Model_building_code

January 30, 2022

0.0.1 Data Preprocessing

Reading the data

```
[ ]: import pandas as pd
     ibm = pd.read_csv("daily_IBM.csv")
     print(ibm.head())
```

```
    timestamp    open      high      low   close    volume
0  2022-01-28  133.19  134.5300  131.790  134.50   5471497
1  2022-01-27  133.66  134.7500  132.080  132.52   5499566
2  2022-01-26  136.47  137.0700  133.130  134.26   8335992
3  2022-01-25  129.14  137.3361  128.300  136.10  19715698
4  2022-01-24  127.99  129.1500  124.193  128.82  13777648
```

Manipulating the Data

```
[ ]: ibm.fillna(0,inplace=True)

     ibm.drop(['open','high','low','volume'],axis=1,inplace=True)

     ibm
```

```
[ ]:        timestamp   close
     0     2022-01-28  134.50
     1     2022-01-27  132.52
     2     2022-01-26  134.26
     3     2022-01-25  136.10
     4     2022-01-24  128.82
     ...          ...     ...
     5593  1999-11-05   90.25
     5594  1999-11-04   91.56
     5595  1999-11-03   94.37
     5596  1999-11-02   94.81
     5597  1999-11-01   96.75

     [5598 rows x 2 columns]
```

[ ]:
```
ibm.index = pd.to_datetime(ibm['timestamp'], format='%Y-%m-%d')
ibm_mod = ibm.drop(['timestamp'],axis=1,inplace=False)
ibm_mod
```

[ ]:
```
             close
timestamp
2022-01-28  134.50
2022-01-27  132.52
2022-01-26  134.26
2022-01-25  136.10
2022-01-24  128.82
...            ...
1999-11-05   90.25
1999-11-04   91.56
1999-11-03   94.37
1999-11-02   94.81
1999-11-01   96.75

[5598 rows x 1 columns]
```

[ ]:
```
```

0.0.2 Data Visualizing

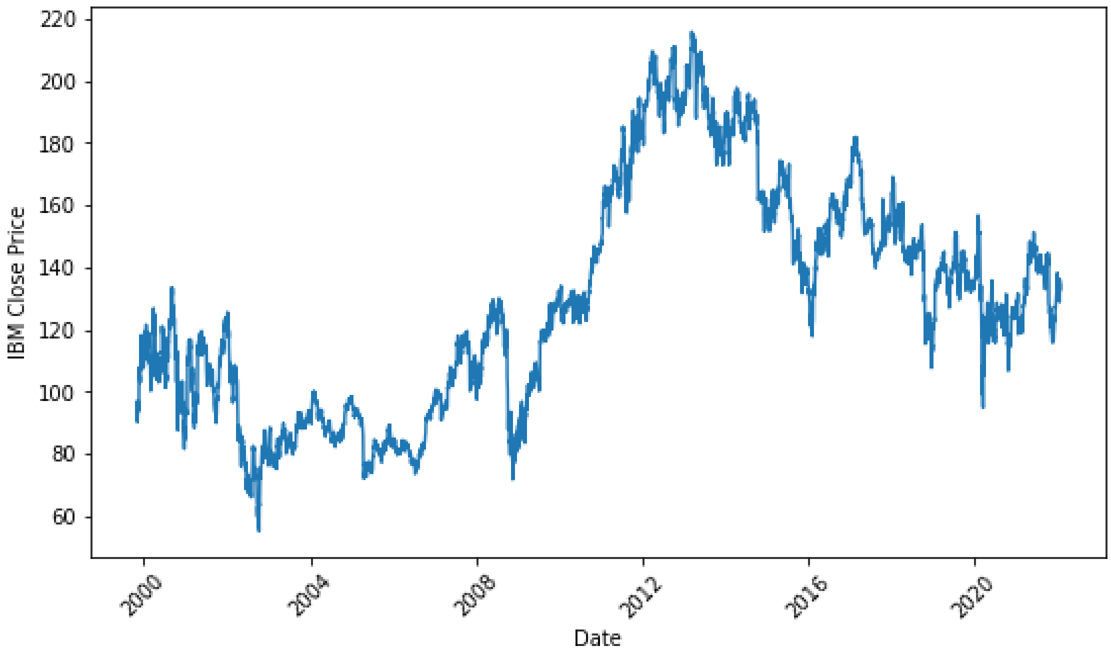
A few graphs below showcase what our data looks like

[ ]:
```
#@title
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
matplotlib.rcParams['font.size'] = 10
matplotlib.rcParams['figure.figsize'] = (9, 5)
```

