Stelmaszek Karolina 11942491

Ferrand Adrien 11942895

Logi Marco 11942485

**Python in Finance 2**

**Final project – report**

**Introduction**

The goal of the project is to predict the change in yield of price of 10-Year Treasury Notes. We assumed that the price of the treasury notes is influenced by messages issued by the Federal Reserve Board, especially by their sentiment scores. Such positive correlation was proved in a related work, but for GDP forecasts.[[1]](#footnote-1) We also supposed that the change in yields will be affected by phrases that appear in narrative of the FRB and other features gained by text mining. In addition to that, we considered other variables connected with the market - the previous change of the treasury notes' yield and returns on the stock market. The data were downloaded from following sites: <https://fred.stlouisfed.org/series/DGS10> (yields of 10-year treseary bonds), <https://fred.stlouisfed.org/series/NASDAQ100> (NASDAQ Index) and <https://www.federalreserve.gov/newsevents.htm> (News of Federal Reserve Board). The data concerned the period between mid-January 2006 till mid-June 2020.

**Data preparation**

In this part we dealt with the retrieve of the data. To build our model and to check if the estimations it makes are reliable, we needed different type of data: the FED’s speeches and press releases and the various prices of the 10-Year Treasury Notes together with the value of the NASDAQ index.

After we installed the web drivers, we created two functions: *wait* and *download\_sites*. The first one is used to find the article element on the page. This function is important because there are a couple of articles with a whole different appearance of the page, and thus we decided to not include these articles inside of our set of data. The second function is used to get all the links of the articles and the relative information. We have used this function for both the speeches and the press releases. The amount of press releases articles is higher than the speeches (3624 vs 809). At the end of the download we have inserted all the data inside two different dataframes and saved them to CSV (“all\_pressreleases” and “all\_speaches”).

Then we worked on the file containing the various yields of the Treasury Note. We created a new column with the lagged yield, we changed the values to numeric and then we computed the change in the yield from one period to another. We then computed the daily percentage change in the NASDAQ index and inserted it in a dataframe together with the yield’s change of the 10-Year Treasury Note. Lastly, we save the dataframe to CSV(“indexes2”)

**Feature engeneering**

In this part we computed some text analysis.

We wrote a class *Preprocessing*, which allowed us to do many things on both files, Press and Speech. Using *change\_time* method, we changed the date format so that manipulation will be easier with the time data. We tokenized the text and then did the sentiment analysis (for titles and for article’s texts) with the Vader module from nltk (*count\_sentiment* method). This created new columns in the data base which represented the sentiment analysis scores.

First, we worked on the Press releses file. As the next step, we used *most\_common* method from *Preprocessing*, and saw that most articles have no authors (3393). The articles also had different publication statuses. Some had the annotation "For immediate release". This information can be considered important and should be published as soon as possible. This annotation appears 2025 times (75% of all articles). After this we built another class, *TextFeatures,* to work on the dataframe. Methods in this class compute different kind of features like number of sentences, words and percentages of nouns, verbs, adverbs and adjectives in the articles texts.

.

To the dataframe we added columns that check if in the article’s title appear words: approval, termination, credit and results (one of the most common unigrams discovered by *most\_common\_unigrams\_in\_titles* method from *Preprocessing*), and also columns showing the most common unigram, bigram and trigram in each article, some of them became features (such variables check if the most frequent trigram/bigram is i.g. federal reserve board/written agreement, etc.) We also computed the number of articles published per day and if there are more articles written in one day, we take the mean of the features’ values.

The same steps were conducted for the speech table (with small changes in the author column and without release column), the variables that check if some unigrams appear in titles and if a bigram/trigram is the most common one in the article, were appropriately changed.

