Evaluating the Effectiveness of Machine Learning and News Sentiment Data in the Prediction of Bear Market Periods of the S&P 500

Computer Science with Data Analytics MSc

University of York

Domantas Vaicys

February 2023

Word Count: 8797

Acknowledgements

I would like to thank my project supervisor, Mominul Ahsan, for suggesting a valuable structure for this report, answering my questions and offering his wisdom to keep the project on track. I would also like to thank every person in my life with whom I've had interesting conversations as they have helped form my understanding of the world and make me who I am today.

Executive Summary

Bear markets are long periods of falling stock market values and investor confidence. Besides investor losses, their effects reverberate through the wider economy, making them of interest to both investors and policymakers. Yet, the few studies predicting this phenomenon have used older logistic regressions and historical price and macroeconomic data. This research gap is tackled by evaluating the effectiveness of using newer machine learning models and news sentiment data in predicting bear markets. XGBoost and neural networks have seen excellent performance in the computer science literature. Therefore, alongside random forests, support vector classifiers and logistic regressions, these were evaluated across varying data sources.

Logistic regressions were only optimal with simpler datasets. XGBoost was the most generalisable and applicable; only it had above-average predictive accuracy across all prediction time-periods and the best investment returns across all combinations of datasets. The best investment strategy, using XGBoost trained on a diversified quantitative macroeconomic and qualitative news sentiment dataset, reached 88.8% accuracy and generated 14.1% annual compound interest, greatly outperforming the received wisdom of buy-and-hold which generated an annual compound interest of 7.9%, challenging the efficient market hypothesis. Thus, predicting bear markets is both possible and profitable, with the benefits of machine learning models and diversified data sources made clear. The results are of obvious interest to investors, but policymakers are also encouraged to take these findings seriously, otherwise, automated trading strategies selling ahead of a bear market risks their quicker arrival.

Contents

List of Tables/ Figures

(Page numbers listed on the left-hand side)

1. Introduction

Stock markets are central to modern economies. They are a means for individuals to accumulate wealth and for society to efficiently allocate scarce capital. Both outcomes are improved through better understanding of stock market behaviour. Yet, while the literature overwhelmingly focuses on short-term day-to-day price fluctuations, this study evaluates the utility of new machine learning (ML) techniques and news sentiment analysis in predicting longer-term ‘bear markets’. Stock markets are underpinned by two long-term states: bull markets and bear markets. During bull markets investor confidence is high with prices generally rising[1]. During bear markets investor confidence is low with prices generally falling, resulting in decreasing individual wealth and less capital being efficiently allocated.

A longer-term focus benefits more parties than just those benefiting from short-term noise such as full-time traders and hedge funds. Two particularly stand out. Firstly regular investors, with less time for daily trading, may profit from superior market timing strategies over the received wisdom of buy-and-hold[83]. Secondly policymakers, for whom day-to-day stock price fluctuations have little bearing whereas long-term decreases in investor confidence reverberate throughout the wider economy, may make better policy responses to stock markets[2,3]. Predicting bear markets requires finding patterns in longer-term, subtler phenomena. Predictions using slow-moving macroeconomic forces have been successfully attempted, albeit with older econometric techniques[4,5]. Two novel approaches are to integrate new ML developments and qualitative news sentiment data.

The inspiration of this study, the stock market's gyrations during the Covid-19 pandemic, can be explained through sentiments. Despite earlier warnings in newspapers of an exponential virus, it took weeks for fear to reach a critical mass[6]. Once it did, investor confidence fell rapidly and the S&P 500 lost 32% of its value. Within a few months, Western governments pumped enough stimulus into the economy to boost macroeconomic stability and investor sentiment, making the S&P 500 overtake its previous peak. This study thus hypothesises that by also analysing news sentiments with newer ML models, bear markets could be predicted with greater effectiveness, allowing policymakers to pre-empt the rapid declines in investor confidence and asset values that define bear markets.

1.1. Research Questions (RQ)

The RQs flow from the hypothesis to explore a series of related themes. The first two aim to improve bear market predictability across two dimensions: model type and data type. The third tests whether the theoretical results translate into real world benefits.

1. How does the performance of newer machine learning models compare to the state-of-the-art logistic regression models for bear market prediction?
2. What is the most effective combination of data sources from historical price data, macroeconomic data and news sentiment data for predicting bear markets with machine learning?
3. Can a market timing strategy based on the best performing bear market prediction model(s) generate higher returns than a buy-and-hold strategy?

1.2. Aims and Objectives

The aim of the research is to find the optimal model type and data sources for a bear market prediction model. 5 objectives work towards achieving this:

1. Identify the research gap: Conduct a literature review of bear market prediction and related topics to integrate findings and specify the research gap. This ensures the objectives of the study fill this gap.

2. Collect and clean S&P 500, macroeconomic and news sentiment data into three time series: The price history of the S&P 500, the IMF's US International Financial Statistics database and the Economist's back-catalogue of news articles are used, with the most meaningful feature sets elicited through feature creation and reduction. Three combinations of data sources are built to answer RQ2.

3. Train ML models on the data sources to predict bear markets: Logistic regressions, support vector classifiers, random forests, XGBoost and neural network models are trained on each of the dataset combinations.

4. Evaluate the models to find the most effective data source and model type combination: Model testing using time-split cross-validation allows classification metrics to uncover the most effective data source and model type combinations

for bear market prediction. By evaluating any additional effectiveness of newer ML models and sentiment analysis, this answers RQ1 and RQ2.

5. Undertake profit analysis using the best model: The theoretical results of the best model combination are tested for their real-world profitability by undertaking an investment strategy based on the model's predictions that will be put against the received wisdom for stock market investment (buy-and-hold) to answer RQ3.

1.3. Scope of Work

Since bear market prediction is under-explored, this study is more foundational. By applying the broadest range of data sources and model types yet, it uncovers the most promising future research paths. However it is ultimately limited by the data sources and the breadth of models tested. Future research could explore more specialised ML techniques related to the best-performing ones identified in this study or incorporate additional data sources such as geopolitical events or social media sentiment data.

1.4. Contribution

This work illuminates the under-explored area of bear market prediction, with a series of contributions to the topic:

1. First use of news sentiment; current published research only uses macroeconomic and historic price-derived variables.

2. First application of support vector machines, random forests, XGBoost and neural networks; past literature has only used econometric techniques.
3. Clarifies the relative importance of different classification metrics in profit analysis.

1.5. Chapter Outline

Chapter 2 completes objective 1 by summarising the relevant literature and identifying the research gap. Chapter 3 outlines the methodology from data collection and preprocessing, achieving objective 2, to model training, testing and evaluation, achieving objective 3. Chapter 4 presents the results and discusses them to answer the RQs, achieving objectives 4 and 5. The paper finishes with a conclusion in chapter 5.

2. Literature Review

The review is organised into thematic sections. Firstly, stock market theory explains the functioning of stock markets, showing the validity in predicting them. Next, a history of stock market prediction starts with econometric techniques but gets superseded by better-performing computer science techniques such as neural network models. Then, diversification of data sources is discussed, as this provides additional performance above changing model types since ML outputs ultimately depend on their input data. Lastly, we cover bear market prediction and summarise the research gap.

2.1. Stock Market Theory

Stocks specify fractional ownership of a company[7]. Since companies' present values are based on future values which are unknown, the stock market represents investors' collective expectations on company values[8]. Theoretically, overvalued stocks have their supply exceed demand causing falling prices, and vice-versa for undervalued stocks[9]. As the first overarching theory on the functioning of stock markets, the Efficient Market Hypothesis (EMH) influences most stock market studies. It proposes that the entire market considers *all* publicly available information, thus stocks are always priced at the closest possible to the company's true underlying value (i.e. priced efficiently)[10]. EMH's consequence is the unpredictability of future stock prices since they are only changed by new (future) information which is necessarily unknowable. EMH helps explain why stock markets are notoriously difficult to predict, as any newly-discovered methods are quickly integrated into stock prices[11].

If EMH were absolute, professionals would have no incentive to uncover the information that gets integrated into market prices, yet a lucrative industry exists[12]. Further, behavioural economics has proposed psychological explanations for human behaviour that undermine the rational assumptions underpinning EMH, such as periods of mass irrationality and a tendency to underreact to new information, with the latter being empirically proved[11,13,14]. Stock market studies frequently predict future returns (profits) significantly above EMH's conclusions, making them evidence against the theory (sections 2.2-2.3). Thus, EMH has gathered increasing counter-evidence, making stock market prediction valid.

2.2. Econometric Stock Market Prediction

Econometrics, the fusion of statistical techniques with economic theory, underpinned the first stock market prediction attempts. Their performances were grounded by profit analysis which measures the returns generated by an investment strategy based on a model's predictions. Simple moving average trading rules, and later regression techniques integrating financial ratios, generated returns exceeding those implied by EMH[15,16,17]. However, in parallel, computer science developed sophisticated ML techniques with less restrictive assumptions than statistical models[18]. These were integrated into stock market prediction, consistently showing superior performance, to which the discussion moves[19].

2.3. Machine Learning Stock Market Prediction

2.3.1. Neural Networks (NN)

Early applications of NNs outperformed linear econometric models, with for instance daily directional changes in the Brazilian stock market predicted to a 93.62% accuracy[20,21]. Stock market prediction literature reviews have found NN performances to be mostly dominant[22,23]. Increasingly specialised NN techniques are continuously being developed and successfully applied. For instance, Long-Short-Term Memory NNs predict multiple values simultaneously, whereas Knowledge-Driven Temporal Convolutional NN can react to rapid structural changes in data, such as switches between bull and bear market states[24,25].

However, without large datasets NN generalisability suffers. Therefore, sample sizes 50 times larger than the number of parameters are recommended, limiting NN applications[26]. Thus, despite the general dominance of NNs in daily stock market prediction, other ML techniques may be more suitable given the reduced sample sizes in bear markets prediction.

2.3.2. Support Vector Machines (SVM)

Some studies show SVMs outperforming NNs in stock market prediction especially using adaptive parameters, albeit in comparison with basic back-propagation NN models which are no longer cutting edge[27]. SVMs benefit from structural risk minimisation; whereas most models just minimise the training error, SVMs additionally minimise the confidence interval[28]. However, with high-dimensional datasets, SVMs

tend to overfit, reducing generalisability[29]. Therefore, best practice involves feature reduction to create a feature set uncorrelated with each other but highly correlated with the output.

2.3.3. Random Forest (RF)

RFs applied to stock market prediction occasionally outperform NNs, but can be beaten by SVMs[30]. Whilst the authors attributed this to the structural risk minimisation principle, limited parameter tuning for NNs may have undermined their performance. RF is computationally efficient and generalisable, allowing a huge number of trees in a model[31]. 500 trees are commonly used as errors often stabilise before then[32].

2.3.4. XGBoost

XGBoost is a relatively new technique partially related to random forests that has seen excellent performance on many standard classification benchmarks[33,34]. XGBoost has been applied to stock market prediction, beating the performance of EMH's suggested buy-and-hold returns by 49.26% to 32.41%[35].

Ultimately the studies covered in 2.3 show two themes:

1. ML models outperform statistical models
2. Best performing models vary by study; no model type is best in absolute terms

Theme 2 is unsurprising given the variability of data, variables and parameter tuning opportunities. This highlights the importance of using a diversity of models and carefully undertaking parameter tuning to uncover the best model for the given task.

2.4. Sentiment Analysis

Different model types can generate superior predictive performance, but so can different data types. Unlike macroeconomic data, sentiment analysis is a more recent development due to its expensive computation. As company information enters public knowledge, stock prices change to better match reality. However, publicly available information may not cover a company's entire production activities leading researchers to theorise that quantifying language sentiments may provide additional information about fundamental company values[14].

Sentiment analysis can be split into three categories. Firstly, unsupervised approaches have models learn word associations by mapping them onto a vector space. However, high dimensionality and unrealistic complexity, with the breakthrough study using a multi-billion word dataset, makes this technique infeasible[36,37]. Secondly, a middle ground technique manually marks text sentiments in a small dataset and then uses automated methods to extend the dictionary, e.g. SenticNet 5[38,39]. However, to limit the study's scope, the literature review and study will focus on the final approach: bespoke and nuanced linguist-made dictionaries, which fortunately tend to be the most effective of the three categories for stock market prediction[40].

Man-made dictionary techniques have successfully predicted daily stock price movements[41]. The results follow intuition, with one study using the commonly-used Harvard IV-4 Sentiment Dictionary finding that negative (positive) news sentiments create permanent decreases (increases) in unexpected stock returns[14]. However, 73.8% of ‘negative’ sentiment words in the dictionary do not necessarily have negative connotations in economic or financial contexts (e.g. ‘cost’, ‘liability’ and ‘tax’), which leads to increased noise and a muddying of sentiment relationships to outcome variables[42]. Thus the specialised Loughran-McDonald Financial Dictionary was created, which greatly outperformed other dictionaries (including Harvard IV-4) when applied by unrelated researchers in processing WSJ reports to predict stock returns[43,14].

However, news sentiment should be integrated with other data sources as it lacks market context. To illustrate, take a stock with a strong upward trend. A negative news article may not decrease the price of the stock, but merely slow its upward trend. Studies combining news sentiments with historical price data found superior performance over any single source of data[44,45]. Even Twitter sentiment data improved stock price change predictive accuracies[46]. However, the referenced studies all used under a year’s worth of data, thus not accounting for seasonality, nor likely fundamental market conditions (i.e. bear/bull states) to change. A study which expanded the time period found sentiment’s predictive power to be insignificant[47]. Another which increased data source diversity by comparing social and traditional

media found mixed results[48]. Thus, a new approach is particularly valuable as the predictive power of sentiments on the stock market remains inconclusive.

2.5. Longer-Term Stock Market Studies

Every mentioned study hitherto focused on short-term day-to-day price changes. This overwhelming focus in academia is puzzling as a longer-term angle benefits more parties (section 1). Bear and bull markets are a natural framework to examine longer-term stock market trends. Bear markets are periods of falling stock prices and investor sentiment, whereas both rise during bull markets[49]. Whilst studies examined bear/ bull periods *retroactively*, for instance to uncover consistent quantifiable rules for past market states, the literature review has uncovered only two studies attempting to *predict* bear and bull markets[50]. This emphasises a glaring gap in the literature.

The first study demonstrated that macroeconomic variables, particularly yield spreads and inflation rate, significantly predict bear markets[4]. Investment strategies using a single relevant macroeconomic variable greatly outperformed a basic buy-and-hold strategy. The same authors revisited the topic a decade later with a multivariate regression model, finding more macroeconomic variables indicating bear market predictability[5]. Despite logistic regression being the most advanced prediction model used, the two studies demonstrate the real-world significance and technical feasibility of predicting bear markets.

