Machine Learning Engineer Nanodegree

Capstone Project

Eidolon
April 24th, 2019

I. Definition

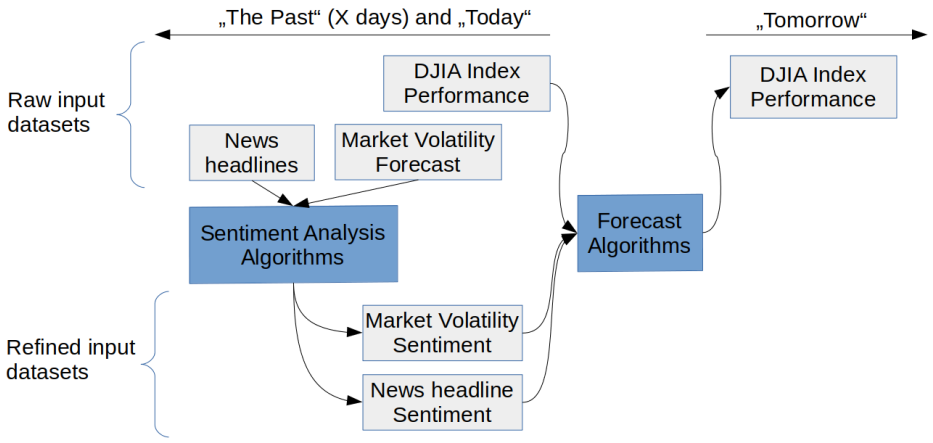
Project Overview

This Capstone project attempts to determine if future stock market index performance (index: Dow Jones Industrial Average - DJIA) can be predicted by creating a prediction model combining past DJIA performance with text sentiment analysis of past news headline and market volatility forecast data. The „sentiment“ of news headlines & market volatility is either negative (bad news / high volatility) or positive (good news / no or low volatility).

Even though economic theory states that „markets are rational“, studies such as this [1] and this [2] show that as long as humans are in the driver‘s seat, irrationality is at least sitting in the co-driver‘s seat. This has driven a field in finance named „behavioural finance“, investigating the psychological aspects of stock market trading. In the domain of artificial intelligence and machine learning, an important research domain intersecting with psychology is „text sentiment analysis“, aiming at predicting the sentiment (i.e. mood such as „negative“ or positive“) of written texts.

The basic idea of this capstone project is that the „psychological state of humankind“, which is theorised to be a factor in stock market development, is reflected in the sentiment of global news and market volatility forecast data, and can be used to predict future stock market development.

This is visualised in the following high level project overview:



[1] Rational or Irrational? A Comprehensive Studies on Stock Market Crashes

[2] The Impact of Behavioural Finance on Stock Markets

Problem Statement

ETFs (Exchange Traded Funds) are on the rise in stock market trading. This investment class replicates the companies represented in a stock market index such as the DJIA, either directly (direct DJIA company stocks) or „synthetically“ (meaning a part of ETF is invested outside DJIA companies, but still replicates DJIA performance). Bottom line: By investing in an DJIA-based ETF, one invests in the performance of the DJIA index.

Each day, ETF investors holding DJIA-based ETF stock are faced with the question to hold, buy or sell their assets, based on their expectation of the short, medium and long-term development of the DJIA.

> **Core problem statement:**
> **A machine learning model shall predict the *future development direction* of the DJIA index based on historical data to support the ETF investor in decision-making.**

The machine learning model shall use historical data of the DJIA index performance, and data about the „sentiment“ (mood) of news and market forecasts. The stock index prediction should address various timeframes (e.g. „tomorrow“, „1 week ahead“) so as to give the ETF investor a baseline for decision-making for various timeframes of trading. The development direction of stock index is expressed categorically only („small increase“; „huge decrease“ etc.). Rationale for choosing categorization (instead of keeping it a regression problem) is that it is expected upfront that the prediction accuracy will not be very high. Therefore, the goal is that the result of the model should not lead the investor to believe that it is a highly accurate forecast. By using categorical forecast values (i.e. ML classification problem) instead of numerical forecast values (i.e. ML regression problem), this goal is achieved.

Metrics

The appropriate evaluation metric is Accuracy, defined as: Accuracy of DJIA performance category prediction compared with actual DJIA performance category.

Ethics

The datasets used to tackle the problem are uncritical from an ethical point of view. They do not contain any personalised data; only publicly available non-personalised data is used.

However, if it is really possible to build a stock market predicition model with consistently high accuracy (>>50%) which enables investors to consistently „outwit the market“, serious ethical implications would arise which would need to be further discussed. Especially regarding the propagation and publication of the model. Is it ethical if one investor, or a small group of investors, exploit the market? What would happen if the model is publicised and everyone starts using it? What kind of stock market would such a „global-scale feedback loop“ create? Also, philosophical questions would arise, like „is it really that easy to understand the psyche of humankind with regard to stock market participancy?

II. Analysis

Data Exploration including Exploratory Visualizations

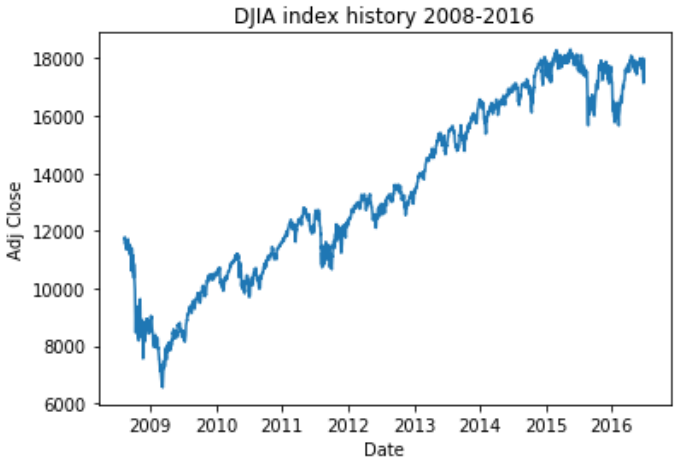
NOTE: Because for the text-based raw input datasets exploratory visualizations are not possible, additional data exploration, exploration and vizualisation steps are including in the „Data Preprocessing“ chapter as well, after text-based datasets have been transformed into numerical.

