ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

PREDICTING STOCK MOVEMENT FROM PATENT DATA IN THE PHARMACEUTICAL INDUSTRY

MACHINE LEARNING FOR FINANCE

William Martin

14th January 2021

Contents

INTRODUCTION

Predicting the movement of stock prices is a tedious task and depends on various factors ranging from fundamental data to public news. Under the efficient-market hypothesis (EMH), stock prices should reflect all publicly available information. In this context, we are interested in assessing the predictive power of innovation within a company, namely through patent activity, on the movement of its stock price. In view of the current situation, we will namely focus on the pharmaceutical industry and determine if information contained in patent applications, granted patents, and R&D activity in general are an ex-ante indicator of company profitability. To this end, we will collect patent and stock data of major pharmaceutical companies and assess through machine learning models the predictive power of the data on stock price movement (upward or downward). In particular, we will use natural language processing (NLP) to extract key subjects from patents and feed them to deep learning models.

Some research already exists pertaining to determining the relation between stock returns and patent activities. Vitt et al [1] investigated the connection between the patent activities of high-tech companies and their stock returns, drift and volatility using an autoregressive model. Hong et al [2] take a more traditional approach and extend the Fama-French multi-factor model with factors related to patents and R&D expenditure. A similar approach to the methodology we propose has been explored by Yu-Shan et al [3], who use numerical data of patents as independent variables to an artificial neural network. Finally, a thorough technical report from the Fraunhofer Institute [4] studies the effect of various patent factors (number of refused patents and number of forward citations just to name a few) on firm performance using multivariate models.

By using natural language processing on patent text and deep learning for predicting stock movements, we hope to differ from research already done in the patent-and-stock field and determine if combining machine learning techniques can recover non-linear relationships between patent information and stock returns.

DATA

This chapter describes the procedure followed for collecting, analyzing, cleaning, and transforming the data needed to train our machine learning models. For this project we have limited our study to the 22 best companies in drug research according to a Forbes article [5] and to 10 full consecutive years (2010 to 2019 included).

2.1 DATA COLLECTION

Perhaps the most tedious part of any data science project is the collection of data, whether it be open-source or not. Luckily for us, a patent is disclosed to the public in a patent application [6] and there exist a plurality of frameworks to extract patent metadata. Open-source *Python* libraries such as *pypatent* [7] and *patent-client* [8] are available, but after running into limitations in both packages we have decided to go for *PatentsView API* [9], which is supported by the US Patent and Trademark Office (USPTO). This API allows users to freely explore US intellectual property (IP) without having to use a token and is conveniently based on *HTTP* requests. Their database can be accessed from different API endpoints, each granting access to various different fields. Table 2.1 enumerates the list of accessible field names and their corresponding endpoint that we have chosen for this project.

API field name	Endpoint
app_date	applications
app_number	applications
assignee_organization	assignees
cited_patent_number	cited_patents
citedby_patent_number	citedby_patents
cpc_group_title	cpcs
cpc_subgroup_title	cpcs
cpc_subsection_title	cpcs
inventor_id	inventors
nber_subcategory_title	nbers

API field name	Endpoint
patent_abstract	patents
patent_date	patents
patent_num_claims	patents
patent_number	patents
patent_processing_time	patents
patent_title	patents
uspc_mainclass_title	uspcs
uspc_subclass_title	uspcs
wipo_field_title	wipos
wipo_sector_title	wipos

TABLE 2.1
Table of endpoints and field names of *PatentsView API*

The idea is to extract sensible features that would be fed into our machine learning models, namely the filing date, number of inventors and claims of the patent, and sufficient text describing the patents in order to extract themes or topics. The latter is built from a concatenation of text found not only in the patent abstract and title but also in the classification of patents in technology categories by four distinct schemes:

1. US Patent Classification (USPC)
2. Cooperative Catent Classification (CPC)
3. The National Bureau of Economic Research (NBER)
4. World Intellectual Property Organization (WIPO)

To this end, we have compiled a set of *Python* functions[1] to download, parse, and save the data into compressed *.csv* files. The query was performed by filtering patent application dates between the 1st of January 2010 and the 31st of December 2019 and by the company name contained in the assignee organizations. Since we are dealing with a significant amount of data, namely in the form of text, compressing the data into different files divided by company and year is a judicious choice.

The second category of data needed for this project is the return of company stocks. Luckily, *EPFL* credentials give access to data of the Center for Research in Security Prices, LLC (CRSP) and the Fama-French Portfolios & Factors databases through the Wharton Research Data Services (WRDS) [10]. The documentation is clear and thorough and a set of *Python* functions[2] were written to download (via raw *SQL*) and save daily stock returns of the companies of interest, the daily return of the benchmark S&P500 index, the daily riskless rate, which corresponds to the one-month Treasury Bill rate, and finally the quarterly R&D and total operating expenditure of each company.

