Authors Note

Please excuse the formatting of this thesis. A requirement was to make this thesis in overleaf, the academic standard for white papers. Due to this some of the figure/table formatting and positioning may not be ideal.

Regards,

Fabio Greenwood

+44 787 040 1486

Fabio.Greenwood@gmail.com

Social Media Sentiment Analysis for Stock Market Automated Trading

Fabio Greenwood

June 2024

Contents

1 Summary Report

1.1 Introduction

In recent years, the landscape of stock trading has evolved significantly, with a marked shift from traditional manual strategies to automated trading systems. This thesis explores the possible integration of sentiment analysis and topic clustering into automated stock trading, aiming to enhance the prediction accuracy of stock price movements. The non-typical approach adopted in this research combines traditional statistical models with insights derived from social media and news sentiment, particularly from sources like Twitter.

Historically, day traders have relied on their understanding of market dynamics and personal judgment of market sentiment, using patterns in stock price movements to make profitable trades. Experienced traders would design "trading rules" based on their understanding, which they would manually follow. Over time, these manually enforced rules have increasingly been replaced with automated bots. Today, it is estimated that at least 50% of trades are executed by these automated systems. Given the high potential for profit for anyone who can successfully develop these automated processes, there is always a large amount of research trying to incorporate novel methods to improve models in this domain and some of this research is in integrating sentiment analysis to these trading models.

1.2 Primary Business Question (Objectives)

The primary objective of this thesis is to benchmark a statistical model's potential improvement in its profitability for stock market prediction after elements of sentiment analysis and topic clustering are added to the model. To this end, the study will attempt to answer the following questions:

1. Is there an improvement in stock prediction with the addition of linear sentiment analysis as a training feature?

2. Is there an improvement in stock prediction with the addition of multiple training features tracking the sentiment across multiple topics?

Below is a summary of the contents of each section of the larger report.

1.3 Literature Review and General Domain Information

While not detailed in this summary, the literature review (section 4) provides an overview of relevant literature reviewed in the preparation of this project while highlighting areas that have influenced this study. It will also give the reader a general understanding of the overall state of the art for the relevant technical domains.

1.4 System Design

The functional aim of the system is to achieve these big three challenges

- Train and evaluate multiple instances of a stock market prediction model
- Tune the hyper-parameters of new models based on the performance of previous models (optimisation)
- Evaluate any potential improvement from adding sentiment analysis and topic clustering

To achieve this there is an overarching module called the "experiment manager". This module initiates the creation of new models by requesting the generation of various bits of metadata from other component modules to train and evaluate an instance of the stock market predictor. It then logs the validation and testing results of the models produced and in a process described later (section 5.2), uses the validation scores to inform the hyper-parameters of subsequent instances of the stock market predictor. Please note the following distinction. The "system" refers to all the components as a whole. A "model" or "design" refers to an instance of the statistical predictor or its internal design.

The key sub-components manipulated by the experiment manager are:

1. Finance Data Preparation
   (a) This section takes standard OHLCV data[1] purchased from firstratedata.com[2] and then calculates various market indicators[3] such as: sma, ema, macd and bollinger bands

[1] Open, High, Low, Close, Volume data, see section 5.3 for more details
[2] see reference [1]
[3] advanced indicators used in understanding a financial market's underlying conditions

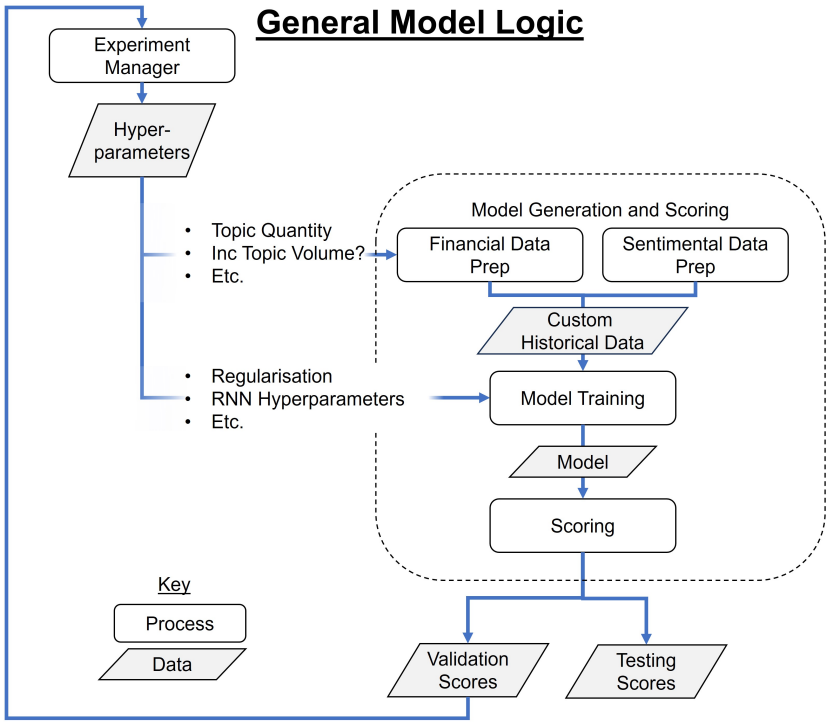

Figure 1: General system logic. Experiment manager issues commands to the other modules to create new instances of the stock market predictor, then based on the validation scores of previous models, tunes the parameters of subsequent models

2. Custom Sentiment Data Preparation

   (a) Tweets concerning Apple were data sourced from kaggle.com[4]

   (b) Each of these tweets were then assigned weightings across X topics (depending on the number of topics) using the python package gensim and an overall sentiment score (happy/sad) based on the python package vaderSentiment's assessment of the mood of the tweet's body of text

   (c) The above-generated sentiment score and topic weighting data are then combined with each tweet's timestamp to produce a continuous sentiment score across each topic cluster over each time step of the analysis

   (d) Optionally a normalised score is also produced for the relative volume of talk for each topic for each time-step in the study

   (e) See section 5.4 for model information

3. Ensemble Statistical-Learning Model

   (a) This is a recurring neural network (RNN) that takes the data prepared during the previous two steps

   (b) The exact design and hyper-parameters of each instance of this model are controlled by the "experiment manager"

   (c) The RNN is further enhanced with the following features: random subspace, ensemble prediction, K-folds validation

   (d) see section 5.5 for more information.

4. Experiment Manager

   (a) controls the above process and logs their results, discussed above and in section 5.2

Like all statistical models, the overall performance is affected by hyper-parameters. The experiment manager will use parametric optimisation to optimise these hyper-parameters faster than traditional methods like sklearn.model_selection.RandomizedSearchCV and sklearn.model_selection.GridSearchCV.

[4] see reference [2]

To achieve this, it will run a set of random designs with random hyper-parameter combinations. Then using the validation results of these experiments, it will select new combinations of hyper-parameters, using statistics, to attempt to maximise performance. This process is called "parametric optimisation" This has the time advantage over GridSearchCV as not all combinations have to be attempted.

All the components are also designed to reuse assets, particularly the assets produced in the calculation of sentiment data and the predictors themselves, which are computationally expensive to produce. These assets are stored for potential reuse in future model runs (details in appendix 5.1.1), the financial data preparation only needs to be completed once.

1.5 Analysis Process Overview

Once the system was completely developed, these are the major steps in the production of results:

1. Complete a design of experiments of around 20 designs based on random configurations of the selected hyperparameters (these being model hyperparameters selected to vary for the analysis)

2. They use GuRobi to design new hyperparameters based on the previous results to try and maximise one of 2 performance measures. This will consider the validation results and not the testing results.

    - These measures are the validation MAE and a custom performance/profitability function[5]
    - This process is repeated 14 times

3. Select a set of high scoring designs based the models with the highest validation scores across the each prediction horizon and each model scope

    - prediction horizon: 5 or 25 mins
    - model scope: sentiment with topic clustering, sentiment without topics or no sentiment

4. Compare and analyse the validation and testing scores across the optimum designs for each of the 6 combinations for prediction horizon and model scope

[5]equaling the average profit made per euro staked, discussed in 6.3

(a) These means that each "optimal" design selected will be repeated across each model scope, so a comparison of the performance for equal designs across each scope can be made

1.6 Limitations

Between the model development and the above implementation stage, exploration was done to experiment with the design to improve performance. This process found the following design changes improved model performance and therefore were implemented into the model:

- Changing the model to use RNNs instead of NNs
  - Recurrent Neural Networks (RNN) are Neural Networks (NN) which consider multiple time-steps for calculating a prediction[6]
- Increasing the keras parameter of allowable loop-back
  - The number of time-steps an RNN is simultaneously trained on
- Finally additional data was purchased to increase the time period studied in the project

As each of these changes were implemented to improve predictive performance they incrementally increased the calculation time of the model from 0.1 - 1 hour to 1.25 - 4 hours (depending on conditions) and consumed calendar time in development work.

Because of this the scope of the analysis was reduced to what was described in the previous section. There were reductions in:

- The number of models instances created and the model of models designed according to the parametric optimisation process
- The number of model hyper-parameters available to be adjusted by the experiment manager
- Limiting the data resolution to 5 min increments, instead of also doing studies for 1 min time steps as well

[6] please see section 5.5.1 for more information

- The outer bounds of certain hyper-parameters that were strongly related to calculation time such as RNN look-backs, number of parallel models, number of k-folds etc

1.7 Model Hyper-Parameters

The hyper-parameters that were allowed to be edited for the parametric optimisation and their ranges are listed in section 6.2

1.8 Results and Discussion

1.8.1 Comparing Model Scopes

To answer the questions posed in section 1.2:

1. Is there an improvement in stock prediction with the addition of linear sentiment analysis as a training feature?

2. Is there an improvement in stock prediction with the addition of multiple training features tracking the sentiment across multiple topics?

A selection of the highest performing models from each model scope (e.g. topics, no-sentiment etc) were selected and rerun multiple times, in their own scope and other scopes with the same prediction horizon. A paired t-test was then run to understand if there was a noticeable uptick in performance between models with different scope.

For the high performing designs from the “no-topics” or “no-sentiment” scopes were selected, the missing parameters which concern sentiment and topics, are taken from the high performing ”topics” scope designs.

All the above process is followed to produce a set of equal runs to compare performance, while ensuring that the A/B is performed for the higher performing designs produced.

The hyper-parameter ”Sentiment - Factor Topic Volume” was also disabled (and removed from this table) as this could introduce data leakage as the number of likes/comments/retweets would happen after the postdate (time of tweeting), meaning allowing the model to see this, would introduce foresight into the model.

The metrics which were compared were % Correct Direction, Profitability Score and MAE error (considering both their validation and testing scores). Tables 1 and 2 shows the t-stats and p-value for the comparison between “multi-topic” and “no-sentiment” and tables 3 and 4 show the

High Performance Designs Paired T-Test					
Multi-Topic vs. No-Sentiment					
5 Mins - 0.9 Confidence					
Performance Metric	observations	multi-topic average	no-sentiment average	t-statistic	p-value
Validation % Correct Direction	24	52.4%	52.9%	-2.01	0.06
Testing % Correct Direction	24	53.5%	53.8%	-1.32	0.20
Validation Profitability	24	1.36E-03	1.40E-03	-1.44	0.16
Testing Profitability	24	4.03E-04	3.91E-04	1.10	0.28
Validation MAE	24	9.24E-04	9.05E-04	1.34	0.19
Testing MAE	24	7.98E-04	7.94E-04	0.60	0.55

Table 1: Paired T-Test comparing the 6 top designs many selected from the multi-topic optimisation for the 5 min context, re-run twice and compared in a paired t-test. The original values that marked these values as high performing were not used as they may have been partly from lucky variation.

comparison of “no-topics” and ”no-sentiment”.

In 17/24 of the criteria measured above, the no-sentiment models outperformed either the no-topics and topics models. While only values in table 4, show statistically significant difference between groups, the continued poor performance from the models involving sentiment, is likely a sign that the technique applied didn't work.

The closest to seeing a sentiment model outperform is testing profitability for the 5-minute context between the topic and no-sentiment models (see Table 1), however there is a high p-value of 0.28. It should also be noted that this table had its observations increased from 12 to 24, which had increased the p-value in question from 0.19 to 0.28. This suggests that this out-performance is due to a random chance in the first 12 observations, which is being evened out in the later observations, not an inherent trend. Also there are 24 paired t-tests (4 tables multiplied by 6 rows), so with these large number of categories there is likely to be one group with some random outliers throwing up an anomalous composite result when working with so few observations per row (mode: 12).

Therefore currently the only conclusion to draw in relation to the business questions is that the approaching attempted to integrate sentiment analysis and topic clustering, haven't yielded an increase prediction performance in this study. Any appearance of such seems to likely be due to chance. There also seems to be cases where no-sentiment models out performed the models using sentiment.

High Performance Designs Paired T-Test					
Multi-Topic vs. No-Sentiment					
30 Mins - 0.9 Confidence					
Performance Metric	observations	multi-topic average	no-sentiment average	t-statistic	p-value
Validation % Correct Direction	24	55.4%	55.8%	-1.46	0.16
Testing % Correct Direction	24	53.2%	54.4%	-3.90	0.00
Validation Profitability	24	5.50E-03	5.61E-03	-0.48	0.64
Testing Profitability	24	6.02E-03	6.22E-03	-1.41	0.17
Validation MAE	24	2.30E-03	2.28E-03	0.82	0.42
Testing MAE	24	2.25E-03	2.19E-03	2.06	0.05

Table 2: Same method as discussed in table 1, however the top designs for the 30 mins prediction horizon were selected.

High Performance Designs Paired T-Test					
No-Topics vs. No-Sentiment					
5 Mins - 0.9 Confidence					
Performance Metric	observations	no-topic average	no-sentiment average	t-statistic	p-value
Validation % Correct Direction	24	52.3%	0.52.9%	-2.06	0.05
Testing % Correct Direction	24	53.5%	53.8%	-1.51	0.15
Validation Profitability	24	1.44E-03	1.40E-03	0.91	0.37
Testing Profitability	24	3.90E-04	3.91E-04	-0.08	0.94
Validation MAE	24	9.16E-04	9.05E-04	0.92	0.37
Testing MAE	24	7.95E-04	7.94E-04	0.10	0.92

Table 3: Same method and designs as discussed in table 1, however the designs are run as no-topics and a no-sentiment models and compared

High Performance Designs Paired T-Test					
No-Topics vs. No-Sentiment					
30 Mins - 0.9 Confidence					
Performance Metric	observations	no-topic average	no-sentiment average	t-statistic	p-value
Validation % Correct Direction	24	55.2%	55.8%	-1.74	0.09
Testing % Correct Direction	24	53.9%	54.4%	-1.63	0.12
Validation Profitability	24	5.34E-03	5.61E-03	-1.37	0.18
Testing Profitability	24	5.98E-03	6.22E-03	-1.18	0.25
Validation MAE	24	2.29E-03	2.28E-03	0.83	0.41
Testing MAE	24	2.20E-03	2.19E-03	0.24	0.81

Table 4: Same method as discussed in table 3, however the top designs for the 30 mins prediction horizon were selected.

1.8.2 MAE Error Scores Overview

Figures 2 and 3, show the validation and training MAE for each design (an iteration of the model) produced for the study. The figures show the context of predictions of 5 mins and 30 mins respectively. Figures 4 and 5 show the same results with the high MAE designs removed.

In each figure, the top left instance shows the designs for all three model scopes: "Multi-Topics" (sentiment and topics), "No-Topics" (sentiment with no-topics) and "No-Sentiment", whereas the other three subplots show the designs for just a single scope. In those three plots, the randomly selected "design of experiments" (DoE) designs and the later "optimisation" designs which were selected by the optimisation algorithm are marked differently so that the user can see progress that the optimisation applied to the model.

It should be noted that, as discussed in section 5.2, the optimisation algorithm alternated in trying to minimise MAE error or maximise profitability score, meaning that on a given figure, not all "Optimisation" points are trying to optimise the metric displayed.

All four figures show a strong correlation between validation and training MAE. While this seems positive it should be noted that the MAEs displayed are all larger than the average stock price jump in the same time period. This means that if one was to predict that the stock price would be the same after the prediction horizon has passed they would make a smaller MAE error than the model, see figure 5.

