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.

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.

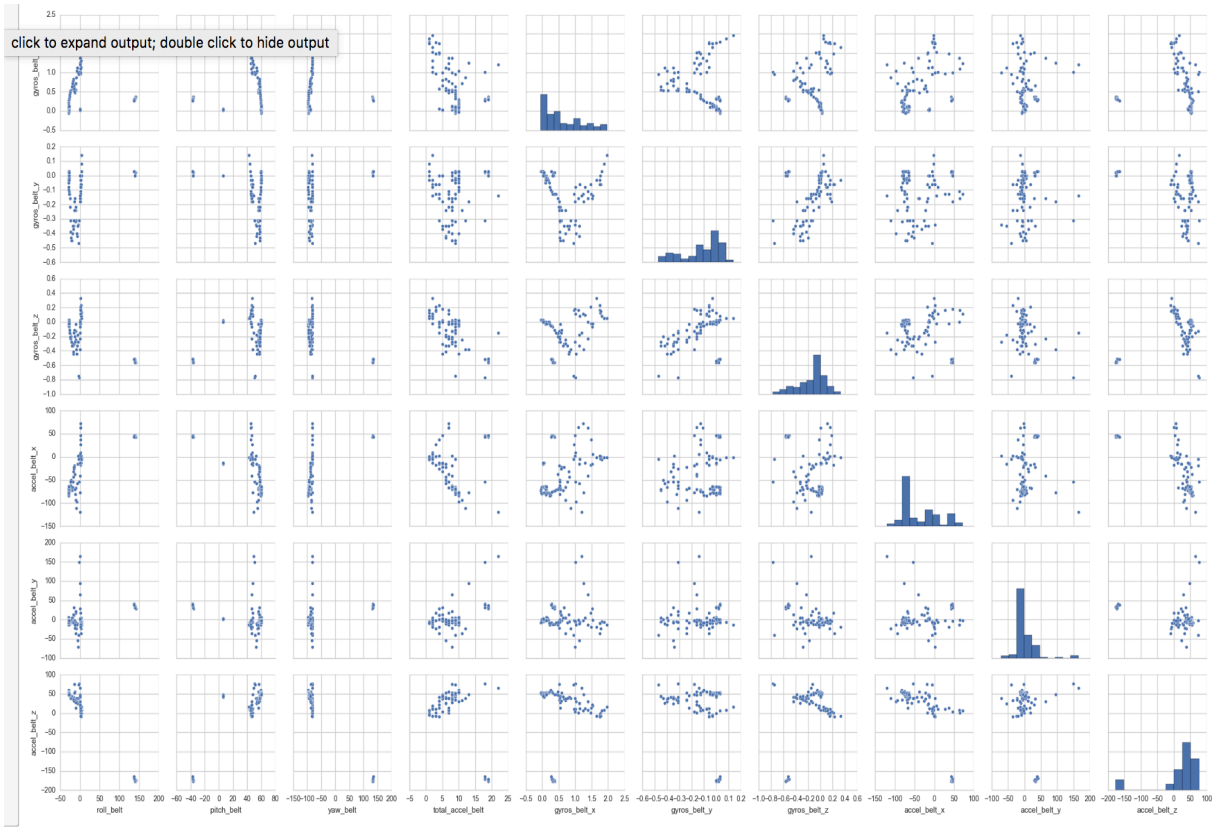


Figure 2

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

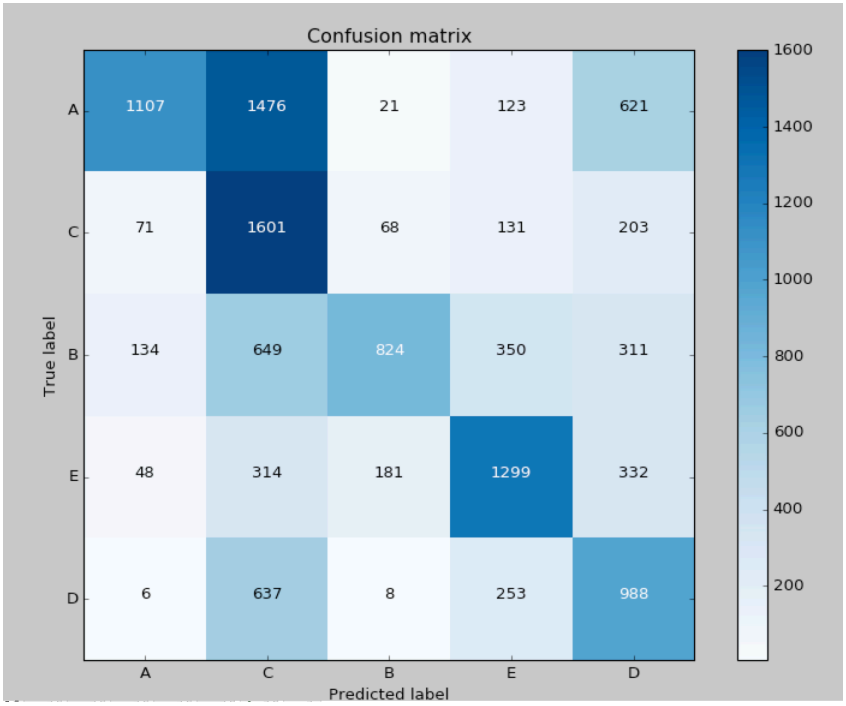


Figure 4

Weka Result:

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:

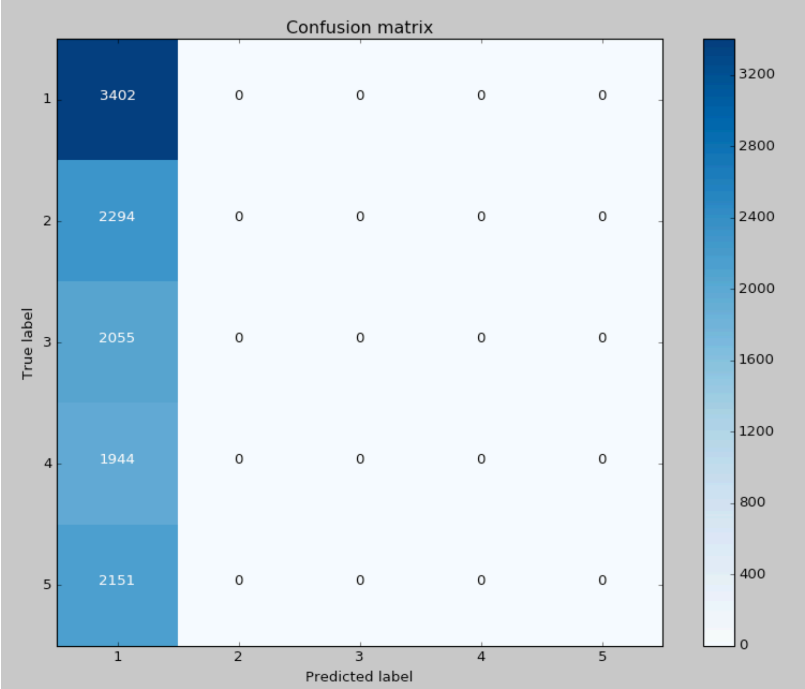
It was clear that the correlation between features made the algorithm to perform poor. We need to focus more on the feature engineering part to identify the variables that have more correlation and measures to remove the correlation by retaining the features that has no correlation and by creating new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

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 5



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.

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.

