DOĞUŞ UNIVERSITY

CLASSIFICATION ALGORITHMS AND APRIORI ALGORITHM USING WEKA

Data Warehousing and Mining (ISE 401)

Computer Engineering, Engineering Department

Adam Janowski

202103001118

08.01.2022

ISTANBUL, 2022

Table of Contests

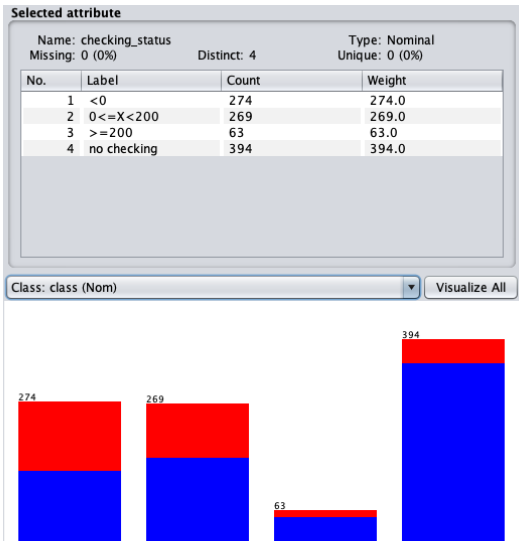
German Credit Dataset

The original dataset contains 1000 entries with 20 numerical/nominal attributes prepared by Prof. Hofmann.

In this dataset, each entry represents a person who takes a credit by a bank. Each person is classified as good or bad credit risks according to the set of attributes.

Attribute 1: checking_status

- Status of existing checking account. In provided dataset numerical values are already converted to nominal values by dividing values on 4 groups.



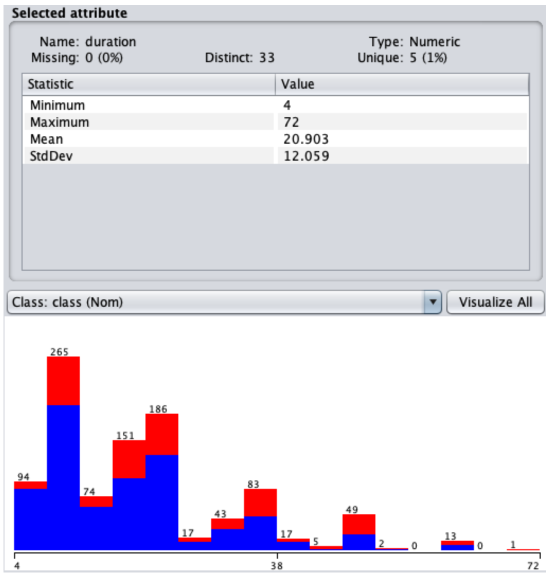
Selected attribute

Name: checking_status　Type: Nominal
Missing: 0 (0%)　Distinct: 4　Unique: 0 (0%)

No.	Label	Count	Weight
1	<0	274	274.0
2	0<=X<200	269	269.0
3	>=200	63	63.0
4	no checking	394	394.0

Attribute 2: duration

- Duration of a credit in months.

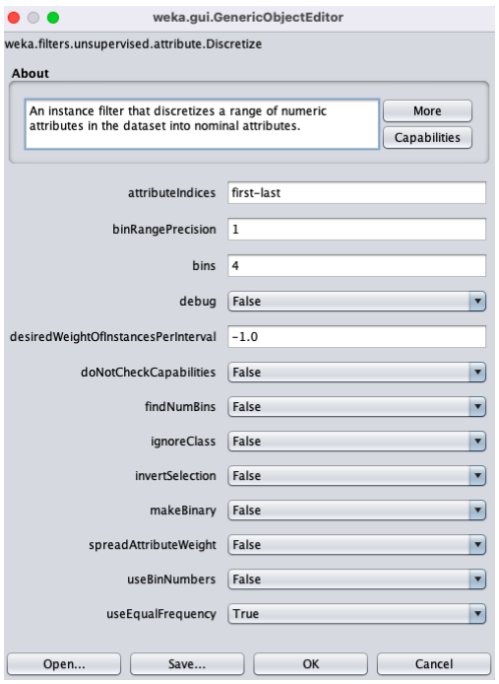


Selected attribute

Name: duration　Type: Numeric
Missing: 0 (0%)　Distinct: 33　Unique: 5 (1%)

Statistic	Value
Minimum	4
Maximum	72
Mean	20.903
StdDev	12.059

As it is better to have all variables as categorical, duration attribute should be converted manually, in similar way as it was done in attribute 1. As original data is left skewed it is better to mark "useEqualFrequency" as true, what will make categorical bins more symmetric.

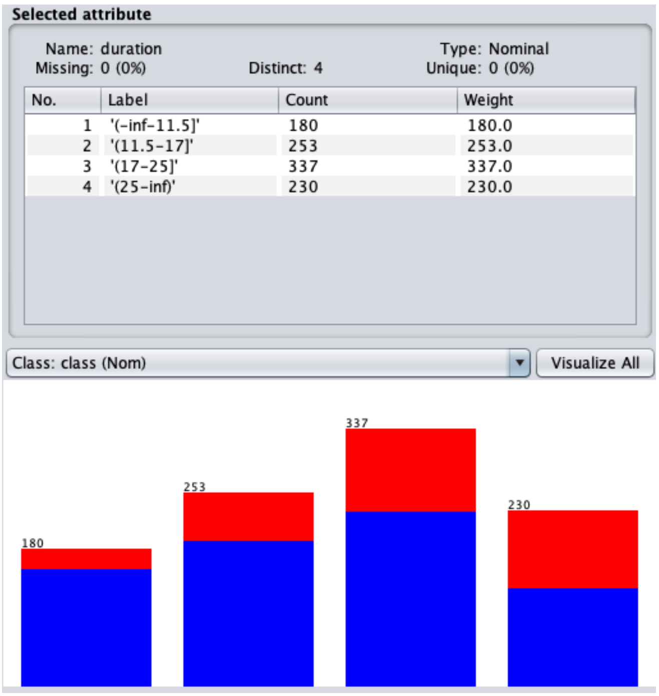

weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.Discretize

About

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

attributeIndices	first-last
binRangePrecision	1
bins	4
debug	False
desiredWeightOfInstancesPerInterval	-1.0
doNotCheckCapabilities	False
findNumBins	False
ignoreClass	False
invertSelection	False
makeBinary	False
spreadAttributeWeight	False
useBinNumbers	False
useEqualFrequency	True


Selected attribute

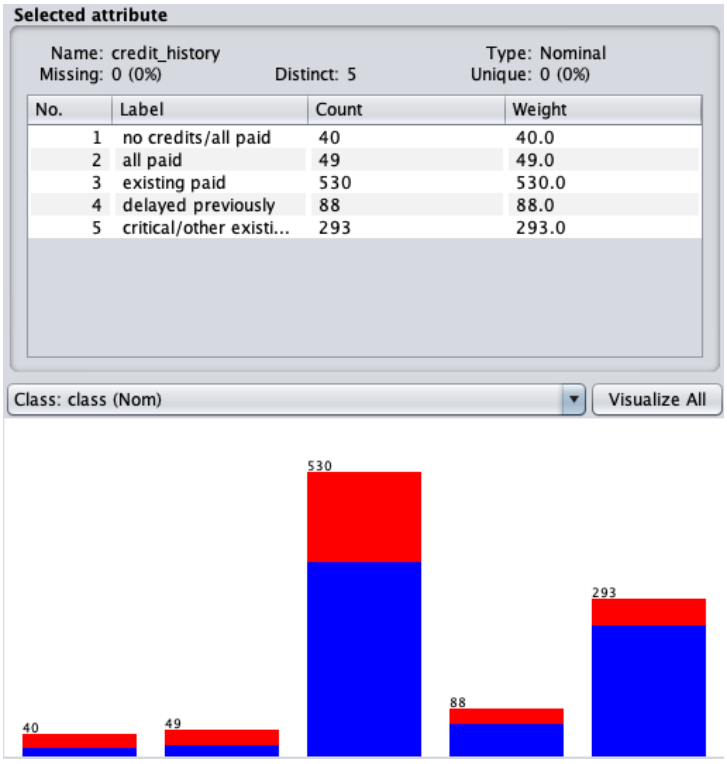
Name: duration
Missing: 0 (0%)
Distinct: 4
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-11.5]'	180	180.0
2	'(11.5-17]'	253	253.0
3	'(17-25]'	337	337.0
4	'(25-inf)'	230	230.0

Class: class (Nom)

Attribute 3: credit_history

- Shows credit history of a customer. May take the following values: 'no credits/all paid', 'all paid', 'existing paid', 'delayed previously', 'critical/other existing credit'.


Selected attribute

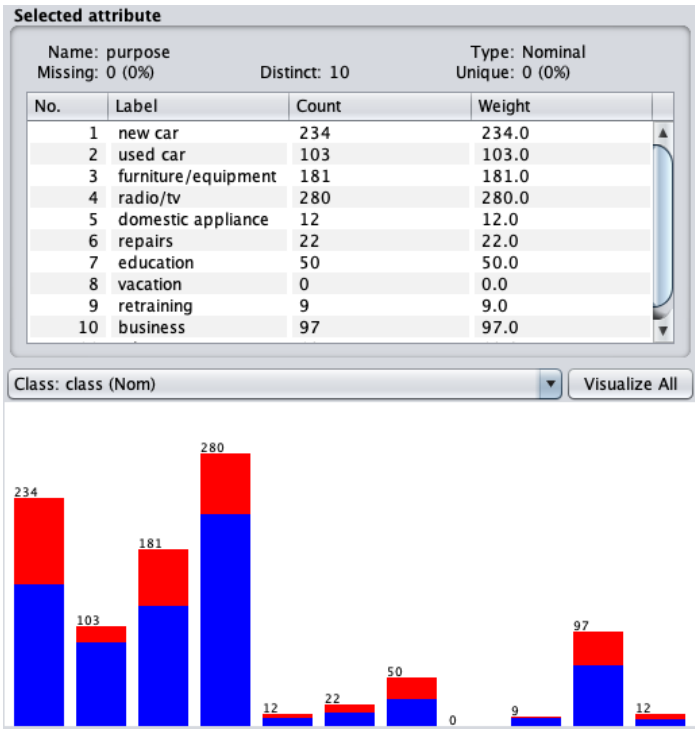
Name: credit_history
Missing: 0 (0%)
Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	no credits/all paid	40	40.0
2	all paid	49	49.0
3	existing paid	530	530.0
4	delayed previously	88	88.0
5	critical/other existi...	293	293.0

Class: class (Nom)

Attribute 4: purpose

- Tells us the purpose of the customer for the taken credit.