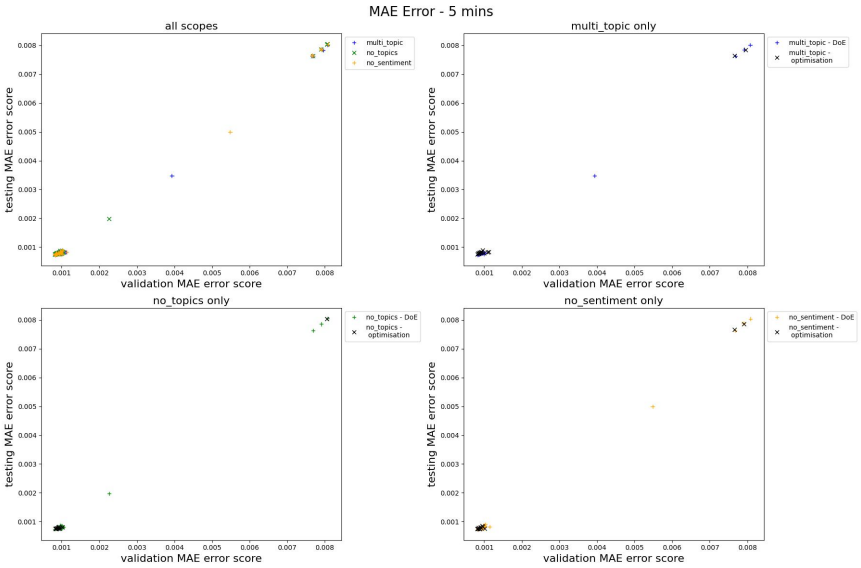

Figure 2: Validation MAE vs testing MAE for all scopes predicting for 5 mins in the future

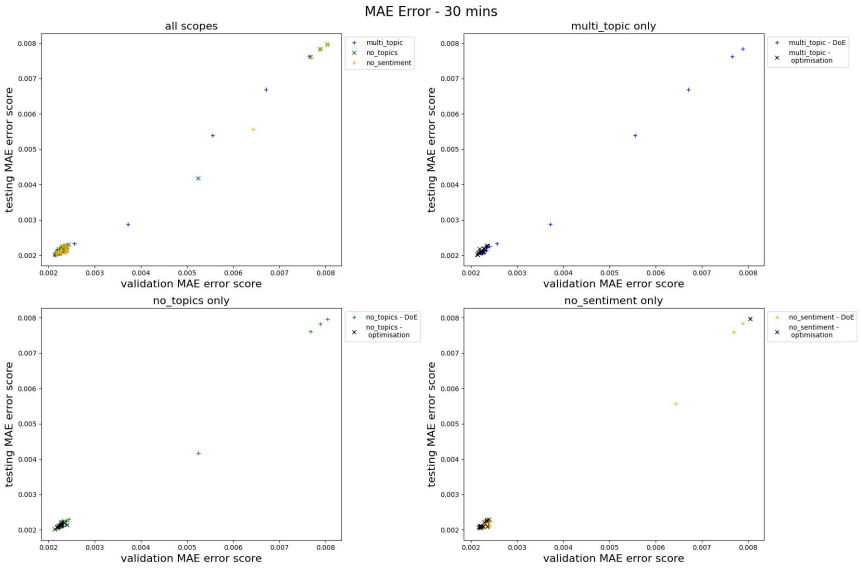

Figure 3: Validation MAE vs testing MAE for all scopes predicting for 30 mins in the future

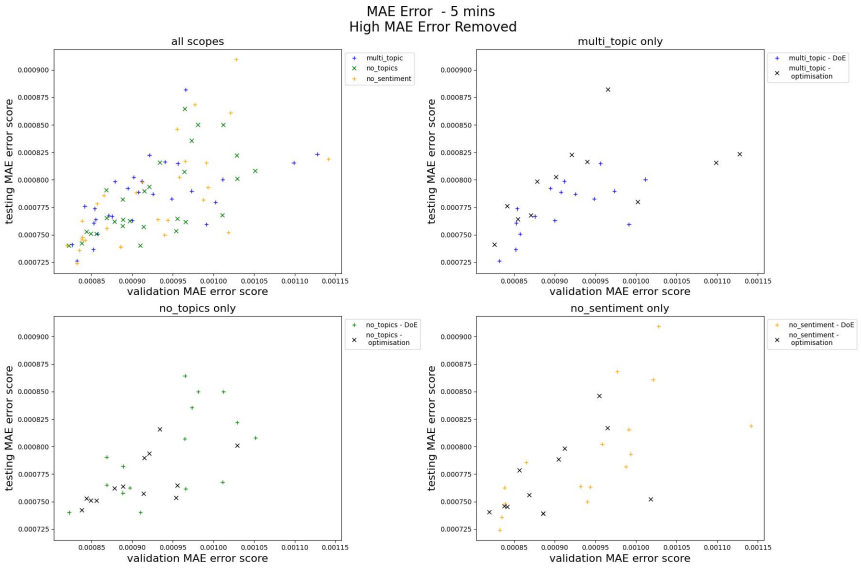

Figure 4: Validation MAE vs testing MAE for all scopes predicting for 5 mins in the future

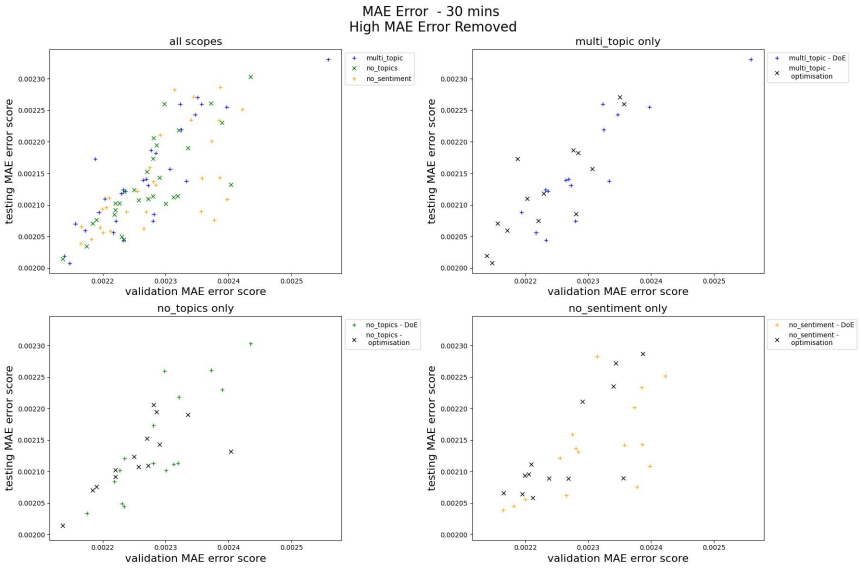

Figure 5: Validation MAE vs testing MAE for all scopes predicting for 30 mins in the future

MAE Comparision Between Best in Class Models and Average Stock Price Jump		
	5-min	30-mins
Average Stock Price Jump	7.4e-4	1.9e-4
Min MAE Recorded (Validation or Testing, all scopes)	8.0e-4	2.0e-4

Table 5: Table comparing the average jump in stock price and the minimum MAE error achieved, validation or testing, recorded across all model scopes for that prediction horizon. This table shows that the current models appear to be subject to an internal noise causing their predictions to have larger jumps then the standard stock markets natural noise/movement

The root cause for this large amount of noise, is likely caused from a combination of some of the following issues: using MSE as a loss function, underfitting/overfitting, poorly tuned regularisation, the model being too simple, early stopping criteria, learning rate. Interestingly section 1.8.10 shows preliminary results which suggest that increasing the training epochs or number of component models will have little effect on this.

Also the model was trained using MSE as a training metric, its was to attempt to create models that would avoid massive losses. This is potentially limiting the ability to minimise MAE, potentially a second round of training to minimise this could be a novel solution to reduce MAE.

These results shouldn't be used for comparing between different model scope (i.e. "multi-topic", "no-topics" etc) is not constructive yet, as the "optimisation" designs differ in parameters between scopes, therefore they can't be compared, please refer to section 1.8.1 for where high performing designs from each scope are reran in each scope for comparison.

In conclusion, this high error is not a massive issue as is discussed in sections 1.8.1,1.8.3 and 1.8.3, the other measures are positive (or ¿50 %), this means that while the predictions have a lot of noise attached, the models are able to predict the coming trends of the market, more often than not.

1.8.3 Profitability Scores Overview

Figures 6 & 7 show the validation and testing estimated profitability scores for the 5 & and 30 minute prediction horizons respectfully. Both figures show the estimated profitability scores when positions are taken 10% of the time (described as 0.9 confidence). For 30 mins there is a strong correlation between validation and testing profit, where as for the 5 minutes prediction horizon this correlation is far weaker, however the designs with a higher validation score, tend to be close or above the 50th

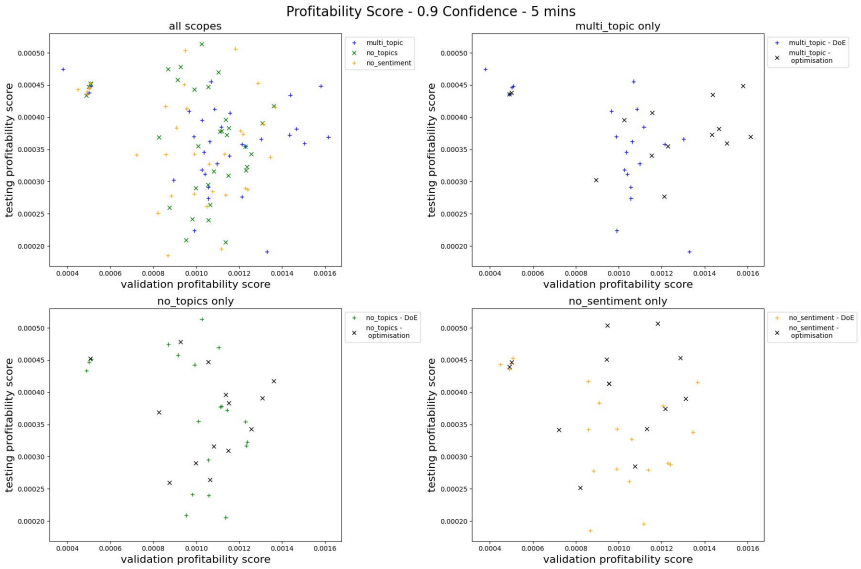

Figure 6: Validation profitability score vs testing profitability score for all scopes predicting for 5 mins in the future. Details of the calculation can be found in section 6.3

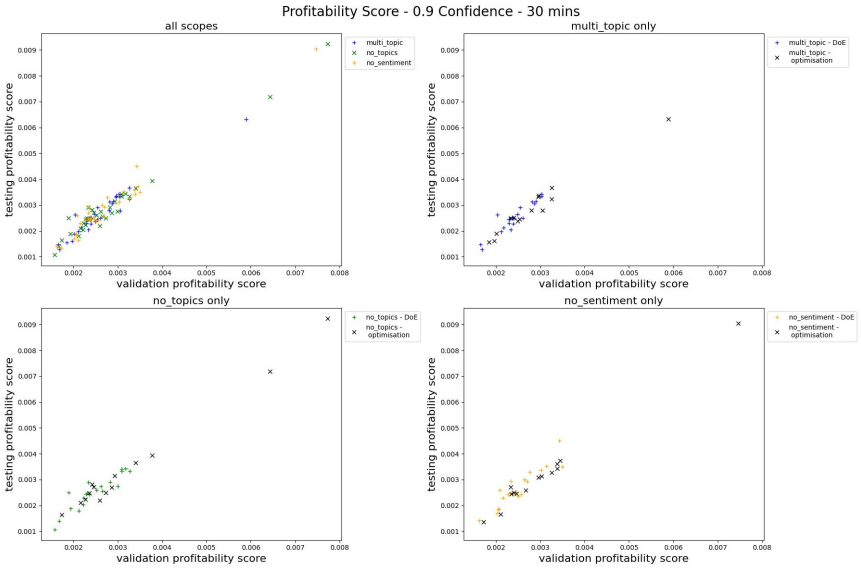

Figure 7: Similar to fig 6 only for the 30 mins prediction horizon

percentile for testing score. This suggests for the 5 minute context, a higher validation profitability score tends to ensure a mid-to-high testing score at least; however more data is needed for this assertion.

In terms of comparing performance between model scopes for the 30 min context, "no sentiment" and "no-topics" have 1 and 2 designs that massively outperform "topics" models but these could be outliers that a multi-topic model iteration could eventually also achieve, given how early in the optimisation process these designs were generated as a single, potentially lucky case as there wasn't repetition. See section 1.8.6 for more information.

For the 5 min context, the multi-topic models have 4-5 designs with a higher validation and high testing score. This doesn't mean that the multi-topic models are out-performing the other model scopes and as discussed previously in section 1.8.1.

1.8.4 % Correct Next Direction Prediction

Figures 8 and 9 show the proportion of time that the model would predict the next direction of stock movement correctly. This was for both the 5 and 30 minute prediction horizons and shows the score when predictions were recorded/taken for the 10% highest predicted price movements (described as 0.9 confidence). These results have had the cluster of high MAE results mentioned in previous sections and discussed in section 1.8.8, removed as they consistently scored low in terms of prediction score. These results can be found in appendix 7.9.

All the above charts show a statistically significant relationship between prediction accuracy for validation and testing sets. While there is a large spread across the trend-line, this still proves that models with a high validation score can be trusted to have a similar testing performance

1.8.5 Temporal Analysis on the Effects of Bias

The historical Apple stock price data used in this analysis has an upwards trend, both in the training/validation and testing datasets, see figure 10. This is important as there is the risk that the model will likely be trained to have an upwards bias, meaning that when tested on a period of upwards movement the profitable results may just be a longer-term result of the bias and not of any innate prediction ability.

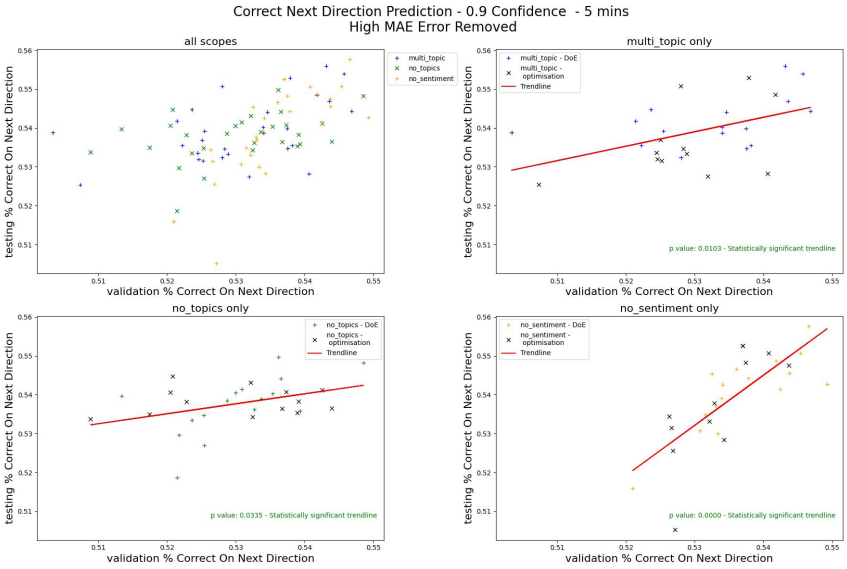

Figure 8: Shows the proportion of time that the model would predict the next direction of stock movement correctly, when predictions are recorded/taken for the 10% highest predicted price movements (described as 0.9 confidence). The validation score for this figure is on the x-axis and the testing is on the y-axis. This figure has any designs (results) with a high validation MAE error removed. Figures with these included can be found in the appendix 7.9

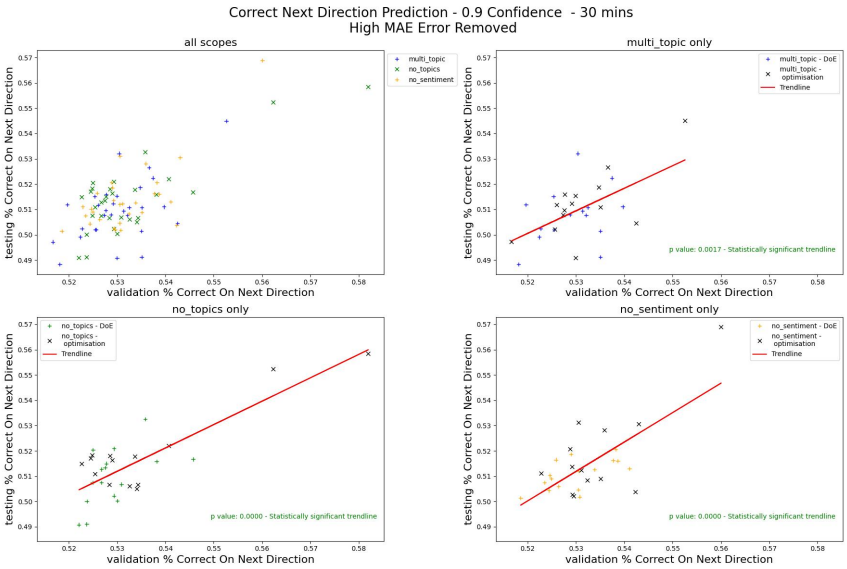

Figure 9: Similar to fig 8 only for the 30 minute prediction horizons

Figure 10: Apple stock price from 1/1/16 to 1/1/20. The model was trained/validated from 1/1/16 to 1/7/19 and tested from 1/7/19 to 1/1/20. Each grey/white background column corresponds to a 6-month period, therefore the final white background column corresponds with the final 6-month testing period, source [3]

To test the potential merit of models that have been trained on almost exclusive upwards trends, on their ability to predict stocks "successfully" in periods of downward movement, the model training process was manipulated that one or more of the validation periods (see section 5.5.3) falls completely on a period of downwards trend. The validation score for this downward period, could be compared to the other validation periods that are scored against upwards periods. A downwards period from 12/10/18 and 04/01/19 was identified as the focus of this study and is pictured in figure 11. The early stopping parameter discussed in section 1.8.9, was also set to zero to avoid this feature from affecting the validation scores. The number of k-folds was increased to 13, which made the validation period for fold ID: 10 falls on the above period. Tables 6 and 7 show the MAE, % correct prediction and profitability scores for each validation fold for the 5 and 30 mins prediction horizons respectfully. Figures 12 and 13 show various scores in a graphical form while figures 14 and 15 show the related MAE values.

