732A92 Text Mining

# CLASSIFYING STOCK PRICE CHANGES BASED ON 8-K SEC FILINGS WITH GRADIENT BOOSTING DECISION TREES (CATBOOST)

December 27, 2019

Name: Julius Kittler
Student ID: julki092
Linköping University

**Abstract**

This text mining project makes use of filings from the U.S. Securities and Exchange Commission (SEC), in particular 8-K filings, to forecast short-term percentage changes in stock prices. 8-K filings have to be published by public companies in the U.S. for certain defined, business-relevant events. The forecasting problem is approached as a classification problem with five classes: large decrease, small decrease, no change, small increase and large increase (of the stock price before and after filing date of the 8-K filing). Instead of using existing data sets, a new, up-to-date data set is created, with data for the year 2018 and the first quarter of 2019. Furthermore, this project makes a contribution by testing a new approach to this problem: gradient boosting decision trees (GBDT) with tf-idf bag-of-words. Previous research had found random forests to perform very well on this problem. Hence, the methodological choice for this project fell on GBDT as a more advanced ensemble model. Exclusively text data was used for training the models. The best model could achieve an additional 10 percentage points of accuracy on the test data compared to the majority classifier, which is comparable to previous findings from the literature.

# Contents

## List of Figures

## List of Tables

## 1 Introduction

Forecasting stock prices has been a relevant problem since the existence of publicly traded companies. Today, it is an even more relevant problem than ever before because we have the technological infrastructure to put our forecasts to practice in form of automatic trading systems. Not only can we obtain massive amounts of financial data via APIs, but we can also execute trades via APIs. With commission free trading becoming the industry standard in the U.S., executing trades via APIs is even offered for free by some companies nowadays [1].

The goal of this project is to classify discretized stock price percentage changes based on text data from SEC filings, in particular 8-K filings. SEC filings are official documents that publicly traded companies in the U.S. are required to publish to inform their shareholders about the business. The SEC, the U.S. Securities and Exchange Commission, is the governmental agency that enforces this [3]. While there are various different types of filings, e.g. for annual and quarterly reports, this project focusses exclusively on 8-K filings. Because 8-K filings correspond to major events for the company and because they need to be published shortly after an event occurred (within 4 days), 8-K filings seem interesting for predicting short-term volatility in the stock market.

Companies need to publish 8-K filings for major events such as a change in the board of directors, a potential delisting from a stock exchange or a merger. In total, there are 31 different 8-K filing events from 9 different sections. A complete list of all events and sections, taken from the official SEC website [3], is shown in the table 1. One 8-K filing document may contain information for several such events. For instance, one particular 8-K filing document may contain information about Item 5.02, Item 6.02 and Item 9.01 at the same time. Every 8-K filing clearly states for which events it contains information. To see some examples of 8-K filings, one may go to the official SEC website. In particular, three examples can be found here: example 1, example 2, example 3.

## 2 Theory

### 2.1 Stock Price Forecasting with 8-K filings

This section gives a short overview of previous work about forecasting stock prices with 8-K filings. Since the focus of this project is classification, this literature review also focuses on previous work with classification. There are few public research papers that made use of 8-K filings for stock price forecasts. However, the papers that are publicly available show promising results, with between 4 and 9 additional percentage points of accuracy achieved by adding text features from 8-K filings to the classification models. An increase of 4 percentage points was achieved compared to a strong baseline in [5] and an increase of 9 additional percentage points was achieved compared to a rather weak baseline in form of a majority classifier in [6].

Lee et al. could achieve an increase in accuracy by 5 percentage points when including text data from 8-K filings into a baseline model that only used financial metrics [5]. This study was solving a classification task with three classes: price increase ($> 1\%$), price decrease ($< -1\%$) or no relevant change ($< |1\%|$). Compared to a majority classifier with 35% accuracy, and a baseline model using financial metrics with 50% accuracy, the best model of the study could achieve a 55% accuracy on the test data. This model was a random forest classifier, which outperformed other models such as multi layer perceptron and logistic regression. Unigram features of the text data were used along with non-negative matrix factorization for dimensionality reduction.

Another study by Saleh et al. extended the research by Lee et al. [6]. In addition to the data from the 8-K filings, the researchers used text data from Twitter. Furthermore, they used convolutional neural networks and recurrent neural networks instead of random forests. Compared to a majority classifier with 42% accuracy without twitter data and 49% accuracy with twitter data, best test accuracies were 51% and 53% respectively. Instead of unigram features, word embeddings from GloVe were used.

Holowczak et al. compared the performance of various common algorithms (random forest, naive bayes, support vector machine, k-nearest neighbor and ridge classifier) on a binay classification task (price increase vs. decrease) [7]. However, only 8-K filings for a particular event, item 4.01, were used. The best results were achieved with a linear support vector classifier with a classification accuracy of 54.4% on the test data (compared to a 50% accuracy by a majority classifier). The text data had been transformed with frequency-inverse document frequency (tf-idf) vectorization.

### 2.2 Gradient Boosting Decision Trees (GBDT)

Gradient boosting with decision trees became popular with the development of XGBoost in 2016 [8] and lightGBM in 2017 [9]. In 2018, another GBDT library called CatBoost was published, which was shown to outperform

Table 1: Overview of all 8-K sections and events

| Section | Item | Event |
|---|---|---|
| Registrant's Business and Operations | 1.01 | Entry into a Material Definitive Agreement |
| | 1.02 | Termination of a Material Definitive Agreement |
| | 1.03 | Bankruptcy or Receivership |
| | 1.04 | Mine Safety - Reporting of Shutdowns and Patterns of Violations |
| Financial Information | 2.01 | Completion of Acquisition or Disposition of Assets |
| | 2.02 | Results of Operations and Financial Condition |
| | 2.03 | Creation of a Direct Financial Obligation or an Obligation under an Off-Balance Sheet Arrangement of a Registrant |
| | 2.04 | Triggering Events That Accelerate or Increase a Direct Financial Obligation or an Obligation under an Off-Balance Sheet Arrangement |
| | 2.05 | Costs Associated with Exit or Disposal Activities |
| | 2.06 | Material Impairments |
| Securities and Trading Markets | 3.01 | Notice of Delisting or Failure to Satisfy a Continued Listing Rule or Standard; Transfer of Listing |
| | 3.02 | Unregistered Sales of Equity Securities |
| | 3.03 | Material Modification to Rights of Security Holders |
| Matters Related to Accountants and Financial Statements | 4.01 | Changes in Registrant's Certifying Accountant |
| | 4.02 | Non-Reliance on Previously Issued Financial Statements or a Related Audit Report or Completed Interim Review |
| Corporate Governance and Management | 5.01 | Changes in Control of Registrant |
| | 5.02 | Departure of Directors or Certain Officers; Election of Directors; Appointment of Certain Officers; Compensatory Arrangements of Certain Officers |
| | 5.03 | Amendments to Articles of Incorporation or Bylaws; Change in Fiscal Year |
| | 5.04 | Temporary Suspension of Trading Under Registrant's Employee Benefit Plans |
| | 5.05 | Amendment to Registrant's Code of Ethics, or Waiver of a Provision of the Code of Ethics |
| | 5.06 | Change in Shell Company Status |
| | 5.07 | Submission of Matters to a Vote of Security Holders |
| | 5.08 | Shareholder Director Nominations |
| Asset-Backed Securities | 6.01 | ABS Informational and Computational Material |
| | 6.02 | Change of Servicer or Trustee |
| | 6.03 | Change in Credit Enhancement or Other External Support |
| | 6.04 | Failure to Make a Required Distribution |
| | 6.05 | Securities Act Updating Disclosure |
| Regulation FD | 7.01 | Regulation FD Disclosure |
| Other Events | 8.01 | Other Events |
| Financial Statements and Exhibits | 9.01 | Financial Statements and Exhibits |

XGBoost and lightGBM on various datasets with default parameters [12]. For this reason, CatBoost is used in this project. The implementational details of CatBoost are rather involved and are therefore not fully covered in this report. Instead, what follows is a short introduction to GBDT. Differences betweeen CatBoost and other implementations of GBDT are mainly that CatBoost handles categorical features very well and that it uses unbiased gradient estimates.

In GBDT, several concepts commonly used in machine learning are combined: gradients, boosting and decision trees. The main idea of **boosting** is to iteratively fit simplistic, additive models (so-called weak models) such that each model reduces the errors made by the ensemble model in the corresponding previous iteration. This can be achieved by fitting each model directly on the residuals of the ensemble model in the previous iteration. Gradient boosting generalizes this approach: each weak model is fitted on the **gradient** of the residuals from the previous iteration [8] [10] [11], where the term *residuals* refers to a differentiable loss function.

