Faculty of Natural, Mathematical & Engineering Sciences
Department of Informatics

King's College London
Strand Campus, London,
United Kingdom

7CCSMPRJ

Individual Project Submission 2024

Name:	Patrick Cresswell
Student Number:	23103862
Degree Programme:	Artificial Intelligence
Project Title:	Joint Human-AI decision making using xAI
Supervisor:	Ian Kenny
Word Count:	37,610 including source code

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☑ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature:

Date: August 6, 2024

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ Individual Project

Joint Human-AI decision making using xAI

Name: **Patrick Cresswell**
Student Number: 23103862
Course: Artificial Intelligence

Supervisor: Ian Kenny

This dissertation is submitted for the degree of MSc in Artificial Intelligence.

Abstract

The aim of this project is to show how Explainable Artificial Intelligence (xAI) methodologies can help us to understand the reasoning behind "back box" AI model decisions, and so enable models to be used in joint decision making with a human domain expert. This is explored on a use case in financial services which enables further analysis into how these techniques can be used to support model approval in a heavily regulated industry.

A feed-forward neural network is implemented to predict day on day equity price movement. Running across a portfolio of top, middle and low performers within the S&P 500, the model performs better than simpler trading strategies. Four xAI methodologies are implemented to provide a justification for these predictions. The methodologies cover decision rules, feature importance, similarity and counterfactuals. These approaches are integrated into an end-to-end process with the user at its centre. It is shown that these approaches are complementary and deliver a clear and consistent justification - understandable by a non technical user.

Additional related information was sourced outside of the model (market sentiment) to demonstrate insight that a human Subject Matter Expert (SME) brings to the decision making process. It was shown that having a user in the loop can result in better decisions than the model or the user behaving independently.

Finally, work was performed to develop alternative benchmark and challenger models. Supporting the xAI results, it was demonstrated how this information allows internal model validation or external regulators to assess model effectiveness. The level of transparency provided enables a reduction in the risk rating and so easier approval for use in business critical applications.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project background

The finance domain offers a great opportunity to apply AI approaches. On a personal level, where our money comes from and how we spend it is logged at the transaction level. Aggregating this across a household will allow you to forecast the spending patterns of a family. Aggregating this across a country will allow you to forecast an economy. At the global markets level, intra-day prices are openly published (albeit with a lag) across all exchange traded products allowing very detailed trend and forecasting analysis. These forecasts, if monetised, could lead to significant gains.

But markets can be irrational : Investors deciding to sell on a "belief" or following a "stock-bubble", or trainers selling for multiples of their retail price as they have become the next "must have" on TikTok. Humans are best placed to understand human behaviour – especially when irrational. So the optimal decision making model is surely a joint one across humans and AI. Pattern recognition approaches of machine learning coupled with human judgement and domain expertise can give us the best of all worlds.

The issue however is one of transparency. The reasoning behind an AI system outcome is not always available – they are often "black box" models. It is not possible to have a dialogue around a decision taken if one party cannot explain their thinking. This is then a significant barrier to usage.

Additionally, any algorithm or model needs to be tested and validated to ensure its correct operation and lack of bias. This is particularly important in highly regulated industries such as finance. Issues around transparency will make this approval difficult if not impossible.

1.2 Explainability

A solution to both of these challenges is proposed by explainable AI. xAI is a set of methods to "open up the black box" around artificial intelligence models. Explainability is the ability to understand why AI models make their decisions. This is separate from Interpretability which is looking to understand how AI models make their decisions. There are two fundamental approaches:

- Explainability by design : a model is able to make transparent its inner workings which led to the final prediction. For example : Decision trees. These simple models are typically good for non-complex datasets.
- Post-hoc explainability : Methods to extract information about patterns which the model has learnt in order to explain why it made its decision after the event. Typically more complex, these models can handle more complicated datasets and relationships between attributes. I will focus in this project on post-hoc explainability as the more complex models offer the greatest potential reward.

Post-hoc methods can be further divided into:

- Global scope : Methods looking to understand the overall logic and behaviour of a black box model.
- Local scope : Methods looking to explain the decisioning behind the final outcome for a specific instance.

Approaches for xAI can be grouped by their methodology as below. I explore each type in more detail in the next section:

- **Feature importance**. This approach quantifies the relative importance of different features to the outcome. This can be calculated by perturbation of the inputs or backward allocation of the final output. The output is typically numeric : ie the contribution of a feature to the outcome.
- **Decision rules**. This approach looks to copy the behaviour of the model being explained using simple easy-to-understand rules. The challenge however is if we can trust that this "simpler" model accurately reflects the essence of the decision being made by the actual model. The output is typically a tree or list : eg a decision tree outlining the path to the outcome.
- **Counterfactuals**. This approach identifies the (usually minimum set of) features to be changed in order to obtain the alternative outcome. The output is typically textual eg increasing X and decreasing Y would result in a new prediction of Z.

1.3 Joint Human-AI decisioning

A key outcome of this project is to show how AI generated recommendations can be integrated into an end-to-end business process along with a human expert in order to derive an outcome "better" than that of the AI or human working alone. To do this we need the justifications coming out of the xAI methodologies to be both clear and compelling. The user interface design is an important part of delivering this clarity. This could be the topic for a whole other study, and so I chose for this project to keep the implementation simple, focussing on demonstrating just the key results and including relevant contextual information to help the user judge the prediction accuracy.

Just as important as helping the user "trust" the recommendations made by the AI, is making sure that the users don't just accept the proposal without effective challenge. A 2021 paper [1] looked at the effects of AI explanations on complementary team performance. In some cases, adding explanations **increased** the chance that the human accepted an AI recommendation even when incorrect. I will address this risk through the end to end process design.

1.4 Project aim

To show how explainable AI methods can be used to enable joint decisioning across humans & AI – as well as to manage the model risk around their usage in a business critical / highly regulated domain.

1.5 Project objectives

- To implement a "black box" machine learning model for a typical finance use-case.
- To implement a number of explainable AI methods to justify its outcomes.
- To show how the output of these methods can be integrated together to enable joint human-AI decisioning.
- To show how the output of these methods can be used to manage model risk.

1.6 Report structure

I begin with three separate literature reviews. The first looks at where AI modelling techniques have been used within the finance space, and concludes with selection of the business use case for this project. The second literature review focuses on the different model types which have been used to implement this use case, and concludes with selection of the model to be used for this project. The third literature review looks across the landscape of xAI methodologies and concludes with selection of approaches to be implemented.

The report then covers the design and implementation of a solution for the selected use case, using the selected model, and including the selected xAI methodologies. The solution is then evaluated with regards model performance, scalability, and how well the xAI methods support joint decisioning with a human SME.

The report wraps up with an assessment of related legal, social, ethical and professional issues, before a final summary of findings, conclusions and next steps. The detailed code behind the implementation is then included within an appendix.

2 Literature Review

2.1 Business Use Case

My first objective was to look at where artificial intelligence modelling techniques have been used within the finance space and use the results of this analysis to pick the focus for my project. We were given a number of suggested digital libraries and databases of academic publications to use within this project:

- IEEE Xplore Digital Library.
- ACM (Association for Computing Machinery) Digital Library.
- Scopus.

I decided to focus on Scopus as it indexes both of the other sites. Searching proved difficult - many articles returned were completely unrelated to finance, with abstracts containing the word "finance" as a generic reference to areas where machine learning techniques have been used - rather than being the actual focus of the paper. I therefore reshaped my online database search to focus on keywords. I also focused my review on the most recent papers – filtering on the date range 2023-2024. I reviewed the abstracts for all 196 articles returned by Scopus(as of Feb 24) pulling together into the five themes as below.

- Keywords – "finance" and "artificial intelligence".
- Date range : 2023-2024.

2.1.1 Forecasting

- Stock prediction and portfolio selection. From my research, this is the most studied use case. Typical is Noxtrader [2] which uses a Long Short-Term Memory (LSTM) model to capture continuous price trends. Many implementations look at end of day data only, but Noxtrader updates the model intra-day in order to better adapt to market trends.
- Residual vehicle value prediction. Calculating the future residual value of a vehicle is a key part of car financing. Working with Hyundai in their paper [3] this group built a Deep Neural Network model leveraging used car data. With regards model validation and regulatory compliance, the team did not use xAI but showed consistency of operation using mean absolute error (MAE).
- Exchange rates. Similar to the above use case predicting stock returns, a number of articles look to predict foreign exchange rates. The typical approach uses a Long Short-Term Memory (LSTM) model - in this example [4] combined with a Convolutional Neural Network. Interesting in this paper is their including sentiment analysis (positive, negative, neutral) into the model input vector.

- Corporate financial performance. Assessing its financial performance allows a company to validate both its own operations, but also to benchmark itself against the competition. In this paper [5] the team use a variety of machine learning approaches to analyse the multiple metrics used. Interesting here is the extraction of qualitative information from annual financial reports.

- House rents. Focussing on the needs of property developers and financial institutions, this team [6] leveraged Deep Neural Network models to predict rental prices. Interesting in this paper is a recognition of the risks around the use of personal data, and the risk of information asymmetry with these sorts of tools only being available to large developers rather than renters - enabling market monopolies and unfair treatment.

2.1.2 Identifying fraud

- As one might expect, fraud detection is also a very well researched application for machine learning within finance. When compared to traditional techniques such as logical regression and decision trees, studies have shown [7] a material improvement in the use of modern approaches such as neural networks. Of interest in this paper was their use of ANOVA analysis across the different models selected. This approach looks to split out the variance seen within a dataset between systemic factors and random factors.

- The possibilities available using AI methods for natural language processing have also been explored with relation to fraud detection. This article [8] discusses methodologies for text mining and network analysis, and then shows their usage in analysing the Scopus database of papers in this topic.

- The specific case of detecting a rogue trader on a trading desk was discussed in [9]. A timeseries of behaviour is collected and then used to train a Long Short-Term Memory (LSTM) model. Predictions made by this model are then compared to actual behaviour.

2.1.3 Modelling financial stress

- Stress testing is a key part of both capital planning and risk management within a bank. A number of articles have been published which look to improve this activity using AI techniques. At the individual firm level, analysis was performed [10] into the default probabilities of over 2000 banks over 17 years. A number of machine learning models were assessed and the Shapley Additive Explanation (SHAP) xAI methodology was used to identify the most important features.

- Analysis has also been performed using machine learning techniques focussing specifically on the COVID-19 period. This research [11] used a Bidirectional Long Short-Term Memory Network (BLSTM) model and also used a variation of the

SHAP xAI methodology for feature selection. Analysis was performed at the macro level looking at the USA, Emerging and Emerged countries. Results were positive regarding the models prediction accuracy.

2.1.4 Risk assessment

- Credit Risk is another well researched area. Banks collect a huge amount of data on their customers and recent advances in machine learning techniques have opened up the opportunity to leverage it more widely. I know from my own experience however, the challenges in bringing together a single customer view across multiple systems. This was addressed in [12] through the use of a federated learning approach where the multiple systems involved each train a local model rather than pushing all data to a central server. This is also an advantage from a data protection point of view.

- Operational Risk. More of an art than a science, I also know from my own experience that improved modelling of operational risk can save banks millions in capital charges. This paper [13] proposes a paradigm shift in the modelling and management of operational risk underpinned by ontologies, machine learning and natural language processing technologies.

2.1.5 Operations

- We have all become more used to talking to companies through chatbots rather than face to face, or even over the phone. Finance is no exception and this paper [14] looks at how IVAs (Intelligent Virtual Assistants) influence customer responses in the financial service setting. One interesting finding was that today customers are preferring IVA over (human) agents since they can obtain solutions to their questions considerably faster and the interactions are a lot smoother.

- On the logistics side, one very practical use case for a retail bank is that of planning the replenishment policies for Automated Teller Machines (ATM). This paper [15] shows how machine learning approaches can be used to optimize both the experience for the customer and the costs for the bank.

- Payment processing using distributed ledger technologies continue to be part of the Fintech landscape. This paper [16] shows how machine learning algorithms can be used to streamline payment operation capabilities and process promptness across instant payment networks and infrastructures.

- Manual process automation : On a very practical level, machine learning approaches can be used to automate manual tasks which (I know from my own experience) can take up a significant part of the working day. This paper [17] uses a Cascade-TableNet model to extract and classify financial tables in enterprise analysis reports.

2.1.6 Final selection

I have chosen within this project to focus on stock movement prediction. This is primarily driven by data availability - many of the above use cases leverage private-side data which will be very challenging to get hold of. Stock prediction and portfolio selection is also a very active area of research and so provides a wealth of material available to review. It will also allow me to look across both tabular data and textual data - eg historic prices as well as news articles.

2.2 Model Landscape

With the business use case decided, my second objective was to pick the model to use within my project. To do this, I performed a second literature review using Scopus, focussing on articles covering stock prediction using machine learning. I performed my search with the filter outlined below. I then reviewed the abstracts for all 261 articles returned (as of Feb 24) to understand the type of model used within each piece of work. I refreshed this analysis in Jul 24 as part of my write-up. The article count increased to 387 and from these a total of 313 included details in the abstract or index detailing the model types covered. The table below shows how many articles out of these 313 referenced that model type. If an article covers multiple models it is counted against each model type covered.

- Keywords : stock AND prediction AND "machine learning".
- Date range : 2023-2024.

Methodology	**Sub-type**	**Articles**	**% of total**
Regression Models		109	35%
Decision Trees		21	7%
Boosting & Bagging		52	17%
K Nearest Neighbour		20	6%
Naïve Bayes		14	4%
Fuzzy Classifiers		8	3%
Support Vector Machine		56	18%
Neural Networks	FF-NN	138	44%
	LSTM / RNN / GRU	150	48%
	Transformers	15	5%
Large Language Models		8	3%

Table 1: Model Landscape Literature Review Summary

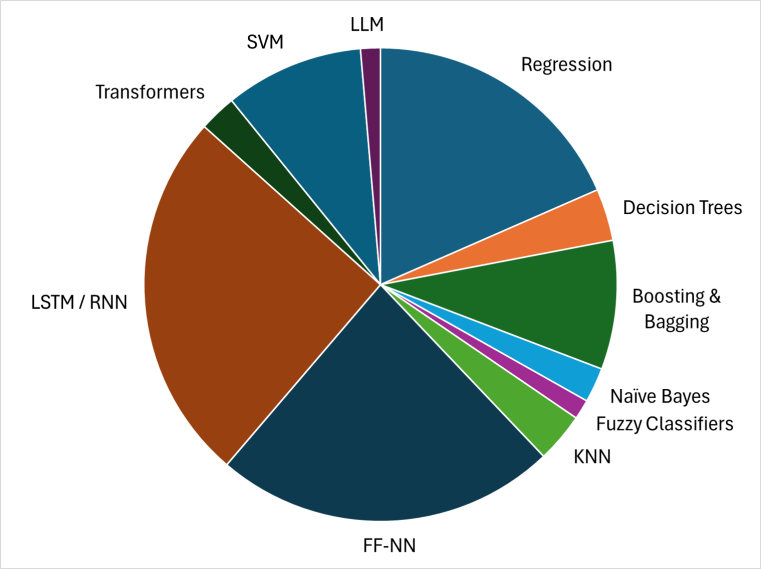


Figure 1: Model Landscape Literature Review Summary

2.2.1 Regression Models

When looking to predict a future numeric value, the simplest approach is to use a linear regression model. Multivariable linear regression models were included in 109 of the 388 papers assessed. Their primary use however was one of acting as a comparison benchmark for more sophisticated approaches. They are simple to understand but have significant limitations when used in higher dimensions - especially around over-fitting [18].

2.2.2 Decision Trees

Decision Tree solutions were assessed in a total of 21 articles across the analysis period. Like Regression Models, they have an advantage in being easy to understand and explainable by design. As outlined by Russell & Norvig [18] their expressiveness is however limited and certain functions cannot be represented concisely. They are however a good comparison model to other more advanced approaches. I saw this in the literature review where all but one decision tree implementation did so alongside one or more alternative approaches.

2.2.3 Boosting

Boosting is an ensemble method which attempts to mitigate some of the limitations of the Decision Tree approach - eg the ADABOOST algorithm. Here, a decision tree is trained using the full training dataset with all samples weighted equally. The weights are then adjusted to pay more attention to the examples which the first model got wrong. A second, and then a third model is added - each focussing on "boosting" the overall

performance across the ensemble until the overall performance meets the desired level. Gradient boosting extends this further by paying attention not to specific examples but to the gradient between the right answers and the answers given by the previous hypothesis.

2.2.4 Random Forests

Random forests are another ensemble methodology using decision trees - extending the idea of bagging. Within bagging, we generate X unique training sets from the original data using sampling with replacement. We then train X decision trees using these datasets. To generate a prediction, we aggregate the predictions from the X models on the new data - eg using the majority vote for a binary classification. The Random forest approach extends this further by randomly varying the features chosen rather than the data when creating each tree.

Gradient Boosting in particular has become very popular in this space and these ensemble approaches are included in 52 articles assessed. While the ensemble methods increase performance, they do however reduce explainability - requiring the use of post-hoc explainability approaches.

2.2.5 K - Nearest Neighbour

K-nearest neighbour models are also seen within a material number of articles within my review - 20 across the study. Like Decision Trees, their primary value is however as a benchmark against which to assess a more complex model.

2.2.6 Naive Bayes

The sheer number of drivers behind a stock price makes its analysis a good candidate for probability theory. Naive Bayes models are included in 14 articles across the study. Naive Bayes models are widely used for natural language processing, and drilling down further it is seen that their use here is mainly around sentiment analysis [18].

2.2.7 Fuzzy Classifiers

Whereas normal boolean logic works in a certain world of true and false, fuzzy logic works with shades of grey - rather more like we do as humans, with degrees of acceptance or rejection of a proposition [18]. A total of 8 articles within my review look at various flavours of fuzzy classifier with regards stock price prediction, typically compared against a Neural Network implementation.

2.2.8 Support Vector Machines

Support Vector Machines (SVM) were (according to Russell & Norvig [18]) the most popular approach in the early 2000's for supervised machine learning - overtaken by deep learning and random forests. They offer benefits in terms of simplicity but are also

often able to handle data which is non-linear in the original input space by mapping into a higher dimension - the so-called kernel trick. SVM were included within 56 articles assessed within my research.

2.2.9 Neural Networks

Neural Networks (of various flavours) are by far the most common model type seen in studies into stock market movement prediction - referenced in 224 of the 313 articles within my study. As outlined by Russell & Norvig [18] Neural Networks are sufficiently expressive to handle the inherent complexity required to solve real-world problems. The following network architectures are all explored within articles seen within my literature review.

- Feed-Forward Neural Network : A deep (multi-layered) network with connections in one (forward) direction only. Each node runs a calculation on its inputs and sends the results forward to its successors[18].
- Recurrent Neural Networks : Allow for cycles in the computation process - ie neurons may take as an input a value computed from their own output at an earlier step in the process. This gives the model an internal "memory" which is particularly useful in analysing time-series data[18].
- Long Short Term Memory : Similar to a Recurrent Neural network, the LSTM architecture consists of a memory cell along with a number of gating units controlling whether each element of the memory cell is : remembered or forgotten (forget gate); updated additively by the new input vector information (input gate); transferred to short term memory or not (output gate)[18].
- Transformers : A more recent deep learning architecture developed by Google. The key concept is one of attention where at each layer the data element is contextualized within the scope of a context window including other data elements. Transformers require less training time that RNN or LSTM and are the building blocks behind large language models. Within my use case, the context is across the dimension of time rather than a sentence or paragraph within a document.

2.2.10 Large Language Models

A small number (8) of articles looked to large language models such as Chat GPT in order to perform "sentiment trading" - where the decision to long or short a stock is based on analysis of market sentiment. These were typically then compared to more "traditional" approaches.

2.2.11 Final selection

For my main model, I will be using a Neural Network - specifically a Feed-Forward Neural Network with multiple hidden layers. This type of model meets the criteria of being a

"black-box" model and is mature enough to be well supported within the various xAI frameworks. I will also be implementing a Decision Tree model as a "challenger" to my main model. As above, this type of model is explainable by design and so will be useful when it comes to comparing results. Finally, I will be implementing seven alternative models as listed below to act as comparison benchmarks for model performance analysis. This gives me good coverage on the various model types discussed within the literature review.

- K - Nearest Neighbour
- Support Vector Machines
- Gaussian Process Classifier
- Random Forest
- Gradient Boosting
- Ada Boost
- Naive Bayes

2.3 xAI Methodologies

With the business use case and model decided, my final goal before starting implementation was to review the range of xAI methodologies available, understand their strengths and weaknesses, and then pick those which I will integrate within my project. I performed a third Scopus literature review using the following criteria:

- Keywords : ("explainable AI" OR "explainable artificial intelligence") AND finance.
- Date range : 2023-2024.

I reviewed the full text for all 52 articles returned (as of Feb 24) and followed the references within these articles to review a total of 115 papers. From these papers, I created a list of 109 different methodologies which I then filtered with regards my use case - applicable for an feed-forward neural network with an example usage given for tabular data (ie not computer vision specific). This reduced the list to 31 possible approaches which I then assessed - grouped as below.

Method	Category	Models Supported	Notes
Ada-WHIPS	N/A	Ensemble	Not applicable for Neural Networks
ALMMo-0*	N/A	NFuzzy	Not applicable for Neural Networks
Anchors	Decision Rules	Agnostic	
ANFIS	N/A	Fuzzy; GA; NeuralNetwork	Not applicable for Neural Networks
ApparentFlow-net	N/A	NeuralNetwork	Medical domain specific
Attention Maps	N/A	NeuralNetwork	Computer vision specific
BB-BC IT2FLS	N/A	Fuzzy	Not applicable for Neural Networks
BEN	N/A	NeuralNetwork	Computer vision specific
BN	N/A	Bayesian	Not applicable for Neural Networks
BRL	N/A	Bayesian	Not applicable for Neural Networks
CAM	N/A	NeuralNetwork	Computer vision specific
Candlestick Plots	Other	NeuralNetwork	Linked to how traders perform techninal analysis
CART	N/A	TreeBased	Not applicable for Neural Networks
CASTLE	Decision Rules	Agnostic	
Causal Importance	Decision Rules	NeuralNetwork	
CFCMC	N/A	Fuzzy	Not applicable for Neural Networks
CGP	N/A	Unsorted	Not applicable for Neural Networks
CIE	Decision Rules	Ensemble; NeuralNetwork; SVM	Confident Itemsets
CIT2FS	N/A	Fuzzy	Not applicable for Neural Networks
Cluster-TREPAN	Decision Rules		
Concept Attribution	N/A	NeuralNetwork	Computer vision specific
Contrastive Explanations Method (CEM)	Counterfactuals		
CTree	N/A	TreeBased	Not applicable for Neural Networks
Decision Tree	N/A	NeuralNetwork; TreeBased	A model - not an XAI approach
DeconvNet	N/A	NeuralNetwork	Computer vision specific
Deep-LIFT	Feature Importance		Similar to LRP. Backpropogation Method
DICE : Diverse counterfactuals	Counterfactuals		
DIFFI	N/A	Ensemble	Not applicable for Neural Networks
DTD	N/A	NeuralNetwork	Computer vision specific
ELI5	N/A	Agnostic	Manufacturing specific
Encoder-Decoder	N/A	NeuralNetwork	Computer vision specific
eUD3.5	N/A	Ensemble	Not applicable for Neural Networks
Explainable Neural-Symbolic Learning	Other		
FACE	Counterfactuals	NeuralNetwork	Feasible and Actionable Counterfactual Explanations
FDE	N/A	Ensemble; NeuralNetwork; NearestNeighbour; SVM	Manufacturing specific
Feature Pattern	N/A	Ensemble	Not applicable for Neural Networks
FFT	N/A	TreeBased	Not applicable for Neural Networks
FINGRAM	N/A	TreeBased	Not applicable for Neural Networks
FormuCaseViz	N/A	CaseBasedReasoning	Not applicable for Neural Networks
FURIA	N/A	Fuzzy	Not applicable for Neural Networks
Fuzzy LeNet	N/A	Fuzzy	Not applicable for Neural Networks
Fuzzy Relations	N/A	Fuzzy	Not applicable for Neural Networks
gbt-HIPS	N/A	Ensemble	Not applicable for Neural Networks
Generation	N/A	NeuralNetwork	Social Media specific
GLAS	N/A	Agnostic	Computer vision specific
Goldeneye	Feature Importance		
GRACE	Counterfactuals		
Grad-CAM	N/A	NeuralNetwork	Computer vision specific
Gradient Boosted Trees	N/A		A model - not an XAI approach
Growing Spheres	N/A	Ensemble; SVM	Not applicable for Neural Networks
HFS	N/A	Fuzzy	Not applicable for Neural Networks
iChIMP	N/A	NFuzzy	Not applicable for Neural Networks
ICM	N/A	CaseBasedReasoning	Not applicable for Neural Networks
iNNvestigate	N/A	NeuralNetwork	Medical domain specific
Integrated Gradients	Feature Importance		
Interpretable Filters	N/A	NeuralNetwork	Computer vision specific
J48	N/A	TreeBased	Not applicable for Neural Networks
Knowledge Graph	N/A	NeuralNetwork	Recommendation use case specific
KSL	N/A	NeuralNetwork	Medical domain specific
LENS	Other		Local Explanations via Necessity and Sufficiency (LENS)
LEWIS	Counterfactuals	Agnostic	Paper gives an example in the finance space
LGNN	N/A	NeuralNetwork	Changes backpropogation approach, doesn't actually produce an explain
LIME + extensions : ALIME / DLIME / ILIME	Feature Importance		Putting more structure around the LIME random perturation and feature selection
LORE	Decision Rules	Ensemble; NeuralNetwork; SVN	Local rule-based explanation method
LPS	N/A	NeuralNetwork	Linguistic Protoforms. Helps make xAI results more readable when generated
LRCN	N/A	NeuralNetwork	Computer vision specific
LRP	Feature Importance	NeuralNetwork; SVN	Layer-Wise Relevance Propagation. Core technique
LSP	N/A	Unsorted	Not applicable for Neural Networks
MANE	Feature Importance		
MAPLE	N/A	Agnostic	A model - not an XAI approach
MTDT	N/A	TreeBased	Not applicable for Neural Networks
MUSE	Decision Rules		
Mutual Importance	N/A	Linear	Not applicable for Neural Networks
MWC, MWP	Feature Importance	NeuralNetwork	Most weighted path / Most weighted combination
Neuro Fuzzy Model	N/A		A model - not an XAI approach
Nilpotent Logic Operators	N/A	NeuralNetwork	Changes the activation function, doesn't actually produce an explain
NLG	N/A	NeuralNetwork	Specific for Hopfield NN - used for implememnting associative memory
NMF	N/A	NeuralNetwork	Computer vision specific
OC-Tree	N/A	TreeBased	Not applicable for Neural Networks
Ontological Perturbation	N/A	NeuralNetwork	Medical domain specific
PAES-RCS	N/A	Fuzzy	Not applicable for Neural Networks
PASTLE	Feature Importance	Agnostic	Pivot-aided space transformation for local explanations
Prescience	N/A	Ensemble	Not applicable for Neural Networks
PRVC	N/A	CaseBasedReasoning	Not applicable for Neural Networks
PSL	N/A	Unsorted	Not applicable for Neural Networks
QMC	N/A	NeuralNetwork	Computer vision specific
QSAR	N/A	NeuralNetwork	Medical domain specific
Rational Shapley Values	Feature Importance		David's paper
RAVA	N/A	Agnostic	Tool fof exploriing multidimensional data, doesn't actually produce an explain
RBIA	N/A	CaseBasedReasoning	Not applicable for Neural Networks
RetainVis	N/A	NeuralNetwork	Medical domain specific
RISE	N/A	NeuralNetwork	Computer vision specific
RPART	N/A	TreeBased	Not applicable for Neural Networks
RuleMatrix	Decision Rules	Agnostic	
SHAP + Extensions	Feature Importance	Agnostic	Deep-SHAP, Kernel-SHAP
Shapelet Tweaking	N/A	Ensemble	Not applicable for Neural Networks
SLRP	Feature Importance	NeuralNetwork	Selective Layer-wise Relevance Propagation
Sparsity	Counterfactuals		
SRM	N/A	Unsorted	Not applicable for Neural Networks
SurvLIME-KS	N/A	Agnostic	Medical domain specific
TCBR	N/A	CaseBasedReasoning	Not applicable for Neural Networks
Template-Based Natural Language Generation	N/A	Unsorted	Not applicable for Neural Networks
Time-Varying Neighbourhood	N/A	NeuralNetwork	Recommendation use case specific
TREPAN	Decision Rules	NeuralNetwork	
Tripartite Graph	N/A	Unsorted	Not applicable for Neural Networks
"What I know" (WIK)	Other	Agnostic	
WM Algorithm	N/A	Fuzzy	Not applicable for Neural Networks
xNN	Other	NeuralNetwork	Explainability by design
XRAI	N/A	NeuralNetwork	Self-driving car specific

Figure 2: xAI Approach Review

2.3.1 Feature importance

The feature importance approach outputs the ranked set of features that lead to the decision made.

