# Research on the Data Quantity's influence on Model performance

Joey Xia(zx2325), Weiran Wang(ww2584), Wenqing Zhong(wz2557)

## Abstract

As powerful algorithms are popping out, data is also an important factor that can influence model performances in different tasks. When predicting the stock price, many machine learning models choose to only train on the base price data, whereas human traders often use technical indicators such as RSI to predict the market trend. In this project, we study whether adding additional features including 'Volume','RSI','ROC', and 'OBV' in the training of machine learning models improves the model performance. Our experiments compare the testing mean square errors of models fitted with only basic price data and models fitted with combinations of additional features. Our models include Linear Regression, Decision Tree and Deep Neural Network.

## 1 Introduction

Stock market plays an important role in the global economy. It not only provides a platform for companies to raise capital but also provides ways for individual investors to grow their own wealth. In order to yield more profit, investors are interested in predicting future stock prices. Since stock price depends on economical, political, and many other unpredictable reasons, its unstable nature makes predicting future stock price a universal challenging topic. Machine learning (ML), with its specialty of fitting data's underlying distribution, becomes a handy tool for stock prediction. However, many prior works only fit the price data to their models that they use past 10 days' price to predict future price. Usually, human traders will refer to different technical indicators to understand the situation and momentum of the stock market. We wonder whether feeding these indicators to the ML models can help them "understand" the market better, and thus having better prediction results.

In this project, we work on the question of how adding additional features besides price influence the performance of ML model on stock price prediction. Our hypothesis is that adding additional features to the training dataset will improve the accuracy of models. We download the historical data of top 10 technology companies and compute three selected technical indicators. The top 10 technology companies are chosen from the Forbes' world's biggest tech companies in 2022, including: Apple Inc., Alphabet Inc., Microsoft Corporation, Meta Platforms, Intel Corporation, Cisco Systems Inc., IBM, Oracle Corporation, Broadcom Limited, and Dell Technologies Inc.

We investigate features including volume, price, ROC, RSI, and OBV since these features are not highly correlated to each other and popular in stock analysis. We fit the dataset with Linear Regression model, Decision Tree model, and Deep Neural Network model. Then we evaluate the result using mean squared error and generate the rank of all feature combinations. The result shows the performance of each feature combinations on each model. We hope our study will help researchers choose appropriate data features to fit their models and improve accuracy of future stock price predictions.

The remaining paper is displayed in the following order: Section 2 discusses previous research, Section 3 introduces data we choose, Section 4 discusses methods we use, Section 5 discusses results and evaluation, and Section 6 displays our conclusion and future work.

## 2 Previous Research

In recent years, predicting future stock prices is increasingly popular in the area of finance. Plenty of researchers work on predicting stock prices using classic machine learning models, while other researchers work on predicting stock prices using advanced machine learning models such as Deep Neural Network. Deep Neural Network is effective in predicting future stock prices by its great ability to learn complex patterns in stock data. We want

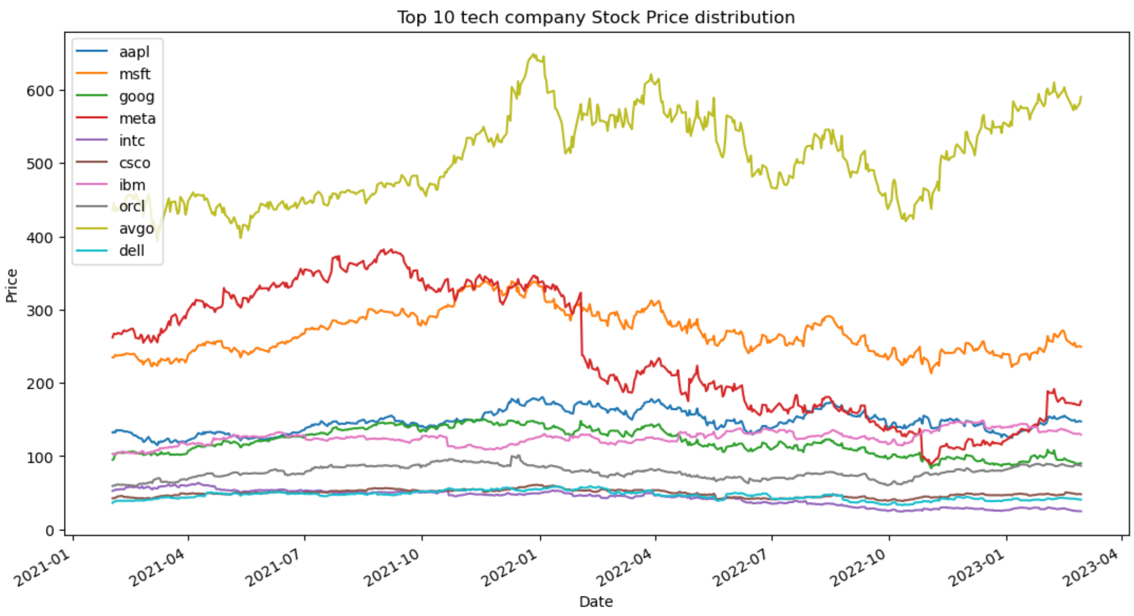


Figure 1: Historical prices of the 10 chosen stocks

to review the previous work of predicting stock prices using machine learning models and identify the gaps.