Selected attribute

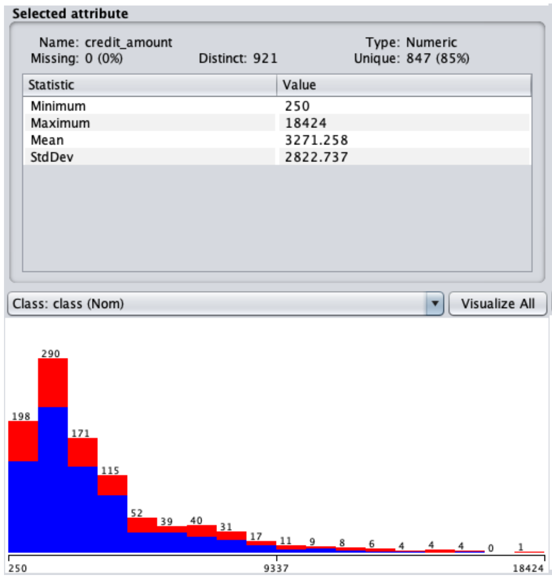
Name: purpose   Type: Nominal
Missing: 0 (0%)   Distinct: 10   Unique: 0 (0%)

No.	Label	Count	Weight
1	new car	234	234.0
2	used car	103	103.0
3	furniture/equipment	181	181.0
4	radio/tv	280	280.0
5	domestic appliance	12	12.0
6	repairs	22	22.0
7	education	50	50.0
8	vacation	0	0.0
9	retraining	9	9.0
10	business	97	97.0

Class: class (Nom)

Attribute 5: credit_amount

- Shows the amount of taken credit. Similarly, is in attribute 2, original values should be converted to nominal one using equally frequent bins.

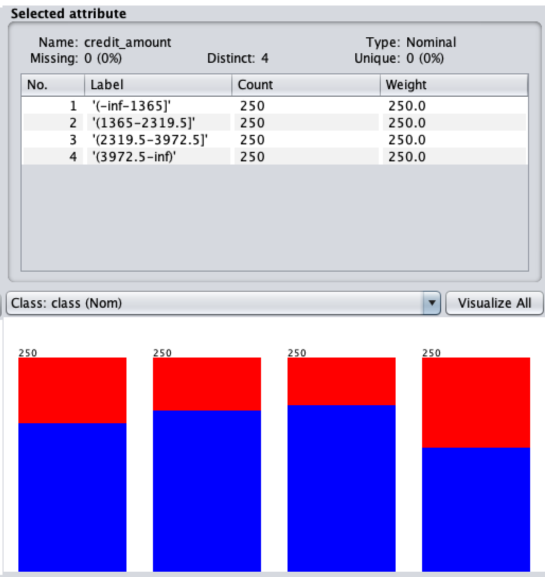


Selected attribute

Name: credit_amount   Type: Numeric
Missing: 0 (0%)   Distinct: 921   Unique: 847 (85%)

Statistic	Value
Minimum	250
Maximum	18424
Mean	3271.258
StdDev	2822.737

Class: class (Nom)



Selected attribute

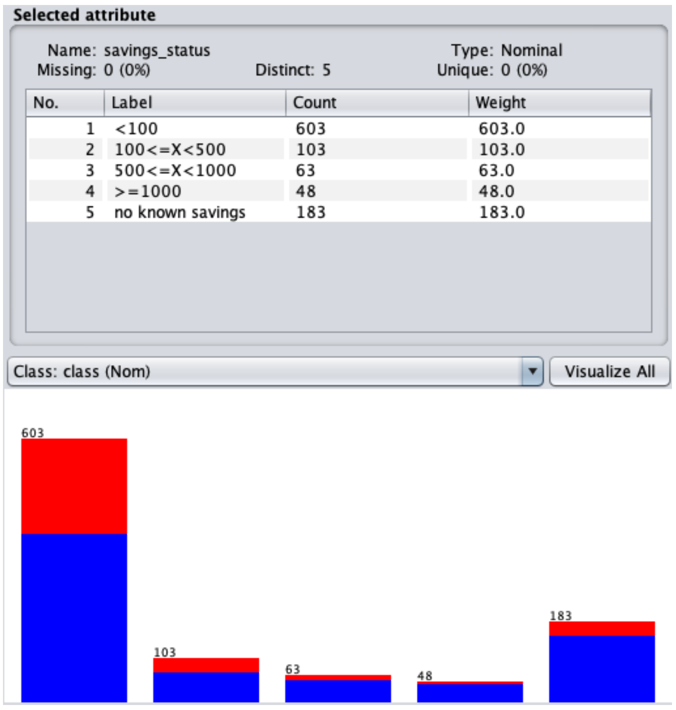
Name: credit_amount   Type: Nominal
Missing: 0 (0%)   Distinct: 4   Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-1365]'	250	250.0
2	'(1365-2319.5]'	250	250.0
3	'(2319.5-3972.5]'	250	250.0
4	'(3972.5-inf)'	250	250.0

Class: class (Nom)

Attribute 6: savings_status

- Savings account/bond.


Selected attribute

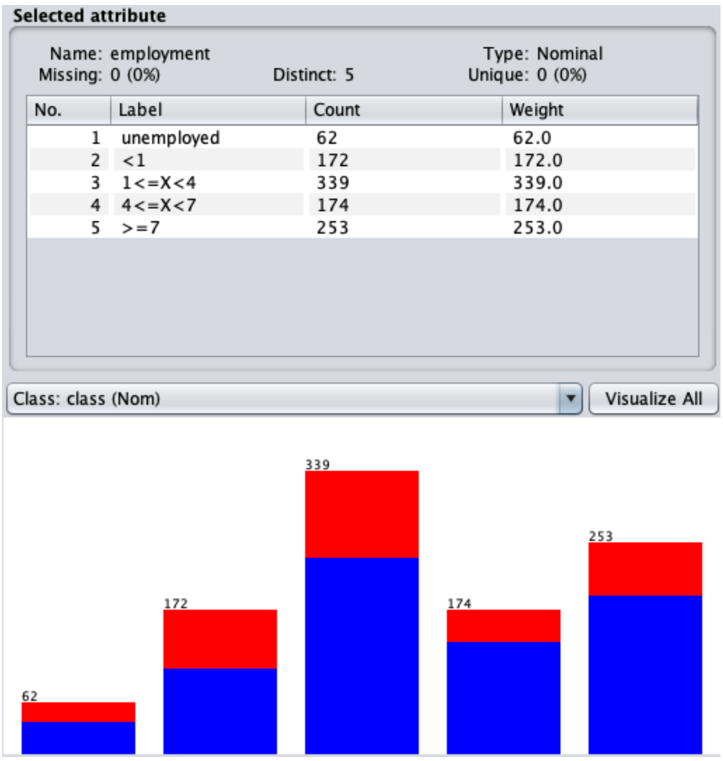
Name: savings_status
Missing: 0 (0%)
Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	<100	603	603.0
2	100<=X<500	103	103.0
3	500<=X<1000	63	63.0
4	>=1000	48	48.0
5	no known savings	183	183.0

Class: class (Nom)

Attribute 7: employment

- Shows for how long the customer is employed.


Selected attribute

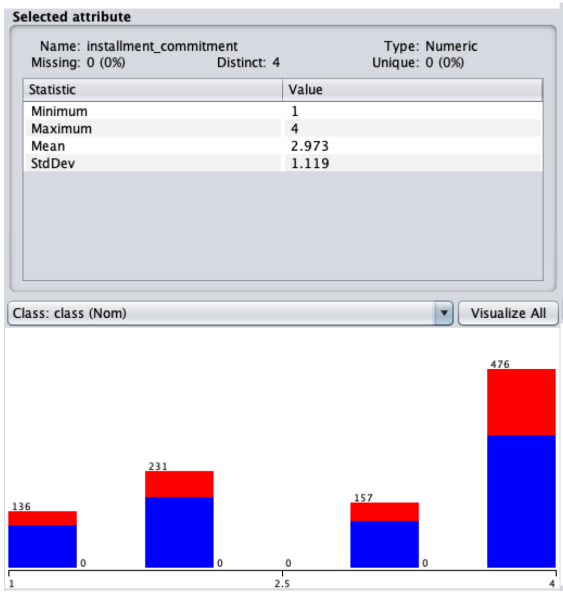
Name: employment
Missing: 0 (0%)
Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	unemployed	62	62.0
2	<1	172	172.0
3	1<=X<4	339	339.0
4	4<=X<7	174	174.0
5	>=7	253	253.0

Class: class (Nom)

Attribute 8: installment_commitment

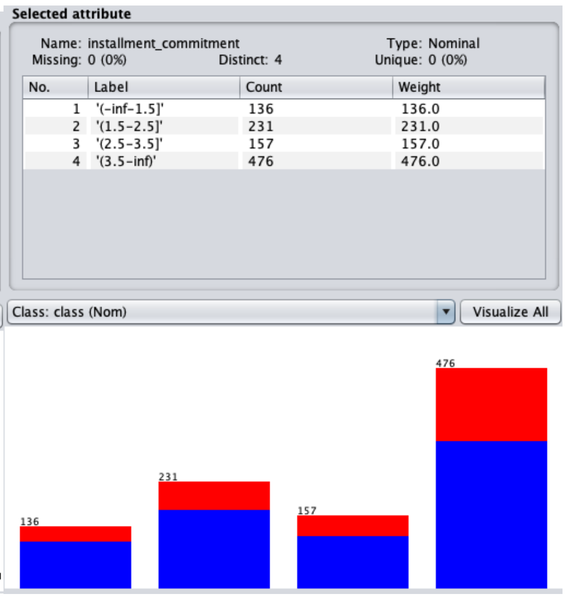
- Installment rate in percentage of disposable income. As this attribute is expressed only as integers in the range 1-4, changing the attribute type to categorical will not affect the values themselves.