- SHAP. Scope : Global explanation. Approach : Perturbation method. Additional detail : Models the features as "players" and leverages game theory to calculate the marginal contribution of each player [19].
- Layer-wise Relevance Propagation (LRP). Scope : Global explanation. Approach : Backpropogation method. Additional detail : Starting from a network's output layer and backpropagating up to the input layer, LRP redistributes the prediction functions in their opposite order [20].
- Deep LIFT. Scope : Global explanation. Approach : Backpropogation method. Additional detail : Similar to LRP but focussed on Deep NN [21].
- Deep SHAP. Scope : Global explanation. Approach : Backpropogation method. Additional detail : Integrates SHAP with DeepLIFT [19].
- LIME. Scope : Local explanation. Approach : Perturbation method. Additional detail : Generates dummy instances in the neighbourhood of an instance and approximates to an interpretable linear model [22].
- PASTLE. Scope : Local explanation. Approach : Perturbation method. Expands on LIME by introducing pivots - representative points of regions where the model exhibits different behaviours [23].
- MANE - Model-Agnostic Non-linear Explanations. Scope : Local explanation. Approach : Perturbation method. Expands on LIME by looking across multiple features – allowing linear regression models to accurately fit the local non-linear boundary of a deep learning model [24].
- Rational Shapley Values. Scope : Global explanation. Approach : Perturbation method. Looks to improve on SHAP by the use of counterfactuals – real or synthetic data points that differ from the instance being explained in a specific way [25].
- Other models. Goldeneye : Focuses on the associations between attributes in a dataset [26]; Integrated Gradients : Similar in approach to DeepLIFT [27]; Selective LRP : Enhances LRP by selecting only those activations with a positive gradient [28].

2.3.2 Decision rules

The decision rule approach outputs a decision tree approximation of the model.

- TREPAN. Scope : Global explanation. Approach : Perturbation method. Additional detail : Queries the neural network to build a decision tree that approximates the concepts represented by the networks by maximizing the gain ratio [29].
- Anchors. Scope : Local explanation. Approach : Perturbation method. Additional detail : An anchor explanation is a rule that sufficiently "anchors" the prediction locally – such that changes to the rest of the feature values of the instance do not matter [30].
- LORE. Scope : Local explanation. Approach : Perturbation method. Additional detail : Combines decision rules and Counterfactuals. Creates the rules to explain the decision made along with a set of counterfactual rules suggesting the changes in the instance features that lead to a different outcome [31].
- MUSE. Scope : Local explanation. Approach : Perturbation method. Additional detail : Integrates user input around features of interest [32].
- Other models. Cluster-TREPAN : Looks to create a better decision tree through clustering [29] (many others like this); RuleMatrix : Focuses on the visualisation of the rules generated [33]; Confident Item sets : Discretizes the feature space and constructs explanations for smaller subspaces, then the explanations can be combined to represent the black-box model's behaviour in larger subspaces [34].

2.3.3 Counterfactuals

The Counterfactual approach outputs the features that need to be changed in order to obtain a certain outcome.

- Contrastive Explanations Method (CEM). Scope : Global explanation. Approach : Perturbation method. Additional detail : Highlights not only the pertinent positives but also the pertinent negatives. A pertinent positive (PP) is a factor whose presence is minimally sufficient in justifying the final classification. A pertinent negative (PN) is a factor whose absence is necessary in asserting the final classification [35].
- GRACE. Scope : Global explanation. Approach : Perturbation method. Additional detail : Generating interventive contrastive samples for model explanation [36].
- DICE : Diverse Counterfactual Explanations. Scope : Local explanation. Approach : Perturbation method. Additional detail : Generating a set of counterfactuals that are diverse and well approximate local decision boundaries [37].
- Other models. Other approaches focus on how to generate the examples. FACE : Feasible and Actionable Counterfactual Explanations [38]; Sparcity CEM : Minimising the number of features in the counterfactual [39]; LEWIS : Local and Global explanations [40].

2.3.4 Other approaches

- Example Based Explanations : Showing an example of an instance in the training set that is similar to the input data to be inferred. "This X is a Y because a similar X' is a Y" [41].
- Local Explanations via Necessity and Sufficiency (LENS). Scope : Local explanation. Approach : Perturbation method. Additional detail : Brings together feature attributions, rule lists and counterfactuals. Focuses on what is needed for logical sufficiency and necessity [42].
- Explainability of Neural Networks through Architecture Constraints (xNN). Scope : Global explanation. Approach : Explainability by design. Additional detail : Balancing prediction accuracy and model explainability in the design stage. Focuses on additivity, sparsity, orthogonality, and smoothness [43].
- Integrating deep learning with expert knowledge graphs. Scope : Local explanation. Approach : Perturbation method. Additional detail : Explainable Neural-Symbolic Learning [44].
- Explainable Deep Convolutional Candlestick Learner. Scope : Local explanation. Approach : Perturbation method. Additional detail : Uses candlestick plots to explain the output predictions of a deep CNN. The timeseries is encoded into a matrix and a local search attack used to understand which "pixels" are important to the final classification [45].

2.3.5 Final selection

My plan is to implement one of each approach type listed above in order to understand how best to integrate into a joint decisioning framework.

2.3.5.1 Feature importance : SHAP Reference : Russell & Norvig [18]. Shapley values are a concept introduced by Lloyd Shapley in the 1950's relating to cooperative game theory. In a cooperative game, multiple agents form agreements with one another in order to receive extra value compared to what they would get by acting alone. The game has N players and a utility function u for which every subset of players $S \subseteq N$ gets the value obtained should the group choose to work together. The question that Shapley values look to answer is how to best allocate the utility $u(S)$ among the players within the coalition.

Shapleys answer was to allocate the value related to how much each player contributed to creating it. The marginal contribution that a player y makes to a coalition S is the value that y would add (or remove) should y join the coalition S.

In calculating Shapley values we need to consider all possible ways that the coalition can form, and then consider the value that y adds to the players preceding them in the ordering. The player y should be rewarded the **average marginal contribution** that they make over all possible orderings to the set of players preceding them in the ordering.

Applying this concept to our xAI explain gives us a framework for understanding feature importance. The Shapley value is the average marginal contribution of a feature value to the outcome, across all possible orderings of features. Full details are in the original paper "A Unified Approach to Interpreting Model Predictions" [19]

2.3.5.2 Decision rules : Anchors Anchors [30] is a model-agnostic approach to explain the behaviour of complex models with if-then rules called "anchors". An anchor explanation is a rule that sufficiently "anchors" the prediction such that changes to the rest of the feature values will not affect the outcome. For all instances where the anchor holds, the prediction will be the same. Anchors are constructed iteratively as described below - with full details in the original paper "Anchors: High-Precision Model-Agnostic Explanations" [30].

1. Generate candidates : A candidate anchor is constructed incrementally - starting with an empty rule and adding to its definition additional feature predicates which increase coverage.

2. Find the set of "best" candidates : The best candidates generated each iteration are defined as those with the highest precision.

3. Select the anchor with the highest coverage as the new benchmark. If the precision target is met, stop. Otherwise go back to step one.

2.3.5.3 Counterfactuals : DICE Counterfactual explanations show feature perturbed versions of the prediction input data which result in the opposite outcome. As outlined in the original paper "Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations" [37] it is important that counterfactual examples be balanced between including a wide range of suggested changes (diversity) and the relative ease of adopting the proposal (proximity to the original input). Counterfactual examples are found using gradient descent to optimise the following function:

$$C(x) = \arg\min_{c_1,...c_k} \frac{1}{k}\sum_{i=1}^{k} yloss(f(c_i), y) + \frac{\lambda_1}{k}\sum_{i=1}^{k} dist(c_i, x) - \lambda_2 dpp(c_1, ...c_k)$$

- yloss : A perturbation of inputs c into the model f which minimises the loss against the opposite prediction class y.

- dist : A proximity term which measures the vector difference between the original input and the counterfactual example features c. The objective is to minimise distance.

- dpp : A diversity term which leverages the determinantal point processes - a concept introduced in the paper "Determinantal point processes for machine learning" [46] in order to find diverse sets of high quality search results in real-world applications.

2.3.5.4 Other approaches : Example based explanation Perhaps one of the most intuitive explanation techniques is to show an example training instance which led to the same outcome and has similar input data to that of the date to be explained - ie "This X is a Y because a similar X' is a Y". This approach also provides insight around the training dataset. If the selected "most similar" example returns a low match rate then we have a reduced confidence in the model results as its training has not included the pattern being seen today [41].

3 Specification and Design

3.1 An integrated Human-AI decisioning process

As mentioned within my introduction, in order to drive an outcome "better" than that of the AI or human expert working alone, the xAI approaches detailed above need to be integrated together into an-end-to-end process with the user. Across my literature review, articles compared and contrasted one approach with that of others, but I found no examples of work to integrate different, complementary approaches together into an overall framework. This section contains my proposal to address this question agnostic to a specific use-case. Design criteria are as follows:

- From the system side, maintain the clarity of justification messaging - only present information if it adds-value.
- From the user side, reflect the fact that they may or may not know the impact of additional information on the model inputs.
- Support multiple iterations of analysis and offer a what-if capability.
- Keep the user interface simple. I use the term "report" within the remainder of this section but this can also be read as a "view" within a wider user environment.

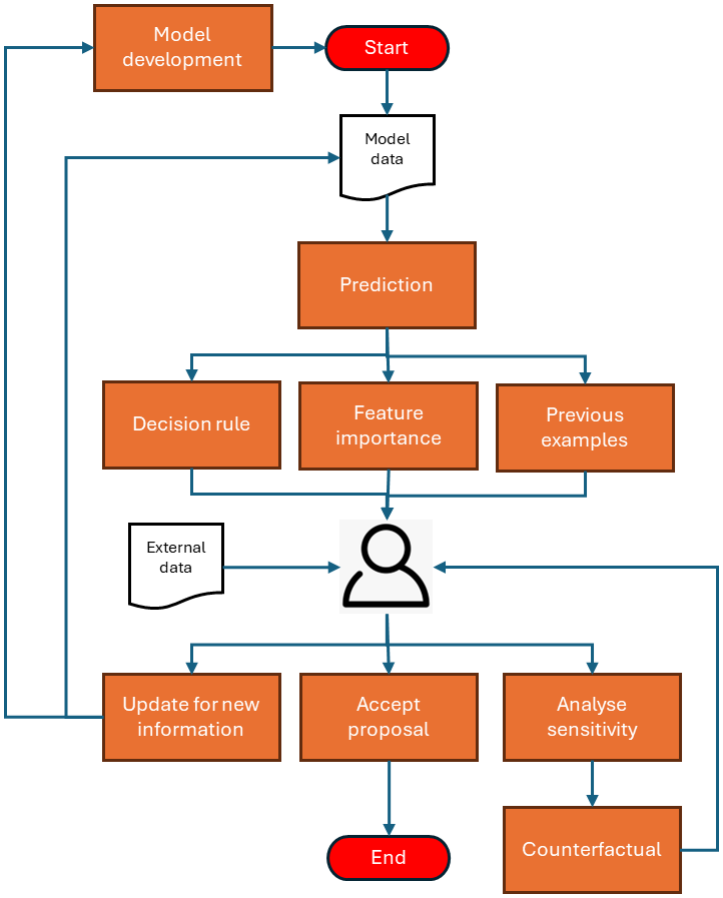


Figure 3: Integrated User-AI Decisioning

3.1.1 Model Development

We begin with model development and the release into production of the model. The model uses specific feature data in generating its prediction.

3.1.2 Prediction

Using the latest feature data, the model runs the prediction function.

3.1.3 Justification

Three complementary explain reports are then generated to justify the proposal:

- Decision rule report : Using the Anchor methodology. "The decision taken was X due to predicates A and B both being True". This is our starting point as is the most intuitive, easy to understand representation of the decision made.

- Feature importance : Using the SHAP methodology. From the decision rule, we know the features involved in driving the decision, but which features were most important and which "pulled" the prediction towards the outcome / "pushed" the prediction away from the outcome ? The SHAP feature importance view puts the features included in the Decision rule output into context.
- Previous examples : So we understand the logic behind the decision, but have we seen this pattern before ? Using the Example based explanation technique gives us confidence (or not) in the model based on whether its training included a similar set of inputs to that being decided.

3.1.4 User Review

The system now presents its proposal and justifications to the user. The user brings to the decision making additional information not currently known to the model. There are then a number of outcome paths:

3.1.5 Agreement

The justification is compelling and the proposal is agreed. The process concludes.

3.1.6 Re-run analysis based on quantified feature impact

The user is able to update the input data used in the prediction to reflect new knowledge. This update is performed and the user re-runs the prediction along with the explains. Example for my use case : The Desk Head has additional information about a company and we're coming into earnings season. He disagrees with what the market has priced in for the price to earnings ratio and thinks that should be X rather than Y. In a human-human dialogue, this would be a case of "what-if" that the Desk Head would ask of the Junior Analyst.

3.1.7 Re-run analysis based on uncertain feature impact

The user is uncertain as to the impact of new knowledge on the model data and needs to run one or more what-if analysis. This is done through tailoring the configuration of the Counterfactual xAI methodology and feeding the results back to the user. Example for my use case : The Desk Head has additional information about a company which will impact its price but is unable to quantify what the impact to the feature(s) will be. The user updates the configuration and re-runs the explain. This answers the question - what would need to be the changed in feature X in order for the prediction to change ?

3.1.8 The model is invalid and needs re-training / further development

The new knowledge available to the user invalidates the model proposal. Further model development is needed. Example for my use case : The Desk Head has additional

information about a company which may impact its price but the model system as currently implemented is unable to reflect this. An example would be a stock-split not yet reflected in the dataset. Re-training or further model development is needed.

3.2 A stock market movement prediction framework

Starting with the above generic process design, this section details the specification and design for my selected use case. The objective of the framework is to predict an increase or decrease in tomorrow's price of a stock in order to then take a long or short position in the next day's trading. Design criteria are as follows:

- Leverage Python. From my experience working in the financial services sector, this is the language of choice across all Tier-1 banks.
- Deliver predictions quickly. The system needs to generate predictions for tomorrow within 1-2 hours of today's market close in order to then be ready to make the trades at market open on the following day.
- Support different model configurations. Allow the hidden layer architecture to be flexible in order to provide the best performance.
- Support different run configurations. Allow the user to run one, many or all of the different elements of the process.

3.2.1 Actors

The end to end system involves the following actors

- Desk Head : Runs an equities desk within the front office. Responsible for PnL. Makes the final call on trades.
- Stock movement prediction system : Predicts the future direction of stocks, proposes a course of trading to maximise profit and explains its reasoning.
- Model Developer : Builds, tests and deploys the prediction system.

3.2.2 System architecture

The system design components are broken down in the diagram below. The rest of this section discusses each component (in blue) in more detail.

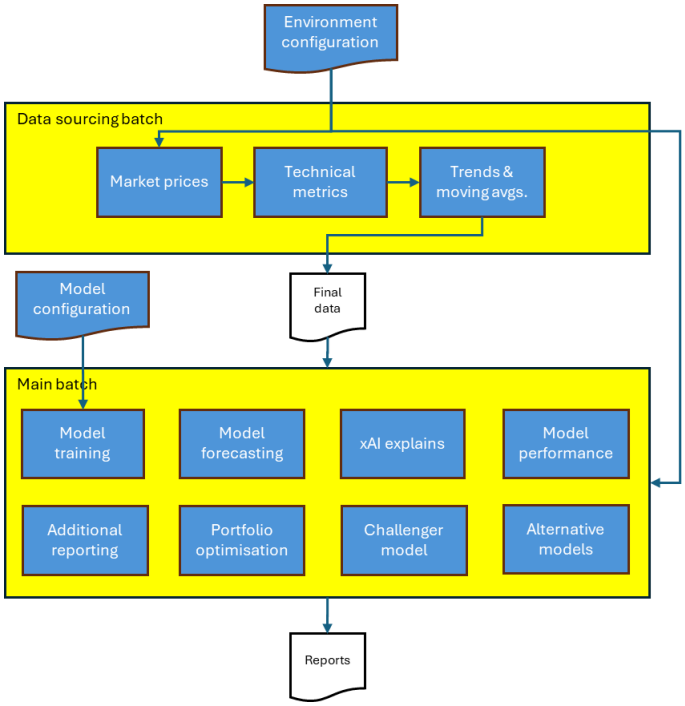


Figure 4: Overall system architecture

3.2.3 Environment Configuration

A configuration file (.env) will be used to set up the different batch runs and also to support the different use cases within the overall end to end process. The configuration file will be fully editable by the user and control the following elements of the solution:

- Where to store the data, reports and models.
- The amount of historical data to be used.
- Boolean batch configuration flags to control which processes are run each time the main batch is executed.

3.2.4 Model Configuration

A model configuration file will list each stock ticker to include within the model system. A model will be trained for each stock and this file will also store the details of the hidden layer architecture to use for each stock when creating the model.

3.2.5 Data sourcing batch

The data sourcing batch will run daily following market close and will source the end of day (EOD) prices for each stock included within the portfolio. To complement the

market data I will also source a number of technical financial metrics. These calculated measures are used by portfolio analysts and traders within the real world to decide when to buy and sell specific stocks, and so will prove useful in the automated solution. As an example, the simplest such metric is a moving average indicator. This calculates each day the average price over a specified time period (eg 5 days). The final data sourcing element within my design is sentiment data. The goal of sentiment data is to use historical news feeds to assess the market sentiment (bullish / bearish / neutral) on the future stock price. Once all external data is sourced, the data sourcing batch will calculate a number of trends across the different metrics which will also be used by the model. The final set of measures is defined in detail within the data sourcing implementation within Section 4.3.

3.2.6 Main batch

The main batch will run daily on completion of the data sourcing run. The core process within the main batch is the model prediction function. A movement prediction will be made for all stocks within the portfolio. The remainder of the batch execution is controlled by the Environment Configuration file. Environment variables will be created for each process as listed below, and the processes will be executed if the associated flag is set to be TRUE.

3.2.6.1 Model training A model will be trained for each stock specified within the Model Configuration file using the hidden layer architecture defined within the same file. The model will be trained using the measures, trends and averages generated within the data sourcing batch.

3.2.6.2 xAI explains Execution of the xAI explains will be controlled by a specific flag for each approach. If set, an explain will be generated for each stock within the portfolio. Four explain methodologies will be covered namely Decision Rules, Feature Importance, Counterfactuals and Similarity.

3.2.6.3 Model performance To support model testing, a model performance function will score the accuracy of the model predictions across both the training and testing datasets. The results will also be backtested - ie compared to realised results in order to assess the return delivered by the model over that period.

3.2.6.4 Additional reporting To support the user review, the following three additional reports will be created: A daily report for each stock within the portfolio summarising the input data used in today's prediction; A market sentiment report for each stock within the portfolio pulling together relevant news articles; A trade history report for each stock within the portfolio showing the predictions and positions taken over the last 20 days.

3.2.6.5 Portfolio optimisation The output from the previous stages is a justified recommendation to take a long or short position against each stock within a given list. These have been analysed on individually trained models. The next step in the process is to look at the impact of correlations between these stocks when assessed as an overall portfolio. To perform this optimisation I will use the mean-variance approach [48].

Portfolios of different stocks will have different levels of expected risk and return. In portfolio theory, portfolios that provide the largest possible expected return for a specific level of risk are called efficient portfolios. One assumption made in the construction of efficient portfolios is that investors are risk-averse – ie they will always choose the portfolio with the lowest risk when offered choices with the same expected return. Within the mean-variance approach we quantify the concept of risk using the standard deviation and the variance.

Variance captures the dispersion of a dataset relative to its mean. The Standard Deviation σ is widely used in Risk Management and is the square root of the Variance $V(X)$ as shown below, with x_i being a sample within the dataset X, μ being the mean across this dataset, and n the number of samples) . For most datasets, approx 95% of results will be within two standard deviations above and below the mean.

$$\sigma = \sqrt{V(X)} = \sqrt{\frac{\sum_i (x_i - \mu)^2}{n}}$$

To calculate the variance of a portfolio consisting of multiple assets we need to look not only at the variance of individual assets, but also how closely the returns of one asset track those of the others. The following formula captures the covariance across two assets, with $E(X)$ being the expected value across the dataset X.

$$Cov(X, Y) = E[(X - E(X))(Y - E(Y))]$$

The variance of the portfolio return is the sum of the squared weighted variances of the two assets plus two times the weighted covariance between the two assets. This equation can be generalized where there are more than two assets in the portfolio.

$$PortfolioVariance = w_1^2\mu_1^2 + w_2^2\mu_2^2 + 2w_1w_2Cov_{1,2}$$

Our goal in portfolio optimisation is then to combine assets into well diversified portfolios (ie not correlated in the same direction) to lower risk without affecting the return - and then search through these portfolios to find the one with the minimum risk for a target level of return, or if required the highest return for a given level of risk.

To allow the user flexibility in adding or removing stocks from the portfolio optimisation process, the details of the stocks to include will be stored within the Environment Configuration file - separate from the Model Configuration, although one would expect these portfolios to be largely consistent.

3.2.6.6 Challenger model To support model testing, the user will be able to run a challenger model alongside the "main" model. This challenger model will be a deci-

sion tree classifier. This type of model is explainable by design and so allows not just performance but feature importance to be assessed.

3.2.6.7 Alternative models Also to support model testing, the user will be able to run a number of alternative models to benchmark the performance of the "main" model. These alternative approaches will cover the majority of those found within my literature review, as per Section 2.2.11.

4 Implementation and Testing

This system implementation consists of the following files. Full source code is included within the Appendix A:

- .env : the environment configuration file.
- model_portfolio : the model configuration file.
- data_sourcing.py : the data sourcing script.
- main.py : the main process script.
- reporting.py : reporting functions used by the main batch process.

4.1 Environment configuration

The system implementation uses the following environment variables stored within an environment .env file. As outlined within the design, the granularity of batch related flags gives the user full flexibility when running.

Environment Variable	Purpose	Example
REPORT_DIR	Where to store the generated reports	Reports/
DATA_DIR	Where to store the generated data	Data/
MODEL_DIR	Where to store the trained models	Models/
INDEX_TICKERS	Index tickers used in model prediction	'["^VIX", "^GSPC"]'
HISTORICAL_DATA	Historical data to source	"10y"
CF_COLUMNS	Columns to flex when generating counterfactuals	["_AD_trend", "_ULT_BIN", "_RSI_BIN"]'
PORTFOLIO_STOCKS	Stock tickers for Portfolio opt.	["BAC", "AAPL", "MSFT"]'
COVARIANCE_HORIZON	Covariance horizon for Portfolio opt.	10y
PORTFOLIO_VALUE	Funds available for Portfolio opt.	1000000
TRAINING	Run model training	TRUE
DAILY	Run daily reporting	TRUE
SENTIMENT	Run sentiment reporting	TRUE
HISTORY	Run trade history reporting	TRUE
ANCHORS	Run Anchors xAI	TRUE
SHAP	Run SHAP xAI	TRUE
SIMILARITY	Run Similarity xAI	TRUE
DICE	Run Counterfactuals xAI	TRUE
PORTFOLIO	Run Portfolio opt.	TRUE
MODEL_PERF	Run model performance analysis	TRUE
CHALLENGER	Run challenger model analysis	TRUE
ALTERNATIVE	Run alternative model analysis	TRUE

Table 2: Environment Configuration Variables

The Python library "OS" is used to access environment variables stored as strings (Link). Example usage code snippet:

```
data_dir = os.getenv('DATA_DIR')
```

A second library "DOTENV" (Link) is used to read in environment variables stored as key-value pairs (eg the list of stock tickers). Example usage code snippet:

```
from dotenv import load_dotenv
load_dotenv()
import json
core_stocks = json.loads(os.environ['INDEX_TICKERS'])
```

4.2 Model configuration

The model configuration file contains one row for each stock to include within the stock movement prediction system. The columns are as follows:

- Stock Ticker : The stock ticker to use for data sourcing - eg BAC
- Hidden Layers : The hidden layer architecture to use when training this stock - eg [16, 16, 3]

4.3 Data sourcing script

The data sourcing process is centralised into one script in order to allow easy scheduling. It consists of three key functions:

4.3.1 Source Market data

End of day close prices are sourced into a Pandas dataframe using the Python library "YFINANCE". There are a number of free sources of EOD market data. I chose to use yFinance for this implementation as it is a mature and reliable platform. (Link). The yFinance API is capable of sourcing market data for multiple stock tickers at the same time. Example usage code snippet:

```
import yfinance as yf
api_object = yf.Tickers(core_stocks)
raw = api_object.history(period=10y)
market_data = raw['Close']
```

I also source EOD prices for two related indices:

- S&P 500 : I am focussing my implementation on US equities and so source the S&P 500 index price. The index includes 500 of the largest companies within the US, covering approximately 80 % of available market capitalization - source : (Link).

- VIX : The VIX (Volatility Index) measures volatility in the stock market. Generated by the Chicago Board of Options Exchange, it is based on the implied volatility of options taken on the S&P 500 Index.

The end of day close prices are sourced for both the stock tickers and index tickers. The period of history to retrieve is specified in the Environment Configuration file. The results are then saved down into the Data Directory.

4.3.2 Source Technical metrics

The Alpha Vantage API only allows one metric to be sourced for one ticker within a single call. It does however return all historic data available (back to 1999). My code therefore calls the API once for each metric for each stock. Example usage code snippet below. The results are saved down into the Data Directory.

```
ticker = 'BAC'
url = 'https://www.alphavantage.co/query?function=AD&symbol=' + ticker +
    '&interval=daily&datatype=csv&apikey=I2UR68ST032EG0J5'
AD = pd.read_csv(url)
```

The "technical_metrics_batch" function iterates through each stock in turn - calling the function "source_metrics" which in turn calls the API as per the above example - and then concatenates the information retrieved into a Pandas dataframe to form the final results. The results are then saved down into the Data Directory.

4.3.3 Generate trends and moving averages

The market data and technical metric datasets are joined together and then the "process_data" function is called which generates a number of trends and averages which are used within the model. The final results are saved down into the Data Directory.

4.3.3.1 Process return data The daily return on a stock is today's price divided by yesterday's price. It is common practice to use the log return in order to make additive over time. As my model does not use timeseries data, I am then processing the return data to generate a number of derived features relating to price movement.

- Return and Return direction : As well as saving down the log return, I add features to capture the return direction (+1 for positive or -1 for negative) and binary return direction (1 for positive and 0 for negative). The return direction is used for calculating the cumulative returns. The binary return is used by the model and explain functions.

- Shifted log returns : I create five new features capturing the daily log returns from one to five days previously. I then digitize by the first and second moments of the distribution to form the final measures actually used within the model. This approach to create feature data based on lagged returns and digitize by the first

and second moments of the distribution comes from Section 15 (Trading Strategies) of the book "Python for Finance" (Yves Hilpisch) [47].

- Simple moving averages : I finally create additional features for short term and long term simple moving averages. I use the same binning approach as for the Shifted log returns.

4.3.3.2 Process technical metrics Depending on the metric, I either calculate the trend across a set time period or look at the difference day on day and use binning to divide the values into ranges - as described above for the Shifted log returns. When complete, the final dataset is saved down into the Data Directory.

4.3.4 Final feature list

The final list of 21 features within the model is included below. The descriptions of the technical metrics (within quotes) are taken from FMLabs Technical Indicator Reference Link.