### 2.1 Machine Learning Model Review

Researchers had started trying machine learning methods in financial tasks very early. (Shah, 2007) worked on using Neural Network on predicting stock price of Chinese firms and found that Neural Network is outperformed than other traditional machine learning models such as linear regression.(Nygren, 2004) worked on using popular machine learning algorithms on stock price prediction in the beginning of 21st century. With the great success of AlexNet (Krizhevsky et al., 2017) in ImageNet (Deng et al., 2009), people's interest and confidence in deep neural network dramatically grew. Research groups adopted popular models that had good performance in other applications to fiannce. (Chen and He, 2018) uses convolutional neural network on stock prediction, while (Rather et al., 2015) uses recurrent neural network. (Yang et al., 2018) tries to exert the potential of deep reinforcement learning on decision making tasks. They uses DDPG algorithm to do stock trading like a human trader.

Since plenty of powerful models have been employed, some researchers (Polyzotis and Zaharia, 2021; Zha et al., 2023) bring forward the idea of data-centric AI. They argue that most of the previous research focuses on models, however data is another crucial part that might be equally important to the model. Their idea of data-centric AI inspired us. Thus we would like to find out that fixing the model and task, how adding additional features could influence the performance result.

### 2.2 Technical Indicators Review

Researchers did studies on importance of technical indicators. (Dr. Bhargavi. R, 2017) showed the validity of RSI technical indicators and resulted that stock companies with RSI greater than 50 has strong momentum. A trader should avoid companies with momentum lower than 50.

## 3 Data

Tech companies play important roles in the overall stock market and are growing fast in recent years. Thus, tech stock prediction becomes an attractive and interesting area for all the investors. Selecting top tech companies is a good choice for stock analysis, since the company's financial situation is relatively stable and sound for investors. Selecting a relatively small group of company's stock can help us to obtain a more detailed understanding of the company's stock price and selected technical indicators.
As shown in Fig. 1, we select the top 10 technology companies in US referring the Forbes [1]. We down-

[1]https://www.forbes.com/sites/jonathanponciano/2022/05/12/the-worlds-largest-technology-companies-in-2022-apple-still-

load the historical data of the 10 companies from Yahoo Finance. The 10 companies are: Apple Inc., Alphabet Inc., Microsoft Corporation, Meta Platforms, Intel Corporation, Cisco Systems Inc., IBM, Oracle Corporation, Broadcom Limited, Dell Technologies Inc. Each row of the dataset represents one day, and columns represent open, high, low, close, adjustment close price, and volume. The data start from 2021/2/1 to 2023/3/1 with a total number of 523 days. We also calculate three technical indicators that are frequently used by traders:

1. **RSI**: Relative strength Index (RSI) is a technical indicator that is used to measure the strength of a security's price action. The RSI compares the magnitude of recent gains to the magnitude of recent losses in order to determine if a security is overbought or oversold.

$$RSI = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}} \quad (1)$$

   where average gain or loss is the average percentage return or loss during a certain period.

2. **ROC**: The Rate of Change (ROC) of price is a technical indicator that calculates the percentage difference between the current price and the price from a specific number of periods ago, and is used to gauge momentum.

$$ROC = \frac{P_0 - P_n}{P_n} \times 100 \quad (2)$$

   where $P_0$ is the closing price of most recent period, and $P_n$ is the closing price of $n$ periods before.

3. **OBV**: On Balance Volume (OBV) is an indicator that evaluates the pressure of buying and selling in the market. It works by adding the volume on days when the price goes up and subtracting the volume on days when the price goes down. If the stock closes higher than the previous day, all of the day's volume is classified as up-volume. Conversely, if the stock closes lower than the previous day, all of the day's volume is classified as down-volume.

$$OBV = OBV_{prev} + \begin{cases} V & P > P_{prev} \\ 0 & P = P_{prev} \\ -V & P < P_{prev} \end{cases} \quad (3)$$

   where $V$ is the current volume, $P$ is the current close price, and $P_{prev}$ is the close price in last period.

dominates-as-brutal-market-selloff-wipes-trillions-in-market-value/?sh=557caee33448

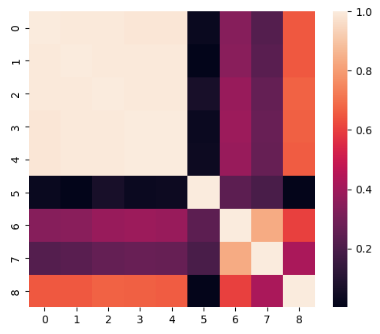


Figure 2: Correlation matrix of the OHLCV and three chosen features

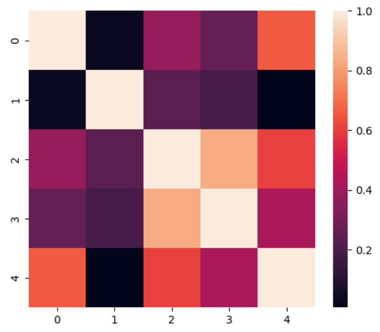


Figure 3: Correlation matrix of the chosen features

To ensure the quality of our data, we remove highly correlated features high, low, open, and close price. After data processing, we have 5 features: price, volume, RSI, ROC, OBV. Overall, the dataset contains 10 companies and 523 rows, 5 columns.

