Stock Price Prediction of Microsoft

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM,Dense,Dropout
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.dates as mdates
from sklearn import linear_model
```

Reading data using parse date

As it is timeseries data so we have to read the data by parsing date means making data column as index.

In [2]:
```
df_final = pd.read_csv("MSFT.csv",na_values=['null'],index_col='Date',parse_dates='
```

In [3]:
```
df_final.head()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
1986-03-13	0.088542	0.101563	0.088542	0.097222	0.061434	1031788800
1986-03-14	0.097222	0.102431	0.097222	0.100694	0.063628	308160000
1986-03-17	0.100694	0.103299	0.100694	0.102431	0.064725	133171200
1986-03-18	0.102431	0.103299	0.098958	0.099826	0.063079	67766400
1986-03-19	0.099826	0.100694	0.097222	0.098090	0.061982	47894400

In [4]:
```
df_final.shape
```

Out[4]:
```
(9083, 6)
```

In [5]:
```
df_final.describe()
```

Out[5]:

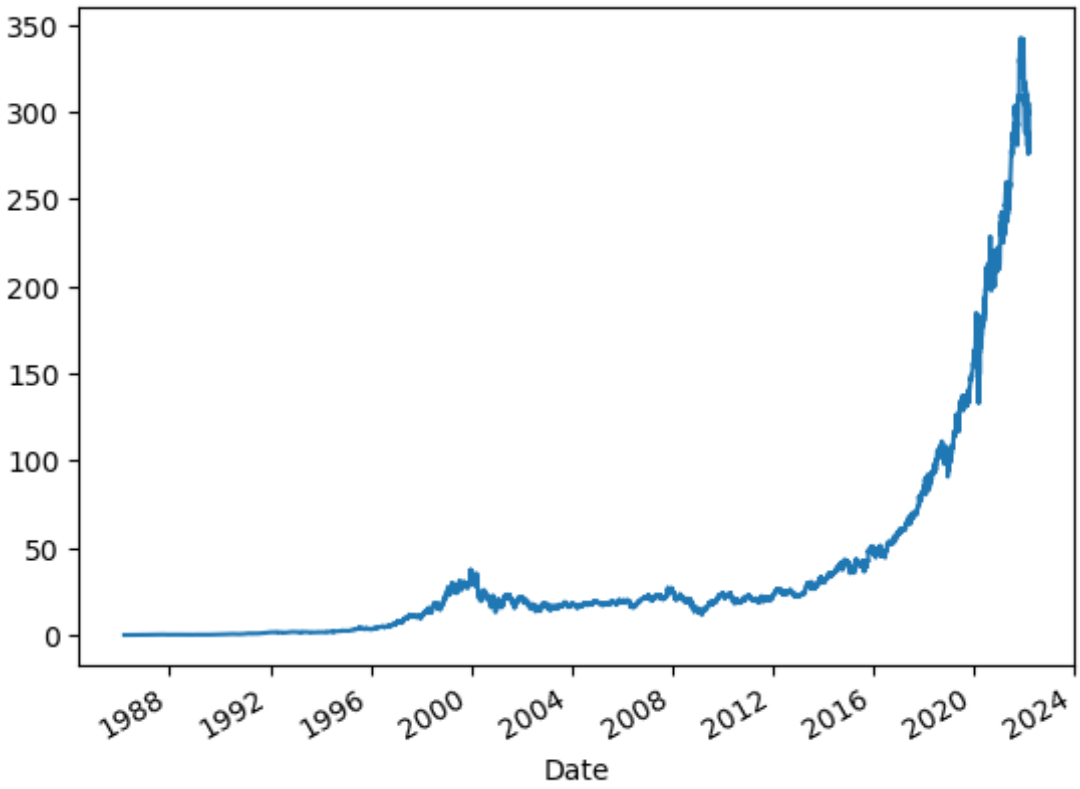
	Open	High	Low	Close	Adj Close	Volume
count	9083.000000	9083.000000	9083.000000	9083.000000	9083.000000	9.083000e+03
mean	41.324936	41.760887	40.878488	41.335628	36.256120	5.875055e+07
std	59.696905	60.272218	59.081728	59.714567	59.981436	3.845200e+07
min	0.088542	0.092014	0.088542	0.090278	0.057046	2.304000e+06
25%	4.050781	4.102051	4.027344	4.075195	2.575089	3.461230e+07
50%	26.820000	27.100000	26.520000	26.840000	18.948530	5.203200e+07
75%	40.034999	40.443751	39.500000	39.937500	29.244812	7.265400e+07
max	344.619995	349.670013	342.200012	343.109985	342.402008	1.031789e+09

