

Applied machine learning

mini project

December 9, 2016

**Abstract**:

We are a three-member team implementing 3 algorithms in Python using open source datasets from UCI & from Human Activity Recognition. We will evaluate our implementation of the algorithm and also compare those with the output of Weka as well. We draw conclusions from these comparison, provide pitfalls of our implementation, performance of the algorithm, and ways to improve our implementation and also the algorithm’s output.

# Group Members

Balaji Rajaram(IU id)

Dipankar Biswas(dbiswas)

Malabika Biswas(mbiswas)

Table of Contents

[Dataset 1 3](#_Toc468990764)

[**Dataset Name**: Human Activity Recognition 3](#_Toc468990765)

[**Dataset Description**: 3](#_Toc468990766)

[**Naïve Bayes Algorithm**: 3](#_Toc468990767)

[Dataset 2 7](#_Toc468990768)

# Dataset 1

### **Dataset Name**: Human Activity Recognition

### **Dataset Description**:

Human Activity Recognition (HAR) - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community, especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises. This dataset has 5 classes (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects [1].

### **Naïve Bayes Algorithm**:

In this section, we discuss the analysis performed on the dataset using Naïve Bayes algorithm, what are the pre-processing steps that we have carried out, analysis on the dataset, output of the algorithm from our python implementation, Weka output, comparison of both the results, learnings, pitfalls of our implementation and potential ways to improvise the performance.

#### Analysis & pre-processing:

This dataset has 159 columns including the target class. The first analysis we performed was to find out columns that can be dropped. We have identified that the below columns may not be helpful in the classification problem and hence we dropped these columns. Figure 1 shows the sample values for these columns.

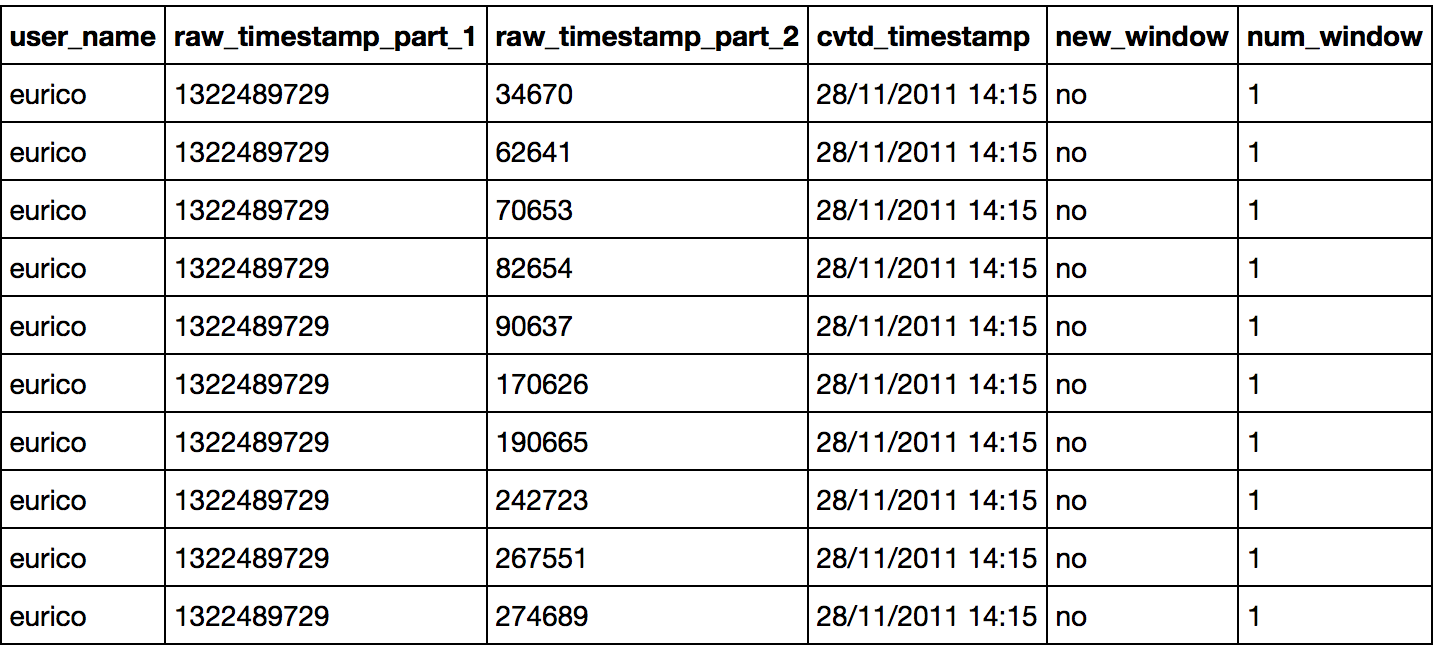


Figure 1

The next analysis we carried out to minimize the features was to find out the columns that has less than 5% value of the total record count in the dataset. Table-x shows the total record count and the threshold value to drop columns.

Table 1

|  |  |
| --- | --- |
| Total Record Count | 39242 |
| Threshold Record Count | 1962.1 |

We identified 100 columns that were below the threshold and we dropped those columns. With this initial pre-processing, we came up with 53 columns including the target class. We then did feature correlation analysis to identify how the features are correlated. Figure 2 shows the correlation analysis chart.

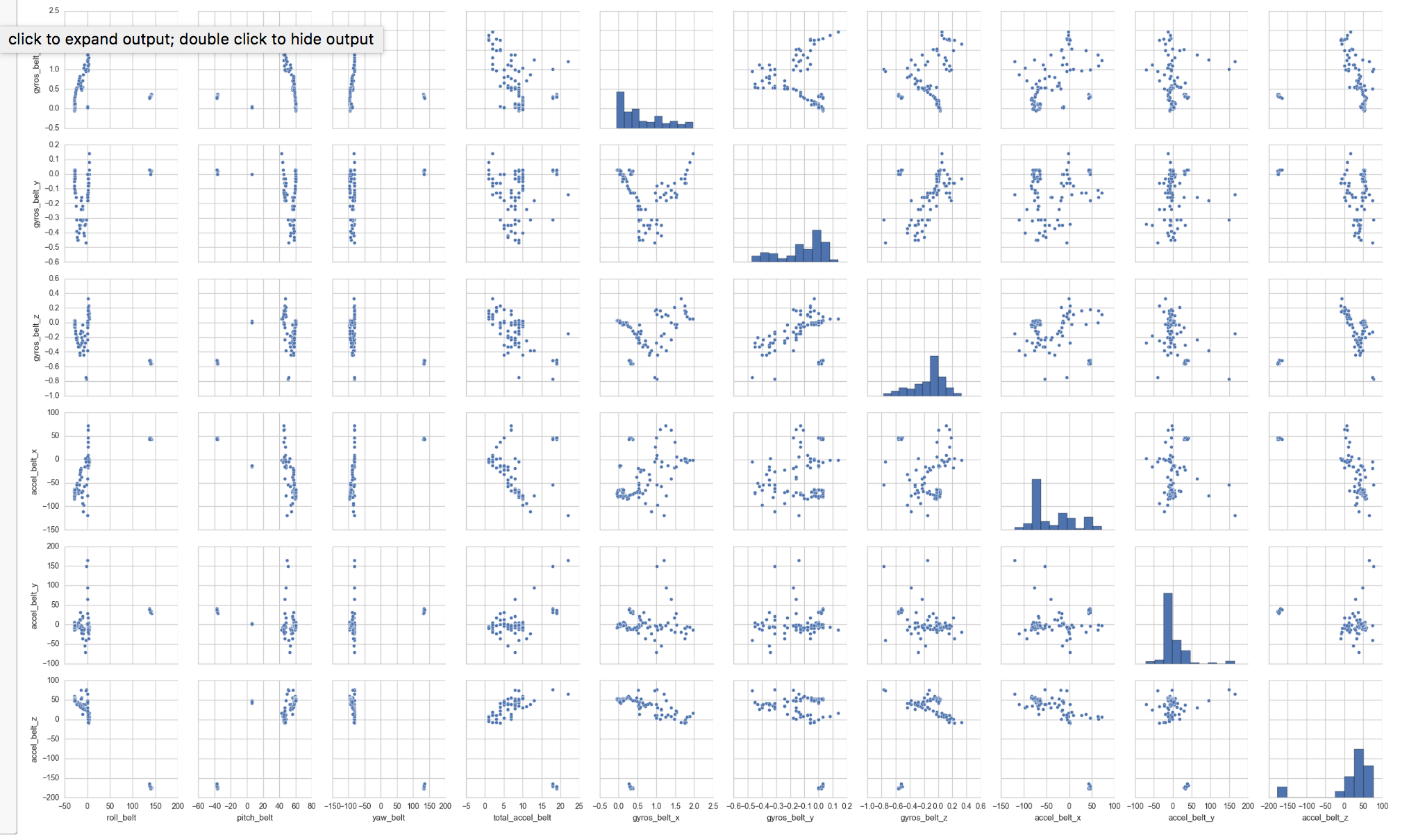


Figure 2

We could see that there is correlation between variables, which is not a good sign to perform Naïve Bayes as the algorithm works based on naive assumptions that all the variables are uncorrelated to each other, which is not true in this case.