2.6. Research Gap

Stock market prediction has overwhelmingly focused on daily fluctuations. The two attempts at bear market prediction used older econometric techniques and only added macroeconomic data[4,5]. Therefore bear market prediction has many opportunities to integrate the computer science developments applied to daily stock market prediction (sections 2.3 and 2.4). This paper tackles two gaps by using four new non-linear ML models and the additional data source of news sentiments. With the research gap specified, objective 1 of this paper is met.

3. Methodology

A mixed methods methodology is employed. As an intervention (a bear market prediction model onto stock market investment) with numerically-measured results, it is mostly quantitative[51]. However, the methodology integrates yet-unused qualitative data into bear market prediction. It converts qualitative human-written news articles into quantitative variables via sentiment analysis for use in a quantitative model.

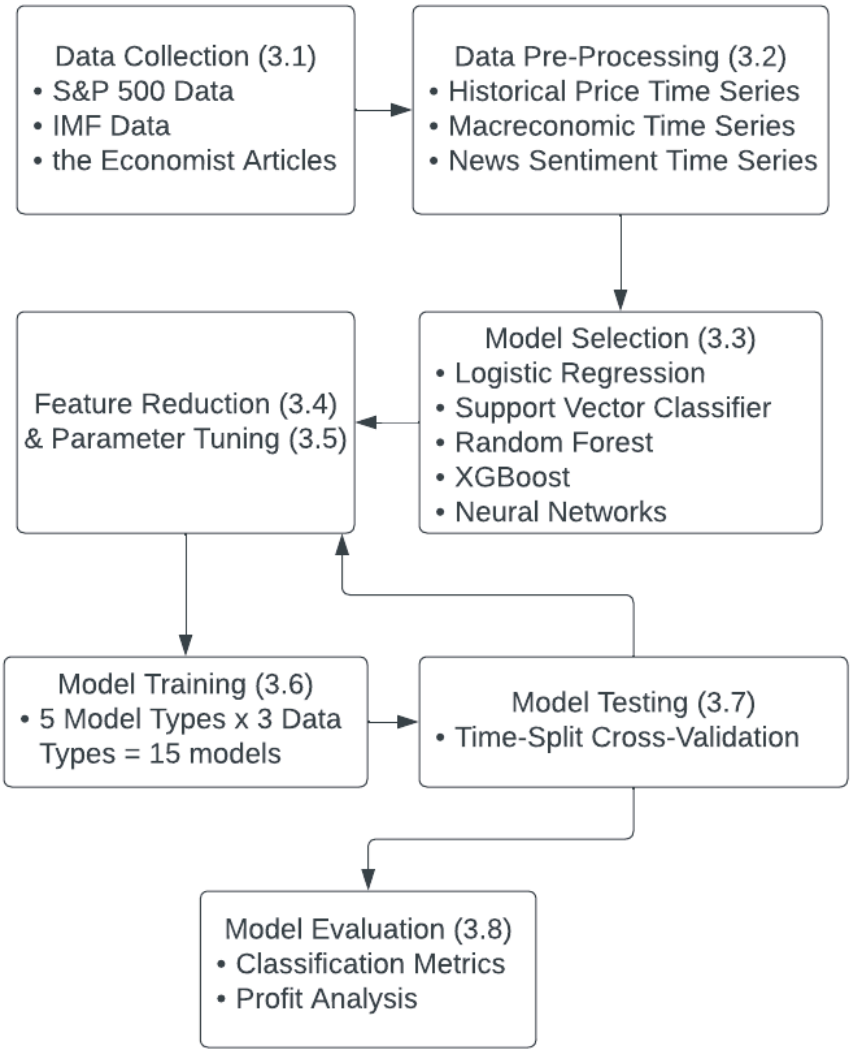

Figure 1: Flow of methodology

The workflow shown above is mirrored in this section's structure. Sections 3.1 and 3.2 will iterate through the creation of the historical price, macroeconomic and sentiment time series individually and achieve objective 2. Sections 3.3-3.7 achieve objective 3 and section 3.8 evaluates the resultant models to achieve objectives 4-5.

3.1. Data Collection

3.1.1. Historical Price Data

Historical price data were downloaded directly from Yahoo! Finance's S&P 500 daily history going back to 1998; the earliest availability of the Economist's online news archive[52]. The S&P 500 was chosen to keep consistency with most stock market index prediction studies.

3.1.2. Macroeconomic Data

Macroeconomic data were downloaded from the IMF's International Financial Statistics database, keeping consistency with past bear market studies[53,4]. The IMF is well-respected and international, meaning their data is at less risk of domestic politicisation, unlike, say, the Federal Reserve's database. As the home of the S&P 500, all US macroeconomic data were downloaded.

3.1.3. News Sentiment Data

News sentiment data were downloaded from the Economist for three reasons[54]. Firstly, as a well-respected newspaper with an economic slant, the Economist is likely more relevant to the stock market. Secondly, by focusing on analysis rather than mere reporting (e.g. the Financial Times), the Economist may be more effective for predicting *future* behaviour of the stock market. Thirdly, the website is well organised with their back-catalogue of articles going back to 1998. Combining more sources is left to further research. A crawler was built which traverses their archive to extract article URLs in their Business, Finance & Economics, Leaders and United States sections(Appendix B). GET requests to the HTTP pages associated with the URLs downloaded the article names, publishing dates and contents(Appendix C). Each section's data were saved into four separate sqlite3 tables[55].

3.2. Data Pre-Processing

3.2.1. Historical Price Pre-Processing

Given the single news source, articles were organised into weekly format to reduce noise. Thus, the S&P 500 daily dataset was also converted into a weekly format by finding the closing price for each Friday. However, null bank holiday Fridays were replaced with the most recent available values, generally the preceding Thursday(Appendix D).

The S&P 500 has grown fourfold in price over the study's period. This non-stationarity means comparing absolute weekly price differences in 1998 to 2022 may bias the

model towards undercounting the relatively smaller earlier differences. To solve this, closing prices were replaced with the relative differences in the percentage of price(Appendix E)[27].

The study's dependent variable is bear market state, not weekly price movements. Yahoo! Finance's dataset does not specify bear markets, nor is there a universal definition[1]. Yardeni Research defines bear market and correction periods as a 20% and 10% decline peak-to-trough respectively[56]. Both declines significantly indicate declining stock prices and investor confidence. Yet, only 4 periods declined over 20% while 13 declined over 10%. More bear periods allow the training of a better model, thus, this study defined a bear market period as any contraction over 10%(Appendix F). Figure 2 clearly indicates significant losses across all identified bear market periods.

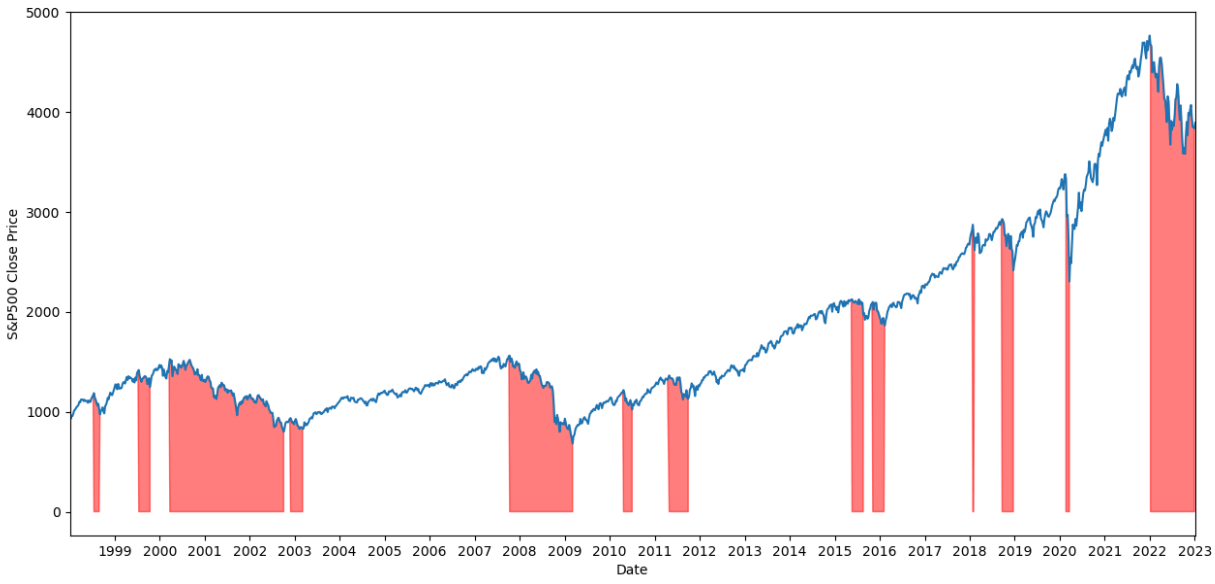

Figure 2: S&P 500 close price over study period, bear market periods marked red

A class imbalance problem, when models bias towards the statistically more frequent majority class, arises when the majority class dominates the samples[57]. 370 out of 1297 samples were bear periods. A 29/71 split is not strongly imbalanced, thus, undersampling or oversampling were not undertaken to avoid removing valuable data or generating synthetic data.

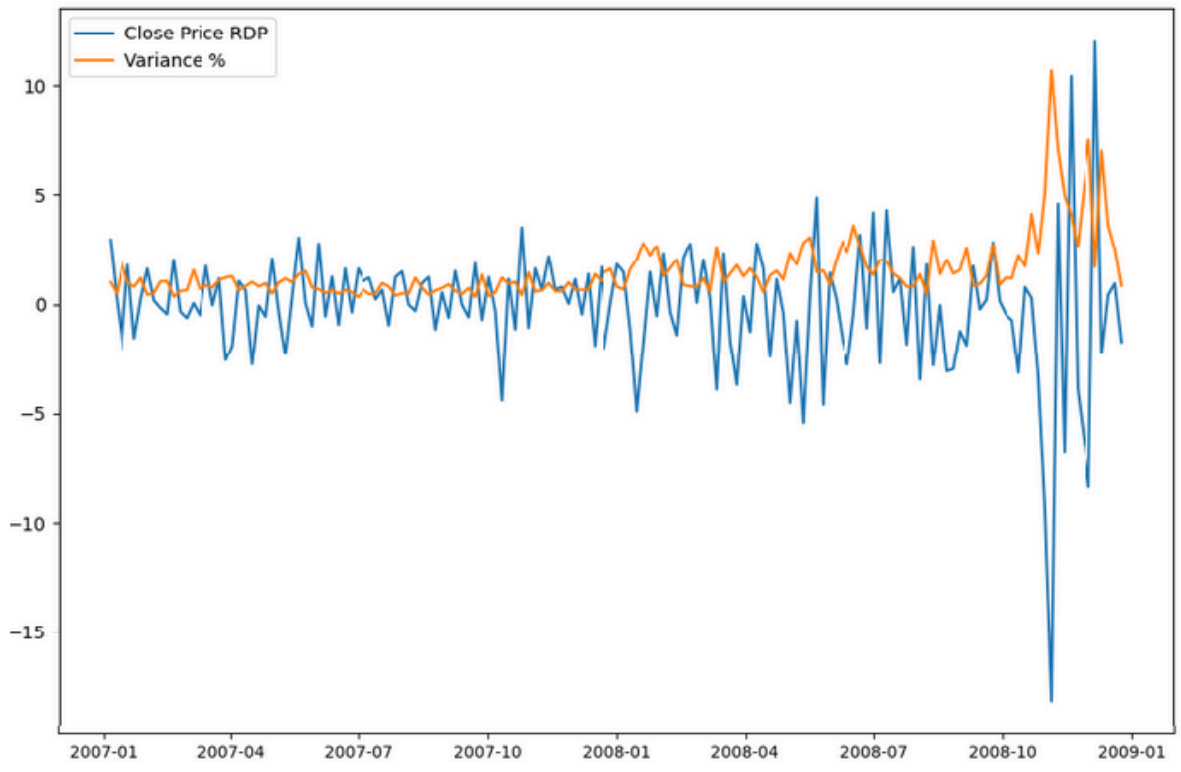

Figure 3: Weekly relative difference in price and price variance, 2007-2008

Figure 3 shows the noisiness of price data over a two-year sample. The effectiveness of a moving average of the previous four periods, as commonly undertaken in

econometrics, in reducing noise and emphasising the longer term trends is shown below(Appendix G)[58].

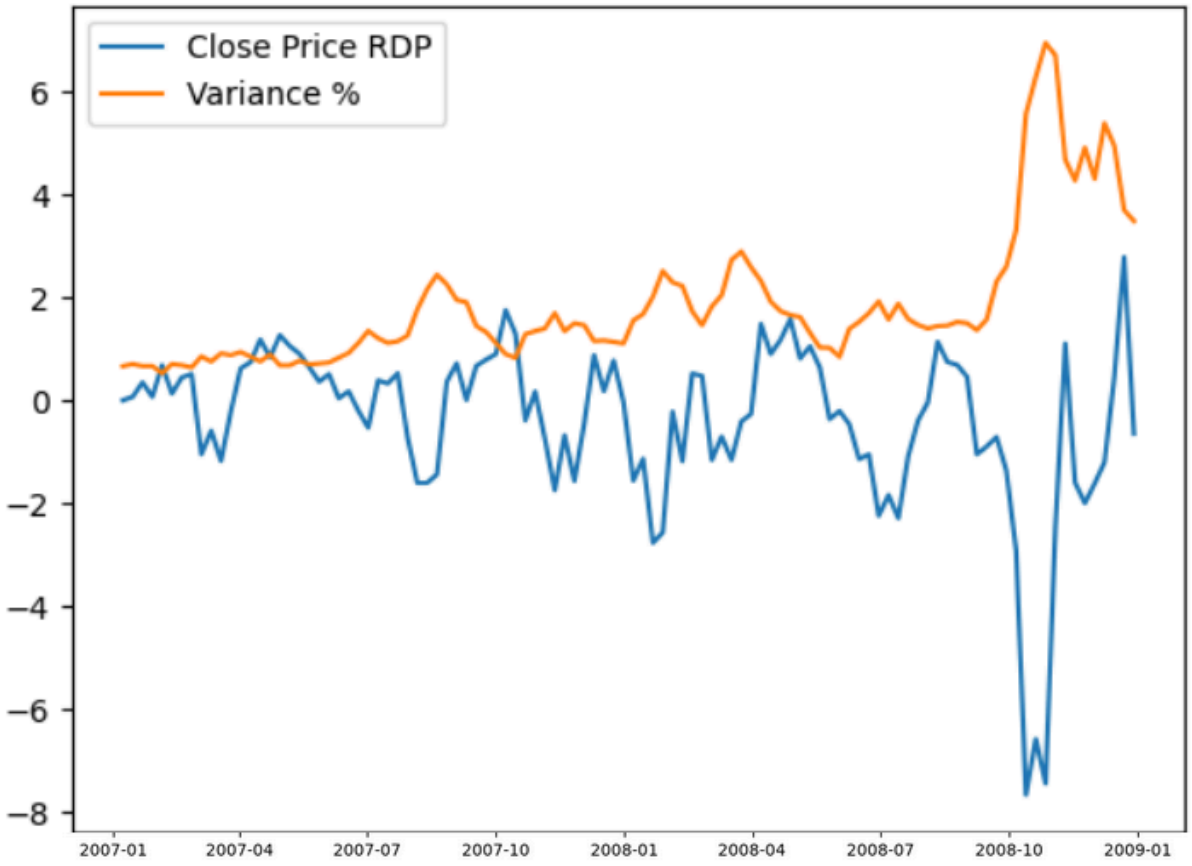

Figure 4: Figure 3, post-smoothing

Next, several features were created. Stock markets show strong seasonality. Dummy variables for winter, spring and summer accounted for this. The variance in prices within a period as a percentage of closing price were also calculated. Models evaluate a sample at a time, but price data has value in its history. Thus, varying length lagged values were created for closing prices, trading volumes and bear market

states(Appendix H). Bear market states were lagged a minimum of one month to ensure validity with the dataset whose shortest defined bear market is two weeks. The results of pre-processing are shown below.

