The Chinese University of Hong Kong, Shenzhen

FIN3210 Fintech Theory and Practice

Final Project Report

Author:	*Student Number*:
Zhang Runmin	119020073
Zhou Xitong	120030067
Yu Kangqi	121090735
Gu Tianxia	121090146
Xiang Chuyi	121090630
Di Xinyue	121020039
Chen Bingshang	121090026
Zhan Yufei	121090759
Lou Nan	121090380
Chen Nuo	121020014

DECEMBER 2023

1 Stock and Factor

1.1 Stock Selection

We have chosen ten stocks from the automotive industry: 000338.SZ WEICHAI POWER, 000625.SH CHANGAN AUTOMOBILE, 000951.SZ CNHTC-JNTC, 002594.SZ BYD, 600104.SH SAIC MOTOR, 600418.SH JAC, 600660.SH FYG, 601238.SH GAC GROUP, 601633.SH GREAT WALL MOTOR, 601689.SH TUOPU GROUP. Such selection may potentially yield better factor performance, offer comparability, and eliminate the need for industry neutralization.

1.2 Dataset Introduction

The data used in the research is collected from Wind and CSMAR with the range from the beginning of 2016 to the end of 2022. In details, the return data is constructed manually through the close price data from WIND database; the three momentum factors are constructed manually from the close price data and the related return data; the periodical/quarter factors are constructed through data from the last period.

1.3 Factor Introduction

For factor filtration, we computed the correlation between the return and the factors, and chose factors with $IC > 0.01$, $IR > 0.03$. $IC_A = Corr(f_A, r)$, f_A: Factor A loading at time t, r: Stock return at time $t+1$, $IR_A = Mean(IC_A)/std(IC_A)$ After filtration, we get 27 factors including 6 valuation factors like Dividend Yields, 3 momentum factors like 20-Day Momentum, 6 risk analysis factors like 30-Day Variance and 12 financial analysis factors like Cash Recovery Ratio and Cost of Goods Sold Ratio. (Details about factors can be seen from the table in the appendix.)

2 Baseline Model

2.1 Our Model

The Arbitrage Pricing Theory Model (APT) was adopted to be our baseline model for our analysis, due to its simplicity. As per the APT, returns on a risky asset are assumed to follow a factor-based structure, which can be expressed as: $r = \alpha + \beta_1 f_1 + \beta_2 f_2 + ... + \beta_n f_n + \epsilon$. This simple linear regression model was estimated utilizing the Ordinary Least Squares (OLS) methodology. It will be used as a comparative benchmark to evaluate the performance of more sophisticated models. The model employs a standard least squares objective function: $\mathcal{L}(\theta) = \frac{1}{NT} \sum_i^N \sum_t^T (r_{it} - \hat{r_{it}})^2$. Minimization of $\mathcal{L}(\theta)$ yields the pooled OLS estimator.

2.2 Regression Result

Upon running OLS regression on training data, we applied it to the test data, yielding the regression results visualized in Figure 18 (in the Appendix C). The Mean Square Error (MSE) on the test data is reported at 0.000875, and the model's R^2 is -0.008846. This indicates the APT model's poor predictive power on out-of-sample data as we expected.

3 Regularized Linear Models

3.1 Sample Splitting and Training Methodology

Prior to analysing specific models and regularization approaches, knowledge on how to divide datasets for estimation and testing as well as model tuning shall be briefly discussed.

We initially split data into training (In-Sample) and testing (Out-of-Sample) samples. Within the training sample, K-fold cross-validation was employed to tune the hyperparameters of the model.

Finally, the testing sample, which is isolated from the estimation and tuning process, served for the out-of-sample test to assess the predictive capability of our model.

