Stock Market Prediction Using a Diverse Set of Variables

Data Science: Principles and Practice - **Final Practical**

Maleakhi Agung Wijaya

Department of Computer Science and Technology
University of Cambridge

Word count: 4918

Darwin College

January 2021

Contents

1 Introduction **1**

2 Data Exploration **1**
2.1 Descriptions of Datasets 1
2.2 Standard Data Analysis 2
2.3 Time Series Analysis 7

3 Machine Learning Algorithms **8**
3.1 Preprocessing and Design Choices 8
3.2 Standard Machine Learning Classifiers 9
3.3 Ensemble Classifiers 12
3.4 Neural Network Classifiers 14

4 Further Evaluation of the Best Performing Classifiers **17**
4.1 Experimental Setup 17
4.2 Results 18

5 Dimensionality Reduction and Embeddings **22**
5.1 Principal Component Analysis 22
5.2 t-Distributed Stochastic Neighbour Embedding 24

6 Conclusion **25**

7 Appendix **26**

1 Introduction

Accurately predicting the future behaviours of stock markets would be extremely valuable for traders. However, the task is challenging, as financial markets can be dynamic, non-stationary, and noisy [2,4]. Above all, the *efficient market theory* states that current stock prices reflect all available knowledge, indicating the futility of using past historical data and financial news to forecast the future prices of stocks.

Despite pessimistic induction, recent studies have shown that markets can be inefficient, making it possible to gain returns beyond the market average by simply observing past data and financial news [9]. Numerous methods for making such predictions have been proposed. Among these, machine learning (ML) approaches are popular due to their proven success [6,8,10,11].

This report discusses the performance results of various ML algorithms, including standard, ensemble, and neural networks, for predicting the daily movement of stock markets conditioned on historical data of diverse market-specific, general economic, and other relevant financial variables. Particularly, we focus on predicting the next day's direction of movements for the indices of Dow Jones Industrial average (DJI), National Association of Securities Dealers Automated Quotations exchange (NASDAQ), New York Stock Exchange (NYSE), RUSSELL 2000, and Standard & Poor's 500 (S&P 500) markets.

The remainder of the report is organised as follows. Section 2 presents data exploration results. Subsequently, Section 3 describes the preprocessing steps, implementation details, and mean training set cross-validation performance of all algorithms for all datasets. Section 4 discusses further optimisation of the best performing classifiers and reports their performance on the test set across various metrics. Section 5 describes how we applied additional dimensionality reduction and embedding analysis and presents the results. Finally, Section 6 summarises the main findings of the study.

2 Data Exploration

This section introduces the datasets used in the project. We then discuss standard data analysis results – examining relationships between different features and target variable. Finally, we report the results of further analysis, taking into account temporal data structures. The code for this section is available in `Data Exploration.ipynb`.

2.1 Descriptions of Datasets

We considered six datasets from different markets, amounting to DJI, NASDAQ, NYSE, RUSSELL, and S&P 500. Each dataset comprised 82 numerical features with a varying range of values representing each day of each market. These features were categorised into eight different groups: primitive, technical indicator, economic, commodities, world stock market indices, the exchange rate of USD against other currencies, data from large US companies, and future contracts. The complete list and description of features are shown in Table 18.

The ML task of predicting stock markets' daily movements was modelled as a binary classification problem. We introduced a new target variable, `Movement`, which could take two different values: 'Up' (1) or 'Down' (0). This was computed according to the following equation:

$$target = \begin{cases} 1, & Close_{t+1} > Close_t \\ 0, & \text{otherwise} \end{cases}$$

On labelling the datasets, we obtained five datasets corresponding to different market indices comprising 1,783 instances. We observed that 669 rows (37.5%) contained missing values, where most of them were missing exchange rate features.

2.2 Standard Data Analysis

Class distribution Figure 1 visualises the ’Up’ and ’Down’ class distributions for all datasets. We noted that classes were roughly balanced, with most instances being from the ’Up’ class. Since classes were roughly balanced, we did not need to resample the training data.

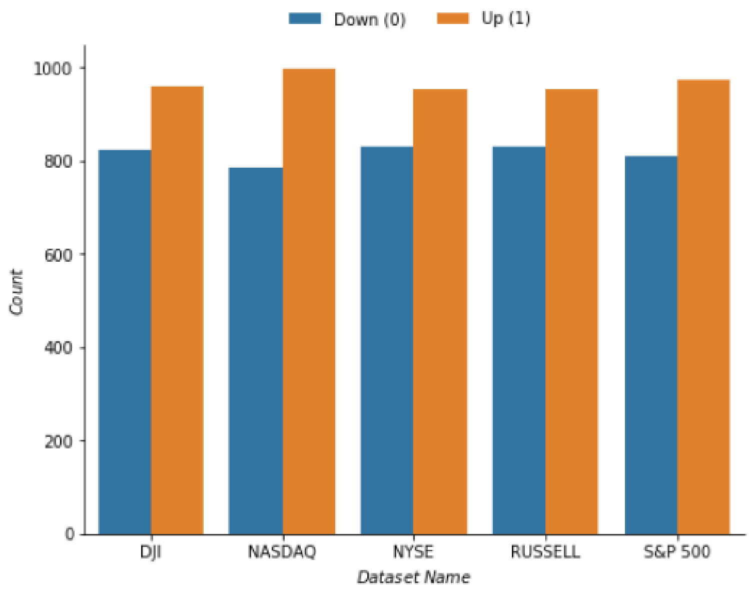


Figure 1: The ’Up’ and ’Down’ class distributions for all datasets. We observed that classes were roughly balanced.

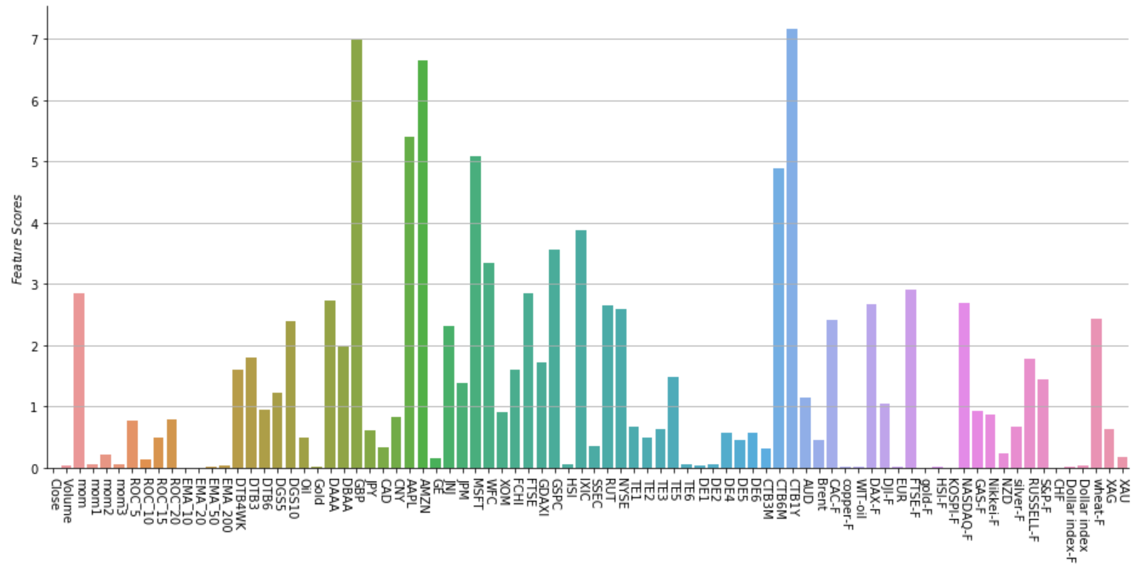
Feature histograms Figure 4 displays histograms of the distributions of feature values for all features. It depicts three feature value histograms for each feature: one considers all instances in the dataset (blue), one considers only ’Down’ class instances (red), and one considers only ’Up’ class instances (gold). Based on the figure, two observations were made. Firstly, most features had values distributed around the mean, suggesting that logarithm transformation was unnecessary for these features. However, some of the features were *tail-heavy* (e.g., `DTB3`, `DTB4WK`, `DTB6`), implying that logarithm transformation can be beneficial for them. During experimentation, we applied logarithm transformation to these features; however, no performance difference was observed. Consequently, we reverted to using the original dataset, performing only essential preprocessing, discussed further in Section 3.2.

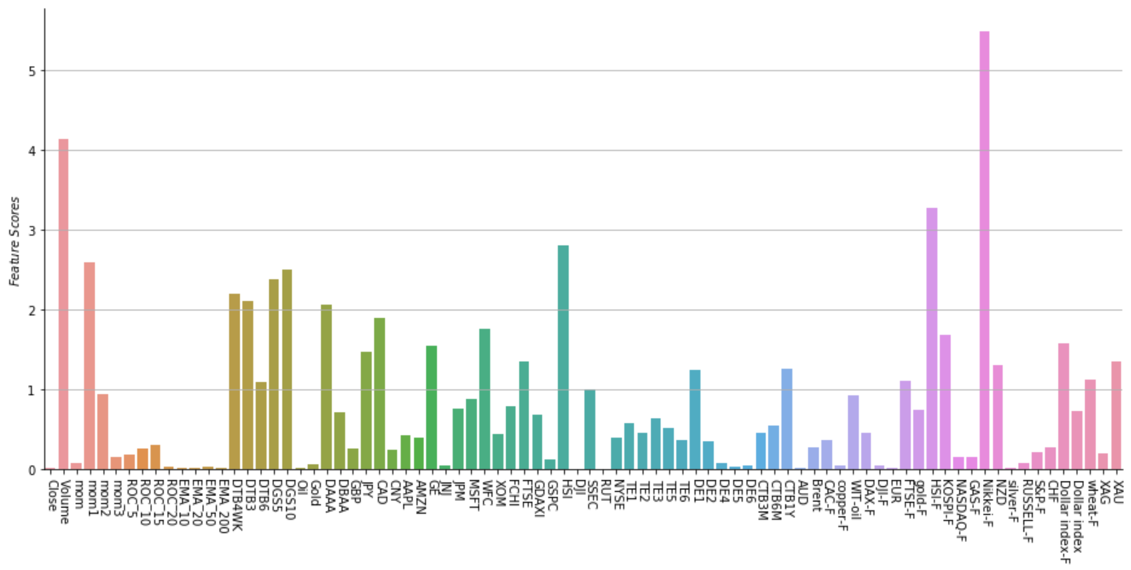
Secondly, we observed that histograms comprising samples from different classes (red and yellow) overlapped and had similar means. This signalled that none of the features would help to discern between classes.

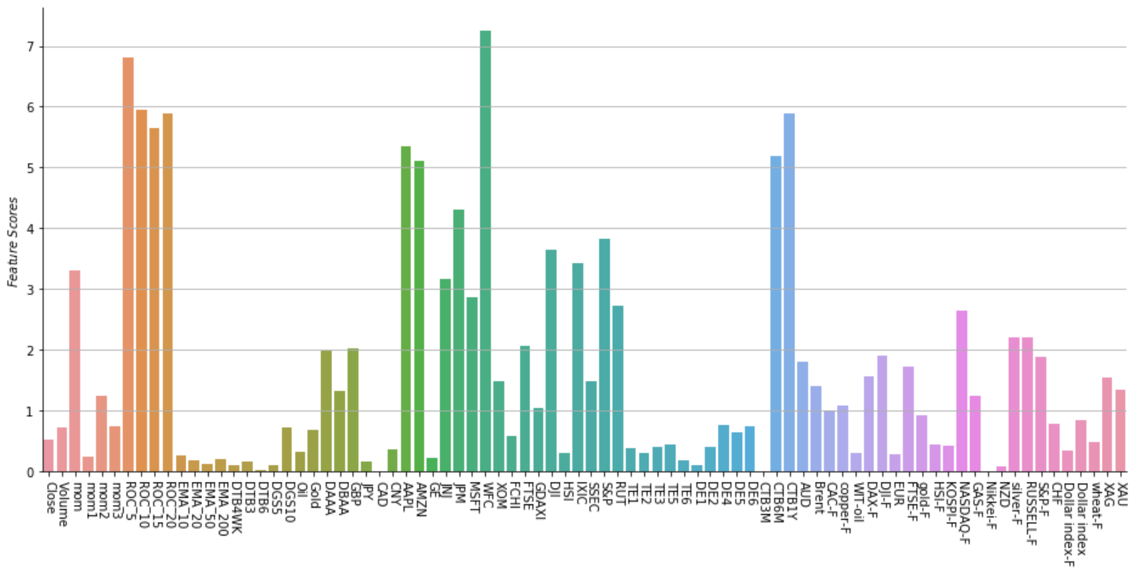
Feature correlation Figure 3 shows the correlation between all numerical features. We observed that some features were strongly correlated (depicted by red and blue cells) with others (e.g., `EMA` was strongly positively correlated (> 0.5) with `Close`, `DTB` was strongly negatively correlated (< -0.5) with `DE`, `DBAA`, and `TE`). The highly correlated features suggested that some features were redundant and could be removed, as they provided minimal additional information for the task, but introduced unnecessary complexity. One method for reducing complexity is to apply dimensionality reduction techniques, such as principal component analysis (PCA), further discussed in Section 5. We applied PCA and relied on reduced representation extracted from deep learning models to address the issue.

