Project Title and Team Members

Feature Engineering and Machine learning for Stock Analysis and Prediction

Group 1: Paul Phillips, Qi Cai, Xin Gao

GitHub link at https://github.com/pcp0019/CSCE-5222-Group-1/

Goals and Objectives:

Our goal is to use feature engineering, data preprocessing, and machine learning together to predict stock price. We will download a dataset, adjust, add and remove features as needed, then use machine learning models to make predictions and test how salient these predictions are based on metrics such as r2 score.

Motivation

There are a number of factors that affect a stock's price, and these factors change every second. Finding the relationships between these factors is key to determining the value of a stock. This raises the question: how can we know which feature is more important than the other? How can we know how one feature affects another feature? What we need to do is to use feature engineering techniques to get this answer, and alter features as needed to improve predictive modeling. After we find out or get some patterns among them, then we may use these patterns and several regression machine learning models to apply to make stock predictions.

Significance

A stock's price changes often, and if we can use feature engineering to improve predictive models it will enable a better understanding about what goes into a stock's valuation and the factors that can cause this valuation to change. This knowledge can be useful for companies looking to improve their present or future stock value and prevent factors that will lead to valuation decline.

Objectives

Perform feature engineering techniques to examine and alter a feature set to create a better machine learning model for predicting stock price.

Features

The features we start with from our dataset are: High, Low, Open, Close, Volume, Adj Close. High represents the highest value of the stock for that day and Low is the lowest value. Close represents the value of the stock before trading ends for that day. Volume represents how much of that stock (represented by shares, not cost) was actually bought or sold in a trading day. Adj Close is similar to Close but it has some modifiers applied to it to take into account actions like dividend distribution. Different mathematical methods are then applied to these features to get new features for use in our dataset, such as H-L (High price minus Low price), and O-C (Open price minus Close price), and Daily Value Proportion.

Related Work:

Sidra Mehtab, Jaydip Sen & Abhishek Dutta have used NIFTY 50 index values of the National Stock Exchange (NSE) of India and built eight regression models that predicted the open values of NIFTY 50 for the period December 31, 2018 till July 31, 2020. [1]

Rebwar M. Nabi and his research team collected data from NASDAQ and S&P, constructing new features and used the WEKA evaluation method. They finally proposed a new feature engineering approach for stock prediction.[2]

Jigar Patel and his research team used ANN, SVC, random forest and naïve-Bayes to predict direction of movement of stock price index for Indian stock markets by analyzing data like open, high, low & close prices. They found the random forest model showed a best performance than the other three.[3]

Tsai and Wang focus on combining ANN and decision trees to create a stock price forecasting model to predict Taiwan stock prices. The experimental result shows that the combined DT+ANN model has 77% accuracy, which is higher than the single ANN and DT models over the electronic industry. [4]

EK Ampomah, Z Qin, G Nyame compare the effectiveness of tree-based ensemble ML models (Random Forest (RF), XGBoost Classifier (XG), Bagging Classifier (BC), AdaBoost Classifier (Ada), Extra Trees Classifier (ET), and Voting Classifier (VC)) in forecasting the direction of stock price movement. They use eight different stock data from three stock exchanges (NYSE, NASDAQ, and NSE). They found that AdaBoost Classifier has the best performance on their training and test dataset. [5]

Mojtaba Nabipour and his research team compare nine machine learning models and two l deep learning methods to reduce the risk of trend prediction by using stock price data from Tehran stock exchange. They found that RNN and LSTM outperform other prediction models with a considerable difference [6]

Dataset:

Our dataset was downloaded from Yahoo finance. Using Costco stock price from 2013-1-1 to 2023-11-1.