Dataset 1: DJIA index performance data ***– raw target prediction feature***

A dataset spanning the daily DJIA values for the years 2008-2016 is available from [2]. It has 1989 samples with 4 features each. For the purpose of this project, only the daily closing value of the DJIA is considered, adjusted for later stock splits. Opening or intra-day developments are discarded.

	Date	Adj Close
0	2016-07-01	17949.369141
1	2016-06-30	17929.990234
2	2016-06-29	17694.679688
3	2016-06-28	17409.720703
4	2016-06-27	17140.240234

	Adj Close
count	1989.000000
mean	13463.032255
std	3144.006996
min	6547.049805
25%	10913.379883
50%	13025.580078
75%	16478.410156
max	18312.390625



Dataset 2: Reddit news headline data ***– raw input feature***

Reddit is an aggregator for global news, providing a discussion forum for them. A pre-cleaned headline corpus with timeframe 2008-2016 is available from [2]. It contains 73,608 news headlines.

	Date	News
0	2016-07-01	A 117-year-old woman in Mexico City finally re...
1	2016-07-01	IMF chief backs Athens as permanent Olympic host
2	2016-07-01	The president of France says if Brexit won, so...
3	2016-07-01	British Man Who Must Give Police 24 Hours' Not...
4	2016-07-01	100+ Nobel laureates urge Greenpeace to stop o...

Dataset 3: ABC news headline data ***– raw input feature***

ABC is an Australian news broadcaster with global news. A pre-cleaned headline corpus with timeframe 2003-2017 is available from [3]. It contains 1,103,663 news headlines.

	publish_date	headline_text
0	2003-02-19	aba decides against community broadcasting lic...
1	2003-02-19	act fire witnesses must be aware of defamation
2	2003-02-19	a g calls for infrastructure protection summit
3	2003-02-19	air nz staff in aust strike for pay rise
4	2003-02-19	air nz strike to affect australian travellers

Dataset 4: Market Volatility Forecast data ***– raw input feature***

The economic calendar from investing.com contains a multitude of global market volatility forecasts. A pre-cleaned dataset with timeframe 2011-2019 is available from [4]. For the purpose of sentiment analysis, only the feature „expected volatility“ is important, because this is the only sentiment-including data from this dataset. Therefore, the remaining input features are discarded.

	Date	Vol
0	2011-01-01	Low Volatility Expected
1	2011-01-01	Low Volatility Expected
2	2011-01-01	Moderate Volatility Expected
3	2011-01-01	Low Volatility Expected
4	2011-01-01	Low Volatility Expected

Dataset sources:

[2] https://www.kaggle.com/aaron7sun/stocknews

[3] https://www.kaggle.com/therohk/million-headlines

[4] https://www.kaggle.com/devorvant/economic-calendar

Exploratory Visualization

NOTE: Due to the nature of the input data (text), exploratory visualizations are included in the chapters „Data Exploration“ and „Data Preprocessing“ instead.

Algorithms and Techniques

As outlined in the figure in the project overview chapter, this project has three major domains of algorithms and techniques. The description of each domain follows.

1. Sentiment Analysis algorithms and techniques

Depending on dataset, the technique to transform raw input datasets differs:

1. Reddit and ABC news headline data

- Using the global parameter „`daysback`“, the following algorithm is applied for all days in the range „today“ to „today minus `daysback`“
- For transforming the news headline into sentiment data, the Python text processing library TextBlob is used, which includes two ready-to-use text sentiment analyzers. First one is based on the „Pattern“ library, second is based on NaiveBayesClassifier being trained on an existing text corpus. Both are used for creating the numerical input features.
- Using TextBlob, every news headline (multiple per day) is turned into numerical values. TextBlob assigns each text sentiment values in the range of -1 (negative) to 1 (positive). The

output values differ slightly by analyzer. The Pattern analyzer creates a „sentiment“ value in range -1 to 1, and a „subjectivity“ value in the range 0 to 1. The NaiveBayes analyzer creates a „positive sentiment“ value in range 0 to 1, and a „negative sentiment“ value in the range 0 to 1. The final NaiveBayes „sentiment“ value in the range -1 to 1 is computed by substracting the negative from the positive sentiment value.

- For each of the „days back“ set in the global parameters, the mean, min and max values of the TextBlob sentiment values of both sentiment analyzers (Pattern and NaiveBayes) are calculated to be used as input features to the DJIA performance prediction algorithm.

2. Market Volatility data

- Aligned with the results of the TextBlob sentiment analyzers, the textual market volatility data for each of the „days back“ is transformed into numerical data using a very simple approach, based on the notion that market *volatility* is always negatively connotated:
    - text „low/moderate/high volatility expected“ = numerical -0.25 / -0.5 / -0.75

2. DJIA categorization algorithm

- For each day „today“, the future DJIA performance for „x days ahead“ is calculated and then categorized into eight categories.
- Based on the value of the `daysahead` global parameter, the absolute numerical value of the future DJIA performance is calculated

3. DJIA performance prediction algorithms and techniques

Due to the nature of the problem (multi-class classification), I benchmark all available such algorithms in the SciKit-Learn library, as outlined in the respective documentation here.

The technique for using a full-scale algorithm benchmark is using the model_selection.cross_val_score function/technique using the standard parameters of the algorithms. StratifiedKFold with 25 splits is used due to the finding from data processing that the DJIA performance class labels are imbalanced.

After extensive full-algorithm benchmarking, the accuracy of all classifiers are compared. For the top algorithms, GridSearchCV is then used to expand the hyperparameter space. Multi-Layer Perceptron is always included in GridSearch, in order to evaluate if it may be interesting to start setting up a Keras/TensorFlow model pipeline if time permits.

Benchmark

Stock market forecasts based on machine learning models are a focus area of financial research. Several papers focused on applying the ML technique to single stocks of individual companies show that prediction accuracy can reach the mid-end 70ies percent range.

For the stock market as a whole, I could not find any ML research papers which tackled this problem and thus might serve as a benchmark. However, I could find a meta-analysis of stock

market forecasts from professional stock market analysts in [X]. This shows that overall, there is an average prediction accuracy of 48% of all stock market forecasts in scope of the analysis.

