

ECMM445 Learning from Data Coursework

Introduction

Stock prices prediction is one of the toughest tasks for any investor who wishes to maximise his returns. The movement of prices depends on various market factors such as supply and demand, profit made by company, dividend it pays and so on. However, many of these factors cannot be quantified. Hence widely used practice is to analyse charts and movement patterns of stock over a period of time to predict next price. Hence, pattern of stock movement is a key factor in determination of future prices.

Objective

In this work, we will predict the stock prices. The data is obtained from Kaggle for AT & T stock prices from 2000 to 2020. We will extract data, identify any inconsistency, missing data, outliers in data, clean and pre-process data and then predict future prices using multiple linear regression, Lasso, Ridge and LSTM (Long Short-Term Memory) and try to regularise model to make it robust. Our objective is to predict the stock price based on previous prices. The price of stock for last few days will be feature vector to predict next day price. The features of data is discussed in Data Acquisition section.

Case Study

The model we use should give better accuracy on prediction. Since stock prices are volatile, a good selection model and regularisation needs to be done for predictions to be robust. Previous works on stock price prediction using machine learning techniques shows that, prices mainly depend on previous prices of stock and market sentiment. Work by B. Panwar et.al. shows that using linear regression, accuracy obtained was 97.8% [2]. Also, Aaryan et. al. shows that using last 60 days stock prices on model to predict next day price, with dropout of 20% good accuracy of 95% was obtained using LSTM-RNN [3]. Similar study by H. Zhang et. al. has showed that prediction for short term can be done better using linear models than non-linear model such as LSTM. This is due to offset becoming insignificant due to short time frame [4].

Methodology

Data Acquisition

AT & T stock prices data is obtained from Kaggle [1]. This data consists of stock prices from 2000 to 2020 for AT & T. This data has column named marker SBC for stock prices till 30 Nov 2005. Later marker is changed as T in stock listing, due to changes in organisation. We take the same data comprising of both markers SBC and T for our study. Daily stock prices are available in the dataset. "DATE" column indicates the for which date the data is collected. For each date, data consisting of High, low, price, volume and so on are available. High indicates the highest price for the day, low corresponds to lowest price. Volume is the total quantity of stocks traded on that day. Similar stock market indicators are present. However, from previous works we found that prediction can be done using previous prices. Hence, we will use relevant fields like DATE and PRICE of the stock. The data header and columns are as below fig 1 and fig 2.

In [2]: data.head()

Out[2]:

	a_permno	PERMNO	DATE	CUSIP	NCUSIP	COMNAM	TICKER	PERMCO	SHRCD	SHRCLS	...	NAICS	PRIMEXCH	TRDSTAT	SECST
0	66093	66093	20000103	00206R10	78387G10	S B C COMMUNICATIONS INC	SBC	21645	11	NaN	...	NaN	N	A	
1	66093	66093	20000104	00206R10	78387G10	S B C COMMUNICATIONS INC	SBC	21645	11	NaN	...	NaN	N	A	
2	66093	66093	20000105	00206R10	78387G10	S B C COMMUNICATIONS INC	SBC	21645	11	NaN	...	NaN	N	A	
3	66093	66093	20000106	00206R10	78387G10	S B C COMMUNICATIONS INC	SBC	21645	11	NaN	...	NaN	N	A	
4	66093	66093	20000107	00206R10	78387G10	S B C COMMUNICATIONS INC	SBC	21645	11	NaN	...	NaN	N	A	