In [6]:
```
df_final.isnull().values.any()
```

Out[6]:
```
False
```

In [7]:
```
df_final['Adj Close'].plot()
```

Out[7]:
```
<AxesSubplot:xlabel='Date'>
```

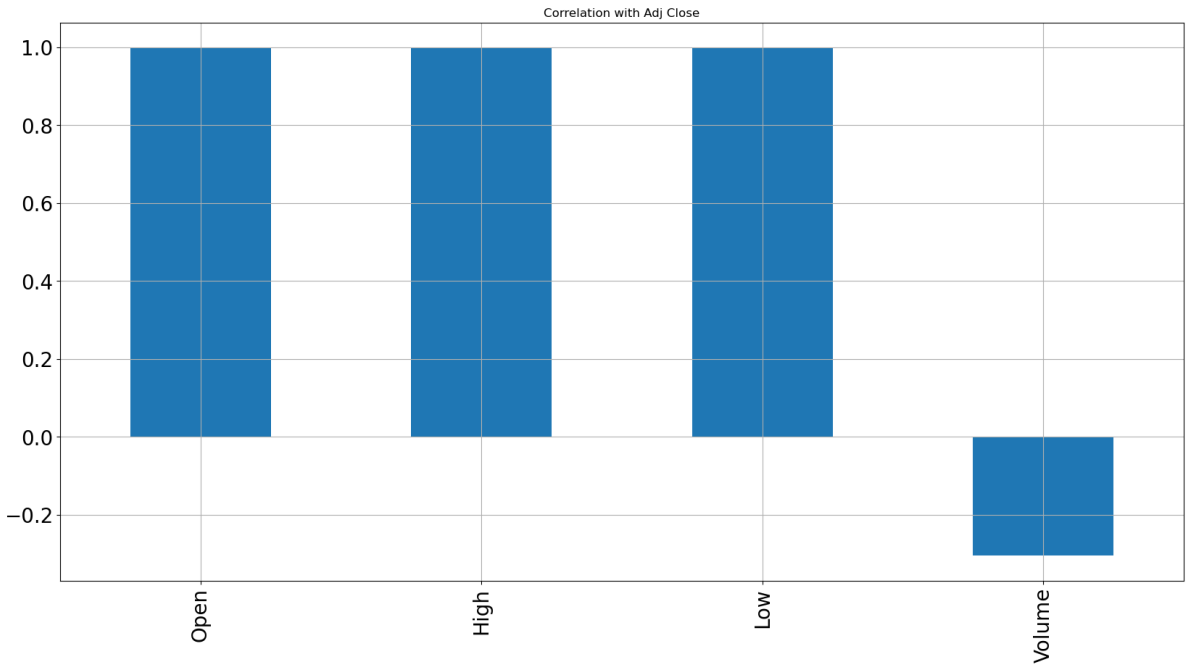


Correlation Analysis

In [8]:
```
X=df_final.drop(['Adj Close'],axis=1)
X=X.drop(['Close'],axis=1)
```

In [9]:
```
X.corrwith(df_final['Adj Close']).plot.bar(
        figsize = (20, 10), title = "Correlation with Adj Close", fontsize = 20,
        rot = 90, grid = True)
```