Detail design of Features:

We used feature engineering to calculate additional features to add to the original dataset. Those features are:

1) Daily Value Proportion, an original feature that is calculated by first calculating the 75th percentile value and the 25th percentile closing stock value. After that, we perform the calculation:

Daily Value Proportion = (Daily Close Price – 25th percentile)/ (75th percentile – 25th percentile)

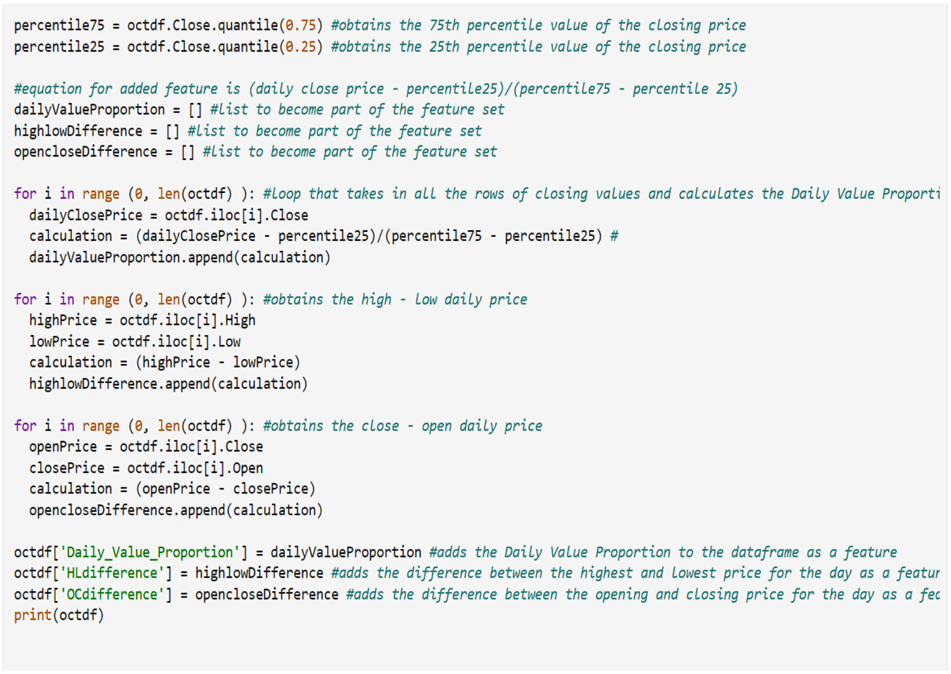
This calculation is performed for all the stock values in the entire timeframe of our dataset. The numerator's calculation is to keep the stock price in the range of -2 to 2, a form of normalization. The 75-25 percentile difference in the denominator was chosen to account for a stock's regular pattern of value across a period of time, without regard for the higher or lower extremes. In this way, Daily Value Proportion demonstrates how the daily stock price relates to its regular pattern of value, allowing it to become useful for predicting the overall price of the stock over time.

For example, if we assume the prices are 100, 110, 115, 120, 125, 130, 140, 150, 160, 170, the 75% percentile is 157.5, the 25% percentile is 115. If daily value is 170, then the Daily Value Proportion is (170 – 115)/(157.5 - 115) = 1.29, showing that it is higher at this point in time than the stock's regular value.

2) H-L is the High Low Difference, calculated as the stock's price at the high point of the day minus its price at the low point of the day. It will show the difference between the maximum and minimum value that could have been gotten from trading that stock in a certain day.

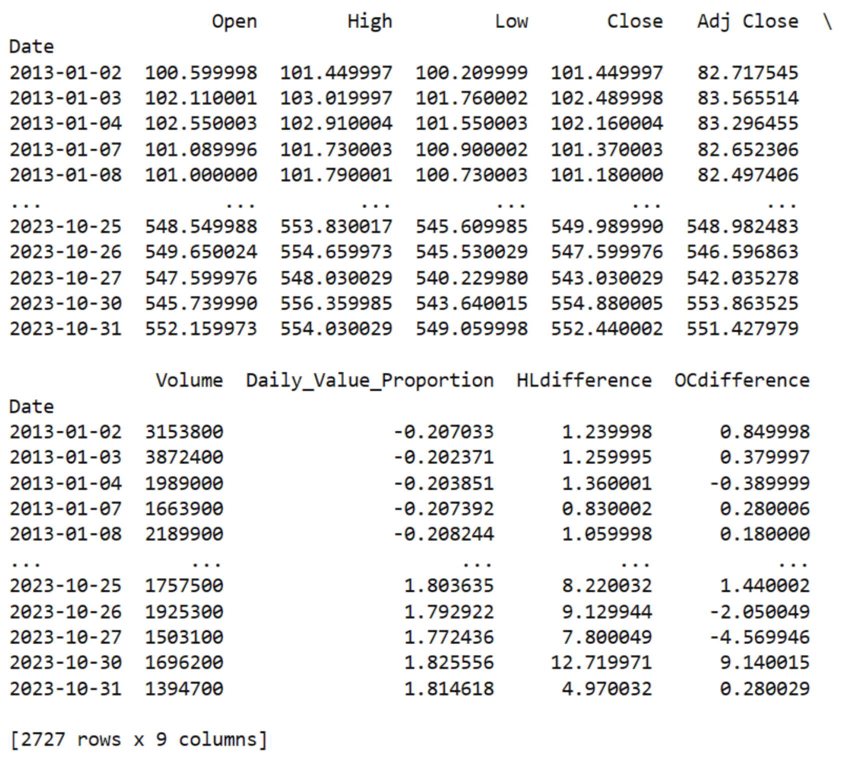
3) O-C is the Open Close Difference, calculated as the stock's daily opening price minus it's daily closing price. This shows how much value it had at the earliest tradable moment as compared to its last tradable moment for a given day.

