STAT4012 Statistical Principles of Deep Learning with Business Applications

2ed Term, 2022-2023

Group 7 Project Report

Stock Return Prediction with CNN and Attention-based LSTM models

Written by

LIU Xiaolin

ZHANG Haoxiang

ZHANG Puming

Abstract

In this project, we aimed to predict the future stock returns of Tesla using effective deep learning models, including Convolutional Neural Networks (CNN) and Attention-based Long Short-Term Memory (LSTM) networks. To construct profitable trading strategies, we conducted both regression and classification for return prediction. For the regression task, we attempted to predict future stock returns and evaluated them using Mean Squared Error (MSE). For the classification task, we predicted the sign of future stock returns (positive or negative) and evaluated using measures such as F1 score and ROC curve.

To test the effectiveness of our models, we conducted back-testing using historical data and compared the results with a traditional GARCH model. Our results showed that both the CNN and Attention-based LSTM models outperformed the GARCH model in predicting the future stock return of Tesla and could generate sound returns. Moreover, the classification result is more reliable than the regression result when serving as the trading signal.

Overall, our project demonstrates the effectiveness of deep learning models in predicting stock returns. However, further improvements are still needed to achieve better performance.

Table of Contents

1. Introduction

Predicting stock returns is a challenging task due to the complexity and unpredictability of the financial market. According to the S&P SPIVA report analysis (Coleman, 2019), more than 80% of actively managed funds underperform their respective benchmarks. However, we hope to leverage the power of deep learning tools to detect the hidden determinants of stock returns and find their relationships.

2. Data Preprocessing

2.1 Data Collection

In this project, the data used was obtained from the Bloomberg terminal. We focused on Tesla, Inc. (TSLA), which is a volatile stock with high liquidity. The data was collected for more than five years, from Oct. 24, 2017, to Apr. 13, 2023, which provides a total of 1376 observations for our analysis. The historical pricing data retrieved includes open, close, high, low, and trading volume. Additionally, we retrieved other indicators including relative strength index (RSI(14)), bollinger bands width, percent B, BIAS (25), and rate of change (ROC(1)), which may provide more embedded information to predict future return movements.

2.2 Normalization

To ensure that no single feature dominated our analysis due to differing magnitudes, we employed min-max normalization to scale our feature variables to the same range [Figure 1]. The formula applied to the d-th feature variable ($x^{(d)}$) with N observations:

$$\tilde{x}^{(d)} = \frac{x^{(d)} - \min\limits_{n=1,\dots,N}\{X_n^d\}}{\max\limits_{n=1\dots,N}\left\{x_n^{(d)}\right\} - \min\limits_{n=1,\dots,N}\left\{x_n^{(d)}\right\}} \in [0,1]$$

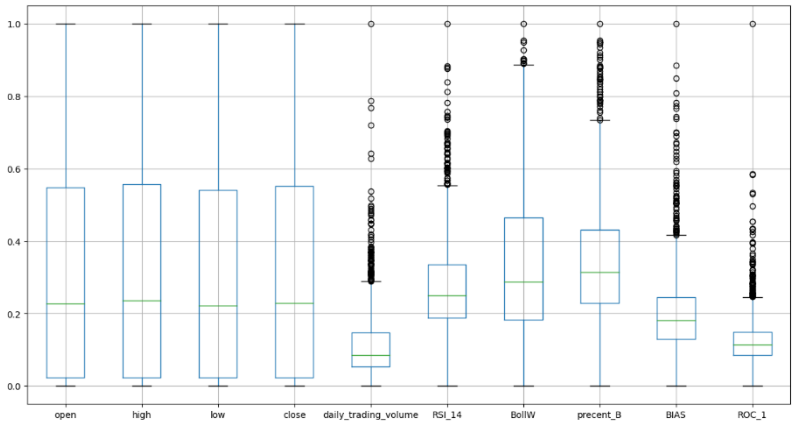


Figure 1: Boxplot of 10 Normalized Feature Variables

2.3 Principal Component Analysis (PCA)

In order to address the problem of high collinearity among our features [Figure 2] and reduce redundant information, we performed Principal Component Analysis (PCA) to reduce input feature dimensionality and promote independence. We used two PCA approaches: one directly performed PCA on normalized features, while the other performed PCA on the difference of the normalized features. The latter approach aimed to extract features that could represent changes in the features.

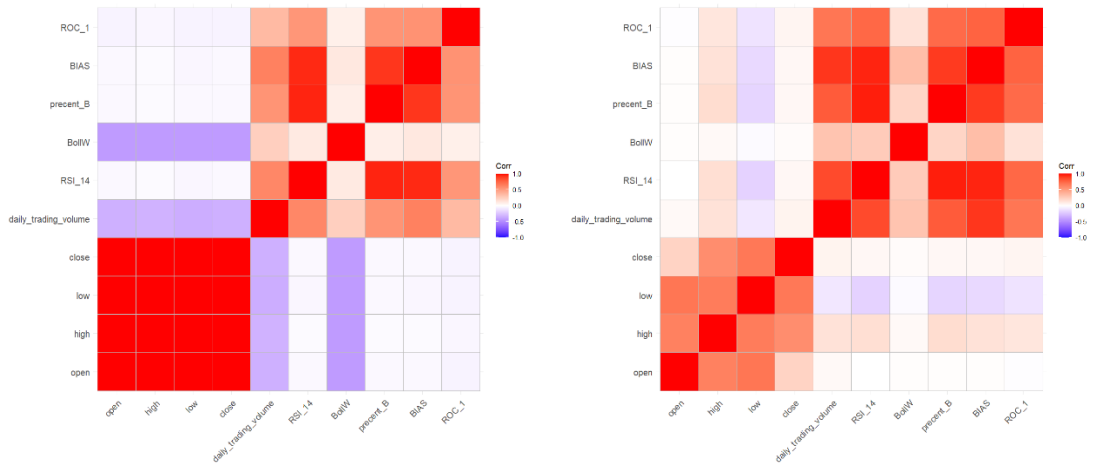


Figure 2: Correlation Heatmap of Normalized Data (Left) & Changes of Normalized Data (Right)

We performed PCA on the covariance matrix for both methods and utilized the cumulative explained variance ratio to determine the optimal number of principal components to retain. Our threshold for retention was set at 95%. We found that the first three principal components could explain over 95% of the total variance for both of our PCA methods [Figure 3].