**Modelling**

We deal with a regression problem here, so we used decision trees, xgboost and neural net (MLPRegressor). After reading processed data for speeches, press releases and financial indexes, we binded them by the date, and filled empty values (common case for speech and press releases – they are not published every day) with lagged values. After splitting the dataset into the train and test data, we created a class named *AlgorithmFinder* that uses *Pipelines* and *GridSearch Cross Validation* to find the best hyperparameters (from the given combinations of hyperparameters) for each model (*find\_the\_best\_tree, find\_the\_best\_xgboost, find\_the\_best\_net*). Each method printed the Mean Square Error for the training and test dataset an returned the optimal set of hyperparametrs. The next useful class was *AlgorithmPresenter*, that plots the importance of features (it’s possible for decision tree and xgboost) with *plot\_importance* method and plots the true and predicted values of the model (*plot\_tree\_and\_prediction* method).

**Results**

The first algorithm tested was a decision tree (with following optimal hyperparameters: max\_depth=6, min\_samples\_leaf=1 and min\_samples\_split=2) and with Training MSE: -4.283 and Test MSE: -6.419. But, looking at the plot showing true and predicted values of changes in yields, we noticed that at the end of the dataset, we have outliers, possibly due to the corona crisis. We decided to remove approx.. the last for months from the analysis in order to improve the quality of the model and we found a new best decision tree, with the same hyperparameters as before except from max\_depth=2. This time, the Training MSE and Test MSE improved a lot (-3.867 -4.25, respectively), but it turned out that only one feature – change in NASDAQ – is important, so we built one more tree without this feature. Training MSE and Test MSE got worse (-4.54 , -4.4), and the plot with true and predicted values showed that this algorithm is not able to give satisfying results.

Next, we used the xgboost algorithm and the best hyperparameters were learning\_rate=0,1 max\_depth = 3, min\_samples\_leaf = 5 and min\_samples\_split=2. The training MSE improved a lot (respectively -2,773 and -4,131), which was the best result among all analysed models. The most important features were change in NASDAQ index (43%), and the lag of change in yields of treasury bond (14%). The rest of variables had rather insignificant influence on the change in yield, between 0 and 2 %. The plot with true and predicted values shows a similar course, but still the predicted values revealed a lower range than the true values of changes in yields.

We also looked at xgboost model for the whole dataset, but again, the outliers that can be found at the end of the sample led to much worse Test MSE (-6,034), which was also confirmed by the plot with the true and predicted values.

The last algorithm used in the project were neural nets. The best parameters found for this algorithm (and the dataset without outliers) were activation (function) = identity, alpha (L2 penalty parameter) = 0,01, 3 hidden layers with number of neurons equal to 64, 32 and 32, max\_iter (iterations) = 1000 and tol (Tolerance for the optimization) = 0,0001. The training MSE (-3,796) is a bit worse than in the xgboost algorithm, but the result in the test dataset was comparable to xgboost’s results (-4,17). The predicted values are placed a bit higher above the 0 than predictions of the xgboost model.

**Summary**

To sum up, the best results were obtained with the xgboost algorithm and neural nets – in aspect of the test dataset – but due to the fact that we generally prefer more interpretable algorithms, the best algorithm in the described project was xgboost, which allows to see which variables were of the highest importance. The hypothesis that features obtained from text (especially sentiment analysis) have a significant impact on changes in yield of the 10-year treasury bonds was not confirmed.

We also noticed some areas for improvement. First of all, a good idea would be to use a bigger dataset (which could solve problems with overfitting – differences in MSE between train and test dataset), but to achieve that, we should consider some other sources of texts – there were no more articles of Federal Reserve Board available. And even if the number of articles would be larger, the scrapper worked relatively slow and for example, downloading twice more articles could take even the whole day. In addition to that, more financial indicators should be taken into account, as they seem to be more effective then text features and sentiment analysis. Last but not least, we could have used another tool for sentiment analysis than VADER (unfortunately we didn’t found a dictionary for financial data)

1. Sharpe, Steven A., Nitish R. Sinha, and Christopher A. Hollrah (2020). “The Power of Narratives in Economic Forecasts,” Finance and Economics Discussion Series 2020-001. Washington: Board of Governors of the Federal Reserve System, <https://doi.org/10.17016/FEDS.2020.001> [↑](#footnote-ref-1)