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 algorithms against these datasets and also compare the results with the output of Weka. We draw conclusions from these comparisons, 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(brajaram)
Dipankar Biswas(dbiswas)
Malabika Biswas(mbiswas)

Table of Contents

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.

user_name	raw_timestamp_part_1	raw_timestamp_part_2	cvtd_timestamp	new_window	num_window
eurico	1322489729	34670	28/11/2011 14:15	no	1
eurico	1322489729	62641	28/11/2011 14:15	no	1
eurico	1322489729	70653	28/11/2011 14:15	no	1
eurico	1322489729	82654	28/11/2011 14:15	no	1
eurico	1322489729	90637	28/11/2011 14:15	no	1
eurico	1322489729	170626	28/11/2011 14:15	no	1
eurico	1322489729	190665	28/11/2011 14:15	no	1
eurico	1322489729	242723	28/11/2011 14:15	no	1
eurico	1322489729	267551	28/11/2011 14:15	no	1
eurico	1322489729	274689	28/11/2011 14:15	no	1

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.

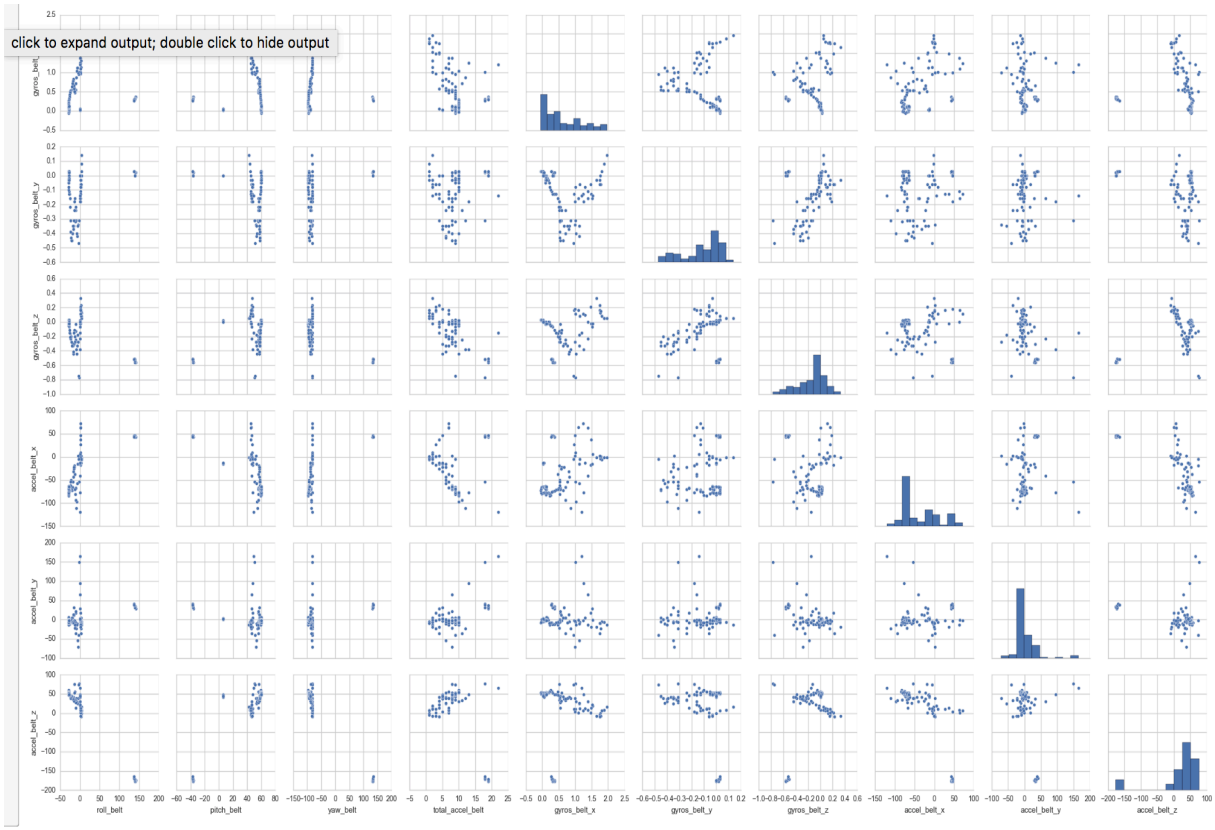


Figure 2

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.

Observations:

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.

```
Naive Bayes
Number of columns:  53
Split 39241 rows into train=27485 and test=11756 rows
class_value    :  A
# of instances :  7811
class_value    :  C
# of instances :  4770
class_value    :  B
# of instances :  5325
class_value    :  E
# of instances :  5039
class_value    :  D
# of instances :  4540
Accuracy: 49.4981286152%
```

Figure 3

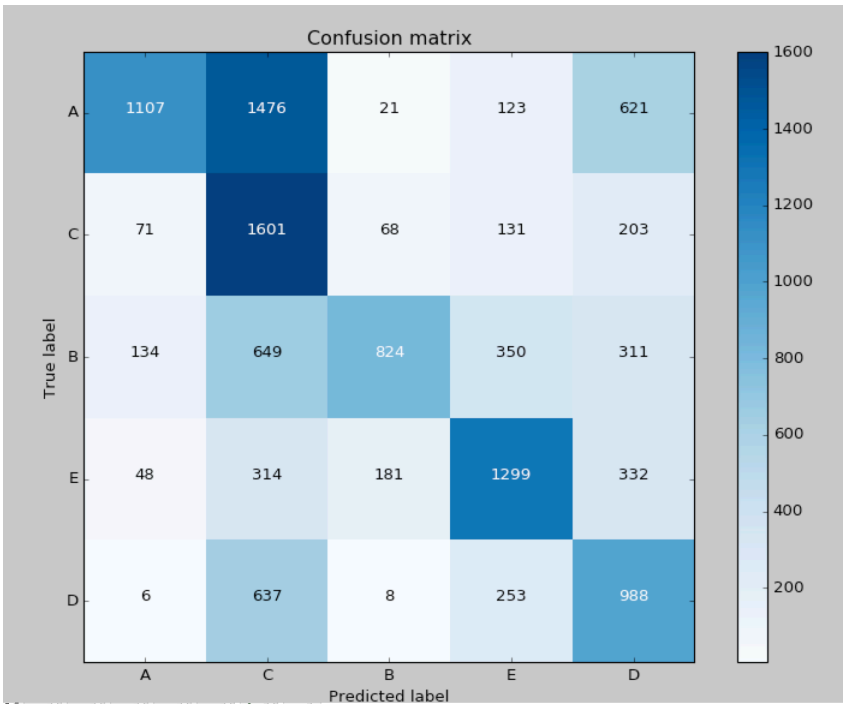


Figure 4

Weka Result:

```
Correctly Classified Instances        5931               50.378  %
Incorrectly Classified Instances      5842               49.622  %
Kappa statistic                          0.3866
Mean absolute error                      0.2051
Root mean squared error                  0.399
Relative absolute error                 64.8624 %
Root relative squared error            100.3395 %
Total Number of Instances            11773

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.611    0.102    0.576      0.611    0.593      0.497    0.853     0.612     E
                 0.397    0.039    0.800      0.397    0.531      0.464    0.809     0.694     A
                 0.496    0.140    0.411      0.496    0.449      0.330    0.778     0.462     D
                 0.366    0.028    0.757      0.366    0.493      0.457    0.861     0.649     B
                 0.728    0.296    0.339      0.728    0.462      0.338    0.810     0.481     C
Weighted Avg.    0.504    0.110    0.607      0.504    0.510      0.425    0.822     0.595

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
 1328   66  307  150  324 |    a = E
  180 1330  559   31 1247 |    b = A
  280    2  962    6  689 |    c = D
  355  187  288  836  621 |    d = B
  164   77  227   82 1475 |    e = C
```

Figure 5

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. We did not notice much difference between our implementation and Weka output.

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.

Logistic Regression:

We used the processed data that was generated for Naïve Bayes and executed the logistic regression code. The code is written in Python. Figure x shows the output and Figure x shows the confusion matrix.

```
Number of columns:  53

Split 39241 rows into train=27395 and test=11846 rows
  NIT   NF    F                          GTG
    0    1                          NAN                 NAN
    1   27                          NAN                 NAN
    2   53                          NAN                 NAN
    3   79                          NAN                 NAN
    4  105                          NAN                 NAN
    5  131                          NAN                 NAN
    6  157                          NAN                 NAN
    7  183                          NAN                 NAN
    8  209                          NAN                 NAN
    9  235                          NAN                 NAN
   10  261                          NAN                 NAN
   11  287                          NAN                 NAN
   12  313                          NAN                 NAN
   13  339                          NAN                 NAN
   14  365                          NAN                 NAN
   15  391                          NAN                 NAN
   16  417                          NAN                 NAN
   17  443                          NAN                 NAN
   18  469                          NAN                 NAN
   19  495                          NAN                 NAN
   20  521                          NAN                 NAN
   21  529                          NAN                 NAN
   21  529                          NAN                 NAN
tnc: Unable to progress

accuracy = 28.7185547864%
```

Figure 6

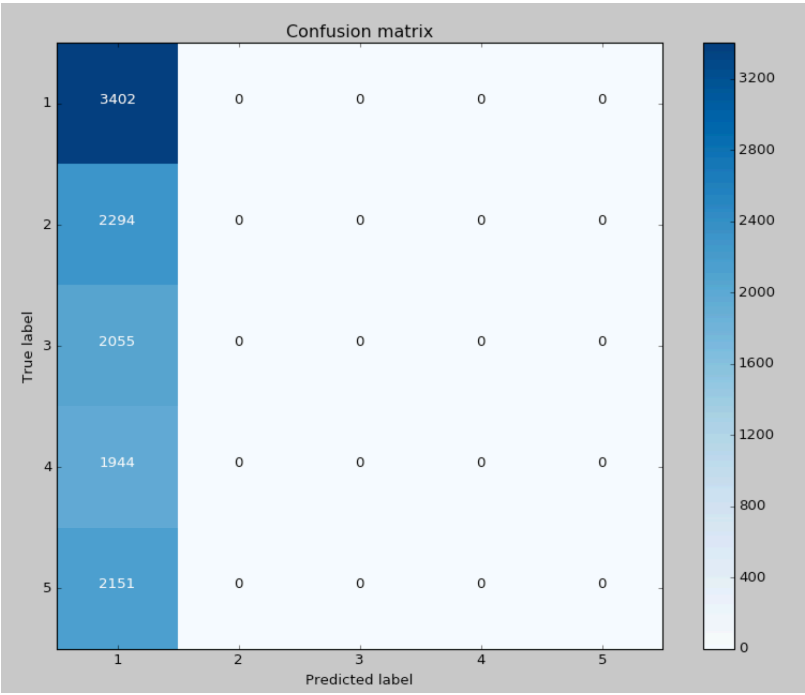


Figure 7

Weka Results:

```
Correctly Classified Instances       29101               74.1578 %
Incorrectly Classified Instances     10141               25.8422 %
Kappa statistic                          0.6723
Mean absolute error                      0.1467
Root mean squared error                  0.2705
Relative absolute error                 46.3728 %
Root relative squared error             68.0315 %
Total Number of Instances            39242

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.686    0.043    0.781      0.686    0.730      0.677    0.920     0.810
                 0.865    0.084    0.804      0.865    0.833      0.765    0.964     0.914
                 0.713    0.052    0.729      0.713    0.721      0.667    0.937     0.782
                 0.660    0.069    0.696      0.660    0.677      0.603    0.914     0.741
                 0.716    0.078    0.661      0.716    0.688      0.619    0.930     0.716
Weighted Avg.    0.742    0.067    0.742      0.742    0.741      0.676    0.936     0.805

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
 4952  341  582  917  422 |    a = E
   92 9649  459  371  588 |    b = A
  341  456 4589  271  775 |    c = D
  662  954  240 5008  729 |    d = B
  297  595  423  626 4903 |    e = C
```

Figure 8

Observations:

We could see that the logistic regression did not perform well as it attempts to predict outcomes based on a set of independent variables which is true in this case and hence the logit model is not providing useful predictions. We also noted that logistic regression is slow compared to Naïve Bayes and it takes time to converge whereas the same dataset performed well in Weka and we used cross-validation 10 folds which implies that there is an opportunity to improve our implementation.