[ ]:
```
#@title
plt.ylabel('IBM Close Price')
plt.xlabel('Date')
plt.xticks(rotation=45)

plt.plot(ibm_mod.index, ibm_mod['close'], );
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f963dfabdd0>]
```

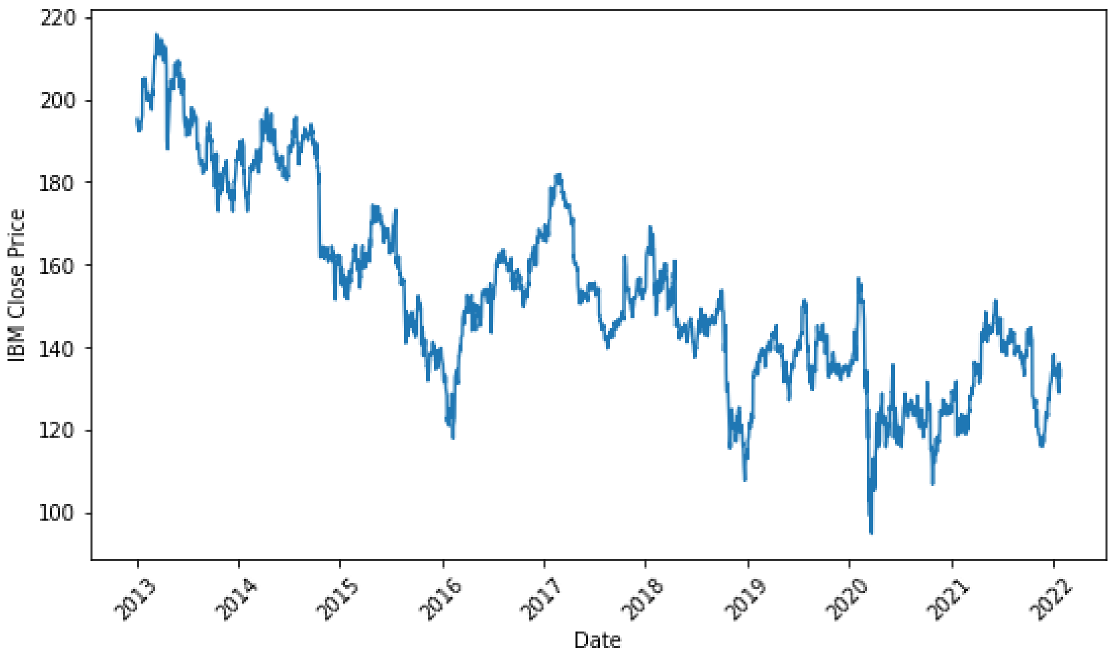



```
[ ]: truncated = ibm_mod[ibm_mod.index > pd.to_datetime("2013-01-02",␣
      ↪format='%Y-%m-%d')]

     plt.ylabel('IBM Close Price')
     plt.xlabel('Date')
     plt.xticks(rotation=45)

     plt.plot(truncated.index, truncated['close'], )
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f963df01d50>]
```



0.0.3 Splitting DataSet in Training and Test sets

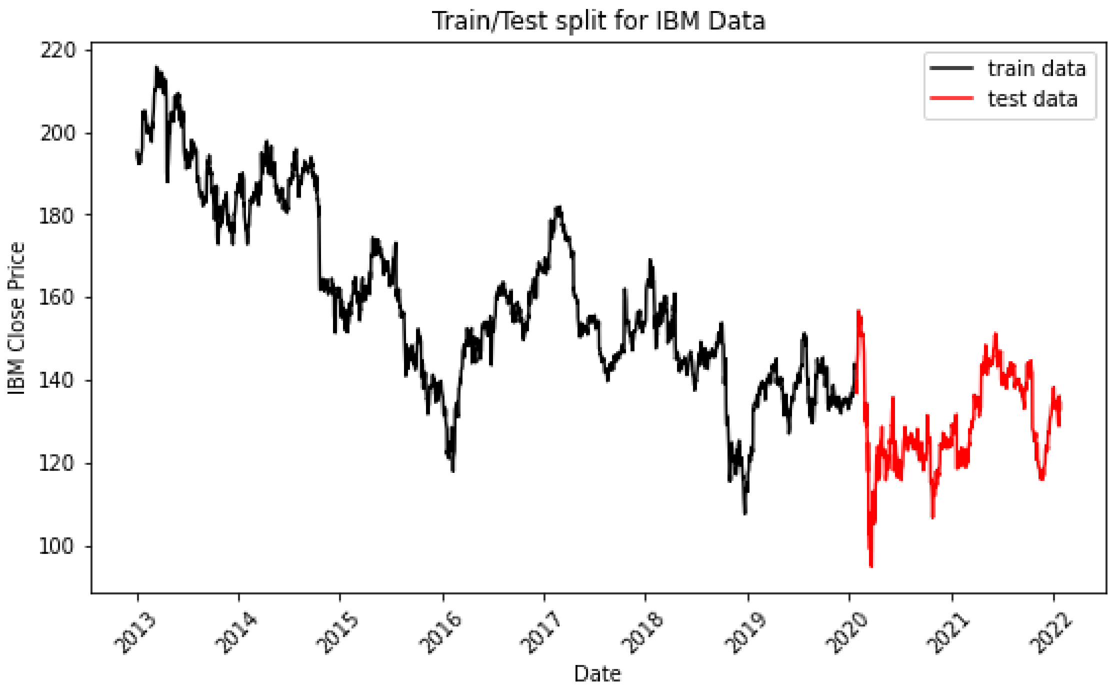
[ ]:
```
train = truncated[truncated.index < pd.to_datetime("2020-01-24",␣
 ↪format='%Y-%m-%d')]
test = truncated[truncated.index > pd.to_datetime("2020-01-24",␣
 ↪format='%Y-%m-%d')]

plt.plot(train, color = "black")
plt.plot(test, color = "red")

plt.ylabel('IBM Close Price')
plt.xlabel('Date')
plt.xticks(rotation=45)

plt.title("Train/Test split for IBM Data")
plt.legend(['train data','test data'])
```

