NATIONAL INSTITUTE OF TECHNOLOGY
KARNATAKA

PROJECT REPORT ON

“A COMPARATIVE STUDY OF ARIMA-LSTM AND ARIMA-MLP HYBRID MODELS FOR STOCK PRICE FORECASTING”

SUBMITTED BY

ROHIT SUNIL MESHRAM
(192090IT017)

VIKASH RAJA MALLICK
(192454IT027)

UNDER THE GUIDANCE OF
Dr. Biju R. Mohan

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY
KARNATAKA

Abstract

Hybrid models are one of the most widely-used hybrid models that in which a time series is assumed to be composed of two linear and nonlinear components. In this project, the performance of two types of these hybrid models is evaluated for predicting stock prices in order to introduce the more reliable series hybrid model. For this purpose, ARIMA, LSTM and MLPs are elected for constructing hybrid models. Empirical results for forecasting data sets indicate that despite of more popularity of the conventional ARIMA-LSTM model, the ARIMA-MLP hybrid model can overall achieved more accurate results.

Table of contents

Introduction

Among many factors that are involved in making decision about selecting a right forecasting model, accuracy is known as a most effective criteria. Thus, improving forecasting accuracy has become vital for decision makers and managers in various fields of science especially time series forecasting. Many researchers believe that combining different models or using hybrid models can be an effective solution to improve forecasting accuracy and to overcome limitations of single models. Theoretical, as well as, empirical evidences in the literature suggest that by combining inhomogeneous models, the hybrid models will have lower generalization variance or error. On the other hand, the main aim of combined models is to reduce the risk of using an inappropriate model by using several models in structure of hybrid models simultaneously. Typically, the theory of combination evolves from this fact that the underlying process of data generation cannot easily be determined and either one individual model cannot identify the true data generation process. In a other word, a single model may not be totally sufficient to identify all the characteristics of the time series. In recent decades, combination techniques of different forecasting models have attracted the attention of many researchers in many areas especially in financial time series forecasting. Hybrid combination models, especially linear/nonlinear hybrid models, are one of the most common hybrid techniques that used widely used for time series forecasting in various fields.

The literature review indicates that ARIMA and ANN models are the most popular and widely-used linear and nonlinear models, which are applied for constructing series linear/nonlinear combination models, respectively. The reason of this popularity is unique advantages of these two types of models in modeling linear and nonlinear structures. The auto-regressive integrated moving average model is one of the most popular linear time series models that have enjoyed useful applications. The popularity of the ARIMA model is due to its statistical properties as well as to the well-known Box-Jenkins methodology in the model building process. The ARIMA model assumes that there is a linear correlation between the values of a time series and by analyzing historical data, extrapolate the linear relationships in the data, so this statistical model is suitable for linear problems. Due to the ability of ARIMA models in linear modeling, this model is frequently used for building series hybrid models.

Artificial neural networks are also one of the most important and widely used types of nonparametric nonlinear time series models, which have been proposed and examined for time series forecasting. The highlighted feature of neural networks is their nonlinear pattern recognition without any information about the relationships exists in the data. In artificial neural networks, no longer need to specify the form of the particular model. Moreover, the model is adaptively formed based on the features presented from the data. Because of the capacity of these models in nonlinear modeling, they are also frequently used in several studies as a part of series hybrid models. Of course, it must be noted that the real word problems are rarely pure linear or nonlinear, thus the ARIMA and MLP models based on their unique features in linear and nonlinear modeling, are not comprehensive to capture both linear and nonlinear patterns simultaneously. Thus, it seems that combing these models together can be an effective way to forecast real world systems such as financial markets.

In recent years, several series combination models have been developed in the literature, incorporating autoregressive integrated moving average and artificial neural networks and have been applied to time series forecasting with good performance. Generally, by changing the sequence of using ARIMA and MLP models in the series combination methodology, two possible hybrid models e.g. ANN-ARIMA and ARIMA-ANN can be presented.

In this project, the performance of two types of hybrid models for financial time series forecasting are compared together. The main aim of this project is to determine the relative predictive capabilities of the ARIMA-LSTM and ARIMA-MLP models. On the other hand, this project aims to conclude that which hybrid combination is better for constructing series hybrid models for financial time series forecasting. We can use variety of dataset for traning tesing and predicting the future close value of the stock price. One of the dataset we have used is the SPDR S&P 500 ETF Trust (SPY). basic concepts and modeling procedures of ARIMA, LSTM and ANN models for time series forecasting is briefly introduced in this report the series hybrid methodology and basic concepts of the ARIMA-LSTM and the ARIMA-MLP models are descripted. The description of used data sets and obtained results of both hybrid models are also presented in this report. the performance of models in forecasting benchmark data sets are compared together in the next section. And lastly we are concluding the result.

Methodology

Autoregressive integrated moving average (ARIMA) models

ARIMA model is a procedure of forecasting future values of time series that same as statistical models by using historical data generate forecasting value of variables. ARIMA model consists of two main parts, Auto Regressive (AR) and Moving Average (MA) which are combined together and build ARIMA models. The formula of ARIMA model for time series forecasting is shown in Eq. below.

$$\phi(B)\nabla^{d}(y_t - \mu) = \theta(B)\,a_t$$

The modeling procedure of ARIMA models based on the Box–Jenkins methodology always contain three iterative steps, including model identification, parameter estimation and diagnostic checking. These steps are described as follows in detail.

1) Identification: In this step we are searching for the actual values of p (the number of auto regressive), d (the number of differencing) and q (the number of moving average). For this purpose Box and Jenkins proposed the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the sample data as the basic tools to identify the order of the autoregressive integrated moving average models.

2) Estimation: After choosing a specify ARIMA (p, d, q), the parameters which were identified in the previous phase, should be estimated by the ordinary least squared (OLS) method. If the model is not adequate, a new structure of ARIMA model will be identified, and the three previous steps should be repeated until the best structure is found.

