Capstone Project

Predicting Stock Price Movement Using Machine Learning

Project Definition

Problem Statement

Analyzing and predicting stock price movement have been a challenging but intriguing task for many analysts and researchers. A model that effectively predicts the direction of future stock price movement will aid investors and traders in making more profitable investments and being better at managing the risks of their portfolios.

Project Overview

Machine learning is gaining popularity in problem-solving in the finance domain. Financial institutions, such as investment companies, trading firms and hedge funds, build various financial models employing machine learning techniques to analyze market information in order to make better investment and trading decisions. As an engineer in a trading firm, I am also keen in attempting to solve a problem in the finance domain using machine learning algorithms and techniques learned in this course.

In this project, I aim to build a classification model that predicts the direction of stock price movement, namely up or down, N days ahead, with more focus on the 1-day ahead prediction. The classifier takes daily historical trading data over a certain date range for selected stocks as train and test datasets. The classifier outputs whether the stock price will move up or down N-day ahead from a selected date. In view of the time series financial data used, I explored several models suited for time series prediction such as Recurrent Neural Networks (RNN) [1], including Gated Recurrent Unit (GRU) and Long Short-Term Memory neural network (LSTM), using the Keras framework. I also explored a Random Forest classifier using scikit-learn. After a few iterations and model tuning, the model results were evaluated against the benchmark, Naive Method, using the accuracy metric.

Evaluation Metrics

This project deals with a classification problem where the target class is the direction of stock price movement, up or down, denoted as n-day_future_trend. A value of 1 indicates

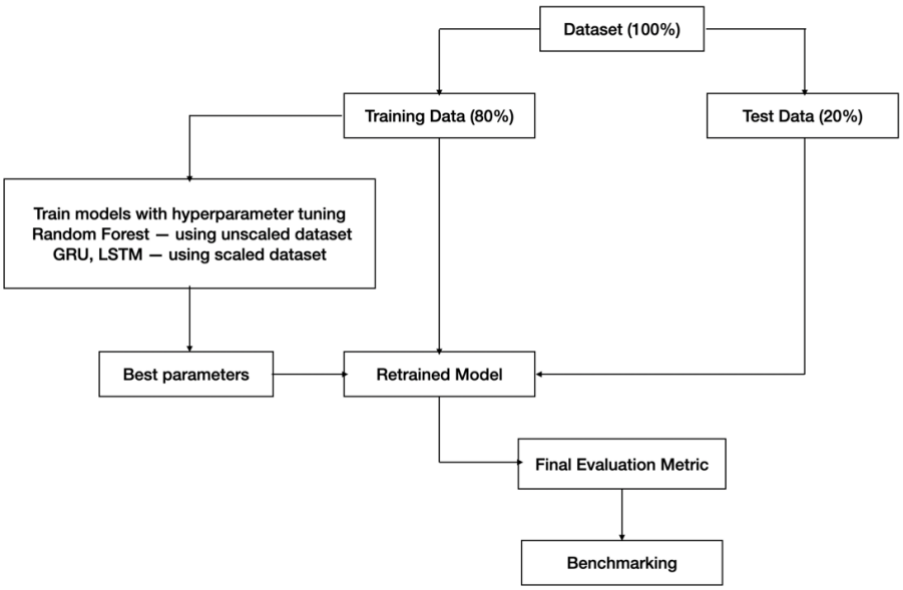
an upward movement, whereas 0 indicates otherwise. Accuracy, which is calculated as the total correct predictions divided by the total number of predictions, is used as the main evaluation metric.

$$Accuracy = \frac{Number\ of\ True\ Postives + Number\ of\ True\ Negatives}{Total\ Number\ of\ Predictions}$$

The distribution of the observations in the target class is also examined to identify any data imbalance [2], in which case accuracy alone might not adequately assess whether a model can identify the minority class. As shown in the analysis section below, the datasets used in this project are quite balanced. Therefore, Recall and Precision scores are not used as additional evaluation metrics in this project.