	Date	Close	Volume
	Filter	Filter	Filter
1	1998-01-02	975.04	366730000.0
2	1998-01-09	927.69	746420000.0
3	1998-01-16	961.51	670080000.0
4	1998-01-23	957.59	635770000.0
5	1998-01-30	980.28	613380000.0
6	1998-02-06	1012.46	569650000.0
7	1998-02-13	1020.09	531940000.0
8	1998-02-20	1034.21	594300000.0
9	1998-02-27	1049.34	574480000.0
10	1998-03-06	1055.69	665500000.0

Figure 5: Original historical price data

	Date	Bear20	Bear10	Variance_P	Close_RDP	Volume_RDP	Winter	Spring	Summer	Bear10_1M	Bear10_3M	Bear10_6M
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1998-01-02	0	0	0.95	NULL	NULL	1	0	0	0.0	0.0	0.0
2	1998-01-09	0	0	2.325	-4.86	103.53	1	0	0	0.0	0.0	0.0
3	1998-01-16	0	0	2.05	-0.605	46.65	1	0	0	0.0	0.0	0.0
4	1998-01-23	0	0	1.945	-0.54	29.3933333333333	1	0	0	0.0	0.0	0.0
5	1998-01-30	0	0	1.905	0.1875	21.165	1	0	0	0.0	0.0	1.0
6	1998-02-06	0	0	1.22	2.2225	-6.5	1	0	0	0.0	0.0	1.0
7	1998-02-13	0	0	1.0025	1.4975	-5.5975	1	0	0	0.0	0.0	1.0
8	1998-02-20	0	0	0.8725	1.945	-1.3875	1	0	0	0.0	0.0	1.0
9	1998-02-27	0	0	0.8475	1.7175	-1.3425	1	0	0	0.0	0.0	1.0
10	1998-03-06	0	0	1.0975	1.05	4.4	0	1	0	0.0	0.0	1.0

Figure 6: Snippet of pre-processed historical price data

3.2.2. Macroeconomic Pre-Processing

Given the varying difficulties in generating macroeconomic data, the IMF's macroeconomic variables vary in their frequencies of observations from monthly to yearly. This creates missing rows of data since price and news sentiment data have a weekly frequency. Thus, macroeconomic variables were extended until their next observation, e.g. GDP for every week in 2008 January-March was set to 2008Q1's value, keeping consistency with EMH stating stocks are priced given the most recent available information(Appendix I). Data extension was chosen over data smoothing in order to prevent the inclusion of future information, which is obviously impossible in reality[19].

Macroeconomic data has a decreasing significance of absolute differences over time as economies grow and money becomes less valuable. Thus, under the same logic as price data, most variables (with exceptions like interest rates) were converted into relative percentage differences. The effects of pre-processing are shown below:

Indicator			Scale	Base Year	1998	1998Q1	1998M01	1998M02	1998M03
Prices					...	...	...	...	...
Financial Market Prices, Equities, Index	Index	FPE_IX	Units		80.89	77.66	73.23	78.06	81.68
Prices, Producer Price Index, All Commodities, Index	Index	PPPI_IX	Units	2010=100	67.36	68.05	69.52	69.29	69.13
Prices, Consumer Price Index, All items, Index	Index	PCPI_IX	Units	2010=100	74.76	74.25	74.11	74.25	74.38
Production					...	...	...	...	...
Economic Activity, Oil Production, Crude, Index	Index	AOMPC_IX	Units	2010=100	...	...	...	...	...
Economic Activity, Industrial Production, Manufacturing, Index	Index	AIPMA_IX	Units	2010=100	...	...	...	...	...
Economic Activity, Industrial Production, Index	Index	AIP_IX	Units	2010=100	93.63	92.50	91.43	92.65	93.50
Industrial Production, Seasonally adjusted, Index	Index	AIP_SA_IX	Units	2010=100	93.63	92.45	92.35	92.46	92.53
Labor					...	...	...	...	...
Labor Force, Persons, Number of	Number of	LLF_PE_NUM	Thousands		137,673.13	136,401.03	135,950.68	136,285.59	136,966.83

Figure 7: Macroeconomic Excel file as downloaded from IMF

	Date	US Dollar per SDR, Period Average	Interest_x	Net/gross investment in nonfinancial assets_x	Central Bank Policy Rate
	Filter	Filter	Filter	Filter	Filter
1	1998-01-02 00:00:00	1.34	253.2	27.45	5.5
2	1998-01-09 00:00:00	1.34	253.2	27.45	5.5
3	1998-01-16 00:00:00	1.34	253.2	27.45	5.5
4	1998-01-23 00:00:00	1.34	253.2	27.45	5.5
5	1998-01-30 00:00:00	1.34	253.2	27.45	5.5
6	1998-02-06 00:00:00	1.35	253.2	27.45	5.5
7	1998-02-13 00:00:00	1.35	253.2	27.45	5.5
8	1998-02-20 00:00:00	1.35	253.2	27.45	5.5
9	1998-02-27 00:00:00	1.35	253.2	27.45	5.5
10	1998-03-06 00:00:00	1.34	253.2	27.45	5.5

Figure 8: Snippet of pre-processed macroeconomic data

3.2.3. News Sentiment Pre-Processing

News articles were converted into weekly news sentiments using two word-to-sentiment dictionaries: Python's NRCLex package and the well-performing Loughran-McDonald Financial Dictionary[43,14].

Sentiment analysis generally starts with tokenisation, which splits long strings into individual words (tokens), and stemming, which trims words into a root word

representing all its possibilities, e.g. abandonment and abandoning are stemmed into abandon[59]. NRCLex carries out these operations automatically[60]. Loughran-McDonald's dictionary includes all possible alterations of words[61]. Thus only tokenisation functionality had to be built to match individual words with Loughran-McDonald sentiments.

Next, each article was passed through the relevant functions to convert words into their sentiment categories(Appendix J). NRCLex's functions convert articles into proportions of words in the article being of a certain sentiment. Thus, for consistency on the Loughran-McDonald side, article lengths and sentiment category frequencies were recorded for conversion into sentiment proportions. By this stage, each article had 16 different proportional sentiments based on 2 dictionaries.

Since the frequency of study is weekly, article sentiments were aggregated on publishing date Saturday to Friday since S&P 500 Friday closing prices were used(Appendix K). Thus, the 21/10/2022 sample in our combined dataset contains the percentage change in S&P 500 price from the previous Friday and the average of news sentiments for the week 15/10/2022-21/10/2022. Each article is taken to represent an individual 'event' in the real world, so averaging across article sentiments will average across event sentiments without biasing towards events that require more explanation (more words in an article). Biases from article lengths should be avoided as studies have found summarised news can outperform full articles[62].

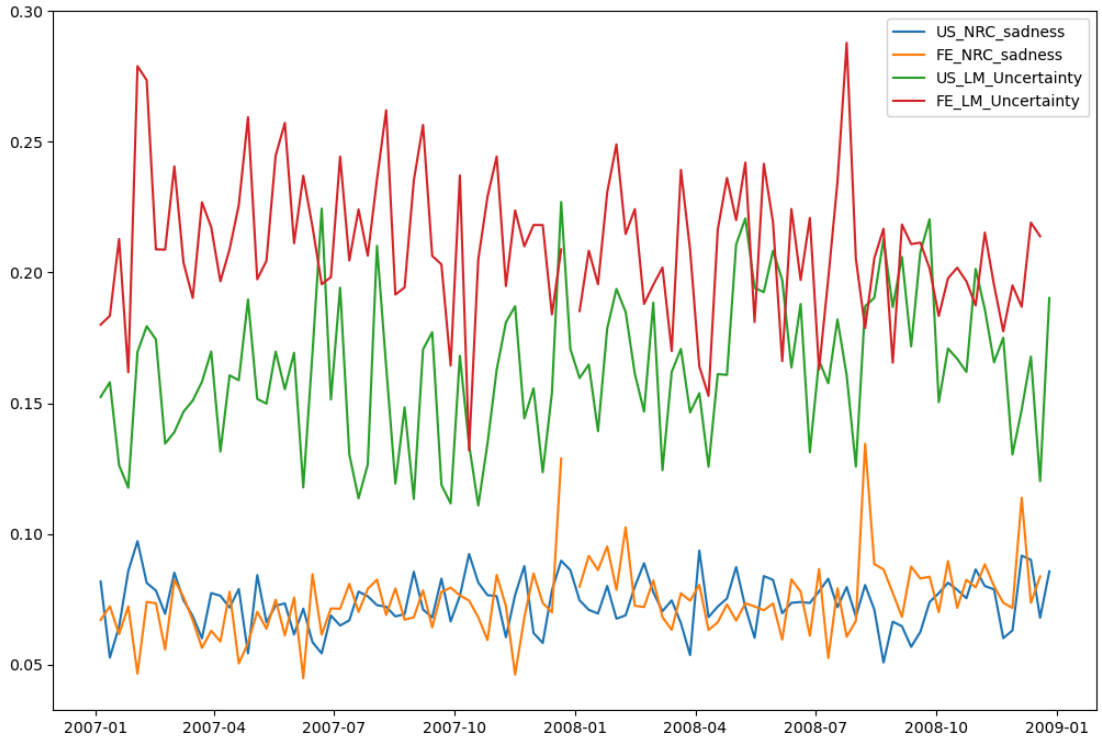

Figure 9: Selection of sentiments, 2007-2008

Weekly sentiment data is noisy like price data, complicating models' abilities to elicit underlying trends. The same smoothing technique of averaging the last four observations was thus undertaken, with its effectiveness in emphasising periods of changing sentiment shown below(Appendix G):

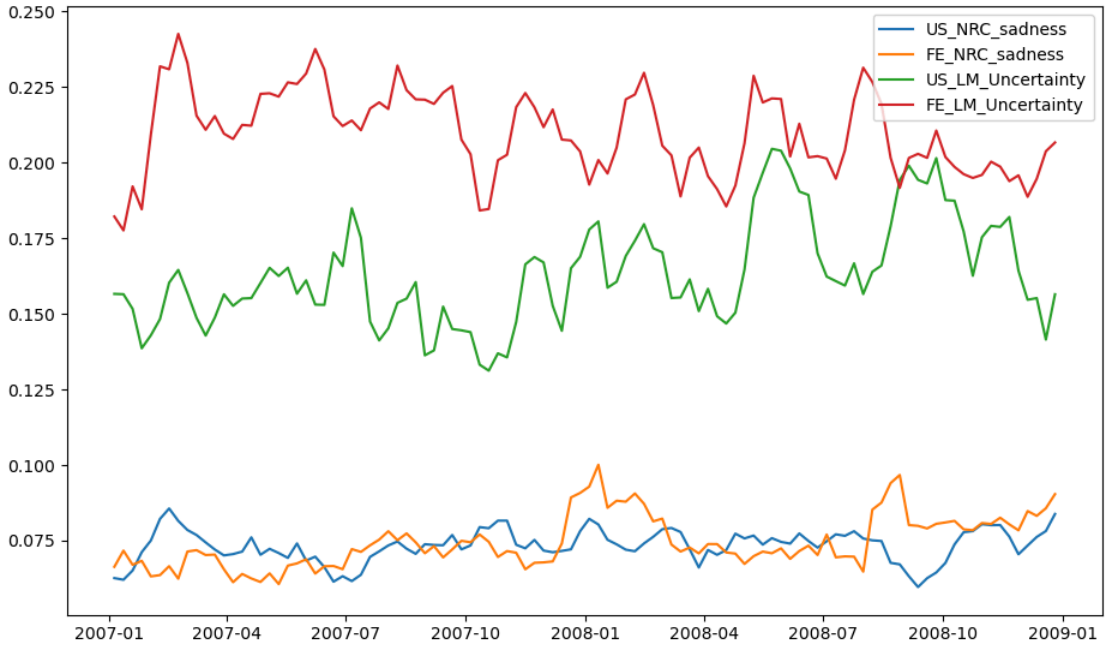

Figure 10: Figure 9, post-smoothing

	title	date	text
	Filter	Filter	Filter
1	The price of friendship	2001/08/23	ReutersIT WAS a puzzle of the cold war that the...
2	Every move you make	2013/11/16	DO YOU understand "the three represents" or "t...
3	Iran cannot fight ...	2020/03/26	Editor's note: The Economist is making some of ...
4	The iPhone turns ten	2017/06/28	NO PRODUCT in recent history has changed ...
5	Float like a butterfly	2016/10/20	MOST people know Elon Musk for his electric ...
6	How China's ...	2020/02/15	WHEN SHOCKS hit the global economy, Wall ...
7	It might still be ...	2021/08/11	"A NEGATIVE OUTCOME, a Taliban automatic ...
8	Absurdly green	1998/07/09	IT IS often ticklish to balance protection of the ...
9	Europe's rescue plan	2011/10/29	YOU can understand the self-congratulation. In t...
10	Why are the rich ...	2022/12/14	The prospect of recession may loom over the ...