#### **Implementation**:

#### The algorithm is implemented in Python. The program is modularized and it accepts the input file name as the input parameter. The assumption is that the last column in the dataset is the target column and also all the columns are of either integer or float type. We go over each module of the program in detail.

Load input file:

This module uses pandas dataframe to load the input csv file and then it converts all the columns to float type and it returns the processed dataframe.

Split dataset:

The next module, split dataset, is to split the given input dataset into training and test dataset. The ratio we used for dividing the dataset is 70:30 for training and test data respectively. This module returns 2 dataframes, one training dataframe and another test dataframe.

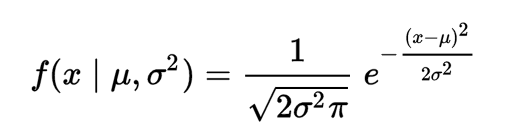
Summarize:

The next module, summarize by class, is invoked with training dataframe. This module first groups the training set into available target classes and its corresponding records. We are doing this to know the statistics about each of the target class. Once the grouping is done, we then calculate the mean and standard deviation for all the attributes for each class. This is done in order to find the probability of each column that belongs to a target class.

Train the model:

In this module, we are trying to calculate the probability of the feature for each target class. To achieve this, we calculate the probability of each attribute using the Gaussian function (Equation 1) with the mean and standard deviation calculated in the previous module.

Equation



Now that we have the class probabilities, we can find the largest probability and return the class associated with that probability.

Prediction:

We then try to predict the classes for test dataset with the summaries derived from the training dataset.

Finding Accuracy & Confusion Matrix:

We used the actual test class and the predicted test class and compare these two lists to come up with the numbers of actual class and predicted class count for each class. This matrix is then used to print the confusion matrix.

#### **Observation**:

We have implemented the Naïve Bayes algorithm in Python and we used the dataset with 53 columns. Figure 3 shows the output of our python implementation and Figure 4 shows the confusion matrix. We divided the dataset into 70-30 ratio for training data and test data respectively.

