**An example of ML Algorithms for Bank Credit Scoring using WEKA**

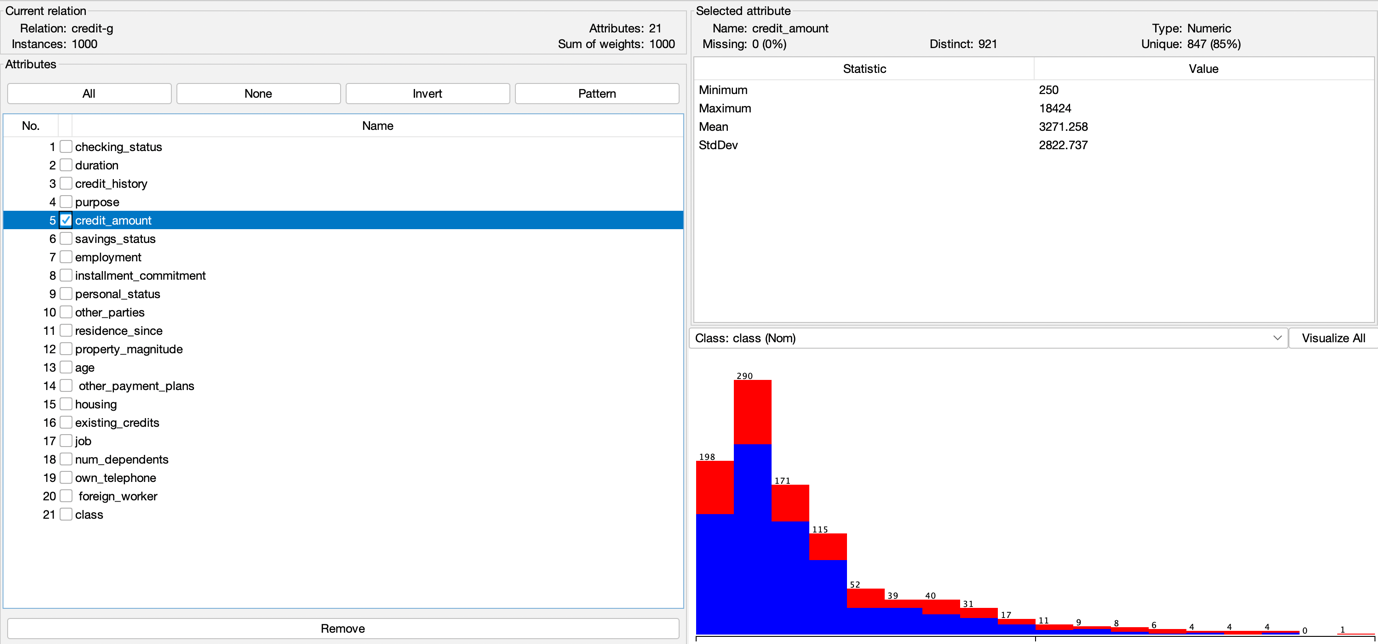
**The business context**

One of the most essential functions of the bank is to provide credit scoring decisions with a minimum of credit-loss rates. Nowadays high-performance models allow financial institutions to define lending parameters more precisely in approving creditworthy customers and rejecting proposals from risky customers (Dash *et al*., 2021). To build even the most primitive model for credit scoring we need some amount of the data and the ability in prioritizing data sources.

For example, the elements for the algorithm should be described by one or more attribute variables and by a single class variable to predict the result. In other words, the features contain data on the customers and on the previous credit report of the customers, if they currently pay for other loans, if they have got a job with an acceptable salary, if they are married and other details. The class value provides information on the short answer – to give loan to the customer or not. Ince & Aktan (2009) examined four different approaches that could be applied to explore credit scoring and evaluate the bank’s credit card policy. The experiment showed that the *Decision Tree (CART)* provided the best average correct classification rate than others – 65.58%, followed by *Logistic Regression* – 62.33%, *Discriminant Analysis* – 62.20% and *Neural Networks* with the prediction accuracy of 61.52%.

In this research we should gain even better results classification rate with WEKA – a collection of machine learning algorithms for data mining tasks.