Learning:

Logistic regression requires that each data point be independent of all other data points. If observations are related to one another, then the model will tend to overweight the significance of those observations. Hence, we need to spend more time analyzing the data including Exploratory Data Analysis (EDA).

Pitfalls & ways to improvise the classification:

The model does not have a regularized term which could have improved the performance a little bit. The main observation is that we need to do extensive feature engineering to come up with new features or identify features that best fit the logit model. We have also implemented our own gradient descent which is not performing as we were not able to identify an optimal learning rate for each dataset and hence we used the fmin_tnc function available in scikit library. This function tries to find out the theta value of global minima.

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.

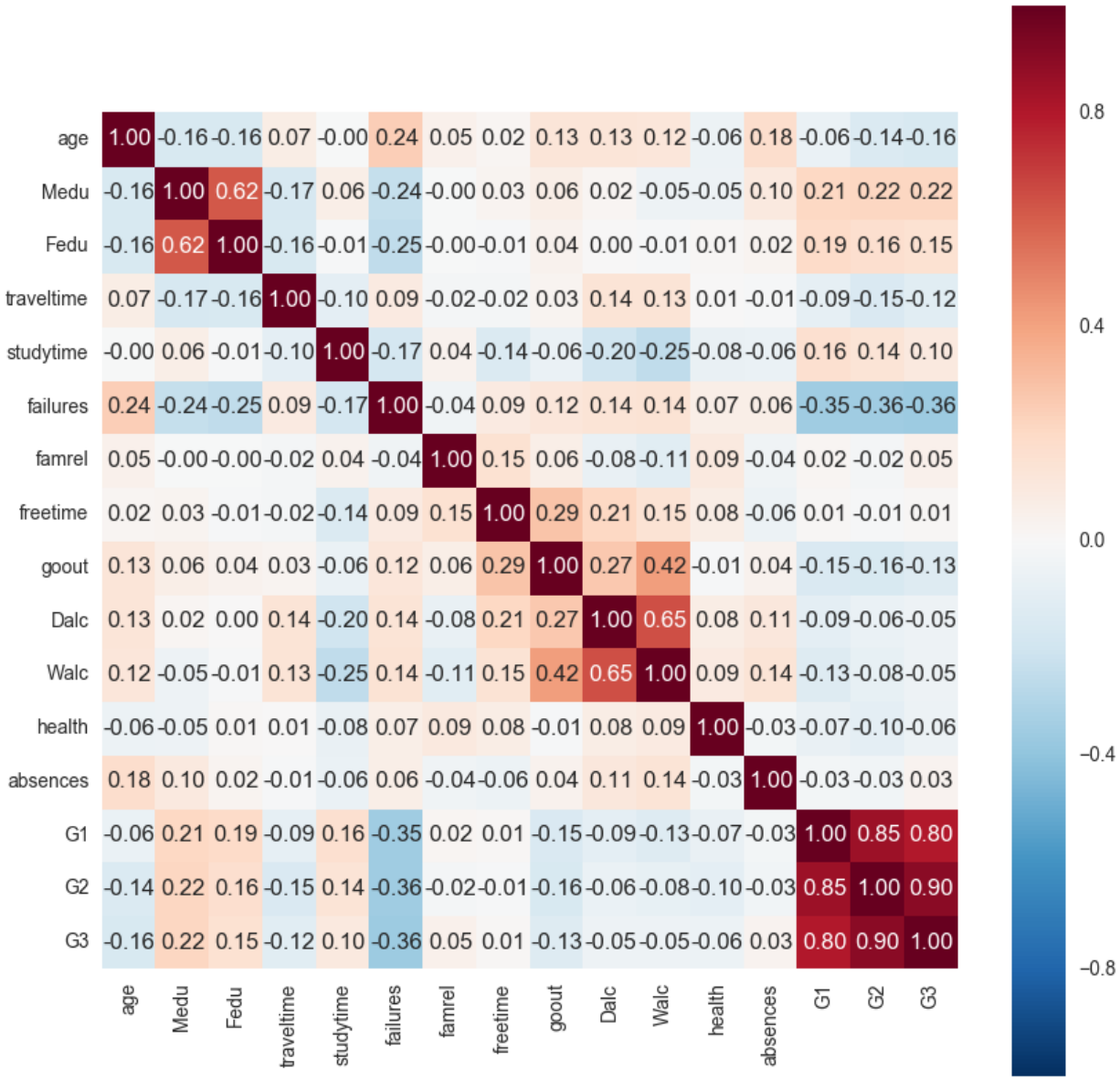

age 1.00 -0.16 -0.16 0.07 -0.00 0.24 0.05 0.02 0.13 0.13 0.12 -0.06 0.18 -0.06 -0.14 -0.16
Medu -0.16 1.00 0.62 -0.17 0.06 -0.24 -0.00 0.03 0.06 0.02 -0.05 -0.05 0.10 0.21 0.22 0.22
Fedu -0.16 0.62 1.00 -0.16 -0.01 -0.25 -0.00 -0.01 0.04 0.00 -0.01 0.01 0.02 0.19 0.16 0.15
traveltime 0.07 -0.17 -0.16 1.00 -0.10 0.09 -0.02 -0.02 0.03 0.14 0.13 0.01 -0.01 -0.09 -0.15 -0.12
studytime -0.00 0.06 -0.01 -0.10 1.00 -0.17 0.04 -0.14 -0.06 -0.20 -0.25 -0.08 -0.06 0.16 0.14 0.10
failures 0.24 -0.24 -0.25 0.09 -0.17 1.00 -0.04 0.09 0.12 0.14 0.14 0.07 0.06 -0.35 -0.36 -0.36
famrel 0.05 -0.00 -0.00 -0.02 0.04 -0.04 1.00 0.15 0.06 -0.08 -0.11 0.09 -0.04 0.02 -0.02 0.05
freetime 0.02 0.03 -0.01 -0.02 -0.14 0.09 0.15 1.00 0.29 0.21 0.15 0.08 -0.06 0.01 -0.01 0.01
goout 0.13 0.06 0.04 0.03 -0.06 0.12 0.06 0.29 1.00 0.27 0.42 -0.01 0.04 -0.15 -0.16 -0.13
Dalc 0.13 0.02 0.00 0.14 -0.20 0.14 -0.08 0.21 0.27 1.00 0.65 0.08 0.11 -0.09 -0.06 -0.05
Walc 0.12 -0.05 -0.01 0.13 -0.25 0.14 -0.11 0.15 0.42 0.65 1.00 0.09 0.14 -0.13 -0.08 -0.05
health -0.06 -0.05 0.01 0.01 -0.08 0.07 0.09 0.08 -0.01 0.08 0.09 1.00 -0.03 -0.07 -0.10 -0.06
absences 0.18 0.10 0.02 -0.01 -0.06 0.06 -0.04 -0.06 0.04 0.11 0.14 -0.03 1.00 -0.03 -0.03 0.03
G1 -0.06 0.21 0.19 -0.09 0.16 -0.35 0.02 0.01 -0.15 -0.09 -0.13 -0.07 -0.03 1.00 0.85 0.80
G2 -0.14 0.22 0.16 -0.15 0.14 -0.36 -0.02 -0.01 -0.16 -0.06 -0.08 -0.10 -0.03 0.85 1.00 0.90
G3 -0.16 0.22 0.15 -0.12 0.10 -0.36 0.05 0.01 -0.13 -0.05 -0.05 -0.06 0.03 0.80 0.90 1.00
age Medu Fedu traveltime studytime failures famrel freetime goout Dalc Walc health absences G1 G2 G3
0.8
0.4
0.0
−0.4
−0.8

Figure 9

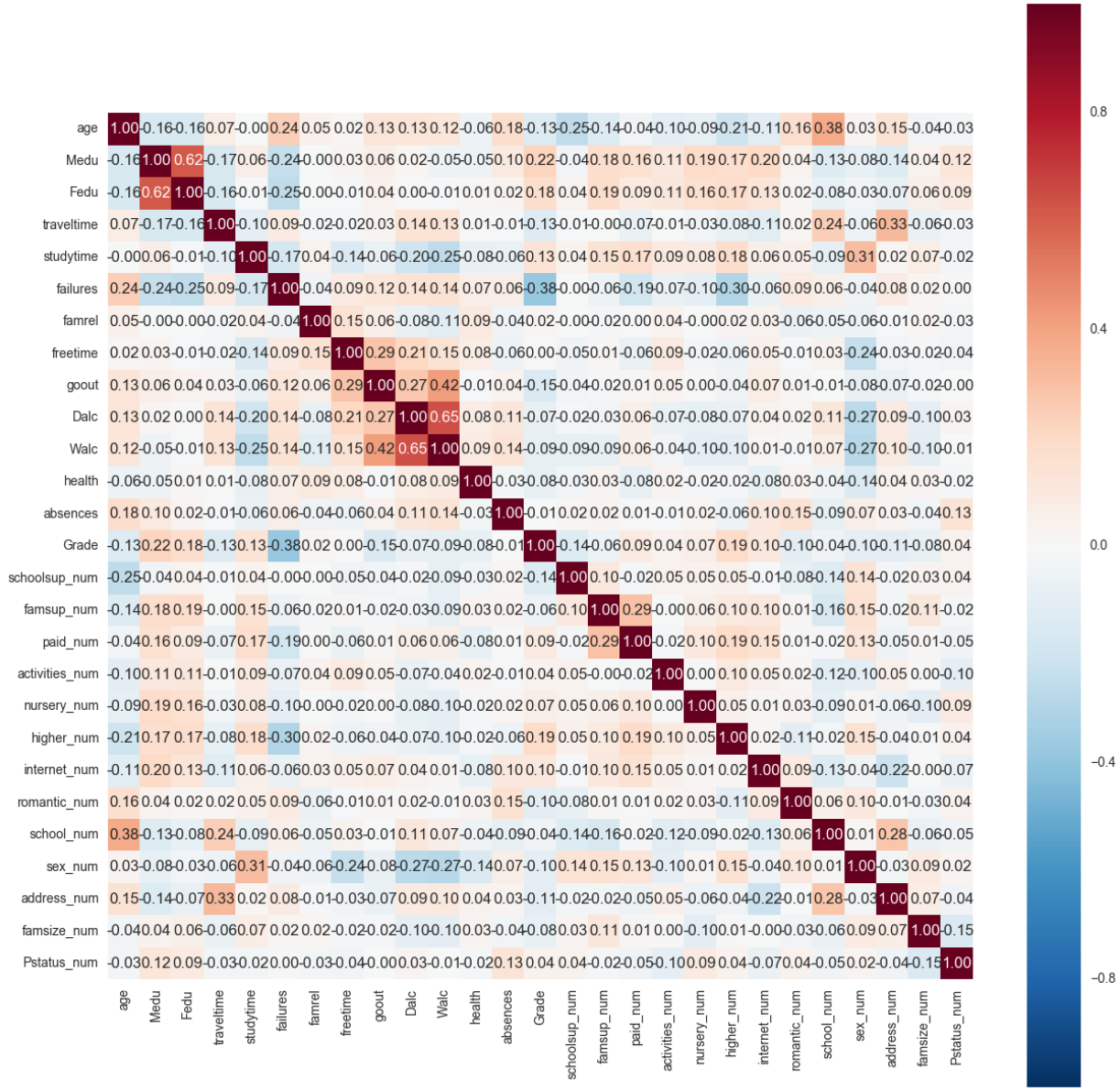
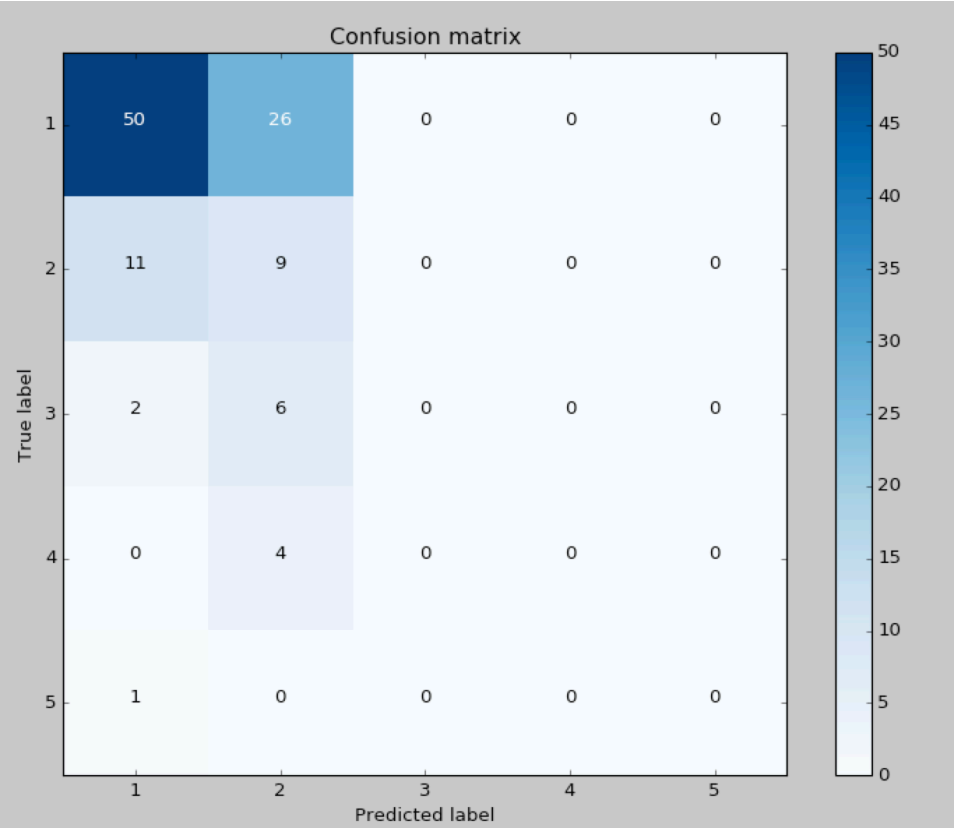

