Jan Heinrich Schlegel 19-747-096
Robert Bibaj 19-747-385
Simon Klaassen 19-934-629
Thomas Meier 19-738-400

Group 2 Executive Summary

Concatenating the Datasets

After importing the given datasets for the years 2014 up until 2018 into our notebook, we started our analysis by comparing their shapes. We realized that the amount of companies that data was collected for varied over the years, the amount of features that were measured however seemed to be constant, implying that the same features were measured every year, which we double checked and verified later on. Then, we created a loop to make some small adaptions to the datasets - creating an *excess return* feature and modifying the existing *class* feature so that it distinguishes between three instead of two advice categories: buy, hold and sell - and to unify the names of all their features, so that we could concatenate the datasets later on. After concatenating the datasets to one large dataset called ”findata”, we analyzed it thoroughly and created multiple graphs to illustrate its most important characteristics and to highlight some of its peculiarities and potential problems.

Dealing With NaN's

Since our dataset contained a plethora of variables with large amounts of NaN values, we set a threshold of 30% which we determined through trial and error, above which the variable gets dropped from the dataset.Applying this rule, we dropped 76 variables, reducing our dataset to 138 variables. The reason for this decision is that imputing nearly a third of the datapoints of a variable could lead to a very biased dataset. Before imputing the values we had to drop one more column bacause we realised that it only consists of one unique value leaving us with 145 features. We settled for the KNN-imputer, because we wanted to try something more sophisticated then just simple mean or median imputation. However, Shrive et al., 2006 find that simple mean imputation doesn't work well with imbalanced datasets and higher NaN percentages, conditions that also apply to our dataset. We are aware of the resulting bias in our dataset because of the imputation, but felt like the trade-off for having more variables to work with was worth it. To minimize said bias we imputed the train and test set separately i.e. we first fit the KNN-imputer to our train set and then used it to transform our train and our test set independently. We also looked at the iterative imputer, but discarded this idea because the computation took much longer than the KNN-alternative. For the same reason we rejected the approach of applying the KNN-imputer seperately to train and validation set in a pipeline which would be even more precise.

Dealing With Outliers

We decided to detect outliers using the Isolation Forest algorithm. Because of its low memory requirements, it is much faster than traditional algorithms and therefore works well with high volume data (Liu et al., 2008). Furthermore, the Isolation Forest method is able to detect data anomalies taking into account multiple variables as well. This leads to a much more compact train set, which is key for training our outlier-sensitive ML-Algorithms. However, the Isolation Forest Method requires a dataframe containing no NaNs. Consequently, we first had to impute the missing values using the KNN imputer. This could potentially lead to a bias, since the KNN imputation might be influenced by outliers that have not yet been removed. However, in our opinion the benefit of a more powerful imputer outweighted the disadvantage of a not completely tidy cleaning process. Again, we first fit the IsolationForest to the train set and then used it to transform the train and the test set separately.

Handling Class Imbalances

Since we had substantial class imbalance in our dataset (see Figure 1), we needed to use a balancing algorithm. We agreed on upsampling our train data in a pipeline so that the cross-validation scores resemble the test scores as closely as possible. We decided to upsample only the train set and not to the validation or test set since new data will likely be unbalanced too. We first tried the SMOTE algorithm for the upsampling but ran into some problems. First of all, we weren't able to implement SMOTE into our sklearn pipelines and had to use imblearn pipelines instead. Secondly, and more troubling is the fact, that SMOTE upsampled our dummies with continuous samples, which messed up the whole column. Next, we tried the SMOTENC algorithm which was specifically developed to handle upsampling of both numerical and categorical variables. However, we could not implement this approach since we had to one-hot encode the variable "Sector" for our KNN-imputer but SMOTENC did not accept one-hot encoded variables as input. Because of these problems, we decided to use the RandomOverSampler function which resolved the aforementioned issues but contrary to SMOTE and SMOTENC does not add variability by creating new samples but only resamples from existing observations. We chose RandomOverSampler from the imblearn package over resample from the sklearn package because its pipeline implementation is straightforward.

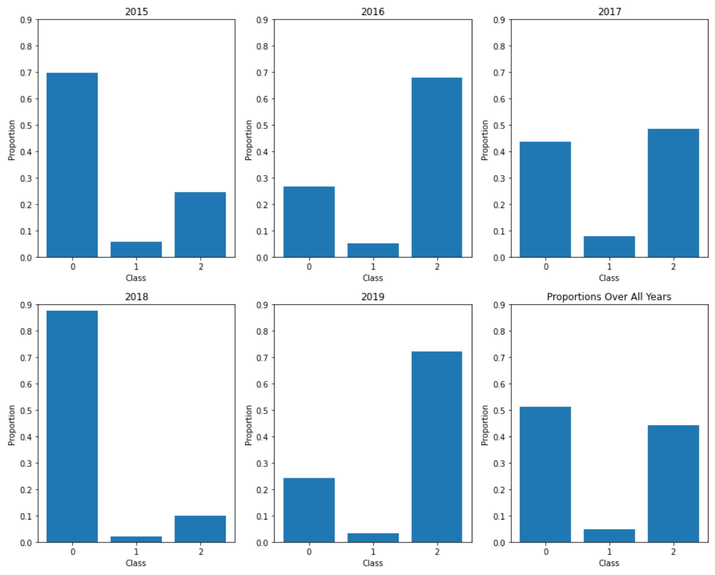

Figure 1: Class Proportions Over the Years

Feature Engineering

The Sustainable Growth Model proposed by Higgins, 1977, describes optimal growth from a financial perspective assuming a given strategy and taking into account liability limitations. One can derive four fundamental drivers of growth from this model, namely profitability,