Selected attribute

Name: installment_commitment Type: Numeric
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	4
Mean	2.973
StdDev	1.119

Class: class (Nom) Visualize All


Selected attribute

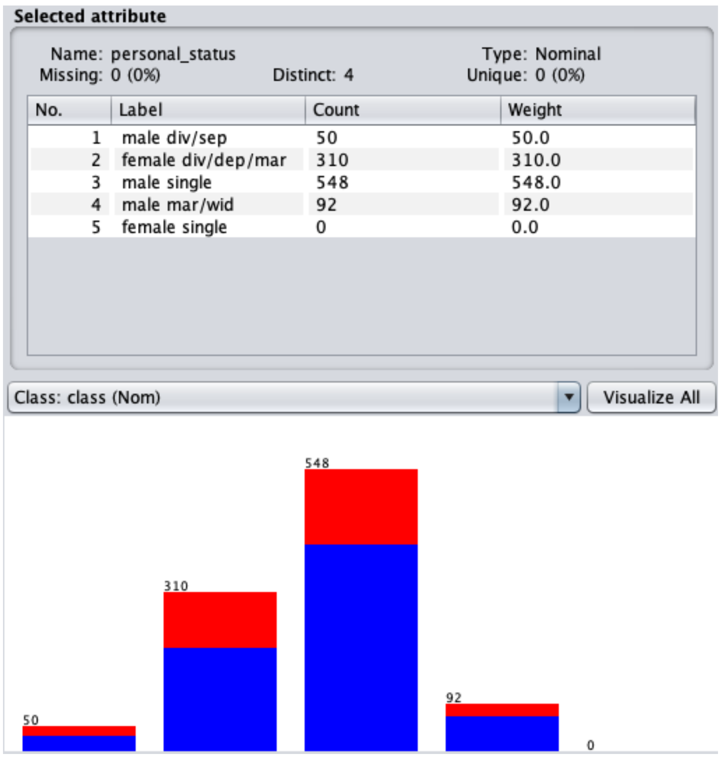
Name: installment_commitment Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-1.5]'	136	136.0
2	'(1.5-2.5]'	231	231.0
3	'(2.5-3.5]'	157	157.0
4	'(3.5-inf)'	476	476.0

Class: class (Nom) Visualize All

Attribute 9: personal_status

- Shows personal status and sex of a customer.


Selected attribute

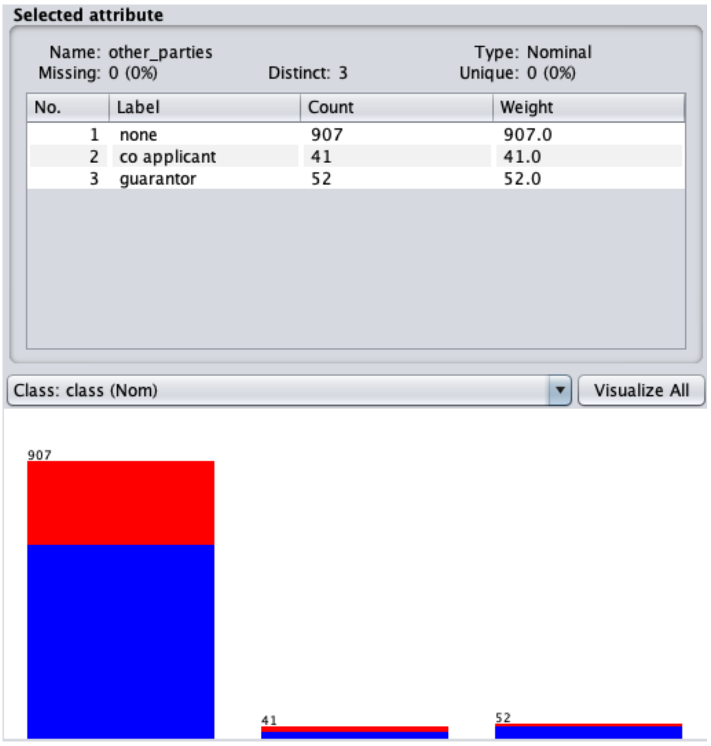
Name: personal_status Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	male div/sep	50	50.0
2	female div/dep/mar	310	310.0
3	male single	548	548.0
4	male mar/wid	92	92.0
5	female single	0	0.0

Class: class (Nom) Visualize All

Attribute 10: other_parties

- Other debtors / guarantors



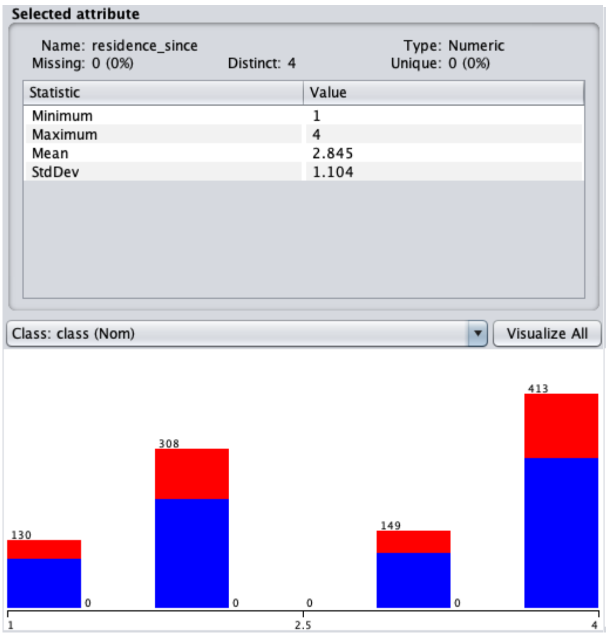
Selected attribute

Name: other_parties
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	none	907	907.0
2	co applicant	41	41.0
3	guarantor	52	52.0

Attribute 11: residence_since

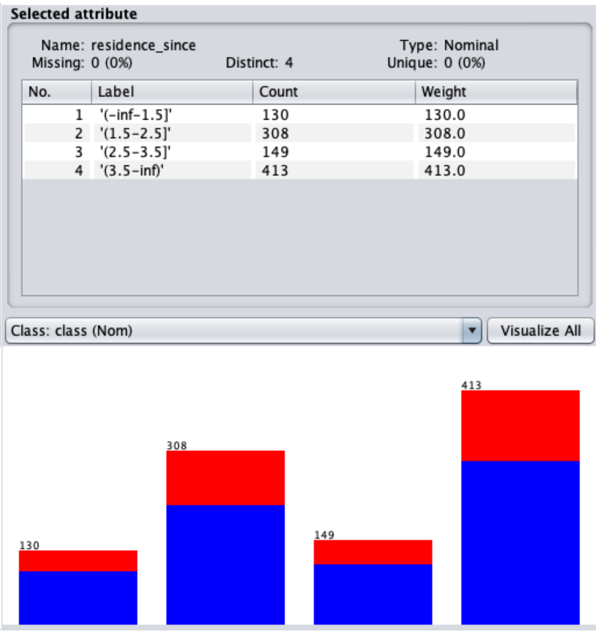
- Present residence since. As this attribute is expressed only as integers in the range 1-4, changing the attribute type to categorical will not affect the values themselves.



Selected attribute

Name: residence_since
Missing: 0 (0%)
Distinct: 4
Type: Numeric
Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	4
Mean	2.845
StdDev	1.104



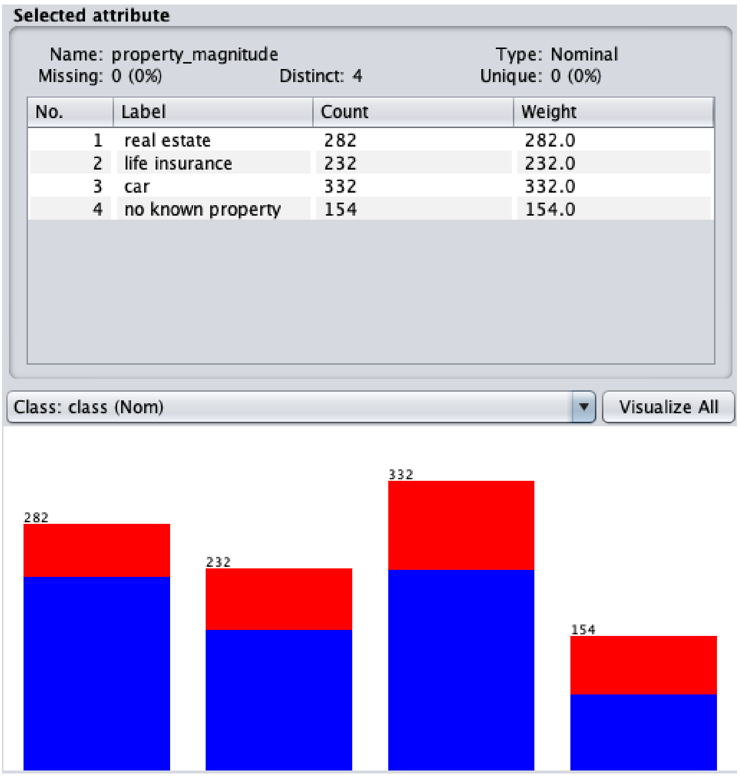
Selected attribute

Name: residence_since
Missing: 0 (0%)
Distinct: 4
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-1.5]'	130	130.0
2	'(1.5-2.5]'	308	308.0
3	'(2.5-3.5]'	149	149.0
4	'(3.5-inf)'	413	413.0

Attribute 12: properties_magnitude

- Property.