Finally, we have chosen to omit a few companies among the ones listed in the Forbes article [5], namely *Celgene*, *Allergan*, *Roche*, *Shire*, *Sanofi*, and *Bayer* since these companies are either not publicly traded, have been acquired by another company, or do not have data available through WRDS. Indeed, the CRSP database only includes prices for stocks traded at the NYSE, AMEX, NASDAQ, and NYSE Arca (*Roche* and *Bayer* for instance do not trade on these markets). Other sources of data have been explored such as the *Python* package *yfinance* and data providers such as Alpha Vantage but these sources are either unreliable, do not give access to sufficient data under a free tier account, or require an API token for a subscription fee.

2.2 Exploratory data analysis

After saving all the data collected, we take a look at the exploratory data analysis part. This section is important in a sense that we need to assess the quality of our data by investigating its contents. Moreover, it is also good practice to get to know the data before training a machine learning model on it.

2.2.1 Patent data

The patent data at hand contains patent information of the biggest pharmaceutical companies over the years 2010 through 2019. Our table has approximately 11'000 entries and 20 exogenous variables. We first examine the number of missing values in each patent feature[3]. Figure 2.1 depicts features for which there is at least one observation with missing data, where the numerical value indicates the number of missing values. We notice that text-only cells are missing. This is not too problematic since we are planning on concatenating all text-related information together.

We can next inspect the numerical features for anomalies. Figure 2.2 depicts the histograms of the number of inventors, number of claims, number of cited patented, and number of patents citing the patent. Our source of data is pretty reliable: there are no noticeable outliers that would require dropping certain observations with regards to numerical features.

In terms of patent application date, the first took place on the 4th of January 2010 and the last on the 20th of December 2019. The first patent was granted on the 31st of August 2010 and the last on the 29th of September 2020. Since the returns of stocks have only been queried until the end of 2019 with respect to the patent application date, we will need to exclude granted dates that are outside our window of analysis and beyond the time-frame of our event study.

[1] Please refer to *Python* script *main_patent_data.py* for the implementation

[2] Please refer to *Python* script *main_returns_data.py* for the implementation

[3] Please refer to the *Jupyter Notebook main_patent_data_clean.ipynb* for results in this section

Finally, we take a look at the evolution of the number of patents granted and their fields of application from years 2012 through 2019. The most common fields are organic fine chemistry, pharmaceuticals and biotechnology which is rather comforting since we are dealing with companies doing research on drugs. The number of granted patents seems relatively low in 2012 and 2019: this can be explained by the fact that we have queried application dates between 2010 and 2019, thus some applications outside these dates would have been granted in this interval or beyond 2019 (Figure 2.3, mind the log scale).

	#nan
cpc_group_title	532
cpc_subgroup_title	532
cpc_subsection_title	532
nber_subcategory_title	7742
patent_abstract	360
uspc_mainclass_title	3319
uspc_subclass_title	7548
wipo_field_title	534
wipo_sector_title	534

FIGURE 2.1
Missing values in each feature from collected data

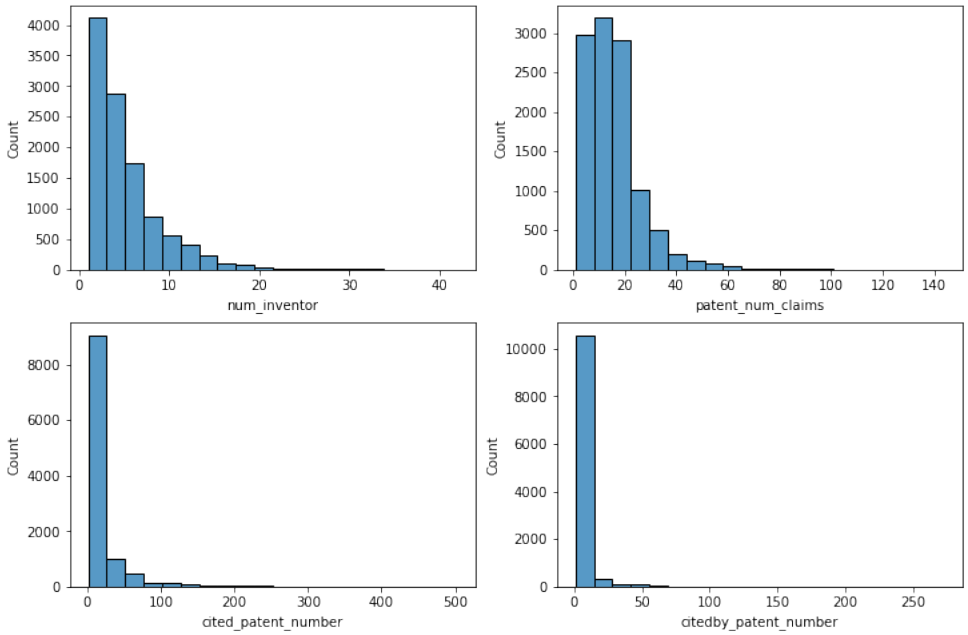

FIGURE 2.2
Histogram of some numerical features

2.2.2 RETURNS DATA

The data at hand contains the daily returns of stocks of the biggest pharmaceutical companies, the return of the S&P, and the riskless rate over the years 2010 through 2019. Our returns of stocks table has approximately 2500 entries and 16 columns corresponding to the companies.