With these three new features, machine learning models should be able to predict the stock prices more clearly.



```python
percentile75 = octdf.Close.quantile(0.75) #obtains the 75th percentile value of the closing price
percentile25 = octdf.Close.quantile(0.25) #obtains the 25th percentile value of the closing price

#equation for added feature is (daily close price - percentile25)/(percentile75 - percentile 25)
dailyValueProportion = [] #list to become part of the feature set
highlowDifference = [] #list to become part of the feature set
opencloseDifference = [] #list to become part of the feature set

for i in range (0, len(octdf) ): #loop that takes in all the rows of closing values and calculates the Daily Value Proporti
  dailyClosePrice = octdf.iloc[i].Close
  calculation = (dailyClosePrice - percentile25)/(percentile75 - percentile25) #
  dailyValueProportion.append(calculation)

for i in range (0, len(octdf) ): #obtains the high - low daily price
  highPrice = octdf.iloc[i].High
  lowPrice = octdf.iloc[i].Low
  calculation = (highPrice - lowPrice)
  highlowDifference.append(calculation)

for i in range (0, len(octdf) ): #obtains the close - open daily price
  openPrice = octdf.iloc[i].Close
  closePrice = octdf.iloc[i].Open
  calculation = (openPrice - closePrice)
  opencloseDifference.append(calculation)

octdf['Daily_Value_Proportion'] = dailyValueProportion #adds the Daily Value Proportion to the dataframe as a feature
octdf['HLdifference'] = highlowDifference #adds the difference between the highest and lowest price for the day as a featur
octdf['OCdifference'] = opencloseDifference #adds the difference between the opening and closing price for the day as a fec
print(octdf)
```

As displayed in a table:



```
                 Open        High         Low       Close   Adj Close  \
Date
2013-01-02  100.599998  101.449997  100.209999  101.449997   82.717545
2013-01-03  102.110001  103.019997  101.760002  102.489998   83.565514
2013-01-04  102.550003  102.910004  101.550003  102.160004   83.296455
2013-01-07  101.089996  101.730003  100.900002  101.370003   82.652306
2013-01-08  101.000000  101.790001  100.730003  101.180000   82.497406
...                ...         ...         ...         ...         ...
2023-10-25  548.549988  553.830017  545.609985  549.989990  548.982483
2023-10-26  549.650024  554.659973  545.530029  547.599976  546.596863
2023-10-27  547.599976  548.030029  540.229980  543.030029  542.035278
2023-10-30  545.739990  556.359985  543.640015  554.880005  553.863525
2023-10-31  552.159973  554.030029  549.059998  552.440002  551.427979

             Volume  Daily_Value_Proportion  HLdifference  OCdifference
Date
2013-01-02  3153800               -0.207033      1.239998      0.849998
2013-01-03  3872400               -0.202371      1.259995      0.379997
2013-01-04  1989000               -0.203851      1.360001     -0.389999
2013-01-07  1663900               -0.207392      0.830002      0.280006
2013-01-08  2189900               -0.208244      1.059998      0.180000
...             ...                     ...           ...           ...
2023-10-25  1757500                1.803635      8.220032      1.440002
2023-10-26  1925300                1.792922      9.129944     -2.050049
2023-10-27  1503100                1.772436      7.800049     -4.569946
2023-10-30  1696200                1.825556     12.719971      9.140015
2023-10-31  1394700                1.814618      4.970032      0.280029

[2727 rows x 9 columns]
```

Data pre-processing step

The features in this dataset vary in scale. The above table demonstrates how some features, such as Volume, are represented by 8 digit numbers, while other features, such as Daily_Value_Proportion, are between -2 and 2. HL difference and OC difference might be in the single or double digits. Meanwhile, Open, High, Low, and Close tend to be values above 100. These scale differences could interfere with the ability of our machine learning models to properly learn from the data, since the larger values could cause bias.