Figure 11: Original 'Leaders' section news articles

	Date	LUS_LM_Weak_Modal	LUS_LM_Litigious	BFE_LM_Negative	LUS_NRC_sadness	LUS_NRC_surprise
	Filter	Filter	Filter	Filter	Filter	Filter
1	1997-03-07 00:00:00	0.0680880744529258	0.0194444444444444	0.366018465867684	0.0271057639707525	0.0192765950889521
2	1997-03-14 00:00:00	0.0584567121367728	0.0150651653854255	0.312265460976684	0.0299211105151774	0.019562516596424
3	1997-03-21 00:00:00	0.0626992656904302	0.0156098216566618	0.315725916167166	0.0281922518549935	0.0228083577587566
4	1997-03-28 00:00:00	0.0612820044078011	0.0222624054942549	0.307359016325768	0.0310517209862944	0.0208508432327644
5	1997-04-04 00:00:00	0.110735237369616	0.109030704235724	0.294787034766856	0.0712098600582858	0.0356499114344248
6	1997-04-11 00:00:00	0.107071576143704	0.113885077456087	0.312379266763779	0.071102101256346	0.0356217578356921
7	1997-04-18 00:00:00	0.106200916427422	0.122542724735636	0.30225547207457	0.0730129829215468	0.0322383551560337
8	1997-04-25 00:00:00	0.107612333503072	0.115877926662785	0.304050287615886	0.0712597323207655	0.0321146277327944
9	1997-05-02 00:00:00	0.0605298878308752	0.0306283621934798	0.306882754333009	0.0320109510454784	0.0179754376284182
10	1997-05-09 00:00:00	0.0641510893905096	0.0296983521123503	0.30040856433591	0.0308212556137823	0.018026104797884

Figure 12: Snippet of pre-processed news sentiments

The results of pre-processing news sentiments are shown above. By this point, three weekly 1998-2022 time series were created for historical prices, news sentiments and macroeconomic data. These datasets were combined into pairs and all three together. The total combined dataset had 1297 samples. Thus, objective 2 has been achieved.

3.3. Model Selection

All the models used, except XGBoost, were from Python's popular scikit-learn library. They were selected for ease of implementation and successful performance histories (section 2.3).

3.3.1. Logistic Regression (LR)[63]

LRs are a common statistical technique and the current state-of-the-art for bear market prediction[5]. They model the relationships of independent variables to a binary outcome variable (bear market state). An s-shaped logistic function generates a 0 to 1 probability of a bear market given the independent variables by maximising the log-likelihood of the observed data through the refinement of independent variable coefficients[64]. LR's linear weights make a highly interpretable model. However, LRs have strict assumptions, the harshest being linear relationships between independent and outcome variables. This is rare for most problems, stock markets included, which is one of the reasons machine learning techniques have superseded LR in most applications.

3.3.2. Support Vector Classifier (SVC)[65]

SVCs use hyperplanes to separate data into classes. Given a potentially infinite number of valid hyperplanes, SVCs find the optimal (most generalisable) hyperplane by maximising support vector distances whilst still correctly classifying data[66]. Support vectors are the most difficult-to-classify points closest to the chosen hyperplane for each class. Since hyperplanes are linear, SVCs use kernel functions to convert data into

higher dimensions where they can be linearly separable and then project them back into original dimensions, allowing them to work with non-linear problems. Different kernel parameters can drastically affect model performance. SVCs were chosen for their strong performance in stock market prediction studies and ability to deal with non-linear problems[27,30].

3.3.3. Random Forest Classifier (RFC)[67]

RFCs combine the benefits of decision trees with bagging (bootstrap aggregation). Decision trees are trained on bootstrap samples (random subsets of features and training data) to reduce overfitting[68]. They then recursively partition data on conditional nodes to maximise purity in child nodes, thus maximising difference on the condition. The leaf nodes predict which class the sample is in. These predictions are then aggregated via an ensemble voting method, meaning all features and training data get covered. The different decision trees are likely to be uncorrelated thanks to bootstrap sampling whilst the aggregation protects predictions from individual errors. RFCs are chosen thanks to a similar intuition to investment strategies in the finance industry which create a portfolio of many successful yet uncorrelated models[69].

3.3.4. XGBoost[70]

XGBoost is related to RFCs, but by using boosting over bagging state-of-the-art results are achieved on many standard classification benchmarks[34]. Decision trees are iteratively fitted on the previous decision tree’s residual errors which incentivises fixing mistakes in each iteration, with the additional benefit of helping with unbalanced classes. Whilst bagging methods train in an additive manner, boosting uses superior

optimisation methods in Euclidean space, e.g. gradient descent loss minimisation[33]. XGBoost specifically, comes with a number of innovations such as parallel computing, novel tree learning algorithms for sparse data and functions penalising overfitting. Many of these make XGBoost much more scalable, giving excellent performance in Kaggle competitions[33]. Its great performance history explains this choice.

3.3.5. Neural Networks (NN)[71]

As a series of interconnected layers of neurons, NNs are inspired by human brains[72]. The input layer's neurons represent individual features. The output layer classifies whether a sample is in a bear market. In between, hidden layers map inputs from the previous layer to some abstract representation of information. As the model makes (in)correct estimates, a process of back-propagation tunes the weights and biases of each neuron in the error-minimising direction until an optimum performance is reached. Neural networks can model complex non-linear decision boundaries giving them dominant performance in many fields, including stock market prediction, thus they have been chosen[22,23].

3.4. Feature Reduction

High dimensionality is an issue for many models, including SVCs[73,74]. By increasing the uniqueness of each sample, while often not adding more information, it risks models overfitting towards the training data. Further, many Gaussian kernels and Euclidean

norms break down in high-dimensional data, risking SVC and XGBoost performance[75]. Three categories of feature reduction techniques have been developed to reduce dimensionality[76]. Filter methods use statistical measures to select features that are best at detecting group-level differences. For similar reasons to econometric techniques being superseded, wrapper methods generally perform better. They use output metrics from a machine learning model (here: accuracy) to select features most beneficial for model performance. Embedded methods embed feature selection within the model training process by incentivising smaller feature sets with penalties on using features. For simplicity, only the first two, alongside basic techniques were used.

Firstly, macroeconomic variables with majority null values were removed. Secondly, variables without statistical meaning (unchanging features, e.g. “domestic currency per US dollar”) were removed. Thirdly, many macroeconomic variables represent similar concepts to one another, such as “national currency” in circulation and “USD” in circulation, which generates unnecessary multicollinearity between variables, decreasing model performances[77]. Here, a filter method using pairwise correlations calculated between all variables was used(Appendix L). Variables above a correlation of >0.95 were considered to represent overly similar concepts; only one per pair was kept, with rare exceptions when they clearly described different concepts (e.g. goods exports and imports surprisingly had a 0.99 correlation). When variables calculated on period average and period end existed, the latter was chosen for being more recent and likely more informative.

Next, wrapper feature reduction methods were used. Backward selection starts from all features and drops features iteratively, whilst forward selection starts from an empty feature set and adds features iteratively. Both continue until no incremental improvement in accuracy is found. Backward selection benefits from lower spurious correlation risk and the ability to assess joint predictive power[78]. Scikit-learn comes with a sequential feature selector, but its k-fold cross-validation is invalid for our time series dataset (section 3.7)[79]. Thus, backward and forward selection functionalities were built from scratch(Appendix M). These processes quickly terminated after finding no incremental improvement in accuracy, likely due to the relatively small dataset raising spurious correlations. Therefore, an iterative approach was taken in which attributes selected by the different models in forward and backward selection were saved. Since RQ1 requires models integrating only historical price data with sentiment or macroeconomic data to stand alone for comparison, wrapper feature selection was undertaken separately for the categories of data. For macroeconomic data, the best average accuracies were with the backward selected features. 65 macroeconomic variables were reduced to 17.

Regarding sentiment data, different ways of combining the news sections were attempted throughout feature selection with positive results arising from combining the economy-related Business and Finance & Economics sections and society-related US and Leaders sections. Results were also better with both NRCLex and Loughran-McDonald dictionaries included. The best performance was from forward

selection. A total of 64 sentiment variables (16 sentiments, applied to 4 sections) were reduced to 6. The wrapper methods were also applied to historical prices with the best performance occurring with 11 lagged variables dropped in backward selection.

The utility of feature reduction was clear with average model accuracies increasing from 0.630 to 0.802. Final feature sets are listed in Appendix tables 7-9.

3.5. Parameter Tuning

ML models have adjustable parameters which affect how the models are trained and perform. These options were found in the relevant online documentation[63,65,67,70,71]. Again, given time-split cross validation (section 3.7), bespoke parameter tuning code was written(Appendix N). Each parameter and each parameter's options were iterated through, with numerical parameter options given a sample covering the range of allowed values. Each model's iteration over a given parameter option would calculate its accuracy in predicting bear markets, with the best-performing option for each parameter saved for model training. One drawback of this method is that it uses default values for all other parameters, potentially missing important parameter interactions.

Neural networks additionally required finding an optimal hidden layer structure. To reduce the decision space two rules of thumbs were used. Firstly, that one hidden layer is enough for most problems, and secondly, that its size should be between the input and output layer sizes[80]. A 17-neuron hidden layer was found to be optimal.

Similarly to feature reduction, average model accuracies increased from 0.802 to 0.818.

3.6. Model training

There are five model types:

- Logistic Regression (LR)
- Support Vector Classifier (SVC)
- Random Forest (RF)
- XGBoost Classifier (XGB)
- Neural Network (Multi-Layer Perceptron Classifier) (NN)

There are also three data types:

- Historical & Sentiment
- Historical & Macroeconomic
- Historical, Macroeconomic & Sentiment

Every model type using its optimal parameters was trained on every data type in order to find the state-of-the-art across the two dimensions, generating 15 different models and achieving objective 3(Appendix O).

3.7. Model Testing

K-fold cross-validation is a commonly-used technique for splitting testing and training data due to its effectiveness in preventing overfitting and maximising data usage[81]. However, the random sampling this involves causes a loss of valuable information in time series datasets with strong inter-time dependence. Thus, a time-split cross-validation technique was used instead to improve model accuracies(Appendix P)[82].

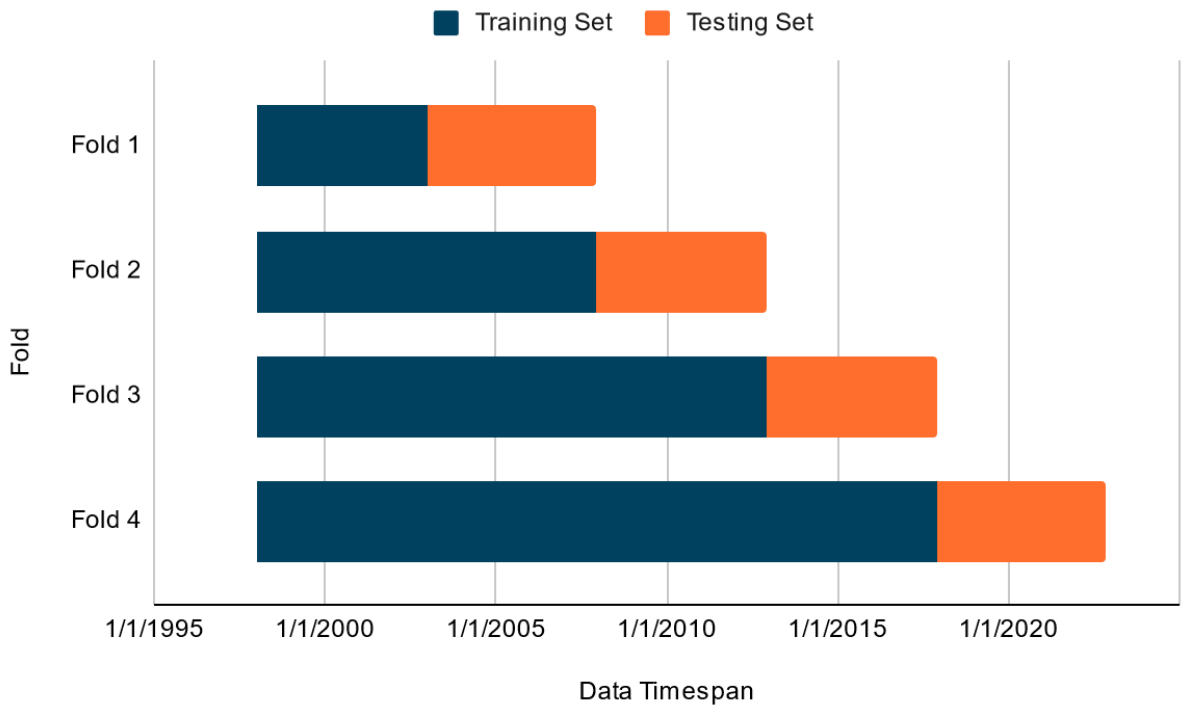

Figure 13: Visualisation of the time-split cross-validation used

The dataset was split into five equal adjacent periods, leading to the four folds shown above. The first fold is trained on the first fifth of data and tested on the second, and so

on until the final fold tests on the first four fifths and trains on the last fifth. After each fold the model is discarded with the metrics saved and averaged at the end. Four folds strike a balance between having more folds and enough bear and bull samples in each fold for the models to adequately learn and be tested.

3.8. Model Evaluation

Model evaluation is done in two ways. Firstly, classification metrics test how the different model types and data sources compare against one another on technical performance. This clarifies whether newer ML models can outperform LR and if news sentiments improve performance, achieving objective 4 by finding the best model type & data type combination. Secondly, profit analysis of an investment strategy suggested by the model versus the received wisdom of buy-and-hold tests whether the results have real-world applicability, achieving objective 5[83]. Evaluation mostly used predictions one week ahead, however one month, three month and six month predictions were also undertaken.

3.8.1. Classification Metrics

Accuracy is the simplest classification metric, stating the proportion of correctly labelled samples. Some researchers consider a predictive accuracy exceeding 50% in the stock market to successfully indicate an improvement over EMH's consequence that investors can ultimately only make random guesses[35]. However, since 71.47% of the dataset were non-bear periods, a model which only predicts non-bear periods would have an

accuracy of 71.47%. This is effectively the buy-and-hold investment approach, and will be the baseline for a model to be considered successful. Accuracy is most effective when classes are balanced. This dataset is only relatively balanced, making it important to consider other metrics for a broader view:

Precision is the proportion of predicted bear market weeks predicted correctly[84].

- Precision = True Positives / (True Positives + False Positives)

Recall is the proportion of genuine bear markets to be predicted as such.

- Recall = True Positives / (True Positives + False Negatives)

F-score is a harmonic mean between precision and recall.

3.8.2. Profit Analysis

Profit analysis tests whether a model generates superior profits by simulating an investment strategy based on its predictions. The received wisdom is that buy-and-hold is the best investment strategy, with market timing being futile[83]**.** Thus, beating buy-and-hold is a considerable achievement and will be used as the baseline strategy. The last fold of data is used, giving the models the best opportunity to learn.

The buy-and-hold strategy invests $1000 at 16-11-2017 prices and keeps the money in the S&P 500 until 04-11-2022. The models meanwhile also start with $1000 investment but then make predictions on whether an upcoming period will be a bear market or not(Appendix Q). The relative differences in closing prices are shifted up in the dataset by the number of periods ahead the model is predicting. Therefore, a ‘1’ or ‘0’ prediction corresponds to the correct return for the predicted period. A prediction of 0 (non-bear) means money is kept in the stock market with the investment multiplied by the percentage price difference. A prediction of 1 means money is taken out of the stock market and no money is lost or gained for the period.