Multilayer perceptron neural networks (MLPS) for forecasting time series

One of the most frequently used intelligent models that has been successfully applied in many fields especially, time series modelling and forecasting is ANN model. The main reason of this popularity is that ANN models extrapolate the underlying data generation without any assumption of the model form. Besides another highlighted features of neural networks is that they are universal approximators that can approximate a large class of function accurately. There are various ANN model architectures available in the literature. Although neural net-works have a similar structure, but based on the how to design, distinction between different types of neural networks have been created. Single hidden layer feed forward network (also known as multilayer perceptrons) is the most widely used neural network model architectures for time series modeling and forecasting. In this paper, this type of neural networks is used for all nonlinear modeling. The MLP network consists of three layers including input, hidden and output layers.

Input layer phase: For the time series problems, a MLP is fitted with past lagged value of actual data (y t−1 ,..., y t−p) as an input vector. Therefore, input layer is composed of p nodes that are connected to the hidden layer.

Hidden layer phase: The hidden layer is an interface between input and output layers. The MLPs model which are designed in this paper have a single hidden layer with q nodes. In this step one of the important tasks is determining the type of activation function (g) which is identifying the relationship between input and output layer. Neural networks support a wide range of activation function such as linear, quadratic, tanh and logistic. The logistic function is often used as the hidden layer transfer function that are shown in Eq.

$$g(x) = \frac{1}{1 + \exp(-x)}$$

Output layer phase: In this step, by selecting an activation transfer function and the appropriate number of nodes, the output of neural network is used to predict the future values of time series. In this paper output layer by designed neural networks contains one node because the one –step ahead forecasting is considered. Also, the linear function as a non-linear activation function is introduced for the output layer. The formula of relationship between input and output layer is presented in Eq.

$$y_t = w_0 + \sum_{j=1}^{q} w_j . g\left[w_{0,j} + \sum_{i=1}^{p} w_{i,j} . y_{t-i} \right] + \varepsilon_t$$

It should be noted that deciding the number of neurons in hidden layer (q) and the number of lagged observations, (p) and the dimension of the input vector in input layer are vital parts of neural network architectures, but no methodical rule exists in order to selecting theses parameters and the only possible way to choose an optimal number of p and q is trial and error.

ARIMA-LSTM hybrid model

This hybrid ARIMA-LSTM model is an application of Stock Price Prediction Based on ARIMA-RNN Combined Model. The model is prepared to forecast the daily closing prices of the SPDR S&P 500 ETF Trust (SPY).

A moving average filter is used to separate the daily closing prices into a low volatility times series (moving average output) and a high volatility time series (close – moving average). The moving average period is optimized such that the resulting output obeys a normal (mesokurtic) distribution. A normal distribution is defined as a kurtosis K value of 3. For $K > 3$, the data is leptokurtic and for $K < 3$, the data is platykurtic. The length of data used for determining the kurtosis value was set to 60 days. The length of data used has a direct effect on the kurtosis value. With longer data sets, K is less than 3 for all values of moving average period length. Several common types of moving averages are evaluated from the Ta-Lib library.

An LSTM (Long Short-Term Memory) recurrent neural network is used to forecast the high volatility time series. The neural network is comprised of a four neuron LSTM tensor, a two neuron LSTM tensor, and a single one neuron dense tensor. The high volatility time series is pre-processed with a scalar function to adjust the feature range to between 0 and 1. The LSTM model is fit with early stopping enabled to minimize potential over-fitting. An ARIMA (AutoRegressive Integrated Moving Average) model is used to forecast the low volatility time series. The model is fit for ARIMA parameters p, q, and d. The predicted low and high volatility time series are summed to generate the predicted closing prices.

The accuracy of the ARIMA-LSTM model is evaluated in several ways. MSE, RMSE, and MAPE are utilized to asses the error between the predicted closing prices and the actual closing prices. Two simulations are conducted to asses the model's predictive accuracy. In first simulation, the model's predictions are compared to the previous day's closing price. If the predicted closing price is greater than the previous closing price, then the SPY ETF is predicted to close higher the following day. In the second simulation, the model's predictions are compared to the previous day's prediction. If the predicted closing price is greater than the previous predicted closing price, then the SPY ETF is predicted to close higher the following day.

ARIMA-MLP hybrid models

In series linear/nonlinear combination models, a time series is considered to be composed of a linear autocorrelation structure and a nonlinear component as follows:

$$y = Sum\ (L, N)$$

where, L denotes the linear and N denotes the nonlinear part. These two components have to be estimated from the data. Thus, in the first stage of these models, linear (nonlinear) component is first modeled by ARIMA (MLP) model. Then, the nonlinear (linear) part is modeled by MLP (ARIMA) model using residuals of the first stage. The main idea of series hybrid models comes from this fact that if a time series is modeled by a linear model such as ARIMA, then residuals of the linear model will only contain nonlinear structure. Therefore, the non-linear part of time series can be modeled by residuals. In the similar fashion, if a time series is modeled by a nonlinear model such as MLP, then residuals of the nonlinear model will only contain linear structure. Therefore, the linear part of time series can be modeled by residuals. In this way, series combination models exploit unique attributes and strengths of the ARIMA. as well as MLP model in determining different patterns. Thus, it could be advantageous to model linear and nonlinear patterns separately by using different models and then combine the forecast to improve the overall forecasting performance. In addition, since it is difficult to completely know the characteristics of the data in real problem, this hybrid methodology that model different parts of time series sequentially by using linear and nonlinear modeling capabilities of each individual model, can be a good strategy for improving forecasting accuracy in practical uses. In the next sections, two possible hybrid models based on the sequence of using ARIMA and MLP models in series combination (e.g. ARIMA- MLP and MLP-ARIMA) are presented.

The ARIMA-MLP model

According to the procedure of the series models, in the ARIMA-MLP model, the ARIMA is used in the first permutation to model the linear component. Let e t denote the residual of the ARIMA model at time t, then:

$$e_t = y_t - \hat{L}_t$$

where, the Lˆt is the output of ARIMA model at time t. The ARIMA model left the nonlinear patterns in its residual. Thus, in the second stage, by modeling residuals using MLPs, nonlinear relationships can be discovered. With n input nodes, the MLP model for the residuals will be:

$$e_t = f(e_{t-1}, e_{t-2}, \ldots, e_{t-n}) + \varepsilon_t \quad \Rightarrow \quad \hat{N}'_t = \hat{e}_t = f(e_{t-1}, e_{t-2}, \ldots, e_{t-n})$$

where f is a nonlinear function determined by the MLP, NˆtJ is the forecasting value at time t from the MLP model on the residual data, and ε t is the random error. Note that if the model f is an inappropriate one, the error term is not necessarily random. Therefore, the correct identification is critical.