So, the first benchmark for my model is 48% Accuracy to determine if the resulting prediction model is better than the average predicition accuracy of professional stock market analysts.

Second benchmark is a comparison with accuracy of guessing. When DJIA performance is categorized into eight categories, likelihood of guessing the correct one is 1/8th.

Therefore, the second benchmark for my model is 12.5% Accuracy to determine if the resulting prediction model is at least better than guessing.

III. Methodology

Data Preprocessing and Exploratory Vizualisations

The data preprocessing implements the algorithms & techniques explained in the previous chapter. Therefore this chapter is is also structured by dataset. First, the input datasets are individually processed. Then, they are merged on Date and further processed (scaled). Finally, the resulting data sets are stored on disk to be consumed by model

1. Normalisation on timeframe

As outlined before, the different input datasets have different timeframes. Therefore, the input datasets have to be normalised on the same timeframe. This reduces processing time later on.

Before normalisation:

```
DJIA   :  2008-08-08 00:00:00  to  2016-07-01 00:00:00
Reddit :  2008-06-08 00:00:00  to  2016-07-01 00:00:00
ABC    :  2003-02-19 00:00:00  to  2017-12-31 00:00:00
FOREX  :  2011-01-01 00:00:00  to  2019-02-13 00:00:00
```

After normalisation:

```
DJIA   :  2011-01-03 00:00:00  to  2016-07-01 00:00:00
Reddit :  2011-01-01 00:00:00  to  2016-07-01 00:00:00
ABC    :  2011-01-01 00:00:00  to  2016-07-01 00:00:00
FOREX  :  2011-01-01 00:00:00  to  2016-07-01 00:00:00
```

2. Data Preprocessing per dataset

Dataset 1: DJIA index performance data

Based on the daysahead variable, the performance of the DJIA from the current date to `daysahead` days in the future is calculated and then categorized into eight performance categories (four for DJIA decrease, four for DJIA increase). The categorization is performed dynamically based on the standard deviation of the resulting distribution for future performance.

The following DataFrame print show the resulting dataset and basic statistics for `daysahead=14`.

	Date	Adj Close	Future14DaysAheadAbs	Future14DaysAheadCat
0	2011-01-03	11670.750000	309.769531	high_increase
1	2011-01-04	11691.179688	286.010742	medium_increase
2	2011-01-05	11722.889648	262.550782	medium_increase
3	2011-01-06	11697.309570	292.520508	medium_increase
4	2011-01-07	11674.759766	148.940429	medium_increase

```
display(data_djia['Future14DaysAheadAbs'].describe())
```

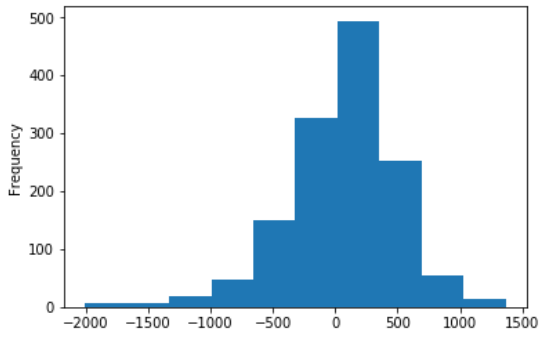
```
count    1370.000000
mean       60.785183
std       440.422621
min     -2004.469726
25%      -174.084717
50%       110.415527
75%       336.929443
max      1367.289063
Name: Future14DaysAheadAbs, dtype: float64
```

The data exploration and visualizations show that the resulting distribution of absolute future performance is always slightly positively skewed, but in general shows a normal distribution. This is explained through the performance of the DJIA as a whole in the timeframe with it‘s steady upwards tendency.

Data Visualisation for Future14DaysAheadAbs and count for resulting stdev-based categorization follows:

```
# show histogram for absolute future values
data_djia[col_name_numeric].plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8d387a908>
```

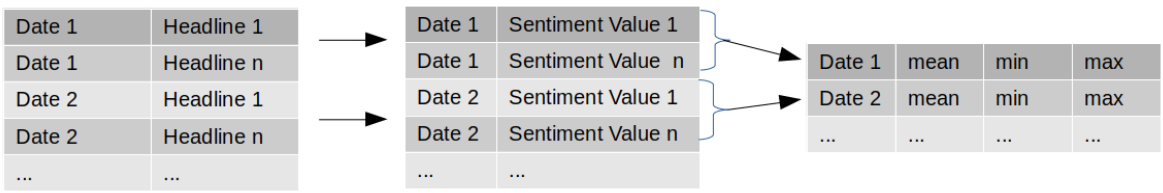



```
# show counts for categorization of absolute values
data_djia[col_name_categorical].value_counts()
```

```
awesome_increase     229
medium_increase      223
low_increase         218
high_increase        172
gruesome_decrease    166
low_decrease         153
medium_decrease      130
high_decrease         79
Name: Future14DaysAheadCat, dtype: int64
```

Dataset 2: Reddit news headline data

For each day, the Reddit dataset contains a multitude of headlines. Because it is impracticable to set up the input feature space with individual headlines, the sentiment analysis condenses the headline sentiment values into daily values (multiple headline sentiment values are aggregated into daily values).