Out[9]: <AxesSubplot:title={'center':'Correlation with Adj Close'}>



In [10]:
```
test = df_final
# Target column
target_adj_close = pd.DataFrame(test['Adj Close'])
display(test.head())
```

Date	Open	High	Low	Close	Adj Close	Volume
1986-03-13	0.088542	0.101563	0.088542	0.097222	0.061434	1031788800
1986-03-14	0.097222	0.102431	0.097222	0.100694	0.063628	308160000
1986-03-17	0.100694	0.103299	0.100694	0.102431	0.064725	133171200
1986-03-18	0.102431	0.103299	0.098958	0.099826	0.063079	67766400
1986-03-19	0.099826	0.100694	0.097222	0.098090	0.061982	47894400

In [11]:
```
# selecting Feature Columns
feature_columns = ['Open', 'High', 'Low', 'Volume']
```

Normalizing the data

In [12]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
feature_minmax_transform_data = scaler.fit_transform(test[feature_columns])
feature_minmax_transform = pd.DataFrame(columns=feature_columns, data=feature_minm
feature_minmax_transform.head()
```

Out[12]:

Date	Open	High	Low	Volume
1986-03-13	0.000000	0.000027	0.000000	1.000000
1986-03-14	0.000025	0.000030	0.000025	0.297096
1986-03-17	0.000035	0.000032	0.000036	0.127119
1986-03-18	0.000040	0.000032	0.000030	0.063588
1986-03-19	0.000033	0.000025	0.000025	0.044285

In [13]:

```
display(feature_minmax_transform.head())
print('Shape of features : ', feature_minmax_transform.shape)
print('Shape of target : ', target_adj_close.shape)

# Shift target array because we want to predict the n + 1 day value


target_adj_close = target_adj_close.shift(-1)
validation_y = target_adj_close[-90:-1]
target_adj_close = target_adj_close[:-90]

# Taking last 90 rows of data to be validation set
validation_X = feature_minmax_transform[-90:-1]
feature_minmax_transform = feature_minmax_transform[:-90]
display(validation_X.tail())
display(validation_y.tail())

print("\n -----After process------ \n")
print('Shape of features : ', feature_minmax_transform.shape)
print('Shape of target : ', target_adj_close.shape)
display(target_adj_close.tail())
```

Date	Open	High	Low	Volume
1986-03-13	0.000000	0.000027	0.000000	1.000000
1986-03-14	0.000025	0.000030	0.000025	0.297096
1986-03-17	0.000035	0.000032	0.000036	0.127119
1986-03-18	0.000040	0.000032	0.000030	0.063588
1986-03-19	0.000033	0.000025	0.000025	0.044285

```
Shape of features :  (9083, 4)
Shape of target :  (9083, 1)
```

Date	Open	High	Low	Volume
2022-03-17	0.851015	0.845356	0.845577	0.027696
2022-03-18	0.857052	0.860775	0.855398	0.039838
2022-03-21	0.867269	0.858315	0.861741	0.025301
2022-03-22	0.869910	0.872217	0.873053	0.024571
2022-03-23	0.871971	0.867154	0.869984	0.022741

Date	Adj Close
2022-03-17	300.429993
2022-03-18	299.160004
2022-03-21	304.059998
2022-03-22	299.489990
2022-03-23	304.100006

```
 -----After process------

Shape of features :  (8993, 4)
Shape of target :  (8993, 1)
```

Date	Adj Close
2021-11-08	334.644562
2021-11-09	329.514557
2021-11-10	331.138214
2021-11-11	335.411560
2021-11-12	334.764099

Train test Split using Timeseries split

In [14]:
```
ts_split= TimeSeriesSplit(n_splits=10)
for train_index, test_index in ts_split.split(feature_minmax_transform):
        X_train, X_test = feature_minmax_transform[:len(train_index)], feature_min
        y_train, y_test = target_adj_close[:len(train_index)].values.ravel(), targ
```

In [15]:
```
X_train.shape
```

Out[15]:
```
(8176, 4)
```

In [16]:
```
X_test.shape
```

Out[16]:
```
(817, 4)
```

In [17]:
```
y_train.shape
```

Out[17]:
```
(8176,)
```

In [18]:
```
y_test.shape
```

Out[18]:
```
(817,)
```

In [19]:
```
def validate_result(model, model_name):
    predicted = model.predict(validation_X)
    RSME_score = np.sqrt(mean_squared_error(validation_y, predicted))
    print('RMSE: ', RSME_score)

