SUBMITTED BY- MAZIZ YAMIN RAHMAN(102008721)

COS30018 - Option B - Task 6: Machine Learning 3

In this task, we undertake the process to develop an ensemble modelling approach for predicting the stock prices.

UNDERSTANDING THE ARIMA MODEL CONCEPT

ARIMA means Auto Regressive Integrated Moving Average model. It is capable of capturing a suite of different standard temporal structures in time-series data. The parameters for this model are;

p = number of lag observations
d = degree of differencing
q = size/width of the moving average window.

During this report we will trial out different hyperparameter configurations for this model and document our findings.

IMPORTS

The additional dependencies needed are as follows.

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean_squared_error

MODEL CREATION

We must need to train and test data. We are predicting the closing stock prices for Facebook. We will import the data from Yahoo finance which can be separated in training and testing it.

1. *We are going to take 70% of the data to train. We will take 30% to test.*
```
train_data, test_data = data[0:int(len(data)*0.7)], data[int(len(data)*0.7):]

  training_data = train_data['Close'].values
  test_data = test_data['Close'].values
```

2. *We will need some additional variables*
```
  model_predictions = []
  history = [x for x in training_data]
  N_test_observations = len(test_data)
```

3. *Building the model*
```
for time_point in range(N_test_observations):
    model = ARIMA(history, order=(4, 1, 0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    model_predictions.append(yhat)
    true_test_value = test_data[time_point]
    history.append(true_test_value)
```

4. *When the model is created, we will predict the values for each test set*

```
elif i == n_layers - 1:
            # last layer
            if bidirectional:
               model.add(Bidirectional(cell(units, return_sequences=False)))
            else:
               model.add(cell(units, return_sequences=False))
```

5. *Pay attention to the parameter configuration used here which is (4, 1, 0) for the (p,d,q)*

6. *After we predict, we will calculate the mean squared error for the actual vs the predicted price*

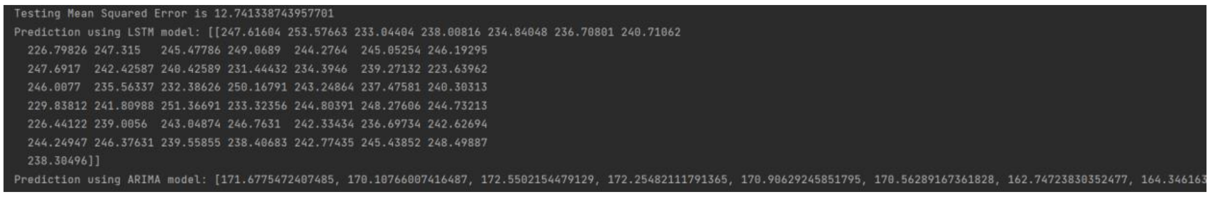
```
MSE_error = mean_squared_error(test_data, model_predictions)
  print('Testing Mean Squared Error is {}'.format(MSE_error))
```

PLOTTING THE PREDICTION

We can plot our predicted values

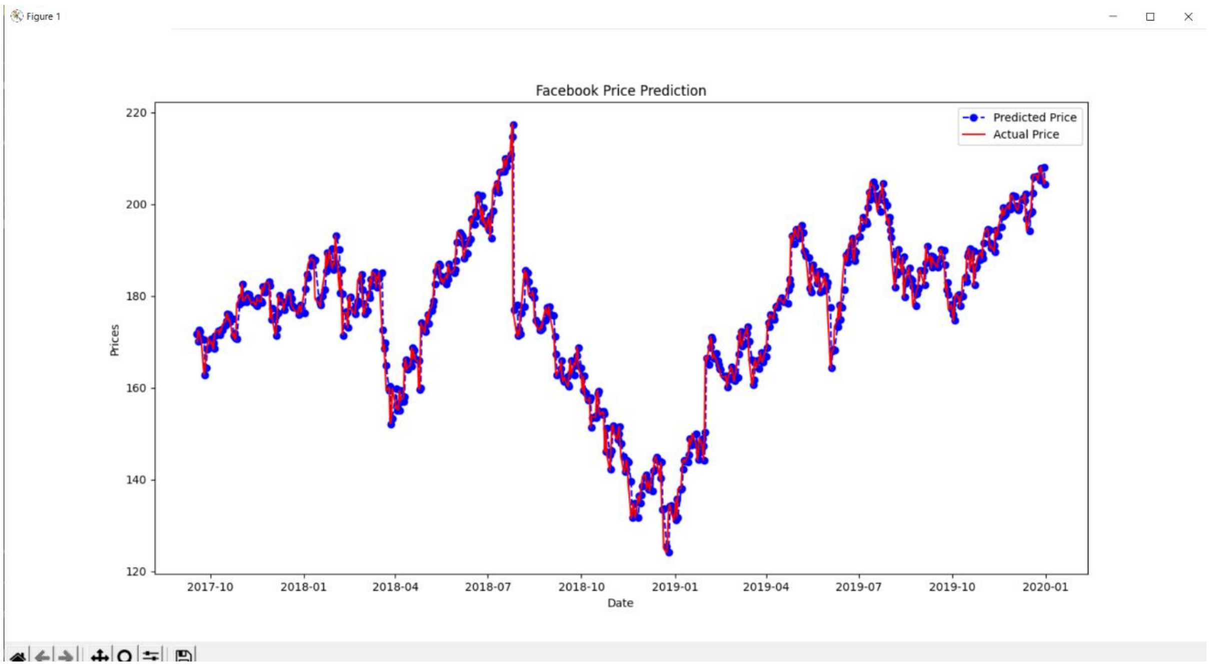
```
test_set_range = data[int(len(data)*0.7):].index


  plt.plot(test_set_range, model_predictions, color='blue', marker='o',

         linestyle='dashed', label='Predicted Price')


  plt.plot(test_set_range, test_data, color='red', label='Actual Price')


  plt.title('Facebook Price Prediction')

  plt.xlabel('Date')

  plt.ylabel('Prices')

  plt.legend()

  plt.show()
```