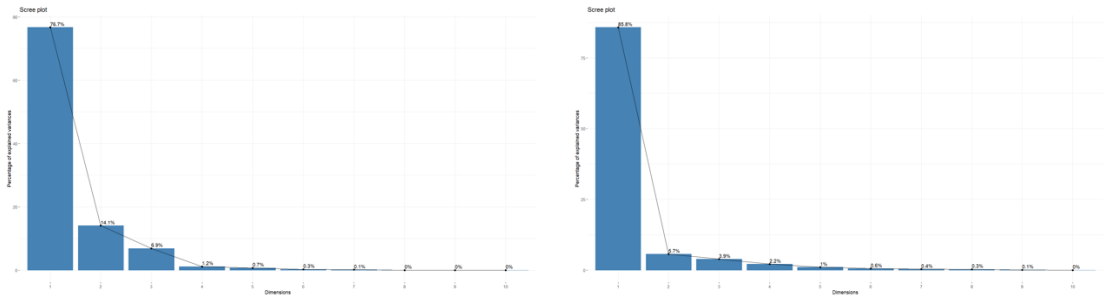


Figure 3: Scree Plot of PCA on Normalized Data (Left) & Changes of Normalized Data (Right)

2.4 Feature and Label Construction

After conducting PCA and selecting the first three principal components, we created two sets of new features. Each set comprises three new features obtained by combining the loadings and normalized data through linear combinations.

We also combined each set of new features with two commonly used technical analysis indicators, the golden cross and the death cross, resulting in a total of 5 input features in each set [Figure 4].

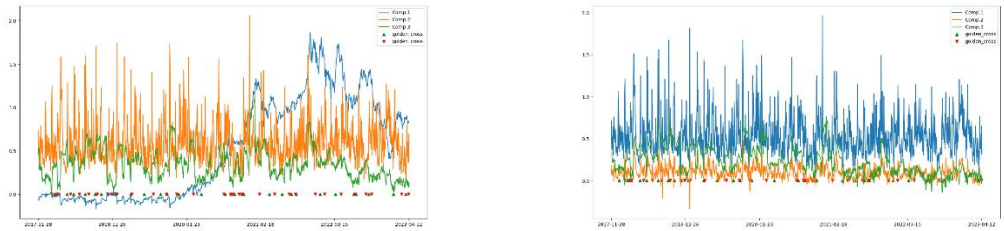


Figure 4: Constructed Features by PCA on Normalized Data (Left) & Changes of Normalized Data (Right)

The golden cross and death cross are formed by the intersection of the 5-day and 25-day moving average of the closing price. The golden cross indicates a bullish trend in

the market when the 5-day moving average crosses above the 25-day moving average. Conversely, the death cross indicates a bearish trend when the 5-day moving average crosses below the 25-day moving average. In our dataset, we found 32 instances of the golden cross and death cross [Figure 5]. We included these patterns as binary variables in our input feature variables, with a value of 1 indicating the presence of a golden cross or death cross. We replaced some missing values with 0 due to the calculation of moving averages for the last few samples.

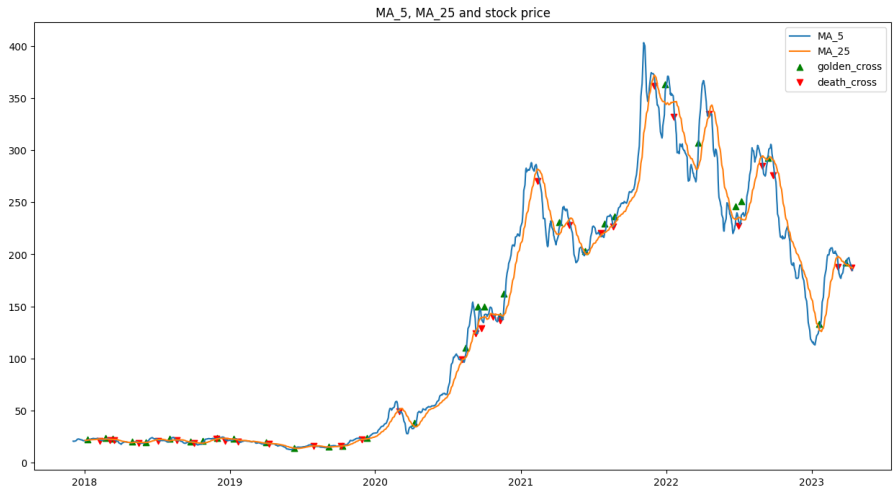


Figure 5: Golden Cross & Death Cross with MA(5) & MA(25)

We chose to use one-day return as the prediction target, which is calculated as the percentage change in the stock's closing price from the previous day's closing price [Figure 6]. i.e.

$$\frac{\text{Today's Close Price} - \text{Previous Day's Close Price}}{\text{Previous Day's Close Price}}$$

Our decision to use the closing price to calculate returns was made under the assumption that pre-market activity would not significantly affect the open price. Therefore, we did not take into account the price gap between the previous day's closing price and today's opening price. Given the previous day's closing price, if our model predicts a positive return, we would buy at the opening price of the current day and sell

at the closing price of the same day.