[ ]:
```
<matplotlib.legend.Legend at 0x7f962ffc8c10>
```



[ ]:

```

```

0.0.4 Training the Model

[ ]:

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

y = train['close']
```

[ ]:

```
from statsmodels.tsa.arima_model import ARIMA
```

[ ]:

```
training_data = train['close'].values
test_data = test['close'].values

history = [x for x in training_data]
model_predictions = []
N_test_observations = len(test_data)

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

```
    history.append(true_test_value)
```

Making predictions

```
[ ]: print(model_predictions)
```

```
[array([195.27540765]), array([134.85300045]), array([134.40600084]),
array([133.55147886]), array([137.9442154]), array([128.66946873]),
array([129.6050535]), array([130.85083478]), array([131.37127538]),
array([132.92683585]), array([134.19927401]), array([134.73316542]),
array([133.58773508]), array([132.92276189]), array([135.07329226]),
array([134.73231377]), array([135.32705764]), array([138.26224958]),
array([137.92060542]), array([136.0399827]), array([133.76373213]),
array([133.98158166]), array([133.29555135]), array([132.5882252]),
array([131.63651283]), array([130.63835281]), array([129.75461277]),
array([128.96494924]), array([127.04319673]), array([127.43584933]),
array([125.88535445]), array([123.08505741]), array([123.84999409]),
array([122.50945545]), array([124.06574135]), array([123.52365455]),
array([122.99603521]), array([121.60887959]), array([119.92021168]),
array([118.85994184]), array([116.87447632]), array([116.92951851]),
array([117.06402351]), array([118.4496431]), array([115.73505502]),
array([116.80641688]), array([116.77968703]), array([116.39310536]),
array([116.062285]), array([116.6636654]), array([118.02991852]),
array([118.39707238]), array([118.8599276]), array([118.96526686]),
array([120.271958]), array([120.17646338]), array([120.8449906]),
array([124.56437207]), array([123.4805367]), array([120.86232488]),
array([127.32079715]), array([125.94273632]), array([126.25378495]),
array([125.21946671]), array([125.84944711]), array([125.13526198]),
array([127.1243319]), array([127.58897703]), array([127.84411906]),
array([128.36119045]), array([141.96381754]), array([141.5724353]),
array([142.35547033]), array([144.9529806]), array([143.32002975]),
array([140.8011411]), array([140.59404376]), array([142.40836646]),
array([143.10144776]), array([141.77925674]), array([142.45565124]),
array([143.1497748]), array([144.05976217]), array([143.30307834]),
array([138.95189736]), array([139.33656545]), array([137.41449619]),
array([138.51296039]), array([137.44881437]), array([136.7185993]),
array([134.66074378]), array([132.99055877]), array([134.33727972]),
array([135.13421235]), array([136.37002876]), array([137.19930081]),
array([136.2130197]), array([138.21811959]), array([136.96780226]),
array([137.75789874]), array([138.69210536]), array([138.00248343]),
array([139.62391632]), array([139.98425876]), array([139.27301063]),
array([140.40170141]), array([138.93744679]), array([139.43884838]),
array([138.78067515]), array([139.85211091]), array([139.81400481]),
array([139.60710462]), array([139.137062]), array([138.02642112]),
array([139.50052688]), array([142.37504639]), array([143.48745036]),
array([143.1850412]), array([143.15279745]), array([142.153239]),
array([141.39407264]), array([141.26575464]), array([144.0861556]),
array([142.66027142]), array([142.80493995]), array([144.13706491]),
```

