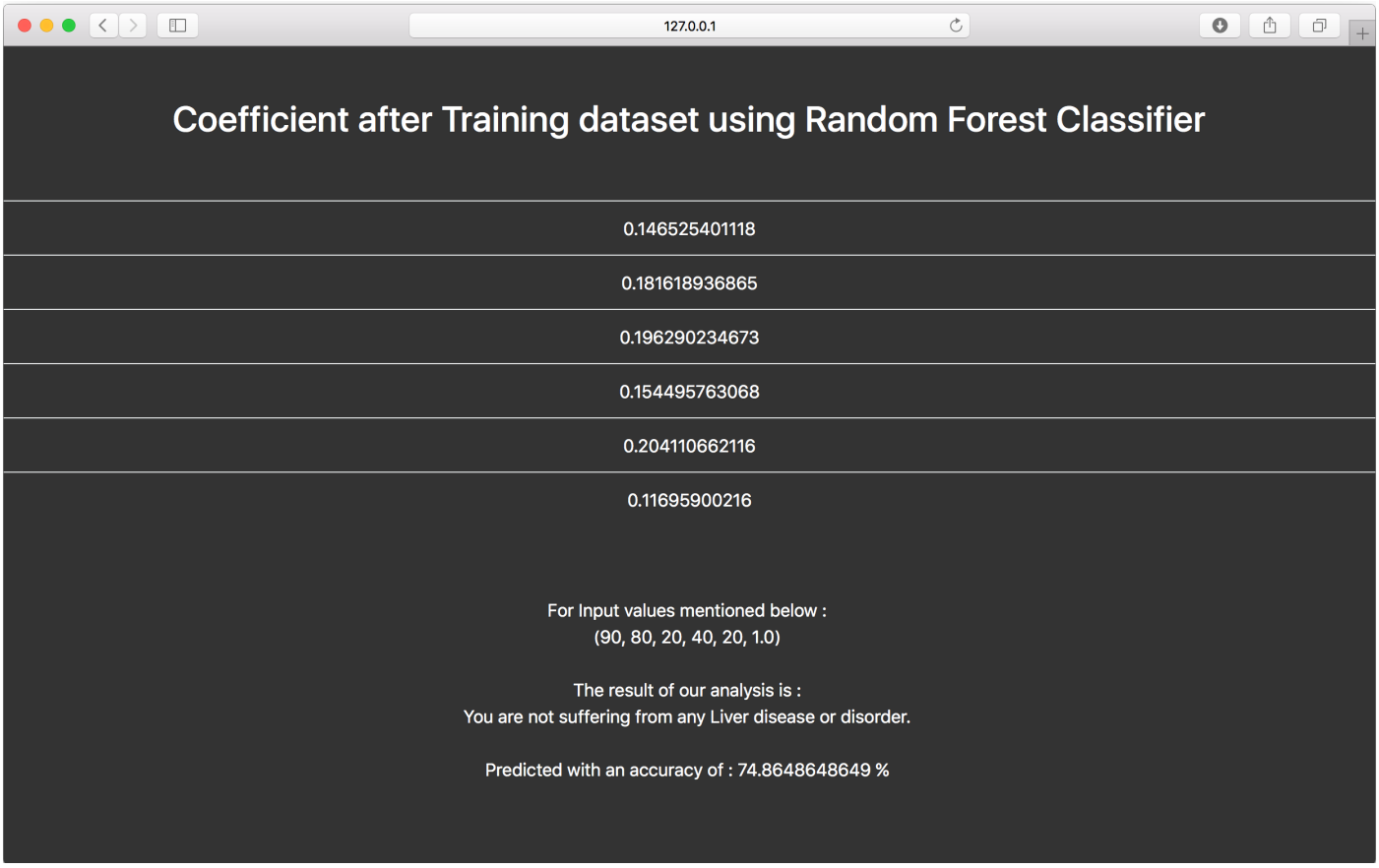


Figure 7.3: Results shown on final page of Liver Care Operations with coefficients with predicted accuracy.

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 ply

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 plot
import os


'''
********************************************************************************
                                START OF PROGRAM
********************************************************************************
'''

###################### FETCHING AND PREPROCESSING DATA ######################

def data_preprocessing():
   print "Prepocessing 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 alogrithm_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”,
eexcessive 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://justbelieverecovery.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