Workflow

Figure 1: Project Workflow



Analysis

Data Exploration and Visualization

Past 10 years' data of 10 well-known US stocks were used for exploratory data analysis and machine learning in this project. The historical stock prices were retrieved from Yahoo! Finance via the yahoofinancials API. The data consists of trading date (Date), opening price (Open), highest and lowest price traded in the day (High/Low), closing price adjusted for stock splits (Close), closing price adjusted for splits and dividends (Adjusted Close), and finally the trading volume (Volume).

The target for prediction is the direction of movement of the adjusted close price N-day ahead. The distribution of the observations in the target class was examined to identify any data imbalance that may add complexity in model training and evaluation. Table 1 summarizes the percentage of daily upward movement and downward movement for each selected stock over the last 10 years.

Table 1: Daily stock price movement from 2012-10-17 to 2022-10-17

	AAPL	MSFT	INTC	AMZN	NVDA	QCOM	TSM	IBM	GOOGL	TSLA
up_pct	52.25	52.96	51.65	53.44	53.24	52.01	51.85	51.13	52.60	52.33
down_pct	47.75	47.04	48.35	46.56	46.76	47.99	48.15	48.87	47.40	47.67

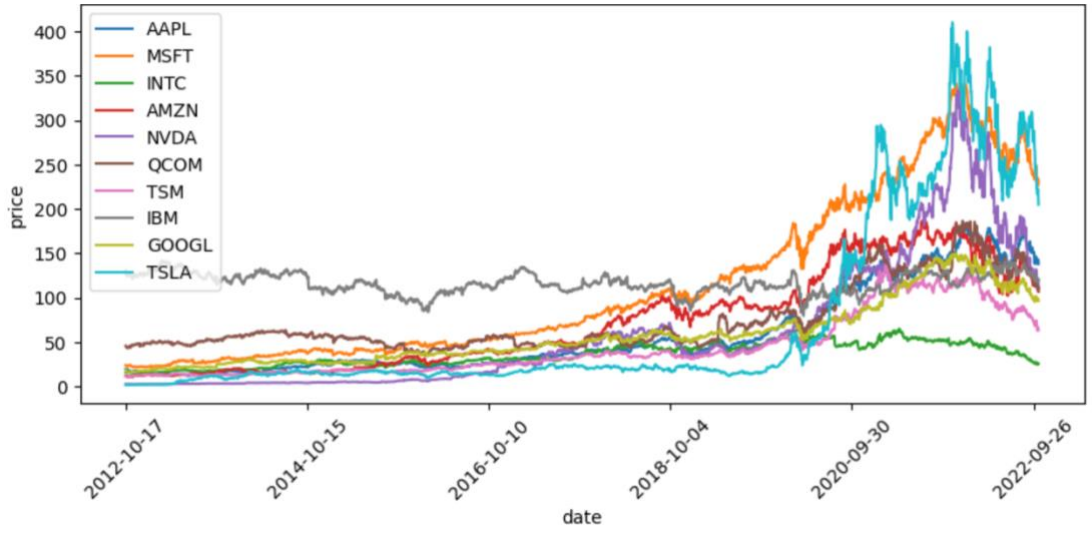
Where:

- up_pct: percentage of the number of days when the adjusted close price is higher than the previous day's
- down_pct: percentage of the number of days when the adjusted close price is lower than the previous day's

The datasets used in this project appear quite balanced as the percentage of upward movement and down movement for selected stocks are all close to 50%.