Recall that the gradient takes a scalar function (returning 1 scalar value given $P$ parameters) and returns a vector (with $P$ values, each of them the partial derivative of the scalar function w.r.t. the $p$th parameter). In our case, the scalar function is the differentiable loss function $L^{(t-1)}$:

$$\mathrm{L^{(t-1)}} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}), \tag{1}$$

where $i = 1, ..., N$ is the index for the training observations, $\hat{y_i}^{(t-1)}$ is the prediction for the $i$th training observation based on the ensemble model from the previous iteration and $l(y_i, \hat{y}_i^{(t-1)})$ is the loss for the $i$th training observation. If we take the gradient with respect to $\hat{y}_i^{(t-1)}$ as parameters, we get the following:

$$\nabla L^{(t-1)} = \left[ \frac{\partial l(y_1, \hat{y}_1^{(t-1)})}{\partial \hat{y}_1^{(t-1)}}, \frac{\partial l(y_2, \hat{y}_2^{(t-1)})}{\partial \hat{y}_2^{(t-1)}}, ..., \frac{\partial l(y_N, \hat{y}_N^{(t-1)})}{\partial \hat{y}_N^{(t-1)}} \right] \tag{2}$$

This is a row vector with N elements, where each element corresponds to the derivative of the loss of the $i$th training observation with respect to its prediction $\hat{y_i}^{(t-1)}$. For instance, if we used the squared error loss function, $l(y_i, \hat{y}_i^{(t-1)}) = \frac{1}{2}\left(y_i - \hat{y}_i^{(t-1)}\right)^2$, the elements in the gradient vector would be $\frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} = -\left(y_i - \hat{y}_i^{(t-1)}\right) = \hat{y}_i^{(t-1)} - y_i$. Since the gradients here are absolute residuals, this corresponds to the simplest case mentioned above: fitting a weak model directly on the residuals of the ensemble model in the previous iteration. Importantly, we are not restricted to the squared error loss function. Instead, we can use any other differentiable loss function.

In every iteration $t$, we would like to add a weak model that helps to decrease the loss from the previous iteration $L^{(t-1)}$. Since the gradient points into the direction of the maximum increase of the loss function $L^{(t-1)}$ but we would like to decrease the loss, we consider the negative gradient $-L^{(t-1)}$. We start by initializing $\hat{y}_i^{(0)}$ as a constant for all training observations $i$, for instance as the mean of $y$ from the training data. In every subsequent iteration $t$, we fit a weak model that predicts the negative gradient. We also learn a parameter that determines the step size towards the negative gradient. When we then add this model to the ensemble classifier, we essentially perform a step of gradient descent (not as usual by changing some parameters $w$, but instead by adding another weak model to an ensemble classifier). The prediction of the ensemble model for any observation $i$ can be represented as:

$$\hat{y}_i = \sum_{t=0}^{T} \hat{y}_i^{(t)}, \tag{3}$$

where $T$ is the number of iterations, i.e. weak models, and $\hat{y}_i^{(t)}$ is the prediction of the weak model from iteration $t$ for observation $i$. Note that $\hat{y}_i$ may not be the final output of the model. For classification, we might pass $\hat{y}_i$ as input to a softmax function to get the predicted class for observation $i$.

Finally, note that different types of weak models can be used with gradient boosting. However, in GBDT specifically, every weak model is a **decision tree**. Decision trees are tree-shaped models that consist of nodes connected with branches. To make a prediction, for an observation $i$, its feature vector $\mathbf{x}_i$ is passed from the root node through the tree until it lands in a leaf node. For all vectors $\mathbf{x}_i$ in the same leaf node, the prediction is made. At each node within the tree, a decision is made about the branch along which the $\mathbf{x}_i$ is to be passed further. The decision is made based on a threshold $\tau_n$ for the value $\mathbf{x}_i^{(j)}$, where $j$ is the index of the variable used at the threshold and $n$ is the index of the node. Among others, the parameters that need to be learned are the thresholds $\tau_n$, the variable indices $j$ used at the thresholds and depending on the implementation also the structure of the tree (e.g. number of nodes, depth of the tree).

# 3 Data

## 3.1 Sources

The data used for this text mining project comes from a variety of sources. The stock price data (with daily resolution) was retrieved with the financial API Tiingo. The SEC filings were downloaded from EDGAR, the official archive for SEC filings. The overview of companies by CIK, necessary to merge the SEC filings with the stock prices, was taken from the service Ranked and Filed. Lastly, the overview of the 8-K events, used to extract the 8-K events from each filing, was taken from the SEC documentation.

## 3.2 Retrieval

The dataset used for training the models was created with the following steps:

1. **Getting 8-K filing overview by CIK:** A list of all 8-K filings for all quarters of 2018 and for the first quarter of 2019 was retrieved from EDGAR. This list contained a total of 76782 8-K filings, including the CIK number (to identify the company) and the filing date.

2. **Getting 8-K filings overview by ticker symbol:** The list was merged with the data from Ranked and Filed to get the exchange market and ticker symbol of the companies corresponding to each 8-K filing.

3. **Filtering out exchanges:** All 8-K filings from companies that were not listed on the NASDAQ, NYSE or AMEX (according to Ranked and Filed) were removed. In particular, the removed 8-K filings corresponded to companies listed on ARCA, OTC or OTCBB and to companies for which no information about the exchange market was available. The resulting list contained a total of 49070 8-K filings.

4. **Filtering out missing stock prices:** All 8-K filings, for which no stock price data was available from Tiingo were removed from the list. The resulting shortlist contained a total of 45262 8-K filings. After further removing 8-K filings that had a stock price split before, on or after the day of the filing date, the shortlist contained 42087 8-K filings.

5. **Adding stock prices by ticker symbol:** For the remaining 8-K filings, the stock price data was extracted from the Tiingo data, making use of the filing date from EDGAR. For each filing, the percentage change from the closing price of the day before the filing date and the open price of the day after the filing date was computed as target variable.

6. **Downloading 8-K filings:** All 8-K filings from the shortlist were successfully downloaded from EDGAR as text documents with HTML format.

7. **Filtering out during pre-processing:** When processing the data (see below) 8-K filings with more than 1 Mio. characters were removed due to file size limitations of the libraries that were used for processing. Furthermore, two outliers with a percentage change of approx. 2900% were removed. This left a total of 41763 SEC filings for the project.

## 3.3 Pre-Processing

### 3.3.1 Text in General

Each raw 8-K filing, a text file containing HTML code, was processed as follows. First, graphics and embedded PDFs were removed and HTML tags were removed as well. Second, the resulting text data was tokenized with the natural language processing library spaCy, using the English language model en_core_web_sm. Third, stop words, non-alphabetical tokens and tokens with only one character were removed. Fourth, the remaining tokens were lemmatized with spaCy. Fifth, the remaining tokens were all converted to lower case text.

### 3.3.2 Extraction of 8-K Events

The 8-K event categories corresponding to each 8-K filing were extracted for analysis. This was done by searching all filing documents for all item identifiers. E.g. each document was searched for "Item 1.01", "Item 1.02", ..., "Item 9.01". Since these identifiers must occur in the titles of the 8-K filings that contain information about the respective events, they could be extracted by a simple search.

### 3.3.3 Negation Encoding

For one experiment, words from negated parts of a sentence were encoded with the suffix _neg. This was done by locating the strings not, 't, never, no, neither, nor, nobody, noone, nothing, nowhere, cannot in each sentence and adding said suffix to all words between such a string and the next punctuation (dot, comma, semicolon etc.).

## 3.4 Description

### 3.4.1 Target Variable Before Discretization

For each 8-K filing, the target variable is computed as the percentage change from the closing price of the day before the filing date to the opening price of the day after the filing date. Only Monday-Friday are considered, i.e. when the filing date is a Monday, then the closing price from the Friday before is used (instead of Sunday).

The percentage changes are normalized with the S&P 500, a stock price based on the 500 largest companies in the U.S. The normalization was done by subtracting the S&P 500 percentage change from the stock price percentage change. For example, when the percentage change of the stock price corresponding to an 8-K filing was +5% and when the percentage change of the S&P 500 in the same period was +3%, then the normalized percentage change is +2%. Table 2 shows descriptive statistics for the percentage changes after normalization but before discretization. The corresponding distribution is visualized in figure 1. The displayed percentage changes are the percentage changes from the closing price of the day before the filing date to the opening price of the day after the filing date, where one data point is one 8-K filing. Visually, the distributions for training, test and validation set seem very similar, and hence the distribution was only plotted for the overall data.