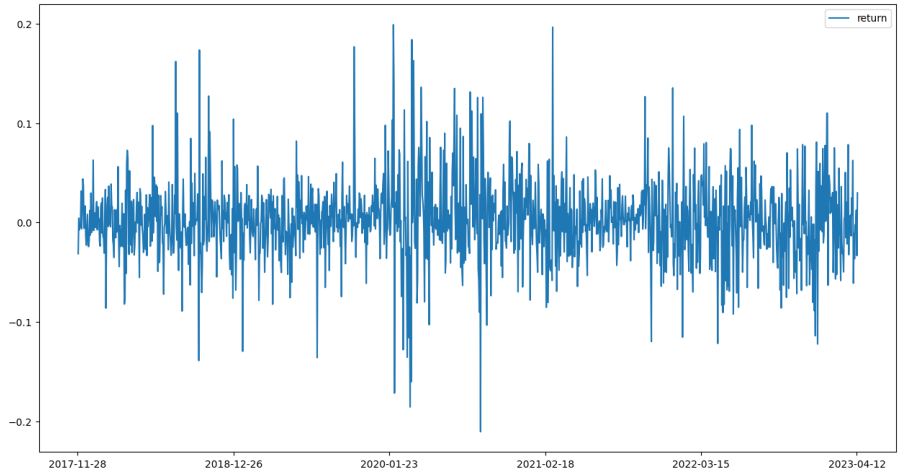


Figure 6: One-day Return Calculated by Close Price

2.5 Feature Reshape and Train Test Split

We divided the data into 5-day or 10-day intervals (n_days) and used the 6th or 11th day's return as the label for each interval. The data was grouped with a stride of 1 day ($stride$), resulting in an input data dimension of $(\frac{1353 - n\_days}{stride}, n\_days, 5)$. After removing the last several label entries to match the length of input features, the label data was reshaped to $(\frac{1353 - n\_days}{stride},)$.

Since our input data is time-series data, we split the data into a training set consisting of the first 80% of the data and a testing set consisting of the last 20%. This ensures that the model is trained on past data and tested on future data [Figure 7].

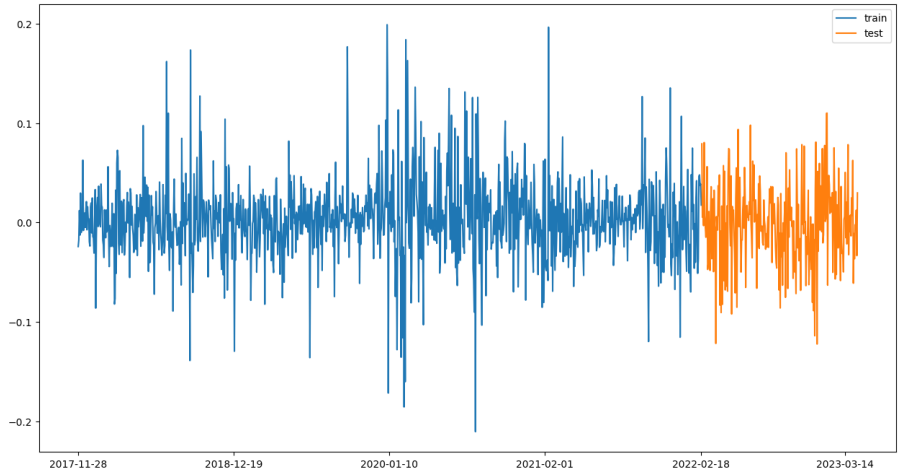


Figure 7: Train Test Split Visualized in Label

3. CNN Model

3.1 CNN Model Architecture

We designed 8 CNN models with a shared basic structure consisting of 1D convolutional layers, a flatten layer, and dense layers. The models were modified by adjusting the number of convolutional layers, the number of filters in each convolutional layer, the number of dense layers, the number of neurons in each dense layer, and n_days to determine the optimal model for generating the best results [Figure 8, Figure 9].

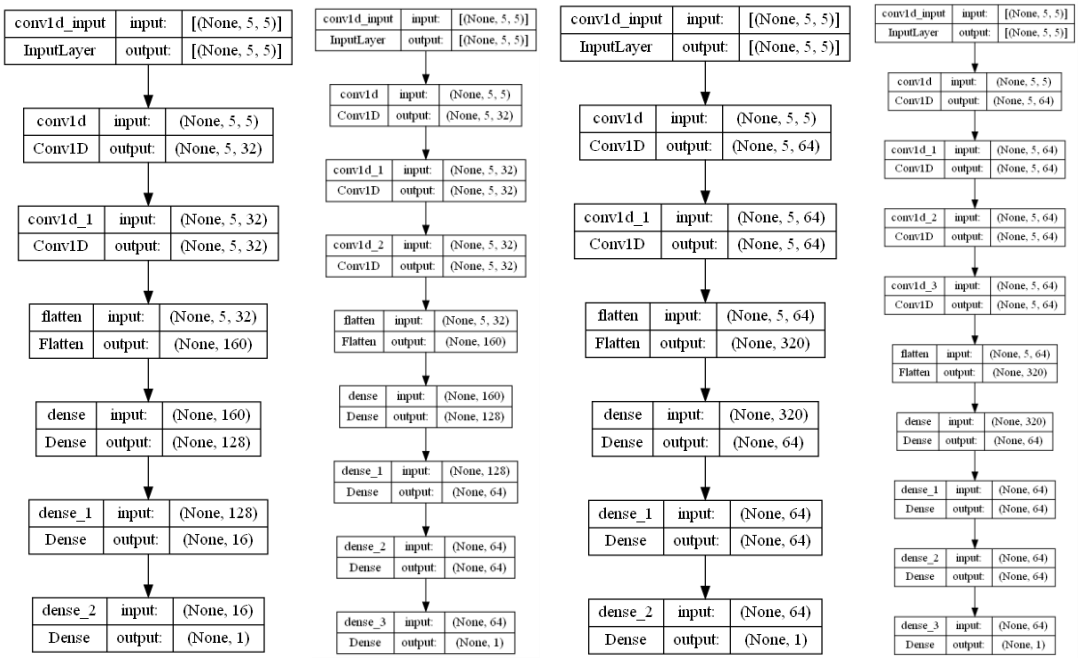


Figure 8: CNN Model 1 (Left 1), CNN Model 2 (Left 2), CNN Model 3 (Left 3), CNN Model 4 (Left 4)