3.2 Regularized Linear

Regularization techniques are employed overcome the issue of overfitting in linear models. It works by incorporating a penalty term ,denoted by (;), to the original loss function to favour an appropriate specification. $\mathcal{L}(\theta) = \frac{1}{NT}\sum_i^N \sum_t^T (r_{it} - \hat{r_{it}})^2 + \phi(\theta; \lambda)$ The specific form of the penalty varies across different models. In our analysis, we mainly focus on three popular regularized linear models: Ridge regression, Lasso regression, and Elastic Net model. They utilize L1, L2, and a combination of L1 and L2 penalties respectively.

3.3 Regression Results and Performance

Upon the training of three models above, the regression results are displayed below. To assess model performance, we employ several metrics, including R^2, Mean Square Error (MSE), Information Criterion (IC), and Information Ratio (IR), as summarized in Table 2 (in the Appendix D). The ridge exhibits the best performance in terms of IC and IR. The performance of lasso and elastic net are almost the same (except their IC and IR on testing data), particularly in training sample, which is demonstrate by their almost identical estimated models, as depicted in Figure 20 and Figure 21 (in the Appendix C). However, none of the models show a significant enhancement in predictive ability for out-of-sample data when compared to the baseline model.

4 Tree Models

4.1 Traditional Tree Models

There are two main categories of Tree ensemble method, boosting and bagging. Boosting algorithms has overfitting issue, and the bagging method can reduce the risk to be overfitting. So, we combine them together to reduce the risk to be overfitting and improve the performance of our model.

4.2 Our methods

4.2.1 Feature Engineering

Given that nearly half of our factors are tied to quarterly data or periodical data, resulting in approximately 60 trading days with identical factor values, we have implemented a novel feature to discern these instances from one another. Referred to as *continuous counting*, this feature entails recording the number of days following the publication date. Its primary utility lies in capturing the influence of these quarterly data points on daily stock returns, accounting for the passage of time. This innovative addition enables us to better gauge and incorporate the temporal impact of these periodic data fluctuations on our predictive models for stock returns.

Besides, we assign more weights on the new data and less weights on the old data to learn the new trend in the market. The weight of the data point is calculated by the following formula: $w_{item_t} = \frac{rank(item_t)}{\sum_{i=1}^{T} rank(item_i)}$.

4.2.2 Ensemble Learning

We use bagging on two type of boosting models, lightGBM and XGBoost, rather than just use one to predict to reduce the risk to be overfitting. The following Figure 1 show our idea exactly.

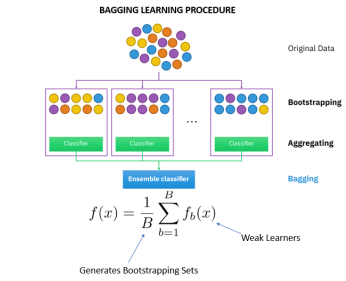

Figure 1: Our model architecture

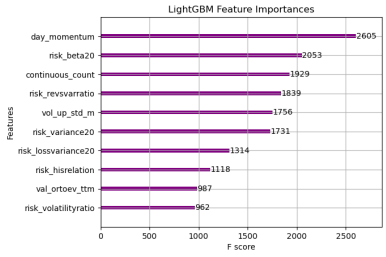

Figure 2: Feature importance of lightGBM

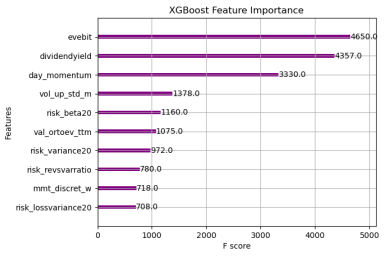

Figure 3: Feature importance of XGBoost

4.3 Feature Importance

After training the data, we get the importance (the split number of each feature) of each features, which is shown in the above figure. As we can see, lightGBM forest capture the features brought by *continuous counting* and all the risk factors, while XGBoost forest capture the features brought by *20day_momentum*, *dividendyield*, and *evebit*, which are more aggressive factors. You will see the difference of these two models in the backtesting part more explicitly.