OUTPUT:

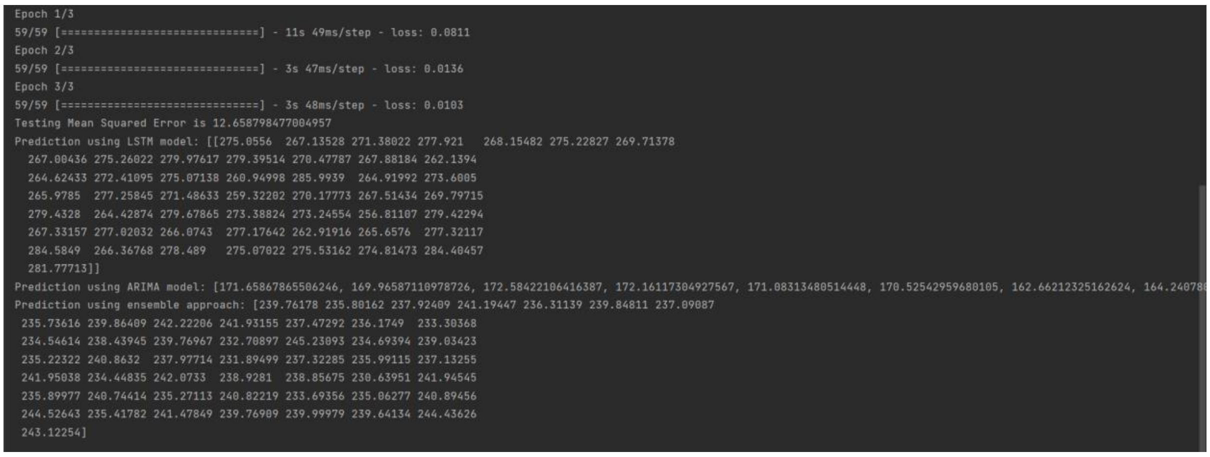


```
Testing Mean Squared Error is 12.741338743957701
Prediction using LSTM model: [[247.61604 253.57663 233.04404 238.00816 234.84048 236.70801 240.71062
  226.79826 247.315   245.47786 249.0689  244.2764  245.05254 246.19295
  247.6917  242.42587 240.42589 231.44432 234.3946  239.27132 223.63962
  246.0077  235.56337 232.38626 250.16791 243.24864 237.47581 240.30313
  229.83812 241.80988 251.36691 233.32356 244.80391 248.27606 244.73213
  226.44122 239.0056  243.04874 246.7631  242.33434 236.69734 242.62694
  244.24947 246.37631 239.55855 238.40683 242.77435 245.43852 248.49887
  238.30496]]
Prediction using ARIMA model: [171.6775472407485, 170.10766007416487, 172.5502154479129, 172.25482111791365, 170.90629245851795, 170.56289167361828, 162.74723830352477, 164.346163
```

OUTPUT



PREDICTION USING ARIMA



```
Epoch 1/3
59/59 [==============================] - 11s 49ms/step - loss: 0.0811
Epoch 2/3
59/59 [==============================] - 3s 47ms/step - loss: 0.0136
Epoch 3/3
59/59 [==============================] - 3s 48ms/step - loss: 0.0103
Testing Mean Squared Error is 12.658798477004957
Prediction using LSTM model: [[275.0556  267.13528 271.38022 277.921   268.15482 275.22827 269.71378
  267.00436 275.26022 279.97617 279.39514 270.47787 267.88184 262.1394
  264.62433 272.41095 275.07138 260.94998 285.9939  264.91992 273.6005
  265.9785  277.25845 271.48633 259.32202 270.17773 267.51434 269.79715
  279.4328  264.42874 279.67865 273.38824 273.24554 256.81107 279.42294
  267.33157 277.02032 266.0743  277.17642 262.91916 265.6576  277.32117
  284.5849  266.36768 278.489   275.07022 275.53162 274.81473 284.40457
  281.77713]]
Prediction using ARIMA model: [171.65867865506246, 169.96587110978726, 172.58422106416387, 172.16117304927567, 171.08313480514448, 170.52542959680105, 162.66212325162624, 164.24078
Prediction using ensemble approach: [239.76178 235.80162 237.92409 241.19447 236.31139 239.84811 237.09087
 235.73616 239.86409 242.22206 241.93155 237.47292 236.1749  233.30368
 234.54614 238.43945 239.76967 232.70897 245.23093 234.69394 239.03423
 235.22322 240.8632  237.97714 231.89499 237.32285 235.99115 237.13255
 241.95038 234.44835 242.0733  238.9281  238.85675 230.63951 241.94545
 235.89977 240.74414 235.27113 240.82219 233.69356 235.06277 240.89456
 244.52643 235.41782 241.47849 239.76909 239.99979 239.64134 244.43626
 243.12254]
```

ENSEMBLE PREDICTION