In this way, the combined forecast will be as follows:

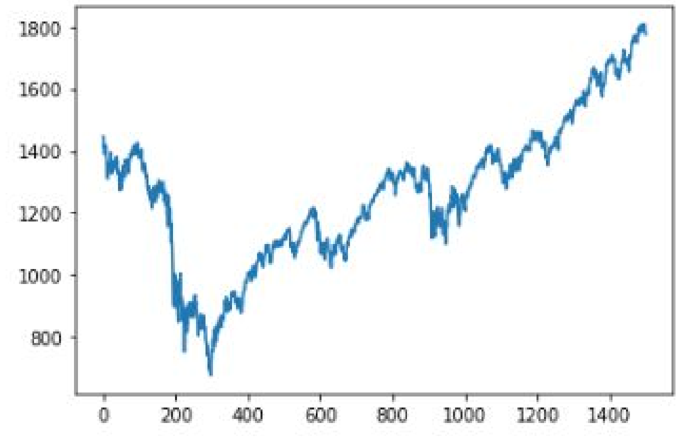
$$\hat{y}_t = \hat{L}_t + \hat{N}'_t$$

Results

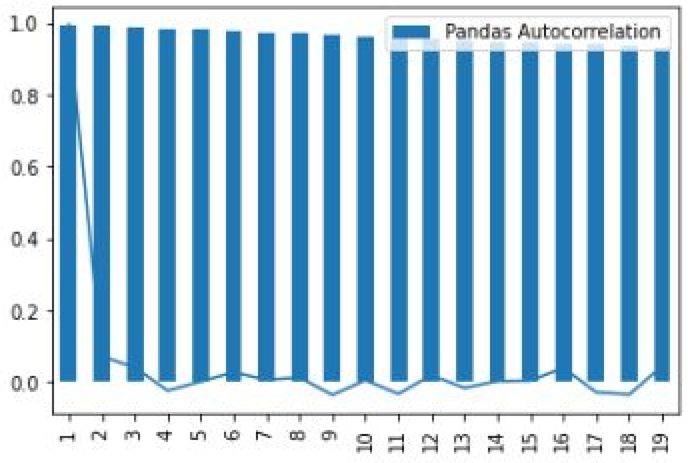
For the implementation of the idea we have written two different programs for each hybrid model. In both the programs we have used a daset file in .csv format. We will see it one after another as follows.

Hybrid-ARIMA-LSTM-Model

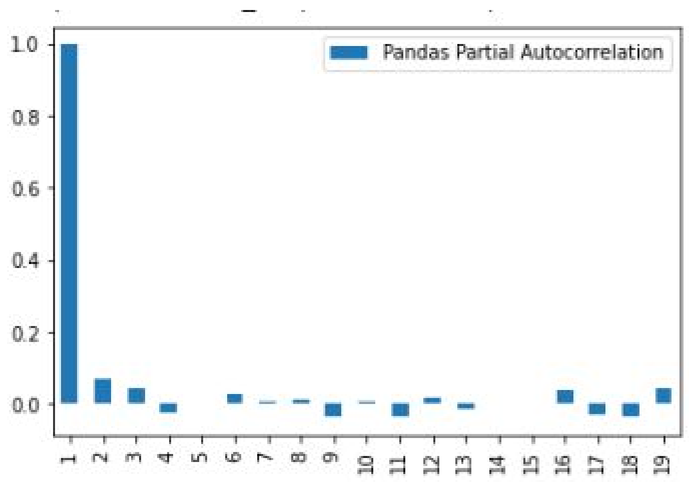
```
(1500, 5)
Index(['open', 'high', 'low', 'close', 'volume'], dtype='object')
```