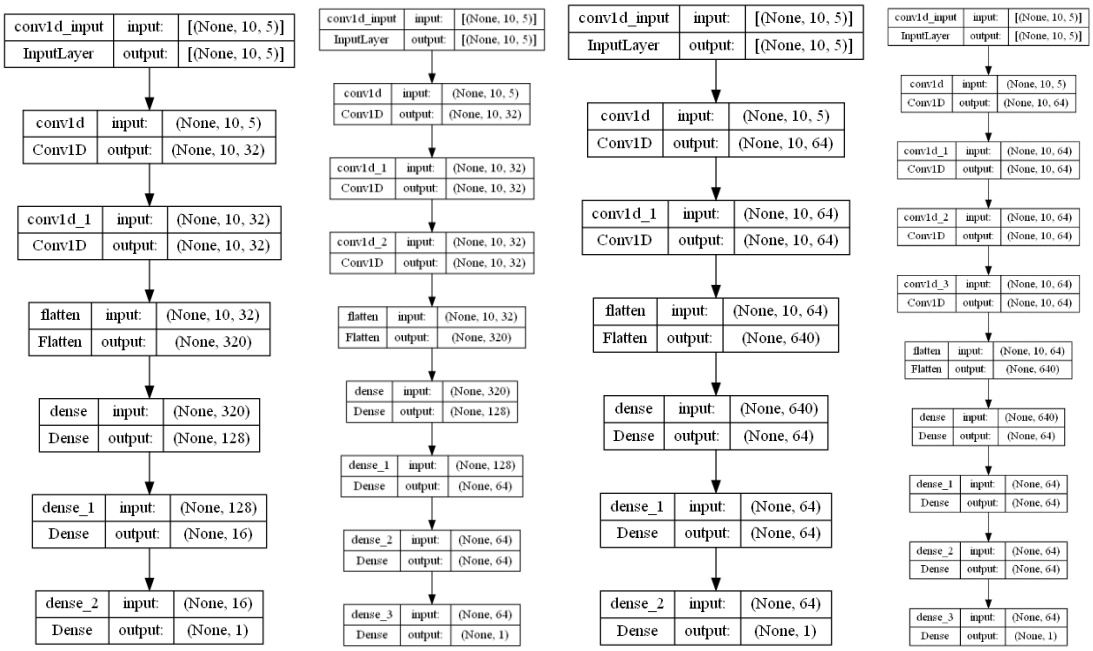


Figure 9: CNN Model 5 (Left 1), CNN Model 6 (Left 2), CNN Model 7 (Left 3), CNN Model 8 (Left 4)

Given that our input data consists of time series data, it is more appropriate to use a 1D convolutional layer rather than a 2D convolutional layer. Since 1D convolutional layer is specifically designed to operate on a sequence of data points. We have decided not to include pooling layers in our models since our dataset has a relatively small number of features and pooling may result in the loss of important information.

3.2 CNN Model Training

To ensure consistency, we established the following configuration for training our CNN models. All models will use Rectified Linear Unit (ReLU) as the activation function, and a kernel size of 2 for all convolutional layers. We will utilize Adaptive Moment Estimation (Adam) as our optimizer, with an initial learning rate of 0.0001. The loss function is specified as Mean Squared Error (MSE). The models will be trained for 100 epochs.

During the model training process, we tuned several hyperparameters to optimize the results. These included the random seed used for initialization, the $stride$ for grouping input feature variables, and the sets of input features, as we formed two sets of input features after performing PCA.

3.3 CNN Model Evaluation

3.3.1 CNN Model Regression Evaluation

The 8 original CNN models were designed for regression prediction, as evidenced by the absence of an activation function in the output layer. These models were used to predict the return for the next day using the testing set. However, since the majority of returns fall between -0.1 and 0.1, even a small deviation between the predicted and true return can have a significant impact on the accuracy score. As a result, the accuracy scores for these models were almost 0.

To better illustrate the prediction results, we can reverse engineer the predicted return for the i-th day ($\hat{r}_i$) to its corresponding close price ($\tilde{p}_{(close,i)}$). This can be done by multiplying the predicted return for the i-th day with the close price for the (i-1)th day ($p_{(close,i-1)}$) plus one. i.e.

$$\tilde{p}_{(close,i)} = \hat{r}_i \times (p_{(close,i-1)} + 1)$$

The estimated close price of the i-th day can be used to visualize the performance of the model. By analyzing the plots of the estimated price and the true price, we can choose three models (Good Model $i, i = 1,2,3$) with the best performance [Table 1]. We can observe that small deviations in the predicted return can result in significant differences in the estimated price [Figure 10, Figure 11]. This indicates that the performance of the regression prediction can still be improved.