Figure 1 Data Header

```
RangeIndex: 5287 entries, 0 to 5286
Data columns (total 29 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   a_permno    5287 non-null   int64
1   PERMNO      5287 non-null   int64
2   DATE        5287 non-null   int64
3   CUSIP       5287 non-null   object
4   NCUSIP      5287 non-null   object
5   COMNAM      5287 non-null   object
6   TICKER      5287 non-null   object
7   PERMCO      5287 non-null   int64
8   SHRCD       5287 non-null   int64
9   SHRCLS      0 non-null      float64
10  ISSUNO      5287 non-null   int64
11  EXCHCD      5287 non-null   int64
12  HEXCD       5287 non-null   int64
13  SICCD       5287 non-null   int64
14  HSICCD      5287 non-null   int64
15  HSICMG      4787 non-null   float64
16  HSICIG      4787 non-null   float64
17  NAMEENDT    7 non-null      float64
18  TSYMBOL     4787 non-null   object
19  NAICS       4170 non-null   float64
20  PRIMEXCH    5287 non-null   object
21  TRDSTAT     5287 non-null   object
22  SECSTAT     5287 non-null   object
23  PRC         5287 non-null   float64
24  VOL         5287 non-null   int64
25  OPENPRC     5287 non-null   float64
26  ASKHI       5287 non-null   float64
27  BIDLO       5287 non-null   float64
28  BID         5263 non-null   float64
dtypes: float64(10), int64(11), object(8)
memory usage: 1.2+ MB
```

Figure 2 Data Columns

All columns are checked for missing values and anomalies. We find that there are no missing data in our data set. We convert date column into pandas datetime format which helps in analysing in next steps. This is done using `datetime.strptime()` method from pandas library. Rolling average for 100 days is plotted along with stock prices for analysis of movement of stock prices. We see that stock prices are non-linear and deviates from average value in both direction in short time frame due to market volatility.

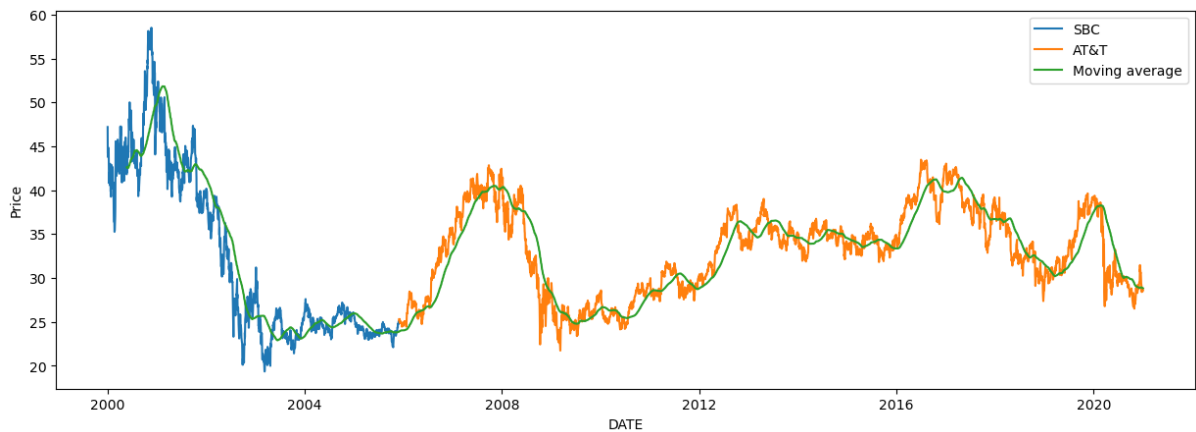


Figure 3 Time series plot of stock prices

Data Cleaning and Training and test data

In this work, previous 100 days stock prices are taken as feature vector to predict next day's (100+1 day) stock prices. We will split data in 70% for training data and 30% for test data. Previous 100 days stock prices are taken as independent variable and next day is dependent variable.

```
>>>df_train = np.array(df[:int(df.shape[0]*0.7)])
>>>df_test = np.array(df[int(df.shape[0]*0.7):])
```