There are quasi no missing values[4]. This is expected since the WRDS database is very reliable. Only ABBV (AbbVie Inc) has a significant number of observations missing (around 750) but this is due to the fact that this company has only been publicly traded since the beginning of 2013. We can plot the distribution of daily returns to assess any strange behaviors (Figure 2.4). The returns seem to follow a normal or Laplace distribution with mean close to 0 and varying standard deviations. We next plot the cumulative returns normalized by the return on the first date in 2010 of our benchmark, chosen to be the S&P500 (Figure 2.5). Comparing this with the level of the S&P500 that we easily find online, we see that the results are proportional.

[4]Please refer to the *Jupyter Notebook main_returns_data_clean.ipynb* for results in this section

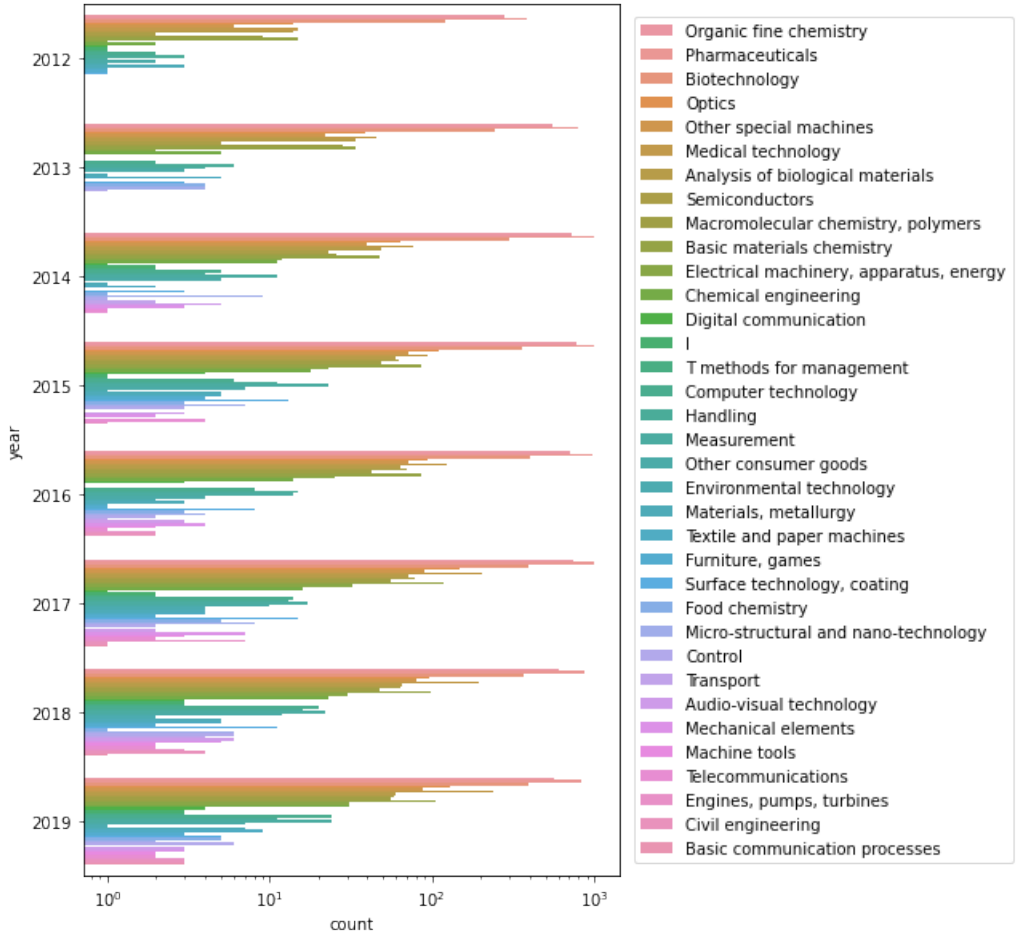

FIGURE 2.3
Evolution of number of patents granted per field

2.3 DATA CLEANING

Cleaning the data of returns is very straightforward[5]. Stocks REGN and VRTX have a combined total of 3 missing values. We simply replace these by 0. This is a wise choice since the common method when dealing with missing stock prices is to propagate the last valid observation forward, thus two of the same consecutive prices result in a return of 0.

Cleaning the patent data is more tedious[6]. First, we assume our goal is to predict daily stock returns for x days after an application is filed or a patent is granted, where x can take any positive integer with maximum value of 2 months. We therefore need to exclude all dates in our patent dataset that are beyond the 31st of December 2019 and 2 months beyond the last patent date. We recall that we queried patents according to the application date so there are surely some patents granted in 2020 and beyond.