4.4 Performance

In this section, we will only use mse, IC and IR to evaluate the performance of our models. The following table shows the result of our models. From the Table 3 (in the Appendix D), we can see the insample performance of lightGBM is exaggerated compared to the insample performance of XGBoost, benifiting from the leaf-wise growing strategy. The performance of lightGBM and XGBoost is almost the same based on outsample IC and IR.

5 Deep Learning Models

5.1 Our Models

As we only select ten stocks, which provides us almost 14000 observations for training and validation. Besides this, the information ratio (IR) of the factors in financial market is relatively low than the normal DL tasks. So, we choose to use the models with simple architectures, NN and CNN.

5.2 Feature Engineering

For NN, we add the *continuous counting* and weighted samples just like we did in Tree models. For CNN, we use last 27 days and 27 factors create snapshoots mimic the image in the financial market. The Figure 4 shows the snapshoots of the stock *000338.SZ* using our faw factor data. You can

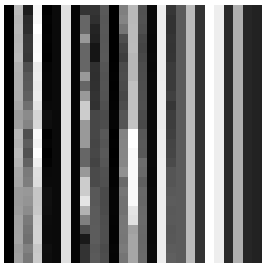

Figure 4: Before processing

Figure 5: After processing

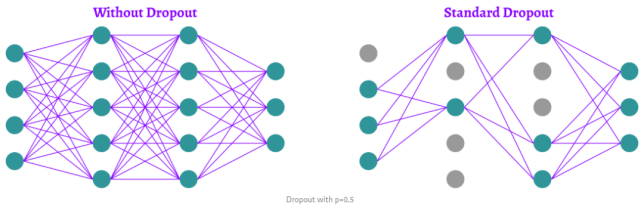

Figure 6: Dropout technique

find that there are some factors have constant value in a period, which is harmful for CNN learning. So, we add the *continuous counting* after the feature with periodical data.

5.3 Model Architectures

For NN, to make the model simple, we use a neural network with only two layers, one layer of 16 nodes and the other layer of only 8 nodes. For CNN, to find more pattern in the market snapshoots,

we use two conv layer in CNN and with different filter sizes. The number of filter is decreasing to reduce the nodes after flatten layer. As there are a lots layer in this model, we use LeakyRuLe to avoid some nodes dying. The details of the architectures can be seen in the Appendix E.

Besides, we also add Dropout layer, just like the Figure 6 illustrated, which help us to train all the nodes in a layer uniformly to reduce the risk of overfitting. An early stop callback is also used in each model to use validation data monitoring the training process to prevent overfitting.

5.4 Performance

The training and validation loss of NN and CNN is shown in the Appendix D. The fitting of NN converges well, but as the trainging data of CNN is a huge pool with many different images and the pattern of these images is hard to recognize, training loss is higher than the validation loss with a small fraction of samples. But, it also alarm us that this model has the propensity to be underfitting due to low quality of data.

6 Out-sample Back Testing

6.1 Portfolio Construction Strategy

Following the assessment of linear regression and machine learning models using evaluation metrics such as MSE or IC, we conduct an out-sample backtesting. To construct a portfolio based on the prediction model, we adopt a strategy outlined in the Appendix F (1). On a daily frequency, stocks with predicted returns below the median are excluded, and weights are assigned based on their predicted return values to form a portfolio.

6.2 Performance of the Linear and Machine Learning Models

Figures 7 and 8 display the results after processing and calculating predicted data, portfolio positions, daily returns, cumulative values, and visualizations. In Figure 7, the LASSO and ElasticNet models exhibit the best performance, followed by the Ridge model, equal weight strategy, with the APT, simple linear regression model performing the worst. In Figure 8, a comparison between the best linear model (LASSO) and the baseline model (APT) is made against machine learning models. Except for the XGBoost model, which outperforms LASSO, other models show varying performances. The lightGBM model, exhibits a significantly lower return, almost equal to the APT model.