3.9. Ethical Considerations

No personal data is used, except the human-written articles from the Economist without author information. The Economist has been contacted to confirm permission to use their articles for the purposes outlined in this study.

Should the results be significant, there are two risks. Firstly, only those with the means to access and implement the outlined methodology would benefit from any additional profit, potentially exacerbating economic inequality. Easy access to knowledge and computing resources are therefore crucial. Secondly, if automated trading algorithms implemented bear market prediction, an upcoming bear market may lead to more rapid drops in stock prices as the algorithms sell to avoid the upcoming losses. Policymakers must therefore pre-empt upcoming bear markets by increasing investor confidence through targeted policies.

However, should the methodology be flawed, there are two different risks. Firstly, investors may invest using an incorrect prediction model and end up with less money than following a buy-and-hold strategy. Readers must therefore exercise a critical eye before implementing new investment strategies. Secondly, institutions may make non-optimal decisions based on flawed prediction models. This is unlikely as the experienced staff would likely spot methodological errors.

4. Results & Discussion

4.1. RQ1: Machine Learning Performance vs Logistic Regression

Table 1: Classification metrics comparison (1 week ahead)

Model	Accuracy	Precision	Recall	F-Score
LR	0.776	0.332	0.435	0.371
SVC	0.797	0.250	0.013	0.024
RF	**0.867**	0.650	**0.794**	**0.673**
XGB	0.849	**0.731**	0.505	0.548
NN	0.802	0.612	0.352	0.340
Average	0.818	0.515	0.420	0.391

We analyse RQ1 with the complete dataset. With accuracies exceeding 71.47%, all models provided additional predictive power beyond EMH's consequences. RF had the best accuracy, with its near-dominance clear when considering the other measures. Its recall means RF correctly predicts 79.4% of all bear market periods. The current state-of-the-art, LR, has the worst accuracy of all the models, therefore the dominance of newer ML techniques in daily stock market prediction literature also applies to bear market prediction. SVC appears particularly ineffective as its low recall score suggests it predicts almost every period as a non-bear market.

Despite NNs requiring datasets much larger than the number of parameters, it performed adequately[26]. Its under-performance relative to RF and XGB could be attributed to the basic NN used, which did not integrate new specialisations discussed in section 2.3.1. Next, we examine model performances predicting bear markets over different periods of time ahead. For brevity, only accuracy is discussed.

Table 2: Model accuracies with all three data sources across prediction time

Model	1 Week	1 Month	3 Months	6 Months
LR	0.776	0.755	0.608	0.698
SVC	0.797	0.797	0.589	0.607
RF	**0.867**	0.666	0.589	0.652
XGB	0.849	**0.808**	0.733	**0.763**
NN	0.802	0.710	**0.761**	0.753
Average	0.818	0.747	0.656	0.695

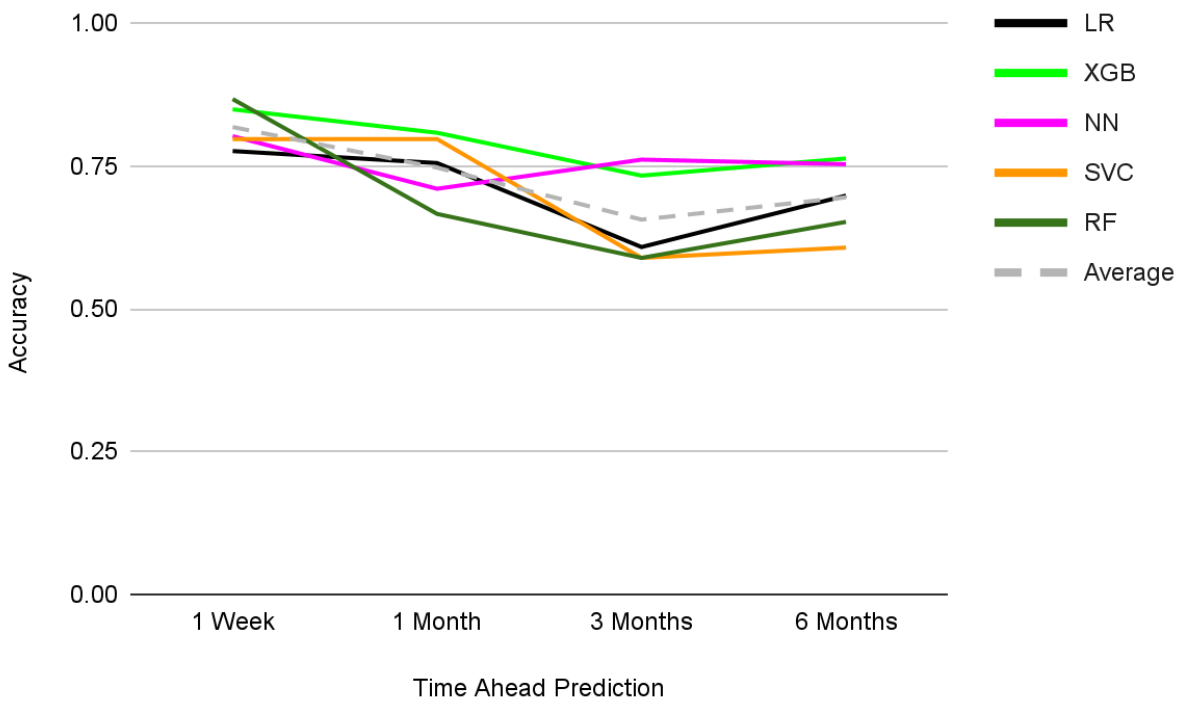

Figure 14: Model accuracies over time

While RF performed the best in short-term prediction, its accuracy quickly declined to be one of the worst performing models, calling its generalisability into question. NN and XGB performed particularly well in later predictions, never falling below the 71.47% threshold, with XGB being the most consistently effective model and the only one to always exceed the average performance. This is consistent with recent literature showing XGB and NNs as the two best performing models generally[33,22,23]. Predicting later periods may be a more complicated and non-linear problem, allowing the two models to benefit from their complex decision boundaries.

Thus, the answer to RQ1 so far is that newer ML techniques outperform LR in bear market prediction, with all models beating LR in the short-term. XGB dominates LR across all time periods and all classification metrics and appears the best model for future research, with the caveat that more specialised NNs may provide a route to superior performance in future research.

4.2. RQ2: Dataset Combination Comparisons

Table 3: Average model metrics across different dataset combinations

Dataset	Accuracy	Precision	Recall	F-Score
Historical & Macro	0.751	0.483	0.480	0.392
Historical & Sentiment	**0.888**	**0.710**	**0.722**	**0.705**
Historical, Macro & Sentiment	0.818	0.515	0.420	0.391

The current state-of-the-art combination of historical price and macroeconomic data had an accuracy of 75.1%, above the 71.47% threshold, reinforcing its effectiveness in predicting bear markets[4,5]. However, accuracy and precision improved with the

addition of sentiment data. This could be because sentiment data changes more frequently than macroeconomic data, allowing the models to learn more intricate patterns.

Interestingly, a dataset combining only historical & sentiment data dominates all other datasets across all metrics, leading to 88.8% of samples being correctly classified and 72.2% of all bear markets being correctly predicted. The inclusion of macroeconomic data brings down average performances considerably. This could be explained using EMH. Macroeconomic data has had decades of analysis within stock market prediction, allowing the relationship between macroeconomic data and the stock market to be well understood and integrated into stock prices. Meanwhile, as a newer technique, novel information from sentiment analysis is yet to be fully integrated into stock prices. More complex sentiment analysis, such as through the use of more news sources or mining social media sentiments may lead to further improvements. Next, we examine model performances in more detail using only the historical & sentiment dataset(Appendix R).

Table 4: Classification metrics comparison with the historic price & sentiment dataset (1 week ahead)

Model	Accuracy	Precision	Recall	F-Score
LR	**0.922**	0.739	0.805	**0.765**
SVC	0.850	0.637	0.750	0.643
RF	0.921	0.752	**0.888**	**0.765**
XGB	0.888	**0.765**	0.638	0.675
NN	0.860	0.655	0.799	0.678
Average	0.888	0.710	0.722	0.705

With the reduced dataset, the answer to RQ1 becomes more nuanced; different models perform best across different metrics. All the best-performing metrics listed in Table 1 have been beaten, and LR no longer appears so under-performing. As the simplest model, this is hypothesised to be due to it benefitting from a simpler dataset with less features. LR and RF appear best-performing with two optimal metrics each. However, it is not yet clear which metric is the most important for bear market prediction, so XGB remains a potential best model due its superior precision. It should be noted that NN, a suggested future research direction in 4.1, has the percentage of correctly predicted bear markets increase from 35.2% to 79.9% when reducing the dataset.

Thus, in examining RQ2, we see the importance of optimising the chosen datasets. Macroeconomic data with historical prices has superior recall and f-scores compared to a combined dataset. However, combining just sentiments and historical prices has superior performance across all metrics. The consequence for bear market prediction is that the inclusion of under-explored qualitative data is vitally important. Further performance improvements may arise from expanding these to be more holistic. Potential sources include other news sources, social media sentiments and geopolitical events.

4.3. RQ3: Profit Analysis vs Buy-and-Hold

As discussed in RQ2, it is not clear which classification metric is most important when it comes to investment. Thus, 3 models will be examined: LR with superior accuracy and f-score, RF with superior recall and f-score and XGB with superior precision. To add further nuance to RQ2, all three combinations of datasets will be examined.

The closing prices on the first and last samples of the testing dataset (the fourth fold) were 2578.85 and 3770.55 respectively. Thus, under buy-and-hold, $1000 invested at the start would lead to $1462, for an annual compound interest rate of 7.9%.

Table 5: Investment returns from models predicting bear markets 1 week ahead

Dataset	LR	RF	XGB
Historical & News Sentiment	$1684	$1483	$1729
Historical, Macroeconomic & News Sentiment	$1486	$1378	**$1935**
Historical & Macroeconomic	$1444	$1356	$1646

Under the best dataset combination uncovered in RQ2, all three models performed better than buy-and-hold. When examining just this dataset, XGB has the best performance with a return of $1729 for an annual compound interest of 11.6%. This suggests precision as the most important metric for investment as it avoids potential lost returns when pulling out of an investment market.

However, this straightforward answer gets more nuanced when considering the other datasets. Whilst LR and RF perform the best under the reduced dataset, XGB has the best investment returns under the combined dataset. This is puzzling as XGB using the combined dataset performed worse under every classification metric than XGB using the reduced dataset (see Tables 1 and 4). Therefore, the more precise prediction of bear markets does not straightforwardly lead to improved investment returns.

A possible explanation is that every bear market was valued equally in the dataset (with a binary value of 1), whereas different periods of bear markets across the dataset had varying severity of declining values. Thus, a combined dataset may have trained XGB to identify more severe bear markets through the integration of macroeconomic data as opposed to just any bear market, explaining the improved investment performance despite worse classification metrics. This interpretation explains the overwhelming focus on daily stock market prediction as this has a more direct relationship to investment returns.

Nonetheless, the answer to RQ3 is that market timing strategies can generate significantly higher returns than buy-and-hold, calling EMH into question. The best-performing model returned $1935, with an impressive 14.1% annual compound interest, almost double that of buy-and-hold(Appendix S). To illustrate the powerful difference, if this pattern held for 20 years, buy-and-hold would generate $4575, whereas the bear market prediction investment strategy would generate $13,987. This result is obviously beneficial to investors, with broader implications for the finance industry. Additionally, whilst policymakers may not directly benefit from superior profits, it highlights the feasibility and real-world effects of predicting bear markets. Creating a methodology more closely aligned with policymakers, such as by aggregating results into months or running macroeconomic simulations based on policymaker responses, is therefore a promising avenue for future research.

4.4. Critical Analysis

4.4.1. Performance Comparison to Existing Work

Performance comparison is challenging given the lack of studies on bear market prediction, emphasising the difficulties in predicting them. Nonetheless, two stock market prediction studies, both using profit analysis and a long-term dataset, were chosen for comparison.

[35] uses XGBoost for daily prediction, allowing the same best-performing model to be compared in daily prediction and bear market prediction. Unfortunately, the other studies into bear market prediction [4,5] used the statistical measures of log probability scores, which are not directly comparable or convertible to accuracy. However, they did undertake profit analysis.

All the studies use their best performing models under profit analysis. Due to the varying period lengths tested in the studies, profit improvements will be calculated by the percentage (note: not percentage point) improvement in the compound interest rate over buy-and-hold.

Table 6: Performance comparison of this study's XGB to two existing works

Model	Accuracy	Compound interest improvement
XGB	88.8%	78.5%
[35]	54.1%	44.7%
[5]	N/A	105.3%

The accuracy in predicting bear markets exceeds that of daily stock market prediction, suggesting bear and bull periods are more predictable, making them worthwhile to study. This translated into an even more profitable investment strategy which is particularly impressive as [35] used a daily prediction model providing it more opportunities to make money relative to the weekly format used in this study. [5] meanwhile, as a bear prediction model, is more directly comparable and it had a more impressive profit improvement. However, it used a more complicated investment strategy and had a questionable methodology which undertook profit analysis over the same period as the model was trained on. Thus, it is difficult to directly compare the studies. Nevertheless, predicting bear markets is clearly both possible and profitable. Future studies are invited to create more sophisticated investment strategies based on bear market prediction models.

4.4.2. Strengths

This study's novel approach provided several strengths. Foremost, it succeeded in answering the three research questions set out in the beginning. The real-world significance of predicting bear markets is indicated with the superior investment strategy performance over buy-and-hold and even over an XGBoost-based daily prediction study.

Secondly, whilst many stock market prediction studies use under a year's data, this study spanned a longer 24-year period. Shorter studies may cover a single bull or bear market period, making it easy for models to perform well by just predicting upwards or downwards respectively. By ensuring this study encompassed 13 bear and bull markets each, the models had to genuinely learn the characteristics of bear and bull markets to avoid under-performance. The transparent methodology shows the periods chosen were not biased to provide better results.

Thirdly, four model types were used for the first time in bear market prediction which clarified directions for future research. Consistent with broader computer science literature, XGBoost and specialised neural networks appear particularly promising. Fourthly, the breadth of models were expanded through diverse data sources, encompassing both quantitative and qualitative data for the first time, with the benefits of diversification shown in profit analysis.

Lastly, all models showed accuracies above the 71.47% accuracy threshold, showing they were able to learn the behaviours of the stock market successfully. These results were achieved despite time-split cross validation making smaller training datasets for three of the four folds.