While the figs 14 and 15 show a marked increase in MAE error for k-fold ID 10, the fold with the downward trend marked by a green vertical line, the other figures show that while there tends to be a decrease in the Profitability and % Correct Predictions values for the downwards trend fold,

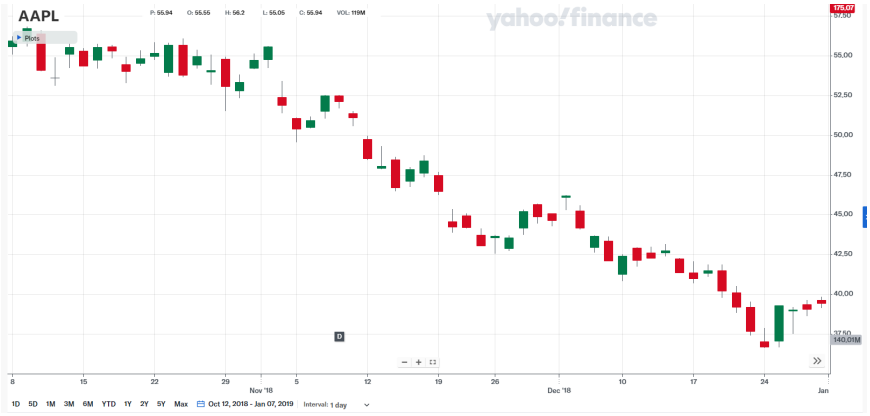

Figure 11: This chart shows the downward period focused on for this micro-study. From 12/10/18 to 04/01/19, the value of Apple stock falls by 32% of its original value

Profitability During Downwards Trend Study - 5mins							
Fold	Start Date	MAE	% Correct On Next Direction (0 con)	% Correct On Next Direction (0.9 con)	Profitability (0 con)	Profitability (0.9 con)	Price Change Over Period
0	01/04/16	0.0009	49.4	46.9	1.7E-04	4.7E-04	-12.7%
1	30/06/16	0.0009	49.9	48.4	4.1E-03	1.3E-02	19.0%
2	03/10/16	0.0009	50.5	52.7	5.3E-05	8.5E-05	5.3%
3	09/01/17	0.0007	48.8	47.1	2.2E-05	1.1E-04	19.1%
4	18/04/17	0.0007	52.9	54.9	1.1E-04	1.8E-04	6.5%
5	20/07/17	0.0007	53.5	55.1	1.2E-04	1.3E-04	4.3%
6	19/10/17	0.0009	50.5	53.8	6.0E-05	1.4E-04	13.4%
7	19/01/18	0.0011	52.9	54.1	2.4E-04	5.1E-04	-0.9%
8	17/04/18	0.0012	50.6	56.1	6.8E-05	4.9E-04	7.7%
9	18/07/18	0.0009	50.3	54.5	4.8E-05	8.3E-05	15.0%
10	12/10/18	0.0015	52.0	54.4	2.5E-04	6.0E-04	-32.3%
11	07/01/19	0.0012	50.2	52.6	6.4E-05	1.4E-04	31.9%
12	03/04/19	0.0013	50.6	53.7	7.5E-05	8.7E-05	1.4%

Table 6: Validation performance statistics for each fold of the ensemble model. Each fold's validation period starts data is marked on its own line and ends on the date marked on following line. Each fold is not trained on its validation period. Each fold is also marked with the overall stock price change over that period (split adjusted). This all combines to allow the reader to compare the performance of the model to be able to perform for periods of different overall price movement trend. This table shows the results for a 5 mins "topics" model and uses the parameters from a high performing model

Profitability During Downwards Trend Study - 30 mins							
Fold	Start Date	MAE	% Correct On Next Direction (0 con)	% Correct On Next Direction (0.9 con)	Profitability (0 con)	Profitability (0.9 con)	Price Change Over Period
0	01/04/16	0.0024	51.7%	53.9%	1.5E-03	4.1E-03	-12.7%
1	30/06/16	0.0023	50.2%	50.7%	2.4E-03	6.1E-03	19.0%
2	03/10/16	0.0019	51.8%	55.5%	6.6E-04	1.5E-03	5.3%
3	09/01/17	0.0015	48.8%	50.0%	3.0E-04	1.1E-03	19.1%
4	18/04/17	0.0017	52.6%	55.4%	5.9E-04	1.3E-03	6.5%
5	20/07/17	0.0019	51.3%	50.9%	1.1E-03	3.5E-03	4.3%
6	19/10/17	0.0017	50.8%	53.3%	6.3E-04	1.4E-03	13.4%
7	19/01/18	0.0029	50.7%	49.2%	9.5E-04	2.2E-03	-0.9%
8	17/04/18	0.0020	50.6%	52.6%	8.6E-04	2.5E-03	7.7%
9	18/07/18	0.0022	51.1%	56.6%	9.9E-04	2.7E-03	15.0%
10	12/10/18	0.0038	51.8%	50.1%	8.2E-04	1.4E-03	-32.3%
11	04/01/19	0.0023	51.2%	50.2%	7.6E-04	1.8E-03	31.9%
12	03/04/19	0.0023	50.9%	52.2%	7.0E-04	1.3E-03	1.4%

Table 7: Shows the equivalent information as table 6, but for the 30 mins prediction horizon

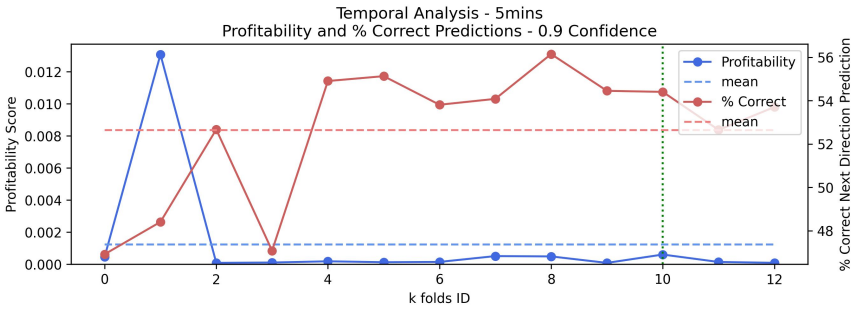

Figure 12: Data displayed for the 0.9 confidence values depicted in table 6

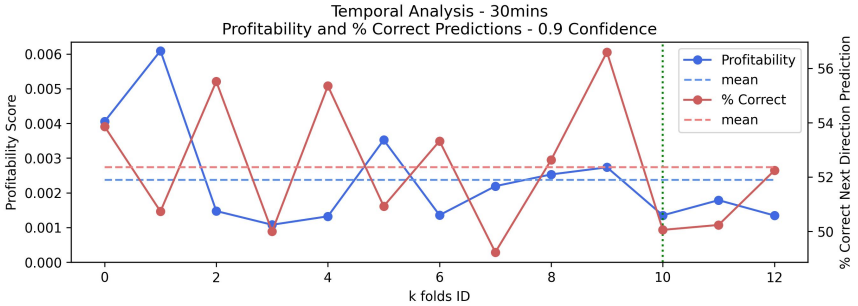

Figure 13: Data displayed for the 0.9 confidence values depicted in table 7

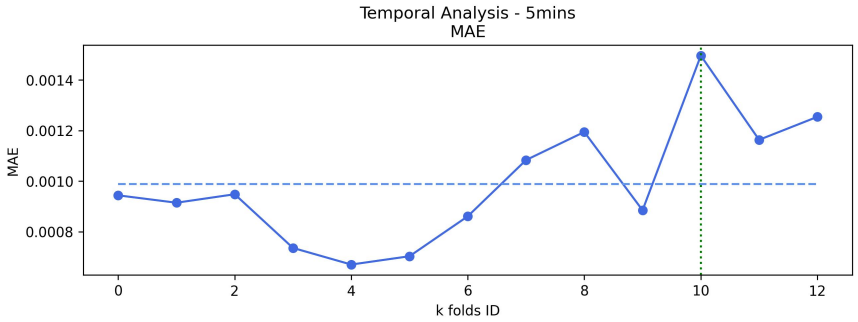

Figure 14: Shows the MAE values displayed in table 6

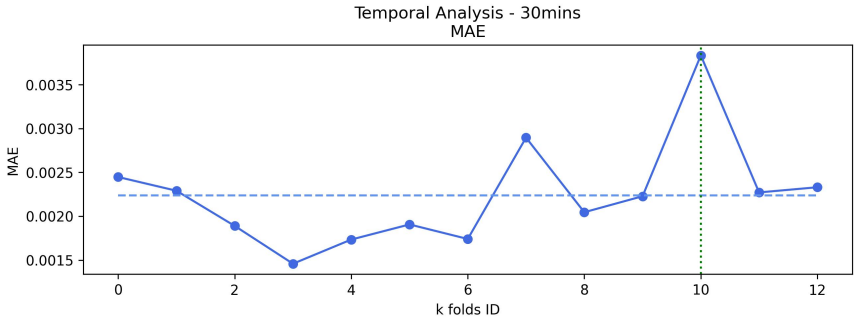

Figure 15: Shows the MAE values displayed in table 7

ID	% Correct	Profitability
training/validation 5 mins, always up	50.3%	3.34E-06
training/validation 5 mins, always down	49.6%	-3.34E-06
training/validation 30 mins, always up	50.7%	2.43E-05
training/validation 30 mins, always down	49.2%	-1.94E-05
testing 5 mins, always up	51.1%	1.20E-05
testing 5 mins, always down	48.8%	-1.20E-05
testing 30 mins, always up	52.8%	7.38E-05
testing 30 mins, always down	47.2%	-6.56E-05

Table 8: Profitability and percent correct direction prediction proportion results for cases where the model always staking in the same direction for 1 euro. The ID column holds information for the period of time (validation or testing), prediction horizon and "position opened" (direction bet). Please note these numbers wouldn't change for if different euro values were placed

this value isn't catastrophic, meaning that while the upwards bias definitely helps similar periods for validation and testing, it doesn't completely negate the ability of the model to predict the upcoming trends/movement of the stock.

It is also worth noting that comparing the profitability score of always betting upwards, table 8 (2.43e-5), against the profitability scores expressed in tables 6 (5 mins horizon, minimum score of 8.30e-5 and mean of 1.23e-3) & 7 (30 mins horizon, minimum score of 1.10e-3 and mean of 2.38e-3). It's clear that the scale of the bias in the historical set is insignificant to the profitability scores being produced. While the effect of bias on a profitability score likely isn't linear, there is still a major difference.

1.8.6 Behaviour During Optimisation

This section discusses how well the parametric optimisation performed.

Figures 16 and 17 show the validation profitability scores for each design produced in order (5 minute and 30 minute respectively). The x-axis shows the order in which the designs were produced. The vertical line between ID 19 & 20 shows the end of the predetermined DoE designs and the start of the generated optimisation designs. Before this each design ID shares the parameters of their matching IDs for other model scopes/prediction horizons and after the line the system generates parameters unique to each model scope according to what parameters the algorithm deems best. Generally, the optimisation algorithms were able to produces decent improvements to the score for each model scope. This is shown with the production of high scores after the red line. Generally,

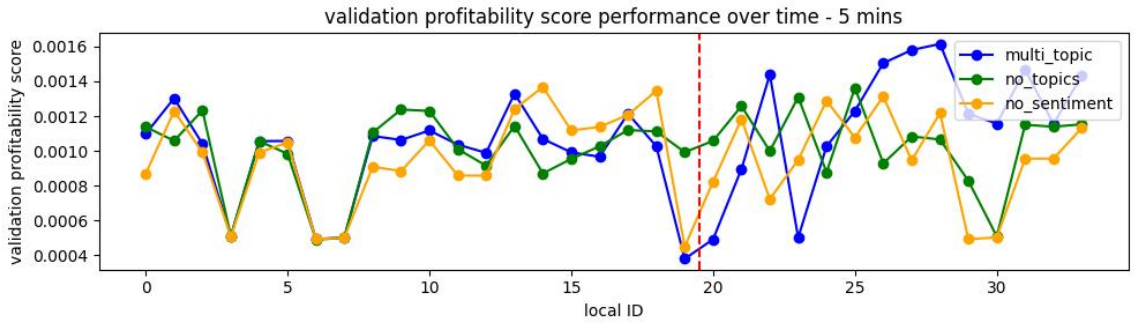

Figure 16: Shows the profitability score for each design produced for the 5 minute prediction horizon

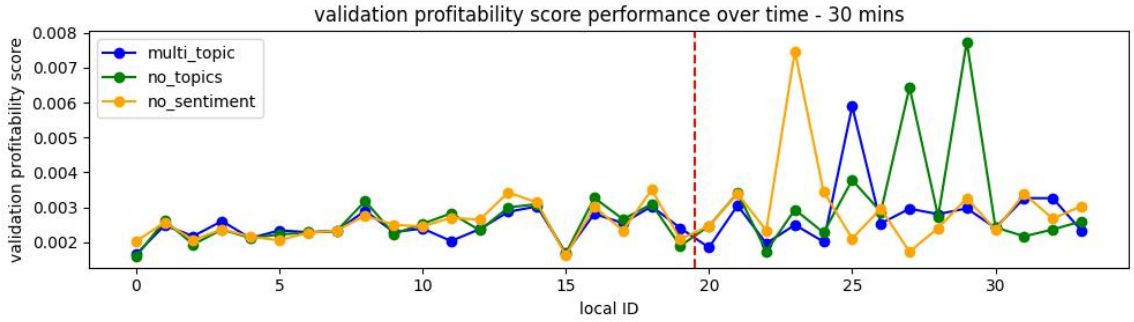

Figure 17: Shows the profitability score for each design produced for the 30 minute prediction horizon

these improvements don't have to be produced instantly, as there is always the potential that the algorithm will look in the wrong area first. It should be noted that the optimisation algorithm applied here is quite simple and is likely prone to get stuck on local optima.

1.8.7 Influential Hyper-Parameters Discussion - Intro

The model hyper-parameter to performance correlation study will have to be limited to the DoE designs as using the optimisation designs will skew the data. The DoE designs are limited to only 20 unique combinations of parameters, each repeated 3 times (per model scope i.e. multi-topics). This means that a correlation could be the product of another parameter, not displayed on the chart's x-axis, which is somehow accidentally correlated to the parameter displayed as a product of the random chance inherit in the production of random DoEs.

Therefore, the results here will need to be treated as initial results and any p-value produced by a correlation study most only be calculated using data points from a single model scope.

If a parameter isn't discussed in this section, then a relationship worth discussing between that parameter and performance wasn't found. Given that there are 12 hyper-parameters varied for the parametric optimisation, discussion of any relationships uncovered will need to be kept brief and will be limited to hyper-parameters concerning the neural network itself, not the pre-processing of Twitter data.

1.8.8 Influential Hyper-Parameters Discussion - L1 Regularisation

Designs with a regularisation above 1e-6 are prone to acting completely differently to other designs. Namely these designs have massive MAE validation and testing errors and in the 5 min context have lower validation profitability scores but decent testing profitability scores, while in the 30 mins context they tend to validate and test for profitability scores quite low. This is show in figures 18 and 19.

The above behaviours are tightly clustered, as can also been seen in the top left of each plot of fig 6.