**The dataset**

There is a quite popular and useful open dataset for the beginners in AI on the internet – German Credit Bank Data by Professor Dr. Hans Hofmann from 1994. Despite its advantaged age, most of the data characteristics are still actual for banks. It contains information about 1000 debtors, described with 20 variables: 13 of them are nominal and 7 are numeric. Nominal data can be both qualitative and quantitative – it is classified without a natural order or rank, while numerical data will always be a number that can be measured. For instance, numeric features in the current credit scoring dataset are duration (from 4 to 72 months) or age (between 19 and 75 years old). The nominal variables are purpose of the loan (car, education, business, etc.), job (skilled, unskilled, high qualification, unemployed, etc.), personal status (a single male, a married man, female, etc.). The classification result looks just like “good” or “bad” regarding the trustworthiness of the potential client. This dataset is available at UCI Machine Learning Repository, a link is in the reference list. We chose the German Credit Bank Data because it is not huge to be stuck with processing for some time, but it has enough attributes to decide which of them is more important than the others when we develop a prediction model, in other words, to provide feature selection. According to the dataset, the minimal amount of approved credit was 250 deutschmarks, and the maximum amount was more than 18,4 thousand deutschmarks (in the picture).   


**First preparations for developing a prediction model**

First of all, we load the dataset to the WEKA and verify the conformity for every variable and for the class, if is nominal or numeric. We are able to tick and change status with the *Filter* tool if it is needed. After that it would be nice to track this raw dataset through a few of the basic WEKA classification tools.

We are get started with using ***ZeroR*** classifier to establish a baseline. According to Witten (1999), it is the most primitive learning scheme in WEKA that can predict the majority class in the training data for problems with a categorical class value (which is our case), and the average class value for numeric prediction problems (not our case). It is useful to generate a baseline performance to compare the other learning schemes to that, because any that does better than it has “skill” on the problem. As a result, we have got the proportion of correctly and incorrectly classified instances like 70% and 30% respectively. For the test mode WEKA uses a K-fold cross-validation procedure, providing a good general estimate of model performance with folds of 10, meaning each fold will contain about 1000/10 or 100 examples.

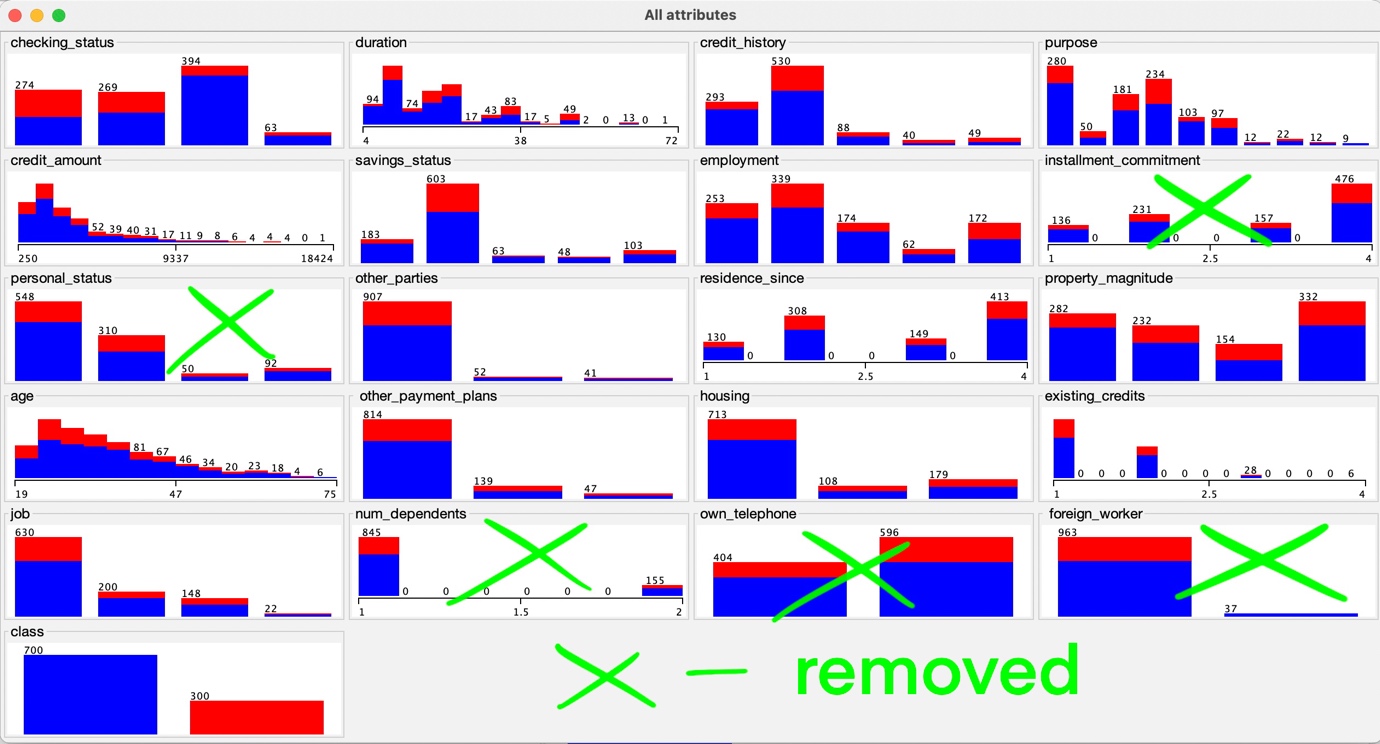
We go further and apply to the dataset ***J48 pruned Decision tree*** classifier model: it shows slightly better results with 70.7% correctly classified and 29.3% incorrectly classified instances. With setting *True* for *Reduce Error Pruning* in *Properties* we evaluate the result for 72.5/27.5% proportion. By the way, with shifting this model to ***J48 unpruned Decision tree***the ratio is even worse than a baseline – 68.3% and 31.7%, that means that pruned tree model is more appropriate for our dataset.