## 4 Method

This study is aim to find the best combination of technical indicators, RSI, ROC, OBV, volume and price. Three machine learning models are implemented in the method.

### 4.1 Data processing and cleaning

After collecting top 10 tech companies stock data from Yahoo Finance, we apply the following steps to clean the data.

- We refer to (Gort et al., 2022) to choose technical indicators that we want to investigate. We choose ROC, RSI, and OBV because they are least correlated to each other. Having a group of highly correlated features will not give additional learning information to the model.

- We drop highly correlated pairs with correlated values > 0.8. As shown in Fig. 2, open, high, low, and close are highly correlated to price, so they are dropped. After dropping, the new correlation matrix is shown in Fig. 3

- We remove rows with missing values in the dataset

## 4.2 Machine learning methods

### 4.2.1 Linear Regression

We start with a vanilla linear regression model. Its loss function is the same as MSE:

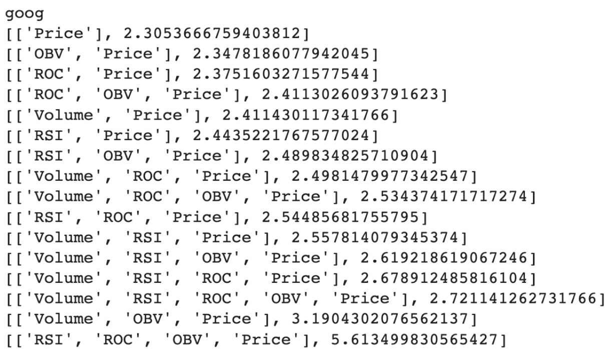
$$Loss = ||Xw - Y||^2$$

Our baseline is only using "Price" to fit the model. The baseline model uses 10 historical stock prices to predict the stock price of the 11th day. For example, using stock prices from 10/01/2021 to 10/11/2021 to predict the stock price of 10/12/2021. In order to study the performance of other feature combinations, we loop through all possible feature combinations and add additional features to the baseline model to build new linear models. There are 16 different feature combinations, such as ['Price'], ['Price','ROC'], ['Price', 'ROC', 'OBV'], ect. We train a new linear regression model on each one of these feature combinations, so we end up having 16 different models for each company's data.

After testing the models and calculating their MSEs, we sort the MSEs in non-descending order to see the rank of feature combinations. For example, Figure 4 shows the ranks for Google's data and Figure 5 shows the ranks for Apple's data. The first element of each row is the features combination used in the model, and the second element is the MSE of this model.

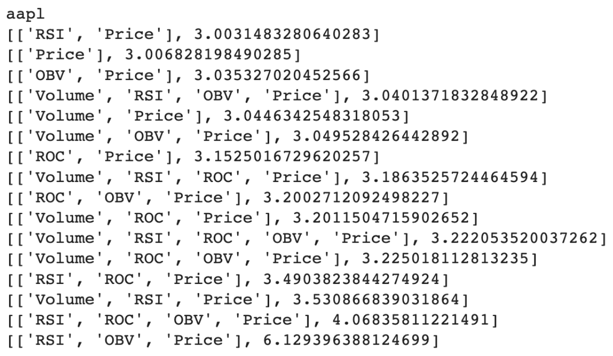
As we can see, when using Google's data, the baseline model that fits on ['Price'] has the smallest MSE, and the model that fits on ['OBV','Price'] has the second smallest MSE. However, when using Apple's data, the models perform differently. In this case, ['RSI','Price'] has the smallest MSE, and ['Price'] has the second smallest MSE. Since each company shows a different rank for feature combinations, we calculate the average rank for each feature combination.

Figure 6 shows the average rank for feature combinations. ['Price'] has an average rank of 2.2, indicating that the baseline model has the best general performance. ['RSI','Price'] has the second-highest average rank, and ['OBV', 'Price'] has the third-highest average rank. We take a closer look at each company's rank and find that ['Price'] has the smallest MSE on Google, Meta, Intel, and Cisco, but it is not significantly better than other fea-



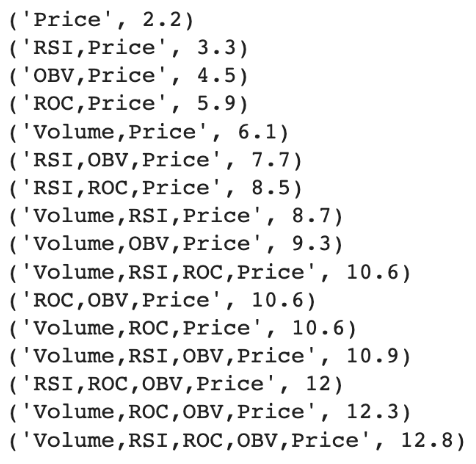
```
goog
[['Price'], 2.3053666759403812]
[['OBV', 'Price'], 2.3478186077942045]
[['ROC', 'Price'], 2.3751603271577544]
[['ROC', 'OBV', 'Price'], 2.4113026093791623]
[['Volume', 'Price'], 2.411430117341766]
[['RSI', 'Price'], 2.4435221767577024]
[['RSI', 'OBV', 'Price'], 2.489834825710904]
[['Volume', 'ROC', 'Price'], 2.4981479977342547]
[['Volume', 'ROC', 'OBV', 'Price'], 2.534374171717274]
[['RSI', 'ROC', 'Price'], 2.54485681755795]
[['Volume', 'RSI', 'Price'], 2.557814079345374]
[['Volume', 'RSI', 'OBV', 'Price'], 2.619218619067246]
[['Volume', 'RSI', 'ROC', 'Price'], 2.678912485816104]
[['Volume', 'RSI', 'ROC', 'OBV', 'Price'], 2.721141262731766]
[['Volume', 'OBV', 'Price'], 3.1904302076562137]
[['RSI', 'ROC', 'OBV', 'Price'], 5.613499830565427]
```