Moreover, we want to adjust the returns of stocks with respect to the performance of the benchmark, that is the S&P500. This will allow us to take into account events that impact the overall market and adjust the returns of stocks appropriately. To do this, we make use of the Capital Asset Pricing Model (CAPM):

$$\mathbb{E}[R_i] - R_f = \alpha_i + \beta_i(\mathbb{E}[R_m] - R_i)$$

where $\mathbb{E}[R_i]$ is the expected return of stock i, R_f the riskless rate, and $\mathbb{E}[R_m] - R_f$ the market risk premium. β_i is the loading on this premium and is a measure of risk of stock i relative to the market and α_i is the measure of out/under-performance of the stock relative to the overall market. We wish to

[5]Please refer to the *Jupyter Notebook main_returns_data_clean.ipynb* for the implementation
[6]Please refer to the *Jupyter Notebook main_patent_data_clean.ipynb* for the implementation

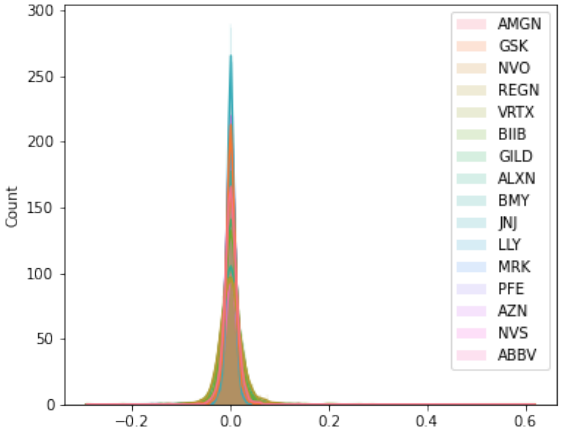

FIGURE 2.4
Distributions and their kernel density estimates of daily stock returns of companies of interest

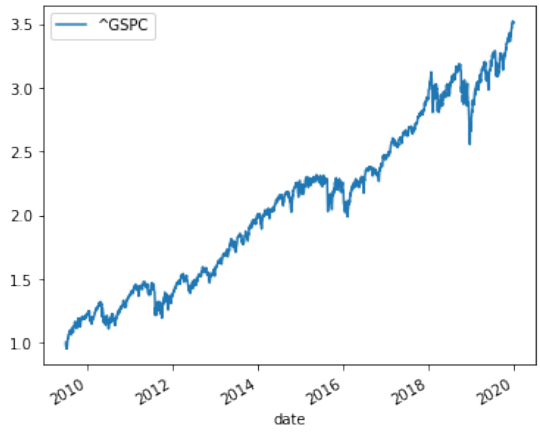

FIGURE 2.5
Compounded returns of the S&P500 from 2010 to 2019

retrieve α_i and use this as a proxy for adjusted stock returns. In order to do this, we first compute β_i of all stocks by deploying a 6-month horizon rolling window, with a time step of a day, on the following formula

$$\beta_i = \frac{Cov(R_m, R_i)}{Var(R_m)}$$

We are therefore required to take into account and download the returns from the the 1st of July 2009 in order to not loose 6 months of data. Finally, this gives us a table of daily betas of the same size of the table of stock returns, and this allows use to extract the daily α_i, which correspond to desired adjusted returns.

2.3.1 OTHER NUMERICAL DATA

The risk-free rate as well as the returns of the S&P500 have no missing values. As for the quarterly R&D expenditures, some intermittent values were missing. A simple linear interpolation method was employed to fill these missing values. Finally, some missing values of expenditures were missing at the beginning of the time series: since we cannot interpolate in this case, we have simply performed a backward fill using the first available datapoint.

2.3.2 TEXT CLEANING

In this part, we prepare the patent data for topic extraction. Our procedure for text cleaning is a combination of the procedure followed in an article from *Medium* [11] and techniques described in the lecture notes. The steps of our procedure are as follows for every patent[7]:

1. Concatenate title and abstract text. Missing values in any fields are replaced by an empty string.
2. From the aggregated text, remove non alphanumerical characters (e.g., encoding, special, and escape characters), convert to lower-case, replace all numbers by the word 'number', remove email addresses and URLs, replace the dollar sign $ by the word 'dollar', remove punctuation characters, and finally remove extra spaces between words.
3. Tokenize the text with the help of the *Python* package *nltk* [12].
4. Remove tokens that are 2 characters or less. We assume that a 2-character word will not be able to provide much information when extracting topics.

[7]Please refer to the *Jupyter Notebook main_patent_data_clean.ipynb*

5. Remove stop words of the English dictionary. Here we have used the *nltk* package again and extended the dictionary of stop words with words that belong to the patent jargon (e.g., 'thereof', 'herein', or 'invention') as well as numbers written in words (e.g., 'one', 'two', etc...).
6. Remove words that are verbs. We assume that verbs do not bring much topic information when it comes to biomedical jargon.
7. Lemmatize the tokens using again the *nltk* package. We have chosen lemmatization over stemming to reduce inflectional forms of tokens because stemming will remove derivational affixes and thus will not always return an existing word [13]. This is an undesirable effect for topic extraction.

Finally, we omit spell checking and removal of misspells since we assume that patents are well-written and thoroughly revised documents.

2.4 FEATURE ENGINEERING

Features that we have created during the collection of data are the number of inventors and the number of cited patents. We have chosen to omit the number of patents citing the patent since this is considered as look-ahead bias, i.e. information that is not readily available at the time a patent application is filed. Moreover, we have created another feature corresponding to the number of patent applications filed in the last $d_{delay} = 60$ days. In this context, we assume that if a company is repeatedly filing patents then it has invested more in its R&D department and therefore promising results will be output and have an effect on the stock price. Finally, we have divided the R&D expenditure by the total operating expenses in order to have a normalized ratio for each company.