Based on the global daysback parameter, the resulting dataset has a varying number of columns, defined dynamically as visualised in the following figure. Column prefixes are explained:

- **D0-** indicates the number of days back the sentiment value belongs to
- **R-** indicates the Reddit dataset origin (as opposed to A- which indicates the ABC dataset)
- **TB-** indicates the technique the sentiment value was generated: **T**ext**B**lob
- **PA/NB-** indicate the analyzer of TextBlob (Pattern Analyzer or NaiveBayes Analyzer)
- **Pol/Sub-** indicates **Pol**arity (positive or negative sentiment) or **Sub**jectivtiy value
- **mean/min/max** indicates the type of aggregated value for the respective day

	Date	D0-R-TB-PA-Pol-mean	D0-R-TB-PA-Pol-min	D0-R-TB-PA-Pol-max	D0-R-TB-PA-Sub-mean	D0-R-TB-PA-Sub-max	D0-R-TB-NB-Pol-mean	D0-R-TB-NB-Pol-min	D0-R-TB-NB-Pol-max	D1-R PA-m
0	2016-07-01	0.008778	-0.5	0.500	0.195556	0.90000	0.375069	-0.857560	0.999968	0.022
1	2016-06-30	0.022841	-0.4	0.500	0.190421	0.70000	0.393066	-0.868612	0.997930	0.038
2	2016-06-29	0.038622	-0.7	0.825	0.220205	0.90625	0.555236	-0.833831	0.995979	0.035
3	2016-06-28	0.035911	-1.0	0.525	0.289912	1.00000	0.228905	-0.983560	0.992596	0.003
4	2016-06-27	0.003085	-0.7	0.800	0.326705	1.00000	0.442804	-0.935693	0.993541	0.083

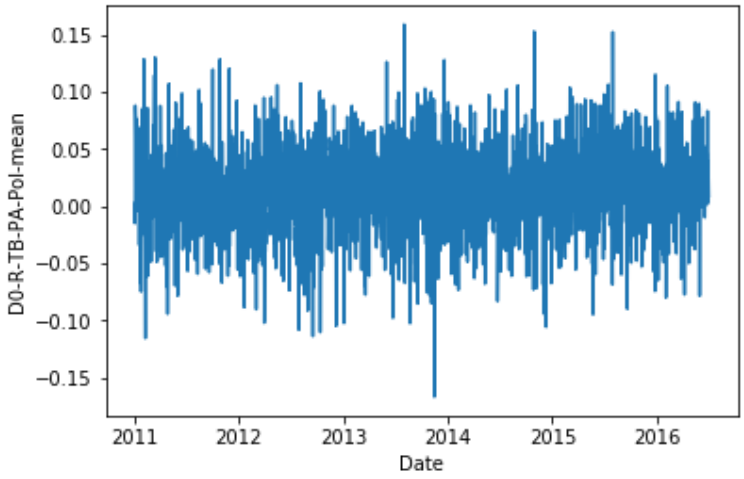
So, for each day back of `daysback`, `8*daysback` columns are generated dynamically. The following is an excerpt of basic statistics for such a resulting datasets:

	D0-R-TB-PA-Pol-mean	D0-R-TB-PA-Pol-min	D0-R-TB-PA-Pol-max	D0-R-TB-PA-Sub-mean	D0-R-TB-PA-Sub-max	D0-R-TB-NB-Pol-mean	D0-R-TB-NB-Pol-min	D0-R-TB-NB-Pol-max	D1-R P
count	2008.000000	2008.000000	2008.000000	2008.000000	2008.000000	2008.000000	2008.000000	2008.000000	2008
mean	0.013777	-0.470423	0.473275	0.244480	0.886088	0.454994	-0.764067	0.995762	0
std	0.039723	0.240676	0.172898	0.058686	0.148565	0.117037	0.195653	0.008068	0
min	-0.166995	-1.000000	0.025000	0.080583	0.416667	0.042316	-0.999947	0.914564	-0
25%	-0.012199	-0.600000	0.357143	0.203798	0.754545	0.377303	-0.912252	0.995613	-0
50%	0.013534	-0.410417	0.500000	0.242702	1.000000	0.459280	-0.813964	0.998624	0
75%	0.040650	-0.300000	0.500000	0.283874	1.000000	0.536621	-0.670013	0.999701	0
max	0.158838	-0.033333	1.000000	0.445943	1.000000	0.778778	0.086880	1.000000	0

These basic statistics as well as the following visualization of one of the sentiment values by date shows that overall, sentiments tend to be rather „positive“ - indicated by the positive mean values of the daily-aggregated sentiment values.

```
sns.lineplot('Date','D0-R-TB-PA-Pol-mean',data=sentiment_reddit)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8d5894e80>
```



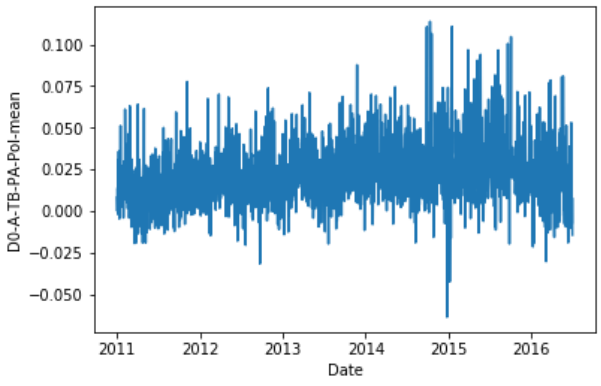
Dataset 3: ABC news headline data

The data prepocessing for the ABC news headlines follows exactly the same method as the one described above for the Reddit news headline dataset. Therefore, here only the basic statistics and example vizualisation of the resulting dataset is shown:

	D0-A-TB-PA-Pol-mean	D0-A-TB-PA-Pol-min	D0-A-TB-PA-Pol-max	D0-A-TB-PA-Sub-mean	D0-A-TB-PA-Sub-max	D0-A-TB-NB-Pol-mean	D0-A-TB-NB-Pol-min	D0-A-TB-NB-Pol-max	D1-A
count	2009.000000	2009.000000	2009.000000	2009.000000	2009.000000	2009.000000	2009.000000	2009.000000	2008
mean	0.021631	-0.745688	0.833809	0.148831	0.995860	0.235498	-0.923975	0.982973	
std	0.018282	0.198893	0.135127	0.028587	0.033317	0.047844	0.059171	0.018025	
min	-0.063717	-1.000000	0.143750	0.054744	0.500000	0.054013	-0.998686	0.720558	-(
25%	0.010061	-1.000000	0.800000	0.130107	1.000000	0.204642	-0.963779	0.978984	
50%	0.019917	-0.750000	0.800000	0.145654	1.000000	0.233923	-0.938664	0.988162	
75%	0.030206	-0.600000	1.000000	0.164218	1.000000	0.266300	-0.900563	0.993741	
max	0.113739	-0.150000	1.000000	0.287512	1.000000	0.428529	-0.395810	0.999930	

```
sns.lineplot('Date','D0-A-TB-PA-Pol-mean',data=sentiment_abc)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8d38a9c88>
```



Dataset 4: Market Volatility Forecast data

The dataset contains multiple values per day – therefore the same aggregation with mean values as for the news headline datasets is done. Because the min and max values are always -0.75 / -0.25 respectively, it does not make sense to include them in the resulting dataset for the input features.