Performing some feature engineering will be necessary here. First of all, the feature Adj Close is to similar to Close (our target variable), so it was dropped from the dataset. Also, Volume lacks relevance to our prediction and is such a large number that is will skew results, so it was dropped. However, there is still a scaling issue with the remaining data, so we decided to do normalization on the dataset.

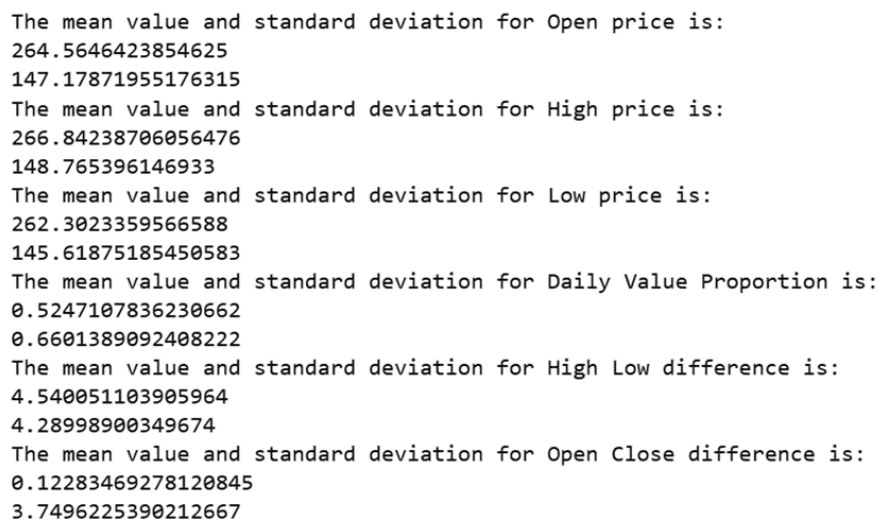
Z Score Normalization step

The chosen normalization method is based on calculating Z-scores for the features. Z scores will keep all the data in a smaller range between around -2 to 2. The Z-score formula used here is:

Z = (feature data point – mean(feature))/(standard deviation of feature)

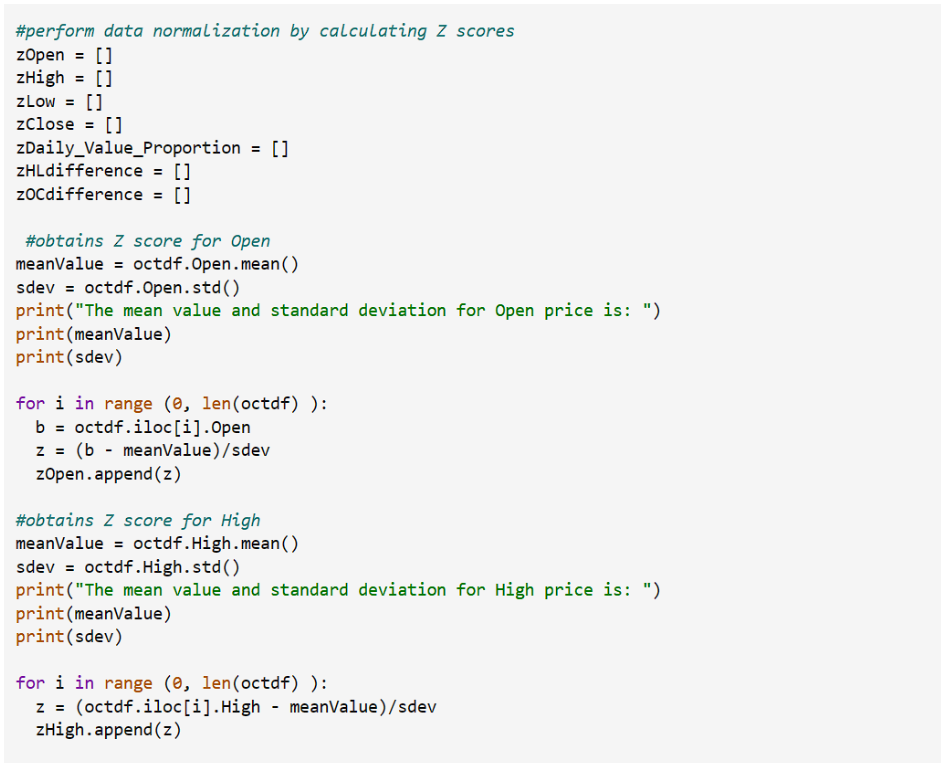
The feature data points are already given in the dataset, but the mean and standard deviation need to be calculated for each feature. They are demonstrated below:

