University of Zurich

Machine Learning in Finance

Group Project Summary

Authors:

Mahamoud Farah

Diego Hager

Franck Huber

Sharon Vögeli

Date of Submission: 14.04.2019

Table of Contents

1. Algorithms 3
1.1. *k*-Nearest Neighbors 3
1.2. Random Forest 3
1.3. Support Vector Machines 3
1.4. Neural Network 4
1.5. Further Techniques Applied 4
1.5.1. Boosting 4
1.5.2. Bagging 4
2. Data 5
2.1. Dataset 5
2.2. Feature Selection 5
2.3. NaN Values 6
3. Results 7
3.1. Process 7
3.2. Performance 7
3.3. Difficulties 7
References 9

1. Algorithms

We have decided to use the algorithms *k*-nearest neighbors, random forest, support vector machines and neural network in our group project. This is mainly due to the individual advantages of these models as well as the successful applications with similar data described in the literature. In the following, the decisive aspects of the algorithms are outlined.

1.1. *k*-Nearest Neighbors

The *k*-nearest neighbors [KNN] algorithm has several advantages and is widely applied. The reason for this is among different properties, as non-parametric or that only one parameter *k* is needed, the simplicity of this method (Abu-Aisheh, Raveaux, and Ramel (2018) and Batista and Silva (2009)). In contrast to neural networks, the user knows exactly what the KNN algorithm does. Here a database is searched with a distance function for the most similar element to a given query element (Angelini, Tollo, and Roli (2008) and Batista and Silva (2009)). However, this requires a lot of time for a large training set (Abu-Aisheh, Raveaux, and Ramel (2018)).

1.2. Random Forest

In their research paper the authors Khandani, Kim, and Lo (2010) use generalized classification and decision trees [CART] to predict the credit-risk of consumers via machine learning algorithms. They use this model because it produces precise decision rules that can be easily understood using a tree. Furthermore, this model can be used for high-dimensional feature spaces. In this case an additional pruning criterion, the Gini measure, can be used to stop the expansion of the tree in order to prevent overfitting the training data. The authors propose a further technique, the so-called boosting, to improve the predictive power when dealing with skewed data (Khandani, Kim, and Lo (2010)). Rather than having all data in the training set weighted equally, the rare occurrences receive more weight than the popular ones (Freund and Schapire (1996)).
Another paper employs the same CART algorithm to identify defaults in mortgage-loan data. The authors obtain with this model the best estimations for default in comparison to other algorithms, such as *k*-nearest neighbors or neural networks (Galindo and Tamayo (2000)).

1.3. Support Vector Machines

The Support Vector Machines [SVM] are explained and applied in several papers. SVM has proven to be a clearly superior method, particularly in the context of high-dimensional data (Wu, Hu, and Huang (2014)). For instance, the authors Shin, Lee, and Kim (2005) use this method in the context of corporate bankruptcy prediction. They achieve significantly better results than with backpropagation neural network [BNN] when the training set size decreases. Other authors Huang et al. (2004) apply both SVM and BNN for the analysis of corporate credit ratings. With these two methods they were able to achieve prediction accuracy of 80% in two different markets. The two artificial intelligence methods performed similarly, but significantly better than traditional statistical methods as discriminant analysis, logistic regression or ordinary least squares.

1.4. Neural Network

Neutral networks were repeatedly used in the context of bankruptcy prediction for credit risk. According to the literature, this method has clearly outperformed other techniques. Moreover, additional ways are being sought to further improve performance, for example through better training methods, architecture selection or inputs (Atiya (2001)). The use of neural networks has also proven to be an efficient method for credit scoring (Pang, Wang, and Bai (2002) and Wu and Wang (2000)). This is mainly due to the fact that neural networks are learning systems that can model the nonlinear relationship between input and output data. However, this method has a black-box nature, which makes it difficult to understand the internal process (Angelini, Tollo, and Roli (2008)).

1.5. Further Techniques Applied

1.5.1. Boosting

We used two versions of boosting algorithms for our script. The GradientBoostingClassifier, which applies a logistic regression as default as well as the AdaBoostClassifier, which uses an exponential function. Both have RND-forest as base learner. Those methods should improve the results. Boosting works sequentially, considering previously trained algorithms and assigning weights to falsely predicted observations. In the end, boosting will generate a weighted mean, where models with better scores are weighted more. Because of the sequential data analysis, boosting does not account for over fitting, yet it could decrease bias within the model.

1.5.2. Bagging

Bagging is also an improvement algorithm with base learners. We took several algorithms as base learners, to try which one works best. In contrast to boosting, bagging does not assess the differences within feature vectors. It makes several predictions with different estimators and then takes the average over those trained models. Because bagging takes an average over different trained models, it could decrease over fitting problems.

2. Data

2.1. Dataset

The dataset consists of the monthly ratios and ratings of the companies listed in the S&P 500 index. The observation period ranges from January 2010 to January 2017. We decided to consider the maximum range since we're interested in the long term credit rating of the companies and a shorter range could possibly lead to a bias.

In order to create our data set, we used the SP_Ratings_GroupProject2019 query, Financial Firm Ratios from Wharton and the SP500_CompanyList.csv file from OLAT. The ratings and the company list could be merged on the ticker, Ratios were merged on gvkey and public_date. After that rows with missing values for the column 'splticrm' were thrown out. The resulting file is named SP500_data_new.csv and has 30'804 rows.

Within our script we worked with two data frames. In the first data frame (data_NaN), we dropped all observations with missing values. For the second data frame (data_y), we used the SimpleImputer function to handle the NaN's, this is discussed in more details in the chapter 2.3. on the topic of NaN Values.