age 1.00 -0.16 -0.16 0.07 -0.00 0.24 0.05 0.02 0.13 0.13 0.12 -0.06 0.18 -0.13 -0.25 -0.14 -0.04 -0.10 -0.09 -0.21 -0.11 0.16 0.38 0.03 0.15 -0.04 -0.03
Medu -0.16 1.00 0.62 -0.17 0.06 -0.24 -0.00 0.03 0.06 0.02 -0.05 -0.05 0.10 0.22 -0.04 0.18 0.16 0.11 0.19 0.17 0.20 0.04 -0.13 -0.08 -0.14 0.04 0.12
Fedu -0.16 0.62 1.00 -0.16 0.01 -0.25 -0.00 -0.01 0.04 0.00 -0.01 0.01 0.02 0.18 0.04 0.19 0.09 0.11 0.16 0.17 0.13 0.02 -0.08 -0.03 -0.07 0.06 0.09
traveltime 0.07 -0.17 -0.16 1.00 -0.10 0.09 -0.02 -0.02 0.03 0.14 0.13 0.01 -0.01 -0.13 -0.01 -0.00 -0.07 -0.01 -0.03 -0.08 -0.11 0.02 0.24 -0.06 0.33 -0.06 -0.03
studytime -0.00 0.06 -0.01 -0.10 1.00 -0.17 0.04 -0.14 -0.06 -0.20 -0.25 -0.08 -0.06 0.13 0.04 0.15 0.17 0.09 0.08 0.18 0.06 0.05 -0.09 0.31 0.02 0.07 -0.02
failures 0.24 -0.24 -0.25 0.09 -0.17 1.00 -0.04 0.09 0.12 0.14 0.14 0.07 0.06 -0.38 -0.00 -0.06 -0.19 -0.07 -0.10 -0.30 -0.06 0.09 0.06 -0.04 0.08 0.02 0.00
famrel 0.05 -0.00 -0.00 -0.02 0.04 -0.04 1.00 0.15 0.06 -0.08 -0.11 0.09 -0.04 0.02 -0.00 -0.02 0.00 0.04 -0.00 0.02 0.03 -0.06 -0.05 -0.06 -0.01 0.02 -0.03
freetime 0.02 0.03 -0.01 -0.02 -0.14 0.09 0.15 1.00 0.29 0.21 0.15 0.08 -0.06 0.00 -0.05 0.01 -0.06 0.09 -0.02 -0.06 0.05 -0.01 0.03 -0.24 -0.03 -0.02 -0.04
goout 0.13 0.06 0.04 0.03 -0.06 0.12 0.06 0.29 1.00 0.27 0.42 -0.01 0.04 -0.15 -0.04 -0.02 0.01 0.05 0.00 -0.04 0.07 0.01 -0.01 -0.08 -0.07 -0.02 -0.00
Dalc 0.13 0.02 0.00 0.14 -0.20 0.14 -0.08 0.21 0.27 1.00 0.65 0.08 0.11 -0.07 -0.02 -0.03 0.06 -0.07 -0.08 -0.07 0.04 0.02 0.11 -0.27 0.09 -0.10 0.03
Walc 0.12 -0.05 -0.01 0.13 -0.25 0.14 -0.11 0.15 0.42 0.65 1.00 0.09 0.14 -0.09 -0.09 -0.09 0.06 -0.04 -0.10 -0.10 0.01 -0.01 0.07 -0.27 0.10 -0.10 0.01
health -0.06 -0.05 0.01 0.01 -0.08 0.07 0.09 0.08 -0.01 0.08 0.09 1.00 -0.03 -0.08 -0.03 0.03 -0.08 0.02 -0.02 -0.02 -0.08 0.03 -0.04 -0.14 0.04 0.03 -0.02
absences 0.18 0.10 0.02 -0.01 -0.06 0.06 -0.04 -0.06 0.04 0.11 0.14 -0.03 1.00 -0.01 0.02 0.02 0.01 -0.01 0.02 -0.06 0.10 0.15 -0.09 0.07 0.03 -0.04 0.13
Grade -0.13 0.22 0.18 -0.13 0.13 -0.38 0.02 0.00 -0.15 -0.07 -0.09 -0.08 0.01 1.00 -0.14 0.06 0.09 0.04 0.07 0.19 0.10 -0.10 -0.04 -0.10 -0.11 -0.08 0.04
schoolsup_num -0.25 -0.04 0.04 -0.01 0.04 -0.00 -0.00 -0.05 -0.04 -0.02 -0.09 -0.03 0.02 -0.14 1.00 0.10 -0.02 0.05 0.05 0.05 -0.01 -0.08 -0.14 0.14 -0.02 0.03 0.04
famsup_num -0.14 0.18 0.19 -0.00 0.15 -0.06 0.02 0.01 -0.02 -0.03 -0.09 0.03 0.02 -0.06 0.10 1.00 0.29 -0.00 0.06 0.10 0.10 0.01 -0.16 0.15 -0.02 0.11 -0.02
paid_num -0.04 0.16 0.09 -0.07 0.17 -0.19 0.00 -0.06 0.01 0.06 0.06 -0.08 0.01 0.09 -0.02 0.29 1.00 -0.02 0.10 0.19 0.15 0.01 -0.02 0.13 -0.05 0.01 -0.05
activities_num -0.10 0.11 0.11 -0.01 0.09 -0.07 0.04 0.09 0.05 -0.07 -0.04 0.02 -0.01 0.04 0.05 -0.00 -0.02 1.00 0.00 0.10 0.05 0.02 -0.12 -0.10 0.05 0.00 -0.10
nursery_num -0.09 0.19 0.16 -0.03 0.08 -0.10 -0.00 -0.02 0.00 -0.08 -0.10 -0.02 0.02 0.07 0.05 0.06 0.10 0.00 1.00 0.05 0.01 0.03 -0.09 0.01 -0.06 -0.10 0.09
higher_num -0.21 0.17 0.17 -0.08 0.18 -0.30 0.02 -0.06 0.04 -0.07 -0.10 -0.02 0.06 0.19 0.05 0.10 0.19 0.10 0.05 1.00 0.02 -0.11 -0.02 0.15 -0.04 0.01 0.04
internet_num -0.11 0.20 0.13 -0.11 0.06 -0.06 0.03 0.05 0.07 0.04 0.01 -0.08 0.10 0.10 -0.01 0.10 0.15 0.05 0.01 0.02 1.00 0.09 -0.13 -0.04 -0.22 -0.00 -0.07
romantic_num 0.16 0.04 0.02 0.02 0.05 0.09 -0.06 -0.01 0.01 0.02 -0.01 0.03 0.15 -0.10 -0.08 0.01 0.01 0.02 0.03 -0.11 0.09 1.00 0.06 0.10 -0.01 -0.03 0.04
school_num 0.38 -0.13 -0.08 0.24 -0.09 0.06 -0.05 0.03 -0.01 0.11 0.07 -0.04 -0.09 0.04 -0.14 -0.16 -0.02 -0.12 -0.09 -0.02 -0.13 0.06 1.00 0.01 0.28 -0.06 -0.05
sex_num 0.03 -0.08 -0.03 -0.06 0.31 -0.04 -0.06 -0.24 -0.08 -0.27 -0.27 -0.14 0.07 -0.10 0.14 0.15 0.13 -0.10 0.01 0.15 -0.04 0.10 0.01 1.00 -0.03 0.09 0.02
address_num 0.15 -0.14 -0.07 0.33 0.02 0.08 -0.01 -0.03 -0.07 0.09 0.10 0.04 0.03 -0.11 -0.02 -0.02 -0.05 0.05 -0.06 -0.04 -0.22 -0.01 0.28 -0.03 1.00 0.07 -0.04
famsize_num -0.04 0.04 0.06 -0.06 0.07 0.02 0.02 -0.02 -0.02 -0.10 -0.10 0.03 -0.04 -0.08 0.03 0.11 0.01 0.00 -0.10 0.01 -0.00 -0.03 -0.06 0.09 0.07 1.00 -0.15
Pstatus_num -0.03 0.12 0.09 -0.03 -0.02 0.00 -0.03 -0.04 -0.00 0.03 -0.01 -0.02 0.13 0.04 0.04 -0.02 -0.05 -0.10 0.09 0.04 -0.07 0.04 -0.05 0.02 -0.04 -0.15 1.00
age Medu Fedu traveltime studytime failures famrel freetime goout Dalc Walc health absences Grade schoolsup_num famsup_num paid_num activities_num nursery_num higher_num internet_num romantic_num school_num sex_num address_num famsize_num Pstatus_num
0.8
0.4
0.0
−0.4
−0.8

Figure 10

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.

```
Naive Bayes

Number of columns:  43

Split 395 rows into train=286 and test=109 rows

class_value    :  1

# of instances :  200

class_value    :  2

# of instances :  55

class_value    :  3

# of instances :  18

class_value    :  4

# of instances :  5

class_value    :  5

# of instances :  8

Accuracy: 54.128440367%
```



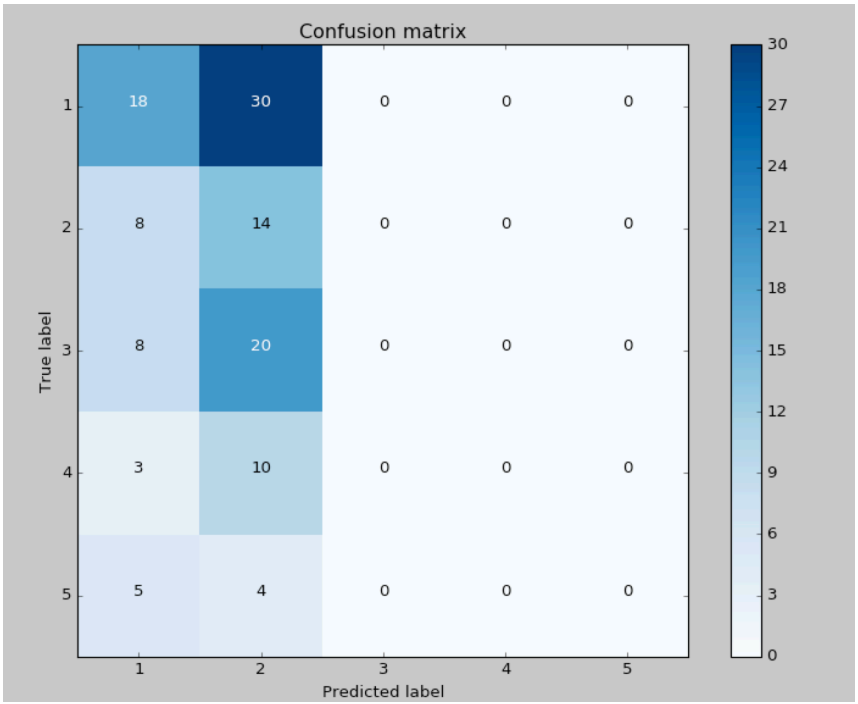
Weka Result:

```
Correctly Classified Instances          78               58.209  %
Incorrectly Classified Instances        56               41.791  %
Kappa statistic                          0.138
Mean absolute error                      0.1684
Root mean squared error                  0.3339
Relative absolute error                 87.8592 %
Root relative squared error            107.7049 %
Total Number of Instances              134

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.772    0.524    0.763      0.772    0.768      0.250    0.706     0.841     1
                 0.214    0.123    0.316      0.214    0.255      0.107    0.650     0.369     2
                 0.100    0.113    0.067      0.100    0.080      -0.011   0.546     0.179     3
                 0.000    0.023    0.000      0.000    0.000      -0.023   0.601     0.044     4
                 0.000    0.030    0.000      0.000    0.000      -0.015   0.962     0.167     5
Weighted Avg.    0.582    0.394    0.595      0.582    0.586      0.192    0.682     0.670

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 71 11  7  2  1 |  a = 1
 17  6  4  0  1 |  b = 2
  5  1  1  1  2 |  c = 3
  0  1  2  0  0 |  d = 4
  0  0  1  0  0 |  e = 5
```