Jan Heinrich Schlegel 19-747-096
Robert Bibaj 19-747-385
Simon Klaassen 19-934-629
Thomas Meier 19-738-400

liquidity, solvency and growth. Since the model fixes growth as a must, which translates very well to stock market performance, we can identify key figures for the other three dimensions. This then allows us to use these indicators as predictors for the stock performance of the company. Specifically, we created the "Cash Flow Margin" ratio as a driver of Liquidity and the "Return on Net Asset" ratio as a driver of profitability. The "Debt to Asset Ratio" seen as a main driver of solvency, was already included in the dataset.

Adding to that, we wanted to include some macroeconomic data for each of the given years to see if that improved the prediction reliability of our algorithm. The first feature we added was the expected inflation for the next year. Because this feature is constant for all the data points during one year (as is most macroeconomic data), it indirectly signalized to which year a data point corresponds, which dramatically improved the performance of our algorithm. Because the algorithm shouldn't have that information, we decided to delete the expected inflation again and not use any further macroeconomic data.

Performance Metric

The F_1-score is a measure of the accuracy of a model's predictions. It is calculated by the harmonic mean between precision and recall. The goal is to achieve the highest possible F1-score (possible values: between 0 and 1). We chose the weighted F1-score because, it favours the majority classes compared to the minority class (in our example the class "hold"). We decided to go with the weighted F_1-score because in our opinion false buy or sell predictions imply greater costs, than a false hold classification. However, we are aware that a precise decision could only be made if we were be able to assign costs to the respective errors, but this is beyond the scope of this project.

Models

We agreed upon using six different models for this project. We included the logistic regression and the gaussian naive bayes classifier because we wanted to see how well relatively simple models perform on this data. Next, we chose the random forest since it delivers good results whilst being relatively easy to understand. We added XGBoost to represent the family of gradient boosting models that regularly place amongst the best classifiers for tabular data in kaggle competitions. Last but not least, we incorporated a support vector machine classifier and several feedforward neural networks with (multiple) hidden layers and dropout. We evaluated each and every model taking different approaches i.e. use all features, select features, reduce the dimension of the feature space and include our own engineered features.

Feature Selection and Dimension Reduction

Due to the large number of inputs in our dataset and thus the high probability of overfitting, we performed feature selection and dimension reduction. Fortunately, sklearn allows for an easy pipeline implementation of both feature selection and dimension reduction. For our models, we observed that the XGBoost approach to feature selection yielded better scores than the random forest approach. As for dimension reduction, we realised in our exploratory data analysis that most of the features are only barely correlated but rather dependent in a non-linear way since many of them are factors of each other. Since PCA requires features

Jan Heinrich Schlegel 19-747-096
Robert Bibaj 19-747-385
Simon Klaassen 19-934-629
Thomas Meier 19-738-400

to be correlated to be effective, we instead used the Kernel PCA approach proposed by Schölkopf et al., 1998, for dimension reduction. We observed that using a sigmoid kernel for dimension reduction led to better results than the use of a linear kernel (which is equivalent to the PCA we have discussed in class). For the logistic regression we additionally tried out different regularization penalties.

Results and Interpretation

In order to see how well our models performed we needed a naive classification as a reference. This baseline consists of randomly choosing a class based on probabilities that coincide with the proportions of each class in the train set. This baseline model achieved a weighted F_1-Score of 0.466. The weighted F_1-Scores of our models are listed in the table below and the approach that yielded the best score for each model/ column is highlighted using a grey cell colour. For the feature engineering, we used the approach that had the highest weighted F_1-Score for the corresponding model and fitted it again using the dataset including our own features.