4.4.3. Limitations

However, to remain manageable, the study required a limited scope resulting in a number of limitations which future studies are invited to address.

Firstly, only one news source was used, the Economist. While bias is unlikely as the sentiment changes are all relative to the Economist, it is still likely insufficient to provide a holistic qualitative view of the world. Future studies are invited to increase the diversity of qualitative data to other news sources, social media and geopolitical events. This led to a second limitation as news data were aggregated into weekly format to avoid extreme noise, limiting the dataset to 1297 samples despite the lengthy period under study. Machine learning datasets frequently reach millions of samples. Finding inventive ways to increase sample sizes will likely be valuable.

Thirdly, some macroeconomic variables had missing data meaning they might have been dropped in feature reduction despite the underlying concept they describe being potentially important. Relatedly, macroeconomic variables were extended to cover a weekly frequency, meaning macroeconomic variables often did not change, while prices and news sentiments did. Thus, the models may have undervalued macroeconomic

variables, potentially explaining their poor performance in section 4.2. However, the difficulties in generating macroeconomic data means this is an issue for most, if not all, related studies.

Fourthly, during parameter tuning, each parameter was examined independently, rather than in a grid search pattern, potentially missing important parameter interactions. Lastly, only a simple neural network was used, which regardless remains cutting-edge for bear market prediction. Future researchers are invited to build bear market prediction models with newer, specialised neural networks to try outperform XGBoost.

5. Conclusion

This paper identified a research gap in bear market prediction. Neither sentiment analysis nor newer ML models have been attempted in published research and thus the paper explored whether either of these approaches provide additional predictive power over using just macroeconomic data with logistic regressions. Therefore, time series datasets of historical S&P 500 prices, macroeconomic data and news sentiments were created. A logistic regression model, representing the state-of-the-art, and the additional ML models of support vector classifiers, random forests, XGBoost and neural networks were trained on various combinations of data source types.

Evaluating the resulting models answered the three research questions, with each answer adding nuance to the previous. Firstly, under more complex datasets, logistic regressions were among the worst performing models with the worst accuracy. However, under a smaller dataset its simplicity allowed it to achieve the highest accuracy of 92.2%. Thus, logistic regressions can only be said to be optimal under the simpler datasets used in past literature. XGBoost appears the most generalisable with it being the only model to always beat the average accuracy across all predicted time periods and having the best investment returns across all dataset combinations.

Secondly, the inclusion of news sentiments to a state-of-the-art dataset consisting of historical prices and macroeconomic data increased predictive accuracy from 75.1% to 81.8% and precision from 48.3% to 51.5%. Interestingly, a dataset consisting of only historical prices and news sentiments dominated the other dataset combinations across

all metrics with the average model classifying 88.8% of samples correctly and predicting 72.2% of all bear markets correctly. Thus, the benefits of including qualitative data are clear.

Thirdly, profit analysis discovered that when undertaking bear market prediction, classification metrics have a less straightforward relationship to investment returns than metrics do in daily stock market prediction, potentially explaining the lack of bear market prediction studies. Despite XGBoost performing worse under every single metric with a reduced historical price and sentiment dataset, XGBoost's investment performance was best with a combined dataset. This best-performing model generated an annual compound interest rate of 14.1%, far exceeding the 7.9% under buy-and-hold.

Overall, the diversification of data sources brings significant benefits. As a consequence, future researchers should use newer ML techniques as these perform considerably better than logistic regressions with the complex datasets this involves. This study partly supports and partly challenges the Efficient Market Hypothesis. EMH helps to explain why sentiment data generated superior benefits as compared to macroeconomic data; unlike sentiment analysis, the relationship between macroeconomic data and the stock market has had decades of research, allowing it to be well integrated into prices. On the other hand, bear markets were also shown to be predictable which generated vastly superior investment returns over buy-and-hold, meaning EMH is certainly not absolute.

As a foundational study, there remains much room for future research. Firstly, future studies should attempt to outperform XGBoost through the use of specialised neural networks. Secondly, this study has made the benefits of diversifying data sources clear, thus, future researchers are likely to find superior performance through further diversification of their qualitative data sources by using more news sources, social media sentiments and geopolitical events. Lastly, researchers are invited to improve the real world benefits for investors or policymakers by creating more complex investment strategies based on bear market prediction models or creating a methodology more aligned to the needs of policymakers respectively.

6. References

1. L. Gonzalez, J. G. Powell, J. Shi, and A. Wilson, “Two centuries of bull and bear market cycles,” *International Review of Economics & Finance*, vol. 14, no. 4, pp. 469–486, 2005
2. P. Shen Market timing strategies that worked: Based on the E/P ratio of the S&P 500 and interest rates Journal of Portfolio Management, 29 (2) (2003), pp. 57-68
3. M.T. Bohl, P.L. Siklos, T. Werner Do central banks react to the stock market? The case of the Bundesbank Journal of Banking and Finance, 31 (3) (2007), pp. 719-733
4. S.-S. Chen, “Predicting the bear stock market: Macroeconomic variables as leading indicators,” *Journal of Banking & Finance*, vol. 33, no. 2, pp. 211–223, 2009.
5. N.-K. Chen, S.-S. Chen, and Y.-H. Chou, “Further evidence on bear market predictability: The role of the External Finance Premium,” *International Review of Economics & Finance*, vol. 50, pp. 106–121, 2017
6. “Will the Wuhan virus become a pandemic?,” *The Economist*, London, 30-Jan-2020
7. A. Hayes, “Stocks: What they are, main types, how they differ from bonds,” *Investopedia*, 06-Jul-2022. [Online]. Available: https://www.investopedia.com/terms/s/stock.asp. [Accessed: 15-Jan-2023].
8. “Surging stocks undermine a hallowed investing rule,” *the Economist*, London, 07-Feb-2023.
9. E. R. McGrattan and E. C. Prescott, “Is the Stock Market Overvalued?,” Cambridge, MA: National Bureau of Economic Research, 2001, pp. 1–39.

10. E. F. Fama, “The behaviour of stock-market prices,” *The Journal of Business*, vol. 38, no. 1, pp. 34–105, 1965
11. B. G. Malkiel, “The efficient market hypothesis and its critics,” *Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59–82, 2003.
12. S. J. Grossman and J. E. Stiglitz, “On the Impossibility of Informationally Efficient Markets,” *The American Economic Review*, vol. 70, no. 3, pp. 393–408, 1980.
13. D. Kahneman, *Thinking, fast and slow*. New York, New York: Farrar, Straus and Giroux, 2011.
14. P. Tetlock, M. Saar-Tsechansky, and S. Mackassy, “More than words: Quantifying language to measure firms' fundamentals,” *The Journal of Finance*, vol. 63, no. 3, pp. 1437–1467, 2008.
15. W. Brock, J. Lakonishok, and B. LeBaron, “Simple technical trading rules and the stochastic properties of stock returns,” *The Journal of Finance*, vol. 47, no. 5, pp. 1731–1764, 1992.
16. J. Pontiff, “Book-to-market ratios as predictors of market returns,” *Journal of Financial Economics*, vol. 49, no. 2, pp. 141–160, 1998.
17. J. Lewellen, “Predicting returns with financial ratios,” *Journal of Financial Economics*, vol. 74, no. 2, pp. 209–235, 2004.
18. B. Weng, “Application of machine learning techniques for stock market prediction,” dissertation, Auburn University, Auburn, Alabama, 2017.
19. W. Jiang, “Applications of deep learning in stock market prediction: Recent progress,” *Expert Systems with Applications*, vol. 184, p. 115537, 2021.

20. X. Ding, Y. Zhang, T. Liu, and J. Duan, “Using structured events to predict stock price movement: An empirical investigation,” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1415–1425, Oct. 2014.
21. F. A. de Oliveira, C. N. Nobre, and L. E. Zárate, “Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index – case study of PETR4, Petrobras, Brazil,” *Expert Systems with Applications*, vol. 40, no. 18, pp. 7596–7606, 2013.
22. I. K. Nti, A. F. Adekoya, and B. A. Weyori, “A systematic review of fundamental and technical analysis of stock market predictions,” *Artificial Intelligence Review*, vol. 53, no. 4, pp. 3007–3057, 2019
23. S. K. Goel, B. Poovathingal, and N. Kumari, “Applications of Neural Networks in Stock Market Prediction,” *International Research Journal of Engineering and Technology*, vol. 3, no. 5, pp. 2192–2197, May 2016.
24. G. Ding and L. Qin, “Study on the prediction of stock price based on the Associated Network Model of LSTM,” *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 6, pp. 1307–1317, Nov. 2019
25. S. Deng, N. Zhang, W. Zhang, J. Chen, J. Z. Pan, and H. Chen, “Knowledge-driven stock trend prediction and explanation via temporal convolutional network,” *Companion Proceedings of The 2019 World Wide Web Conference*, pp. 678–685, May 2019
26. A. Alwosheel, S. van Cranenburgh, and C. G. Chorus, “Is your dataset big enough? sample size requirements when using artificial neural networks for

discrete choice analysis," *Journal of Choice Modelling*, vol. 28, pp. 167–182, 2018.

27. L. Cao and F. Tay, "Support Vector Machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506–1518, 2003.
28. V. N. Vapnik, The Nature of Statistical Learning Theory, New York:Springer-Verlag, 1995
29. Y. Lin, H. Guo, and J. Hu, "An SVM-based approach for stock market trend prediction," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013.
30. M. Kumar and M. Thenmozhi, "Forecasting stock index movement: A comparison of support vector machines and Random Forest," *SSRN Electronic Journal*, 2006.
31. H. Guan, J. Li, M. Chapman, F. Deng, Z. Ji, and X. Yang, "Integration of orthoimagery and LIDAR data for object-based urban thematic mapping using random forests," *International Journal of Remote Sensing*, vol. 34, no. 14, pp. 5166–5186, 2013.
32. R. L. Lawrence, S. D. Wood, and R. L. Sheley, "Mapping invasive plants using hyperspectral imagery and Breiman Cutler classifications (randomforest)," *Remote Sensing of Environment*, vol. 100, no. 3, pp. 356–362, 2006.
33. T. Chen and C. Guestrin, "XGBoost," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

34. P. Li, “Robust logitboost and adaptive base class (ABC) logitboost,” *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 302–311, 2010.
35. J. Nobre and R. F. Neves, “Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets,” *Expert Systems with Applications*, vol. 125, pp. 181–194, 2019.
36. R. Liu and D. F. Gillies, “Overfitting in linear feature extraction for classification of high-dimensional image data,” *Pattern Recognition*, vol. 53, pp. 73–86, 2016.
37. T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Google Inc, Mountain View, California, tech., 2013
38. X. Li, P. Wu, and W. Wang, “Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong,” *Information Processing & Management*, vol. 57, no. 5, p. 102212, 2020.
39. E. Cambria, S. Poria, D. Hazarika, and K. Kwok, “SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
40. X. Li, H. Xie, L. Chen, J. Wang, and X. Deng, “News impact on stock price return via sentiment analysis,” *Knowledge-Based Systems*, vol. 69, pp. 14–23, 2014.
41. J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

42. T. Loughran and B. McDonald, “When is a liability not a liability? textual analysis, dictionaries, and 10-KS,” *The Journal of Finance*, vol. 66, no. 1, pp. 35–65, 2011.
43. P. C. Tetlock, “Giving content to investor sentiment: The role of media in the stock market,” *The Journal of Finance*, vol. 62, no. 3, pp. 1139–1168, 2007.
44. X. Li, C. Wang, J. Dong, F. Wang, X. Deng, and S. Zhu, “Improving Stock Market Prediction by Integrating Both Market News and Stock Prices” in *International Conference on Database and Expert Systems Applications*, 2011, pp. 279–293.
45. R. P. Schumaker and H. Chen, “Textual analysis of stock market prediction using Breaking Financial News,” *ACM Transactions on Information Systems*, vol. 27, no. 2, pp. 1–19, 2009
46. J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011
47. N. Oliveira, P. Cortez, and N. Areal, “On the predictability of stock market behaviour using StockTwits sentiment and posting volume,” *Progress in Artificial Intelligence*, pp. 355–365, 2013.
48. Y. Yu, W. Duan, and Q. Cao, “The impact of social and conventional media on firm equity value: A sentiment analysis approach,” *Decision Support Systems*, vol. 55, no. 4, pp. 919–926, 2013.
49. L. Gonzalez, J. G. Powell, J. Shi, and A. Wilson, “Two centuries of bull and bear market cycles,” *International Review of Economics & Finance*, vol. 14, no. 4, pp. 469–486, 2005

50. G. Bry and C. Boschan, “Cyclical Analysis of Time Series: Selected Procedures and Computer Programs,” Cambridge, MA: National Bureau of Economic Research, 1971
51. J. W. Creswell and J. D. Creswell, “Selection of a Research Approach,” in *Research design: Qualitative, quantitative & mixed methods approaches*, Los Angeles, California: SAGE, 2018, pp. 31–49.
52. “S&P 500 (^GSPC) historical data,” *Yahoo! Finance*, 07-Nov-2022. [Online]. Available: https://finance.yahoo.com/quote/%5EGSPC/history/. [Accessed: 07-Nov-2022].
53. “United States,” *IMF data*. [Online]. Available: https://data.imf.org/regular.aspx?key=61545854. [Accessed: 10-Nov-2022].
54. “World News, Economics, Politics, Business & Finance,” *The Economist*. [Online]. Available: http://www.economist.com/. [Accessed: 23-Dec-2022].
55. “SQLITE3 - DB-API 2.0 interface for SQLite databases,” *Python documentation*. [Online]. Available: https://docs.python.org/3/library/sqlite3.html. [Accessed: 29-Dec-2022].
56. E. Yardeni, J. Abbott, and M. Quintana, “Stock Market Historical Tables: Bull & Bear Markets,” *Yardeni Research, Inc*. [Online]. Available: https://www.yardeni.com/pub/sp500corrbeartables.pdf. [Accessed: 19-Nov-2022].
57. T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.
58. J. C. C. Chan, “Moving average stochastic volatility models with application to inflation forecast,” *Journal of Econometrics*, vol. 176, no. 2, pp. 162–172, 2013.

59. C. C. Aggarwal, “Overview of Text Mining for Web Data Mining” in *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data,* Cham, Switzerland: Springer, 2022, pp. 1–12.
60. S. Mohammad, “NRC Word-Emotion Association Lexicon,” *NRC Emotion Lexicon*. [Online]. Available: https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm. [Accessed: 03-Jan-2023].
61. T. Loughran and B. McDonald, “Loughran-McDonald Master Dictionary w/ Sentiment Word Lists,” *Software Repository for Accounting and Finance*. [Online]. Available: https://sraf.nd.edu/loughranmcdonald-master-dictionary/. [Accessed: 03-Jan-2023].
62. X. Li, H. Xie, Y. Song, S. Zhu, Q. Li, and F. L. Wang, “Does summarization help stock prediction? A news impact analysis,” *IEEE Intelligent Systems*, vol. 30, no. 3, pp. 26–34, 2015.
63. “Sklearn.linear_model.logisticregression,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 07-Jan-2023].
64. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, “Linear Models,” in *Data Mining: Practical Machine Learning Tools and Techniques*, Cambridge, Massachusetts: Morgan Kaufmann, 2017, pp. 128–135.
65. “Sklearn.svm.SVC,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: 07-Jan-2023].