Table 2: Descriptive statistics for percentage change by data type

| Data Type | size | min | max | mean | percentile_25 | percentile_50 | percentile_75 | std |
|---|---|---|---|---|---|---|---|---|
| training | 27030.0 | -0.9386 | 4.9105 | 0.0011 | -0.0158 | 0.0001 | 0.0165 | 0.0765 |
| validation | 6758.0 | -0.9206 | 3.0863 | 0.0005 | -0.0159 | 0.0004 | 0.0167 | 0.0726 |
| test | 7975.0 | -0.8135 | 2.8866 | 0.0021 | -0.0163 | -0.0001 | 0.0157 | 0.0759 |
| total | 41763.0 | -0.9386 | 4.9105 | 0.0012 | -0.0159 | 0.0001 | 0.0164 | 0.0758 |

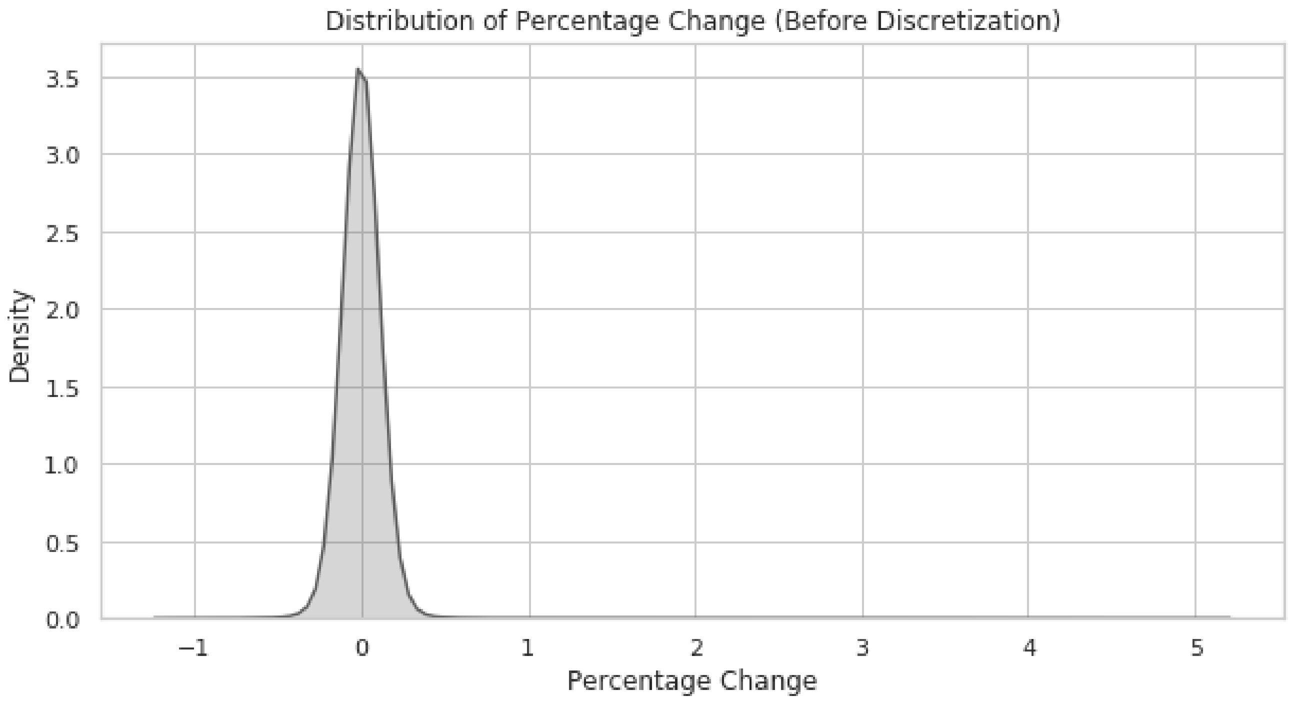

Figure 1: Distribution of the target variable before discretization

### 3.4.2 Target Variable After Discretization

The normalized percentage changes (as described in 3.4.1) were discretized into five bins of equal size, representing the five classes that are to be predicted in this project. Discretization into bins of equal size was used as discretization method to ensure balanced classes without having to over- or undersample. Five bins were chosen to distinguish between large, small and negligible percentage changes. The five classes correspond to large percentage decrease (lg_dec), small percentage decrease (sm_dec), no relevant change (same), small percentage increase (sm_inc) and large percentage increase ( lg_inc ).

Table 3 shows descriptive statistics for the percentage changes after discretization. This table basically illustrates what the classes used in this project actually represent. For instance, the class lg_dec represents any percentage change in the interval $[-0.9386, -0.0217]$. Here, $-0.9386$ represents $-93.86\%$. Figure 2 shows the number of observations by training, validation and test data. Due to the discretization method, the classes are balanced in the overall data as well as across the different data sets, which is visible in table 3 and figure 2.

Table 3: Descriptive statistics for percentage change by class

| Disc. Level | size | min | max | mean | median | std |
|---|---|---|---|---|---|---|
| lg_dec | 8355 | -0.9386 | -0.0217 | -0.0681 | -0.0451 | 0.0687 |
| sm_dec | 8386 | -0.0217 | -0.0051 | -0.0121 | -0.0116 | 0.0047 |
| same | 8365 | -0.0051 | 0.0054 | 0.0002 | 0.0002 | 0.0030 |
| sm_inc | 8315 | 0.0054 | 0.0226 | 0.0126 | 0.0119 | 0.0048 |
| lg_inc | 8342 | 0.0227 | 4.9105 | 0.0736 | 0.0473 | 0.1166 |

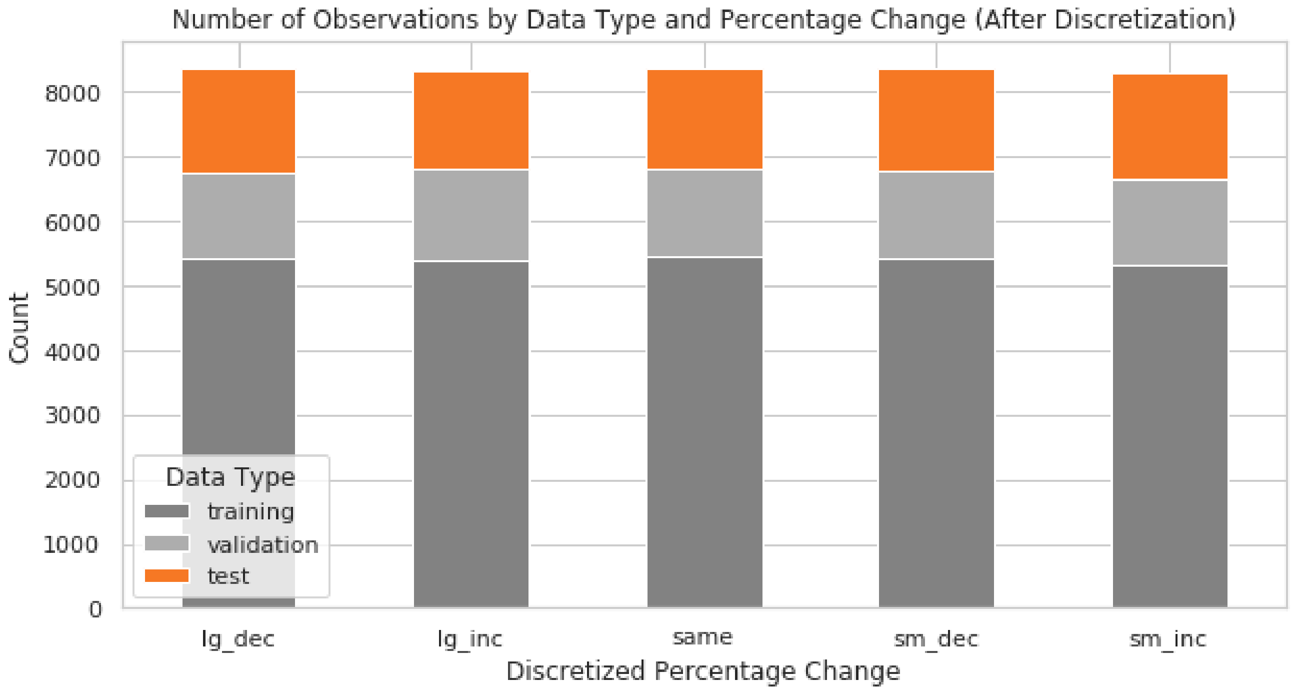

Figure 2: Distribution of the target variable after discretization

### 3.4.3 8-K Filings

The distribution of the number of tokens per document is shown in figure 3. There are few documents with more than 10000 tokens. The largest 8-K filing has approximately 75000 tokens. Note that any filings with more than 1 Mio. characters have been filtered out before during pre-processing.