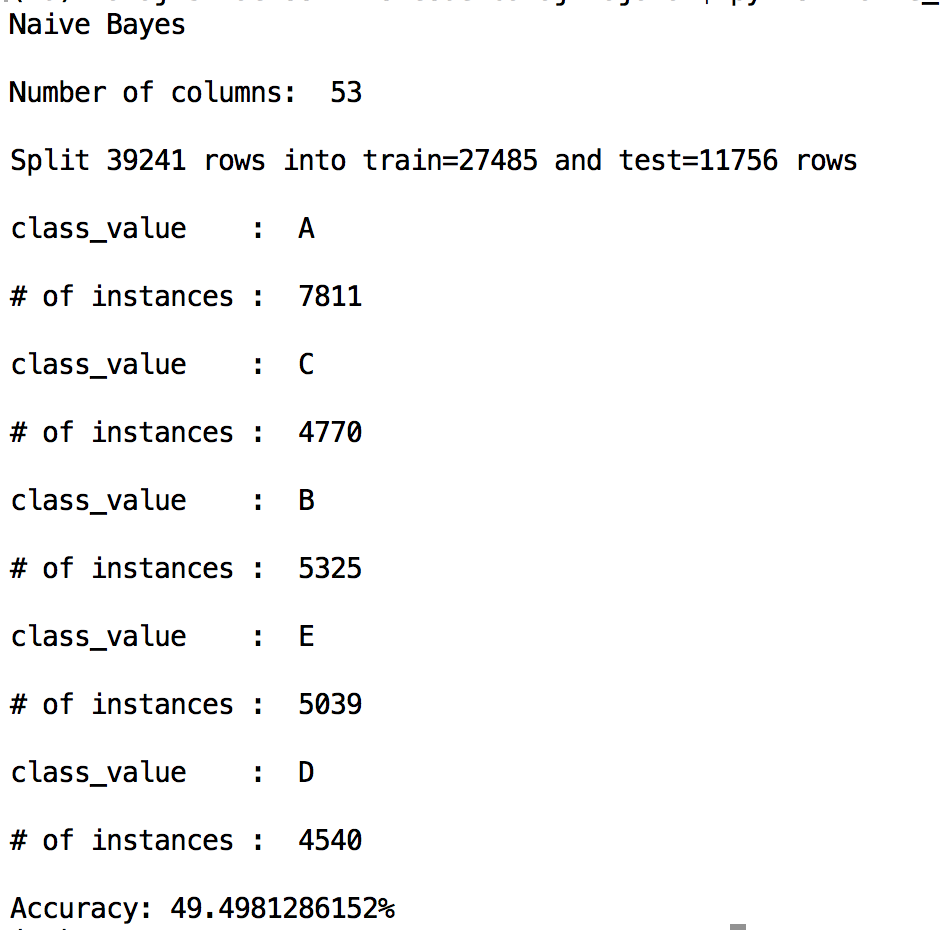


Figure 3

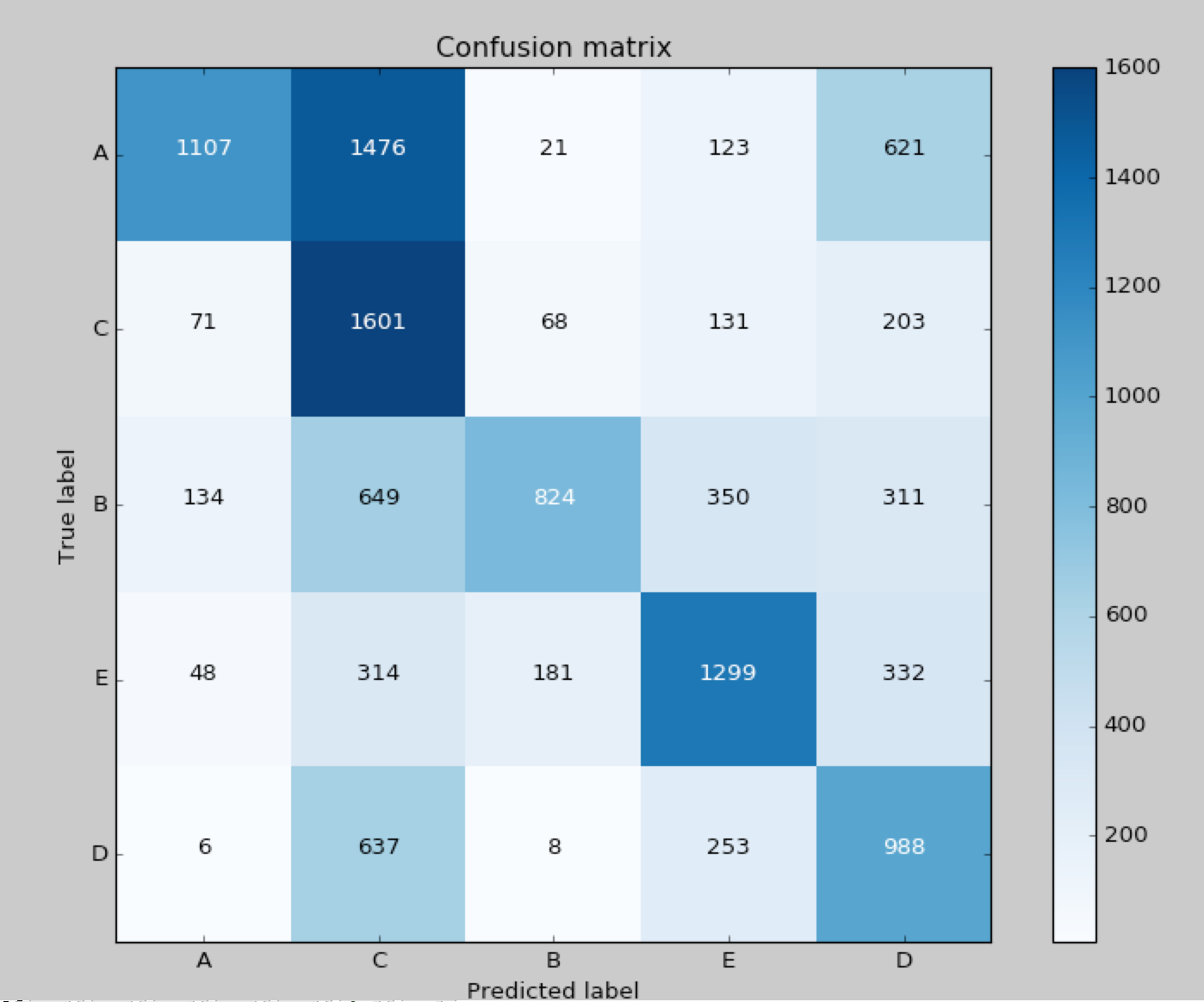
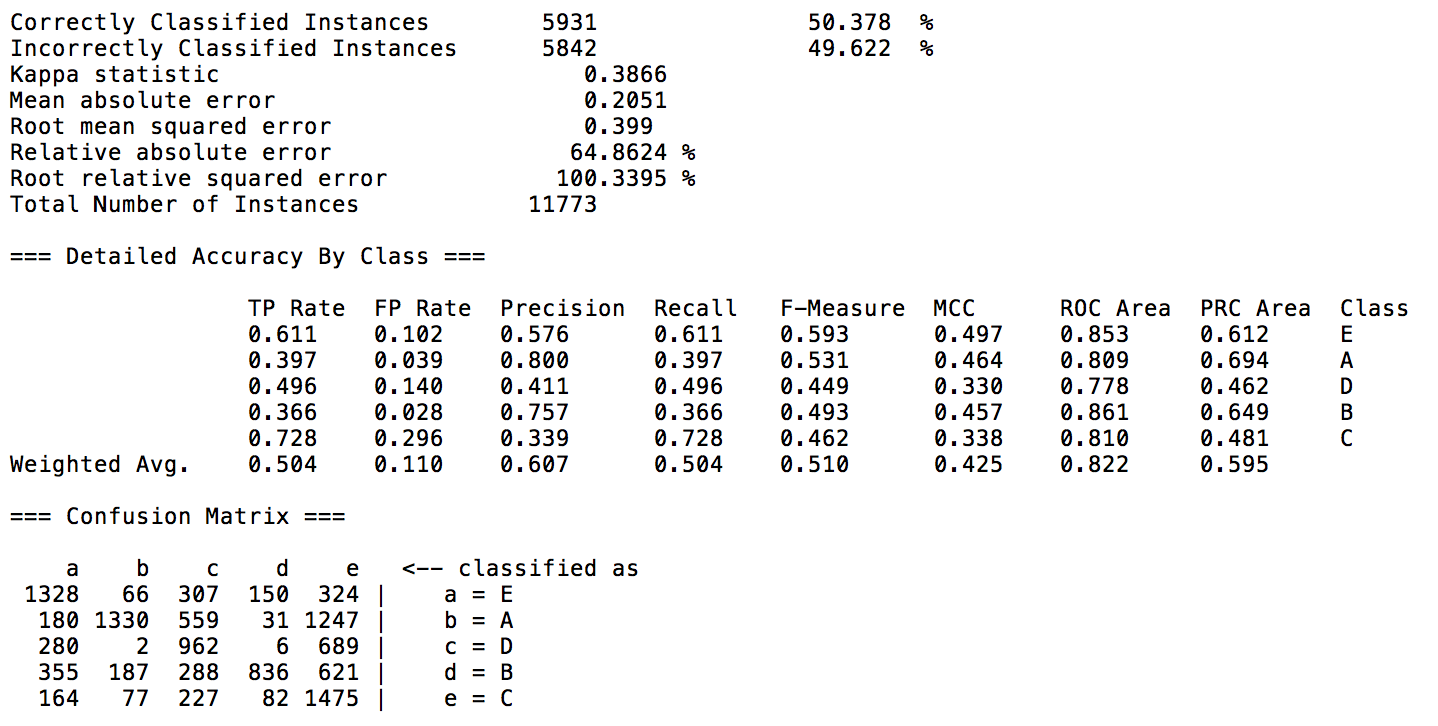


Figure 4

#### Weka Result:



#### Learning:

We were able to correlate the theoretical understanding of the algorithm with the practical implementation especially with the assumption of Naïve Bayes that the features are not correlated to each other.

#### Pitfalls & ways to improvise the classification:

It was clear that the correlation between features made the algorithm to perform poor. We need to focus more on the feature engineering part to identify the variables that have more correlation and measures to remove the correlation by retaining the features that has no correlation and by creating new variables so that we can remove the correlation between variables. Feature engineering plays a vital role in improving the algorithm’s performance.

# Dataset 2

**Dataset Name**: Student Alcohol Consumption

**Dataset Description**:

The dataset provides the result of correlation between alcohol usage and the social, gender and study time attributes for each student. This is a multivariate classification dataset [2]. The dataset has 2 target classes: one is workday alcohol consumption and the other is for weekend alcohol consumption.

**Naïve Bayes Algorithm**:

In this section, we discuss the analysis performed on the dataset using Naïve Bayes algorithm, what are the pre-processing steps that we have carried out, analysis on the dataset, output of the algorithm from our python implementation, Weka output, comparison of both the results, learnings, pitfalls of our implementation and potential ways to improvise the performance.

#### Analysis & pre-processing:

This dataset has 33 columns. Unlike the earlier HAR dataset where all the columns are of either Integer or float type, this dataset has columns with String type as well. The python implementation of our algorithm does not support String types and hence we had to convert the variables from string type to float. We converted the binary valued columns (yes/no) to 1 and 0 respectively and there were 7 such binary valued columns. Next, we converted other columns that had only 2 values other than yes/no to 1 and 0 respectively and there were 5 such columns. We then converted the categorical columns to separate columns for each value in that columns to 1 and 0 and there were 4 such columns. We came up with 46 columns. We also performed feature correlation analysis and Figure x shows the heat map chart of our initial analysis without the string columns. In this analysis, we found that grades G1, G2 and G3 are highly correlated. So, we created a new column with average of these 3 grades and we dropped the 3 source grades from the dataset. We also noted correlation between mother’s education and father’s education column, but the correlation was less than 0.7% and hence we ignored that correlation for this project. We then plotted a final correlation heat map, which is depicted in Figure x. We could notice that there is no much correlation between features. We could see that there is a correlation between 2 target variables, dalc and walc which can be ignored as we have created 2 separate datasets, one for dalc and the other one for walc. We performed individual analysis on these 2 datasets.