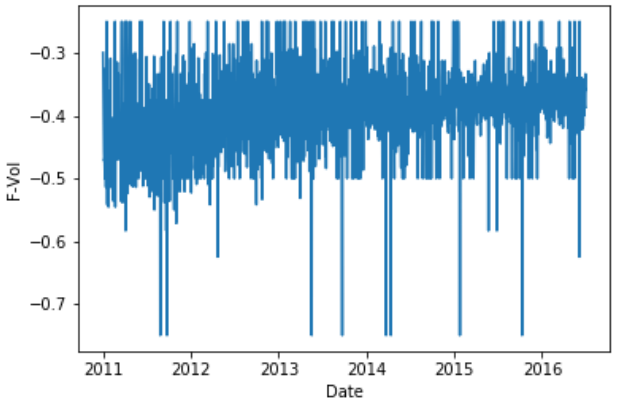
The resulting DataFrame looks like this, and the visualisation over time shows a similar view as for the news headlines, with the exception that there are days when the defined min and max values are occurring directly. An investigation into the dataset shows that there are days were only one input value for the expected market volatility occurs, which results in mean=min=max for these days.

	Date	F-Vol
0	2011-01-01	-0.299685
1	2011-01-03	-0.388889
2	2011-01-04	-0.441176
3	2011-01-05	-0.455882
4	2011-01-06	-0.472222

	F-Vol
count	1789.000000
mean	-0.394516
std	0.064378
min	-0.750000
25%	-0.428571
50%	-0.392045
75%	-0.359649
max	-0.250000

```
sns.lineplot('Date','F-Vol',data=sentiment_forex)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb8d3
```



3. Data Merge

After the sentiment analyses as outlined until now, that four datasets are merged on Date. Naturally, this reduces the dataset to fewer rows – only those of working days were DJIA was open and values are available.

Depending on the given daysback parameter, the resulting dataset may contain many input feature columns, example for `daysback=14` shown below.

```
full_data.head()
```

Out[60]:

	Date	Future14DaysAheadCat	D0-R-TB-PA-Pol-mean	D0-R-TB-PA-Pol-min	D0-R-TB-PA-Pol-max	D0-R-TB-PA-Sub-mean	D0-R-TB-PA-Sub-max	D0-R-TB-NB-Pol-mean	D0-R-TB-NB-Pol-min	D0-R-TB-NB-Pol-max	...	D5-A-TB-NB-Pol-max	D6-A-TB-PA-Pol-mean	D6-A-TB-PA-Pol-min	D6-A-TB-PA-Pol-max	A
0	2011-01-03	high_increase	0.087892	-0.750	0.8	0.381847	1.0000	0.661600	-0.486659	0.999925	...	0.000000	NaN	0.0	0.0	
1	2011-01-04	medium_increase	0.079411	-0.300	1.0	0.262418	1.0000	0.284726	-0.905987	0.999678	...	0.000000	NaN	0.0	0.0	
2	2011-01-05	medium_increase	0.060790	-0.275	0.4	0.340611	0.8000	0.474861	-0.848085	0.999284	...	0.000000	NaN	0.0	0.0	
3	2011-01-06	medium_increase	-0.004385	-0.400	0.5	0.228958	0.9375	0.471216	-0.789812	0.995958	...	0.928516	NaN	0.0	0.0	
4	2011-01-07	medium_increase	0.076277	-0.500	0.5	0.322694	1.0000	0.500884	-0.925325	0.999645	...	0.985030	0.008112	-0.8	0.8	0.1

5 rows × 115 columns

4. Data Preprocessing for merged full dataset

Rows containing NaN entries at the top and bottom of the dataset have to be discarded in order for the prediction algorithms to work. These NaN values occur naturally due to the fact that days-back sentiment analysis cannot be performed at the beginning of the dataset due to missing data before the dataset start date (2011-01-01), and DJIA performance prediction cannot be performed due to missing DJIA performance data after the cutoff date (2016-06-30).

The basic statistics of the sentiment analysis results show that different value ranges are resulting from the Pattern and NaiveBayes analyzers as well as from the simple algorithm for the market volatility forecast. This is all now scaled into range 0..1 using the MinMaxScaler.

Finally, the unnecessary columns are dropped to produce the final datasets (`input_features…` and `target_features...`) to be consumed by the DJIA performance prediction algorithm.

Implementation with iterative refinement

The implementation is performed in a set of two Jupyter Notebook files which were improved with each iteration of my project (see chapter „Refinement“ for details. The final, „best“ implementation of the project is contained in the Iteration 3 Notebooks:

- `01 Capstone Project Eidolon – Iteration 3 Data Preparation and Exploration.ipynb`
- `01 Capstone Project Eidolon – Iteration 3 Modeling and Evaluation.ipynb`

The following table shows a summary of the improvements (refinements) per iteration

Iteration	Data Processing & Exploration	Modeling & Evaluation
1	Initial implementation of data preprocessing pipeline hardcoded to „1 day back“, „1 day ahead“ timeframe, using only mean sentiment values.	Initial implementation of SciKit-Learn multiclass capable classifiers implementing model_selection.cross_val_score and analyzing the resulting ranking of classifiers using boxplots. Initial implementation of GridSearchCV for hyperparameter tuning of many non-MLP classifiers.
2	Include not only mean sentiment values, but also min and max values as input features	Included feature importance investigation. Increased hyperparameter space tuning for more non-MLP classifiers.
3	Rewrote code to be completely parametric for number of days back/ahead to experiment with more timeframes; rewrote code so that categorization of DJIA performance values is dynamic based on stdev of resulting performance distribution.	Improve performance with MLPClassifier – increased hyperparameter space for GridSearchCV. Experiment with different daysback/daysahead timeframes to identify best possible performance.