Selected attribute

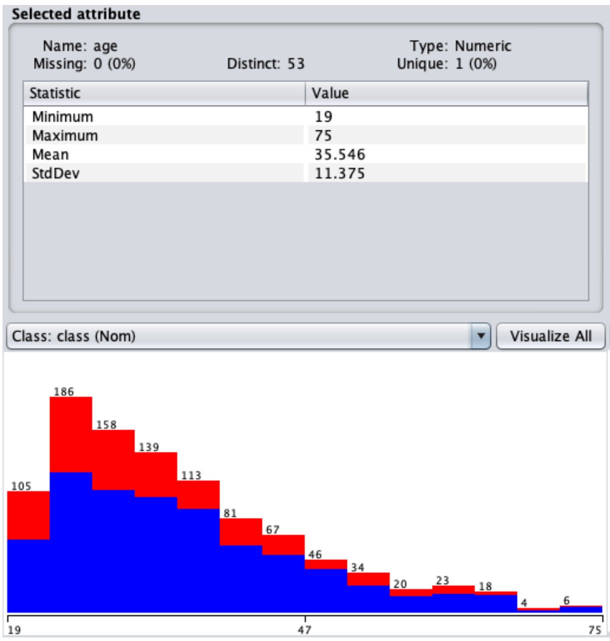
Name: property_magnitude   Type: Nominal
Missing: 0 (0%)   Distinct: 4   Unique: 0 (0%)

No.	Label	Count	Weight
1	real estate	282	282.0
2	life insurance	232	232.0
3	car	332	332.0
4	no known property	154	154.0

Class: class (Nom)

Attribute 13: age

- Age of customer in years. Similarly, is in attribute 2 and 5, original values should be converted to nominal one using equally frequent bins.

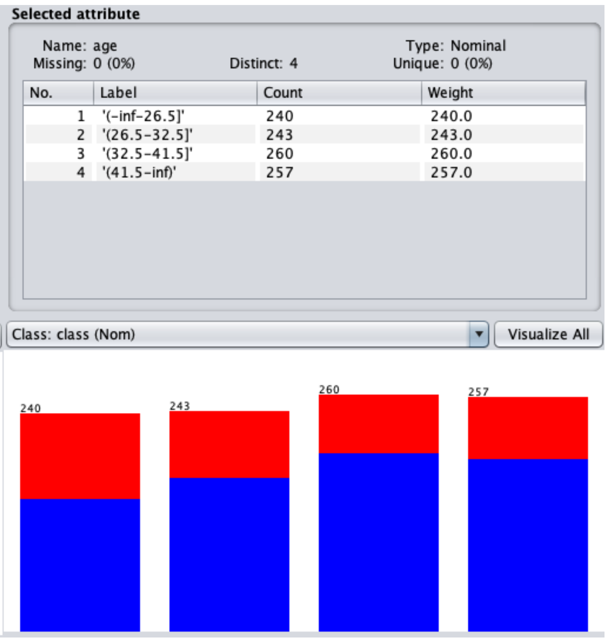


Selected attribute

Name: age   Type: Numeric
Missing: 0 (0%)   Distinct: 53   Unique: 1 (0%)

Statistic	Value
Minimum	19
Maximum	75
Mean	35.546
StdDev	11.375

Class: class (Nom)



Selected attribute

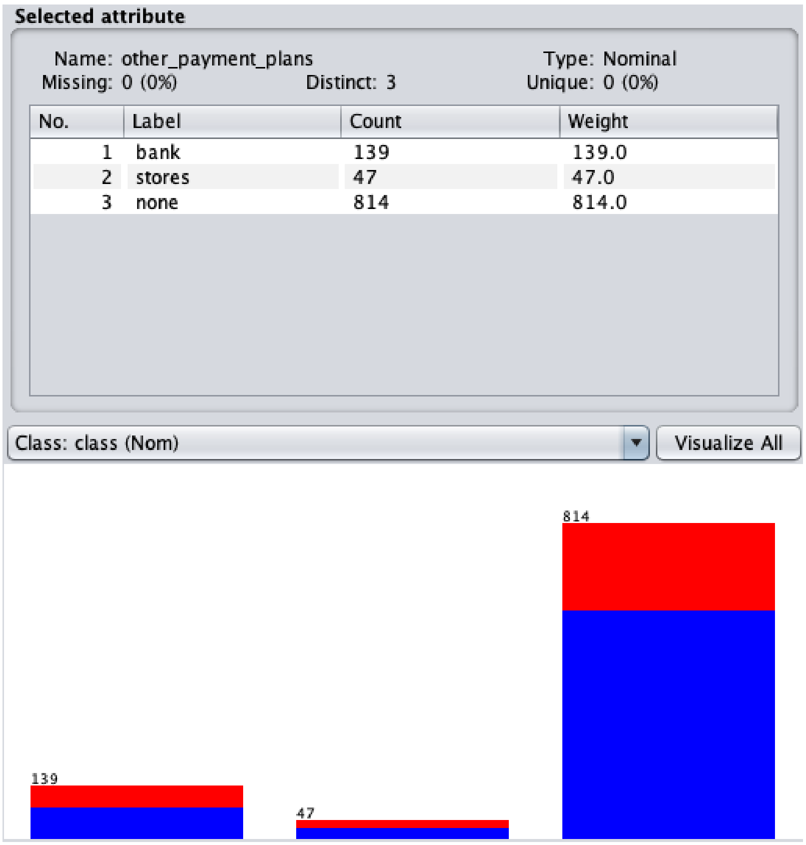
Name: age   Type: Nominal
Missing: 0 (0%)   Distinct: 4   Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-26.5]'	240	240.0
2	'(26.5-32.5]'	243	243.0
3	'(32.5-41.5]'	260	260.0
4	'(41.5-inf)'	257	257.0

Class: class (Nom)

Attribute 14: other_payment_plans

- Other installment plans.



Selected attribute

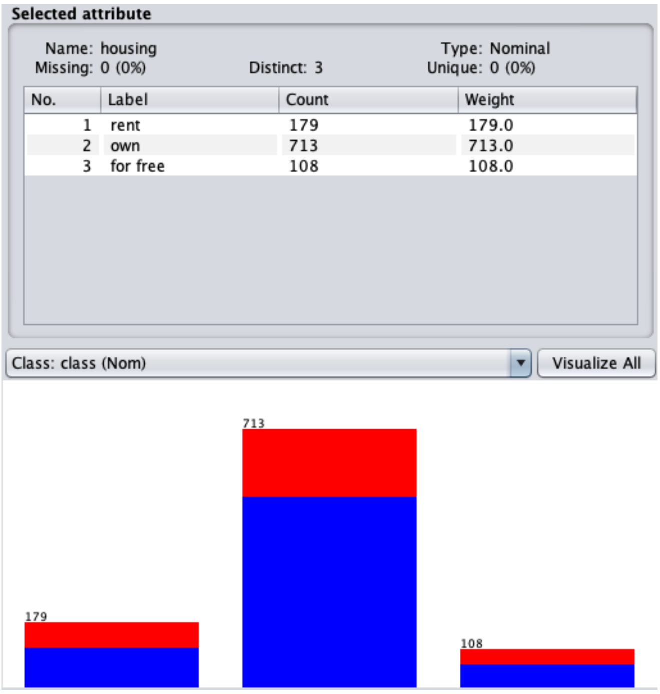
Name: other_payment_plans Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	bank	139	139.0
2	stores	47	47.0
3	none	814	814.0

Class: class (Nom)

Attribute 15: housing

- Housing



Selected attribute

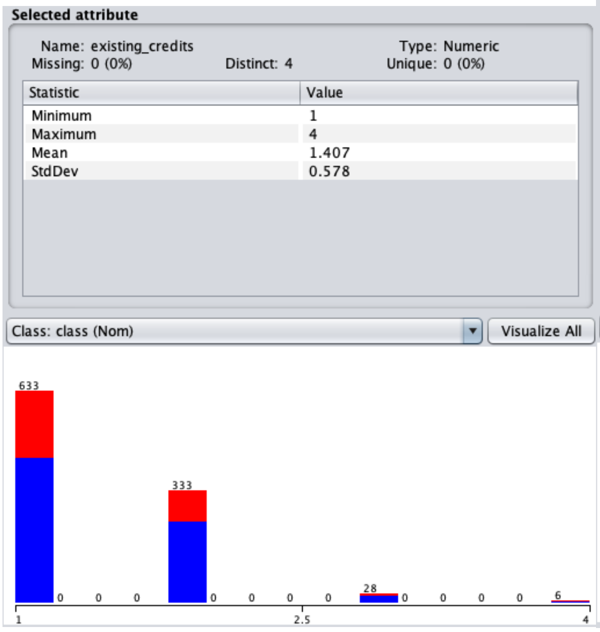
Name: housing Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	rent	179	179.0
2	own	713	713.0
3	for free	108	108.0

Class: class (Nom)

Attribute 16: existing_credits

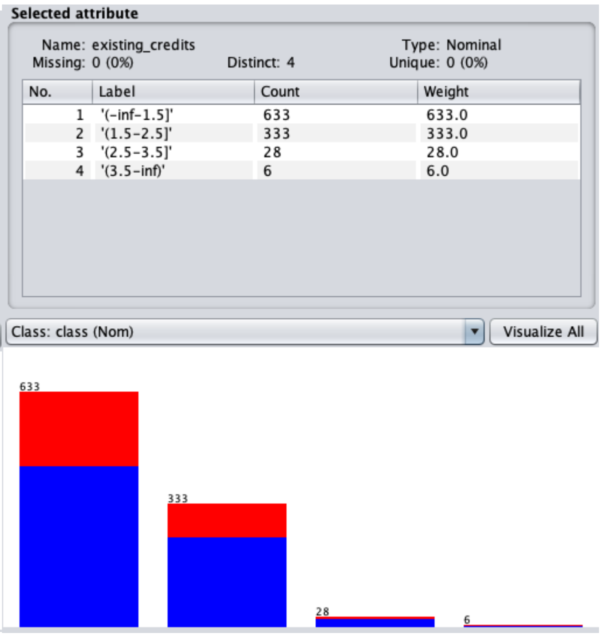
- Number of existing credits at this bank. As this attribute is expressed only as integers in the range 1-4, changing the attribute type to categorical will not affect the values themselves.