1. Lagged log returns 1 day : The Log Returns for the stock from t-1.
2. Lagged log returns 2 days : The Log Returns for the stock from t-2.
3. Lagged log returns 3 days : The Log Returns for the stock from t-3.
4. Lagged log returns 4 days : The Log Returns for the stock from t-4.
5. Lagged log returns 5 days : The Log Returns for the stock from t-5.
6. Stock short horizon moving average : 40 day moving average of the stock.
7. Stock long horizon moving average : 160 day moving average of the stock.
8. S&P short horizon moving average : 40 day moving average of the S&P.
9. S&P long horizon moving average : 160 day moving average of the S&P.
10. VIX short horizon moving average : 40 day moving average of the VIX.
11. VIX long horizon moving average : 160 day moving average of the VIX.
12. On balance volume (OBV) trend : "The On Balance Volume (OBV) is a cumulative total of the up and down volume. When the close is higher than the previous close, the volume is added to the running total, and when the close is lower than the previous close, the volume is subtracted from the running total". I calculate the trend in the On Balance Volume (OBV) values over the preceding 3 days.

13. Chaikin A/D line (AD) trend : "The Accumulation/Distribution Line is similar to the On Balance Volume (OBV), which sums the volume times +1/-1 based on whether the close is higher than the previous close. The Accumulation/Distribution indicator however multiplies the volume by the close location value (CLV). The CLV is based on the movement of the issue within a single bar and can be +1, -1 or zero". I calculate the trend in the Chaikin A/D line (AD) values over the preceding 3 days.

14. Ultimate oscillator (ULTOSC) : "The Ultimate Oscillator is the weighted sum of three oscillators of different time periods. The typical time periods are 7, 14 and 28. The values of the Ultimate Oscillator range from zero to 100. Values over 70 indicate overbought conditions, and values under 30 indicate oversold conditions". I digitise the Ultimate oscillator (ULTOSC) values into three bins (Between 0-40, Between 40-60, Between 60-100).

15. Relative strength index (RSI) : "The Relative Strength Index (RSI) calculates a ratio of the recent upward price movements to the absolute price movement. The RSI ranges from 0 to 100. The RSI is interpreted as an overbought/oversold indicator when the value is over 70/below 30". I digitise the Relative strength index (RSI) values into three bins (Between 0-40, Between 40-60, Between 60-100).

16. Williams %R (WILLR) : "The Williams %R values range from zero to 100, and are charted on an inverted scale, that is, with zero at the top and 100 at the bottom. Values below 20 indicate an overbought condition and a sell signal is generated when it crosses the 20 line. Values over 80 indicate an oversold condition and a buy signal is generated when it crosses the 80 line". I digitise the Williams %R (WILLR) values into four bins (Between -100 and -80, Between -80 and -20, Between -20 and 0, Zero).

17. Directional movement index (DX) : "The DX values range from 0 to 100, but rarely get above 60. To interpret the DX, consider a high number to be a strong trend, and a low number, a weak trend". I digitise the Directional movement index (DX) values into four bins (Between 0-20, Between 20-40, Between 40-60, Between 60 and 100).

18. Average directional movement index (ADX) : "The ADX is a Welles Wilder style moving average of the Directional Movement Index (DX). The values range from 0 to 100, but rarely get above 60". I digitise the Average directional movement index (ADX) values into four bins (Between 0-20, Between 20-40, Between 40-60, Between 60-100).

19. Money flow index (MFI) : "The Money Flow Index calculates the ratio of money flowing into and out of a security. Money Flow Index values range from 0 to 100. Values above 80/below 20 indicate market tops/bottoms". I digitise the Money flow index (MFI) values into four bins (Between 0-30, Between 30-70, Between 70-100, 100).

20. Parabolic SAR (SAR) Flag : "The Parabolic SAR calculates a trailing stop. The SAR assumes that you are always in the market, and calculates the Stop And Reverse point when you would close a long position and open a short position or vice versa. Looks at the difference between the Parabolic SAR (SAR) value and the EOD Close". I return -1 if the difference is less than 0, 0 if the difference is 0 and +1 if the difference is more than 0.

21. Aroon (AROON) UP / DOWN Difference : "The Aroon indicator attempts to show when a new trend is dawning. The indicator consists of two lines (Up and Down) that measure how long it has been since the highest high/lowest low has occurred within an n period range. When the Aroon Up is staying between 70 and 100 then it indicates an upward trend. When the Aroon Down is staying between 70 and 100 then it indicates an downward trend". I digitise the difference between the Aroon Up and the Aroon Down values into four bins (Between -100 and -40, Between -40 and 40, Between 40-100, 100).

4.4 Main batch script

The main batch script is designed to run after completion of the data sourcing batch, and begins by loading in the final prepared data from the specified data directory. This dataset is then split into a training and a testing dataset. This is a sequential split across the overall timeline with 80% allocated to training and 20% allocated to testing. The split needs to be sequential rather than random in order to allow effective backtesting analysis of model returns across the period (comparing predicted to realised results).

4.4.0.1 Load models The process then loads in the latest set of models from the model directory. The trained models are serialised using the Python library "PICKLE". This library provides the functionality to convert a Python object into a binary file which can be saved down. The binary file can then be re-loaded and "un-pickled" to recreate the original object.

4.4.0.2 Prediction Once the latest set of model objects have been re-created, the "run_prediction" function is executed. The prediction function takes in any model object. This allows the function to be re-used by the challenger model and alternative model configurations as described later. The prediction function runs across all dates within the dataset passed in. In this way, the prediction function can also be re-used by the model performance processes, also described later.

The last date within the testing dataset will be "today" - the prediction date. The model predicts the best position to be taken tomorrow based on the feature data seen for today - returned as a 1 or 0. This result is then mapped to be +1 for long and -1 for short. This mapped result is then used to calculate the return on the position. The return from day t is multiplied by the model position from day $t-1$. This reflects the return that would be seen if the model proposed strategy was followed.

4.4.0.3 Training If the Training flag within the environment config file is set to True, the model training process will run. A model is trained for each stock within the portfolio. All models within the system are built using the Python library "SCIKIT-LEARN". I chose this library as it is open source, well established, and one that we used extensively within the MSc Artificial Intelligence Programme. As per Section 2.2.11 the core model selected for this implementation is a Feed-Forward Neural Network. This is implemented in scikit-learn as the Multi Layer Perceptron (MLP) Classifier. Details on the final configuration used are included within Section 4.5.

The trained model is saved down using the pickle functionality as described above. A list of the columns used within the model training are also pickled and saved down. This list is used extensively within the xAI methods later.

4.4.0.4 The User Interface As mentioned within the design in Section 3, my goal is to keep the user interface simple and maintain the clarity of justification messaging - only presenting information if adds-value. Within my implementation I generate a number of PDF reports to capture system outputs, including the models and the xAI justification. One can see how, if needed, these could easily become views within a wider user desktop or dashboard.

All report generation functions are consolidated into the one Python file - "reporting.py". This allows me to maximise reuse and define concepts such as the style sheet in one place only.

Reports are generated as PDF files using the Python library "REPORTLAB". I selected this package from the many available due to it being simple to use and well integrated with both Matplotlib and Pandas. The package offers document templates which automatically format in line with standards such as A4. Report content is built up by appending "flowables" - such as paragraphs of text, images and tables - to a list variable. The document is then built using the defined template populated by the final content flow. Example usage code snippet below.

```
from reportlab.platypus import SimpleDocTemplate
pdf = SimpleDocTemplate(report_dir + "Sentiment_Report_" + ticker + "_" +
    news_date + ".pdf")
content_flow = []
paragraph_txt = Paragraph("Text", sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
image1 = Image('img1.png', width=300, height=150)
content_flow.append(image1)
table1 = []
table1.append(['Row1 Col1', 'Row1 Col1', 'Row1 Col3'])
table1.append(['Row2 Col1', 'Row2 Col1', 'Row2 Col3'])
table_final = Table(table1)
content_flow.append(table_final)
pdf.build(content_flow)
```

4.4.0.5 Generate xAI explain : Anchors If the Anchor explains flag within the environment config file is set to True, an Anchor decision rule explain will be generated for today's model prediction date. My implementation leverages the Python library "ALIBI" which includes a number of xAI approaches. (Link). Following experimentation, I have set the precision threshold at 99%. This means that predictions on instances where the anchor holds will be the same as the original prediction at least 99% of the time.

As we are working with tabular data, an AnchorTabular explainer is initialised and then trained with our target dataset. The trained explainer is then run against the input dataset for the prediction date, returning not just the decision rule but also statistics around precision and coverage. Example usage code snippet below.

```
from alibi.explainers import AnchorTabular
exp = AnchorTabular(predictor=predict_fn, feature_names=columns)
exp.fit(dataset, disc_perc=(25, 50, 75))
explanation = exp.explain(target, threshold=0.99)
```

This information is then sent to the report generation function. The Anchors / Decision Rules report begins by repeating the model prediction and then states the decision rule anchoring this prediction. The report then gives details on the precision and coverage around its calculation.

Figure 5: Example Decision Rules Report

Anchors Report for BAC on 2024-07-26

Recommendation : Go short

Decision rule anchoring the forecast

BAC_OBV_trend <= -1.00 AND BAC_AR_BIN > 1.00 AND BAC_WIL_BIN > 2.00

Confidence :

Precision 0.99, Coverage 0.06

4.4.0.6 Generate xAI explain : SHAP values If the SHAP explains flag within the environment config file is set to True, a SHAP feature importance explain will be generated for today's model prediction date. My implementation leverages the Python library "SHAP" (Link).

The library offers a number of different methods for SHAP generation. I have decided to use the Kernel SHAP method which uses weighted linear regression to compute the importance of each feature. There is another method - Deep SHAP - specifically designed to work with deep learning models, but this only works within the Keras or Tensorflow frameworks. As I was seeing good results using the Kernel approach, I decided it not worth the time porting my model across to a new framework.

The Kernel Explainer is initialised and trained on a cut-down version of the training dataset, generated by performing sampling without replacement on the input data. The SHAP library offers a function to perform this - set as to be 100 samples within my implementation.

Once complete, the explainer is then run on the dataset to be explained. Within my implementation I run the explain on the last 100 samples within the testing dataset - which includes today (our explain date) as the final row. Example usage code snippet below.

```
import shap
small_train = shap.sample(train_data, 100)
explainer = shap.KernelExplainer(model.predict, small_train[columns])
small_test = test_data.tail(100)
shap_values = explainer(small_test[columns])
```

The returned shap values object is then used to generate two separate feature importance reports. Firstly the SHAP feature attribute report which shows the results graphically just for the prediction date:

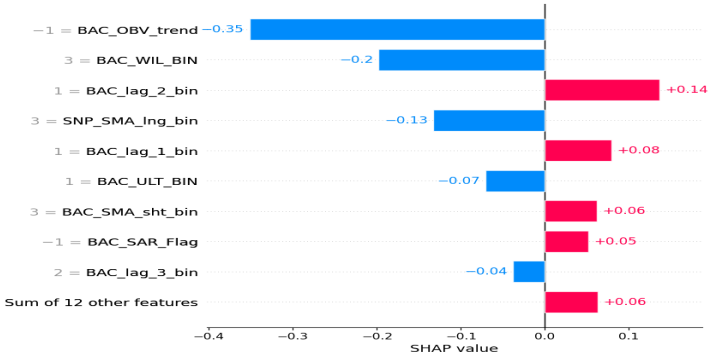
- Bar plot : The importance of each feature for this instance.
- Waterfall plot : For this instance, the bottom of a waterfall plot starts as the expected value of the model output, and then each row working upwards shows how the positive or negative contribution of each feature moves the value from the expected model output to the model output for this prediction.
- Force plot : The force plot provides an overview of how the individual features combine. Positive red features "push" the model output higher while negative blue features "push" the model output lower.

Figure 6: Example SHAP feature attribution report : Specific prediction date

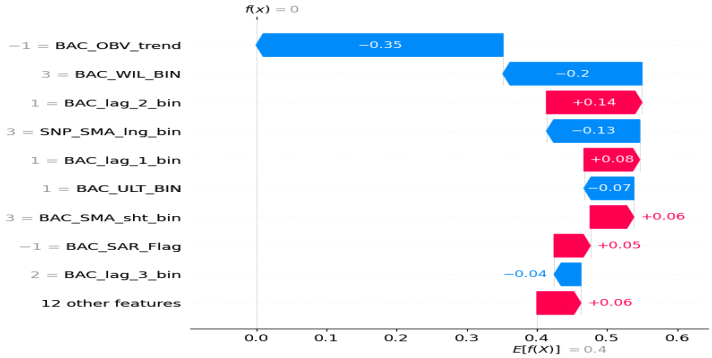
SHAP feature attribution

Drivers behind the forecast for BAC on 2024-07-26

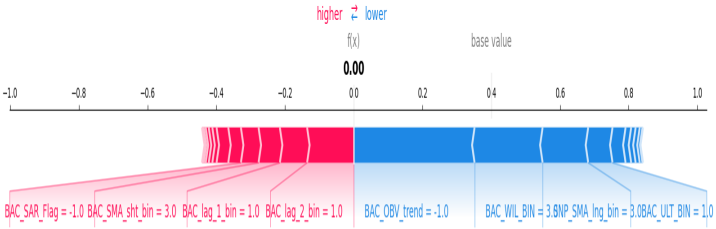
Bar plot



Waterfall plot



Force plot



A model level SHAP attribution report is also generated. Looking at this report allows the user to see if the prediction date attribution is in line with that for the model overall - or an outlier. This report is also used for model performance analysis.

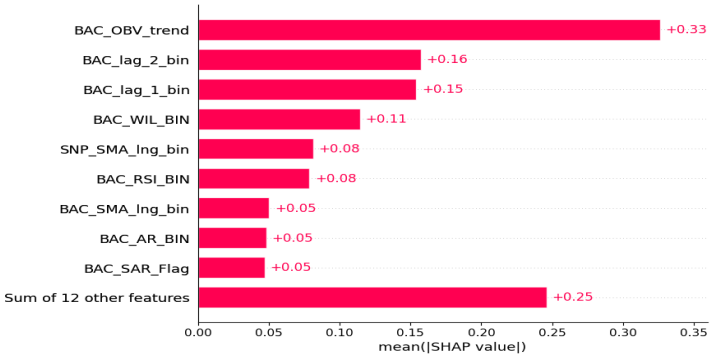
- Bar plot : The global importance of each feature is the mean absolute value for that feature across all given samples.
- Beeswarm plot : Shows how the top features in a dataset impact the models output. Each instance is represented by a single dot on each feature row. The x position is determined by the SHAP value. The colour is determined by the feature value from low to high.
- Summary plot : The "violin" summary plot represents the distribution and variability of SHAP values for each feature. Individual violin plots are stacked by importance of the particular feature on model output. The violins can therefore provide insights into the range, variability, skewness and symmetry of the SHAP value distribution for a specific feature.

Figure 7: Example SHAP feature attribution report : Model level

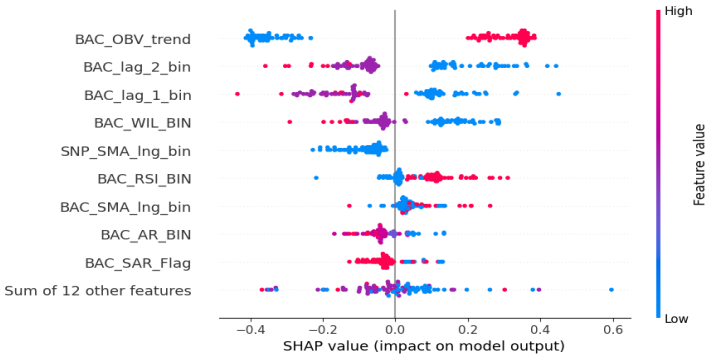
SHAP feature attribution

Model level results for BAC on 2024-07-26

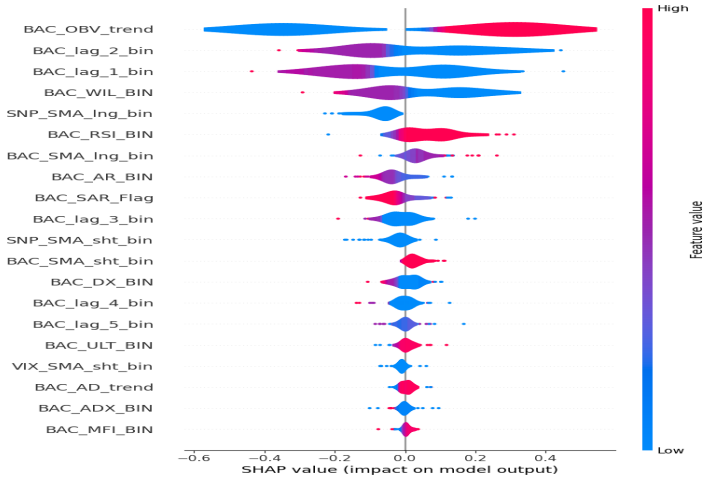
Bar plot



Beeswarm plot



Summary plot



4.4.0.7 Generate xAI explain : Diverse Counterfactuals If the DICE explains flag within the environment config file is set to True, the Diverse Counterfactual generation process will be run for today's model prediction date. My implementation leverages the Python library "DICE" (Link). The first step is to create a data object consisting of the training data itself along with definitions of the feature columns and target. You then create a model object including the trained model along with metadata as to its type. Finally, the explainer object is created using the data and model objects.

Once initialised, the explainer object is called with the number of counterfactuals to be generated (within my implementation set to be 5), the features that are allowed to be varied, and the desired class (eg the opposite to that generated by the model prediction). If it is not possible to generate a counterfactual of the opposite class, then an error is returned. My code traps this using an exception handling framework to ensure that the overall process can continue. Example usage code snippet below.

```
d = dice_ml.Data(dataframe=new_train, continuous_features=columns,
    outcome_name=ticker + '_Direction_bin')
m = dice_ml.Model(model=model, backend="sklearn", model_type="classifier")
exp = dice_ml.Dice(d, m, method="random")
e3 = exp.generate_counterfactuals(explain_target[columns], total_CFs=5,
    desired_class='opposite', features_to_vary=features)
```

The user can edit in the configuration file the specific columns which are flexed in the Counterfactual generation - and re-run the explain. This answers the question - what would need to be the change in feature X in order for the prediction to change ? The counterfactual generation process is therefore performed twice - once where all features are allowed to vary, and once where only the set of features specified by the user in the environment configuration file can vary. Both sets of results are then sent to the reporting function.

The Counterfactual Report includes five examples of perturbed input data for the prediction date which would lead to the opposite outcome. The counterfactual feature values are shown alongside the actual input data in order for the user to compare. Where the feature data is bucketed, the report also includes a helpful reminder.

Figure 8: Example Counterfactuals Report

Counterfactual examples

BAC on 2024-07-26

to get the opposite prediction from model result of -1

Restricted to varying : ['BAC_AD_trend', 'BAC_ULT_BIN', 'BAC_RSI_BIN', 'BAC_WIL_BIN', 'BAC_ADX_BIN', 'BAC_AR_BIN', 'BAC_MFI_BIN', 'BAC_DX_BIN', 'BAC_OBV_trend', 'BAC_SAR_Flag']

Feature attributes	Input data	CF 1	CF 2	CF 3	CF 4	CF 5
1d lagged returns (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-
2d lagged returns (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-
3d lagged returns (bin num) < mean - SD; < mean; > mean; > mean + SD	2.0	-	-	-	-	-
4d lagged returns (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-
5d lagged returns (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-
Chaikin AD trend UP(+1), DN(-1)	-1.0	-	0.6	-	-	-
Ultimate Oscillator (bin num) 0 - 40; 40 - 60; 60-100	1.0	3.0	-	-	3.0	-
Relative strength index (bin num) 0 - 40; 40 - 60; 60-100	2.0	-	-	-	-	3.0
Williams Value (bin num) -100 to -80; -80 to -20; -20 to 0; 0	3.0	-	0.0	-	2.0	-
Avg directional movement (bin num) 0 - 20; 20 - 40; 40 - 60; 60-100	2.0	-	-	-	-	-
Aroon value trend UP(+1), DN(-1)	2.0	-	-	-	-	-
Money flow index (bin num) 0 to 30; 30 to 70; 70 to 100; 100	2.0	-	-	-	-	-
Directional movement (bin num) 0 - 20; 20 - 40; 40 - 60; 60-100	1.0	-	-	-	-	-
On balance volume trend UP(+1), DN(-1)	-1.0	0.5	-	0.6	-	-0.6
Parabolic SAR flag + or - diff to EOD Close	-1.0	-	-	0.4	-	-
Short stock SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	3.0	-	-	-	-	-
Long stock SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	3.0	-	-	-	-	-
Short SnP SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	3.0	-	-	-	-	-
Long SnP SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	3.0	-	-	-	-	-
Short VIX SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-
Long VIX SMA trend (bin num) < mean - SD; < mean; > mean; > mean + SD	1.0	-	-	-	-	-

4.4.0.8 Generate xAI explain : Similarity If the similarity explains flag with the environment config file is set to True, the system determines the most similar date within the training set to the prediction date where the next day movement seen is in line with the model prediction.

For my implementation I calculate the most similar date using cosine similarity. This is a standard measure of similarity between two non-zero vectors used widely in machine learning, calculated as the L2-normalized dot product of two vectors. I compare the vector of input data on the prediction date against each date in the training set, and select the most similar.

$$cosinesimilarity(x, y) = \frac{x \cdot y}{|x||y|}$$

Full details surrounding both the explain data and most similar data are then passed to the reporting function. To give the user additional context, the Similarity Report begins with a number of charts (as below) along with stating the similarity match rate percentage to set the level of confidence:

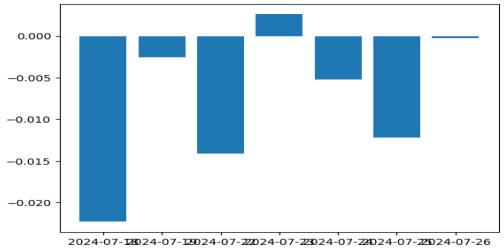
- The price trend leading up to the prediction date.
- The price trend leading up to the similarity date.
- The price trend following the similarity date.

The report then includes full details of the feature date from the similarity date - in the same format (actually using the same code) used for the Daily Report.

Figure 9: Example Similarity Report

Similarity Report for BAC on 2024-07-26

EOD return trend leading up to model date

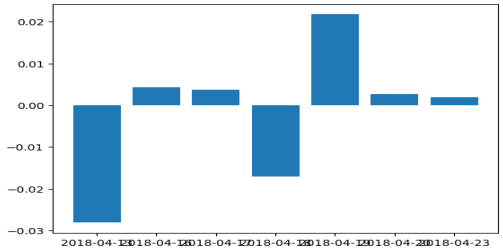


Recommendation : Go short

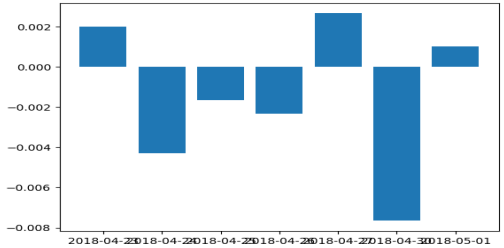
Most similar date in training set : 2018-04-23

Match % of 0.97

EOD return trend leading up to 2018-04-23



EOD return trend following 2018-04-23



Input data on 2018-04-23

COB Market Metrics

Direction/Trend is day over day

Feature	Value	Direction/Trend
EOD Close	26.12	1.0
Log Return	0.0	-1.0
S&P 500	2670.29	1.0
VIX	16.34	-1.0

COB calculated measures, bucketed wrt the overall mean and SD

< mean - SD	< mean	> mean	> mean + SD
	SMA short		
SMA long			
SnP SMA short			
SnP SMA long			
	VIX SMA short		
VIX SMA long			
	1d Lag Returns		
		2d Lag Returns	
3d Lag Returns			
	4d Lag Returns		
	5d Lag Returns		

Technical metrics, included in the model wrt trend

Chaikin A/D line + On balance volume : Trend over the last 3 days

Parabolic SAR flag : Difference between Parabolic SAR and the EOD close. 1 if positive, -1 if negative

Aroon Up/Down values : Difference day on day

Feature	Value	Direction/Trend/Difference
Chaikin A/D line (AD)	-10879272697.09	1.0
On balance volume (OBV)	11411199671.0	1.0
Parabolic SAR (SAR)	26.55	-1.0
Aroon (AROON) UP values	40.0	-10.0
Aroon (AROON) DOWN values	50.0	50.0

Technical metrics, included in the model wrt bucketed value

Ultimate Oscillator + Relative Strength Index : Values over 70 indicate overbought conditions, values under 30 indicate oversold conditions

0 - 40	40 - 60	60-100

Ult Osc
RSI

Directional Movement Index + Average Directional Movement Index : Typically 0-60 with a high number being a strong trend. Values over 70 indicate overbought conditions. Values under 30 indicate oversold conditions

0 - 20	20 - 40	40 - 60	60-100
	DX		
	ADX		

Williams %R values : Below 20 indicates overbought condition. Values over 80 indicate an oversold condition

-100 to -80	-80 to -20	-20 to 0	0
	WILLIAMS %R		

Money Flow Index : Values above 80 indicate market tops. Values below 20 indicate market bottoms

0 to 30	30 to 70	70 to 100	100
	MFI		

4.4.0.9 Run Challenger Model If the Challenger flag with the environment configuration file is set to True, the Challenger model process will be run. As mentioned within the design section 4.4.0.9 the challenger model is a decision tree classifier. As with the main model, this is built within the scikit-learn library. The decision tree classifier is trained and tested on the same training and testing datasets as used by the main model.

The performance of the model is scored across both training and testing datasets using the scikit-learn score function which returns the mean accuracy across all elements. The returns are then calculated both for adopting the model strategy, and for simply holding the stock. This is run across both training and testing datasets and is calculated as the exponential of the sum of the log returns across the period.

A very useful feature within scikit-learn decision tree implementation is the ability to visualise the generated decision tree. I use this within my implementation. The Challenger Model Report displays both model performance results and the decision tree visualisation.