The basic implementation of Data Processing & Exploration has been described in the previous chapters and is reflected in the respective codebase of the first Jupyter notebook (...Data Preparation and Exploration)

The basis implementation of Modeling & Evaluation is the following, and is reflected in the respective codebase of the second Jupyter notebook (...Modeling and Evaluation):

Some part of the project was not automated (implemented), instead it was done manually – see below for rationale.

Automated part:

- Load the input and target feature datasets produced by notebook 1 (...Data Preparation and Exploration)
- Perform a full model evaluation, but only with standard (default) model parameters of SciKit-Learn models to see which models are well suited (worthwhile to pursue to tune) and which are not well suited (not worthwhile to pursue to tune):
    - for this, SciKit-Learns model_selection.cross_val_score is used
    - for details, see below in chapter „Refinement“ - Iteration 1 details ff.
- Perform hyperparameter tuning with a subset of all suitable models
    - for this, SciKit-Learns „GridSearchCV“ is used.
    - For details, see below in chapter „Refinement“ - Iteration 1 details ff.

Manual part:

- Execute the notebooks many times with different combinations of timeframes for input and target feature (`daysback` and `daysahead` global parameters).
- *Rationale for non-automation*: Due to the curse of dimensionality and resulting explosion in required computation time, it was not feasible to completely automate this process into an automated loop to find the optimal prediction model within given ranges of those parameters. Therefore I used a combination of different timeframes manually. Especially when considering many days back (e.g. a whole month), the number of input features explodes, thus exploding model training times.

Refinement

As outlined in the previous chapter, my project went through three major iterations.

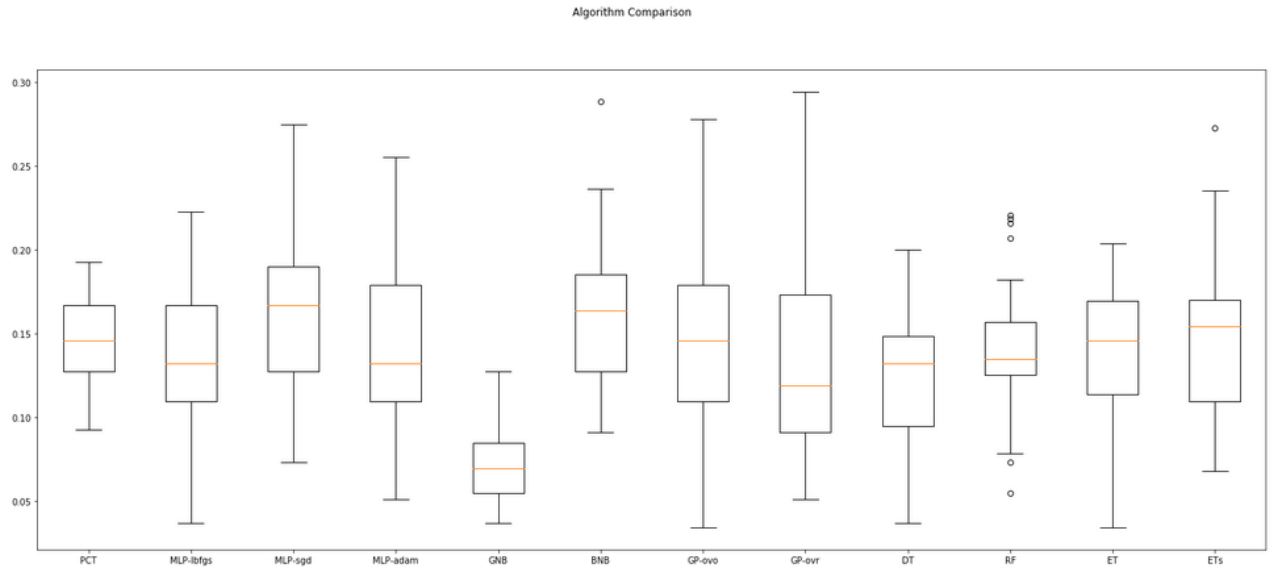
Iteration 1 details (initial code and refinements within iteration 1)

Within iteration 1, I had my data preprocessing pipeline hardcoded to a timeframe. Being restricted to only one timeframe (1 day back, 1 day ahead), I focused my work on getting the full model selection code working and perform a feature importance analysis. In detail, the implementation is:

1. Define the list of models to be evaluated with model_selection.cross_val_score. Set up a StrafiedKFold object, because the target variable is not evenly categorized, as can be seen from the basic statistics (category occurence count) in the previous chapters.

2. Loop through the model list and evaluate each model. For each model, capture the accuracy score and – when available – feature importances for the model.

3. Plot and analyse the resulting „model hitlist“. Example:




```
In [20]: pdresults.sort_values('mean',ascending=False).head()
```

Out[20]:

	Algo	mean	stdev
22	LR-ovr	0.182366	0.042750
5	GP-ovr	0.181075	0.045962
21	LR-i	0.180138	0.040044

3. For the top models from the general evaluation, perform hyperparameter tuning using GridSearchCV. Regardless of the top list, always include the MultiLayerPerceptron Classifier as well in order to constantly see if it may be worthwhile to follow a Keras/TensorFlow implementation approach as well.

Analyse results of GridSearchCV and compare results of „best“ (optimized) classifier with the accuracy result of the non-optimized (default-parameters) classifier. Example:

```
In [31]: pdgridresults.sort_values(by='Acc-optimized',ascending=False)
```

Out[31]:

	Algo	Acc-raw	Acc-optimized
4	MLP-sgd	0.164740	0.176301
0	LR-ovr-saga	0.153179	0.153179
2	LR-ovr-others	0.153179	0.153179

In iteration 1, I also experimented with leaving out entire input datasets (Reddit, ABC or Market Volatility). But regardless of combination, the accuracy results for the top-performing algorithms stayed in the same range. Summarized, This first iteration produced results in the accuracy range of 0.16-0.18. This is better than guessing, but not by much, but left me dissatisfied, therefore I went forward with iteration 2 to improve.