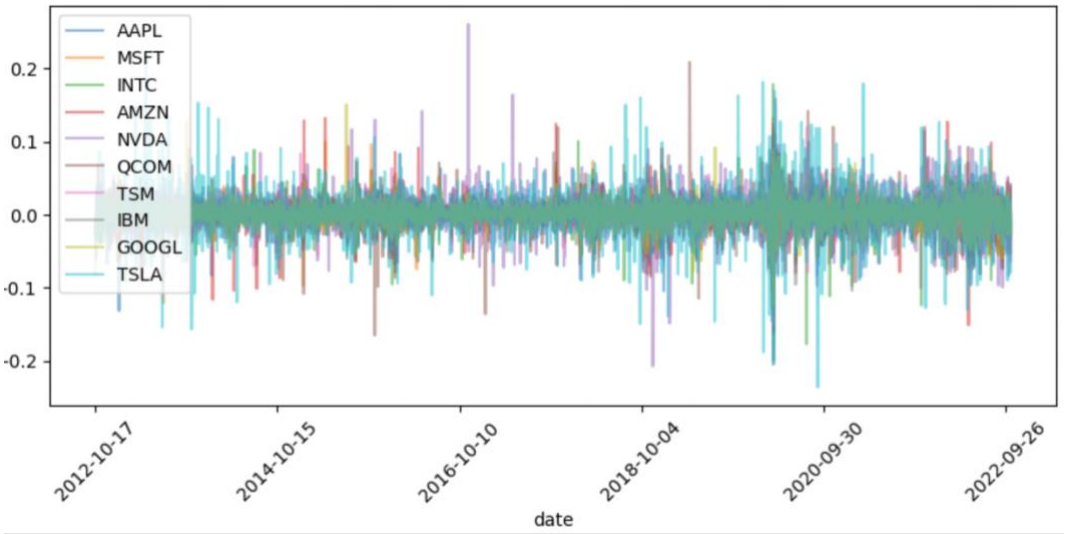
Figure 2 shows the trend of daily adjusted close price for the selected stocks over the last 10 years. An overall upward trend can be observed for most of the selected stocks except for IBM. Such data may not be suitable inputs for Machine Learning models that cannot handle non-stationary data well [3].

Figure 2: Daily Adjusted Close Price



Using first order differencing or percentage change usually is able to convert non-stationary times series data to stationary one. In fact, it is more interesting to know whether the stock price will move up or down and by what percentage rather than look at the absolute price. Figure 3 shows the daily return for all the selected stocks. Logarithmic return, which is commonly used in financial data modelling, is used in this project. No obvious trend of the daily returns can be observed from the plot.

Figure 3: Daily Return



More detailed analysis is done on a selected stock symbol AAPL. The approach to analysis is demonstrated in this report and could be easily adapted to analysing other stock symbols.

Feature Selection

The daily return of AAPL is approximately normally distributed with a slightly longer left tail. Partial autocorrelation [4] is also checked for the time series daily return data, where no strong autocorrelation can be observed. Whether Machine Learning can uncover any hidden patterns in this dataset will be examined later.

Figure 4: Trend and Distribution of Daily Return of AAPL

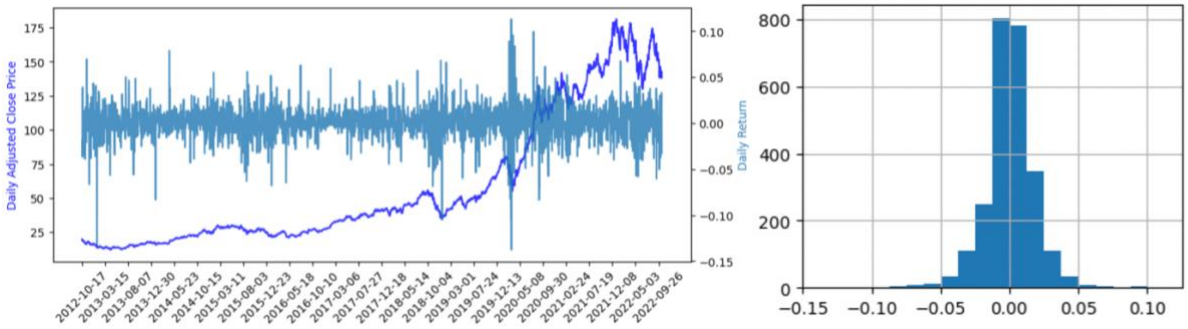
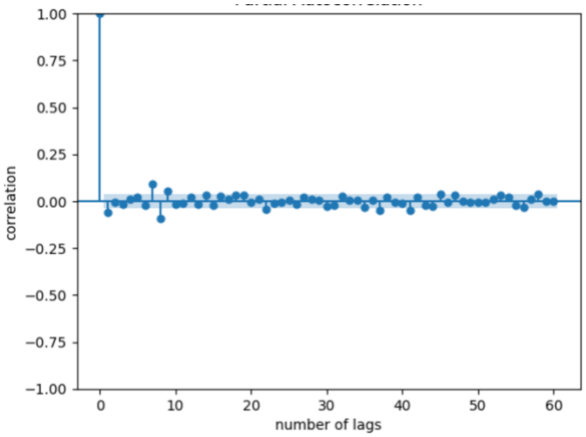