First, we calculate the standard mean and standard deviation of our features. The result of mean and standard deviation among features could range in the hundreds for some features, or less than 10 for other features. Like the following screen shot:



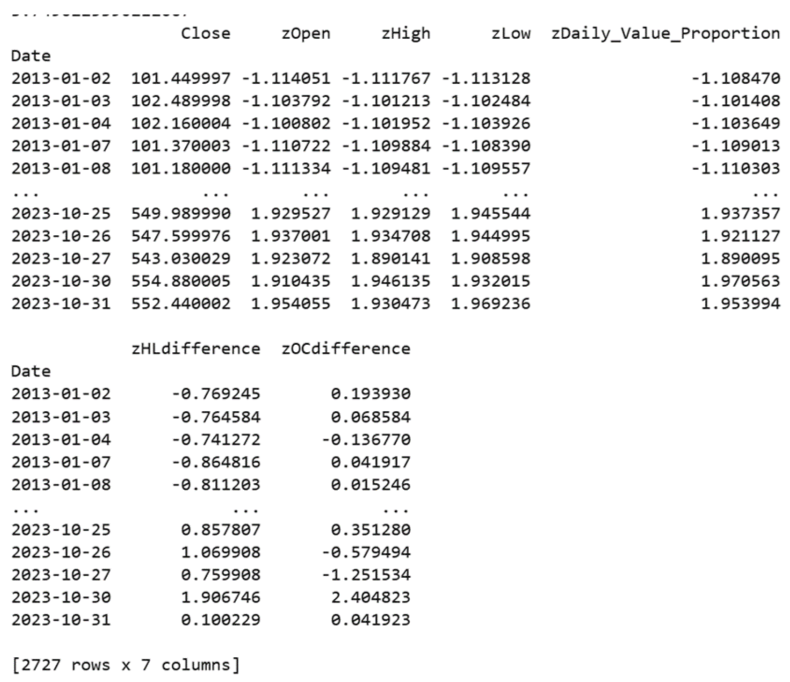
```
The mean value and standard deviation for Open price is:
264.5646423854625
147.17871955176315
The mean value and standard deviation for High price is:
266.84238706056476
148.765396146933
The mean value and standard deviation for Low price is:
262.3023359566588
145.61875185450583
The mean value and standard deviation for Daily Value Proportion is:
0.5247107836230662
0.6601389092408222
The mean value and standard deviation for High Low difference is:
4.540051103905964
4.28998900349674
The mean value and standard deviation for Open Close difference is:
0.12283469278120845
3.7496225390212667
```

These mean and standard deviation values show the difference in scale of the features. Some features have these values as low as 0.1228 and 3.749, while others have them as high as 264.564 and 147.178. Now the Z-score can be calculated and added to a new column in a data frame to replace the unscaled, raw value.



```python
#perform data normalization by calculating Z scores
zOpen = []
zHigh = []
zLow = []
zClose = []
zDaily_Value_Proportion = []
zHLdifference = []
zOCdifference = []

 #obtains Z score for Open
meanValue = octdf.Open.mean()
sdev = octdf.Open.std()
print("The mean value and standard deviation for Open price is: ")
print(meanValue)
print(sdev)

for i in range (0, len(octdf) ):
  b = octdf.iloc[i].Open
  z = (b - meanValue)/sdev
  zOpen.append(z)

#obtains Z score for High
meanValue = octdf.High.mean()
sdev = octdf.High.std()
print("The mean value and standard deviation for High price is: ")
print(meanValue)
print(sdev)

for i in range (0, len(octdf) ):
  z = (octdf.iloc[i].High - meanValue)/sdev
  zHigh.append(z)
```

The new columns are named “zOpen”, “zHigh”, “zLow”, and so on. The old columns are dropped from the dataset as they are no long needed. The new columns will be our new Z-score normalized features.