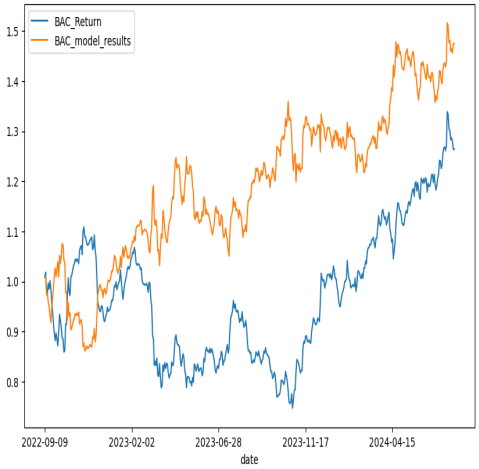
Figure 10: Example Challenger Model Report

Challenger model : Decision Tree Classifier

BAC on 2024-07-26

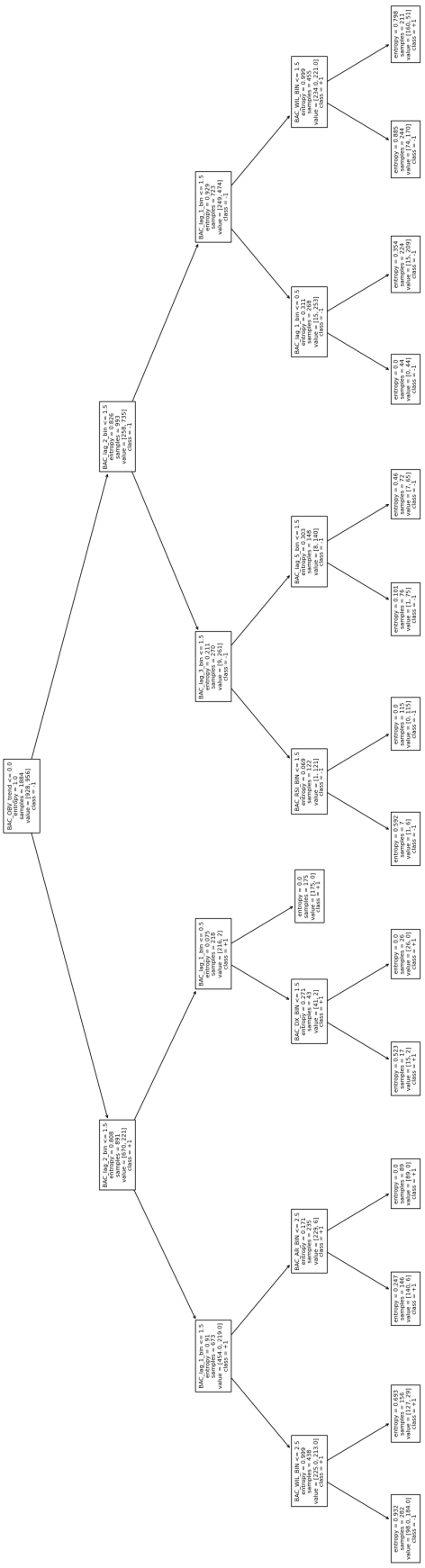
Training set model score = 0.85

Testing set model score = 0.82



Return seen for just holding stock = 1.26

Return seen using model strategy = 1.47


BAC_QNV_trend <= 0.0
entropy = 0.994
samples = 1884
value = [1020, 864]
class = -1
BAC_hig_1_bin <= 1.5
entropy = 0.868
samples = 1153
value = [870, 283]
class = -1
BAC_hig_1_bin <= 1.5
entropy = 0.826
samples = 731
value = [258, 473]
class = +1
BAC_hig_1_bin <= 1.5
entropy = 0.931
samples = 438
value = [454.0, 219.0]
class = -1
BAC_hig_1_bin <= 0.5
entropy = 0.276
samples = 715
value = [716, 2]
class = -1
BAC_hig_1_bin <= 1.5
entropy = 0.929
samples = 372
value = [249, 474]
class = +1
BAC_hig_1_bin <= 1.5
entropy = 0.211
samples = 720
value = [0, 241]
class = +1
BAC_WIL_BIN <= 2.5
entropy = 0.993
samples = 438
value = [214, 210]
class = -1
BAC_AR_BIN <= 2.5
entropy = 0.991
samples = 295
value = [212, 6]
class = -1
BAC_DX_BIN <= 1.5
entropy = 0.0
samples = 43
value = [43, 0]
class = -1
entropy = 0.0
samples = 37
value = [37, 0]
class = +1
BAC_hig_1_bin <= 0.5
entropy = 0.994
samples = 357
value = [357, 469]
class = +1
BAC_hig_1_bin <= 1.5
entropy = 0.211
samples = 220
value = [0, 241]
class = +1
BAC_WIL_BIN <= 4.5
entropy = 0.998
samples = 435
value = [424, 410]
class = -1
BAC_hig_1_bin <= 0.5
entropy = 0.994
samples = 368
value = [357, 469]
class = +1
BAC_RSI_BIN <= 1.5
entropy = 0.969
samples = 122
value = [121, 71]
class = -1
BAC_hig_1_bin <= 1.5
entropy = 0.0
samples = 98
value = [0, 240]
class = +1
entropy = 0.332
samples = 276
value = [38.0, 188.0]
class = +1
entropy = 0.593
samples = 162
value = [176.0, 29.0]
class = -1
entropy = 0.247
samples = 144
value = [140, 6]
class = -1
entropy = 0.0
samples = 151
value = [151, 0]
class = -1
entropy = 0.0
samples = 43
value = [43, 0]
class = -1
entropy = 0.0
samples = 37
value = [37, 0]
class = +1
entropy = 0.523
samples = 43
value = [43, 2]
class = -1
entropy = 0.0
samples = 43
value = [43, 0]
class = -1
entropy = 0.0
samples = 37
value = [37, 0]
class = +1
entropy = 0.0
samples = 98
value = [0, 240]
class = +1
entropy = 0.592
samples = 51
value = [51, 6]
class = -1
entropy = 0.0
samples = 71
value = [70, 0]
class = -1
entropy = 0.354
samples = 224
value = [15, 299]
class = +1
entropy = 0.0
samples = 44
value = [0, 44]
class = +1
entropy = 0.101
samples = 76
value = [1, 75]
class = +1
entropy = 0.48
samples = 46
value = [7, 65]
class = +1
entropy = 0.0
samples = 115
value = [0, 115]
class = +1
entropy = 0.0
samples = 71
value = [71, 0]
class = -1
entropy = 0.885
samples = 170
value = [74, 170]
class = +1
entropy = 0.706
samples = 91
value = [104, 51]
class = +1

4.4.0.10 Run Alternative Models If the Alternative flag with the environment config file is set to True, the Alternative model process will be run. As mentioned within the design in Section 4.4.0.10 a number of alternative models are trained and their performance assessed. For each stock within the portfolio, the Alternative model process trains each of the following classifiers using the "train_model" function previously defined. Code snippet below. This list and the comparison model configurations has been adapted from the scikit-learn documentation (Link)

```
classifiers = [
KNeighborsClassifier(3),
SVC(kernel="linear", C=0.025, random_state=42),
SVC(gamma=2, C=1, random_state=42),
GaussianProcessClassifier(1.0 * RBF(1.0), random_state=42),
RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1,
    random_state=42),
AdaBoostClassifier(algorithm="SAMME", random_state=42),
GaussianNB(),
GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
    max_depth=1, random_state=42)]
```

Each classifier is scored using the scikit-learn score function with the results consolidated into a single report as included below.

Figure 11: Example Alternative Models Report

Alternative model performance report

Testing dataset for BAC on 2024-07-26

Model : KNeighborsClassifier(n_neighbors=3)

Model score = 0.75

Model : SVC(C=0.025, kernel='linear', random_state=42)

Model score = 0.83

Model : SVC(C=1, gamma=2, random_state=42)

Model score = 0.5

Model : GaussianProcessClassifier(kernel=1**2 * RBF(length_scale=1), random_state=42)

Model score = 0.85

Model : RandomForestClassifier(max_depth=5, max_features=1, n_estimators=10, random_state=42)

Model score = 0.75

Model : AdaBoostClassifier(algorithm='SAMME', random_state=42)

Model score = 0.83

Model : GaussianNB()

Model score = 0.69

Model : GradientBoostingClassifier(learning_rate=1.0, max_depth=1, random_state=42)

Model score = 0.83

4.4.0.11 Run Portfolio Optimisation If the Portfolio optimisation flag within the environment config file is set to True, the portfolio optimisation process will be run. My implementation uses the Python library "PYPORTFOLIOOPT" (Link). The function begins by loading in the daily close date for stocks listed within the "Optimisation Portfolio" environment variable. As per the process design, portfolio optimisation may include zero or more of the stocks within the model, along with new tickers of interest to the user, and so the data used for model training and testing is not reused. The process then generates a number of portfolio metrics. Within my implementation I have followed the worked example within the PyPortfolioOpt documentation (link above) as this nicely illustrates the concepts desired - outlined below along with illustrative code snippets:

1. Calculate the expected returns and sample covariance.

```
mu = expected_returns.mean_historical_return(market_data)
S = risk_models.sample_cov(market_data)
```

2. Find the efficient frontier - the set of optimal portfolios which minimise the risk for a target return.

```
ef = EfficientFrontier(mu, S, weight_bounds=(-1, 1))
```

3. Find the portfolio from this set which gives the maximum return for the lowest risk - the maximal Sharpe ratio.

```
ef.max_sharpe()
```

4. Generate the discrete allocation using these weights given the portfolio fund size defined within the environment config file.

```
da = DiscreteAllocation(cleaned_weights, latest_prices,
    total_portfolio_value=int(portfolio_value))
```

Once created, the user may want to revise the portfolio based on additional knowledge - eg the Desk Head is aware of some additional information about a company which impacts its inclusion within the portfolio. In a human-human dialogue, this would be a case of "what-if" that the Desk Head would ask of the Junior Analyst. This is supported within my implementation through the configuration file.

- Use case 1 : The Desk Head wants to add a new stock to the portfolio and understand what this does to the weights. This use case is supported through the user updating the list of portfolio stocks within the configuration file and re-running the portfolio optimisation.

- Use case 2 : The Desk Head wants to remove a stock from the portfolio and understand what this does to the weights. Again, this use case is supported through the user updating the list of portfolio stocks within the configuration file and re-running the portfolio optimisation.
- Use case 3 : The Desk Head wants to change the co-variance horizon used within the portfolio optimisation process - eg to stop a specific period of un-representative data being picked up. This use case is supported through the user updating the Co-variance horizon within the configuration file and re-running the portfolio optimisation.

All results are then consolidated within a single report as included below.

Figure 12: Example Portfolio Optimisation Report

Portfolio report as of 2024-08-01

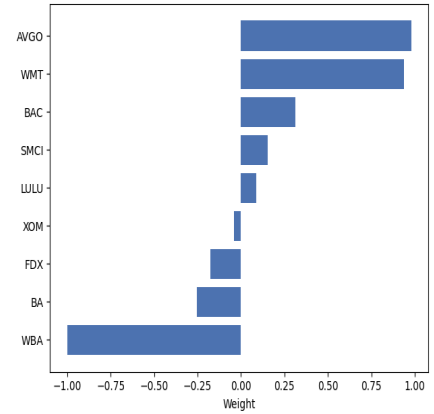
Portfolio : ['SMCI', 'AVGO', 'WMT', 'XOM', 'BAC', 'FDX', 'WBA', 'BA', 'LULU']

Model movement predictions - today and yesterday

Stock Ticker	Date	Movement prediction	Previous date	Previous Movement prediction
BAC	2024-08-01	-1	2024-07-31	-1
MSFT	2024-08-01	-1	2024-07-31	-1
TSLA	2024-08-01	-1	2024-07-31	1

Portfolio weights

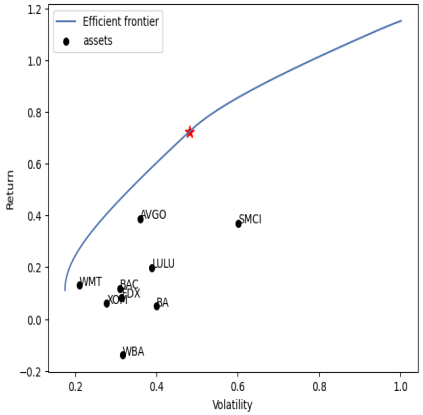
([('AVGO', 0.97912), ('BA', -0.25365), ('BAC', 0.31111), ('FDX', -0.17612), ('LULU', 0.08907), ('SMCI', 0.15177), ('WBA', -1.0), ('WMT', 0.93927), ('XOM', -0.04057)])



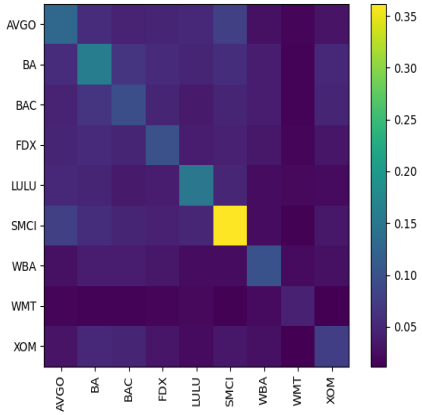
Portfolio details

Expected annual return (%): 0.72
Annual volatility (%) 0.48
Sharpe Ratio: 1.46
Remaining funds from portfolio of $1000000 : $446.12
Discrete allocation : {'AVGO': 2755, 'WMT': 5553, 'BAC': 3351, 'SMCI': 98, 'LULU': 153, 'WBA': -87108, 'BA': -1493, 'FDX': -602, 'XOM': -347}

Portfolio efficient frontier



Portfolio covariance



4.4.0.12 Run Model Performance If the Model Performance flag within the environment config file is set to True, the model performance process will run. The performance of the model is scored across both training and testing datasets using the scikit-learn score function which returns the mean accuracy across the dataset. The returns are then calculated both for adopting the model strategy, and for simply holding the stock - calculated as the exponential of the sum of the log returns across the period. The Model Performance Report consists of the following information:

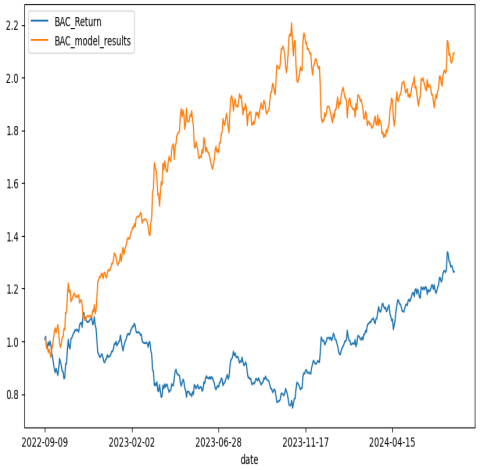
- Performance score on the testing set.
- Return seen across the testing horizon using the model.
- Return seen across the testing horizon just holding the stock.
- Cumulative return plots across the testing window of the above.
- Performance score on the training set.
- Cumulative return plots across the training window as described above for testing.
- The number of trades needed across the testing period to follow the model strategy.

Figure 13: Example Model Performance Report

Model performance report for BAC as of 2024-07-26

Training set model score = 0.88

Testing set model score = 0.8



Return seen for just holding stock = 1.26

Return seen using model strategy = 2.09

Across testing period of 472 days

Trades made using strategy = 188

4.4.0.13 Daily Report A Daily Report is generated for each stock included within the model for each day, and provides visibility into the prediction result (long or short) along with the feature data used to generate the prediction:

- EOD close. The EOD close value for the stock along with the day over day trend (-1 for down, +1 for up).
- Log return. The log return seen on the prediction date along with the day over day trend (as above).
- S&P 500 and VIX closes. The EOD close for the S&P index and the VIX index along with the day over day trend (as above).
- Lagged returns. For each of the 1-5 day lagged returns, where today's values fall with regards the overall distribution.

  Greater than the mean plus the standard deviation.

  Less than the mean minus the standard deviation.

  Between the mean and the mean plus the standard deviation.

  Less than the mean and the mean minus the standard deviation.
- Moving averages. For the long and short moving averages against the stock itself, the S&P index and the VIX index, where these averages fall with regards the overall distribution - categories as listed above.
- Technical metrics included in the model with regards trend. Details of the trend values used within the model for the following metrics : Chaikin A/D line (AD); On balance volume (OBV); Parabolic SAR (SAR); Aroon (AROON) UP values; Aroon (AROON) DOWN values.
- Technical metrics included in the model with regards bucketed values. Details of the digitized values used within the model for the following metrics : Ultimate Oscillator; RSI; ADX; DX; Williams; MFI.

Figure 14: Example Daily Report

Daily report for BAC on 2024-07-26

Recommendation : Go short

COB Market Metrics

Direction/Trend is day over day

Feature	Value	Direction/Trend
EOD Close	41.67	-1.0
Log Return	-0.0	1.0
S&P 500	5459.1	1.0
VIX	16.39	-1.0

COB calculated measures, bucketed wrt the overall mean and SD

< mean - SD	< mean	> mean	> mean + SD
		SMA short	
		SMA long	
		SnP SMA short	
		SnP SMA long	
VIX SMA short			
VIX SMA long			
1d Lag Returns			
2d Lag Returns			
	3d Lag Returns		
4d Lag Returns			
5d Lag Returns			

Technical metrics, included in the model wrt trend

Chaikin A/D line + On balance volume : Trend over the last 3 days

Parabolic SAR flag : Difference between Parabolic SAR and the EOD close. 1 if positive, -1 if negative

Aroon Up/Down values : Difference day on day

Feature	Value	Direction/Trend/Difference
Chaikin A/D line (AD)	-10175969134.24	-1.0
On balance volume (OBV)	11586398418.0	-1.0
Parabolic SAR (SAR)	44.41	-1.0
Aroon (AROON) UP values	30.0	-10.0
Aroon (AROON) DOWN values	0.0	-10.0

Technical metrics, included in the model wrt bucketed value

Ultimate Oscillator + Relative Strength Index : Values over 70 indicate overbought conditions, values under 30 indicate oversold conditions

0 - 40	40 - 60	60-100
Ult Osc		
	RSI	

Directional Movement Index + Average Directional Movement Index : Typically 0-60 with a high number being a strong trend. Values over 70 indicate overbought conditions. Values under 30 indicate oversold conditions

0 - 20	20 - 40	40 - 60	60-100
DX			
	ADX		

Williams %R values : Below 20 indicates overbought condition. Values over 80 indicate an oversold condition

-100 to -80	-80 to -20	-20 to 0	0
		WILLIAMS %R	

Money Flow Index : Values above 80 indicate market tops. Values below 20 indicate market bottoms

0 to 30	30 to 70	70 to 100	100
	MFI		

4.4.0.14 Market Sentiment Report As mentioned within the design in Section 3.2.5 part of my implementation includes the sourcing of sentiment data. There is an option of self calculation from raw text (using a library such as the Python NLTK library (Link)) or an option to use sentiment data already processed and made available through an API.

I decided to follow this second option and again leverage the Alpha Vantage solution (Link). The Alpha Vantage API tags each article with a market sentiment ranging from bullish (a belief that prices are going to rise) to bearish (a belief that prices are going to fall). The API is called for a specific stock ticker and a specific time period and returns up to 1000 results in a JSON object for further processing. Example usage code snippet below.

```
range_start_str = news_date + 'T0130'
range_end_str = news_date + 'T0130'
url = ('https://www.alphavantage.co/query?function=NEWS_SENTIMENT&tickers='
    + ticker + '&time_from=' + range_start_str + '&time_to=' + range_end_str
    + '&limit=1000&apikey=I2UR68ST032EG0J5')
r = requests.get(url).json()
```

My implementation pulls the sentiment out of each article and finds the most common outcome. It then uses one-hot encoding to convert the resulting category into separate features for each sentiment type. Two material issues were however seen during the implementation:

- Historical news data was only available for the last two years.
- Many of the tickers within my test set (for smaller companies) were coming back empty with no relevant news articles.

These issues make the data unsuitable for modelling. It can however be very useful for the user as part of their review stage - forming part of the external environment within which they review the model prediction justification. I therefore refocussed my implementation around the generation of a daily sentiment report for each ticker which would accompany the model predictions. Articles from multiple news providers are tagged with a relevance score (between 0 and 1) for all referenced stock tickers. In order to ensure that only relevant articles are included, the report includes only articles with relevance score more than 0.1.

Figure 15: Example Market Sentiment Report

Sentiment Report for BAC on 20240726

Headline : SoFi Technologies (SOFI) Rises 12% in a Month: Good Time to Buy?

Sentiment : Neutral : Relevance measure 0.1

https://www.zacks.com/stock/news/2310295/sofi-technologies-sofi-rises-12-in-a-month-good-time-to-buy

Headline : Charles Schwab Stock Plummeted 19% in the Days Following Its Earnings Announcement. Here's Why.

Sentiment : Neutral : Relevance measure 0.1

https://www.fool.com/investing/2024/07/26/charles-schwab-stock-plummeted-19-in-the-days-foll/

Headline : Warren Buffett Is Raking in $5.26 Billion in Annual Dividend Income From These 7 Stocks

Sentiment : Neutral : Relevance measure 0.13

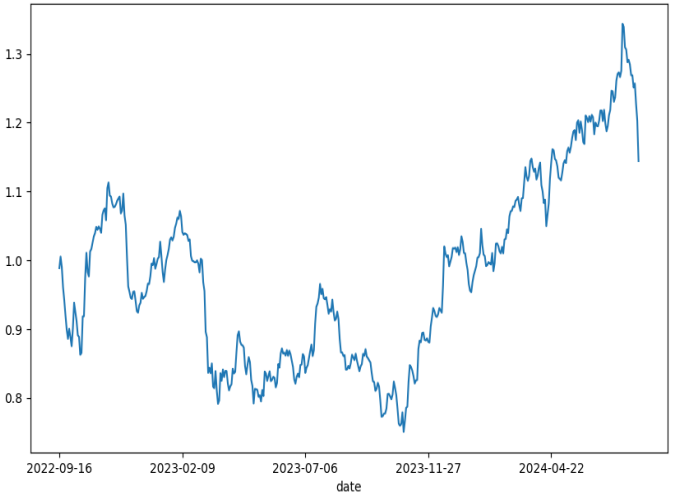
https://www.fool.com/investing/2024/07/26/warren-buffett-53-billion-dividend-income-7-stocks/

4.4.0.15 Trade History Report When deciding on a position to take going forward, it is useful to understand position history. The Trade History report is used to see the position history over the last 20 days. The position prediction generated by the model is shown along with the return. There is also a column on the report to show the actual position taken by the desk. This would be populated by the bank's trading system. Using this view, the Desk Head can see where he has decided to not follow the model proposal - and the resulting outcome.

Figure 16: Example Trade History Report

Trade history report for BAC as of 2024-08-02

Long term return trend



Detailed view over the last 20 days

EOD prices, daily returns and positions taken

Date	EOD Price	Daily Return	Model Position	Actual Position
2024-07-08	40.62	0.01	-1	
2024-07-09	41.42	0.02	1	
2024-07-10	41.74	0.01	1	
2024-07-11	41.81	0.0	1	
2024-07-12	41.59	-0.01	1	
2024-07-15	41.89	0.01	1	
2024-07-16	44.13	0.05	1	
2024-07-17	43.98	-0.0	1	
2024-07-18	43.01	-0.02	-1	
2024-07-19	42.9	-0.0	1	
2024-07-22	42.3	-0.01	-1	
2024-07-23	42.41	0.0	-1	
2024-07-24	42.19	-0.01	-1	
2024-07-25	41.68	-0.01	-1	
2024-07-26	41.67	-0.0	-1	
2024-07-29	41.09	-0.01	-1	
2024-07-30	41.28	0.0	-1	
2024-07-31	40.31	-0.02	-1	
2024-08-01	39.5	-0.02	-1	
2024-08-02	37.58	-0.05	-1	

4.5 Testing

4.5.1 Training and Testing datasets

For purposes of model training and testing, I sourced 10 years of historical data. Generation of the longer term moving averages trims off approx 3 months at the far end and so results in a window to use for both training and testing spanning March 2015 to July 2024. I have configured the training / testing split to be 80:20 resulting in the following windows:

- Training window : 10 Mar 2015 to 31 Aug 2022.
- Testing window : 01 Sep 2022 to 19 Jul 2024.

The diagram below shows the price trend of the S&P 500 over both periods. The training data is largely bullish, but does include material drops in 2020 relating to COVID and 2022 relating to the Russian invasion of Ukraine.

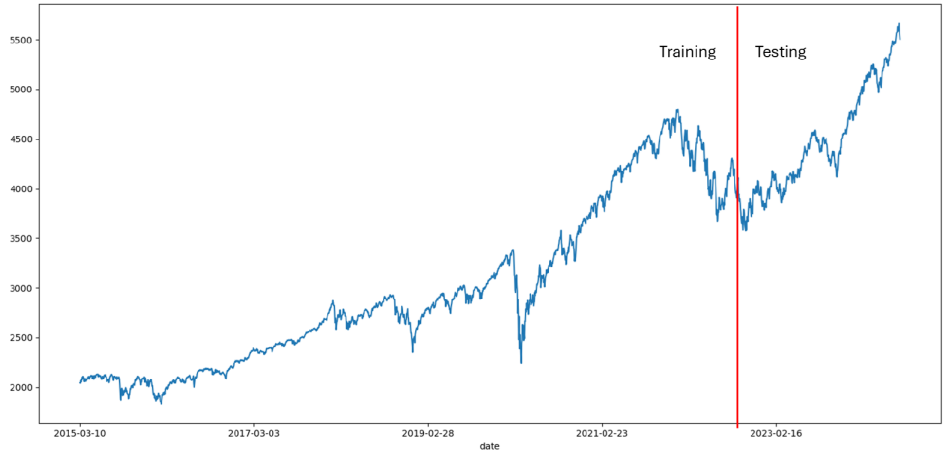


Figure 17: S&P 500 over the training and testing windows

4.5.2 Unit and Integration Testing

All system components as described above have been individually tested using a randomly selected portfolio of 20 stocks. All issues found were fixed before proceeding on to system testing.

4.5.3 System Testing

The overall system was tested by running all components for a different portfolio of 20 stocks each day over a five day period from 8th to the 12th of July. All issues found were fixed before proceeding on to result analysis and evaluation.

5 Results, Analysis and Evaluation

I evaluated my implementation using the following criteria:

1. Which hidden layer architecture leads to the best model performance ?
2. What is the impact on performance from the different types of feature data ?
3. How well does the implementation scale with regards the number of stocks ?
4. How well do the xAI methods support a user understanding and actioning these predictions - specifically, are justifications clear and consistent over time ?
5. How well do the xAI methods support internal model validation ?

5.1 Neural network architecture

5.1.1 Portfolio Selection

To assess model prediction accuracy under different architectures, I selected nine stocks covering the year to date top gainers, top losers and middle performers across the S&P500 as of the start of July 2024. I chose US stocks and include results within this section stated in USD as I have personally spent the last 12 years working in Bank of America and so find this domain more natural. The system will of course also work for UK stocks and one needs only to add a ".L" to the end of the ticker when calling the yFinance API as part of the data sourcing step.

Top Gainers		**Middle Performers**		**Top Losers**	
Ticker	Company	Ticker	Company	Ticker	Company
SMCI	Super Micro Computer Inc	XOM	Exxon Mobil Corp	WBA	Walgreens Boots Alliance Inc
AVGO	Broadcom Inc	BAC	Bank of America	LULU	Lululemon Athletica Inc
WMT	Walmart	FDX	Fedx	BA	Boeing

Table 3: Evaluation Portfolios

5.1.2 Results

In my analysis, the input and output layers are constant - with 21 neurons in the input layer reflecting the full range of features sourced and calculated, and an output layer of 1 neuron representing the classification of long (1) or short(0). For different configurations, I compare the returns seen using the model strategy to that of just holding the stock over the testing window. I also calculate the impact of the number of trades made on the position valuation - ie changing from long to short or vice-versa. To do this I have assumed a trade fee of $4.95 - the standard commission charged by e*trade for customers who perform at least 30 stock trades a quarter (Link). The following tables contain results as of 19 July 2024 across all nine stocks.