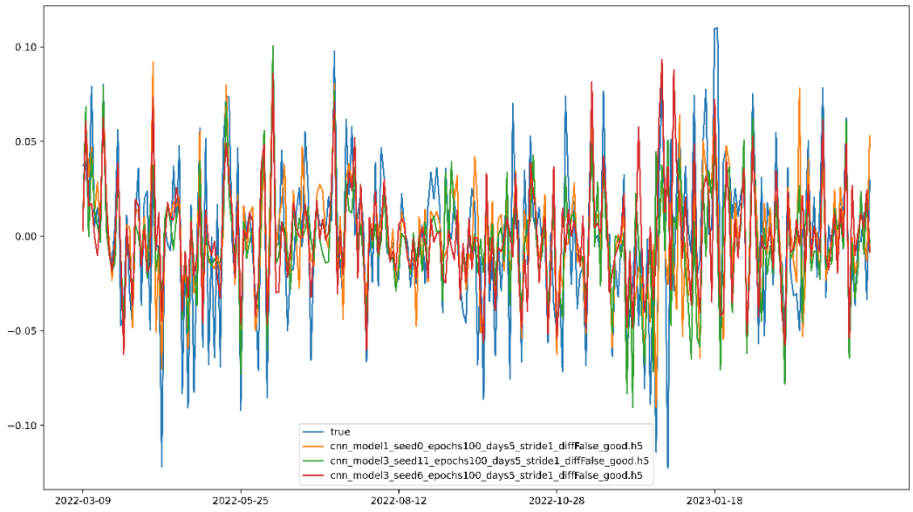


Figure 10: Return Plot of Three Chosen Models

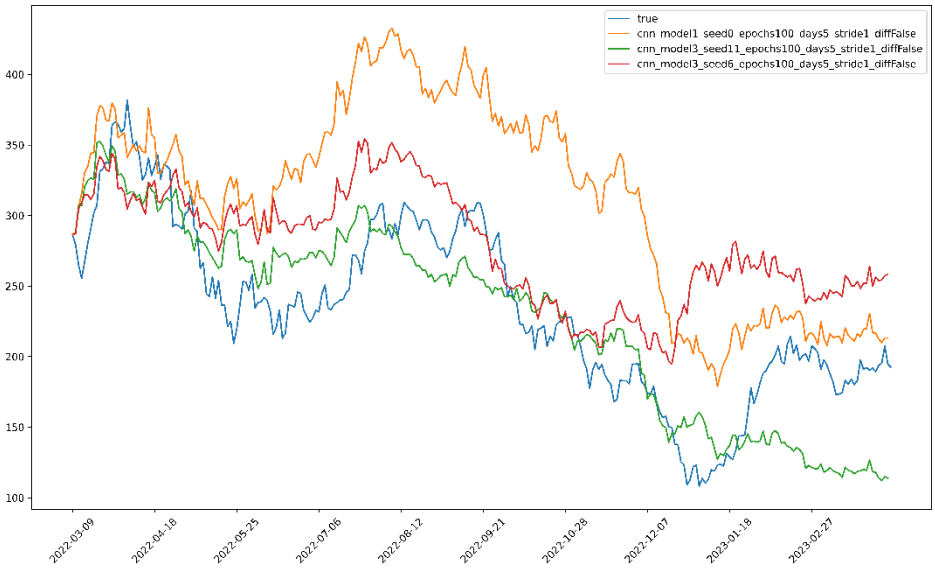


Figure 11: Price Plot of Three Chosen Models

CNN Good Performance Models				
Model Index	CNN Model Numer	Seed	Stride	PCA Type
Good Model 1	1	0	1	PCA on Normalized Data
Good Model 2	3	6	1	PCA on Normalized Data
Good Model 3	3	11	1	PCA on Normalized Data

Table 1:Three Chosen Models under Regression Evaluation with Best Performance

The models are capable of reproducing the overall patterns of return changes and price movements. Nonetheless, they exhibit limitations in accurately capturing the extreme cases of return values, leading to substantial discrepancies between the estimated price and the actual price.

3.3.2 CNN Model Classification Evaluation

A sigmoid function σ(r) can be applied to the predicted return (r), converting it into a probability of a positive return. By setting a threshold probability of 0.5, any probability greater than the threshold can be classified as a positive return. Otherwise, it will be classified as a negative return.

$$\sigma(r) = \frac{1}{1 + e^{-r}}$$

In order to evaluate the effectiveness of the classification, we compared the predicted classification results with the actual classification labels in the test set. To measure the performance, we computed various evaluation metrics including accurac, precision, recall, f1 score, and confusion matrix [Table 2, Table 3]. After analyzing the results, we found that the three models selected based on their performance in regression evaluation also showed relatively good performance in classification.

Model Index	Accuracy	Precision	Recall	F1 Score
Good Model 1	0.72963	0.74265	0.72662	0.73455
Good Model 2	0.73333	0.75969	0.70504	0.73134
Good Model 3	0.71481	0.75833	0.65468	0.70270

Table 2: Accuracy, Precision, Recall, & F1 Score of Three Chosen Models

TRUE Confusion Matrix	Predicted						
	Good Model 1		Good Model 2		Good Model 3		
	+	-	+	-	+	-	Total
+	96	35	100	31	102	29	131
-	38	101	41	98	48	91	139
Total	134	136	141	129	150	120	270

Table 3: Confusion Matrix for Three Chosen Models

The accuracy, precision, recall, and F1 score show a relatively high performance of around 70% for the classification model [Table 2]. The ROC curve also exhibits a good performance [Figure 12]. These results indicate that the selected models for classification in the project can accurately predict whether a stock will have a positive or negative return. Additionally, the models are effective in minimizing false positive rates and false negative rates.