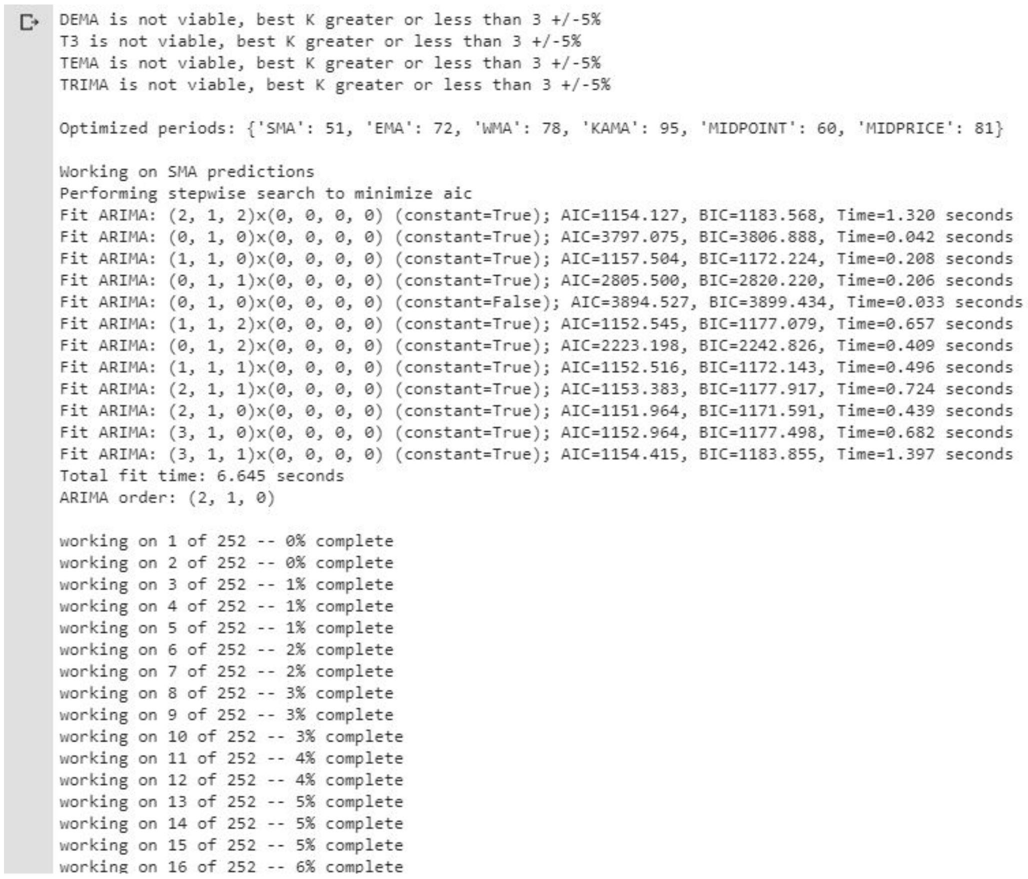
Visualization of the dataset



Pandas Autocorrelation



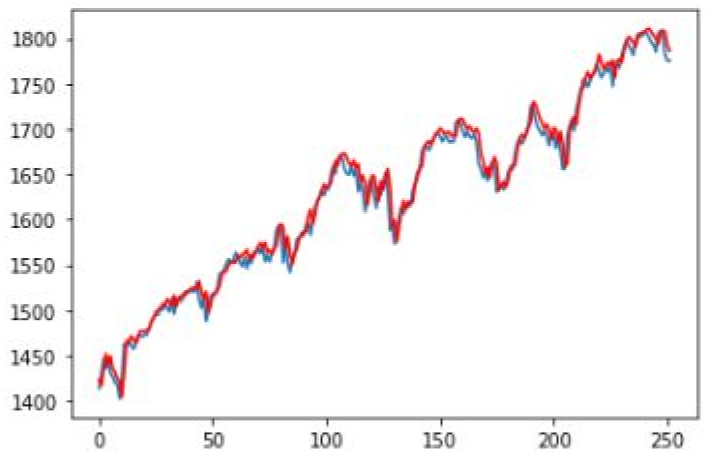
Pandas Partial Autocorrelation



```
DEMA is not viable, best K greater or less than 3 +/-5%
T3 is not viable, best K greater or less than 3 +/-5%
TEMA is not viable, best K greater or less than 3 +/-5%
TRIMA is not viable, best K greater or less than 3 +/-5%

Optimized periods: {'SMA': 51, 'EMA': 72, 'WMA': 78, 'KAMA': 95, 'MIDPOINT': 60, 'MIDPRICE': 81}

Working on SMA predictions
Performing stepwise search to minimize aic
Fit ARIMA: (2, 1, 2)x(0, 0, 0, 0) (constant=True); AIC=1154.127, BIC=1183.568, Time=1.320 seconds
Fit ARIMA: (0, 1, 0)x(0, 0, 0, 0) (constant=True); AIC=3797.075, BIC=3806.888, Time=0.042 seconds
Fit ARIMA: (1, 1, 0)x(0, 0, 0, 0) (constant=True); AIC=1157.504, BIC=1172.224, Time=0.208 seconds
Fit ARIMA: (0, 1, 1)x(0, 0, 0, 0) (constant=True); AIC=2805.500, BIC=2820.220, Time=0.206 seconds
Fit ARIMA: (0, 1, 0)x(0, 0, 0, 0) (constant=False); AIC=3894.527, BIC=3899.434, Time=0.033 seconds
Fit ARIMA: (1, 1, 2)x(0, 0, 0, 0) (constant=True); AIC=1152.545, BIC=1177.079, Time=0.657 seconds
Fit ARIMA: (0, 1, 2)x(0, 0, 0, 0) (constant=True); AIC=2223.198, BIC=2242.826, Time=0.409 seconds
Fit ARIMA: (1, 1, 1)x(0, 0, 0, 0) (constant=True); AIC=1152.516, BIC=1172.143, Time=0.496 seconds
Fit ARIMA: (2, 1, 1)x(0, 0, 0, 0) (constant=True); AIC=1153.383, BIC=1177.917, Time=0.724 seconds
Fit ARIMA: (2, 1, 0)x(0, 0, 0, 0) (constant=True); AIC=1151.964, BIC=1171.591, Time=0.439 seconds
Fit ARIMA: (3, 1, 0)x(0, 0, 0, 0) (constant=True); AIC=1152.964, BIC=1177.498, Time=0.682 seconds
Fit ARIMA: (3, 1, 1)x(0, 0, 0, 0) (constant=True); AIC=1154.415, BIC=1183.855, Time=1.397 seconds
Total fit time: 6.645 seconds
ARIMA order: (2, 1, 0)

working on 1 of 252 -- 0% complete
working on 2 of 252 -- 0% complete
working on 3 of 252 -- 1% complete
working on 4 of 252 -- 1% complete
working on 5 of 252 -- 1% complete
working on 6 of 252 -- 2% complete
working on 7 of 252 -- 2% complete
working on 8 of 252 -- 3% complete
working on 9 of 252 -- 3% complete
working on 10 of 252 -- 3% complete
working on 11 of 252 -- 4% complete
working on 12 of 252 -- 4% complete
working on 13 of 252 -- 5% complete
working on 14 of 252 -- 5% complete
working on 15 of 252 -- 5% complete
working on 16 of 252 -- 6% complete
```

Working on SMA predictions

(similarly it is performing predictions for EMA, WMA, KAMA, MIDPOINT, MIDPRICE)

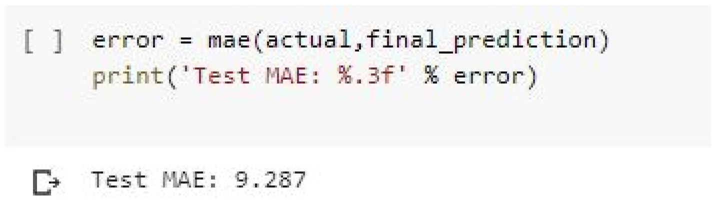


ARIMA-LSTM predictions vs actual close value

(red = predictions, blue=actual close value)