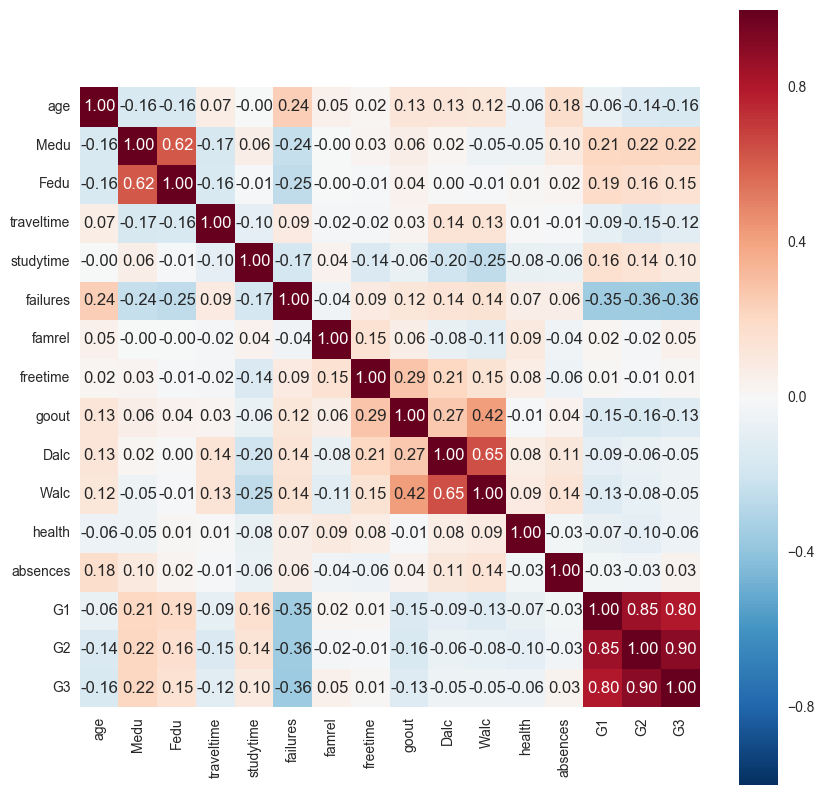


Figure 5

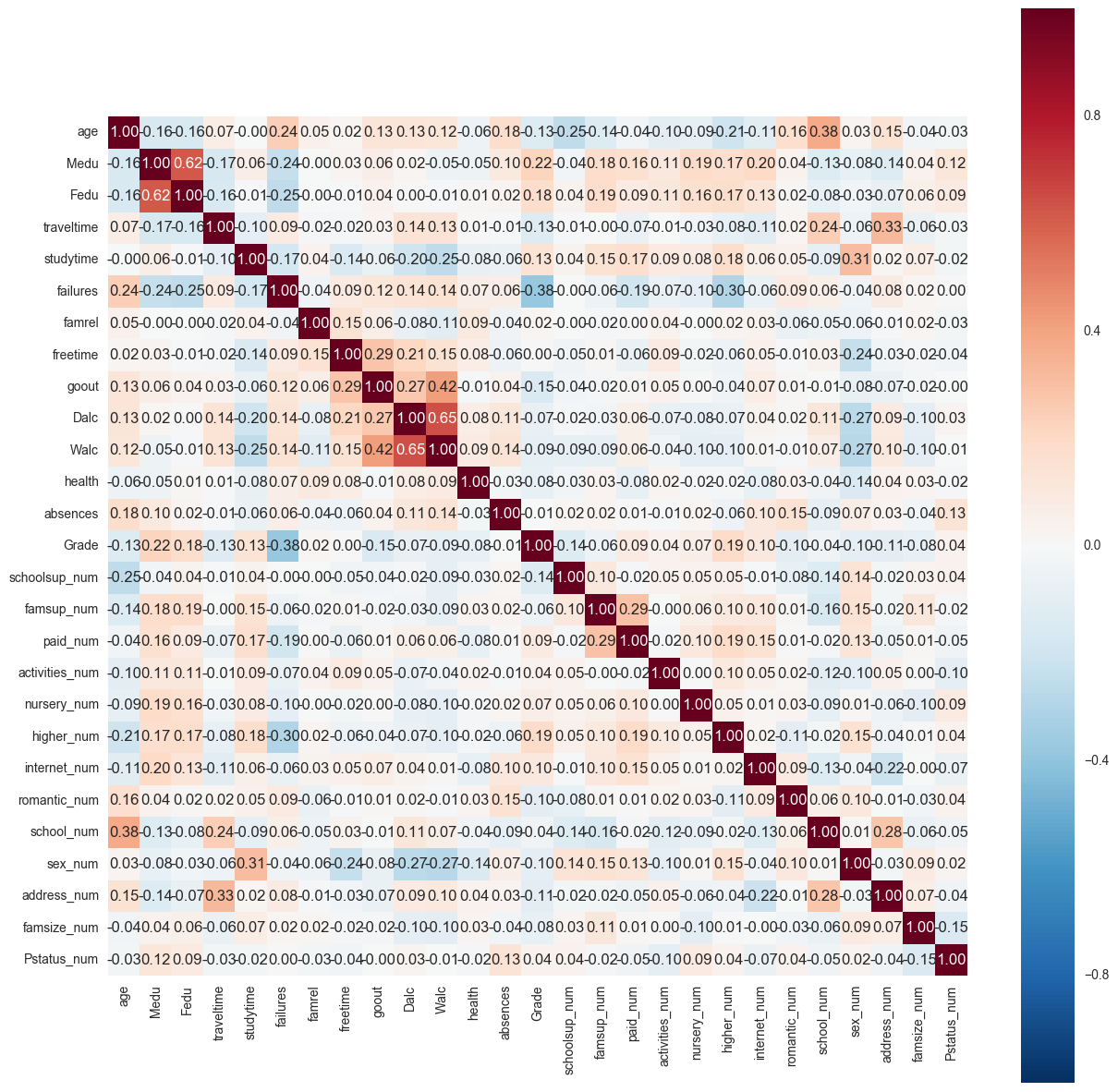
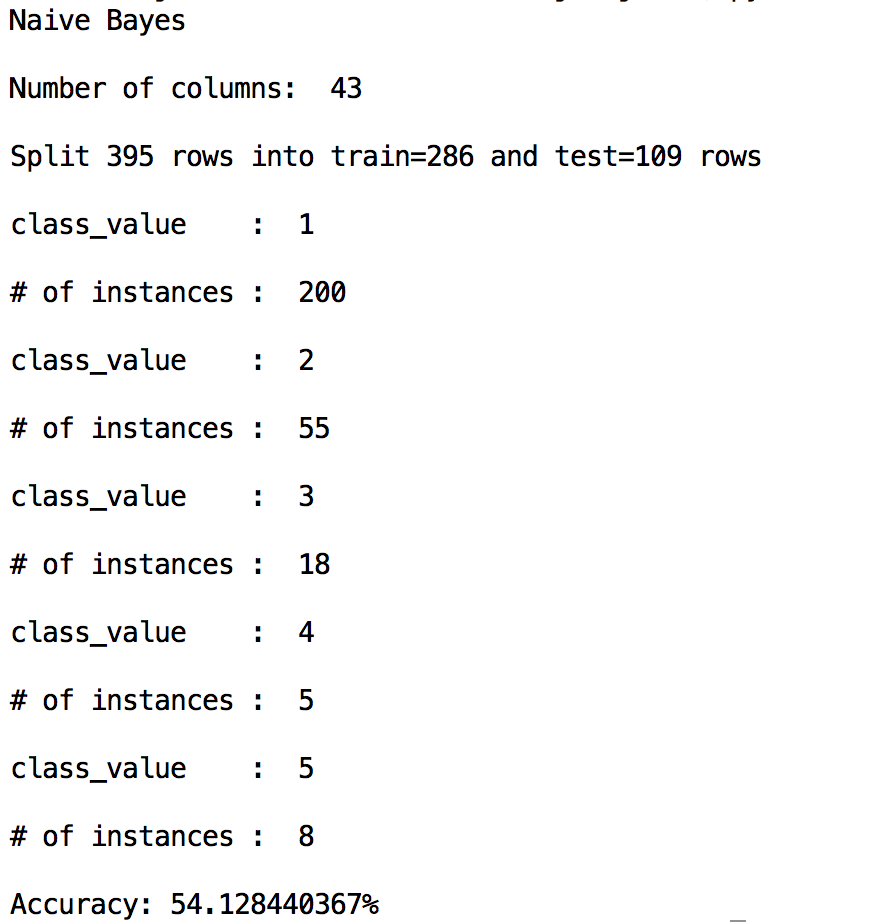
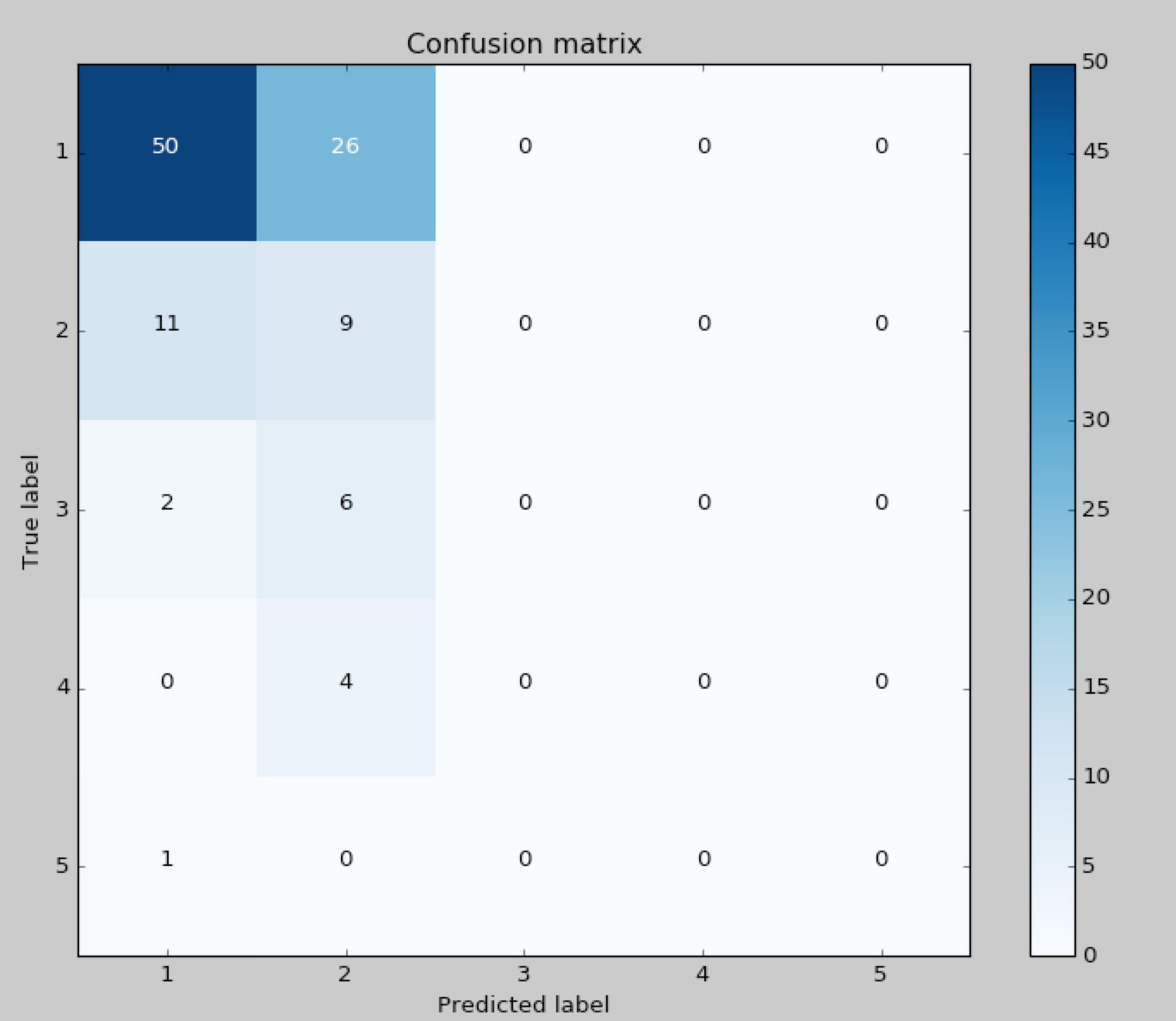