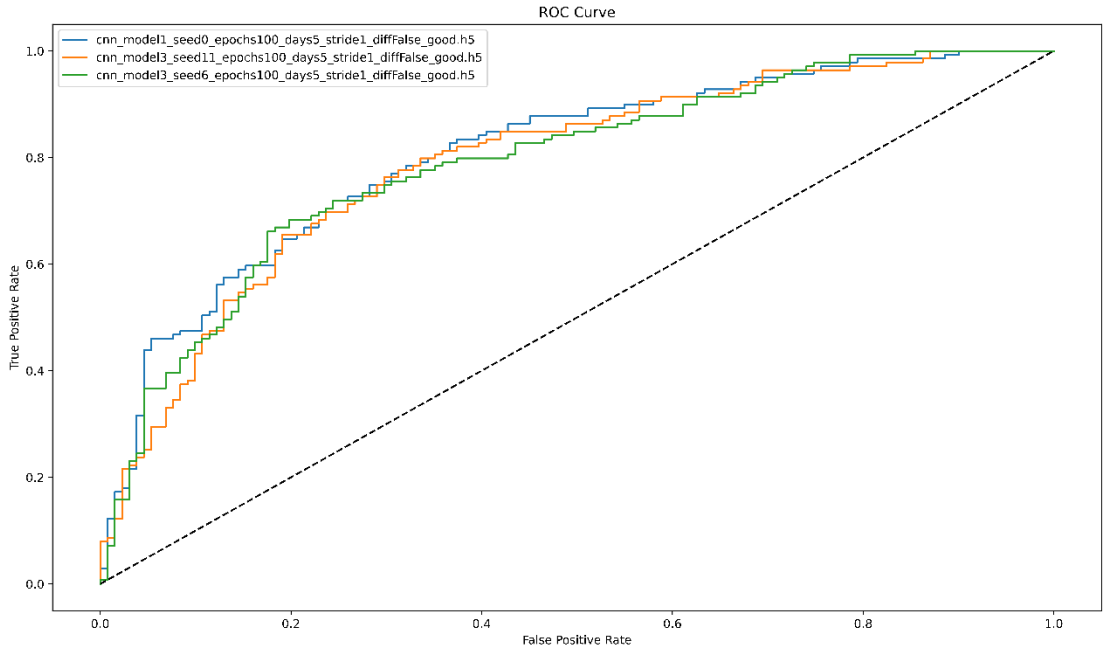


Figure 12: ROC Curve for Three Chosen Models

4. Attention-based LSTM model

4.1 Attention-based LSTM Model Architecture

To capture salient features from the input and improve the model accuracy, we added an Attention layer to the naïve LSTM model. The calculation in the attention layer is:

$$Att_t = \sum_{j=1}^{t} a_j h_j \qquad a_t = \frac{\exp(W_a h_t)}{\sum_{t=1}^{t} \exp(W_a h_t)}$$

After numerous model tuning, we obtained a model with Attention – LSTM(32) – Dense(64) – Dense(32) – Dense(1) structure. We adopted ReLU as the activation function and Adam as the optimizer. After 200 epochs, the model loss dropped to 0.00049 [Figure 13].

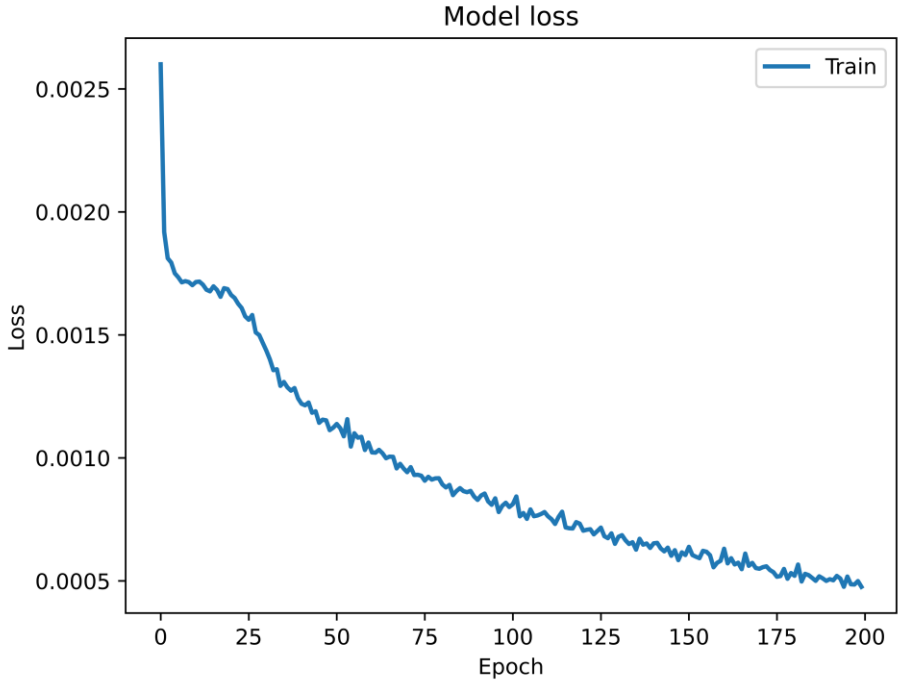


Figure 13: Attention-based LSTM Model Learning Loss Curve

4.2 Attention-based LSTM Model Evaluation

4.2.1 Attention-based LSTM Model Regression Evaluation

The Attention-LSTM model performed quite well in predicting the first 130 days' return, while the accuracy diminished for the rest of the predictions [Figure 14]. Through switching different seeds and hyperparameters to test the robustness of the model, we found that overall, the predicted price curve largely fits with the actual price curve [Figure 15].

