POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Multi-task learning methods for intraday stock trading

Supervisors

Prof. Luca CAGLIERO

PhD. Jacopo FIOR

Candidate

Gabriele SALVO

2021/2022

Abstract

Can news text data add a significant value in a multi-task neural network scenario? Which preprocessing method is best suitable to predict price movements? Which time granularity is the best in an intraday trading system?
This work addresses these questions. In the first experiment we have explored different news sentiment methods, both human-rated and word embedding. Provided the first result, we focus on different time granularities and different quantile labelling. The first trading system has let us choose the 2-hours granularity as this decreases the number of signals and the trading costs. Worth mentioning that it has both a selling strategy based only on the predictions and one based on technical trading signals. The third set of neural network experiments is focused on RNN cells and dense layers. The second trading system explores new time periods in the technical analysis signals, a new buy signal filter and a selling strategy based on the previous price label that shows interesting results. A permutation importance computation has been done to find whether the news feature is helping us in the prediction.

Acknowledgements

It has been a tough journey, a lot of challenges have been faced. I want to thank my family because their support has helped me to overcome them. Several friends have been following my adventure during these years, their sympathy has been an excellent travel companion. It is also a duty to thank Professor Luca Cagliero and Jacopo Fior from Politecnico di Torino for their guidance.

Table of Contents

List of Tables

List of Figures

Acronyms

RNN
Recurrent Neural Network

GRU
Gated Recurrent Unit

LSTM
Long Short Term Memory

GloVe
Global Vectors

RL
Reinforcement Learning

VADER
Valence Aware Dictionary and sEntiment Reasoner

OTC
Over-The-Counter market

NASDAQ
National Association of Securities Dealers Automated Quotation

ATVI
Activision Blizzard, Inc.

ADBE
Adobe Inc.

AMD

Advanced Micro Devices, Inc.

ALXN

Alexion Pharmaceuticals, Inc.

ALGN

Align Technology, Inc.

GOOGL

Alphabet Inc.

GOOG

Alphabet Inc.

AMZN

Amazon.com, Inc.

AAL

American Airlines Group Inc.

AMGN

Amgen Inc.

ADI

Analog Devices, Inc.

AAPL

Apple Inc.

AMAT

Applied Materials, Inc.

ASML

ASML Holding N.V.

ADSK

Autodesk, Inc.

ADP

Automatic Data Processing, Inc.

BIDU

Baidu, Inc.

BIIB

Biogen Inc.

BMRN

BioMarin Pharmaceutical Inc.

BKNG

Booking Holdings Inc.

AVGO

Broadcom Inc.

CDNS

Cadence Design Systems, Inc.

CELG

Celgene Corporation

CERN

Cerner Corporation

CHTR

Charter Communications, Inc.

CHKP

Check Point Software Technologies Ltd.

CTAS

Cintas Corporation

CSCO

Cisco Systems, Inc.

CTXS

Citrix Systems, Inc.

CTSH

Cognizant Technology Solutions Corporation

CMCSA

Comcast Corporation

COST

Costco Wholesale Corporation

CSX

CSX Corporation

CTRP

China Ritar Power Corporation

DLTR

Dollar Tree, Inc.

EBAY

eBay Inc.

EA

Electronic Arts Inc.

EXPE

Expedia Group, Inc.

FB

Facebook, Inc.

FAST

Fastenal Company

FISV

Fiserv, Inc.

GILD

Gilead Sciences, Inc.

HAS

Hasbro, Inc.

HSIC

Henry Schein, Inc.

IDXX

IDEXX Laboratories, Inc.

ILMN

Illumina, Inc.

INCY

Incyte Corporation

INTC

Intel Corporation

INTU

Intuit Inc.

ISRG

Intuitive Surgical, Inc.

JBHT

J.B. Hunt Transport Services, Inc.

JD

JD.com, Inc.

KLAC

KLA Corporation

KHC

The Kraft Heinz Company

LRCX

Lam Research Corporation

LBTYA

Liberty Global plc

LBTYK

Liberty Global plc

LULU

Lululemon Athletica Inc.

MAR

Marriott International, Inc.

MXIM

Maxim Integrated Products, Inc.

MELI

MercadoLibre, Inc.

MCHP

Microchip Technology Incorporated

MU

Micron Technology, Inc.

MSFT

Microsoft Corporation

MDLZ

Mondelez International, Inc.

MNST

Monster Beverage Corporation

MYL

Mylan N.V.

NTAP

NetApp, Inc.

NTES

NetEase, Inc.

NFLX

Netflix, Inc.

NVDA

NVIDIA Corporation

NXPI

NXP Semiconductors N.V.

ORLY

O'Reilly Automotive, Inc.

PCAR

PACCAR Inc

PAYX

Paychex, Inc.

PYPL

PayPal Holdings, Inc.

PEP

PepsiCo, Inc.

QCOM

QUALCOMM Incorporated

REGN

Regeneron Pharmaceuticals, Inc.

ROST

Ross Stores, Inc.

SIRI

Sirius XM Holdings Inc.

SWKS

Skyworks Solutions, Inc.

SBUX

Starbucks Corporation

SYMC

Symantec Corporation

SNPS

Synopsys, Inc.

TMUS

T-Mobile US, Inc.

TTWO

Take-Two Interactive Software, Inc.

TSLA

Tesla, Inc.

TXN

Texas Instruments Incorporated

ULTA

Ulta Beauty, Inc.

UAL

United Airlines Holdings, Inc.

VRSN

VeriSign, Inc.

VRSK

Verisk Analytics, Inc.

VRTX

Vertex Pharmaceuticals Incorporated

WBA

Walgreens Boots Alliance, Inc.

WDAY

Workday, Inc.

WDC

Western Digital Corporation

WLTW

Willis Towers Watson Public Limited Company

WYNN

Wynn Resorts, Limited

XEL

Xcel Energy Inc.

XLNX

Xilinx, Inc.

NDAQ

NASDAQ

Chapter 1

Introduction

Nowadays the interest in buying and selling stocks is really high, actors in play may vary a lot depending on the amount of money invested: we have both independent traders and institutions like banks and funds buying stocks and selling those when the price has risen a certain amount, in case of a LONG position; the profit comes from the price difference between the selling and buying signals. When referring to a SHORT position instead, traders sell a share which is actually not theirs and closes it when the price is below a given threshold. The birth of Internet and World Wide Web has broadened the range of possible acquirers through the use of web platforms that help traders put orders, both by hand and automatically.
We can distinguish traders in *technical analysts* and *fundamental* ones: the former just look and the price chart or at some particular function of these, called technical indicators, to predict price movements in a low-mid term horizon; fundamental analysis focuses on the company itself, searching for those that are underrated and may grow in the future, as share price in a long term will reflect the real stock value.
There are several quantitative approaches to trading: these techniques analyze historical data to take investment decisions. These actions may also be performed automatically. In this work we use machine-learning algorithms to predict price movement in a short term, in an intraday scenario.
Machine Learning is improving performances in many fields of research and technology. The main goal of a machine learning algorithm is to learn from an input, i.e. an image, an audio, a data-point from a machine sensor and a label a representation, either supervised, i.e. a label put in an image telling us whether there is a dog or a cat, or unsupervised, clustering data in groups depending on its features. Setting up a loss function that measures the distance between predicted data and true values results in obtaining a model probability distribution, as close as possible to the real data distribution. The most promising application in the financial field rely on Support Vector Machines, Deep Reinforcement Learning, Naive Bayes or Neural Networks, as reported in [1] and [2].

Multi-task learning [3] [4] is a concept used in several machine learning problems: we leverage the representation of one single problem requiring the machine learning algorithm, in our case a Neural Network, to learn the representation of other related tasks. The objective function in this way is a combination of several objective functions, each one related to a part of the whole problem. The information from these related tasks is a source of inductive bias, i.e. a set of assumptions that the predictor uses to improve generalization.
In descriptive statistics, a time series is a set of variables ordered with respect to the time, and it expresses the dynamics of a phenomenon. We can describe with time series a lot of things, such as earthquake waves, products purchased in a time period, temperatures in a city. If we decide to take measurements in equally-spaced time points, then we talk about discrete time series. In our experiments we start from 1-minute frequency data and we downsample to see what happens. Depending on the number of variables in a single datapoint, we distinguish between univariate and multivariate time series: in our experiments we start considering the open price and we add also technical analysis features.
In this thesis we explore the use of multi-task learning to forecast stock prices and support intraday stock trading. We consider both historical stock price series data and financial news related to the underlying stocks. Another aim of our research is finding out whether using news data might improve results in a multi-task learning approach. Stocks on the same business field may be strongly correlated, so the rise of one may affect the others; similarly, a positive news might influence the whole sector. Moreover we want to find out whether downsampling the data might be useful in terms of signals generated: each transaction has a fee, so decreasing them might improve our porfolio performance.
In the first experiment we have explored different news sentiment methods, count vectors, VADER sentiment and GloVe word embeddings. Provided the first result, we focus on different time granularities and different quantile labelling. The first trading system has both a selling strategy based only on the predictions and one based on technical trading signals. The third set of neural network experiments is focused on different types of RNN cells and different number of dense layers after the RNNs. The second trading system explores new time periods in the technical analysis signals, a new buy signal filter and a selling strategy based on the previous price label that shows interesting results. A permutation importance computation has been done to find whether news feature is helping us in the prediction.
We carried out an empirical evaluation of the ML-based trading strategies on the stocks belonging to the NASDAQ-100 market. Firstly we compared single- with multi-task learning strategies. The results confirm the benefits of using multi-tasking in terms of ROE and volatility. Secondly, we compare the proposed approaches with a state-of-the-art Reinforcement Learning strategy [5]. The classification-based approaches perform consistently better than RL in terms of

annual return and the main portfolio ratio, such as Omega and Sharpe. Strategies are very favorable when we use the previous label strategy. Finally we conducted a permutation importance analysis. It has ruled out the relative importance of the news-related VADER feature: the additional information provided by news data turned out to be negligible compared to the price-related features, as it is not as frequent as the price information. In fact just a few stocks have a significant amount of news headlines, which help in the prediction.

Overview. This document is organized as follows. Chapter 2 explores the main contributions made by other researchers in this field. Chapter 3 gives an overall understanding of neural network concept in the multi-task learning view, recurrent cells and theory behind the news preprocessing methods. Chapter 4 formalizes the proposed approach in all its configurations. Chapter 5 dives into financial data and news preprocessing, addressing timespan, and general statistics about the dataset. Chapter 6 describes in detail all different configuration, both neural network and trading system experiments; it shows training and validation plots, equity line plots and permutation importance charts for the multi-task learning experiments. Chapter 7 sets the conclusions of our work.

Chapter 2

Related work

This section briefly describes methods that other researchers previously used to tackle the same problem, predicting the future of some specific stock index. This leads to several approaches, depending on the input data and the aim of the prediction (either regression of a price point or classification of a trend). In [6] a pattern recognition model is used to identify bull-flags patterns, a common trading signal in the stock prices technical chart. This implementation comes from two common strategies for pattern recognition, perceptually important point identification matching and template matching.

In [7] Genetic Network Programming was combined with Sarsa Reinforcement learning to give trading signals (buy or sell). The technical candlestick chart and several technical indicators are used to decide whether it is the right time to buy or sell stocks.

In [8] one-minute logarithm return is predicted with a feed-forward Neural Network. The network is fed with the logarithm of the returns, average price and standard deviation from 1-minute interval prices. In addition, timestamps are given to the network to estimate day trends and anomalous behaviours.

Chiang, Enke et al. [9] build an adaptive neural network model that uses Particle Swarm Optimization and a neural network to leverage from the data the direction of the stock index price movement instead of the price itself. Particle Swarm Optimization is useful to help the system overcome the risk of finding a local optimum and converging to a solution in a feasible time.

Patel, Shah et al. [10] proposed a two-stage approach for predicting stock market indexes, whose best one was based on Support Vector Regressor and ANNs to predict technical indicators and, from those, the price after n days.

In [11] a forex trading system was presented, based on the Dempster-Shafter theory and fuzzy logic systems.

Deep learning methods may also be useful to extract information from the stock market through autoencoders and understand how stocks are related to each other

using co-variance estimation. Chong et al. [12] observed that correlation distances among stocks might vary over time, and this knowledge may be fruitful for intraday traders.

There are several papers in which researchers try to leverage stock prediction using news text. For example, Liu [13] use a Recurrent Neural network to encode news text context and an Attention Mechanism to focus the predictor on the most valuable words, news and days.

The effectiveness of evaluating the stock market through news data is also evaluated by Geva et al. [14]. It is shown that improving the quality of news encoding would result in larger profits in intraday prediction. Several methods are tried, from news count to Bag Of Words and sentiment scores.

Neural Networks can also be applied to analyze emerging markets, as Mostafa did in [15]: a General Regression Neural Network was used to predict the Kuwait Stock Exchange closing prices. To reduce the risk of overfitting in single series prediction, Ticknor [16] proposed a Bayesian Regularization Network, tested on several technology indices with daily granularity.

Schumaker and Chen [17] investigated the possibility of including subjectivity analysis in their predictions. They found out that it is possible to have better results, including subjectivity. Moreover, they find out that investors may react stronger on negative feeds rather than on positive.

Lavrenko et al. [18] demonstrated the effectiveness of language models in predicting stock price trends. They used time-stamped news stories and minute-frequency financial time series to associate news to different trends, which means surge, purge, slight surge or slight purge. Das and Chen [19] build an ensemble of classifiers (Naive, Bayesian, and Vector Distance) to extract investor sentiment from message boards.

Mittemayer [20] developed a trading system based on a hand-made thesaurus to categorize news press releases from the American PRNewswire.

Liu proposes a system based on Recurrent Neural Networks to leverage information from news data retrieved from Thomson Reuters to predict the S&P 500 index and some technological indexes individually.

Finally, Ma and Ke [21] develop a Multi-task learning model to predict several time-series using Recurrent Neural Networks with an Attention mechanism based on the idea of Capital Asset Pricing Model (Sharpe 1964), as it is well known from the literature that stock indexes from the same sector have similar trends. To the best of our knowledge, we are not aware of systems that mix the multi-task learning approach with news data.

Chapter 3

Background

3.1 Stock market

A share consists of a security representative of the capital of a company. The shareholder holds certain rights in all of the assets of the issuing company. Shareholders have the right to vote for the appointment of the Board of Administration and other matters. Investors may get a return in two ways:

- the share price rises over time;
- the company pays out dividends.

The return will be negative if the price will be lower at the sale; on the contrary, it will be positive if the price increases.

$$return = \frac{p_1 + D}{p_0} \tag{3.1}$$

where p_1 is an arbitrary price after p_0 and D is the dividend.
Among those who invest money to buy stocks, we can distinguish two types:

- **long term investors**: they keep security shares for long periods, as they are interested in administrative rights inside the company (e.g. voting rights);
- **speculators**: they keep security shares for short periods because they want to realize capital gains as the price increases.

There are two main types of stocks to refer to:

- **common stocks**, in which stakeholders have voting rights and receive variable dividends;

- **preferred stocks**, in which stakeholders usually do not vote and receive fixed dividends.

There are two types of stock markets, depending on the rules applied:

- the **regulated market** is subject to strict rules regards the terms of sale;
- the **over-the-counter market**, also known as OTC, is based just on the matching between supply and demand.

The stock indexes of our research are purchased in the NASDAQ (National Association of Securities Dealers Automated Quotation) stock market.

3.2 Neural networks

Neural network are a class of machine learning algorithm able to generalize a given data distribution with a generic function f. We want to approximate a function $f^*(x)$, with our function $y = f(x, w)$. Learning w parameters will result in the approximation of f towards f^*. Information goes from x, our input data, to our labels y, through several layers, where each one represents an intermediate function. So for example $f(x) = f_{l1}(f_{l2}(f_{l3}(x)))$ where f_{l1}, f_{l2}, f_{l3} might be linear layers in the form of

$$f_l(x) = W_l * x + b_l, \tag{3.2}$$

where $W \in \mathbb{R}^{m,n}$ maps the input from a space m to a space n, and b_l is called bias. The output can then be gated with an activation function, such as ReLU (Rectified Linear Unit)

$$ReLU(x) = max(0, x). \tag{3.3}$$

Functions like 3.3 introduce non-linearities in the general mapping f and they may be used to reproduce non-linearities between input and labels in our problem.

Following this view, *human practitioners can encode their knowledge to help generalization by designing families $\varphi(x; w)$ that they expect will perform well.*(Goodfellow et al., 2016) [22]. More layers we stack upon each other, deeper the network: this is why we refer to this kind of machine learning algorithms as "deep learning". In addition to the linear layers mentioned, we also have to cite the convolutional and recurrent layers: the former is not the object of our dissertation, the latter will be discussed in section 3.4.

The predicted output $\hat{y}_i$ from the model is compared with the true value y_i given by the data with a loss function. An example is the Mean Squared Error function (MSE):

$$MSE(y_i - \hat{y}_i) = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}, \tag{3.4}$$

which computes the distance between the predicted and the true value. Our objective is to minimize this distance regards the parameters w through some gradient-based optimization method, such as:

$$w_{t+1} = w_t + \eta * \nabla_w c, \tag{3.5}$$

where η is the learning rate, a number that modifies $\nabla_w c$, the gradient of the cost function w.r.t the parameters w. To optimize our cost function c, we need to update the parameters w at every layer of our network, and we can do this through the **backpropagation algorithm** [23]. Let us suppose we have computed our predicted values $\hat{y} = f(x, w)$, where $\hat{y} \in \mathbb{R}$ and we have our cost function $J(w)$. We can compute the derivative of J w.r.t. w following the **chain rule**:

$$\frac{dJ}{dw} = \frac{dJ}{d\hat{y}} \frac{d\hat{y}}{dw}. \tag{3.6}$$

We derive $\frac{dJ}{d\hat{y}}$ directly from the formula of the cost, the same stands for $\frac{d\hat{y}}{dw}$. The same applies in case we have vectors instead of scalars, if we have a relation f that maps from $\mathbb{R}^m$ to $\mathbb{R}^n$ and J that maps from $\mathbb{R}^n$ to $\mathbb{R}$, then equation 3.7 becomes

$$\frac{dJ}{dw_i} = \sum_j \frac{dJ}{d\hat{y}_j} \frac{d\hat{y}_j}{dw}. \tag{3.7}$$

3.3 Multi-task learning

Multi-task learning [3] is a concept used in several machine learning problems: we leverage the representation of one single problem requiring the machine learning algorithm, a Neural Network, to learn the representation of other related tasks. The objective function in this way is a combination of several objective functions, each one related to a part of the whole problem. The information from these related tasks is a source of inductive bias, i.e. a set of assumptions that the predictor uses to improve generalization. The key concept to obtain this result is to implement a shared layer among the intermediate outputs of our tasks in our net. This is called hard parameter sharing [4].

In soft parameter sharing, the n tasks are trained in parallel with different models, and the layers are constrained so that the parameters will be similar. Regarding a common problem in machine learning, *if we simultaneously train a net to recognize object outlines, shapes, edges, regions, subregions, textures, reflections, highlights, shadows, text, orientation, size, distance, etc., it may learn better to recognize complex objects in the real world* (Caruana 1997).

3.4 Recurrent neural networks

Recurrent neural networks [22] are a class of specific nets that are able to generalize a relationship for sequences of data, such as words in a sentence or temperature through time. The key idea behind this kind of architecture is the ability to share weights among all the time steps of the sequence $x^{(1)}, ..., x^{(t)}$. Again, one may think about 1-D Convolution as an alternative, but in this case, the shared parameters embed the connection between the data point $x^{(t)}$ and its narrow neighbours. In general, a recurrent function can be written as:

$$h^{(t)} = g^{(t)}(x^{(t)}, ..., x^{(1)}), \tag{3.8}$$

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta). \tag{3.9}$$

Where θ are the learnable parameters and $h^{(t)}$ is the general state at time t. One of the major problems of RNNs is vanishing and exploding gradients. Indeed let us consider the hidden state at time t:

$$h^{(t)} = (W^{(t)})^T h^{(0)} \tag{3.10}$$

If we assume that W is diagonalizable, we can rewrite it as

$$W = Q\lambda Q^T, \tag{3.11}$$

then 3.10 becomes

$$h^{(t)} = Q^T \lambda^t Q h^{(0)}. \tag{3.12}$$

In this last equation, we can notice that the eigenvalues mostly contribute to the final state. Large values tend the computation to explode, while W can vanish with eigenvalues close to zero.

Literature has obviated the problem introducing Gated Recurrent Units (**GRU**), invented by Cho et al. [24]. It works like:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z), \tag{3.13}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r), \tag{3.14}$$

$$\tilde{h}_t = \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h), \tag{3.15}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t. \tag{3.16}$$

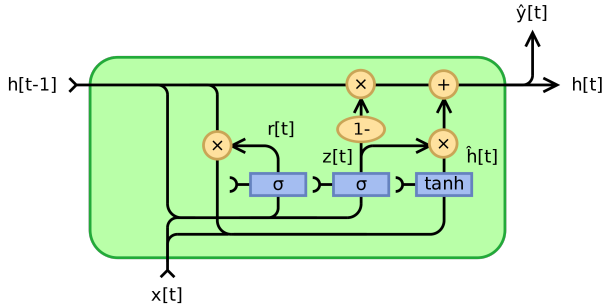


Figure 3.1: GRU graphic diagram (source wikipedia.org)

The network uses a gate z_t (3.13) that decides to update the state in 3.16 or not. An intermediate state $\tilde{h}_t$ (3.15) is computed from the reset gate r_t in 3.14. This architecture reduces the risk of exploding or vanishing gradients. In the image below, a graphical representation of the GRU architecture.

Another Recurrent Unit that obviates the problem of vanishing gradients is the **LSTM** [25] (Long Short Term Memory). Here it is the list of all the formulas involved:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f), \tag{3.17}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i), \tag{3.18}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o), \tag{3.19}$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c), \tag{3.20}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t, \tag{3.21}$$

$$h_t = o_t \circ \sigma_h(c_t), \tag{3.22}$$

The idea behind this algorithm is that the cell state c_t acts as a memory and is modified by the input i_t, the output o_t and the forget gate f_t. h_t is the hidden state vector, and it is the output of the LSTM cell. The σ_g is a sigmoid activation function while the σ_c is an hyperbolic tangent activation function.

3.5 Adam optimization algorithm

Adam is a gradient-based optimization algorithm invented by Diederik Kingma and Jim Ba in 2015 [26]. It belongs to the family of the Stochastic Gradient

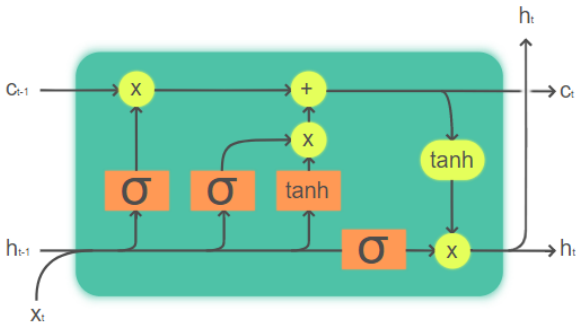


Figure 3.2: LSTM graphic diagram (source wikipedia.org)

algorithms used to train Neural Networks. Still, it has several interesting features that make it more stable and able to converge faster. It derives its functionalities from **AdaGrad** and **RMSprop**. In particular:

- problems with sparse gradients may be leveraged by maintaining a learning rate for each parameter, as **AdaGrad** does;
- non-stationary problems may benefit from the fact that parameters are updated with the use of moving averages, a feature introduced by the **RMSprop** algorithm.

The equations 3.23 - 3.28 show how the parameters are updated during each step within one epoch. In 3.23 the gradient w.r.t, the parameter is computed similarly to the other gradient-based optimization algorithms; in 3.24 and 3.25, moving averages from the gradient and the squared gradient are calculated, with β_1 and β_2 controlling the exponential decay. Then $\hat{m}_t$ and $\hat{h}_t$ (3.26, 3.27) determine the next parameter update in 3.28.

$$g_t = \nabla_\theta f_t(\theta_{t-1}) \tag{3.23}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.24}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.25}$$

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{3.26}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{3.27}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.28}$$

3.6 VADER

In the following section, we briefly describe VADER (for Valence Aware Dictionary for sEntiment Reasoning) [27], a lexicon-based method to retrieve sentiment overall rating from a text. It is already implemented in the Natural Language Toolkit package in Python [28]. A sentiment lexicon is a set of linguistic features labelled by humans as positive or negative. VADER has shown to have excellent results calculating sentiment for social-media content. Its strength consists of leveraging the sentiment's valence by looking at punctualization, capitalization and degree modifiers (such as intensifiers like *extremely* or *marginally*). In the table below an example of the heuristics found to retrieve sentiment intensity:

Condition	Example
Punctuation	Wow. What a beautiful day.
Punctuation1	Wow! What a beautiful day!
Punctuation1 + Intensifier	Wow! What an extremely beautiful day!
Punctuation2	Wow!! What an extremely beautiful day!!
Capitalization	WOW. What a BEAUTIFUL day.
Punct1 + Cap.	WOW! What a BEAUTIFUL day!
Punct2 + Cap.	WOW!! What a BEAUTIFUL day!!
Punct3 + Cap.	WOW!!! What a BEAUTIFUL day!!!
Punct3 + Cap. + Intensifier	WOW!!! What an EXTREMELY BEAUTIFUL day!!!

Amazon Mechanical Turk hired human raters: they were first selected by the English language skills they showed during some tests, a sentiment rating session was held to explain their job. In the end, a monetary reward was given to them for every answer to the sentiment tests.

3.7 Count vectors

There are several techniques in Natural Language Processing to process text data in a form suitable for a machine learning problem. One of these is Count Vectorizing, i.e. the given sentence is transformed in a vector of integers, with vocabulary size as maximum dimensionality (in our case we set a maximum number of features n so that the n most frequent words will be taken into account). Then, for each entry, we substitute the i-th word with the number of word occurrences in the all corpus, in our case the list of news titles.

3.8 Word embeddings

In recent years, research has proposed a different approach to leverage the major semantic features of a text and has come out with a new concept: word embeddings. The aim is to encapsulate a single word in a vector $w \in \mathbb{R}^d$ having similar words in close surroundings of the d-dimensional space. For example, we can extrapolate synonyms of a word looking at a notion of distance, like the cosine similarity:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{3.29}$$

Research has so far produced two types of word embeddings: **Matrix Factorization Methods** and **Shallow Window Methods**. In the first family, matrices are built to get the relations among words in *all* the corpus. Deerwester et al. [29] used a term-document matrix, while Lund and Burgess [30] take a term-term matrix: entries are the count of occurrences a word in the corpus occurs in the context of another word. In the latter, we use local context windows to aid the representation of words. We can either predict a word given its context, as Mikolov et al. do with the Continuous bag-of-words [31] or predict the context given the word itself. From this previous work, Pennington et al. [32] have developed a new approach, called **GloVe**, that stands for Global Vectors. Following empirical evidence, they have found out that word vectors could be established from the *ratio of co-occurrence probabilities*. From here, the model takes the form:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{3.30}$$

The ratio on the right side of the equation is taken from the corpus as $P_{ik} = P(k|i)$ is the probability that word k appears in the context of word k. More in details:

$$P(k|i) = \frac{X_{ik}}{X_i}, \tag{3.31}$$

where X_{ik} equals the number of times word k appears in the context of word i, and X_i counts the times *any word* appears in word i context. We refer the reader to the original paper to find out about the form of F and the final objective function.

3.9 Portfolio metrics

This section briefly describes the portfolio metrics used to evaluate our strategies explained in section 6.5. In general when we talk about ratio, we mean a measure of risk-adjusted return, where return at the numerator are divided to some measure of risk.

3.9.1 Max Drawdown

The maximum drawdown is the largest loss realized by the strategy over a period T, which is usually 36 months.

$$MDD = \max \left[\max(\sum_{t=0}^{T} R_t) - R_T \right] \tag{3.32}$$

3.9.2 Calmar ratio

With the drawdown depicted in 3.9.1 we can compute the CALMAR (California Managed Account Ratio), introduced by Young in 1991 as:

$$CALMAR = \frac{Return\ over\ T\ periods}{MDD\ over\ T\ periods}, \tag{3.33}$$

where MDD is defined in 3.32. The higher ratio, better the strategy provided. Although it is a simple and understandable metric, it ignores the volatility as a risk measure. The risk is just a single event in the portfolio history.

3.9.3 Sharpe ratio

One of the top-five metrics is defined as follows:

$$Sharpe = \frac{R_p - R_F}{\sigma(R_p)}, \tag{3.34}$$

where R_p is the annualized period return over period T, R_F is the risk-free return over a period T, $\sigma(R_p)$ is the standard deviation of the portfolio returns over a period. The volatility on the denominator counts both systematic and unsystematic risk, so if the portfolio is not completely diversified, that is the right metric to look on; volatility on the contrary does not weight the trend: investors are more interested in downward price movements and Sharpe does not help in identifying those.

3.9.4 Stability

As the pyfolio documentation states, stability is *R-squared of a linear fit to the cumulative log returns*. Once computed the least squares linear fit, R^2 tells us to which extent the variation of the cumulative log return can explain the line slope.

3.9.5 Omega ratio

Keating and Shadwick invented the formula that follows:

$$Omega = \frac{\int_{r_d}^{b} 1 - F(r)f(r)dr}{\int_{a}^{r_d} F(r)f(r)dr}. \tag{3.35}$$

On the numerator we have the cumulative probability of an investment outcome above a certain threshold level r_d. On the denominator we consider the cumulative outcome below a threshold level r_d. Unlike the Sharpe, Omega does not depend on the assumption that the distribuion is Gaussian. It may take also investor thresholds as preferences.

3.9.6 Sortino ratio

Inspired by the Sharpe ratio, Sortino and Van Der Meer defined a new risk-adjusted metric as follows:

$$Sortino = \frac{R_p - T_A}{DR}, \tag{3.36}$$

where R_p is the return over a period T, T_A is the target rate of return to take into consideration and DR measures the variability of returns below the minimum target rate.

$$DR = \Big(\sum_{t=1}^{T} \frac{\min[R_t^p - TA_t, 0]^2}{T-1} \Big)^{1/2} \tag{3.37}$$

3.9.7 Skew, kurtosis and tail ratio

We can refer to skew as a measure of distortion in the classical bell curve of a normal distribution. Skewness can be positive, in which mean of data is greater than the median, otherwise negative. Positive skewness in an asset return distribution means that there are more gains than losses. Here is the measure of skewness:

$$Skew = \frac{3 * \overline{X} - Md}{\sigma}, \tag{3.38}$$

where in our case $\overline{X}$ is the mean of the asset return, Md is the median, σ is the standard deviation. Skewness is taken into consideration with the kurtosis, that measures the combined weight of a distribution tails relative to the center of the distribution. In this context, tail ratio is defined as the 95th divided by the 5th percentile of daily returns distribution.

3.9.8 Daily value at risk

With this metric we mean the downside deviation below the threshold zero relative to the daily return distribution. Investors are more concerned about the risk of losing money so that volatility is not enough reliable to evaluate a strategy.

3.9.9 Alpha and beta

Alpha is referred as *excess returns over a market benchmark*, such as the NASDAQ index in our case. It is the ability of our strategy *to beat the market*. It is defined as a difference between the portfolio return and the benchmark return. Beta is indeed a measure of correlation between the portfolio daily return and the benchmark we are considering.

Chapter 4

Model

The following chapter describes in detail the proposed model that incorporates multi-task and single-task learning approach, as seen in [21] and the information retrieved from news, processed in three different ways, VADER (3.6), Count Vectors (3.7) and GloVe (3.8).

4.1 Problem Statement and Objective Function

Let us define our single-task prediction as:

$$E(p_{t+\Delta}|\mathbf{x}_{t-i*\Delta}, i = 1, ..., N) = g(\mathbf{x}_{t-1*\Delta}, ..., \mathbf{x}_{t-N*\Delta}), \tag{4.1}$$

where the expected value $p_{t+\Delta}$ may be the closing price in a 20-minute window, as Gidofalvi et al. have found to be a predictable time span [33]; we are also trying different timespans, 30 and 120 minutes. The input x_{t-i} is a multivariate time series of length N, where each open price is associated with multiple news features, preprocessed with three alternative procedures, as described above; we are also integrating these with technical analysis features used by Fior and Cagliero [34].

$$\mathbf{x}_t = (x_{1*\Delta}, ..., x_{m*\Delta})^T \tag{4.2}$$

g is the approximate function of the neural network, whose details are going to be described below. In our multi-task learning scenario, the network takes 102 stocks from the NASDAQ index as input:

$$E\left(\begin{array}{c|c} p^1_{t+\Delta} & \{\mathbf{x}^1_{t-i*\Delta}\}_{i=1}^N \\ \vdots & \vdots \\ p^k_{t+\Delta} & \{\mathbf{x}^k_{t-i*\Delta}\}_{i=1}^N \end{array}\right) = g\left(\begin{array}{c} \{\mathbf{x}^1_{t-i*\Delta}\}_{i=1}^N \\ \vdots \\ \{\mathbf{x}^k_{t-i*\Delta}\}_{i=1}^N \end{array}\right). \tag{4.3}$$

The 102 stocks taken in exam are fed into RNN units separately, as shown below:

$$h_k = RNN(\{x^k_{t-i*\Delta}\}^N_{i=1}), k = 1, ..., M, \tag{4.4}$$

the h_k output is the base for a linear layer, shared among the four tasks, that outputs the final predicted price:

$$\hat{p}^k_{t+\Delta} = \delta(\sum_{i=1}^{k} w_{ik} h_i). \tag{4.5}$$

The hyperbolic tangent activation function shapes the outputs in the interval [−1,1]. The closing prices, after the linear layer, are compared to the target prices to compute the loss with the Mean Squared Error formula.

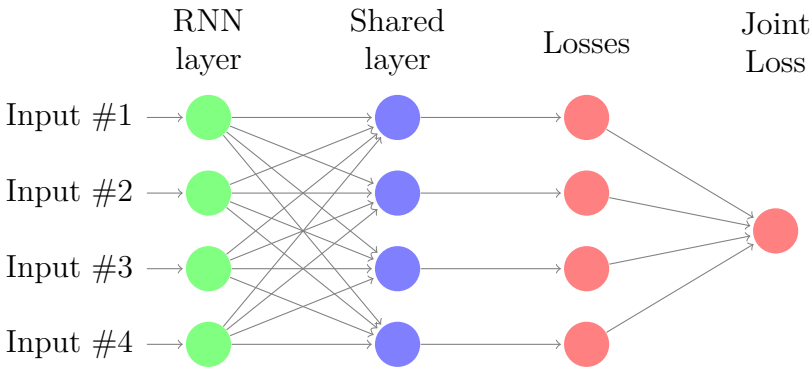
$$L_k = MSE(\hat{p}^k_{t+\Delta}, p^k_{t+\Delta}) \tag{4.6}$$

$$MSE(y_i, \hat{y}_i) = \frac{\sum^N_{i=1}(y_i - \hat{y}_i)^2}{N} \tag{4.7}$$

Finally, the joint loss is computed as the mean of the 102 task-related losses.

