Data Analytics Coursework (SET09120)

Ismael Souf

40326941

German Credits

Content

1 Introduction

The aim of the coursework is to conduct an exploratory data mining and produce a short report about what is discovered from the data. The data provided is a file which contains historic observations on 10 variables for 1000 past applications for credit. Each applicant was given a rate of "good" and "bad" cases. Based on the applicant's profile a bank can make reasonable decisions about whether or not to award a loan. This report consists of four sections including the introduction. The second part represent the way the data is cleaned by the use of OpenRefine and data conversion. The third part provides different methodology using algorithms in order to analyse the data such as "Classification". Finally, the last part concludes the first three part including the limitations of the case.

2 Data Preparation

2.1 Data Cleaning

Data Cleaning is the process of detecting and correcting corrupt or inaccurate records from a data set. In order to clean our data, we will use OpenRefine, a powerful tool to deal with messy data. This tool can be used to complete four essential tasks such as remove duplicates records, separate multiple values contained in the same field, analyse the distribution of values throughout a data set and group together different representation of the same reality. In this section the steps will be described and illustrated with the credit file given. The credits in German provides loan based on the applicant's profile. Before using OpenRefine we need to add the name of each column provided by the file. Then, on OpenRefine page create a new project using the file credit and click "Next" and you will see 1000 rows. The first step to follow is to go through the data and inspect the different values in facets. Go to a column, click on the box containing a triangle, select "Facet" followed by "Text facet" (See Fig.1). A facet window will appear on the left, you can either sort by count or sort alphabetically. Faceting is an interesting feature that help us get an overview of the data containing text, numbers or dates. In fact, OpenRefine also have many other facets such as "Numeric Facet". This is the first step to clean the data. After faceting the columns, we can find duplicates values and see outliers in certain attributes. The duplicates values are due to spelling mistakes, to fix the spelling mistake we can place the cursor over the text in the facet window and click on "edit" and then fix the error in the text box provided and finally save it by clicking on "Apply" (See Fig.2). For example, we have found in the data a purpose called "Eduction", this is a mistake and should be replace by "Education". Therefore, a more interesting feature is cluster. Clustering function allow us to merge the field with the correct name for repeating records. In our example, the data show the value "business", "busness" and "busines", those attributes must be merged: click on cluster in the facet window, select the three attributes, correct the spelling mistake and click on Merge Selected and Re-Custer (See Fig3). We have now one attribute of business with 97 rows.

Outliers can be removed by showing them in the rows. For example, taking the facet window of the attribute “Age”, it can clearly be seen that the range of Age is -40 to 340 years old (See Fig4). An age cannot be negative. In order to remove the outliers, we can move the facet cursor and have an overview of the corrupted or inaccurate data. Most of the time those outliers are typing error, as it can be seen the age of 222 should be 22 and need to be changed on clicking “edit” and removing a 2 without forgetting to select the right data type (number). OpenRefine allow us different features such as facets, filters and clusters in order to have a quick overview of classic issues found in the German credits. By applying those different features, the file has been cleaned and can now proceed to the Data conversion.

2.2 Data Conversion

In this section of the report, we are going to describe how the German credit data is converted to data sets which can be analysed by algorithms. Different software takes different format of data as input. The first step is to load the cleaned data into Excel and then save it as CSV file. After this step, load this file into Microsoft Word and make the beginning of the ARFF file such as @relation(title), @attribute (data type), @data. A second option is to open the CSV files following few steps in Weka to convert it to ARFF file. Open file, select the save button and write the name of the new project, finally chose the Arff extension: you can now view the data more easily by clicking the “Edit” button in Weka. In order to verify if the dataset has been correctly converted to Arff file we can open it in Word. If the file contains at the beginning declaration of the attributes, the file has been converted successfully otherwise an error occurred go back to the first step. I have made different copies of the dataset during the data conversion. The transform feature allows us to check whether the values are text, numbers. The different steps are illustrated (See Fig5), to convert records, we can select a column for example Credit_amount then select “edit cells” and select “common transform” and choose between the different option such as “To number”. The transform feature can be used with the language python. In our dataset I have decided to transform the column Class from nominal to numeric. This can be done with few steps, “edit cells”, Transform and a window will appear. In this window (See Fig6), select the language Python/Jython. The task is to transform the values on Class from nominal to numeric values such as Good -> 1 and Bad -> 2. The following Python code can be used in the expression box: “if value == "good": return 1 elif value == "bad": return 2”. Similarly, those steps can be done in order to convert numeric values to nominal. Most of the conversion has been done in Weka. In fact, in order to use different algorithm, we need to convert numeric values to nominal values. This can be done in the tab pre-process tab in Weka explorer. We can choose to filter the values with different such as Discretize, if we want to use Association rule Mining in our dataset, we need to convert the Credit_amount, Age or Case_no from numeric type to nominal type (See Fig7). Data conversion is very important and is widely use. Viewing our data in different format can help us find insight that would be impossible to find without conversion. It is important to understand that certain software can be accessed only via specific format.