Figure 4: MSEs for Linear Regression Model Trained on Google's Data



```
aapl
[['RSI', 'Price'], 3.0031483280640283]
[['Price'], 3.006828198490285]
[['OBV', 'Price'], 3.035327020452566]
[['Volume', 'RSI', 'OBV', 'Price'], 3.0401371832848922]
[['Volume', 'Price'], 3.0446342548318053]
[['Volume', 'OBV', 'Price'], 3.049528426442892]
[['ROC', 'Price'], 3.1525016729620257]
[['Volume', 'RSI', 'ROC', 'Price'], 3.1863525724464594]
[['ROC', 'OBV', 'Price'], 3.2002712092498227]
[['Volume', 'ROC', 'Price'], 3.2011504715902652]
[['Volume', 'RSI', 'ROC', 'OBV', 'Price'], 3.222053520037262]
[['Volume', 'ROC', 'OBV', 'Price'], 3.225018112813235]
[['RSI', 'ROC', 'Price'], 3.4903823844274924]
[['Volume', 'RSI', 'Price'], 3.530866839031864]
[['RSI', 'ROC', 'OBV', 'Price'], 4.06835811221491]
[['RSI', 'OBV', 'Price'], 6.129396388124699]
```

Figure 5: MSEs for Linear Regression Model Trained on Apple's Data



```
('Price', 2.2)
('RSI,Price', 3.3)
('OBV,Price', 4.5)
('ROC,Price', 5.9)
('Volume,Price', 6.1)
('RSI,OBV,Price', 7.7)
('RSI,ROC,Price', 8.5)
('Volume,RSI,Price', 8.7)
('Volume,OBV,Price', 9.3)
('Volume,RSI,ROC,Price', 10.6)
('ROC,OBV,Price', 10.6)
('Volume,ROC,Price', 10.6)
('Volume,RSI,OBV,Price', 10.9)
('RSI,ROC,OBV,Price', 12)
('Volume,ROC,OBV,Price', 12.3)
('Volume,RSI,ROC,OBV,Price', 12.8)
```

Figure 6: Average Rank for Feature Combinations on Linear Regression Model

ture combinations with small MSEs. For example, when using Google's dataset, ['Price'] has a 2.30 MSE, and ['OBV','Price] has a 2.34 MSE. When using Meta's dataset, ['Price'] has a 6.63 MSE, and ['RSI','Price] has a 6.67 MSE. We also check companies where ['Price'] doesn't show the smallest MSE. On Microsoft's dataset, ['RSI', 'Price'] has a 5.31 MSE, and ['Price] has a 5.32 MSE. On Apple's dataset, ['RSI','Price'] has a 3.003 MSE, and ['Price] has a 3.006 MSE. In general, ['Price'] and ['ROC'/'RSI'/'OBV'/'Volume', 'Price'] show very similar MSEs on most company's datasets. In other words, if we only add one additional feature to the baseline model, no matter what that feature is, the model performance usually won't change much. It sometimes goes slightly better, sometimes goes slightly worse, which seems totally random. But if we add more than two additional features to the baseline model, the model will behave worse on every company's dataset, except IBM. A potential reason for this is that no additional features can provide more information to help predict future price. Additional features are considered as noise in Linear Regression model, so the more features we add, the larger is the MSE.

These above observations seem to suggest that adding additional features to the baseline model won't improve model accuracy. In order to further inspect this hypothesis, we check cases where ['Price'] doesn't produce the smallest MSE. Figure 7 is the feature importance graph for a Linear Regression model trained on Apple's dataset. This model has the smallest MSE when using ['RSI','Price']. We can see that Price10, which means the price at the 10th day, has the largest feature importance. Although all RSIs have some level of importance, they are far less important than Prices. The similar pattern is found in Oracle's dataset, too (Figure 8). ['Price','ROC'] has the smallest MSE on Oracle's dataset, but all ROCs are far less important than Prices. This finding suggests that additional features don't contribute much to Linear Regression model accuracy. Adding additional features to the baseline model cannot notably improve model performance.