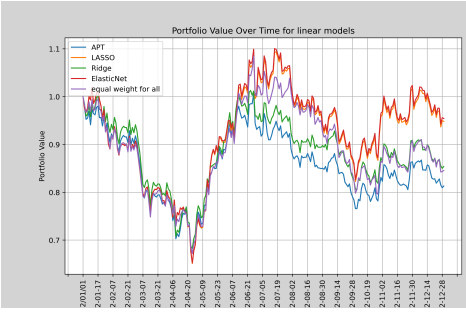

Figure 7: Linear models

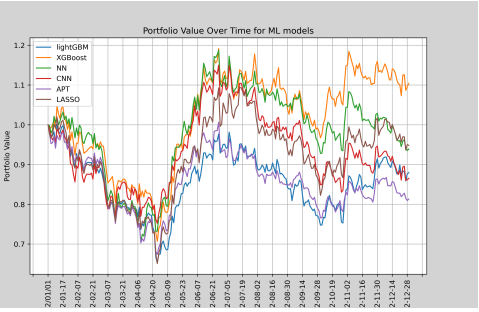

Figure 8: Machine learning models

6.3 Max-drawdown of Models

Profitability is not the sole metric for evaluating a portfolio. Figure 9 visualizes the max- drawdown of each model, with the XGBoost model demonstrating the best performance. Notably, the two deep-learning models, NN and CNN, exhibit more significant drawdowns.

6.4 Sharpe Ratio on Exceed Return

In the following figures, we calculate the Sharpe ratio of the excess return using the APT model as the baseline due to the general negative return on each portfolio during this period. Tree models,

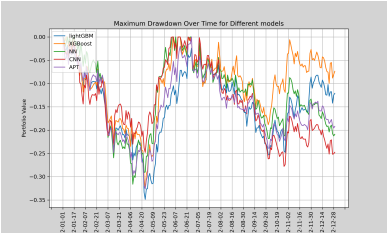

Figure 9: Max drawdowns of models

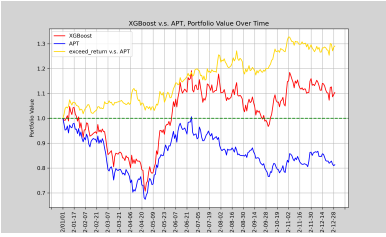

Figure 10: Exceeded Sharpe ratio of XGBoost

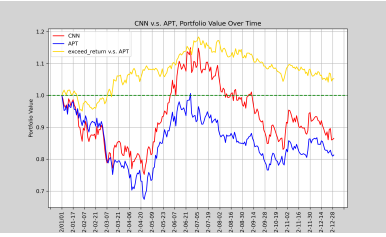

Figure 11: Exceeded Sharpe ratio of CNN

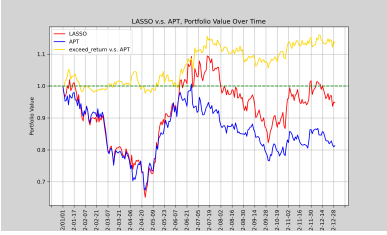

Figure 12: Exceeded Sharpe ratio of LASSO

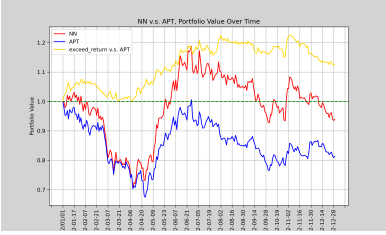

Figure 13: Exceeded Sharpe ratio of NN

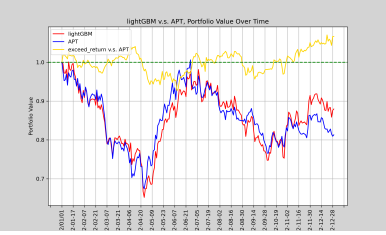

Figure 14: Exceeded Sharpe ratio of lightGBM

XGBoost and lightGBM, outperform the best linear regression model LASSO, shows the advantage of tree models in applications. With overfitting in the training result of the NN model and underfitting for the CNN model, caution is advised in drawing conclusions of deep learning models based solely on Sharpe ratio.