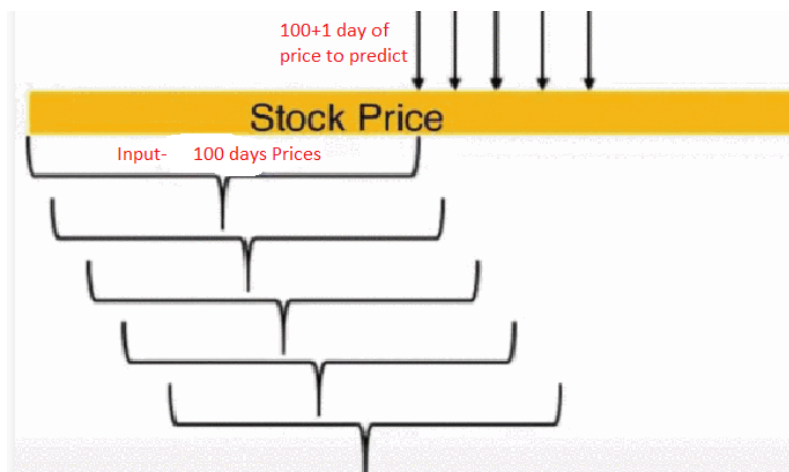
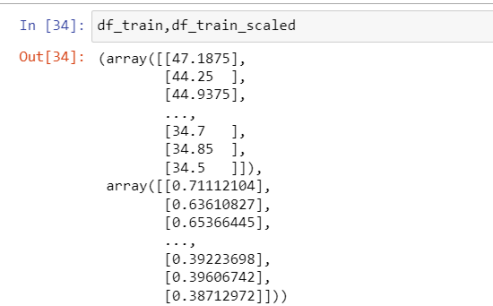


Figure 4 Input feature vector and output

The stock prices range from 20\$ to 60\$. This needs to be scaled in the range [0 to 1]. This gives better performance in prediction of values. Hence, we use MinMaxScaler from scikit library to implement this method. After we predict the output, we take inverse scaling to get the output in same scale as input.

```
>>>from sklearn.preprocessing import MinMaxScaler
>>>scaler=MinMaxScaler()
```

```
>>>df_train = scaler.fit_transform(df_train)
>>>df_test = scaler.transform(df_test)
```



```
In [34]: df_train, df_train_scaled
Out[34]: (array([[47.1875],
 [44.25 ],
 [44.9375],
 ...,
 [34.7   ],
 [34.85  ],
 [34.5   ]]),
 array([[0.71112104],
 [0.63610827],
 [0.65366445],
 ...,
 [0.39223698],
 [0.39606742],
 [0.38712972]]))
```

Figure 5 Scaling features

Model Fitting

Linear Regression

We will fit scaled data into linear regression model. Linear regression is the form $y = b_0 + b_1 X_1 + \dots + b_{100} X_{100}$.

$X_1 \dots X_{100}$ are prices for set of 100 days. Y is the price for $X_{100} + 1$ day.

We have 100 features in one feature vector. Hence this will be multiple linear regression. This works on principle of ordinary Least squares without any regularisation. The purpose of this model is to minimise the squared error of the predicted and actual output. The graph obtained for actual test data and predicted data. The MSE obtained for this model is 0.2323. Other metrics are tabulated in table mentioned below. We will try to regularize this model to be robust by using Lasso, Ridge and tuning hyperparameters to get acceptable value of metrics.

```
>>>from sklearn.linear_model import LinearRegression
>>>model=LinearRegression()
>>>model.fit(x_train, y_train)
>>>pred = model.predict(x_test)
```

The graph for predicted and actual values of stock prices is plotted below using linear regression.



Figure 6 Linear regression prediction and actual values

Lasso Regression

Lasso regression estimated coefficients using L1 regularisation. Alpha is the parameter that multiplies the L1 term controlling the regularisation term. After tuning hyper parameters, we find the suitable value for alpha is 0.005. The MSE is found to be 0.2286. The equation for lasso regression loss function is given as:

$$\min_w \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha ||w||_1$$

Table 1 Lasso regression Hyperparameter tuning

Alpha	MSE	R2 Score	MAE
0.0005	0.2286	0.9854	0.3335
0.001	0.2468	0.9843	0.3593
0.01	2.6569	0.8312	1.3190

Ridge Regression

Ridge regression used L2 regularisation. Alpha parameter is used to control amount of shrinkage in the below loss function. The MSE obtained for alpha 0.05 is 0.2320. The loss function for ridge regression is given as below.

$$\min_w ||Xw - y||_2^2 + \alpha ||w||_2^2$$

The relevant code snippet will be as follows for ridge regression.

```
>>>from sklearn.linear_model import Ridge
>>>model=Ridge(alpha=1)
```