Figure 5: Partial Autocorrelation for Daily Return of AAPL



Several technical indicators [5] that are relevant to the stock price movement, including Exponentially Weighted Moving Average(EWMA), Relative Strength Index(RSI) and Moving Average Convergence/Divergence (MACD), are generated as additional features for exploratory data analysis and model training. To satisfy stationarity and ensure more accurate results, features in price terms are normalized against the adjusted close price.

Meanwhile, correlations across all features and the target class are examined. 1d_return has the strongest correlation at -0.055 to the target class. High, low, open, close and adjusted close price have a strong positive correlation close to 1 and all show an upward trend. Therefore, they are removed from the feature list.

Another consideration for feature selection is the requirement for the dimensionality of model inputs. In this project, two sets of features are used depending on individual model's requirements. As a Random Forest model requires 2-Dimentional inputs, utilizing past sequence data forces the number of features to be one so that the input dataset can be reshaped into 2D. Recurrent Neural Networks(RNN) take 3-Dimentional inputs, allowing multiple features to be used. Specifically:

- The first feature set only contains the 1d_return data, to be used for single/multiple-timestep RF and RNN models.
- The second feature set consists of 1d_return, volume and all technical indicators created above, to be used for single-timestep RF and RNN models.

Benchmark

Naïve method, which is often used for evaluating the accuracy of time-series forecasting, is chosen as the benchmark model. In the absence of seasonality, the naïve method is based on a random walk and each prediction is simply equal to the last observed value, in this case the direction of the previous day's stock price movement. Although naïve method itself might be overly simplistic, it serves as a useful benchmark for other forecasting methods and machine learning models in financial time series prediction. Using AAPL's data, the 1-day and 2-day ahead prediction accuracy using Naïve Method are 50% and 49% respectively. A machine learning model needs to beat the naïve method in order to be considered practical.

Methodology

Data Preprocessing

The past 10 years' stock data along with additional features created were split into training and test dataset by a default split ratio of 0.8, which is also configurable. Note that the train/test split cannot be random when dealing with time series data, especially when some of the models use past sequence data in a single data sample. Thus, the first 80% of days' data is used for training and the remaining 20% is used for testing.

Besides, look-ahead bias is carefully examined in order to avoid using future unknown data for prediction. As the target class is the direction of stock price movement for the next day, the last day's data from the training set is removed as its target class is unknown given the information in the training dataset itself. Similarly, the second last day's data is removed for a 2-day ahead predication.

Additional data preprocessing is done for different models based on their respective model input requirements. Before moving into the model-level details, below notation is used to facilitate the discussion of data dimensionality.

The training data fed into a machine learning model can be generalized to be in a shape of N x T x D, where:

- N is the number of samples
- T is the number of timesteps (in days in this project)
- D is the number of features

A Random Forest model requires a 2-Dimetional input data. This project examines two types of data fed into a Random Forest model:

i. The data contains no past sequence and is simply in the form of N x D, where T=1.
ii. The data contains past sequences and is in the shape of N x T, where D=1.