Figure 6

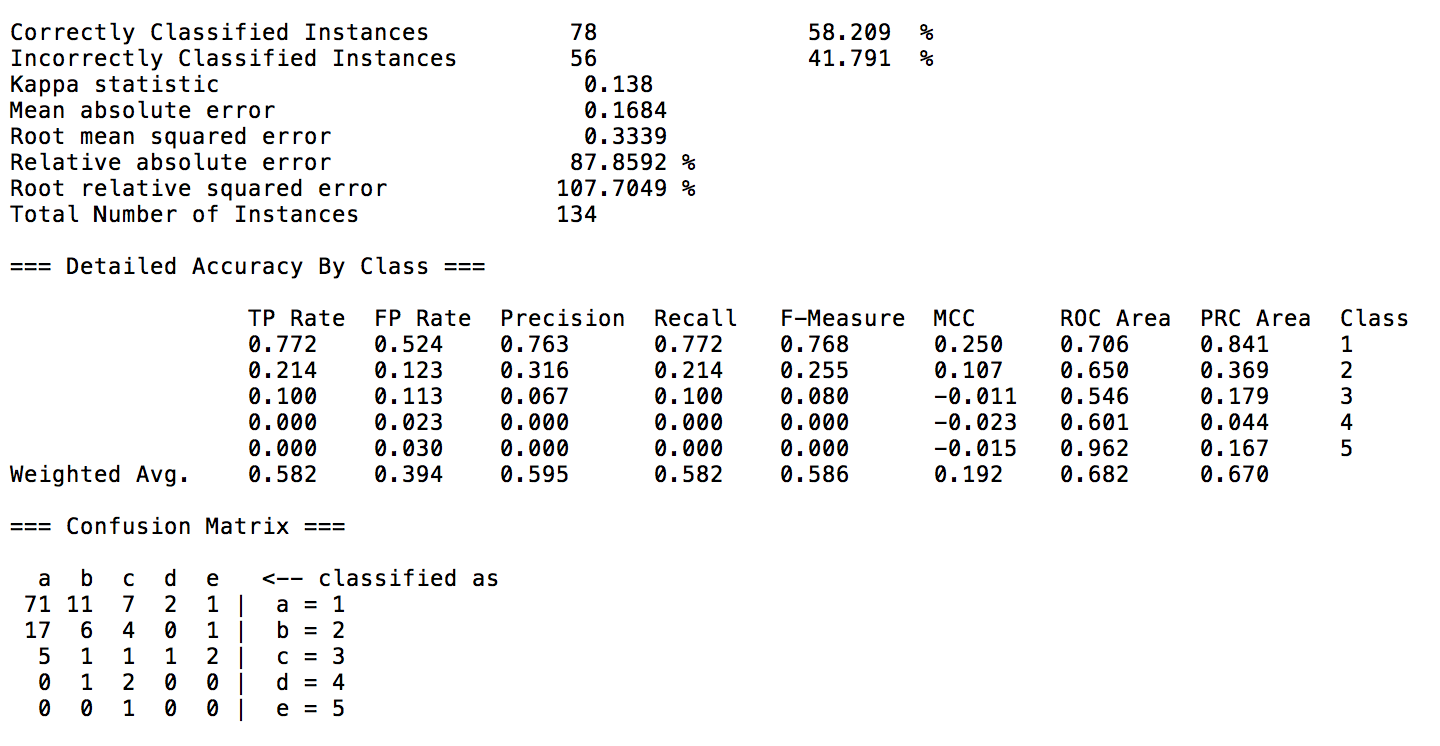
#### Work Day Alcohol Consumption Results:

We have implemented our implementation of Naïve Bayes on both the datasets separately. This section shows the result of work-day alcohol consumption. Figure x shows the terminal output and Figure x shows the confusion matrix.