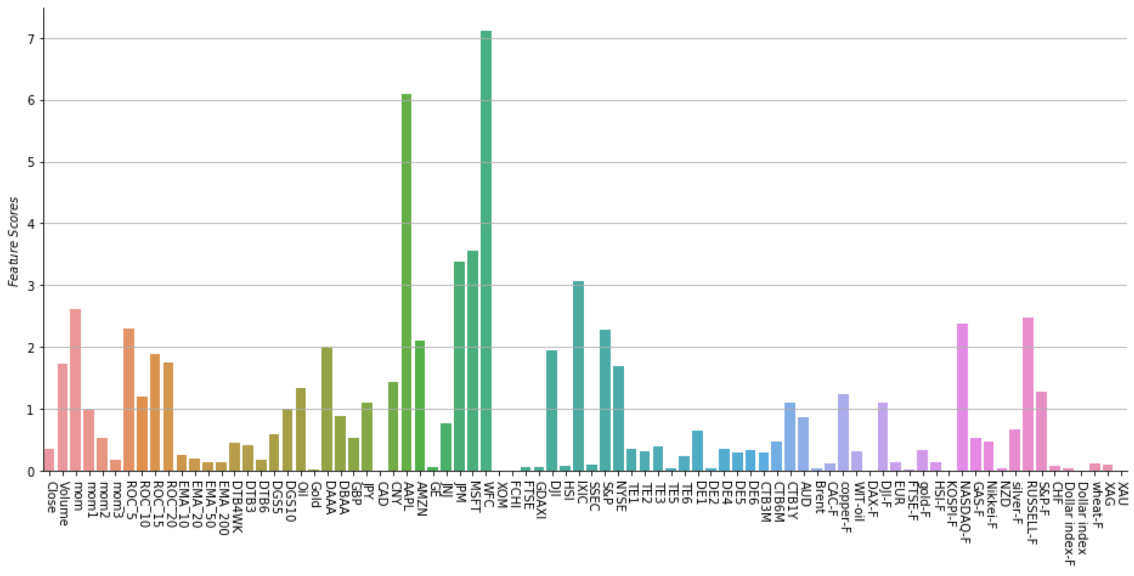
The most important column in the heatmap for the prediction task was the `Close` column (left-most column), as it was a proxy for the target variable. On observing the `Close` column, we noted that some features were strongly correlated (> 0.5) with it, namely `EMA`, `DTB`, and `DE` features, suggesting that these features can be vital for the classification task. We experimented on training the classifiers on subsets of strongly correlated features that are further discussed in Section 3.

Feature importance A more appropriate feature importance metric to measure the importance of numerical features with categorical labels is the analysis of variance (ANOVA) correlation coefficient. We calculated the ANOVA F-value for verifying the existence of equal variance between a group of a categorical feature to continuous responses for all features for all datasets, as shown in Figure 2. We observed that the features considered important differed for all datasets (e.g., `CTB1Y` was considered the most important for DJI, `Nikkei-F` for NASDAQ, `WFC` for NYSE, `WFC` for RUSSELL, and `CTB1Y` for S&P 500). However, we observed similar distributions in scores and agreements in features considered unimportant (e.g., `DE` and `TE` features).




Feature Scores
0
1
2
3
4
5
Close
Volume
mom
mom1
mom2
mom3
ROC_5
ROC_10
ROC_15
ROC_20
EMA_10
EMA_20
EMA_50
EMA_200
DTB4WK
DTB3
DTB6
DGS5
DGS10
Oil
Gold
DAAA
DBAA
GBP
JPY
CAD
CNY
AAPL
AMZN
GE
JNJ
JPM
MSFT
WFC
XOM
FCHI
FTSE
GDAXI
GSPC
HSI
DJI
SSEC
RUT
NYSE
TE1
TE2
TE3
TE5
TE6
DE1
DE2
DE4
DE5
DE6
CTB3M
CTB6M
CTB1Y
AUD
Brent
CAC-F
copper-F
WIT-oil
DAX-F
DJI-F
EUR
FTSE-F
gold-F
HSI-F
KOSPI-F
NASDAQ-F
GAS-F
Nikkei-F
NZD
silver-F
RUSSELL-F
S&P-F
CHF
Dollar index-F
Dollar index
wheat-F
XAG
XAU


Feature Scores
0
1
2
3
4
5
6
7
Close
Volume
mom
mom1
mom2
mom3
ROC_5
ROC_10
ROC_15
ROC_20
EMA_10
EMA_20
EMA_50
EMA_200
DTB4WK
DTB3
DTB6
DGS5
DGS10
Oil
Gold
DAAA
DBAA
GBP
JPY
CAD
CNY
AAPL
AMZN
GE
JNJ
JPM
MSFT
WFC
XOM
FCHI
FTSE
GDAXI
DJI
HSI
IXIC
SSEC
S&P
RUT
NYSE
TE1
TE2
TE3
TE5
TE6
DE1
DE2
DE4
DE5
DE6
CTB3M
CTB6M
CTB1Y
AUD
Brent
CAC-F
copper-F
WIT-oil
DAX-F
DJI-F
EUR
FTSE-F
gold-F
HSI-F
KOSPI-F
NASDAQ-F
GAS-F
Nikkei-F
NZD
silver-F
RUSSELL-F
S&P-F
CHF
Dollar index-F
Dollar index
wheat-F
XAG
XAU


Feature Scores
0
1
2
3
4
5
6
7
Close
Volume
mom
mom1
mom2
mom3
ROC_5
ROC_10
ROC_15
ROC_20
EMA_10
EMA_20
EMA_50
EMA_200
DTB4WK
DTB3
DTB6
DGS5
DGS10
Oil
Gold
DAAA
DBAA
GBP
JPY
CAD
CNY
AAPL
AMZN
GE
JNJ
JPM
MSFT
WFC
XOM
FCHI
FTSE
GDAXI
DJI
HSI
IXIC
SSEC
S&P
NYSE
TE1
TE2
TE3
TE5
TE6
DE1
DE2
DE4
DE5
DE6
CTB3M
CTB6M
CTB1Y
AUD
Brent
CAC-F
copper-F
WIT-oil
DAX-F
DJI-F
EUR
FTSE-F
gold-F
HSI-F
KOSPI-F
NASDAQ-F
GAS-F
Nikkei-F
NZD
silver-F
RUSSELL-F
S&P-F
CHF
Dollar index
Dollar index-F
wheat-F
XAG
XAU

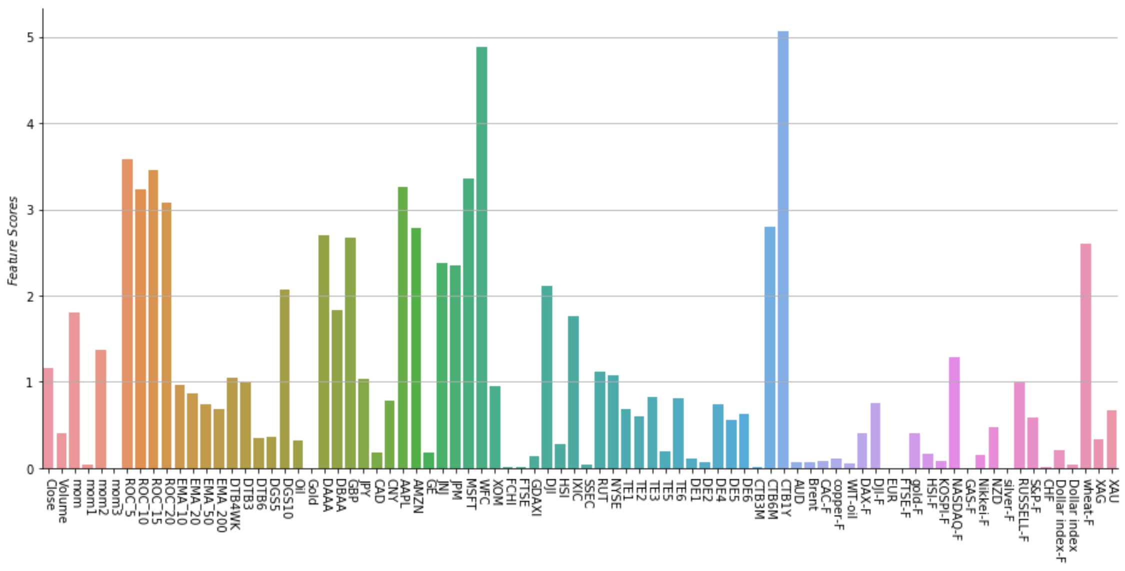


Figure 2: Feature importance scores according to ANOVA F-value for all datasets. The plots are ordered as follows: DJI, NASDAQ, NYSE, RUSSELL, and S&P 500.

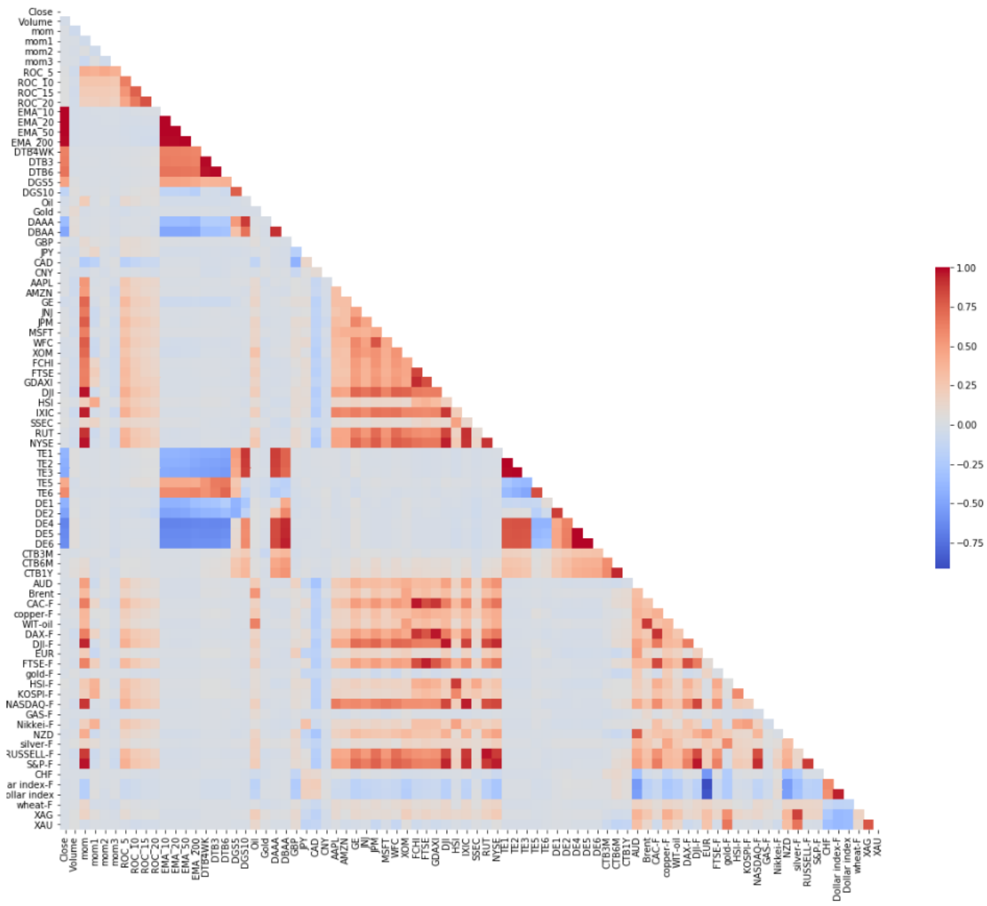


Figure 3: Correlation matrix of all features. Colours closer to red indicate more positive correlations, while those closer to blue indicate negative correlations. Similar correlation matrices were observed in all datasets.