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{4.8}$$

Here is an explanatory chart with a reduced number of inputs to better explain the process.



Given the network, we can finally define the hypothesis space as the following:

$$\mathcal{H}^{(k,n,m)} = \{h^{(w)} : \mathbb{R}^{k,n,m} \to \mathbb{R} : h^{(w)} = g^{(w)}(\mathbf{x})\} \tag{4.9}$$

4.2 Classification and single-task mode

We also explore the possibility to use the same network in classification mode, it will be held in all sets of experiments. The network will output three logits instead of a continued price value. In this case, we compute the Categorical Crossentropy as loss function.

$$CCE(y_i, \hat{y}_i) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij})) \tag{4.10}$$

We also try to exploit differences between the single-task learning approach against the multi-task one because we want to determine whether one has better results. The experiments will explore combinations of the four modes (multi-task or single-task, VADER, count vectorizing or GloVe embeddings, regression or classification, 20, 30 or 120 minute granularity). Each experiment section (6.1, 6.2, 6.4) will describe in detail which features are included.

Chapter 5

Data collection and preprocessing

5.1 Financial data preprocessing

We briefly describe the financial and news data, that we have retrieved using the AlphaVantage API. The model was built in Python, using the Tensorflow 2.2 framework. In table 5.1 there are the major statistics of the first dataset, which consists of four stocks, Apple, Amazon, Google, Microsoft. This dataset was used for the first experiment in subsection 6.1.

Days	Initial timestamp	Final timestamp	Datapoints
359	2018-03-06 09:30:00	2019-02-28 16:00:00	517810

Table 5.1: Description of the first small financial dataset from Alpha Vantage

Stock	Number of headlines
AAPL	17881
AMZN	12655
GOOGL	7943
MSFT	8404

Table 5.2: Number of headlines per stock in the first dataset

We then describe in table 5.3 the second broader dataset, used in the experiments detailed in subsections 6.2, 6.3, 6.4, 6.5.

News headlines and stock prices about the 103 companies and the NASDAQ

Days	Initial timestamp	Final timestamp	Datapoints
359	2018-10-24 09:30:00	2019-10-23 16:00:00	524551

Table 5.3: Description of financial data from Alpha Vantage

index were also retrieved from AlphaVantage. The news are just in english. Timepoints that are not present in this range are linearly interpolated during the preprocessing. Stock data from FOX and FOXA were corrupted so not taken into consideration for the experiments.

Regarding neural network input data, it is very important that the prices will be normalized in a range between 0 and 1, since a neural network tends to find a point of minima faster: the error surface is smoother and gradient descent algorithms work better. In our model we choose to use the `MinMaxScaler` function from the library `scikit-learn` [35], which mathematical formula is displayed below:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}. \tag{5.1}$$

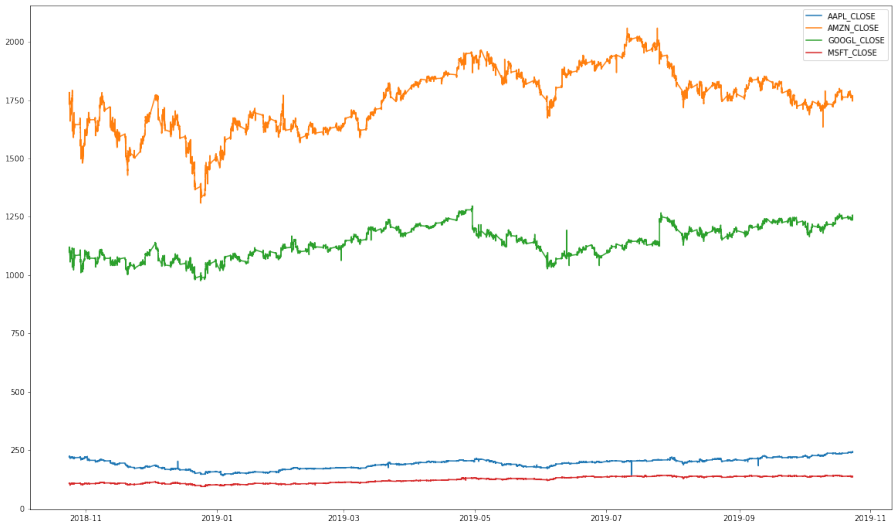


Figure 5.1: Close stock prices for the all time span (24/10/2018 - 23/10/2019).

Figure 5.2: Normalized close stock prices for the all time span (24/10/2018 - 23/10/2019).

5.2 Technical indicators

In this subsection we are going to describe the technical indicators used as additional features to the open, close, high and low price and the volume. We are going to discuss later whether these are improving the model in predicting the price labels. We divide these in groups:

- **momentum indicators** are meant to identify strength in price movements;
- **volume indicators** are based on volume data, that is the amount of shares exchanged and help the trader understand the interest of the market in a given stock;
- **trend indicators** help the trader understand if the market is either bullish, that is prices are rising due to the prevalence of buy over sell orders, or bearish, where prices are decreasing;
- **volatility indicators** indicate how much a stock is distant from its mean price. Higher the volatility higher, given a time period, the expected returns.

5.2.1 Percentage price oscillator

Percentage Price Oscillator is a momentum indicator that relates two exponential moving averages with 12 and 26 periods p. The exponential moving average weights the members of the last p periods with a function that decreases exponentially, so that last closing price data points are more discounted; it reacts faster to changes than the Simple Moving Average. PPO is defined as:

$$PPO_t = \frac{EMA_{12}(close) - EMA_{26}(close)}{EMA_{26}(close)} * 100, \tag{5.2}$$

where the exponential moving average is defined as follows:

$$EMA_{t,p} = \begin{cases} Y_t & t = 1 \\ \alpha * Y_t + (1 - \alpha) * EMA_{t-1,p} & t > 1 \\ \alpha = \frac{2}{p+1} \end{cases} \tag{5.3}$$

PPO is generally used with its $signalline = EMA_9(PPO)$ to spot buy or sell opportunities: buy signal when the PPO crosses its signal upward, sell signal viceversa. PPO itself above 0 may confirm an upward trend, while below 0 may indicate a downward trend.

5.2.2 Percentage volume oscillator

PVO is the PPO counterpart for volume data and is defined as:

$$PVO_t = \frac{EMA_{t,12}(Volume) - EMA_{t,26}(Volume)}{EMA_{t,26}(Volume)} * 100. \tag{5.4}$$

It is also used to ensure the current trend.

5.2.3 True strength index

Another momentum indicator is the True Strength Index, useful to determine whether the asset is overbought or oversold. Trend reversal may be highlighted by the divergence with the price chart. It is defined as follows:

$$TSI = 100 * \frac{DoubleSmoothedPriceChange}{DoubleSmoothedAbsolutePriceChange}, \tag{5.5}$$

$$DoubleSmoothedPriceChange = EMA_{13}[EMA_{25}(PC)], \tag{5.6}$$

$$DoubleSmoothedAbsolutePriceChange = EMA_{13}[EMA_{25}(|PC|)], \tag{5.7}$$

$$PC = ClosePrice_t - ClosePrice_{t-1}. \tag{5.8}$$

A trader might note trading signals with centerline crossovers: TSI above 0 suggests a buy, a sell viceversa. Signal line referred to this oscillator is usually a 7-to-12 EMA: long or short position are signaled respectively with an upward or downward crossing.

5.2.4 Relative strength index

This is a very popular oscillator invented by J. Welles Wilder [36]. It swings from 0 to 100 and it announces overbought shares over 70 and oversold under 30. Its calculation may be resumed as:

$$RSI = 100 - \frac{1}{RS + 1}, \tag{5.9}$$

$$RS = \frac{AverageGain}{AverageLoss}, \tag{5.10}$$

$$AverageGain = \frac{1}{14} \sum_{i=0}^{14} \max(PC_{t-i}, 0), \tag{5.11}$$

$$AverageLoss = \frac{1}{14} \sum_{i=0}^{14} |\min(PC_{t-i}, 0)|, \tag{5.12}$$

where PC_t is defined in 5.8. Important to notice that losses are taken as absolute values in the formula 5.12.

5.2.5 Bollinger Bands

This technical indicator will be used in the trading system (see algorithms 5 and 7) to further filter trading signals generated by the neural network. John Bollinger designed a method to spot oversold and overbought opportunities looking at closing price simple moving average. Then generally two standard deviations are added and subtracted to get upper and lower bands: traders get signals if the price either breakouts over the upper or below the lower. Here we show the formulas:

$$upperBand = SMA_{20}(close) + 2 * \sigma_{20}(close), \tag{5.13}$$

$$upperBand = SMA_{20}(close) - 2 * \sigma_{20}(close). \tag{5.14}$$

.

5.2.6 Chande momentum oscillator

Instead of considering the average of gains and losses, Tushar Chande developed an indicator using the sum over a period of 20. Due to the 100 multiplier, it ranges from -100 to +100, where oversold is below -50 and overbought over 50. The divergence among high peaks and this indicator may announce a trend reversal:

$$CMO = \frac{sumOfGains - sumOfLosses}{sumOfGains + sumOfLosses} * 100, \tag{5.15}$$

$$sumOfGains = \sum_{t=0}^{14} \max(PC_t, 0), \tag{5.16}$$

$$sumOfLosses = \sum_{t=0}^{14} |\min(PC_t, 0)|, \tag{5.17}$$

where PC_t is defined as 5.8

5.2.7 Stochastic oscillator

In the 50s George C. Lane introduced a momentum indicator that relates the position of the close to he high-low range in a given period. Here are the formulas:

$$\%D = SMA_3(\%K), \tag{5.18}$$

$$\%K = \frac{ClosePrice_t - L_{14}}{H_{14} - L_{14}}, \tag{5.19}$$

$$L_{14} = \begin{cases} low_t & t = 1 \\ \min(low_t, \min_{t-1}) & 2 \leq 14 \end{cases}, \tag{5.20}$$

$$H_{14} = \begin{cases} high_t & t = 1 \\ \max(high_t, \max_{t-1}) & 2 \leq 14 \end{cases}, \tag{5.21}$$

where $\%D$ is slower than the $\%K$ and acts as a signal for the latter. The equations 5.20 and 5.21 find the lowest and the highest price in the period considered.

5.2.8 Money flow index

Similar to the RSI in 5.2.4, Money Flow Index takes into account also the volume data. Traders rely on that to find trend reversals when the chart diverges with price. Combined with RSI, it can confirm price signals.

$$MFI = 100 - \frac{1}{1 - MFR}. \tag{5.22}$$

$$MFR = \frac{mfH_{14}}{mfL_{14}}. \tag{5.23}$$

$$mfH_{14} = \sum_{i=0}^{14} \max(TypicalPrice_{t-i} * Volume_{t-i}). \tag{5.24}$$

$$mfL_{14} = \sum_{i=0}^{14} |\min(TypicalPrice_{t-i} * Volume_{t-i})|. \tag{5.25}$$

$$TypicalPrice_t = \frac{close_t + high_t + low_t}{3}. \tag{5.26}$$

5.2.9 On balance volume

In 1963 Joseph Granville invented OBV to leverage the information sentiment lead by volume data [37]. He noticed that large investors like pension funds increase the trade volume before the price springs upward or downward, a precursor highlighted by this formula:

$$OBV_t = OBV_{t-1} + \begin{cases} Volume & if\ close_t > close_{t-1} \\ 0 & if\ close_t = close_{t-1} \\ -Volume & if\ close_t > close_{t-1} \end{cases}. \tag{5.27}$$

5.2.10 Accumulation distribution index

This volume-momentum indicator was developed by Marc Chaikin in 80s, it is more sophisticated than the OBV as it relates the close price in the high-low range. It is computed as:

$$A/D_T = \sum_{t=0}^{T} MoneyFlow_t * Volume_t, \tag{5.28}$$

$$MoneyFlow_t = \frac{(close_t - low_t)(high_t - close_t)}{high_t - low_t}. \tag{5.29}$$

Understanding the meaning of $MoneyFlow_t$ is fundamental, as it reaches +1 when the close price is near the high, and consequently there is a buy pressure, conversely it is -1 when close is equal to the low. $MoneyFlow_t$ is in fact a weight for the volume.

5.2.11 Force index

Invented by [38] is a function of price and volume as the latter two. It is unbounded as it resumes as follows:

$$FI = EMA_{13}(FI_t), \tag{5.30}$$

$$FI_t = (close_t - close_{t-1}) * volume. \tag{5.31}$$

Like the MFI it helps assess the strength of a trend and spot potential price reversals.

5.2.12 Moving average convergence divergence

This is one of the most popular indicators used to identify if we are in a bullish or bearish trend. Usually traders compare it to 0.

$$MACD_t = EMA_{t,12}(close) - EMA_{t,26}(close). \tag{5.32}$$

5.2.13 Aroon indicator

Tushar Chande aimed at extracting information from strong uptrends and downtrends measuring the time from the last high or the last low, as follows:

$$Aroon = AroonUp - AroonDown, \tag{5.33}$$

$$AroonUp = 100 * \frac{25 - Periods\ since\ the\ highest}{25}, \tag{5.34}$$

$$AroonDown = 100 * \frac{25 - Periods\ since\ the\ last\ lowest}{25}. \tag{5.35}$$

5.2.14 Average true range percentage

ATRP measures the volatility of the stock over a range of periods, it can be used both for intraday and for interday data. It is not directional as it might indicate a buying or selling pressure. Traders often use it to signal stop loss.

$$ATRP = \frac{ATR}{close} * 100, \tag{5.36}$$

$$ATR = \frac{\sum_{i=1}^{n} TR_i}{n}, \tag{5.37}$$

$$TR_i = \max(high - low, |high - close|, |low - close|). \tag{5.38}$$

5.2.15 Average directional index

ADX is one of the most important trend indicator, it ranges from 0 to 100 and it qualifies the trend strength. It means weak trend below 20 and strong trend above 30. Trading strategies vary depending on this, as we will show in our trading system. Usually a low ADX allows to trade with resistance and support level of price, waiting for a breakout at ADX above 30, where trade on trend happen. Chartists look at ADX in relation to $+DI_t$ and $-DI_t$

$$ADX_t = \frac{DX_{t-1} * 13 + DX_t}{14}, \tag{5.39}$$

$$DX_t = \frac{|+DI_t - -DI_t|}{|+DI_t + -DI_t|} * 100, \tag{5.40}$$

$$+DI_t = \frac{+SDM}{ATR} * 100, \tag{5.41}$$

$$-DI_t = \frac{-SDM}{ATR} * 100, \tag{5.42}$$

$$+/-SDM = \sum_{i=1}^{14} DM_{t-i} + 1/14 * \sum_{i=1}^{14} DM_{t-i} + DM_t, \tag{5.43}$$

$$+/-DM = \begin{cases} \max(high_t - high_{t-1}, 0) & high_t - high_{t-1} > low_{t-1} - low_t \\ \max(low_{t-1} - low_t, 0) & high_t - high_{t-1} \leq low_{t-1} - low_t \end{cases}. \tag{5.44}$$

5.3 News data preprocessing

versionCreated	text
2019-09-01 23:17:02.549000+00:00	Dow Jones Selected Stocks - September 02
2019-09-01 11:30:17.771000+00:00	Apple Tag: Here's everything you need to know ...
2019-09-01 10:47:08.437000+00:00	Apple apologizes for use of contractors to eav...
2019-09-01 10:15:22.600000+00:00	Apple and Samsung sued over phones' radiofrequ...
2019-09-01 10:15:16.477000+00:00	Google experts say websites infected iPhones w...

Figure 5.3: Example of raw news headlines for the AAPL stock.

The table 5.3 shows how many title are listed per stock and how these are spaced in time. We can notice that just a few companies have a consistent amount of datapoint in the time interval considered. The first step about news data is organizing the headlines, so that the data frame is evenly spaced in time. We noticed that the dataframe has some news occurring in the same minute. Applying the `round_time` function will move the points to the same minute. Here it is an example of the function behaviour:

2018-10-25 10:42:05 -> 2018-10-25 10:43:00
2018-10-25 10:42:37 -> 2018-10-25 10:43:00

In case the headlines overlap, a special field `cumcount` in the table marks the number of news headlines in the minute.

round_time	text	CUMCOUNT
2018-03-06 13:00:02	Infinite Peripherals Improves Patient Care with the Infinea X for Apple iPhone	1
2018-03-06 13:20:52	APPLE REPORTEDLY WORKING ON NEW PREMIUM OVER-THE-EAR HEADPHONES WITH ACTIVE NOISE CANCELLATION	1
2018-03-06 15:27:16	Apple targets high end headphones	1
2018-03-06 16:30:57	Review: Bose SoundSport Free	1
2018-03-06 17:26:06	FANG names open strong at the open	1

Figure 5.4: News headlines with rounded time and cumulative minute count.

The `cumcount` field is used for the table pivoting. The result will be a new table with a unique `DateTimeIndex` and several columns with eventual n columns filled, if n headlines occured in that minute.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	Infinite Peripherals Improves Patient Care with the Infinea X for Apple iPhone																			
1																				
2																				
3																				
4																				

Figure 5.5: News headlines after pivoting.

Once the pivoting process ends, we treat news headlines in different ways depending on the three different approaches that we explored, VADER (3.6), Count Vectors (3.7) and GloVe (3.8). In the figures 5.6, 5.7 and 5.8 we do not represent additional technical analysis features as described in sections 6.4 and 5.2. The VADER method outputs a number in a continuous range $[-1,1]$, where 1 stands for positive, -1 for negative and 0 for neutral. Since we fill the empty headlines with zeros, we have also to provide a boolean to ensure if the news is present or not. In the image below there is an example of the dataframe after the VADER function.

	0	1	2	3	4	5	6	7	8	9	...	23	24	25	26	27	28	29	30	31	32
2018-03-06 15:20:00	0.390300	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:21:00	0.389917	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:22:00	0.389205	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:23:00	0.389534	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:24:00	0.388189	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:25:00	0.390190	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:26:00	0.390081	0.2023	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:27:00	0.390628	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:28:00	0.390738	-0.0644	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:29:00	0.389972	0.0000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 5.6: Dataframe after VADER preprocessing.

In the count-vectorizing preprocessing procedure, words with a positive and negative impact are kept according to the list provided by each sentence is replaced by the 50-length vector of word occurrences. Multiple news occurring at the same minute are eventually aligned. In the table below the 50 most used words for each company set of headlines:

	0	1	2	3	4	5	6	7	8	9	...	491	492	493	494	495	496	497	498	499	500
2018-03-06 09:30:00	0.298520	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:31:00	0.298520	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:32:00	0.299113	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:33:00	0.299064	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:34:00	0.299014	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:35:00	0.298965	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:36:00	0.298965	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:37:00	0.298965	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:38:00	0.299079	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 09:39:00	0.299194	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 5.7: Dataframe after count-vectorizing.

Finally, if we follow the third choice, the GloVe word embeddings, then we substitute *each word* by the corresponding 50-length vector. The headlines have different number of words, so we take an average m of these and we change the first m words in the phrase.

	0	1	2	3	4	5	6	7	8	9	...	5991	5992	5993	5994	5995	5996	5997	5998	5999	6000
2018-03-06 15:20:00	0.310813	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:21:00	0.309385	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:22:00	0.310908	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:23:00	0.309654	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:24:00	0.309910	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:25:00	0.310958	0.16719	0.073081	0.77842	0.66745	-0.30701	-1.01440	-0.28204	-0.14262	-0.081355	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:26:00	0.312471	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:27:00	0.312121	0.56819	-0.956090	0.51209	1.20990	-0.41932	-0.95927	-1.63950	-0.54107	1.992700	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:28:00	0.312310	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-06 15:29:00	0.311625	0.00000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 5.8: News dataframe substituted with the GloVe word embeddings.

Stock	News	TimedeltaMean	TimedeltaStd
0	NASDAQ	364	1 days 00:03:23
1	AAL	1914	0 days 04:34:39
2	AAPL	13518	0 days 00:38:51
3	ADBE	706	0 days 12:21:48
4	ADI	593	0 days 14:46:59
5	ADP	598	0 days 14:38:17
6	ADSK	637	0 days 13:38:44
7	ALGN	645	0 days 13:34:52
8	ALXN	642	0 days 13:38:57
9	AMAT	697	0 days 12:20:58
10	AMD	1523	0 days 05:44:59
11	AMGN	1482	0 days 05:54:22
12	AMZN	16675	0 days 00:31:31
13	ASML	590	0 days 14:49:33
14	ATVI	993	0 days 08:48:53
15	AVGO	1217	0 days 07:11:59
16	BIDU	1141	0 days 07:40:00
17	BIIB	1206	0 days 07:15:51
18	BKNG	628	0 days 13:56:54
19	BMRN	424	0 days 20:36:06
20	CDNS	580	0 days 15:06:19
21	CELG	1429	0 days 06:06:09
22	CERN	407	0 days 21:32:24
23	CHKP	498	0 days 17:29:36
24	CHTR	696	0 days 12:33:13
25	CMCSA	2253	0 days 03:53:04
26	COST	1364	0 days 06:24:38
27	CSCO	3186	0 days 02:44:46
28	CSX	686	0 days 12:44:44
29	CTAS	336	1 days 01:57:04
30	CTRP	468	0 days 18:39:12
31	CTSH	1278	0 days 06:50:39
32	CTXS	692	0 days 12:37:36
33	DLTR	670	0 days 13:01:40
34	EA	896	0 days 09:44:04
35	EBAY	1295	0 days 06:45:34
36	EXPE	860	0 days 10:10:16
37	FAST	310	1 days 04:17:09
38	FB	15516	0 days 00:33:52

Stock	News	TimedeltaMean	TimedeltaStd
39	FISV	736	0 days 11:51:05
40	FOX	1008	0 days 08:41:19
41	GILD	1557	0 days 05:36:53
42	GOOG	8240	0 days 01:03:46
43	GOOGL	8239	0 days 01:03:47
44	HAS	740	0 days 11:49:01
45	HSIC	561	0 days 15:21:05
46	IDXX	287	1 days 06:28:26
47	ILMN	613	0 days 14:15:31
48	INCY	492	0 days 17:46:45
49	INTC	4472	0 days 01:57:32
50	INTU	462	0 days 18:51:19
51	ISRG	384	0 days 22:51:05
52	JBHT	421	0 days 20:47:55
53	JD	884	0 days 09:51:20
54	KHC	1680	0 days 05:12:33
55	KLAC	464	0 days 18:53:03
56	LBTYA	1068	0 days 08:11:04
57	LBTYK	1069	0 days 08:10:36
58	LRCX	623	0 days 13:59:44
59	LULU	898	0 days 09:45:32
60	MAR	2183	0 days 04:00:36
61	MCHP	625	0 days 14:00:22
62	MDLZ	1172	0 days 07:28:07
63	MELI	243	1 days 12:01:14
64	MNST	401	0 days 21:32:52
65	MSFT	8839	0 days 00:59:28
66	MU	1556	0 days 05:37:26
67	MXIM	402	0 days 21:44:15
68	MYL	942	0 days 09:10:09
69	NFLX	7438	0 days 01:10:39
70	NTAP	777	0 days 11:15:17
71	NTES	385	0 days 22:44:04
72	NVDA	2292	0 days 03:49:22
73	NXPI	716	0 days 12:13:29
74	ORLY	356	1 days 00:36:55
75	PAYX	360	1 days 00:17:07
76	PCAR	549	0 days 15:57:39
77	PEP	1561	0 days 05:35:28

Stock	News	TimedeltaMean	TimedeltaStd
78	PYPL	1369	0 days 06:23:44
79	QCOM	2683	0 days 03:15:50
80	REGN	723	0 days 12:01:37
81	ROST	485	0 days 17:19:56
82	SBUX	2150	0 days 04:04:19
83	SIRI	540	0 days 16:12:48
84	SNPS	728	0 days 12:01:29
85	SWKS	589	0 days 14:50:14
86	SYMC	927	0 days 09:24:17
87	TMUS	1760	0 days 04:58:06
88	TSLA	5922	0 days 01:28:46
89	TTWO	522	0 days 16:43:04
90	TXN	1052	0 days 08:19:59
91	UAL	2177	0 days 04:01:22
92	ULTA	428	0 days 20:27:53
93	VRSK	935	0 days 09:18:59
94	VRSN	217	1 days 16:13:28
95	VRTX	1369	0 days 06:23:34
96	WBA	1087	0 days 08:02:40
97	WDAY	588	0 days 14:55:16
98	WDC	950	0 days 09:12:32
99	WLTW	748	0 days 11:41:19
100	WYNN	1065	0 days 08:12:50
101	XEL	547	0 days 15:57:26
102	XLNX	851	0 days 10:17:16

Chapter 6

Experiments

This chapter will describe in detail all the machine learning experiments held to predict future prices. Computational resources were provided by HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)

6.1 First experiments with different news preprocessing

6.1.1 Training description

Following the hyperparameter choice in 6.1, the training consists of evaluating the model performance and optimizing the model parameters. Finally, we feed the network with **random sequences of training data**. This implementation improves the chances of generalization, as for each epoch, a different representation of the whole dataset is given. In the code snippet below we can inspect the implementation of this concept.

Algorithm 1 Training-validation feeding.

```
for each sample in the batch do
    choose an index in the x
    assign 60 1-minute-frequency data points to x
    assign 1 data point, 20 minutes after the last x point, to y
end for
```

Learning rate	0,001
Optimizer	Adam
Train - validation - test split	64% / 16% / 20%
Batch size	8
Epochs	10
Number of steps	1000

Table 6.1: Choice of hyperparameters

6.1.2 Test description

Once the training is completed, we need to evaluate the performances on the test set, on which the network has not evaluated any loss function: the data points are new for the model. Since for each sequence we evaluate just the next price in 20 minutes, during the test we slide a window of 60 data points through the all test set. In the snippets below you can find the details.

Algorithm 2 Test feeding.

```
for each base_index do
  base = base_index * batch_size
  for each j in batch_size do
    assign 60 data points to x
    assign 1 data point to y
  end for
end for
```

6.1.3 Grid description

We have explored our model to work in different modalities to determine whether using text in a multi-task learning approach is useful. We use the first dataset consisting of 4 stocks (Apple, Amazon, Google, Microsoft) to choose an approach based on one preprocessing method, and test the latter with a more extensive dataset, consisting of all stocks from Nasdaq. In particular, our model differs from:

- multi-task or single-task learning approach;
- the preprocessing made on the news text, either VADER, Count Vectorizing, GloVe or without news text;
- the neural network's output type, either a continuous value (regression) or a class label (classification) indicating the price rise or fall.

6.1.4 Results

Given the combination of those three characteristics, we have gathered training and validation loss curves to decide which preprocessing method was the best and we have chosen VADER: if we look at the training-validation loss plots we can observe that the most stable curves are the ones given by the VADER preprocessing, where a continuous number in a range $[-1, 1]$ is substituted to the headline. Considering the Count Vectorizing and GloVe simulations, we had gradient explosion problems, that caused us to apply gradient clipping in the neural network itself. We can see this effect both in multi-task (6.2, 6.3, 6.6, 6.7) and single task, (6.13-6.16, 6.17-6.20, 6.29-6.32, 6.33-6.36) If we also consider the directional accuracy shown in table 6.3, we can notice that VADER and NoNews perform better, especially with Apple and Google. If we extend the simulation with a broader dataset, we might expect more numerical problems, as the number of parameters increases. The overfitting curves also testify this insight in these two implementations.

		Loss							
		Training				Validation			
Classification		AAPL	AMZN	GOOGL	MSFT	AAPL	AMZN	GOOGL	MSFT
Multi	VADER	0,75				0,81			
	CountVec	0,73				1,07			
	GloVe	0,63				1,11			
	NoNews	0,74				0,82			
Single	VADER	0,76	0,73	0,72	0,78	0,85	0,72	0,84	0,82
	CountVec	0,74	0,66	0,70	0,79	1,73	0,93	0,77	0,85
	GloVe	0,69	0,62	0,69	0,78	1,47	0,94	0,77	0,83
	NoNews	0,76	0,72	0,73	0,80	0,85	0,72	0,85	0,83
Regression									
Multi	VADER	8,44E-05				9,04E-03			
	CountVec	2,16E-04				2,00E-02			
	GloVe	4,01E-04				4,40E-03			
	NoNews	4,50E-05				8,70E-03			
Single	VADER	4,30E-04	4,90E-04	3,20E-04	1,10E-04	3,70E-02	3,20E-02	1,80E-02	3,80E-02
	CountVec	6,40E-04	2,00E-03	4,70E-03	8,57E-05	1,00E-02	6,00E-02	4,70E-02	8,90E-02
	GloVe	4,90E-05	5,30E-05	4,10E-05	7,30E-05	1,50E-03	2,00E-02	2,00E-02	1,00E-02
	NoNews	1,80E-05	1,16E-04	4,70E-04	6,90E-05	1,00E-02	2,50E-02	2,00E-02	2,60E-02

Table 6.2: Training-validation loss. In the first vertical half we can see the classification metric (categorical cross-entropy), while in the second half the regression loss (mean-squared error). The multi-task value is a mean of the four losses.

		Accuracy			
		Test			
Classification		AAPL	AMZN	GOOGL	MSFT
Multi	VADER	0,59	0,51	0,49	0,45
	CountVec	0,47	0,48	0,49	0,49
	GloVe	0,51	0,48	0,49	0,49
	NoNews	0,44	0,48	0,52	0,49
Single	VADER	0,56	0,51	0,49	0,49
	CountVec	0,43	0,48	0,49	0,48
	GloVe	0,43	0,48	0,50	0,49
	NoNews	0,56	0,51	0,49	0,49
Regression					
Multi	VADER	0,47	0,52	0,44	0,42
	CountVec	0,48	0,55	0,51	0,51
	GloVe	0,44	0,49	0,50	0,51
	NoNews	0,51	0,46	0,30	0,41
Single	VADER	0,53	0,52	0,60	0,40
	CountVec	0,48	0,51	0,53	0,52
	GloVe	0,52	0,53	0,52	0,51
	NoNews	0,44	0,54	0,65	0,39

Table 6.3: Test accuracy for the first set of experiments. In the first vertical half we can see the classification experiments, while in the second half the regression experiments.

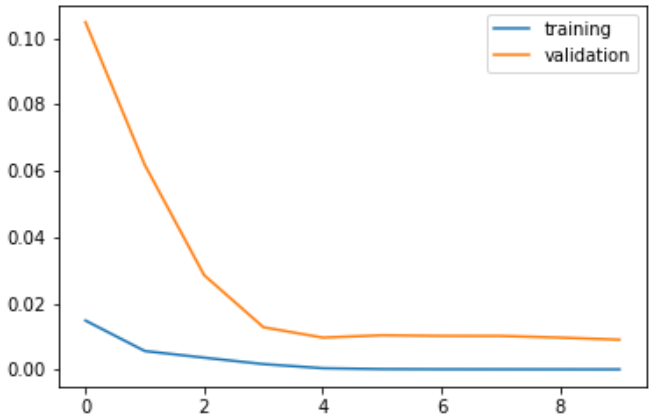


Figure 6.1: Loss plot for neural network trained in **regression multi-task** learning mode, **with Vader**.

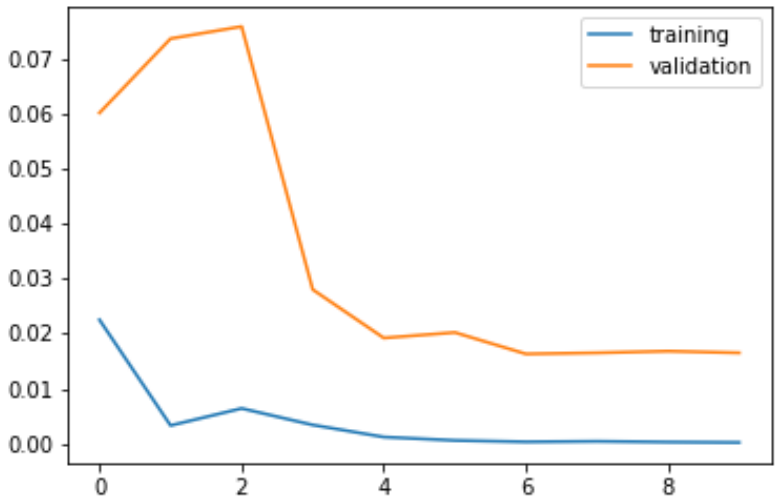


Figure 6.2: Loss plot for neural network trained in **regression multi-task** learning mode, **with count-vectorizing**.

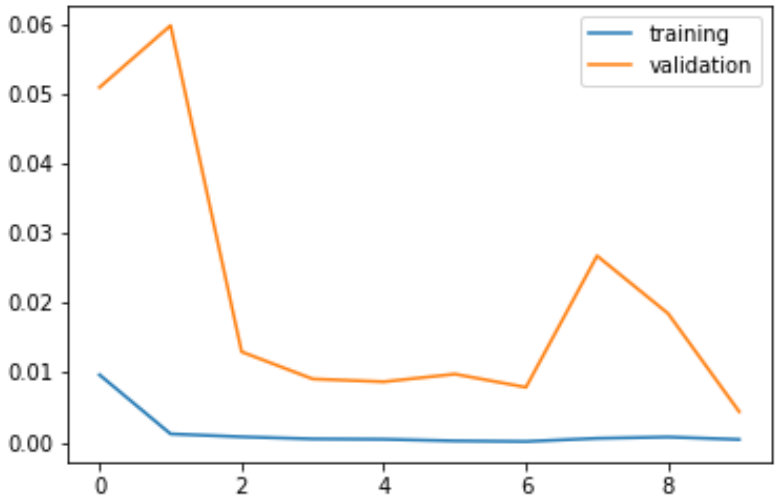


Figure 6.3: Loss plot for neural network trained in **regression multi-task** learning mode, **with GloVe**.

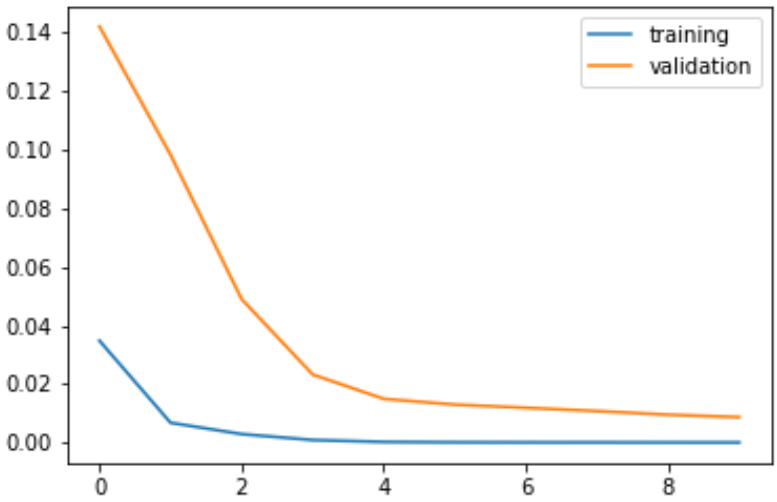


Figure 6.4: Loss plot for neural network trained in **regression multi-task** learning mode, **without news feature**.