array([141.33851874]), array([141.04195469]), array([141.9671958]),
array([141.6794061]), array([142.75285841]), array([142.76041038]),
array([141.33216554]), array([140.7709723]), array([141.31663175]),
array([139.91121509]), array([137.93714085]), array([138.97345897]),
array([140.38890724]), array([139.72602319]), array([140.32772745]),
array([140.93887324]), array([141.49018369]), array([140.72909729]),
array([139.85477052]), array([138.8117062]), array([140.0364052]),
array([146.81656531]), array([146.35420458]), array([145.60169029]),
array([145.46895914]), array([146.85020916]), array([145.3598026]),
array([144.65113595]), array([146.42800681]), array([146.56198419]),
array([143.07985368]), array([145.77025219]), array([147.75160524]),
array([149.22363828]), array([150.05018666]), array([151.32562177]),
array([150.53586014]), array([150.71932503]), array([149.08649223]),
array([148.05408595]), array([147.45109731]), array([145.52072736]),
array([145.75800133]), array([144.15583297]), array([143.74616849]),
array([143.83332729]), array([143.33859966]), array([143.79719372]),
array([144.71258552]), array([144.70232537]), array([143.8882678]),
array([143.23574967]), array([143.93413594]), array([145.07181557]),
array([144.62616361]), array([144.20314848]), array([141.32540986]),
array([144.31805994]), array([146.0628942]), array([145.33682611]),
array([148.53429968]), array([145.14889694]), array([145.85440442]),
array([144.78219861]), array([141.83626498]), array([144.349856]),
array([142.88645399]), array([142.00056247]), array([141.64176161]),
array([142.41761977]), array([141.2250398]), array([143.5898552]),
array([138.08711612]), array([133.2580794]), array([133.77408058]),
array([132.42241461]), array([132.56161422]), array([131.17068162]),
array([134.62529641]), array([135.61399366]), array([135.06358984]),
array([135.01982404]), array([134.23888275]), array([135.94327081]),
array([133.15271481]), array([133.3366471]), array([134.74144508]),
array([135.75765853]), array([136.34868214]), array([133.07044047]),
array([130.73817731]), array([130.53146068]), array([130.47383868]),
array([128.83224972]), array([130.1101836]), array([128.9753989]),
array([128.23532708]), array([128.6190585]), array([127.56304974]),
array([127.14795103]), array([127.88334338]), array([124.11090489]),
array([124.9188064]), array([122.78998584]), array([120.08266848]),
array([122.4473161]), array([120.18613619]), array([120.75770463]),
array([118.93181641]), array([122.48668027]), array([123.08866018]),
array([120.65213223]), array([121.0042222]), array([118.97002165]),
array([120.73519209]), array([119.89546132]), array([120.05526094]),
array([120.81684773]), array([120.86180427]), array([122.23421228]),
array([122.06150258]), array([123.61629772]), array([121.75370527]),
array([121.0701066]), array([119.1498024]), array([119.45041224]),
array([120.49787401]), array([119.02606661]), array([120.12651813]),
array([122.44714604]), array([122.38118821]), array([118.59010624]),
array([118.77243786]), array([131.6492925]), array([129.50514069]),
array([129.13791]), array([128.70499203]), array([128.94712332]),
array([126.86357424]), array([129.26685466]), array([128.49293686]),
array([128.5094391]), array([129.03361256]), array([129.25163843]),

```
array([126.12327165]), array([124.05987449]), array([125.94770628]),
array([124.17730482]), array([123.80754382]), array([124.86731011]),
array([124.60523059]), array([123.88878288]), array([123.65304191]),
array([123.38467522]), array([125.82910866]), array([125.44115626]),
array([125.55942722]), array([125.97577365]), array([123.50134173]),
array([124.35264026]), array([124.91929501]), array([126.70464661]),
array([125.65305151]), array([124.75637158]), array([127.26425493]),
array([123.48363787]), array([124.73431919]), array([123.15467901]),
array([123.4892059]), array([124.34093619]), array([124.13015586]),
array([124.42767655]), array([120.09863668]), array([117.09045529]),
array([117.27046018]), array([116.64208803]), array([117.63662758]),
array([118.31585981]), array([116.81030621]), array([114.56736718]),
array([117.27927355]), array([117.75289008]), array([115.45566352]),
array([114.18085952]), array([114.81786776]), array([111.80287575]),
array([114.22243013]), array([112.81568757]), array([111.64520344]),
array([108.99019267]), array([106.70620432]), array([110.58484648]),
