

**Machine Learning in Finance**

**Group Project Summary**

Institute for Banking and Finance

Spring Semester 2021

Seminar Introduction to Machine Learning

Aaron Hauser

Lorena Tassone

Lukas Dekker

Nick Vogel

## Missing data

When first looking at the data it became apparent rather quickly that missing data would require a lot of consideration – since there is a considerable number of NAs in the data. However, this is usually the case with ‘real’ data so it didn’t come as a surprise.

Therefore, we spent some time researching how to tackle this issue. Our research came up with some papers, as well as blog entries surrounding the topic from which we took inspiration – some of which are listed in the appendix.

Albeit some of the proposed methods were too involved for our understanding and the scope of this project – as a conclusion of our research regarding this topic we’ve come up with some methods which seem to be suited for our problem.

Some basic methods include mean, median or most frequent value (mode) but these approaches seemed to be rather simplistic for our purposes. Therefore, we settled for more elaborate measures, such as KNN, MissForest and iterative imputer, the latter of which is still experimental according to the documentation page.

Before we got started with implementing these methods, we conducted some EDA to see what we were dealing with. One thing that caught our eye was the fact that quite some columns had a ton of NAs. Since imputing, with whatever method, does not seem reasonable for such factors we had to decide which ones we wanted to drop. Eventually, we settled for a benchmark of 1’500 NAs per column (out of roughly 4000-5000 values per year) and year.

After finding these high-NA variables we wanted to check whether some of them could be useful to assign stocks to one of the predefined labels (buy/hold/sell). We achieved this by creating a classification tree.

Since at that point of time we were not actually going to make a prediction based on the results, but were only concerned with finding factors that may be ‘important’, we used the predefined variable ‘Class’ as our response variable.

From this we got eight variables which were used in a decision tree for any of the years, which had a high NA-count. We then looked at each of them individually to determine whether to drop them or not. For this decision we took into account the difference between the actual NA-count and the predefined threshold. This, of course, was based on highly subjective judgement.

What is more, we have found that there were also a number of rows which had a rather high NA-count, which can be seen in the missing value matrix at the beginning of chapter: “Missing Values/ NAs”. For this we’ve used a threshold of 150 missing values per row, which accounted for approximately 10% of all firms (where a company is counted twice if it appears in two years, three times if it appears thrice, etc.). We then went on and also dropped these rows out of our dataset. We were able to see a clear improvement regarding the total NA-count (this is made visible in a graphic in the subsequent code-chunk).

## Zeroes

Another problem that had to be addressed were the zero-values found throughout the data set. This seemed to be a rather delicate matter, since we were and still cannot be sure whether these zero-values represent actual zeroes, NAs or something else which we weren’t considering.

After some plotting and further exploration of the data it showed that a lot of zeroes were connected to either dividends, R&D or things like Revenue from discontinued Operations, short or long term investments which are metrics that are not entirely unlikely to be zeroes if taken in an economical sense (see appendix for a graph).

Eventually, we decided to leave the zero-values untouched, by the reasoning that we would possibly do more harm than good by changing these values. Another point in favour of this course of action one might add is that, according to our research, changing (or even imputing) values in financial data is avoided as much as possible.

## Outliers

A next step in approaching our main task was the handling of outliers, which we met with a method that is widely used according to scientific literature: the IQR-approach. As an alternative we’ve looked at the Winsorization method to deal with the outliers which is very close to the IQR-approach, given the fact that both use percentiles and set abnormally large or small values on predefined boundaries

Some of the problems that we’ve encountered for this part of the project are firstly, the fact that a lot of methods assume normality which is violated for the given data as we’ve checked with the Henze-Zirkler-test. Secondly a lot of the literature is concerned with univariate data which is obviously not given either so finding methods that can be applied for multivariate data was rather challenging.

Again, one might add that this might bring more misinformation than good to the table so another way would be to simply leave the outliers alone. As one user has mentioned on the Kaggle webpage some values e.g., for revenue have not been converted to USD but rather left in some currency which has a multiplier of 100 such as the Japanese Yen. If one were to be extremely invested one could perform an extensive EDA and correct ‘obvious’ mistakes. Since we’ve got a rather tight schedule, we refrain from such actions.

## Feature Engineering

Since our original data set has 222 financial features, we are dealing with a rather high dimensional set. Therefore, a pre-processing method that identifies the most important features can enhance the predictive accuracy (John, Kohavi, Pfleger (1994)) and improve both the computational and the general performance of our learning algorithm (Shukla et al. (2020)).

For selecting the most important features, we decided to use the random forest algorithm, which ranks the features importance based on an impurity decrease (Pedregosa et al. (2011)).

Looking at the ten most important features in figure 2 in the appendix we see that their importance is already low. Hence to cover about 70% of the importance, we decided to select all the features whose importance is higher or equal the median of all ranked importance values.

However, in figure 3 of the appendix, representing the selected 64 features and their importance, we can see that only the importance of the first two features is significantly higher. Additionally, it is interesting to see, that the first three indicators those are, which were added to the dataset by us.