Iteration 2 details

This iteration increased the input feature space by not only including the mean sentiment values of each day, but also the min and max values. My thinking behind this was that sometimes, one really negative headline (or really positive headline) may drive the markets more significantly than all news headlines for that day combined (i.e. mean value).

In order to evaluate my thinking, I added a feature importance investigation in the general model evaluation loop, which allowed me to analyse the feature importance aggregate over all model evaluations in the main loop. Example results are:

```
pdresults_fi = pd.DataFrame(feature_importances,columns=pdresults_fi_columns)
```

```
with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be spec
    display(pdresults_fi)
```

	Algo	D0-R-TB-PA-Pol-mean	D0-R-TB-PA-Pol-min	D0-R-TB-PA-Pol-max	D0-R-TB-PA-Sub-mean	D0-R-TB-PA-Sub-max	D0-R-TB-NB-Pol-mean	D0-R-TB-NB-Pol-min	D0-R-TB-NB-Pol-max	D1-R-TB-PA-Pol-mean	D1-R-TB-PA-Pol-min	D1-R-TB-PA-Pol-max	D1-R-TB-PA-Sub-mean	D1-R-TB-PA-Sub-max	D1-R-TB-NB-Pol-mean	D1-R-TB-NB-Pol-min	D1-R-TB-NB-Pol-max	D2-R-TB-PA-Pol-mean	D2-R-TB-PA-Pol-min
0	PCT	False	True	True	True	True	False	True	False	True	True	False	True	True	True	False	False	False	True
1	BNB	False	True	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	True
2	DT	True	False	False	True	False	False	True	False	True	True	False	True	False	False	True	True	False	False
3	RF	True	False	False	True	False	False	True	True	False	False	False	True	False	True	True	True	True	True
4	ET	True	True	True	False	False	False	False	True	False	True	False	False	False	True	False	False	False	False
5	ETs	False	True	True	True	False	True	False	True	True	True	True	True	False	False	False	False	True	True
6	LR-ovr	False	True	True	True	False	False	False	False	True	True	False	True	True	False	False	False	False	False
7	LRC-	False	True	True	True	True	False	False	False	False	True	True	False	True	False	False	False	False	True

```
with pd.option_context('display.max_rows', None, 'display.max_columns
    display(pdresults_fi.sum())
```

```
Algo                    PCTBNBDTRFETETsLR-ovrLRC-ovrLDARCRCCGBCSGDPAC
D0-R-TB-PA-Pol-mean                                                 5
D0-R-TB-PA-Pol-min                                                  9
D0-R-TB-PA-Pol-max                                                  8
D0-R-TB-PA-Sub-mean                                                 9
D0-R-TB-PA-Sub-max                                                  4
D0-R-TB-NB-Pol-mean                                                 4
D0-R-TB-NB-Pol-min                                                  6
D0-R-TB-NB-Pol-max                                                  6
D1-R-TB-PA-Pol-mean                                                 9
D1-R-TB-PA-Pol-min                                                 10
D1-R-TB-PA-Pol-max                                                  3
D1-R-TB-PA-Sub-mean                                                10
D1-R-TB-PA-Sub-max                                                  6
D1-R-TB-NB-Pol-mean                                                 6
D1-R-TB-NB-Pol-min                                                  2
D1-R-TB-NB-Pol-max                                                  5
D2-R-TB-PA-Pol-mean                                                 2
D2-R-TB-PA-Pol-min                                                  6
```

The value of this feature important analysis is that it can be used to consider dropping those features / feature types which have low sums, meaning which have proven to be unimportant in many classifiers.

This feature importance investigation showed that two newly added input features seemed to be utterly unimportant – therefore, for iteration 3, I removed them again from the input feature space.

From a prediction performance point of view, the iteration was disappointing, all „top“ model performances stayed in the same range of 0.16-0.18.

Iteration 3 details

As finding from iteration 2, the ...PA-Sub-min input feature was dropped entirely.

In this iteration, the whole data preprocessing codebase was rewritten to accept global parameters `daysback` and `daysahead` to become flexible with the input feature and target feature timeframes, which were hardcoded to daysback=1 and daysahead=1 in the previous iterations.

After this, the main effort in iteration 3 was to – manually – experiment with many combinations of these parameters. I chose an approach to investigate in blocks of „x weeks“ back/ahead. With this approach, I came to my final accuracy score as a conclusion for this project as reported in the following „Results“ chapter.

IV. Results

Model Evaluation and Validation

This chapter presents model evaluation and validation from Iteration 3 (final iteration). Intermediate results of model evaluation and validation are discussed in the previous chapter in the details of each previous iteration.

After experimenting with many input and target feature timeframes in iteration 3, the highest predication accuracy I could achieve was the following: **0.274 (27,4%)**

Input feature timeframe: 28 days (1 month) back

Target feature timeframe: 56 days (2 months) ahead

The „winning model“ is a Support Vector Classifier – method „one-vs-one“ or „one-vs-rest“ did not make a difference. Runner-up is a Multi-Layer Perceptron using sgd solver.

	Algo	Acc-default-params	Acc-optimized-params
0	SV-ovo	0.256098	0.274390
1	SV-ovr	0.256098	0.274390
3	MLP-sgd	0.253049	0.259146

```
GridSearchCV.best_estimator_ =
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovo', degree=3, gamma='scale', kernel='poly',
  max_iter=-1, probability=False, random_state=47, shrinking=True,
  tol=0.001, verbose=False)

SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
  max_iter=-1, probability=False, random_state=47, shrinking=True,
  tol=0.001, verbose=False)
```