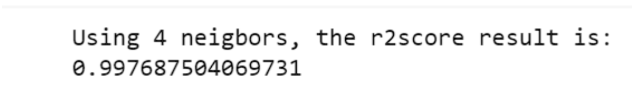
```
                 Close     zOpen     zHigh      zLow  zDaily_Value_Proportion
Date
2013-01-02  101.449997 -1.114051 -1.111767 -1.113128                -1.108470
2013-01-03  102.489998 -1.103792 -1.101213 -1.102484                -1.101408
2013-01-04  102.160004 -1.100802 -1.101952 -1.103926                -1.103649
2013-01-07  101.370003 -1.110722 -1.109884 -1.108390                -1.109013
2013-01-08  101.180000 -1.111334 -1.109481 -1.109557                -1.110303
...                ...       ...       ...       ...                      ...
2023-10-25  549.989990  1.929527  1.929129  1.945544                 1.937357
2023-10-26  547.599976  1.937001  1.934708  1.944995                 1.921127
2023-10-27  543.030029  1.923072  1.890141  1.908598                 1.890095
2023-10-30  554.880005  1.910435  1.946135  1.932015                 1.970563
2023-10-31  552.440002  1.954055  1.930473  1.969236                 1.953994

            zHLdifference  zOCdifference
Date
2013-01-02      -0.769245       0.193930
2013-01-03      -0.764584       0.068584
2013-01-04      -0.741272      -0.136770
2013-01-07      -0.864816       0.041917
2013-01-08      -0.811203       0.015246
...                   ...            ...
2023-10-25       0.857807       0.351280
2023-10-26       1.069908      -0.579494
2023-10-27       0.759908      -1.251534
2023-10-30       1.906746       2.404823
2023-10-31       0.100229       0.041923

[2727 rows x 7 columns]
```

Analysis, implementation and preliminary results:

1. K Neighbors Regression

We use this Z-score normalized dataset to train six different machine learning models. One of these models is K-Nearest Neighbor, a common machine learning technique that makes predictions based on identifying a certain number of close datapoints. The number of these datapoints is assigned by the user (the K value, set to 4 in this case). While this method can be used for classification, predicting the value of a stock is a regression problem so an average of values is from this number of datapoints is used to make the regression prediction. R2 score will be the metric used to measure how well the model has performed this task. In our K Neighbors Regression model, Train/Test split of the data was at 70% reserved for training, 30% reserved for testing.

The screenshot below demonstrates results:

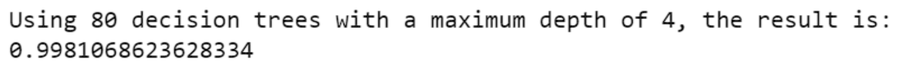


```
Using 4 neigbors, the r2score result is:
0.997687504069731
```

2. Random Forest regression

Random Forest regression is an algorithm designed around making a “forest” of decision trees to make a prediction. Hyperparameters such as the number or depth of these trees can be set by the user. The forest will be built and the decision trees trained separately, and eventually each individual tree’s prediction will be used as “ensemble input” to generate an overall value. Since our stock price dataset is changeable, this kind of complex non-linearity is very suitable for using random forest. Besides, random forest shows good robustness when facing abnormal value and noise data, which are very common in our dataset. We use a 30% train/test split, and 80 decision trees with a maximum depth of 4 to train a random forest regression model. Overfitting is often a problem when using machine learning models, so limiting the maximum depth can help with that.