Interestingly for the 5 min context (fig 18) the high regulation has ensured that there isn't a deterioration between validation and testing profitability score, an effect normally seen on MAE

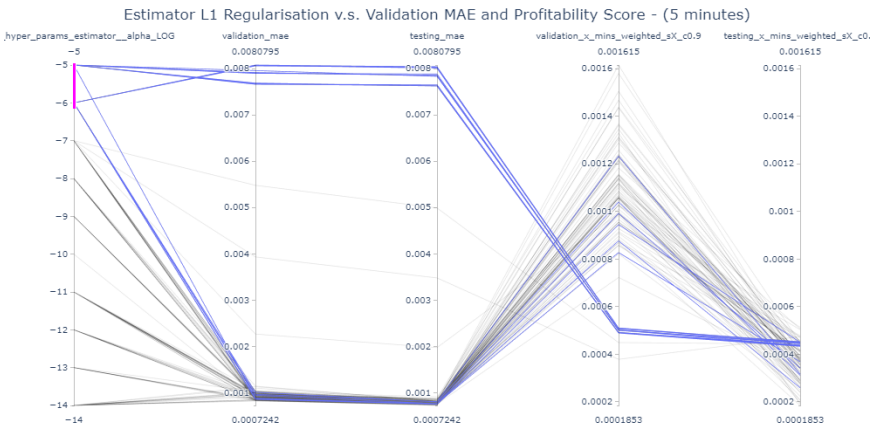

Figure 18: Parallel axis chart showing the designs filtered according to L1 Regularisation value that are equal to the pink band in the axis. This axis is log scaled and labelled "hyper_params_estimator__alpha_LOG". These charts show each design as a single line, with the point at which a line passes an axis giving the design's value for a given parameter. Both charts are filtered for designs that are equal or above a -6 (log) for the L1 linearisation. Designs meeting the filter are shown in blue and not grey. These show all scopes of models with a 5 minute prediction horizon

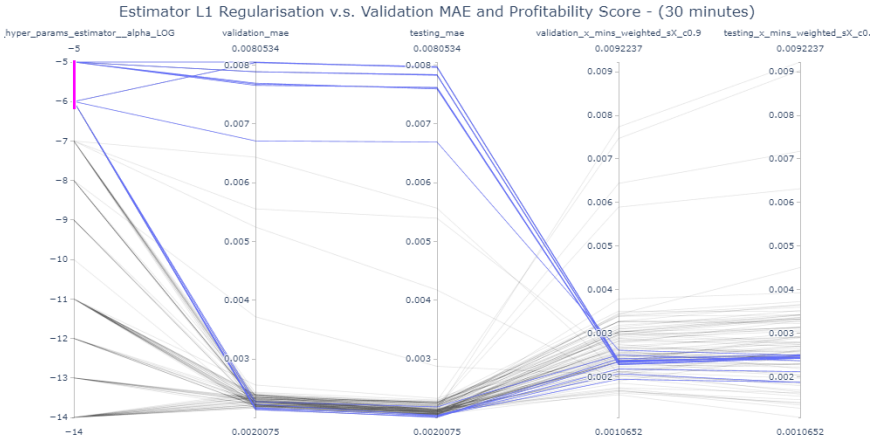

Figure 19: Similar chart to 18 but showing the designs for the 30 minute prediction horizon

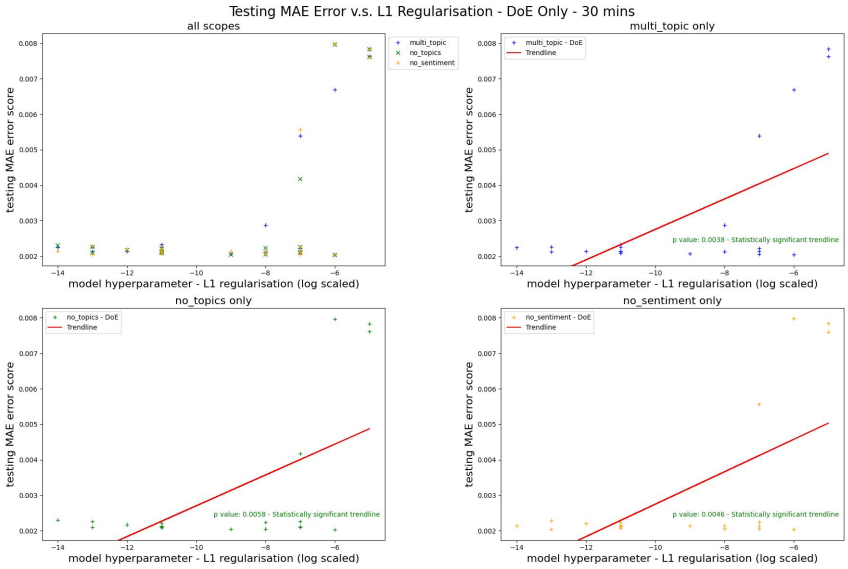

Figure 20: L1 regularisation (x-axis) vs MAE error (y-axis) for the initial 20 DoE designs, shown for the 30 mins prediction horizon

error.

A potential reasons for this similar behaviour is due to the fact that neural networks with high L1 (linear) regularisation models tend to undergo "sparsity", reducing the number of active neurons and connections, making them simple and potentially similar, internally.

Figures 20 and 21 show the relationship for the 30 minutes prediction horizon, before and after the high MAE error designs are removed respectively.

Once these high MAE error designs are removed there is a noticeable negative relationship between L1 regularisation and MAE error, suggesting that for accuracy of the prediction some regularisation is useful however too much can cause a sudden switch in behaviour, potentially the results of either sparsity or the training optimisation getting stuck on false minima.

The relationship between this variable and profitability score is too variable on the scope and prediction horizon that it is not worth commenting on currently.

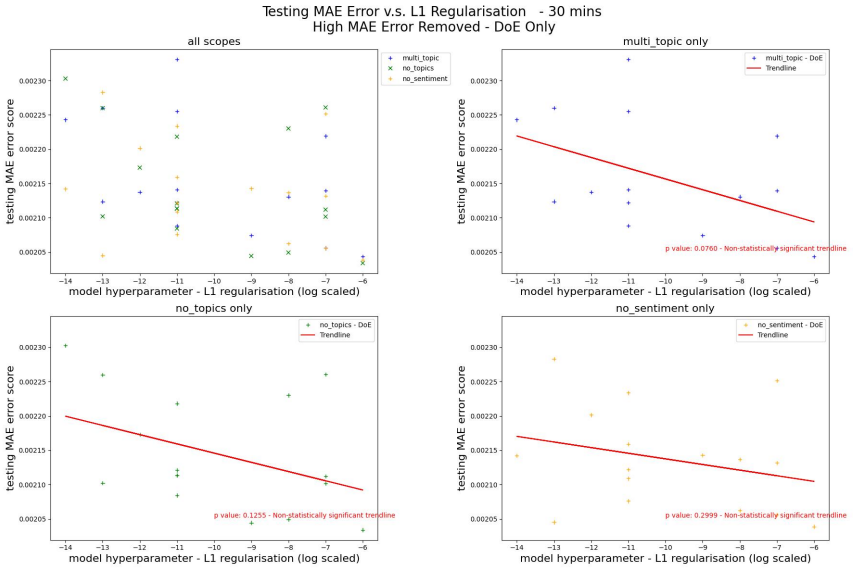

Figure 21: Similar chart to 18 only shows the designs for the 30 minute prediction horizon

1.8.9 Influential Hyper-Parameters Discussion - Early Stopping

This parameter controls a function that stops the training if the validation error doesn't decrease for n steps. In such cases, the model reverts to the state with the lowest error score. However, a potential issue with this function is that models can effectively "self-select" a good validation score by terminating training if it stops improving the fit for a specific validation period (for example, the market between March and June 2018). While the argument for implementing this method is to avoid over-fitting, in this case, with a significant amount of training data available (3.5 years or 2e5 time steps), this issue is less pertinent.

For this project this can cause two potential issues: one potentially reducing testing performance and more subtly, skewing the optimisation towards sub-optimal combinations of parameters that achieve higher validation scores but would ultimately score lower testing scores because the algorithm can only optimise validation scores.

Likely due to the obscuring factors discussed previously it is difficult to draw any conclusions on the above points or the general effect of the early stopping parameter.

1.8.10 Computing Limitations and Effects on Performance

This project was ran on a remote computer, with an advertised AMD EPYC™ (up to 3,7 GHz) 8 vCores CPU and a NVIDIA® RTX™ A4000 for GPU. With this hardware, runs could take anywhere from 2-9 hours, with the availability to run 3 instances at once. In terms of scale of 9e5 tweets were processed and 2e5 time steps of financial data were processed per model. Models were run for a maximum of 40 epochs and the ranges for lookbacks and number of nodes are reported in section 6.2.

The above computational load was why the analysis was limited to 180 designs/runs/models along with the maximum number of epochs (40) and ensemble models (5) per run. With additional computing power and more development time, the above limitations could have be loosened, and a system to ensure that each feature would be used at least once per ensemble model would be implemented.

To check the effect of the above limitation on the predictive performance of the model, a few high-performing designs were selected from the larger analysis and re-run with 100 epochs, no early stopping and with 15 ensemble models per run.

Effect of Increased Epochs and Component Models on 5 Mins Prediction Horizon Models, 0.9 Confidence, Design 1, 30-mins Prediction Horizon								
Scope	Epochs	Comp Models	% Correct On Next Direction		Profitability Score		MAE	
-	-	-	Val	Test	Val	Test	Val	Test
No Sent...	40	5	53.9%	51.3%	3.15E-03	3.36E-03	2.27E-03	2.12E-03
Topics	40	5	52.5%	51.9%	3.12E-03	3.54E-03	2.39E-03	2.25E-03
No Sent...	100	15	56.4%	55.9%	5.62E-03	6.49E-03	2.33E-03	2.16E-03
Topics	100	15	57.0%	55.9%	5.50E-03	6.39E-03	2.33E-03	2.14E-03

Table 9: The first example of a design ran twice with the computation allowed for other models in the study, with and without sentiment and then ran twice with an increased allowance for training epochs and total component (ensemble) models. A marked improvement is seen following this increase. Done for the 30-mins prediction horizon.

The results in tables 9, 10 & 11 don't suggest that increasing the size of the model increases classical accuracy (MAE), however, it seems to sync better with matching the trajectory or future trends of the market better. This is reflected in the repeated large increases in profitability scores for the designs with more epochs and component models. There is a slighter increase for the 5-mins

Effect of Increased Epochs and Component Models on 5 Mins Prediction Horizon Models, 0.9 Confidence, Design 2, 30-mins Prediction Horizon								
Scope	Epochs	Comp Models	% Correct On Next Direction		Profitability Score		MAE	
-	-	-	Val	Test	Val	Test	Val	Test
No Sent...	40	5	53.9%	50.5%	3.05E-03	3.16E-03	2.43E-03	2.21E-03
Topics	40	5	53.0%	49.3%	3.01E-03	3.19E-03	2.45E-03	2.27E-03
No Sent...	100	15	54.2%	51.9%	4.30E-03	5.99E-03	2.42E-03	2.25E-03
Topics	100	15	53.6%	50.3%	3.58E-03	4.20E-03	2.48E-03	2.38E-03

Table 10: Similar to table 9, only with a different design

Effect of Increased Epochs and Component Models on 5 Mins Prediction Horizon Models, 0.9 Confidence, Design 3, 5-mins Prediction Horizon								
Scope	Epochs	Comp Models	% Correct On Next Direction		Profitability Score		MAE	
-	-	-	Val	Test	Val	Test	Val	Test
No Sent...	40	5	53.4%	54.0%	1.10E-03	4.56E-04	9.16E-04	7.57E-04
Topics	40	5	52.9%	53.4%	1.17E-03	3.93E-04	8.70E-04	7.31E-04
No Sent...	100	15	54.1%	53.3%	1.47E-03	4.28E-04	7.85E-04	7.26E-04
Topics	100	15	53.2%	53.0%	1.21E-03	5.15E-04	8.42E-04	7.31E-04

Table 11: Similar to table 9, only with a different design, also for the 5-mins prediction horizon

prediction horizon, which could potentially be dismissed to chance, however the first two tables showing the 30-mins prediction horizon designs, tend to show significant increases of profitability score of around 31.6-80.5% depending on the case.

1.9 Conclusion

In this study, we conducted an in-depth analysis of various model scopes and hyper-parameters to predict stock price movements using sentiment analysis and topic clustering on available Twitter data.

In terms of the business questions set by this paper, asking if we could use Twitter data to help predict the stock market; it doesn't seem that we have been able to prove it, as there hasn't been an improvement in models that use Twitter data. This is discussed in more detail in section 1.8.1. Potential reasons for the lack of performance in models with additional Twitter data include the quality of the data, or the difficult requirements for this project. It seems that just using financial data and related indicators were enough for the moment. The models have a positive profitability

score, suggesting profitability, however it is hard to say how this would translate into real life.

Overall, our results demonstrate correlations between the validation and training scores for all our metrics (MAE, profitability, % Correct Next Direction Prediction) indicating the robustness of our models in generalizing to unseen data. However, it should be noted that our analysis primarily focused on periods characterized by an upward trend in stock prices. We have initially explored the consequence of using these types of models in periods of downward trend in section 1.8.5, which has shown that while the models perform slightly worse, they still seem to be trading profitably within these periods of downwards trend. Which is positive and the best result expectable in this situation.

Assessing the effect of parameters is difficult given the limited number of designs with unique combinations of parameters not effected by the optimisation algorithm. Other obscuring factors included the high variability of any results produced. That said it was found that L1 regularisation was able to improve the 30 minutes prediction MAE for testing, however in all cases above a value of 1e-6, the models would find it hard to converge to an acceptable MAE error. Unfortunately, it was also difficult to assess the effect of the early stopping parameter and the deep random subspace technique, due the small number of data points in the DoE.

This study was limited in terms of computational time, given the survey and optimisation dimension of it, meaning that models had to be limited in training epochs and number of ensemble models. A side study found that increasing the epochs and number of component models led to the model making better trades, while not currently reducing MAE error.

1.10 Potential Next Steps

It's worth briefly discussing potential next steps for this study. Initially it's worth noting that the current approach of taking the general Twitter sentiment isn't working. Given the good scores of the model if performance was the primary concern, the pragmatic approach would be to improve the core model that only uses financial data and at a later date attempt to add a sentiment analysis again. This is because there are various low hanging fruit available in terms of improving the non-sentiment areas of the model. Discussion around improving the sentiment analysis and fixing any potential errors in the current approach are discussed after.

Non-sentiment analysis areas for improvement:

- Purchase of better hardware. Current hardware unable to utilise GPU's effectively, greatly increasing the computational load on the machines. CUDA was installed but didn't function properly.
- The above point could allow from the increase of training epochs, quantity of component models and complexity of the models. The first two points were proven as easy wins in terms of increasing profitability in section 1.8.10.
- Reduce the time series granularity to 1-min time steps. This would mean there is 5 times the data to train on. This would mean increasing the required look-backs by 5 and increasing the training epochs. Utilising GPUs would be needed to support this most likely.
- Intergration of the model to a real world training trading account. A proof of concept would be required before the investment of more research. Metatrader 4 or 5 is likely a good software choice.
- Repeat of analysis on mid-level CAP stocks as they are likely less heavily traded by automated systems. Examples include stocks like: CAT, Brembo, Luxottica etc
- Consideration of the removal of early stopping based on validation loss. There is still the potential for self-selection for models using this.
- Focusing current efforts on a single prediction horizon and model scope, likely "30-mins no-sentiment". This will limit computation load and focus efforts. Other prediction horizons and model scopes can be studied later.
- Changing the error function from MAE to incorporate profitability i.e. rewarding the model for making large jumps in the correct direction. This would effectively turn the predictions in price change reflect the model's confidence in the change

Sentiment analysis areas for improvement:

- The sentiment analysis method needs a redesign. Using a technique like that in the review [11] could potentially be strong. There, multiple high volume/profile posters were identified and their sentiments were each converted into individual training features, therefore if the mood

of a single person changed, that would change the value of a feature. This could likely change the inherent noise in the system. Using Twitter this would require access to live data and a more nuanced filtering process, to find more serious and relevant posters. Unfortunately for this project this approach couldn't be pursued, due to the lack of available data.

- There is a large potential for approaches with access to this much data, for example the filtering of posters that tend to have a great correlation to stock movement, effectively performing feature selection on the twitter data available
- The paper mentioned above used Sina Weibo, a microblogging site from China. This site has typically longer posts and is considered to contain more serious and thoughtful opinions/posts than twitter.
- Potentially using the sentiment from different news sources as individual training feature is another option (i.e. what is the opinion on Apple stock stated in the Financial Times, Wall Street Journal, Il Sole 24 Ore etc)
- Currently the approach of sourcing data from kaggle, while useful for an academic exercise, would need to be replaced with a more high quality data source for a real world application.

2 Report Long Form

3 Introduction

An introduction to this project and its subject domain is given in section 1.1.

The rest of the report will contain technical information concerning the techniques used to obtain the results in the summary section of the analysis and is split into the following sections:

- Literature Review
  - Review of the key papers that influenced the design of the final project
- System Design
  - An overview of the model with a brief discussion of any non-standard features implemented
- Implementation and Results
  - Discussion of the experimentation process and any metrics used

4 Literature Review and Other Important Information

This section provides an overview of relevant literature reviewed in the preparation of this project, highlighting areas that have influenced this study. At the end there is also an introduction on any relevant technical domain.