66. P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, “Classification: Alternative Techniques,” in *Introduction to data mining*, Harlow, England: Pearson Education Limited, 2020, pp. 396–560.
67. “Sklearn.ensemble.randomforestclassifier,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 07-Jan-2023].
68. M. Belgiu and L. Drăguţ, “Random Forest in remote sensing: A review of applications and Future Directions,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016.
69. T. Yiu, “Understanding Random Forest,” *Towards Data Science*, 12-Jun-2019.
70. “XGBoost Python package,” *XGBoost Python Package - xgboost 1.7.4 documentation*. [Online]. Available: https://xgboost.readthedocs.io/en/stable/python/index.html. [Accessed: 07-Jan-2023].
71. “Sklearn.neural_network.MLPClassifier,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. [Accessed: 07-Jan-2023].
72. S. Russell and P. Norvig, “Artificial Neural Networks,” in *Artificial Intelligence: A modern approach*, New York, New York: Larsen & Keller Education, 2017, pp. 727–737.
73. G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation, " Journal of Machine Learning Research, vol. 11, pp. 2079-2107, 2010

74. A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, "Support vector machines and kernels for computational biology, " PLoS Comput. Biol., vol. 4, no. 10, 2008
75. M. Verleysen and D. Francois, “The Curse of Dimensionality in Data Mining and Time Series Prediction” in *International Work-Conference on Artificial Neural Networks*, 2005, pp. 758–770.
76. B. Mwangi, T. S. Tian, and J. C. Soares, “A review of feature reduction techniques in neuroimaging,” *Neuroinformatics*, vol. 12, no. 2, pp. 229–244, 2013.
77. T. W. Hua and Dougherty, E. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition, 42*, 409–424
78. M. Chowdhury and T. Turin. “Variable selection strategies and its importance in clinical prediction modelling”, *Fam Med Com Healt*h, 2020
79. “Sklearn.feature_selection.Sequentialfeatureselector,” *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html#sklearn.feature_selection.SequentialFeatureSelector. [Accessed: 14-Jan-2023].
80. J. Heaton, “The Number of Hidden Layers,” in *Introduction to Neural Networks in Java*, St Louis, Missouri: Heaton Research, 2005, pp. 128–131.
81. D. Berrar, “Cross-validation,” *Encyclopedia of Bioinformatics and Computational Biology*, pp. 542–545, 2019.

82. A. P. Ratto, S. Merello, L. Oneto, Y. Ma, L. Malandri, and E. Cambria, “Ensemble of Technical Analysis and machine learning for market trend prediction,” *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018.
83. T. Hale, *Smarter Investing*, 3rd ed. Upper Saddle River, New Jersey: Financial Times, 2013.
84. B. Liu, “Evaluation Measures,” in *Web data mining: Exploring hyperlinks, contents, and usage data*, 2nd ed., Berlin, Germany: Springer, 2011, pp. 223–227.

7. Appendix

Appendix A: artefacts contents

- **Model Instructions** holds detailed instructions to reproduce the results of this study and proof of successful run on the University-provided VMs
- **Background documentation** folder
    - **Economist Research Use Email** confirmation of permission to use the Economist articles for this study
    - **Gmail - Economist research use** HTML file of the above permission
    - **Fast Track Ethics Form-Domantas Vaicys** the ethics form sent to and approved by the University of York
    - **University of York Mail - Fast-Track Ethics Application - Vaicys20221110** email confirmation of ethics approval
- **Code** folder stores the code to build the models
    - **1_web_crawler.py** pulls URLs and article contents from the Economist
    - **2_sentiment_analyser.py** converts article contents into weekly sentiments
    - **3_preprocessing.py** carries out the bulk of pre-processing
    - **4_feature_reduction.py** implements feature reduction logic
    - **5_model_training.py** trains the models and evaluates them using classification metrics and profit analysis
    - **visualitation.py** holds miscellaneous visualisation functionality
- **Data** folder stores the data used by the models
    - **data.db** the final database used to achieve results detailed in section 4

- **macroeconomic_correlations.csv** output of pairwise correlations between macroeconomic variables
- **Macroeconomic_IMF** folder stores the original and converted macroeconomic variables downloaded from the IMF
- **News_Sentiment** folder
  - **Articles** folder stores the raw article contents from the Economist in their relevant section
  - **URLs** folder stores the URLs for the Economist articles in their relevant sections
  - **LM_Financial_Dictionary** the original Loughran-McDonald financial dictionary
  - **LM_Financial_Dictionary_Cleaned** the cleaned Loughran-McDonald financial dictionary for use in this study

Appendix B: Web crawler which downloads the Economist article URLs

```
def download_article_urls():
    for section in Section._registry: #Leaders, United States, Business and Finance & Economics sections
        file = open(section.file, "w")
        for page_number in range(1, section.pages): #Iterates through the number of pages in back-catalogue
            url = section.archive_url + str(page_number) #URL per page in back-catalogue
            page = requests.get(url)
            soup = BeautifulSoup(page.content, "html.parser") #HTML elements of page
            article_elements = soup.find_all('h3') #Almost every article's contents are within the 'h3' section
            for element in article_elements:
                url = element.find('a', href=True)
                file.write(url['href'] + '\n') #The HTML element describing the article URL
            print(page_number)
        file.close()
```

Appendix C: Web crawler which downloads the Economist article contents from URLs

```
def download_articles():
    for section in Section._registry: #Leaders, United States, Business and Finance & Economics sections
        file = open(section.file, "r") #File with downloaded Economist URLs
        counter = 1
        for url in file.readlines():
            try: #A few articles with idiosyncratic page structures are skipped
                json_elements = json.loads(Section.extract_http_elements(url))
                article_text, url, headline = Section.extract_article_content(json_elements)
                section_name, article_date, article_date_name = Section.extract_article_metadata(url)
                assert section_name == section.name #Validation check for expected section
                file_path = 'articles/' + section_name + '/'
                file_name = article_date_name + ' ' + headline
                with open(file_path + file_name + '.txt', 'w') as f: #Saving article contents into .txt files
                    f.write(headline + '\n')
                    f.write(article_date  + '\n')
                    f.write(article_text)
                print(section_name, counter)
            except:
                with open('failed_articles.txt', 'a') as f:
                    f.write(section_name + ' ' + str(counter) + ' has failed \n')
            counter = counter + 1
```

Appendix D: Function to convert the daily S&P dataset into weekly format

```
def make_SandP_weekly():
    df = pd.read_csv('../Data/S&P_500/S&P_500.csv')
    df['Date'] = pd.to_datetime(df['Date'])
    df_fridays = df[df['Date'].dt.weekday == 4] #Python starts with 0; the 4th Weekday is a Friday. Data reduced to Fridays
    rows_to_add = [] #Holding list for weekly data
    for index, row in df_fridays.iterrows():
        if 'previous_row' in locals(): #Ensures the below is only run when previous_row variable is defined
            if (row['Date'] - previous_row['Date']) < timedelta(-7): #Checks if a week is missing due to bank holidays
                missing_date = previous_row['Date'] - timedelta(7)
                date_found = False
                days_back = 1
                while date_found == False: #Iterates going back a day over daily dataset until a replacement date is found
                    replacement_date = df.loc[df['Date'] == missing_date - timedelta(days_back)]
                    if len(replacement_date) == 0:
                        days_back = days_back + 1
                    else:
                        date_found = True
                replacement_date['Date'] = missing_date
                rows_to_add.append(replacement_date) #Replacement sample for missing Fridays added to dataset
                if (row['Date'] - previous_row['Date']) < timedelta(-14): #Repeats logic for periods with two missing weeks
                    second_missing_date = previous_row['Date'] - timedelta(14)
                    date_found = False
                    days_back = 1
                    while date_found == False:
                        replacement_date = df.loc[df['Date'] == second_missing_date - timedelta(days_back)]
                        if len(replacement_date) == 0:
                            days_back = days_back + 1
                        else:
                            date_found = True
                    replacement_date['Date'] = second_missing_date
                    rows_to_add.append(replacement_date)
        previous_row = row
    for row in rows_to_add: #Converts holding list into a dataframe
        df_fridays = df_fridays.append(row)
    df_fridays = df_fridays.sort_values('Date')
    df_fridays.to_csv('../Data/S&P_500/S&P_500_weekly.csv', index=False)
```

Appendix E: Function to create the relative percentage difference features

```
def convert_SandP_data_to_percentages():
    columns_to_exclude = ['Bear20', 'Bear10', 'Variance_P']
    df = pd.read_sql_table('S&P_500', engine)
    df = df.rename(columns={'Indicator': 'Date'})
    df = df.set_index('Date')
    for column in df.columns:
        if column in columns_to_exclude:
            print(column, 'excluded')
        else:
            df[column + '_RD'] = df[column].diff() #Calculates difference in value from preceding sample
            df[column] = df[column].shift(1) #Moves original prices up so differences can be calculated to current week
            df[column + '_RDP'] = round((100 / df[column]) * df[column + '_RD'], 2) #Calculates percentage difference
            df[column + '_RDP'] = df[column + '_RDP'].replace(0, np.nan) #Deals with the first sample which has no difference
            df[column + '_RDP'] = df[column + '_RDP'].fillna(method='ffill') #First sample filled with second
            df = df.drop([column, column + '_RD'], axis=1) #Absolute difference no longer needed
    df = df.reset_index()
    df.to_sql(name='S&P_500_relative', con=conn, if_exists='replace', index=False)
```

Appendix F: Function to create bear market dependent variables

```
def create_bear_market_dummies():
    df = pd.read_sql_table('S&P_500', engine)
    df = fix_numbers(df) #Commas removed, e.g. 40,000 -> 40000
    df['Bear20'] = df['Date'].apply(lambda x: in_bear_market(x)) #If date within defined bear market periods
    df['Bear10'] = df['Date'].apply(lambda x: in_correction(x)) #If date within defined correction period
    df.to_sql(name='S&P_500', con=conn, if_exists='replace', index=False)
```

Appendix G: Functions for data smoothing

```
def smooth_SandP():
    df = pd.read_sql_table('S&P_500_relative', engine)
    for column in df.columns:
        if column not in ['Bear20', 'Bear10', 'Date']: #Excludes columns not suitable for smoothing
            df[column] = df[column].rolling(window=4, min_periods=1).mean() #Averages over last 4 periods
    df.to_sql(name='S&P_500_relative_smoothed', con=conn, if_exists='replace', index=False)

def smooth_news_sentiment():
    df = pd.read_sql_table('News_Sentiments', engine)
    for column in df.columns:
        if column not in ['Date']: #Excludes columns not suitable for smoothing
            df[column] = df[column].rolling(window=4, min_periods=1).mean() #Averages over last 4 periods
    df.to_sql(name='News_Sentiments_smoothed', con=conn, if_exists='replace', index=False)
```

Appendix H: Function for adding lagged variables

```
def add_lagged_variables():
    df = pd.read_sql_table('S&P_500_relative_smoothed', engine)
    for lag in [1, 2, 4, 8, 12]: #1-12 week lags
        df['Close_RDP_lag_' + str(lag)] = df['Close_RDP'].shift(periods=(lag)) #Data shifted up by size of lag
        df['Volume_RDP_lag_' + str(lag)] = df['Volume_RDP'].shift(periods=(lag))
    for lag in [4, 8, 12]: #4-12 week lags
        df['Bear10_lag_' + str(lag)] = df['Bear10'].shift(periods=(lag))
        df['Bear10_lag_' + str(lag)] = df['Bear10_lag_' + str(lag)].replace(np.nan, 0) #Late 1997 was a non-bear period
    df.to_sql(name='S&P_500_relative_smoothed', con=conn, if_exists='replace', index=False)
```

Appendix I: Function for macroeconomic data extension

```
def convert_IMF_data_to_weekly():
    relative_directory = '../Data/Macroeconomic_IMF/Monthly Excels'
    save_directory = '../Data/Macroeconomic_IMF/Weekly Excels/'
    for filename in os.listdir(relative_directory):
        filepath = os.path.join(relative_directory, filename)
        if os.path.isfile(filepath):
            print(filepath)
            df = pd.read_excel(filepath, index_col=False)
            df['Indicator'] = df['Indicator'].str.replace('M',',01,') #Convert into valid datetime format
            df['Indicator'] = pd.to_datetime(df['Indicator'])
            if filename == 'US_IMF_Trade_of_Goods.xlsx': #Bespoke logic for an idiosyncratic file
                df = df.sort_values(['Indicator']) #Sort values by date (column not yet renamed)
                df = df.reset_index()
                df = df.ffill() #Forward fill the values to extend until next valid observation
            df_weekly = df.resample("W-FRI", on="Indicator").mean() #Changes the date to be the date of the Friday
            df_weekly = df_weekly.reset_index()
            df_weekly = df_weekly.ffill() #Forward fill the values to extend until next valid observation
            df_weekly.to_excel(save_directory + filename, index=False)
```

Appendix J: Function to calculate article sentiments using LM financial dictionary

```
def calculate_lm_sentiments():
    financial_dictionary = pd.read_csv('../Data/News_Sentiment/LM_Financial_Dictionary_Cleaned.csv') #LM Financial Dictionary
    lm_sentiments = ['Negative', 'Positive', 'Uncertainty', 'Litigious', 'Strong_Modal', 'Weak_Modal', 'Constraining']
    columns = ["test_title", "words"]
    for sentiment in lm_sentiments:
        columns.append(sentiment) #New sentiment columns
    #section-level
    for table in ['business', 'leaders', 'finance_and_economics', 'united_states']:
        cur.execute('SELECT * FROM ' + table) #Selects all articles per section
        data = []
        #article-level
        for row in cur: #Iterates over articles per section
            sentiment_counts = {'Negative': 0,  #Set all sentiments to 0
                'Positive': 0,
                'Uncertainty': 0,
                'Litigious': 0,
                'Strong_Modal': 0,
                'Weak_Modal': 0,
                'Constraining': 0}
            df_row = [] #Holding list for sentiment sample for dataframe
            number_of_words = calculate_words_in_article(row[2])
            df_row.append(row[0]) #Article date
            df_row.append(number_of_words)
            word_list = row[2].split(' ') #Article contents tokenised
            #word-level
            for word in word_list: #Iterates over words per article
                word_exists = financial_dictionary['Word'].isin([word.upper()]) #Finds all relevant sentiments per word
                sentiment_word = financial_dictionary.loc[word_exists] #Finds the sentiment for the word
                if len(sentiment_word) > 0: #If a sentiment exists for the word
                    sentiment_type = sentiment_word.columns[sentiment_word.isin([1]).any()]
                    for sentiment in sentiment_type:
                        sentiment_counts[sentiment] = sentiment_counts[sentiment] + 1 #Increments the sentiment count
            for sentiment in sentiment_counts:
                df_row.append(sentiment_counts[sentiment]) #Adds the relevant value to the relevant sentiment column
            data.append(df_row) #Adds holding list for row into dataframe
            print(table, len(data))
        df = pd.DataFrame(data, columns=columns)
        df.to_sql(table + '_LM', con=engine, if_exists='replace')
```