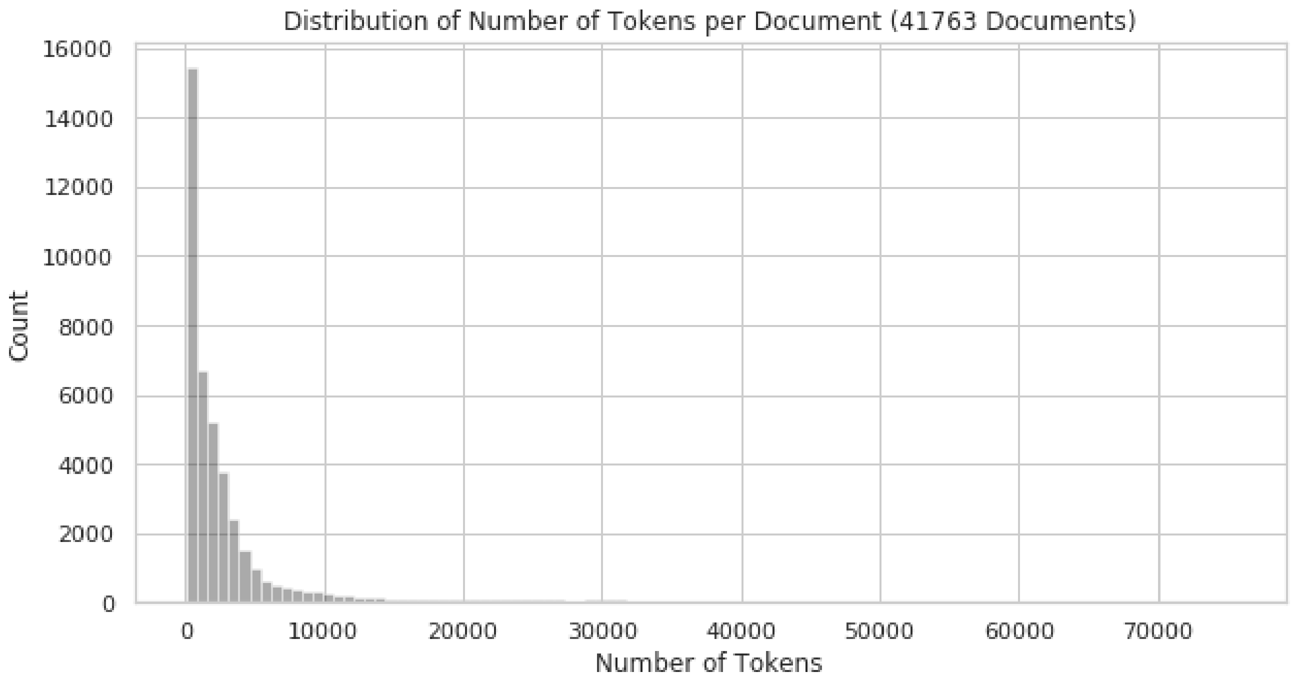

Figure 3: Distribution of the number of tokens per document

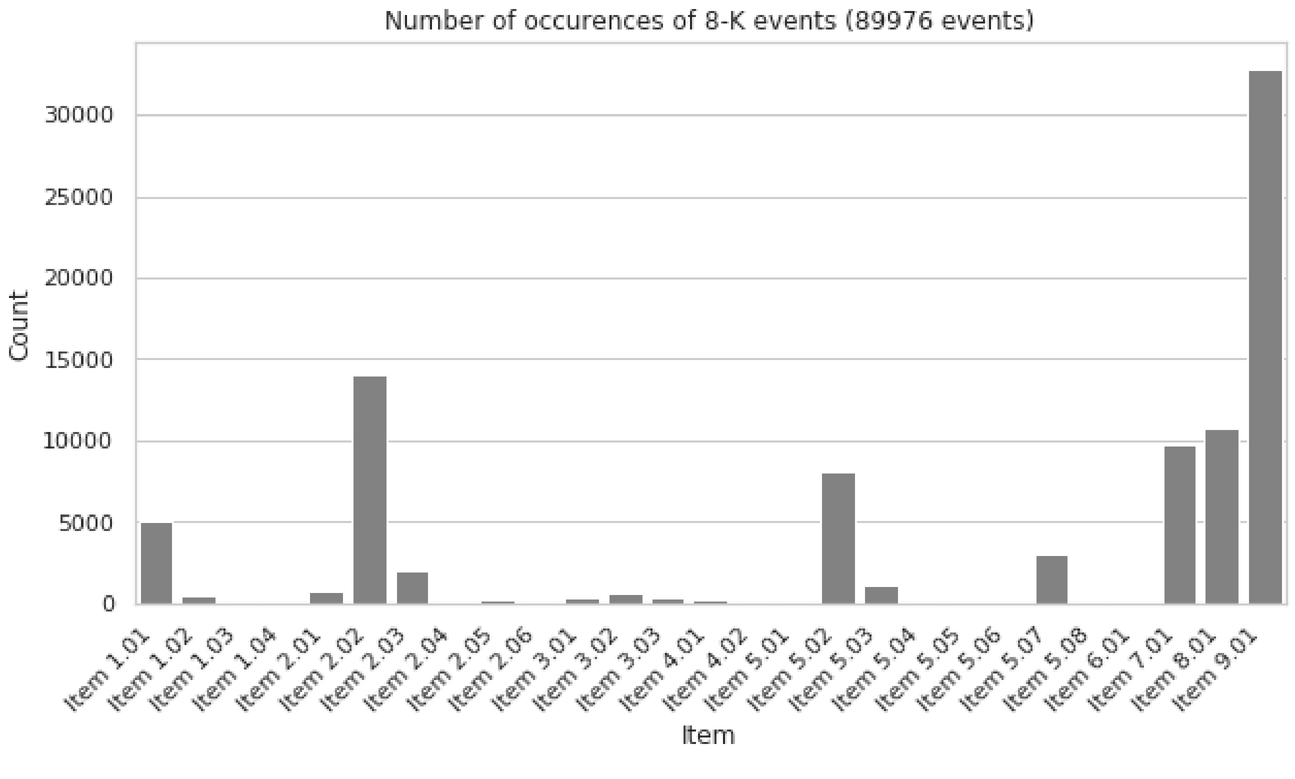

Figure 4: Number of occurences of 8-K events

The number of occurences of each 8-K event is displayed in figure 4. Note that one 8-K filing can contain information about more than one event. Therefore, the total number of events, 89976, is more than twice as large as the total number of 8-K filings. The five most frequent events are: 9.01 (financial statements and exhibits),