Next, we employ the *Python* library *gensim* [14] to perform topic extraction using Latent Dirichlet Allocation (LDA) and the procedure followed in [11][8]. An LDA model is fitted directly on the corpora of cleaned and tokenized text of all patents resulting from section 2.3.2. In this context, we thus generate n_{topics} topics, where a topic is a distribution over words. We then apply the trained model on each of the patents and recover the probabilities of a patent belonging to any of the n_{topics} topics. These probabilities (that add up to 1) are then used as exogenous variables and serve as factors to determine if the "subject" of the patent plays a role in the predictive power of the underlying stock price.

There exists various metrics to evaluate an LDA model in the context of topic extraction such as the perplexity or coherence scores [15]. However, we assume this is outside the scope of the goals of this project and rather decide to eyeball the results of the model.

Finally, we create the dependent variables that will serve as labels to our machine learning model. To this ends, we take the cumulative returns of x days, starting from one day before the application or granted date. This ensures us to capture a possible shift in stock price at the time the patent is filed or granted. We also define a binary value, which takes 1 if the cumulative returns are positive, and 0 otherwise.

[8] Please refer to the *Jupyter Notebook main_patent_lda.ipynb*

MODELS

In this chapter, we establish the general framework under which we carry out the modeling of the machine learning algorithms. We will compare the performance of the benchmark **logistic regression** against an **artificial neural network** (ANN) in a classification problem. The classification in question will be do categorize stock returns that have either increased or decreased following a patent application or grant from the underlying company.

We first recall the structure of our data. The independent data constructed from section 2.4 contain the following features:

- Number of patents cited.
- Number of inventors.
- Number of claims of the patent.
- Number of patent applications of the company prior to a particular patent application.
- n_{topics} columns corresponding to the probability of a patent belonging to any of the n_{topics} topics.

And every row of the data is either a granted patent or a patent application with an associated grant date or application date respectively. The variable to classify is the binary variable taking 1 if the cumulative returns are positive and 0 otherwise. The cumulative returns are computed on x days, starting from one day before the application or grant date of the patent.

3.1 FEATURE SELECTION AND TRANSFORMATION

Before feeding the data to our models, it is important to select features that are relevant to the problem we are trying to solve. It is important to go through this process for the following reasons:

- Reduce the complexity and training time of our models by selecting a subset of the current features.
- Reduce over-fitting and make our models more robust to the introduction of new data, for instance new patents.
- Remove features that could make our models worse, i.e. features that are negatively correlated with our target variable.

To this end, the data will go through feature filtering by principal component analysis (PCA) and we select features that represent 60% of the explained variance. Before doing this however, we normalize each independent variable by subtracting the mean and scaling to unit variance.

Finally, we split the data into 70% of training set and the remainder as the test set in a stratified fashion, meaning we make sure the proportions of each class are approximately the same in the training and test set.

3.2 HYPERPARAMETERS

When constructing our models, we will not only consider the fine-tuning of hyperparameters specific to each model but also of the high-level parameters that are used to construct the independent and dependent variables. The parameters that we will vary are the x days used to compute the cumulative returns and n_{topics}, the number of topics generated from the tokenized text of each patent. For hyperparameters specific to each model, we will use K-fold cross validation for the fine-tuning.

3.3 ARTIFICIAL NEURAL NETWORK

We make use of a simple feed-forward fully-connected neural network with sigmoid activation functions at each node and layer, which is the most suitable activation function when it comes to binary classification and convenient with our transformed data (ranging from 0 to 1). Moreover, we use stochastic gradient descent in the back-propagation process for faster computation and potentially faster convergence with a batch size of 256 (power of 2) and learning rate of 0.01. The learning rate should not be too high or else the model will not be able to properly converge towards a minimum. We choose an input layer of size equal to the number of features and an output layer of size 1, corresponding to the probability of either belonging to class 1 or 0. As for the hidden layers, we choose a single layer with a number of nodes being half the number of features. We observed from our experiments that having too many nodes in the hidden layers and more than one hidden layer often led to over-fitting (accuracy on training set significantly higher than on the test set). Moreover, we have selected at least one hidden layer in order to capture non-linearities from our data (no hidden layer would be a simple regression). Finally, in order to improve the generalization of the model, we make use of $l1$ activity regularization to apply a penalty on the layer's output. For the implementation of the neural network we make use of the library *Keras* [16] from *Tensorflow* [17].

3.4 MODEL EVALUATION

In order to evaluate the quality of our models, we use the accuracy score that is commonly used in classification problems and defined as

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where the terms in the fraction are depicted in the confusion matrix of Figure 3.2.

		Predicted		
		0	1	Total
Actual	0	TN	FP	$TN + FP$
	1	FN	TP	$FN + TP$
	Total	$TN + FN$	$FP + TP$	$TP + FP + TN + FN$

TABLE 3.2
Confusion matrix

As such, training the models with maximizing the accuracy in mind means we are giving as much important in correctly predicting an upward movement of a stock than a downward movement, as opposed to other metrics such as sensitivity and specificity that mainly focus on measuring the proportions of true positives and true negatives respectively.