Error calculations

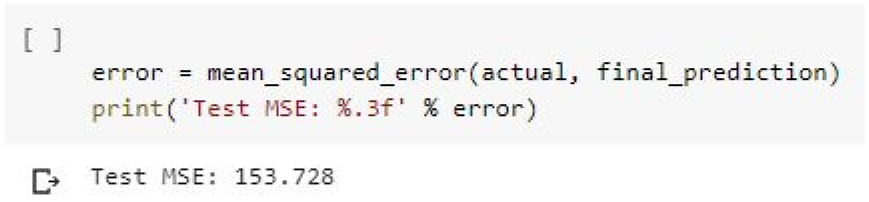
MAE (Mean Absolute Error)



```
[ ]  error = mae(actual,final_prediction)
     print('Test MAE: %.3f' % error)
```

```
Test MAE: 9.287
```

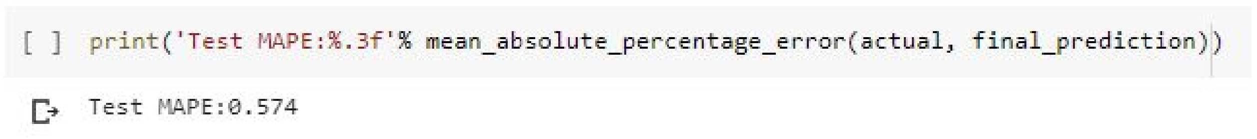
MSE (Mean Square Error)



```
[ ]
     error = mean_squared_error(actual, final_prediction)
     print('Test MSE: %.3f' % error)
```

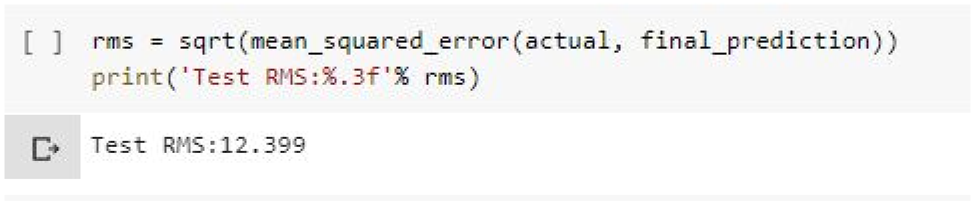
```
Test MSE: 153.728
```

MAPE (Mean Absolute Percentage Error)



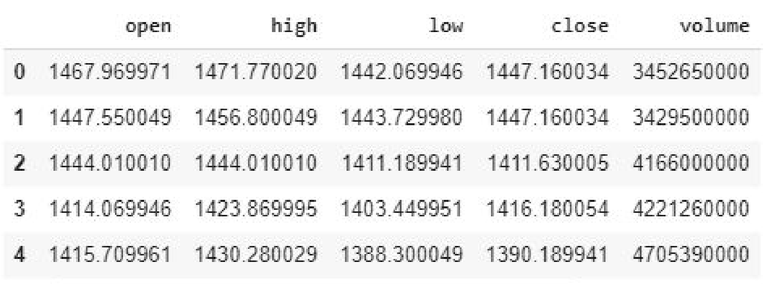
```
[ ]  print('Test MAPE:%.3f'% mean_absolute_percentage_error(actual, final_prediction))

     Test MAPE:0.574
```

RMSE (Root Mean Square Error)

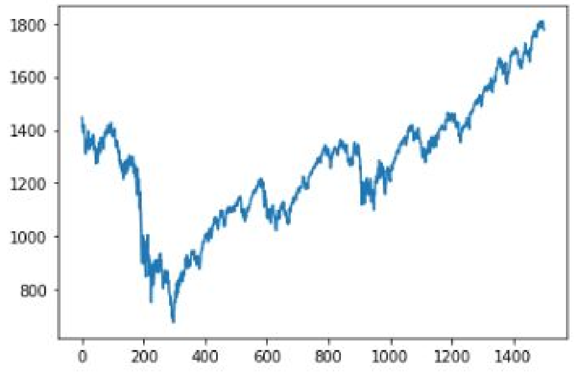


```
[ ]  rms = sqrt(mean_squared_error(actual, final_prediction))
     print('Test RMS:%.3f'% rms)

     Test RMS:12.399
```

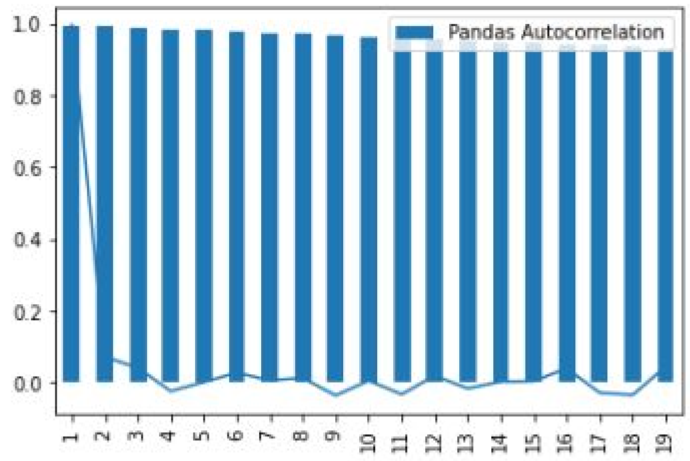
Hybrid_ARIMA_MLP_Model



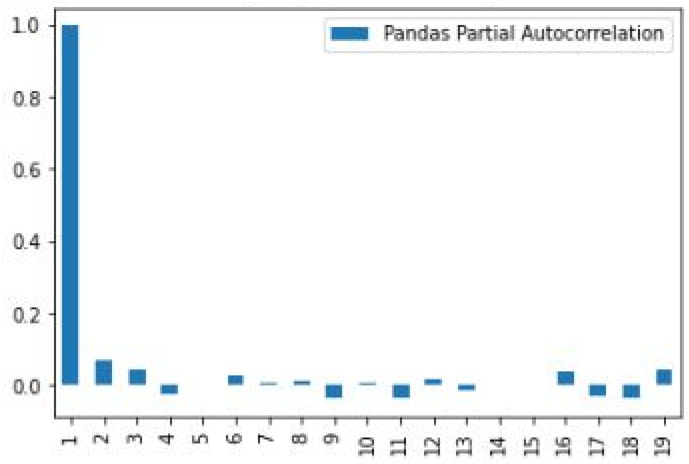
	open	high	low	close	volume
0	1467.969971	1471.770020	1442.069946	1447.160034	3452650000
1	1447.550049	1456.800049	1443.729980	1447.160034	3429500000
2	1444.010010	1444.010010	1411.189941	1411.630005	4166000000
3	1414.069946	1423.869995	1403.449951	1416.180054	4221260000
4	1415.709961	1430.280029	1388.300049	1390.189941	4705390000