Selected attribute

Name: existing_credits Type: Numeric
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	4
Mean	1.407
StdDev	0.578



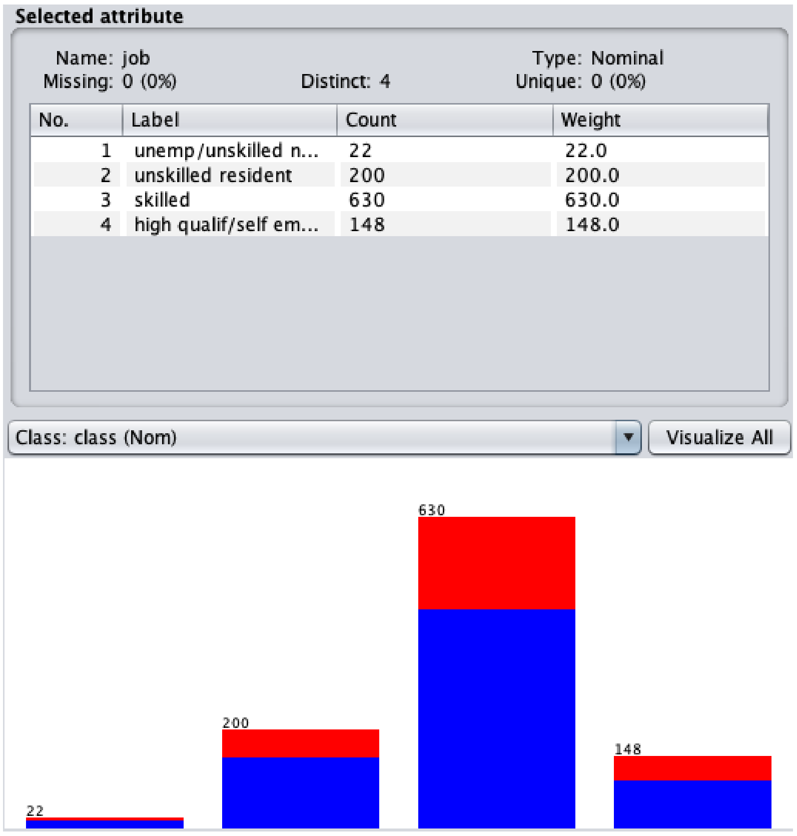
Selected attribute

Name: existing_credits Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-1.5]'	633	633.0
2	'(1.5-2.5]'	333	333.0
3	'(2.5-3.5]'	28	28.0
4	'(3.5-inf)'	6	6.0

Attribute 17: job

- Description of customer's job.



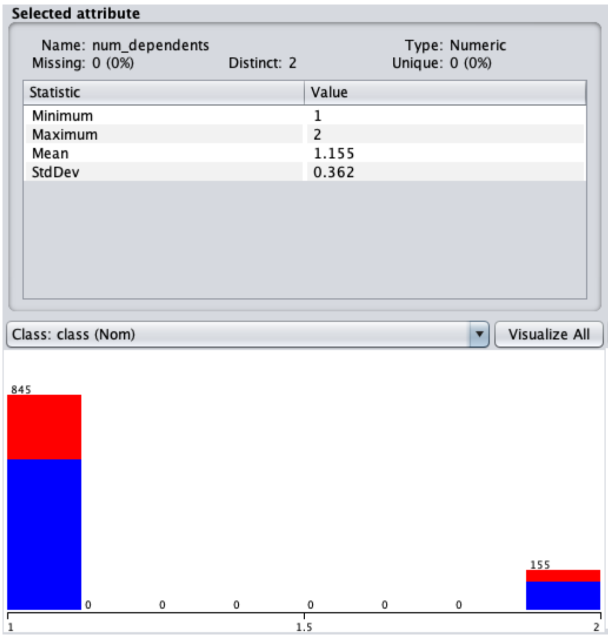
Selected attribute

Name: job Type: Nominal
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

No.	Label	Count	Weight
1	unemp/unskilled n...	22	22.0
2	unskilled resident	200	200.0
3	skilled	630	630.0
4	high qualif/self em...	148	148.0

Attribute 18: num_dependents

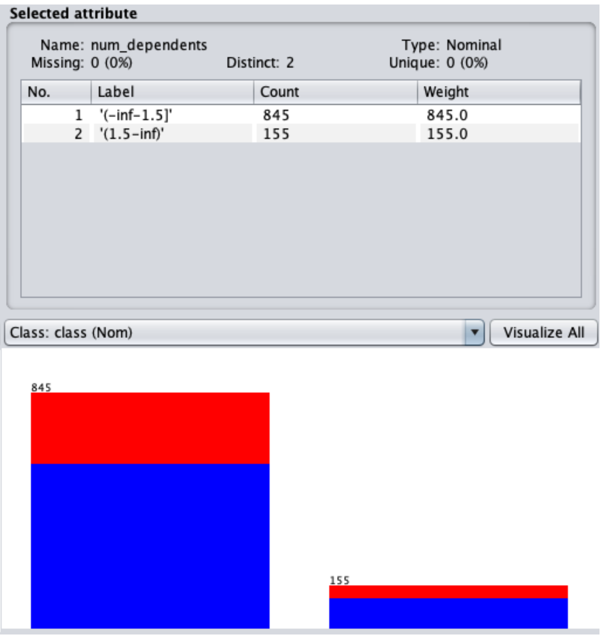
- Number of people being liable to provide maintenance for.



Selected attribute

Name: num_dependents Type: Numeric
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

Statistic	Value
Minimum	1
Maximum	2
Mean	1.155
StdDev	0.362



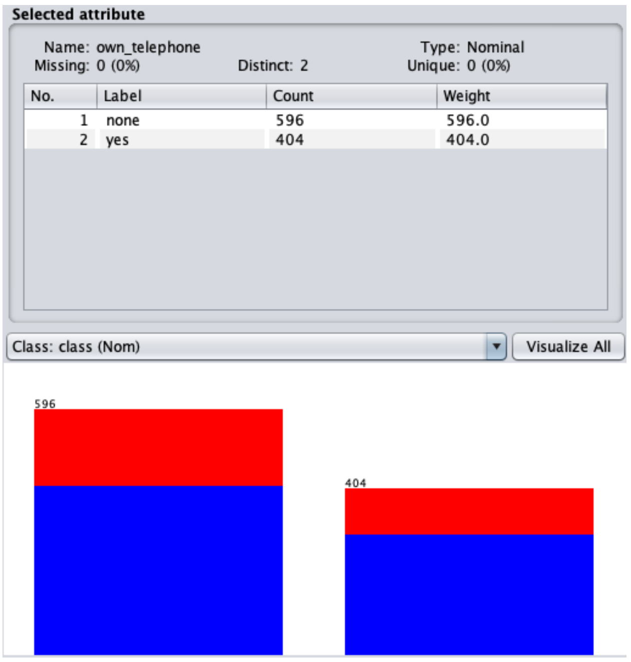
Selected attribute

Name: num_dependents Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	'(-inf-1.5]'	845	845.0
2	'(1.5-inf)'	155	155.0

Attribute 19: own_telephone

- If customer has a telephone.



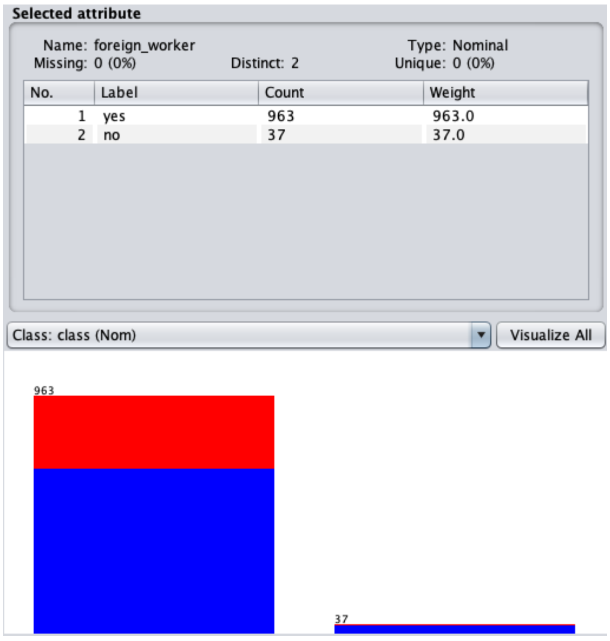
Selected attribute

Name: own_telephone Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	none	596	596.0
2	yes	404	404.0

Attribute 20: foreign_worker

- If a customer is a foreign worker.