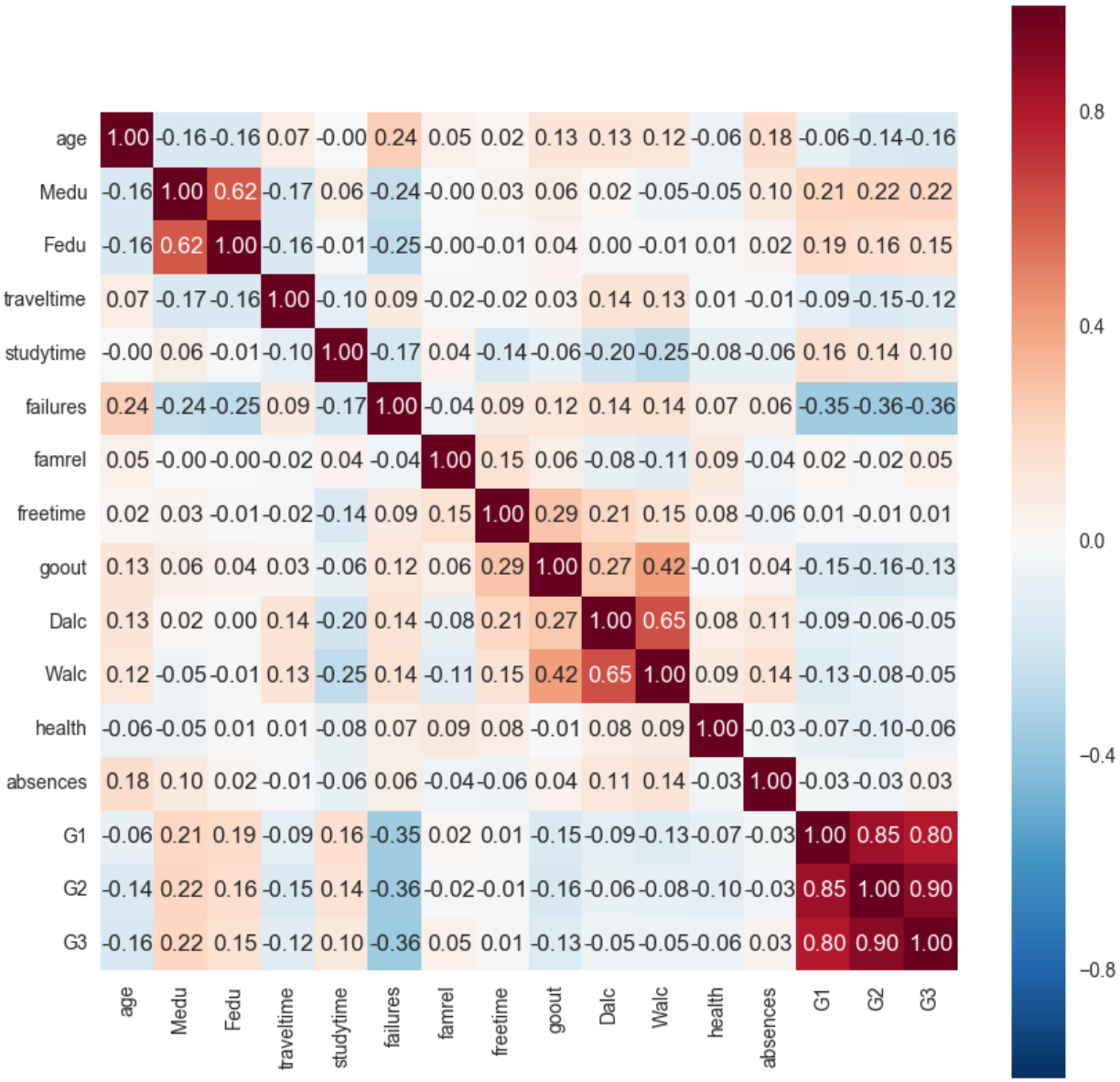

age 1.00 -0.16 -0.16 0.07 -0.00 0.24 0.05 0.02 0.13 0.13 0.12 -0.06 0.18 -0.06 -0.14 -0.16
Medu -0.16 1.00 0.62 -0.17 0.06 -0.24 -0.00 0.03 0.06 0.02 -0.05 -0.05 0.10 0.21 0.22 0.22
Fedu -0.16 0.62 1.00 -0.16 -0.01 -0.25 -0.00 -0.01 0.04 0.00 -0.01 0.01 0.02 0.19 0.16 0.15
traveltime 0.07 -0.17 -0.16 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 6