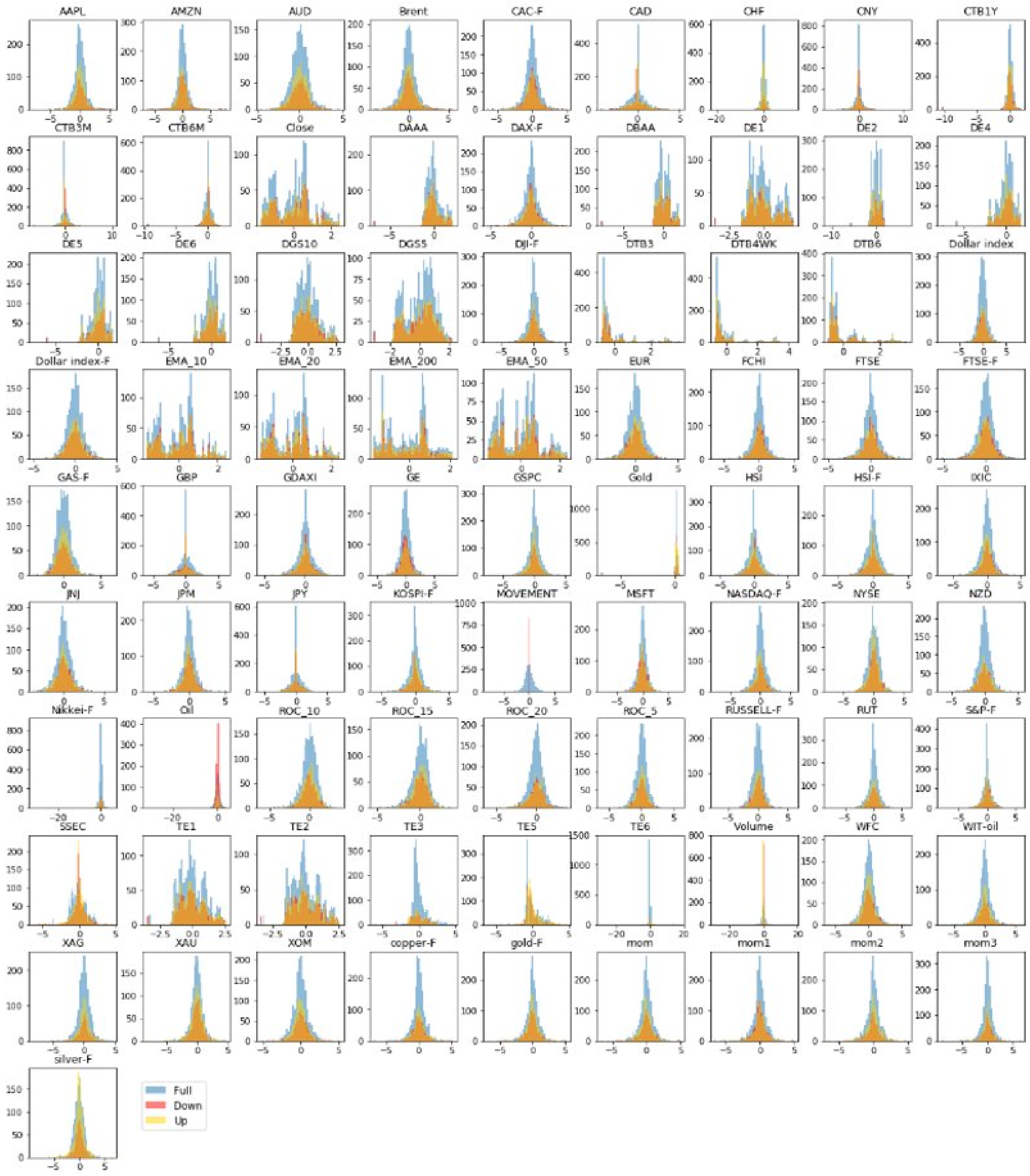


Figure 4: Histograms of feature values for all features. Each plot comprises three histograms, one considers all instances in the dataset (blue), one considers only 'Down' class instances (red), and one considers only 'Up' class instances (gold). Similar histograms were observed for all datasets.

2.3 Time Series Analysis

Stock market price trend As we were working with time series data, we aimed to analyse *seasonality* – regular and predictable patterns that recur over a certain period – by plotting the daily `Close` price movement for 2011-2016. Overall, apart from the 2011 and 2015 lines, none of the lines followed similar patterns, suggesting that the `Close` price did not follow any yearly trend. This demonstrated the difficulty of the prediction task, as the `Close` price followed a random walk consistent with the *random walk hypothesis* popularised by Malkiel [14].

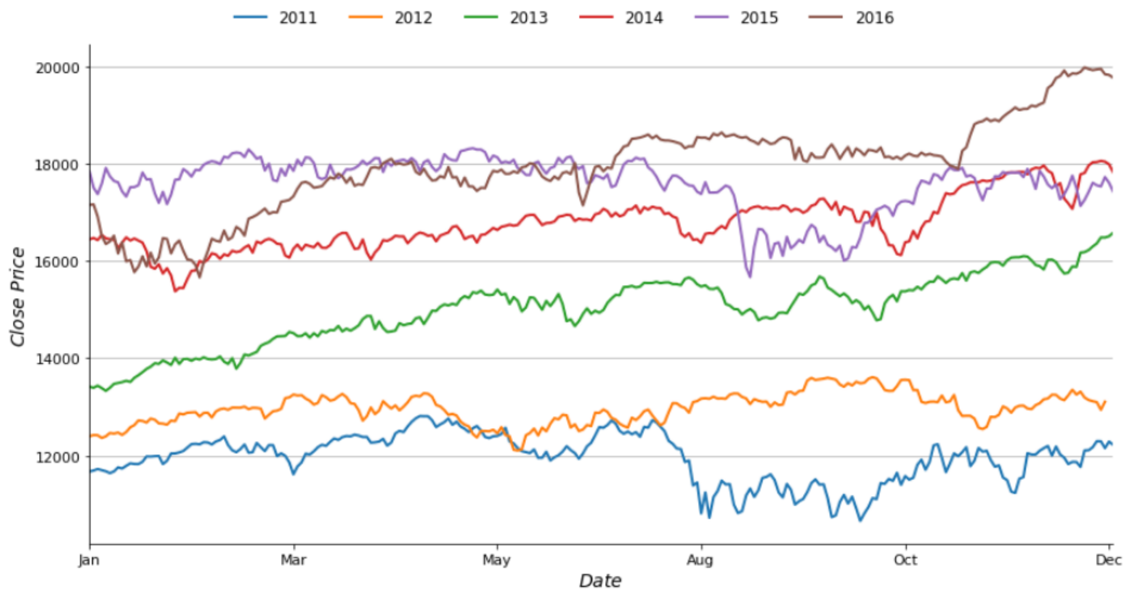


Figure 5: Daily Close price movement for 2011-2016. Similar patterns were observed for all datasets.

Autocorrelation To further analyse seasonality, we plotted the *autocorrelation* plot, depicting the correlation between current feature values with a delayed copy of itself over varying time lags. Overall, we observed a strong positive correlation before 200 days' lag and a strong negative correlation around 800 days' lag. For the experiments, we observed that conditioning the prediction task on the values of diverse variables over the previous 50-200 days appeared appropriate, as shown by the relatively high autocorrelation while providing sufficient historical information. We selected 60 days as the window size following our observation and recommendation by Hoseinzade and Haratizadeh [7].

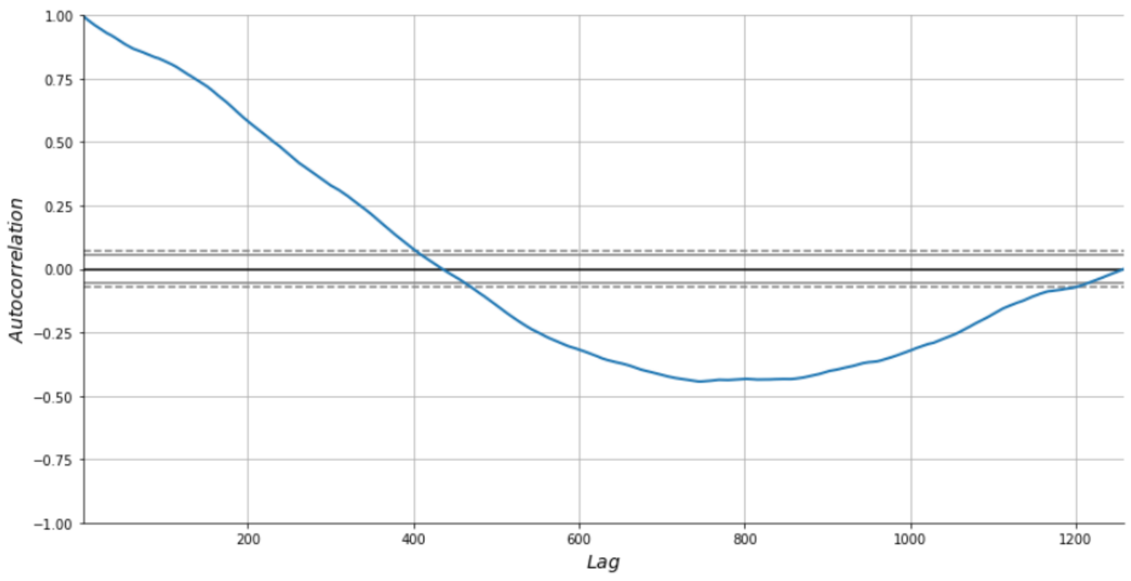


Figure 6: Autocorrelation plot of the Close price feature from 2011-2016. The horizontal lines in the plot correspond to 95% (solid) and 99% (dashed) confidence bands. Similar patterns were observed for all datasets.

3 Machine Learning Algorithms

This section first explains the initial preprocessing steps and implementation details of various ML algorithms we used to predict stock markets' daily movements. It then reports the mean training set cross-validation accuracies and macro-average F1s of all ML models for all datasets. Further evaluation was performed on three best classifiers according to the cross-validation result discussed in Section 4.

3.1 Preprocessing and Design Choices

Before implementing ML models, we preprocessed the datasets, ensuring that they were error-free and in a suitable format for fitting to the models. The preprocessing steps and design choices are detailed below.

Missing values The given market datasets had several features with missing values totalling 37.5% of instances for all datasets (Section 2.1). We assumed that these values were missing completely at random due to occasional misrecording of information during the holiday season [15]. To address this issue, we imputed the missing values using several imputation methods appropriate for time series data, including the *forward fill*, which propagates the previous record to the missing record, *zero fill*, which imputes missing records with zero [7], and *mean imputation* over a time window considering seasonality assumption. The choice of imputation methods did not affect the classifiers' performances. Consequently, we employed the zero fill method following Hoseinzade and Haratizadeh [7].

Feature scaling All 82 features were numeric with different ranges of values. As some ML algorithms (e.g., distance- and gradient descent-based) are sensitive to feature scaling, we normalised all features to ensure that learning algorithms could converge faster and that all features were treated equally. We standardised all features following Hoseinzade and Haratizadeh [7,8].

Feature sets We considered three different feature sets: a *full* feature set with 82 total features, a reduced *PCA* feature set comprising 30 principal components explaining 90.7% variance of the original dataset (inspired by the method discussed in [18]), and a *technical indicator* feature set comprising 13 technical indicator variables, shown in Table 18 (inspired by the method discussed in [10]).

We selected the length of history or window size as 60 days – the model could use information from the past 60 days to make a prediction. Depending on the ML model architecture, we concatenated 60 days of data in four different ways. For flat classifiers (classifiers discussed in Sections 3.2, 3.3, and feedforward neural network (FFNN)), we concatenated the features along the columns. We thus producing 4,920 features for a prediction on data with the full feature set; 1,800 features for a prediction on data with the PCA feature set; and 780 features for a prediction on data with the technical indicator feature set. For 2D-CNNpred [7] architecture, we stacked the features into a 2D tensor where each input had size $60 \times$ *number of features*, which could be 82, 30, or 13 depending on the feature set we used. For 3D-CNNpred [7], we stacked the features into a 3D tensor, where each input was of size $60 \times$ *number of markets* $\times$ *number of features*. The procedure for building the tensor for the 2D- and 3D-CNNpred is detailed in [7]. Finally, for the long short term memory (LSTM) classifier, we used `TimeseriesGenerator` from `TensorFlow` [1] to generate 3D tensors of size *batch size* $\times 60 \times$ *number of features*.