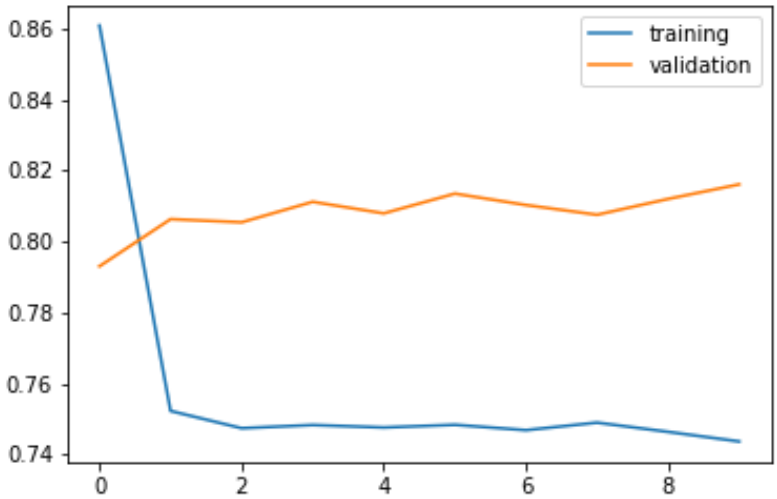


Figure 6.5: Loss plot for neural network trained in **classification multi-task** learning mode, **with Vader**.

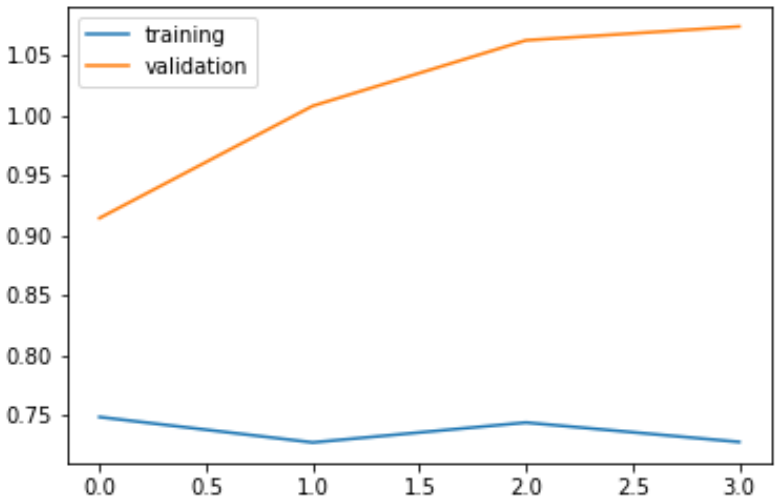


Figure 6.6: Loss plot for neural network trained in **classification multi-task** learning mode, **with count-vectorizing**.

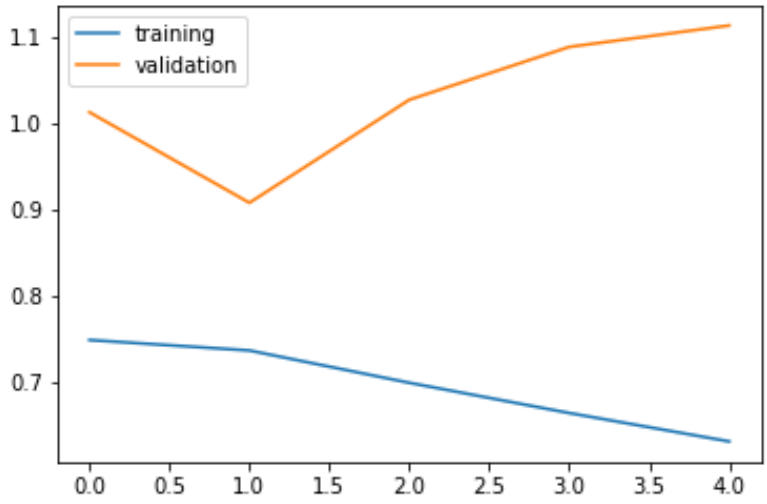


Figure 6.7: Loss plot for neural network trained in **classification multi-task** learning mode, **with GloVe**.

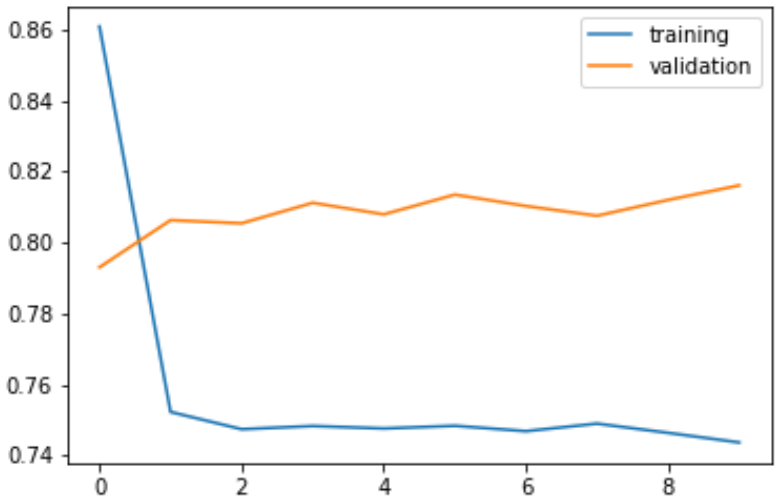


Figure 6.8: Loss plot for neural network trained in **classification multi-task** learning mode, **without news feature**.

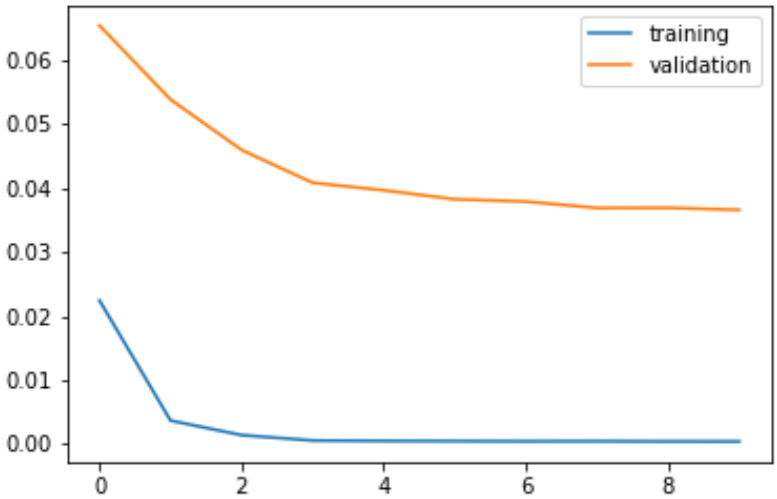


Figure 6.9: Loss plot for neural network trained in **regression single-task** learning mode for the Apple stock, **with Vader**.

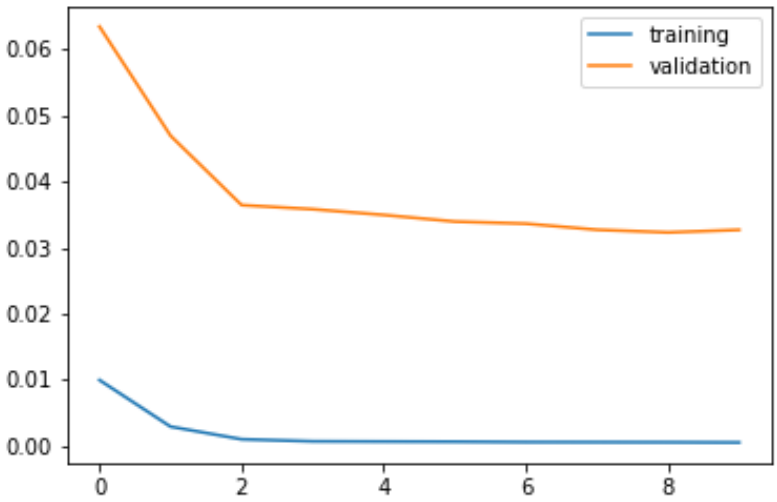


Figure 6.10: Loss plot for neural network trained in **regression single-task** learning mode for the Amazon stock, **with Vader**.

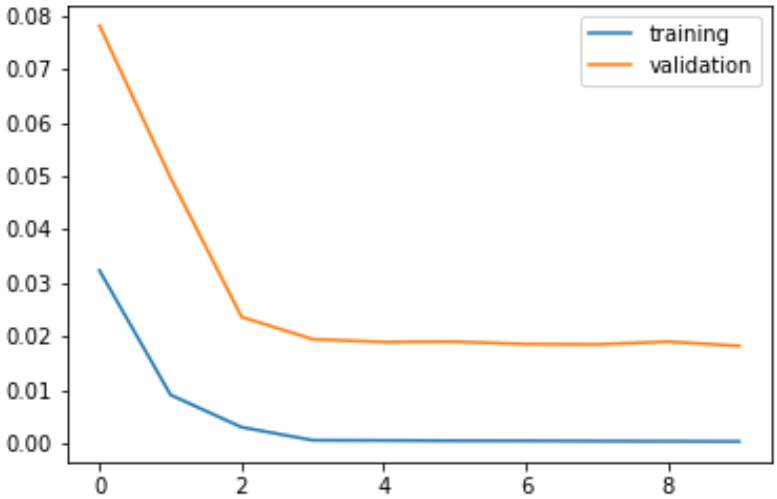


Figure 6.11: Loss plot for neural network trained in **regression single-task** learning mode for the Google stock, **with Vader**.

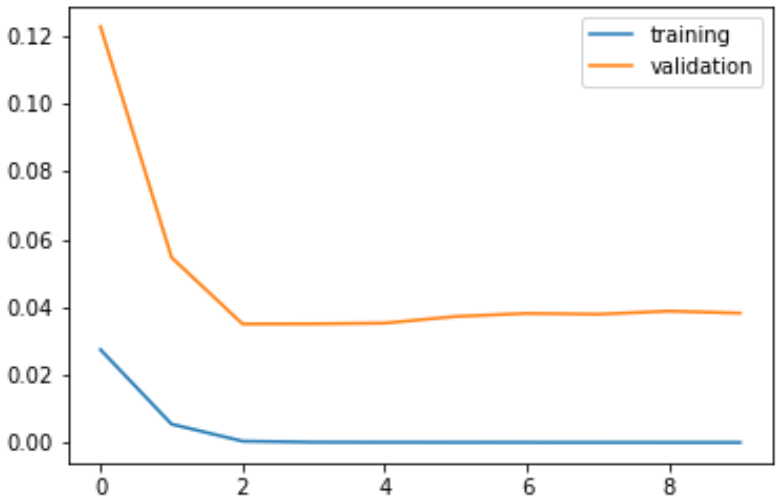


Figure 6.12: Loss plot for neural network trained in **regression single-task** learning mode for the Microsoft stock, **with Vader**.

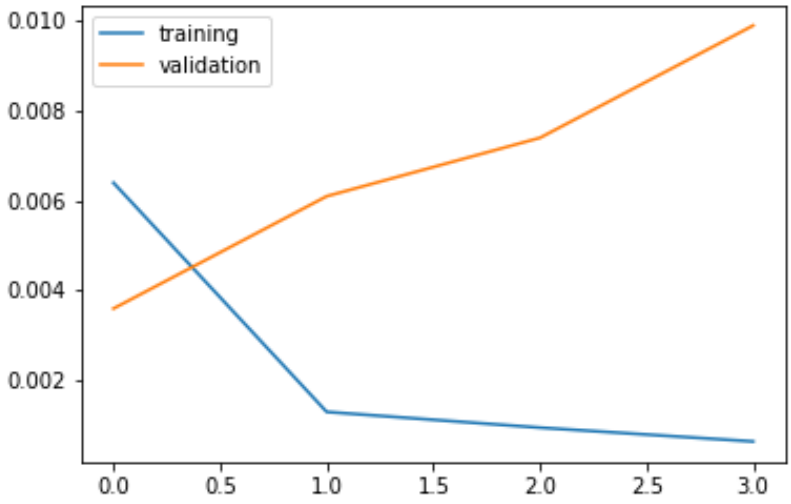


Figure 6.13: Loss plot for neural network trained in **regression single-task** learning mode for the Apple stock, **with count-vectorizing**.

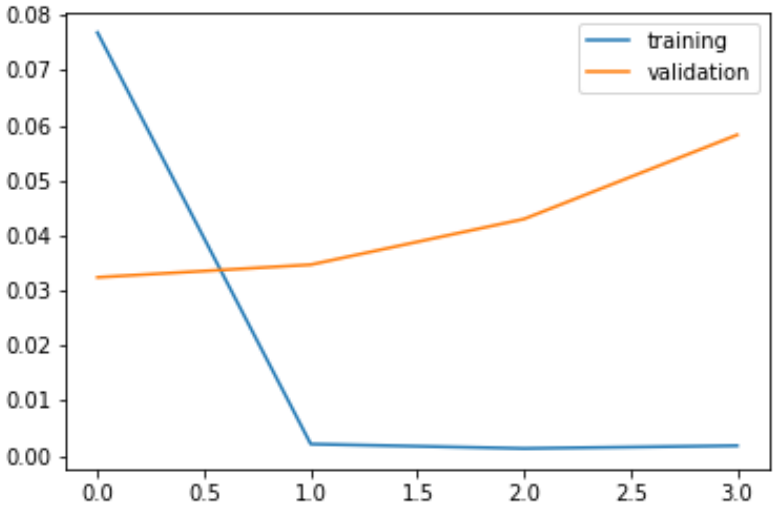


Figure 6.14: Loss plot for neural network trained in **regression single-task** learning mode for the Amazon stock, **with count-vectorizing**.

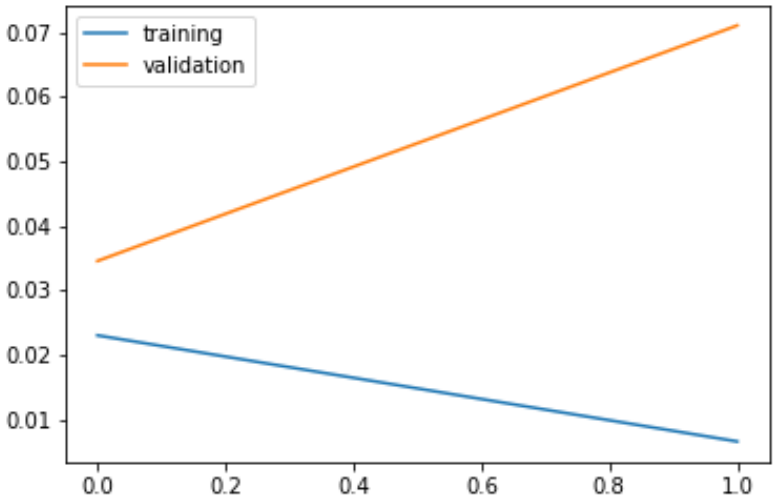


Figure 6.15: Loss plot for neural network trained in **regression single-task** learning mode for the Google stock, **with count-vectorizing**.

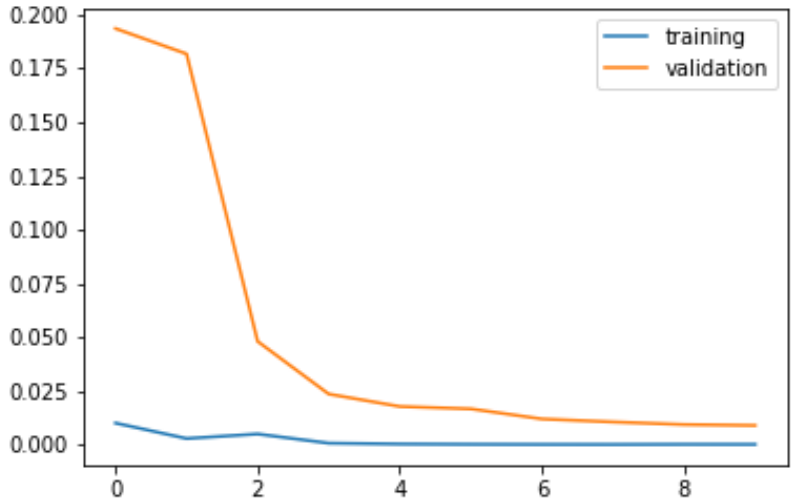


Figure 6.16: Loss plot for neural network trained in **regression single-task** learning mode for the Microsoft stock, **with count-vectorizing**.

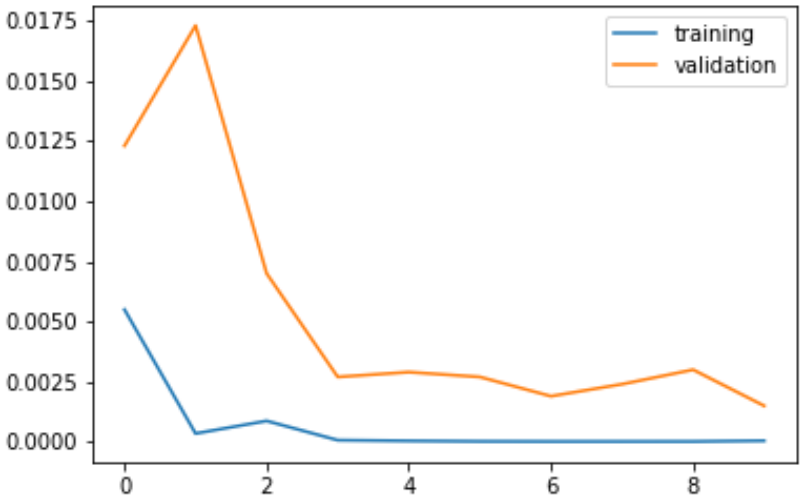


Figure 6.17: Loss plot for neural network trained in **regression single-task** learning mode for the Apple stock, **with GloVe**.

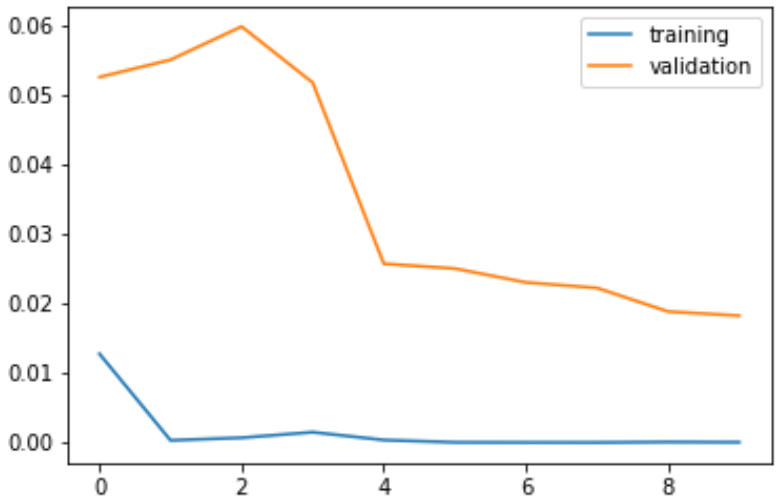


Figure 6.18: Loss plot for neural network trained in **regression single-task** learning mode for the Amazon stock, **with GloVe**.

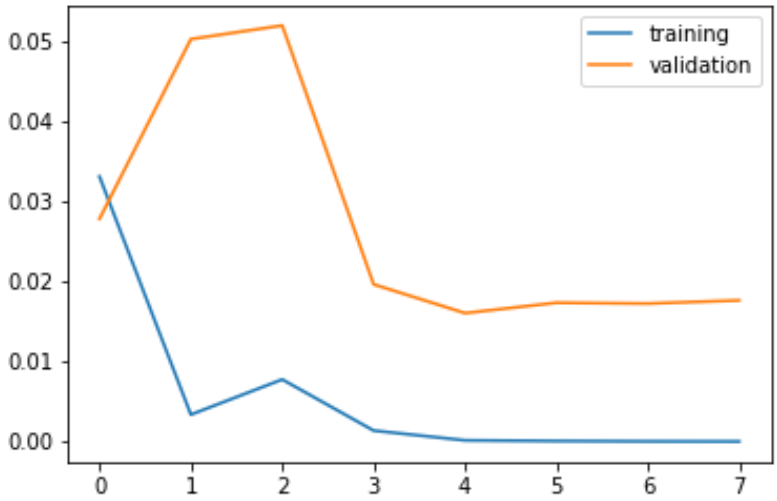


Figure 6.19: Loss plot for neural network trained in **regression single-task** learning mode for the Google stock, **with GloVe**.

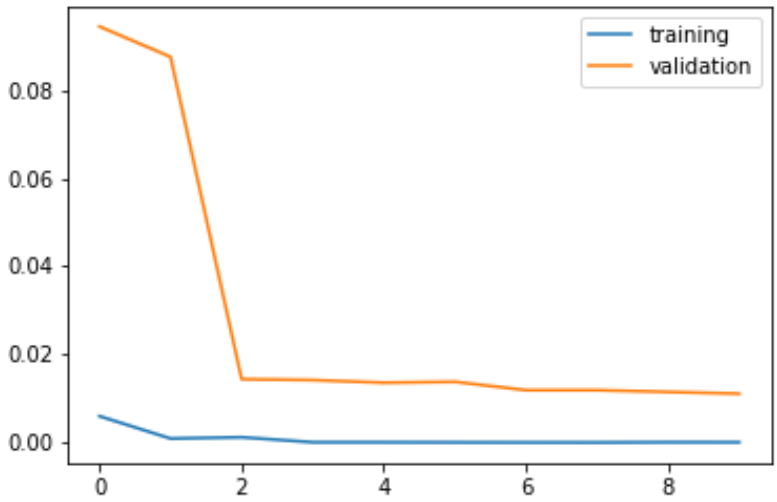


Figure 6.20: Loss plot for neural network trained in **regression single-task** learning mode for the Microsoft stock, **with GloVe**.

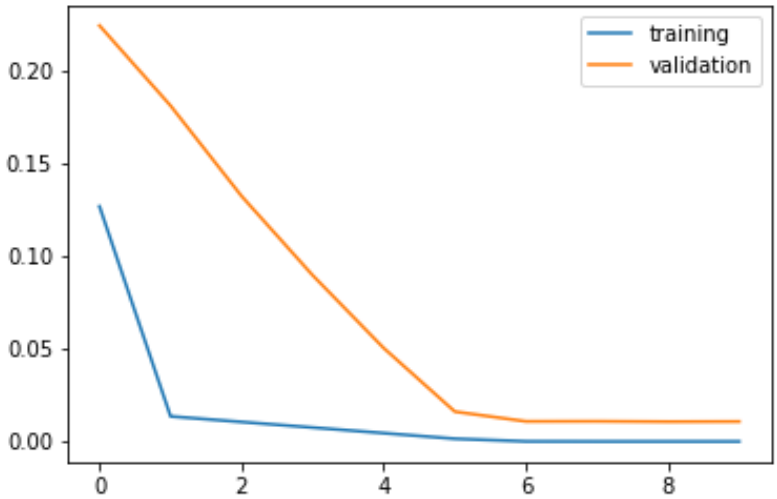


Figure 6.21: Loss plot for neural network trained in **regression single-task** learning mode for the Apple stock, **without news feature**.

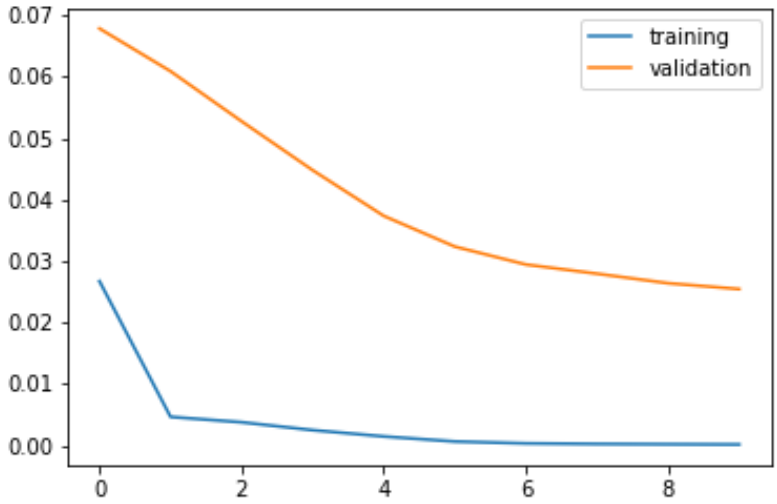


Figure 6.22: Loss plot for neural network trained in **regression single-task** learning mode for the Amazon stock, **without news feature**.

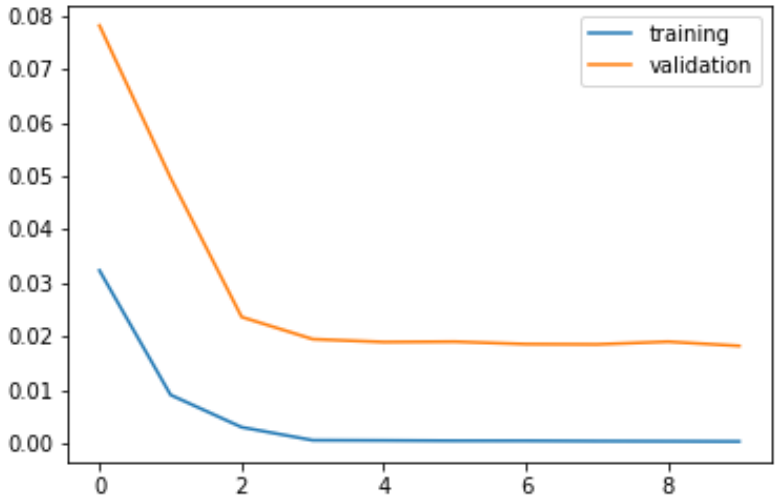


Figure 6.23: Loss plot for neural network trained in **regression single-task** learning mode for the Google stock, **without news feature**.

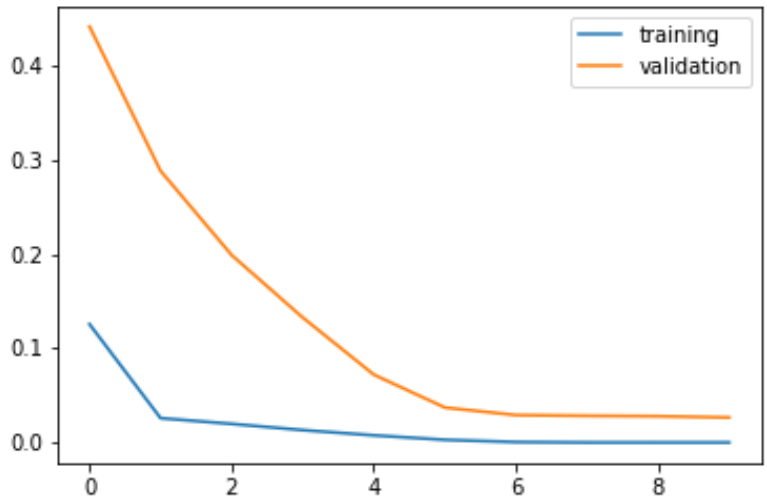


Figure 6.24: Loss plot for neural network trained in **regression single-task** learning mode for the Microsoft stock, **without news feature**.

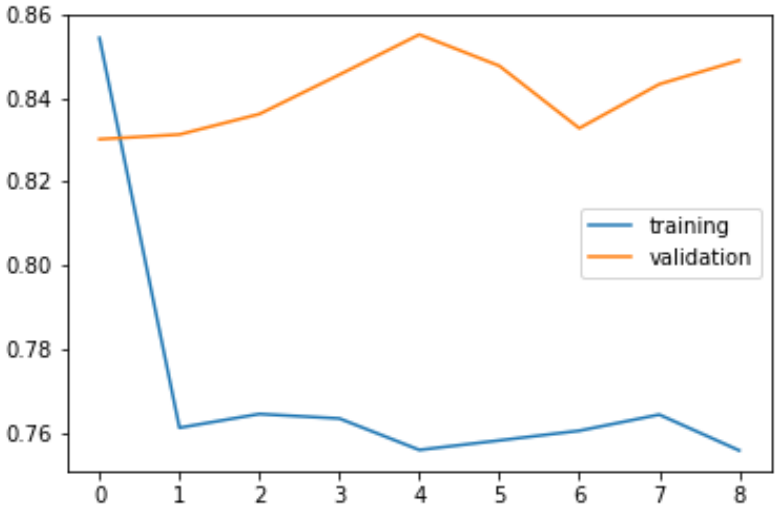


Figure 6.25: Loss plot for neural network trained in **classification single-task** learning mode for the Apple stock, **with Vader**.

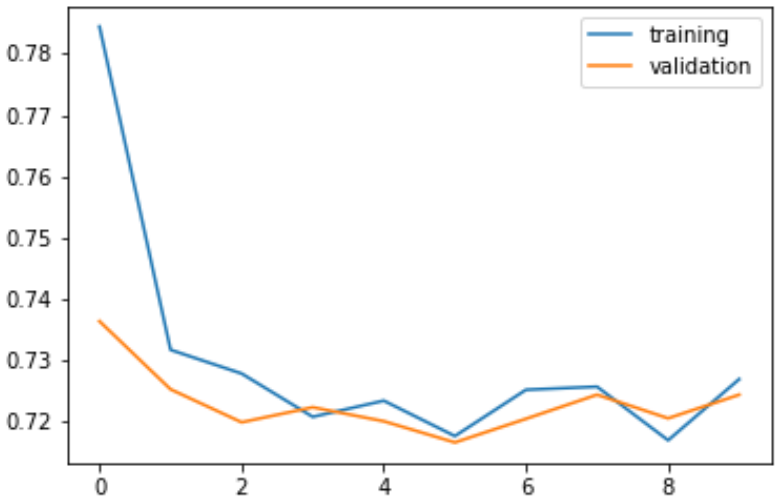


Figure 6.26: Loss plot for neural network trained in **classification single-task** learning mode for the Amazon stock, **with Vader**.

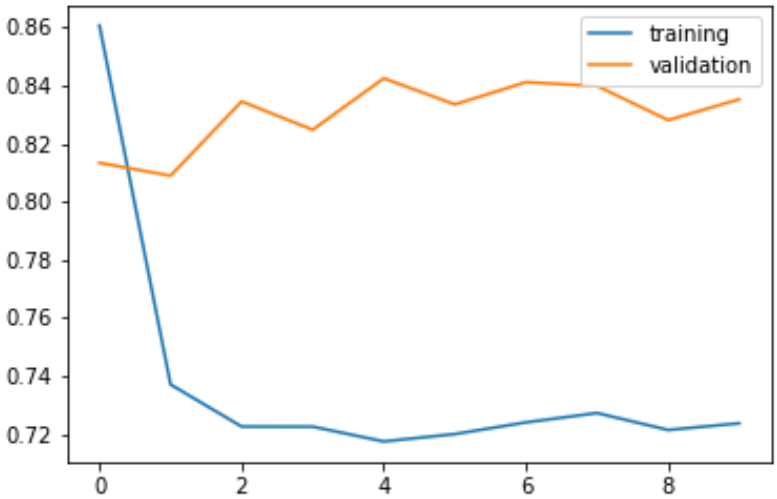


Figure 6.27: Loss plot for neural network trained in **classification single-task** learning mode for the Google stock, **with Vader**.

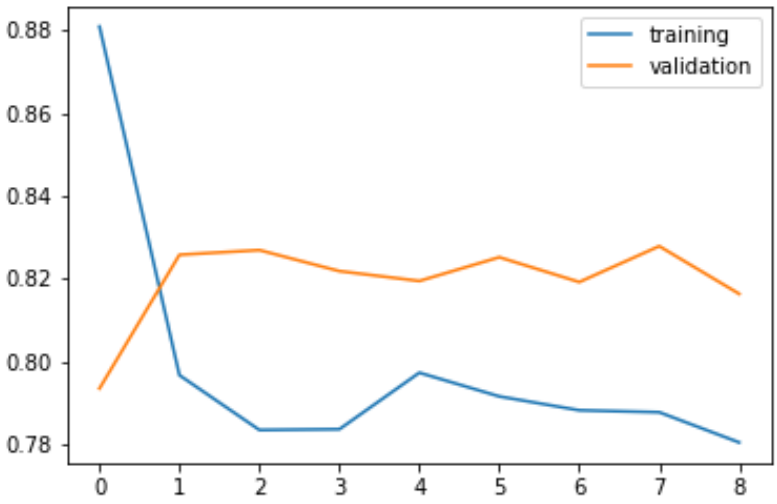


Figure 6.28: Loss plot for neural network trained in **classification single-task** learning mode for the Microsoft stock, **with Vader**.

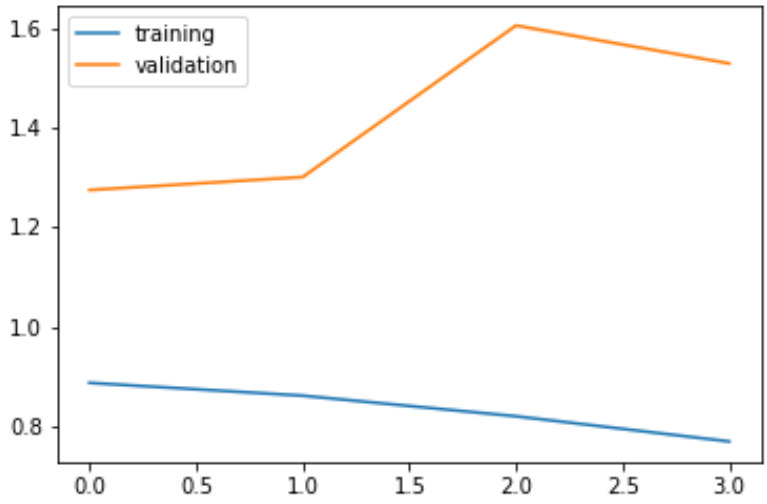


Figure 6.29: Loss plot for neural network trained in **classification single-task** learning mode for the Apple stock, **with count-vectorizing**.