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	81%	61%	12.24	$ 12,240.00	6.01	68	$ 6,010.00	$ 5,673.40
AVGO	Broadcom Inc	$ 1,000.00	87%	79%	3.15	$ 3,150.00	0.9	211	$ 900.00	$ -144.45
WMT	Walmart	$ 1,000.00	87%	80%	1.64	$ 1,640.00	1.38	170	$ 1,380.00	$ 538.50
XOM	Exxon Mobil Corp	$ 1,000.00	87%	68%	1.29	$ 1,290.00	1.37	115	$ 1,370.00	$ 800.75
BAC	Bank of America	$ 1,000.00	88%	83%	1.35	$ 1,350.00	1.9	201	$ 1,900.00	$ 905.05
FDX	Fedx	$ 1,000.00	86%	85%	1.52	$ 1,520.00	1.09	186	$ 1,090.00	$ 169.30
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	84%	80%	0.35	$ 350.00	0.81	177	$ 810.00	$ -66.15
LULU	Lululemon Athletica Inc	$ 1,000.00	84%	80%	0.93	$ 930.00	1.04	161	$ 1,040.00	$ 243.05
BA	Boeing	$ 1,000.00	86%	81%	1.12	$ 1,120.00	1.03	186	$ 1,030.00	$ 109.30
		$ 9,000.00				$ 23,590.00			$ 15,530.00	$ 8,228.75

Table 4: One hidden layer of 16 neurons

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	84%	60%	12.24	$ 12,240.00	16.37	63	$ 16,370.00	$ 16,058.15
AVGO	Broadcom Inc	$ 1,000.00	88%	75%	3.15	$ 3,150.00	0.87	189	$ 870.00	$ -65.55
WMT	Walmart	$ 1,000.00	86%	79%	1.64	$ 1,640.00	1.08	176	$ 1,080.00	$ 208.80
XOM	Exxon Mobil Corp	$ 1,000.00	89%	75%	1.29	$ 1,290.00	1.19	143	$ 1,190.00	$ 482.15
BAC	Bank of America	$ 1,000.00	88%	78%	1.35	$ 1,350.00	2.35	187	$ 2,350.00	$ 1,424.35
FDX	Fedx	$ 1,000.00	87%	80%	1.52	$ 1,520.00	1.93	181	$ 1,930.00	$ 1,034.05
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	86%	80%	0.35	$ 350.00	2.02	169	$ 2,020.00	$ 1,183.45
LULU	Lululemon Athletica Inc	$ 1,000.00	87%	75%	0.93	$ 930.00	1.09	164	$ 1,090.00	$ 278.20
BA	Boeing	$ 1,000.00	50%	47%	1.12	$ 1,120.00	0.79	9	$ 790.00	$ 745.45
		$ 9,000.00				$ 23,590.00			$ 27,690.00	$ 21,349.05

Table 5: Two hidden layers of [16, 3] neurons

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	81%	61%	12.24	$ 12,240.00	8.28	64	$ 8,280.00	$ 7,963.20
AVGO	Broadcom Inc	$ 1,000.00	53%	53%	3.15	$ 3,150.00	3.19	1	$ 3,190.00	$ 3,185.05
WMT	Walmart	$ 1,000.00	87%	78%	1.64	$ 1,640.00	1.3	154	$ 1,300.00	$ 537.70
XOM	Exxon Mobil Corp	$ 1,000.00	51%	52%	1.29	$ 1,290.00	1.31	1	$ 1,310.00	$ 1,305.05
BAC	Bank of America	$ 1,000.00	89%	83%	1.35	$ 1,350.00	3.09	167	$ 3,090.00	$ 2,263.35
FDX	Fedx	$ 1,000.00	87%	83%	1.52	$ 1,520.00	1.18	179	$ 1,180.00	$ 293.95
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	85%	81%	0.35	$ 350.00	1.65	179	$ 1,650.00	$ 763.95
LULU	Lululemon Athletica Inc	$ 1,000.00	86%	82%	0.93	$ 930.00	1.41	165	$ 1,410.00	$ 593.25
BA	Boeing	$ 1,000.00	84%	80%	1.12	$ 1,120.00	0.91	183	$ 910.00	$ 4.15
		$ 9,000.00				$ 23,590.00			$ 22,320.00	$ 16,909.65

Table 6: Three hidden layers of [16, 16, 3] neurons

Ticker	Stock	Position at start	Return with model strategy	Trades made	Position at end	Position minus fees	**Hidden layer arrangement**
SMCI	Super Micro Computer Inc	$ 1,000.00	16.37	63	$ 16,370.00	$ 16,058.15	**[16, 3]**
AVGO	Broadcom Inc	$ 1,000.00	3.19	1	$ 3,190.00	$ 3,185.05	**[16, 16, 3]**
WMT	Walmart	$ 1,000.00	1.38	170	$ 1,380.00	$ 538.50	**[16]**
XOM	Exxon Mobil Corp	$ 1,000.00	1.37	115	$ 1,370.00	$ 800.75	**[16]**
BAC	Bank of America	$ 1,000.00	3.09	167	$ 3,090.00	$ 2,263.35	**[16, 16, 3]**
FDX	Fedx	$ 1,000.00	1.93	181	$ 1,930.00	$ 1,034.05	**[16, 3]**
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	2.02	169	$ 2,020.00	$ 1,183.45	**[16, 3]**
LULU	Lululemon Athletica Inc	$ 1,000.00	1.41	165	$ 1,410.00	$ 593.25	**[16, 16, 3]**
BA	Boeing	$ 1,000.00	1.03	186	$ 1,030.00	$ 109.30	**[16]**
		$ 9,000.00			$ 31,790.00	$ 25,765.85	

Table 7: Optimal hidden layer configuration by stock

5.1.3 Analysis and Evaluation

The choice of architecture has a big impact on performance. Finding the right layout is largely trial and error, but there are many helpful rules to guide assessment. For my evaluation with one hidden layer, I used as a guide setting the number of neurons as 2/3 of the sum of the neurons in the input and output layers - experimenting with settings between 13 and 17, with 16 giving the best performance. With one layer of [16], performance scores are good - averaging 86% across the training window and 77% across the testing window. The returns across the Middle and Low performers is above that seen in just holding the stock, but the returns on High performers are much lower. Starting from an even allocation across all stocks of $9m, just holding the stocks over the training period returned $23m but following the model proposals returned only $15m, as shown in Table 4.

When adding the second hidden layer, my initial assessment repeated the same architecture as the first - namely [16, 16]. This however led to a reduction in performance with the model over-fitting the training data - returning scores of 99% across many stocks. I therefore experimented with lower configurations, finding the best performance with a configuration of [16, 3] as shown in Table 5. Returns on some stocks are much higher - eg SMCI, but others drop eg Broadcom. Overall though, the aggregate position at the end of the training period is now materially **higher** than just holding the stock at $27m.

When adding a third hidden layer, I focussed on finding an architecture which best fitted stocks which until now showed only poor performance - eg Broadcom. After some experimentation, a configuration of [16, 16, 3] proved most useful - for example, increasing the return seen for Broadcom from 0.87 to 3.19. Other stocks however showed a reduced performance - most notably SMCI whose return dropped from 16.37 to 8.28, as shown in Table 6.

Overall then, results clearly show the advantages within my design - ie using an individual model for each stock with an individually optimised architecture. As shown in Table 7 this approach delivers a positive return on **all** stocks (greater than 1) and an overall return of $31m as opposed to $23m under a simple static position approach.

When looking at the impact of trade fees, across the 472 trading days within the testing window, the average number of trades to follow the model predictions for a single

stock was 135. For four out of the 9 stocks assessed with low returns (Walmart, Exxon, Lululemon and Boeing) these fees drive an overall negative return with the position value after fees being less that the position value at the start - as shown in Table 8. Clearly, as starting position sizes rise, the impact of these fees becomes less and less material. With a starting position of $100k in each stock, all position valuations minus fees are $100k+ at the end of the testing period and the overall valuation has increased from $900k to $3.1m.

5.2 Model feature analysis

I next look to assess the importance of the different feature categories on performance - training and running the model with a static architecture of two hidden layers [16, 3] with just returns and moving averages, then with just the technical metrics and indicators, and finally using all features. For each configuration, I compare the return seen using the model strategy to that seen just holding the stock over the window - as before. For this analysis I used the same portfolios as defined within Section 5.1.

5.2.1 Results

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	55%	49%	12.24	$ 12,240.00	4.63	225	$ 4,630.00	$ 3,516.25
AVGO	Broadcom Inc	$ 1,000.00	58%	50%	3.15	$ 3,150.00	1.34	216	$ 1,340.00	$ 270.80
WMT	Walmart	$ 1,000.00	54%	47%	1.64	$ 1,640.00	0.89	107	$ 890.00	$ 360.35
XOM	Exxon Mobil Corp	$ 1,000.00	57%	53%	1.29	$ 1,290.00	0.84	144	$ 840.00	$ 127.20
BAC	Bank of America	$ 1,000.00	59%	53%	1.35	$ 1,350.00	0.93	242	$ 930.00	$ -267.90
FDX	Fedx	$ 1,000.00	57%	49%	1.52	$ 1,520.00	0.82	185	$ 820.00	$ -95.75
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	54%	57%	0.35	$ 350.00	1.66	160	$ 1,660.00	$ 868.00
LULU	Lululemon Athletica Inc	$ 1,000.00	57%	51%	0.93	$ 930.00	0.88	144	$ 880.00	$ 167.20
BA	Boeing	$ 1,000.00	58%	52%	1.12	$ 1,120.00	0.87	199	$ 870.00	$ -115.05
		$ 9,000.00				$ 23,590.00			$ 12,860.00	$ 4,831.10

Table 8: Feature testing - Returns and moving averages

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	75%	72%	12.24	$ 12,240.00	2.12	143	$ 2,120.00	$ 1,412.15
AVGO	Broadcom Inc	$ 1,000.00	76%	75%	3.15	$ 3,150.00	2.39	121	$ 2,390.00	$ 1,791.05
WMT	Walmart	$ 1,000.00	77%	76%	1.64	$ 1,640.00	0.85	144	$ 850.00	$ 137.20
XOM	Exxon Mobil Corp	$ 1,000.00	78%	74%	1.29	$ 1,290.00	0.83	153	$ 830.00	$ 72.65
BAC	Bank of America	$ 1,000.00	78%	76%	1.35	$ 1,350.00	1.11	136	$ 1,110.00	$ 436.80
FDX	Fedx	$ 1,000.00	76%	77%	1.52	$ 1,520.00	1.54	137	$ 1,540.00	$ 861.85
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	76%	74%	0.35	$ 350.00	1.24	133	$ 1,240.00	$ 581.65
LULU	Lululemon Athletica Inc	$ 1,000.00	76%	70%	0.93	$ 930.00	1.02	112	$ 1,020.00	$ 465.60
BA	Boeing	$ 1,000.00	77%	75%	1.12	$ 1,120.00	0.99	141	$ 990.00	$ 292.05
		$ 9,000.00				$ 23,590.00			$ 12,090.00	$ 6,051.00

Table 9: Feature testing - Technicals and indicators

Ticker	Stock	Position at start	Training set score	Testing set score	Return holding stock	Position at end	Return with model strategy	Trades made	Position at end	Position minus fees
SMCI	Super Micro Computer Inc	$ 1,000.00	84%	60%	12.24	$ 12,240.00	16.37	63	$ 16,370.00	$ 16,058.15
AVGO	Broadcom Inc	$ 1,000.00	53%	53%	3.15	$ 3,150.00	3.19	1	$ 3,190.00	$ 3,185.05
WMT	Walmart	$ 1,000.00	87%	78%	1.64	$ 1,640.00	1.3	154	$ 1,300.00	$ 537.70
XOM	Exxon Mobil Corp	$ 1,000.00	51%	52%	1.29	$ 1,290.00	1.31	1	$ 1,310.00	$ 1,305.05
BAC	Bank of America	$ 1,000.00	89%	83%	1.35	$ 1,350.00	3.09	167	$ 3,090.00	$ 2,263.35
FDX	Fedx	$ 1,000.00	87%	80%	1.52	$ 1,520.00	1.93	181	$ 1,930.00	$ 1,034.05
WBA	Walgreens Boots Alliance Inc	$ 1,000.00	86%	80%	0.35	$ 350.00	2.02	169	$ 2,020.00	$ 1,183.45
LULU	Lululemon Athletica Inc	$ 1,000.00	87%	75%	0.93	$ 930.00	1.09	164	$ 1,090.00	$ 278.20
BA	Boeing	$ 1,000.00	50%	47%	1.12	$ 1,120.00	0.79	9	$ 790.00	$ 745.45
		$ 9,000.00				$ 23,590.00			$ 31,090.00	$ 26,590.45

Table 10: Feature testing - All features

5.2.2 Analysis and Evaluation

Training the models with a feature set containing just the lagged returns and moving averages results in poor performance - with scores across both training and testing windows just above - or in some cases below 50% - what you would get from tossing a coin - as shown in Table 8.

Results seen just using the technical metrics are better - averaging 75% - as shown in Table 9. This intuitively makes sense. These metrics are used by real traders in making portfolio investment decisions and so capture useful patterns in their design.

Combining both together proves most beneficial however with performance scores increasing to 80%+ across the majority of the portfolios - as shown in Table 10.

5.3 Scalability

I then looked to assess how well the system performs as the number of stocks increases - ie does this work as a system expected to run every day across the full S&P 500 ? I looked first at data sourcing, running the process with numbers of stock tickers from 3 up to 30. Results are shown in Table 11. I then assessed model training times, again running the process with numbers of stock tickers from 3 up to 30, with results shown in Table 12.

Number of stocks	3	6	10	14	20	30
Data sourcing time (seconds)	5.47	24.15	44.62	73.18	101.3	162

Table 11: Scalability testing - data sourcing

Number of stocks	3	8	13	18	23	30
Model training time (seconds)	2.13	5.76	8.99	11.76	15.59	19.43

Table 12: Scalability testing - model training

5.3.1 Analysis and Evaluation

It takes roughly 5 seconds to source and process the data for each stock - leading to approximately 42 minutes to source and process all 500 stocks within the S&P 500. This is more than acceptable for an overnight batch run. These results were generated on a single (reasonably high spec) Windows PC and so significant improvement would be expected when running in a distributed environment such as Microsoft Azure for AI.

Model training times are very quick and performance is more than acceptable to support daily execution at full volume. One would however expect a system such as this to be retrained only once a week.

5.4 End to End Process : xAI supported decisioning

In order to assess the effectiveness of the end-to-end process, I ran the full process every day from 19 July to 25th July for a subset of five stocks - taking the role of the Desk Head. Reminder below of the process flow as defined in Section 3

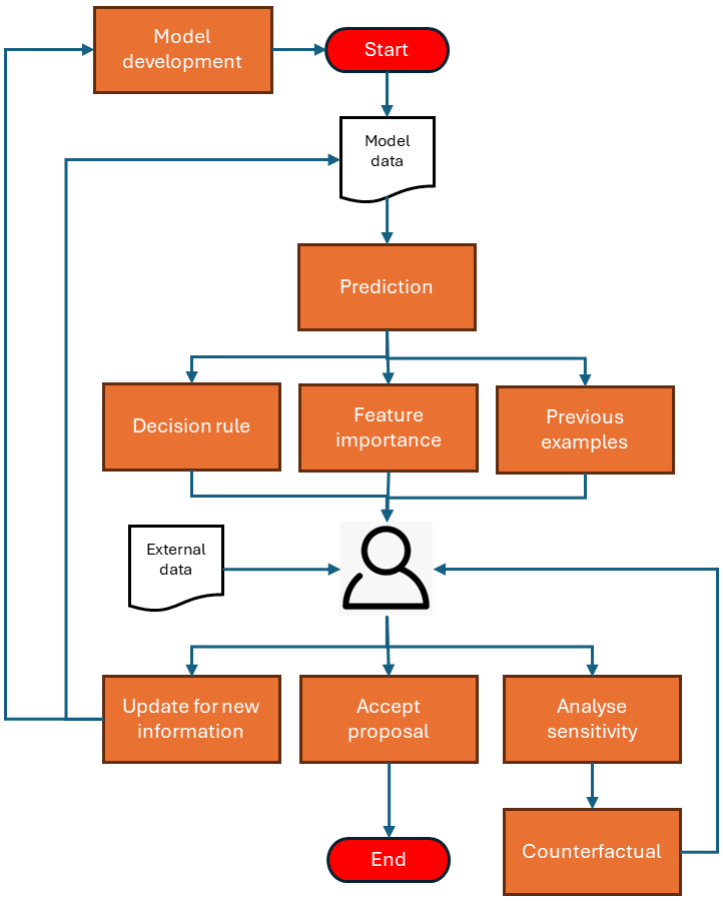


Figure 18: Integrated User-AI decisioning process

5.4.1 Super Micro Computer Inc (SMCI)

- Super Micro Computer Inc is one of the top gainers on the S&P 500 across 2024.
- The model consistently prediction LONG each day across the period.
- The decision rule logic was clear and did not change across the period. The prediction was driven by the long horizon moving average of the stock return.
- The SHAP report confirmed this - highlighting the long horizon moving average as a major +ve contributor along with the shorter term moving average and 2d return lag. Negative drivers came from three technical metrics - WIL, OBV and RSI - and were again consistent across the period.
- The Similarity Report varied with dates across Jun-Oct 2018 but all had a reasonably low match rate at 88-90%. This is not overly surprising - as mentioned, this stock has massively outperformed this year.
- So as a user, I am getting a clear and confident prediction and justification.
- Turning to additional sources of data and the Sentiment Report, this was not clear with some neutral, some somewhat-bearish, and some somewhat-bullish. A number of options traders were highlighted as changing their positions from bullish to bearish. Again, not a massive surprise - given how long this stock has been overperforming, it is overdue a turn.
- Looking at the Counterfactuals, this again supported the messaging from the other methods - namely that the only way that the prediction would turn to SHORT would be for the short and long term moving averages to flip from one side to the other of the distribution.
- My decision was therefore to follow the model proposals and stay LONG.
- This unfortunately proved costly as the returns were negative each day.
- The process however worked well with the justification behind the proposed position being clear and consistent.

5.4.2 FedEx (FDX)

- Fedex is one of the middle performers on the S&P 500 across 2024.
- The model prediction alternated between SHORT and LONG across the period.
- The decision rule logic was clear and did not change across the period. The prediction was driven by five technical indicators.
- The SHAP report was very useful - highlighting which indicator was providing positive, and which negative directional impact.

- The overriding feature was the OBV - whose value ultimately led to the prediction direction.
- The Similarity Report varied with dates across Jan-Jun 2018 but all had a reasonably high match rate of 94%+ giving me confidence as a user that the model had seen a pattern such as this before.
- So as a user, I am getting a clear and confident prediction and justification.
- Turning to additional sources of data and the Sentiment Report, this ranged from neutral to somewhat bullish, but a number of articles focussed on option traders going bullish on Fedex.
- Based on this information, my decision was therefore to disagree with the model proposals and go LONG.
- This proved the correct call towards the end of the testing period with returns moving from negative to positive.

5.4.3 Walgreens Boots Alliance (WBA)

- Walgreens Boots Alliance Inc is one of the biggest losers on the S&P 500 across 2024.
- The model consistently prediction SHORT each day across the period.
- The decision rule logic fluctuated across the testing period, highlighting different combinations of technical metrics and lagged returns.
- The SHAP report however again proved useful highlight the OBV metric as the overriding driver.
- The Similarity Report varied with dates across Apr-Jun 2018 but all had a reasonably solid match rate of 90%+ giving me confidence as a user that the model had seen a pattern such as this before.
- So as a user, I am getting a clear and confident prediction and justification.
- Turning to additional sources of data and the Sentiment Report, all of the news articles referencing WBA were focussed on a class action suit being brought against the company relating to the company making misleading statements to the market.
- Based in this, my decision was therefore to follow the model proposals and stay SHORT.
- Results were mixed with some days showing a negative and some a positive return. The process however worked well with the justification behind the proposed position being clear and consistent.

5.4.4 Lululemon Athletica (LULU)

- Lululemon Athletica Inc is one of the biggest losers on the S&P 500 across 2024.
- The model prediction SHORT on all but one day across the period.
- The decision rule logic fluctuated across the testing period, highlighting different combinations of technical metrics and lagged returns.
- The SHAP report however again proved useful highlight the OBV metric as the overriding driver.
- The Similarity Report varied with dates across Apr-Oct 2018 but all had a reasonably solid match rate of 94%+ giving me confidence as a user that the model had seen a pattern such as this before.
- So as a user, I am getting a clear and confident prediction and justification.
- Turning to additional sources of data and the Sentiment Report, the week started with sentiment being bullish or somewhat bullish on future potential. Mid week an earnings report came in lower than expectations and the week ended with the bullish sentiment over.
- Based on the clarity of the explain, my decision was to follow the model proposals and stay SHORT.
- This position proved correct as returns were negative on all but one day across the testing period.

5.4.5 Boeing (BA)

- Boeing is one of the biggest losers on the S&P 500 across 2024.
- Model predictions were mixed between LONG and SHORT across the period.
- The decision rule logic was unclear and changed materially across the week, consisting of a mixture of technical metrics and lagged returns across multiple horizons.
- The SHAP feature attribution reports were clearer - indicating that the 1d lagged return was the key driver behind the final prediction.
- The Similarity Report varied with dates across Jun-Sep 2018 but all had a reasonably solid match rate of 92%+.
- So as a user, I am NOT getting a clear and confident prediction and justification.
- Turning to additional sources of data and the Sentiment Report, this again is very mixed with some articles neutral, some somewhat-bearish and some somewhat-bullish. The company received a large order of planes mid-week but many other articles focussed on production delays impacting the industry.

- Given a lack of clear information my decision was to remain neutral on Boeing. This proved a good one as returns fluctuated between positive and negative across the week.

5.4.6 Summary of findings

- The implemented xAI frameworks tools valuable insight, and integrating together can work very well.
- Having a user in the loop however resulted in better decisions than the model on its own.
- The OBV metric proved the most insightful across the 10 technical metrics implemented.

5.5 Model Performance Monitoring

Within an environment such as a bank, before a model can be used by the business it needs to be validated and approved for use by an independent validation team, typically within the Chief Risk Officer (CRO) Organisation. This analysis is performed both to initially approve a model to go live, and then again typically monthly or quarterly to attest to its continued operation. The range of tests to be performed is at the discretion of the individual validator, but regulatory guidance is available. The Bank of England outline four categories [49] which I have assessed below as to how well each is supported by my implementation. Overall the scope of functionality delivered aligns well with the capabilities needed for model validation.

5.5.1 Sensitivity testing

Within sensitivity testing you are assessing the robustness and stability of the model. A combination of views and capabilities are available to the model validator to enable this. Taking as an example, analysis into how sensitive the BAC model is to the On Balance Volume (OBV) trend:

- The Decision Tree/Anchors report shows this to be a driver behind the prediction, see Figure 5.
- The SHAP model level report - specifically the Beeswarm plot - shows both the impact on the model output for each instance within the training set, but also how this impact ranges from high to low feature values, see Figure 7.
- The Counterfactual report generated to vary just the technical metrics includes the OBV metric in 3 from 5 examples - taking both positive and negative values. Taking example 5, this shows the OBV remaining negative and the Relative Strength Index (RSI) shifting from bucket 2 (40-60) to bucket 3(60-100). The validator is now able to see not just sensitivity to the target measure but the relative sensitivities to movement across multiple measures, see Figure 8.

- Finally, the model validator can directly edit the feature data within the final processed dataset to reflect a "what-if" of interest and re-run the model prediction, generating a full suite of views for this new input configuration.

5.5.2 Backtesting

Within backtesting you are comparing model generated predictions to actual results. I have used Backtesting extensively within the Model Performance Report covered in Section 4.4.0.12 to show the model performance across both testing and training windows.

5.5.3 Benchmarking

Within benchmarking you are comparing model estimates with comparable model outcomes. I have demonstrated Benchmarking as part of my implementation with the Alternative Models Report as defined in Section 4.4.0.10. Within Table 13 I include results generated as of 02 Aug 2024 for the five stocks used in Section 5.4. Results of performance testing across the testing dataset are equivalent or better than those generated by the primary deep neural network model, with the AdaBoost and Gradient Boosting Classifiers performing the best.

5.5.4 Parallel running of a challenger model

Finally, an overall tool to assess model performance is through the use of a challenger model. This model is run alongside the main / production model to allow detailed comparison of performance. As part of my implementation I created a decision tree classifier as a challenger model - as described in Section 4.4.0.9. Table 13 also includes Challenger Model results generated as of 02 Aug 2024 for the five stocks used in Section 5.4. Excluding SMCI as an outlier, returns across the testing period are similar, if not quite as strong as those from the primary model - showing the strength of a simpler, explainable by design model approach.

Ticker	Stock	Model	Training	Testing	Return	Model Type
BA	Boeing	Primary	86%	83%	1.08	FF Neural Network
		Challenger	85%	82%	1.00	Decision Tree
		Best Alternative		85%		Gradient Boosting Classifier
FDX	Fedx	Primary	89%	82%	1.47	FF Neural Network
		Challenger	84%	83%	1.38	Decision Tree
		Best Alternative		85%		Gradient Boosting Classifier
LULU	Lululemon Athletica Inc	Primary	86%	81%	1.52	FF Neural Network
		Challenger	83%	82%	1.29	Decision Tree
		Best Alternative		82%		Gradient Boosting Classifier
SMCI	Super Micro Computer Inc	Primary	84%	59%	4.53	FF Neural Network
		Challenger	79%	85%	12.31	Decision Tree
		Best Alternative		83%		AdaBoost
WBA	Walgreens Boots Alliance Inc	Primary	86%	81%	1.48	FF Neural Network
		Challenger	83%	84%	0.58	Decision Tree
		Best Alternative		84%		AdaBoost

Table 13: Primary, Benchmark & Challenger Model Results

6 Legal, Social, Ethical and Professional Issues

6.1 Legal Issues

Before a risk or capital model can be used within a highly regulated industry such as finance, it needs to be assessed and approved by the appropriate regulatory body - for us in the UK this will be the Bank of England Prudential Regulation Authority (PRA). They publish a set of regulations and guidelines which have to be followed from a legal and compliance perspective.

One of my objectives within the introduction (Section 1.5) was to show how the output of xAI methods can be used to manage model risk. The Bank of England Supervisory Statement on Model Risk [49] defines model risk not only as the risk of the model generating incorrect results, but also the risk of the model output being used inappropriately. Within the same source, a model is defined as any quantitative method with an uncertain outcome - ie which includes an element of uncertainty - and so very much applicable to the models being covered within this project.

6.1.1 Model Risk Classification

When a model is identified, the first step is that of classifying the level of model risk. The following factors input into this classification [49]:

1. The potential impact on a firms activities. The more business critical the process using the model output, the higher the risk classification.

2. The nature and quality of the input data. For example, does the model use unstructured as well as structured data ? How well is the provenance of the data sourcing understood ? What controls are in place ?

3. The choice of methodology. Is the methodology established and well understood or new and largely untried ?

4. The level of model interpretability, explainability and transparency. How easy is it to understand both what the model is doing and how it reached its outcome ?

5. The potential for bias either within the design or the data. What dataset has the model been trained on ? How representative is this of the environment that it will be used in when live ?

As shown in Section 2, AI approaches are being used in more and more business critical activities and part of their value is in the ability to generate semantic understanding from unstructured data. They are also relatively new and so would lead to a high classification rating from the first three criteria listed above.

6.1.2 The use of xAI approaches

The use of xAI approaches can directly address the above requirement for transparency and explainability. I have shown for my selected use case how the user can be provided with transparency both into the decision criteria and the training data most closely aligned to that underpinning the prediction. I have shown how an integrated view across xAI approaches can provide a high level of explainability.

6.1.3 General Data Protection Regulation (GDPR)

Another key law which one needs to always be aware of when implementing a system - AI or otherwise - is that of GDPR, which ensures the protection of personal data. This is not applicable within this project as I am only working with publicly available company stock prices and news articles.

6.2 Commercial factors

6.2.0.1 Trading fees The first commercial factor of note is that of trading fees. I have shown within my evaluation within Section 5 how including trading fees can have a dramatic impact on the returns made using the system. One needs to either optimise the model to minimise the number of trades needed while delivering accurate predictions, or have the luxury of a very large portfolio fund which makes trading costs relatively immaterial.

6.2.0.2 Data sourcing Commercial factors also come into play regarding data sourcing. Stock prices are sourced from yFinance. This is a free resource provided for research and educational purposes only. For a real-world implementation, a robust enterprise source should be used which will need to be purchased. It is a similar situation for the technical metrics. My implementation currently makes use of a free student account which is unsupported if anything goes wrong. A real-world implementation would require a service level agreement (SLA) between the two companies related to data provision.

6.3 Social and Ethical Issues

6.3.1 Keeping a human in the loop

One of the key challenges in the use of AI is the amount to which we are comfortable in a machine making the "ultimate decision" - regardless of how accurate it is. This area was explored within my MSc within the Philosophy and Ethics of AI module where we looked at lethal autonomous weapons systems - where the AI was better at targetting and calculating proportionality, but there was a desire to keep a "human in the loop" as they are better placed to weigh up all of the implications before making a final decision. This again was one of my objectives within this project - to show how a human in the loop can result in a better decision making process than an AI on its own.

6.3.2 Algorithmic bias

Another key challenge in the use of an AI system is ensuring that its operation is not biased. Algorithmic bias is a systematic issue within the system that creates an **unfair** outcome for one group of users over another. There are a number of different types of algorithmic bias. The below categories are again as covered in the Philosophy and Ethics of AI module taken within this MSc.

- Data bias. This is where the original training data is biased and so the model will maintain this bias going forward. The last 10 years in the US have been biased towards a bull market - as shown in the trend in Section 4.5. The model as currently trained would therefore benefit from training on a more representative dataset covering bull and bear markets.
- Data bias relative to statistical standards. This is where training data is not representative for use in a new context. For example, the models would need to be retrained in order to work on UK as opposed to US stocks.
- Data bias relative to moral standards. An example here would be the model prioritising portfolio investment in companies based on the wealthy east or west coasts of the US rather than the under-privileged central communities. The inclusion of the human SME can again help this by ensuring that objectives around diversity are included in portfolio selection.
- Algorithm bias. This is where the model is biased by design - eg a restriction to show to the user only the top two stocks. This is not the case within my implementation.
- User and interpretation bias. This **is** a risk within my implementation - as for any AI system. There is a risk that the human expert assumes that the output is correct without challenge. The purpose of this process is for the user to consider alternative sources of data and explore options based on their own knowledge and experience.
- Feedback bias. This is where the model receives feedback on whether predictions were correct or not by the human SME - thereby embedding any of the above biases. There is no model feedback loop within my implementation.