7 Performance under Different Strategies

In assessing the informativeness and accuracy of models under diverse strategies, we assign weights to stocks based on predicted return rankings, as illustrated in the Appendix F (2). This strategy, requiring less information than the "by value" strategy, results in a general decline in the performance of machine learning models, while linear models remain stable.

Combining this observation with the factor contribution figures for each model discussed above, we conclude that machine learning models can effectively incorporate multiple factors, reducing the

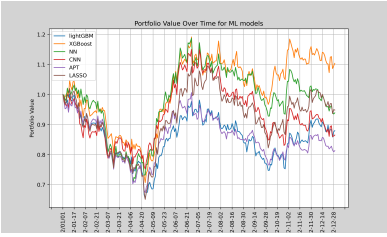

Figure 15: Performance of investing by value

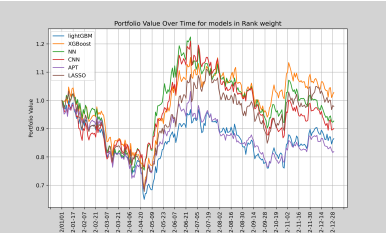

Figure 16: Performance of investing by rank

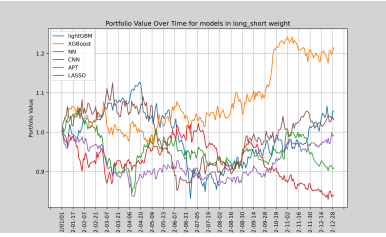

Figure 17: Performance of long-short strategy

likelihood of factors with substantial weights and enhancing the information output. In contrast, linear regression models are more prone to having a few factors with large coefficients, with the majority contributing almost zero, resulting in less informative output. Therefore, when seeking to minimize information requirements and employing the "by rank" strategy, linear models remain unchanged, while machine learning models experience a decline.

Additionally, we explore a strategy involving longing stocks above the median and shorting stocks below the median, with weights determined by predicted return values, as presented in the Appendix F (3). Ideally, by taking long positions on the first half of stocks and short positions on the second half, the portfolio aims to achieve stable returns. Figure 17 reveals the performance ranking: XGBoost > > lightGBM = LASSO > NN > CNN, providing strong confirmation of the conclusions drawn in Section 6 regarding the predictive capabilities of each model.

A Reference

- https://data.csmar.com
- https://www.wind.com.cn/portal/zh/WFT/index.html
- V. DeMiguel, L. Garlappi, and R. Uppal, Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? The review of Financial studies, vol. 22, no. 5, pp. 1915-1953, 2007.

B Factors

Factor	Classification	Frequency	Description
Dividend Yield(%)	Valuation Factor	Daily	Total annual dividend payments/ Stock market capitalization
val_lnmv	Valuation Factor	Daily	Logarithmic market capitalization
val_lntotassets	Valuation Factor	Periodically	Logarithmic total assets
val_ortoev_ttm	Valuation Factor	Daily	Operating Income (TTM)/Enterprise Value
dividendyield2	Valuation Factor	Daily	Dividend Yield (12 Months)
val_floatmv	Valuation Factor	Daily	Floating Market Cap_PIT
20day_momentum	Momentum Factor	Daily	20-Day Momentum = Closing Price of the day / (Average of the previous 20 days' closing prices)
mmt_discret_W	Momentum Factor	Daily	One-week information dispersion momentum: the difference between the number of days with positive returns and the number of days with negative returns in the past five trading days.
Vol_up_std_M	Momentum Factor	Daily	One-month upward volatility: the standard deviation of positive returns over the first twenty trading days
risk_variance20	Risk Analysis	Daily	20-Day Variance_PIT
risk_lossvariance20	Risk Analysis	Daily	20-Day Loss Variance_PIT
risk_beta20	Risk Analysis	Daily	20-Day Beta_PIT