Appendix K: Function to convert article sentiments into weekly sentiments

```
def group_articles_by_week():
    for table in ['business_sentiments', 'finance_and_economics_sentiments', 'leaders_sentiments', 'united_states_sentiments']:
        df = pd.read_sql_table(table, engine)
        df['Date'] = pd.to_datetime(df['Date'])
        df = df.resample('W-Fri', on='Date').mean().reset_index().sort_values(by='Date') #Averages weekly sentiments, starting Friday
        df.to_sql(name=table + '_weekly', con=conn, if_exists='replace', index=False)
```

Appendix L: Function finding pairwise correlations between macroeconomic features

```
def create_correlations_csv():
    df = pd.read_sql_table('Macroeconomic', engine)
    corr_matrix = df.corr().abs() #Calculates a correlation matrix between all variables
    corr_values = corr_matrix.unstack()
    corr_values = corr_values.sort_values(kind="quicksort", ascending=False) #Sorts correlations in descending order
    corr_values.to_csv('../Data/macroeconomic_correlations.csv') #Creates a CSV that was examined to choose variables to remove
```

Appendix M: Functions for feature reduction

```
def feature_selection(direction):
    df = pd.read_sql_table('Data_SN', engine) #Different tables need to be added per dataset
    df = df.drop(['Variance_P', 'Close_RDP', 'Volume_RDP',], axis=1) #Removes base S&P 500 features
    folds = Feature.create_walk_forward_folds(df) #Creates time-split cross-validation folds
    Feature.record_features(folds[0][2], direction) #Saves the features in dataset
    while Feature.best_feature_set_found == False: #Iterates until no incremental improvement in accuracy
        for fold in folds:
            Feature.find_fold_accuracies(fold[0], fold[1], fold[2], fold[3], direction)
        Feature.find_avg_accuracies(direction) #Finds average accuracy across folds
        Feature.check_if_best_feature_is_improvement(direction) #Prints feature name if accuracy improves
```

```
def find_fold_accuracies(y_train, y_test, x_train, x_test, direction):
    chosen_set = Feature.find_chosen_features() #Creates the chosen set of features
    if direction == 'forward': #Forward feature reduction
        for feature in Feature.instances: #Iterates through features
            if feature.chosen == False: #If feature not yet chosen
                chosen_set.append(feature.column_name)
                reduced_x_train = x_train[chosen_set] #Reduces featurs to chosen set
                reduced_x_test = x_test[chosen_set]
                chosen_set.remove(feature.column_name) #Resets the chosen set
                accuracy = find_svc_accuracy(y_train, y_test, reduced_x_train, reduced_x_test)
                feature.fold_accuracies = feature.fold_accuracies + [accuracy] #4 folds for later averaging
    else: #Backward feature reduction
        for feature in Feature.instances:
            if feature.chosen == True: #If feature not yet removed
                chosen_set.remove(feature.column_name)
                reduced_x_train = x_train[chosen_set] #Reduces features to chosen set
                reduced_x_test = x_test[chosen_set]
                chosen_set.append(feature.column_name) #Resets the chosen set
                accuracy = find_svc_accuracy(y_train, y_test, reduced_x_train, reduced_x_test)
                feature.fold_accuracies = feature.fold_accuracies + [accuracy]
```

Appendix N: Function for parameter tuning

```
def tune_parameters(y_variable):
    df = pd.read_sql_table('Data_SMN', engine)
    folds = Model.create_walk_forward_folds(df, y_variable) #Time-split cross-validation folds
    for model in Model.instances: #Iterates through ML models
        for parameter in model.parameters: #Iterates over model parameters
            best_option = None
            best_accuracy = 0
            for option in model.parameters[parameter]: #Iterates over model parameter options
                try: #Builds model with the parameter option
                    classifier_build_statement = model.class_name + '(' + model.default_parameters + ', ' + parameter + '="' + option + '")'
                except TypeError: #Fixes error from numerical parameter options
                    classifier_build_statement = model.class_name + '(' + model.default_parameters + ', ' + parameter + '=' + str(option) + ')'
                print(classifier_build_statement)
                model.classifier = eval(classifier_build_statement) #Builds the model
                for fold in folds:
                    Model.y_train, Model.y_test, Model.x_train, Model.x_test = fold[0], fold[1], fold[2], fold[3]
                    Model.create_y_pred(model) #Creates the predicted dependent variable values
                    Model.find_fold_accuracies(model) #Calculates accuracies per fold
                model.avg_accuracy = Model.find_average(model.fold_accuracies) #Averages fold accuracies
                print(model.avg_accuracy)
                if model.avg_accuracy > best_accuracy: #If the parameter option is best-performing it is saved
                    best_option = option
                    best_accuracy = model.avg_accuracy
            print('************')
            print('For', model.name, parameter, 'works best with', best_option, 'at', best_accuracy, 'accuracy')
            print('************')
```

Appendix O: Example of logistic regression model with its best parameters and how it is used in calculating model metrics

```
lr = Model('Logistic Regression',
           LogisticRegression(C=0.2, class_weight='balanced', solver='liblinear', multi_class='auto', random_state=42),
           #LogisticRegression(random_state=42),
           class_name='LogisticRegression',
           parameters = lr_parameters,
           default_parameters='random_state=42')
```

```
def calculate_model_metrics(df, y_variable):
    df = pd.read_sql_table(df, engine)
    folds = Model.create_walk_forward_folds(df, y_variable) #Time-split cross-validation folds
    for fold in folds:
        Model.y_train, Model.y_test, Model.x_train, Model.x_test = fold[0], fold[1], fold[2], fold[3]
        for model in Model.instances: #Iterates over created models
            Model.create_y_pred(model) #Calculates the predictions for the dependent variable
            Model.find_fold_accuracies(model) #Calculates the metrics for each fold
            Model.find_fold_precisions(model)
            Model.find_fold_recalls(model)
            Model.find_fold_fscores(model)
    for model in Model.instances:
        model.avg_accuracy = Model.find_average(model.fold_accuracies) #Averages the metrics across folds
        model.avg_precision = Model.find_average(model.fold_precisions)
        model.avg_recall = Model.find_average(model.fold_recalls)
        model.avg_fscore = Model.find_average(model.fold_fscores)
```

```
def create_y_pred(model):
    model.classifier.fit(Model.x_train, Model.y_train)
    model.y_pred = model.classifier.predict(Model.x_test)
```

```
def find_fold_precisions(model):
    precision = Model.find_precision(model.y_pred)
    model.fold_precisions = model.fold_precisions + [precision]
```

```
def find_precision(y_pred):
    precision = precision_score(Model.y_test, y_pred)
    return precision
```

Appendix P: Function to carry out time-split cross-validation

```python
def create_walk_forward_folds(df, y_variable):
    df['Date'] = pd.to_datetime(df['Date'])
    basedate = pd.Timestamp('1998-01-02') #First sample date
    df['Datedelta'] = df['Date'].apply(lambda x: (x - basedate).days) #Days since first sample
    df = df.drop(['Date'], axis=1)
    folds = []
    for quantile in [0.2, 0.4, 0.6, 0.8]: #Five quantils to create four folds
        test_quantile = quantile + 0.2 #Test quantile will be the quantile above training quantiles
        quantile_date = df['Datedelta'].quantile(quantile) #The date that splits quantiles
        test_quantile_date = df['Datedelta'].quantile(test_quantile)
        df = df[df[y_variable].notna()]
        df = df.bfill() #Any missing beginning data is back-filled
        df = df.ffill() #Any missing ending data is forward-filled
        train = df[df['Datedelta'] < quantile_date] #Dataset split on date splitting the quantiles
        test = df[df['Datedelta'] < test_quantile_date]
        test = test[test['Datedelta'] > quantile_date]
        y_train = train[y_variable] #Dependent variable column
        y_test = test[y_variable]
        #Invalid columns to keep in model training are removed
        x_train = train.drop(['Bear20', 'Bear10', 'Datedelta', 'Bear10_1M', 'Bear10_3M', 'Bear10_6M'], axis=1)
        x_test = test.drop(['Bear20', 'Bear10', 'Datedelta', 'Bear10_1M', 'Bear10_3M', 'Bear10_6M'], axis=1) #
        folds.append([y_train, y_test, x_train, x_test])
    return folds
```

Appendix Q: Function to undertake profit analysis

```python
def profit_analysis(df, y_variable, model):
    df = pd.read_sql_table(df, engine)
    df = df[df['Date'] < '2022-11-05'] #Keeping dataset sizes equal across datasets
    df = df[df['Date'] > '1998-01-01']
    Model.y_train, Model.y_test, Model.x_train, Model.x_test = Model.create_final_fold(df, y_variable)
    Model.create_y_pred(model) #Creates the predictions for the dependent variable
    prices = pd.read_sql_table('S&P_500_relative', engine)
    period_offset = {'Bear10': -1, #The correct number of weeks to shift data up to find correct returns for predicted period
                     'Bear10_1M': -5,
                     'Bear10_3M': -13,
                     'Bear10_6M': -25}
    prices['Close_RDP'] = prices['Close_RDP'].shift(periods=period_offset[y_variable]) #Shifts data as per above
    prices = prices[prices['Date'] > '2017-11-16'] #Testing dataset for profit analysis (fourth fold)
    prices = prices[prices['Date'] < '2022-11-05']
    assert model.y_pred.size == len(prices.index) #Checks prediction size matches testing dataset
    prices['Prediction'] = model.y_pred #Column to hold 0 or 1 based on prediction
    money = 1000
    for index, row in prices.iterrows(): #Iterates through weeks to calculate profit
        if row['Prediction'] == 0: #If non-bear prediction then the loss or gain for the week is calculated
            money = money * (1 + (row['Close_RDP'] / 100))
        print('Given the RDP of', row['Close_RDP'], 'and prediction of', row['Prediction'], 'money is now', money)
```

Appendix R: Terminal output of model metrics using the historical price & sentiment dataset

```
Logistic Regression accuracy: 0.9222516946654877
Logistic Regression precision: 0.7394634923168725
Logistic Regression recall: 0.8051452655325895
Logistic Regression fscore: 0.7654840328753372
Support Vector Classifier accuracy: 0.8504052460949013
Support Vector Classifier precision: 0.6366611477201232
Support Vector Classifier recall: 0.7499390281080421
Support Vector Classifier fscore: 0.642860192857631
Random Forest accuracy: 0.9212827880931329
Random Forest precision: 0.7516017316017316
Random Forest recall: 0.7953821413328456
Random Forest fscore: 0.7651708705689076
XGBoost accuracy: 0.8876289419392868
XGBoost precision: 0.7648249619482497
XGBoost recall: 0.6382780585025303
XGBoost fscore: 0.6752946841555553
Neural Network accuracy: 0.8599690539345711
Neural Network precision: 0.6550598264789974
Neural Network recall: 0.7990499817084324
Neural Network fscore: 0.6781537778226567
```

Appendix S: Terminal output of the final few weeks of profit analysis on XGBoost with combined dataset

```
Given the RDP of -1.21 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -4.04 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -3.29 and prediction of 1 money is now 1819.1131385756255
Given the RDP of 3.65 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -4.77 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -4.65 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -2.91 and prediction of 1 money is now 1819.1131385756255
Given the RDP of 1.51 and prediction of 1 money is now 1819.1131385756255
Given the RDP of -1.55 and prediction of 1 money is now 1819.1131385756255
Given the RDP of 4.74 and prediction of 1 money is now 1819.1131385756255
Given the RDP of 3.95 and prediction of 0 money is now 1890.968107549363
Given the RDP of -3.35 and prediction of 0 money is now 1827.6206759464594
Given the RDP of 5.9 and prediction of 0 money is now 1935.4502958273004
```

Table 7: Feature reduction final sets for historical price data

Forward Selection (to keep)	Backward Selection (to remove)
Bear10_lag_4	-Volume_RDP_lag_4 -Volume_RDP_lag_12 x3 -Volume_RDP_lag_1 -Volume_RDP_lag_2 -Volume_RDP_lag_8 -Close_RDP_lag_4 -Close_RDP_lag_12 x3 -Close_RDP_lag_8 x3 -Close_RDP_lag_2 -Close_RDP_lag_1 x2 -Bear10_lag_12 x2

Table 8: Feature reduction final sets for macroeconomic data

Forward Selection (to keep)	Backward Selection (to remove)
US Dollar per SDR x2 Consumption of fixed capital_x_RDP Social benefits_x_RDP Revenue_x_RDP	-Interest_x x2 -US Dollar per SDR, Period Average -Net/gross investment in nonfinancial assets_x -Change in Inventories, Nominal, Undjusted, Domestic Currency_RDP

Table 9: Feature reduction final sets for news sentiment data

Forward Selection (to keep)	Backward Selection (to remove)
BFE_LM_Negative x2	-LUS_LM_Negative x3
BFE_NRC_disgust x2	-LUS_LM_Uncertainty
LUS_NRC_sadness x2	-LUS_LM_Constraining
LUS_NRC_surprise	-LUS_LM_Weak_Modal
LUS_NRC_fear	-LUS_NRC_Positive
LUS_LM_Constraining	-LUS_NRC_trust x2
	-LUS_NRC_disgust
	-LUS_NRC_anger
	-BFE_NRC_anger
	-BFE_LM_Positive
	-BFE_LM_Uncertainty
	-BFE_LM_Constraining
	-BFE_LM_Weak_Modal
	-BFE_LM_Litigious
	-BFE_NRC_fear
	-BFE_NRC_Positive x2
	-BFE_NRC_trust

Table 10: Classification Metrics comparison (1 week ahead) with only historic price & macroeconomic data

Metrics	Accuracy	Precision	Recall	F-Score
LR	0.784	0.346	0.431	0.380
SVC	0.797	0.250	0.013	0.024
RF	0.805	0.614	**0.799**	**0.641**
XGB	**0.835**	**0.691**	0.658	0.595
NN	0.533	0.514	0.499	0.322
Average	0.751	0.483	0.480	0.392