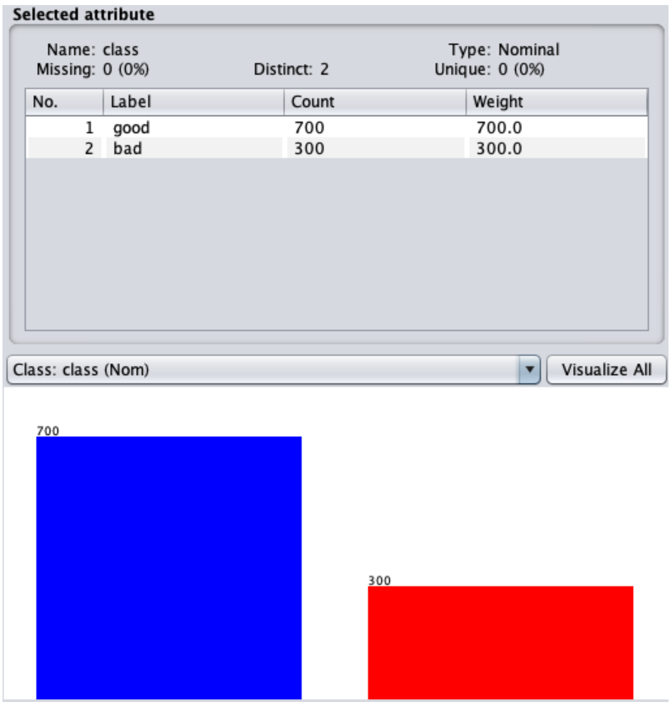
Selected attribute

Name: foreign_worker   Type: Nominal
Missing: 0 (0%)   Distinct: 2   Unique: 0 (0%)

No.	Label	Count	Weight
1	yes	963	963.0
2	no	37	37.0

Attribute 21: class

- Dependent variable, also called outcome variable. It shows if a customer is “good” or “bad” for a creditor. As it is shown below the attribute is imbalanced, what mean that in the dataset there is more “good” values than “bad”. this is an important feature of the variable as some algorithms can only do well with the equally numeric search variable. Due to the specifics of the dataset, it is worse to class a customer as good when they are bad, than it is to class a customer as bad when they are good.



Selected attribute

Name: class   Type: Nominal
Missing: 0 (0%)   Distinct: 2   Unique: 0 (0%)

No.	Label	Count	Weight
1	good	700	700.0
2	bad	300	300.0

Naïve Bayes

Theory of Naïve Bayes

*Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, **mutual independence of features**. If any two events A and B are mutually independent, then:*

P(A,B) = P(A)P(B)

*Naïve Bayes classifier works on **conditional probability**. Conditional probability is the probability that something will happen, given that something else has already occurred. Using the conditional probability, we can calculate the probability of an event based on its prior knowledge using below formula:*

$$P(H\ |E) = \frac{P(E\ |H) * P(H)}{P(E)}$$

In plain English, using Bayesian probability terminology, the above equation can be written as:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Pros:

- *It is easy and fast to predict class of test data set. It also performs well in multi class prediction.*
- *When assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression and you need less training data.*
- *It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).*
- *The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features.*

Cons:

- *If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.*
- *On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.*
- *Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.*

Naïve Bayes Classification in Weka

Naïve bayes classification in Weka allows attributes numeric/nominal. However, only nominal attributes give better results as it can be noticed in comparison below.

Before transormation

```
=== Summary ===

Correctly Classified Instances         149               74.5    %
Incorrectly Classified Instances        51               25.5    %
Kappa statistic                          0.3657
Mean absolute error                      0.2879
Root mean squared error                  0.4129
Relative absolute error                 70.6169 %
Root relative squared error             93.9316 %
Total Number of Instances              200

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,799    0,412    0,850      0,799    0,824      0,368    0,796     0,923     good
                 0,588    0,201    0,500      0,588    0,541      0,368    0,796     0,539     bad
Weighted Avg.    0,745    0,358    0,761      0,745    0,751      0,368    0,796     0,825

=== Confusion Matrix ===

   a   b   <-- classified as
 119  30 |   a = good
  21  30 |   b = bad
```

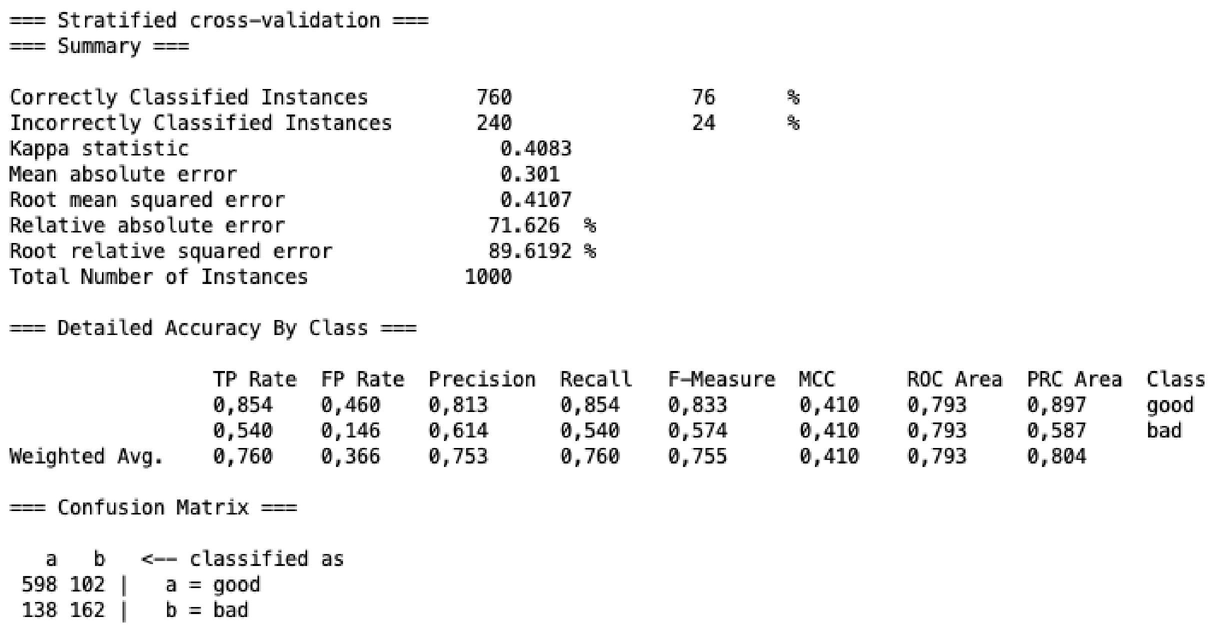
After transformation

```
=== Summary ===

Correctly Classified Instances         156               78      %
Incorrectly Classified Instances        44               22      %
Kappa statistic                          0.4561
Mean absolute error                      0.2841
Root mean squared error                  0.3934
Relative absolute error                 69.6645 %
Root relative squared error             89.5135 %
Total Number of Instances              200

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,819    0,333    0,878      0,819    0,847      0,460    0,816     0,931     good
                 0,667    0,181    0,557      0,667    0,607      0,460    0,816     0,545     bad
Weighted Avg.    0,780    0,295    0,796      0,780    0,786      0,460    0,816     0,833

=== Confusion Matrix ===

   a   b   <-- classified as
 122  27 |   a = good
  17  34 |   b = bad
```

Dataset split (80%/20%) is not the best tool for evaluation of the classification model. Much better is Cross-validation for multiple of reasons.

- When a dataset like “German Credit Dataset” has only 1000 instances what is not enough, because only 20 instances are test data. Based on statistics principles, we can get almost any performance on this set only due to chance. If we use cross-validation in this case, we build K different models, so we are able to make predictions on all of our data. For each instance, we make a prediction by a model that didn’t see this example, and so we are getting 1000 examples in our test set.
- As mentioned in above, when we create five different models using our learning algorithm and test it on five different test sets, we can be more confident in our algorithm performance. When we do a single evaluation on our test set, we get only one result. This result may be because of chance or a biased test set for some reason. By using Cross-Validation, we are able to get more metrics and draw important conclusion both about our algorithm and our data.

Bringing back German Credit Dataset and comparing both ways of model validation we can notice that simple data split has better results. It does not mean that it is better. It just means that there the result in it is biased.