4.1 Lit Review - Example of Standard Approach

Wang Qili et al [11] showed a good standard approach to the challenge of using stock market and sentiment data to predict the stock market. That example quantized social media sentiment on a website called "Sina Weibo", a Chinese micro-blogging site (analogous to a form of Twitter with longer post sizes), by assigning positive or negative sentiment values to various words according to a preexisting word dictionary. Words were also added from other sources to ensure that finance specific words were added. This was achieved by highlighting finance specific words via TF-IDF being applied to finance news sources in comparison to general news items. (This stands "Term Frequency - Inverse Document Frequency" a technique of highlighting which words frequently happen in one document that are unusual in other documents).

The post database [7]was filtered for users that frequently posted referring to the "stock market", then the sentiment of each individual user was tracked according to the posts they made each day.

If a user didn't post on a given day, their sentiment was "imputed" (extrapolated) from their previous mood score and the market movements.

The daily sentiment of each user was then treated as a separate feature, which generated a large number of features that were then handled using a method called deep random subspace ensembles (DRSE). This is the practice of creating ensemble-methods from models that use different combinations of the available features. This approach was powerful in mitigating overfitting from using a large number of features.

This paper also augmented the stock market information with around 20 technical indicators such as moving average and stochastic oscillator. These indicators are used by traders to understand hidden patterns of stock movements and are likely of use to the statistical models/predictors.

[7]post here being the equivalent of tweet

This is a good example to discuss because it clearly outlines the basic process that any model trying to achieve our goals would have to do and discusses a good technique for handling a large number of features.

4.2 Lit Review - The Effectiveness of Sentiment Data

On its own sentiment data can be used to predict the movement of stocks. However, cases that use only sentiment data, excluding data on the minute-by-minute movement of stock price and other such data, seem to only work on the time scale of predicting the movement for the next day (and not for example the next 1-20 mins) [[9], [6]] These papers also showed a strong positive correlation between negative sentiment and volatility, suggested by the papers as the effect of increased market uncertainty. The available literature suggests that any potential insight from a solely sentiment technical analysis would not be timely enough to profit from in a short time-frame trading scenario as discussed in this paper (we are aiming to trade on 5 min intervals). Bin Wang et al[4], suggests that trading on smaller times-scales such as we are aiming to, short-term sentiment analysis can also be added to the data for a combined solution. Quote: "online sources [sentiment data] does not substitute the traditional finance metrics, but rather supplements"

4.3 Lit Review - The Preparation and Analysis of Sentiment Data

[9] reports that when sentiment data was combined with information from the stock market, there was an increased ability to predict the future movements of stocks. This paper pre-processed tweets by the following process:

1. Tokenisation: The splitting of a text into a series of individual words
2. Stopword Removal: The removal of words that don't express emotion[8]
3. Regex Matching for special character removal: The removal of non-letter characters such as URLs or #hashtags and the names of people to placeholders such as "USER".

[8]Stopwords often refers also to the removal of common words that don't express information i.e. "the", "a", "an", "so", "what" [8], the removal of common words is how it is utilised in this thesis project

This paper also warns against the use of sentiment analysers that were originally trained on different text of a different subject/type (i.e. about news articles about film not stocks). This paper also labelled tweets that were positive, neutral or negative manually by the researchers and then a classifier was trained on top of this to create a custom tweet classifier, which was then taken forward into a larger stock movement prediction model. Then this classifier used feature extraction like word2vec representation on data prepared in the manner previously discussed to convert the series of words into a vector representation.

While the manual training for a custom sentiment analysis analyser would be ideal, there isn't scope in the project to do so, therefore an off the shelf solution used in another example[6] called "vader" was be implemented in this thesis project. In the cited case it was also used to analyse if a tweet is positive, neutral or negative. This was then fed into a classification model to predict if a stock was going up or down the next day.

4.4 Lit Review - Other Examples of Novel Approaches

Predicting short-term stock prices using ensemble methods and online data sources A paper by Weng et al [4], expanded the sources of sentiment data, including trends on Google searches for items, number of unique visitors for pertinent Wikipedia pages and counts of bull or bearish news items for a given stock per day. This paper also utilised the following statistical models: neural network regression bagged ensemble, support vector regression bagged ensemble, boosted regression tree and random forest regression. It was stated that in the cross validation phase the AI platform could automatically pick the "best" ensemble for a given stock. This paper made the important point that it is more useful to build a model that tries to predict the exact price change instead of a boolean up or down prediction as whatever model is made it can always be scaled back to the simpler "classification" predictor.

On the Effectiveness of Candlestick Chart Analysis for the Brazilian Stock Market Prado [10], found that the application of patterns developed for other markets when applied to the Brazilian markets were not guaranteed to work, depending on the pattern. This means that any set of established patterns developed for one market may have to be adapted when applied to another market.

Stock Price Movement Prediction Using Sentiment Analysis and Candlestick Chart Representation [7], tested various neural networks on predicting stock market movements based on creating images of stock market movements in their candlestick form. They found conventional neural networks best for predicting stock price movements base on visual patterns alone. This also used the language processing tool "vader" to classify tweets as either positive, negative or neutral which was used to add additional features to the predictor. [12] also used a similar NN approach but converted the candlestick patterns into a data format called "wavelet" which simplified the images of stock "candlestick" patterns to grey-scale matrices with a lower resolution.

4.5 Long and Short Trading

Long and short trading is the practice of effectively betting on whether a stock price will go up or down. A "position" is effectively an open wager, positions are "opened" and "closed" to begin and terminate them. "Going long"/"longing a stock" is a wager that a stock will increase in value and "going short"/"shorting" is wagering a decrease.

When a position is taken, the person must consider the "spread", this is the gap between the highest price that someone else is willing to buy the stock at (bid price) and the lowest price someone is willing to sell the stock at (ask price). When they close a position, lets say long, the current bid price, must have gone above the ask price at the point they opened the position, to make a profit on the position[9]. Shorting is effectively the reverse and is facilitated by the trader making a promise to purchase a quantity of a stock at a later date and being given the cash for the current price right away, should they be able to buy the stock at a later time for less, they can pocket the difference. Obviously in the two above scenarios, if the price of the stock doesn't move in the planned direction, the position will close at a loss.

Long and short trading is attractive because of the practice of leverage. This is where any losses and gains are multiplied via a complex lending mechanism. The trading broker opens parallel positions to yours and then upon completion, hand any resulting wins/losses to the trader. The amount of leverage allowed depends on various factors such as the agreement with the broker, the

[9]This sounds complex but effectively, when you open a position you buy X of a stock, you then need to wait until the price someone else will buy it at is higher than your original buying price to make a profit.

balance of the cash account with the broker, local laws etc. The above also incurs lending and transaction fees from the broker.

Three major risks of this system are, firstly leverage can cause the trader to lose money quickly and more than is in their trading account, secondly potential losses in shorting are technically infinite as there is no limit to how high a stock can increase in value, given say, sudden news, short squeezes or other market forces, thirdly brokers have a mechanism that should the potential losses on a position threaten to eclipse the available balance on a trading account, they often, automatically close the position to protect themselves, handing the trader any resulting losses.

This is a short overview of long and short positions. More detail and maths on the subject can be found in appendix 7.2.

5 System Design

This section discusses the design of the system, including the underlying mathematics and theory used. After reading this section, the reader could design a system with a similar approach used to produce the final results.

5.1 Design - System Overview

This subsection outlines the general system that was developed. The following subsections will cover more detail about individual modules.

Figure 22 shows the overall layout of the proposed system's logic. The "experiment manager" process controls and records the output of all the other processes. It does this by issuing a set of design parameters for a given iteration of the model, which are fed into the below processes, affecting their exact workings and outputs, these are the:

1. Preparation of custom finance data
2. Preparation of custom sentiment data
3. Preparation and training of a custom ensemble statistical-learning model

Effectively these parameters act like hyper-parameters for the statistical model. They effect factors such as the quantity of topics, how topics are weighed, the half-life of a tweet's "relevance" (weighting over time), the size of the neural network, the neural network's L1 regularisation etc.

The produced model is then fed into the scoring process which produces a range of training, validation and testing scores. The experimental manager then considers the validation scores in the selection of the next set of design parameters. Due to the high computation cost of a single run of this system, a rudimentary form of multi-objective optimisation is implemented to reduce the number of runs needed to explore different combinations of the hyper-parameters, the exact process for is discussed in section 5.2.

The model configurations with the highest validation scores will then be examined to understand if they also produce good testing scores, to validate the overall process.

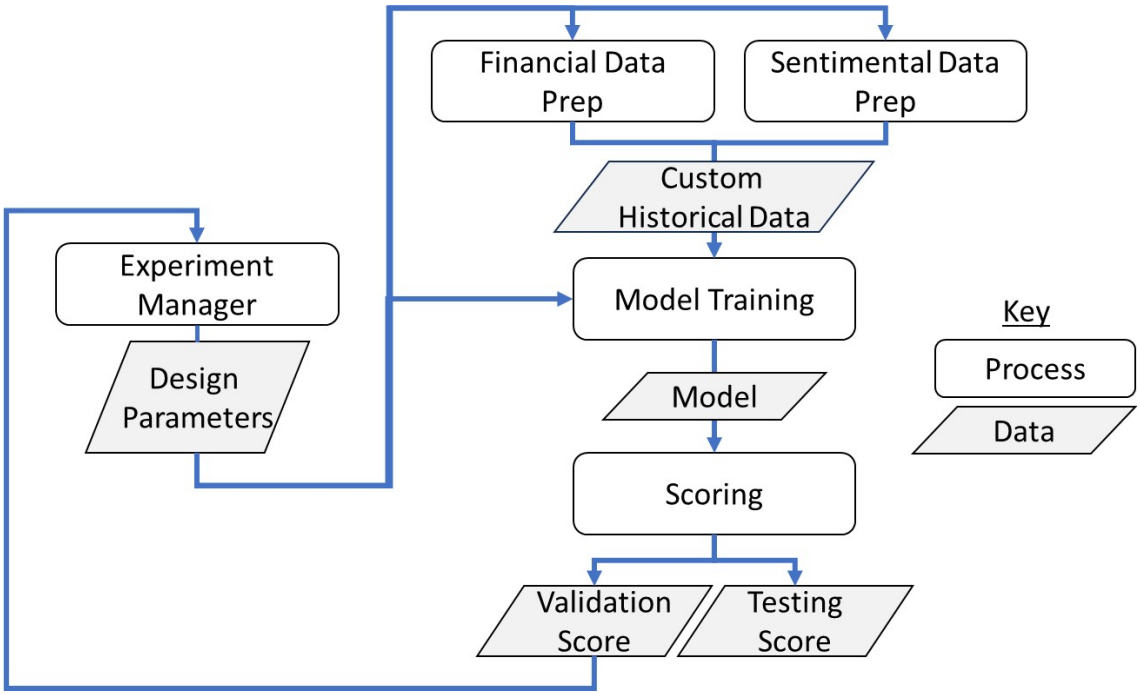

Figure 22: Overview of the system created for the project. The experiment manager produces a set of design parameters, which control the subsequent processes of data prepossessing and model training. The model is finally validated and tested. The validation scores are fed back to the experimental manager to inform later deesigns before the process is repeated

5.1.1 Reuse of Assets

Multiple assets are produced for each model, including sets of annotated tweets, sentiment data and predictive models, many of which are quite computationally expensive to produce. Therefore they are saved for potential reuse meaning that they can be reused should a future run have need for them. Please refer to appendix 7.4 for more detail.

5.2 Design - Experiment Manager

As described in the previous section the experiment manager decides and records all models created by the rest of the system. The schedule for a set of experiments is to:

1. Run a random design of experiments: a collection of randomly selected combinations of design parameters, designed to take a survey of the effect of each input parameter

2. Order additional models to be produced according to a rudimentary multi-objective optimisation, which looks at the validation scores of previous models and selects a set of design parameters based on what it projects to be potentially a high scoring design

   (a) One in ten designs is picked at random to continue global search and to prevent a convergence on a local minimum (the exact number is set according to the parameters of the optimisation process, currently disabled)

   (b) If a design has already been run, it is rejected in favour of a random design to encourage increased global search

   (c) Currently this is a multi-objective optimisation in the sense that is cycles between different mono-objective functions as targets for maximisation for when generating a new set of design parameters.

There are a wide range of more powerful techniques available for multi-objective optimisation for example such as NSGA-II, various deep learning methods, pareto surface generation for convex multi-objective instances etc. These were not implemented due to time constraints.

5.2.1 Advantages of Multi-Objective Optimisation

Regarding the multi-objective design optimisation method implemented. while there are more nuanced approaches to multi-objective optimisation this simple solution still offers an increased diversification of design solutions generated. Diversity among generated solutions has a range of benefits including:

- The generation of more unique solutions, which may in time provide unique design features that inform the final optimal solutions
  - This is effectively the effect of mitigating against the pitfall of focusing on a single false local minima
- Potentially the generation of model features that are potentially more profitable while scoring lower on classical statistical performance scores such as MAE/MSE, part of the final objective id to predict the next movement of the stock not necessarily the exact next price.

5.3 Design - Finance Data Prep

This section explains the preparation of the financial data. Nothing is explained for the implementation and theory as this section was quite straightforward.

Detailed long term finance data was purchased from firstratedata.com for the stock ticker AAPL (Apple which contained the following data for the period studied:

- OHLCV: Standard information about the trading of a given stock/index during a time-interval (5 min for this project)
  - Open: The first price the stock was traded for at the start of the time step
  - High: The highest price the stock was traded for during the time-step
  - Low: The lowest price the stock was traded for during the time-step
  - Close: The final price the stock was traded for during/at the end of the time-step
  - Volume: The quantity of stocks exchanged during the time-step

The following financial indicators were then calculated using the OHLCV data and then added in parallel. These are composite values used by traders to understand trends in the market. This was achieved using the module "pandas_ta":

- Financial Indicators
  - Simple Moving Average
  - Exponential Moving Average
  - Moving Average Convergence/Divergence
  - Bollinger Bands
  - Pivot Points
  - Doji Movement Indicators

Most of the above indicators are quite simple and definitions can be found with simple google searches.

The following data was not included as it was not available for purchase at the project's budget:

- Bid Price: the highest open offer on the market to buy a stock
- Ask Price: the lowest open offer on the market to sell a stock
- Market Depth Data: the quantity of open bids and open asks at different price levels, for example this would tell you how many open offers to sell a stock at price X are currently available and is an advanced indicator of the markets "depth"[10] of interest in a stock, beyond the best bid/ask price.

5.4 Design - Sentiment Data Prep

This section describes the justification, theory and an overview of the implementation of how the raw Twitter data was converted into both a set of continuous sentiment scores; tracking the overall mood positive/negative for each subject, and a continuous normalised interest score tracking the relative volume of discussion for each topic. This is done to give models the ability to respond to perceived changes in sentiment and/or interest across multiple topics. The system also allows the inclusion of one or both of these scores into a produced model.

This process is done in the steps as visualised in figure 23:

1. Generate x topic clusters - section 5.4.2
    - these clusters are referred to as the "topic models"
2. Annotate each tweet with a normalised weighting according to its belonging to each topic cluster and a single value for its overall sentiment - section 5.4.3
    - sentiment is a continuous value anywhere from -1 negative to +1 for positive
3. Finally, calculate the estimated sentiment score and topic interest for each topic for each time step - section 5.4.4

More detail and pseudo-code can be found in the appendix 7.4

[10] depth here meaning the number of people currently interested in the stock

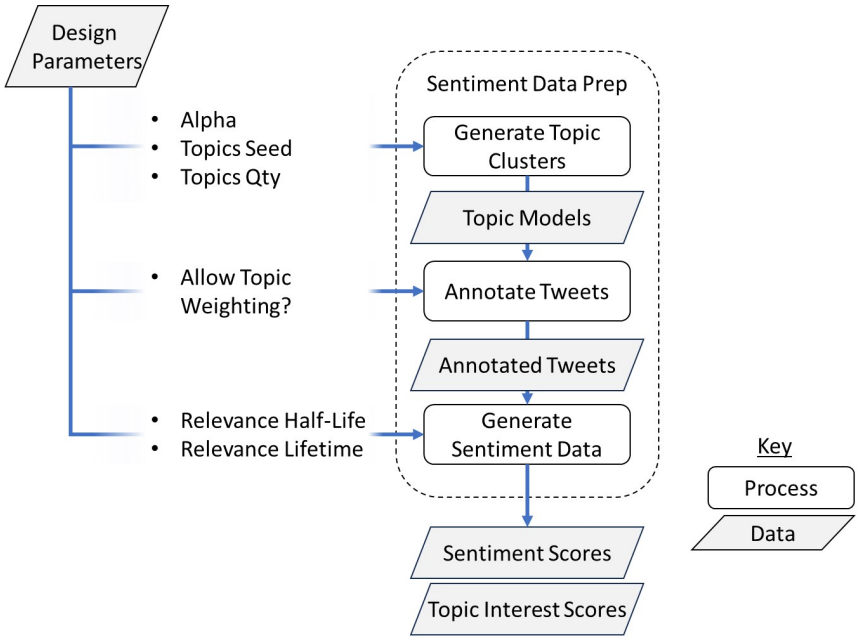

Figure 23: An overview of the process steps for the preparation of sentiment data

5.4.1 Justification for the Multi-Topic Sentiment Analysis Approach

As discuss in the lit review the objective of sentiment analysis is largely (but not completely limited to) the analysis of digital text to determine the emotive tone of the message. Generally, these methods convert a body of text to a single value signifying their overall mood from -1 (signifying strongly negative) to 0 (for neutral) until +1 (for strongly positive). There are various examples of using this data to gauge the movement of stocks during the next day of trading [7].