3 Data Analytics

Four categories of data mining methods were introduced in order to find patterns in dataset. They are also described as four styles of learning commonly appear data mining applications. Through the report we will find patterns using a specific algorithm in each method.

3.1 Classification

In our classification learning, in order to work with our dataset, I have chosen the J48 algorithm. This algorithm generates a decision tree in order to be used for classification. The software Weka allow us to use the algorithm J48 and provides interesting patterns to find and analyse. In our different attributes it has been found that the credit history of applicant's is very relevant to find which applicant are safe to get a loan. The decision tree provided by the algorithm show us that applicant's with existing debts paid back are likely to be good applicants using 5 attributes: Credit_history, Credit_amount,Employment, Age and Class.

J48 Pruned Tree using training set (See Fig8)

IF Credit_history = critical/other existing credit THEN class: "good"
IF Credit_history = existing paid
AND Credit_amount <= 8648 THEN class: "good"
AND Credit_amount > 8648 THEN class: "bad"

The credit_history attribute is an important part (75,9% instances correctly classified) to offer a loan or not. 293 applicants with a critical or another existing credit are classified as "good" applicants and 50 were incorrectly classified. i.e 293 had the class=good and 50 had the class=bad. To be more specific, it depends on the credit amount, because 502 applicants have a credit amount under 8648 euros paid back duly till now. It is important to highlight that the number after the "/" show us that the values are misclassified. The bank considers the applicants as safe to receive a loan. Therefore, more than 40% (293/690 total instance classified as good) of the applicants which are safe to receive a loan have a critical account or an existing credit in another bank. Applicants with all their debts at the bank paid back are likely to be good applicants except a little part considered as "bad" (20%). As we can see in the above rules if the credit_amount of the applicants has been paid back but above 8648 euros they are considered as "bad" applicants.The percentage has been found by selecting the applicants with their class qualified as bad divided by the applicants with all their debts at the bank paid back. Checking the status of an existing account in other banks for the applicants is an interesting attribute to analyse.

Jpruned Tree 3 attributes: Checking_status,Job and Class(See Fig9)

IF Checking_status = <0
AND Job = skilled THEN class: “bad”
AND Job = unskilled resident THEN class: “good “
AND Job = high qualif/self emp/mgmt. THEN class: “good”
AND Job = unemp/unskilled non res THEN class: “bad”
IF Checking_status = 0<=X<200 THEN class: “good”
IF Checking_status = no checking THEN class: “good”
IF Checking_status = >=200 THEN class: “good”

With an overall of 72,2% instance correctly classified the above rules are represented. Applicants with 0 or more than 200 status of existing account are considered as good applicants, they represent more than 26,9% of the applicant. The best section to be safe for a loan is the one without a current account in the bank. 394 applicants without a current account in the bank are considered as “good” applicants which represent 39,4% with 46 applicants misclassified. However, if the status of a current account is less than 0 the category of the applicants is divided in four. Applicants without a current account are only safe if they are unskilled resident or work in management, officer or are self-employee. They are safe to receive a loan and represent half of the applicants qualified as “good” while having a status of less than 0 account. Applicants in that category with a skilled job represent (172 applicants) 63% of applicants with less than 0 account. In other words, the majority applicants with less than 0 current account are considered as “bad” applicants excepts a little part. Above 1 account the applicants are qualified as “good” applicants without having to check their job.