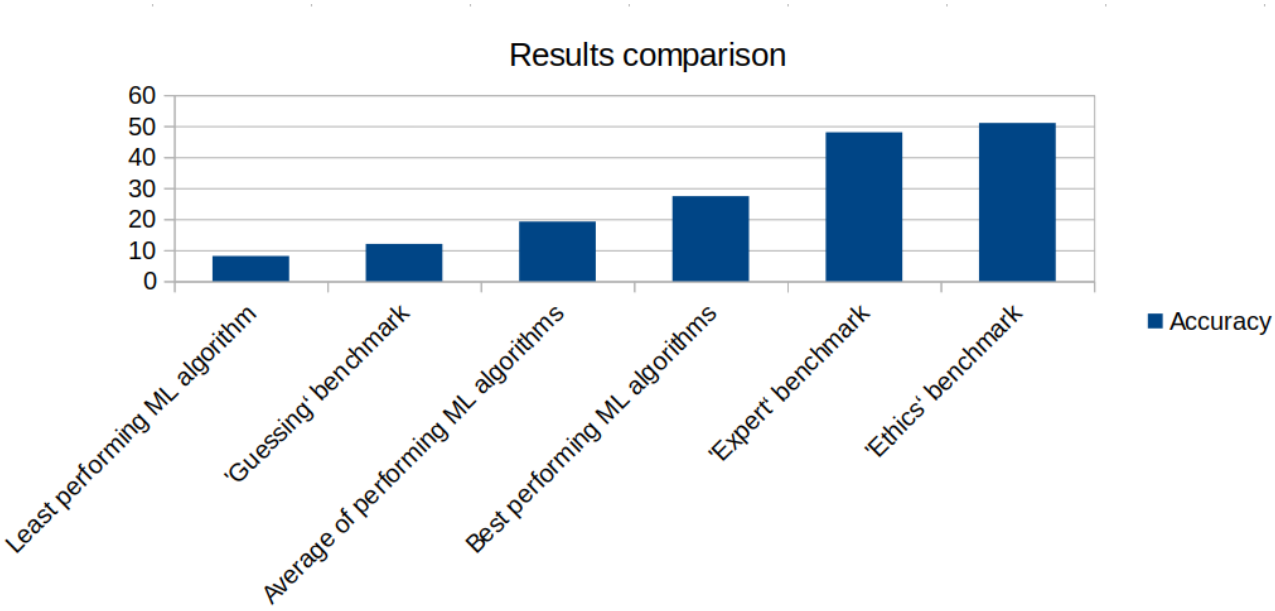
Unfortunately, for a model sensitivity analysis, I do not have any data from other timeframes outside the model training timeframe to work with.

Justification

The project results are not significant enough to have solved the problem posed in the project. Even though the achieved DJIA performance prediction accuracy (27.4%) is better than just *guessing* the future DJIA performance (with eight performance categories, this would be 100/8 = 12.5% accuracy), the achieved accuracy is definitively not enough to actually help ETF investors in their everyday investment decisions.

V. Conclusion

Free-Form Visualization



The above figure shows the minimum, mean and maximum accuracy results of my capstone project compared with the set benchmarks, plus the „ethics“ benchmark.

It is interesting to note how important it is to pick the right machine learning algorithm and to tune the hyperparameters correctly for optimization of the algorithm performance. The accuracy ranges from 8,1% (least performing, not optimized) to 27,4% (best performing, optimized) with an average of 19,2%.

The most important quality of my project is that I could indeed outperform the „guessing“ benchmark of 12,5%, meaning my resulting model is twice as good as just guessing the evolution of the DJIA. But even this „important“ quality is useless in reality.

Most significant is the fact that a „simple“ machine learning pipeline like this one has by far worse prediction accuracy than expert market analysts (‚Expert‘ benchmark at 48%), and clearly fails to raise any ethical red flags (‚Ethics‘ benchmark at 51%).

Reflection

The aim of the project was to predict the future performance of the DJIA stock market index with as high an accuracy as possible, based on text sentiment analysis („mood“ of texts) of publicly available news headlines and market volatility forecasts. The problem to be solved with this are the

investment decisions of stock market investors investing in DJIA-based ETFs, which are tied to the DJIA performance. To tackle this problem, suitable machine-readable datasets of DJIA performance, news headlines and market volatility forecasts have been processed and transformed into a format required to apply machine learning techniques.

Evaluating a multitude of machine learning algorithms and data combinations to improve accuracy, three main ones – Logistic Regression, Support Vector Machines and Multi-Layer Perceptrons (MLP) – turned out to be the best ones, with the SVM clearly generating the „best“ results with around 27,4% prediction accuracy. Getting to achieve this „high“ accuracy was the most interesting part of the project, as it was counterintuitive to the perceived reality of the prediction task. Perceived reality is that predictions are better the nearer the future they address is. But predicting the DJIA performance for 2 months into the future was more accurate than predicting the DJIA performance for only 1 day ahead; this was quite a surprise during the project.

However, even this „best“ result of 27,4% is not useful to address the problem an ETF investor has – which is „outperforming the market“. This would only be possible in the case where prediction accuracy consistenly is in the >>50% domain.

Therefore, the model should **not** be used to support investment decisions.

Then again, this outcome of my project fits to my expectation for the problem – an „unsolvable“ one. In case there really was a model to consistently achieve >>50% accuracy, the ethical problems outlined in the introductory chapter would arise.

Come to think of it: If I had *really* solved this problem – i.e. achieving a prediction accuracy of >>50% - maybe I would not have written this project report, but instead keep the model for myself and get rich with it? Or would I have reported the model to the authorities? Who knows… ;-)

Improvement

There are a lot of possibilities for improvements which could not be pursued due to the restricted timeline and effort budget available for this project. These are:

- Use larger timescale input data. Project was restricted to timeframe 2011/01/01-2016/0630, only 5,5 years.
- Use different sentiment analysis algorithms. Project used TextBlob only. Alternatives are using Stanford‘s CoreNLP package or even creating an own deep neural network.
- Use different DJIA performance prediction models and technologies. Project only used scikit-learn as technologies. Other technologies (e.g. Keras/TensorFlow, PyTorch or even commercial ones such as H2O.AI) could not be pursued
- Use better input data for sentiment analysis. The current text corpora do contain the full scope of global news, including trivia such as entertainment and sports. It is possible that the prediction accuracy can be increased by discarding such kind of trivial news headlines and focus on stock market driving news such as economical, financial, political news
- Experiment with more prediction timescale combinations. Due to the computing effort required for each combination, only a few combinations could be tried, e.g. 1back/1head – 3back/1ahead – 7back/7ahead etc. Maybe there are combinations to be found which increase the prediction accuracy.