RESULTS

4.1 LOGISTIC REGRESSION

We first train the benchmark logistic regression model following the general procedure described in the previous chapter and train various models by varying the data-related hyperparameters x (number of days used to compute cumulative returns) and n_{topics} (number of topics explaining each patent)[9]. For each model, we perform cross-validation on the model-specific hyper-parameter C (regularization term) in order to find a model that will generalize well over the overall data. Figure 4.6 depicts the accuracy score of the training set (left) and test set (right) as a function of x and n_{topics} for patent applications. Figure 4.7 on the other hand depicts the accuracy score for granted patents.

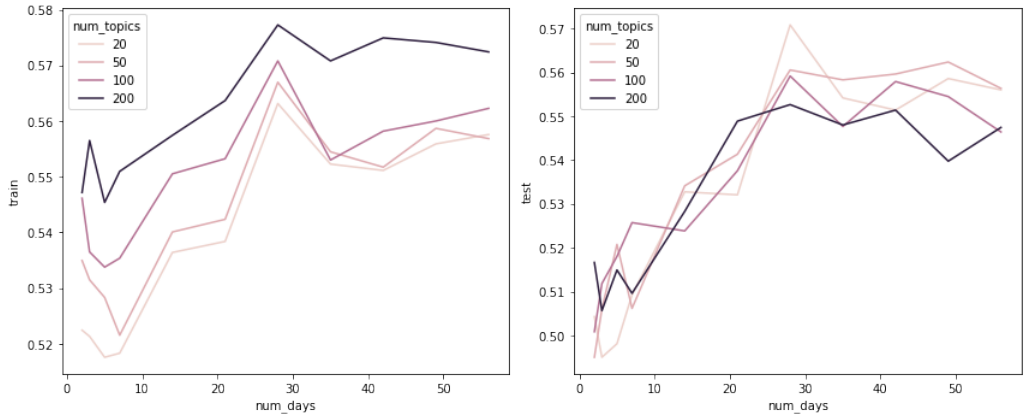

FIGURE 4.6
Accuracy vs. x vs. n_{topics} taking into account application dates of patents

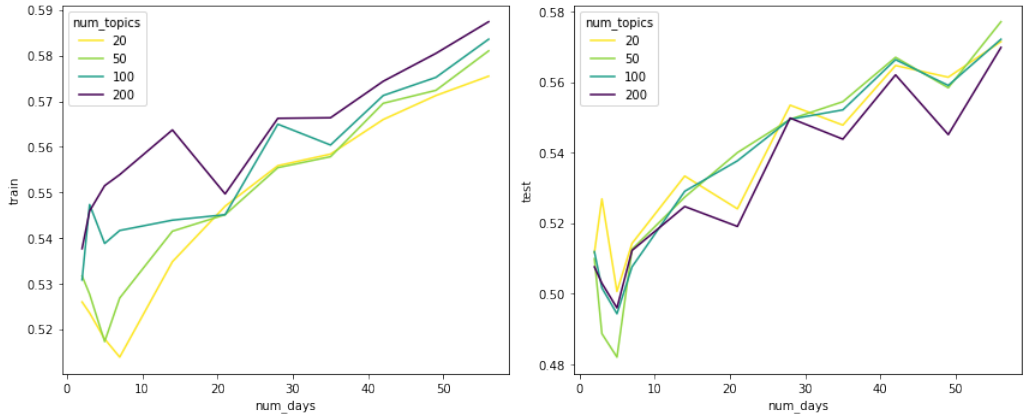

FIGURE 4.7
Accuracy vs. x vs. n_{topics} taking into account grant dates of patents

We observe that the overall accuracy score on the training and test set are similar, which means the trained models generalize well on the overall dataset and in general over-fitting is avoided. The accuracy score in itself however is far from ideal. Indeed, accuracy scores over all cases (application

[9]Please refer to the *Jupyter Notebook logistic_model.ipynb*

and grant dates included) range from 0.50 to 0.58 approximately, which means the models are just slightly better than random guessing (which would give 50% accuracy on average).

However, we can take away some interesting results: the models seems to better predict stock movements about a month after patent applications and this phenomenon is consistent across any number of topics. This might indicate that one month is the optimal time patent applications are reflected in the stock price. Indeed, as soon as a patent is filed, it will take some time and research for big investors to assess if the invention at hand is a groundbreaking innovation and a sign for company value growth. For granted patents, the accuracy score increases as x increases, regardless of the number of topics. This could be a signal that approved patents contain long-term information on the stock price direction. Indeed, an approved patent signifies that the underlying company has the exclusive legal rights to exploit the innovation behind the patent and exercise some sort of monopoly in the field of application of the patent.

In terms of number of topics describing the patents, a higher number gives a better score on the training set than the test set (for both patent applications and granted patents). This could be a sign that a high number of topics (and therefore a high number of features and more complex models) could be an overfitting factor. In general, 100 topics seem to give generalized results on both the training and test set.