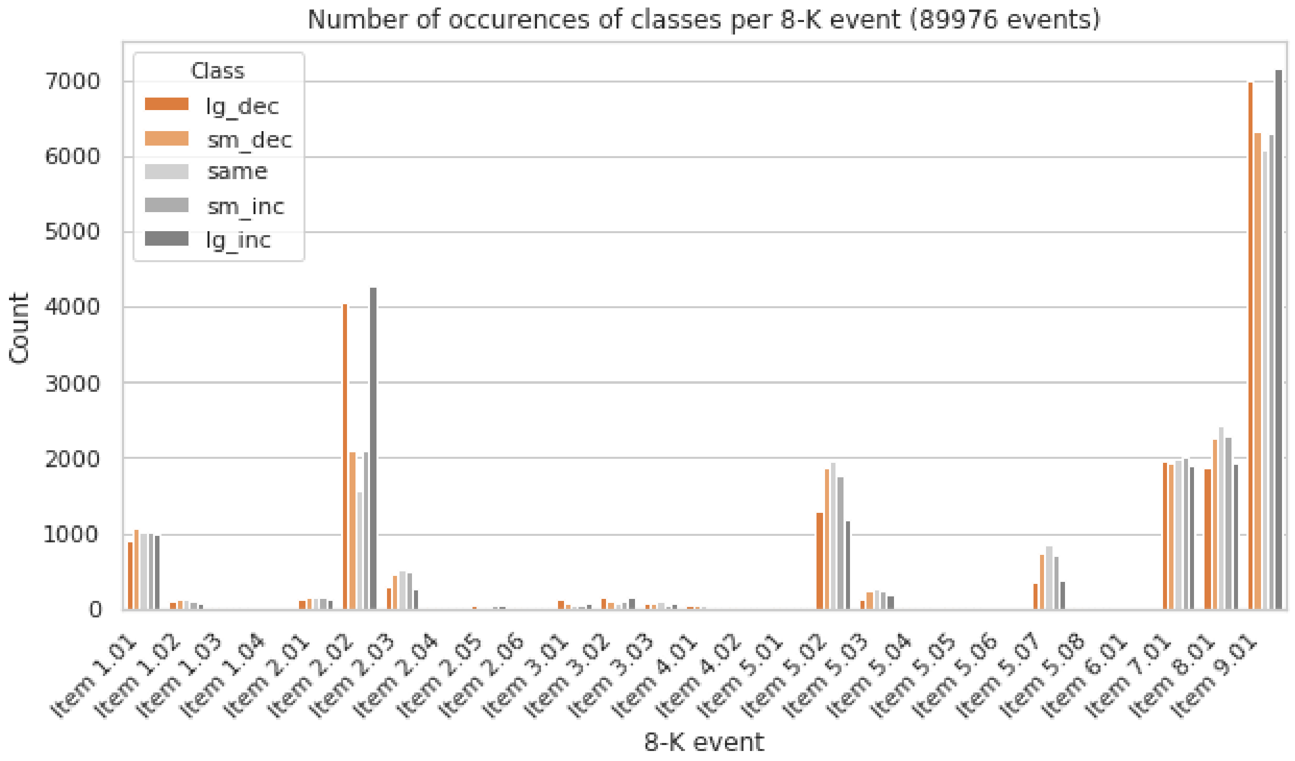

Figure 5: Number of occurences of classes by 8-K event

2.02 (results of operations and financial condition), 8.01 (other events), 7.01 (regulation FD disclosure) and 5.02 (Departure of directors or certain officers; election of directors; appointment of certain officers; compensatory arrangements of certain officers).

In figure 5, the number of occurences of each 8-K event is again displayed, this time however with additional information about the corresponding discretized percentage change. For instance, one can see that there were approx. 7000 8-K filings with information about ”Item 7.01” that were followed by a large *decrease* in the stock price (dark orange) and another 7000 8-K filings with information about ”Item 7.01” that were followed by a large *increase* in the stock price (dark grey). Importantly, one can see that the same 8-K event, e.g. ”Item 7.01”, generally corresponds to an equally large number of stock price increases and decreases. In other words, whether a particular 8-K event contributes to a stock price increase or decrease does not seem to depend on the event itself but rather on the actual content of the 8-K filing about this event.

## 4 Method

### 4.1 Training, Validation, Testing

The available data covers the complete year 2018 and the first quarter of 2019, i.e. five quarters in total. The four quarters of 2018 were used as training and validation data and the first quarter of 2019 was used as test data. The data for 2018 was split randomly into 80% training and 20% validation data.

The training data was used to train the models (with and without cross validation). In particular, cross validation with three folds based on the training data was used when doing grid search for hyperparameter optimization. The baseline models were trained on the training data without cross validation. After training, all models were evaluated on the validation data. Based on the performance on the validation data, the best model was chosen. This model was then evaluated on the test data to obtain a generalization error and evaluate whether we could use the model in practice.

## 4.2 Evaluation Metrics

For evaluating the results, accuracy, as well as average precision, recall and f1-score are reported. Accuracy was used as main criterion to determine the best model. When interpreting the performance of the best model on the test data, the focus was on the confusion matrix and on class-wise precision, an important metric for considering potential trading strategies based on the model.

Class-wise precision is computed for each class individually. For a given class c, the class-wise precision is defined as the number of observations of class c that were classified as class c (TP for the number true positives), divided by the number of any observations that were classified as class c, regardless of whether they are of class c or not (TP + FP for the number of all true positives and false positives).

$$\text{precision}_c = \frac{TP_c}{FP_c + TP_c} \tag{4}$$

Using precision as prioritized evaluation metric for the final model reflects the assumptions that a) we want to avoid loosing money in a trade and that b) missing out on a good trade is acceptable. To explain this, it is best to consider an example. Assume that we take a long position whenever the model classifies a large stock price increase (`lg_inc`), i.e. we buy stocks expecting that their stock price will increase. If the stock price decreases instead, we would loose money. To avoid this loss, we ideally want that among all cases where the model classifies a large increase (FP + TP), the stock price actually increases (TP). This means that we care about the precision.

## 4.3 Loss Function

For multi-class classification problems, the softmax function is generally used as loss function. Hence, it is also used for this project. However, the chosen loss function differs slightly from the softmax function because in GBDT, each observation $i$ is weighted with $w_i$. The loss function used for CatBoost is called MultiClass loss function and is essentially a weighted softmax function [13]. It is computed as a scalar value across all observations and classes. The definition is as follows:

$$L = \frac{\sum_{i=1}^{N} w_i \log\left(\frac{\exp a_{i,t_i}}{\sum_{j=0}^{M-1} \exp a_{i,j}}\right)}{\sum_{i=1}^{N} w_i}, \tag{5}$$

where $i = 1, ..., N$ is the index for the observation, $j = 0, ..., M-1$ is the index of the class, $a_{i,j}$ is the model output for observation $i$ and class $j$, $w_i$ is the weight assigned for observation $i$ during training, and $t_i$ is the index of the true class of observation $i$ so that $a_{i,t_i}$ is the model output for the true class of observation $i$.

Essentially, the term in the parentheses is the softmax function, which returns a value between 0 and 1 for each observation $i$ to belong to its true class $t_i$. If the model performs well for observation $i$, then the softmax value will be large. For instance, if the value is 1, this means that according to the model, we have a probability of 1 that observation $i$ is from class $t_i$, i.e. from the true class of the observation. In other words, the model would perfectly identify the true class of observation $i$. Since the log is taken, and since $log(1) = 0$, the loss incurred for this perfectly classified observation will be 0. If the softmax value is smaller than 1, the loss for an observation will be negative and will hence affect the training process.

## 4.4 Experiment Description

### 4.4.1 Baseline

Four commonly used models were included as baseline models. The data was processed in the same way for all these models: the CountVectorizer from sklearn was used. min_df was set to 0.001 and max_df was set to 0.9, i.e. only tokens that were in fewer than 90% of the documents and in more than 0.1% of the documents were included. These parameter values were identified during a grid search experiment with CatBoost and then, for simplicity, used for the baseline models as well.

### 4.4.2 GBDT

Various experiments and parameter searches were performed with CatBoost. Due to long run-times, an exhaustive grid search was not possible. Instead, a mixture of simple comparisons, grid search, and random search was used. Since experiments depended on each other, some results are reported here as well. Only the final model was evaluated on the validation data. For training and parameter tuning only the training data was used, with a simple percentage split (for step 1-2) and with 3-fold cross validation (for step 3-4).

1. A comparison between the **CountVectorizer and the TfidfVectorizer** was conducted (with 67% of the training data for training and 33% of the training data for validation). min_df was set to a small number, 0.001, so that the vectorized data could fit in memory. CatBoost was used with default parameters. Since accuracy, as well as average precision and recall were better by 0.01 with the TfidfVectorizer, only the TfidfVectorizer was used in subsequent experiments.
2. A comparison between text data **with and without negation encoding** was conducted (with 67% of the training data for training and 33% of the training data for validation). The default CatBoost classifier with TfidfVectorizer was used. Since adding the negation encoding changed the evaluation metrics by only $< 0.01$, the data without negation encoding was used in subsequent experiments for simplicity.
3. A grid search for the TfidfVectorizer parameters **min_df and max_df** was conducted. The values 0.001, 0.005, 0.01 and 0.85, 0.9, 0.95 were considered respectively. The best parameters, 0.001 for min_df and 0.9 for max_df, were used in subsequent experiments. Note that the chosen parameter value for min_df was the smallest among the tested values. However, it was not decreased further due to memory limitations.
4. A random search (with 15 iterations) was done for the **CatBoost hyperparameters** learning_rate (0.015, 0.02, 0.025, 0.03, 0.035), depth (5, 6, 7, 8, 9), and iterations (750, 800, 850, 900, 950). The best parameters were a depth of 8, a number of iterations of 900 and a learning_rate of 0.03.

In summary, the final CatBoost model was a CatBoost classifier with depth of 8, 900 iterations and a learning_rate of 0.03. The TfidfVectorizer was used with parameters 0.001 for min_df and 0.9 for max_df. Only unigrams were used.