    R2_score = r2_score(validation_y, predicted)
```

```
    print('R2 score: ', R2_score)

    plt.plot(validation_y.index, predicted,'r', label='Predict')
    plt.plot(validation_y.index, validation_y,'b', label='Actual')
    plt.ylabel('Price')
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
    plt.title(model_name + ' Predict vs Actual')
    plt.legend(loc='upper right')
    plt.show()
```

Benchmark Model

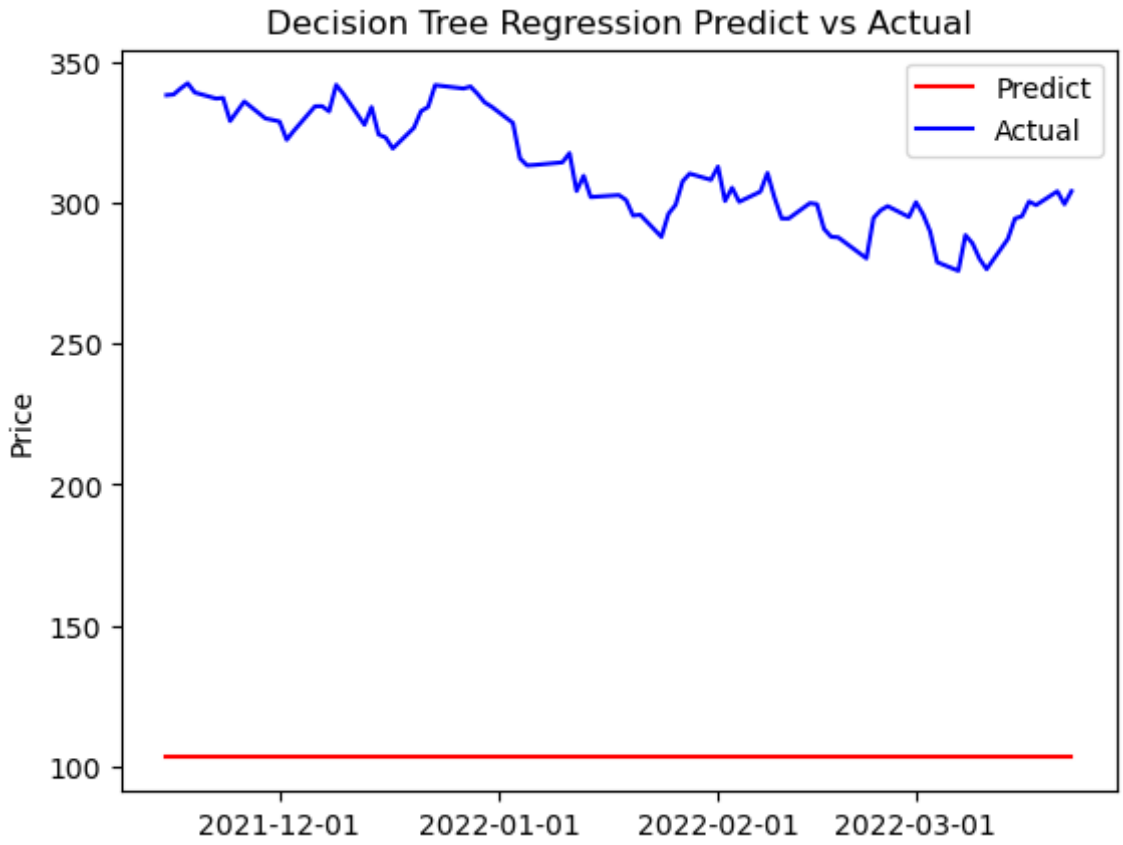
In [20]:
```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(random_state=0)

benchmark_dt=dt.fit(X_train, y_train)

validate_result(benchmark_dt, 'Decision Tree Regression')
```

```
RMSE:  209.66947344119035
R2 score:  -119.13211468450538
```



Process the data for LSTM

In [21]:
```
X_train =np.array(X_train)
X_test =np.array(X_test)