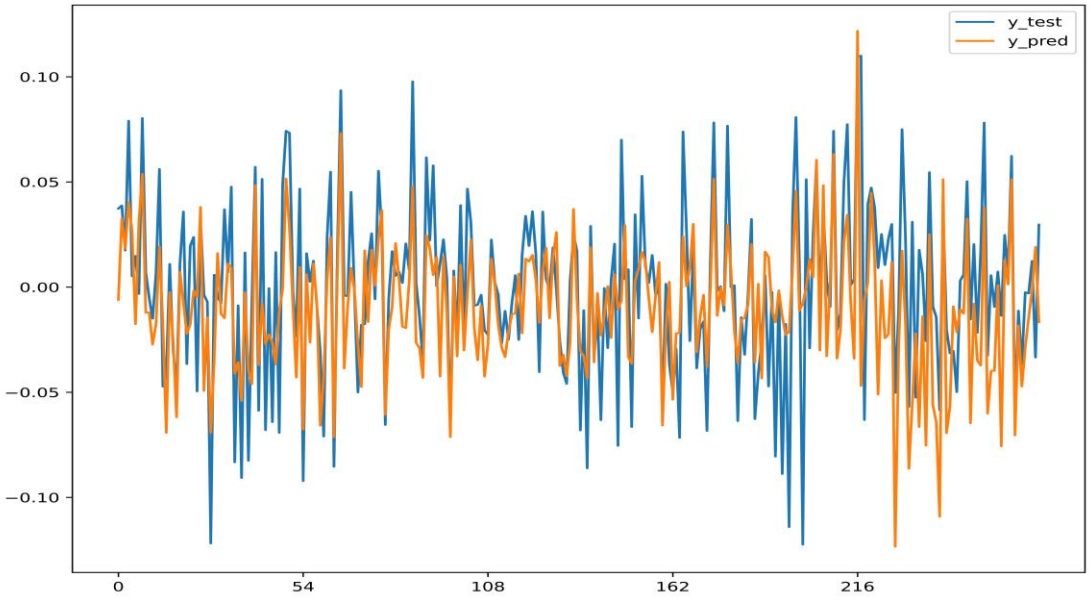
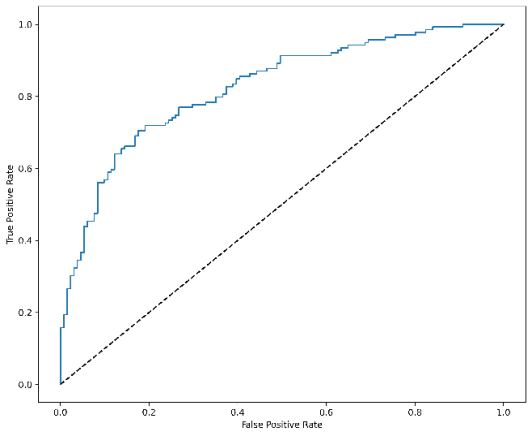


Figure 14: Return Prediction for 270 days

Figure 15: Price Prediction for 270 days

4.2.2 Attention-based LSTM Model Classification Evaluation

Through adding a sigmoid function at the last layer, we obtained the predicted probability of positive return. The classification result is also satisfactory, with a 73.3% accuracy and a 0.74 F1 score.



Training Set			
TARGET / OUTPUT	Class0	Class1	SUM
Class0	115 42.59%	16 5.93%	131 87.79% 12.21%
Class1	56 20.74%	83 30.74%	139 59.71% 40.29%
SUM	171 67.25% 32.75%	99 83.84% 16.16%	198 / 270 73.33% 26.67%

Figure 16: : ROC Curve (Left)and Confusion Matrix for Attention-LSTM model (Right)

5. GARCH Model

5.1 GARCH Model Architecture

To better assess our models trained through CNN and attention-based LSTM, we use the time series model as our comparison. Since ARIMA models assume an unconditionally non-random and constant variance, it is more appropriate to use GARCH(1, 1) model in accordance with the heteroscedasticity of stock return in the market.

$$r_t = \mu + \epsilon_t, \qquad \epsilon_t = \sigma_t e_t,$$

$$\sigma_t{}^2 = \omega + \alpha\epsilon_{t-1}{}^2 + \beta\sigma_{t-1}{}^2, \qquad e_t \sim N(0,1)$$

5.2 GARCH Model Regression Evaluation

We fit the actual return from Nov 29th, 2017 to Mar 8th, 2022 into the GARCH model and utilize the trained model to do a five-step ahead forecast for the return of the periods from Mar 17th, 2022 to Apr 13th, 2023. We get the following prediction result.

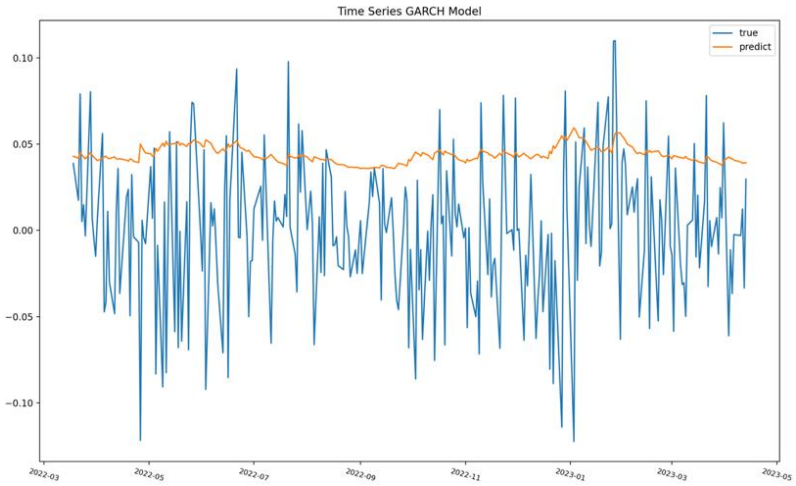


Figure 17: Return prediction for 270 days of GARCH model

6. Backtesting

To further analyze the performance of our models, we design an intraday strategy and evaluate our models based on the portfolio value.

6.1 Intraday Strategy

Based on our predicted return, we will long our target stock at the open price and close it at the close price on the same day. For the regression method, the trading signal is predicted return>3%; for the classification method, the trading signal is classification class=1. The trading will follow the setting below.