An issue with the above approach is that it doesn't differentiate the type of tweet that is being read. For example, a 3rd party advertisement will likely tend to sound positive. However, if it mentions Apple, it shouldn't really indicate any market sentiment around that company but may still skew a model's perceived sentiment around that stock.

Another important example is the type of news being shared. The sharing of a rumour is likely to signal a steady increase in stock price, where there is a possibility that a big breaking news story will cause a small positive spike in stock price followed by a larger drop in price. This is often because insightful traders have often bought upon hearing the rumour and await the rush of less

insightful, excited traders on the "splash" of the big news story, to sell them their shares during the price spike. This has led to the trading maxim "Buy the rumour, sell the news"[5].

These are two examples as why just tracking sentiment of tweets based solely on the positive/negative can lose detail that a human reader would pick up on.

5.4.2 Topic Clustering

A python package called gensim was used to create topic clusters. These clusters were based on LDA ("Latent Dirichlet Allocation"),

A python package called Vader (Valence Aware Dictionary and sEntiment Reasoner) was used for the topic clustering. The major inputs of the software were:

- The corpus of tweets
- The number of topics/clusters desired
- Alpha
  - A training parameter, in the case of this python module, it controls the average number of topics a document tends to belong to

Before the tweets could be used for any processes they were prepared by:

- Filtering to just tweets that mentioned Apple
- Any mentions of cash amount were changed to "MONEYAMOUNT"
- Any mention of Apple, its stock ticker or any other textual indicator of Apple was replaced with the stockword "Apple"
  - This is to stop the software clustering false topics around different methods of mentioning the company
- Any mentions of other companies are replaced with "OTHERCOMPANY"
  - A special list of shortened company names with any business entity suffixes, such as "incorporated", "inc.", "ltd" etc, removed, had to be created, to also allow the software to

defect these shortened versions of names, which were also replaced with the "OTHER-COMPANY" place holder

- Filtering to remove any tweets that mentioned more than one other company as these ended up more often being advertisements completely off useful topics for the study
- Removal of any website link
- Removal of grammar marks
- Removal of stopwords, words such as "and", "that" etc that don't convey meaning or subject
- Removal of business entity suffixes such as "incorporated", "inc.", "inc"

Product keywords were allowed to remain in the text so words like "jobs", "iphone", "iphone", "airpods" as these ended up being major contributors to the overall topic clusters, concerning product lines.

Additionally the allowable ranges for the topic model's input parameters required effective ranges established. These ranges were established by manually generating and introspecting the produced topic clusters with the python modules native visualisations as shown in figure 24. Resultant values for perplexity were also examined. The sensible parameter ranges that were established can be seen in table 12.

Figure 24 shows the visualisation that can be used to inspect the topic clusters produced. In the example pictured topic cluster "2" is selected and the words along the right side are associated with that cluster. The length of the bars by each word denotes the overall mentions of the word, where the red section of the bar denoted the mentions of the word associated with the cluster. In the image below we can see that this cluster is closely associated with words like "tech", "trial", "ip", "patent", "research". This suggests that this cluster is around tweets discussing new technology and research.

Some of the clusters examined remained difficult to fully interpret but this is an acceptable condition, as in many cases there were visible links between groups of the words within the clusters or potentially they were multiple real-life subjects combined into a single topic.

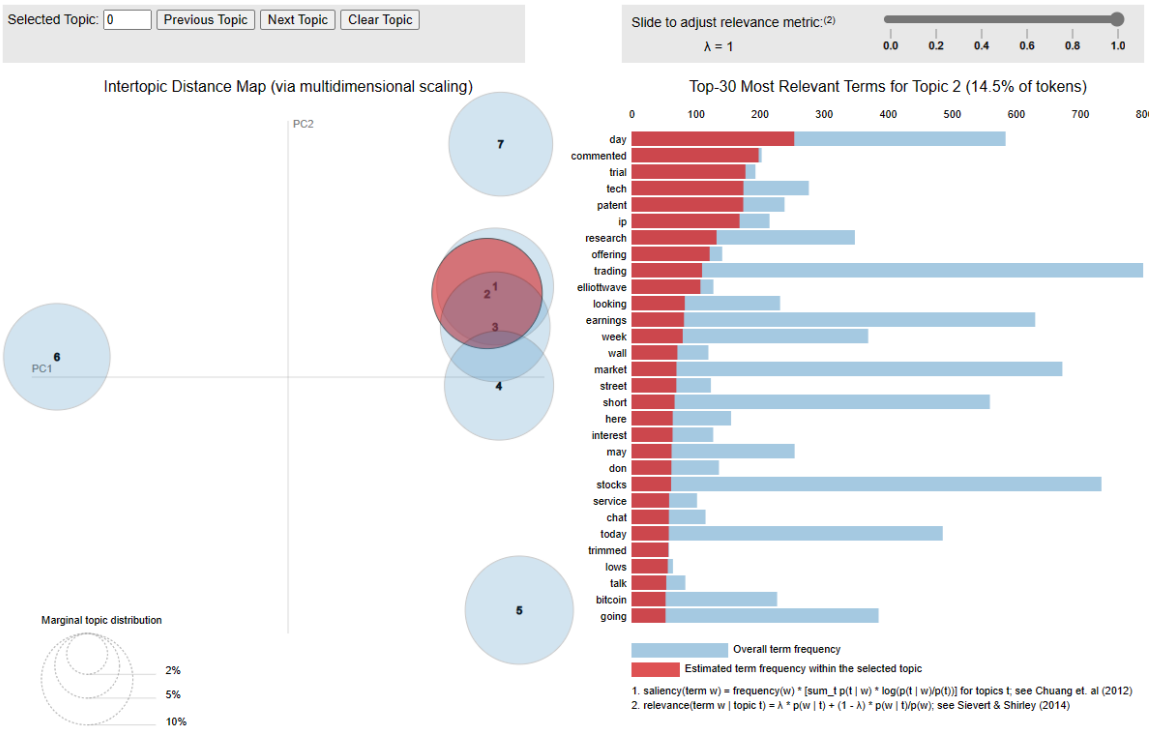

Figure 24: Example topic cluster created in the designing phase of the project, here prominent words include: "trail", "tech", "patent", "ip" and "research", words with an obvious thematic link

Finally, this was integrated into the larger process so that a topic model is automatically created according to the specification sent from the experiment manager for each iteration of the model produced.

5.4.3 Topic Annotation (step: Annotate Tweets)

This stage is quite straightforward. Here every tweet has its text examined by the LDA (module "gensim") topic model and is assigned its weighting across all the topic clusters and then the sentiment analysis package ("vader") assigns is a score for the sentiment from very positive +1 to very negative -1.

5.4.4 Sentiment Analysis Mathematics (step: Generate sentiment Data)

In the final stage of the sentiment data preparation, the objective of the sentiment analysis is to combine the individual tweet topic weights and sentiment scores into a time series estimating the overall sentiment and relative topic interest for each topic cluster and time step.

To calculate the continuous sentiment score and topic interest for a given topic cluster the following factors are considered for each tweet:

- The time that a tweet was made, with more recent tweets holding more weight
  - The value for time_weight decays as a half-life defined by the parameter "relevance half-life", starting as 1 if the tweet was produced zero seconds before the time step in question
- The weighting of the tweet to the topic cluster in question
- The overall sentiment of the tweet

Sentiment Score

This is the calculation for the sentiment score for a topic at a time step:

$$sent_{t,j} = \frac{\sum_{i=1}^{n} tweet\_sentiment_i \cdot topic\_weight_{i,j} \cdot time\_weight_{i,t}}{\sum_{i=1}^{n} topic\_weight_{i,j} \cdot time\_weight_{i,t}} \tag{1}$$

This is the sentiment score for:

- time step - t
- topic - j

Every tweet (index i) that meets the following criteria is considered in the above equation:

- It must be created before the start of the timestep in question
- It must not be older than the period "(tweet) relevance lifetime" from the start of the time-step
- Must mention the company name, symbol or a generated shortened version of the company in question's name i.e. "Microsoft" for "Microsoft Corporation".

This process is repeated per topic cluster and time step to create a continuous weighted average sentiment score for each topic cluster.

The values for tweet_sentiment and topic_weight for each tweet are assigned in the previous step. The exact calculation for time_weight for a tweet is:

$$\text{time\_weight}_{i,t} = 0.5^{\left(\frac{\text{time since timestep}}{\text{relevance half-life}}\right)} \tag{2}$$

Topic Interest

The second score calculated by this larger section is the relative topic interest, a measure of the volume of talk concerning a topic, relative to the other topics. This is calculated as:

$$topic\_interest_{t,c,j} = \frac{\sum_{i=1}^{n} topic\_weight_{i,j} \cdot time\_weight_{i,t}}{\sum_{i=1}^{n} time\_weight_{i,t}} \tag{3}$$

Where the factors and indexes are the same as above.

5.4.5 Model Scopes (Multi-Topics, no-topics, No Sentiment)

The exact configuration of equation 1 depends on the "scope" of the model. The "Multi-Topic" models have all the stated expressions, whereas the "no-topics" will always equate the "topic_weight" value to one as this isn't considered in this example. The "No Sentiment" scopes completely ignore above calculations and any information from Twitter.

5.5 Design - Model Training

This section covers the major statistical techniques/features used and their underlying theory. After reading this section the reader could create a similar model following the design philosophy applied. The statistical model was based around a multi-layer recurrent neural network ("RNN") model [11]. The major features used in the design of this model are:

- Recurrent Neural Networks
- Deep Random Subspace Ensembles

[11] https://keras.io/api/models/sequential/

  - a form of bootstrapping/bagging, not included in the scikit module, which needed to be manually applied within the model

- K-Folds Validation

5.5.1 Recurrent Neural Networks

The basic statistical model was based on a multi-layer recurrent neural network ("RNN")model[12]. The python package keras was used.

Recurrent neural networks (or "RNN") are a type of neural network that allows information from previous time steps to affect the output. Information about standard neural networks is available in appendix 7.3. How RNNs analyse multiple time-steps at once is dependent on the exact type of RNN employed, usually, there is a typical feedback loop within the neuron itself although more complex configurations are available.

The types of RNNs used in this example are "Long Short Term Memory" (LSTM) and "Gate Recurrent Units" (GRU). These methods are generally used in language tasks and both work by feeding back information to themselves while having an internal state which can decide if information being passed to the node can be forgotten, stored or fed back to the current calculation/other nodes. This active system allows more complex neural networks as these nodes don't suffer from diminishing backward gradient issues. NNs based on "Simple RNN" layers were also attempted but proved too long to train. This potentially is due to these networks not having the internal state functionality described above, therefore having to maintain more internal information.

RNNs are particularly powerful in predicting time series, as the shape of previous financial data over time can be considered. This consideration of these shapes is popular in financial trading circles for predicting stock movements, fig 25 shows an example of typical patterns used to predict future stock movements.

5.5.2 Deep Random Subspace Ensembles

Deep Random Subspace Ensembles, (also known as "attribute bagging" or "feature bagging"), is a method of ensemble modelling where component models are trained on different subsets of features.

[12]https://keras.io/api/models/sequential/

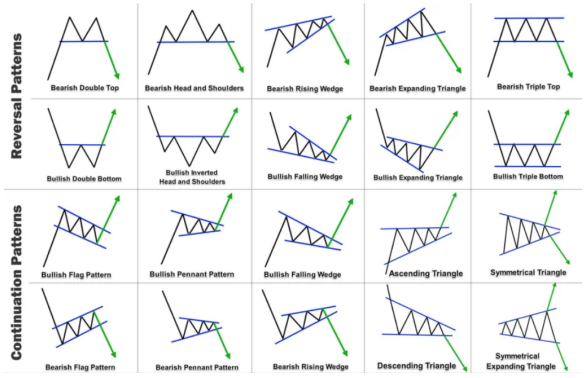

Figure 25: Typical patterns used to predict stock movement manually

```
for i_random in range(model_hyper_params["n_estimators_per_time_series_blocking"]):
    # randomly select features to drop out
    n_features = X.shape[1]
    dropout_cols = return_columns_to_remove(columns_list=X.columns, self=self)
    X_sel = X.loc[X.index[train_index].values].copy()
    X_sel.loc[:, dropout_cols] = 0
    y_sel= y.loc[y.index[train_index].values].copy()
    estimator.random_state = global_random_state
    global_random_state += 1
    estimator.dropout_cols_ = dropout_cols
    estimator.fit(X_sel, y_sel)
    self.estimators_ = self.estimators_ + [estimator]
```

Figure 26: Code implementation of the deep random subspace

This can help avoid over-fitting by creating multiple "weaker" models with fewer features as more weaker models create a more robust analysis and fewer features reduce the possibility of over-fitting. This was discussed in the lit review and was used by Wang et al to achieve this goal [11].

In this project there are many features and given the complexity of the relationship between the inputs (historical data) and the output (prediction of stock movements), it would be difficult to implement effective feature selection. Hence the above solution of random removal was selected.

As this method doesn't come native to the scikit-learn module, this approach was implemented by giving each component regressor a customised X input where the features which were to be ignored were changed to zero pre-training fig 26. Then L1 regularisation was used to ensure that the model would naturally remove any consideration of these features fig 27. A record of these features were attached to the model so that they could also be neutralised for any later prediction operation.

The method "return_columns_to_remove" selects quasi-random features for removal/neutralisation

```python
def return_RNN_ensamble_estimator(model_hyper_params, global_random_state, n_features, dropout_cols):

    ensemble_estimator = Sequential()

    for id, layer in enumerate(model_hyper_params["estimator__hidden_layer_sizes"]):
        # prep key word arguments
        kwargs = {
            "units" : layer[1], "activation" : model_hyper_params["estimator__activation"], "return_sequences" : True
        }
        if id == 0 or id == 1:
            kwargs["input_shape"] = (model_hyper_params["lookbacks"], n_features)
        if id == len(model_hyper_params["estimator__hidden_layer_sizes"]) - 1:
            kwargs["return_sequences"] = False
        kwargs["kernel_regularizer"]   = tf.keras.regularizers.L1(model_hyper_params["estimator__alpha"])
        kwargs["bias_regularizer"]     = tf.keras.regularizers.L1(model_hyper_params["estimator__alpha"])
        kwargs["activity_regularizer"] = tf.keras.regularizers.L1(model_hyper_params["estimator__alpha"])
        # add layer
        if layer[0] != "simple":
            kwargs["recurrent_activation"] = "sigmoid"
            kwargs["recurrent_dropout"]= 0
            kwargs["unroll"] = False
            kwargs["use_bias"] = True

        if layer[0] == "simple":
            ensemble_estimator.add(SimpleRNN(**kwargs))
        elif layer[0] == "GRU":
            ensemble_estimator.add(GRU(**kwargs))
        elif layer[0] == "LSTM":
            ensemble_estimator.add(LSTM(**kwargs))

    ensemble_estimator.add(Dense(units=1, activation='linear'))
    opt = keras.optimizers.Adam(learning_rate=model_hyper_params["learning_rate"])
    ensemble_estimator.compile(optimizer=opt, loss=model_hyper_params["testing_scoring"])
    ensemble_estimator.random_state = global_random_state
    ensemble_estimator.dropout_cols_ = dropout_cols

    return ensemble_estimator
```

Figure 27: Code implementation of L1 (linear) regularisation added to RNNs

for the component model in question. The probability of an individual feature being removed is according to what type of feature they are, and the removal rate assigned to that feature is set by the dictionary displayed in figure 28. This ensures that critical values like the close price (the feature being predicted) are always included.

5.5.3 Validation, K-Folds Blocked Validation, Early Stopping and Testing

The models for this project were trained using k-folds blocked validation with a configuration similar to as shown in figure 29. Each component model is trained on a different validation period i.e. row on the figure.