Finally, the poor accuracy score can be explained by the results from the feature selection process (via PCA). Indeed, the PCA algorithm always needs to choose about half of the original features in order to reach 60% of explained variance. This means that almost all features have equally poor predictive power on the labels and this is a sign that either the features that we have generated/chosen are not capable of predicting stock movements or that there are missing features that could better explain stocks prices.

4.2 NEURAL NETWORK

In this section, we choose to train a neural network using $x = 28$ and $n_{topics} = 100$, which corresponds to the model with the best consistent results from the logistic regression model[10]. Moreover, we follow, as usual, the same procedure described in the previous chapter. For each run of training, we make use of the early stopping method which involves stopping the training phase at a certain epoch. This is used to avoid over-fitting of the model when there are too many epochs. Figure 4.8 depicts the evolution of the accuracy (left) and of the loss function (right) for both the training and test set for patent applications. Figure 4.9 depicts the same for granted patents.

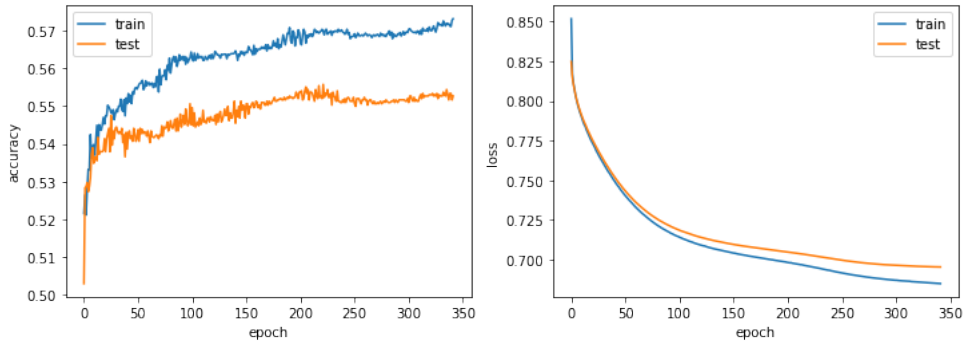

FIGURE 4.8
Evolution of accuracy (left) and loss function (right) for patent applications

Both runs stopped early at approximately 350 epochs (max. 1000) indicating that further training the model would lead to loss of generalization. The training is successful in a sense that accuracy scores

[10]Please refer to the *Jupyter Notebook ann_model.ipynb*

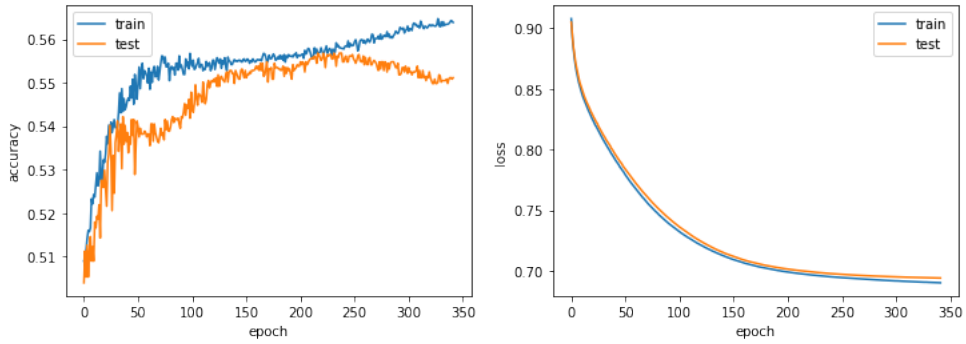

FIGURE 4.9
Evolution of accuracy (left) and loss function (right) for granted patents

increased and the loss function decreased. The result however are not ideal as in the case of logistic regression. Indeed, the accuracy is in the same range as in Figure 4.6 and Figure 4.7 (for 100 topics and 28 days), with logistic regression actually beating the neural network by a small margin. Exploring different combinations of number of nodes on each layer, number of layers, activation functions and regularization methods do not give better results.

4.3 DISCUSSION AND OTHER MODELS

We have tried deploying other models such as a decision trees classifier with bagging but the results are similar as for the logistic regression and neural network[11].

By exploring different machine learning algorithms and fine-tuning each model carefully, we have not managed to surpass an accuracy score of 60%. The models that we have trained perform slightly better than randomly predicting price movements (up or down) which is undesirable in trading and investing, especially if we do not know the intensity of the movement. However, one could argue that any strategy that is better than random is a good strategy but there are other models taking account different data and risk factors that could outperform the models we have developed here by a large margin. In order to properly assess the quality of our models, we would need to implement a trading strategy and backtest it with the results of our machine learning models. This could provide an extending scope for future works.

There are a few reasons for our suboptimal results. The first, as explained in the last section, is that all features contribute very little to the explained variance of the data. Indeed, in the case of 100 features, about 50 are selected to explain 60% of the variance, which is about 1.2% on average for each feature. This is a sign that the features that we have generated do not contain valuable information for stock movement prediction. The second reason is that even if we have accounted for the market movement in our stock returns (section 2.3), the patent metadata that we have considered cannot solely explain stock price trends and that there are a multitude of external factors (outside the patent and research domains) that can impact returns.