## 4.5 Best Model Evaluation

In addition to evaluating class-wise precision and the confusion matrix, a backtesting with simplified assumptions (below) was conducted to evaluate the final model on the test data from QTR1 2019. The backtesting was conducted in two variations. First, a trade was placed whenever the model classified lg_inc. Second, a trade was placed whenever the model classified lg_inc with a probability larger than 40%. The threshold of 40% was determined as a reasonable threshold based on the distribution of the predicted probabilities of the classified classes for the test data, displayed in figure 6. Basically, this figure shows the softmax probabilities corresponding to the classes that were predicted by CatBoost for the test observations from QTR1 2019. Note that for each observation, the class corresponding to the largest softmax probability gets predicted. The backtesting was done with the following simplifying assumptions:

1. Amount of money placed on each trade: $2K
2. Cash in portfolio: enough to place any trade
3. Commission per trade: $0
4. Max. loss of per trade: 5% (i.e. $100)
5. Max. percentage change realized in case of profit: 50%

When a trade was placed for a certain 8-K filing, the corresponding normalized percentage change before discretization (as described in 3.4.1) was used to calculate the profit. For each trade $2K were used as investment. With a positive percentage change of e.g. +10%, the profit would be $100, computed as $\$2000 \cdot 0.1 \cdot 0.5$. The max. percentage change realized in case of profit is only set to 50% because parts of the price increase might happen before we might enter a trade (in pre- or after-market hours). With a negative percentage change of e.g. -10%, the loss would be -$100, computed as $\$2000 \cdot \max(-0.1, -0.05)$. The max. loss per trade reflects a stop-loss, i.e. we would abort a trade when reaching a 5% loss (e.g. by selling all shares in case of a long position).

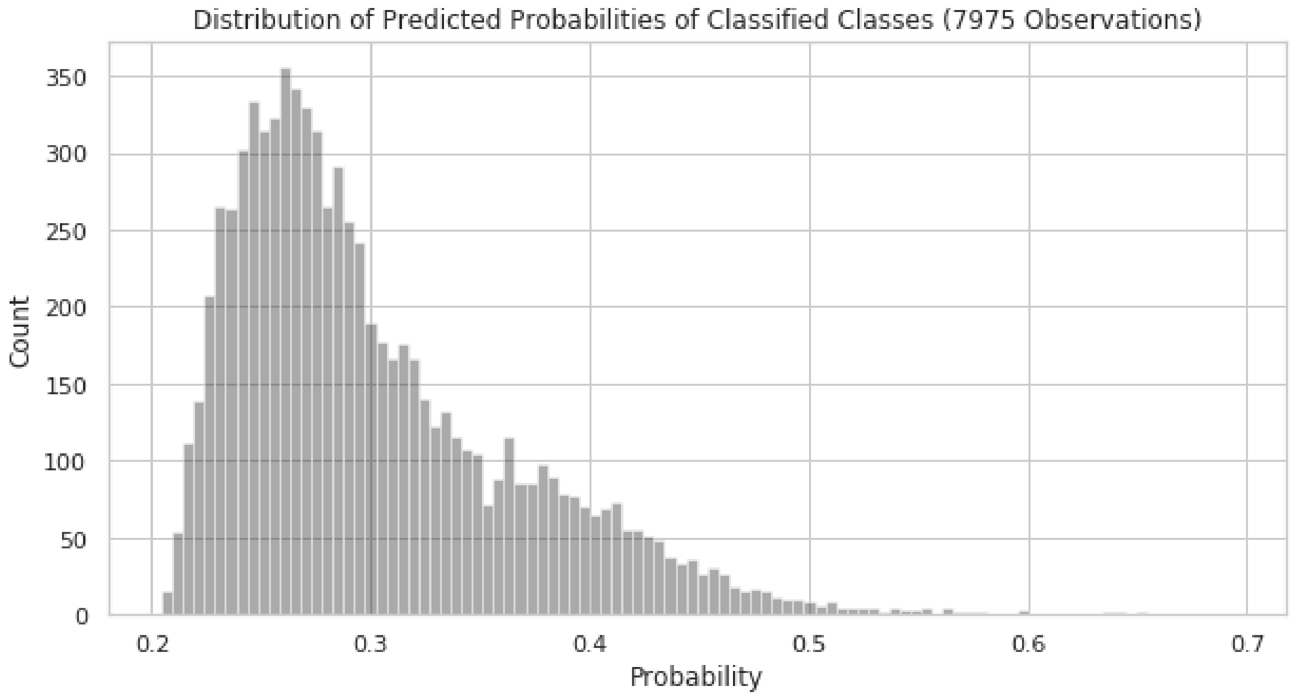

Figure 6: Distribution of predicted probabilities of classified classes

## 5 Results

### 5.1 Model Comparison

The performance of all classifiers on the training and validation data can be found in table 4. The best model on the validation data was the CatBoost classifier with an accuracy of 31%, representing an increase of 11% compared to the majority classifier with 20% accuracy. The best model is followed by logistic regression with an accuracy of 29%.

Table 4: Classifier performance on the training and validation data

| Data | Model | accuracy | precision | recall | f1 |
|---|---|---|---|---|---|
| | Random Guess | 0.20 | 0.20 | 0.20 | 0.20 |
| | Majority Class | 0.20 | 0.20 | 0.20 | 0.20 |
| | Naive Bayes | 0.31 | 0.30 | 0.31 | 0.29 |
| Training | SVM | 0.43 | 0.44 | 0.43 | 0.42 |
| | Random Forrest | 0.98 | 0.98 | 0.98 | 0.98 |
| | Logistic Regression | 0.42 | 0.43 | 0.42 | 0.42 |
| | CatBoost (Final) | 0.62 | 0.63 | 0.62 | 0.62 |
| | Random Guess | 0.20 | 0.20 | 0.20 | 0.20 |
| | Majority Class | 0.20 | 0.20 | 0.20 | 0.20 |
| | Naive Bayes | 0.25 | 0.25 | 0.25 | 0.23 |
| Validation | SVM | 0.26 | 0.26 | 0.26 | 0.25 |
| | Random Forrest | 0.28 | 0.27 | 0.28 | 0.27 |
| | Logistic Regression | 0.29 | 0.29 | 0.29 | 0.28 |
| | **CatBoost (Final)** | **0.31** | **0.30** | **0.31** | **0.30** |

## 5.2 Best Model Evaluation

### 5.2.1 Evaluation Metrics

The performance of the final CatBoost classifier on the test data from QTR1 2019 is displayed in table 5. The corresponding confusion matrix with class-wise precision is shown in table 6. The performance on the test data is very similar to the performance on the validation data (with an accuracy of 30% instead of 31%). The largest precision is achieved for the class lg_inc, with a value of 0.36. This means that when classifying lg_inc, the model is correct 36% of the time. The model makes rather many mistakes when classifying sm_inc and sm_dec.

Table 5: Classifier performance on the test data

| Model | accuracy | precision | recall | f1 |
|---|---|---|---|---|
| CatBoost (Final) | 0.30 | 0.30 | 0.30 | 0.30 |

Table 6: Confusion matrix of best model evaluated on test data

| | | Predicted | | | | |
|---|---|---|---|---|---|---|
| | | lg_dec | sm_dec | same | sm_inc | lg_inc |
| True | lg_dec | 496 | 248 | 235 | 226 | 435 |
| | sm_dec | 200 | 281 | 542 | 340 | 220 |
| | same | 136 | 293 | 688 | 357 | 161 |
| | sm_inc | 200 | 258 | 513 | 335 | 218 |
| | lg_inc | 409 | 187 | 204 | 200 | 593 |
| Precision | | 0.34 | 0.23 | 0.32 | 0.22 | 0.36 |