Dataset split As we used cross-validation to evaluate the performance of various ML algorithms in this section, we split the dataset into an 80%/20% train/test split with 1,379 train and 345 test instances. However, for the neural network experiments, we reported the average held-out results; thus, we split the dataset into an 80%/10%/10% train/validation/test split with 1,395 train, 156 validation, and 173 test instances. Since some algorithms (2D- and 3D-CNNpred) were trained on aggregated market datasets, datasets must not be shuffled and stratified when splitting to avoid *cheating* – using future data or same-date data from different markets for making predictions in another market[1].

Evaluation metrics To compare the results of different models, we primarily use two metrics: *accuracy* and *macro-average F1*. Accuracy was used as it is a common metric that has been used extensively in the area. The macro-average F1 calculates the mean of F1 scores for each class, providing insight into the model capabilities of distinguishing both classes, as suggested in the literature [5, 7, 16].

3.2 Standard Machine Learning Classifiers

We implemented standard ML models, amounting to Gaussian naive Bayes (NB), logistic regression, k-nearest neighbours (k-NN), decision tree, and support vector machine (SVM) for the classification task to predict the daily price movement of stock markets. As a baseline, we implemented the ZeroR classifier that predicts every instance as the majority class label, ignoring the features. All models in this section were implemented using the `scikit-learn` [17] package. We used mixtures of grid and random search depending on the hyperparameter

[1]This cheating can lead to classifiers that are trained with aggregated shuffled datasets to achieve accuracy and F1 exceeding 80% as markets moved similarly on the same date.

range of values and reported the best model performance. Several hyperparameters of interest are further discussed in Section 4.1 and the code for the experimentation is available in `Standard ML Analysis.ipynb`.

Tables 1 - 5 summarise the 5-fold cross-validation training set accuracies and macro-average F1s for all datasets. Overall, we observed that all classifiers distinguished both classes better than the ZeroR baseline, as shown by a significant difference in their macro-average F1 ($46 - 54\%$ for all classifiers, $34 - 36\%$ for the baseline). However, we observed that the baseline was slightly better in terms of accuracy ($53 - 56\%$) than the trained classifiers ($49 - 56\%$).

Apart from the overall difference in performance between classifiers, we observed that all classifiers achieved similar performances irrespective of whether they were trained on full, PCA, or technical indicator features. This suggested that both the PCA and technical indicator features captured sufficient information to make the prediction. This finding supports the technical analysis view in finance that strongly believe that technical indicator features are sufficient for predicting stock market futures [13].

The following is a summary of the best models for each dataset:

- **DJI**: logistic regression trained on the full feature set achieved the best performance with an accuracy of 52.9% and a macro-average F1 of 52.7%;
- **NASDAQ**: decision tree trained on the technical indicator feature set achieved the best performance with an accuracy of 53.9% and a macro-average F1 of 53.1%;
- **NYSE**: NB trained on the technical indicator feature set achieved the best performance with an accuracy of 51.3% and a macro-average F1 of 51.0%;
- **RUSSELL**: logistic regression trained on the technical indicator feature set achieved the best performance with an accuracy of 52.1% and a macro-average F1 of 51.9%;
- **S&P 500**: decision tree trained on the PCA feature set achieved the best performance with an accuracy of 55.7% and a macro-average F1 of 52.2%.

		Baseline	**NB**	**Logistic Regression**	**k-NN**	**Decision Tree**	**SVM**
Full	Accuracy	**53.8%**	49.2%	52.9%	50.4%	51.5%	53.5%
	Macro F1	34.9%	48.8%	**52.7%**	50.2%	51.2%	48.2%
PCA	Accuracy	**53.8%**	50.2%	49.6%	50.3%	50.6%	52.0%
	Macro F1	34.9%	**50.0%**	49.4%	49.6%	49.1%	46.5%
TI	Accuracy	**53.8%**	51.7%	49.4%	52.5%	50.8%	50.8%
	Macro F1	34.9%	51.5%	49.3%	**52.2%**	49.6%	46.7%

Table 1: Mean training set cross-validation accuracies and macro-average F1s for Dow Jones Industrial Average index using baseline, naive Bayes, logistic regression, k-nearest neighbours, decision tree, and SVM classifiers, trained on full, PCA, and technical indicator feature sets.

		Baseline	NB	Logistic Regression	k-NN	Decision Tree	SVM
Full	Accuracy	**55.6%**	47.1%	51.9%	49.5%	49.2%	55.5%
	Macro F1	35.7%	46.9%	**51.5%**	48.3%	49.8%	44.6%
PCA	Accuracy	**55.6%**	51.4%	51.1%	50.2%	49.2%	56.4%
	Macro F1	35.7%	**51.0%**	50.6%	49.2%	49.8%	46.1%
TI	Accuracy	**55.6%**	49.4%	51.3%	48.7%	53.9%	53.4%
	Macro F1	35.7%	48.8%	49.6%	47.1%	**53.1%**	44.0%

Table 2: Mean training set cross-validation accuracies and macro-average F1s for NASDAQ Composite index using baseline, naive Bayes, logistic regression, k-nearest neighbours, decision tree, and SVM classifiers, trained on full, PCA, and technical indicator feature sets.

		Baseline	NB	Logistic Regression	k-NN	Decision Tree	SVM
Full	Accuracy	**53.4%**	47.3%	45.2%	48.8%	50.6%	50.3%
	Macro F1	34.8%	47.2%	45.1%	47.9%	**51.0%**	43.8%
PCA	Accuracy	**53.4%**	49.6%	49.2%	50.0%	50.6%	52.1%
	Macro F1	34.8%	**49.3%**	49.0%	49.2%	49.1%	45.6%
TI	Accuracy	**53.4%**	51.3%	50.0%	49.2%	51.5%	50.9%
	Macro F1	34.8%	**51.0%**	50.0%	48.5%	50.8%	46.5%

Table 3: Mean training set cross-validation accuracies and macro-average F1s for NYSE Composite index using baseline, naive Bayes, logistic regression, k-nearest neighbours, decision tree, and SVM classifiers, trained on full, PCA, and technical indicator feature sets.

		Baseline	NB	Logistic Regression	k-NN	Decision Tree	SVM
Full	Accuracy	**53.3%**	48.8%	49.6%	48.5%	50.7%	52.2%
	Macro F1	34.8%	48.5%	49.4%	48.1%	**50.0%**	45.9%
PCA	Accuracy	**53.3%**	48.9%	49.6%	49.6%	51.6%	52.0%
	Macro F1	34.8%	48.8%	**49.4%**	48.0%	49.0%	47.1%
TI	Accuracy	**53.3%**	48.5%	52.1%	50.1%	50.2%	51.9%
	Macro F1	34.8%	47.8%	**51.9%**	50.0%	50.0%	48.5%

Table 4: Mean training set cross-validation accuracies and macro-average F1s for RUSSELL 2000 using baseline, naive Bayes, logistic regression, k-nearest neighbours, decision tree, and SVM classifiers, trained on full, PCA, and technical indicator feature sets.

		Baseline	NB	Logistic Regression	k-NN	Decision Tree	SVM
Full	Accuracy	**54.4%**	47.7%	51.3%	51.0%	49.7%	53.5%
	Macro F1	35.2%	47.5%	**51.0%**	49.7%	48.6%	46.6%
PCA	Accuracy	54.4%	51.4%	50.2%	48.2%	**55.7%**	55.0%
	Macro F1	35.2%	51.3%	50.0%	47.8%	**52.2%**	49.0%
TI	Accuracy	**54.4%**	46.8%	50.5%	48.7%	52.4%	52.4%
	Macro F1	35.2%	46.6%	49.2%	48.2%	**52.1%**	47.4%

Table 5: Mean training set cross-validation accuracies and macro-average F1s for S&P 500 index using baseline, naive Bayes, logistic regression, k-nearest neighbours, decision tree, and SVM classifiers, trained on full, PCA, and technical indicator feature sets.

3.3 Ensemble Classifiers

We implemented various ensemble classifiers as follows:

1. A soft *voting classifier* based on NB, logistic regression, k-NN, decision tree, and SVM;
2. The *bagging* and *pasting* ensemble methods on decision trees, implemented as *random forest* classifiers;
3. The adaptive boosting (AB) ensemble method on decision tree classifiers;
4. The gradient boosting (GB) ensemble method on decision tree classifiers;
5. The stacking ensemble method comprising logistic regression, decision tree, k-NN, NB, and SVM as *base* classifiers and a logistic regression as the *meta* classifier that learn to combine the predictions of base classifiers to make the final prediction.

All models in this section were implemented using the `scikit-learn` [17] package. We used mixtures of grid and random search depending on the hyperparameter range of values and reported the best model performances. The code for the experimentation is available in `Ensemble ML Analysis.ipynb`.

Tables 6 - 10 summarise the 5-fold cross-validation training set accuracies and macro-average F1s for all datasets. Overall, we observed similar conclusions to Section 3.2: all ensemble-based classifiers could significantly better distinguished both classes compared to the ZeroR baseline, albeit with lower accuracies. Similarly to Section 3.2, all ensemble classifiers achieved similar performances, irrespective of whether they were trained on full, PCA, or technical indicator feature sets. Interestingly, stacking classifiers consistently achieved lower macro-average F1s for all datasets than other ensemble methods, as shown in the final columns of the tables. After further inspection, we observed a significant overlap of predictions on the base model, making them insufficiently uncorrelated, reducing the stacking method's effectiveness. Other than for stacking, we observed similar performances for ensemble and base models.

The following is a summary of the best models for each dataset:

- **DJI**: AB trained on the PCA feature set achieved the best performance with an accuracy of 54.8% and a macro-average F1 of 53.1%;

- **NASDAQ**: AB trained on the PCA feature set achieved the best performance with an accuracy of 54.8% and a macro-average F1 of 53.1%;
- **NYSE**: GB trained on the PCA feature set achieved the best performance with an accuracy of 52.7% and a macro-average F1 of 50.6%;
- **RUSSELL**: pasting trained on the technical indicator feature set achieved the best performance with an accuracy of 52.7% and a macro-average F1 of 52.3%;
- **S&P 500**: pasting trained on the technical indicator feature set achieved the best performance with an accuracy of 52.6% and a macro-average F1 of 52.2%.

		Voting	**Bagging**	**Pasting**	**AB**	**GB**	**Stacking**
Full	Accuracy	51.4%	51.2%	49.1%	51.1%	51.1%	**51.5%**
	Macro F1	**51.0%**	50.0%	48.9%	50.3%	49.5%	42.0%
PCA	Accuracy	48.2%	52.6%	49.5%	**54.8%**	50.3%	54.0%
	Macro F1	48.6%	50.1%	48.5%	**53.1%**	49.8%	43.8%
TI	Accuracy	50.4%	49.6%	51.5%	52.4%	51.1%	**53.5%**
	Macro F1	50.0%	49.2%	48.4%	**50.2%**	48.8%	43.5%

Table 6: Mean training set cross-validation accuracies and macro-average F1s for Dow Jones Industrial Average index using voting, bagging, pasting, AB, GB, and stacking ensembles, trained on full, PCA, and technical indicator feature sets.

		Voting	**Bagging**	**Pasting**	**AB**	**GB**	**Stacking**
Full	Accuracy	50.0%	53.1%	53.1%	53.1%	52.8%	**54.2%**
	Macro F1	48.8%	47.3%	47.2%	**52.1%**	51.6%	38.1%
PCA	Accuracy	50.8%	54.0%	54.5%	54.8%	53.1%	**55.1%**
	Macro F1	50.8%	48.6%	47.4%	**53.1%**	52.1%	42.7%
TI	Accuracy	52.2%	53.4%	51.3%	52.5%	51.0%	**55.6%**
	Macro F1	**50.8%**	46.5%	48.6%	50.2%	48.6%	39.5%

Table 7: Mean training set cross-validation accuracies and macro-average F1s for NASDAQ Composite index using voting, bagging, pasting, AB, GB, and stacking ensembles, trained on full, PCA, and technical indicator feature sets.