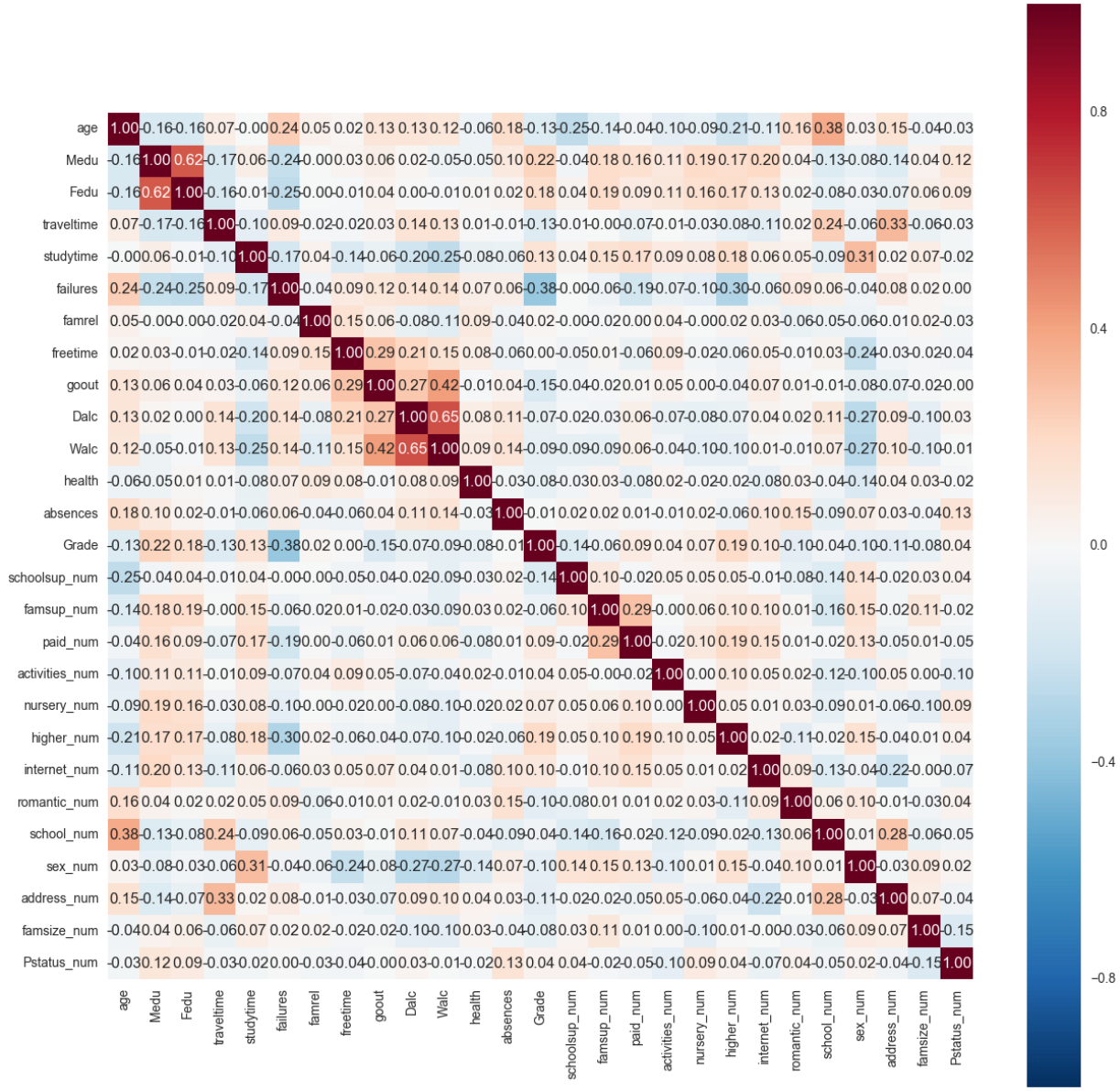
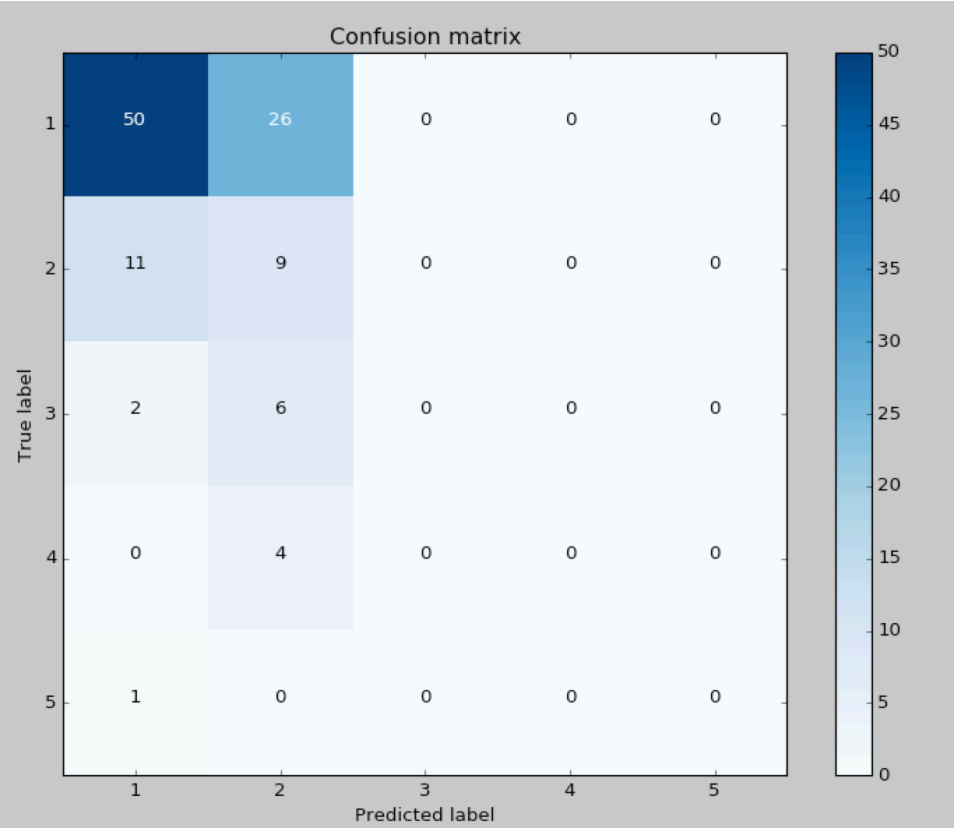