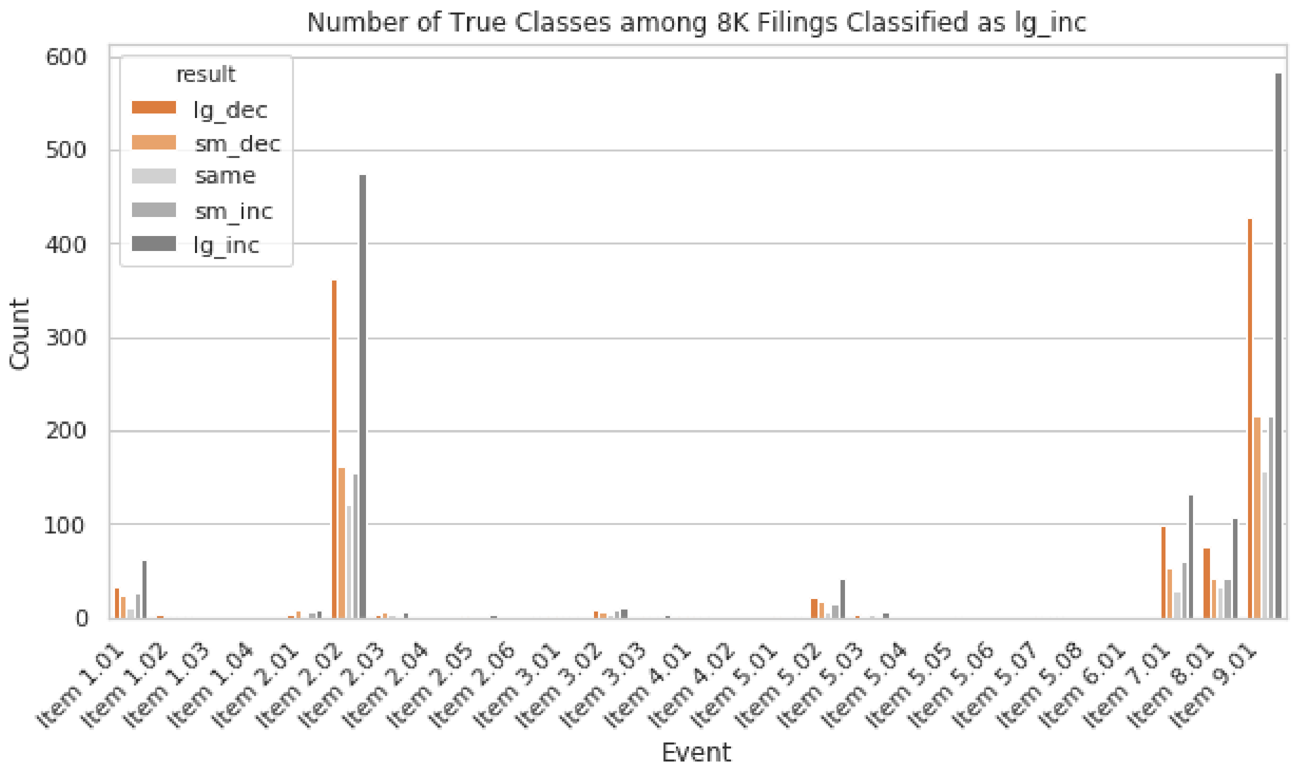

Figure 7: Number of true classes among 8-K filings classified as lg_inc by 8-K event

Since lg_inc has the highest precision, figure 7 shows the number of true classes among all observations that were classified as lg_inc by 8-K event. E.g. approx. 570 of the 9.01 events classified as lg_inc had the true class lg_inc but approx. 440 of the 9.01 events classified as lg_inc had the true class lg_dec. There are only slight differences in precision across the different 8-K events. However, for events with little available data (e.g. 2.01, 2.03 and 3.02) the number of correct classifications is as large as the number of false classifications.

### 5.2.2 Backtesting

The backtesting results are summarized in table 7 and the cumulative profits across trades are shown in the figures 8 and 9. The figures also show a random trader in comparison, which randomly places one trade for each trade of the non-random trader. Any 8-K filings from the corresponding date (of the non-random trader's trade) can be traded by the random trader. Visibly, the random traders are not profitable over time whereas the non-random traders are.

The table shows that making use of the predicted class probabilities from the CatBoost classifier improves the trading results. If only trades are placed when lg_inc is classified with a probability of $> 0.4$, then the percentage of trades with profit increases from 55% to 62%, the total number of trades reduces to 27% while the total profit only reduces to 85% (compared to the trader that always places a trade whenever lg_inc is classified).

Table 7: Backtesting results

| Type | No. trades | % of trades with profit | % of trades with loss | Total profit |
|---|---:|---:|---:|---:|
| Trade all lg_inc | 1628 | 55% | 45% | \$8797 |
| Trade lg_inc with prob. $> 0.40$ | 443 | 62% | 38% | \$7450 |

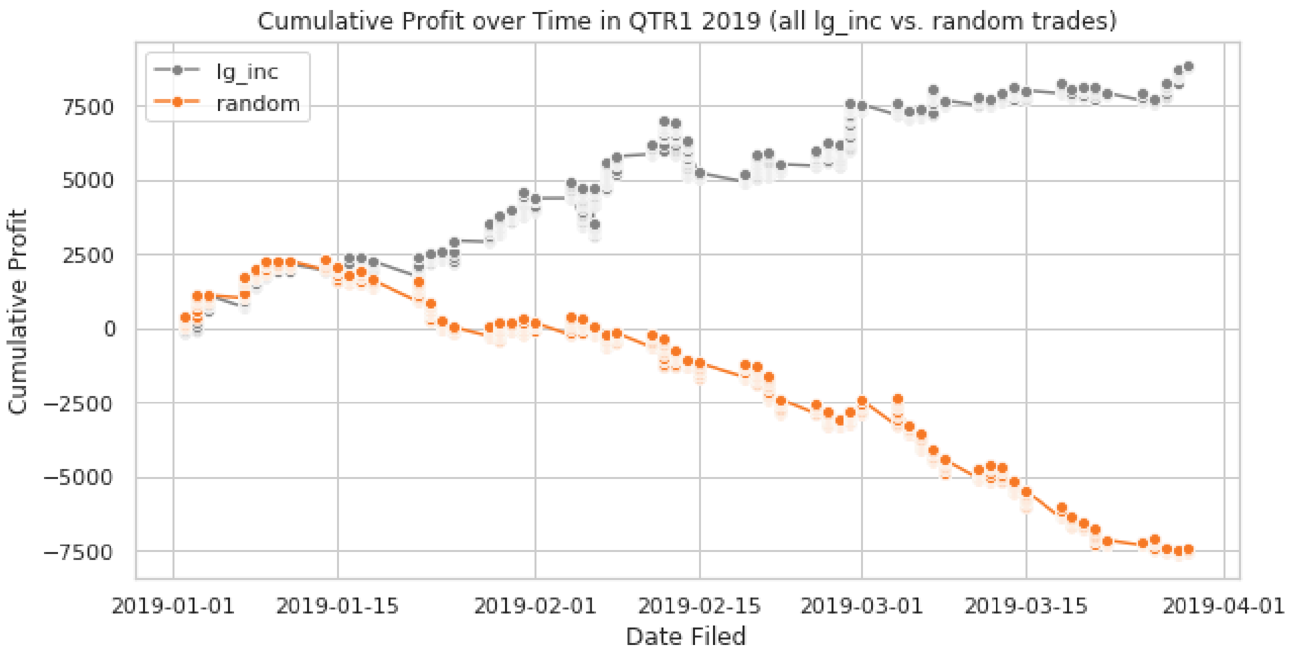

Figure 8: Cumulative profit over time in QTR1 2019 (all lg_inc vs. random trades)

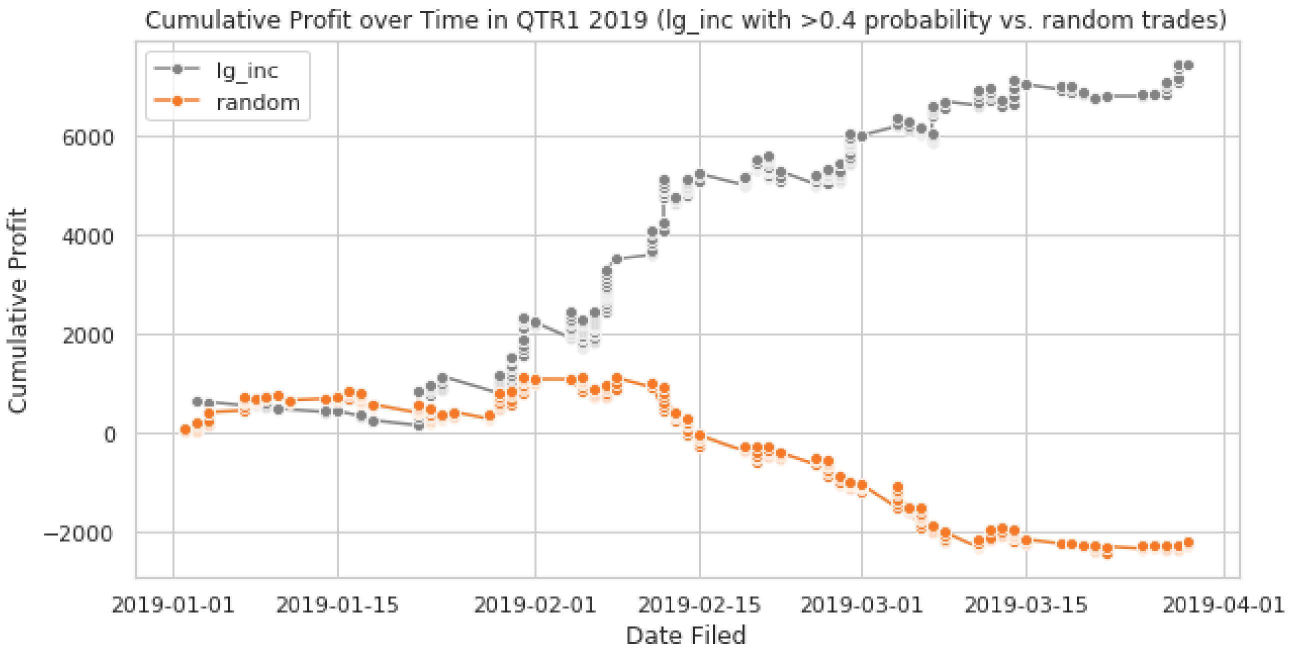

Figure 9: Cumulative profit over time in QTR1 2019 (lg_inc with $> 0.4$ probability vs. random trades)

### 5.2.3 Important Features

A word cloud with features weighted by feature importance is shown in figure 10. The feature importance is a normalized CatBoost metric that reflects how much the prediction values change when the feature changes [14]. In the word cloud, larger words have a larger feature importance.