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.

```
Naive Bayes

Number of columns:  43

Split 395 rows into train=275 and test=120 rows

class_value    :  1

# of instances :  103

class_value    :  2

# of instances :  63

class_value    :  3

# of instances :  52

class_value    :  4

# of instances :  38

class_value    :  5

# of instances :  19

Accuracy: 26.6666666667%
```



Weka Result:

```
Correctly Classified Instances          46               34.3284 %
Incorrectly Classified Instances        88               65.6716 %
Kappa statistic                          0.1291
Mean absolute error                      0.2676
Root mean squared error                  0.4201
Relative absolute error                 89.1965 %
Root relative squared error            107.9933 %
Total Number of Instances              134

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.480    0.286    0.500      0.480    0.490      0.196    0.665     0.538     1
                 0.364    0.295    0.195      0.364    0.254      0.055    0.543     0.225     2
                 0.206    0.160    0.304      0.206    0.246      0.053    0.514     0.286     3
                 0.200    0.079    0.308      0.200    0.242      0.146    0.711     0.333     4
                 0.375    0.048    0.333      0.375    0.353      0.310    0.697     0.327     5
Weighted Avg.    0.343    0.210    0.362      0.343    0.344      0.136    0.616     0.380

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 24 17  7  2  0 |  a = 1
  9  8  2  2  1 |  b = 2
  9  9  7  5  4 |  c = 3
  5  4  6  4  1 |  d = 4
  1  3  1  0  3 |  e = 5
```

Logistic Regression:

Work Day Alcohol Consumption Results:

```
  NIT   NF   F                             GTG
    0    1  6.931471805599453E-01   1.40183624E+00
tnc: stepmx = 1000
    1    5 -5.749251696342640E+00   3.30718691E-01
    2   42 -1.543128536091840E+01   3.30652139E-01
tnc: |fn-fn-1] = 0 -> convergence
    3   96 -1.543128536091840E+01   3.30652139E-01
tnc: Converged (|f_n-f_(n-1)| ~= 0)

accuracy = 69.8734177215%


   1     | 2     | 3     | 4     | 5    |<--- predicted
-----------------------------------------
1| 276   | 0     | 0     | 0     | 0    |
-----------------------------------------
2| 75    | 0     | 0     | 0     | 0    |
-----------------------------------------
3| 26    | 0     | 0     | 0     | 0    |
-----------------------------------------
4| 9     | 0     | 0     | 0     | 0    |
-----------------------------------------
5| 9     | 0     | 0     | 0     | 0    |
-----------------------------------------
```

Weekend Alcohol Consumption Results:

```
  NIT   NF   F                          GTG
    0    1  6.931471805599453E-01   4.76684826E+00
tnc: stepmx = 1000
    1    5 -1.574174759383600E+01   2.47826843E+00
    2   44 -4.175939962573737E+01   2.47807082E+00
tnc: |fn-fn-1] = 0 -> convergence
    3   98 -4.175939962573737E+01   2.47807082E+00
tnc: Converged (|f_n-f_(n-1)| ~= 0)

accuracy = 38.2278481013%


   1     | 2     | 3     | 4     | 5     |<--- predicted
-----------------------------------------
1| 151   | 0     | 0     | 0     | 0     |
-----------------------------------------
2| 85    | 0     | 0     | 0     | 0     |
-----------------------------------------
3| 80    | 0     | 0     | 0     | 0     |
-----------------------------------------
4| 51    | 0     | 0     | 0     | 0     |
-----------------------------------------
5| 28    | 0     | 0     | 0     | 0     |
-----------------------------------------
```

Weka Result:

```
Correctly Classified Instances          84               62.6866 %
Incorrectly Classified Instances        50               37.3134 %
Kappa statistic                          0.2507
Mean absolute error                      0.1762
Root mean squared error                  0.3614
Relative absolute error                 91.9111 %
Root relative squared error            116.5759 %
Total Number of Instances              134

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.837    0.381    0.828      0.837    0.832      0.459    0.749     0.847     1
                 0.143    0.075    0.333      0.143    0.200      0.096    0.481     0.249     2
                 0.300    0.048    0.333      0.300    0.316      0.264    0.647     0.213     3
                 0.000    0.046    0.000      0.000    0.000      -0.033   0.634     0.047     4
                 0.000    0.105    0.000      0.000    0.000      -0.030   0.624     0.020     5
Weighted Avg.    0.627    0.283    0.663      0.627    0.637      0.354    0.682     0.650

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 77  6  1  2  6 |  a = 1
 14  4  4  3  3 |  b = 2
  2  1  3  1  3 |  c = 3
  0  0  1  0  2 |  d = 4
  0  1  0  0  0 |  e = 5
```

Dataset 3

Dataset Name: Caravan Insurance

Dataset Description:

This data set used in the CoIL 2000 Challenge contains information on customers of an insurance company. The data consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes [3].

Analysis & pre-processing:

This dataset has more volume of data and hence we decided to subset of data for better analysis and comparison. We did feature selection in Weka for this dataset. Hence, we ran algorithm with all the columns and again we ran algorithms with the selected columns both in Weka as well as our code. Apart from three algorithms, we also ran J48 and RandomForest algorithms as well.

Feature Selection:

We used Weka's feature selection functionality. The dataset has 86 columns. We used CfsSubsetEval attribute evaluation method as it values subsets that correlate highly with the class value and low correlation with each other. Figure x shows the output of the feature selection for this dataset.

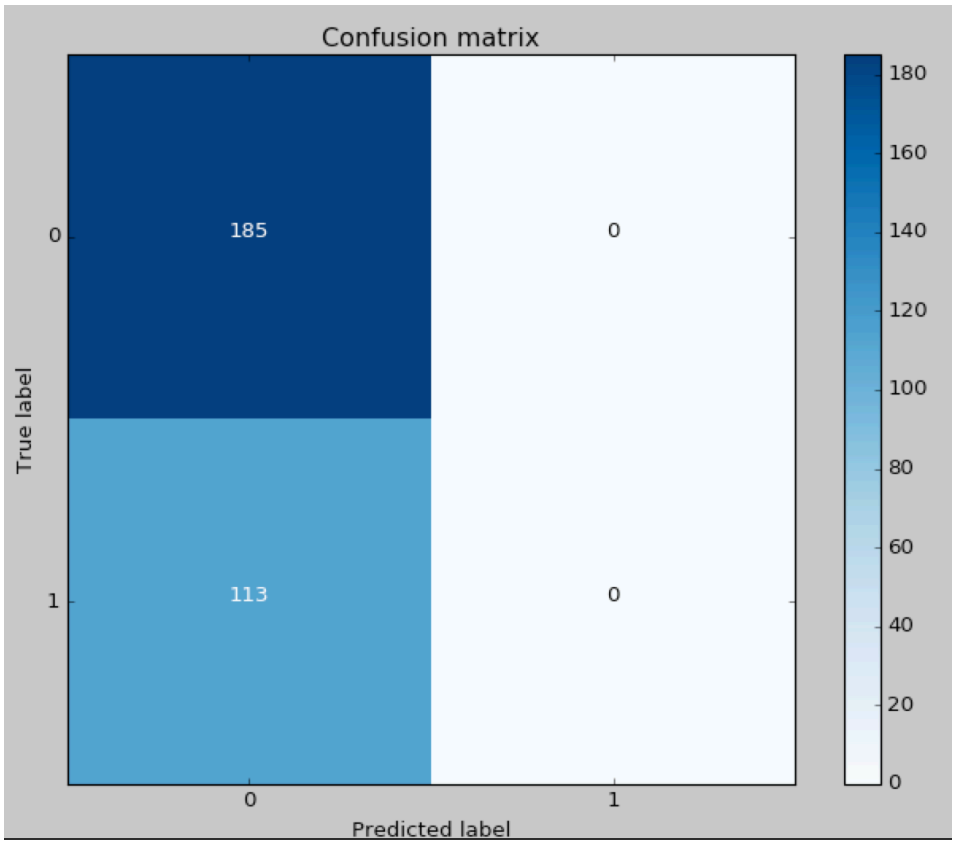
```
Attribute Subset Evaluator (supervised, Class (nominal): 86 CARAVAN):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 1,3,12,25,39,42,43,44,46,47,55,59,61,68 : 14
                     MOSTYPE
                     MGEMOMV
                     MRELOV
                     MSKA
                     MINK4575
                     MINKGEM
                     MKOOPKLA
                     PWAPART
                     PWALAND
                     PPERSAUT
                     PLEVEN
                     PBRAND
                     PPLEZIER
                     APERSAUT
```

Figure 11

Naïve Bayes:

Results before Attribute Selection:

```
Naive Bayes

Number of columns:  86

Split 947 rows into train=649 and test=298 rows

class_value    :  0

# of instances :  415

class_value    :  1

# of instances :  234

Accuracy: 62.0805369128%
```



Results after Attribute Selection:

```
Naive Bayes

Number of columns:  15

Split 837 rows into train=587 and test=250 rows

class_value    :  0

# of instances :  413

class_value    :  1

# of instances :  174

Accuracy: 74.8%
```

Figure 12

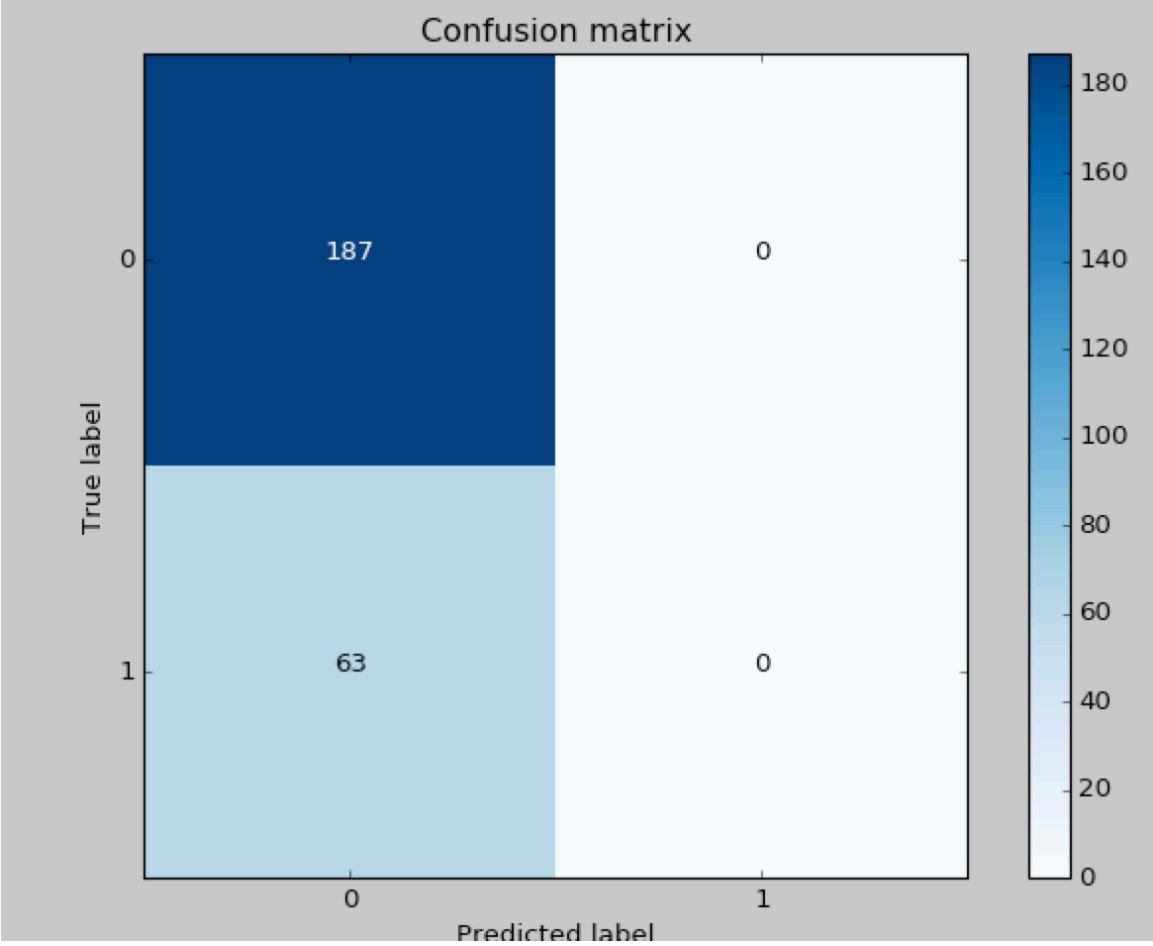


Figure 13

Weka:

Results before Attribute Selection:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         553               65.9905 %
Incorrectly Classified Instances       285               34.0095 %
Kappa statistic                          0.2164
Mean absolute error                      0.3455
Root mean squared error                  0.5417
Relative absolute error                 84.889  %
Root relative squared error            120.1366 %
Total Number of Instances              838

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.718    0.487    0.788      0.718    0.752      0.219    0.678     0.830
                 0.513    0.282    0.419      0.513    0.461      0.219    0.678     0.465
Weighted Avg.    0.660    0.429    0.683      0.660    0.669      0.219    0.678     0.726

=== Confusion Matrix ===

   a   b   <-- classified as
 431 169 |   a = 0
 116 122 |   b = 1
```

Figure 14

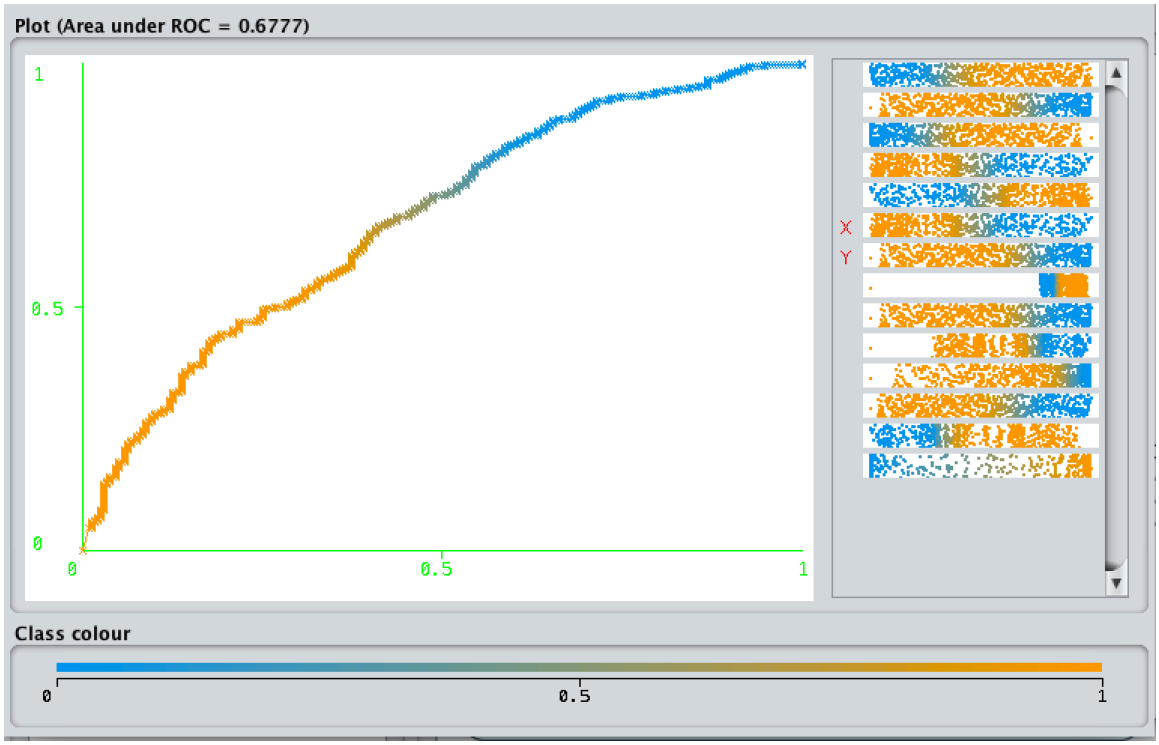


Figure 15

Results after Attribute Selection:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         598               71.3604 %
Incorrectly Classified Instances       240               28.6396 %
Kappa statistic                          0.2904
Mean absolute error                      0.3244
Root mean squared error                  0.4596
Relative absolute error                 79.7222 %
Root relative squared error            101.9164 %
Total Number of Instances              838

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.805    0.517    0.797      0.805    0.801      0.290    0.707     0.843
                 0.483    0.195    0.496      0.483    0.489      0.290    0.707     0.488
Weighted Avg.    0.714    0.425    0.711      0.714    0.712      0.290    0.707     0.742

=== Confusion Matrix ===

   a   b   <-- classified as
 483 117 |   a = 0
 123 115 |   b = 1
```

Figure 16

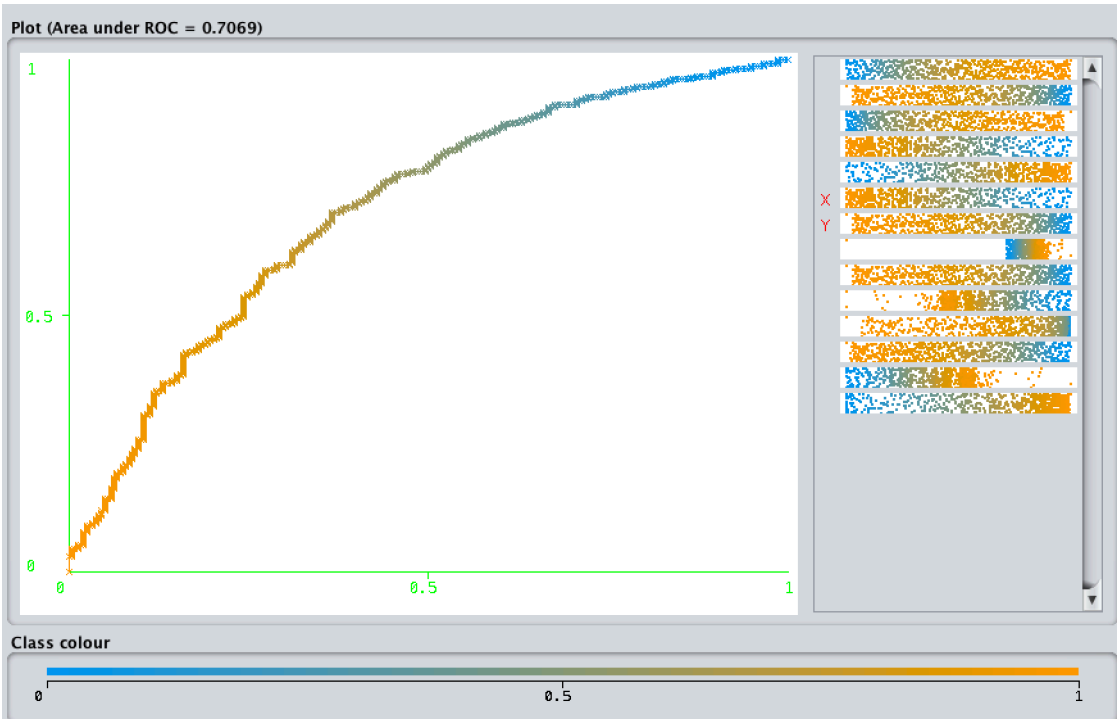


Figure 17

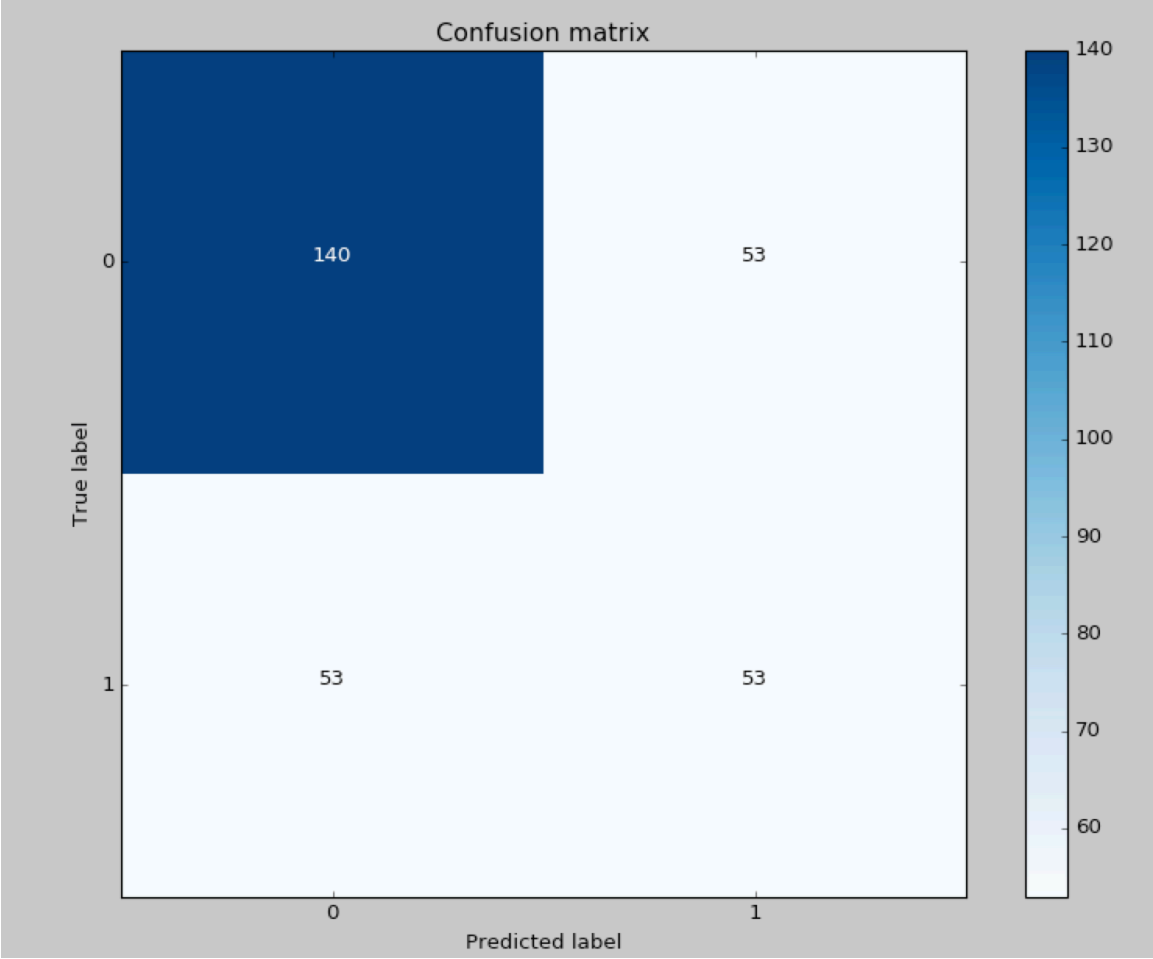
Observations:

We could see the importance of feature selection as there is an increase in the accuracy. The initial dataset shows 65% accuracy whereas the feature selected data shows 71% accuracy. We also observed that our implementation classifies most of them in the majority class, which implies that there is an opportunity to improve the implementation. We could also notice the improvement in ROC curve after feature selection is done.