3.2 Association

Apriori algorithm is a breadth first search method to find frequent item sets in a transaction database. In this part we will find different rules of the dataset provided and analyse them to find interesting patterns in order to look for favourable loan towards the applicants. Before we run Apriori algorithm to find association rules, we need to use our dataset converted to Nominal by a filter “NumericToNominal” for the numeric attributes such as Credit_amount. Select “Associate tab” and then I have selected different parameters. Apriori stops once the lowerBoundMinSupport (default 10%) is reached or required number of rules, numRules will have the number of rules to be generated. The different rules will be ranked by metricType by default Confidence. Therefore, the minMetric is 0.90 by default which mean the rules with that confidence are considered and delivered as the output. I have tried two options for the outputItemSets, the first one show all the frequent itemsets by selecting True in the parameter found and the second one False only the show the rules.
I have decided to set up the following configuration (See Fig10):

car: *False*; classIndex: *-1*; delta: *0.05*; lowerBoundMinSupport: *0.0*; metricType: *Confidence*; minMetric: *0.9*; numRules: *10*; outputItemSets:*False*; removeAllMissingCols: *False*; significanceLevel: *-1.0*; upperBoundMinSupport: *1.0*; verbose: *False*

The 6 best rules found using those configurations in our dataset of German credits are as follows:

IF Credit_history=critical/other existing credit AND Purpose=radio/tv AND Employment=>=7 AND Job=skilled THEN Class=good <conf:(1)>
IF Credit_history=critical/other existing credit AND Purpose=radio/tv AND Employment=>=7 AND Personal_status=male single AND Job=skilled THEN Class=good <conf:(1)>
IF Credit_history=critical/other existing credit AND Purpose=radio/tv AND Employment=>=7 AND Personal_status=male single THEN Class=good <conf:(1)>
IF Credit_history=critical/other existing credit AND Saving_status=no known savings AND Employment=>=7 AND Job=skilled THEN Class=good <conf:(1)>
IF Purpose=used car AND Personal_status=male single AND Job=high qualif/self emp/mgmt THEN Class=good <conf:(1)>
IF Purpose=radio/tv AND Saving_status=no known savings AND Employment=>=7 Personal_status=male single THEN Class=good <conf:(1)>

We can see six rules are represented in antecedent => consequent format. Here the number with the antecedent is the absolute coverage in the dataset. We can see in the six rules we have a consequence of class (See Fig11). The most important aspect to highlight in the rules found in that we have a confidence equal to 1 and a very high lift. Those two parameters show us very strong association without coincidence. Most of the rules represent credit_history which is critical/other existing credit. We will go through each rule to interpret them. The first rules show us among 27 applicants, if the applicant who have been working for a long time (7 years or more) with a skilled job and have a critical account or an existing credit in another bank with a purpose of buying a radio/tv then all 27 applicants are qualified as “good”. Secondly, if 24 male single applicants who have been working for a long time (7 years or more) with a skilled job and have a critical account or an existing credit in another bank with a purpose of buying a radio/tv, all of those applicants are considered as “good”. The term all is used to show the meaning of confidence. The third rule show us that if the credit_history of 30 male single applicants is critical account or other existing credit and the purpose is radio or tv and have been working seven years or more then all 30 applicants are qualified as good. The fourth rule show us that if the credit_history of 20 applicants is critical account or other existing credit and have an unknow or no saving account and have been working seven years or more with skilled job then all 20 applicants are qualified as “good”. The fifth rule show us that if 23 male single applicants with a high qualified/self-employee/management job with a purpose of loan being a used car they are qualified as “good”. Finally, if 20 male single applicants have been working for 7 years or more with an unknow saving account with the purpose of radio/tv, those applicants are qualified as “good” applicants. The above rules have been found using specific parameter, if we change the configuration, we can get more rules. These association are helpful to understand that the credit_history and the purpose of the

applicants are essential in order to select good applicants for the bank, but the results are not accurate until the confidence is 1. The rules can be expanded to find other patterns by removing attributes and modifying the configurations.

3.3 Clustering

Clustering data is a method that can be used in Weka by different algorithm. In our situation we are going to use the K means algorithm with the Euclidean distance by default. The algorithm is interesting because in our situation it automatically normalizes numerical attributes. Before running the algorithm, we are going to select few parameters and make sure to set the "Cluster Mode". First of all, we are going to filter the attributes in the Pre-Process tab, clicking on the choose button, select unsupervised, attribute and "AddCluster" then "Apply". Secondly, to perform clustering, select the "Cluster" tab in the explorer and click on the "Choose" button and select "SimpleKMeans". For the configuration, we are going to set the number of clusters to k=2 because we are looking for "bad" and "good" applicants. We make sure the distance is Euclidean. Then go to the cluster mode and click on the "Classes to cluster evaluation" and select "cluster". We can now click on the button start and interpret the result by visualising the cluster results by right-clicking the result and "visualise clusters assignments". The clustering model show us the centroid of each cluster and statistics. The Cluster 0 show the applicants favourable to get a loan and the Cluster 1 show the applicants unfavourable.