6.4 Professional Issues

All work undertaken for this MSc project has been undertaken following the British Computer Society Chartered Institute for IT Code of Conduct and the Institute of Engineering and Technology Rules of Conduct.

7 Conclusions and Future Work

To summarise what I have achieved within this project:

- I have built a feed-forward neural network for stock price movement prediction.
- The model produces better returns than taking a static position, even in an extended bull market.
- I have implemented four xAI methods covering decision rules, feature importance, similarity and counterfactuals.
- I have shown how integrating these approaches together can provide a user clear justification as to the decision made by the model as well as answering what-if questions.
- I have shown that having a user in the loop can result in better decisions than the model or the user behaving independently.
- I have shown how one can use xAI tools to manage model risk and meet regulatory model risk management requirements.

So xAI methods **can** provide a clear, consistent and non-technical user understandable justification behind the decisions made by a "black-box" AI model. Multiple complementary methods are however needed - integrated together into an end-to-end process in order to deliver actionable "insight" rather than just "data". Finally, while I have been able to demonstrate the potential for excellent performance from a feed-forward neural network, often a simpler "explainable by design" model such as a decision tree performs nearly as well.

7.1 Future work

As mentioned within my exploration of algorithmic bias in Section 6.3.2 additional analysis would be beneficial to see how well the models perform in very different market environments. This has been addressed in part by my selection of the training window to include COVID and the design proposal to re-train the model every week, but the current dataset is biased to a bull-market.

Additionally, there would be value in future work around model improvement - the current model is only marginally better than simple trading strategies. In particular, it would be worthwhile investigating the AdaBoost classifier which is performing very well within the Alternative Model analysis, and the Long Short-Term Memory (LTSM) model architecture which is used extensively within this space as mentioned within my literature review.

Finally, future work is needed to scale up the solution - currently only tested with a maximum of 30 stocks - and productionise on an enterprise platform such as Microsoft Azure for AI. I have however shown that the design and implementation **can** scale to volumes needed to run across a regional trading desk.

References

[1] G. Bansal, T. Wu, J. Zhou, R. Fok, B. Nushi, E. Kamar, M. Ribeiro, and D. Weld, "Does the whole exceed its parts? the effect of ai explanations on complementary team performance," *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

[2] H.-H. Liu, H.-J. Shu, and W.-N. Chiu, "Noxtrader: Lstm-based stock return momentum prediction for quantitative trading," *Advances in Artificial Intelligence and Machine Learning, 3(4), pp. 1671–1684, 95*, 2023.

[3] M. Kim, J. Choi, J. Kim, Y. Ryou, and K.-E. Kim, "Trustworthy residual vehicle value prediction for auto finance," *AI Magazine, 44(4), pp. 394–405*, 2023.

[4] L. Xueling, X. Xiong, and S. Yucong, "Exchange rate market trend prediction based on sentiment analysis," *Computers and Electrical Engineering, 111, 108901*, 2023.

[5] E. Abdellatif, S. Saleh, and H. Hamed, "Corporate financial performance prediction using artificial intelligence techniques," *Lecture Notes in Networks and Systems, 753 LNNS, pp. 25–32*, 2023.

[6] Y. Yang, H.-M. Dai, C.-H. Chao, S. Wei, and C.-F. Yang, "Training a neural network to predict house rents using artificial intelligence and deep learning," *Sensors and Materials, 35(10 P2), pp. 4671–4680*, 2023.

[7] S. Dash, S. Das, S. Sivasubramanian, G. Harsha, and T. Sathish, "Developing ai-based fraud detection systems for banking and finance," *Proceedings of the 5th International Conference on Inventive Research in Computing Applications, ICIRCA 2023, pp. 891–897*, 2023.

[8] P. Sood, C. Sharma, S. Nijjer, and S. Sakhuja, "Review the role of artificial intelligence in detecting and preventing financial fraud using natural language processing," *International Journal of System Assurance Engineering and Management, 14(6), pp. 2120–2135*, 2023.

[9] M. Neyret, J. Ouaggag, and C. Allain, "Trading desk behavior modeling via lstm for rogue trading fraud detection," *DATA 2020 - Proceedings of the 9th International Conference on Data Science, Technology and Applications, pp. 143–150*, 2020.

[10] J. de Moraes Souza, D. de Castro, Y. Peng, and I. Gartner, "A machine learning-based analysis on the causality of financial stress in banking institutions," *Computational Economics*, 2023.

[11] I. Ghosh, R. Jana, R. David, P. Wanke, and Y. Tan, "Modelling financial stress during the covid-19 pandemic: Prediction and deeper insights," *International Review of Economics and Finance, 91, pp. 680–698*, 2024.

[12] Y. Li and G. Wen, "Research and practice of financial credit risk management based on federated learning," *Engineering Letters, 31(1), pp. 271–278*, 2023.

[13] T. Butler and R. Brooks, "Time for a paradigm change: Problems with the financial industry's approach to operational risk," *Risk Analysis*, 2023.

[14] B. Priya and V. Sharma, "Exploring users' adoption intentions of intelligent virtual assistants in financial services: An anthropomorphic perspectives and socio-psychological perspectives," *Computers in Human Behavior, 148, 107912*, 2023.

[15] D. Orhan and M. Genevois, "Analyzing replenishment policies for automated teller machines," *Lecture Notes in Mechanical Engineering, pp. 546–554*, 2024.

[16] G. Lăzăroiu, M. Bogdan, M. Geamănu, L. Ionescu, and R. Ștefănescu, "Artificial intelligence algorithms and cloud computing technologies in blockchain-based fintech management," *Oeconomia Copernicana, 14(3), pp. 707–730*, 2023.

[17] H.-J. Kim, Y.-E. Jeon, W.-S. Jung, Y.-I. Park, and D.-O. Won, "An automated framework for accurately classifying financial tables in enterprise analysis reports," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 14406 LNCS, pp. 374–384*, 2023.

[18] S. Russell and P. Norvig, *Artificial Intelligence : A modern approach, Fourth Edition.* Pearson, 2022.

[19] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems, 2017-December, pp. 4766–4775*, 2017.

[20] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11700 LNCS, pp. 193–209*, 2019.

[21] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *34th International Conference on Machine Learning, ICML 2017, 7, pp. 4844–4866*, 2017.

[22] M. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you? explaining the predictions of any classifier," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, pp. 1135–1144*, 2016.

[23] V. La Gatta, V. Moscato, M. Postiglione, and G. Sperlì, "Pastle: Pivot-aided space transformation for local explanations," *Pattern Recognition Letters, 149, pp. 67–74*, 2021.

[24] Y. Tian and G. Liu, "Mane: Model-agnostic non-linear explanations for deep learning model," *Proceedings - 2020 IEEE World Congress on Services, SERVICES 2020, pp. 33–36, 9283900*, 2020.

[25] D. Watson, "Rational shapley values," *ACM International Conference Proceeding Series, pp. 1083–1094*, 2022.

[26] A. Henelius, K. Puolamäki, H. Boström, L. Asker, and P. Papapetrou, "A peek into the black box: Exploring classifiers by randomization," *Data Mining and Knowledge Discovery, 28(5-6), pp. 1503–1529*, 2014.

[27] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *34th International Conference on Machine Learning, ICML 2017, 7, pp. 5109–5118*, 2017.

[28] Y.-J. Jung, S.-H. Han, and H.-J. Choi, "Explaining cnn and rnn using selective layer-wise relevance propagation," *IEEE Access, 9, pp. 18670–18681, 9320473*, 2021.

[29] T. De, P. Giri, A. Mevawala, R. Nemani, and A. Deo, "Explainable ai: A hybrid approach to generate human-interpretable explanation for deep learning prediction," *Procedia Computer Science, 168, pp. 40–48*, 2020.

[30] M. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018, pp. 1527–1535*, 2018.

[31] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," 2018.

[32] H. Lakkaraju, R. Caruana, E. Kamar, and J. Leskovec, "Faithful and customizable explanations of black box models," *AIES 2019 - Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, pp. 131–138*, 2019.

[33] Y. Ming, H. Qu, and E. Bertini, "Rulematrix: Visualizing and understanding classifiers with rules," *IEEE Transactions on Visualization and Computer Graphics, 25(1), pp. 342–352, 8440085*, 2019.

[34] M. Moradi and M. Samwald, "Post-hoc explanation of black-box classifiers using confident itemsets," *Expert Systems with Applications, 165, 113941*, 2021.

[35] A. Dhurandhar, P.-Y. Chen, R. Luss, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," *Advances in Neural Information Processing Systems, 2018-December, pp. 592–603*, 2018.

[36] T. Le, S. Wang, and D. Lee, "Grace: Generating concise and informative contrastive sample to explain neural network model's prediction," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 238–248*, 2020.

[37] R. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," *FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 607–617*, 2020.

[38] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach, "Face: Feasible and actionable counterfactual explanations," *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, pp. 344–350*, 2020.

[39] D. Wang, Z. Chen, I. Florescu, and B. Wen, "A sparsity algorithm for finding optimal counterfactual explanations: Application to corporate credit rating," *Research in International Business and Finance, 64, 101869*, 2023.

[40] S. Galhotra, R. Pradhan, and B. Salimi, "Explaining black-box algorithms using probabilistic contrastive counterfactuals," *Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 577–590*, 2021.

[41] I. Shin-nosuke, T. Masato, Tand Masato, U. Yasunobu, M. Kazunari, L. Peihsuan, O. Taiki, and Y. Masao, "Example-based explainable ai and its application for remote sensing image classification," *International Journal of Applied Earth Observation and Geoinformation*, 2023.

[42] D. Watson, L. Gultchin, A. Taly, and L. Floridi, "Local explanations via necessity and sufficiency: Unifying theory and practice," *Minds and Machines, 32(1), pp. 185–218*, 2022.

[43] Z. Yang, A. Zhang, and A. Sudjianto, "Enhancing explainability of neural networks through architecture constraints," *IEEE Transactions on Neural Networks and Learning Systems, 32(6), pp. 2610–2621, 9149804*, 2021.

[44] N. Díaz-Rodríguez, A. Lamas, G. Sanchez, J.and Franchi, I. Donadello, S. Tabik, D. Filliat, P. Cruz, R. Montes, and F. Herrera, "Explainable neural-symbolic learning (x-nesyl) methodology to fuse deep learning representations with expert knowledge graphs: The monumai cultural heritage use case," *Information Fusion*, 2022.

[45] J.-H. Chen, S.-C. Chen, Y.-C. Tsai, and C.-S. Shur, "Explainable deep convolutional candlestick learner," *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, PartF162440, pp. 234–237*, 2020.

[46] A. Kulesza, "Determinantal point processes for machine learning," *Foundations and Trends in Machine Learning*, 2012.

[47] Y. Hilpisch, *Python for Finance*. O'Reilly, 2019.

[48] F. Fabozzi, H. Markowitz, P. Kolm, and F. Gupta, "Mean-variance model for portfolio selection," *Encyclopedia of Financial Models*, 2012.

[49] "Model risk management principles for banks," *Bank of England Supervisory statement — SS1/23*, 2023.

A Appendix

This Appendix contains the source code for the Implementation covered in Section 4

A.1 data_sourcing.py

```
import numpy as np
import pandas as pd

# Load in the environment variables for where we store the datasets and how
    much history to source
from dotenv import load_dotenv
import os
load_dotenv()
data_dir = os.getenv('DATA_DIR')
historical_data = os.getenv('HISTORICAL_DATA')

# Turn off the intrusive performance warnings
# Approach taken from stackoverflow.com with the setting of copy_on_write
    to be true suggested by pandas itself within the warning
#
    https://stackoverflow.com/questions/51521526/python-pandas-how-to-supress-performancewarni
pd.options.mode.copy_on_write = True
from warnings import simplefilter
simplefilter(action="ignore", category=pd.errors.PerformanceWarning)

def source_market_data(api_object, stocks):
'''
A function to source the market data from the Yahoo Finance API
:param api_object: the Yahoo Finance API object
:param stocks: the list of stock tickers to source
:return: none
'''
print('Sourcing the EOD market data')
raw = api_object.history(period=historical_data)
# We just need the close data
market_data = raw['Close']
# Set the date to be the index
market_data.index = market_data.index.date
market_data.index.names = ['date']
# Calculate the day over day return
for ticker in stocks:
market_data[ticker + '_Return'] = np.log(market_data[ticker] /
    market_data[ticker].shift(1))
market_data.dropna(inplace=True)
# Save the data down
market_data.to_csv(data_dir + 'market_data.csv')
```

```
def source_metrics(ticker):
'''
A function to source the technical metrics from the Alpha Vantage API.
Approach taken as outlined in the API documentation at
    https://www.alphavantage.co/documentation/.
The code loads the data, sets the index and adds the ticker to the start of
    each column
:param ticker: the list of stock tickers to source
:return: individual pandas dataframe objects containing the sourced
    information for each technical metric
'''
print('Sourcing technical metrics for ' + ticker)
# Source AD
url = 'https://www.alphavantage.co/query?function=AD&symbol=' + ticker +
    '&interval=daily&datatype=csv&apikey=I2UR68ST032EG0J5'
AD = pd.read_csv(url)
AD.set_index('time', inplace=True)
AD.rename(columns={'Chaikin A/D': ticker + '_AD'}, inplace=True)
# Source Ultimate oscillator
url = 'https://www.alphavantage.co/query?function=ULTOSC&symbol=' + ticker
    + '&interval=daily&datatype=csv&apikey=I2UR68ST032EG0J5'
Ult = pd.read_csv(url)
Ult.set_index('time', inplace=True)
Ult.rename(columns={'ULTOSC': ticker + '_ULT'}, inplace=True)
# Source RSI
url = 'https://www.alphavantage.co/query?function=RSI&symbol=' + ticker +
    '&interval=daily&time_period=10&series_type=close&datatype=csv&apikey=I2UR68ST032EG0J5'
RSI = pd.read_csv(url)
RSI.set_index('time', inplace=True)
RSI.rename(columns={'RSI': ticker + '_RSI'}, inplace=True)
# Source Williams
url = 'https://www.alphavantage.co/query?function=WILLR&symbol=' + ticker +
    '&interval=daily&time_period=10&datatype=csv&apikey=I2UR68ST032EG0J5'
WIL = pd.read_csv(url)
WIL.set_index('time', inplace=True)
WIL.rename(columns={'WILLR': ticker + '_WIL'}, inplace=True)
# Source Average Directional Movement Index (ADX)
url = 'https://www.alphavantage.co/query?function=ADX&symbol=' + ticker +
    '&interval=daily&time_period=10&datatype=csv&apikey=I2UR68ST032EG0J5'
ADX = pd.read_csv(url)
ADX.set_index('time', inplace=True)
ADX.rename(columns={'ADX': ticker + '_ADX'}, inplace=True)
# Source AROON
url = 'https://www.alphavantage.co/query?function=AROON&symbol=' + ticker +
    '&interval=daily&time_period=10&datatype=csv&apikey=I2UR68ST032EG0J5'
AROON = pd.read_csv(url)
AROON.set_index('time', inplace=True)
AROON.rename(columns={'Aroon Down': ticker + '_AR_DN', 'Aroon Up': ticker +
    '_AR_UP'}, inplace=True)
```

```
# Source Money Flow Index (MFI)
url = 'https://www.alphavantage.co/query?function=MFI&symbol=' + ticker +
    '&interval=daily&time_period=10&datatype=csv&apikey=I2UR68ST032EG0J5'
MFI = pd.read_csv(url)
MFI.set_index('time', inplace=True)
MFI.rename(columns={'MFI': ticker + '_MFI'}, inplace=True)
# Source DX
url = 'https://www.alphavantage.co/query?function=DX&symbol=' + ticker +
    '&interval=daily&time_period=10&datatype=csv&apikey=I2UR68ST032EG0J5'
DX = pd.read_csv(url)
DX.set_index('time', inplace=True)
DX.rename(columns={'DX': ticker + '_DX'}, inplace=True)
# Source SAR
url = 'https://www.alphavantage.co/query?function=SAR&symbol=' + ticker +
    '&interval=daily&datatype=csv&apikey=I2UR68ST032EG0J5'
SAR = pd.read_csv(url)
SAR.set_index('time', inplace=True)
SAR.rename(columns={'SAR': ticker + '_SAR'}, inplace=True)
# Source On Balance Volume (OBV)
url = 'https://www.alphavantage.co/query?function=OBV&symbol=' + ticker +
    '&interval=daily&datatype=csv&apikey=I2UR68ST032EG0J5'
OBV = pd.read_csv(url)
OBV.set_index('time', inplace=True)
OBV.rename(columns={'OBV': ticker + '_OBV'}, inplace=True)
return AD, Ult, RSI, WIL, ADX, AROON, MFI, DX, SAR, OBV

def technical_metrics_batch(stocks):
'''
A function to build up the technical metric data table one stock at a time.
Calls the sourcing function for each one then concatenates into the overall
    dataframe
:param stocks: the list of stock tickers to source
:return: none
'''
df = pd.DataFrame()
for name in stocks:
# Source technical metrics for each stock
AD, Ult, RSI, WIL, ADX, AROON, MFI, DX, SAR, OBV = source_metrics(name)
df = pd.concat([df, AD, Ult, RSI, WIL, ADX, AROON, MFI, DX, SAR, OBV],
    axis=1)
df.dropna(inplace=True)
df.sort_index(inplace=True)
# Save down the file
df.to_csv(data_dir + 'technical_metrics.csv')

def process_data(tickers):
'''
A function to create the simple moving averages and trends. Called once we
    have the market data and technical metrics.
```

```
    The approach to digitize feature data by the first and second moments of
        the distribution comes from the book "Python for Finance"
    (Yves Hilpisch, 2019, Section 15 on Trading Strategies, Page 508)
    :param tickers: the list of stock tickers
    :return: none
    '''
    print('Pulling together the datasets')
    df_tec = pd.read_csv(data_dir + 'technical_metrics.csv', index_col='time')
    df_tec.index.rename('date', inplace=True)
    df_mkt = pd.read_csv(data_dir + 'market_data.csv', index_col='date')
    df_com = pd.merge(df_mkt, df_tec, how='inner', on='date')
    df_com.dropna(inplace=True)

    # Create a short horizon moving average metric on the S&P
    # Allocate to bins based on the mean and standard deviation
    df_com['SNP_SMA_sht'] = df_com['^GSPC'].rolling(window=40).mean()
    mu = df_com['SNP_SMA_sht'].mean()
    v = df_com['SNP_SMA_sht'].std()
    bins = [mu - v, mu, mu + v]
    df_com['SNP_SMA_sht_bin'] = np.digitize(df_com['SNP_SMA_sht'], bins=bins)

    # Create a long horizon moving average metric on the S&P
    # Allocate to bins based on the mean and standard deviation
    df_com['SNP_SMA_lng'] = df_com['^GSPC'].rolling(window=160).mean()
    mu = df_com['SNP_SMA_lng'].mean()
    v = df_com['SNP_SMA_lng'].std()
    bins = [mu - v, mu, mu + v]
    df_com['SNP_SMA_lng_bin'] = np.digitize(df_com['SNP_SMA_lng'], bins=bins)

    # Create a short horizon moving average metric on the VIX
    # Allocate to bins based on the mean and standard deviation
    df_com['VIX_SMA_sht'] = df_com['^VIX'].rolling(window=40).mean()
    mu = df_com['VIX_SMA_sht'].mean()
    v = df_com['VIX_SMA_sht'].std()
    bins = [mu - v, mu, mu + v]
    df_com['VIX_SMA_sht_bin'] = np.digitize(df_com['VIX_SMA_sht'], bins=bins)

    # Create a long horizon moving average metric on the VIX
    # Allocate to bins based on the mean and standard deviation
    df_com['VIX_SMA_lng'] = df_com['^VIX'].rolling(window=160).mean()
    mu = df_com['VIX_SMA_lng'].mean()
    v = df_com['VIX_SMA_lng'].std()
    bins = [mu - v, mu, mu + v]
    df_com['VIX_SMA_lng_bin'] = np.digitize(df_com['VIX_SMA_lng'], bins=bins)

    for ticker in tickers:
    print('Processing the technical metrics for ' + ticker)
    # Process the AD : Look at the trend
```

```
# If the Accumulation/Distribution Line is trending upward it indicates
    that the price may follow
source_col = '{}_AD'.format(ticker)
new_col = '{}_AD_trend'.format(ticker)
df_com[new_col] = np.sign(df_com[source_col].diff(periods=3))
# Process Ultimate oscillator : Ranges from 0 to 100
# Interpreted as an overbought/oversold indicator when the value is over
    70/below 30
bins = [0, 40, 60, 100]
source_col = '{}_ULT'.format(ticker)
new_col = '{}_ULT_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process RSI : Ranges from 0 to 100
# The RSI is interpreted as an overbought/oversold indicator when the value
    is over 70/below 30
bins = [0, 40, 60, 100]
source_col = '{}_RSI'.format(ticker)
new_col = '{}_RSI_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process Williams : Ranges from 0 to 100
# Inverted scale. Sell signal when crosses 20. Buy signal when crosses 80
bins = [0, -20, -80, -100]
source_col = '{}_WIL'.format(ticker)
new_col = '{}_WIL_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process Average Directional Movement Index (ADX) : Ranges from 0 to 100
    but rarely above 60
# High number is a strong trend. Low number is a weak trend
bins = [0, 20, 40, 60, 100]
source_col = '{}_ADX'.format(ticker)
new_col = '{}_ADX_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process AROON : Up and Down indicators
# Up between 70 and 100 indicates an upward trend. Down between 70 ans 100
    indicates a downward trend
dn_col = '{}_AR_DN'.format(ticker)
up_col = '{}_AR_UP'.format(ticker)
diff_col = '{}_AR_DIFF'.format(ticker)
df_com[diff_col] = df_com[up_col] - df_com[dn_col]
bins = [-100, -40, 40, 100]
final_col = '{}_AR_BIN'.format(ticker)
df_com[final_col] = np.digitize(df_com[diff_col], bins=bins)
# Process Money Flow Index (MFI). Ranges from 0 to 100
# Values above 80 and below 20 indicate market top / bottoms
bins = [0, 30, 70, 100]
source_col = '{}_MFI'.format(ticker)
new_col = '{}_MFI_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process DX. Ranges from 0 to 100 but rarely above 60
```

```
# High number is a strong trend. Low number is a weak trend
bins = [0, 20, 40, 60, 100]
source_col = '{}_DX'.format(ticker)
new_col = '{}_DX_BIN'.format(ticker)
df_com[new_col] = np.digitize(df_com[source_col], bins=bins)
# Process On Balance Volume (OBV)
# Look at the trend
source_col = '{}_OBV'.format(ticker)
new_col = '{}_OBV_trend'.format(ticker)
df_com[new_col] = np.sign(df_com[source_col].diff(periods=3))
# Calculate the Stop And Reverse flag
df_com[ticker + '_SAR_Flag'] = np.sign(df_com[ticker] - df_com[ticker +
    '_SAR'])

# Create a column holding the direction of the return - will be +1 for
    positive and -1 for negative
df_com[ticker + '_Direction'] = np.sign(df_com[ticker +
    '_Return']).astype(int)
# Create an additional column remapping this direction indicator to 1 or 0.
    Used in the xAI functions
df_com[ticker + '_Direction_bin'] = df_com[ticker +
    '_Direction'].apply(lambda x: 1 if x > 0 else 0)

# The approach to create feature data based on lagged returns and digitize
    by the first and second moments of the distribution comes from the book
    "Python for Finance" (Yves Hilpisch, 2019)
# The code below create features based on lagged returns for the last five
    days and is taken from Section 15 (Trading Strategies) of the book
lags = 5
cols = []
# Create the lagged log returns
for lag in range(1, lags + 1):
col = ticker + '_lag_{}'.format(lag)
df_com[col] = df_com[ticker + '_Return'].shift(lag)
cols.append(col)
# Allocate to bins based on the mean and standard deviation
mu = df_com[ticker + '_Return'].mean()
v = df_com[ticker + '_Return'].std()
bins = [mu - v, mu, mu + v]
for col in cols:
col_bin = col + '_bin'
df_com[col_bin] = np.digitize(df_com[col], bins=bins)

# Create a short horizon moving average metric on the EOD close
df_com[ticker + '_SMA_sht'] = df_com[ticker].rolling(window=40).mean()
# Create a long horizon moving average metric on the EOD close
df_com[ticker + '_SMA_lng'] = df_com[ticker].rolling(window=160).mean()
# Allocate to bins based on the mean and standard deviation
mu = df_com[ticker].mean()
```

```
v = df_com[ticker].std()
bins = [mu - v, mu, mu + v]
df_com[ticker + '_SMA_sht_bin'] = np.digitize(df_com[ticker + '_SMA_sht'],
    bins=bins)
df_com[ticker + '_SMA_lng_bin'] = np.digitize(df_com[ticker + '_SMA_lng'],
    bins=bins)
df_com.dropna(inplace=True)
df_com.to_csv(data_dir + 'final_processed_dataset.csv')

def main():
'''
The main function running the data sourcing process
:return: none
'''
import yfinance as yf
import time
import json

# Execution is timed to track how well it scales as the volume of stocks
    increases
start_time = time.time()
# Load in the list of stock tickers from the model_portfolio file
model_portfolio = pd.read_csv(data_dir + 'model_portfolio.csv')
core_stocks = model_portfolio["Stock_Ticker"].tolist()
# Add on the index tickers stored in the environment config file
index_tickers = json.loads(os.environ['INDEX_TICKERS'])
all_tickers = core_stocks + index_tickers
# Create the Yahoo Finance API object
api_object = yf.Tickers(all_tickers)
# Source market data
source_market_data(api_object, all_tickers)
# Source technical metrics
technical_metrics_batch(core_stocks)
# Process the data
process_data(core_stocks)
end_time = time.time()
print('Process execution took %s seconds' % (end_time - start_time))
if __name__ == "__main__":
main()
```