Logistic Regression:

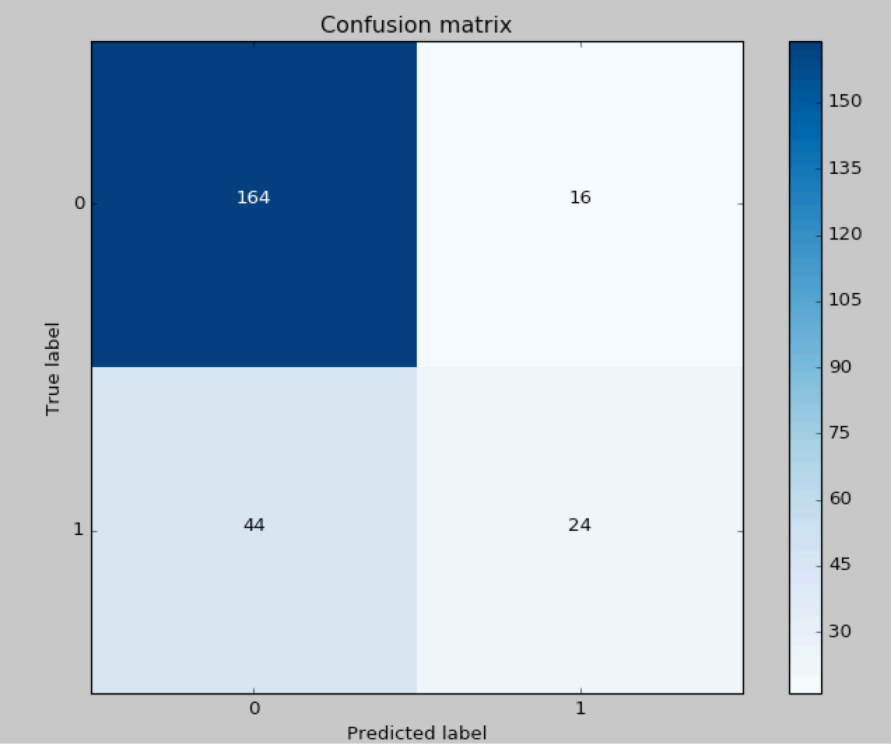
Results before Attribute Selection:

```
Number of columns:  86

Split 947 rows into train=648 and test=299 rows
  NIT   NF   F                        GTG
    0    1  6.931471805599453E-01   2.65848694E+01
    1    6  5.580914770664417E-01   6.57335587E-01
    2   13  5.186088531795111E-01   1.89490597E-02
    3   20  5.078626137256140E-01   9.14736724E-02
    4   33  4.985482516008095E-01   7.64731265E-04
tnc: fscale = 36.1614
    5   40  4.978070017951317E-01   4.05520288E-04
    6   49  4.938871534031049E-01   2.82623078E-02
    7   58  4.892935536116020E-01   1.66337133E-02
    8   67  4.874752047524525E-01   5.24642993E-04
    9   70  4.872951233248895E-01   1.79303647E-03
   10   79  4.855634654655550E-01   2.31803922E-03
   11   88  4.843688689559685E-01   4.23464298E-04
   12   97  4.836426355518541E-01   1.07599475E-03
   13  114  4.826615807385363E-01   1.20152551E-03
   14  131  4.822687030566191E-01   7.73536461E-04
   15  141  4.820765290055486E-01   5.40184148E-04
   16  152  4.818828885215704E-01   3.32402063E-05
   17  163  4.817638882704077E-01   7.30843608E-04
   18  173  4.817058036349773E-01   1.85821064E-04
   19  199  4.815788054937769E-01   8.18456879E-06
   20  223  4.815134215014409E-01   5.80554009E-05
   21  233  4.815022390859386E-01   3.77701008E-05
   22  243  4.814831361740699E-01   6.18054237E-06
   23  282  4.814036050656041E-01   4.55315122E-05
   24  292  4.813925245503271E-01   9.96078501E-05
   25  303  4.813745771969914E-01   3.29168444E-05
   26  314  4.813506382160970E-01   3.75558941E-05
   27  323  4.813377257240205E-01   2.22919857E-05
   28  340  4.812972188868376E-01   3.26393344E-05
   29  351  4.812801504552149E-01   5.71756643E-05
   30  362  4.812615630533085E-01   1.30420549E-05
tnc: stepmx = 1000
   31  401  4.811647979065000E-01   6.68961759E-06
   32  418  4.811455596610498E-01   1.63475078E-05
   33  444  4.809815957152553E-01   5.19356370E-04
   34  453  4.809494397898437E-01   1.53517299E-04
   35  474  4.808488854497731E-01   2.05173120E-04
   36  484  4.807971846127403E-01   2.37116651E-04
   37  511  4.805682296350736E-01   4.53018521E-04
   38  523  4.804847296913786E-01   1.90848378E-05
   39  534  4.804058670127000E-01   7.61840727E-05
   40  545  4.803559933612098E-01   7.66453230E-05
   41  582  4.800166944225616E-01   4.66129626E-04
   42  616  4.799738781448623E-01   1.77841858E-04
tnc: |fn-fn-1] = 3.76292e-12 -> convergence
   43  657  4.799738781410993E-01   1.77841851E-04
tnc: Converged (|f_n-f_(n-1)| ~= 0)

accuracy = 64.5484949833%
```



Results after Attribute Selection:

```
Number of columns:  15

Split 837 rows into train=589 and test=248 rows
  NIT   NF   F                      GTG
    0    1  6.931471805599453E-01   3.59141995E+01
    1    4  5.787377435072255E-01   4.74059620E-01
    2   11  5.433675544643787E-01   7.29134135E-02
    3   14  5.411563956411107E-01   5.86620684E-03
    4   22  5.260855668521572E-01   1.63117349E-02
    5   29  5.241782620717383E-01   1.96847330E-03
tnc: fscale = 22.539
    6   35  5.240540938575907E-01   1.04414596E-03
    7   43  5.238819915768296E-01   1.72551468E-04
    8   49  5.238218077420342E-01   3.86602234E-05
    9   52  5.238122563420722E-01   3.98682353E-05
   10   59  5.237796778554140E-01   5.03688745E-05
   11   63  5.237756946107465E-01   9.16112359E-07
tnc: fscale = 1044.78
   12   69  5.237746509708392E-01   3.22344812E-06
   13   77  5.237683504560067E-01   8.40259094E-06
   14   81  5.237670027184276E-01   1.59555698E-07
   15   86  5.237660190215380E-01   1.03088212E-07
   16   89  5.237659049286932E-01   1.88032413E-07
   17   95  5.237655811233682E-01   5.62281862E-07
   18   98  5.237655395197561E-01   7.89944300E-07
   19  104  5.237654305346399E-01   3.77678198E-07
   20  111  5.237653757641200E-01   7.29010678E-08
   21  118  5.237653552539970E-01   7.74100974E-08
tnc: |fn-fn-1] = 1.07255e-08 -> convergence
   22  123  5.237653445284817E-01   1.13966668E-08
tnc: Converged (|f_n-f_(n-1)| ~= 0)

accuracy = 75.8064516129%
```



Observations:

We could see the importance of feature selection as there is an increase in the accuracy. The initial dataset shows 64% accuracy whereas the feature selected data shows 75% accuracy on our python implementation.

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.

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.

Weka Result:

```
Correctly Classified Instances         607               64.0295 %
Incorrectly Classified Instances       341               35.9705 %
Kappa statistic                          0.2264
Mean absolute error                      0.3992
Root mean squared error                  0.5073
Relative absolute error                 85.8876 %
Root relative squared error            105.244  %
Total Number of Instances              948

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.715    0.489    0.716      0.715    0.716      0.226    0.644     0.726     0
                 0.511    0.285    0.510      0.511    0.511      0.226    0.644     0.481     1
Weighted Avg.    0.640    0.414    0.641      0.640    0.640      0.226    0.644     0.636

=== Confusion Matrix ===

   a   b   <-- classified as
 429 171 |   a = 0
 170 178 |   b = 1
```

Figure 18

J48 – Weka Result:

```
Correctly Classified Instances         608               72.5537 %
Incorrectly Classified Instances       230               27.4463 %
Kappa statistic                          0.2716
Mean absolute error                      0.3181
Root mean squared error                  0.4879
Relative absolute error                 78.1617 %
Root relative squared error            108.1973 %
Total Number of Instances              838

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.857    0.605    0.781      0.857    0.817      0.276    0.591     0.745
                 0.395    0.143    0.522      0.395    0.450      0.276    0.591     0.414
Weighted Avg.    0.726    0.474    0.708      0.726    0.713      0.276    0.591     0.651

=== Confusion Matrix ===

   a   b   <-- classified as
 514  86 |   a = 0
 144  94 |   b = 1
```

Figure 19

Random Forest – Weka Result:

```
Correctly Classified Instances         613               73.1504 %
Incorrectly Classified Instances       225               26.8496 %
Kappa statistic                          0.2179
Mean absolute error                      0.3434
Root mean squared error                  0.423
Relative absolute error                 84.393  %
Root relative squared error             93.8122 %
Total Number of Instances              838

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.915    0.731    0.759      0.915    0.830      0.241    0.725     0.857
                 0.269    0.085    0.557      0.269    0.363      0.241    0.725     0.493
Weighted Avg.    0.732    0.548    0.702      0.732    0.697      0.241    0.725     0.754

=== Confusion Matrix ===

   a   b   <-- classified as
 549  51 |   a = 0
 174  64 |   b = 1
```

Figure 20

Observations:

The J48 and random forest was performed on the initial dataset. The algorithm performed same on both the datasets (before & after attribute selection). Hence, we did not include that in this section.

Dataset 4

Dataset Name: Pima Indians Diabetes Data Set

Dataset Description:

We collected this dataset from UCI repository. This dataset has 9 attributes including the target class variable. This is a binary classification dataset. All patients here are females at least 21 years old of Pima Indian heritage [4].

Analysis:

Using this dataset, tested our Naïve Bayes implementation and Logistic Regression implementation and compared with the Weka's result. We also did analysis on decision tree and random forest algorithms. Since this dataset had all the features in either Integer or of float type, we did not do any pre-processing. However, we performed the feature correlation analysis. Figure 19 shows the feature correlation analysis chart. We could see that there is no major correlation between variables which is good sign that the models will perform good on this dataset.

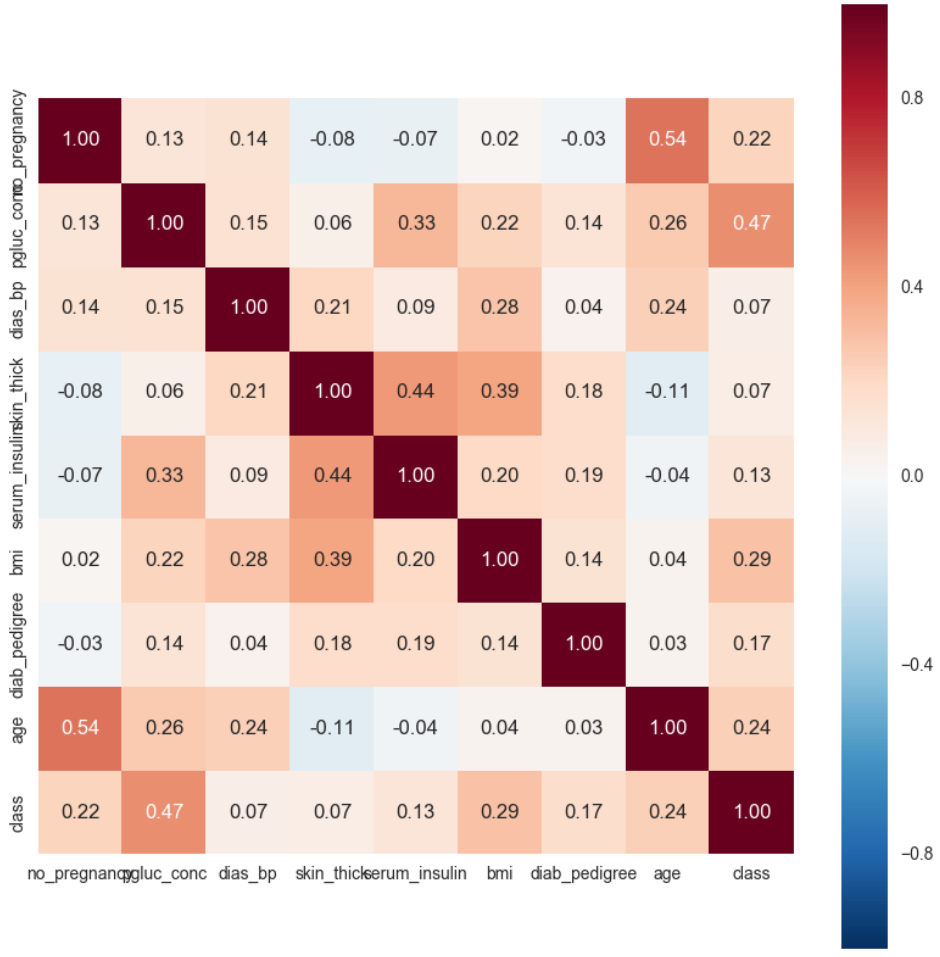


Figure 21

Naïve Bayes:

```
Number of columns:  9

Split 767 rows into train=534 and test=233 rows

class_value    :  0

# of instances :  349

class_value    :  1

# of instances :  185

Accuracy: 72.9613733906%
```