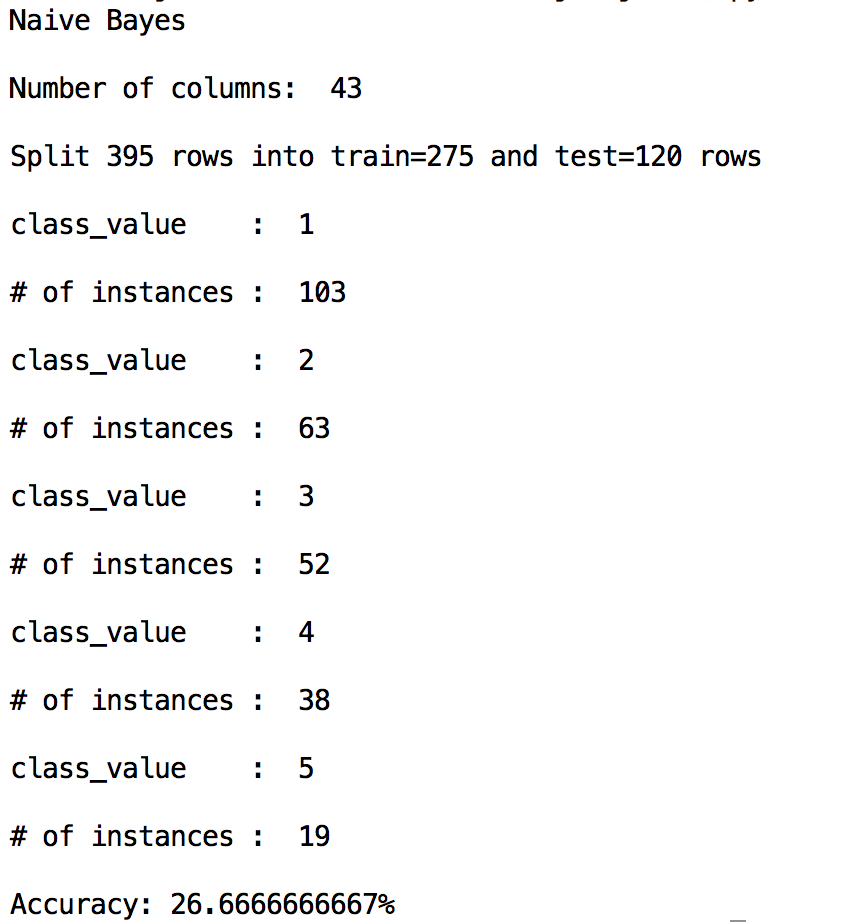
****

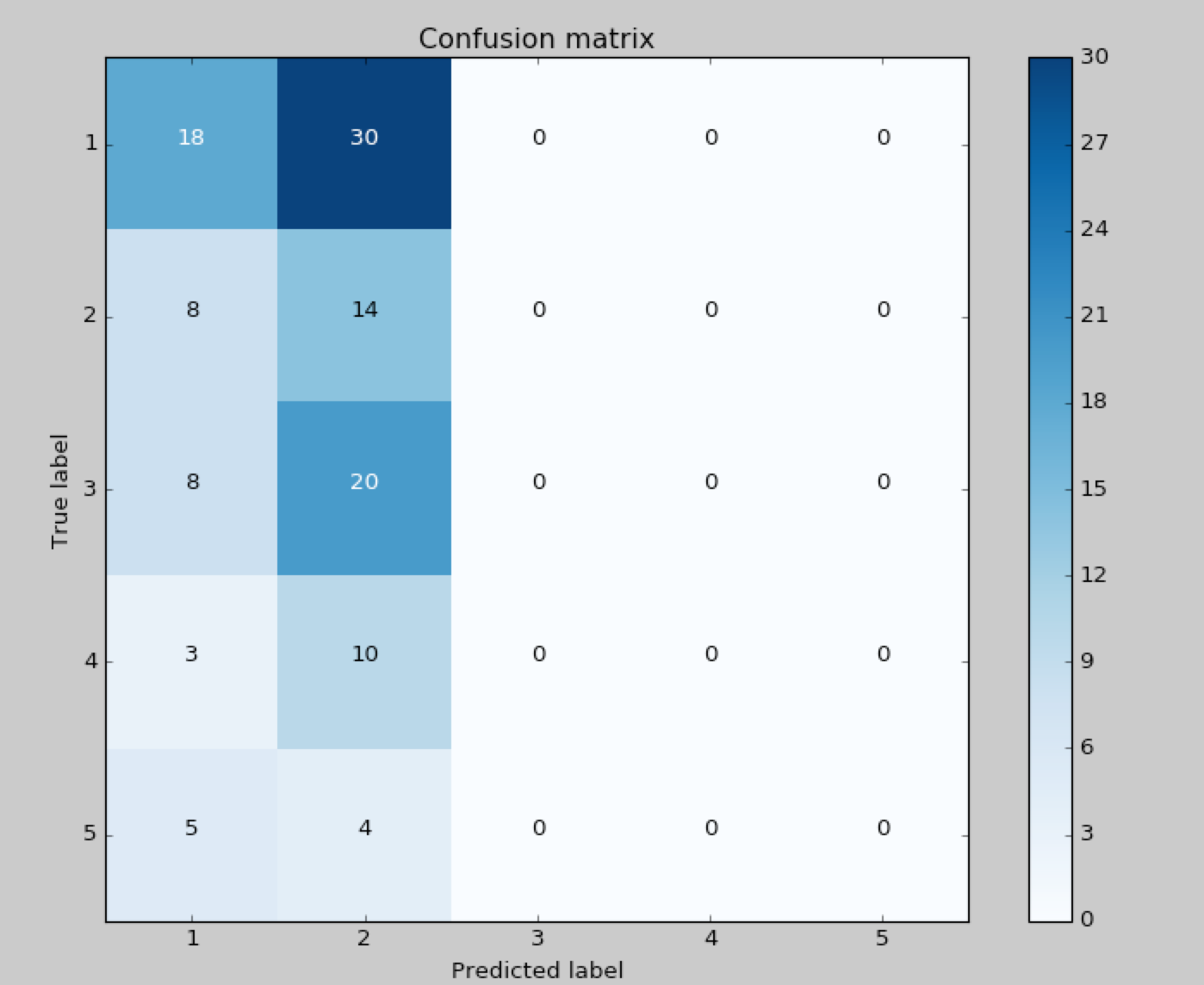
#### Weka Result:



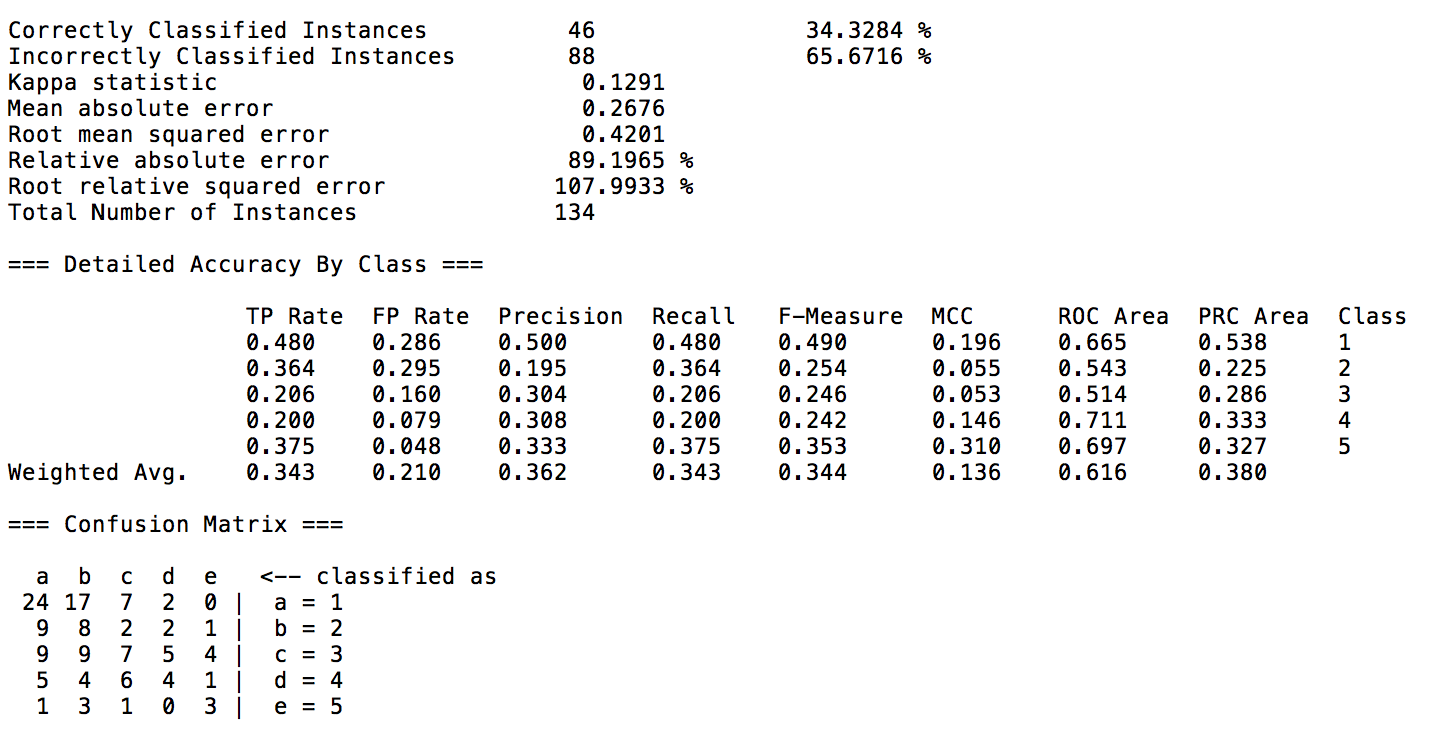
#### Weekend Alcohol Consumption Results:

This section shows the result of weekend alcohol consumption. Figure x shows the terminal output and Figure x shows the confusion matrix.





#### Weka Result:

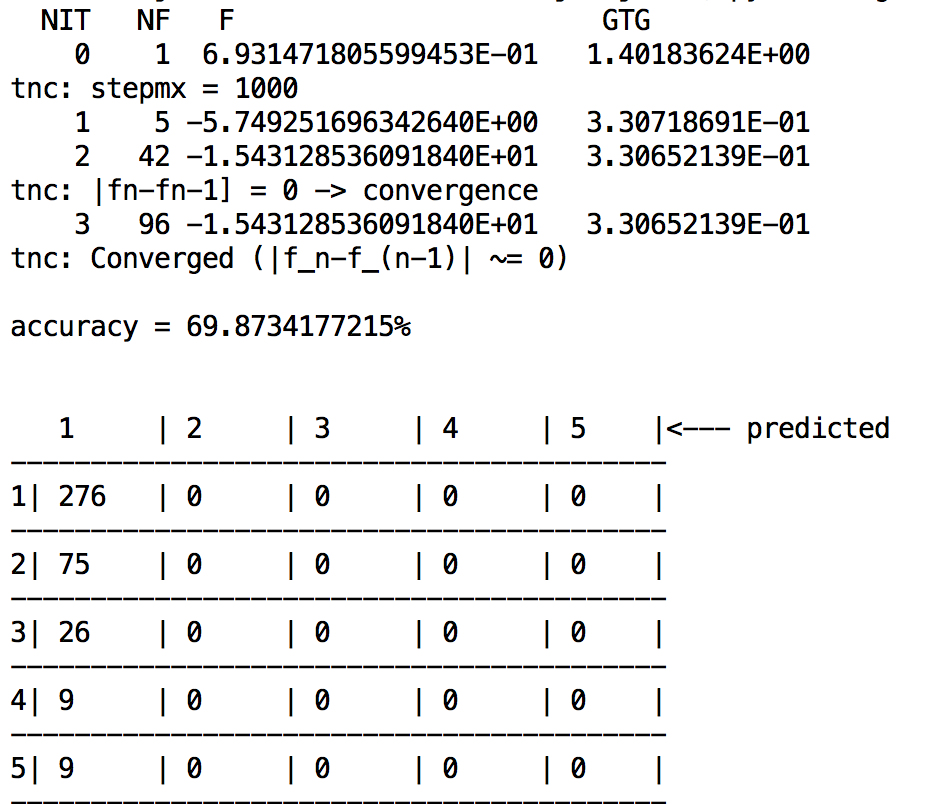


#### **Logistic Regression**:

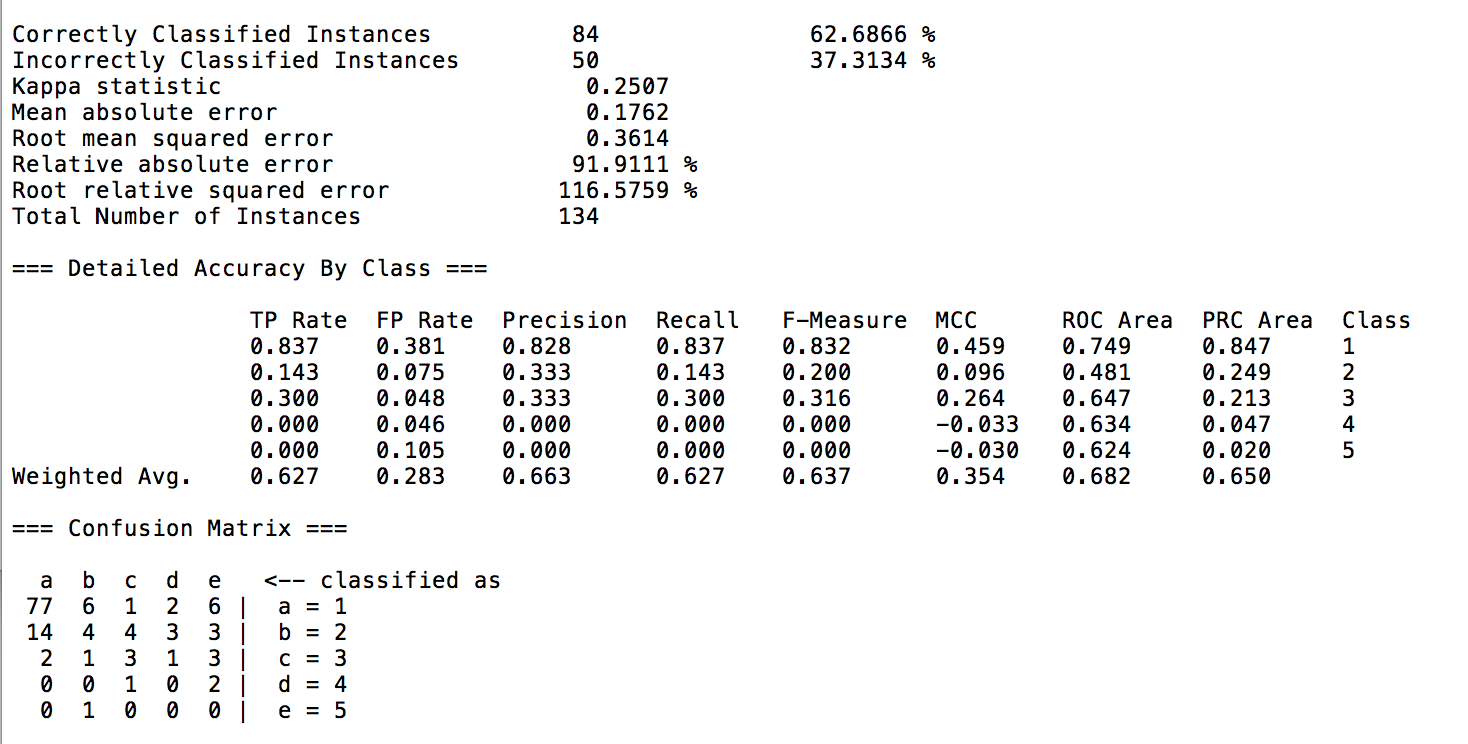
Implementation:

The algorithm is implemented in Python.