X_tr_t = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_tst_t = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
```

Model building : LSTM

In [22]:
```
from keras.models import Sequential
from keras.layers import Dense
import keras.backend as K
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.models import load_model
from keras.layers import LSTM
K.clear_session()
model_lstm = Sequential()
model_lstm.add(LSTM(16, input_shape=(1, X_train.shape[1]), activation='relu', retu
model_lstm.add(Dense(1))
model_lstm.compile(loss='mean_squared_error', optimizer='adam')
early_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)
history_model_lstm = model_lstm.fit(X_tr_t, y_train, epochs=200, batch_size=8, verl
```

```
Epoch 1/200
1022/1022 [==============================] - 8s 5ms/step - loss: 202.0299
Epoch 2/200
1022/1022 [==============================] - 3s 3ms/step - loss: 197.9244
Epoch 3/200
1022/1022 [==============================] - 3s 3ms/step - loss: 179.2041
Epoch 4/200
1022/1022 [==============================] - 4s 4ms/step - loss: 144.2872
Epoch 5/200
1022/1022 [==============================] - 3s 3ms/step - loss: 109.7324
Epoch 6/200
1022/1022 [==============================] - 3s 3ms/step - loss: 76.9869
Epoch 7/200
1022/1022 [==============================] - 3s 3ms/step - loss: 48.6590
Epoch 8/200
1022/1022 [==============================] - 3s 3ms/step - loss: 27.4613
Epoch 9/200
1022/1022 [==============================] - 4s 4ms/step - loss: 14.3307
Epoch 10/200
1022/1022 [==============================] - 4s 4ms/step - loss: 7.7830
Epoch 11/200
1022/1022 [==============================] - 3s 3ms/step - loss: 5.1858
Epoch 12/200
1022/1022 [==============================] - 5s 5ms/step - loss: 4.2444
Epoch 13/200
1022/1022 [==============================] - 4s 4ms/step - loss: 3.8032
Epoch 14/200
1022/1022 [==============================] - 4s 4ms/step - loss: 3.5185
Epoch 15/200
1022/1022 [==============================] - 4s 4ms/step - loss: 3.3090
Epoch 16/200
1022/1022 [==============================] - 5s 5ms/step - loss: 3.1494
Epoch 17/200
1022/1022 [==============================] - 4s 4ms/step - loss: 3.0260
Epoch 18/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.9300
Epoch 19/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.8547
Epoch 20/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.7953
Epoch 21/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.7478
Epoch 22/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.7089
Epoch 23/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.6764
Epoch 24/200
1022/1022 [==============================] - 5s 5ms/step - loss: 2.6485
Epoch 25/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.6238
Epoch 26/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.6014
Epoch 27/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.5806
Epoch 28/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.5610
Epoch 29/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.5423
Epoch 30/200
1022/1022 [==============================] - 5s 5ms/step - loss: 2.5243
Epoch 31/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.5068
Epoch 32/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.4898
```