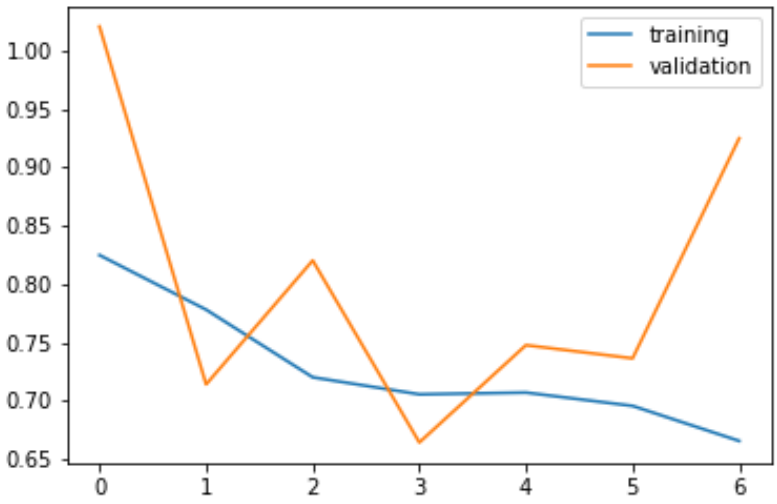


Figure 6.30: Loss plot for neural network trained in **classification single-task** learning mode for the Amazon stock, **with count-vectorizing**.

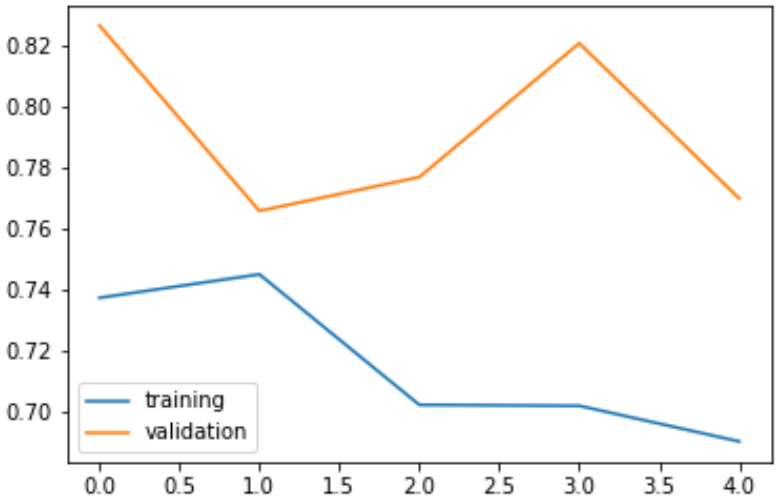


Figure 6.31: Loss plot for neural network trained in **classification single-task** learning mode for the Google stock, **with count-vectorizing**.

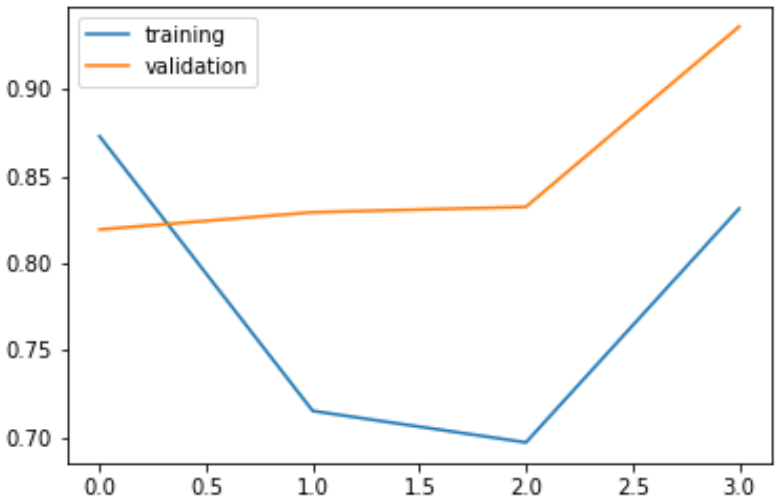


Figure 6.32: Loss plot for neural network trained in **classification single-task** learning mode for the Microsoft stock, **with count-vectorizing**.

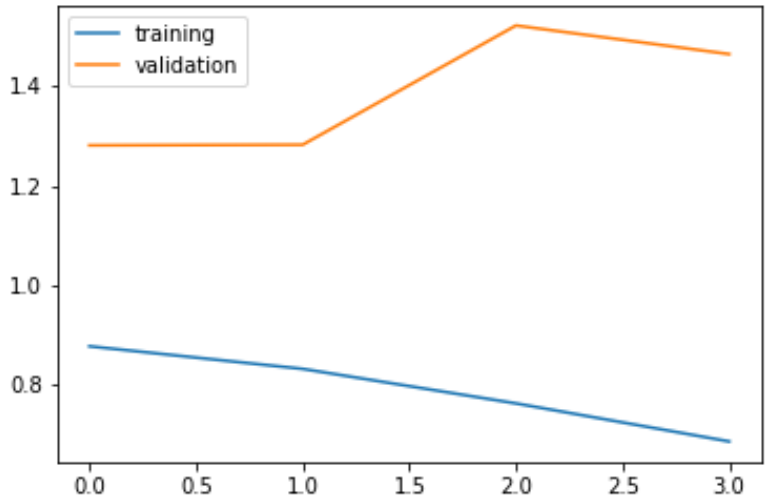


Figure 6.33: Loss plot for neural network trained in **classification single-task** learning mode for the Apple stock, **with GloVe**.

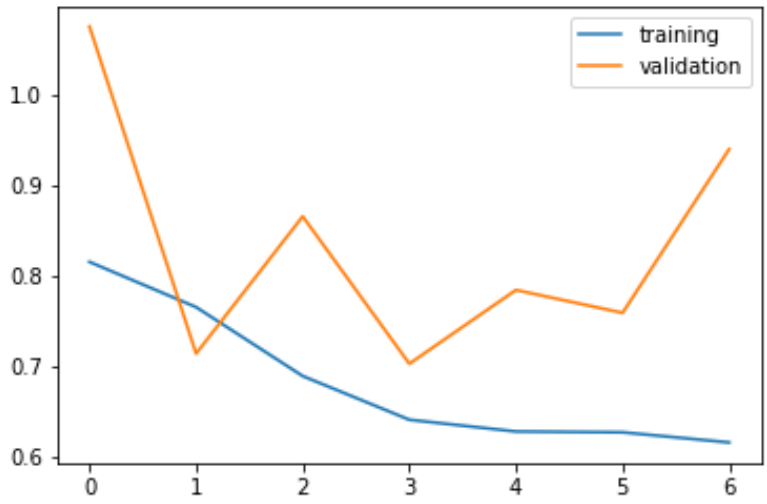


Figure 6.34: Loss plot for neural network trained in **classification single-task** learning mode for the Amazon stock, **with GloVe**.

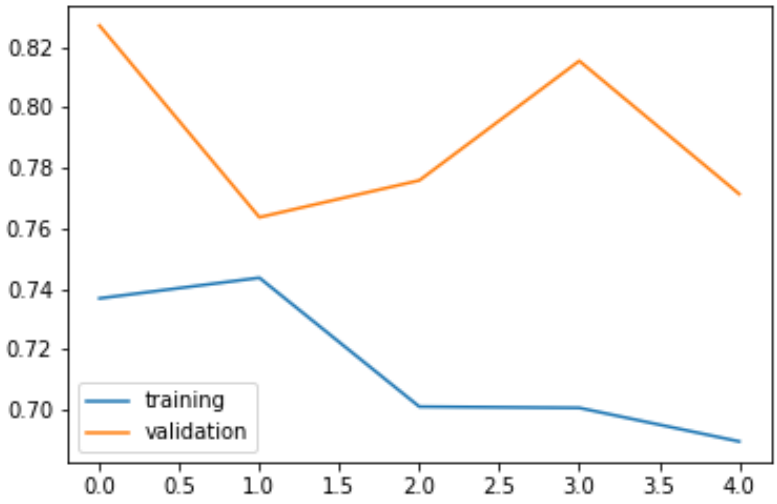


Figure 6.35: Loss plot for neural network trained in **classification single-task** learning mode for the Google stock, **with GloVe**.

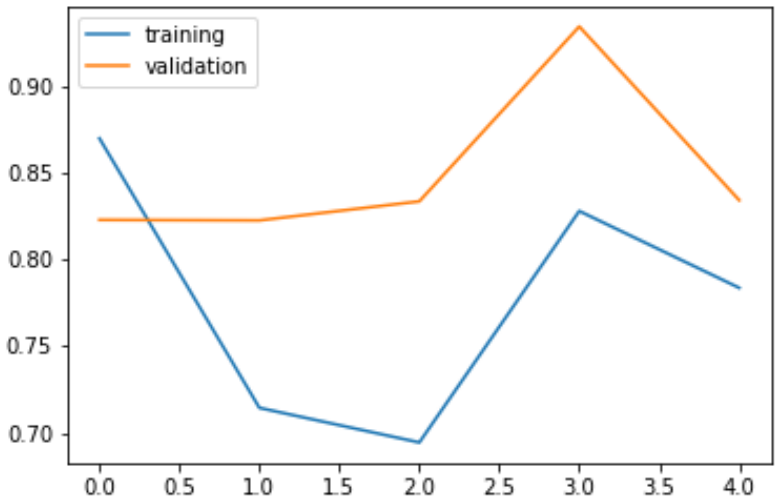


Figure 6.36: Loss plot for neural network trained in **classification single-task** learning mode for the Microsoft stock, **with GloVe**.

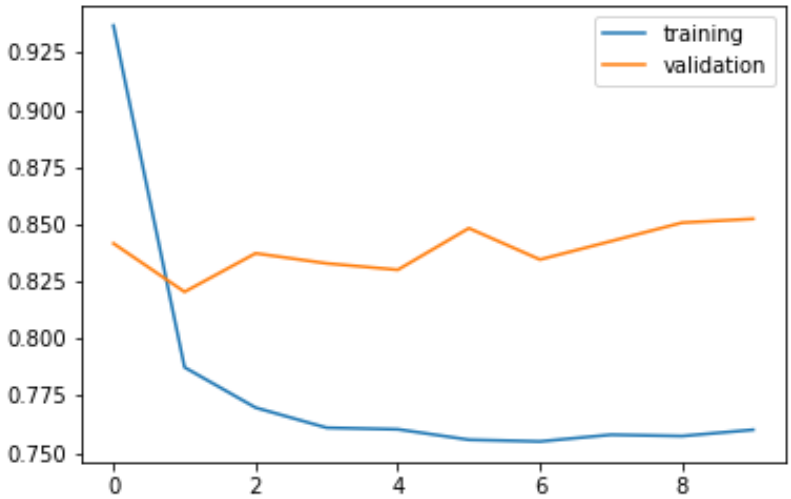


Figure 6.37: Loss plot for neural network trained in **classification single-task** learning mode for the Apple stock, **without news feature**.

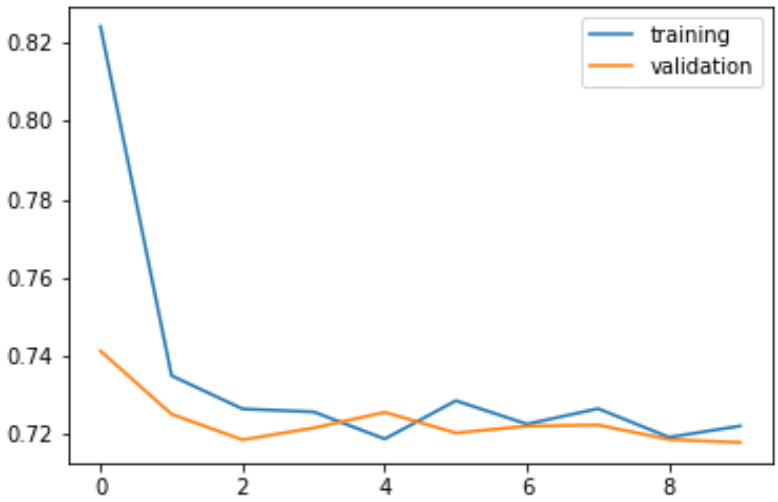


Figure 6.38: Loss plot for neural network trained in **classification single-task** learning mode for the Amazon stock, **without news feature**.

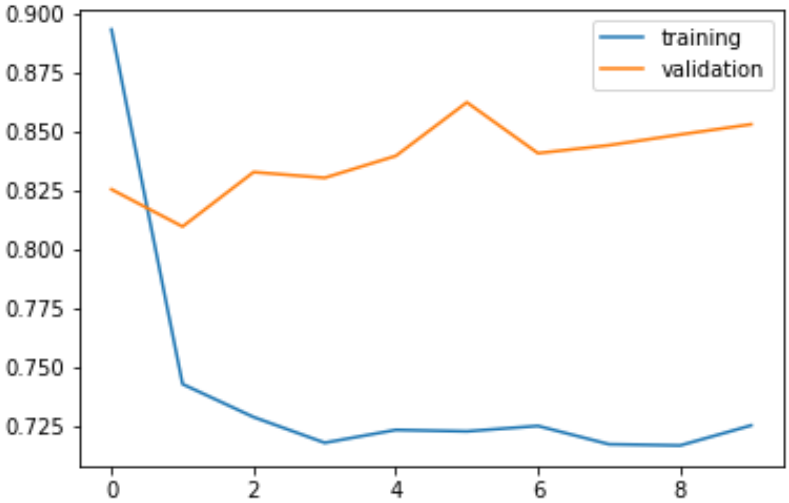


Figure 6.39: Loss plot for neural network trained in **classification single-task** learning mode for the Google stock, **without news feature**.

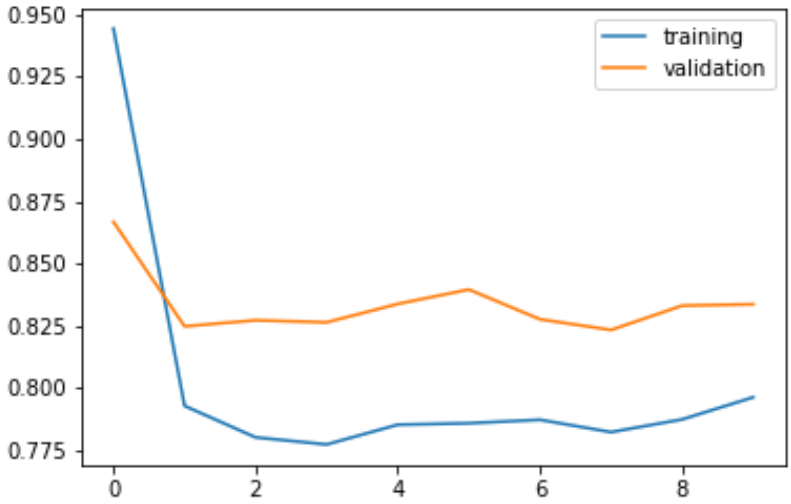


Figure 6.40: Loss plot for neural network trained in **classification single-task** learning mode for the Microsoft stock, **without news feature**.

6.2 Second experiments with all the stocks from NASDAQ

6.2.1 Grid description

We set up the next experiments keeping the same hyperparameters shown in the table 6.1. We set an experiment grid with the following parameters:

- multi-task learning or single-task learning;
- vader (also referred in the plots as sia) preprocessing or no news features;
- classification or regression;
- regarding experiments in classification, label creation with 10-90, 20-80 and 33-66 quantiles;
- different timeseries granularities, i.e. 20, 30, 120 minutes.

Regarding this parameter, training and test sequence feeding has been changed, so that now points in the sequence are subsampled and the reference close price $\hat{y}$, from where the label is computed, is the next point in the timeseries; in this way we aim to compare results with the work made by Cagliero and Fior [34].

configurations			epochs
multi	regression	vader	10
		countvec	10
		glove	10
		nonews	10
	classification	vader	10
		countvec	3
		glove	4
		nonews	10
single	regression	vader	10
		countvec	4.25
		glove	9.25
		nonews	10
	classification	vader	10
		countvec	4.25
		glove	4.5
		nonews	10

Table 6.4: Table with all the epochs occurred to let either the neural network converge to a solution or let the early stopping trigger. The values in the single-task experiments are a simple average of all the experiments.

6.2.2 Results

Loss plots from the different experiments show different behaviours whether we use multi-task training or not: indeed experiments held in single-task (6.47 - 6.52) last for the 10 epochs required for training than those in multi-task (6.41 - 6.46), which take from 4 to 6 epochs before early stopping, due to the rise of validation loss, occurs. Adding information about how other stocks are moving is helping the network generalizing, but more data is needed to prevent overfitting. Also, taking larger quantiles for classifying labels - we refer to the multi-task classification experiments - leads to more epochs in the training process: overfitting phenomenon delays as an imbalanced dataset is more difficult to predict. This may also lead to higher returns and less trading signals. Experiments in regression mode have a smoother loss curve: training and validation loss reach easier a plateau. Regarding the work from Fior et Cagliero [34], we see that there are differences in terms of performance: indeed the granularity that worked best in those experiments was the 30 minutes one, in terms of Maximum Drawdown.

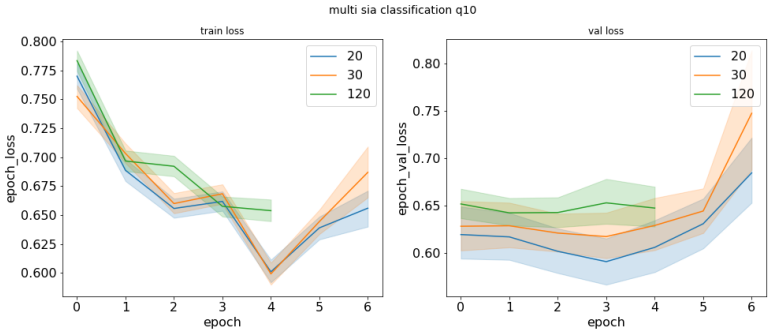


Figure 6.41: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader** and quantiles **10-90**.

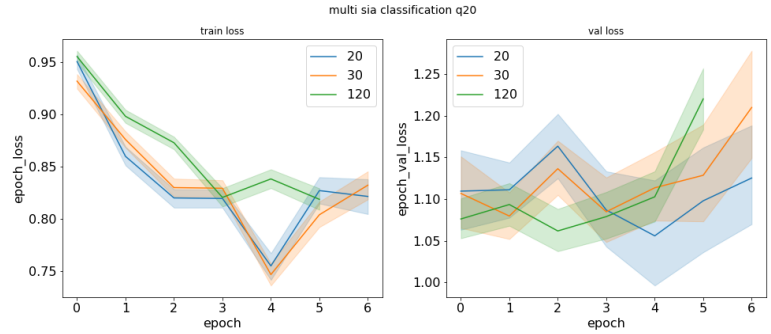


Figure 6.42: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader** and quantiles **20-80**.

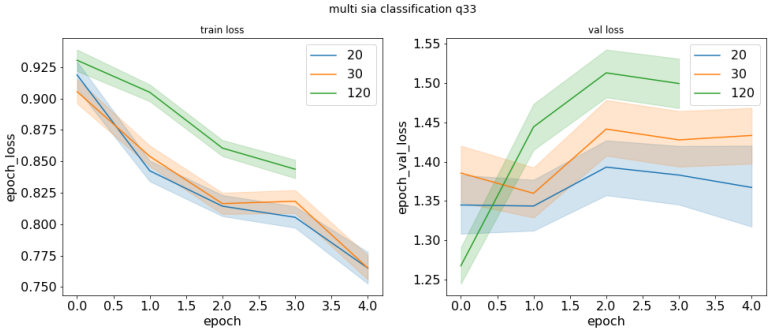


Figure 6.43: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader** and quantiles **33-66**.

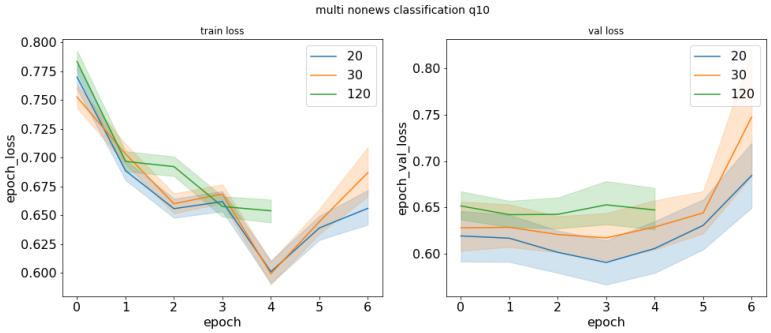


Figure 6.44: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature** and quantiles **10-90**.

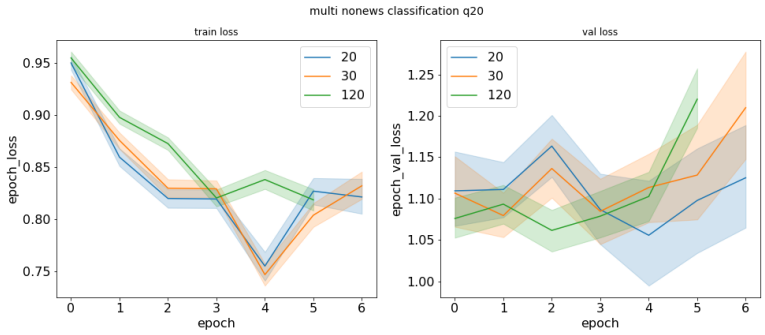


Figure 6.45: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature** and quantiles **20-80**.

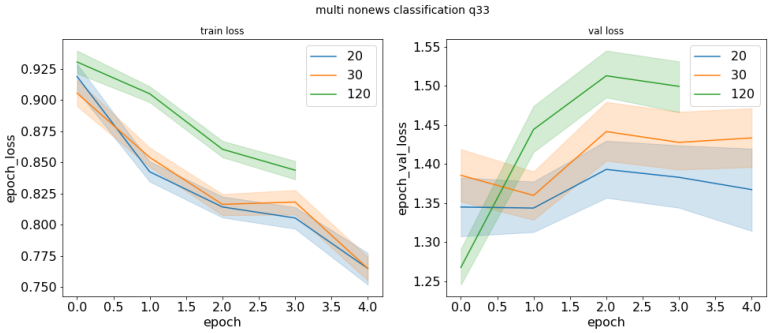


Figure 6.46: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature** and quantiles **33-66**.

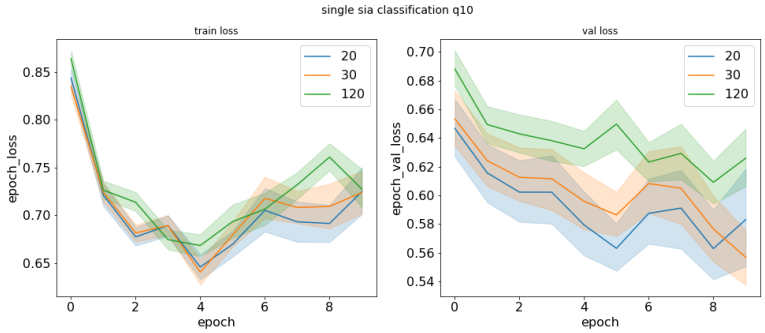


Figure 6.47: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader** and quantiles **10-90**.

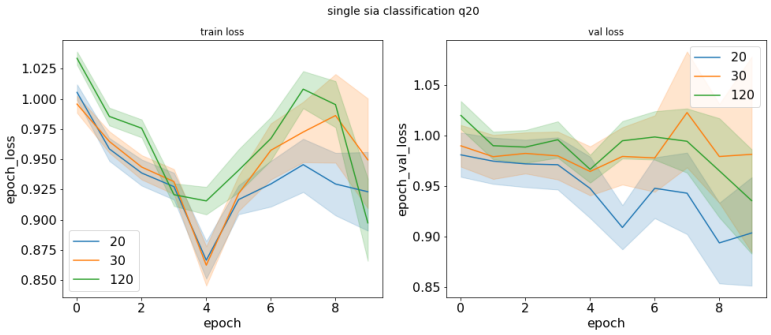


Figure 6.48: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader** and quantiles **20-80**.

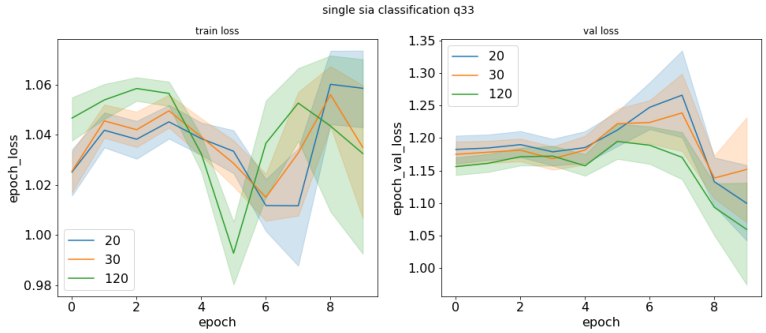


Figure 6.49: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader** and quantiles **33-66**.

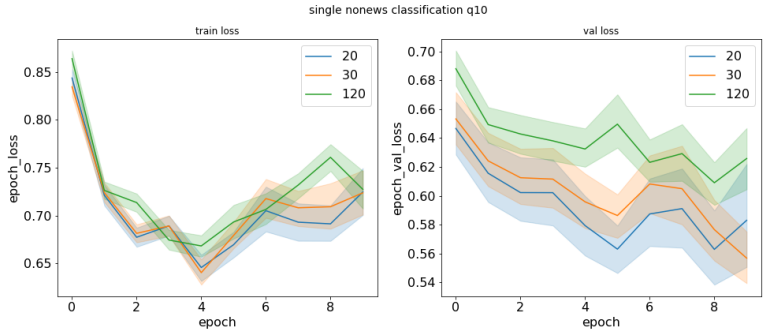


Figure 6.50: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature** and quantiles **10-90**.

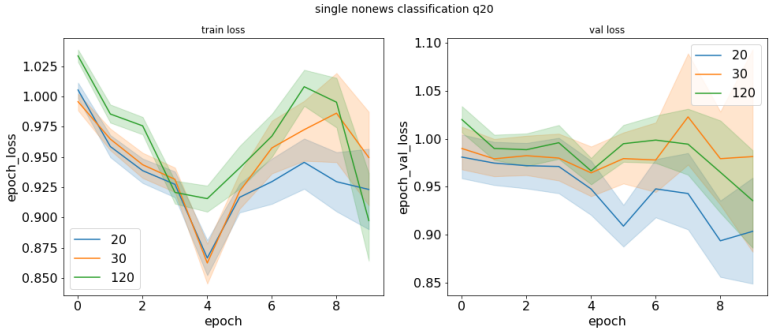


Figure 6.51: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature** and quantiles **20-80**.

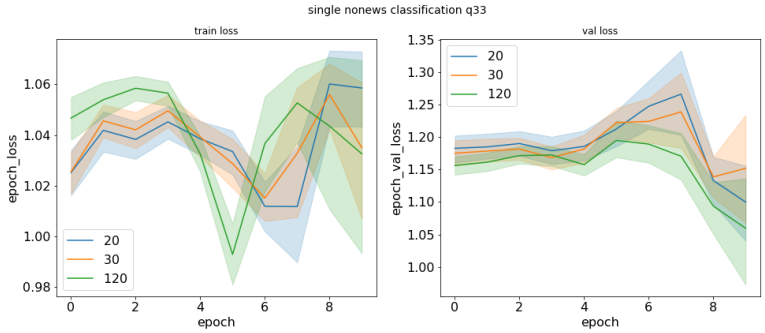


Figure 6.52: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature** and quantiles **33-66**.

6.3 First trading system

6.3.1 Pseudocode

Given the experiments in section 6.2, we develop an algorithm to use the test predictions to place orders and trace an equity line. First thing to mention, we simulate an initial capital of 100000 $: every order takes its quota from this amount. Second, we distinguish two algorithms: one for the experiments run in regression mode and the other for classification. The former considers the percentiles thresholds that label the price in classification, the latter checks directly the true label from the test set. The exact conditions are described in table 6.5, 6.6, 6.7.
Third, we sell the position either using the predictions from the machine learning experiments, as listed in algorithm 4, or using technical analysis indicators, as shown in algorithm 5.

Position	Regression	Classification
Long	pred_price > cur_price + delta_long[stock]*cur_price	predicted_label == LONG
Short	pred_price < cur_price + delta_short[stock]*cur_price	predicted_label == SHORT

Table 6.5: Buy criteria in regression and classification mode.

Position	Regression	Classification
Long	last_price > current_price + delta_long[stock] * current_price	prev_true_label != LONG
Short	last_price < current_price - delta_short[stock] * current_price	prev_true_label != SHORT

Table 6.6: Stop loss criteria in regression and classification mode.

6.3.2 Results

This first attempt does not give us good performance, as none of the experiments is earning profits from the orders we place. Figure 6.53 shows one of the performances: we can see that the 120-minute granularity gives us the lowest number of signals compared to the 20 and 30-minutes granularity, so it loses less money. This in addition to the fact that 10-90 percentiles reduce the number of signals from the predictions. We will continue the experiments keeping these two characteristics.

Position	Regression	Classification
Long	pred_price > current_price + delta_long[stock] * current_price	predicted_label == LONG
Short	pred_price < current_price - delta_short[stock]*current_price	predicted_label == SHORT

Table 6.7: Hold position criteria in regression and classification mode.

Algorithm 3 First trading system - Buying part

```
1: for each time t+delta do
2:   budget = capital
3:   order stocks by equity_line
4:   for each stock in last_equity_sorted do
5:     perc = 0.1
6:     if position not taken then
7:       if buy_signal_long() then
8:         expense = perc * budget
9:         if expense + current_expenses < budget  then
10:          BUY LONG
11:        else
12:          STOP BUYING
13:        end if
14:      else if buy_signal_short() then
15:        expense = cur_price * shares_lent
16:        if expense + current_expenses < budget then
17:          BUY SHORT
18:        else
19:          STOP BUYING
20:        end if
21:      else
22:        HOLD
23:      end if
24:    else
```

Algorithm 4 First trading system - Selling part

```
25:         if position == LONG then
26:           if is_stop_loss_long() then
27:             CLOSE POSITION
28:           else
29:             if is_still_long() then
30:               HOLD
31:             else
32:               CLOSE POSITION
33:             end if
34:           end if
35:         end if
36:         if position == SHORT then
37:           if is_stop_loss_short() then
38:             CLOSE POSITION
39:           end if
40:         else
41:           if is_still_short() then
42:             HOLD
43:           else
44:             CLOSE POSITION
45:           end if
46:         end if
47:       end if
48:     end for
49: end for
```

Algorithm 5 First trading system - Selling strategy with technical analysis filter

```
1: if position == LONG then
2:    if is_stop_loss_long() then
3:       CLOSE POSITION
4:    else if adx > 30 then
5:       if ema_20 < ema_50 then
6:          CLOSE POSITION
7:       else
8:          HOLD
9:       end if
10:   else if adx < 20 then
11:      if cur_price > upper_band then
12:         CLOSE POSITION
13:      else
14:         HOLD
15:      end if
16:   else
17:      HOLD
18:   end if
19: end if
20: if position == SHORT then
21:   if is_stop_loss_short() then
22:      CLOSE POSITION
23:   else if adx > 30 then
24:      if ema_20 > ema_50 then
25:         CLOSE POSITION
26:      else
27:         HOLD
28:      end if
29:   else if adx < 20 then
30:      if cur_price < lower_band then
31:         CLOSE POSITION
32:      else
33:         HOLD
34:      end if
35:   else
36:      HOLD
37:   end if
38: end if
```

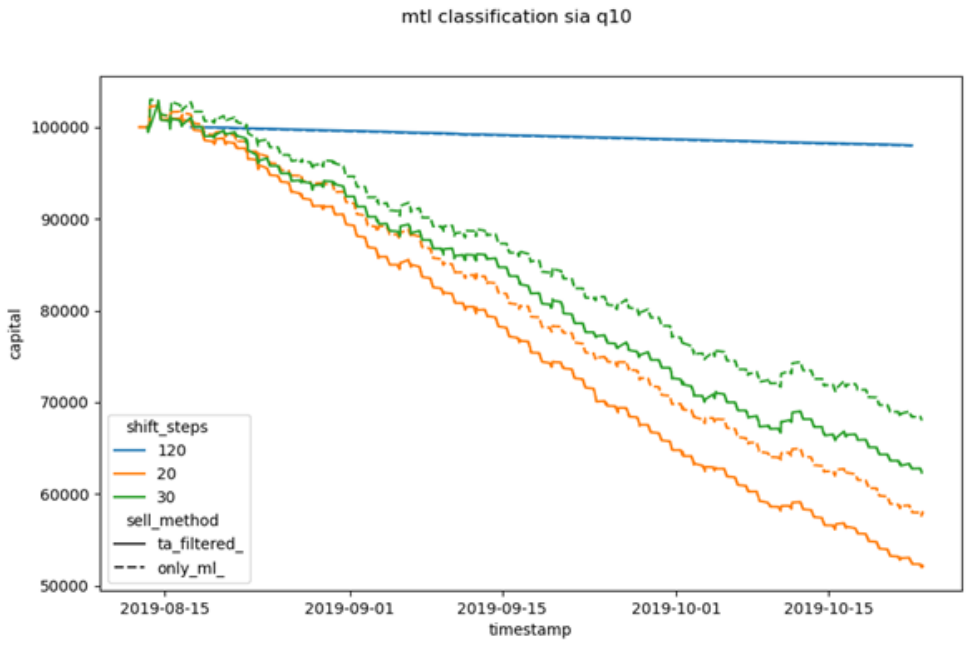


Figure 6.53: Equity line of **multi-task vader classification** neural network, with **10-90** quantiles. The 120-minute granularity experiment generates the least number of signals.

6.4 Third experiments with additional features and different RNN cells

6.4.1 Grid description

After selecting the 10-90 quantile and the 120 minutes granularity, due to the fewer number of signals in the trading system, we conducted further experiments to validate other ideas. Hyperparameters are listed in table 6.1. Here it is the next experiment grid:

- multi-task learning or single-task learning;
- vader (also referred as sia) preprocessing or no news features;
- GRU or LSTM recurrent cells;
- a fixed number of cells calculated as:

$$numCells = \left\lceil \frac{2}{3}(numFeatures + rnnOutput) \right\rceil; \tag{6.1}$$

- fixed time-series granularity, i.e. 120 minutes;
- only the classification mode with 10-90 quantiles for labelling;
- one, two or three dense layers with hyperbolic tangent as activation function;
- optional technical analysis features, described in section 5.2.

It has to be mentioned that the VADER news sentiment is now a mean of all the news sentiment in the 120 minutes interval taken into account, since pivoting the table, as made in 6.2, created too many signals in the news features: stacking numerous news in the same period wasn't beneficial.