Given that we have found ['Price'] is the best feature to use on Linear Regression model, we want to further improve the model accuracy by adding L2-regularization. Linear regression model with L2-regularization is also called Ridge Regression. It adds penalty to large coefficients and could potentially improve accuracy by reducing the effect of multicollinearity. The loss function of Ridge Regression is

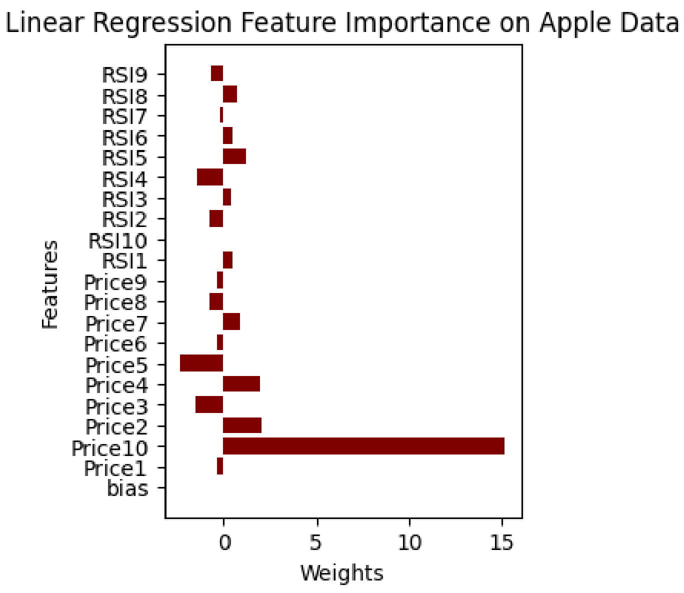


Figure 7: Linear Regression Feature Importance on Apple Data

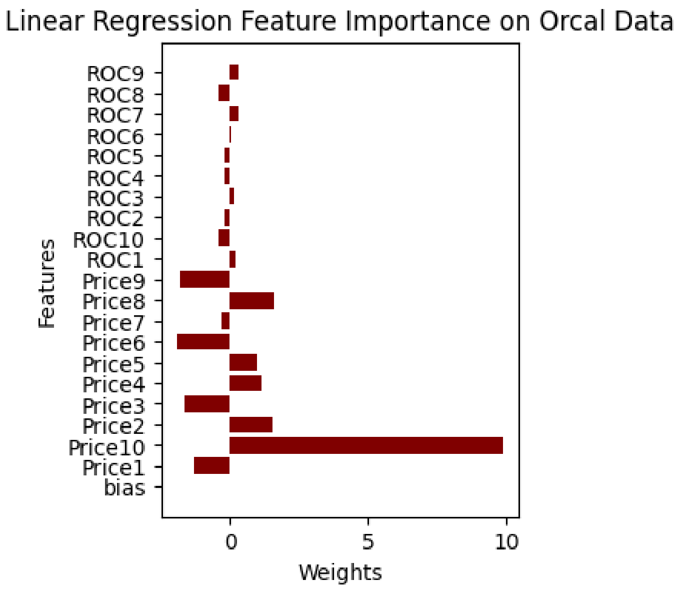


Figure 8: Linear Regression Feature Importance on Oracle Data

$$Loss = ||Xw - Y||^2 + \alpha \sum_{1}^{n} w_i^2$$

$\alpha$ is the penalty term. The higher the values of alpha, the bigger is the penalty. We use $r^2$ score to evaluate the effect of $\alpha$. $r^2$ score shows how well the data fit the regression model, so a higher $r^2$ score indicates a better model.

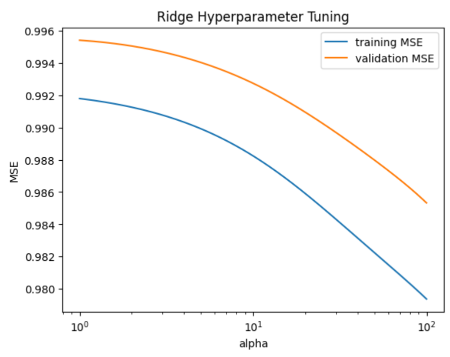


Figure 9: R-squared Score for Ridge Regression

However, our results in Figure 9 show that adding L2-regularization cannot improve model accuracy at all. As we increase the value of $\alpha$, $r^2$ score of the model decreases. The potential reason for this is that Prices on different days don't have strong multicollinearity, so the penalty is mainly added to Price10, which is the most important feature and has the largest coefficient. The importance of Price10 is reduced by the penalty term, so it contributes less to the model accuracy. This result again proves that ['Price'] is the best feature combination to use for linear regression model because it doesn't show strong multicollinearity.

### 4.2.2 Decision Tree

The second method we use is Decision Tree. We pick Decision Tree because its mechanism is vastly different from Linear Regression. The performance of Decision Tree does not depend on linear relationship between independent and dependent variables. We want to test which feature combination works best on Decision Tree.

We take a different approach to test the performance of feature combinations in Decision Tree. Instead of looping through all possible combinations and build a model for each combination, we first fit the Decision Tree with all features, i.e. ['Price','Volume','ROC','RSI','OBV']. We use the default Decision Tree model from sklearn. After fitting, it gives us a Decision Tree with the maximum depth, i.e. every leaf in the tree is a pure node. However, this Decision Tree is usually over-fitted. It uses features that do not contribute to the general model accuracy. In order to find the tree with the best performance, we prune the tree layer by layer, re-train the tree, and record its MSEs. Figure 10 shows the pruning process of the Decision Tree using Microsoft's data.