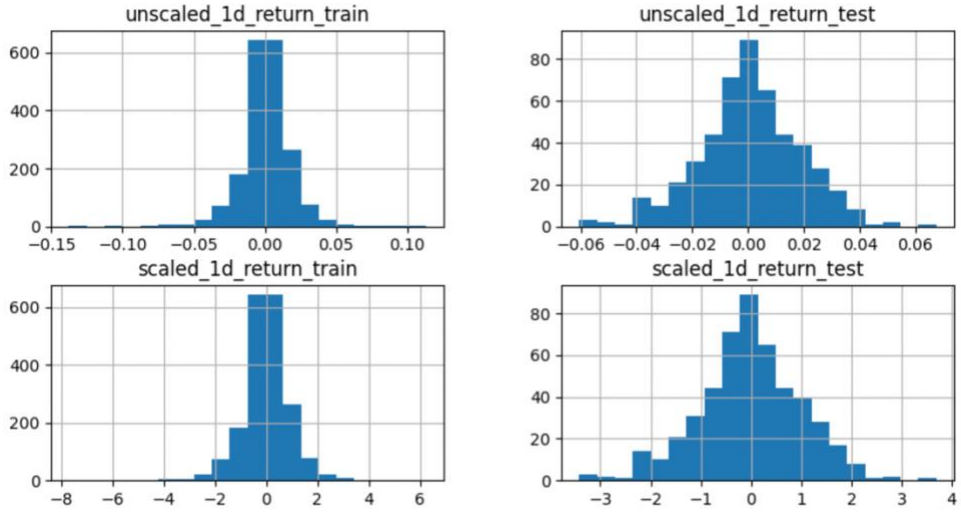
For Type II, 1d_return is chosen as the only feature due to its highest correlation with the target class. Past 30 days' 1d_return data is retrieved for each data sample and the entire dataset is stored as a N x 30 array. The number 30 is chosen based on hyperparameter tuning.

On the other hand, RNN models, in this project namely GRU and LSTM, require 3-Dimentional inputs. As such, more features can be used for training on top of past sequences. Past 30 or 60 days' data for all features is retrieved for each data sample and the entire dataset is stored as a N x T x D array. In this case, two sets of data are created:

i. In the shape of N x T x 1, which has 1 feature and is to be compared to the Random Forest model using past sequence data
ii. In the shape of N x T x 12, which includes all 12 features.

In addition, as the values of some of the features are small, Standard Scaling, which normalize the data into a distribution with a mean of 0 and standard deviation of 1, is applied to achieve more efficient and accurate model training. Figure 6 illustrates the distribution of the 1d_return feature before and after scaling.

Figure 6: Effect of Standard Scaling for AAPL



Implementation

Three types of machine learning models were used to build a classification model for predicting the direction of stock price movement 1-day or 2-day ahead. The workflow for each iteration includes the following steps:

- Parameterize and construct the ML model
- Train the model using the train dataset
- Evaluate the model using the test dataset
- Obtain the 1-day and 2-day ahead prediction accuracy for both train and test sets
- Evaluate the results against the benchmark model

Random Forest

Four variants of Random Forest models are constructed and evaluated using the scikit-learn framework:

- A simple model with max_depth=5 and trained using a single feature 1d_return, denoted as 'rfr'
- A model with max_depth=5 and trained using all 12 features, denoted as 'rfr_mf'
- A model trained using all 12 features and with hyperparameter tuning, denoted as 'rfr_hypo'
- A model trained using the best hyperparameters from the previous step and using 30-day past sequence data of 1d_return, denoted as 'rfr_recur'

Gated Recurrent Neural Network(GRU)

A recurrent neural network (RNN) is a type of artificial neural network that deals with sequential data or time series data. Gated Recurrent Unit(GRU) is a type of RNN which is controlled by "gates" that determine the weights of the past and new information. Two variants of GRU models are constructed and evaluated using the Keras framework:

- The first model, denoted as 'gru', consists of
    - an input layer of size 1 (1d_return)
    - a GRU layer with 50 neurons and 20 percent dropout to prevent overfitting
    - another GRU layer with 50 neurons and 20 percent dropout
    - an output layer with size 1 (target class)

- The second model, denoted as ‘gru_mf’, only differs from the first in the input size of 12 (12 features)

Long Short-Term Memory neural network (LSTM)

LSTM is another type of RNN model that has one additional gate compared to GRU. It is known to work well for longer sequences. Two variants of LSTM models are constructed and evaluated using the Keras framework:

- The first model, denoted as ‘lstm’, consists of
  - an input layer of size 1 (1d_return)
  - a LSTM layer with 50 neurons and 20 percent dropout to prevent overfitting
  - another LSTM layer with 50 neurons and 20 percent dropout
  - an output layer with size 1 (target class)
- The second model, denoted as ‘lstm_mf’, only differs from the first in the input size of 12 (12 features)

Refinement

Hyperparameter tuning has been done for all three types of models by looping through a parameter grid consisting of different of combinations of hyperparameter values.

Random Forest

Initial hyperparameter ranges tried:

- n_estimators: list(range(100,501,100))
- max_depth: list(range(5,31,5))
- min_samples_splits: list(range(2,7,1))
- min_samples_leaf: list(range(1,4,1))
- max_features: ['sqrt', 'log2', None],
- criterion: ['gini', 'entropy', 'log_loss']
- bootstrap: [True, False]

Observations:

- Increasing the number of estimators, min_sample_splits and min_samples_leaf did not significantly improve model performance.

- Larger max_depth tends to overfit -- giving high accuracy for the training dataset but low accuracy for test dataset. Training accuracy reached > 90% with 9+ max depths, while test accuracy stayed between 48% to 54%.
- No restriction on max_features yielded slightly higher accuracy than with restriction.
- Gini is used as the criterion for training since it slightly outperformed the other two.
- Bootstrap performed better than without boostrap.

After a few iterations, the best hyperparameters are used for subsequent model training and evaluation.

GRU and LSTM

Parameters tuned include batch_size in [16, 32, 64] and epochs in [10, 20, 30, 40, 50, 100]. In view of the training and testing sample size, which are 1903 and 432 respectively for a timestep of 60 days, larger batch size was not used. Besides, the training and testing accuracy did not seem to improve much with epochs larger than 100. The best-performing hyperparameters tried are used for subsequent model training and evaluation.

Results

Model Evaluation and Validation

Table 2 and Table 3 summarize the 1-day and 2-day ahead prediction accuracy for the selected stock using various models.

Table 2: 1-day ahead prediction accuracy for AAPL

1d_ahead	naive	rfr	rfr_mf	rfr_hypo	rfr_recur	gru	gru_mf	lstm	lstm_mf
train	0.486996	0.619776	0.672783	0.579001	0.604759	0.527588	0.538623	0.655807	0.716763
test	0.5	0.494908	0.507128	0.535642	0.521645	0.5231	0.5347	0.5694	0.5486

Table 3: 2-day ahead prediction accuracy for AAPL

2d_ahead	naive	rfr_recur	gru_mf	Lstm_mf
train	0.502551	0.600414	0.523134	0.659306
test	0.486708	0.509761	0.5012	0.5151

Most of the machine learning models yield higher test accuracy, up to 6.9% for 1-day ahead prediction and up to 2.8% for 2-day ahead prediction, compared to the Naive benchmark model. However, the improvement is not considered significant.

Among Random Forest models, the one trained with multiple features and with hyperparameter tuning yield the highest test accuracy. However, the difference in model performance is not significant.

GRU's performance appears comparable with that of Random Forest models with hyperparameter tuning.