We are proceeding with the experimentation and choose ***Naive Bayes*** classifier and finally in this case we see 75.4% of correctly classified instances and 24.6%

incorrectly classified. It is better, but we are still not absolutely satisfied with the results.

**Data Treatment & Feature Evaluation**

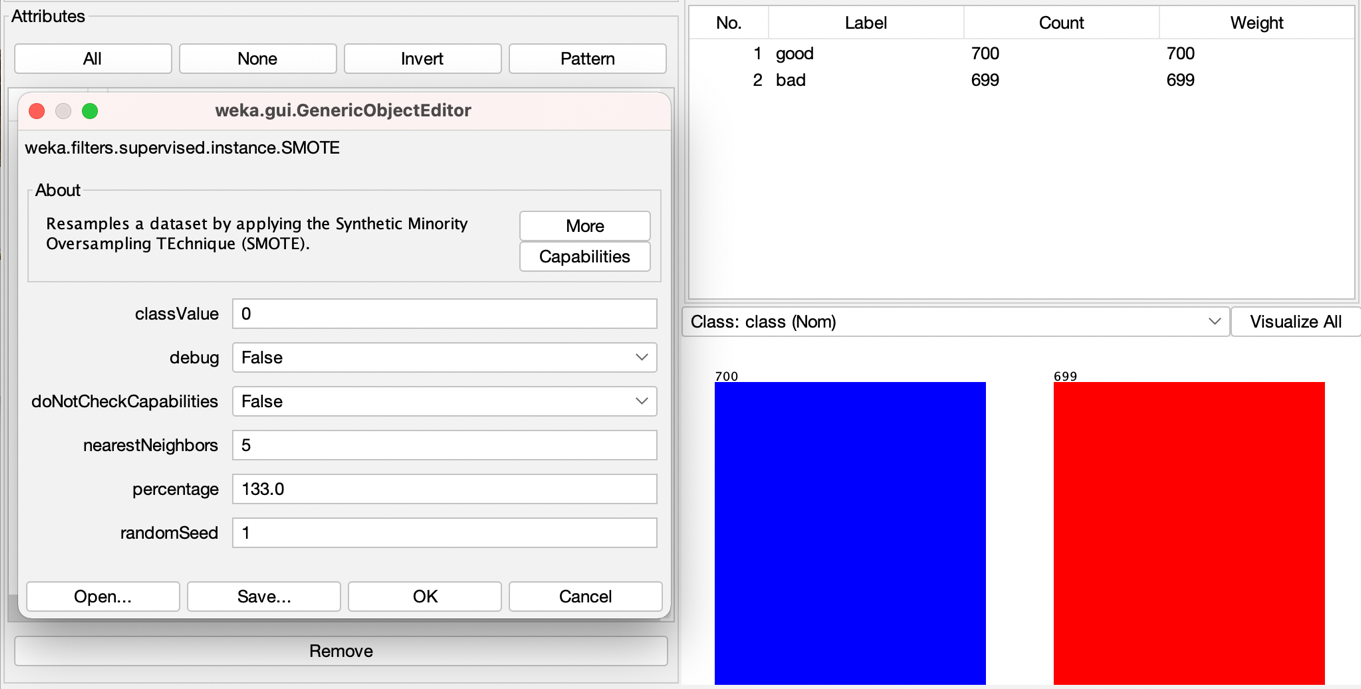
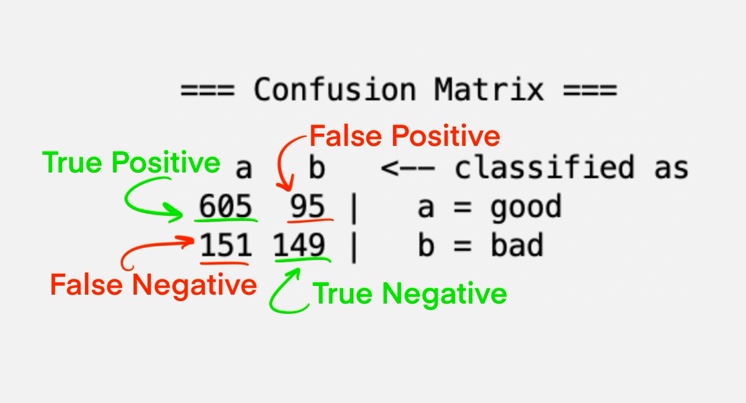
We are not banking specialists; however, we keep in mind that our experimental dataset is from 1994. Once we have tested it, next comes the most tedious task to select the most important and actual attributes. For that we should provide evaluation – measuring the results of a learning system to increase the predictive performance, identifying the best-suited learning algorithm in comparison with the others. For the beginning let WEKA use the automatic tool for this special job. In *Preprocess* mode choose *Filters – Supervised – Attribute – Attribute Selection* and click the *Apply* button. We have got four main attributes as an outcome in the list, three of them are variables (checking status, duration, credit history), and one is a class. Now we know – we are not going to move them since we have another look at the rest of the initial list. For example, the “own telephone” option probably seems odd in the 21st century, “foreign worker” has a very low number of cases for “no” (just 37 from 1000), “personal status”, “installment commitment”, and “num dependents” look not very informative according to the classification ratio, therefore let us dare remove them.