IF Checking_status = no checking THEN class= good
IF Checking_status = 0 <= X < 200 THEN class= bad
IF Saving_status < 100 THEN class=good/bad
We can see that the average of applicants qualified as good have not a current account in the bank whereas in "bad" applicants have between 0 and 200 for their checking_status. However, those results are a mean/average and cannot be considered as 100% accurate. In the third rule, we can interpret it by saying that the saving status do not have a huge influence on the applicants, this is not mandatory to choose if we will give a loan to the applicants.
IF Employment >= 7 AND Job= Skilled THEN class = good
IF Employment 1 <= X < 4 AND Job= Skilled THEN class=bad

Depending on the number of years that an applicant has been working in their current employment it can be a reason to offer a loan or not. In the rules we have found, if an applicant has been working seven years or more in their current employment with a skilled job, they are qualified as "good". However, if an applicant has been working between 1 or 4 years in their current employment with a skilled job then they are considered as "bad" applicants. Those rules mean that in the average, out of 617 instance applicants with a skilled job are more likely to receive a loan if they have been working seven years or more. Finally, even though the applicants have critical or an existing debt most of the applicants are qualified as good applicants because the rules explained above show the important attributes to be studied in order to provides a loan.

In order to support our results, I have decided to include another algorithm in Clustering: EM.
IF Checking_status (no checking) THEN class= good
IF Credit_history (existing paid) THEN class= good
IF Credit_amount mean THEN class = good IF Credit_amount THEN class = bad
Employment > 7 considered as good applicants
IF Job = skilled THEN class = good

The cluster 0 represent the "good" applicants and the cluster 1 represent "bad" applicants. As we can see in the Fig12, if the checking_status is no checking then the applicants are qualified as "good" and they represent 40% (275/682) of the good applicants. If applicants have an existing debt paid back duly now, they are qualified as "good" applicants (397 out of 683). Then, the credit_amount mean differ from the two cluster. We can see that a high credit_amount among applicants is often a reason to not offer a loan. Finally, 466 applicants are considered as "good" applicants if they have a skilled job and represent almost 70% of the "good "applicants. However, only 50% of "bad" applicants have a skilled job this mean that the reason of being "bad" comes from another aspect.

4 Conclusion

To sum up, through our previous research we have found with our classification method that the credit history of each applicants should be checked in order to choose to offer a loan. Moreover, checking the status of an existing account has been found to be a critical criterion to analyse. In our association, with an accuracy of a 100% we can support our first conjecture by saying that most of the "good" applicants have a critical account or an existing debt in another bank. Finally, the clustering method go through each attribute to confirm the results. We can conclude first that most of the applicants are favourable and have different criteria but the majority of them have not a current account in the bank. Therefore, clients are qualified as "good" applicants when they pay their credit on time. Finally, results show us that the clients who have higher amount on their current account and who are working for longer period of time are better debtor. SimpleKMeans Clustering is one of the easiest unsupervised learning algorithms. It is simple to use and efficient in order to find the good patterns to convince the managers of the bank. However, it is more interesting to use an algorithm in the classification method such as J48 because we have found many rules and can have an overview of them with a decision tree. Apriori algorithm is also effective, treating data is not easy and require rigor. This algorithm helps us be more accurate with statistics results using a confidence, lift or coverage. Weka is a very good software when the data is cleaned and prepared, this is not the case in every data set. But the platform allows us to run different algorithms very fast.