Figure 23

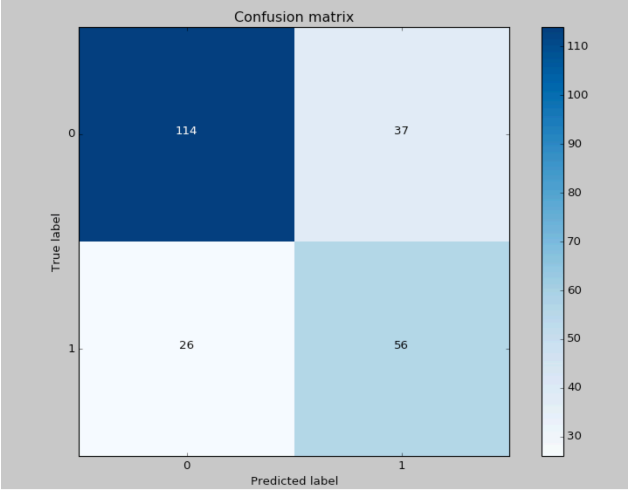


Figure 22

Weka Result:

```
=== Summary ===

Correctly Classified Instances         584               76.1408 %
Incorrectly Classified Instances       183               23.8592 %
Kappa statistic                          0.4626
Mean absolute error                      0.2859
Root mean squared error                  0.4201
Relative absolute error                 62.9797 %
Root relative squared error             88.1767 %
Total Number of Instances              767

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area
                 0.842    0.390    0.802      0.842    0.821      0.464    0.815
                 0.610    0.158    0.674      0.610    0.640      0.464    0.815
Weighted Avg.    0.761    0.309    0.757      0.761    0.758      0.464    0.815

=== Confusion Matrix ===

   a   b   <-- classified as
 421  79 |   a = 0
 104 163 |   b = 1
```

Figure 24

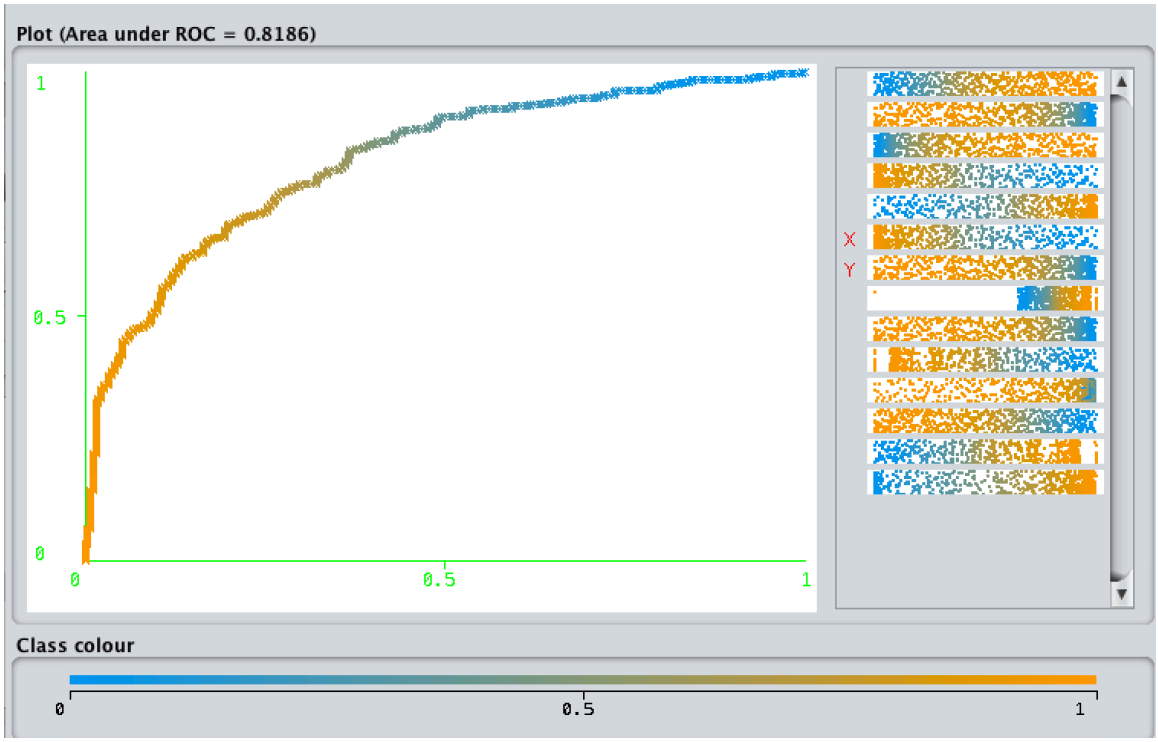


Figure 25

Observations:

We could see that Naïve Bayes performed reasonably well on this dataset given the reason that there is no major correlation between features. However, we can do standardization of variables to have all the features on same scale which may improve the performance of the algorithm.

Logistic Regression:

```
Number of columns:  9

Split 767 rows into train=517 and test=250 rows
  NIT   NF   F                         GTG
    0    1  6.931471805599453E-01   3.46222875E+02
    1    6  6.095366152062257E-01   1.47398790E+00
    2   11  5.776791716182732E-01   2.73089472E+01
    3   16  4.969761131134418E-01   6.36166658E+00
    4   21  4.755819051950658E-01   2.40012356E+00
    5   24  4.719443913349738E-01   5.09851782E-01
    6   29  4.649990668661332E-01   3.82824037E-03
    7   34  4.647900293389640E-01   1.36348884E-02
    8   38  4.647215067912678E-01   5.13769634E-03
    9   42  4.646893291941960E-01   9.03691307E-03
   10   46  4.646668638566780E-01   7.35450860E-04
tnc: fscale = 36.8742
   11   49  4.646666008381842E-01   3.46088568E-05
   12   54  4.646651780131549E-01   1.29691015E-06
tnc: fscale = 878.102
tnc: |fn-fn-1] = 4.03652e-09 -> convergence
   13   59  4.646651739766313E-01   7.00762229E-09
tnc: Converged (|f_n-f_(n-1)| ~= 0)

accuracy = 79.6%
```

Figure 26

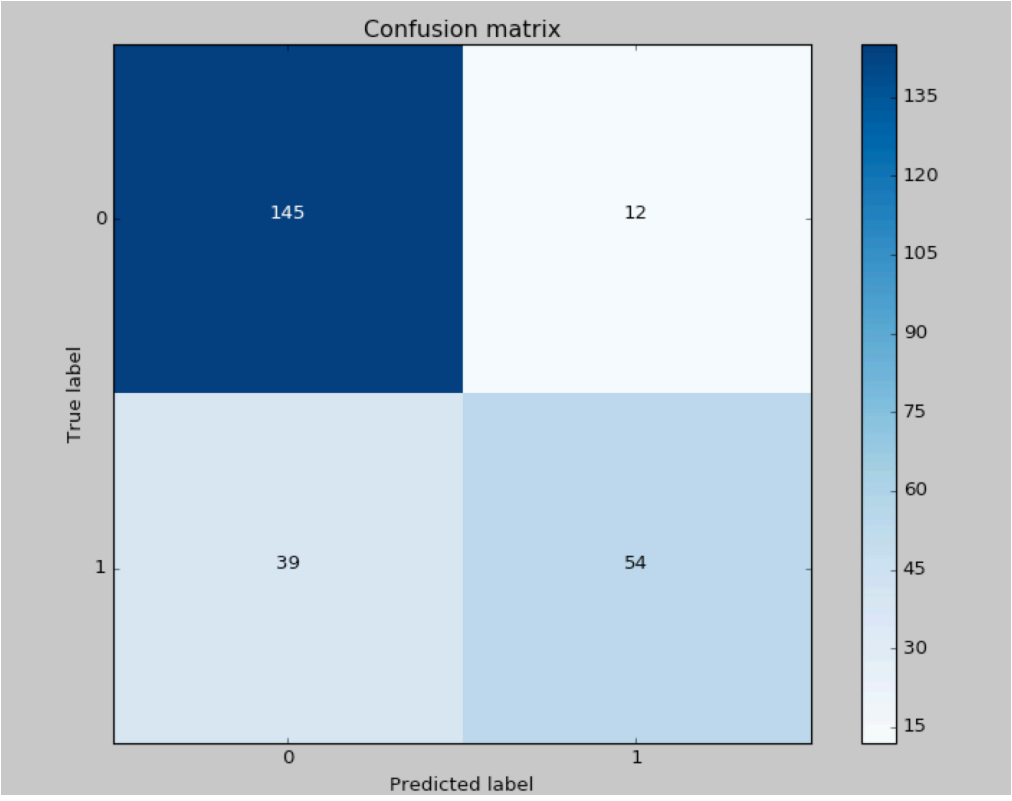


Figure 27

Weka Result:

```
=== Summary ===

Correctly Classified Instances         598               77.9661 %
Incorrectly Classified Instances       169               22.0339 %
Kappa statistic                          0.4882
Mean absolute error                      0.3099
Root mean squared error                  0.3961
Relative absolute error                 68.257  %
Root relative squared error             83.1502 %
Total Number of Instances              767

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area
                 0.890    0.427    0.796      0.890    0.840      0.496    0.830
                 0.573    0.110    0.736      0.573    0.644      0.496    0.830
Weighted Avg.    0.780    0.317    0.775      0.780    0.772      0.496    0.830

=== Confusion Matrix ===

   a   b   <-- classified as
 445  55 |   a = 0
 114 153 |   b = 1
```

Figure 28

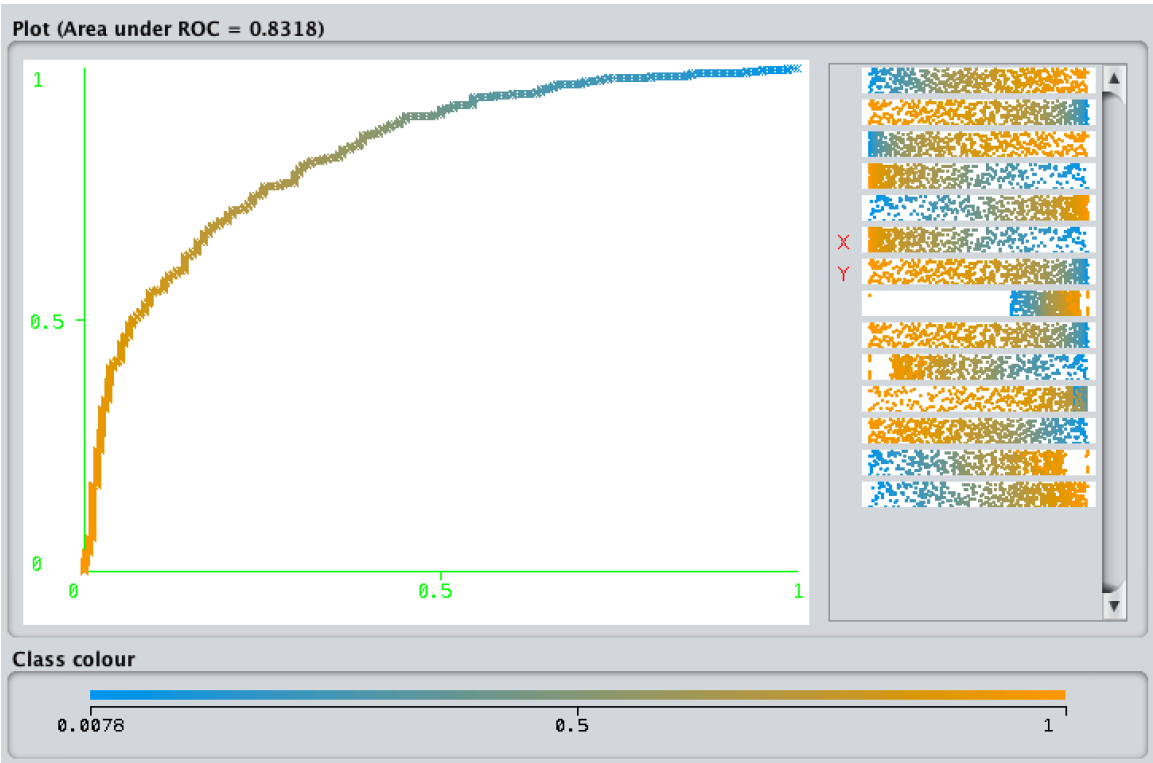


Figure 29

J48 – Weka Result:

```
=== Summary ===

Correctly Classified Instances         567               73.8281 %
Incorrectly Classified Instances       201               26.1719 %
Kappa statistic                          0.4164
Mean absolute error                      0.3158
Root mean squared error                  0.4463
Relative absolute error                 69.4841 %
Root relative squared error             93.6293 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area
                 0.814     0.403     0.790      0.814     0.802      0.417     0.751
                 0.597     0.186     0.632      0.597     0.614      0.417     0.751
Weighted Avg.    0.738     0.327     0.735      0.738     0.736      0.417     0.751

=== Confusion Matrix ===

   a   b   <-- classified as
 407  93 |   a = 0
 108 160 |   b = 1
```