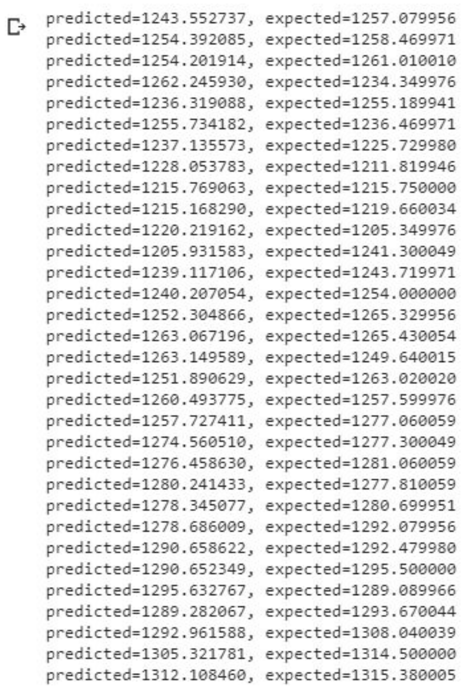
Visualization of dataset with first 5 row



Pandas Autocorrelation

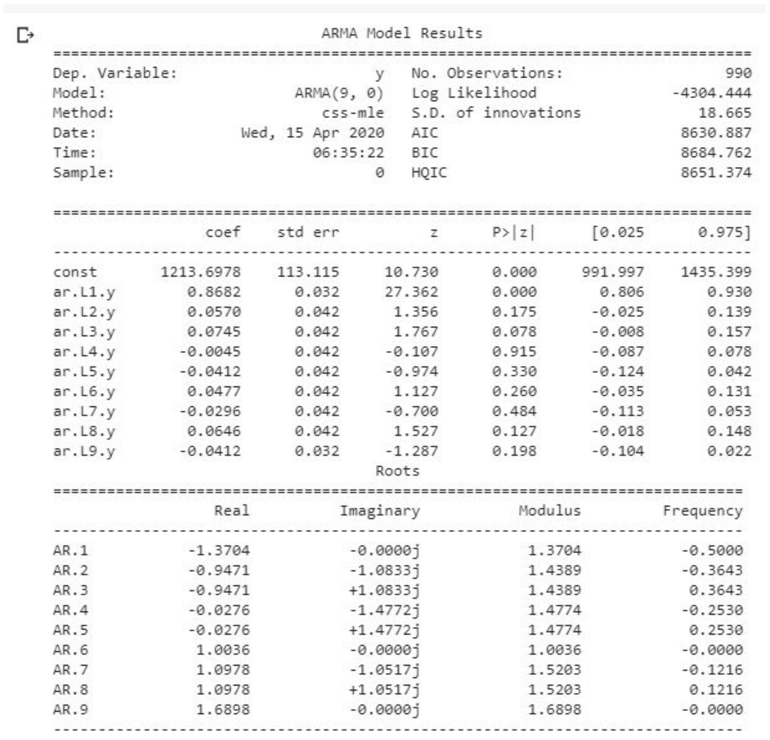


Pandas Partial Autocorrelation



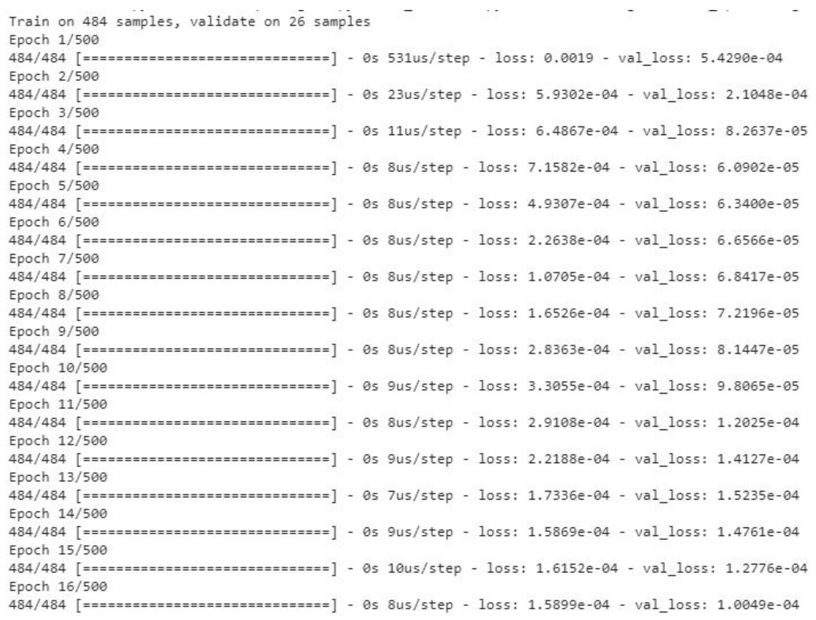
```
predicted=1243.552737, expected=1257.079956
predicted=1254.392085, expected=1258.469971
predicted=1254.201914, expected=1261.010010
predicted=1262.245930, expected=1234.349976
predicted=1236.319088, expected=1255.189941
predicted=1255.734182, expected=1236.469971
predicted=1237.135573, expected=1225.729980
predicted=1228.053783, expected=1211.819946
predicted=1215.769063, expected=1215.750000
predicted=1215.168290, expected=1219.660034
predicted=1220.219162, expected=1205.349976
predicted=1205.931583, expected=1241.300049
predicted=1239.117106, expected=1243.719971
predicted=1240.207054, expected=1254.000000
predicted=1252.304866, expected=1265.329956
predicted=1263.067196, expected=1265.430054
predicted=1263.149589, expected=1249.640015
predicted=1251.890629, expected=1263.020020
predicted=1260.493775, expected=1257.599976
predicted=1257.727411, expected=1277.060059
predicted=1274.560510, expected=1277.300049
predicted=1276.458630, expected=1281.060059
predicted=1280.241433, expected=1277.810059
predicted=1278.345077, expected=1280.699951
predicted=1278.686009, expected=1292.079956
predicted=1290.658622, expected=1292.479980
predicted=1290.652349, expected=1295.500000
predicted=1295.632767, expected=1289.089966
predicted=1289.282067, expected=1293.670044
predicted=1292.961588, expected=1308.040039
predicted=1305.321781, expected=1314.500000
predicted=1312.108460, expected=1315.380005
```