Approach	Log Reg	Naive Bayes	Random Forest	XGBoost	SVM	Neural Net[1]
All Features	0.508[2]	0.358	0.544	0.578	0.470	0.540
Feature Sel RF[3]	0.440	0.390	0.492	0.470	0.416	0.535
Feature Sel XGB [4]	0.466	0.505	0.494	0.446	0.273	0.536
(Kernel) PCA	0.456	0.508	0.507	0.526	0.459	-
Feature Engineering	0.513	0.463	0.535	0.581	0.493	0.531

Table 1: Weighted F_1-Scores of the models

Interestingly, only the gaussian naive bayes classifier performed best when upsampling and standardizing. The neural networks and the logistic regression classifier performed best when standardizing the data but not upsampling it. The support vector machine yielded the best results when normalizing the data but not upsampling it. In contrast, the random forest classifier performed best when not scaling but upsampling the data and the Gradient Boosting/ XGBoost Classifier when neither scaling nor upsampling it. Further, we can recognize in the table that Kernel PCA yielded better results for most models than feature selection. In particular we observed that using a sigmoid kernel for PCA yielded better results than normal, linear PCA. We can also observe that our feature engineering yielded mixed results but most importantly we achieved the best score i.e. 0.581 using XGBoost and our engineered features. Notably, one can see in table above, that a finely tuned, but simple logistic regression was able to perform nearly as good as the much more complex ML algorithms. Following this parsimony logic, it is also surprising, that the gradient boosting outperformed the highly complex neural network. This is inline with Shwartz-Ziv and Armon, 2022, who find that tree based algorithms outperform neural networks on tabular data.

[1]Feedforward neural network with 2 hidden layers and dropout
[2]All features using L2 regularization as penalty i.e. some parameters are shrunken towards zero
[3]Feature Selection with random forest
[4]Feature Selection with XGBoost

Relevance Concerning Economic/Finance Theory

According to Lo and MacKinlay, 2011, it is impossible to predict stock prices. Therefore, we have to critically question our results, even though it looks like we are able to beat a random guess (described by our baseline model in respect to the proportions of our dataset).

The popular efficient market hypothesis by Fama, 1960 argues, that it is impossible to beat the market (in our example the S&P500) in the long run. Since our model is built mostly on fundamental analysis, it appears at first glance as our model violates the semi-strong form of the efficient market hypothesis. However, it is to be noted that this is not necessarily the case since it is not outside the realm of possibility that our model classified a larger proportion of stocks correctly than the baseline but does not have a better performance in terms of actual performance on the stock market.

Apart from traditional literature, we also have to critically question the validity of our results due to strong biases in the stock selection. From the descriptive analysis, we are able to see, that the our sample stocks on median had a performance of 3.9%, with a mean performance of 270%. This led us to believe, that there have to be a lot of small cap stocks in our sample. On the other side, the S&P500 only includes the large caps, thereby this could lead to a certain bias in our model, because small caps are known for their higher return volatility. This high volatility also explains our small portion of "Hold" cases in the dataset. Interestingly, the S&P500 outperformed the sample stocks on median, but underperformed on arithmetic average. This is partially due to some extreme outliers in the dataset.

Omitted variables result in biased model estimates (Clarke, 2005). Since our dataset only includes a vast but finite set of independent variables (IVs), we can never be sure that we don't suffer of the Omitted Variable Bias. We tried to reduce the likelihood of this bias by including additional IVs but had problems implementing macroeconomic variables.

Further Analysis

We now know how well our models performed in terms of the weighted F_1-Score but not in terms of actual performance on the stock market. Thus, one could further investigate the models by creating portfolios and buy/ hold/ sell according to the model. Then, for each model different metrics for the portfolio performance could be calculated. One could also recreate our project with a longer time horizon and/ or more features. Another possibility is to use non-US stock data to check how our models would perform for different countries' stocks. Additionally, one could take a different approach to splitting the data into train and test set to reduce the leakage problem when engineering new features such as (expected) inflation. Obviously, further finetuning of the hyperparameters is also possible (in particular those of the neural networks). Especially since we sometimes had to restrict ourselves to the usage of the selective RandomizedSearchCV instead of the more thourough GridSearchCV. Since further hyperparametertuning might be computationally demanding, the experimental HalvingGridSearchCV and HalvingRandomizedSearchCV from the package scikit-learn (Pedregosa et al., 2011) might be of interest. Lastly, we recognise that it is possible to be more meticulous in the analysis of the dataset and e.g.check whether some features are linear combinations or determine the NaN-threshold more formally than we did.

References

Clarke, K. A. (2005). The phantom menace: Omitted variable bias in econometric research. *Conflict Management and Peace Science*, *22*(4), 341–352. https://doi.org/10.1080/07388940500339183

Fama, E. F. (1960). Efficient market hypothesis. *Diss. PhD Thesis, Ph. D. dissertation.*

Higgins, R. C. (1977). How much growth can a firm afford? *Financial Management*, *6*(3), 7–16. http://www.jstor.org/stable/3665251

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

Lo, A. W., & MacKinlay, A. C. (2011). A non-random walk down wall street. *A non-random walk down wall street*. Princeton University Press.

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, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, *10*(5), 1299–1319. https://doi.org/10.1162/089976698300017467

Shrive, F. M., Stuart, H., Quan, H., & Ghali, W. A. (2006). Dealing with missing data in a multi-question depression scale: A comparison of imputation methods. *BMC medical research methodology*, *6*(1), 1–10.

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, *81*, 84–90. https://doi.org/https://doi.org/10.1016/j.inffus.2021.11.011