Appendix

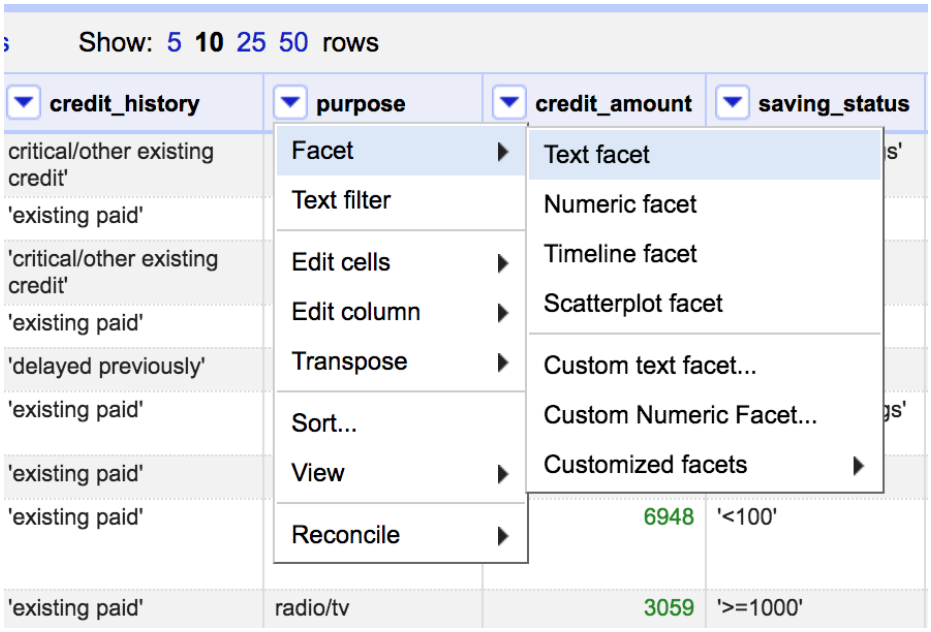

Fig1. Text Facet

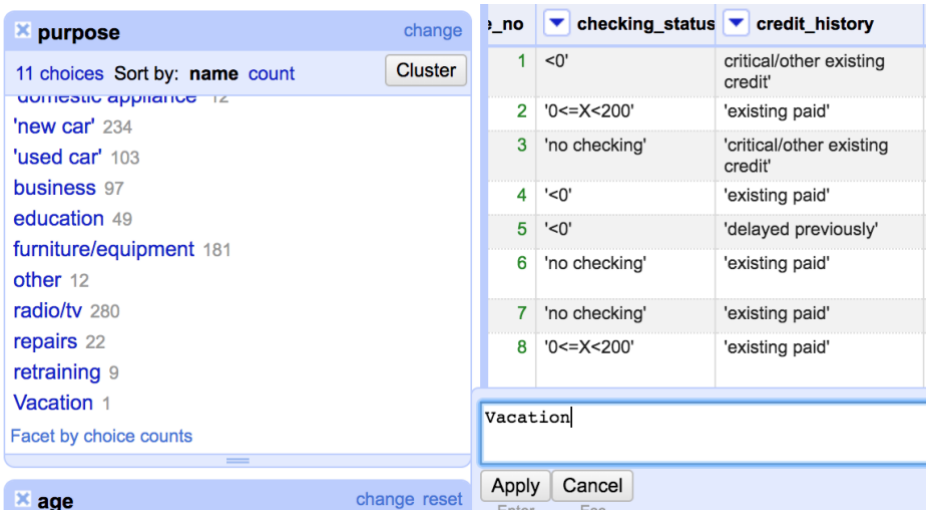

Fig2. Spelling mistakes

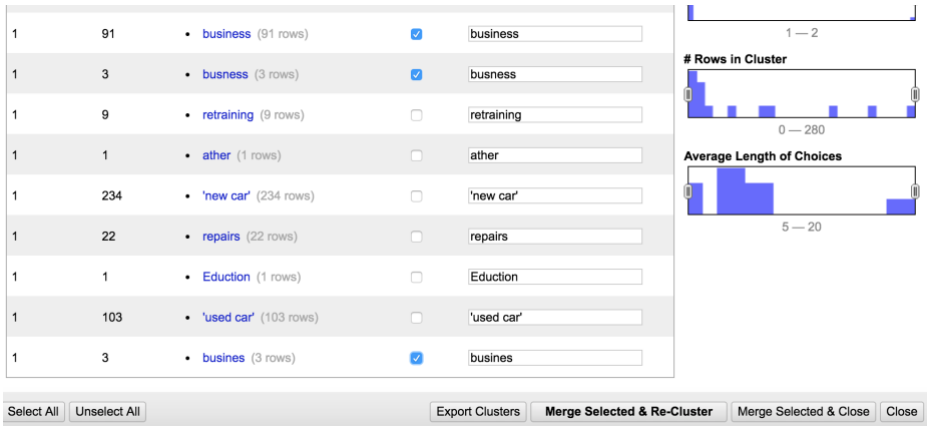

Fig3. Merge Selected and Re-Cluster “business”