A.2 main.py

```
import numpy as np
import pandas as pd

# Import the reporting functions
import reporting
```

```
# Load in the environment variables for where we find the data and store
    the reports and models
from dotenv import load_dotenv
import os
load_dotenv()
data_dir = os.getenv('DATA_DIR')
report_dir = os.getenv('REPORT_DIR')
model_dir = os.getenv('MODEL_DIR')

import json
import pickle
import time

# Turn off the intrusive performance warnings
# Approach taken from stackoverflow.com with the setting of copy_on_write
    to be true returned by pandas itself within the warning
#
    https://stackoverflow.com/questions/51521526/python-pandas-how-to-supress-performancewarni
from warnings import simplefilter
simplefilter(action="ignore", category=Warning)

def split_data(all_data):
'''
A function to split the model data into training and testing datasets.
    Configured to be 80% train, 20% test.
Uses a sequential split (as opposed to a random split) in order to then
    calculate the daily returns across each window
:param all_data: the full set of market data sourced
:return: the individual training and testing datasets
'''
split_point = int(len(all_data) * 0.8)
training_set = all_data.iloc[:split_point].copy()
testing_set = all_data.iloc[split_point:].copy()
return training_set, testing_set

def train_model(classifier, train, ticker):
'''
A function to train the model.
Called with different classifiers - used for the production MLP model
    training as well as the alternative models
:param classifier: the details of the classifier model to be trained
:param train: the training dataset
:param ticker: the ticker of interest
:return: the model object and the columns used to train the model
'''
print('Training ' + str(classifier) + ' model for ' + ticker)
# Define the columns that we will use to train the model
columns = ([
```

```
# The bucketed lagged returns
ticker+'_lag_1_bin', ticker+'_lag_2_bin', ticker+'_lag_3_bin',
    ticker+'_lag_4_bin', ticker+'_lag_5_bin',
# The stock based technical metrics
ticker+'_AD_trend', ticker+'_ULT_BIN', ticker+'_RSI_BIN',
    ticker+'_WIL_BIN', ticker+'_ADX_BIN',
ticker+'_AR_BIN', ticker+'_MFI_BIN', ticker+'_DX_BIN', ticker+'_OBV_trend',
    ticker+'_SAR_Flag',
# The short and long horizon moving averages
ticker+'_SMA_sht_bin', ticker+'_SMA_lng_bin', 'SNP_SMA_sht_bin',
    'SNP_SMA_lng_bin', 'VIX_SMA_sht_bin', 'VIX_SMA_lng_bin',
])
# Train the model to predict the movement direction using the training
    dataset
model = classifier
model.fit(train[columns], train[ticker + '_Direction_bin'])
return model, columns

def run_prediction(model, columns, train, test, ticker):
'''
A function to run the model to predict the movement direction for that
    stock. Run across both the training and testing datasets.
Then calculates the return based on that prediction. Multiplies yesterday's
    position prediction by today's return.
Needed to reflect that you take action based on the prediction and then see
    the results of this action the next day
:param model: the model object
:param columns: the columns used to train the model
:param train: the training dataset
:param test: the testing dataset
:param ticker: the ticker of interest
:return: none
'''
print('Running prediction for ' + ticker)

# Calculate the predicted model position on the testing dataset - will be 0
    or 1
test[ticker + '_model_position'] = model.predict(test[columns])
# Remap to +1 and -1. Needed by the following step
# eg if you go short and the return is negative, then the return is positive
test[ticker + '_model_position'] = test[ticker +
    '_model_position'].apply(lambda x: 1 if x > 0 else -1)
# Calculate what this then gives in terms of return on the testing dataset
test[ticker + '_model_results'] = test[ticker + '_model_position'].shift(1)
    * test[ticker + '_Return']
# Calculate the predicted model position on the training dataset - will be
    0 or 1
train[ticker + '_model_position'] = model.predict(train[columns])
# Remap to +1 and -1. Needed by the following step
```

```
train[ticker + '_model_position'] = train[ticker +
    '_model_position'].apply(lambda x: 1 if x > 0 else -1)
# Calculate what this then gives in terms of return on the training dataset
train[ticker + '_model_results'] = train[ticker +
    '_model_position'].shift(1) * train[ticker + '_Return']

def explain_ANCHORS(date, stock, model, test, columns, explain_target,
    forecast):
'''
A function to run the Anchor decision rule explain
:param date: the prediction date
:param stock: the stock of interest
:param model: the model object
:param test: the testing dataset
:param columns: the columns used to train the model
:param explain_target: the input data on the prediction date to be explained
:param forecast: the model movement prediction
:return: none
'''
from alibi.explainers import AnchorTabular

# Link to the example integration of the ANCHOR functionality using
    scikit-learn and tabular data which I adapted for my implementation
#
    https://docs.seldon.io/projects/alibi/en/stable/examples/anchor_tabular_iris.html

# There are different explainer options. We are using the tabular explainer
    as using tabular data
# Takes a callable prediction function. Implemented for my model using a
    lamdba function
predict_fn = lambda x: model.predict(x)
exp = AnchorTabular(predictor=predict_fn, feature_names=columns)
dataset = test[columns].to_numpy()
# Fit the explainer to the test dataset
# Bins numerical features as given. I have left values to be as per the
    default
exp.fit(dataset, disc_perc=(25, 50, 75))
# Calculate the anchor result for just the explain target to a threshold of
    99%
target = explain_target[columns].to_numpy()
explanation = exp.explain(target, threshold=0.99)
anchor_string = '%s' % (' AND '.join(explanation.anchor))
# Call the function to generate the report
reporting.anchors_report(date, stock, anchor_string, explanation.precision,
    explanation.coverage, forecast)

def explain_SHAP(ticker, explain_date, model, train_data, test_data,
    columns):
'''
```

```
A function to generate the SHAP feature importance explain
:param ticker: the ticker of interest
:param explain_date: the report execution date
:param model: the model object used for the prediction
:param train_data: the training dataset
:param test_data: the testing dataset
:param columns: the columns used to train the model
:return: none
'''
import shap

# Link to the example integration of the SHAP functionality using
    scikit-learn which I adapted for my implementation
#
    https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/model_agnostic/Ir
# My first step is to cut back our 10 years of historical data into 100
    samples from the training set
small_train = shap.sample(train_data, 100)
# I then create the explainer
# There are different explainer options. I am using the Kernel explainer
explainer = shap.KernelExplainer(model.predict, small_train[columns])
# I now generate the SHAP values for the last 100 days of the test dataset
small_test = test_data.tail(100)
# The last row in the resulting SHAP values table will be our explain date
shap_values = explainer(small_test[columns])
# I now call the report generation function
reporting.SHAP_report(shap_values, ticker, explain_date)

def explain_DICE(date, ticker, model, train, columns, explain_target,
    forecast):
'''
A function to run the diverse counterfactuals (DICE) explain
:param date: the prediction date
:param ticker: the stock of interest
:param model: the model object
:param train: the training dataset
:param columns: the columns used to train the model
:param explain_target: the input data on the prediction date to be explained
:param forecast: the model movement prediction
:return: none
'''
import dice_ml

# Link to the example integration of the DICE functionality with
    scikit-learn which I adapted for my implementation
# https://interpret.ml/DiCE/readme.html#getting-started-with-dice

# Define the data to be used. Takes a copy of the training data and adds a
    column with the predicted direction
```

```
# Tells the explainer that this prediction direction is the outcome to be
    varied
# Needs to be either 0 or 1 in order to use the "give me the opposite"
    functionality
new_train = train[columns + [ticker + '_Direction_bin']].copy()
d = dice_ml.Data(dataframe=new_train, continuous_features=columns,
    outcome_name=ticker + '_Direction_bin')
# Define the model to be used to generate the counterfactuals
m = dice_ml.Model(model=model, backend="sklearn", model_type="classifier")
# Create the explainer using the data and model specified
exp = dice_ml.Dice(d, m, method="random")
# Store the original values of the input data used by the model
org = explain_target[columns].values.tolist()

# If no counterfactuals are possible, an error is raised
# This is actually a valid outcome and so this error is neatly trapped and
    reported
try:
# First try and generate counterfactuals using all features
# Generates 5 examples
e2 = exp.generate_counterfactuals(explain_target[columns], total_CFs=5,
    desired_class='opposite')
# Save down
e2.cf_examples_list[0].final_cfs_df.to_csv(path_or_buf=report_dir +
    'counterfactuals_all_' + ticker + '.csv', index=False)
# Call the function to generate the report with the results
# Pass in the context that uses all features
reporting.DICE_report(e2, org, forecast, date, ticker, columns, 'ALL')
except Exception as ex:
print("Not possible to generate counterfactuals for " + ticker + " using
    all features")
try:
# Load in the user defined columns which can be varied from the environment
    configuration
cf_columns = json.loads(os.environ['CF_COLUMNS'])
features = [ticker + column for column in cf_columns]
# Try and generate counterfactuals using the restricted feature list
    provided by the user
e3 = exp.generate_counterfactuals(explain_target[columns], total_CFs=5,
    desired_class='opposite', features_to_vary=features)
# Save down
e3.cf_examples_list[0].final_cfs_df.to_csv(path_or_buf=report_dir +
    'counterfactuals_filtered_' + ticker + '.csv', index=False)
# Call the function to generate the report with the results
# Pass in the context that uses specified features
reporting.DICE_report(e3, org, forecast, date, ticker, columns,
    str(features))
except Exception as ex:
```

```
print("Not possible to generate counterfactuals for " + ticker + " using
    the restricted list of features")

def explain_Similarity(ticker, train, test, explain_date_data, columns,
    forecast):
'''
A function to find the most similar date in the training dataset to that
    being explained.
Views the input data on each date as a vector. Uses cosine similarity to
    find the closest match.
Calls the reporting function to generate the report showing details on both
    the explain and matched dates
:param ticker: the stock of interest
:param train: the training dataset
:param test: the testing dataset
:param explain_date_data: the input data on the prediction date to be
    explained
:param columns: the columns used by the model
:param forecast: the model movement prediction
:return: none
'''
from sklearn.metrics.pairwise import cosine_similarity

print('Running similarity explain')
dir_col = ticker + '_Direction'
# Filter out from the training dataset just those dates with the same
    forecast as the explain date
samples = train[train[dir_col] == forecast]
# For all dates left, calculate the cosine similarity between this date and
    the explain date
result = cosine_similarity(samples[columns], explain_date_data[columns])
# Find the date with the maximum similarity across all in the training set
index_max = np.argmax(result)
# Find the confidence in the match
confidence = max(result)
compdate = train[index_max:index_max + 1].index[0]
explain_date = explain_date_data.index[0]
print('Explain date : ' + str(explain_date))
print('Most similar date in training set : ' + str(compdate))
# Save down
pd.concat([explain_date_data[columns], train[columns][index_max:index_max +
    1]]).to_csv(path_or_buf=report_dir + 'similarity' + ticker + '.csv',
    index=False)
# The Similarity report needs various details on the preceding and
    following dates
# Defined here
explain_date_window = test[ticker + '_Return'].tail(7)
comp_date_previous = train.iloc[index_max - 7 + 1:index_max + 1]
comp_date_following = train.iloc[index_max:index_max + 7]
```

```
comp_date_values = train.loc[[compdate]]
comp_date_min1_values = train.shift(1)[train.index == compdate]
# Finally we call the report
reporting.daily_report(compdate, ticker, comp_date_values,
    comp_date_min1_values, 'SIMILAR', explain_date, confidence,
    explain_date_window, comp_date_previous, comp_date_following, forecast)

def run_challenger_model(train, test, portfolio, date):
'''
A function to run the Decision Tree Classifier Challenger Model. Also
    includes the reporting component
:param train: the training dataset
:param test: the testing dataset
:param portfolio: the portfolio of stocks to be assessed
:param date: the execution date
:return: none
'''
from sklearn.tree import DecisionTreeClassifier

#  A Decision Tree Classifier challenger model is trained and a performance
    report generated for each stock in the portfolio
for stock in portfolio:
# Define the model with a maximum depth of 4
# Needs to be easily explainable so keeps simple
model = DecisionTreeClassifier(criterion="entropy", max_depth=4,
    random_state=42)
# Train the model
model_trained, columns_trained = train_model(model, train, stock)
# Take a copy of the training and testing datasets for processing separate
    from the main model datasets
train_cp = train.copy(deep=True)
test_cp = test.copy(deep=True)
# Generate the model predictions
run_prediction(model_trained, columns_trained, train_cp, test_cp, stock)
# Now generate the report
reporting.challenger_report(model_trained, columns_trained, train_cp,
    test_cp, stock, date)

def portfolio_opt(test, core_stocks, date_t, date_tmin1):
'''
A function to run the portfolio optimisation analysis
:param test: the testing dataset
:param core_stocks: the list of stocks within the portfolio
:param date_t: the date of execution
:param date_tmin1: the date before the execution date
:return: none
'''
import yfinance as yf
from pypfopt import EfficientFrontier
```

```
from pypfopt import risk_models
from pypfopt import expected_returns
from pypfopt.discrete_allocation import DiscreteAllocation,
    get_latest_prices

position_results = []
# Pull out the model generated movement predictions for the last two days
for stock in core_stocks:
new_position = test[stock + '_model_position'].tail(1).iloc[0]
old_position = test[stock + '_model_position'].tail(2).iloc[0]
position_results.append([stock, date_t, new_position, date_tmin1,
    old_position])
# Source the list of stocks and the covariance horizon from the environment
    variables
portfolio_stocks = json.loads(os.environ['OPTIMISATION_PORTFOLIO'])
historical_data = os.getenv('COVARIANCE_HORIZON')
position_results_df = pd.DataFrame(position_results)

# The approach and code here comes directly from the PyPortfolioOpt
    documentation
# https://pypi.org/project/pyportfolioopt/
# First, source the historical close data for the stocks in the portfolio
tickers = yf.Tickers(portfolio_stocks)
raw = tickers.history(period=historical_data)
market_data = raw['Close']

# Next we calculate the expected returns and sample covariance using this
    data
mu = expected_returns.mean_historical_return(market_data)
S = risk_models.sample_cov(market_data)

# Find the efficient frontier - the set of optimal portfolios which
    minimise the risk for a target return
# Allows both long and short positions
ef = EfficientFrontier(mu, S, weight_bounds=(-1, 1))
# Find the portfolio from this set which gives the maximum return for the
    lowest risk - the maximal Sharpe ratio
ef.max_sharpe()
# Helper method provided by the package to set any weights whose absolute
    values are below the cutoff to zero, and round the rest
cleaned_weights = ef.clean_weights()
# Save down the weights
ef.save_weights_to_file(report_dir + "weights.csv") # saves to file

# Use these weights to generate a discrete allocation
# Firstly, load in the money available from the environment configuration
    file
portfolio_value = os.getenv('PORTFOLIO_VALUE')
latest_prices = get_latest_prices(market_data)
```

```
# Generate the discrete allocation
da = DiscreteAllocation(cleaned_weights, latest_prices,
    total_portfolio_value=int(portfolio_value))
allocation, leftover = da.greedy_portfolio()
# Finally, generate the report
reporting.portfolio_report(portfolio_stocks, date_t, position_results_df,
    cleaned_weights, mu, S, ef, portfolio_value, allocation, leftover)

def main():
'''
The main batch function. Controls the execution of each individual process
    using the environment variables
:return: none
'''
from sklearn.neural_network import MLPClassifier

# Load in the list of stock tickers from the model_portfolio file
model_portfolio = pd.read_csv(data_dir + 'model_portfolio.csv')
core_stocks = model_portfolio["Stock_Ticker"].tolist()
model_portfolio.index = model_portfolio["Stock_Ticker"]

# Source the model input data
final_data = pd.read_csv(data_dir + 'final_processed_dataset.csv',
    index_col='date')

# Set the prediction date = date_t = the last row in the dataset
# Also sets the previous date = date t-1, which is used in the reporting
date_t = final_data.tail(1).index[0]
date_t_data = final_data.loc[[date_t]]
date_tmin1 = final_data.tail(2).index[0]
date_tmin1_data = final_data.loc[[date_tmin1]]

# Split the input data into testing and training sets
train, test = split_data(final_data)

# All of the core processes are controlled in the run by environment
    variables
# If set to be True then the process is run

if os.getenv('TRAINING') == 'True':
# The process is timed in order to assess how well it
# scales with volumes
start_time = time.time()
# Get the hidden layer settings from the model_portfolio dataframe
layers = model_portfolio["Hidden_Layers"]
for stock in core_stocks:
# Set the hidden layer settings for this specific stock
architecture = layers.get(stock)
hidden_layers = json.loads(architecture)
```

```
# Train a new MLP classifier for each stock
model = MLPClassifier(max_iter=1000, hidden_layer_sizes=hidden_layers,
    random_state=42)
model_res, columns_res = train_model(model, train, stock)
# Pickle the model object and save down
with open(model_dir + stock + '_model.pkl', 'wb') as f:
    pickle.dump(model_res, f)
with open(model_dir + stock + '_columns.pkl', 'wb') as f:
    pickle.dump(columns_res, f)
end_time = time.time()
# Report the training time
print('Process execution took %s seconds' % (end_time - start_time))

# For each stock in the portfolio, we run the prediction
# We then run each of the reporting or explain processes as set in the
    environment variables
for stock in core_stocks:
# Load in the model
with open(model_dir + stock + '_model.pkl', 'rb') as f: model_trained =
    pickle.load(f)
with open(model_dir + stock + '_columns.pkl', 'rb') as f: model_columns =
    pickle.load(f)
# Run the prediction
run_prediction(model_trained, model_columns, train, test, stock)
forecast = int(test[stock + '_model_position'].tail(1).iloc[0])

if os.getenv('DAILY') == 'True':
print('Generating daily report for ' + stock)
reporting.daily_report(date_t, stock, date_t_data, date_tmin1_data,
    'INPUT', date_t, 1, [], [], [], forecast)
if os.getenv('SENTIMENT') == 'True':
print('Generating sentiment report for ' + stock)
trimmed_explain_date = date_t.replace('-', '')
reporting.sentiment_report(stock, trimmed_explain_date)
if os.getenv('HISTORY') == 'True':
print('Generating trade history for ' + stock)
reporting.history_report(test, stock, date_t)
if os.getenv('ANCHORS') == 'True':
print('Generating anchors report for ' + stock)
explain_ANCHORS(date_t, stock, model_trained, test, model_columns,
    date_t_data, forecast)
if os.getenv('SHAP') == 'True':
print('Generating feature importance report for ' + stock)
explain_SHAP(stock, date_t, model_trained, train, test, model_columns)
if os.getenv('SIMILARITY') == 'True':
print('Generating similarity report for ' + stock)
explain_Similarity(stock, train, test, date_t_data, model_columns, forecast)
if os.getenv('DICE') == 'True':
print('Generating counterfactuals for ' + stock)
```

```
explain_DICE(date_t, stock, model_trained, train, model_columns,
    date_t_data, forecast)
if os.getenv('MODEL_PERF') == 'True':
print('Generating report on model performance')
reporting.performance_report(model_trained, train, test, model_columns,
    stock, date_t)

if os.getenv('PORTFOLIO') == 'True':
print('Running portfolio optimisation')
portfolio_opt(test, core_stocks, date_t, date_tmin1)

if os.getenv('CHALLENGER') == 'True':
print('Running challenger model training and testing')
run_challenger_model(train, test, core_stocks, date_t)

if os.getenv('ALTERNATIVE') == 'True':
print('Running alternative model training and testing')
reporting.alt_models_report(train, test, core_stocks, date_t)

if __name__ == "__main__":
main()
```

A.3 reporting.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Import the reportlab document templates and style sheets
# The approach to build up the page content using an empty list and add
    content as flowables comes from the reportlab userguide
# https://docs.reportlab.com/reportlab/userguide/ch5_platypus/
from reportlab.platypus import SimpleDocTemplate, Paragraph, Image,
    NextPageTemplate, BaseDocTemplate, Frame, PageTemplate, Table, TableStyle
from reportlab.lib.pagesizes import landscape, A4
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib import colors
sample_style_sheet = getSampleStyleSheet()

# Load in the environment variable for where to store the reports
from dotenv import load_dotenv
import os
load_dotenv()
report_dir = os.getenv('REPORT_DIR')

# Set the table style used within all reports
table_style = TableStyle([
```

```
('BACKGROUND', (0, 0), (-1, 0), colors.blue),
('TEXTCOLOR', (0, 0), (-1, 0), colors.white),
('ALIGN', (0, 0), (-1, -1), 'LEFT'),
('FONTNAME', (0, 0), (-1, 0), 'Times-Bold'),
('FONTSIZE', (0, 0), (-1, 0), 8),
('BOTTOMPADDING', (0, 0), (-1, 0), 4),
('BACKGROUND', (0, 1), (-1, -1), colors.antiquewhite),
('TEXTCOLOR', (0, 1), (-1, -1), colors.black),
('ALIGN', (0, 1), (-1, -1), 'LEFT'),
('FONTNAME', (0, 1), (-1, -1), 'Times-Roman'),
('FONTSIZE', (0, 1), (-1, -1), 8),
('BOTTOMPADDING', (0, 1), (-1, -1), 4)
])

def process_news_article(article, ticker, content_flow):
'''
A function to process an individual news article to pull out the headline,
    sentiment and relevance score
:param article: the article object
:param ticker: the stock ticker of interest
:param content_flow: the list variable where we are building up the report
    content
:return: none
'''
# Each article, will be linked to a number of stock tickers. We look at
    each one in turn
tickers = article.get('ticker_sentiment')
for j in range(0, len(tickers)):
company = tickers[j].get('ticker')
# Check to see if this article refers to the stock we want
if company == ticker:
relevance = tickers[j].get('relevance_score')
relevance_int = float(relevance)
# Check to see if the relevance is above a set threshold. Gets rid of the
    noise
if relevance_int >= 0.1:
title = article.get('title')
url = article.get('url')
sentiment = tickers[j].get('ticker_sentiment_label')
# Add to the report content variable
paragraph_txt = Paragraph("Headline : " + title,
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Sentiment : " + str(sentiment) + ' : Relevance
    measure ' + str(round(relevance_int, 2)), sample_style_sheet['Bullet'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph(url, sample_style_sheet['Italic'])
content_flow.append(paragraph_txt)
```

```
def sentiment_report(ticker, news_date):
'''
A function to generate the sentiment report
:param ticker: the stock ticker of interest
:param news_date: the report execution date
:return: none
'''
import requests

# The list variable containing the report content. Built up as we go along
content_flow = []
# The API needs a range start and end date. Set both of these to be the
    explain date
range_start_str = news_date + 'T0130'
range_end_str = news_date + 'T0130'
paragraph_txt = Paragraph("Sentiment Report for " + ticker + ' on ' +
    news_date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# Source the sentiment data on the explain date for this stock ticker
url = ('https://www.alphavantage.co/query?function=NEWS_SENTIMENT&tickers='
    + ticker + '&time_from=' + range_start_str + '&time_to=' + range_end_str
    + '&limit=1000&apikey=I2UR68ST032EG0J5')
r = requests.get(url).json()
# Pull the results into a dataframe
df = pd.DataFrame.from_dict(r)
# Check to see if anything has been returned
if len(df) > 0:
# Filter to just the date we want
df['time'] = df['feed'].apply(lambda i: i.get('time_published'))
df['date'] = df['time'].str[0:8]
new_df = df[df['date'] == news_date]
# Pull out the details for each news article returned
# Vectorised call on each row in the dataframe calling the
    process_news_article function
df['sentiment'] = new_df['feed'].apply(process_news_article, args=(ticker,
    content_flow))
# Save down the report created
pdf = SimpleDocTemplate(report_dir + "Sentiment_Report_" + ticker + "_" +
    news_date + ".pdf")
pdf.build(content_flow)

def daily_report(date, ticker, t, tminus1, context, comp_date, matchrate,
    window1, window2, window3, forecast):
'''
A function to generate the tables showing the input data for a specific
    date.
Used by the Daily Report to show the input data which went into the
    prediction.
```

```
Used by the Similarity Report to show the input data on the date deemed to
    have the closest match.
:param date: the report execution date
:param ticker: the stock ticker of interest
:param t: the input data for the date on which the report is being run
:param tminus1: the input data for the day before the report execution date
:param context: which report is required - Daily or Similarity
:param comp_date: for the similarity report - the comparison date
:param matchrate: for the similarity report - the match rate
:param window1: for the similarity report - the EOD price trend leading up
    to the model date
:param window2: for the similarity report - the EOD price trend leading up
    to the identified similar date
:param window3: for the similarity report - the EOD price trend following
    the identified similar date
:param forecast: the model prediction
:return: none
'''
# Helper functions used when creating the report for features which put
    values into one of a number of bins
# Creates a blank mask and then inserts the feature into the appropriate
    slot
def allocate_from_3_bins(t, feature, feature_name):
mask = [' ', ' ', ' ']
bin = t[feature][0]
mask[bin - 1] = feature_name
return mask
def allocate_from_4_bins(t, feature, feature_name):
mask = [' ', ' ', ' ', ' ']
bin = t[feature][0]
mask[bin - 1] = feature_name
return mask
# The list variable containing the report content. Built up as we go along
content_flow = []
if context == 'SIMILAR':
# Create the report header if we're running a similarity report
pdf = SimpleDocTemplate(report_dir + "Similarity_Report_" + ticker + "_" +
    comp_date + ".pdf")
paragraph_txt = Paragraph("Similarity Report for " + ticker + " on " +
    comp_date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# The most important thing here is to provide context to the user around
    the comparison
# We start by showing the price trend for the seven days leading up to the
    prediction date
# Saves down as an image which is then added to the report flow
paragraph_txt = Paragraph("EOD return trend leading up to model date",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
```

```
plt.bar(window1.index, window1)
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=300, height=150)
content_flow.append(image1)
# We then output the recommendation based on the model prediction
if forecast == 1:
paragraph_txt = Paragraph("Recommendation : Go long",
    sample_style_sheet['Heading3'])
else:
paragraph_txt = Paragraph("Recommendation : Go short",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# We then output the most similar date found along with the match rate
# If the match rate is low then an example like today was not found in the
    training set
paragraph_txt = Paragraph("Most similar date in training set : " + date,
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Match % of " + str(round(matchrate[0], 2)),
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# For comparison we then show the price trend for the seven days leading up
    to the most similar date
# Saves down as an image which is then added to the report flow
paragraph_txt = Paragraph("EOD return trend leading up to " + date,
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
plt.bar(window2.index, window2[ticker + '_Return'])
plt.savefig('img2.png', bbox_inches="tight")
plt.clf()
image2 = Image('img2.png', width=300, height=150)
content_flow.append(image2)
# And finally we show the price trend for the seven days following the most
    similar date
# Saves down as an image which is then added to the report flow
paragraph_txt = Paragraph("EOD return trend following " + date,
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
plt.bar(window3.index, window3[ticker + '_Return'])
plt.savefig('img3.png', bbox_inches="tight")
plt.clf()
image3 = Image('img3.png', width=300, height=150)
content_flow.append(image3)
paragraph_txt = Paragraph('Input data on ' + date,
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
else:
# Create the report header if we're running a Daily Report
```

```
pdf = SimpleDocTemplate(report_dir + "Daily_Report_" + ticker + "_" + date
    + ".pdf")
paragraph_txt = Paragraph("Daily report for " + ticker + ' on ' + date,
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# Output the recommendation based on the model prediction
if forecast == 1:
paragraph_txt = Paragraph("Recommendation : Go long",
    sample_style_sheet['Heading2'])
else:
paragraph_txt = Paragraph("Recommendation : Go short",
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# Show the input data used by the model in a user-friendly format
# Will either be the prediction date or the similarity date based on the
    context from which the function was called
paragraph_txt = Paragraph("COB Market Metrics",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Direction/Trend is day over day",
    sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
# The first table displays the core market metrics along with their
    direction or trend
table1 = []
table1.append(['Feature', 'Value', 'Direction/Trend'])
table1.append(['EOD Close', round(t[ticker][0], 2), np.sign(t[ticker][0] -
    tminus1[ticker][0])])
table1.append(['Log Return', round(t[ticker + '_Return'][0], 2),
    np.sign(t[ticker + '_Return'][0] - tminus1[ticker + '_Return'][0])])
table1.append(['S&P 500', round(t['^GSPC'][0], 2), np.sign(t['^GSPC'][0] -
    tminus1['^GSPC'][0])])
table1.append(['VIX', round(t['^VIX'][0], 2), np.sign(t['^VIX'][0] -
    tminus1['^VIX'][0])])
table_final = Table(table1)
table_final.setStyle(table_style)
content_flow.append(table_final)
# The second table displays the moving average and lagged return data which
    has been bucketed related to the mean and moving average
# I decided not to include the actual values as makes too busy - losing the
    overall messaging
paragraph_txt = Paragraph("COB calculated measures, bucketed wrt the
    overall mean and SD", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
table2 = []
table2.append(['< mean - SD', '< mean', '> mean', '> mean + SD'])
table2.append(allocate_from_4_bins(t, ticker+'_SMA_sht_bin', 'SMA short'))
table2.append(allocate_from_4_bins(t, ticker+'_SMA_lng_bin', 'SMA long'))
table2.append(allocate_from_4_bins(t, 'SNP_SMA_sht_bin', 'SnP SMA short'))
```