array([111.98547833]), array([115.88749786]), array([115.68869861]),
array([115.09997677]), array([117.45597065]), array([125.41854657]),
array([125.60551589]), array([124.9445604]), array([126.16000712]),
array([125.05575614]), array([127.21607298]), array([127.72254513]),
array([131.45375863]), array([123.98398421]), array([122.26718997]),
array([122.14866124]), array([120.38778559]), array([121.08858361]),
array([121.63638462]), array([120.87822795]), array([121.76015147]),
array([118.92099009]), array([118.17097581]), array([118.8593123]),
array([120.40839826]), array([120.15856311]), array([122.77913776]),
array([124.85102385]), array([124.13131647]), array([122.51686672]),
array([122.19235534]), array([121.44360166]), array([120.53483514]),
array([122.27531288]), array([121.11923454]), array([122.3156539]),
array([124.43101957]), array([128.07392736]), array([123.27964786]),
array([123.54095917]), array([125.13160895]), array([124.46845891]),
array([124.18142057]), array([124.68689804]), array([125.64154476]),
array([123.10362241]), array([123.25379335]), array([123.84554417]),
array([124.82691438]), array([124.39085611]), array([125.29815172]),
array([125.01166093]), array([126.69119922]), array([126.69437227]),
array([127.09951913]), array([124.97806905]), array([126.19684577]),
array([125.40011918]), array([125.81169033]), array([124.31376952]),
array([122.978317]), array([122.9494576]), array([125.27157149]),
array([124.33602671]), array([126.23635329]), array([125.76832113]),
array([127.31964465]), array([128.64024777]), array([125.99919024]),
array([126.50144615]), array([125.11695838]), array([123.99492174]),
array([123.03933694]), array([120.601623]), array([119.25549585]),
array([118.36346923]), array([115.6729646]), array([117.76109059]),
array([117.46500746]), array([120.12569217]), array([119.63676684]),
array([118.55281384]), array([120.85582573]), array([119.6429866]),
array([117.20365339]), array([119.15809344]), array([116.3138137]),
array([119.43763872]), array([120.97163887]), array([122.34854558]),
array([124.16264065]), array([124.11786246]), array([125.17482076]),
array([121.64792124]), array([122.03172283]), array([118.01210289]),
```

```
array([129.84370802]), array([131.43374644]), array([135.5593759]),
array([132.20217969]), array([129.03762735]), array([129.2453557]),
array([125.95040808]), array([124.92803969]), array([124.9453621]),
array([124.4719279]), array([125.51952731]), array([121.73695833]),
array([118.50460816]), array([119.25374155]), array([121.2841625]),
array([120.14030702]), array([121.58414883]), array([116.99410571]),
array([117.06319436]), array([115.77385399]), array([120.18080353]),
array([122.43003637]), array([122.84697651]), array([121.27686398]),
array([123.25094813]), array([122.53057748]), array([121.64617963]),
array([121.9296467]), array([125.5251191]), array([128.53132343]),
array([126.14000852]), array([126.04084179]), array([124.80718477]),
array([121.352314]), array([119.42099688]), array([116.82445519]),
array([120.42870336]), array([119.97363352]), array([115.66729532]),
array([118.885494]), array([123.80098358]), array([120.85628351]),
array([121.61725218]), array([119.39298185]), array([114.97493775]),
array([114.98392303]), array([106.33803112]), array([110.17621714]),
array([105.09350572]), array([110.80902387]), array([112.83324927]),
array([107.88381868]), array([113.03801644]), array([105.84610077]),
array([105.58861476]), array([95.04143138]), array([95.49974047]),
array([100.3416356]), array([103.23040019]), array([106.45912603]),
array([99.07111555]), array([107.94992967]), array([102.85674873]),
array([117.28049635]), array([124.5937696]), array([117.53769724]),
array([127.5543641]), array([129.37049232]), array([133.96516928]),
array([129.04132576]), array([134.18790167]), array([130.32159447]),
array([132.92453038]), array([139.59988249]), array([141.49607335]),
array([146.25138269]), array([149.76473323]), array([151.15043724]),
array([150.89416924]), array([151.13100323]), array([150.73032687]),
array([154.21402176]), array([155.26268487]), array([153.49839624]),
array([154.46186451]), array([153.4567827]), array([156.65502095]),
array([156.33670672]), array([149.28475723]), array([146.46062151]),
array([143.86331853]), array([136.8811401]), array([137.69432604]),
array([139.49330788])]
```