Before the final pipeline selects these features, we removed some columns which in our opinion do not contribute much information to the classification. First, features that have the same value for all stocks were dropped, since they provide no information. Second, we removed one of two correlated features with a correlation coefficient higher than 0.8 since they convey redundant information to the model. The threshold 0.8 is still quite high but setting it lower would lead to a removal of too many features. Further, there was no significant improvement of performance when setting this threshold lower. The removal of these features led to an increase of the feature’s importance values.

When testing some model with all the features instead of this selection, not only was the computational time higher, but also there was no real improvement of accuracy or performance.

## Class Imbalance

Since the “hold” classification is defined by a small range (within +/- 2.5% of S&P500), we expected the amount of “hold” recommendations to be way lower than “buy” and “sell”. Looking at the distribution of figure 4 in the appendix confirms this assumption. Therefore, we built a function to balance the data set by up sampling the class “hold” by taking 8000 more samples. With “sell” having 10’092 and “buy” 8970 classification results, 8000 seemed to be accurate since it is still a smaller value then the other two but more or less in the same range.

When fitting the models, we additionally checked whether the balanced data set leads to better train and test scores and performance in general than the unbalanced. This was proven true by nearly all classifiers, which is why a balanced data set should finally be used.

## Classifiers

### Decision Tree/Random Forest

decision tree test accuracy with grid search for unbalanced data: 0.6031  
decision tree test accuracy with grid search for balanced data: 0.6729

randomized parameter search:

for the unbalanced data we have 0.6390 (randomized search) vs. 0.6262 (no parameter specification)

for the balanced data we have 0.7548 (randomized search) vs. 0.7532 (no parameter specification)

### SVM

The tuning of the parameters for SVM (computationalyl and manually) resulted in this final selection: kernel: ‘rbf’, C: 0.8, gamma: 0.06. This led to a train score of 0.9437 and a test score of 0.7254.   
The hyperparameter tuning preferred a C value of 1, however, we chose to try smaller C values, since a high C value aims to classify all training samples perfectly and we had a high overfitting problem (Scikit-learn (2011)). Since C penalizes misclassification, lowering it reduced the overfitting but also lowered the test and train score. Finally, a C of 0.8 was chosen since by looking at the amount of missed ‘sell’ classifications this seemed to be the most reasonable compromise.

### KNN

The tuning of the parameters for KNN (computationally and manually) resulted in this final selection: leaf\_size: 100, n\_neighbors: 5, p: 2, weights: ‘uniform’. This led to a train score of 0.7468 and a test score of 0.6321.   
When testing KNN higher leaf\_size values and smaller n\_neighbors (K) values led to better train and test scores. However, the difference between train and test score was significant since in some cases we generated a train score of 1.000. This was an obvious sign of overfitting which is why a higher n\_neighbors value was used. Obviously, the train and test score decreased but the difference between them decreased as well. It was not easy to find a small enough K to avoid oversimplifying but also a large enough to not overfit the samples. Additionally, we tried to reduce the overfitting by using the bagging method, however it got worse. Further, increasing leaf\_size also increased the computational time. Therefore, a leaf\_size of 100 was chosen.

## Results?

## Difficulties and Problems

* One of the main problems that we have encountered throughout the project was the lacking processing power, i.e., at times we have spent quite some time waiting for python to execute code. This was especially a problem for the imputation of missing values and the hyperparameter tuning when fitting the models. It was not possible to tune a lot of values for each parameter at once, since this took several hours. Therefore, some steps were performed manually.
* Although we all had some experience in coding, the adaption to Jupyter Notebooks needed some time, especially for the handling of import statements, which sometimes had to be imported via the terminal first before we were able to access it in the Notebook
* We frequently used the documentation of different tools and functions to get a more in-depth understanding of functions and their respective parameters, this however, often was quite complex.
* One problem that did not have a big impact but was still somewhat annoying is the labelling of plots. It seemed as though sometimes the same settings gave different results with respect to the size of the axis labels.
* When fitting the models, we often came across the problem of overfitting the train data. One way to solve this was bagging the classifier, however, this did not work for all models. Another way was to vary the parameters manually, for example lowering the parameter C for SVM or increase the parameter n\_neighbors (K) for KNN (whose optimal value was previously chosen by grid search). The difficulty was here to find a low but at the same time high enough value to neither oversimplify nor overfit the data.

## Further Actions

If we had more resources (mainly time and processing power) some further steps that we would take are the following:

* We have created different datasets, one for each different imputing method, and an additional one for the Winsorization method (see chapter on outliers), but we’ve only fitted our models on the dataset obtained via KNN imputer, and the IQR-method, as a future step we’d also fit the models on the other datasets (i.e., other imputing and outlier-handling methods)
* Create new features, complementing our inflation and momentum terms
* Rest the significance of added variables such as e.g., momentum or the added interaction terms
* Check the validity of the models with more/ different performance metrics
* More thorough analysis of feature importance and selection by using for example a causality-based features selection. Since classical feature selection algorithms, like the one used by us, only select features based on their correlation to the class variable, Yu et al. (2020) showed that taking causal relationships between the features into account supports building more interpretable and robust prediction models
* Tuning the models with more values for each parameter at once to get the best parameter.