The maximum depth of this tree is 15, but the

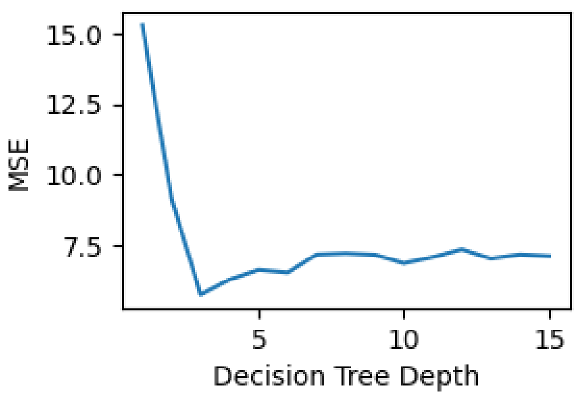


Figure 10: Decision Tree Pruning using Microsoft's data

tree with depth of 2 has the smallest MSE, meaning the best feature combination for this decision tree is used at depth of 2. We visualized the 2-layer tree to see which features are used in tree nodes. From Figure 11 we can see that the only feature used in all tree nodes is Price10. This indicates that ['Price'] is the best feature combination for Decision Tree trained on Microsoft's data.

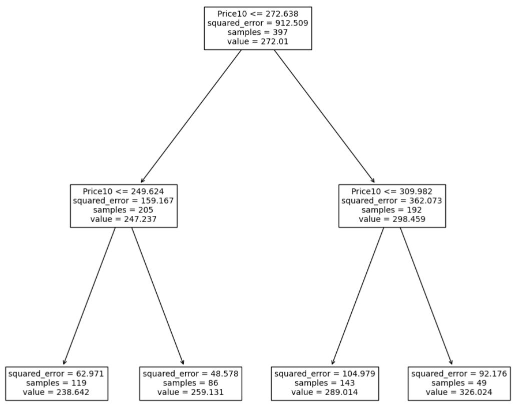
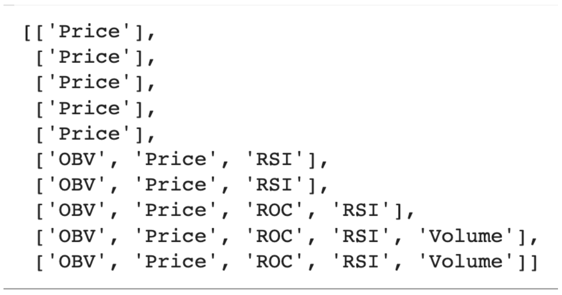


Figure 11: Decision Tree fitted with Microsoft data

If we train and prune the Decision Tree using Meta's data, the tree will have 4 layers and use ['RSI','OBV','Price']. If we use Intel's data, the tree will have 12 layers and use all fetaures. Decision Trees trained on different datasets have different structures and use different features in tree nodes. We loop through every tree and record the feature combination it uses. The result is shown in Figure 12.



```
[['Price'],
 ['Price'],
 ['Price'],
 ['Price'],
 ['Price'],
 ['OBV', 'Price', 'RSI'],
 ['OBV', 'Price', 'RSI'],
 ['OBV', 'Price', 'ROC', 'RSI'],
 ['OBV', 'Price', 'ROC', 'RSI', 'Volume'],
 ['OBV', 'Price', 'ROC', 'RSI', 'Volume']]
```

Figure 12: Feature Combinations Frequency on Decision Tree Model

The result indicates that ['Price'] is used in most tress, meaning that adding more features to the baseline model doesn't improve model performance in general. This result is the same as the result for Linear Regression. There are 5 models that don't behave the best on ['Price']. We check their feature importance graphs to see which feature contributes the most to model accuracy. Figure 13 shows feature importance for Decision Tree trained on Google's data. This tree has the smallest MSE when using all features.

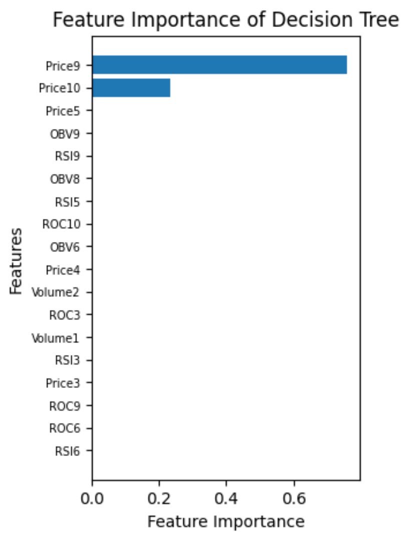


Figure 13: Feature Importance for Decision Tree Trained on Google's Data

We can see that although this tree uses all features, Price9 and Price10 are the most important ones. Other features' importance values are so small that their contributions can safely be ignored. Similar feature importance graphs are produced by other trees that use additional features. This result indicates that ['Price'] is the best feature to use on Decision Tree.

### 4.2.3 Deep Neural Network

The third method we use is deep neural network. Deep neural network has shown great power in the last decade. It has the potential to fit all kinds of distributions and functions. Besides, it has strong flexibility of structure, a lot of powerful models are constructed based on neural network: convolutional neuralnetwork (Krizhevsky et al., 2017), residual neural network (He et al., 2016), trnaformers (Vaswani et al., 2017), LSTM (Hochreiter and Schmidhuber, 1997), deep reinforcement learning (Mnih et al., 2015), etc. So we would like to see how different combinations of features influence the performance of neural network.