In this figure, the blue sections represent training periods and the orange validation periods. During training at the end of an epoch the model is validated using the validation period and the relative change in the "val_loss" function is used to determine if the training continues. If this validation loss doesn't decrease for a specified number of epochs, then the model is restored to the version that scored the lowest "val_loss" and training is completed. This number of epochs is defined

```
cohort_retention_rate_dict_strat1 = {
            "£_close" : 1, #output value
            "£_*": 1, #other OHLCV values
            "$_*" : 0.5, # technical indicators
            "match!_*" : 0.8, #pattern matchs
            "~senti_*" : 0.5, #sentiment analysis
            "*": 0.5} # other missed values
```

Figure 28: Screenshot of the default rate that feature was not removed in the training of an individual

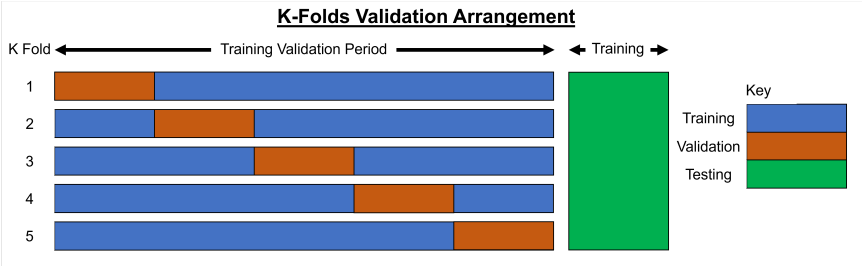

Figure 29: Example k-folds validation for a time series. Training sets are blue and validation in orange. Once the ensemble model is assembled, it is tested on a separate testing period

as the hyperparameter "early stopping" (see table 12) which is managed by the experiment manager for the entire system. It must be noted that each component model is only trained using training data. For both the training and the training validation the measure used is MSE.[13]

After this training process each component model is tested again against the validation set using the custom scoring method designed to approximate model profitability (detailed in section 6.3) and new value is saved as the component model's validation profitability score. These scores are averaged across all component models to create the final model validation score for the ensemble model overall. This score is used in the selection of new designs by the optimisation algorithm as detailed in section 5.2.

5.5.4 Testing

Testing is finally done in a separate time-period as used for the training/validation process. Models are tested on a range of measures both traditional and the custom measures detailed in section 6.3.

[13]this produced better results than MAE and R2

A main measure of approach validity is that the models with a high validation profitability score also have a high testing profitability score, as this means that the profitability score is repeatable for different periods of time. This is confirmed in sections 1.8.2 and 1.8.3 of the results.

5.5.5 Data Leakage

Technically this model has a negligible amount of data leakage in the validation stage as the RNNs are trained off sequences of 8-40 consecutive time steps (at 5 min intervals), the training and validation folds could then share up to 7-39 timesteps at their limits. As this is in period of about 200k time-steps over the 3.5 years used in the training/validation dataset, this is an allowable amount of data leakage.

6 Implementation and Results

With the system logic defined and justified, this section defines how the experiment was carried out concerning the handling of results, justification of the parameters selected with their assigned ranges and any scoring systems used with their justifications/limitations.

6.1 Multi-Objective Optimisation Implementation

As discussed in section 1.1, this paper will benchmark the potential for including topic clustering to support automated stock trading. It will do this by creating a standard stock market predicting model and then asking the following questions:

- Is there an improvement in stock prediction with the addition of linear sentiment analysis as a training feature?
- Is there an improvement in stock prediction with the addition of multiple training features tracking the sentiment across multiple topics?

This model has been designed with several potential training hyper-parameters, many of which have been fixed (but not hard-coded). 12 hyper-parameters have been identified as potentially adjustable to maximise model performance see table 12. Potential values for each of these hyper-parameters were then listed, which predicted a potential 33'177'600 combinations of these parameters. Due to this a parametric optimisation approach was selected to attempt to quickly find as optimal a configuration as possible for the limited number of runs available.

Please note this section assumes that the reader understands the general process for a parametric optimisation as discussed in section 5.2 when the experiment manager is discussed.

To answer each of the above questions, 3 separate scopes of the model were optimised, these were:

- multi-topics - With all sentiment analysis included
- no-topics - sentiment analysis without splitting the grouping of tweets into topic clusters
- no-sentiment - The model ran without any Twitter data

To ensure parity, each of these optimisations were ran from the same random initial design of experiments (DoE)and then allowed to optimise independently.

6.2 Model Hyper-parameters

Below is a table of all the experimental variables edited by the optimisation. Please note that all model input parameters were controlled by a multi-levelled dictionary which is edited and passed into a method that then creates, trains and tests a new version of the model based on the contents of that dictionary. This dictionary is divided into sub-dictionaries, each controlling different aspects of the design:

- Temporal – controls the starts, ends and time step lengths of the training and testing periods
- Financial - controls the OHLCV, technical indicators and other financial instruments compiled into the input data
- Sentiment – controls aspects of the generation of the final quantized sentiment data as defined in section 5.4
- Outputs - defines the number of time steps ahead the prediction is required to be
- Model - defines the model's hyperparameters
- Reporting - defines the parameters concerning the reporting of model performance

The above was mentioned to explain what "sub-dictionaries" are in the definition of the optimisation design parameters.

6.3 Scoring Method and Model Limitations

To fully model a trading scenario, as described in appendix 4.5, the trading bot would have to consider the following factors:

- forecasting the bid price
- forecasting the ask price

Section & Name	**Description**	**Values**
Sentiment - Topic Qty	Controls the number topics created by the topic	5, 9, 13, 17, 25
Sentiment - Topic Model Alpha	Controls the alpha variable used when generating topic models. Roughly translates to the average number of topics per cluster. The module used also controls the beta variable through the alpha	0.3, 0.7, 1, 2, 3, 5, 7, 13
Sentiment - Relative Half-life (Seconds)	Controls the speed of a tweet's "weight decay". A high value means that a tweet's effect/weight on overall sentiment reduces quickly	180 (3 mins), 900 (15 mins), 7200 (2 hr), 25200 (7 hr)
Sentiment - Apply IDF	Controls if IDF (Inverse Density Frequency) is applied to words in the weighting of a tweet to a topic cluster	False, True
Sentiment - Topic Weight Square Factor	Each tweet's probabilistic weighting value is changed by a power factor of this value. Then each weighting is divided by the new sum total, returning the probabilistic weighting to equal one. This can allow the model to increase all the tweets supplied to be more strongly weighted to thier dominant topics. This can be used to increase the non-homogeneity of the input data	1, 2, 4
Sentiment - Factor Topic Volume	For "Multi-Topic" scoped designs, controls if either the sentiment of each topic, the volume interest of in each topic or both are included in the historical data used to train each model	0 : Sentiment Only, 1 : Sentiment and Topic Interest, 2 : Topic Interest Only
Model – Hidden Layer Sizes	Controls the size and quantity of hidden layers in the neural network	*See appendix F
Model – General Adjusting Square Factor	Controls the rate that input features are removed for the deep random subspace functionality	3, 2, 1, 0
Model - Estimator Alpha	The neural network's Regularisation value. Note Regularisation is needed for the deep random subspace functionality to work	1e-11, 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-4
Model - Lookbacks	Controls the number of time steps the model considers at once	8, 10, 15, 20, 25, 50
Model – Early Stopping Patience	Controls the number of epochs the model will train for without decrease in loss function before finishing training of ensemble model	0, 5, 7, 9, 12

Table 12: Description of adjustable model hyperparameters

- identifying windows of trading opportunity
- simulate forced liquidations
- lending and trading fees

However, in the various sources of available free historical stock market data, historical ask and bid prices along with historic spreads were not offered. Also, to avoid over-scaling the project the focus was centred on improving the use of sentiment analysis for the forecasting of price movements. For these two reasons the trading bot was simplified along the following assumptions:

- Long and short positions (bets) will be placed when the model predicts a % change in the stock larger than the "confidence threshold" (calculated as the percentile of estimated price jumps)
- Any long and short positions will be closed at the time step the prediction was made for
- Buying and selling can always happen at the close price
  - the definition of close price is the price of the stock at its last exchange on the market
  - this ignores the bid and ask prices and assumes that both equal the close price
  - this eliminates the spread and makes achieving a profit easier than in a real-life scenario
  - data for the bid and ask prices were not available
- Any profit/loss is proportional to the proportional change of stock value multiplied by the betting stake
- When opening a long/short the stake is proportional to the difference between the expected change in price and the "confidence threshold" multiplied by the original price for the stock (see equation 6.3)
- It is assumed that the trader has an unlimited account with the broker
  - this also removes the potential for forced liquidations

It should be noted some of the assumptions are because historical ask and bid prices were not available on free versions of the trading data available. Due to these assumptions, a positive profitability score may not translate to profitable trading in a real-world setting.

These assumptions lead to the following scoring system, which estimates the average profit per euro staked in a position with 1:1 leverage, without considering the spread or any trading fees/delays:

$$\text{profitability score} = \frac{\sum_{i=1}^{n}((p_1 - p_0) \times \text{betting stake}_i)}{\sum_{i=1}^{n} \text{betting stake}_i}$$

where:

- n is the total number of time-steps
- p_0 is the close price at the original time step
- p_1 is the actual close price at the predicted future time step

The betting stake is set as so:

$$BettingStake = \begin{cases} (p_e/p_0) - (0.5 * p_0 * Con), & \text{if } abs(p_e - p_0)/p_0 \geq Con \\ 0, & else \end{cases}$$

where:

- Con is the "confidence threshold" - the expected ratio increase required to lay a bet
- p_e is the estimated future close price

Notes:

- This approximates the average potential profit per trade
- The number of time steps forward that predictions are made is a parameter on the model and stays constant throughout a run
- Every time step where the condition for betting stake is met adds one to the value "quantity of bets placed"

- If a large downwards change is predicted, the betting stake value is negative, inverting the scoring equation and modelling a short position
- If there are multiple time steps with a large positive/negative prediction n steps ahead there will be a position is opened on each time step, all of which are to be closed on the time-step which they are predicting the value of
  - n being the parameter set for the time steps ahead for the prediction

7 Appendixes

7.1 Appendix A - Glossary

- System
  - refers to all the components as a whole, trains and tracks multiple instances of the model
- Model, Design(s)
  - an instance of statistical predictor, can refer to the internal design of the system
- Model Scope
  - refers to if the model uses sentiment data with topic clustering "multi-topic", sentiment data without topic clustering "no-topics" or without any sentiment data "no-sentiment"
- Prediction Horizon
  - the amount of time in the future required for the model to predict
- Model Context
  - refers to a group of models with the same prediction horizon and model scope
- Design of Experiments, DoE
  - the initial set of random designs generated to give the optimisation algorithm a wide range of diverse points to inform later design selection

7.2 Appendix B - Long and Short Trading

The scenario this model looks at is long and short position trading. Below is a brief description of the process, however more detailed information can be found at various online sources [14].

This is covered so that when the limitations of the model are described, their implication can be better understood.

Opening a long or short position (also known as longing and shorting) is a practice of placing a bet on whether a stock will go up or down in the future. It should also be noted that stocks can only be bought at the ask price (the lowest price on the market offered for that stock) and then can only be later sold at the bid price (the highest price someone else is willing to buy that stock at). The ask price is always higher than the bid price[15] and the gap between these two values is called the spread. Therefore, anyone who buys a stock would then need to wait for the bid price to overtake the original ask price at which they entered the market to avoid making a loss on that trade/position.

Should someone instantly open and close a long position they would in theory lose the value of the spread. The trader would likely lose a slightly different amount as there are lags in the system, the ask/bid prices are constantly changing, and they would need to factor whatever broker fees they have incurred.

Shorting is similar but in reverse, to give a brief explanation, the trader is effectively selling a quantity of a share without owning it at the current bid price. They would then owe their broker that quantity of that share, they are said to be "short X of that share", hence the name. This can be thought of like having an overdraft. The trader's objective is to then buy their missing quantity of the share at a lower price. So, if at a later point the ask price goes below that past previous bid price, they can "close" their position by buying the stock at a lower price, fulfilling their commitment to the broker and pocketing the difference. If they can't as the stock goes up in value, they will have to buy the stock all the same and swallow the resulting loss.

Long and short trading are popular as brokers provide leverage, this is a form of lending that sees the broker match a trader's long and short positions but in multiples noted by the leverage

[14] https://www.investopedia.com/ask/answers/100314/whats-difference-between-long-and-short-position-market.asp#toc-the-bottom-line

[15] otherwise those two orders would fulfil each other and cancel out

ratio, i.e. if someone is trading at 10:1 leverage, for every 1 stock they take a position on, their broker will take a position on another 9 on their behalf. When a position is closed the broker will pass any resulting profit/loss onto the trader, along with a small lending fee incurred for services and the capital required to buy the 9 stocks originally.

For example, say the bid price for Apple is at £100 and our trader places £22 long on Apple. They later close the position when the ask for Apple is £110, this means that out trader made a 10% profit of £2.2 (£22*10%). However, with say a 10:1 leverage, this same trade would profit £22 (ignoring fees). This works because while our trader placed a stake of £22, the broker bought another 9 times the original stake on the trader's behalf.

There are two interlinked hazards with leverage. Firstly, there is the possibility to lose more money than originally staked. In the previous example, should the stock drop by 20%, while the trader staked £22 on said stock with 10:1 leverage, they would lose £44 (£22*20%*10), more than originally staked. This leads into the second major risk of "forced liquidation". When trading a trader normally has a balance with a broker. When taking a position, the broker will be tracking the potential risk that at a future point, the trader's balance would not be able to cover any future losses for an open position. They therefore reserve the right to force open positions to be closed. This can be especially devastating as short-term swings in the market can force accounts with smaller balances to close their positions before the market self-corrects and returns the stock to its original value.

Short trading is riskier than "going long" for various reasons including the theoretically infinite potential for loss as there is no limit how high as stock can grow in value while a trader has a commitment to buy it back later. There is also the dangerous phenomenon of a short squeeze, this is where a large section of the market is shorting a stock, causing a large potential demand for the later purchase of said stock. However, if the stock in question has a low relative availability for purchase, then the lower ask orders (offers to sell) for the stock will disappear, leaving the trader in a position where only higher ask offers for the stock are available. This can have a runaway effect of sellers taking advantage of this situation by price gouging the "short sellers" (the trader with a short position). This increased potential loss can also force short sellers (depending on their broker argument) to face forced liquidations from their brokers, forcing them to close their positions, buying

their required shares at an inflated price, causing massive losses. This phenomenon is what drove the famous 2021 GameStop incident.

7.3 Appendix C - Neural Networks

Neural networks are based around the idea of trying to simulate a brain's neurons and how they work in concert within the human brain.

This is achieved by having a network of simulated neurons sending signals to each other. Similarly as in the brain, each neuron receives a signal input from various nodes upstream from it and outputs a signal to nodes downstream to it based on this.

The relationship between input and output of a node/neurons is referred to as the activation function. Below are some examples of the typical ones used. The shape of the activation function is generally selected before training and neurons using different activation functions can be combined in the same network.

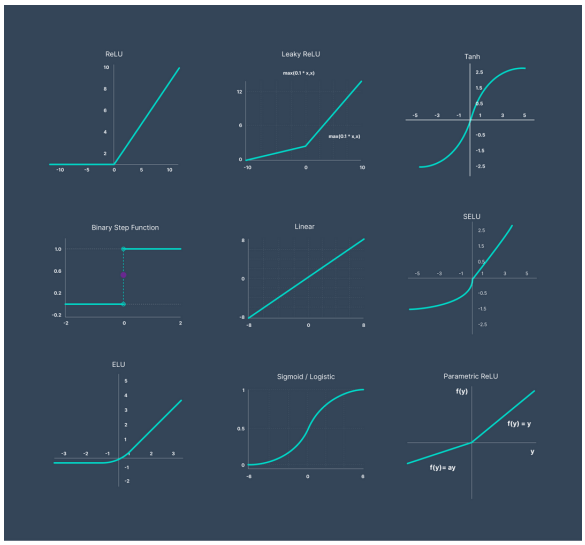

Figure 30: NN activation functions examples

Generally, neurons are arranged in "layers", with the neurons from one layer receiving a modified input from each of the neurons from the previous layer, as displayed in the image below:

In the above figure "Dense 1" is the first layer in the NN, the input to each neuron is a linear

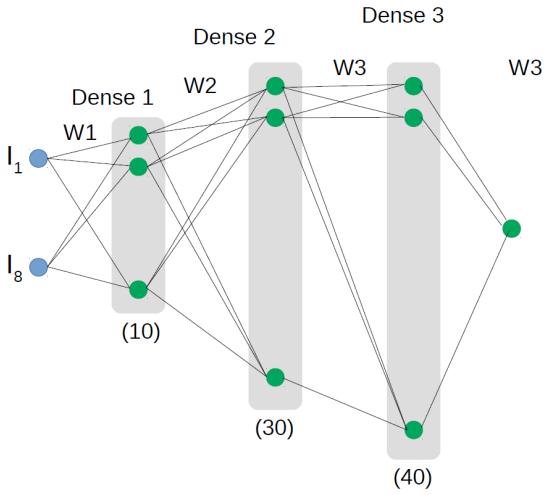

Figure 31: NN layers graphic
source: University of Studies Verona - Alberto Castellini

combination of the input values i_1 & i_8. Each neuron then calculates an output according to their input and then feeds that output to each neuron in the next layer, here labelled "Dense 2". Each neuron in this layer then makes a linear combination of each previous output to feed their own activation function and so-on until the output neurons, which are then fed to the user/whatever lies outside the neural network. For a single neuron this process can be expressed as:

$$x_j = f(\sum_{i=1}^{n} w_{ij}x_i + b_j) \tag{4}$$

Where: f = the activation function (fig 30) Indexes: i: node previous layer j: node in current layer Variables: w_{ij} = the weighting of the output from node i to the input of node j x_i = the output value of a given node i b_j = the bias of node j's input Output: x_j = the output value of the node in question (j)

This means that throughout the network the parameters w_{ij}, b_j and whatever parameters within the activation function can be tuned to the training set.