Factor	Classification	Frequency	Description
risk_volatilityratio	Risk Analysis	Daily	Ratio of Individual Security Volatility and Market Volatility_PIT
risk_hisrelation	Risk Analysis	Daily	252-Day Correlation between Individual Security and the Market_PIT
risk_revsvarratio	Risk Analysis	Daily	30-Day Variance / 120-Day Variance_PIT
fa_cashrecovratio_ttm	Financial Analysis	Periodically	Cash Recovery Ratio (TTM)_PIT
fa_blev	Financial Analysis	Periodically	Book Leverage_PIT
fa_current	Financial Analysis	Periodically	Current Ratio_PIT
fa_apturn_ttm	Financial Analysis	Periodically	Accounts Payable Turnover (TTM)_PIT
fa_ncgr_ttm	Financial Analysis	Periodically	Growth Rate - Net Cash Flow (TTM)_PIT
fa_salestocost_ttm	Financial Analysis	Periodically	Cost of goods sold ratio (TTM)
fa_sellexpensetogr_ttm	Financial Analysis	Periodically	Selling expenses/total operating income (TTM)
enebit	Financial Analysis	Quarterly	Enterprise Value multiplier: Total market value/EBITDA

Factor	Classification	Frequency	Description
fa_operincometopbt	Financial Analysis	Periodically	Net Income from Operating Activities / Total Profit_PIT
fa_octogr_ttm	Financial Analysis	Periodically	Total Operating Cost / Total Operating Revenue (TTM)_PIT
fa_netprofitmargin_ttm	Financial Analysis	Periodically	Net Profit Margin (TTM)_PIT
fa_salescashtoor	Financial Analysis	Periodically	Cash Received from Sales of Goods and Rendering of Services / Operating Revenue_PIT