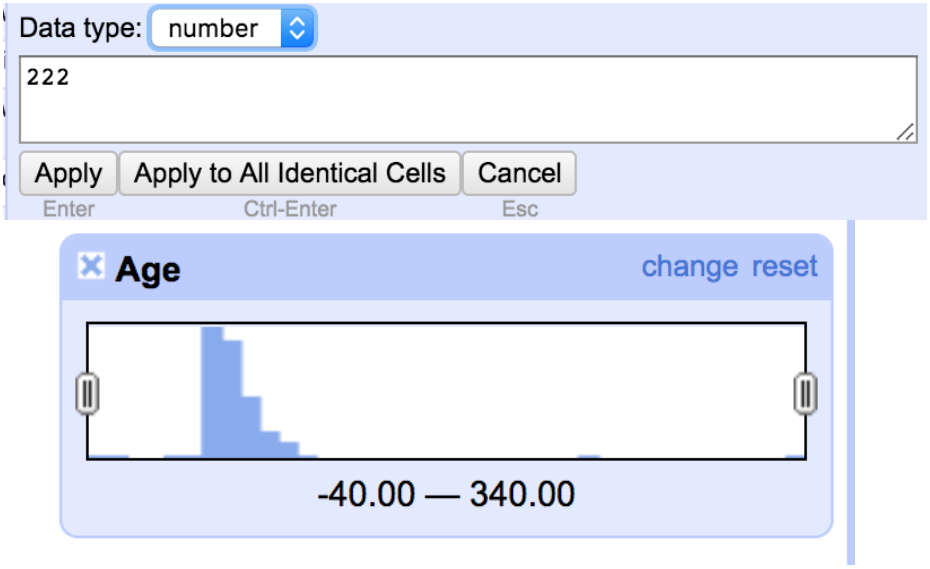

Fig4. Outliers in Age to be corrected.

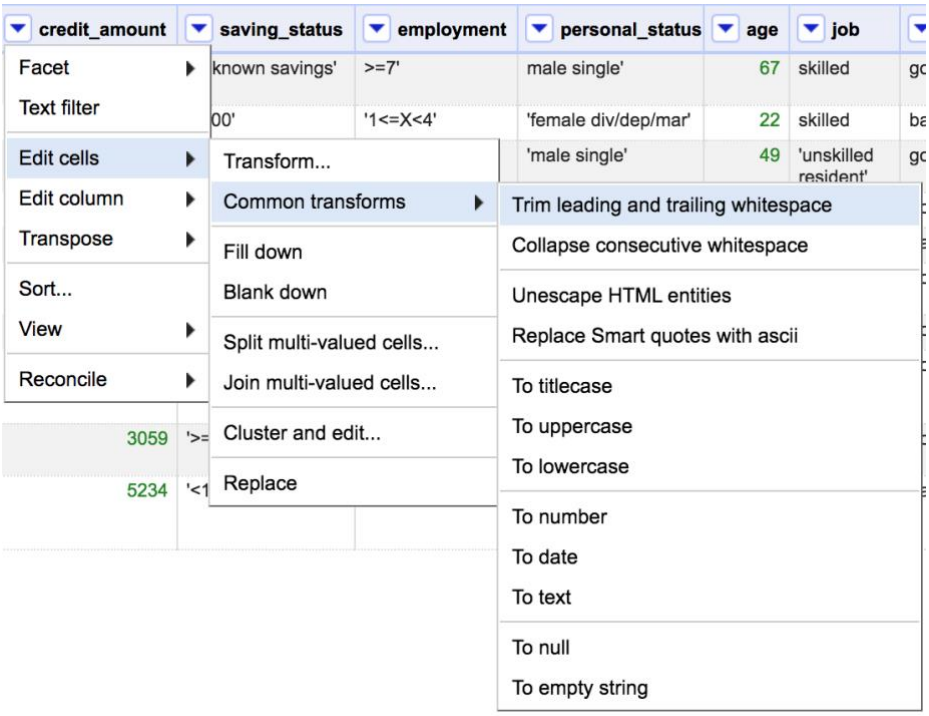

Fig5. Common transform credit_amount.