```
Epoch 33/200
1022/1022 [==============================] - 4s 3ms/step - loss: 2.4733
Epoch 34/200
1022/1022 [==============================] - 4s 3ms/step - loss: 2.4570
Epoch 35/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.4412
Epoch 36/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.4256
Epoch 37/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.4103
Epoch 38/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.3954
Epoch 39/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.3807
Epoch 40/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.3663
Epoch 41/200
1022/1022 [==============================] - 4s 3ms/step - loss: 2.3523
Epoch 42/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.3385
Epoch 43/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.3249
Epoch 44/200
1022/1022 [==============================] - 4s 3ms/step - loss: 2.3117
Epoch 45/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2987
Epoch 46/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2860
Epoch 47/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2735
Epoch 48/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2613
Epoch 49/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2493
Epoch 50/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2376
Epoch 51/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2262
Epoch 52/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.2149
Epoch 53/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.2040
Epoch 54/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.1932
Epoch 55/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.1827
Epoch 56/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.1724
Epoch 57/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.1623
Epoch 58/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.1524
Epoch 59/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.1427
Epoch 60/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.1332
Epoch 61/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.1240
Epoch 62/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.1149
Epoch 63/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.1060
Epoch 64/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.0973
```

```
Epoch 65/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.0888
Epoch 66/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.0804
Epoch 67/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0723
Epoch 68/200
1022/1022 [==============================] - 2s 2ms/step - loss: 2.0642
Epoch 69/200
1022/1022 [==============================] - 3s 2ms/step - loss: 2.0564
Epoch 70/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0488
Epoch 71/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0412
Epoch 72/200
1022/1022 [==============================] - 4s 4ms/step - loss: 2.0339
Epoch 73/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0267
Epoch 74/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0196
Epoch 75/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0127
Epoch 76/200
1022/1022 [==============================] - 3s 3ms/step - loss: 2.0060
Epoch 77/200
1022/1022 [==============================] - 3s 3ms/step - loss: 1.9993
Epoch 78/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9928
Epoch 79/200
1022/1022 [==============================] - 3s 2ms/step - loss: 1.9865
Epoch 80/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9802
Epoch 81/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9741
Epoch 82/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9682
Epoch 83/200
1022/1022 [==============================] - 3s 2ms/step - loss: 1.9623
Epoch 84/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9565
Epoch 85/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9510
Epoch 86/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9455
Epoch 87/200
1022/1022 [==============================] - 3s 2ms/step - loss: 1.9400
Epoch 88/200
1022/1022 [==============================] - 3s 2ms/step - loss: 1.9349
Epoch 89/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9298
Epoch 90/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9247
Epoch 91/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9199
Epoch 92/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9152
Epoch 93/200
1022/1022 [==============================] - 3s 3ms/step - loss: 1.9105
Epoch 94/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9060
Epoch 95/200
1022/1022 [==============================] - 2s 2ms/step - loss: 1.9012
Epoch 96/200
1022/1022 [==============================] - 3s 3ms/step - loss: 1.8967
```

```
Epoch 97/200
1022/1022 [==============================] - 4s 4ms/step - loss: 1.8920
Epoch 98/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8877
Epoch 99/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8834
Epoch 100/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8792
Epoch 101/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8751
Epoch 102/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8711
Epoch 103/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8671
Epoch 104/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8632
Epoch 105/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8593
Epoch 106/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8555
Epoch 107/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8518
Epoch 108/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8481
Epoch 109/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8444
Epoch 110/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8408
Epoch 111/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8373
Epoch 112/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8338
Epoch 113/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8303
Epoch 114/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8269
Epoch 115/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8235
Epoch 116/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8202
Epoch 117/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8168
Epoch 118/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8135
Epoch 119/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8103
Epoch 120/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8071
Epoch 121/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8040
Epoch 122/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.8009
Epoch 123/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7978
Epoch 124/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7946
Epoch 125/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7915
Epoch 126/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7884
Epoch 127/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7852
Epoch 128/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7821
```