C Linear model coefficients

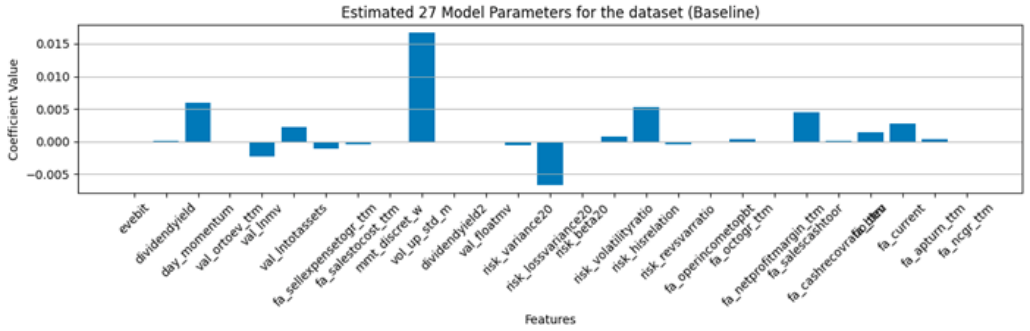

Figure 18: Regression result of APT model

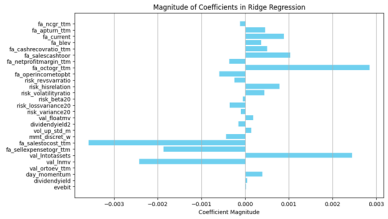

Figure 19: Ridge regression coefficients

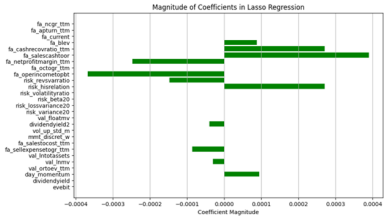

Figure 20: Lasso regression coefficients

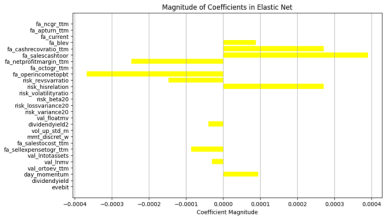

Figure 21: Elastic Net regression coefficients

D Model Performance

Model and dataset	Metrics			
	mse	R^2	IC	IR
Ridge Insample	0.000674	0.00481	0.0459	0.135
Ridge Outsample	0.000789	-0.0112	0.0273	0.0734
Lasso Insample	0.000676	0.00214	0.0339	0.0964
Lasso Outsample	0.000782	-0.00300	0.00192	0.00583
Elastic Net Insample	0.000676	0.00214	0.0338	0.0962
Elastic Net Outsample	0.000782	-0.00305	0.00411	0.0124

Table 2: Linear models performance

Model and dataset	Metrics			
	mse	R^2	IC	IR
LightGBM Insample	0.00010	0.90170	0.82614	4.80196
LightGBM Outsample	0.00091	0.00010	0.04098	0.10894
XGBoost Insample	0.00046	0.68502	0.59617	1.73243
XGBoost Outsample	0.00079	0.00207	0.03940	0.10844