LSTM has up to 23% higher training accuracy than the benchmark model, but relatively smaller advantage in test accuracy, up to 6.9% improvement.

Table 4 and Table 5 summarize how 1-day ahead prediction accuracy changes with different timestep, batch_size and epochs settings for the GRU and LSTM models respectively. It can be observed that there is no obvious correlation between timestep/batch_size and the train/test accuracy for both models, and between epochs and the train/test accuracy for the GRU model. Although training accuracy increases with longer epochs for LSTM, the test accuracy does not improve proportionally.

Other tunings, such as adjusting dropout ratio/number of neurons in each layer for both GRU and LSTM model, and trying different combination of features available in the current datasets, are explored. However, the results are not promising and thus the details are not included in this report.

Table 4: 1-day ahead prediction accuracy for AAPL with GRU hyperparameter tuning

GRU	Batch Size	16		32		64	
Timestep	Epochs	Train	Test	Train	Test	Train	Test
60	10	0.5208	0.5162	0.5271	0.5069	0.5244	0.5069
	20	0.5376	0.5023	0.5339	0.5093	0.5381	0.5069
	30	0.5386	0.5046	0.5307	0.4815	0.5397	0.5208
	40	0.5533	0.5208	0.5465	0.5116	0.5307	0.5185
	50	0.5691	0.5532	0.5439	0.4815	0.5255	0.4977
	100	0.5612	0.4838	0.6217	0.5301	0.5638	0.5116
30	10	0.5178	0.5173	0.5349	0.4848	0.5323	0.513
	20	0.5365	0.5195	0.5468	0.5065	0.537	0.5
	30	0.5308	0.4913	0.5427	0.5216	0.5194	0.5541
	40	0.5504	0.5325	0.5473	0.5043	0.5515	0.5173
	50	0.5577	0.5346	0.5458	0.5238	0.5561	0.513
	100	0.6177	0.5043	0.5654	0.5195	0.598	0.5325

Table 5: 1-day ahead prediction accuracy for AAPL with LSTM hyperparameter tuning

LSTM	Batch Size	16		32		64	
Timestep	Epochs	Train	Test	Train	Test	Train	Test
60	10	0.5391	0.4931	0.5286	0.5324	0.5544	0.5324
	20	0.5812	0.4907	0.578	0.5278	0.5607	0.5324
	30	0.6322	0.5162	0.6132	0.5509	0.5885	0.5208
	40	0.6726	0.5255	0.6379	0.5394	0.64	0.5417
	50	0.732	0.5162	0.69	0.5162	0.6448	0.5417
	100	0.8807	0.5347	0.7977	0.5093	0.7514	0.5093
30	10	0.5499	0.5303	0.5303	0.5108	0.537	0.5043
	20	0.5851	0.5216	0.5794	0.5303	0.5613	0.5238
	30	0.6342	0.5195	0.6234	0.5238	0.5841	0.5325
	40	0.6684	0.5238	0.6461	0.5325	0.6446	0.5433
	50	0.7227	0.487	0.7181	0.5216	0.6596	0.526
	100	0.9364	0.5108	0.8769	0.5455	0.777	0.513

Justification

In this project, machine learning models suitable for time series data prediction are used to build a classification model for predicting the direction of stock price movement of the next n-day. Proper ML methodology, techniques and workflow are followed for model training and optimization. However, by up to 7% higher accuracy, these models only barely outperform the Naïve Method, which requires no parameterization and simply assumes each prediction being equal to the last observed value.

Why is the accuracy of ML models not high in this specific case? One possible reason is that crucial factors that affect stock price movement are not captured in the available dataset used for model training. To name a few -- overnight news, unexpected events, politics, market sentiments, or even some celebrity posting a tweet -- these factors may not be quantified into the training data and used for model training in the previous day. On the other hand, if a model can easily predict the direction of stock price movement with publicly available data at high accuracy, such model would be exploited in a real investment or trading strategy and make a plenty of money. Meanwhile, majority of market participants might seek to adopt the same model whereby market dynamics could change again, making the model's superior performance no longer hold.