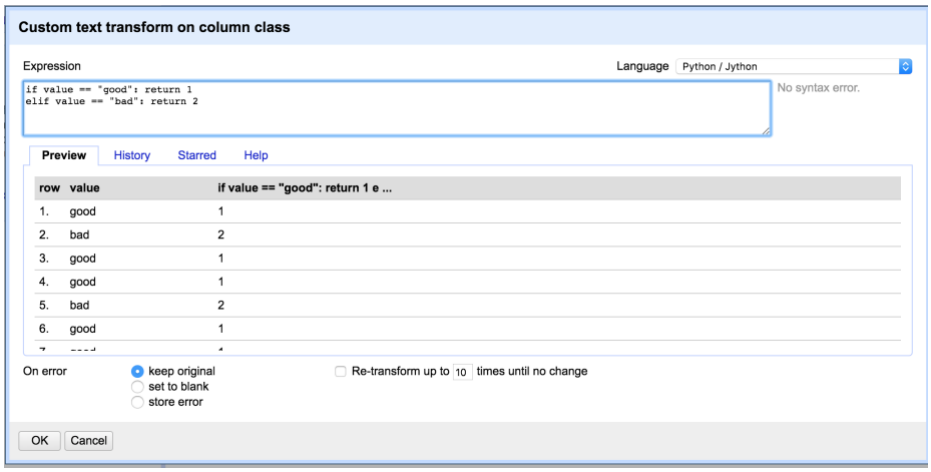

Fig6. OpenRefine and Python/Jython class: *good/bad*

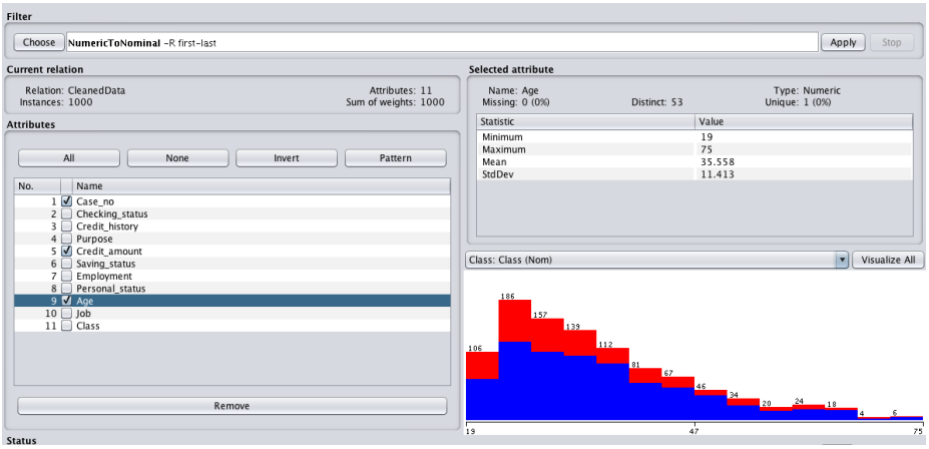

Fig7. Convert Case_no, Credit_amount and Age to nominal with Weka filter.

```
J48 pruned tree
------------------

Credit_history = critical/other existing credit: good (293.0/50.0)
Credit_history = existing paid
|   Credit_amount <= 8648: good (502.0/148.0)
|   Credit_amount > 8648: bad (28.0/7.0)
Credit_history = delayed previously
|   Employment = >=7
|   |   Age <= 38: good (6.0)
|   |   Age > 38: bad (14.0/6.0)
|   Employment = 1<=X<4: good (33.0/6.0)
|   Employment = 4<=X<7
|   |   Age <= 28
|   |   |   Credit_amount <= 4736: bad (4.0/1.0)
|   |   |   Credit_amount > 4736: good (2.0)
|   |   Age > 28: good (11.0)
|   Employment = unemployed
|   |   Age <= 39: good (4.0/1.0)
|   |   Age > 39: bad (2.0)
|   Employment = <1
```

Fig8. J48 pruned Tree to support.

```
Attributes:   3
              Checking_status
              Job
              Class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
------------------

Checking_status = <0
|   Job = skilled: bad (172.0/76.0)
|   Job = unskilled resident: good (59.0/23.0)
|   Job = high qualif/self emp/mgmt: good (37.0/12.0)
|   Job = unemp/unskilled non res: bad (6.0/2.0)
Checking_status = 0<=X<200: good (269.0/105.0)
Checking_status = no checking: good (394.0/46.0)
Checking_status = >=200: good (63.0/14.0)

Number of Leaves  :     7

Size of the tree :      9


Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances         722               72.2    %
Incorrectly Classified Instances       278               27.8    %
Kappa statistic                          0.2511
Mean absolute error                      0.3629
Root mean squared error                  0.4259
Relative absolute error                 86.361  %
Root relative squared error             92.9483 %
Total Number of Instances             1000
```

Fig9. J48 Pruned Tree Checking_status.