6.4.2 Results

The main result that affects our dissertation is that multi-task experiments, as depicted in the previous iteration, overfits in all cases, while single-task performs for all the epochs defined. Adding just one dense layer gets the minimum training loss in both multi and single task sets

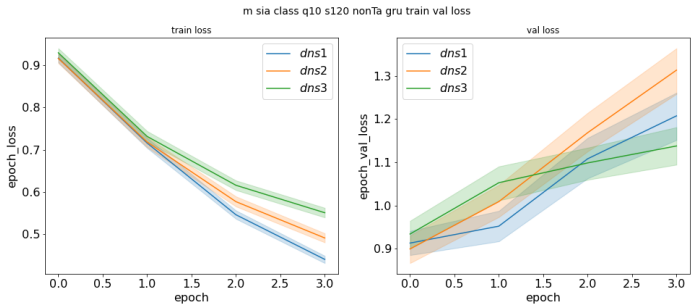


Figure 6.54: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **gru** cells, **without technical analysis features**.

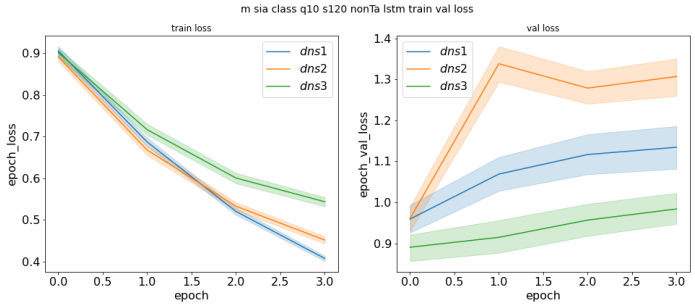


Figure 6.55: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **lstm** cells, **without technical analysis features**.

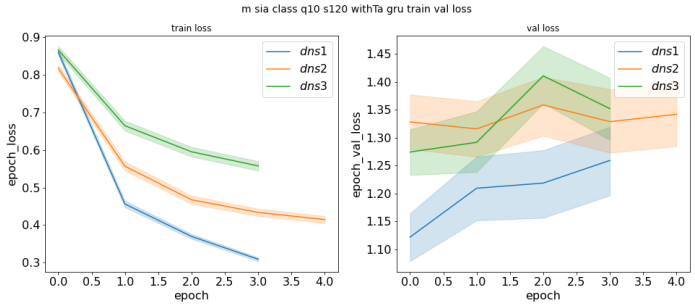


Figure 6.56: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **gru** cells, **with technical analysis features**.

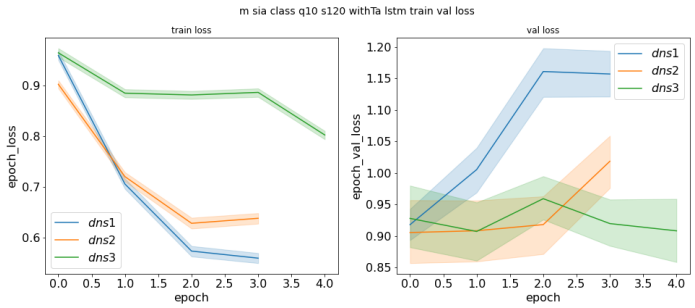


Figure 6.57: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **lstm** cells, **with technical analysis features**.

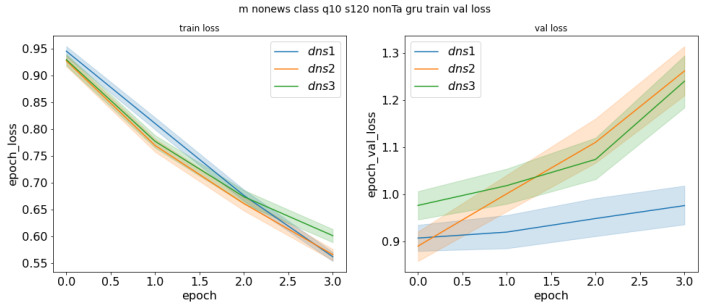


Figure 6.58: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **gru** cells, **without technical analysis features**.

Figure 6.59: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **lstm** cells, **without technical analysis features**.

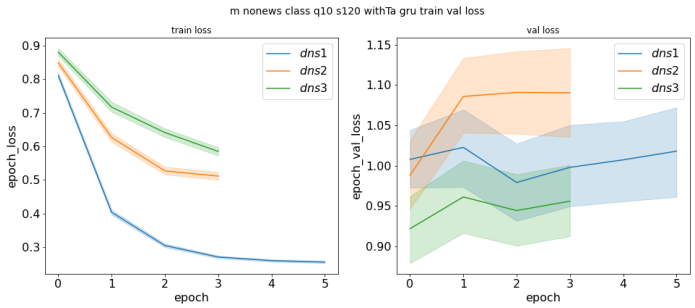


Figure 6.60: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **gru** cells, **with technical analysis features**.

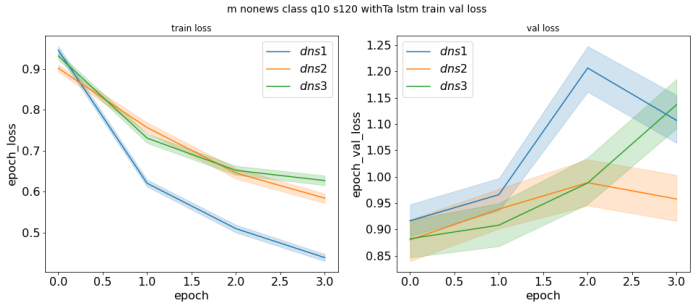


Figure 6.61: Loss plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **lstm** cells, **with technical analysis features**.

Figure 6.62: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader**, **gru** cells, **without technical analysis features**.

Figure 6.63: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader**, **lstm** cells, **without technical analysis features**.

Figure 6.64: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader**, **gru** cells, **with technical analysis features**.

Figure 6.65: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **with Vader**, **lstm** cells, **with technical analysis features**.

Figure 6.66: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature**, **gru** cells, **without technical analysis features**.

Figure 6.67: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature**, **lstm** cells, **without technical analysis features**.

Figure 6.68: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature**, **gru** cells, **with technical analysis features**.

Figure 6.69: Loss plot for neural network trained with Nasdaq 100 dataset in **classification single-task** learning mode, **without news feature**, **lstm** cells, **with technical analysis features**.

6.5 Second trading system

6.5.1 Pseudocode

We have decided to keep just the experiments in classification and change the trading system for that mode. We implemented an ADX filter that only assesses predictions signals if we are in a good directional index: this option is highlighted in red6. The stop loss criteria now considers the low or high price in the last period and fixed stop-loss thresholds are chosen: $0.01, 0.015, 0.02, 0.05$; the previous label mode now closes the position only if the label is the opposite of the one taken. Details are shown in table 6.9.
The technical analysis selling strategy now explores the moving average and Bollinger band different periods, as shown in the table 6.8

Ema min	3, 10, 14
Ema max	12, 14, 26
Bollinger	10, 14

Table 6.8: Exponential moving average and Bollinger bands periods.

First we try in subsection 6.5.2 to choose the best stop loss threshold in terms of performance, then we run again the same experiments including the *previous label* mode.

Position	Threshold	Previous
Long	last_low_price < price_bought - thres_stop_loss * price_bought	prev_true_label == SHORT
Short	last_high_price > price_bought + thres_stop_loss * price_bought	prev_true_label == LONG

Table 6.9: Stop-loss criteria in classification mode.

Algorithm 6 Second trading system - Buying part

```
1:  for each time t+120 do
2:    budget = capital
3:    order stocks by equity_line
4:    for each stock in last_equity_sorted do
5:      if position is taken then
6:        if  position == LONG then
7:          if  is_stop_loss_long() then
8:            CLOSE POSITION for stop loss
9:          end if
10:       end if
11:       if  position == SHORT then
12:         if  is_stop_loss_long() then
13:           CLOSE POSITION for stop loss
14:         end if
15:       end if
16:     end if
17:   end for
18:   for each stock in last_equity_sorted do
19:     perc = 0.1
20:     if position not taken then
21:       optionally -> if adx < 20 or adx > 30
22:       if buy_signal_long() then
23:         expense = perc * budget
24:         if expense + current_expenses < budget  then
25:           BUY LONG
26:         else
27:           STOP BUYING
28:         end if
29:       else if buy_signal_short() then
30:         expense = cur_price * shares_lent
31:         if expense + current_expenses < budget then
32:           BUY SHORT
33:         else
34:           STOP BUYING
35:         end if
36:       else
37:         HOLD
38:       end if
39:     else
```

Algorithm 7 Second trading system - Selling strategy with technical analysis filter

```
40:         if position == LONG then
41:           if is_long_prediction_wrong() then
42:             CLOSE POSITION
43:           else if adx > 30 then
44:             if ema_min < ema_max then
45:               CLOSE POSITION
46:             else
47:               HOLD
48:             end if
49:           else if adx < 20 then
50:             if cur_price > upper_band_period then
51:               CLOSE POSITION
52:             else
53:               HOLD
54:             end if
55:           else
56:             HOLD
57:           end if
58:         end if
59:         if position == SHORT then
60:           if is_short_prediction_wrong() then
61:             CLOSE POSITION
62:           else if adx > 30 then
63:             if ema_min > ema_max then
64:               CLOSE POSITION
65:             else
66:               HOLD
67:             end if
68:           else if adx < 20 then
69:             if cur_price < lower_band_period then
70:               CLOSE POSITION
71:             else
72:               HOLD
73:             end if
74:           else
75:             HOLD
76:           end if
77:         end if
78:       end if
79:     end for
80: end for
```

6.5.2 Choice of stop loss threshold

The tables 6.10 and 6.11 represent an aggregation, intended as mean and standard deviation, of portfolio performances respectively for experiments run without using the stop loss and for stop loss thresholds 0.01, 0.015, 0.02, 0.05.

metrics	no sl	sl 0.01	sl 0.015	sl 0.02	sl 0.05
Annual return	-23.67	-23.32	-23.38	-23.51	-23.64
Cumulative returns	-6.72	-6.63	-6.64	-6.67	-6.72
Annual volatility	6.0	5.36	5.64	5.79	5.98
Sharpe ratio	-4.61	-5.1	-4.83	-4.73	-4.61
Calmar ratio	-3.15	-3.24	-3.21	-3.19	-3.15
Stability	0.86	0.87	0.87	0.87	0.86
Max drawdown	-7.42	-7.17	-7.25	-7.3	-7.41
Omega ratio	0.43	0.4	0.41	0.42	0.42
Sortino ratio	-5.12	-5.55	-5.33	-5.23	-5.12
Skew	-0.45	-0.38	-0.4	-0.43	-0.45
Kurtosis	3.45	3.47	3.37	3.29	3.42
Tail ratio	0.66	0.65	0.67	0.66	0.66
Daily value at risk	-0.86	-0.78	-0.82	-0.84	-0.86
Alpha	-0.23	-0.23	-0.23	-0.23	-0.23
Beta	-0.04	-0.04	-0.04	-0.04	-0.04

Table 6.10: Mean performance metrics for the four stop loss thresholds (sl x) and the option without stop loss (no sl).

6.5.3 Experiments with the previous label sell strategy

This section summarizes the performances from the 21504 trading system combinations: the figures 6.70, 6.71, 6.72, 6.73, 6.74, 6.75, 6.77 contain one point for each trading system, located in the x axis for the final capital value and the y axis for the percentage of trading signals, excluding stop loss and last trade in the day.

Figure 6.70 shows clearly that filtering buy signals with the ADX condition reduces the stop loss and last trade percentage, so we enter the position in a better condition. The exponential moving average with periods 14 and 26 gives the greatest *other signals* percentage in figure 6.72. The higher the difference between the periods the more probable this will give us a signal; it is not indicative of best performances. Selling with technical analysis signals gives us excellent performances, as shown in figure 6.73.

Lastly, it is evident that in figure 6.80 stopping loss looking at the previous true label (6.9) outperforms all the classic threshold stop-loss practices.

metrics	no sl	sl 0.01	sl 0.015	sl 0.02	sl 0.05
Annual return	7.89	8.48	8.03	7.9	7.87
Cumulative returns	2.43	2.64	2.49	2.44	2.43
Annual volatility	1.43	1.28	1.35	1.36	1.42
Sharpe ratio	1.7	1.96	1.76	1.71	1.69
Calmar ratio	0.52	0.37	0.37	0.43	0.52
Stability	0.15	0.13	0.13	0.14	0.15
Max drawdown	2.28	2.48	2.37	2.31	2.28
Omega ratio	0.15	0.15	0.14	0.15	0.15
Sortino ratio	1.47	1.59	1.47	1.45	1.47
Skew	0.93	1.0	0.93	0.9	0.93
Kurtosis	2.54	2.85	2.67	2.54	2.54
Tail ratio	0.25	0.27	0.26	0.26	0.24
Daily value at risk	0.2	0.18	0.19	0.2	0.2
Alpha	0.08	0.08	0.08	0.08	0.08
Beta	0.05	0.04	0.04	0.05	0.05

Table 6.11: Standard deviation of performance metrics for the four stop loss thresholds (sl x) and the option without stop loss (no sl).

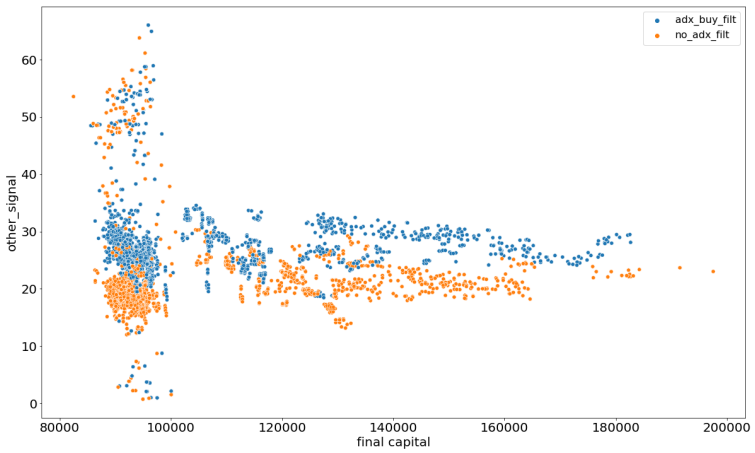


Figure 6.70: Equity-signal plot for the trading system experiments. In blue the experiments with the ADX filter, in orange those without.

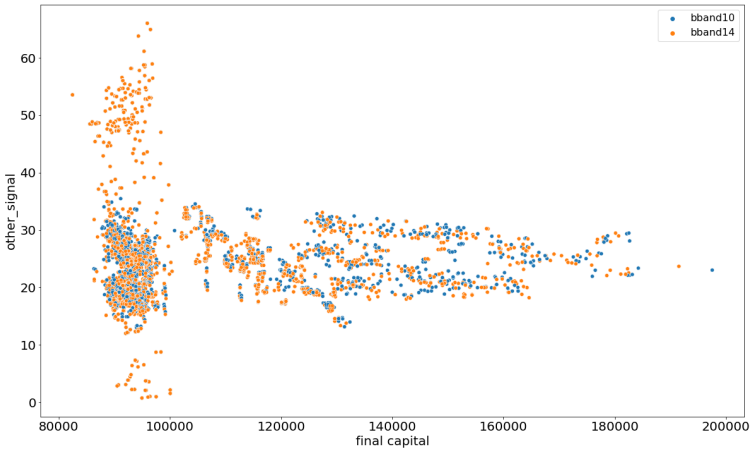


Figure 6.71: Equity-signal plot for the trading system experiments. In blue the experiments with 10-period Bollinger Band, in orange those with 14-period Bollinger Band.

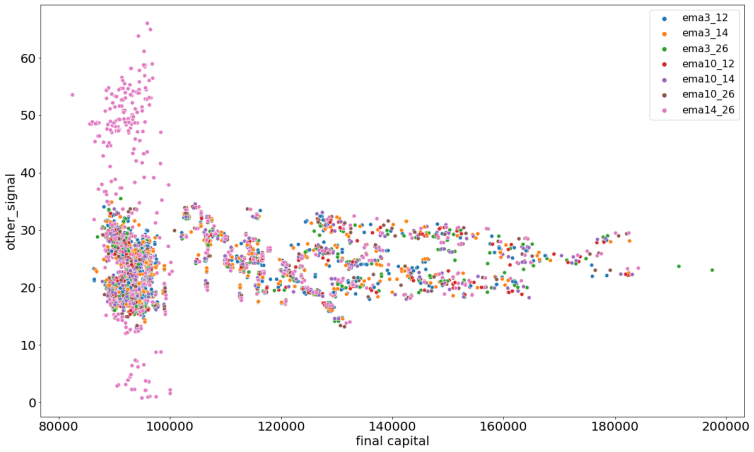


Figure 6.72: Equity-signal plot for the trading system experiments. Points are blue for EMA(3,12), orange for EMA(3,14), green for EMA(3,26), red for EMA(10,12), violet for EMA(10,14), purple for EMA(10,26), pink for EMA(14,26).

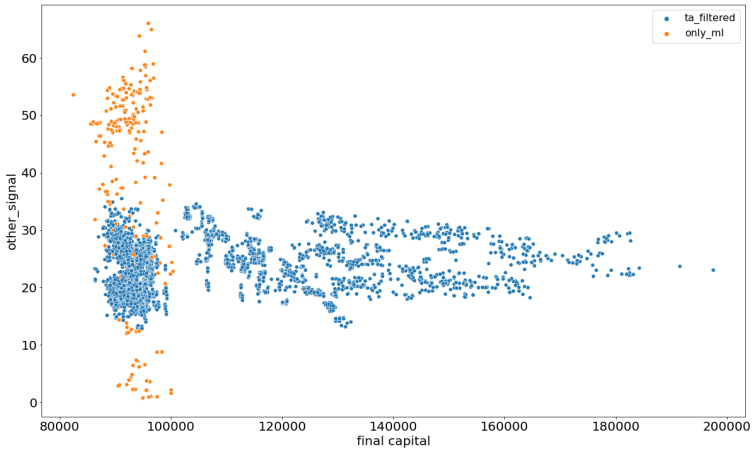


Figure 6.73: Equity-signal plot for the trading system experiments. In blue the experiments with technical analysis selling strategy, in orange those using just the signals from the neural networks.

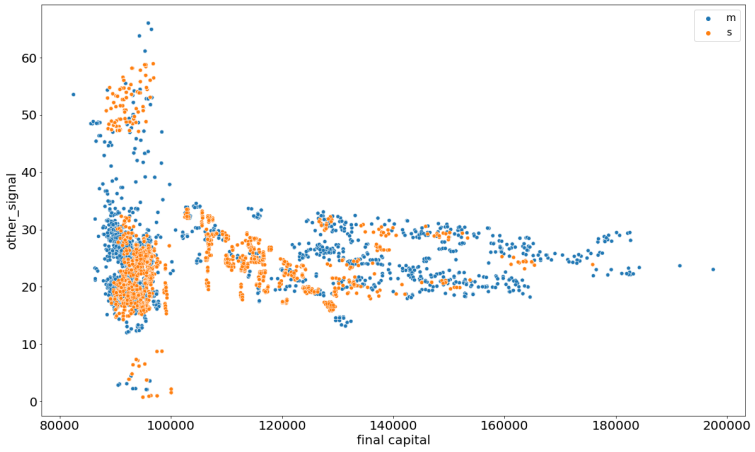


Figure 6.74: Equity-signal plot for the trading system experiments. In blue the experiments in multi-task learning, in orange those in single-task learning.

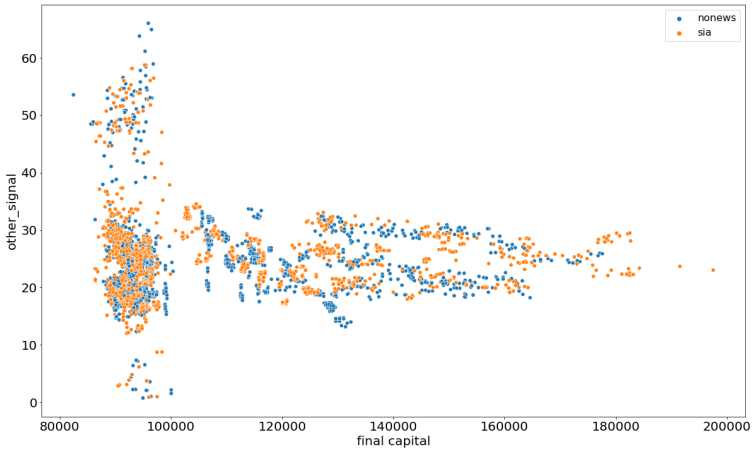


Figure 6.75: Equity-signal plot for the trading system experiments. In blue the experiments without news encoding, in orange those with VADER encoding.

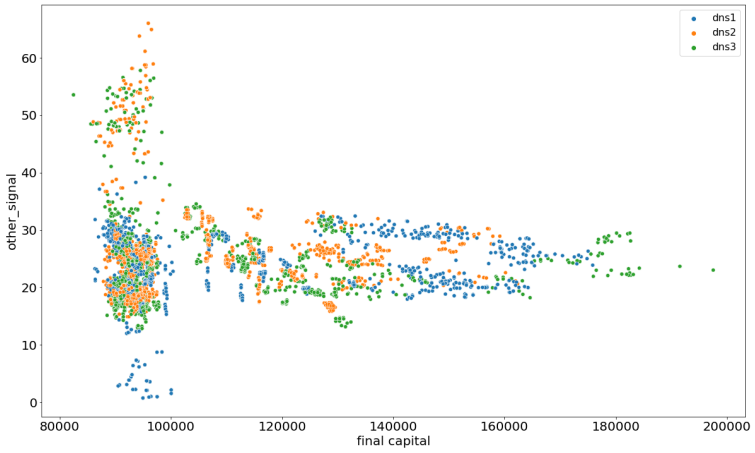


Figure 6.76: Equity-signal plot for the trading system experiments. In blue the experiments with one dense layer after the RNNs, in orange those with two dense layers, in green those with three.

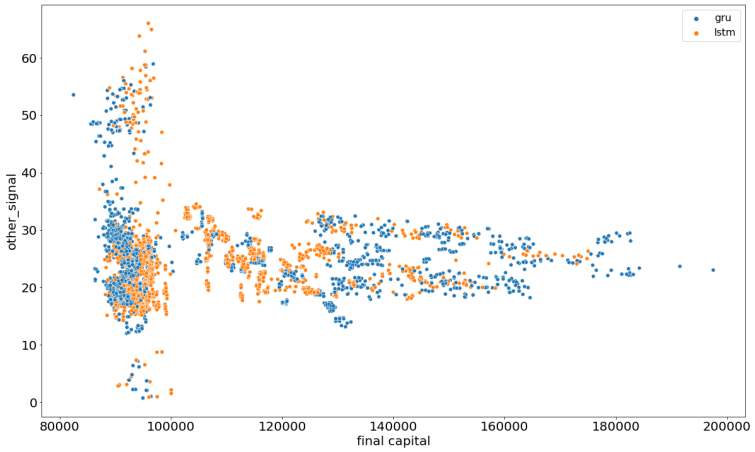


Figure 6.77: Equity-signal plot for the trading system experiments. In blue the experiments with GRU cells, in orange those with LSTM cells.

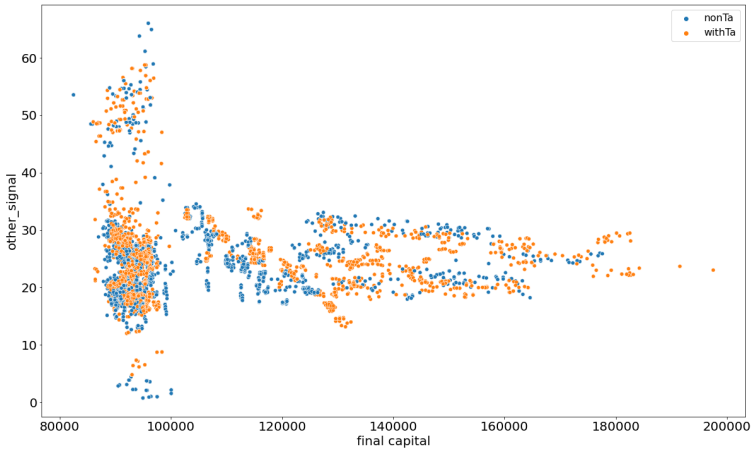


Figure 6.78: Equity-signal plot for the trading system experiments. In blue the experiments without technical analysis features, in orange those with technical analysis features.

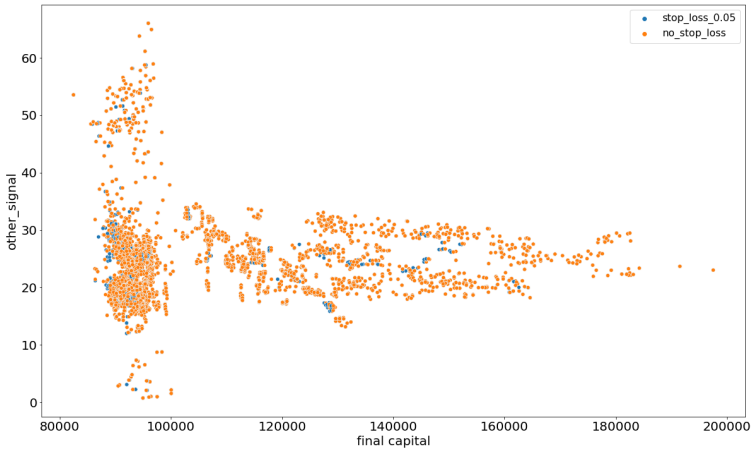


Figure 6.79: Equity-signal plot for the trading system experiments. In blue the experiments without stop loss, in orange those with stop loss threshold 0.05

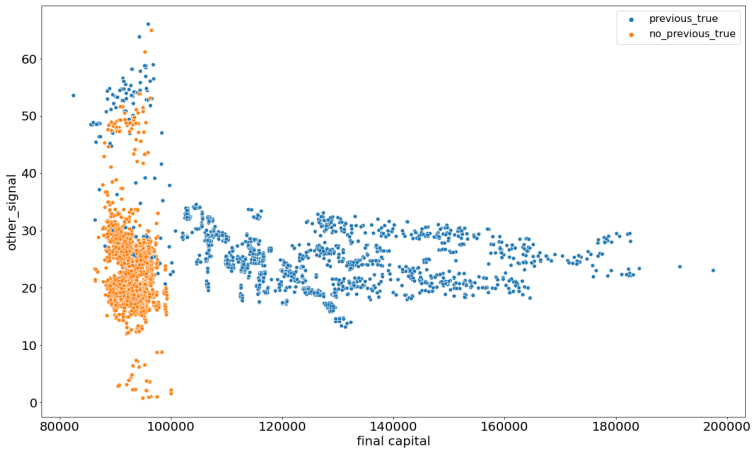


Figure 6.80: Equity-signal plot for the trading system experiments. In blue the experiments previous label selling strategy, in blue those without it.

In order to give a quantity of our results, we select a subset of these experiments and we compute performance statistics using pyfolio. In the table below we show how we have chosen the baseline, referred as **b** and the relative changes. Metrics are shown as mean among the groups displayed in tables 6.13 and 6.14. Standard

deviation for the same performance metrics is shown in tables 6.15 and 6.16.

b	single, nonews, no ta filter, nonTa, sl 0.05, no previous
b+n	single, vader, no ta filter, nonTa, sl 0.05, no previous
b+n+ta	single, vader, no ta filter, withTa, sl 0.05, no previous
b+n+ta+f	single, vader, ta filter, withTa, sl 0.05, no previous
b+n+ta+f+p	single, vader, ta filter, withTa, sl 0.05, previous
m+b	multi, nonews, no ta filter, nonTa, sl 0.05, no previous
m+b+n	multi, vader, no ta filter, nonTa, sl 0.05, no previous
m+b+n+ta	multi, vader, no ta filter, withTa, sl 0.05, no previous
m+b+n+ta+f	multi, vader, ta filter, withTa, sl 0.05, no previous
m+b+n+ta+f+p	multi, vader, ta filter, withTa, sl 0.05, previous

Table 6.12: Baseline groups for performance statistics.

metrics	b	b+n	b+n+ta	b+n+ta+f	b+n+ta+f+p
Annual return	-21.43	-28.18	-23.73	-19.94	163.2
Cumulative returns	-6.17	-8.13	-6.75	-5.54	25.13
Annual volatility	4.62	5.42	5.55	5.31	11.39
Sharpe ratio	-5.0	-6.3	-5.13	-4.22	6.93
Calmar ratio	-2.9	-3.25	-3.16	-3.27	113.78
Stability	0.78	0.94	0.83	0.86	0.92
Max drawdown	-6.38	-8.58	-7.32	-6.08	-1.47
Omega ratio	0.41	0.32	0.41	0.46	4.52
Sortino ratio	-5.04	-6.42	-5.39	-4.81	25.54
Skew	-1.11	-0.62	-0.62	-0.36	1.15
Kurtosis	4.19	2.32	2.39	2.33	1.6
Tail ratio	0.65	0.54	0.61	0.64	4.44
Daily value at risk	-0.68	-0.82	-0.8	-0.76	-1.09
Alpha	-0.21	-0.27	-0.23	-0.2	1.64
Beta	-0.06	-0.06	-0.05	-0.03	0.03

Table 6.13: Mean of performance metrics for single-task baselines

For the sake of clarity, we mind the reader that annual return, cumulative return, annual volatility, maximum drawdown and daily value at risk are expressed as percentage. It is obvious that only the combinations with the *previous label* exiting strategy were successful, especially in the multi-task learning scenario. Although Calmar Ratio is quite high in both experiments with *previous label*, that assumes the risk as the maximum drawdown, which does not represent the probability

metrics	m+b	m+b+n	m+b+n+ta	m+b+n+ta+f	m+b+n+ta+f+p
Annual return	-25.32	-30.45	-25.75	-26.37	467.85
Cumulative returns	-7.27	-8.83	-7.45	-7.56	49.74
Annual volatility	6.3	7.22	7.05	6.77	16.59
Sharpe ratio	-4.69	-5.12	-4.16	-4.52	8.7
Calmar ratio	-3.16	-3.21	-2.85	-3.06	313.03
Stability	0.83	0.93	0.85	0.88	0.93
Max drawdown	-8.02	-9.5	-8.68	-8.56	-1.49
Omega ratio	0.42	0.4	0.46	0.4	7.5
Sortino ratio	-5.41	-5.85	-4.7	-5.09	39.25
Skew	0.09	0.08	-0.44	-0.48	1.26
Kurtosis	2.77	2.58	2.57	2.94	1.8
Tail ratio	0.69	0.7	0.62	0.63	8.19
Daily value at risk	-0.9	-1.05	-1.02	-0.98	-1.47
Alpha	-0.25	-0.3	-0.25	-0.26	4.89
Beta	-0.02	-0.07	-0.06	-0.05	-0.12

Table 6.14: Mean of performance metrics for multi-task baselines

metrics	b	b+n	b+n+ta	b+n+ta+f	b+n+ta+f+p
Annual return	11.98	6.94	8.14	6.4	126.62
Cumulative returns	3.6	2.16	2.46	1.95	15.45
Annual volatility	0.92	0.53	0.81	1.18	3.93
Sharpe ratio	2.58	2.0	2.32	1.16	2.5
Calmar ratio	1.43	0.39	0.41	0.27	88.31
Stability	0.35	0.05	0.19	0.09	0.1
Max drawdown	3.23	1.63	2.06	1.8	0.55
Omega ratio	0.28	0.15	0.2	0.1	1.65
Sortino ratio	2.53	1.53	1.87	1.07	11.94
Skew	0.8	0.65	0.6	0.26	0.39
Kurtosis	2.73	0.9	0.74	0.58	1.87
Tail ratio	0.3	0.28	0.24	0.16	1.56
Daily value at risk	0.19	0.07	0.12	0.17	0.31
Alpha	0.12	0.07	0.08	0.06	1.27
Beta	0.03	0.02	0.02	0.02	0.04

Table 6.15: Standard deviation of performance metrics for single-task baselines

distribution of the returns and might be misled by outliers. Sharpe, Omega and ratio show good performance, especially the Sharpe for the single-task learning one. Omega is good for both. A skewness indicator close to zero states that return distribution is close to be normal. Kurtosis below 3 is typical of a platyokurtosis. Alpha is more relevant (0.53) in the multi task scenario, while in both strategies

metrics	m+b	m+b+n	m+b+n+ta	m+b+n+ta+f	m+b+n+ta+f+p
Annual return	8.0	4.64	10.34	7.29	313.72
Cumulative returns	2.57	1.56	3.26	2.3	25.41
Annual volatility	1.1	0.87	0.89	1.0	4.99
Sharpe ratio	1.43	1.17	1.61	1.24	2.47
Calmar ratio	0.31	0.11	0.49	0.19	208.67
Stability	0.11	0.05	0.18	0.08	0.09
Max drawdown	2.24	1.37	2.51	2.1	0.23
Omega ratio	0.1	0.06	0.1	0.08	2.2
Sortino ratio	1.2	1.22	1.59	1.18	15.96
Skew	0.62	0.54	0.4	0.6	0.45
Kurtosis	1.58	0.78	2.13	1.49	1.75
Tail ratio	0.21	0.2	0.22	0.24	3.05
Daily value at risk	0.16	0.13	0.17	0.15	0.37
Alpha	0.08	0.05	0.1	0.07	3.3
Beta	0.04	0.05	0.08	0.07	0.06

Table 6.16: Standard deviation of performance metrics for multi-task baselines

beta is negative, which means that the portfolio value is negatively correlated with the Nasdaq index. The *previous true label* exiting strategies gives good results: not trusting the neural network that has given us the predictions prevents us from keeping the position were the probability distribution in the test set does not reflect the one registered by the network in the training set.

6.5.4 Benchmark against the AI4Finance paper

We have then decided to deploy our Nasdaq dataset with one of the state-of-art trading system strategy based on Deep Reinforcement Learning, made by Yang et al. [5]. Here we show the same benchmark indexes. Testing period is from the 24th of June 2019 until the 23rd of October 2019.

metrics	value
Annual return	5.0
Cumulative returns	1.6
Annual volatility	16.7
Sharpe ratio	0.38
Calmar ratio	0.65
Stability	0.06
Max drawdown	-7.7
Omega ratio	1.06
Sortino ratio	0.55
Skew	0.11
Kurtosis	0.27
Tail ratio	0.84
Daily value at risk	-2.1
Alpha	-0.02
Beta	0.95

6.6 Permutation importance calculation

We have decided to give more insights in what our multitask neural network predicted with a permutation importance calculus, following in part the approach given by Parr et al. [39]: for each feature column in both train, validation and test set we shuffle the values in that column, in every stock input, and see if categorical cross-entropy loss decreases or not. A positive difference is actually a sign that a particular experiment did learn in predicting the price, since the permutation loss is greater than the baseline loss. Due to performance constraints, the feature column had had to be permuted once for all the stocks taken into account.