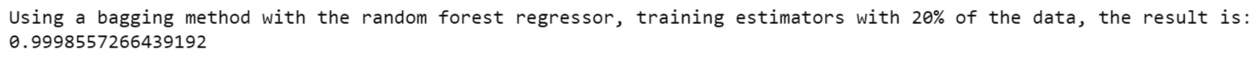
We get a r2 score as 0.9981, which is also a higher score than the K Neighbors Regression model.


```
Using 80 decision trees with a maximum depth of 4, the result is:
0.9981068623628334
```

3. Bagging with random forest regression

After applying random forest regression, we are looking for trying other random forest-based algorithms. Bagging with random forest could be a choice. It is also an ensembled method. The training dataset will be randomly sampled according to a percentage of the total data, and then the algorithm fits a model on these samples. This has the effect of increasing randomness in the training phase, which is helpful so that the model does not overfit on the training data, leaving it unable to perform properly in the test or on other data samples. The algorithm will then take the results found in each subset, and average them out and predict this average value. Bagging can also reduce the variance of the model and improve the generalization performance of the model.

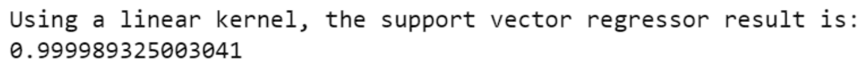
In this model, the Random Forest Regressors here uses 20 decision trees, and each of the 10 subsets uses 20% of the whole data, which will be the main idea of this model.


```
Using a bagging method with the random forest regressor, training estimators with 20% of the data, the result is:
0.9998557266439192
```

We also got a very high R2 score as 0.9998, an improvement on the first Radom Forest Method.

4. Support Vector Regression

Support Vectors are commonly used in machine learning, and we choose to use this algorithm to solve our regression task. An Support Vector Regression model has the ability to lower the amount of error in a prediction by finding a proper hyperplane that will ensure most of the data points will be inside the margin. Based on way the feature types relate to one another the data is probably linearly separable to a significant degree. This linear separability means that it would probably be best to use a linear kernel function, rather than a polynomial kernel function to do the stock price prediction.


```
Using a linear kernel, the support vector regressor result is:
0.999989325003041
```

From the Support Vector Regression model with linear kernel, we get a 0.99998 r2 score, which is very high among our models.

5. Naïve Bayes

Naïve Bayes is a very different model attempted for this dataset. It makes a prediction by calculating the most likely outcome from a Bayes Theorem probability calculation. However, since it assumes feature independence and our features have relations to one another, we do not expect it to yield very accurate results in this case. Since this model is used for classification and not regression, we create a new y set called rise or fall. We use the next day's close price to minus today's close price to know if the close price between these two days is up or down. Up will be recorded as 1 and down will be 0. A new y set is created full of 1s and 0s, and we remove the last row to remove NaN value.

With this feature as our new target, we trained and tested the Naïve Bayes model. The results can be seen in the screenshot below:

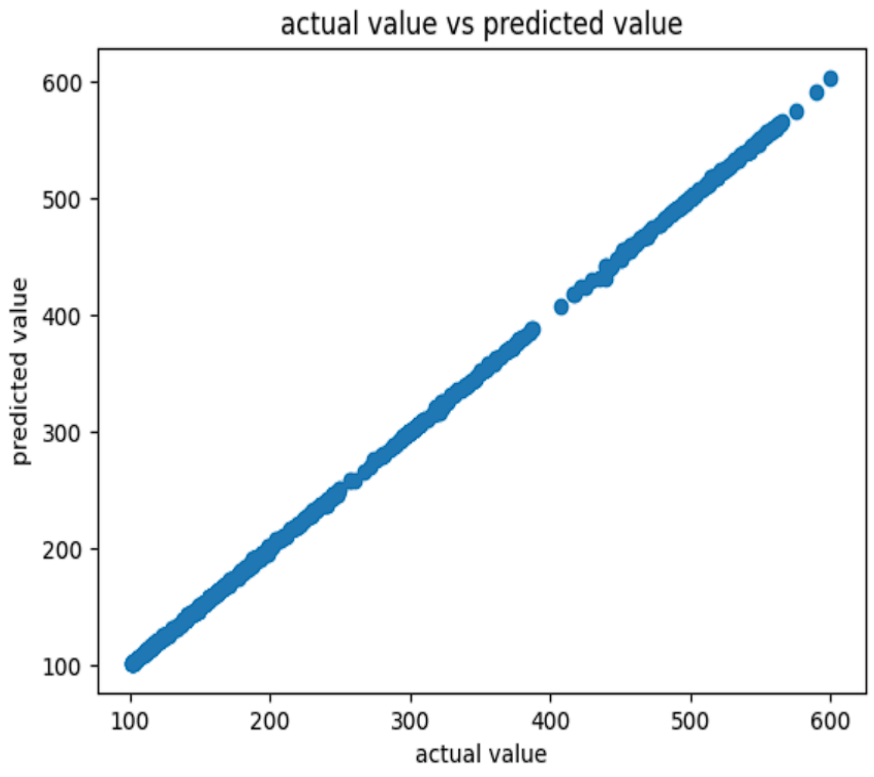
```
Accuracy: 0.5164835164835165
              precision    recall  f1-score   support

           0       0.45      0.16      0.24       255
           1       0.53      0.82      0.65       291

    accuracy                           0.52       546
   macro avg       0.49      0.49      0.44       546
weighted avg       0.49      0.52      0.46       546
```