Figure 30

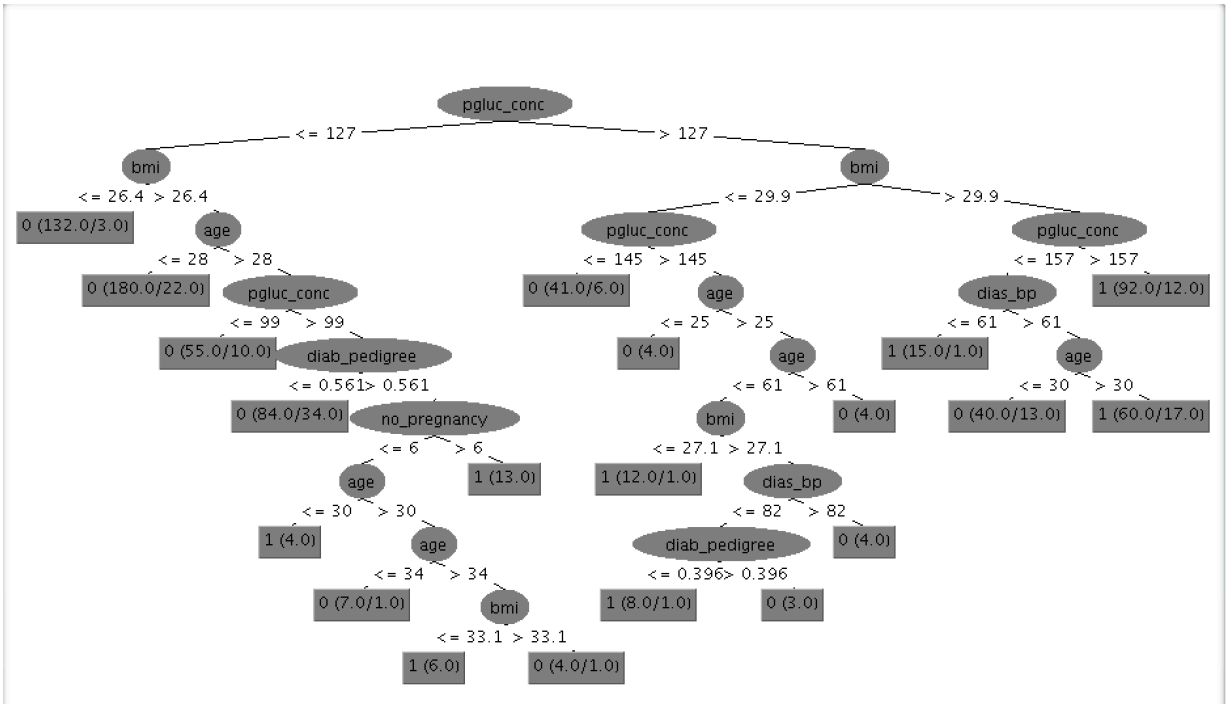


Figure 31

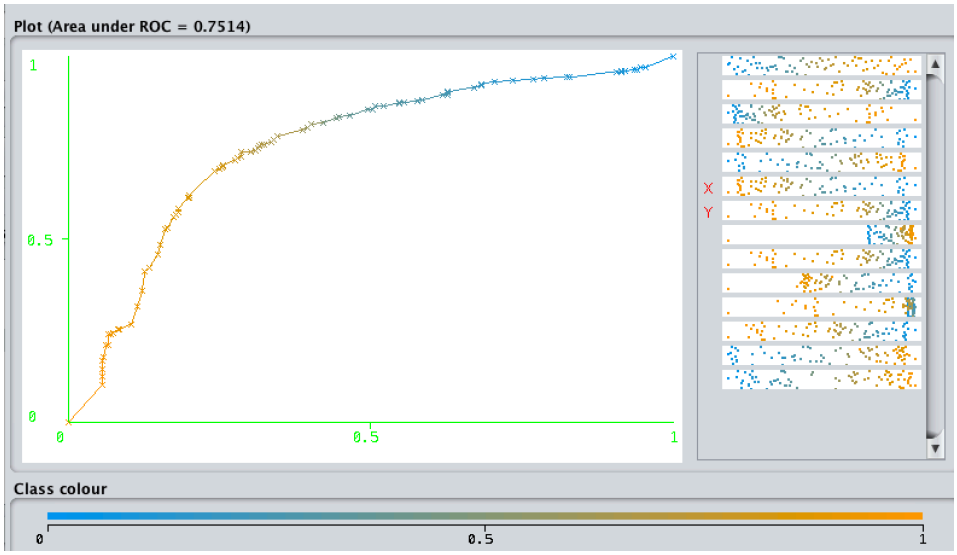


Figure 32

Observations:

The J48 algorithm also performed reasonably well and we are able to visualize the tree as well on how the tree is built for prediction. We have also plotted the ROC curve for better understanding of the performance.

Random Forest:

```
=== Summary ===

Correctly Classified Instances         578               75.2604 %
Incorrectly Classified Instances       190               24.7396 %
Kappa statistic                          0.4459
Mean absolute error                      0.3132
Root mean squared error                  0.4052
Relative absolute error                 68.9163 %
Root relative squared error             85.0156 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area
                 0.830     0.392     0.798      0.830     0.814      0.447    0.818
                 0.608     0.170     0.657      0.608     0.632      0.447    0.818
Weighted Avg.    0.753     0.314     0.749      0.753     0.750      0.447    0.818

=== Confusion Matrix ===

   a   b   <-- classified as
 415  85 |   a = 0
 105 163 |   b = 1
```

Figure 33

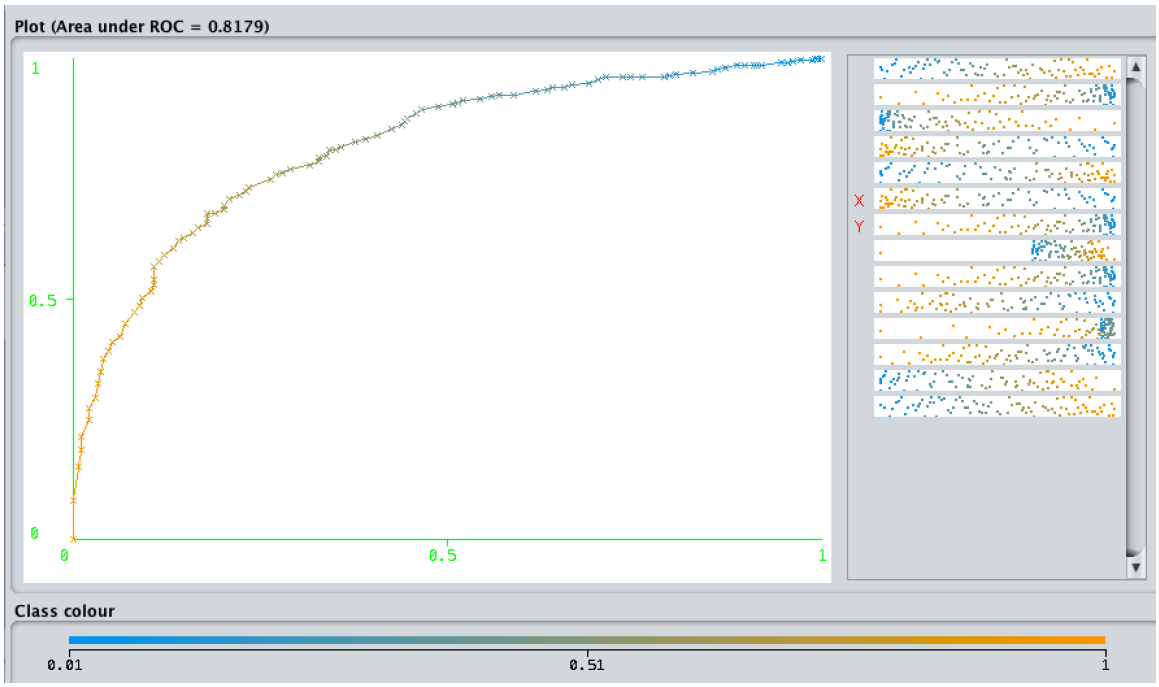


Figure 34

Observations:

From the ROC curve of all the algorithms, we could see that all the algorithms perform reasonably well and shows that the classifier can better separate the positive and negative classes reasonably well.

Code:

We have used Python 2.7 to implement the algorithms. We used PyCharm IDE for development and Jupyter notebook for our analysis. We have included the code as well as the notebooks in the github repo. Github URL is documented at the end of Code section.

Naïve Bayes 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 with the mean and standard deviation calculated in the previous module.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

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.

Logistic Regression Implementation:

The algorithm is implemented in Python. The program is modularized and it accepts two parameters:

1. input file name
2. conversion flag (yes/no). If the target class is having character literals, then providing 'yes' will automatically map it to a integer values.

The assumption for the program is that the last column in the dataset is the target column and also all the columns are of either integer or float type except the target column as it can be handled with the conversion flag. 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.

Convert target string literal to numeric:

If the flag is provided as 'yes' then conversion module will be called that will first generate an integer value for unique target values starting 1. It then the assigns the generated integer value to the target column in all the records.

Split dataset:

We then add a column with value 1 to the dataset at the beginning. This is done in order to simplify the calculation in the cost function. 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.

Cost Function:

We then split the training and test dataset into X and y values. We initialize theta with zeros and we use the below formula to calculate the cost function. The cost function is implemented completely in vectorized form without any regularized term.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i)\log(1 - h_\theta(x^i))$$

where $h_\theta(x)$ is defined as follows

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

And then we using fmin_tnc function to find the global minima theta values.

Prediction:

We then use this the minimized theta to make predictions of the test data.

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.

github URL:

https://github.com/brajaram/aml_spring_2016

github directory view:

```
aml_spring_2016/
├── README.md
├── code
│   ├── LogisticRegression.py
│   ├── knn.py
│   ├── ml_utils.py
│   ├── naive_bayes.py
│   └── pre_process.py
├── data
│   ├── har_processed.arff
│   ├── har_subset.arff
│   ├── student_dalc
│   └── student_walc
├── pynb
│   ├── Pima_Indian_Diabetes_Dataset.ipynb
│   ├── School_Alcohol_Consumption_Analysis.ipynb
│   └── har_preprocessing_book.ipynb
├── report
│   ├── report.docx
│   ├── weka_result
│   │   ├── logistic_har_processed
│   │   └── naive_har_processed
│   └── ~$report.docx
└── report.pdf
```

Figure 35

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.
3. P. van der Putten and M. van Someren (eds) . CoIL Challenge 2000: The Insurance Company Case. Published by Sentient Machine Research, Amsterdam. Also a Leiden Institute of Advanced Computer Science Technical Report 2000-09. June 22, 2000.
4. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.