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

Correctly Classified Instances         760               76      %
Incorrectly Classified Instances       240               24      %
Kappa statistic                          0.4083
Mean absolute error                      0.301
Root mean squared error                  0.4107
Relative absolute error                 71.626  %
Root relative squared error             89.6192 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,854    0,460    0,813      0,854    0,833      0,410    0,793     0,897     good
                 0,540    0,146    0,614      0,540    0,574      0,410    0,793     0,587     bad
Weighted Avg.    0,760    0,366    0,753      0,760    0,755      0,410    0,793     0,804

=== Confusion Matrix ===

   a   b   <-- classified as
 598 102 |   a = good
 138 162 |   b = bad
```

Accuracy explanation

TP Rate(True Positive Rate also called sensitivity): Calculated as TP/TP+FN. TPR is the probability that an actual positive will test positive.

FP Rate(False Positive Rate): Calculated as FP/FP+TN, where FP is the number of false positives and TN is the number of true negatives (FP+TN being the total number of negatives). It’s the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

Precision: Ratio between the True Positives and all the Positives. $\frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$

Recall: Measure of model correctly identifying True Positives. $\frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$

F-Measure: Harmonic mean of the Precision and Recall. $2 * \frac{Precision * Recall}{Precision + Recall}$

DECISION TREE

Theory of Decision Tree

The Decision Tree Algorithm belongs to the family of supervised machine learning algorithms. The goal of this algorithm is to create a model that predicts the class of a target, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Assumptions that we make while using the Decision tree:

- In the beginning, we consider the whole training set as the root.
- Feature values are preferred to be categorical, if the values continue then they are converted to discrete before building the model.
- Based on attribute values records are distributed recursively.
- We use a statistical method for ordering attributes as a root node or the internal node.

Before going to the Information Gain it is important to understand **entropy**, which is the measures of impurity, disorder, or uncertainty in a bunch of examples. Entropy as well as Gini Inpurity is a cost function that controls how a Decision Tree decides to split the data. It affects how a Decision Tree draws its boundaries. “Entropy values range from 0 to 1”, Less the value of entropy more it is trusting able.

$$H(s) = -probablity\ of\ \log_2(p+) - -probability\ of\ \log_2(p-)$$

Information Gain: Is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. A commonly used measure of purity is called information.

$$Gain(S, A) = H(s) \sum \frac{/Sv/}{/s/} H(Sv)$$

Gini Impurity: is a measurement used in CART (Classification and Regression Tree) methods to build Decision Trees to determine how the features of a data set should split nodes to form the tree. More precisely, the Gini Impurity of a data set is a number between 0-0.5, which indicates the likelihood of new, random data being miss classified if it were given a random class label according to the class distribution in the data set.

The difference between Gini Impurity and Information Gain is that the maximum value for entropy is 1 whereas the maximum value for Gini impurity is 0.5. As the Gini Impurity does not contain any logarithmic function to calculate it takes less computational time as compared to entropy.

$$\mathrm{Gini}(K) = \sum_{i \in N} P_{i,K}(1 - P_{i,K}) = 1 - \sum_{i \in N} P_{i,K}^2$$

- N is the list of classes (In this case $N = \{\text{'Yes'}, \text{'No'}\}$)
- K is the category
- $P_{i,K}$ is the probability of category K having class i

Lastly Gini Impurity has Gini index which combines the category noises together to get the feature noise. Gini Index is the weighted sum of Gini Impurity based on the corresponding fraction of the category in the feature. The formula is:

$$I_{\mathrm{Gini}}(a) = \sum_{k \in M} P_{k,a} \cdot \mathrm{Gini}(k)$$

- a is the feature
- M be the list of all categories in feature a
- $P_{k,a}$is the fraction of category k in feature a

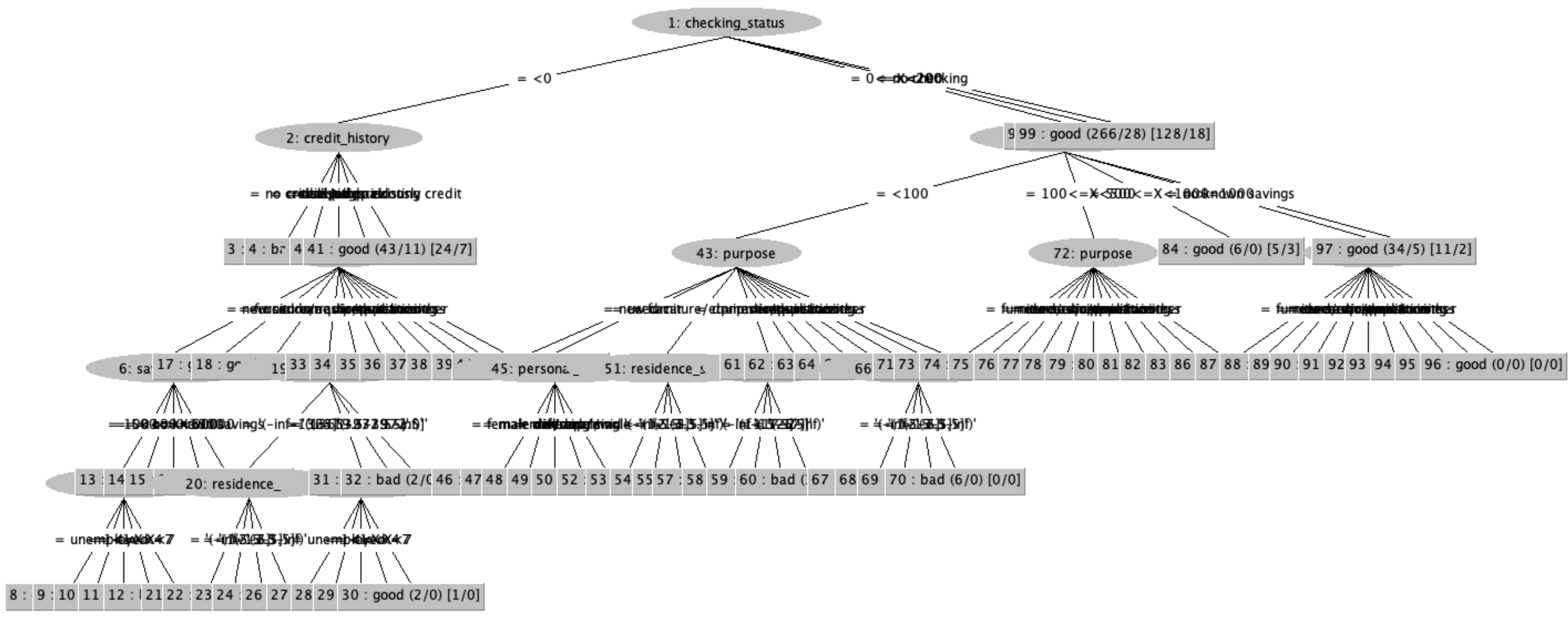
Decision Tree in Weka

Among many classifying algorithms there is REPTree, which is a method to generate a decision tree from a given dataset. It is considered to be an extension of C45 by improving the pruning phase by using Reduced Error Pruning (REP). For every subtree it checks whether the subtree could be replaced by a single node, without lowering the performance of the classifier on this pruning set. As such, the pruning method is simple, but often considered to be too agressive, ie. it might remove subtrees which are actually relevant.

Comparing results of German Credit Dataset we see that Precision, which is the most relevant in this case, is lower than in results of Naïve Bayes algorithm.

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

Correctly Classified Instances         715               71.5    %
Incorrectly Classified Instances       285               28.5    %
Kappa statistic                          0.2662
Mean absolute error                      0.3457
Root mean squared error                  0.4536
Relative absolute error                 82.2638 %
Root relative squared error             98.977  %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,853    0,607    0,766      0,853    0,807      0,272    0,703     0,815     good
                 0,393    0,147    0,534      0,393    0,453      0,272    0,703     0,450     bad
Weighted Avg.    0,715    0,469    0,697      0,715    0,701      0,272    0,703     0,706

=== Confusion Matrix ===

   a   b   <-- classified as
 597 103 |   a = good
 182 118 |   b = bad
```

Weka also gives option to visualize a decision tree, however in case of German Credit Dataset, the number of attributes is too big and the decision tree in too expand to visualize it easily.



K-Nearest Neighbors

Theory of K-Nearest Neighbors

K Nearest Neighbors is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbors are classified.

‘k’ in KNN is a parameter that refers to the number of nearest neighbors to include in the majority of the voting process. Choosing the right value of K is a process called parameter tuning and is important for better accuracy. Finding the value of k is not easy. It is needed to find out the best value of k by trial-and-error process and assuming that training data is unknown. Choosing smaller values for K can be noisy and will have a higher influence on the result. On the other hand, larger values of K will have smoother decision boundaries which mean lower variance but increased bias. Also, computationally expensive. In general, practice, choosing the value of k is **k = sqrt(N)** where N stands for the number of samples in your training dataset. It is better to keep the value of k odd number, in order to avoid confusion between two classes of data.

K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen” observation. Similarity is defined according to a distance metric between two data points. A popular one is the Euclidean distance method showed below, however there are Other methods are Manhattan, Minkowski, and Hamming distance methods. For categorical variables, the hamming distance must be used.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Pros of KNN

- Simple to implement
- Flexible to feature/distance choices
- Naturally handles multi-class cases
- Can do well in practice with enough representative data

Cons of KNN

- Need to determine the value of parameter K (number of nearest neighbors)
- Computation cost is quite high because we need to compute the distance of each query instance to all training samples.
- Storage of data
- Must know we have a meaningful distance function.

K-Nearest Neighbors in Weka

As the German Credit dataset counts 1000 instances, good choice seems to 'K' value at 31 because it's an odd number and is close to root of 1000. Although KNN algorithm is created generally for nominal attributes, it allows to apply this algorithm also to nominal variables even though it does not offer hamming distance. Bearing that in mind, and the fact that in general KNN algorithm works better with balanced (dataset where target class has the same number of positive and negative values), result of 0,711 of Precision seems to be a satisfactory outcome. It is not the highest result gotten in the process, however it is still better than decision tree outcome.

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

Correctly Classified Instances         730               73      %
Incorrectly Classified Instances       270               27      %
Kappa statistic                          0.2114
Mean absolute error                      0.3474
Root mean squared error                  0.4184
Relative absolute error                 82.6782 %
Root relative squared error             91.3    %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,946    0,773    0,740      0,946    0,831      0,257    0,758     0,881     good
                 0,227    0,054    0,642      0,227    0,335      0,257    0,758     0,544     bad
Weighted Avg.    0,730    0,558    0,711      0,730    0,682      0,257    0,758     0,780

=== Confusion Matrix ===

   a   b   <-- classified as
 662  38 |   a = good
 232  68 |   b = bad
```

Summary

Summarizing the obtained results of three different algorithms. It can be said that Naive Bayes performed best giving Precision equal to 75,3%, followed by K-Nearest Neighbors obtaining a Precision result of 71,1% and lastly Decision Tree giving a Precision result of 69,7%.

Lastly, last row was separated from original dataset and a class attribute was hidden to check how Weka determines unknown class of the test instance. As it is seen below, using the best resulting Naïve Bayes algorithm, Weka correctly classify last row as "good" value.