## References

*Scikit Learn - Imputation of Missing Values*. https://scikit-learn.org/stable/modules/impute.html

*Tutorial on 5 Powerful R Packages used for imputing missing values*. https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

Armina, R., Mohd Zain, A., Ali, N. A., & Sallehuddin, R. (2017). A Review On Missing Value Estimation Using Imputation Algorithm. *Journal of Physics: Conference Series*, *892*, 12004. https://doi.org/10.1088/1742-6596/892/1/012004

Gautam, C., & Ravi, V. (2015). Data imputation via evolutionary computation, clustering and a neural network. *Neurocomputing*, *156*(4), 134–142. https://doi.org/10.1016/j.neucom.2014.12.073

John, C., Ekpenyong, E. J., & Nworu, C. C. (2019). Imputation of Missing Values in Economic and Financial Time Series Data Using Five Principal Component Analysis (PCA) Approaches. *Central Bank of Nigeria Journal of Applied Statistics*(Vol. 10 No. 1), 51–73. https://doi.org/10.33429/Cjas.10119.3/6

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*(12), 2825–2830.

Ping, X.-O., Lai, F., & Tseng, Y.-J. (Eds.). (2014). *Evaluation of Imputation Methods for Missing Data and Their Effect on the Reliability of Predictive Models*. IARIA. http://www.thinkmind.org/index.php?view=instance&instance=BIOTECHNO+2014

Raschka, S. *Scikit Learn - Support Vector Machines.* Packt Publishing Ltd. https://scikit-learn.org/stable/modules/svm.html#svm-classification

Schmidt, P., Mandel, J., & Guedj, M. A Comparison of six Methods for Missing Data Imputation. *Journal of Biometrics and Biostatistics*, *6*(224), 1–6.

Shukla, A. K., Pippal, S. K., Gupta, S., Ramachandra Reddy, B., & Tripathi, D. (2020). Knowledge discovery in medical and biological datasets by integration of Relief-F and correlation feature selection techniques. *Journal of Intelligent & Fuzzy Systems*, *38*(5), 6637–6648. https://doi.org/10.3233/JIFS-179743

Yu, K., Guo, X., Liu, L., Li, J., Wang, H., Ling, Z., & Wu, X. (2020). Causality-based Feature Selection. *ACM Computing Surveys*, *53*(5), 1–36. https://doi.org/10.1145/3409382

## Appendix

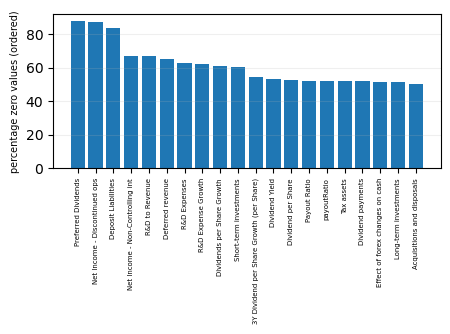
Figure 1: Graph showing the variables with the highest percentage of zero values

Figure 2: Ten most important features according to Random Forest

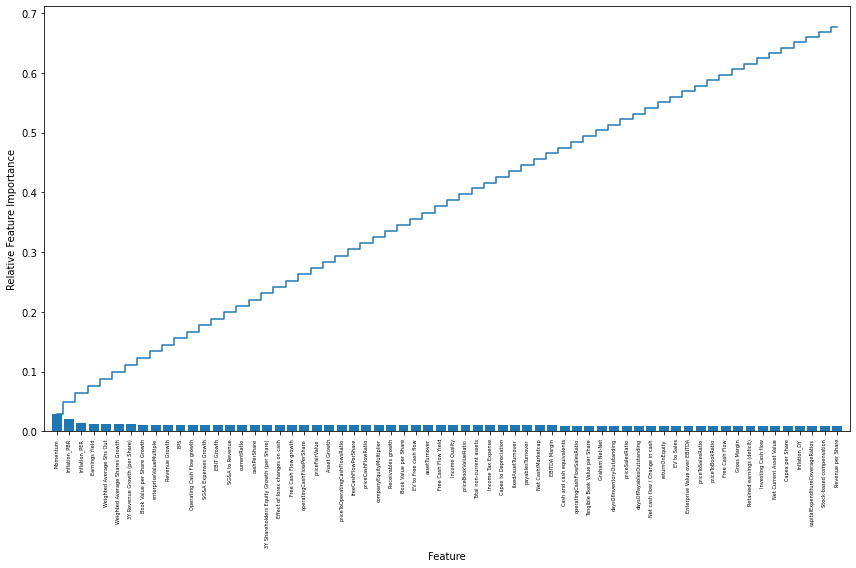
Figure 3: Graph showing all the selected features by the random forest classifier whose importance is higher than the median importance and the cumulative importance.

Figure 4: Number of classifications per class before and after up sampling