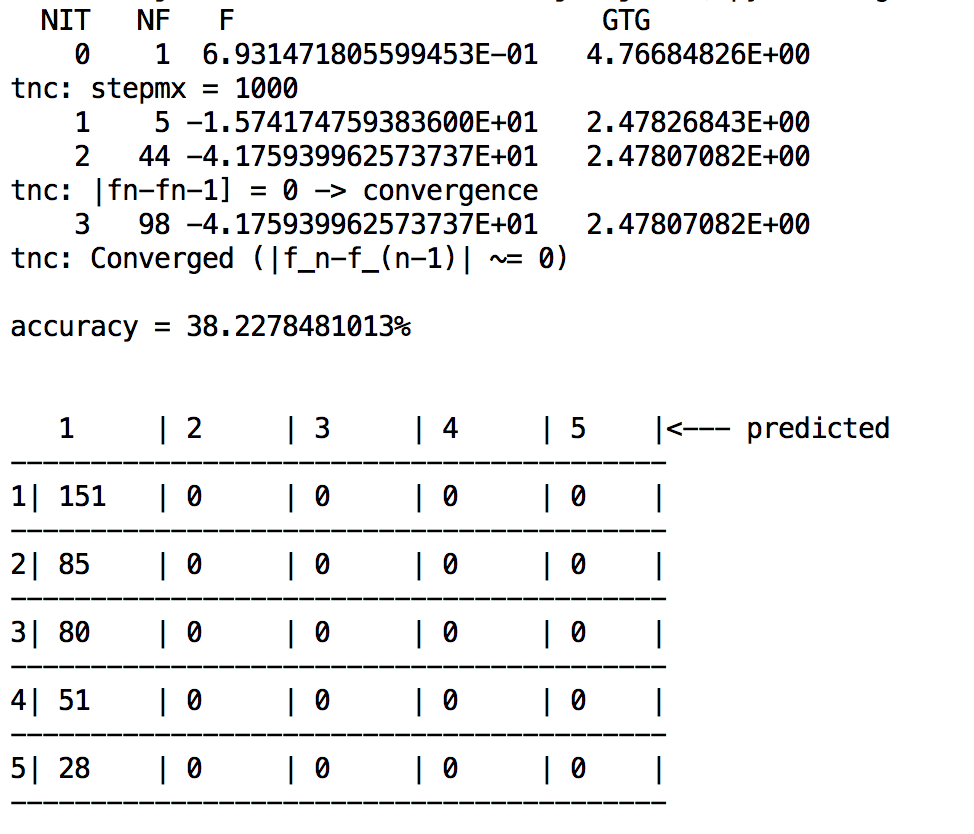
#### Work Day Alcohol Consumption Results:



#### Weka Result:

****

#### Weekend Alcohol Consumption Results:



#### Weka Result:

# Dataset 3

### **Dataset Name**: Caravan Insurance

### **Dataset Description**:

### **K Nearest Neighbor (knn) Algorithm**:

In this section, we discuss the analysis performed on the dataset using KNN algorithm, what are the pre-processing steps that we have carried out, analysis on the dataset, output of the algorithm from our python implementation, Weka output, comparison of both the results, learnings, pitfalls of our implementation and potential ways to improvise the performance.

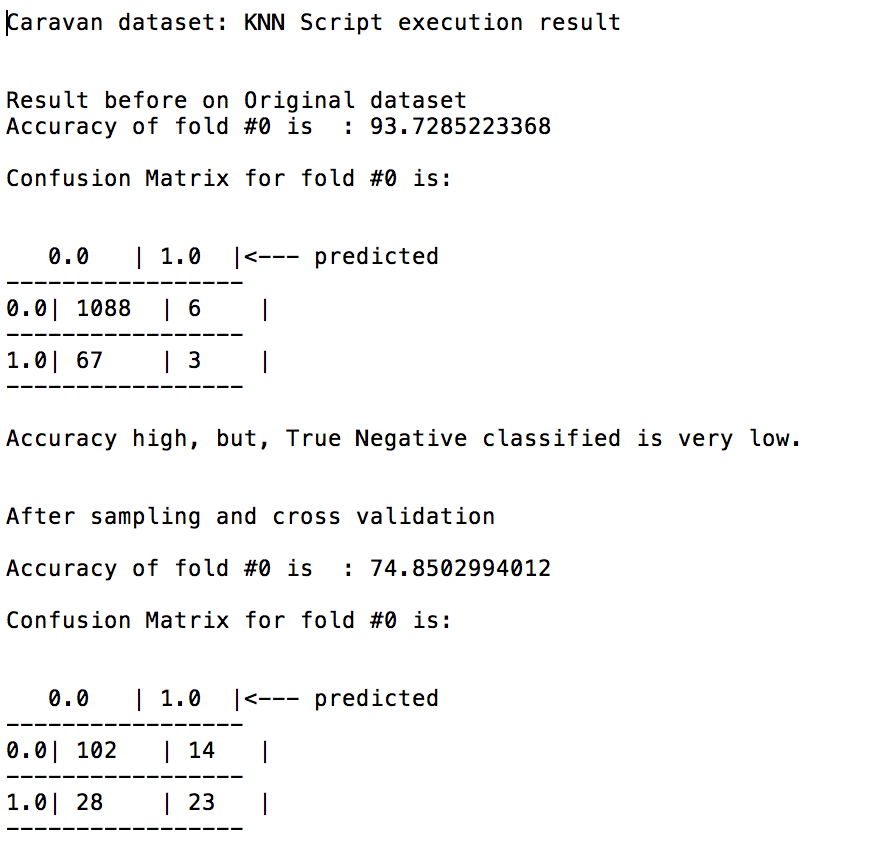
#### Analysis & pre-processing:

#### **Implementation**:

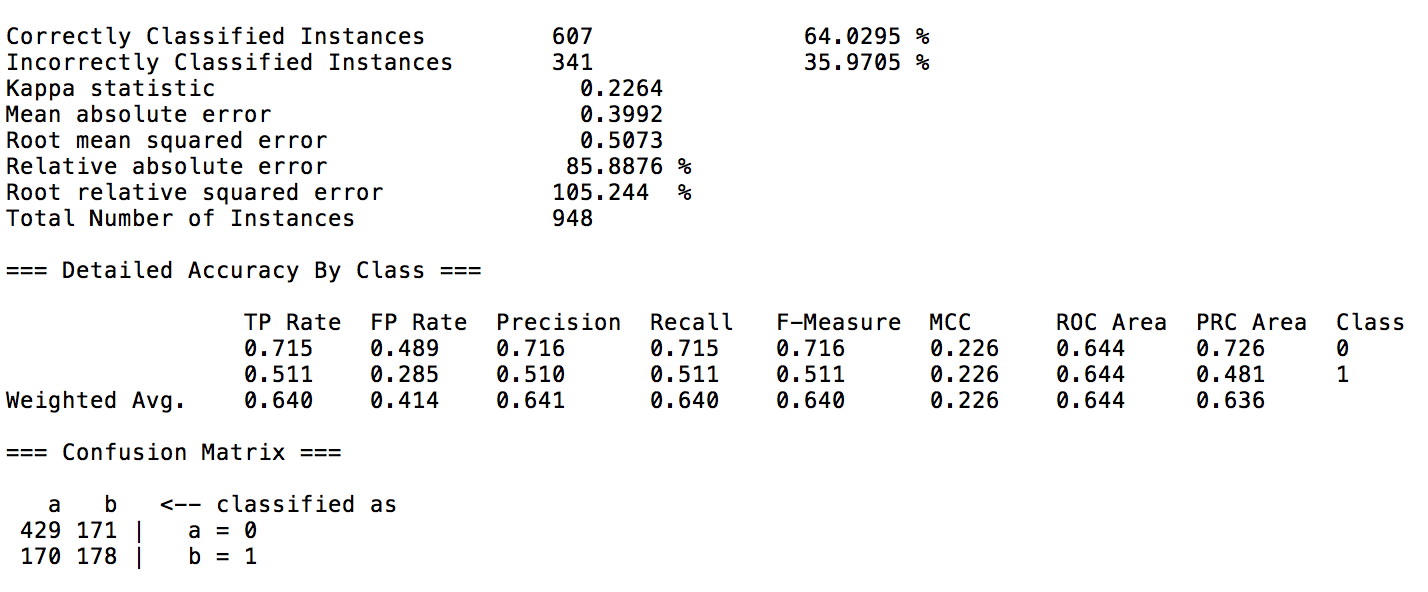
#### **Observation**:

We have implemented the KNN algorithm in Python and we used the dataset with 86 columns. We divided the dataset into 70-30 ratios for training data and test data respectively.

Below figure shows the output of our python implementation.



#### Weka Result:



References:

1. Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6\_6.
2. P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.