As the prediction task relatively simple, we choose a simple fully connected network with two hidden layers.

**Layer structure.** Deep neural network is powerful. However, too large structure when on simpler tasks might cause problems such as overfitting. In order to reduce the redundancy of the network, we've tried several different setup of hidden layer sizes. We've tried the two hidden layers with size: [32,32], [32,16], [16,16], [16,8], and [8,8] with each running on 100 episodes. We choose the final structure to be [16,8] that balance the loss and training efficiency. The input layer has size from 1 to 4 based on how many features we are feeding into the model. The output layer has size 1 because we only expecting the neural network output a one-step prediction. Fig. 14 gives a visualization of our neural network structure.

**Activation function.** Activation functions add non-linearity to the neural network. Different choices of activation functions might influence the performance. Here, from candidates including: ReLU, ELU, Softmax, Sigmoid, and Tanh. ReLU has been investigated to be the activation function that is similar to the activation mechanism of human neurons. And it has the best performances in wide range of tasks, including prediction/regression tasks as ours. Thus we choose ReLU as our activation function.

**Loss function.** Neural network support different types of loss functions when evaluating the performance and updating the weights and bias using backpropagation. Several popular loss functions include: mean squared error (MSE), binary cross-entropy, categorical cross-entropy, hinge loss, hubor loss, etc. MSE is the most common choice in prediction tasks. Besides, our regression model

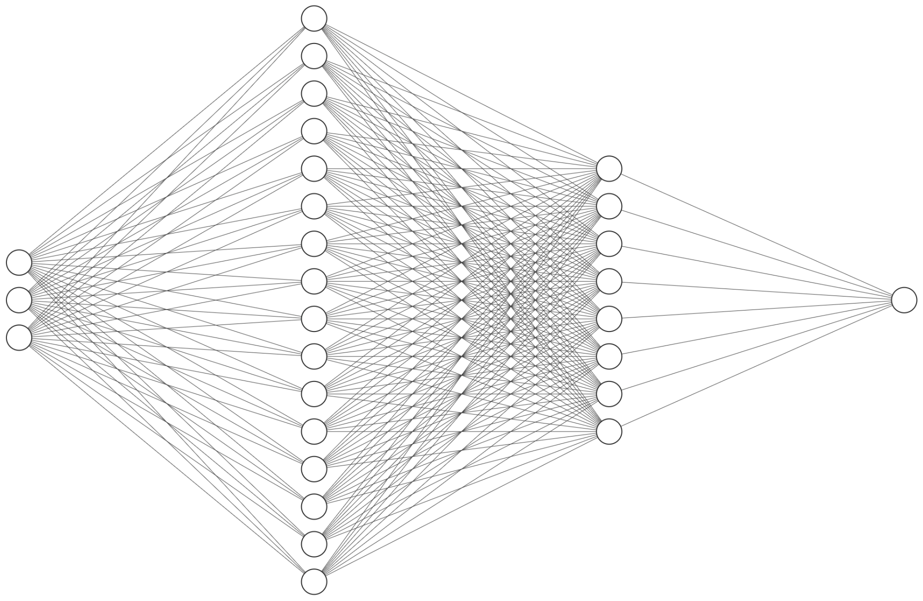


Figure 14: Layer structure of our neural network

also uses MSE. Thus we choose MSE as our loss function.

**Stochasticity.** By the default implementation of TensorFlow, initial weights and biases of a neural network are randomly generated. This gives the neural network more flexibility and avoid it stuck in a local minimum during the training process. Besides, during the training process, we do not specifically split batches for training the neural network. Then the network will use stochastic gradient descent, which samples a mini batch randomly and use their corresponding gradients to update weights and biases. Thus, the prediction results are not exactly same for each trial using neural network. We choose the trial with the minimum final loss to do our final evaluation.

## 5 Results and evaluation

For each company, we calculate the average rank for all 16 combinations of features. The result shows that ['Price'] is the best feature combination to use on Linear Regression, Decision Tree and DNN, which means adding additional features to the training dataset does not improve the model performance. We use average rank instead of average MSE to evaluate feature combinations because MSEs varies a lot between different companies. Using MSEs within each company to calculate average rank can more accurately show the general performance of feature combinations. The ranks for each model are recorded in the table below. Although we have shown that adding technical indicators in the training dataset cannot significantly improve the model accuracy, we are not sure why. Intuitively, 'Volume' should tell the model more information about the stock price because it is the number of shares traded over a specific period of time. Volume reflects how "popular" a stock is, so we expect that adding volume to the dataset will improve model accuracy. Similarly, RSI, ROC and OBV show the general market trend in different ways, so we expect them to improve model accuracy, too. However, the result of our study is different from our expectation. We think the potential reason is that our models are not complex enough to maximize the potential and efficacy of technical indicators. Linear Regression and Decision Tree are very basic models that treated each feature equally without considering their meaning. They cannot "understand" that technical indicators show the macro trend of the market instead of the stock price of a certain day, so they ignore technical indicators or treat them like non-informative noise. Deep Neural Network has the potential to learn from technical indicators, but our model only has two layers, so it also doesn't have the ability to make good use of technical indicators. However, this is just our conjecture based on current knowledge. Knowing whether this conjecture is actually true will need further study.