[ ]:
```
y_pred_df = test.copy()

y_pred_df['Predictions'] = model_predictions
#y_pred_df.drop(['close'],axis=1,inplace=True)

y_pred_df
```

[ ]:

timestamp	close	Predictions
2022-01-28	134.50	[195.2754076545199]
2022-01-27	132.52	[134.85300044792874]
2022-01-26	134.26	[134.40600083506013]
2022-01-25	136.10	[133.55147886272576]
2022-01-24	128.82	[137.94421540188145]

```
...              ...                      ...
2020-01-31  143.73  [146.46062150853083]
2020-01-30  136.77  [143.86331853091127]
2020-01-29  137.69  [136.88114009678463]
2020-01-28  139.55  [137.69432603677538]
2020-01-27  138.62   [139.4933078776248]

[508 rows x 2 columns]
```

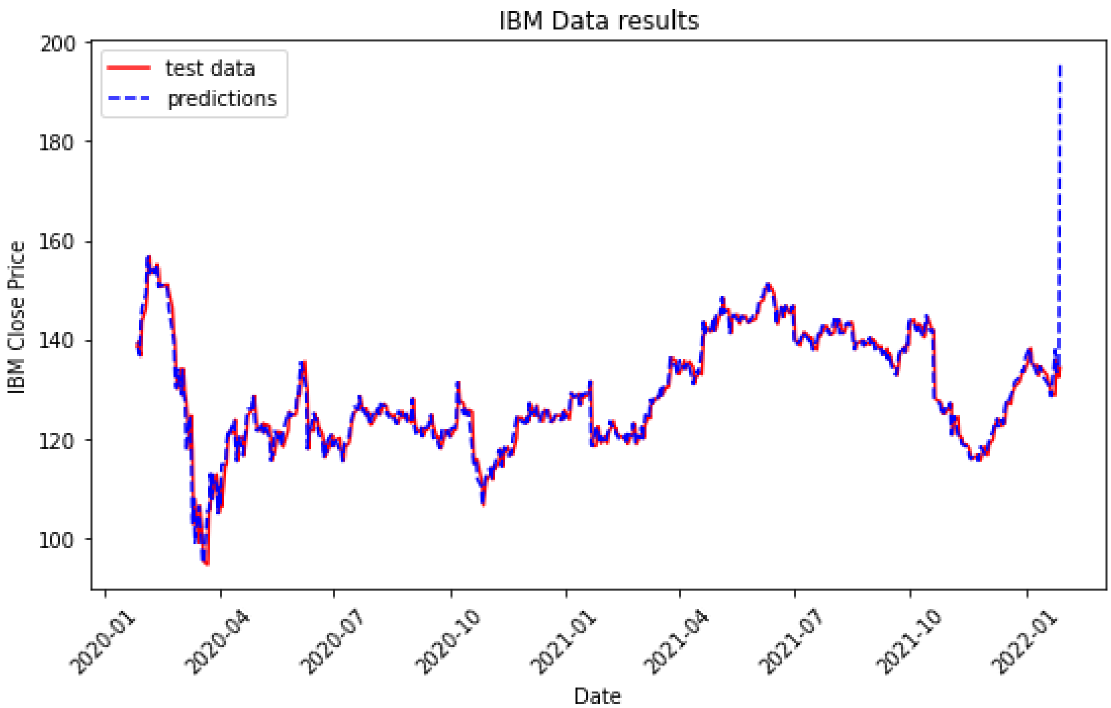
[ ]:
```
y_pred_df.drop(['close'],axis=1,inplace=True)
```

[ ]:
```
plt.plot(test, color = "red")
plt.plot(y_pred_df, color = "blue", ls = '--')

plt.ylabel('IBM Close Price')
plt.xlabel('Date')
plt.xticks(rotation=45)

plt.title("IBM Data results")
plt.legend(['test data','predictions'])
```

[ ]:
```
<matplotlib.legend.Legend at 0x7f963da06b90>
```



[ ]:

0.0.5 Using LSTM approach

[ ]:
```
import math
import matplotlib.pyplot as plt
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping
```

[ ]:
```
raw = pd.read_csv('daily_IBM.csv').loc[::-1].reset_index(drop=True)
raw.shape
```

[ ]:
```
(5598, 6)
```

[ ]:
```
raw.head()
```

[ ]:
```
    timestamp   open   high    low  close    volume
0  1999-11-01  98.50  98.81  96.37  96.75   9551800
1  1999-11-02  96.75  96.81  93.69  94.81  11105400
2  1999-11-03  95.87  95.94  93.50  94.37  10369100
3  1999-11-04  94.44  94.44  90.00  91.56  16697600
4  1999-11-05  92.75  92.94  90.19  90.25  13737600
```