After removing 5 attributes from the list ***ZeroR*** performance has the same 70/30% ratio, ***J48 pruned Decision tree*** performs +1.8% percentage points better for correctly classified instances (72.5%), as well as ***J48 unpruned Decision tree*** results raise +1.3 percentage points (69.6%). However ***Naive Bayes*** classifier performance drops down -0.9 percentage points (74.5%). Thus, for the reduced list of attributes for this dataset ***Decision Tree*** is a better option, while for the original list ***Naive Bayes*** would be more relevant.

**Imbalanced data problem**

This is time for a very essential question: why all the outcomes from the tests above revolved more or less around a 70/30 ratio? The answer is an imbalanced dataset – the disproportional data has some risks for overfitting model and algorithm bias. Looking through *Class* attribute we realize that German Credit Bank Data is quite imbalanced. There are two classes, for “good” customers and for “bad” customers, however they are not represented in the dataset as 50/50. The truth is that 70 percent of the examples are “good” customers (a majority, negative for potential problems), whereas the remaining 30 percent of examples are “bad” (a minority, positive for possible troubles). The dataset is provided with the *cost matrix* that gives penalties to each misclassification error. In our case marking a “bad” customer as “good” is considered like *false negative* and costs 5, while marking a “good” customer as “bad” costs 1 which is assigned for *false positive*. That means it is more costly to the bank to give money to a non-reliable customer than to not give money to a reliable one. There is another *confusion matrix* sensitive measure in WEKA – it is similar to the *cost matrix* except the fact that we are calculating the cost of wrong prediction or right prediction. We are able to count predicted output results at the best WEKA performance we have got before with ***Naive Bayes*** (75,4%).