		Voting	Bagging	Pasting	AB	GB	Stacking
Full	Accuracy	46.5%	50.3%	50.6%	50.2%	49.2%	**51.7%**
	Macro F1	46.3%	47.7%	47.4%	46.3%	**48.5%**	48.0%
PCA	Accuracy	48.7%	49.5%	51.0%	51.6%	**52.7%**	52.6%
	Macro F1	48.0%	49.3%	50.6%	**50.8%**	50.6%	43.5%
TI	Accuracy	52.3%	51.1%	50.4%	50.0%	50.8%	**52.5%**
	Macro F1	**50.5%**	48.1%	48.4%	49.2%	50.0%	44.1%

Table 8: Mean training set cross-validation accuracies and macro-average F1s for NYSE Composite index using voting, bagging, pasting, AB, GB, and stacking ensembles, trained on full, PCA, and technical indicator feature sets.

		Voting	Bagging	Pasting	AB	GB	Stacking
Full	Accuracy	50.1%	52.4%	52.1%	50.0%	51.2%	**52.6%**
	Macro F1	48.5%	49.0%	47.6%	49.4%	**49.6%**	42.4%
PCA	Accuracy	48.7%	49.7%	50.4%	51.6%	49.4%	**53.7%**
	Macro F1	49.1%	49.3%	48.4%	**51.0%**	48.4%	47.8%
TI	Accuracy	49.2%	50.8%	**52.7%**	52.1%	51.0%	51.6%
	Macro F1	47.9%	51.2%	**52.3%**	51.8%	51.1%	46.2%

Table 9: Mean training set cross-validation accuracies and macro-average F1s for RUSSELL 2000 using voting, bagging, pasting, AB, GB, and stacking ensembles, trained on full, PCA, and technical indicator feature sets.

		Voting	Bagging	Pasting	AB	GB	Stacking
Full	Accuracy	50.1%	**54.9%**	51.6%	50.2%	51.5%	53.9%
	Macro F1	49.6%	48.1%	**51.7%**	49.6%	51.2%	42.3%
PCA	Accuracy	50.8%	51.3%	**53.2%**	50.7%	52.4%	52.8%
	Macro F1	**50.4%**	50.0%	49.6%	49.3%	49.4%	41.2%
TI	Accuracy	48.6%	53.3%	52.6%	50.5%	51.8%	**54.4%**
	Macro F1	47.7%	51.1%	**52.2%**	49.2%	50.0%	39.3%

Table 10: Mean training set cross-validation accuracies and macro-average F1s for S&P 500 index using voting, bagging, pasting, AB, GB, and stacking ensembles, trained on full, PCA, and technical indicator feature sets.

3.4 Neural Network Classifiers

We implemented various neural network classifiers as follows:

1. A FFNN with two hidden layers consisting of 64 and 32 neurons with rectified linear unit (ReLU) activation functions, followed by a sigmoid layer for making the final prediction;

2. A 2D-CNNpred with architecture and settings as specified in [7] comprising a convolutional layer with filters of size $1 \times$ *number of features* and ReLU activation functions, two convolutional layers with ReLU activation functions and max-pooling layers with filters of size 3×1 and 2×1, respectively, and a sigmoid layer for making the final prediction;

3. A 3D-CNNpred with architecture and settings as specified in [7] comprising a convolutional layer with filters of size 1×1 and ReLU activation functions, a convolutional layer with filters of size $3 \times$ *number of markets* and ReLU activation functions, and the same configurations for the next three layers as the 2D-CNNpred, comprising a max-pooling layer with a filter of size 2×1, a convolutional layer with filters of size 3×1, a max-pooling layer with a filter of size 2×1, and a sigmoid layer for making the final prediction;

4. An LSTM network with two LSTM layers consisting of 128 and 64 neurons with LeakyReLU activation functions, followed by a sigmoid layer for making the final prediction.

All models in this section were implemented using the `TensorFlow` [1] package. We used mixtures of grid and random search depending on the hyperparameter range of values and reported the best models. The code for the experimentation is available in
`Neural Network Analysis.ipynb`.

Tables 11 - 15 summarise the mean of five-run held-out test accuracies and macro-average F1s for all datasets. Overall, we observed that the neural-based models achieved slightly better accuracy and macro-average F1 than the standard and ensemble ML models, and significantly better accuracy and F1 than the ZeroR baseline. Contrary to Sections 3.2 and 3.3, neural network classifiers trained on full feature sets achieved better performance in accuracy and macro-average F1 compared to those trained on reduced representations, namely PCA and technical indicator feature sets. This demonstrates the strength of neural networks as *automatic feature extractors* that can automatically construct representation from raw features without the need for manual feature engineering, unlike standard and ensemble methods [3]. Particularly, we noted the effectiveness of 2D-CNNpred compared to other neural network architectures.

Finally, the performance difference between 2D- and 3D-CNNpred compared to other traditional, ensemble, and neural methods was slight; their performance differences were less significant than those reported by Hoseinzade and Haratizadeh [7, 8].

The following is a summary of the best models for each dataset:

- **DJI**: 2D-CNNpred trained on the full feature set achieved the best performance with an accuracy of 59.5% and a macro-average F1 of 58.9%;

- **NASDAQ**: 2D-CNNpred trained on the full feature set achieved the best performance with an accuracy of 57.8% and a macro-average F1 of 52.5%;

- **NYSE**: 2D-CNNpred trained on the technical indicator feature set achieved the best performance with an accuracy of 50.4% and a macro-average F1 of 49.6%;

- **RUSSELL**: 2D-CNNpred trained on the full feature set achieved the best performance with an accuracy of 53.2% and a macro-average F1 of 52.5%;

- **S&P 500**: 2D-CNNpred trained on the PCA feature set achieved the best performance with an accuracy of 52.6% and a macro-average F1 of 51.9%.

		FFNN	2D-CNNpred	3D-CNNpred	LSTM
Full	Accuracy	53.2%	59.5%	59.5%	**63.9%**
	Macro F1	52.8%	**58.9%**	55.3%	43.0%
PCA	Accuracy	52.0%	48.6%	**60.1%**	52.0%
	Macro F1	**51.8%**	47.7%	45.3%	48.8%
TI	Accuracy	49.1%	47.4%	47.4%	**49.9%**
	Macro F1	**48.6%**	46.8%	46.4%	46.1%

Table 11: Mean held-out accuracies and macro-average F1s for Dow Jones Industrial Average index using feed-forward neural network, 2D-CNNpred, 3D-CNNpred, and LSTM classifiers, trained on full, PCA, and technical indicator feature sets.

		FFNN	2D-CNNpred	3D-CNNpred	LSTM
Full	Accuracy	54.3%	**57.8%**	53.3%	52.9%
	Macro F1	**53.9%**	52.5%	50.2%	48.5%
PCA	Accuracy	53.2%	**54.3%**	54.2%	52.5%
	Macro F1	**52.7%**	51.8%	49.9%	50.0%
TI	Accuracy	**51.4%**	46.8%	46.8%	47.7%
	Macro F1	**50.4%**	45.1%	45.5%	46.3%

Table 12: Mean held-out accuracies and macro-average F1s for NASDAQ Composite index using feedforward neural network, 2D-CNNpred, 3D-CNNpred, and LSTM classifiers, trained on full, PCA, and technical indicator feature sets.

		FFNN	2D-CNNpred	3D-CNNpred	LSTM
Full	Accuracy	42.2%	52.0%	51.1%	**52.9%**
	Macro F1	42.2%	**47.7%**	47.5%	40.2%
PCA	Accuracy	50.3%	**50.9%**	**50.9%**	50.2%
	Macro F1	**49.7%**	46.7%	45.5%	45.5%
TI	Accuracy	50.3%	**50.4%**	50.1%	50.1%
	Macro F1	49.4%	**49.6%**	46.9%	44.5%

Table 13: Mean held-out accuracies and macro-average F1s for NYSE Composite index using feedforward neural network, 2D-CNNpred, 3D-CNNpred, and LSTM classifiers, trained on full, PCA, and technical indicator feature sets.

		FFNN	2D-CNNpred	3D-CNNpred	LSTM
Full	Accuracy	43.9%	**53.2%**	51.2%	53.0%
	Macro F1	43.9%	**52.5%**	47.7%	42.0%
PCA	Accuracy	49.1%	48.0%	47.7%	**49.9%**
	Macro F1	**48.8%**	46.7%	46.9%	45.5%
TI	Accuracy	**52.0%**	46.8%	46.6%	48.9%
	Macro F1	**52.0%**	46.4%	46.8%	47.7%

Table 14: Mean held-out accuracies and macro-average F1s for RUSSELL 2000 using feedforward neural network, 2D-CNNpred, 3D-CNNpred, and LSTM classifiers, trained on full, PCA, and technical indicator feature sets.

		FFNN	2D-CNNpred	3D-CNNpred	LSTM
Full	Accuracy	50.9%	53.8%	53.5%	**56.3%**
	Macro F1	**50.6%**	49.0%	48.9%	39.3%
PCA	Accuracy	45.1%	**52.6%**	52.5%	51.1%
	Macro F1	44.5%	**51.9%**	51.9%	50.3%
TI	Accuracy	**52.0%**	48.6%	49.0%	49.9%
	Macro F1	**51.1%**	47.7%	48.0%	50.3%

Table 15: Mean held-out accuracies and macro-average F1s for S&P 500 index using feedforward neural network, 2D-CNNpred, 3D-CNNpred, and LSTM classifiers, trained on full, PCA, and technical indicator feature sets.

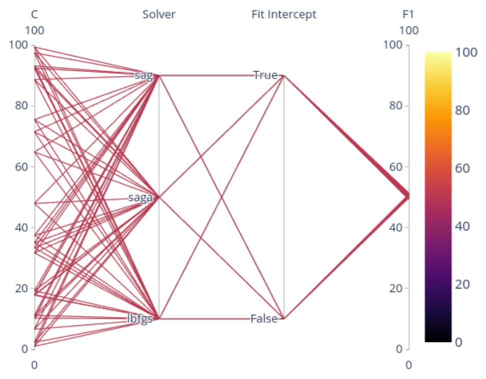
4 Further Evaluation of the Best Performing Classifiers

This section describes further experimentation on the three best classifiers from Sections 3.2, 3.3, and 3.4, amounting to the logistic regression, AB classifier, and 2D-CNNpred. We report these models' test set performances across various evaluation metrics, namely accuracy, macro-average F1, recall, precision, and the area under the receiver operating characteristic curve (AUC). The code for the section is available in `Evaluation.ipynb`.

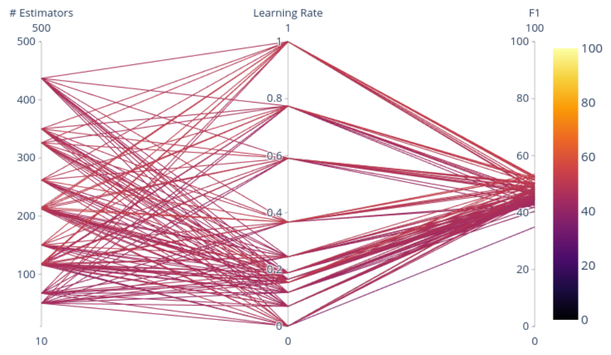
4.1 Experimental Setup

For the experimentation, we followed steps similar to those discussed in Section 3.1, except that we only focused on training the models on the full feature set. We built general models following the approach discussed in the paper [7] – training them using samples from different markets before evaluating their performance on each market's test set to measure the extent to which these models will generalise on different markets.

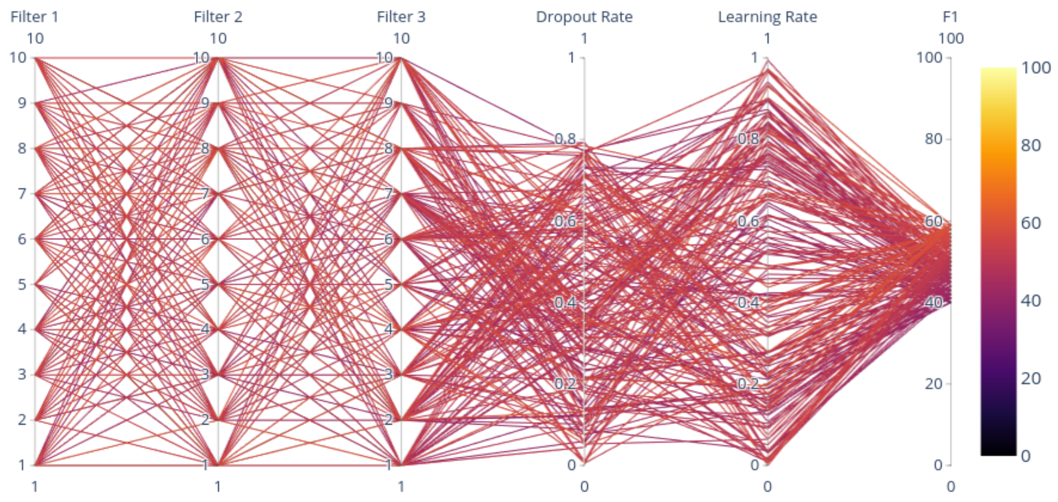
We conducted a further hyperparameter search using mixtures of grid search and random search, selecting the best model according to the validation set. Figure 7 depicts the hyperparameter search results for logistic regression, AB classifier, and 2D-CNNpred. Overall, we observed that the hyperparameter search improved the performance over the classifier with default settings by $3-5\%$.