[ ]:
```
raw['timestamp']
```

[ ]:
```
0       1999-11-01
1       1999-11-02
2       1999-11-03
3       1999-11-04
4       1999-11-05
           ...
5593    2022-01-24
5594    2022-01-25
5595    2022-01-26
5596    2022-01-27
5597    2022-01-28
Name: timestamp, Length: 5598, dtype: object
```

[ ]:
```
close = raw['close'].copy()
```

```
[ ]: training_set = (close.iloc[:5000]).to_numpy().reshape(-1,1)
     test_set = (close.iloc[5000:]).to_numpy().reshape(-1,1)
```

```
[ ]: # Feature Scaling
     sc = StandardScaler()
     sc.fit(training_set)
     training_set_scaled = sc.transform(training_set)
```

```
[ ]: # Creating a data structure with 14 time-steps and 1 output
     X_train = []
     y_train = []
     for i in range(14, 5000):
         X_train.append(training_set_scaled[i-14:i, 0])
         y_train.append(training_set_scaled[i, 0])

     X_train, y_train = np.array(X_train), np.array(y_train)
     X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
```

```
[ ]: X_train.shape, y_tra
```

```
[ ]: (4986, 14, 1)
```

```
[ ]: model = Sequential()

     #Adding the first LSTM layer and some Dropout regularisation
     model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.
      ↪shape[1], 1)))
     model.add(Dropout(0.2))

     # Adding a second LSTM layer and some Dropout regularisation
     model.add(LSTM(units = 50, return_sequences = True))
     model.add(Dropout(0.2))

     # Adding a third LSTM layer and some Dropout regularisation
     model.add(LSTM(units = 50, return_sequences = True))
     model.add(Dropout(0.2))

     # Adding a fourth LSTM layer and some Dropout regularisation
     model.add(LSTM(units = 50))
     model.add(Dropout(0.2))

     # Adding the output layer
     model.add(Dense(units = 1))

     # Compiling the RNN
     model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

[ ]:

```python
# Fitting the RNN to the Training set
model.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

```
Epoch 1/100
156/156 [==============================] - 12s 33ms/step - loss: 0.0793
Epoch 2/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0290
Epoch 3/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0270
Epoch 4/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0246
Epoch 5/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0251
Epoch 6/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0223
Epoch 7/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0222
Epoch 8/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0185
Epoch 9/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0176
Epoch 10/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0183
Epoch 11/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0173
Epoch 12/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0160
Epoch 13/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0150
Epoch 14/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0145
Epoch 15/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0149
Epoch 16/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0144
Epoch 17/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0143
Epoch 18/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0134
Epoch 19/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0131
Epoch 20/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0129
Epoch 21/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0134
Epoch 22/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0125
Epoch 23/100
```

```
156/156 [==============================] - 5s 33ms/step - loss: 0.0127
Epoch 24/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0115
Epoch 25/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0115
Epoch 26/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0119
Epoch 27/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0119
Epoch 28/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0117
Epoch 29/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0107
Epoch 30/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0114
Epoch 31/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0106
Epoch 32/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0124
Epoch 33/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0109
Epoch 34/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0109
Epoch 35/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0111
Epoch 36/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0110
Epoch 37/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0102
Epoch 38/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0105
Epoch 39/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0103
Epoch 40/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 41/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0104
Epoch 42/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0102
Epoch 43/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0102
Epoch 44/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 45/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 46/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 47/100
```

```
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 48/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0100
Epoch 49/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 50/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 51/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0093
Epoch 52/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0105
Epoch 53/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0099
Epoch 54/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0090
Epoch 55/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0094
Epoch 56/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0100
Epoch 57/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 58/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 59/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0097
Epoch 60/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0099
Epoch 61/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 62/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0097
Epoch 63/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0099
Epoch 64/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0099
Epoch 65/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 66/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 67/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 68/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0089
Epoch 69/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0101
Epoch 70/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0093
Epoch 71/100
```

```
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 72/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0096
Epoch 73/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0094
Epoch 74/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 75/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0093
Epoch 76/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 77/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0091
Epoch 78/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0094
Epoch 79/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0100
Epoch 80/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0098
Epoch 81/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0094
Epoch 82/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0094
Epoch 83/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0089
Epoch 84/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0105
Epoch 85/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0094
Epoch 86/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0098
Epoch 87/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0095
Epoch 88/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0094
Epoch 89/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0092
Epoch 90/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0095
Epoch 91/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0094
Epoch 92/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0097
Epoch 93/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0093
Epoch 94/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0093
Epoch 95/100
```

```
156/156 [==============================] - 5s 34ms/step - loss: 0.0094
Epoch 96/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0095
Epoch 97/100
156/156 [==============================] - 5s 33ms/step - loss: 0.0091
Epoch 98/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0089
Epoch 99/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0091
Epoch 100/100
156/156 [==============================] - 5s 34ms/step - loss: 0.0098
```