According to the *confusion matrix,* the model predicts 756 as “good” with 151 mistakes and 244 as “bad” with 95 mistakes. That means our model is ready to give a loan to 151 unreliable customers which is probably not perfect for the bank. Misclassification errors on the minority class are more important than others for such kinds of imbalanced classification tasks. The good news is we have tools to fix imbalanced dataset like this. An easy way is to remove the data on 400 “good” customers from 700 to get 300/300 instances for training, however it looks that we lost a big amount of data. Another proposed solution is applying an oversample technique like SMOTE is built into WEKA software. SMOTE (Synthetic Minority Oversampling Technique) – is a filter used to increase the minority group when such a disproportion occurs. In *Preprocess* mode we choose *Filters – Supervised – Instance – SMOTE* anddouble click on SMOTE to customize. After that we maintain the *Class Value* at zero to let the machine identify the minority group, set *Percentage* at 133% to increase the minority class to a new equal proportion of 700/699, and then click *Apply*.

According to Chawla *et al.* (2002) with this approach the minority class is over-sampled by creating “synthetic” examples rather than by over-sampling with replacement or “copy-pasting” straightforward techniques. The synthetic examples cause the classifier to create larger and less specific decision regions.

The next important step is to apply randomization to the SMOTE results. In *Preprocess* mode choose *Filters – Unsupervised – Instance – Randomize* andclick on *Apply*.

Applying ***ZeroR*** to the rebalanced dataset we have got the result of 50.0357% correctly classified instances and 49.9643% incorrectly classified instances, we are absolutely satisfied with that. ***J48 pruned Decision tree*** performs +8.1 percentage points better than raw dataset (78.842%) which is the best accuracy result by far. Also, we have obtained impressive results for ***J48 unpruned Decision tree***: +8.3 percentage points (76.6262%). The ***Naive Bayes*** with our new balanced dataset is able to classify about +3,5 percentage points better (78.7706%).

Although the SMOTE preprocessing algorithm is considered “de facto” as a standard in the framework of learning from imbalanced data, during the last 15 years researchers have been presenting other significant SMOTE-based approaches: Borderline-SMOTE, ROSE, MWMOTE, etc.

However, there are still some current challenges, Fernández et. al (2018) highlighted among them the need for enhancing the treatment of small disjuncts, noise, lack of data, overlapping, dataset shift and “the curse of dimensionality”. In other words, the ML designer should be careful in using SMOTE.

**Conclusion**

We thereby set an example of some possible ways to improve the Machine Learning algorithm for Bank Credit Scoring in case we have got a small imbalanced dataset. There are a lot of different techniques to apply for better performance, as well as many bigger datasets exist to put them into research. Because of the limit to 2000 words, we have presented just the general description, however we outperformed Ince & Aktan better result, where the best correct classification rate in 2009 was 65.58%. Using SMOTE techniques, we have increased this level of accuracy to 78.842% and we are getting more.

**References:**

Dash, R. Kremer, A. & Petrov, A. (2021) Designing next-generation credit-decisioning models. *McKinsey & Company.* Available from:

https://www.mckinsey.com/business-functions/risk-and-resilience/our-insights/designing-next-generation-credit-decisioning-models [Accessed 22 May 2022]

Ince, H. & Aktan, B. (2009) A comparison of data mining techniques for credit scoring in banking: A managerial perspective. *Journal of Business Economics and Management*, 10(3), pp.233-240.

UCI Machine Learning Repository. German dataset. Available from: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data) [Accessed 15 May 2022]

Witten, I.H., Frank, E., Trigg, L.E., Hall, M.A., Holmes, G. & Cunningham, S.J. (1999) *Weka: Practical machine learning tools and techniques with Java implementations.* Working Paper Series.

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence resear*ch, 16, pp.321-357.

Fernández, A., Garcia, S., Herrera, F. & Chawla, N.V. (2018) SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61, pp.863-905.