| Average Rank | | | |
|---|---|---|---|
| Feature Combinations | LR | DT | DNN |
| $['Price']$ | 2.2 | 1 | 2.6 |
| $['ROC','Price']$ | 5.9 | NAN | 2.9 |
| $['RSI','Price']$ | 3.3 | NAN | 3 |
| $['OBV','Price']$ | 4.5 | NAN | 7.7 |
| $['Volume','Price']$ | 6.1 | NAN | 5.6 |
| $['Volume','RSI','Price']$ | 8.7 | NAN | 6.6 |
| $['Volume','ROC','Price']$ | 10.6 | NAN | 10.5 |
| $['Volume','OBV','Price']$ | 9.3 | NAN | 9.2 |
| $['RSI','ROC','Price']$ | 8.5 | NAN | 3.4 |
| $['RSI','OBV','Price']$ | 7.7 | 2 | 9.8 |
| $['ROC','OBV','Price']$ | 10.6 | NAN | 12.9 |
| $['Volume','RSI','ROC','Price']$ | 10.6 | NAN | 9.7 |
| $['Volume','RSI','OBV','Price']$ | 10.9 | NAN | 13.4 |
| $['Volume','ROC','OBV','Price']$ | 12.3 | NAN | 12.6 |
| $['RSI','ROC','OBV','Price']$ | 12 | 3 | 12.8 |
| $['RSI','ROC','OBV','Volumn','Price']$ | 12.8 | 4 | 13.3 |

Note: LR stands for linear regression, DT stands for Decision Tree, DNN stands for Deep Neural Network.

## 6 Conclusion

In this paper, we have shown a comparison of how different combinations of technical indicators could influence the prediction results by using three different models: linear regression, decision tree, and neural network. And we've discussed about the techniques that we use when we process the data and choose and design the models we use. Our result shows that baseline models fitted on ['Price'] has the best performance on all three models. Thus, we would claim that adding additional features to the training dataset cannot improve model accuracy.

## 7 Future Work

### 7.1 Data

We restrict our data in the field of stock price of technology companies. However, the financial markets is extremely complex. There are different types of securities and each with different fields such as technology, energy, finance, sports, etc. Besides that, we only choose 10 top technology companies. Although these 10 companies could be a good representation of technology industry, there bias exists. In our future, we would like to find out how to include more types of industries data, in order to have a more generalizable result.

### 7.2 Features

On the other hand, our chosen technical indicators are also limited. RSI, ROC, and OBV are three typical indicators that are less correlated with each other, however, there are decades of different indicators based on price and volume. Although some of them might be highly correlated, they might bring different effect to the prediction. Thus we would also try more different types of technical indicators to find the one with best potential.

### 7.3 Methods

We've worked on choosing different hyperparameters of the three models: linear regression, decision tree, and deep neural network. However, there are a lot more possible variations we could work with. As we mentioned in the evaluation section, maybe more complex models will learn better from technical indicators.

## References

Sheng Chen and Hongxiang He. 2018. Stock prediction using convolutional neural network. In *IOP Conference series: materials science and engineering*, volume 435, page 012026. IOP Publishing.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Anith.R Dr. Bhargavi. R, Dr. Srinivas Gumparthi. 2017. Relative strength index for developing effective trading strategies in constructing optimal portfolio. *Research India Publications*.

Berend Jelmer Dirk Gort, Xiao-Yang Liu, Xinghang Sun, Jiechao Gao, Shuaiyu Chen, and Christina Dan Wang. 2022. Deep reinforcement learning for cryptocurrency trading: Practical approach to address backtest overfitting. *arXiv preprint arXiv:2209.05559*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518:529–33.

Karl Nygren. 2004. Stock prediction–a neural network approach.

Neoklis Polyzotis and Matei Zaharia. 2021. What can data-centric ai learn from data and ml engineering? *arXiv preprint arXiv:2112.06439*.

Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. 2015. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241.

Vatsal H Shah. 2007. Machine learning techniques for stock prediction. *Foundations of Machine Learning| Spring*, 1(1):6–12.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu. 2018. A practical machine learning approach for dynamic stock recommendation. In *2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, pages 1693–1697. IEEE.

Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. 2023. Data-centric ai: Perspectives and challenges. *arXiv preprint arXiv:2301.04819*.

## 8 Appendix

### 8.1 Code and Data Files

Our code and data files can be found on Github

### 8.2 Contributions

1. Wenqing Zhong: Write code for data processing, Linear Regression, Decision Tree. Write 4.1 Data processing and cleaning, 4.2.1 Linear Regression, 4.2.2 Decision Tree, 5. Result and evaluation

2. Ziyi Xia: Find prior work and data source. Write code for data fetching, Deep Neural Network. Write 1. Abstract and Introduction, 3. Data, 4.2.3 Deep Nerual Network, 6. Conclusion, 7. Future Work

3. Weiran Wang: Write code for data visualization. Write 2. Previous Reasearch, 3. Data