(a) Logistic regression



(b) AB classifier



(c) 2D-CNNpred

Figure 7: Hyperparameter search results on the validation set for logistic regression, AB classifier, and 2D-CNNpred. Each line in the plots represents a single model configuration, and its macro-average F1 result on the validation set.

4.2 Results

A summary of the evaluation results for the models trained using samples from all datasets and tested on the testing set of each dataset is depicted in Table 16. The confusion matrices are shown in Figure 8.

Overall, we observed that 2D-CNNpred achieved the highest accuracy compared to logistic regression and the AB classifier for all datasets (**57%, 54%, 53%, 54%, 58%**). Not only did 2D-CNNpred achieved high accuracy; it achieved high macro-average recall, precision, and F1 for all datasets. In particular, it achieved significantly higher macro-average recall, precision, and F1 on DJI (**55%, 55%, 54%**) and S&P 500 datasets (**57%, 56%, 56%**). Concerning logistic regression and the AB classifier, we observed that both models achieved similar performances in the range of 48-54% on accuracy, macro-average recall, precision, and F1. This demonstrates the benefit of automatic feature learning; in this case, automated feature extraction using the convolutional layers in 2D-CNNpred learned superior feature representation to manual features engineered by domain experts. However, the performances of all models were only marginally better than that of a stratified random baseline, showing the complexity of the problem, consistent with our initial assumption made during data exploration – that the datasets do not display seasonal trends and that the `Close` price changes following a random walk (Section 2.3).

Further inspection of the confusion matrices, indicated that all classifiers on the NASDAQ, NYSE, RUSSELL, and S&P 500 could predict the 'Up' price movement well, but mispredict the 'Down' price. Interestingly, for the DJI dataset, both logistic regression and AB classifiers behaved oppositely.

Datasets	**Classifiers**	**Accuracy**	**Macro-Average**		
			Recall	**Precision**	**F1**
DJI	Logistic regression	50.0%	50.0%	50.0%	50.0%
	AB classifier	49.0%	49.0%	49.0%	48.0%
	2D-CNNpred	**57.0%**	**55.0%**	**55.0%**	**54.0%**
NASDAQ	Logistic regression	51.0%	**52.0%**	**52.0%**	**51.0%**
	AB classifier	51.0%	**52.0%**	**52.0%**	**51.0%**
	2D-CNNpred	**54.0%**	50.0%	50.0%	49.0%
NYSE	Logistic regression	48.0%	48.0%	48.0%	48.0%
	AB classifier	52.0%	**52.0%**	**52.0%**	**52.0%**
	2D-CNNpred	**53.0%**	51.0%	51.0%	50.0%
RUSSELL	Logistic regression	53.0%	**54.0%**	**54.0%**	**53.0%**
	AB classifier	53.0%	53.0%	53.0%	**53.0%**
	2D-CNNpred	**54.0%**	52.0%	52.0%	51.0%
S&P 500	Logistic regression	50.0%	50.0%	50.0%	50.0%
	AB classifier	49.0%	49.0%	49.0%	48.0%
	2D-CNNpred	**58.0%**	**57.0%**	**56.0%**	**56.0%**

Table 16: Summary of performance of the optimised logistic regression, AB classifier, and 2D-CNNpred for all datasets' testing sets. We observed that 2D-CNNpred achieved the highest accuracy for all datasets while maintaining high macro-average recall, precision, and F1.

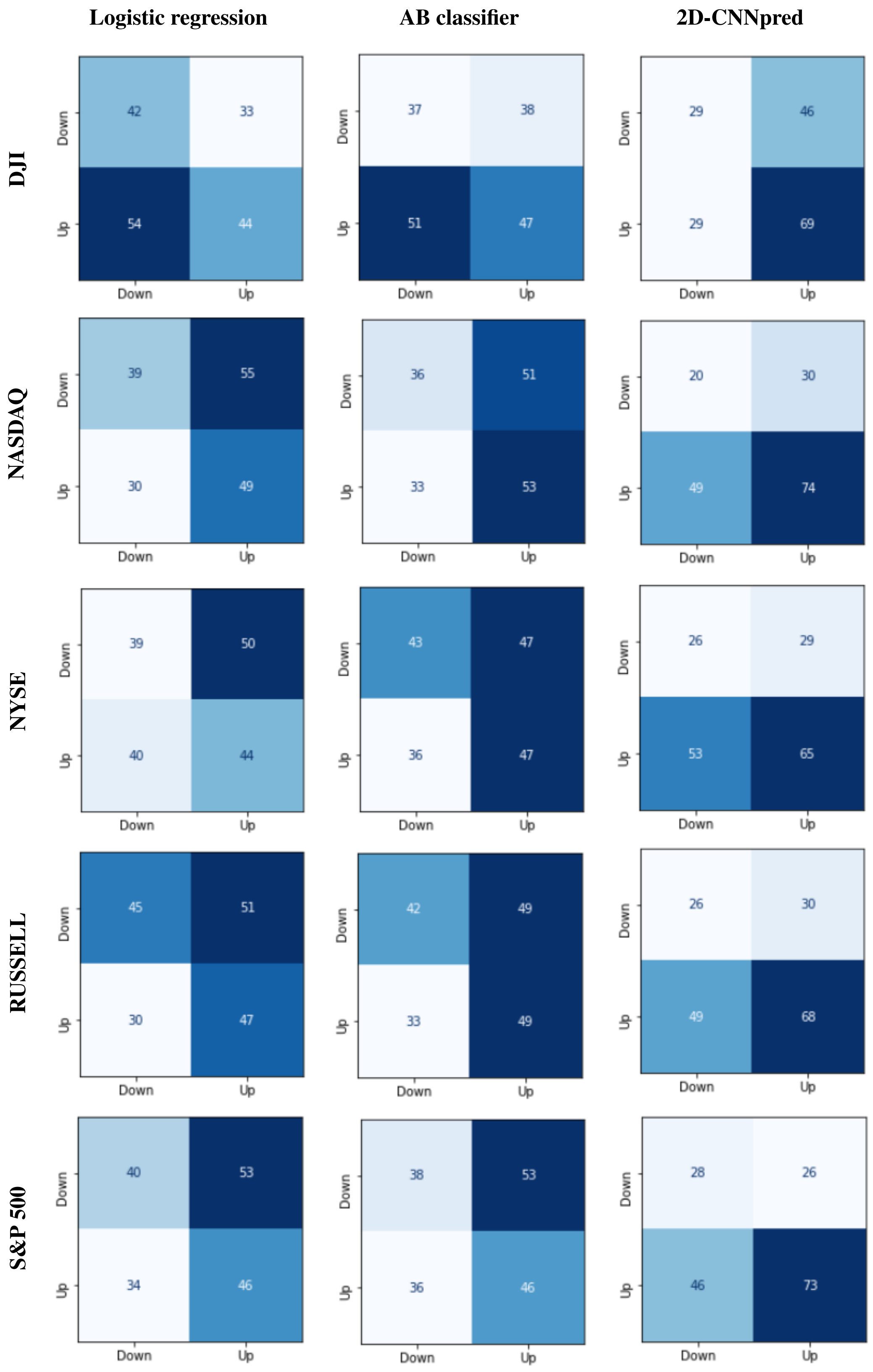
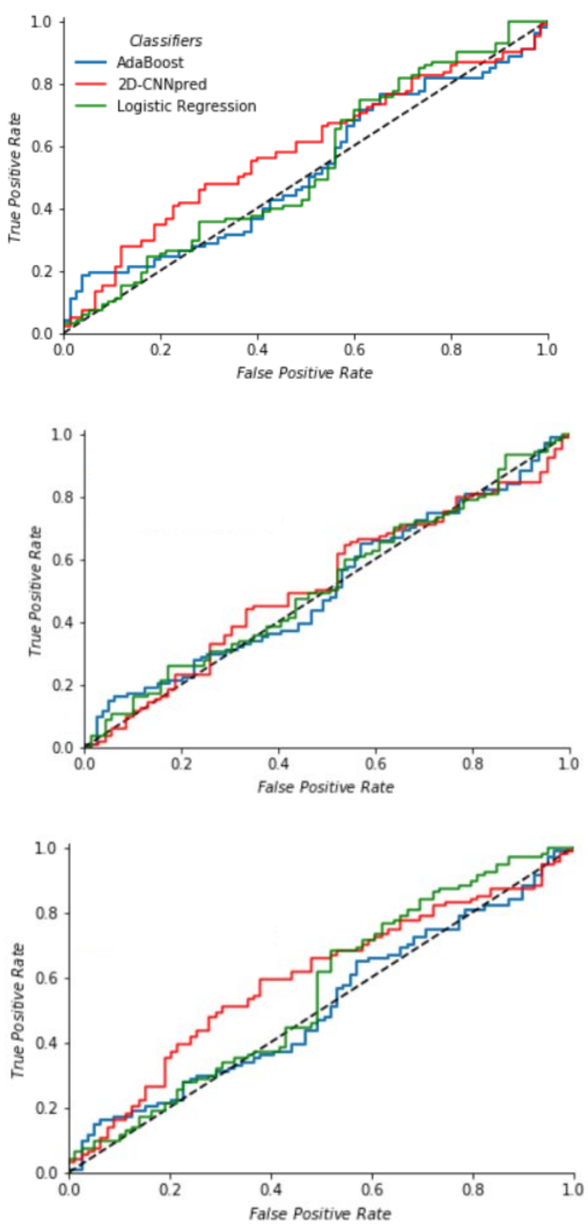
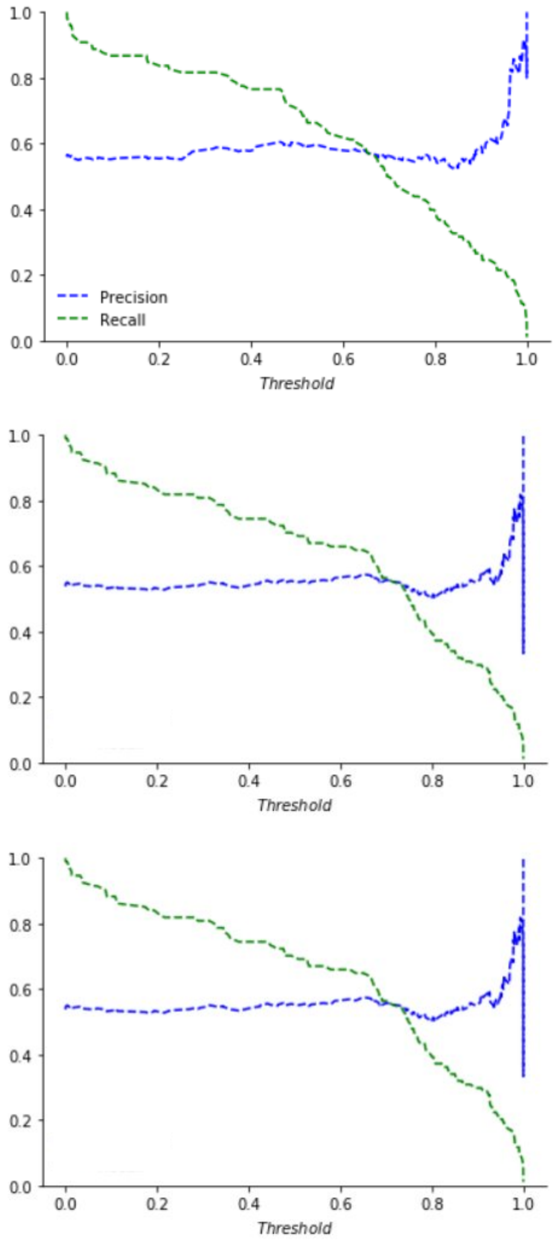
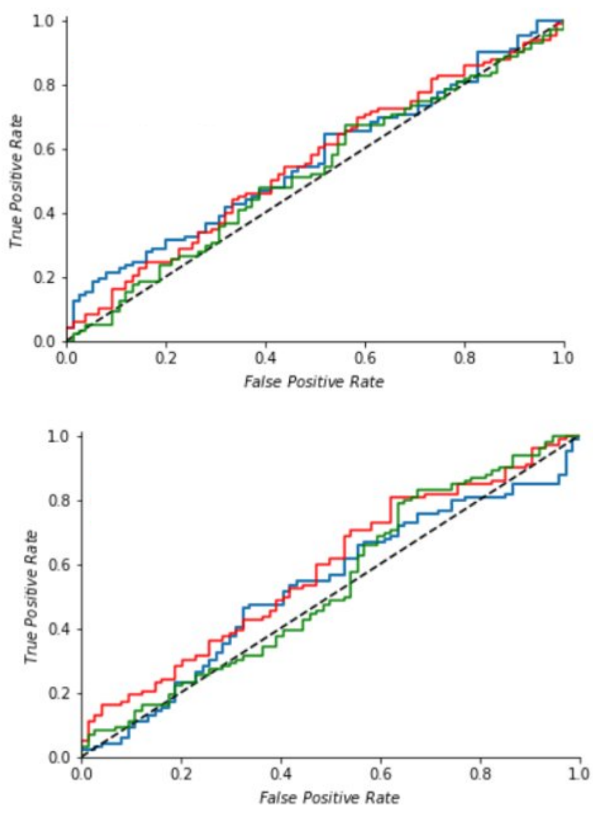