```
table2.append(allocate_from_4_bins(t, 'SNP_SMA_lng_bin', 'SnP SMA long'))
table2.append(allocate_from_4_bins(t, 'VIX_SMA_sht_bin', 'VIX SMA short'))
table2.append(allocate_from_4_bins(t, 'VIX_SMA_lng_bin', 'VIX SMA long'))
table2.append(allocate_from_4_bins(t, ticker + '_lag_1_bin', '1d Lag
    Returns'))
table2.append(allocate_from_4_bins(t, ticker + '_lag_2_bin', '2d Lag
    Returns'))
table2.append(allocate_from_4_bins(t, ticker + '_lag_3_bin', '3d Lag
    Returns'))
table2.append(allocate_from_4_bins(t, ticker + '_lag_4_bin', '4d Lag
    Returns'))
table2.append(allocate_from_4_bins(t, ticker + '_lag_5_bin', '5d Lag
    Returns'))
table_final = Table(table2)
table_final.setStyle(table_style)
content_flow.append(table_final)
# The third table displays the technical metrics which are included in the
    model relating to direction of movement or trend
# The only exception are the Aroon up and down values. The difference is
    used by the model but there is value to the user in showing both
paragraph_txt = Paragraph("Technical metrics, included in the model wrt
    trend", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Chaikin A/D line + On balance volume : Trend
    over the last 3 days", sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Parabolic SAR flag : Difference between
    Parabolic SAR and the EOD close. 1 if positive, -1 if negative",
    sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Aroon Up/Down values : Difference day on day",
    sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
table3 = []
table3.append(['Feature', 'Value', 'Direction/Trend/Difference'])
table3.append(['Chaikin A/D line (AD)', round(t[ticker+'_AD'][0], 2),
    t[ticker+'_AD_trend'][0]])
table3.append(['On balance volume (OBV)', round(t[ticker+'_OBV'][0], 2),
    t[ticker+'_OBV_trend'][0]])
table3.append(['Parabolic SAR (SAR)', round(t[ticker+'_SAR'][0], 2),
    t[ticker + '_SAR_Flag'][0]])
table3.append(['Aroon (AROON) UP values', round(t[ticker+'_AR_UP'][0], 2),
    t[ticker+'_AR_UP'][0] - tminus1[ticker+'_AR_UP'][0]])
table3.append(['Aroon (AROON) DOWN values', round(t[ticker+'_AR_DN'][0],
    2), t[ticker+'_AR_DN'][0] - tminus1[ticker+'_AR_DN'][0]])
table_final = Table(table3)
table_final.setStyle(table_style)
content_flow.append(table_final)
```

```
# The last four tables display the technical metrics which are included in
    the model relating to specific bucketed values
# Before each table, text is added to remind the user how to read the
    results
paragraph_txt = Paragraph("Technical metrics, included in the model wrt
    bucketed value", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Ultimate Oscillator + Relative Strength Index :
    Values over 70 indicate overbought conditions, values under 30 indicate
    oversold conditions", sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
table4 = []
table4.append(['0 - 40', '40 - 60', '60-100'])
table4.append(allocate_from_3_bins(t, ticker+'_ULT_BIN', 'Ult Osc'))
table4.append(allocate_from_3_bins(t, ticker+'_RSI_BIN', 'RSI'))
table_final = Table(table4)
table_final.setStyle(table_style)
content_flow.append(table_final)
paragraph_txt = Paragraph("Directional Movement Index + Average Directional
    Movement Index : Typically 0-60 with a high number being a strong trend.
    Values over 70 indicate overbought conditions. Values under 30 indicate
    oversold conditions", sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
table5 = []
table5.append(['0 - 20', '20 - 40', '40 - 60', '60-100'])
table5.append(allocate_from_4_bins(t, ticker + '_DX_BIN', 'DX'))
table5.append(allocate_from_4_bins(t, ticker+'_ADX_BIN', 'ADX'))
table_final = Table(table5)
table_final.setStyle(table_style)
content_flow.append(table_final)
paragraph_txt = Paragraph("Williams %R values : Below 20 indicates
    overbought condition. Values over 80 indicate an oversold condition",
    sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
table6 = []
table6.append(['-100 to -80', '-80 to -20', '-20 to 0', '0'])
table6.append(allocate_from_4_bins(t, ticker+'_WIL_BIN', 'WILLIAMS %R'))
table_final = Table(table6)
table_final.setStyle(table_style)
content_flow.append(table_final)
paragraph_txt = Paragraph("Money Flow Index : Values above 80 indicate
    market tops. Values below 20 indicate market bottoms",
    sample_style_sheet['Heading5'])
content_flow.append(paragraph_txt)
table7 = []
table7.append(['0 to 30', '30 to 70', '70 to 100', '100'])
table7.append(allocate_from_4_bins(t, ticker+'_MFI_BIN', 'MFI'))
table_final = Table(table7)
table_final.setStyle(table_style)
```

```
content_flow.append(table_final)
# Save down the report
pdf.build(content_flow)
# Tidy up by removing the images
if context == 'SIMILAR':
os.remove("img1.png")
os.remove("img2.png")
os.remove("img3.png")

def anchors_report(date, ticker, rule, precision, coverage, forecast):
'''
A function to generate the report of the results of the Anchors Decision
    Rule explain
:param date: the report execution date
:param ticker: the ticker of interest
:param rule: the anchor rule
:param precision: the anchor rule precision
:param coverage: the anchor rule coverage
:param forecast: the model movement prediction
:return: none
'''
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph("Anchors Report for " + ticker + ' on ' + date,
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# Output the recommendation based on the model prediction
if forecast == 1:
paragraph_txt = Paragraph("Recommendation : Go long",
    sample_style_sheet['Heading2'])
else:
paragraph_txt = Paragraph("Recommendation : Go short",
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# Output the decision rule generated
paragraph_txt = Paragraph("Decision rule anchoring the forecast",
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph(rule, sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Output the confidence and precision of the decision rule
paragraph_txt = Paragraph("Confidence :", sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Precision " + str(round(precision, 2)) + ",
    Coverage " + str(round(coverage, 2)), sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Save down the report
pdf = SimpleDocTemplate(report_dir + "Anchors_Report_" + ticker + "_" +
    date + ".pdf")
```

```
pdf.build(content_flow)

def DICE_report(counterfactuals, original, forecast, date, ticker, columns,
    filter):
'''
A function to generate the report containing the counterfactual results
:param counterfactuals: the counterfactual examples generated
:param original: the original input data which went into the model
    prediction
:param forecast: the model movement prediction
:param date: the report execution date
:param ticker: the ticker of interest
:param columns: the list of columns included within the counterfactual
    generation
:param filter: context for the report - whether the counterfactuals include
    all features or a user filtered set
:return: none
'''
import math

newdf = counterfactuals.cf_examples_list[0].final_cfs_df.values.tolist()
# In the report, I only want to show the changed values - if not changed,
    show as a "-" for clarity
# I therefore need to go through each element in each counterfactual
    example in turn to see if different
# The function visualize_as_dataframe() within the DICE package can do this
    but only from within the IPython / Jupyter environment
# The code below comes from display_df() which is called by
    visualize_as_dataframe() and does just what I need
# https: // interpret.ml / DiCE / dice_ml.html #
    dice_ml.diverse_counterfactuals.CounterfactualExamples.visualize_as_dataframe
for ix in range(counterfactuals.cf_examples_list[0].final_cfs_df.shape[0]):
for jx in range(len(original[0])):
if not isinstance(newdf[ix][jx], str):
if math.isclose(newdf[ix][jx], original[0][jx], abs_tol=0.01):
newdf[ix][jx] = '-'
else:
newdf[ix][jx] = str(newdf[ix][jx])
else:
if newdf[ix][jx] == original[0][jx]:
newdf[ix][jx] = '-'
else:
newdf[ix][jx] = str(newdf[ix][jx])
# Add the actual model prediction to the end of the dataframe
results = pd.DataFrame(original + newdf, columns=columns + ['Forecast'])
# Transpose so the features are rows and each counterfactual example is a
    column
results_tran = results.transpose()
```

```
# Add a column containing a short description of each feature and what it's
    measuring re bucketing, trends etc
results_tran['Feature Attribute'] = ['1d lagged returns (bin num)\n< mean -
    SD; < mean; > mean; > mean + SD',
'2d lagged returns (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'3d lagged returns (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'4d lagged returns (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'5d lagged returns (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Chaikin AD trend\nUP(+1), DN(-1)',
'Ultimate Oscillator (bin num)\n0 - 40; 40 - 60; 60-100',
'Relative strength index (bin num)\n0 - 40; 40 - 60; 60-100',
'Williams Value (bin num)\n-100 to -80; -80 to -20; -20 to 0; 0',
'Avg directional movement (bin num)\n0 - 20; 20 - 40; 40 - 60; 60-100',
'Aroon value trend\nUP(+1), DN(-1)',
'Money flow index (bin num)\n0 to 30; 30 to 70; 70 to 100; 100',
'Directional movement (bin num)\n0 - 20; 20 - 40; 40 - 60; 60-100',
'On balance volume trend\nUP(+1), DN(-1)',
'Parabolic SAR flag\n+ or - diff to EOD Close',
'Short stock SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Long stock SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Short SnP SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Long SnP SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Short VIX SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Long VIX SMA trend (bin num)\n< mean - SD; < mean; > mean; > mean + SD',
'Model forecast\nLong(+1), Short(-1)']
# Make this column containing the feature descriptions the first one in the
    table
col = results_tran.columns.tolist()
col.insert(0, col.pop())
results_tran = results_tran[col]
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_1 = Paragraph("Counterfactual examples",
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_1)
paragraph_1 = Paragraph(ticker + " on " + date,
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_1)
# Remind the user of the prediction made and so what this counterfactual
    example is telling them
paragraph_1 = Paragraph("to get the opposite prediction from model result
    of " + str(forecast), sample_style_sheet['Heading2'])
content_flow.append(paragraph_1)
# This function is called under two contexts : Add a row to the report
    explaining which one
if filter == 'ALL':
# Counterfactuals generated with no restrictions
paragraph_1 = Paragraph("No user restrictions",
    sample_style_sheet['Heading3'])
```

```
pdf = SimpleDocTemplate(report_dir + "Counterfactuals_All_" + ticker + "_"
    + date + ".pdf")
else:
# Counterfactuals generated under user restrictions
paragraph_1 = Paragraph("Restricted to varying : " + filter,
    sample_style_sheet['Heading3'])
pdf = SimpleDocTemplate(report_dir + "Counterfactuals_Filtered_" + ticker +
    "_" + date + ".pdf")
content_flow.append(paragraph_1)
# Finally, convert the dataframe containing the counterfactual examples to
    a table able to be added to the report
table1_data = []
table1_data.append(['Feature attributes', 'Input data', 'CF 1', 'CF 2', 'CF
    3', 'CF 4', 'CF 5'])
for i, row in results_tran.iterrows():
table1_data.append(list(row))
table1_data = table1_data[:-1]
table_final = Table(table1_data)
table_final.setStyle(table_style)
content_flow.append(table_final)
# Save down the report
pdf.build(content_flow)

def SHAP_report(shap_values, ticker, explain_date):
'''
A function to generate both the individual example SHAP report, and the
    model level SHAP report
:param shap_values: the explains object containing the generated SHAP values
:param ticker: the ticker of interest
:param explain_date: the report execution date
:return: none
'''
import shap

# During development, the SHAP reporting library functions very
    occasionally raised errors
# I have therefore wrapped this function in an exception handling framework
    to ensure the overall batch run is never affected
try:
# We generate first the model level report
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph("SHAP feature attribution",
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Model level results for " + ticker + " on " +
    explain_date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
```

```
# The report consists of three plots. Each is generated and saved down as
    an image before being loaded into the report content variable
# Includes the values across all examples included within the shap_values
    results
# The bar plot
paragraph_txt = Paragraph("Bar plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.plots.bar(shap_values, show=False)
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=300, height=150)
content_flow.append(image1)
# The beeswarm plot
paragraph_txt = Paragraph("Beeswarm plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.plots.beeswarm(shap_values, show=False)
plt.savefig('img2.png', bbox_inches="tight")
plt.clf()
image2 = Image('img2.png', width=300, height=150)
content_flow.append(image2)
# The summary plot
paragraph_txt = Paragraph("Summary plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.summary_plot(shap_values, plot_type='violin', show=False)
plt.savefig('img3.png', bbox_inches="tight")
plt.clf()
image3 = Image('img3.png', width=300, height=200)
content_flow.append(image3)
# Build the report
pdf = SimpleDocTemplate(report_dir + "Model_SHAP_Values_" + ticker + "_" +
    explain_date + ".pdf")
pdf.build(content_flow)
# Tidy up by removing the images
os.remove("img1.png")
os.remove("img2.png")
os.remove("img3.png")

# Now produce the SHAP report for the specific explain date
content_flow = []
paragraph_txt = Paragraph("SHAP feature attribution",
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Drivers behind the forecast for " + ticker + "
    on " + explain_date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# Again the report consists of three plots. Each is generated and saved
    down as an image before being loaded into the report content variable
# Includes values for just the last example included within the shap_values
    results - ie today, the explain date
```

```
# The bar plot
paragraph_txt = Paragraph("Bar plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.plots.bar(shap_values[-1], show=False)
plt.savefig('img4.png', bbox_inches="tight")
plt.clf()
image4 = Image('img4.png', width=300, height=150)
content_flow.append(image4)
# The waterfall plot
paragraph_txt = Paragraph("Waterfall plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.plots.waterfall(shap_values[-1], show=False)
plt.savefig('img5.png', bbox_inches="tight")
plt.clf()
image5 = Image('img5.png', width=300, height=150)
content_flow.append(image5)
# The force plot
paragraph_txt = Paragraph("Force plot", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
shap.plots.force(shap_values[-1], matplotlib=True, show=False)
plt.savefig('img6.png', bbox_inches="tight")
plt.clf()
image6 = Image('img6.png', width=300, height=150)
content_flow.append(image6)
# Build the report
pdf = SimpleDocTemplate(report_dir + "SHAP_Values_" + ticker + "_" +
    explain_date + ".pdf")
pdf.build(content_flow)
# Tidy up by removing the images
os.remove("img4.png")
os.remove("img5.png")
os.remove("img6.png")
except Exception as ex:
print('Issue generating the SHAP reports for ' + ticker + ' : Code ' +
    str(ex))

def history_report(test, ticker, date):
'''
A function to generate the trade history report
:param test: the testing dataset
:param ticker: the ticker of interest
:param date: the report execution date
:return: none
'''
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph('Trade history report for ' + ticker + ' as of '
    + date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
```

```
# First, creates a plot of the EOD price across the testing period
paragraph_txt = Paragraph('Long term return trend',
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
test[ticker + '_Return'].cumsum().apply(np.exp).plot(figsize=(10, 6))
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=400, height=300)
content_flow.append(image1)
# Add a table containing the detailed information over the last 20 days
# Includes the predicted model position
# Includes a placeholder to store the actual position taken - will come
    from integration with a "real" trading system
paragraph_txt = Paragraph('Detailed view over the last 20 days',
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph('EOD prices, daily returns and positions taken',
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
test['date_field'] = test.index
test['rnd_cob_price'] = round(test[ticker], 2)
test['rnd_return'] = round(test[ticker + '_Return'], 2)
table1_data = []
table1_data.append(['Date', 'EOD Price', 'Daily Return', 'Model Position',
    'Actual Position'])
for i, row in test[['date_field', 'rnd_cob_price', 'rnd_return',
    ticker+'_model_position']].tail(20).iterrows():
table1_data.append(list(row))
table_final = Table(table1_data)
table_final.setStyle(table_style)
content_flow.append(table_final)
pdf = SimpleDocTemplate(report_dir + "Trade_History_" + ticker + "_" + date
    + ".pdf")
# Build the report
pdf.build(content_flow)
# Tidy up by removing the image
os.remove("img1.png")

def portfolio_report(portfolio, date, position_results, cleaned_weights,
    mu, S, ef, portfolio_value, allocation, leftover):
'''
A function to generate the portfolio optimisation report
:param portfolio: the list of stocks to include within the portfolio
    optimisation
:param date: the report execution date
:param position_results: the results of the model movement predictions for
    the date (t) and the day before (t-1)
:param cleaned_weights: the cleaned weights for te calculated max sharpe
    value portfolio
```

```
:param mu: the expected returns for the portfolio
:param S: the sample covariance for the portfolio
:param ef: the efficient frontier object
:param portfolio_value: the funds available for allocation
:param allocation: what that discrete allocation is
:param leftover: funds left over
:return: none
'''

from pypfopt import EfficientFrontier
from pypfopt import plotting

# We start by generating the header for the report
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph("Portfolio report as of " + date,
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
# Lists the stocks included - is configurable by the user in the
    environment file
paragraph_txt = Paragraph("Portfolio : " + str(portfolio),
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# For each stock, output the model movement predictions for the explain
    date (today) and the previous date (yesterday)
paragraph_txt = Paragraph("Model movement predictions - today and
    yesterday", sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# For each stock, output the model movement predictions for the explain
    date (today) and the previous date (yesterday)
table1_data = []
table1_data.append(['Stock Ticker', 'Date', 'Movement prediction',
    'Previous date', 'Previous Movement prediction'])
for i, row in position_results.iterrows():
table1_data.append(list(row))
table_final = Table(table1_data)
table_final.setStyle(table_style)
content_flow.append(table_final)

# Now add the results of the portfolio optimisation
paragraph_txt = Paragraph("Portfolio weights",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Print our the portfolio weights
paragraph_txt = Paragraph(str(cleaned_weights).strip('OrderedDict'),
    sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)
# Also show the weights graphically
plotting.plot_weights(cleaned_weights)
```

```
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=250, height=250)
content_flow.append(image1)

# Print out the various portfolio portfolio statistics available
paragraph_txt = Paragraph("Portfolio details",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
er, vol, sharpe = ef.portfolio_performance()
paragraph_txt = Paragraph('Expected annual return (%): ' + str(round(er,
    2)), sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph('Annual volatility (%) ' + str(round(vol, 2)),
    sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph('Sharpe Ratio: ' + str(round(sharpe, 2)),
    sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)

# Output to the report the discrete allocation along with any remaining
    funds
paragraph_txt = Paragraph("Remaining funds from portfolio of $"+
    portfolio_value + " : ${:.2f}".format(leftover),
    sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Discrete allocation : " + str(allocation),
    sample_style_sheet['Normal'])
content_flow.append(paragraph_txt)

# Finally, two plots provided by the package supporting the calculation
fig, ax = plt.subplots()
paragraph_txt = Paragraph("Portfolio efficient frontier",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Firstly, the efficient frontier
# Shows the individual stock volatility and returns - along with the set of
    optimal portfolios which minimise the risk for a target return
ef = EfficientFrontier(mu, S, weight_bounds=(-1, 1))
ef1 = ef.deepcopy()
plotting.plot_efficient_frontier(ef, ax=ax, ef_param='return',
    show_tickers=True)
ef1.max_sharpe()
ret_tangent, std_tangent, _ = ef1.portfolio_performance()
ax.scatter(std_tangent, ret_tangent, marker="*", s=100, c="r", label="Max
    Sharpe")
plt.savefig('img2.png', bbox_inches="tight")
plt.clf()
image2 = Image('img2.png', width=250, height=250)
```

```
content_flow.append(image2)

# Next the covariance matrix across the stocks within the portfolio
paragraph_txt = Paragraph("Portfolio covariance",
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
plotting.plot_covariance(S)
plt.savefig('img3.png', bbox_inches="tight")
plt.clf()
image3 = Image('img3.png', width=250, height=250)
content_flow.append(image3)
# Build the report
pdf = SimpleDocTemplate(report_dir + 'Portfolio_Report_' + date + '.pdf')
pdf.build(content_flow)
# Tidy up afterwards by removing the images
os.remove("img1.png")
os.remove("img2.png")
os.remove("img3.png")

def performance_report(model, train, test, columns, ticker, date):
'''
A function to generate the model performance report
:param model: a model object
:param train: the training dataset
:param test: the testing dataset
:param columns: the columns used to train the model
:param ticker: the ticker of interest
:param date: the report execution date
:return: none
'''
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph('Model performance report for ' + ticker + ' as
    of ' + date, sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)

# My approach uses vectorized backtesting to assess model performance
    (multiplying the log returns by the positions) as well as
# comparing against simply holding the stock over the horizon. Adapted from
    the book "Python for Finance" (Yves Hilpisch, 2019) Section 15 : Trading
    Strategies

# The model score on the training set
paragraph_txt = Paragraph('Training set model score = ' +
    str(round(model.score(train[columns], train[ticker + '_Direction_bin']),
    2)), sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# The model return on the training set along with that for just holding the
    stock
```

```
# The plot across the window is generated and saved down before being
    loaded into the report flow object
train[[ticker + '_Return', ticker +
    '_model_results']].cumsum().apply(np.exp).plot(figsize=(10, 6))
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=200, height=200)
content_flow.append(image1)

# The model score on the testing set
paragraph_txt = Paragraph('Testing set model score = ' +
    str(round(model.score(test[columns], test[ticker + '_Direction_bin']),
    2)), sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# The model return on the testing set along with that for just holding the
    stock
test[[ticker + '_Return', ticker +
    '_model_results']].cumsum().apply(np.exp).plot(figsize=(10, 6))
plt.savefig('img2.png', bbox_inches="tight")
plt.clf()
image2 = Image('img2.png', width=200, height=200)
content_flow.append(image2)
# Overall returns across the test period for both static and model based
    strategies
paragraph_txt = Paragraph('Return seen for just holding stock = ' +
    str(round(np.exp(test[ticker + '_Return'].sum()), 2)),
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph('Return seen using model strategy = ' +
    str(round(np.exp(test[ticker + '_model_results'].sum()), 2)),
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# Calculate the number of trades made if following the model strategy
    across the testing period
# Making a trade costs money and so impacts the return of any strategy
paragraph_txt = Paragraph('Across testing period of ' + str(len(test)) + '
    days', sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
trades_made = (test[ticker + '_model_position'].diff() != 0).sum()
paragraph_txt = Paragraph('Trades made using strategy = ' +
    str(trades_made), sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# Generate the report
pdf = SimpleDocTemplate(report_dir + 'Model_Perf_' + ticker + '_' + date +
    '.pdf')
pdf.build(content_flow)
# Tidy up afterwards by removing the images
os.remove("img1.png")
os.remove("img2.png")
```

```
def challenger_report(model, columns, train_cp, test_cp, stock, date):
'''
A function to generate the challenger report
:param model: the trained challenger model
:param columns: the columns on which the model was trained
:param train_cp: the enhanced training datset
:param test_cp: the enhanced testing dataset
:param stock: the stock of interest
:param date: the report execution date
:return: none
'''
import sklearn.tree as tr

# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph("Challenger model : Decision Tree Classifier",
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph(stock + " on " + date,
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# The challenger model score on the training set
paragraph_txt = Paragraph('Training set model score = ' +
    str(round(model.score(train_cp[columns], train_cp[stock +
    '_Direction_bin']), 2)), sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# The challenger model return across the training set
train_cp[[stock + '_Return', stock +
    '_model_results']].cumsum().apply(np.exp).plot(figsize=(10, 6))
plt.savefig('img1.png', bbox_inches="tight")
plt.clf()
image1 = Image('img1.png', width=200, height=200)
content_flow.append(image1)
# The challenger model score on the testing set
paragraph_txt = Paragraph('Testing set model score = ' +
    str(round(model.score(test_cp[columns], test_cp[stock +
    '_Direction_bin']), 2)), sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# The challenger model return across the testing set
test_cp[[stock + '_Return', stock +
    '_model_results']].cumsum().apply(np.exp).plot(figsize=(10, 6))
plt.savefig('img2.png', bbox_inches="tight")
plt.clf()
image2 = Image('img2.png', width=200, height=200)
content_flow.append(image2)
# Print out the returns over the test period for both static and model
    based strategies
```

```
paragraph_txt = Paragraph('Return seen for just holding stock = ' +
    str(round(np.exp(test_cp[stock + '_Return'].sum()), 2)),
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph('Return seen using model strategy = ' +
    str(round(np.exp(test_cp[stock + '_model_results'].sum()), 2)),
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Use the sklearn plot_tree function which generates a pictorial view of
    the decision tree
# Allows the user to compare against the production model anchor report
plt.figure(figsize=(40, 12))
tr.plot_tree(model, feature_names=columns, class_names=['+1', '-1'],
    fontsize=8)
plt.savefig('img3.png', bbox_inches="tight")
plt.clf()
# For the decision tree plot, we need to change the page orientation to be
    landscape to enable it to be seen clearly
# To do this I have used the following code taken from stackoverflow
#
    https://stackoverflow.com/questions/50660395/reportlab-how-to-change-page-orientation/5066
pdf = BaseDocTemplate(report_dir + 'Challenger_Report_' + stock + "_" +
    date + '.pdf', pagesize=A4, rightMargin=25, leftMargin=25, topMargin=25,
    bottomMargin=25)
portrait_frame = Frame(pdf.leftMargin, pdf.bottomMargin, pdf.width,
    pdf.height, id='portrait_frame ')
landscape_frame = Frame(pdf.leftMargin, pdf.bottomMargin, pdf.height,
    pdf.width, id='landscape_frame ')
pdf.addPageTemplates([PageTemplate(id='portrait', frames=portrait_frame),
    PageTemplate(id='landscape', frames=landscape_frame,
    pagesize=landscape(A4))])
content_flow.append(NextPageTemplate('landscape'))
image3 = Image('img3.png', height=400, width=700)
content_flow.append(image3)
# Build the report
pdf.build(content_flow)
# Tidy up afterwards by removing the images
os.remove("img1.png")
os.remove("img2.png")
os.remove("img3.png")

def alt_models_report(train, test, portfolio, date):
'''
A function to generate benchmark results for the same dataset using a
    number of different classifiers
:param train: the training dataset
:param test: the testing dataset
:param portfolio: the portfolio of stocks
:param date: the execution date
```

```
:return: none
'''
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier,
    GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.gaussian_process.kernels import RBF
from sklearn.neighbors import KNeighborsClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.svm import SVC
from __main__ import train_model

# Specifies the classifiers to be included
# I have used the list and configuration as specified in the sklearn
    documentation
#
    https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.htm
# I have excluded the MLP Classifier as this is our production model and
    the Decision Tree Classifier which is our challenger
# I have added to the sklearn list the Gradient Boosting Classifier which
    came up in the literature review
classifiers = [
KNeighborsClassifier(3),
SVC(kernel="linear", C=0.025, random_state=42),
SVC(gamma=2, C=1, random_state=42),
GaussianProcessClassifier(1.0 * RBF(1.0), random_state=42),
RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1,
    random_state=42),
AdaBoostClassifier(algorithm="SAMME", random_state=42),
GaussianNB(),
GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
    max_depth=1, random_state=42)
]
# A report is generated for each stock in the portfolio for each
    alternative model
for stock in portfolio:
# The list variable containing the report content. Built up as we go along
content_flow = []
paragraph_txt = Paragraph("Alternative model performance report",
    sample_style_sheet['Heading1'])
content_flow.append(paragraph_txt)
paragraph_txt = Paragraph("Testing dataset for " + stock + " on " + date,
    sample_style_sheet['Heading2'])
content_flow.append(paragraph_txt)
# Add the details for each classifier in turn
for classifier in classifiers:
paragraph_txt = Paragraph("Model : " + str(classifier),
    sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Train the model
```

```
model_trained, columns_trained = train_model(classifier, train, stock)
# Calculate the model score when run on the training dataset
paragraph_txt = Paragraph('Model score = ' +
    str(round(model_trained.score(test[columns_trained], test[stock +
    '_Direction_bin']), 2)), sample_style_sheet['Heading3'])
content_flow.append(paragraph_txt)
# Build the report
pdf = SimpleDocTemplate(report_dir + 'Alt_Model_Report_' + stock + "_" +
    date + '.pdf')
pdf.build(content_flow)
```

A.4 .env

```
REPORT_DIR = "Reports/"
DATA_DIR = "Data/"
MODEL_DIR = "Models/"

INDEX_TICKERS = '["^VIX", "^GSPC"]'
HISTORICAL_DATA = "10y"

CF_COLUMNS = '["_AD_trend", "_ULT_BIN", "_RSI_BIN", "_WIL_BIN", "_ADX_BIN",
    "_AR_BIN", "_MFI_BIN", "_DX_BIN", "_OBV_trend", "_SAR_Flag"]'

OPTIMISATION_PORTFOLIO = '["SMCI", "AVGO", "WMT", "XOM", "BAC", "FDX",
    "WBA", "BA", "LULU"]'
COVARIANCE_HORIZON = "10y"
PORTFOLIO_VALUE = 1000000

TRAINING = True
DAILY = True
SENTIMENT = True
HISTORY = True
ANCHORS = True
SHAP = True
SIMILARITY = True
DICE = True
PORTFOLIO = True
MODEL_PERF = True
CHALLENGER = True
ALTERNATIVE = True
```

A.5 model_portfolio.csv

Stock_Ticker	Hidden_Layers
SMCI	[16, 3]
AVGO	[16, 16, 3]
WMT	[16]
XOM	[16]
BAC	[16, 16, 3]
FDX	[16, 3]
WBA	[16, 3]
LULU	[16, 16, 3]
BA	[16]