ARIMA forcasting



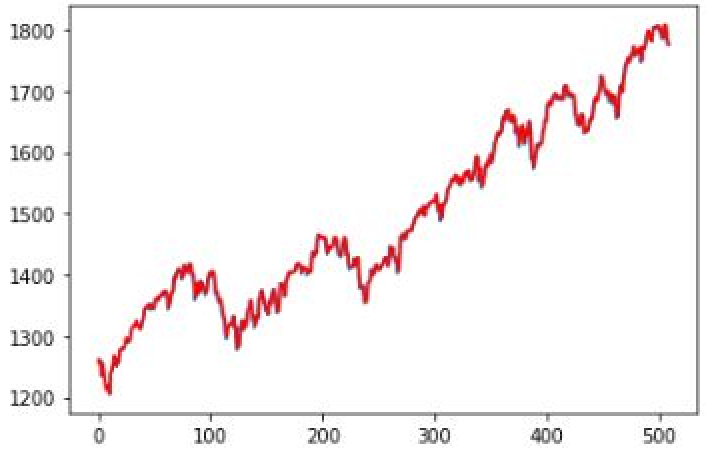
```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  990
Model:                     ARMA(9, 0)   Log Likelihood               -4304.444
Method:                       css-mle   S.D. of innovations             18.665
Date:                Wed, 15 Apr 2020   AIC                           8630.887
Time:                        06:35:22   BIC                           8684.762
Sample:                             0   HQIC                          8651.374

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       1213.6978    113.115     10.730      0.000     991.997    1435.399
ar.L1.y        0.8682      0.032     27.362      0.000       0.806       0.930
ar.L2.y        0.0570      0.042      1.356      0.175      -0.025       0.139
ar.L3.y        0.0745      0.042      1.767      0.078      -0.008       0.157
ar.L4.y       -0.0045      0.042     -0.107      0.915      -0.087       0.078
ar.L5.y       -0.0412      0.042     -0.974      0.330      -0.124       0.042
ar.L6.y        0.0477      0.042      1.127      0.260      -0.035       0.131
ar.L7.y       -0.0296      0.042     -0.700      0.484      -0.113       0.053
ar.L8.y        0.0646      0.042      1.527      0.127      -0.018       0.148
ar.L9.y       -0.0412      0.032     -1.287      0.198      -0.104       0.022
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.3704           -0.0000j            1.3704           -0.5000
AR.2           -0.9471           -1.0833j            1.4389           -0.3643
AR.3           -0.9471           +1.0833j            1.4389            0.3643
AR.4           -0.0276           -1.4772j            1.4774           -0.2530
AR.5           -0.0276           +1.4772j            1.4774            0.2530
AR.6            1.0036           -0.0000j            1.0036           -0.0000
AR.7            1.0978           -1.0517j            1.5203           -0.1216
AR.8            1.0978           +1.0517j            1.5203            0.1216
AR.9            1.6898           -0.0000j            1.6898           -0.0000
-----------------------------------------------------------------------------
```

ARIMA Model Result



```
Train on 484 samples, validate on 26 samples
Epoch 1/500
484/484 [==============================] - 0s 531us/step - loss: 0.0019 - val_loss: 5.4290e-04
Epoch 2/500
484/484 [==============================] - 0s 23us/step - loss: 5.9302e-04 - val_loss: 2.1048e-04
Epoch 3/500
484/484 [==============================] - 0s 11us/step - loss: 6.4867e-04 - val_loss: 8.2637e-05
Epoch 4/500
484/484 [==============================] - 0s 8us/step - loss: 7.1582e-04 - val_loss: 6.0902e-05
Epoch 5/500
484/484 [==============================] - 0s 8us/step - loss: 4.9307e-04 - val_loss: 6.3400e-05
Epoch 6/500
484/484 [==============================] - 0s 8us/step - loss: 2.2638e-04 - val_loss: 6.6566e-05
Epoch 7/500
484/484 [==============================] - 0s 8us/step - loss: 1.0705e-04 - val_loss: 6.8417e-05
Epoch 8/500
484/484 [==============================] - 0s 8us/step - loss: 1.6526e-04 - val_loss: 7.2196e-05
Epoch 9/500
484/484 [==============================] - 0s 8us/step - loss: 2.8363e-04 - val_loss: 8.1447e-05
Epoch 10/500
484/484 [==============================] - 0s 9us/step - loss: 3.3055e-04 - val_loss: 9.8065e-05
Epoch 11/500
484/484 [==============================] - 0s 8us/step - loss: 2.9108e-04 - val_loss: 1.2025e-04
Epoch 12/500
484/484 [==============================] - 0s 9us/step - loss: 2.2188e-04 - val_loss: 1.4127e-04
Epoch 13/500
484/484 [==============================] - 0s 7us/step - loss: 1.7336e-04 - val_loss: 1.5235e-04
Epoch 14/500
484/484 [==============================] - 0s 9us/step - loss: 1.5869e-04 - val_loss: 1.4761e-04
Epoch 15/500
484/484 [==============================] - 0s 10us/step - loss: 1.6152e-04 - val_loss: 1.2776e-04
Epoch 16/500
484/484 [==============================] - 0s 8us/step - loss: 1.5899e-04 - val_loss: 1.0049e-04
```

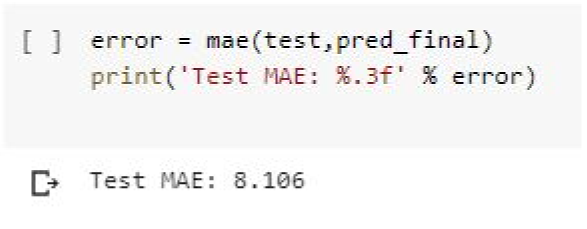
ARIMA-MLP hybrid model training and validation