age 1.00 -0.16 -0.16 0.07 -0.00 0.24 0.05 0.02 0.13 0.13 0.12 -0.06 0.18 -0.13 -0.25 -0.14 -0.04 -0.10 -0.09 -0.21 -0.11 0.16 0.38 0.03 0.15 -0.04 -0.03
Medu -0.16 1.00 0.62 -0.17 0.06 -0.24 -0.00 0.03 0.06 0.02 -0.05 -0.05 0.10 0.22 -0.04 0.18 0.16 0.11 0.19 0.17 0.20 0.04 -0.13 -0.08 -0.14 0.04 0.12
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 7

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.

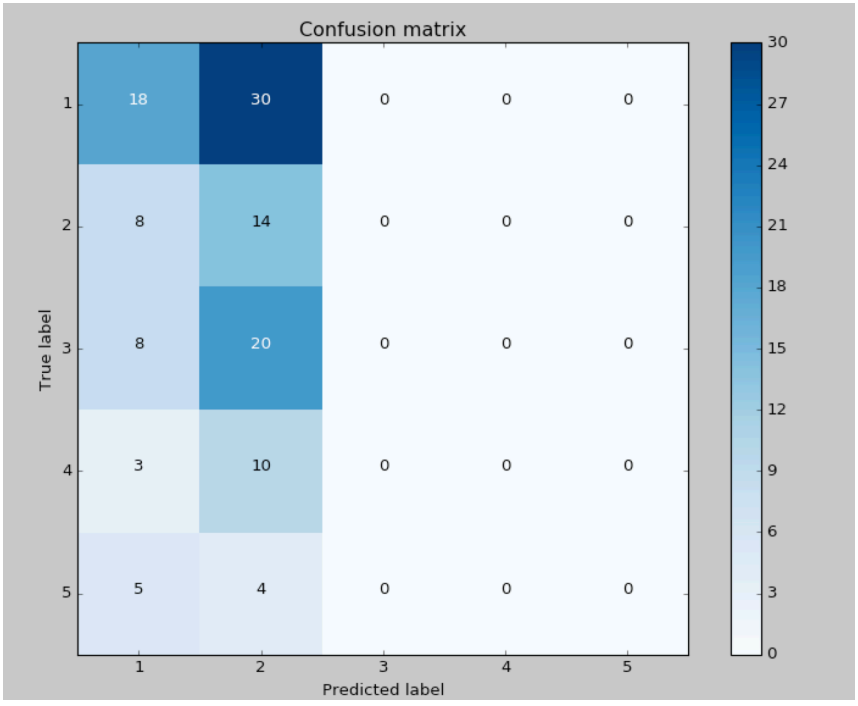
```
Naive Bayes

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%
```



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%
```



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     |
-----------------------------------------
```

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

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.