The accuracy of 0.51 was to be expected given the inherent assumptions of Naïve Bayes. The F1 score in predicting fall is 0.24, while in predicting rise it is 0.65, much higher. This difference is also shown in the recall value: 0.82 in predicting rise and 0.16 in predicting fall.

6. Gradient Boost Regressor

The last model we used was Gradient Boost Regressor. This method makes predictions based on decision trees, with a special focus on lowering the prediction error rate. Since the gradient boost method is being used for regression and not classification, the target will be the closing price, like in the K-Neighbors model. The data was trained on 70% of the data and tested on 30%, using decision trees of depths 3 and 100 estimators. The results are demonstrated in the graph screenshot below:

```
R^2 score: 0.9999488884550308
```



We got the highest R2 score as 0.9999 among our six models. The model showed a good linear relationship and the predicted price matched very closely with the actual price.

Conclusion

K Neighbors Regression	0.9976
Random Forest Regression	0.9981
Bagging with random forest Regression	0.9985
Support Vector Regression	0.9999
Naive Bayes	0.5164
Gradient Boost Regressor	0.9999

After training and testing 6 models with different algorithms, we get several different but good predictions results, except for Naïve Bayes. This is probably because of the relationship between our features, and this may make influence to its' prediction. Our best prediction model is Gradient Boosting Regressor, with a 0.9999 r2 score, make its predicted price match the actual price very closely.

Project Management

Work completed:

Paul Phillips: Data preprocessing, Daily Value Proportion implementation, K Neighbors model implementation, Z-score normalization implementation, Random Forest model implementation, Bagging with Random Forest model implementation, PPT making, project draft report writing, final project report writing

33.3%

Xin Gao: Comment modification, Daily Value Proportion concept creation, idea generation, Naïve Bayes model implementation, project draft report writing, final project report writing

33.3%

Qi Cai: Comment modification, Daily Value Proportion concept creation, project planning, Gradient boost regression model implementation, project draft report writing, final project report writing

33.4%

References

[1] Mehtab, Sidra, Jaydip Sen, and Abhishek Dutta. "Stock price prediction using machine learning and LSTM-based deep learning models." Machine Learning and Metaheuristics Algorithms, and Applications: Second Symposium, SoMMA 2020, Chennai, India, October 14– 17, 2020, Revised Selected Papers 2. Springer Singapore, 2021. (Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models | SpringerLink)

[2] Nabi, Rebwar M., Soran AB Saeed, and Abdulrahman MW Abdi. "Feature Engineering for Stock Price Prediction." Int. J. Adv. Sci. Technol 29.12s (2020): 2486-2496.(https://www.researchgate.net/profile/Rebwar- Nabi/publication/342339410

[3] Patel, Jigar, et al. "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques." Expert systems with applications 42.1 (2015): 259-268.

[4] Tsai, Chih F., and Sammy P. Wang. "Stock price forecasting by hybrid machine learning techniques." Proceedings of the international multiconference of engineers and computer scientists. Vol. 1. No. 755. 2009.

[5] Ampomah, Ernest Kwame, Zhiguang Qin, and Gabriel Nyame. "Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement." Information 11.6 (2020): 332.

[6] Nabipour, Mojtaba, et al. "Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis." IEEE Access 8 (2020): 150199-150212.