Finally, other factors could have been taken into consideration such as entire patent text (but this would require custom web scrapping scripts as opposed to the ready-to-use API), more companies in order to have more data for our models, different sectors in order to identify out/under-performing groups of sectors, or FDA (U.S Food and Drug Administration) approval/disapproval which can have an immediate effect on stock movement. The latter however would require complex web scrapping scripts or a subscription with a data provider since free-tier providers such as *WRDS* do not provide this kind of data.

[11] Please refer to the *Jupyter Notebook trees_model.ipynb*

CONCLUSION

In this project, we collect patent data with the help of an existing API and other metadata containing information on the R&D activity of major pharmaceutical companies from a reliable data provider. Moreover, a corpora is created from all text data of patents and natural language processing (NLP) is deployed to generate a number of subject topics. A probability of belonging to a topic is then associated to each patent. These probabilities together with the collected metadata of patents are then used to predict the stock movement (upward or downward) of the underlying companies.

To predict stock movements, we make use of three machine learning classification algorithms: a benchmark logistic regression classifier, decision tree classifier, and finally a simple feed-forward fully-connected neural network. In order to properly assess the prediction quality and generalization of our models, we make use of various methods to avoid over-fitting and evaluate the trained models by accuracy. Moreover, the models are trained by not only fine-tuning the model-specific hyperparameters but also the parameters that dictate the form of the exogenous and indigenous variables. In this context, we vary the number of different topics a patent can belong to and the time-horizon of the stock price used to determine its movement.

Our results reveal that information contained in patent data and R&D expenditure are poor stock movement predictors. Indeed, the accuracy threshold of 60% has never been reached but the models perform better than a strategy of random prediction (with an accuracy of 50% on average). These results could further be explained by the fact that each feature contributes to very little explained variance of the dataset, where 1.2% of variance is explained on average by the filtered features. Moreover, another reason can be that there are actually a large number of other factors that effect the stock market such as fundamental data and company news, despite the fact that we have taken into account the general behavior of the market. Furthermore, the results from logistic regression tend to indicate that stock price movements are more predictable when considering the evolution of the stock over one month and generating 100 topics from the patent corpora.

Finally, the results we obtain allow for more space for improvement. Indeed, we could look for other patent-related features with more predictive power such as FDA approval/disapproval. Moreover, we could further extend the scope of patents to more companies and companies in other sectors.

BIBLIOGRAPHY

[1] C. A. Vitt and H. Xiong. 'The Impact of Patent Activities on Stock Dynamics in the High-Tech Sector'. In: *2015 IEEE International Conference on Data Mining*. 2015, pp. 399–408. DOI: 10.1109/ICDM.2015.95.

[2] Gun Jea Yu and KiHoon Hong. 'Patents and R&D expenditure in explaining stock price movements'. In: *Finance Research Letters* 19 (2016), pp. 197–203. ISSN: 1544-6123. DOI: https://doi.org/10.1016/j.frl.2016.07.012. URL: http://www.sciencedirect.com/science/article/pii/S1544612316301350.

[3] Yu-Shan Chen and Ke-Chiun Chang. 'Exploring the nonlinear effects of patent citations, patent share and relative patent position on market value in the US pharmaceutical industry'. In: *Technology Analysis & Strategic Management - TECHNOL ANAL STRATEG MANAGE* 22 (Feb. 2010), pp. 153–169. DOI: 10.1080/09537320903498496.

[4] Peter Neuhäusler et al. *Patents and the financial performance of firms - An analysis based on stock market data*. Discussion Papers "Innovation Systems and Policy Analysis" 28. Fraunhofer Institute for Systems and Innovation Research (ISI), Feb. 2011. URL: https://ideas.repec.org/p/zbw/fisidp/28.html.

[5] Matthew Herper. *Who's The Best In Drug Research? 22 Companies Ranked*. May 2014. URL: https://www.forbes.com/sites/matthewherper/2014/05/22/new-report-ranks-22-drug-companies-based-on-rd/?sh=1db2001c4268.

[6] *World Intellectual Property Organization (WIPO)*. URL: https://www.wipo.int/portal/en/index.html.

[7] *pypatent, Python package*. URL: https://pypi.org/project/pypatent/.

[8] *patent-client, Python package*. URL: https://pypi.org/project/patent-client/.

[9] *PatentsView API*. URL: https://api.patentsview.org/doc.html.

[10] Wharton Research Data Services. *Wharton Research Data Services*. URL: https://wrds-www.wharton.upenn.edu/.

[11] Susan Li. *Topic Modelling in Python with NLTK and Gensim*. Apr. 2018. URL: https://towardsdatascience.com/topic-modelling-in-python-with-nltk-and-gensim-4ef03213cd21.

[12] *Natural Language Toolkit*. URL: http://www.nltk.org/.

[13] *Stemming and lemmatization*. Apr. 2009. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[14] *Gensim: topic modelling for humans*. Nov. 2020. URL: https://radimrehurek.com/gensim/.

[15] Selva Prabhakaran. *Topic Modeling in Python with Gensim*. July 2020. URL: https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/.

[16] François Chollet et al. *Keras*. https://keras.io. 2015.

[17] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.