6.6.1 Code

```
import numpy as np
import pandas as pd

def permutation_importances(self, X, y, is_train=True):
    for stock in self.config.subset_names:
        X[stock] = pd.DataFrame(X[stock])
        y[stock] = pd.DataFrame(y[stock])
    ds = tf_dataset(X, y, self.config)
    baseline = self.model.evaluate(ds, verbose=0)
    imp = []
    save = {}
    for col in range(X[stock].shape[1]):
        #random permutation on the column of each stock
        for stock in self.config.subset_names:
            save[stock] = X[stock][:,col].copy()
            X[stock][:,col] = np.random.permutation(X[stock][:,col])
        # build again dataframe
        for stock in self.config.subset_names:
            X[stock] = pd.DataFrame(X[stock])
            y[stock] = pd.DataFrame(y[stock])
        ds = tf_dataset(X, y, self.config)
        m = self.model.evaluate(ds, verbose=1)
        importances = [baseline[i]-m[i] for i in range(len(baseline))
    ]
        imp.append(importances)
        for stock in self.config.subset_names:
            X[stock][:,col] = save[stock]
    imp = np.array(imp)
    return imp
```

6.6.2 Training set

Among those that had a positive importance without technical features, where permutating a column increased the loss, there are four experiments with only the open price feature, which are those with the GRU recurrent cells (figures 6.83, 6.93, 6.93, 6.95, 6.97. If we consider experiments with all features, just figure 6.88 shows a positive importance in all features, but news sentiment (VADER column) is negligible compared to the other features. Even in 6.83 open price has a positive impact, but not VADER. Experiments with a good permutation importance regarding the news feature was just 6.90.

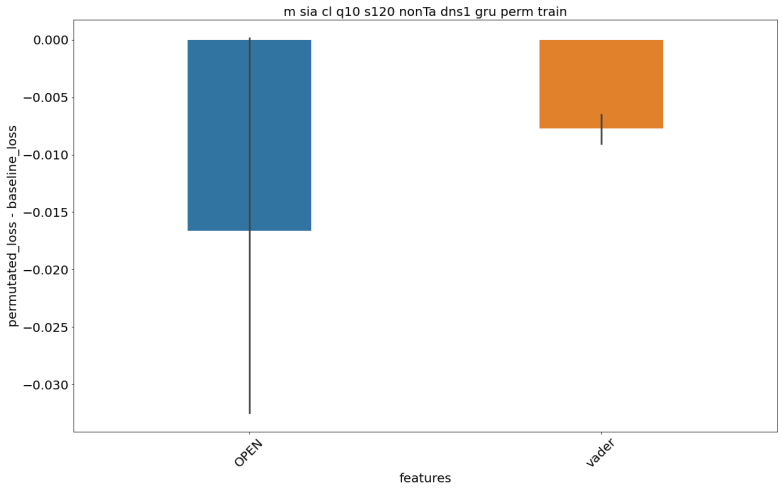


Figure 6.81: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **one dense layer**.

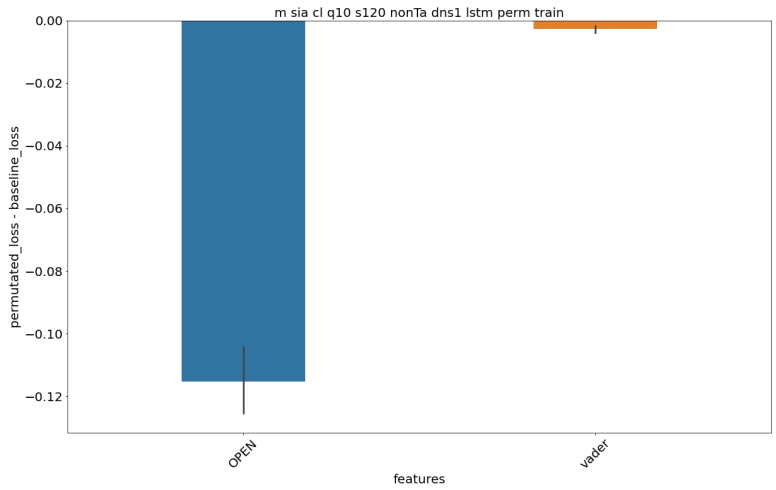


Figure 6.82: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **one dense layer**.

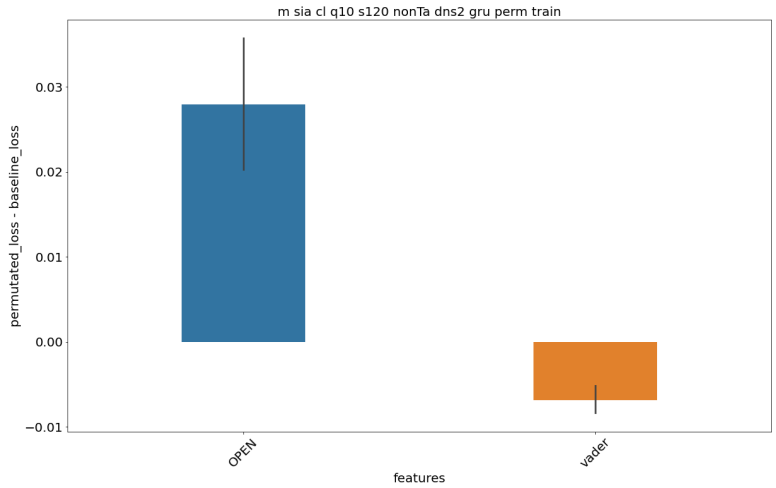


Figure 6.83: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **two dense layers**.

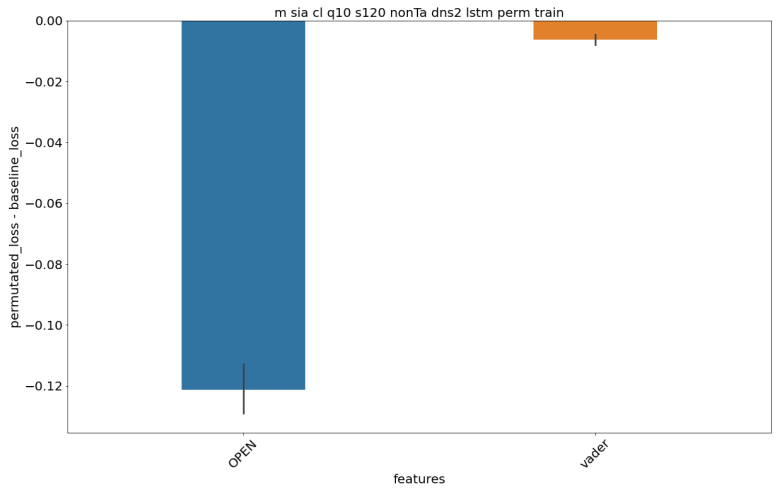


Figure 6.84: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **two dense layers**.

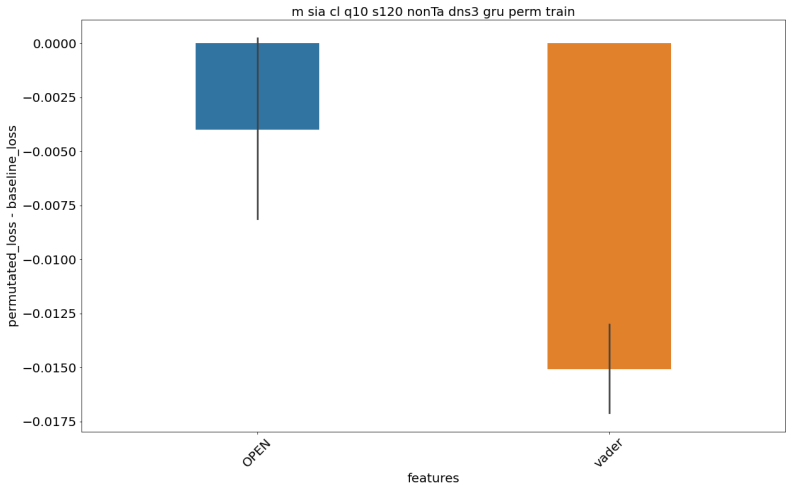


Figure 6.85: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **three dense layers**.

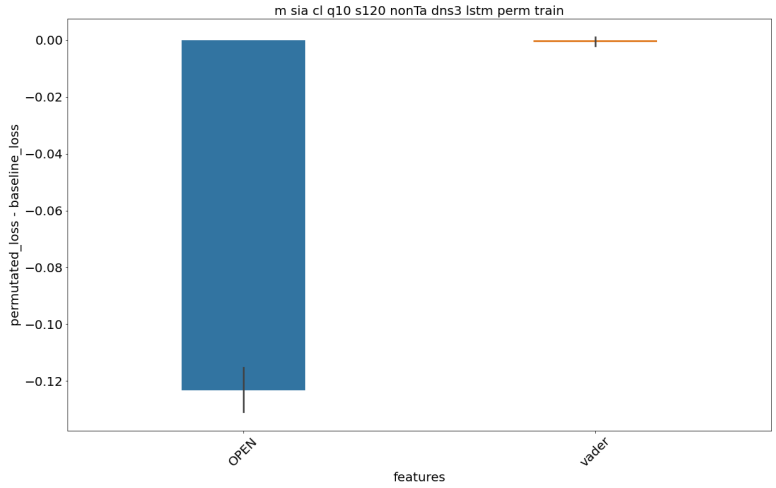


Figure 6.86: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **three dense layers**.

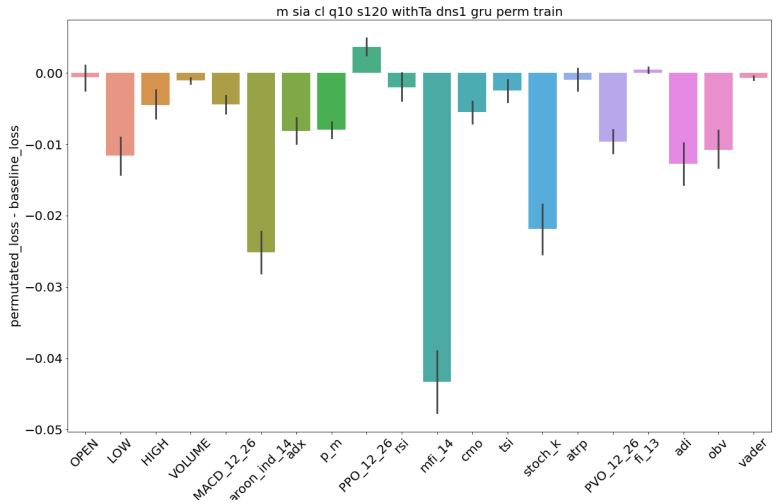


Figure 6.87: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **one dense layer**.

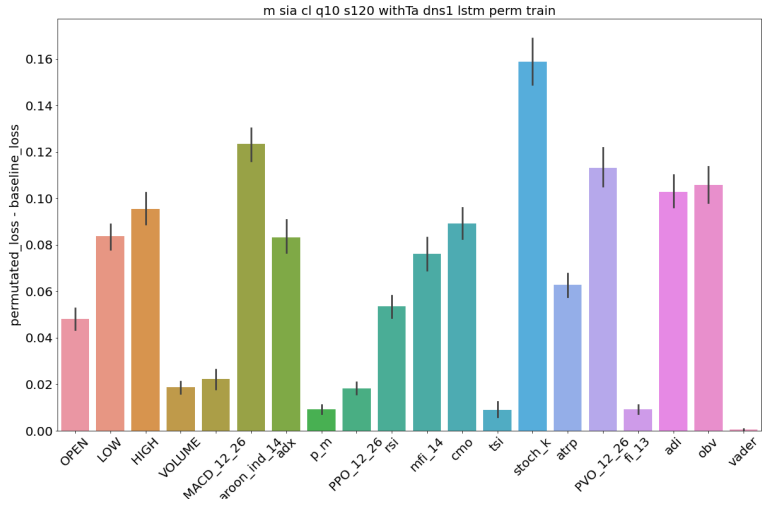


Figure 6.88: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **one dense layer**.

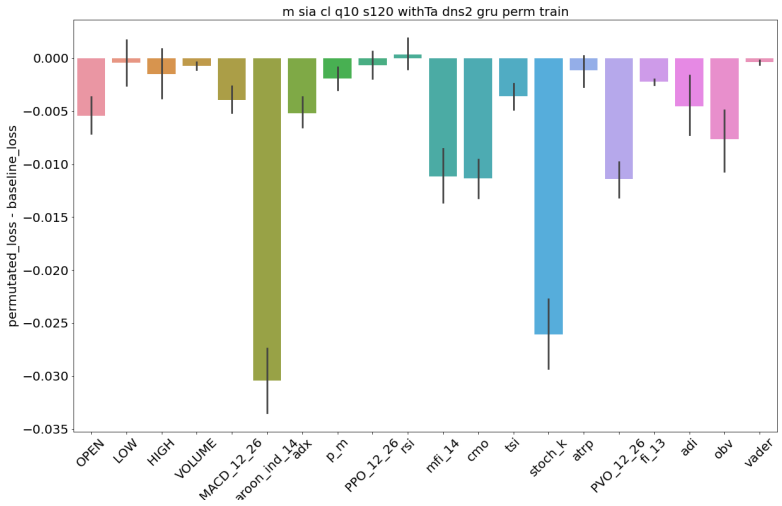


Figure 6.89: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **two dense layers**.

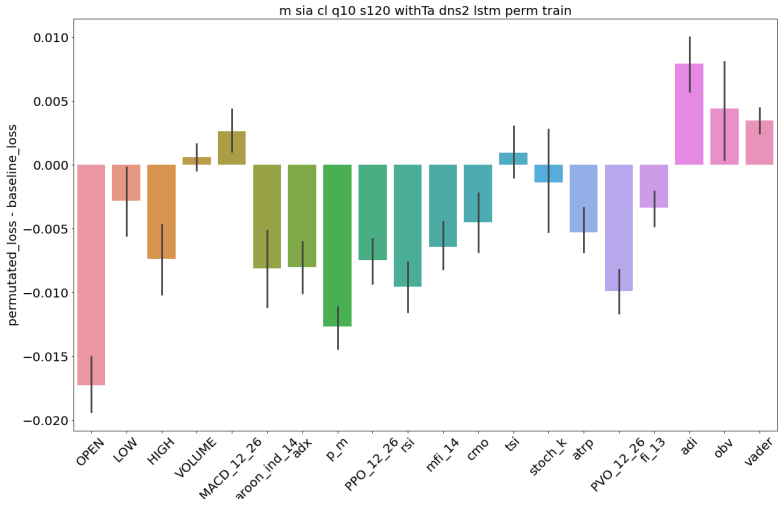


Figure 6.90: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **two dense layers**.

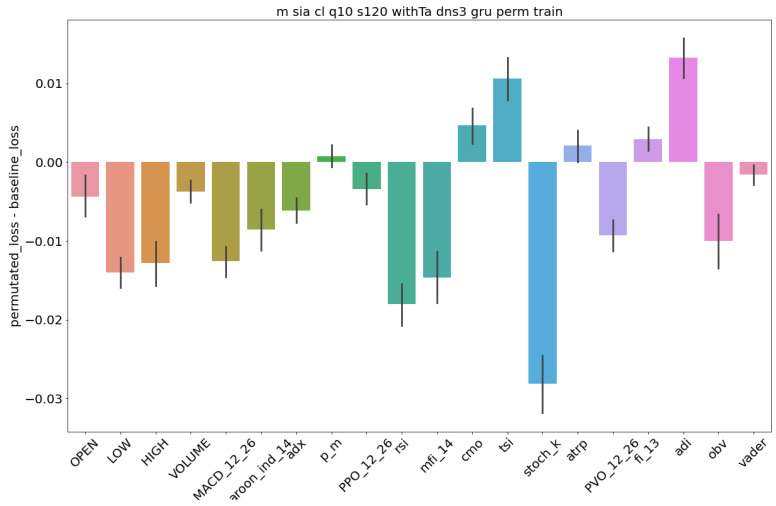


Figure 6.91: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **three dense layers**.

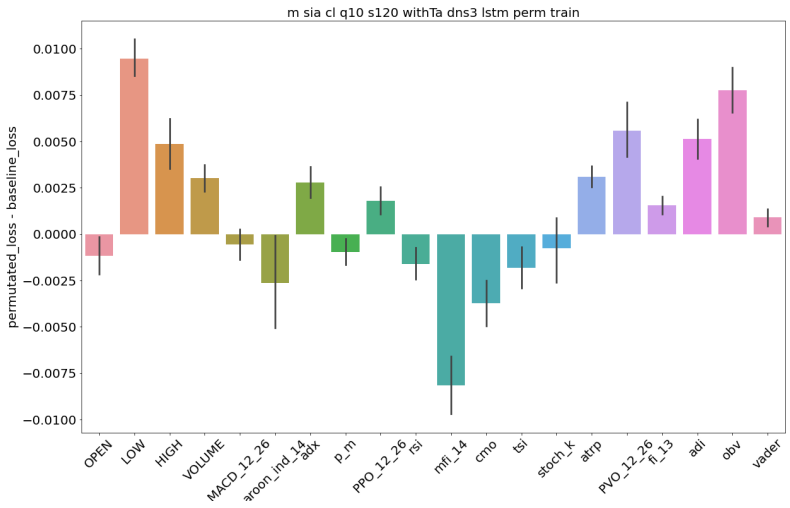


Figure 6.92: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **three dense layers**.

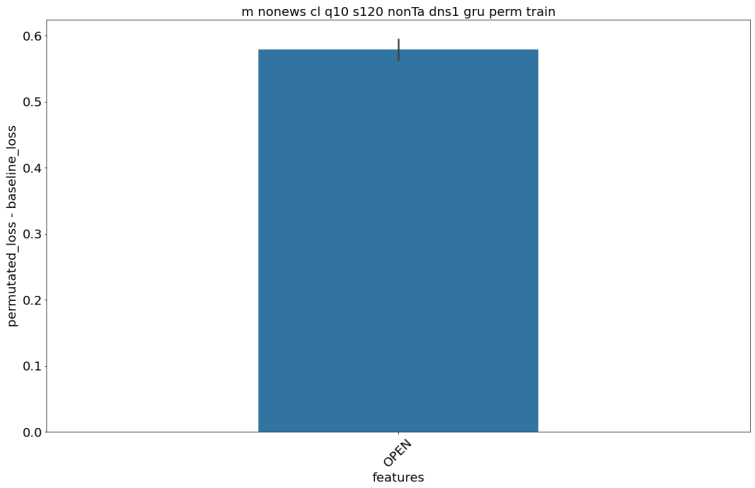


Figure 6.93: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **one dense layer**.

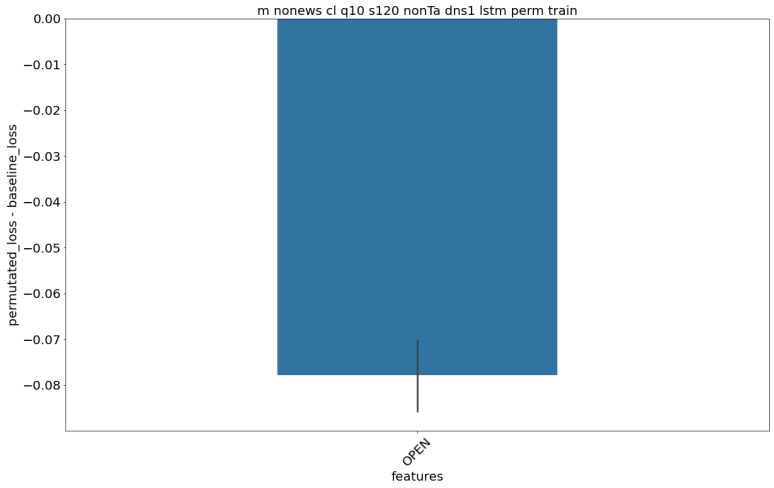


Figure 6.94: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **one dense layer**.

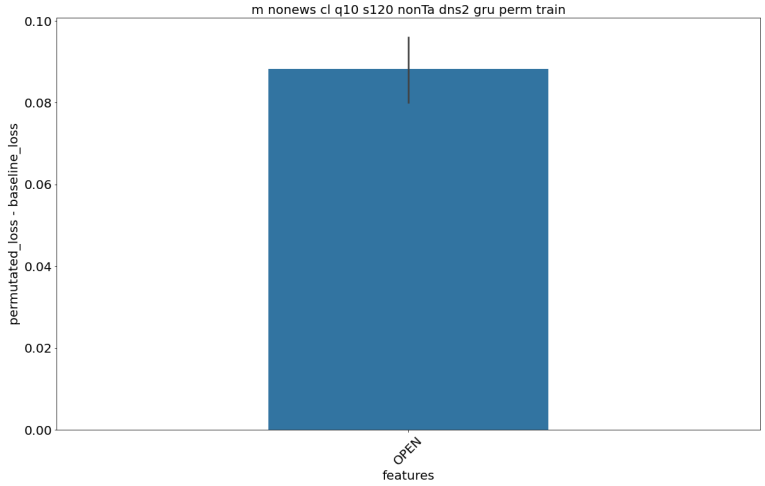


Figure 6.95: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **two dense layers**.

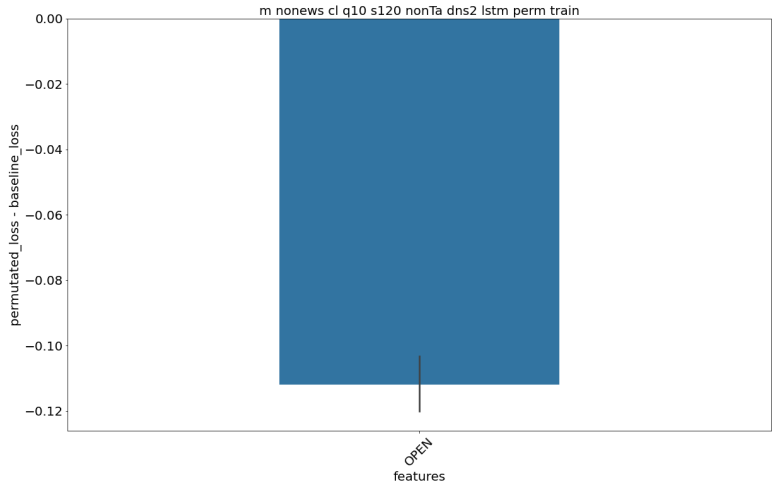


Figure 6.96: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **two dense layers**.

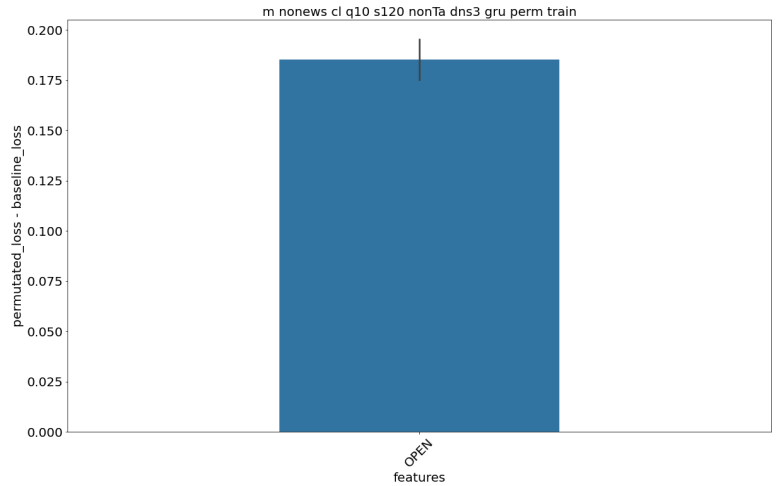


Figure 6.97: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **three dense layers**.

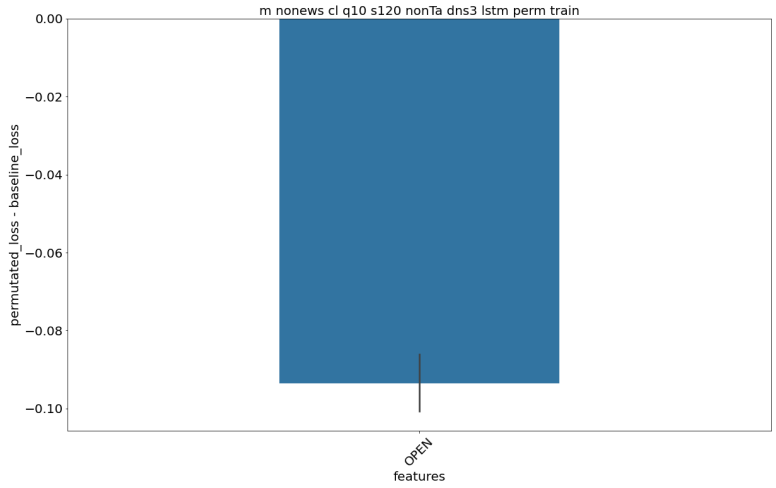


Figure 6.98: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **three dense layers**.

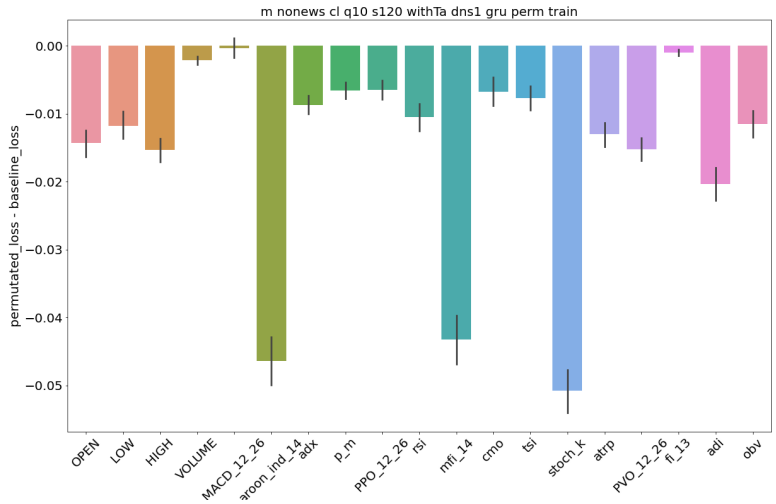


Figure 6.99: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **one dense layer**.

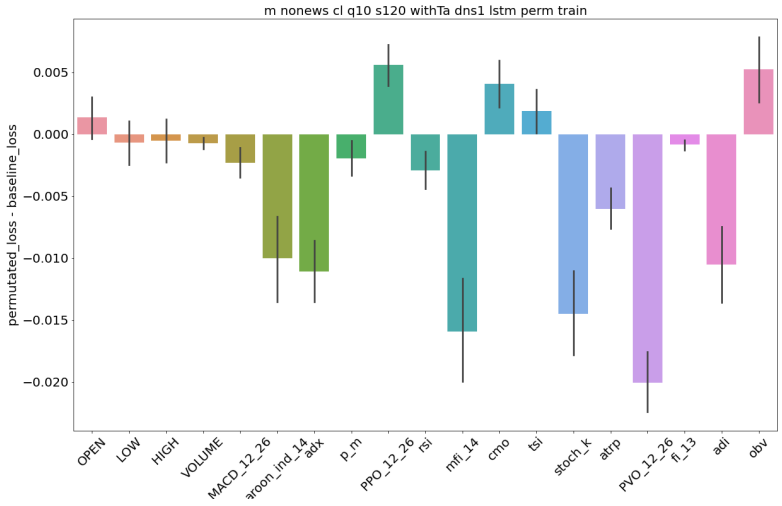


Figure 6.100: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **one dense layer**.

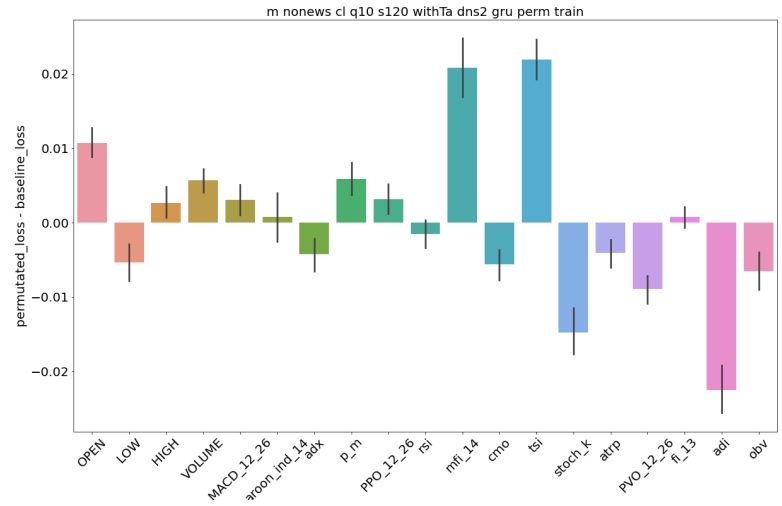


Figure 6.101: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **two dense layers**.

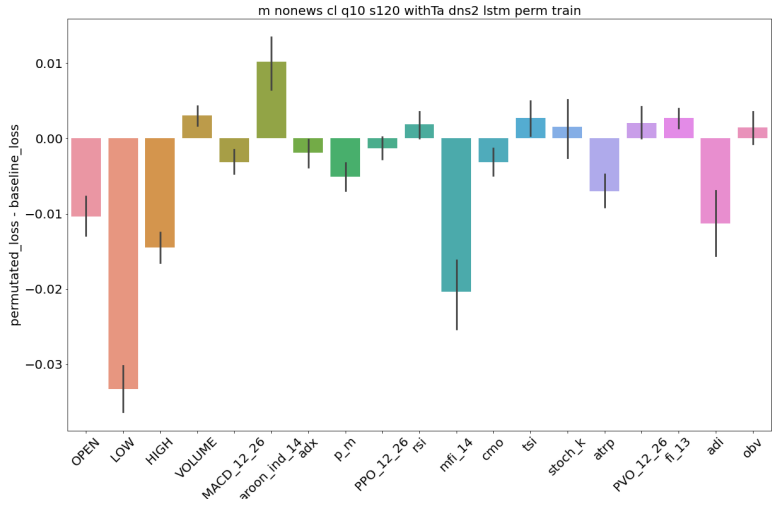


Figure 6.102: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **two dense layers**.

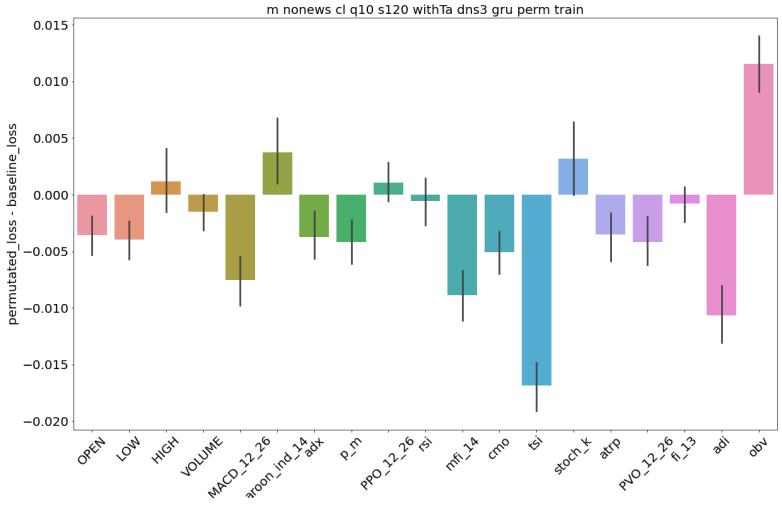


Figure 6.103: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **three dense layers**.

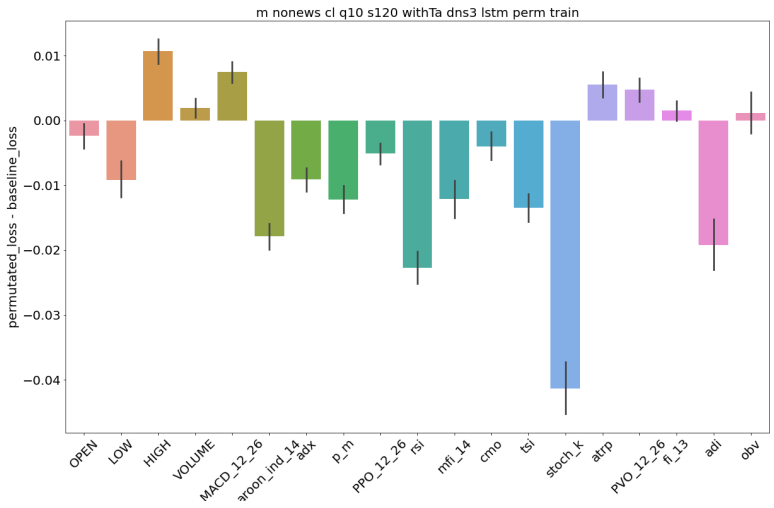


Figure 6.104: Training permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **three dense layers**.

6.6.3 Validation set

Among those without technical features, vader does not have a positive impact on the predictions. We further mention 6.123, 6.126, 6.114, 6.115 all with technical features included. Experiments 6.114 and 6.115 have also the VADER feature, but its importance is negligible compared to other features.

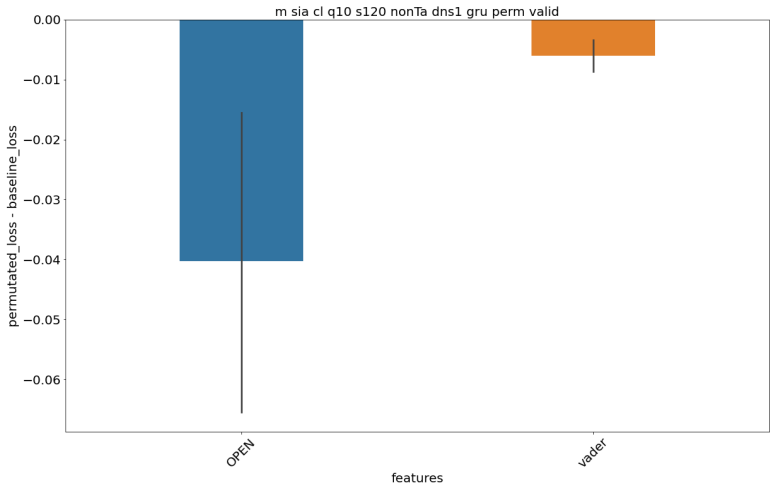


Figure 6.105: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **one dense layer**.

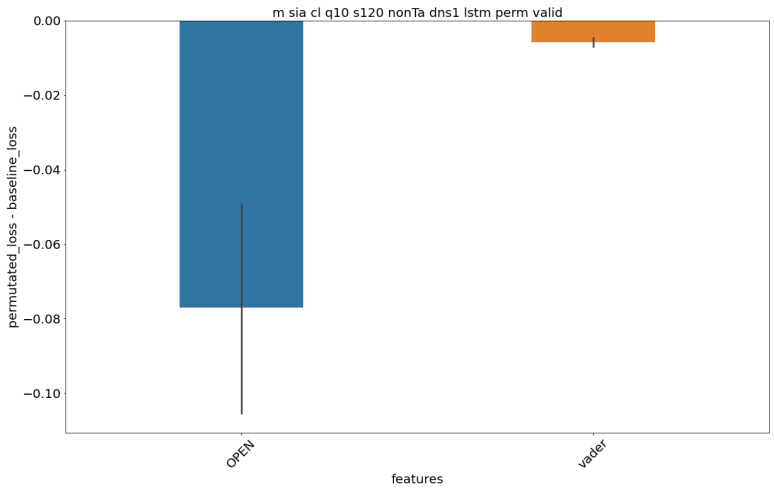


Figure 6.106: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **one dense layer**.