ARIMA-MLP prediction vs actual close value

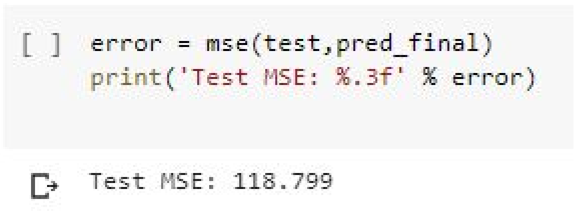
(red = predictions, blue=actual close value)

MAE (Mean Absolute Error)



```
[ ]  error = mae(test,pred_final)
     print('Test MAE: %.3f' % error)

     Test MAE: 8.106
```

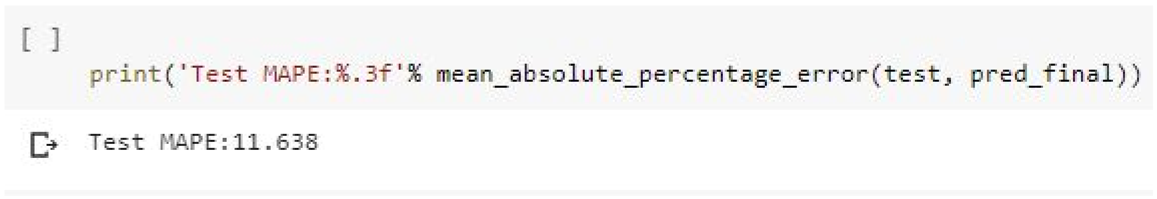
MSE (Mean Square Error)



```
[ ]  error = mse(test,pred_final)
     print('Test MSE: %.3f' % error)
```

```
Test MSE: 118.799
```

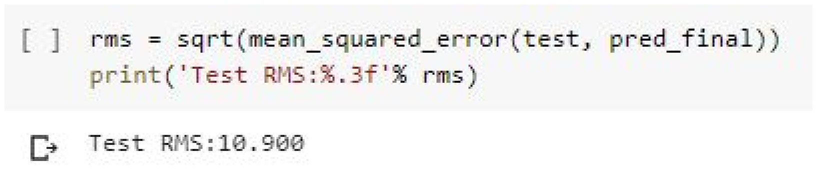
MAPE (Mean Absolute Percentage Error)



```
[ ]
     print('Test MAPE:%.3f'% mean_absolute_percentage_error(test, pred_final))
```

```
Test MAPE:11.638
```

RMSE (Root Mean Square Error)



```
[ ]  rms = sqrt(mean_squared_error(test, pred_final))
     print('Test RMS:%.3f'% rms)
```

```
Test RMS:10.900
```

Comparison of forecasting results

In this section, the predictive capabilities of hybrid models are compared together and with either of their components multilayer perceptrons and autoregressive integrated moving average− in three above mentioned data sets. Four performance indicators including mean absolute error (MAE), mean square error (MSE), mean absolute percentage error (MAPE), and root mean square error (RMSE), which are computed from the following equations, are employed in order to compare forecasting performance of hybrid models and their components.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |e_i|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (e_i)^2$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N} \frac{|e_i|}{|y_i|}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (e_i)^2}$$

	ARIMA-LSTM	ARIMA-MLP
MAE	9.287	8.106
MSE	153.728	118.799
MAPE	57.4	11.638
RMSE	12.399	10.900

Comparison between ARIMA-LSTM and ARIMA-MLP hybrid model

Conclusion

Almost all financial decision makers, i.e. investors, money managers, investment banks, hedge funds, etc. need to predict prices of financial assets such as stocks, bonds, options, interest rates, exchange rates, etc. in order to make accurate financial decisions. It is the main reason that why efforts for improving the efficiency of forecasting models has never been stopped in the finance. However, literature indicates that achieving accurate financial forecasts is an important yet often difficult task facing financial decision makers. Combining different models together is one of the most accepted and widely used methods in the literature for improving the forecasting accuracy. In the literature, several different combination techniques have been developed in order to overcome the deficiencies of single models and yield results that are more accurate. Hybrid techniques that decompose a time series into its linear and non- linear components are one of the most popular hybrid models, which have been theoretically and practically shown to be successful for single models.These models are jointly used unique advantages of linear and nonlinear models in order to capture different forms of relationship in the time series data.

In this project, the predictive capabilities of two hybrid linear/nonlinear models which are ARIMA-LSTM and ARIMA-MLP are compared together and with their components. Empir- ical results with SPDR S&P 500 ETF Trust (SPY) data sets of stock prices indicate that using these hybrid models can be a worthy idea in order to yield more accurate results than all three com- ponents used separately. In general, it can be theoretically demonstrated that hybrid models can obtain results that at least is better than one of the component models. We can conclude that the ARIMA-MLP performs very well as compare to ARIMA-LSTM as we know that every error value of the ARIMA-LSTM is more than ARIMA-MLP hybrid model.

Acknowledgements

I would like to express my special thanks of gratitude to my teacher Dr. Biju R. Mohan as well as Nagraj sir who guide us while our project work on the topic “A comparative study of ARIMA-LSTM and ARIMA-MLP hybrid models for stock price forecasting”, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

Secondly i would also like to thank my friends who helped me a lot in finalizing this project within the limited time frame.

References

Time series forecasting using a hybrid ARIMA and LSTM model, https://velvetconsulting.com/wp-content/uploads/2019/03/Seasonality_modeling_using_ARIMA_LSTM_Hybrid_Model.pdf

How to Create an ARIMA Model for Time Series Forecasting in Python, https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

LSTM using python, https://www.youtube.com/watch?v=zwqwlR48ztQ

How To Build Multi-Layer Perceptron Neural Network Models with Keras, https://machinelearningmastery.com/build-multi-layer-perceptron-neural-network-models-keras/

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras, https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

A comparative study of series arima/mlp hybrid models for stock price forecasting, https://www.tandfonline.com/loi/lssp20

HYBRID-ARIMA-LSTM-Model, Christopher Naimo posted Sep 24, 2019