```
Epoch 129/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7793
Epoch 130/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7762
Epoch 131/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7733
Epoch 132/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7704
Epoch 133/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7679
Epoch 134/200
1022/1022 [==============================] - 6s 6ms/step - loss: 1.7650
Epoch 135/200
1022/1022 [==============================] - 6s 6ms/step - loss: 1.7624
Epoch 136/200
1022/1022 [==============================] - 6s 5ms/step - loss: 1.7597
Epoch 137/200
1022/1022 [==============================] - 6s 5ms/step - loss: 1.7574
Epoch 138/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7548
Epoch 139/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7523
Epoch 140/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7498
Epoch 141/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7474
Epoch 142/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.7451
Epoch 143/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7431
Epoch 144/200
1022/1022 [==============================] - 4s 4ms/step - loss: 1.7407
Epoch 145/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.7385
Epoch 146/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7364
Epoch 147/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7343
Epoch 148/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7322
Epoch 149/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7302
Epoch 150/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7282
Epoch 151/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7263
Epoch 152/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7244
Epoch 153/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7226
Epoch 154/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7208
Epoch 155/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7190
Epoch 156/200
1022/1022 [==============================] - 6s 5ms/step - loss: 1.7173
Epoch 157/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7156
Epoch 158/200
1022/1022 [==============================] - 6s 6ms/step - loss: 1.7140
Epoch 159/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7124
Epoch 160/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7109
```

```
Epoch 161/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7094
Epoch 162/200
1022/1022 [==============================] - 4s 4ms/step - loss: 1.7079
Epoch 163/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.7065
Epoch 164/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7051
Epoch 165/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7037
Epoch 166/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.7024
Epoch 167/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7012
Epoch 168/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.7000
Epoch 169/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6988
Epoch 170/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6977
Epoch 171/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6966
Epoch 172/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6955
Epoch 173/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6945
Epoch 174/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6935
Epoch 175/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6926
Epoch 176/200
1022/1022 [==============================] - 4s 4ms/step - loss: 1.6917
Epoch 177/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6909
Epoch 178/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6901
Epoch 179/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6894
Epoch 180/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6887
Epoch 181/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6880
Epoch 182/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6874
Epoch 183/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6868
Epoch 184/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6863
Epoch 185/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6859
Epoch 186/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6854
Epoch 187/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6851
Epoch 188/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6847
Epoch 189/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6845
Epoch 190/200
1022/1022 [==============================] - 4s 4ms/step - loss: 1.6843
Epoch 191/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6841
Epoch 192/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6840
```

```
Epoch 193/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6839
Epoch 194/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6839
Epoch 195/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6840
Epoch 196/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6841
Epoch 197/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6843
Epoch 198/200
1022/1022 [==============================] - 5s 4ms/step - loss: 1.6845
Epoch 199/200
1022/1022 [==============================] - 5s 5ms/step - loss: 1.6847
Epoch 199: early stopping
```

Evaluation of Model

In [23]:
```
y_pred_test_lstm = model_lstm.predict(X_tst_t)
y_train_pred_lstm = model_lstm.predict(X_tr_t)
print("The R2 score on the Train set is:\t{:0.3f}".format(r2_score(y_train, y_trai
r2_train = r2_score(y_train, y_train_pred_lstm)

print("The R2 score on the Test set is:\t{:0.3f}".format(r2_score(y_test, y_pred_t
r2_test = r2_score(y_test, y_pred_test_lstm)
```

```
26/26 [==============================] - 1s 3ms/step
256/256 [==============================] - 1s 4ms/step
The R2 score on the Train set is:       0.947
The R2 score on the Test set is:        0.836
```

Predictions made by LSTM

In [24]:
```
score_lstm= model_lstm.evaluate(X_tst_t, y_test, batch_size=1)
```