We looked for high correlations within the features on the label vector. There are higher correlations in the sample data_NaN, than in the other sample. It could be that NaNs are not distributed randomly. They might appear in several entries per observation vector. When counting NaNs for every column, we see a pattern which strengthens this hypothesis. Most columns have either 500-600, 2100-2300 or roughly 5000 missing values.

2.2. Feature Selection

In the context of the selection of adequate input parameters, we have found an instructive paper. The authors Hajek and Michalak (2013) discuss how the feature selection in corporate credit rating prediction should be approached. For this they use the so-called wrapper approach, which describes an iterative selection mechanism that picks specific features based on performance evaluation. What is particularly striking in this paper is that the authors emphasize the importance of feature selection several times. This is because the dimension of the feature space can be narrowed, learning algorithms are faster and classification precision is improving.

Therefore, we have adopted this professional example of Hajek and Michalak (2013) and selected similar features for our script. Furthermore, we applied the feature selection function from Random Forest. This process is described in more detail in the chapter 3.1 on the subject of process.

2.3. NaN Values

In order to solve the problem regarding the NaN values, we decided to apply two datasets. We applied the *SimpleImputer* function from the Scikit-Learn package for the first dataset (data_y). It's a relatively straightforward application. We set the strategy as 'median' and let the function delete the entire column if it only consisted of NaN values.

In addition, we created a second dataset where all NaNs were dropped (data_NaN). This would make it possible to compare the different scores and see which approach is better suited. The second dataset seems to have overall better results, measured by the mean accuracy. This could be due to the distribution of NaNs within sample.

3. Results

3.1. Process

We tried the algorithms on default settings to get an idea which algorithms could be best for our purposes. Therefore, we directly used a principal component analysis and resampling. Resampling was necessary because in the data, where we just dropped all missing Label values, we had very few entries for low ratings. We decided to not use synthetic samples, because our algorithms performed very poorly. Hence, a synthetic sample based on such an algorithm would introduce even more estimation errors. That is why, we simply used copies to resample our data.

Moreover, we wanted to look at how features impact the algorithms and if we might be able to get the same scores with less features. Thus, we used the examples from Hajek and Michalak (2013) and implemented similar features in our script. In addition, we used the feature selection function from the random forest algorithm to get features. We took the 40 most important features and selected therefrom, those better than mean, median and the best eight. In order to compare the results with the results from PCA, we took the best eight. Where we decided to take also eight PCA components. We also made random feature samples for the parameter tuning. We wanted to use the GridSearch results from those random features to see if there are substantial differences of the best parameters within the algorithms with different features.

3.2. Performance

During the development of the script, the results were acceptable and improving. However, during the last phase of the development, an error was introduced and the scores dropped drastically except for the PCA results. Unfortunately, we could not find the error till the deadline and it was impossible to make any logical conclusion.

3.3. Difficulties

Some difficulties have arisen within the framework of our group project. Especially the selection of a suitable dataset took a lot of time and discussion. This is because we tried to select the appropriate factors to predict the rating. We also had to agree on the appropriate size of the data set. Furthermore, we spent a long time looking for the best way to eliminate the NaN values without distorting the data set. In the end we were able to solve these difficulties regarding the dataset and decided together which way to proceed.

The actual main part of the project, programming in Python, also brought with it some obstacles. Main difficulties were surely the amount of time consumed during the training of the algorithms. Unfortunately, we realized too late that we would not be able to run all algorithm before deadline. However, we have created a folder on Github to share the current progress with all group members and to solve any problems as fast as possible. Even though the project itself brought with it some difficulties, working together in the group has always been very effective.

References

Abu-Aisheh, Zeina, Romain Raveaux, and Jean-Yves Ramel, 2018, Efficient k-nearest neighbors search in graph space. Pattern Recognition Letters.

Angelini, Eliana, Giacomo di Tollo, and Andrea Roli, 2008, A neural network approach for credit risk evaluation,The quarterly review of economics and finance 48, 733-755.

Atiya, Amir F., 2001, Bankruptcy prediction for credit risk using neural networks: A survey and new results, IEEE Transactions on neural networks 12, 929-935.

Batista, Gustavo E. A. P. A., and Diego F. Silva, 2009, How k-nearest neighbor parameters affect its performance, Argentine symposium on artificial intelligence, 1-12.

Freund, Yoav, and Robert E. Schapire, 1996, Experiments with a new boosting algorithm, *icml* 96, 148-156.

Galindo, Jorge, and Pablo Tamayo, 2000, Credit risk assessment using statistical and machine learning: basic methodology and risk modeling applications, Computational Economics 15, 107-143.

Hajek, Petr, and Krzysztof Michalak, 2013, Feature selection in corporate credit rating prediction, *Knowledge-based Systems* 51, 72-84.

Huang, Zan, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu, 2004, Credit rating analysis with support vector machines and neural networks: a market comparative study, Decision support systems 37, 543-558.

Khandani, Amir. E, Adlar J. Kim, and Andrew W. Lo, 2010, Consumer credit-risk models via machine-learning algorithms., *Journal of Banking and Finance* 34, 2767-2787.

Pang, Su-Lin, Yan-Ming Wang, and Yuan-Huai Bai, 2002, Credit scoring model based on neural network, Proceedings. International Conference on Machine Learning and Cybernetics, 4. 1742-1746.

Shin, Kyung-Shik, Taik Soo Lee, and Hyun-jung Kim, 2005, An application of support vector machines in bankruptcy prediction model, Expert Systems with Applications 28, 127-135.

Wu, Chunchi, and Xu-Ming Wang, 2000, A neural network approach for analyzing small business lending decisions, Review of Quantitative Finance and Accounting 15, 259-276.

Wu, Hsu-Che, Ya-Han Hu, and Yen-Hao Huang, 2014, Two-stage credit rating prediction using machine learning techniques, Kybernetes 43, 1098-1113.