```
=== Re-evaluation on test set ===

User supplied test set
Relation:     german_credit-weka.filters.unsupervised.attribute.Discretize-F-B4-M-1.0-Rfirst-last-precision6
Instances:     unknown (yet). Reading incrementally
Attributes:   21

=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1        1:?      1:good       0.643

=== Summary ===

Total Number of Instances                0
Ignored Class Unknown Instances                  1
```

Apriori Algorythm

Theory of Apriori Algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Pros of Apriori Algorithm:

- *This is the most simple and easy-to-understand algorithm among association rule learning algorithms*
- *The resulting rules are intuitive and easy to communicate to an end user*
- *It doesn't require labeled data as it is fully unsupervised; as a result, you can use it in many different situations because unlabeled data is often more accessible*
- *Many extensions were proposed for different use cases based on this implementation—for example, there are association learning algorithms that take into account the ordering of items, their number, and associated timestamps*
- *The algorithm is exhaustive, so it finds all the rules with the specified support and confidence*

Cons of Apriori Algorithm:

- *A large number of itemsets in the Apriori algorithm dataset.*
- *Low minimum support in the data set for the Apriori algorithm.*
- *The time needed to hold a large number of candidate-sets with many frequent itemsets.*
- *Thus it is inefficient when used with large volumes of datasets.*

Support - *Fraction of transactions that contain an itemset. For example, the support of item I is defined as the number of transactions containing I divided by the total number of transactions.*

$$support(I) = \frac{Number\ of\ transactions\ containing\ I}{Total\ number\ of\ transactions}$$

Confidence - *Measures how often items in Y appear in transactions that contain X. Confidence is the likelihood that item Y is also bought if item X is bought. It's calculated as the number of transactions containing X and Y divided by the number of transactions containing X.*

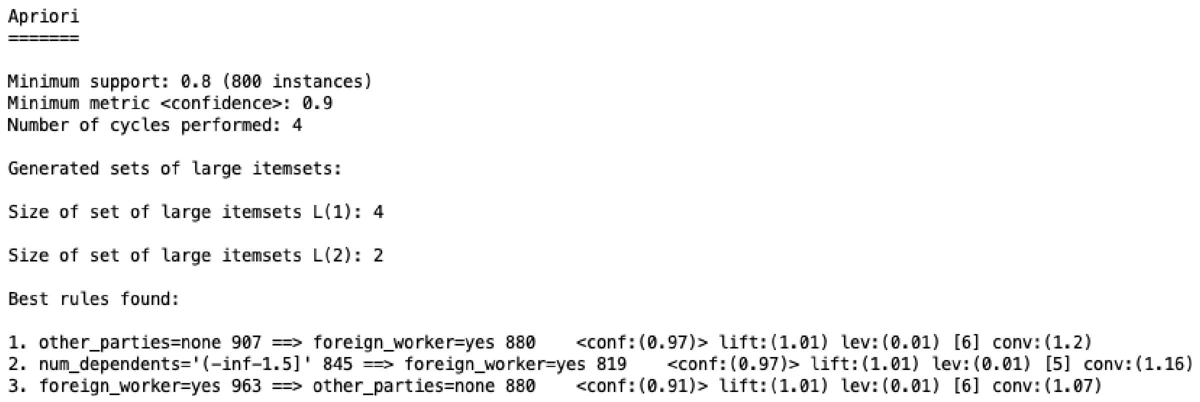
$$confidence(X \rightarrow Y) = \frac{Number\ of\ transactions\ containing\ X\ and\ Y}{Number\ of\ transactions\ containing\ X}$$

Lift - Lift(A -> B) refers to the increase in the ratio of sale of B when A is sold. A Lift of 1 means there is no association between products A and B. Lift of greater than 1 means products A and B are more likely to be bought together. Finally, Lift of less than 1 refers to the case where two products are unlikely to be bought together.

$$Lift = \frac{Support}{Support(X) * Support(Y)}$$

Apriori Algorithms in Weka

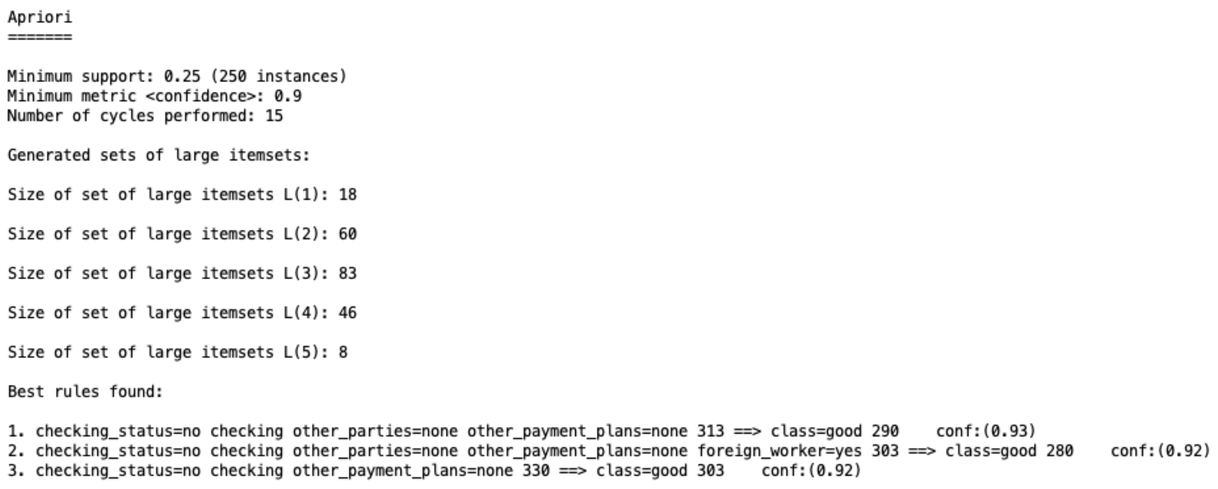
Apriori Algorithm in Weka uses a bit different measures of interest for association rules. Beside common confidence measure which is a base measure when deciding which rule is the best, Weka presents leverage and conviction measures instead of support. ***Leverage*** *measures the difference of X and Y appearing together in the data set and what would be expected if X and Y where statistically dependent. The rational in a sales setting is to find out how many more units (items X and Y together) are sold than expected from the independent sells.* ***Conviction*** *compares the probability that X appears without Y if they were dependent with the actual frequency of the appearance of X without Y.*



```
Apriori
=======

Minimum support: 0.8 (800 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 4

Generated sets of large itemsets:

Size of set of large itemsets L(1): 4

Size of set of large itemsets L(2): 2

Best rules found:

1. other_parties=none 907 ==> foreign_worker=yes 880    <conf:(0.97)> lift:(1.01) lev:(0.01) [6] conv:(1.2)
2. num_dependents='(-inf-1.5]' 845 ==> foreign_worker=yes 819    <conf:(0.97)> lift:(1.01) lev:(0.01) [5] conv:(1.16)
3. foreign_worker=yes 963 ==> other_parties=none 880    <conf:(0.91)> lift:(1.01) lev:(0.01) [6] conv:(1.07)
```

While analyzing best three rules found by Weka, we notice that the biggest association between attributes was ("other_parties" and "foreign_worker") as well as ("num_dependents" and "foreign_worker"). Weka have found that when there is no other debtors or guarantors to the credit, the customer is the most probably a foreign worker. Similarly, it works the other way that when a customer is a foreign worker, the most probably his credit has no other guarantors. Weka points out also that, when a number of people being liable to provide maintenance for is equal to 1(the customer maintenances on his own), then he is the most probably a foreign worker.

However, as the German Credit Dataset is focused to decide if a customer is good or bad, it is better to focus Apriori Algorithm on finding associations with class attribute. From the result below, it can be noticed that associations was found that when a customer has no checking account and he has no other payment plans, the customer is probably a good one. Moreover if this customer's credit has no other debtors or guarantors we can be more sure that he is a good customer.



```
Apriori
=======

Minimum support: 0.25 (250 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 15

Generated sets of large itemsets:

Size of set of large itemsets L(1): 18

Size of set of large itemsets L(2): 60

Size of set of large itemsets L(3): 83

Size of set of large itemsets L(4): 46

Size of set of large itemsets L(5): 8

Best rules found:

1. checking_status=no checking other_parties=none other_payment_plans=none 313 ==> class=good 290    conf:(0.93)
2. checking_status=no checking other_parties=none other_payment_plans=none foreign_worker=yes 303 ==> class=good 280    conf:(0.92)
3. checking_status=no checking other_payment_plans=none 330 ==> class=good 303    conf:(0.92)
```

Bibliography

1. https://www.kaggle.com/uciml/german-credit
2. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
3. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
4. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
5. https://www.geeksforgeeks.org/naive-bayes-classifiers/
6. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
7. https://www.analyticsvidhya.com/blog/2021/02/machine-learning-101-decision-tree-algorithm-for-classification/
8. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html
9. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
10. https://data-en-maatschappij.ai/en/tools/method-reptree
11. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
12. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
13. https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/
14. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
15. https://www.geeksforgeeks.org/apriori-algorithm/
16. https://www.javatpoint.com/apriori-algorithm-in-machine-learning
17. https://en.wikipedia.org/wiki/Apriori_algorithm
18. https://www.softwaretestinghelp.com/apriori-algorithm/
19. https://towardsdatascience.com/the-apriori-algorithm-5da3db9aea95