```
817/817 [==============================] - 3s 3ms/step - loss: 663.2576
```

In [25]:
```
print('LSTM: %f'%score_lstm)
```

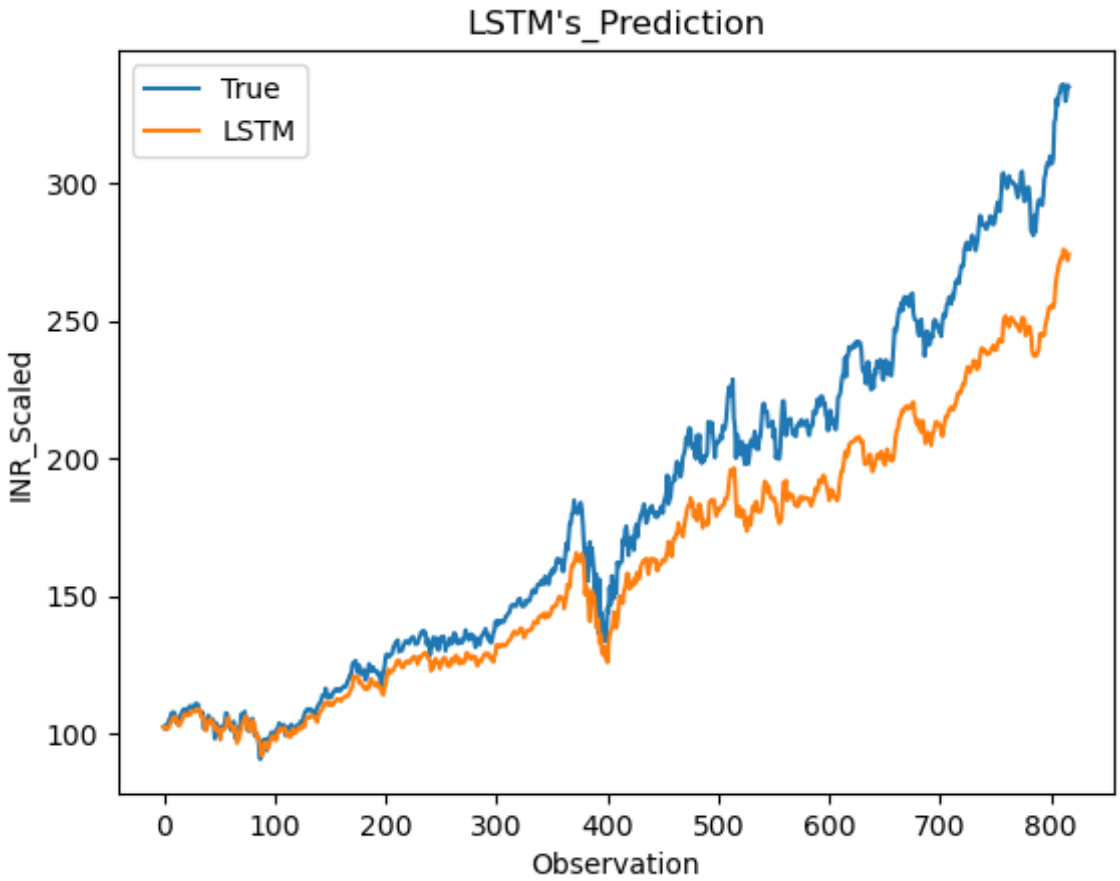
```
LSTM: 663.257568
```

In [26]:
```
y_pred_test_LSTM = model_lstm.predict(X_tst_t)
```

```
26/26 [==============================] - 0s 3ms/step
```

LSTM's Prediction Visual

In [27]:
```
plt.plot(y_test, label='True')
plt.plot(y_pred_test_LSTM, label='LSTM')
plt.title("LSTM's_Prediction")
plt.xlabel('Observation')
plt.ylabel('INR_Scaled')
plt.legend()
plt.show()
```



Converting Prediction data

In this step I have made the prediction of test data and will convert the dataframe to csv so that we can see the price difference between actual and predicted price.

In [28]:
```
col1 = pd.DataFrame(y_test, columns=['True'])

col2 = pd.DataFrame(y_pred_test_LSTM, columns=['LSTM_prediction'])

col3 = pd.DataFrame(history_model_lstm.history['loss'], columns=['Loss_LSTM'])
results = pd.concat([col1, col2, col3], axis=1)
results.to_excel('PredictionResults_LSTM_NonShift.xlsx')
```

Conclusion

It is Not possible to get a model that can 99% predict the price without any error, there are too many factors can affect the stock prices. So, we cannot hope there is a perfect model, but the general trend of predicted price is in line with the actual data, so the trader could have an indicator to reference, and makes trading decision by himself.