Figure 8: Confusion matrices of logistic regression, AB classifier, and 2D-CNNpred on all market datasets. The matrices have been annotated with their scores and coloured such that a higher score has a colour closer to dark blue. The rows of each confusion matrix are actual classes, while columns are predicted classes.

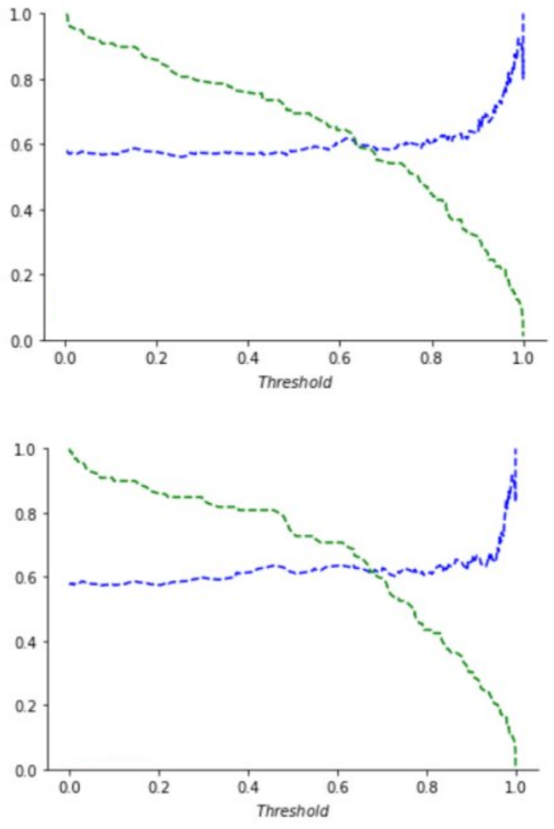
We plotted the ROC curves (Figure 9a) and calculated the AUC scores for all classifiers to directly measure how well these models could distinguish between classes regardless of different threshold values. Additionally, we plotted the precision-recall curve (Figure 9b) against various threshold values to visualise the optimal threshold value that balanced precision and recall for all classifiers for all datasets. Overall, the performances of all classifiers were marginally better than those of random classifiers, with the best AUC scores obtained by 2D-CNNpred (**57%, 52%, 53%, 54%, 58%**), followed by logistic regression (50%, 51%, 48%, 53%, 51%), and finally the AB classifier (50%, 51%, 52%, 53%, 50%).







(a) ROC



(b) Precision-recall

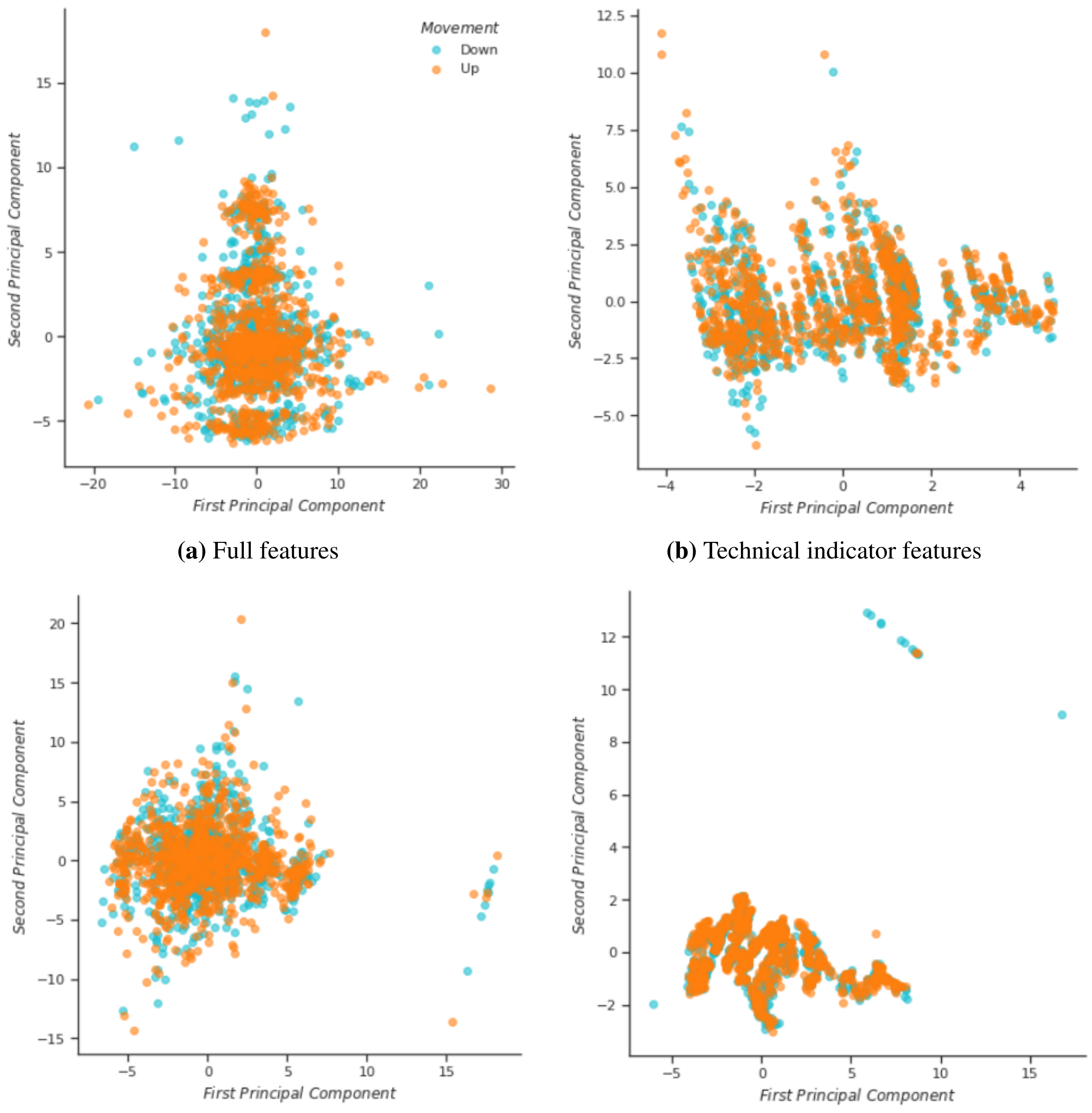
Figure 9: Receiver operating characteristic and precision-recall curves of logistic regression, AB classifier, and 2D-CNNpred for all datasets. Figures are ordered as follows: DJI, NASDAQ, NYSE, RUSSELL, S&P 500.

5 Dimensionality Reduction and Embeddings

This section discusses dimensionality reduction and embedding analysis using PCA and *t*-distributed stochastic neighbour embedding (t-SNE). We first standardised all features to ensure that they had the same variance before applying both PCA and t-SNE algorithms, preventing any single feature from dominating simply because of a different range of values. No other preprocessing steps were conducted, as all features were already in numerical format. The code for this section is available in `Dimensionality Reduction.ipynb`.

5.1 Principal Component Analysis

Figure 10 displays the projection of the first two principal components when applying the PCA algorithm to full features and selected sets of features, namely technical indicator features, economic features, and features strongly correlated (> 0.5) to the `Close` price. Both classes' instances substantially overlapped when applying PCA to both complete and selected feature sets, indicating difficulties for learning algorithms in distinguishing samples from different classes.



(a) Full features

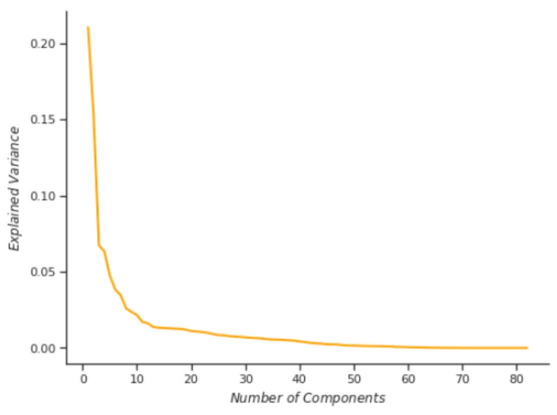
(b) Technical indicator features

(c) Economic features

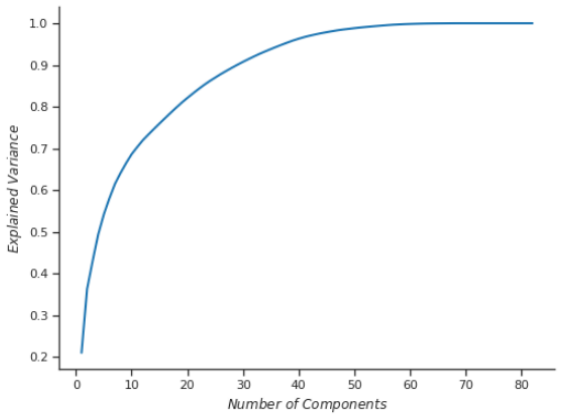
(d) Highly correlated features

Figure 10: Principal component analysis' scatter plot of the first two components when applied to different feature sets. Similar clustering were observed for all datasets.

Using the `explained_variance_ratio_` attribute of PCA, we calculated the amount for which each principal components explained the total variation of the original data, visualised in Figure 11a. We noted that the first 30 components contributed to 90.7% of the total variance of the data (Figure 11b), providing a compact feature set (less than half of the original features) for training classifiers, alleviating the *curse of dimensionality*. This formed a basis for our experimental assumption regarding the PCA feature set, explained in Section 3. We demonstrated similar performances for ML algorithms trained on the full and PCA feature sets.



(a) Explained variance ratio



(b) Cumulative explained variance ratio

Figure 11: Percentage explained variance and its cumulative sum for each principal component. We observed that the first 30 components explained more than 90% of the total variance of the original data.