Conclusion

Reflection

In this project, I used Random Forest, Gated Recurrent Unit(GRU) and Long Short Term Memory neural network(LSTM) along with time series stock data to build a classification model for predicting the direction of stock price movement for the next 1 day or 2 days, and benchmarked the prediction accuracy against the Naive Method. Some key observations:

- Prediction accuracy of machine learning models for this specific problem is not higher than 60%.
- Machine Learning models outperform the Naive Method by up to 7% higher test accuracy. However, this improvement is not significant compared to some of other machine learning classification use cases.
- Prediction accuracy for 1-day ahead is up to 5% higher than for 2-day ahead.
- LSTM which utilizes past sequence data appears to have the best performance among the models evaluated. However, longer timesteps do not significantly improve model performance.
- Models trained using multiple features tend to but not always have higher prediction accuracy than those trained using a single feature for this specific problem.

Although with improved prediction accuracy, the ML classification models examined in this project do not adequately address the problem of predicting the direction of future stock price movement. Several aspects including data completeness, as explained in the earlier section, need to be further evaluated and addressed.

In view of the accuracy level attainable, the python script, mentioned in the project proposal, that implements the classifier using the best-performing model is not coded, as 7% improvement in the prediction accuracy is insignificant for a reliable prediction. Meanwhile, the relevant methodology and code for implementing the classifier have been demonstrated in the jupyter notebook submitted.

Improvement

From data's perspective, adding and tracking additional metrics such as market events and events' impacts to the stock price, e.g. expected percentage contribution to the stock price

change, impact factor on a scale of 0 to 10, etc, may be useful and likely to enable higher predictive power of machine learning models. Web scrapers can be used to retrieve market news and events for establishing such metrics. However, it may be challenging to restore and backfill these metrics for the past few decades for each stock symbol and maintain such data as a private competitive edge. Alternative data sources that provide relevant information may also be explored. On the other hand, more technical indicators such as Bollinger Bands and standard deviation of stock price can be evaluated as additional features for any potential contribution to model performance.

From model training's perspective, if more time were given, additional optimization can be explored, such as cross-validation and further hyperparameter tuning. Besides, the models can be evaluated using more stock symbols to ensure consistency of model performance.

Difficulties

The major difficulty I encountered was during the initial research phase of this project, when I was overwhelmed by the misleading information about time series prediction using LSTM and its overstated prediction performance claimed by many online contents. Instead of predicting returns, such online contents predict one-step ahead stock prices using only past price data, and present predictions closely tracking the actual stock prices. However, after a careful examination, such predictions are merely lagging the actual stock prices. An LSTM model trained using stock price sequence data does not see the full range of the stock prices in the test dataset as the stock prices are trending up. It might put more weight to the latest observations, making each prediction close to the last observed value. The seemingly accurate prediction does not differ significantly from a prediction using the Naïve Method, and has little practical use. A lot of time was spent on understanding, evaluating and recognizing the issues with such prediction methodology and misleading presentation of the results. On the other hand, this exercise helps me to think more critically when reading online contents about machine learning and to cross-check against more reliable sources.

Another difficulty is with package installation on my laptop which uses Apple M1 chip. A lot of time was spent on researching, troubleshooting and reinstalling packages that need to both be compatible with the Apple M1 chip and satisfy interdependencies. For example,

TensorFlow does not officially support Mac M1 yet, therefore I need to research and install the tensorflow-macos package maintained by Apple. These installation procedures have been documented for potential future reference.

Acknowledgements

I would like to thank Udacity for providing the Machine Learning Engineer Nano Degree courses and the capstone project where I can apply the machine learning knowledge and techniques gained to a customized project of my own interest.

I would also like to thank Lazy Programmer Inc. whose Youtube series regarding the common mistakes people make in stock price prediction help me streamline the methodology I use in this project.