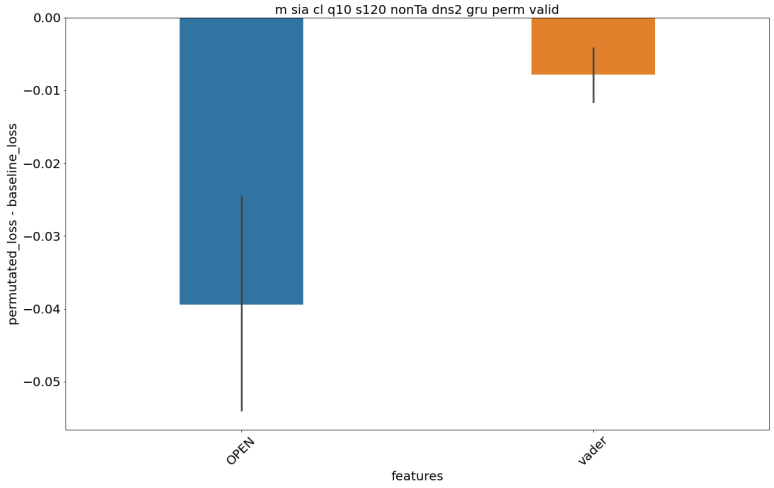


Figure 6.107: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **two dense layers**.

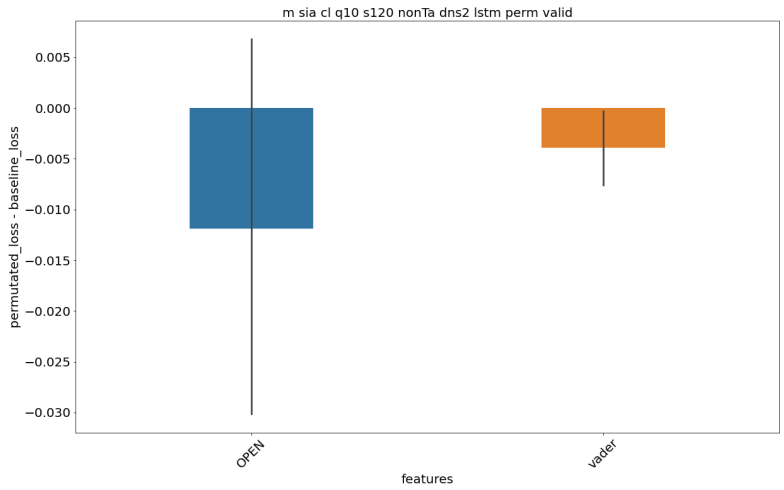


Figure 6.108: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **two dense layers**.

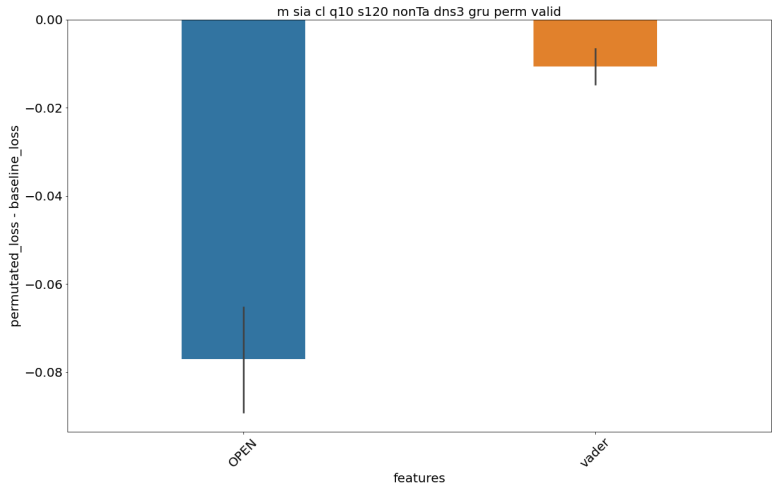


Figure 6.109: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **three dense layers**.

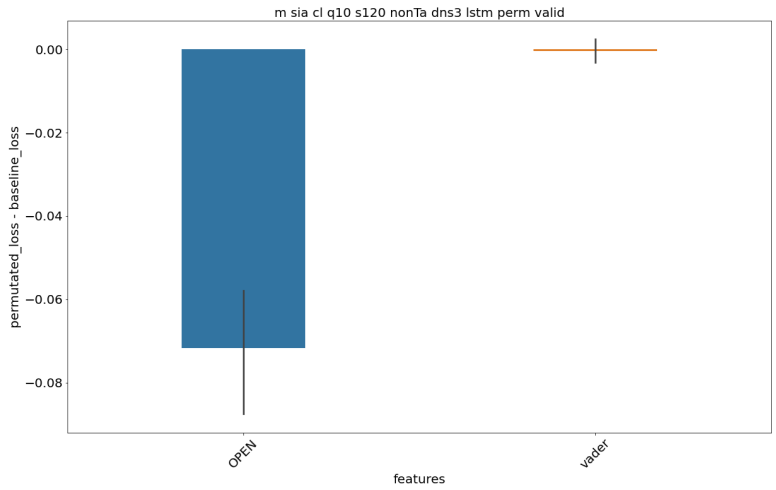


Figure 6.110: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **three dense layers**.

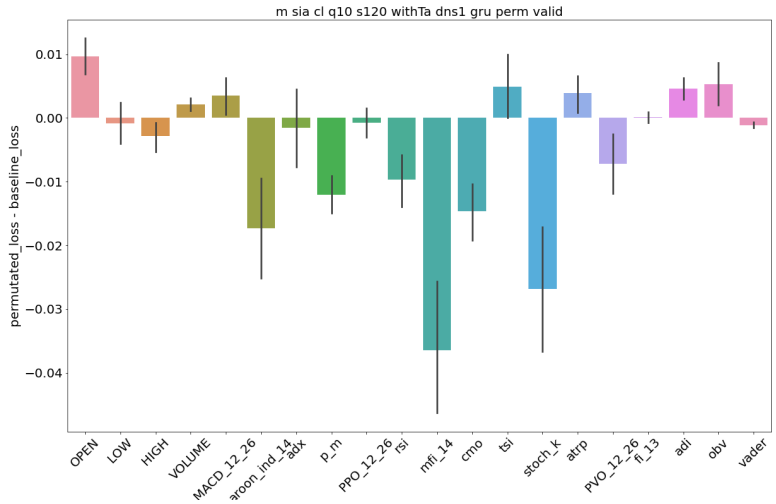


Figure 6.111: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **one dense layer**.

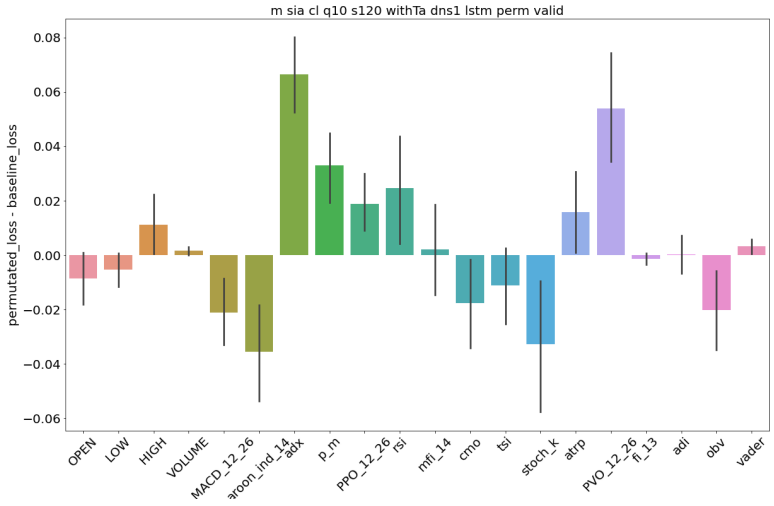


Figure 6.112: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **one dense layer**.

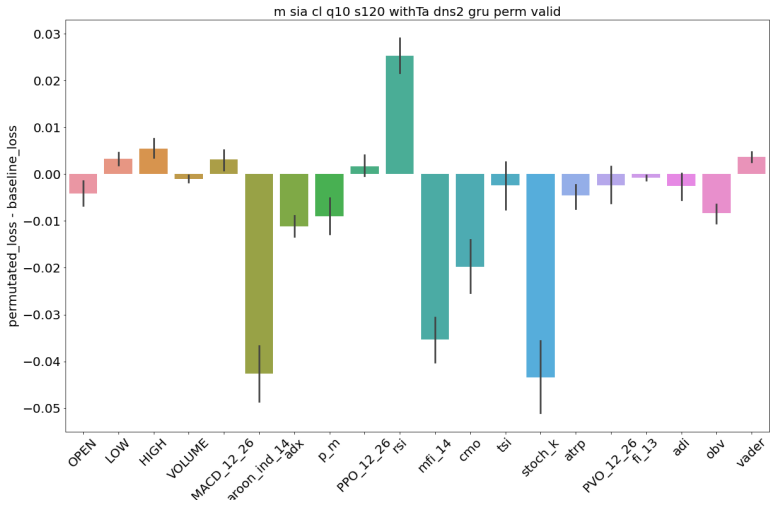


Figure 6.113: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **two dense layers**.

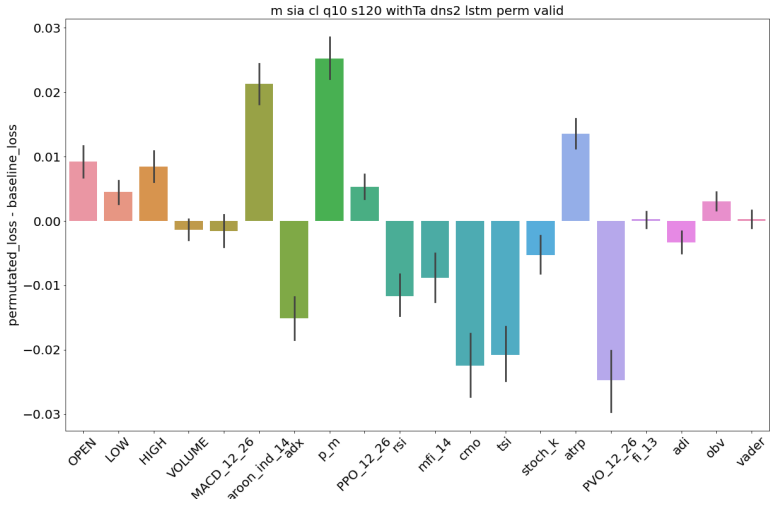


Figure 6.114: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **two dense layers**.

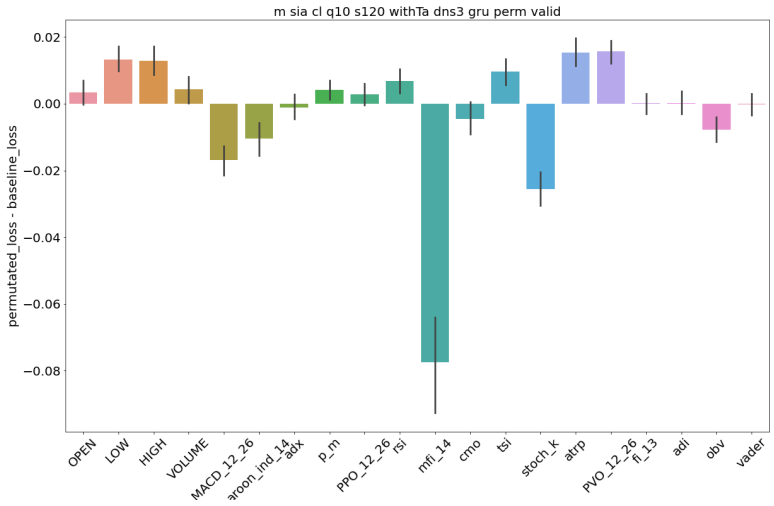


Figure 6.115: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **three dense layers**.

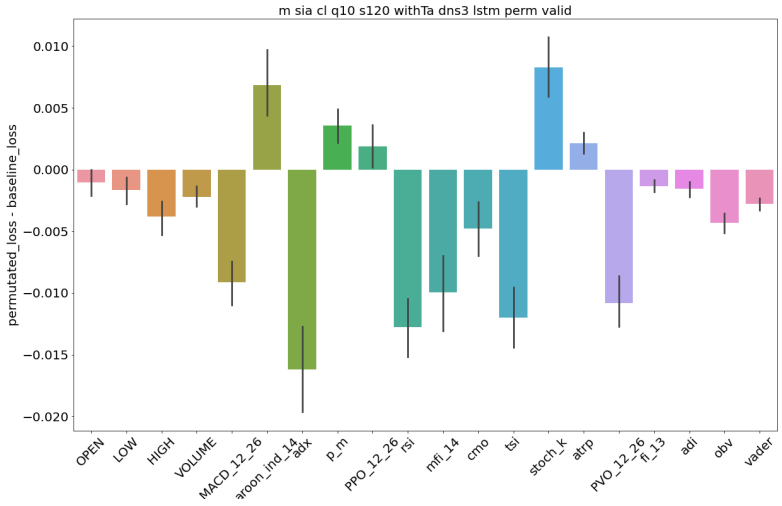


Figure 6.116: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **three dense layers**.

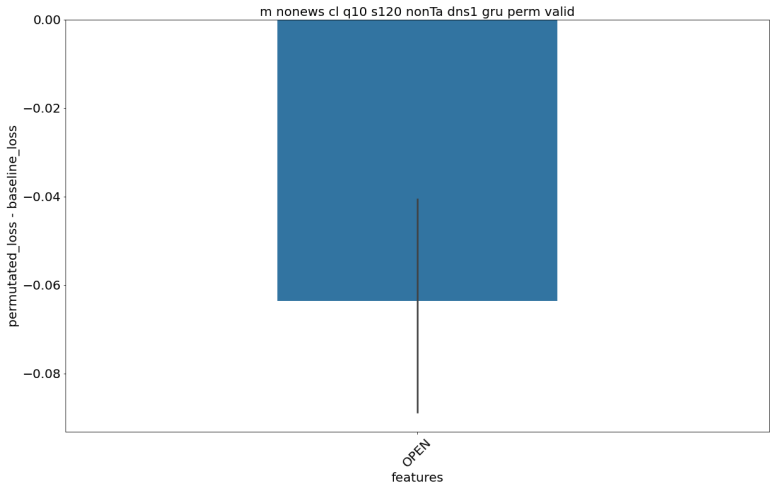


Figure 6.117: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **one dense layer**.

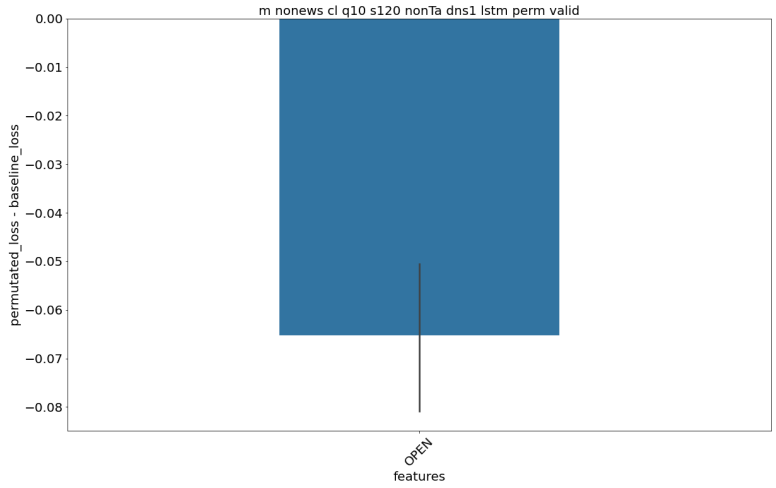


Figure 6.118: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **one dense layer**.

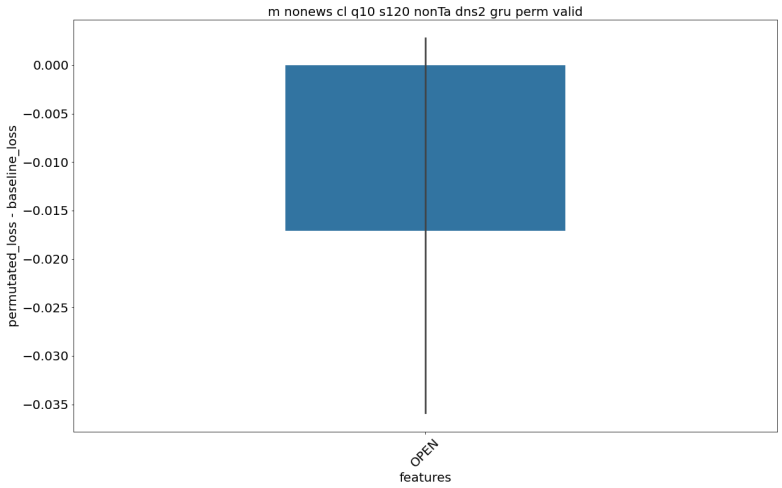


Figure 6.119: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **two dense layers**.

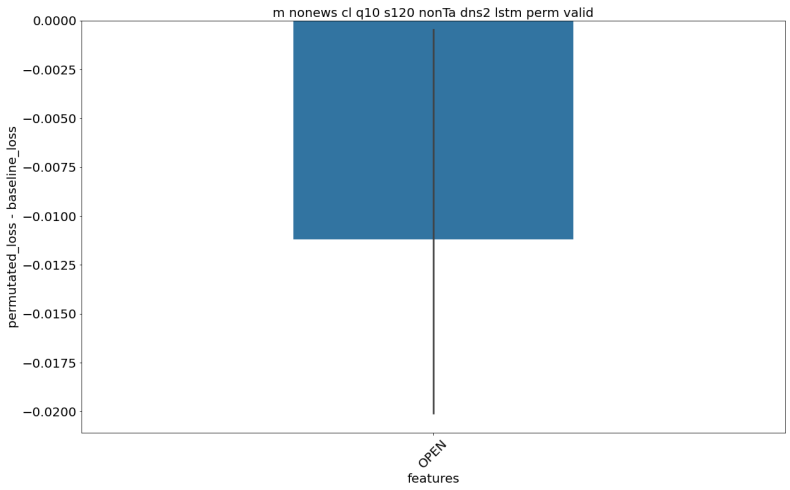


Figure 6.120: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **two dense layers**.

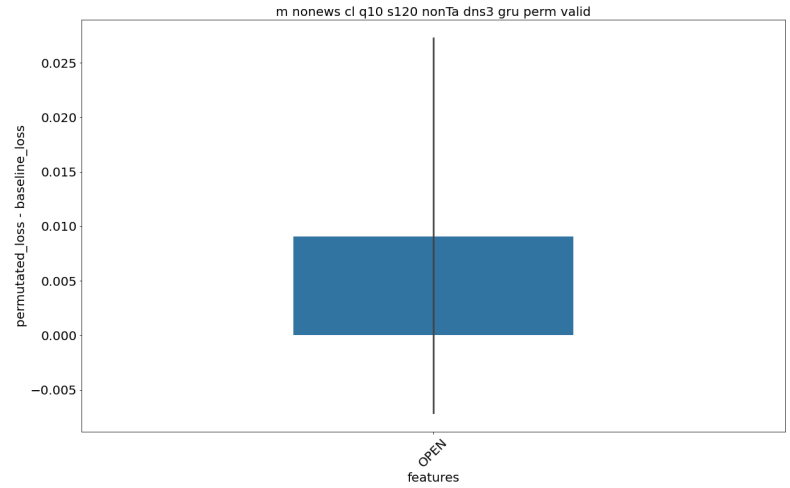


Figure 6.121: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **three dense layers**.

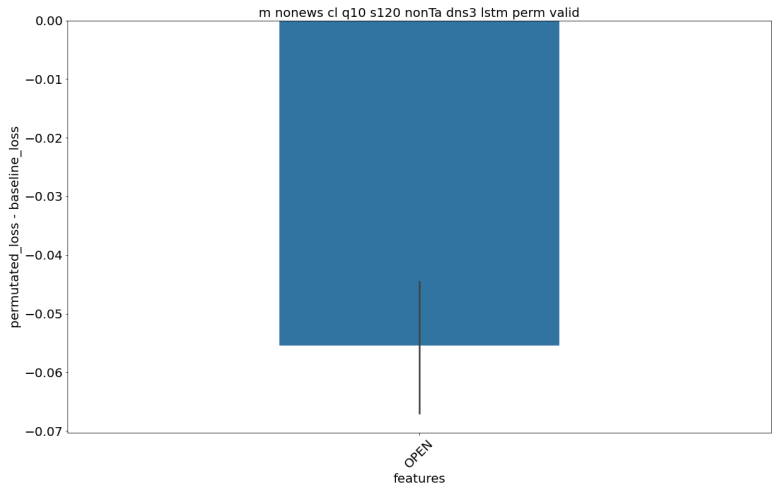


Figure 6.122: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **three dense layers**.

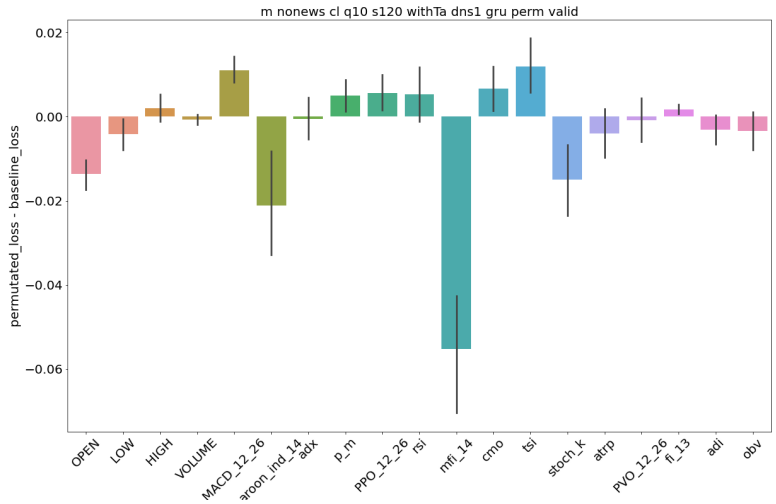


Figure 6.123: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **one dense layer**.

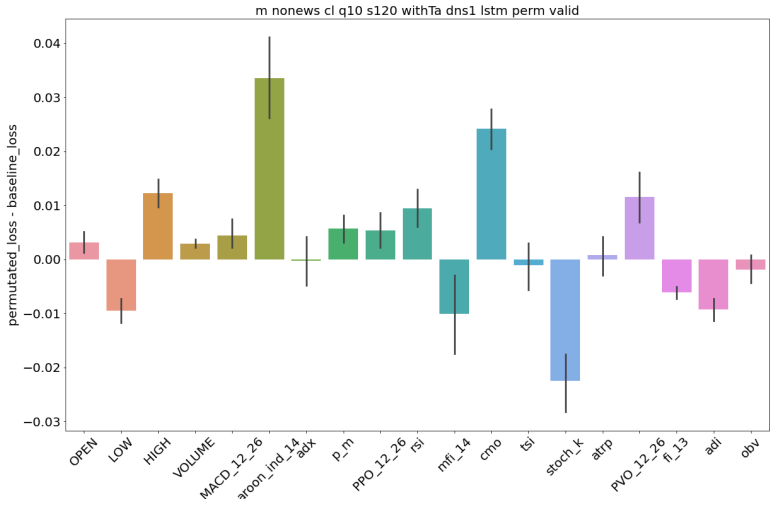


Figure 6.124: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **one dense layer**.

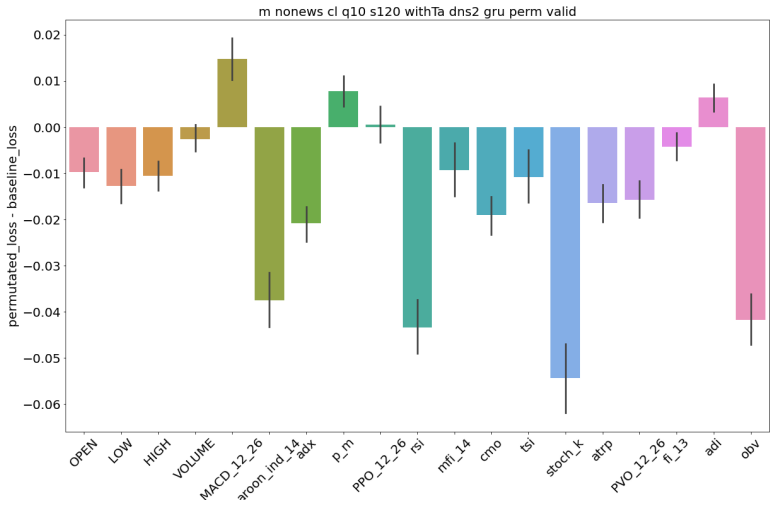


Figure 6.125: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **two dense layers**.

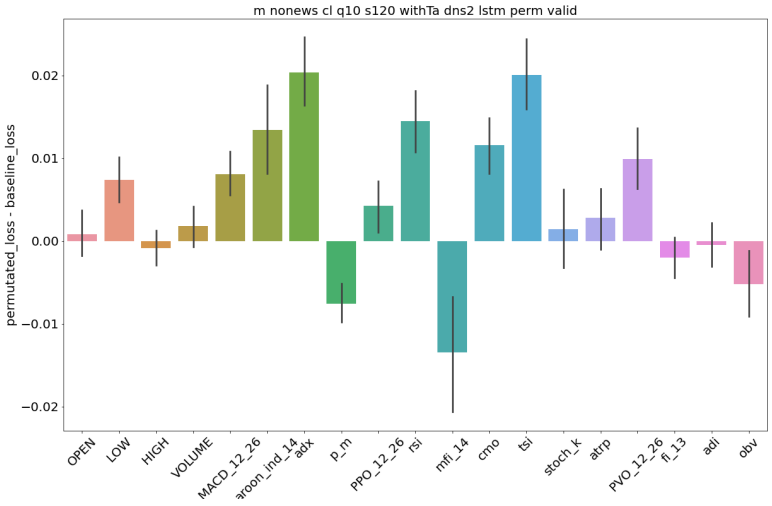


Figure 6.126: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **two dense layers**.

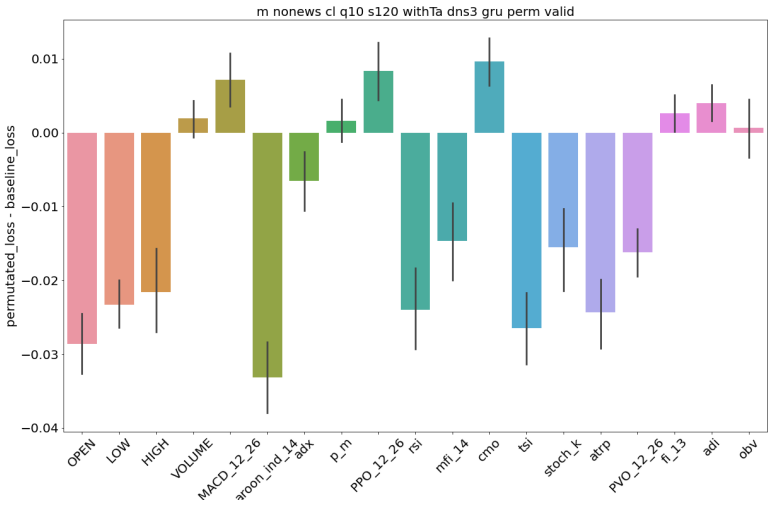


Figure 6.127: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **three dense layers**.

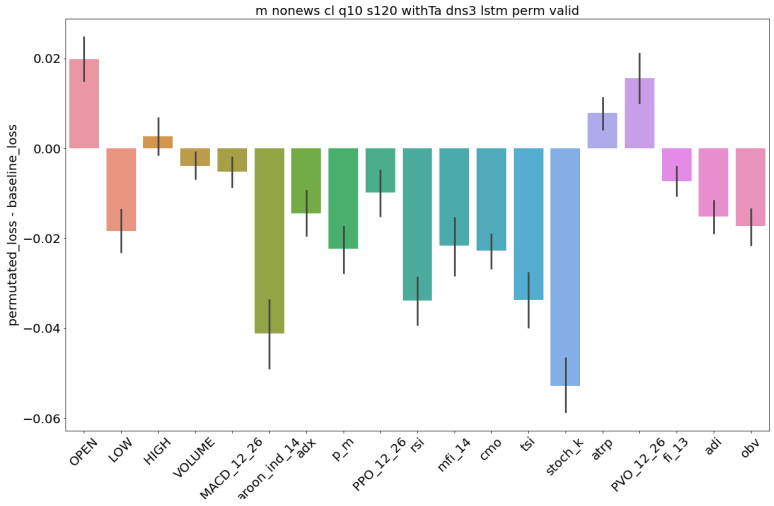


Figure 6.128: Validation permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **three dense layers**.

6.6.4 Test set

Permutation importance in test set highlights three experiments without technical and VADER features nonews (6.143, 6.145, 6.142). Where only VADER is included (6.129, 6.131, 6.132) just in 6.131 VADER has positive importance greater than the open price. Where vader and technical features are included, either vader importance is negative or positive but negligible in respect to the others.

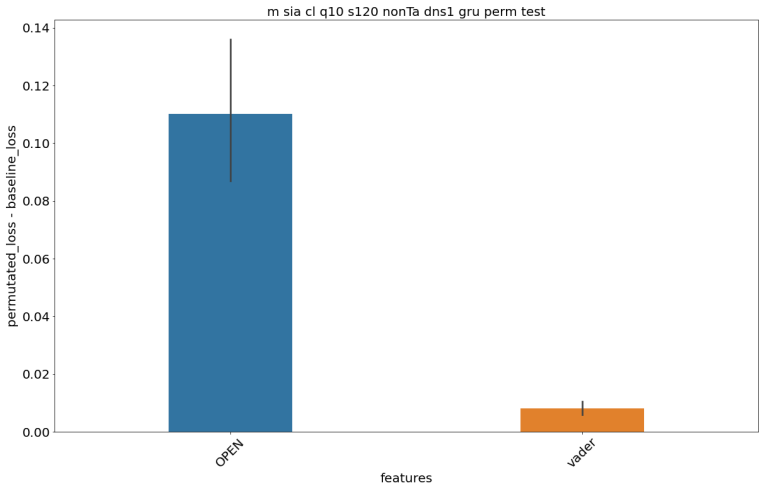


Figure 6.129: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **one dense layer**.

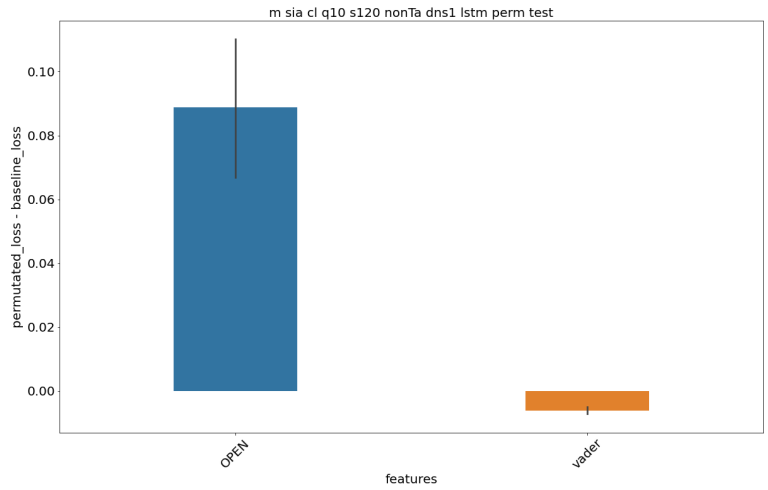


Figure 6.130: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **one dense layer**.

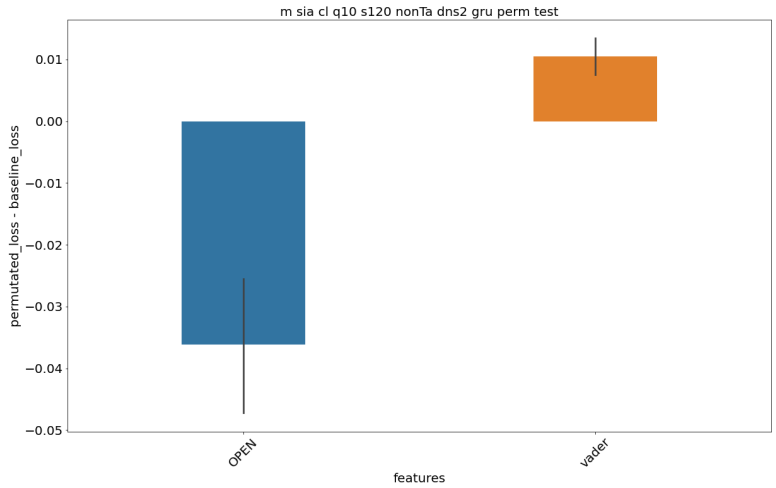


Figure 6.131: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **two dense layers**.

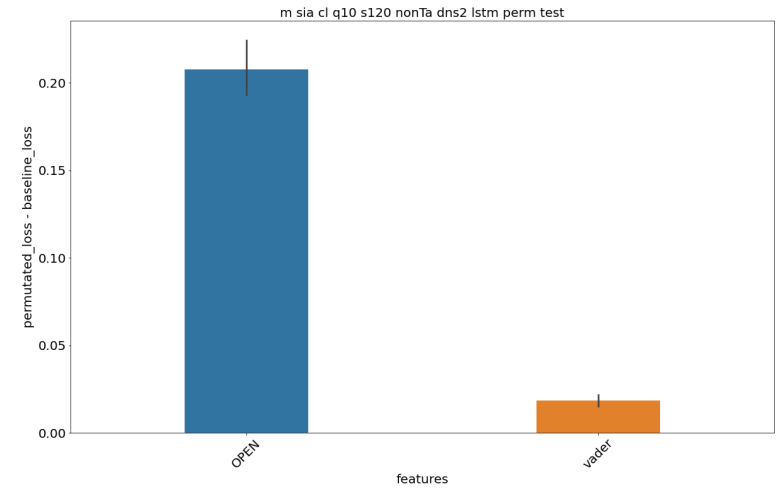


Figure 6.132: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **two dense layers**.