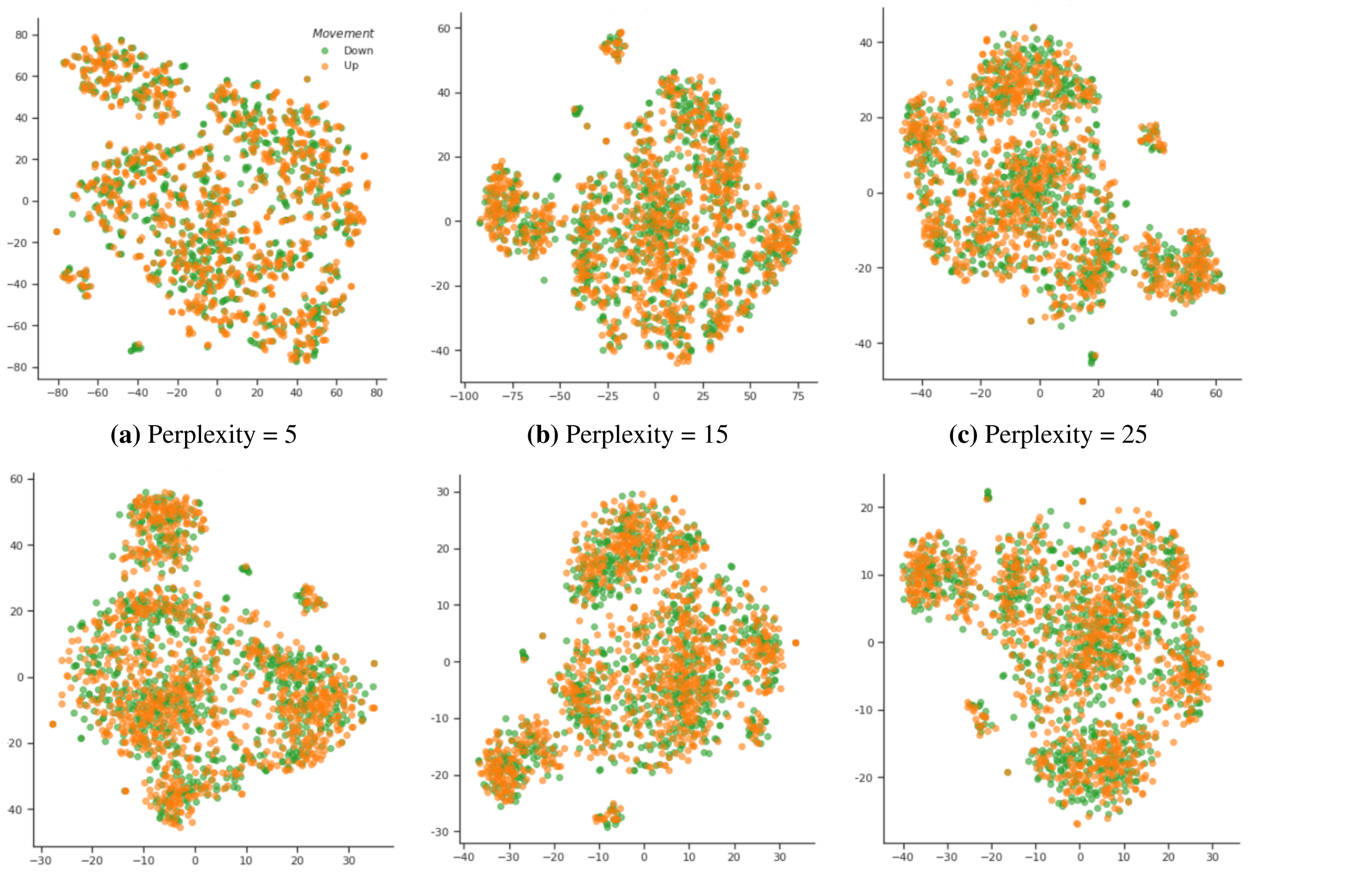
Finally, checking the `components_` attribute of PCA, we recorded the most influential feature of the first five principal components for all datasets, shown in Table 17. We observed the same most influential features for all five principal components for all datasets, except for the NYSE dataset. We observed different most influential feature for the first principal component (`mom`).

PCA	**Dataset**				
	DJI	**NASDAQ**	**NYSE**	**RUSSELL**	**S&P 500**
PC1	NYSE	NYSE	mom	NYSE	NYSE
PC2	DE4	DE4	DE4	DE4	DE4
PC3	DGS5	DGS5	DGS5	DGS5	DGS5
PC4	Dollar index-F	Dollar index-F	Dollar index-F	Dollar index-F	Dollar index-F
PC5	HSI	HSI	HSI	HSI	HSI

Table 17: The most influential feature for the first five principal components for all datasets.

5.2 *t*-Distributed Stochastic Neighbour Embedding

The t-SNE embeddings for different perplexities are shown in Figure 12. The perplexities were selected in the range of 5-50 to balance attention between the local and global aspects of the data [12]. Similarly to PCA, t-SNE embedding did not separate instances from both classes well, indicating difficulty in constructing accurate classifiers.



(a) Perplexity = 5 **(b)** Perplexity = 15 **(c)** Perplexity = 25

(d) Perplexity = 35 **(e)** Perplexity = 45 **(f)** Perplexity = 50

Figure 12: The t-SNE embeddings for different perplexities. Similar shapes were observed for all datasets.

6 Conclusion

This report presents a comprehensive study of how ML technologies are applied to predict the next days' direction of movements for various stock markets, namely DJI, NASDAQ, NYSE, RUSSELL, and S&P 500 indices, given historical financial variables. We applied various data analysis techniques, including standard and time series approaches, and reported the results. From the analysis insights, we built three distinct datasets for each index with different feature sets – full, reduced PCA, and technical indicator.

We implemented various ML algorithms, including standard (NB, logistic regression, k-NN, decision tree, SVM), ensemble (voting, bagging, pasting, AB, GB, stacking), and neural-based classifiers (FFNN, 2D-CNNpred, 3D-CNNpred, LSTM). These algorithms were then evaluated and compared for all five indices. The final results showed the superior performance of neural-based approaches compared to other approaches, highlighting their superior capabilities to automatically learn representations from raw features compared to manually constructed features. Among the algorithms on which we experimented, the 2D-CNNpred network architecture achieved the best performance. It improved the prediction performance in all five indices over the optimised standard and ensemble classifiers by approximately 1-8%, based on accuracy, F1, and AUC metrics.

7 Appendix

The list of raw initial features and its type:

#	Feature	Description	Type
1	Day	Which day of week	Primitive
2	Close	Close price	Primitive
3	Vol	Relative change of volume	Technical Indicator
4	MOM	Return of 1 day before	Technical Indicator
5	MOM-1	Return of 2 days before	Technical Indicator
6	MOM-2	Return of 3 days before	Technical Indicator
7	MOM-3	Return of 4 days before	Technical Indicator
8	ROC-5	4 days Rate of Change	Technical Indicator
9	ROC-10	10 days Rate of Change	Technical Indicator
10	ROC-15	15 days Rate of Change	Technical Indicator
11	ROC-20	20 days Rate of Change	Technical Indicator
12	EMA-10	10 days Exponential Moving Average	Technical Indicator
13	EMA-20	20 days Exponential Moving Average	Technical Indicator
14	EMA-50	50 days Exponential Moving Average	Technical Indicator
15	EMA-200	200 days Exponential Moving Average	Technical Indicator
16	DTB4WK	4-Week Treasury Bill: Secondary Market Rate	Economic
17	DTB3	3-Month Treasury Bill: Secondary Market Rate	Economic
18	DTB6	6-Week Treasury Bill: Secondary Market Rate	Economic
19	DGS5	5-Year Treasury Constant Maturity Rate	Economic
20	DGS10	10-Year Treasury Constant Maturity Rate	Economic
21	DAAA	Moody's Seasoned Aaa Corporate Bond Yield	Economic
22	DBAA	Moody's Seasoned Baa Corporate Bond Yield	Economic
23	TE1	DGS10-DTB4WK	Economic
24	TE2	DGS10-DTB3	Economic
25	TE3	DGS10-DTB6	Economic
26	TE5	DTB3-DTB4WK	Economic
27	TE6	DTB6-DTB4WK	Economic
28	DE1	DBAA-BAAA	Economic
29	DE2	DBAA-DGS10	Economic
30	DE4	DBAA-DTB6	Economic
31	DE5	DBAA-DTB3	Economic
32	DE6	DBAA-DTB4WK	Economic
33	CTB3M	Change in the market yield on US Treasury securities at 3-month constant maturity, quoted on investment basis	Economic
34	CTB6M	Change in the market yield on US Treasury securities at 6-month constant maturity, quoted on investment basis	Economic
35	CTB1Y	Change in the market yield on US Treasury securities at 1-year constant maturity, quoted on investment basis	Economic
36	Oil	Relative change of oil price(WTI), Oklahoma	Commodity
37	Oil	Relative change of oil price(Brent)	Commodity
38	Oil	Relative change of oil price(WTI)	Commodity

39	Gold	Relative change of gold price (London market)	Commodity
40	Gold-F	Relative change of gold price futures	Commodity
41	XAU-USD	Relative change of gold spot US dollar	Commodity
42	XAG-USD	Relative change of silver spot US dollar	Commodity
43	Gas	Relative change of gas price	Commodity
44	Silver	Relative change of silver price	Commodity
45	Copper	Relative change of copper future	Commodity
46	IXIC	Return of NASDAQ Composite index	World Indices
47	GSPC	Return of S&P 500 index	World Indices
48	DJI	Return of Dow Jones Industrial Average	World Indices
49	NYSE	Return of NY stock exchange index	World Indices
50	RUSSELL	Return of RUSSELL 2000 index	World Indices
51	HSI	Return of Hang Seng index	World Indices
52	SSE	Return of Shang Hai index	World Indices
53	FCHI	Return of CAC 40	World Indices
54	FTSE	Return of FTSE 100	World Indices
55	GDAXI	Return of DAX	World Indices
56	USD-Y	Relative change in USD to JPY exchange rate	Exchange Rate
57	USD-GBP	Relative change in USD to GBP exchange rate	Exchange Rate
58	USD-CAD	Relative change in USD to Canadian dollar exchange rate	Exchange Rate
59	USD-CNY	Relative change in USD to CNY exchange rate	Exchange Rate
60	USD-AUD	Relative change in USD to AUD exchange rate	Exchange Rate
61	USD-NZD	Relative change in USD to NZD exchange rate	Exchange Rate
62	USD-CHF	Relative change in US dollar to Swiss franc exchange rate	Exchange Rate
63	USD-EUR	Relative change in US dollar to Euro exchange rate	Exchange Rate
64	USDX	Relative change in US dollar index	Exchange Rate
65	XOM	Return of Exon Mobil Corporation	US Companies
66	JPM	Return of JPMorgan Chase & Co	US Companies
67	AAPL	Return of Apple Inc.	US Companies
68	MSFT	Return of Microsoft Corporation	US Companies
69	GE	Return of General Electric Company	US Companies
70	JNC	Return of Johnson & Johnson	US Companies
71	WFC	Return of Wells Fargo & Company	US Companies
72	AMZN	Return of Amazon.com Inc.	US Companies
73	FCHI-F	Return of CAC 40 Futures	Futures
74	FTSE-F	Return of FTSE100 Futures	Futures
75	GDAXI-F	Return of DAX Futures	Futures
76	HSI-F	Return of Hang Seng index Futures	Futures
77	Nikkei-F	Return of Nikkei index Futures	Futures
78	KOSPI-F	Return of Korean stock exchange Futures	Futures
79	IXIC-F	Return of NASDAQ Composite index Futures	Futures
80	DJI-F	Return of Dow Jones Industrial Average Futures	Futures
81	S&P-F	Return of S&P 500 index Futures	Futures
82	RUSSELL-F	Return of RUSSELL Futures	Futures
83	USDX-F	Relative change in USD index futures	Futures

Table 18: Description of used indices.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Yaser S Abu-Mostafa and Amir F Atiya. Introduction to financial forecasting. *Applied intelligence*, 6(3):205–213, 1996.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Guido J Deboeck. *Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets*, volume 39. John Wiley & Sons, 1994.

[5] Hakan Gunduz, Yusuf Yaslan, and Zehra Cataltepe. Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, 137:138–148, 2017.

[6] Erkam Guresen, Gulgun Kayakutlu, and Tugrul U Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

[7] Ehsan Hoseinzade and Saman Haratizadeh. Cnnpred: Cnn-based stock market prediction using several data sources. *arXiv preprint arXiv:1810.08923*, 2018.

[8] Ehsan Hoseinzade and Saman Haratizadeh. Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

[9] Michael C Jensen. Some anomalous evidence regarding market efficiency. *Journal of financial economics*, 6(2/3):95–101, 1978.

[10] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[11] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*, 2016.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[13] Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[14] Burton Gordon Malkiel. *A random walk down Wall Street: including a life-cycle guide to personal investing*. WW Norton & Company, 1999.

[15] Sohae Oh. *Multiple Imputation on Missing Values in Time Series Data*. PhD thesis, Duke University, 2015.

[16] Arzucan Özgür, Levent Özgür, and Tunga Güngör. Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences*, pages 606–615. Springer, 2005.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Xiao Zhong and David Enke. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139, 2017.