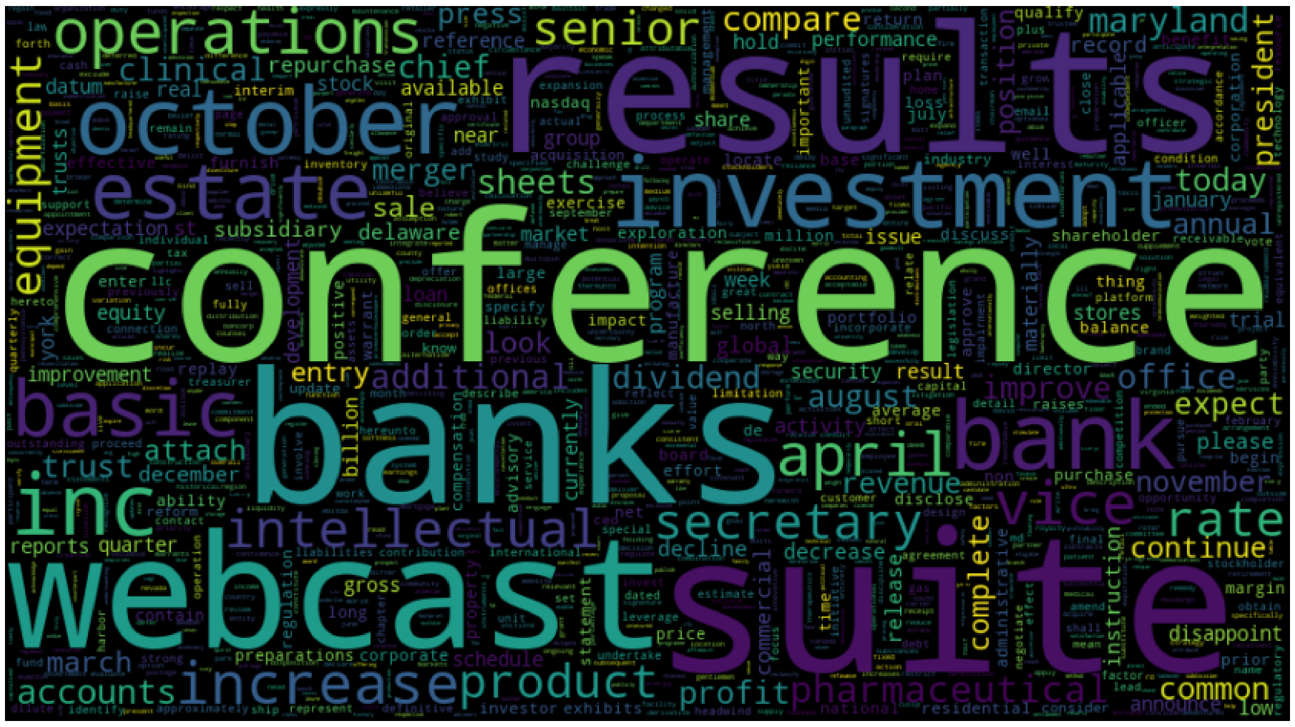

Figure 10: Word cloud of features weighted by feature importance

# 6 Discussion

## 6.1 Model Performance

Considering that the neural networks with GloVe word embeddings by Saleh et al. achieved an increase in accuracy by 9 percentage points compared to a majority classifier, the achieved results in this project are good [5]. The best model of this project improved the test accuracy by 10 percentage points compared to a majority classifier. This comparison is not ideal due to differences in pre-processing, e.g. Saleh et al. used three classes instead of five. However, it is sufficient to serve as a general reference.

Moreover, one can see that the many features used by the best model, i.e. the features with a certain feature importance, seem reasonable. The word cloud in figure 10 shows many words that seem relevant for stock price changes, e.g. increase, decrease, disappoint, profit, and improvement. However, there are also words that seem less relevant, e.g. months like April, abbreviations like inc, and other terms like conference. This indicates that the model did well in identifying relevant tokens as features but that it might be mistaken about other tokens.

Even though the classes lg_inc and lg_dec have the largest class-wise precisions with 36% and 34% respectively, the model has a particular weakness for these classes: among the observations classified as lg_inc, most turn out to truly have the class lg_inc *or* lg_dec. The same is true for observations classified as lg_dec. This means that the model is good at identifying extreme percentage changes but not necessarily at identifying the direction of the percentage changes. In other words, if the model classifies any class with prefix lg_, then the true class will likely have the prefix lg_ but the classified direction _dec or _inc may not match with the true direction. Unfortunately, negation encoding did not lead to any improvement regarding this issue.

## 6.2 Practical Implications

Nevertheless, the results indicate that 8-K filings can be useful in generating trading signals. As the backtesting shows, simply placing trades when the model classifies lg_inc may be sufficient to generate profits in the long-term. However, the backtesting was conducted with simplified assumptions, e.g. the stop-loss of 5%. In practice, we may not always be able to cut the losses at 5%, e.g. during pre- or after-market hours when no trades are possible. Future projects should investigate whether these assumptions hold in practice. In addition, one might compare the 8-K filing strategy with trading strategies other than a random trader to have a more fair comparison.

## 6.3 Future Projects

Above all, future projects should investigate possibilities to further improve the accuracy and class-wise precision. Three potential projects could be the following. First, one could add other data sources such as stock price time series and fundamental data about the companies. Second, one could compare CatBoost with more complex models such as neural networks with word embeddings like BERT. Third, one could investigate the 8-K filings that are associated with certain percentage changes using methods such as topic modeling to understand better which 8-K filings may actually lead to certain percentage changes such as a lg_inc.

# 7 Conclusion

The CatBoost classifier was shown to be suitable model for this problem. It identified relevant text features from the 8-K filings and outperformed all other baseline models with an accuracy of 31% on the validation data (compared to 20% with a majority classifier). This performance generalized well to the test data with an accuracy of 30%. The largest class-wise precision was achieved for the class lg_inc, namely a precision of of 36%. A backtesting, where a trade was placed for every lg_inc classification, turned out to be profitable. In conclusion, forecasting stock price changes based on 8-K filings with GBDT has potential for usage in a trading application.

## References

[1] *Alpaca - Commission-Free API First Stock Brokerage*, en. [Online]. Available: `https://alpaca.markets` (visited on 11/23/2019).

[2] *SEC.gov | Form 8-K.* [Online]. Available: `https://www.sec.gov/fast-answers/answersform8khtm.html` (visited on 11/23/2019).

[3] *SEC.gov | What We Do.* [Online]. Available: `https://www.sec.gov/Article/whatwedo.html` (visited on 11/23/2019).

[4] W. Kenton, *8-K (Form 8k)*, en. [Online]. Available: `https://www.investopedia.com/terms/1/8-k.asp` (visited on 11/23/2019).

[5] H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky, "On the Importance of Text Analysis for Stock Price Prediction.," in *LREC*, 2014, pp. 1170–1175.

[6] M. Saleh and S. Nair, "Neural based event-driven stock rally prediction using SEC filings and Twitter data," en, p. 8,

[7] R. Holowczak, D. Louton, and H. Saraoglu, "Testing market response to auditor change filings: A comparison of machine learning classifiers," en, *The Journal of Finance and Data Science*, vol. 5, no. 1, pp. 48–59, Mar. 2019, ISSN: 24059188. DOI: `10.1016/j.jfds.2018.08.001`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S2405918818300096` (visited on 11/28/2019).

[8] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," en, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pp. 785–794, 2016, arXiv: 1603.02754. DOI: `10.1145/2939672.2939785`. [Online]. Available: `http://arxiv.org/abs/1603.02754` (visited on 12/02/2019).

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[10] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[11] *A Kaggle Master Explains Gradient Boosting | No Free Hunch.* [Online]. Available: `http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/` (visited on 12/06/2019).

[12] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[13] *Multiclassification: Objectives and metrics - CatBoost. Documentation*, en, concept. [Online]. Available: `https://catboost.ai/docs/concepts/loss-functions-multiclassification.html` (visited on 12/02/2019).

[14] *Feature importance - CatBoost. Documentation.* [Online]. Available: `https://catboost.ai/docs/concepts/fstr.html#fstr__regular-feature-importances__lossfunctionchange` (visited on 12/26/2019).