Table 2 Ridge regression hyperparameters tuning

Alpha	MSE	R2 Score	MAE
0.01	0.2312	0.9853	0.3281
0.1	0.2284	0.9859	0.3260
1	0.2723	0.9827	0.3616

LSTM

LSTM (Long Short-Term Memory) is type of Recurrent Neural Network. LSTM resolves the drawback of vanishing gradient and boosting gradient in RNN. In LSTM there are 2 transmission state (cell state and hidden state) instead of 1 in RNN. The structure is shown as below.

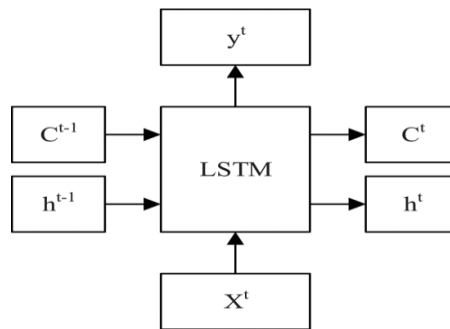


Figure 7 LSTM model

LSTM cells determine c^t and h^t . In LSTM, there are 3 stages. The first stage called forgotten stage. It determines which information c^{t-1} needs to be forgotten when it received an input. Selective memory stage is next stage when information would be selective remembered. The last stage is output stage when h^t is decided by output stage.

In this methodology, we use multi-layer LSTM, with 4 layers. Each has dropout of 20% for regularisation. At the end, there is one dense layer with 1 unit. The model is trained with 30 epoch. The mean squared obtained was 1.4204 and R2 0.9097.



Figure 8 Actual and Predicted value using LSTM model

Evaluation and Results

The evaluation metric used for our analysis here is Mean squared error (MSE), Root Mean squared Error (RMSE) and R2 Score. They can measure how good our model is predicting values closer to actual value. The various metrics are defined as below.

$$RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^T (y_i - \hat{y}_i)^2}$$

$$MSE = \frac{1}{T} \sum_{i=1}^T (y_i - \hat{y}_i)^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^T (y_i - \hat{y}_i)^2}{\sum_{i=1}^T (y_i - \bar{y})^2}$$

The model metrics for models used in our study are tabularised as below.

Table 3 Results of various models

Model	Linear Regression	Ridge regression	Lasso Regression	LSTM
MSE	0.2323	0.2284	0.2286	1.4204
MAE	0.3290	0.3260	0.3335	0.9382
R2 Score	0.9852	0.9859	0.9854	0.9097

Conclusion

In this study we used Linear regression, lasso, ridge and LSTM for prediction of stock prices. From results metrics Ridge regression is performing better as compared to other models. LSTM would have given better results, but various factors included in the stock prices are not included in our dataset. However, linear regression cannot be applied as general rule for stock price prediction, because it can only predict only for short duration of time. We also observe that stock prices are nonlinear function, which should have performed better using a non-linear function such as sigmoid function in LSTM. However, we are only trying to predict it one next step where nonlinear features could be offset.

In future, this could be improved with additional data along with LSTM neural network such as, quarterly profit, dividend price and along with market sentiments such as international policies etc which can be added along with text analytics. This could give better results when all factors of stock market are considered while building model.

References

- [1] <https://www.kaggle.com/datasets/konstantinparfenov/att-sbc-stock-price-data>
- [2] B. Panwar, G. Dhuriya, P. Johri, S. Singh Yadav and N. Gaur, "Stock Market Prediction Using Linear Regression and SVM," *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2021, pp. 629-631, doi: 10.1109/ICACITE51222.2021.9404733.
- [3] Aaryan, B. Kanisha, Vishnupriya Jayaraman, B. Kanisha, "Forecasting stock market price using LSTM-RNN", *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp.1557-1560, 2022
- [4] H. Zhang, "Stock Price Prediction using Linear Regression and LSTM Neural Network," *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, 2022, pp. 182-185, doi: 10.1109/MLISE57402.2022.00043.