7.3.1 Back Propagation

Back propagation is the technique used to train the neural network. The objective of the process is to minimise the average error value of the NN::

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{5}$$

Where: X = the complete set of training input/output pairs (x_i, y_i) θ = the parameters within the NN y_i = the network's outputs $\hat{y}_i$ = the training outputs Here the error function is the root-sum-squared (RRS) although other functions can be used. For a given parameter the partial derivative between it and the mean error is calculated:

$$\frac{\partial E}{\partial p_i} = \delta_{jk} \tag{6}$$

Where: ∂E = change in overall mean error (for the dataset) ∂p_i = change in parameter i δ_{jk} = the parameter's "error"

This error is multiplied by a learning rate to give the recommended adjustment to the parameter overall for that "epoch":

$$\Delta p_i^k = -\alpha * \frac{\partial E(X, \theta)}{\partial p_i^k} \tag{7}$$

For multiple layers the chain rule is used to calculate the error derivatives of parameters a few layers back and the above process can be sped up by splitting the dataset into random cohorts and training the network on a different random subset for each cohort

7.4 Appendix D1 - Pseudocode: Sentiment Data Prep

The objective of the "Sentiment Data Prep" module is to produce a time series scoring the sentiment concerning a stock across a number of topic clusters. To achieve this, it has to follow these steps:

- Define a topic model
- Use that topic model to define a set of annotated tweets
- Use these annotated tweets to define a set of sentiment scores

See section Design - sentiment Data Prep for more detail.

Each of these steps are required to generate the following step, each step tends to be computationally expensive and the assets generated from each step can be reused. Depending on the input parameters coming from the experiment manager it is quite possible that some of the earlier assets can be reused in the generation of later assets, saving computation time, therefore the overall module functions as pictured in figure 32 Finally, all new assets are saved to potentially save computation

```
# SubModule - Sentiment Data Prep
def retrieve_or_generate_sentimental_data(input_dict):
    sentimental_data_path, annotated_tweets_path, topic_model_path = generate_filepaths()

    if os.exists(sentimental_data):
        # the sentimental time series data already exists
        sentimental_data = pickle.load(sentimental_data_path)
    elif os.exists(annotated_tweets):
        # the sentimental data can be generated from an existing annotated tweets file
        annotated_tweets = pickle.load(annotated_tweets_path)
        sentimental_data = generate_sentimental_data(annotated_tweets, input_dict)
    elif os.exists(topic_model):
        # the annotated tweets can be generated from an existing topic model
        # then the sentimental data can be generated
        topic_model      = pickle.load(topic_model_path)
        annotated_tweets = generate_annotated_tweets(annotated_tweets)
        sentimental_data = generate_sentimental_data(annotated_tweets, input_dict)
    else:
        # all assets need to be generated from this set of inputs
        topic_model      = generate_topic_model(input_dict, df_tweets)
        annotated_tweets = generate_annotated_tweets(annotated_tweets)
        sentimental_data = generate_sentimental_data(annotated_tweets, input_dict)

    # save all new assests to save time in future runs
    save_generated_assets(topic_model, annotated_tweets, sentimental_data)

    return sentimental_data
```

Figure 32: Example of the logic implemented for the generation and re-use of sentimental data

time should future runs share similar characteristics. Please note that in the source code the exact order of how the module is organised is different but has been rearranged here to be more readable.

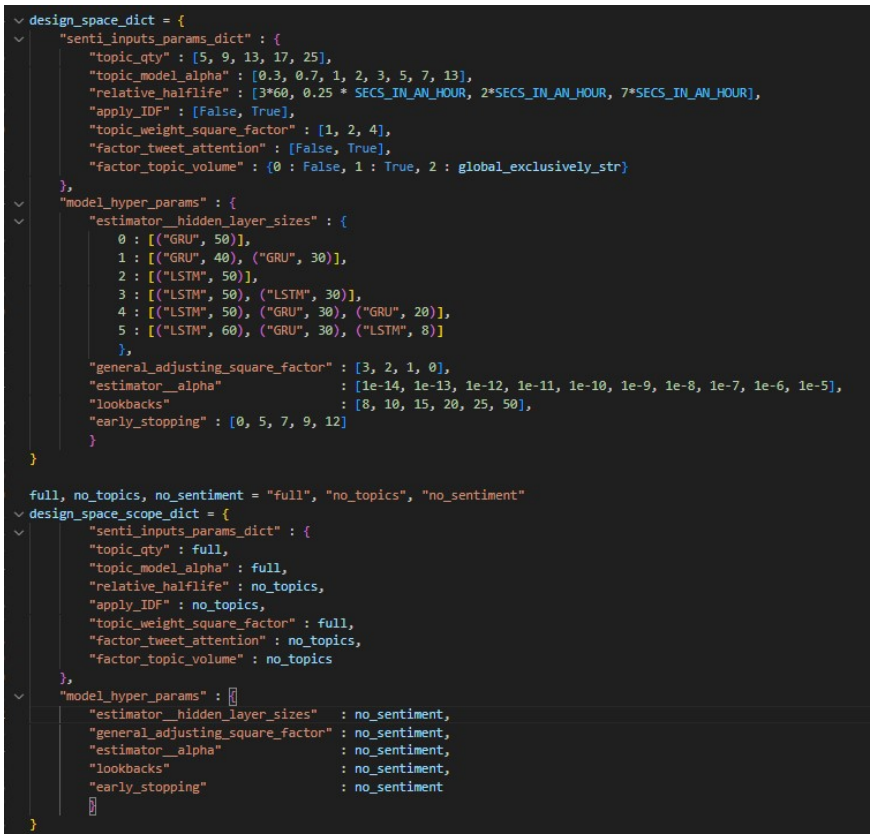

Figure 33: Mutli-layer dictionary, informing the model which variables can be changed for each new iteration of the model. The below dictionary explains which variables belong to which model scope

7.5 Appendix D2 - Pseudocode: Definition of the Design Space (Input Parameter Boundaries)

Figure 33 shows how the design spec for the baseline version of the model was defined. This allows the code to understand which areas of which input dicts to edit to affect the design. This is also used to configure the input to the GPyOpt's package for optimisation [16]. GPyOpt then can be used to define the design of experiments and the larger optimisation.

[16]See definition of GPyOpt objects at: https://www.blopig.com/blog/wp-content/uploads/2019/10/GPyOpt-Tutorial1.html

7.6 Appendix E - Model Improvements and Exploration

Below is a short section on some of the unseen work gone into this model to improve its performance, both by exploratory analysis and changes implemented. This list isn't exhaustive.

- Regularisation Tests
  - Given the large range of potential regularisation values early realisation that this would be a major factor in the general performance of the model, values for regularisation were tested for their effect on profitability score and MAE. These early tests showed that relatively high regularisation value could lead to high profitability scores lead to the regularisation values being allowed to range up to 1e-3
- GPU implementation
  - GPU were implemented for the calculation of these RNNs however this functionality had to be removed as the neural networks were not large enough to effectively leverage GPUs
- Sequence not crossing a day
  - The code was edited to ensure that an input tensor didn't straddle two days
- Financial data scaling
  - experiments to see the model's effectiveness with the financial data normalised in different ways were undertaken. These tests found that scaling all financial data relative (dividing by) to the opening price each day was the most effective. This means that the price at the start of each day would equal 1
  - The second most effective method was to convert all financial data to the price changes per time step. However this performed much worse than the previously mentioned method, suggesting that the RNNs could pick up better on the financial patterns better the previous way

ID	Composition
0	("GRU", 50)
1	("GRU", 40), ("GRU", 30)
2	("LSTM", 50)
3	("LSTM", 50), ("LSTM", 30)
4	("LSTM", 50), ("GRU", 30), ("GRU", 20)
5	("LSTM", 60), ("GRU", 30), ("LSTM", 8)

Table 13: Available configurations for the RNN layers

7.7 Appendix F - Different RNN Size Options

Below are the details for the different RNN layers and sizes available for the optimisation to select: Discussed in section 5.5.1.

7.8 Appendix G - Composite Index Maths

The composite indexes used in this paper returned a weighted average testing score, weighted by their corresponding validation scores, models with a better validation score were given a larger weighting. These weighting were then root-sum-squared to make lower scoring models drop in weighting more. The overall effect of this was to ensure an average testing score, made of only the top validation scores:

$$\text{composite\_index} = \sum_i \frac{\text{testing\_score}_i}{\text{validation\_weighting}_i}$$

However, the weightings from validations were scaled as so:

$$validation\_weighting_i = \left(\frac{x_i - \min(\hat{x})}{\max(\hat{x}) - \min(\hat{x})} \right)^3$$

where: x_i = is an individual validation score x_hat = is the collection of other validation scores

The normalisation and subsequent cubing, ensured that the composite index was overly weighted on the larger validation scores as the assumption was that the lower validation scoring designs would be partically ignored

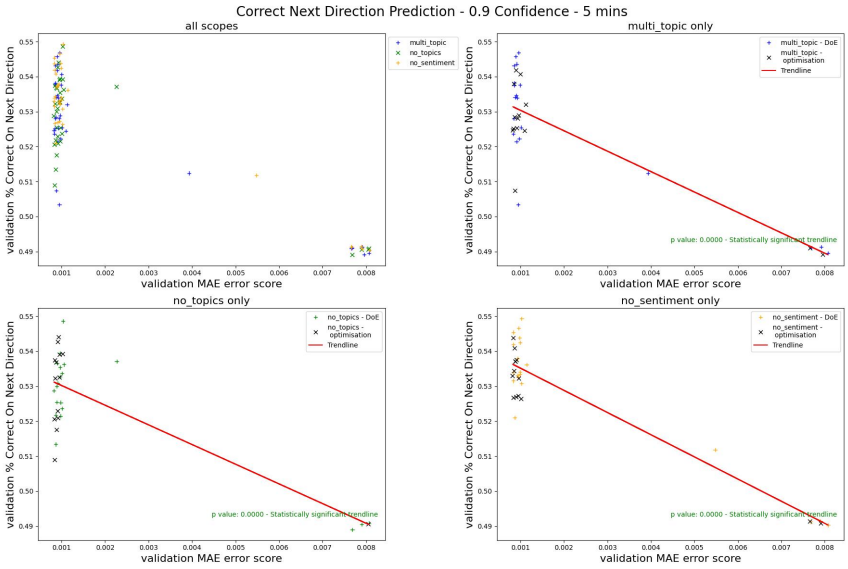

Figure 34: Proportion of time that the model would give the next direction of stock movement correct, predictions 10% of the time, 5 minutes prediction horizon

7.9 Appendix H - Complete Correct Next Prediction Results

Figures 34 & 35 show the proportion of time that the model would predict the next direction of stock movement correctly. This was for both the 5 and 30 minute prediction horizons and shows the score when predictions were taken 10% of the time (described as 0.9 confidence).

These figures show that the high MAE error score clusters are also creating low scoring designs, skewing the analysis.

7.10 Appendix I - Example of Model Variability

Table 14 is an example of three design runs with very similar design parameters and the variation between them. The only difference is a slight difference between predictor L1 regularisation, which forced the system to regenerate the RNN for each instance.

This table shows that for these 3 example designs there is a 12.3% and 19.9% variation for the validation and testing profitability score respectfully, a significant amount.

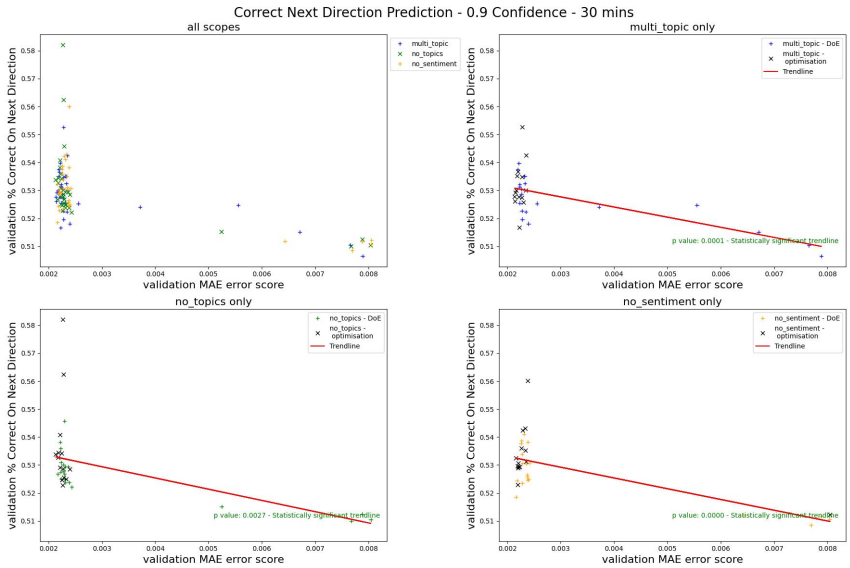

Figure 35: Proportion of time that the model would give the next direction of stock movement correct, predictions 10% of the time, 30 minutes prediction horizon

	L1 Regularisation	Validation Profitability	Testing Profitability	Validation MAE	Testing MAE
0	1.00E-14	3.26E-03	3.22E-03	2.19E-03	2.17E-03
1	1.01E-14	2.91E-03	2.69E-03	2.19E-03	2.10E-03
2	1.02E-14	3.18E-03	3.12E-03	2.18E-03	2.12E-03
Range		3.57E-04	5.35E-04	6.03E-06	6.91E-05
% increase from min to max		12.3%	19.9%	0.3%	3.3%

Table 14: Below is an example of three design runs with very similar design parameters and the variation between them

7.11 Appendix J - Typical Spread Estimation and its Effect on Profitability

Spread is the difference between the bid and the ask price, this difference must be overcome with longing or shorting a stock (see section 7.2 for more information). Unfortunately historical data for the spread was not available for this experiment. However at the time of writing (28/05/24) nasdaq.com reported that the current spread on apple was $0.03 on a bid price of 191.47, meaning that the effect on profitability would be 1.56E-04 (calculated as the spread divided by the price). While this value would be variable moment by moment, the scale of this value gives use the confidence that it would have a small effect of the profitability scores posted in this paper.

The spread in volatile trading conditions could be different from this and more data/research/expertise would be needed to be collected on this before making a final statement

References

[1] source of financial data used. URL: https://firstratedata.com/.

[2] source of twitter data used. URL: https://www.kaggle.com/datasets/omermetinn/tweets-about-the-top-companies-from-2015-to-2020?select=Tweet.csv.

[3] source of financial charts. URL: https://finance.yahoo.com/quote/AAPL.

[4] Lin Lu Bin Weng. *Predicting short-term stock prices using ensemble methods and online data sources*. June 2018. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417418303622.

[5] Markus K Brunnermeier. *Buy on Rumours - Sell on News: A Manipulative Trading Strategy*. Dec. 1998. URL: https://www.researchgate.net/publication/5055159_Buy_on_Rumours_-_Sell_on_News_A_Manipulative_Trading_Strategy.

[6] J Deveikyte. *A Sentiment Analysis Approach to the Prediction of Market Volatility*. 2022.

[7] Trang-Thi Ho. *Sentiment Analysis and CandleStick Chart Representation*. Nov. 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8659448/#sec3-sensors-21-07957title.

[8] Chetna Khanna. *Text pre-processing: Stop words removal using different libraries*. URL: https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a.

[9] Venkata Sasank Pagolu. *Sentiment Analysis of Twitter Data for Predicting Stock Market Movements*. 2016. URL: XXXX.

[10] Hercules Prado. *On the Effectiveness of Candlestick Chart Analysis for the Brazilian Stock Market*. Oct. 2013. URL: https://www.sciencedirect.com/science/article/pii/S1877050913009939.

[11] Wang Qili. *Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles*. July 2018. URL: https://www.sciencedirect.com/science/article/abs/pii/S0925231218303540.

[12] Chih-Fong Tsai. *Stock Prediction by Searching for Similarities in Candlestick Charts*. July 2014.