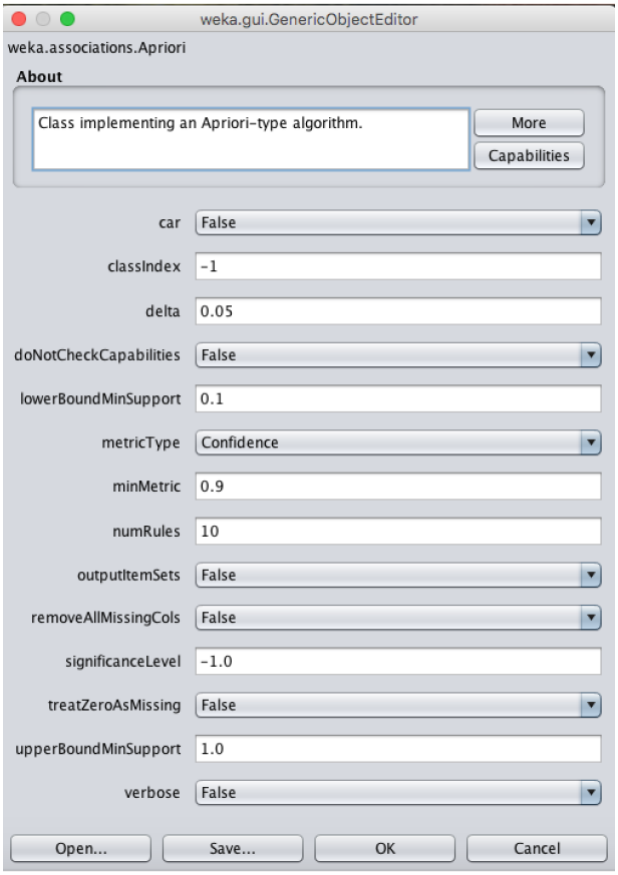

Fig10. Apriori configurations.

```
Best rules found:

1. Credit_history=critical/other existing credit Purpose=radio/tv Employment=>=7 Job=skilled 27 ==> Class=good 27    <conf:(1)> lift:(1.43)
2. Credit_history=critical/other existing credit Saving_status=no known savings Employment=>=7 Job=skilled 20 ==> Class=good 20    <conf:(1)
3. Purpose=radio/tv Saving_status=no known savings Employment=>=7 Job=skilled 20 ==> Class=good 20    <conf:(1)> lift:(1.43) lev:(0.01) [6]
4. Credit_history=critical/other existing credit Purpose=radio/tv Saving_status=<100 Employment=>=7 14 ==> Class=good 14    <conf:(1)> lift:
5. Purpose=furniture/equipment Employment=1<=X<4 Job=unskilled resident 13 ==> Credit_history=existing paid 13    <conf:(1)> lift:(1.89) lev
6. Credit_history=critical/other existing credit Purpose=furniture/equipment Employment=>=7 12 ==> Job=skilled 12    <conf:(1)> lift:(1.59)
```

Fig11. Apriori rules

```
                                        Cluster
Attribute                                    0         1
                                        (0.68)    (0.32)
============================================================
Checking_status
  <0                                   188.7368   87.2632
  0<=X<200                             162.2465  108.7535
  no checking                          275.7511  120.2489
  >=200                                 55.7375    9.2625
  [total]                              682.4718  325.5282
Credit_history
  critical/other existing credit       195.2657   99.7343
  existing paid                        397.1621  134.8379
  delayed previously                    39.5446   50.4554
  no credits/all paid                   17.1073   24.8927
  all paid                               34.392    16.608
  [total]                              683.4718  326.5282
Credit_amount
  mean                                2004.9403 6125.1504
  std. dev.                           1006.1784 3543.0742


Class attribute: Class
Classes to Clusters:

   0   1  <-- assigned to cluster
 535 165 | good
 186 114 | bad

Cluster 0 <-- good
Cluster 1 <-- bad

Incorrectly clustered instances :       351.0    35.1    %
```

Fig12. Results Cluster EM.