Table 3: Tree models performance

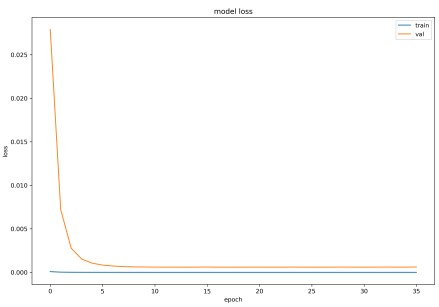

Figure 22: NN

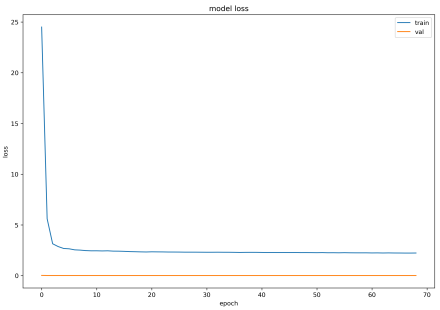

Figure 23: CNN

E Deep Learning Models Architectures

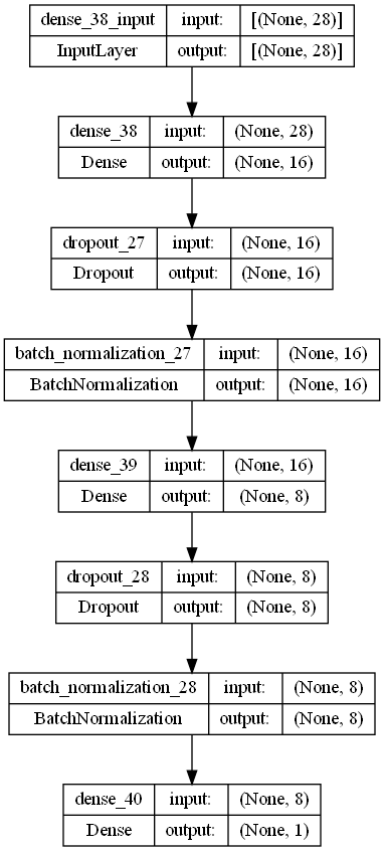

Figure 24: NN

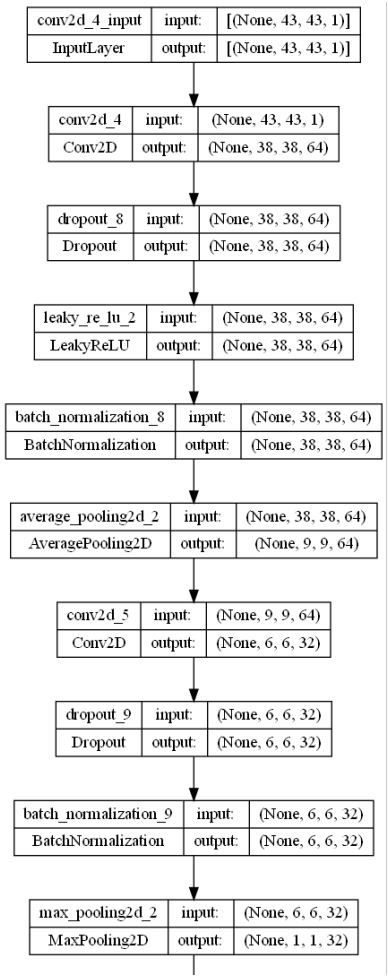

Figure 25: CNN Conv part

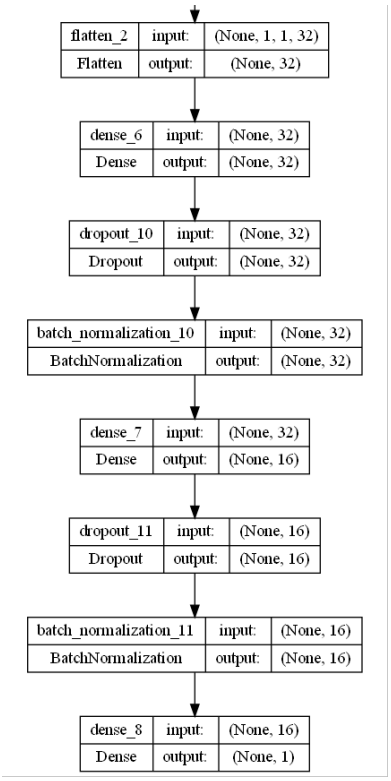

Figure 26: CNN FC part

F Portfolio Construction Strategy

$$
\begin{aligned}
r_t &= [r_1, r_2, \ldots, r_n] \quad \text{for } i = 1, 2, \ldots, n \\
\text{med} &= \text{median}(r_t) \\
\text{map\_to\_zero\_below\_median}(r) &= \max(0, r - \text{med}) \\
\text{sum\_mapped} &= \sum_{i=1}^{n} \text{map\_to\_zero\_below\_median}(r_i) \\
\text{weight}_i &= \frac{\text{map\_to\_zero\_below\_median}(r_i)}{\text{sum\_mapped}} \quad \text{for } i = 1, 2, \ldots, n
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
\text{sorted\_list} &= \text{sorted}(\text{input\_list}, \text{reverse=True}) \\
\text{mapped\_values} &= [0.4, 0.3, 0.2, 0.1] + [0] \times (\text{len}(\text{input\_list}) - 4) \\
\text{mapped\_dict} &= \text{dict}(\text{zip}(\text{sorted\_list}, \text{mapped\_values})) \\
\text{result} &= [\text{mapped\_dict}[num] \text{ for num in input\_list}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
\text{median\_val} &= \text{median}(r) \\
\text{map\_median}(r_i) &= \begin{cases} r_i - \text{median\_val}, & \text{if } r_i \geq \text{median\_val} \\ r_i - \text{median\_val}, & \text{otherwise} \end{cases} \\
\text{mapped\_lst} &= [\text{map\_median}(r_1), \text{map\_median}(r_2), \ldots, \text{map\_median}(r_n)] \\
\text{map\_abs} &= [|\text{map\_median}(r_1)|, |\text{map\_median}(r_2)|, \ldots, |\text{map\_median}(r_n)|] \\
\text{sum\_mapped} &= \sum_{i=1}^{n} \text{map\_abs}(r_i) \\
w_i &= \frac{\text{map\_median}(r_i)}{\text{sum\_mapped}} \quad \text{for } i = 1, 2, \ldots, n
\end{aligned} \tag{3}
$$