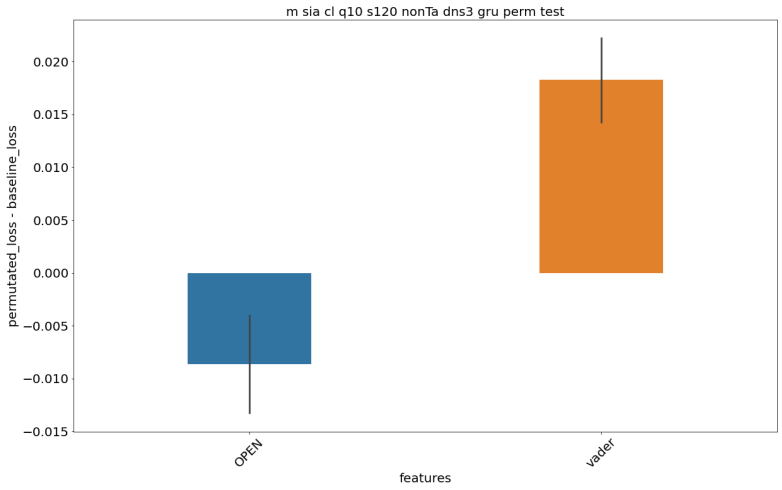


Figure 6.133: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **gru** cells, **three dense layers**.

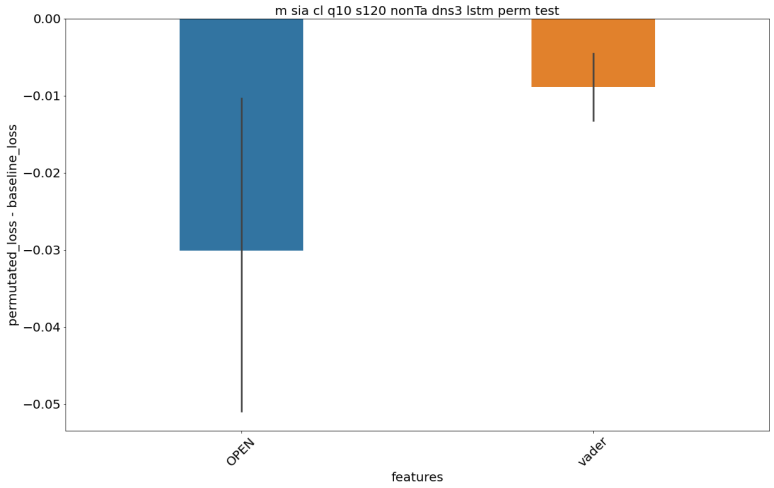


Figure 6.134: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **without technical analysis features**, **lstm** cells, **three dense layers**.

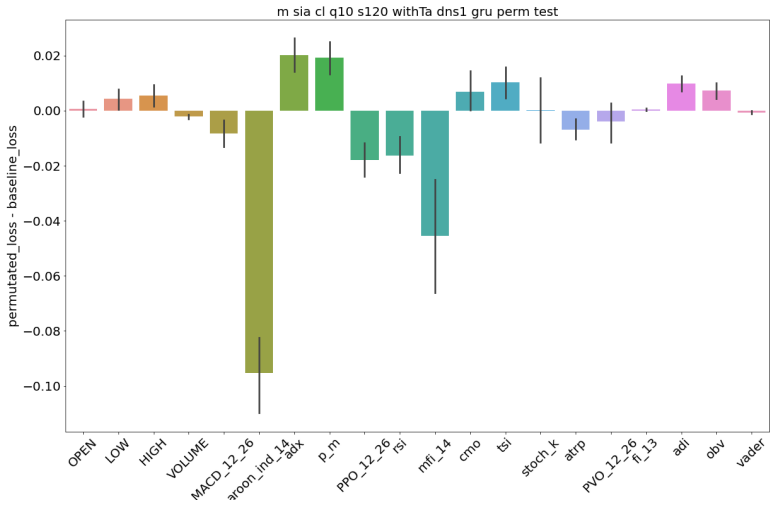


Figure 6.135: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **one dense layer**.

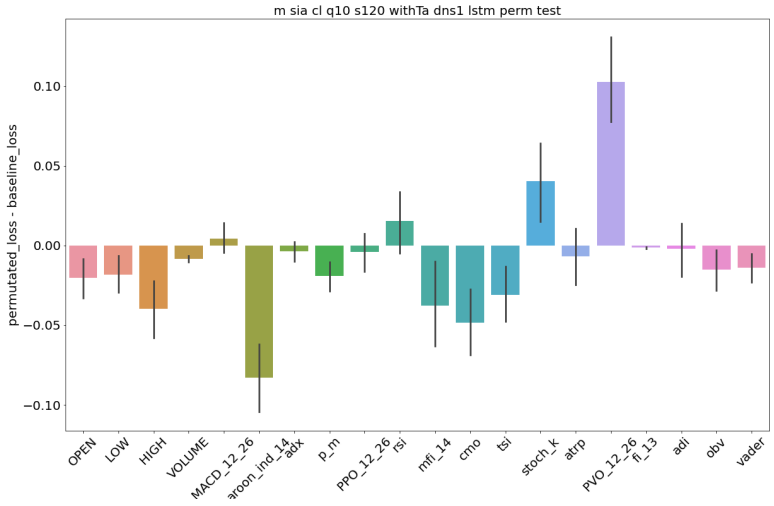


Figure 6.136: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **one dense layer**.

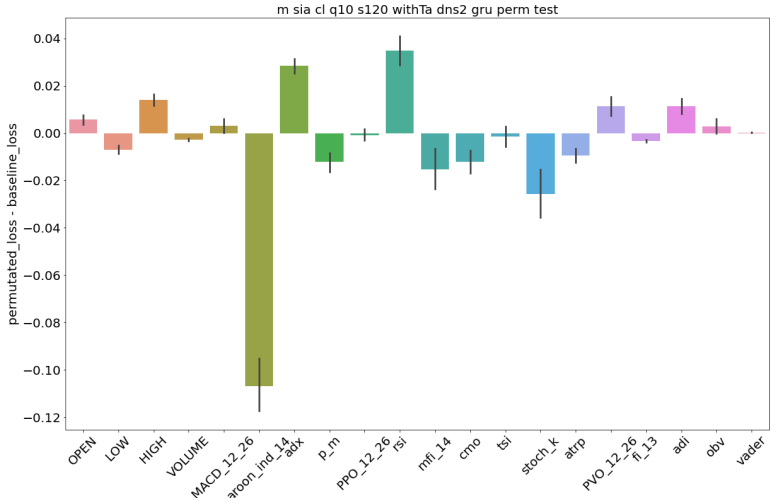


Figure 6.137: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **two dense layers**.

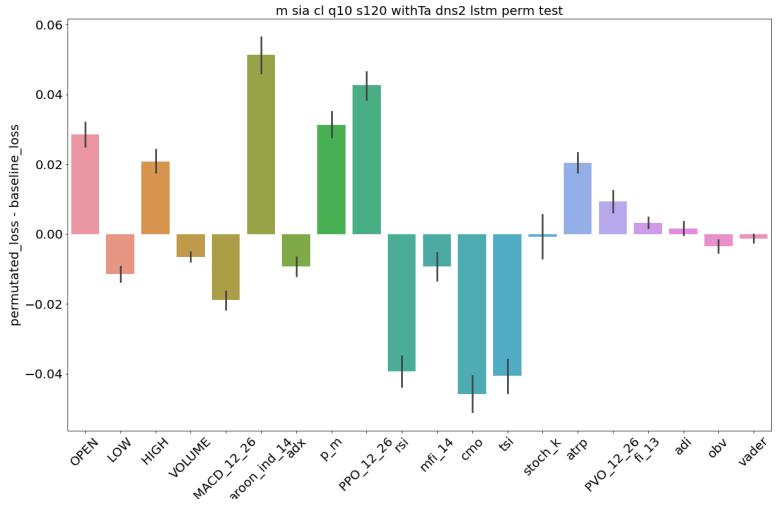


Figure 6.138: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **two dense layers**.

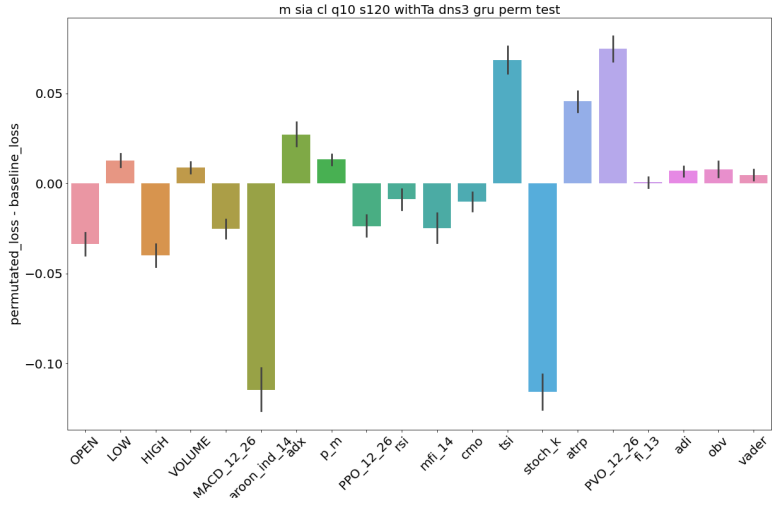


Figure 6.139: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **gru** cells, **three dense layers**.

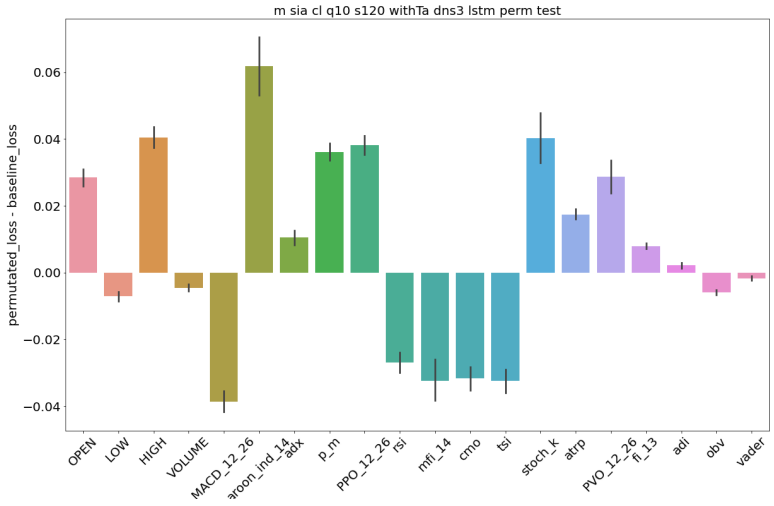


Figure 6.140: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **with Vader**, **with technical analysis features**, **lstm** cells, **three dense layers**.

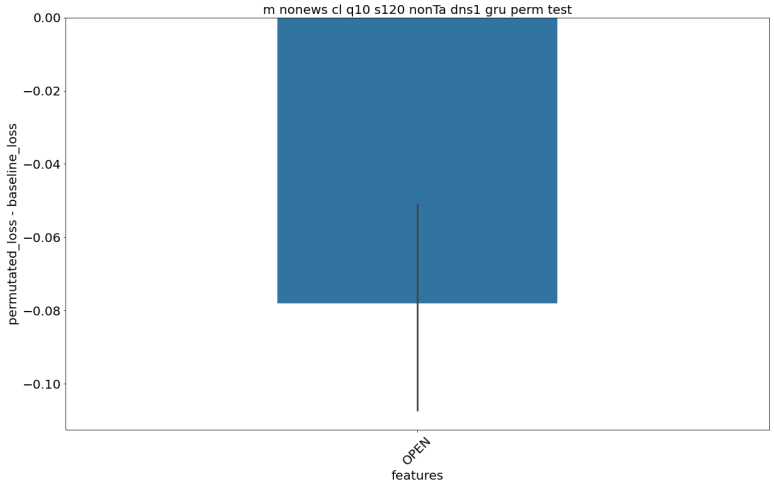


Figure 6.141: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **one dense layer**.

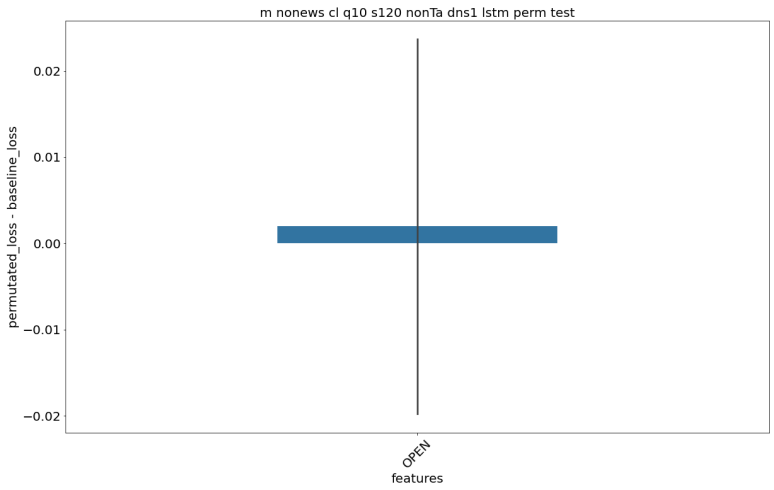


Figure 6.142: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **one dense layer**.

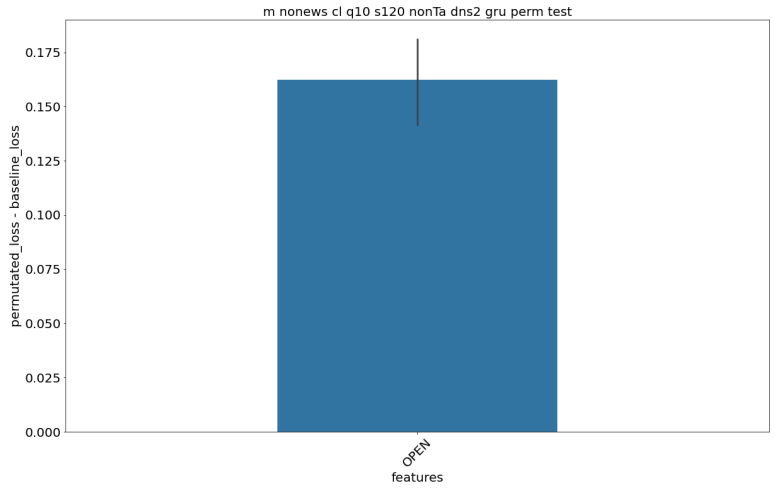


Figure 6.143: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **two dense layers**.

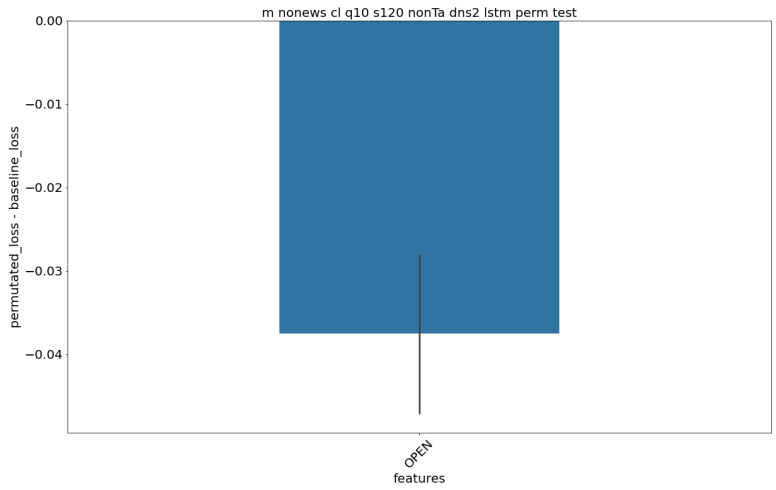


Figure 6.144: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **two dense layers**.

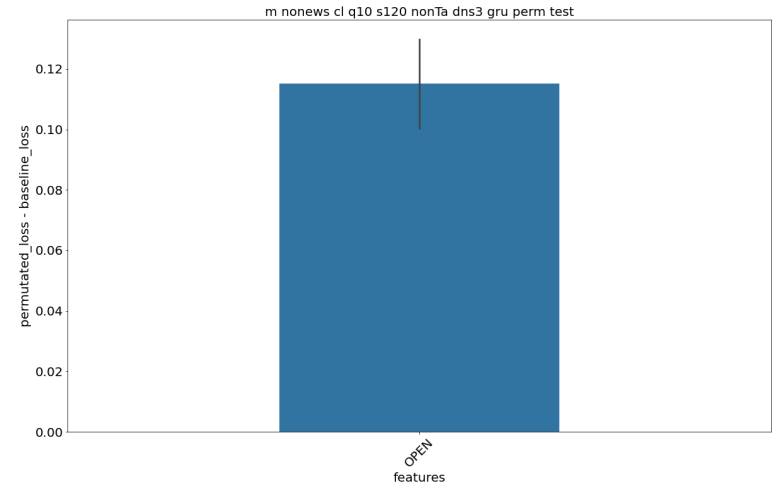


Figure 6.145: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **gru** cells, **three dense layers**.

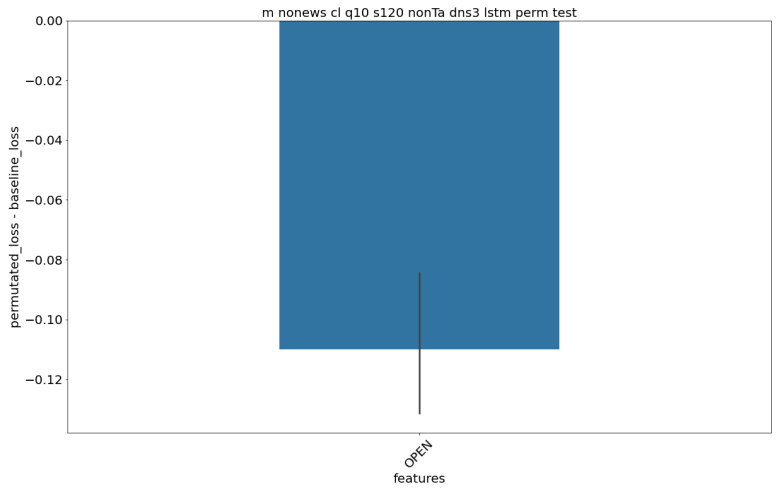


Figure 6.146: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **without technical analysis features**, **lstm** cells, **three dense layers**.

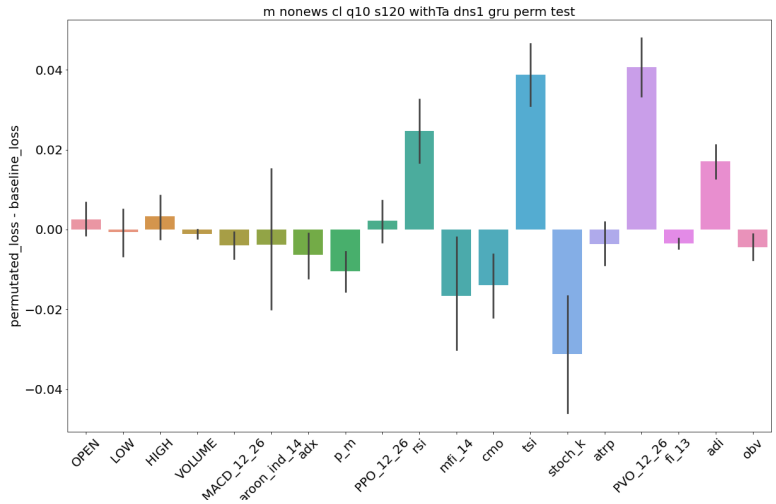


Figure 6.147: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **one dense layer**.

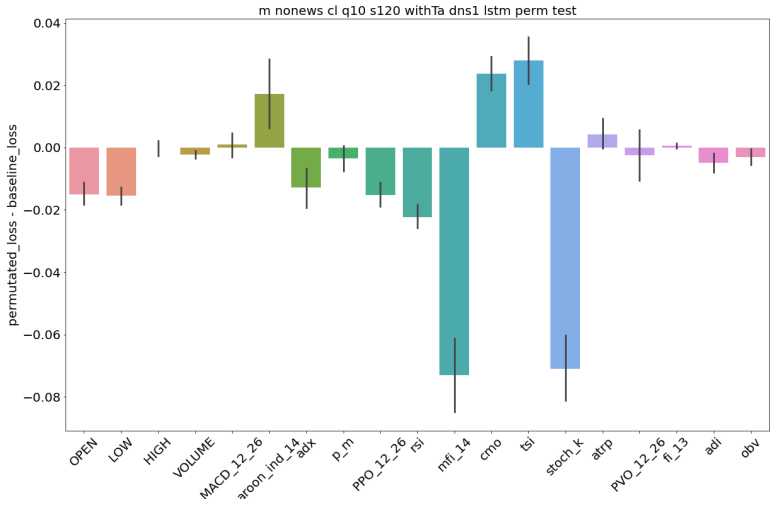


Figure 6.148: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **one dense layer**.

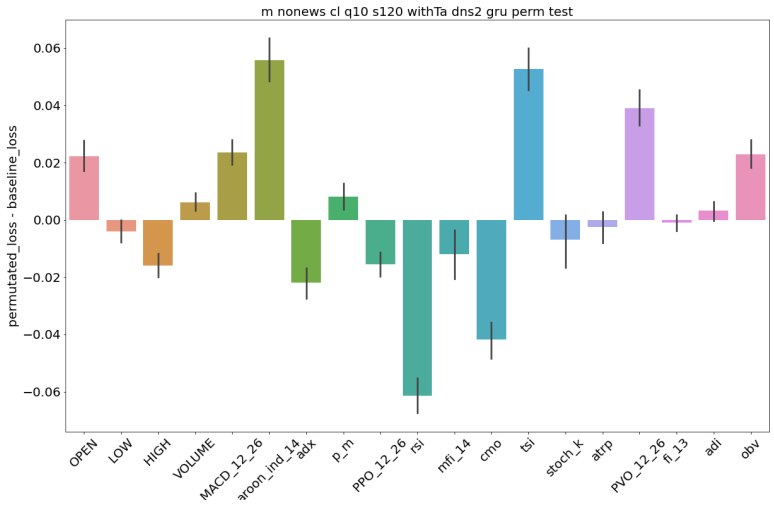


Figure 6.149: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **two dense layers**.

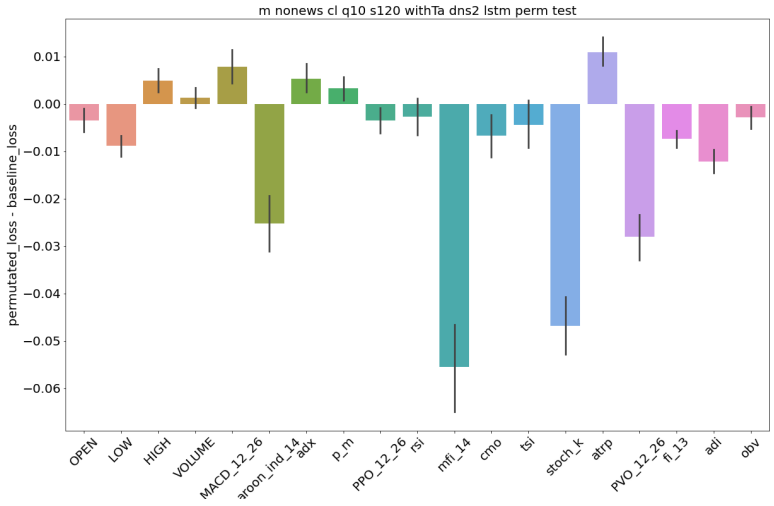


Figure 6.150: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **two dense layers**.

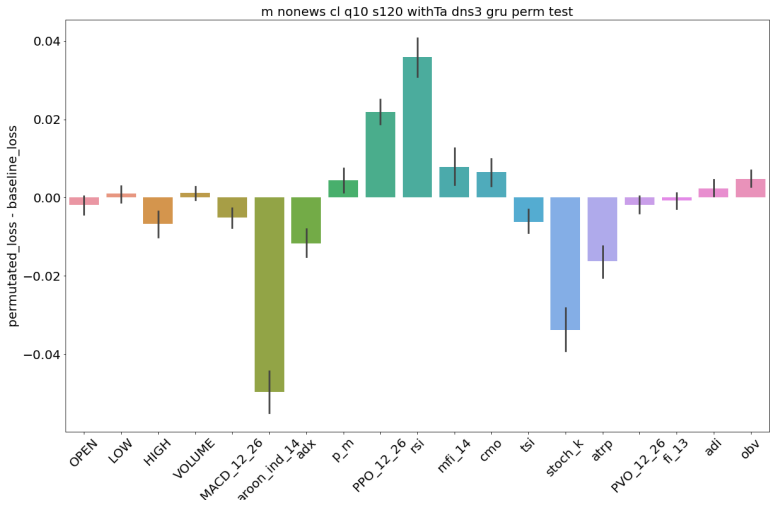


Figure 6.151: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **gru** cells, **three dense layers**.

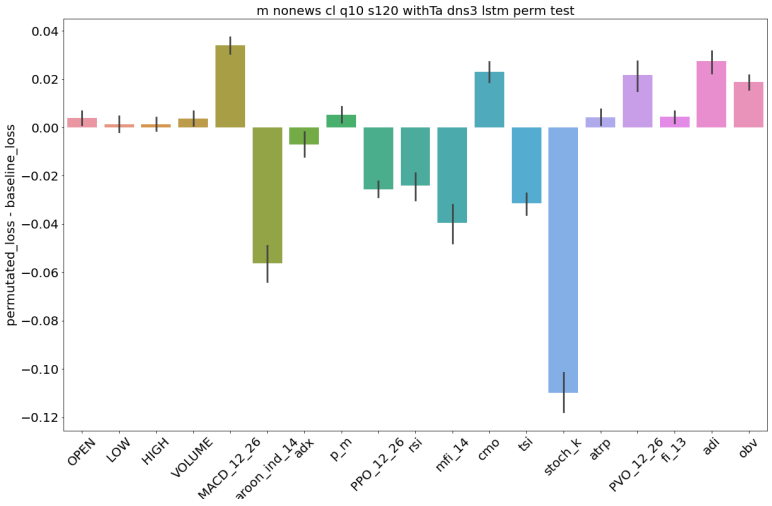


Figure 6.152: Test permutation importance plot for neural network trained with Nasdaq 100 dataset in **classification multi-task** learning mode, **without news feature**, **with technical analysis features**, **lstm** cells, **three dense layers**.

Chapter 7

Conclusions

We have exploited the possibilities of using news data in a wide range of neural network configurations. First we have used different preprocessing methods (GloVe, count vectorizing and VADER) to find the most stable in terms of gradient issues, and we have found the VADER being the most promising. Then with a broader dataset, we have searched different combinations of time granularity and different label bindings, keeping the 120 minutes and 10-90 quantiles, due to less trading signals and less impact on trading costs. The third range of experiments added technical features in input, LSTM cells, a parametrized number of RNN cells and more dense layers. The final trading system grid exploited a wide range of periods in the technical indicators, together with an optional Trend Strength Indicator (ADX) filter and a stop loss criteria based on the previous true label. The results confirm the benefits of using multi-tasking in terms of ROE and volatility. Secondly, we compare the proposed approaches with a state-of-the-art Reinforcement Learning strategy [5]. The classification-based approaches perform consistently better than RL in terms of annual return and the main portfolio ratio, such as Omega and Sharpe. Finally we conducted a permutation importance analysis. It has ruled out the relative importance of the news-related VADER feature: the additional information provided by news data turned out to be negligible compared to the price-related features, as it is not as frequent as the price information: in fact the VADER feature is very sparse. Future work will focus on moving the trading system from a self-made algorithm in python to a more stable trading python framework such as Zipline [40]. This will lead us to test the strategy including a more realistic scenario with slippage, which means simulating the distance between the order and the execution of the trade. Further improvements will be done in tweaking the money put in the order based on the effective accuracy registered in the training and validation sets.

Bibliography

[1] Zexin Hu, Yiqi Zhao, and Matloob Khushi. «A Survey of Forex and Stock Price Prediction Using Deep Learning». In: *Applied System Innovation* 4.1 (2021). ISSN: 2571-5577. DOI: 10.3390/asi4010009. URL: https://www.mdpi.com/2571-5577/4/1/9 (cit. on p. 1).

[2] Gourav Kumar, Sanjeev Jain, and Uday Pratap Singh. «Stock market forecasting using computational intelligence: A survey». In: *Archives of Computational Methods in Engineering* 28.3 (2021), pp. 1069–1101 (cit. on p. 1).

[3] Rich Caruana. «Multitask learning». In: *Machine learning* 28.1 (1997), pp. 41–75 (cit. on pp. 2, 8).

[4] Sebastian Ruder. «An overview of multi-task learning in deep neural networks». In: *arXiv preprint arXiv:1706.05098* (2017) (cit. on pp. 2, 8).

[5] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. «Deep reinforcement learning for automated stock trading: An ensemble strategy». In: *Available at SSRN* (2020) (cit. on pp. 2, 90, 131).

[6] Tai-liang Chen and Feng-yu Chen. «An intelligent pattern recognition model for supporting investment decisions in stock market». In: *Information Sciences* 346 (2016), pp. 261–274 (cit. on p. 4).

[7] Yan Chen, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu. «Genetic network programming with sarsa learning and its application to creating stock trading rules». In: *2007 IEEE Congress on Evolutionary Computation*. IEEE. 2007, pp. 220–227 (cit. on p. 4).

[8] Andrés Arévalo, Jaime Niño, German Hernández, and Javier Sandoval. «High-frequency trading strategy based on deep neural networks». In: *International conference on intelligent computing*. Springer. 2016, pp. 424–436 (cit. on p. 4).

[9] Wen-Chyuan Chiang, David Enke, Tong Wu, and Renzhong Wang. «An adaptive stock index trading decision support system». In: *Expert Systems with Applications* 59 (2016), pp. 195–207 (cit. on p. 4).

[10] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. «Predicting stock market index using fusion of machine learning techniques». In: *Expert Systems with Applications* 42.4 (2015), pp. 2162–2172 (cit. on p. 4).

[11] Ludmila Dymova, Pavel Sevastjanov, and Krzysztof Kaczmarek. «A Forex trading expert system based on a new approach to the rule-base evidential reasoning». In: *Expert Systems with Applications* 51 (2016), pp. 1–13 (cit. on p. 4).

[12] Eunsuk Chong, Chulwoo Han, and Frank C Park. «Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies». In: *Expert Systems with Applications* 83 (2017), pp. 187–205 (cit. on p. 5).

[13] Huicheng Liu. «Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network». In: *arXiv preprint arXiv:1811.06173* (2018) (cit. on p. 5).

[14] Tomer Geva and Jacob Zahavi. «Empirical evaluation of an automated intraday stock recommendation system incorporating both market data and textual news». In: *Decision support systems* 57 (2014), pp. 212–223 (cit. on p. 5).

[15] Mohamed M Mostafa. «Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait». In: *Expert Systems with Applications* 37.9 (2010), pp. 6302–6309 (cit. on p. 5).

[16] Jonathan L Ticknor. «A Bayesian regularized artificial neural network for stock market forecasting». In: *Expert Systems with Applications* 40.14 (2013), pp. 5501–5506 (cit. on p. 5).

[17] Robert P Schumaker, Yulei Zhang, Chun-Neng Huang, and Hsinchun Chen. «Evaluating sentiment in financial news articles». In: *Decision Support Systems* 53.3 (2012), pp. 458–464 (cit. on p. 5).

[18] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. «Language models for financial news recommendation». In: *Proceedings of the ninth international conference on Information and knowledge management*. ACM. 2000, pp. 389–396 (cit. on p. 5).

[19] Sanjiv R Das and Mike Y Chen. «Yahoo! for Amazon: Sentiment extraction from small talk on the web». In: *Management science* 53.9 (2007), pp. 1375–1388 (cit. on p. 5).

[20] Marc-Andre Mittermayer and Gerhard F Knolmayer. «Newscats: A news categorization and trading system». In: *Sixth International Conference on Data Mining (ICDM'06)*. Ieee. 2006, pp. 1002–1007 (cit. on p. 5).

[21] Tao Ma and Guolin Ke. «Multi-task Learning for Financial Forecasting». In: *arXiv preprint arXiv:1809.10336* (2018) (cit. on pp. 5, 17).

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on pp. 7, 9).

[23] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: `10.1038/323533a0`. URL: `https://doi.org/10.1038/323533a0` (cit. on p. 8).

[24] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning phrase representations using RNN encoder-decoder for statistical machine translation». In: *arXiv preprint arXiv:1406.1078* (2014) (cit. on p. 9).

[25] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-term Memory». In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: `10.1162/neco.1997.9.8.1735` (cit. on p. 10).

[26] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 10).

[27] Clayton J Hutto and Eric Gilbert. «Vader: A parsimonious rule-based model for sentiment analysis of social media text». In: *Eighth international AAAI conference on weblogs and social media*. 2014 (cit. on p. 12).

[28] Edward Loper and Steven Bird. «NLTK: the natural language toolkit». In: *arXiv preprint cs/0205028* (2002) (cit. on p. 12).

[29] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. «Indexing by latent semantic analysis». In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407 (cit. on p. 13).

[30] Kevin Lund and Curt Burgess. «Producing high-dimensional semantic spaces from lexical co-occurrence». In: *Behavior research methods, instruments, & computers* 28.2 (1996), pp. 203–208 (cit. on p. 13).

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. «Efficient estimation of word representations in vector space». In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on p. 13).

[32] Jeffrey Pennington, Richard Socher, and Christopher Manning. «Glove: Global vectors for word representation». In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on p. 13).

[33] Gyozo Gidofalvi and Charles Elkan. «Using news articles to predict stock price movements». In: *Department of Computer Science and Engineering, University of California, San Diego* (2001) (cit. on p. 17).

[34] Jacopo Fior and Luca Cagliero. «Exploring the Use of Data at Multiple Granularity Levels in Machine Learning-Based Stock Trading». In: *2020 International Conference on Data Mining Workshops (ICDMW)*. 2020, pp. 333–340. DOI: 10.1109/ICDMW51313.2020.00053 (cit. on pp. 17, 58, 59).

[35] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python ». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 21).

[36] J Welles Wilder. *New concepts in technical trading systems*. Trend Research, 1978 (cit. on p. 24).

[37] Joseph E Granville. *Granville's New Key to Stock Market Profits*. Pickle Partners Publishing, 2018 (cit. on p. 26).

[38] Alexander Elder. *Trading for a living: psychology, trading tactics, money management*. Vol. 31. John Wiley & Sons, 1993 (cit. on p. 27).

[39] *Beware Default Random Forest Importances*. https://explained.ai/rf-importance/. Accessed: 2021-28-11 (cit. on p. 91).

[40] *Zipline Reloaded*. https://pypi.org/project/zipline-reloaded/. Accessed: 2021-02-11 (cit. on p. 131).