Trading Strategy		
Principal	Exposure for Every Trade	Commission Fee Rate
$100,000.00	60% cash	0.1%

Table 4: Intraday Trading Strategy Setting

6.2 CNN Model Results

Regression Method:

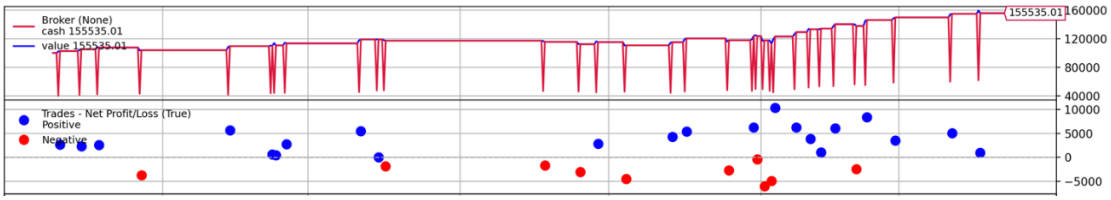


Figure 18: Backtest result for CNN regression method (Good Model 2)

Classification Method:

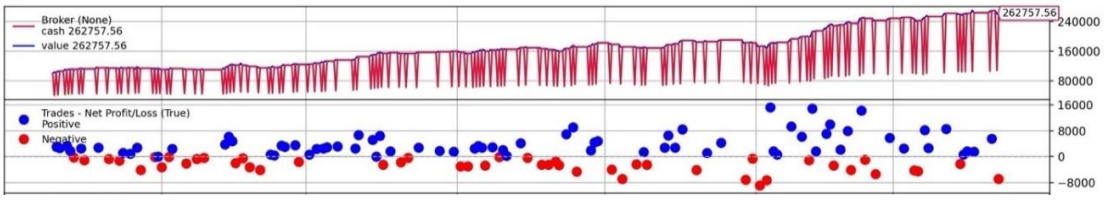


Figure 19: Backtest result for CNN classification method (Good Model 3)

6.3 Attention-based LSTM Model Results

Regression Method:

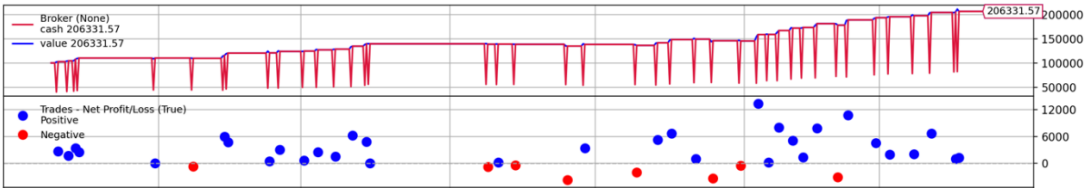


Figure 20: Backtest result for Attention-LSTM regression method

Classification Method:

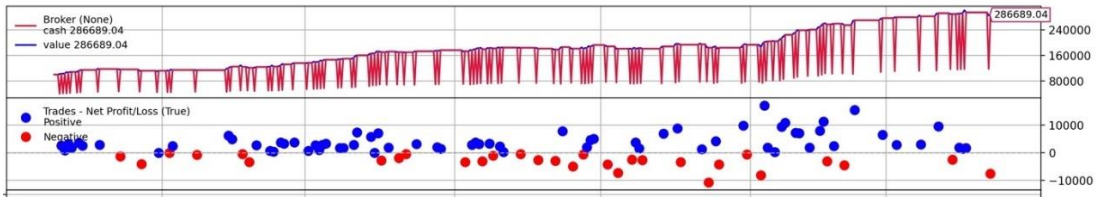


Figure 21: Backtest result for Attention-LSTM classification method

6.4 GARCH Model Results

Regression Method:

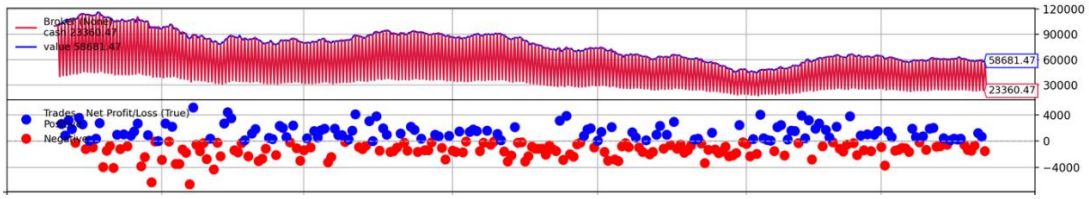


Figure 22: Backtest result for GARCH regression method

7. Conclusion

Based on the backtesting result, we can easily conclude that attention-based LSTM models perform the best, followed by CNN models. Returns predicted by GARCH models hardly change, resulting in constant trading and the worst performance.

Furthermore, classification methods bring much more profits than regression methods. Attention-based LSTM models and CNN models have considerably high accuracy in predicting the sign of return in the near future.

References

Coleman, M. (2019). SPIVA: 2022 Year-End Active vs. Passive Scorecard. *Index Fund Advisor.* Retrieved from https://www.ifa.com/articles/despite_brief_reprieve_2018_spiva_report_reveals_active_funds_fail_dent_indexing_lead_-_works

Appendix[1]

backtesting_backtrader.py: construct an intraday strategy using backtrader package and visualize backtest results

cnn_fit.py: fit CNN model.

cnn_model.py: define CNN model

cnn_model_visualization.py: plot CNN model structure

cnn_model1_seed0_epochs100_days5_stride1_diffFalse_good.h5: Good Model 1

cnn_model3_seed6_epochs100_days5_stride1_diffFalse_good.h5: Good Model 2

cnn_model3_seed11_epochs100_days5_stride1_diffFalse_good.h5: Good Model 3

data.csv: raw_data

data_cleaning.py: clean raw data retrieved from Bloomberg

feature_construction.py: construct feature after PCA

feature_reshape.py: reshape model input

GARCH_model_construction.py: build a GARCH(1, 1) model with evaluation plot

LSTM evaluation.py: load saved h5 files, then draw prediction curves for regression analysis and create confusion matrix and ROC curve for classification analysis

lstm_good_model.h5: a saved good model

LSTM_model_construction.py: build Attention-LSTM models and save into h5 files

model_evaluate.py: evaluate CNN model classification methods

normalization.py: normalized features

principal_component_analysis.R: perform PCA

STAT4012_Presentation_Slides: presentation slides

train_test_split.py: split training and testing set

---

[1] The raw data, coding files, and testing results are available in the GitHub repository: https://github.com/Gavin-OP/stat4012-group-project