[ ]:
```
<keras.callbacks.History at 0x7ffb1325e150>
```

[ ]:
```
# Preparing test data
inputs = close[5000 - 14:].to_numpy().reshape(-1,1)

inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
```

[ ]:
```
inputs.shape
```

[ ]:
```
(612, 1)
```

[ ]:
```
X_test = []
for i in range(14, 612):
    X_test.append(inputs[i-14:i, 0])

X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

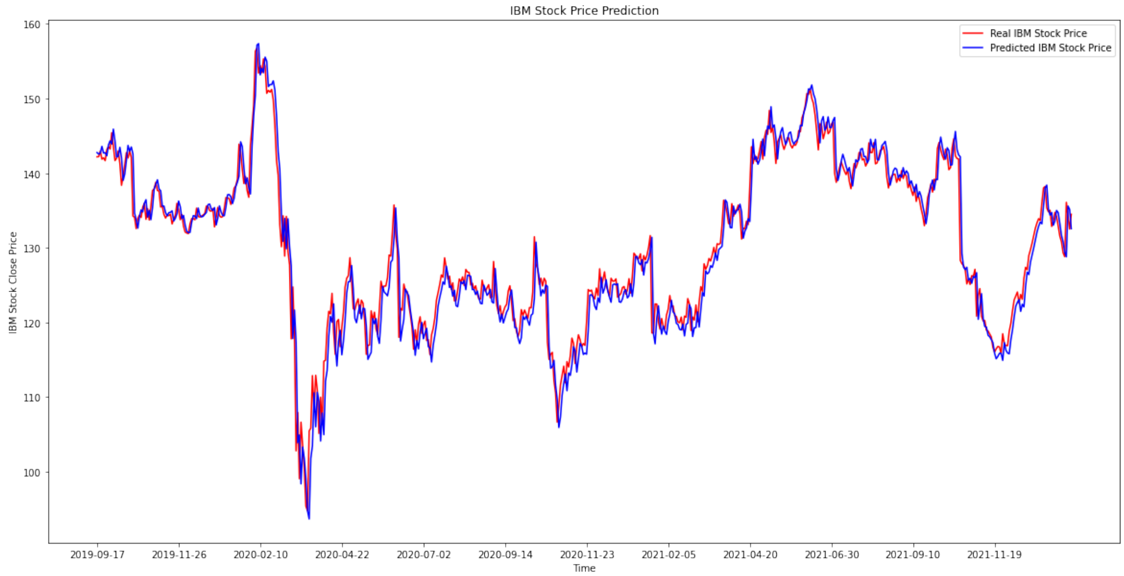
[ ]:
```
X_test.shape
```

[ ]:
```
(598, 14, 1)
```

[ ]:
```
predicted_stock_price = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

[ ]:
```
# Visualising the results
plt.figure(figsize=(20,10))
plt.plot(raw.loc[5000:,"timestamp"], test_set, 'r', label = 'Real IBM Stock␣
 ↪Price')
plt.plot(raw.loc[5000:,"timestamp"], predicted_stock_price, 'b', label =␣
 ↪'Predicted IBM Stock Price')
plt.xticks(np.arange(0,598,50))
plt.title('IBM Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('IBM Stock Close Price')
```

```
plt.legend()
plt.show()
```



[ ]:
```
raw.iloc[3500]
```

[ ]:
```
timestamp    2013-10-01
open             185.34
high             186.65
low              184.65
close            186.38
volume          2681200
Name: 3500, dtype: object
```

0.1 Trying prediction for an arbitrary date

[ ]:
```
def prep_input(input_date):
  idx = raw[raw['timestamp']==input_date].index.values[0]

  input = sc.transform(raw.loc[idx-13:idx,'close'].to_numpy().reshape(14,1))

  return input.reshape(1,14,1)
```

[ ]:
```
predicted_stock_price = model.predict(prep_input('2013-10-01'))
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

[ ]:
```
predicted_stock_price
```

[ ]:
```
array([[187.11333]], dtype=float32)
```

[ ]:
```
raw.iloc[3501]
```

[ ]:
```
timestamp    2013-10-02
open             185.54
high             186.31
low              184.41
close            184.96
volume          3617100
Name: 3501, dtype: object
```