Let's forecast the Closing Prices of Apple Company with the help of Statistics and Machine Learning!

Stock market forecasting is a behavior to determine the future value of corporate stocks or other financial instruments traded on exchanges.

Successful forecast of the future stock price can make considerable profit.

This Notebook attempts to forecast the Closing Prices of Apple Company. So what are we waiting for, let's get into the notebook.

The whole notebook is divided into 2 parts:

1. Data Acquisition: We will be using the yfinance library to extractTime Series data for our reqd task

2. Model Building and Evaluation:

----> Data Preprocessing: We will preprocess the data and convert them into a suitable format for the model we are going to build.

----> We will dwelve into leveraging Recurrent Neural Nets and further look into building Classical Machine Learning models, explore Facebook's Prophet Model and finally end with trying ARIMA.

In [1]:

```
#Importing the reqd libraries:
import math
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Mute sklearn warnings
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
simplefilter(action='ignore', category=DeprecationWarning)
# Mute general warnings
import warnings
warnings.filterwarnings('ignore')

# Chart drawing
import plotly as py
import plotly.io as pio
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

### Machine Learning Libraries
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_absolute_error,mean_squared_error
from xgboost import XGBRegressor,plot_importance

from statsmodels.tsa.arima.model import ARIMA

### Deep Learning Libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import LSTM,Dropout,Dense,GRU
```

In [2]:

```
plt.style.use('fivethirtyeight')
```

In [3]:

```
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.4-py2.py3-none-any.whl (51 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.4/51.4 kB 580.9 kB/s eta 0:00:00
Requirement already satisfied: requests>=2.26 in /opt/conda/lib/python3.7/site-packages (
from yfinance) (2.28.1)
Requirement already satisfied: lxml>=4.9.1 in /opt/conda/lib/python3.7/site-packages (fro
m yfinance) (4.9.1)
Collecting pytz>=2022.5
  Downloading pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 499.4/499.4 kB 3.4 MB/s eta 0:00:00
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/lib/python3.7/site-packages (f
rom yfinance) (1.3.5)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/lib/python3.7/site-pa
ckages (from yfinance) (4.11.1)
Requirement already satisfied: cryptography>=3.3.2 in /opt/conda/lib/python3.7/site-packa
ges (from yfinance) (37.0.2)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: appdirs>=1.4.4 in /opt/conda/lib/python3.7/site-packages (
from yfinance) (1.4.4)
Requirement already satisfied: frozendict>=2.3.4 in /opt/conda/lib/python3.7/site-package
s (from yfinance) (2.3.4)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/lib/python3.7/site-packages (f
rom yfinance) (1.21.6)
Requirement already satisfied: html5lib>=1.1 in /opt/conda/lib/python3.7/site-packages (f
rom yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.7/site-packages (f
rom beautifulsoup4>=4.11.1->yfinance) (2.3.1)
Requirement already satisfied: cffi>=1.12 in /opt/conda/lib/python3.7/site-packages (from
cryptography>=3.3.2->yfinance) (1.15.0)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.7/site-packages (fr
om html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.7/site-packages (from h
tml5lib>=1.1->yfinance) (1.15.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-pa
ckages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-pac
kages (from requests>=2.26->yfinance) (1.26.12)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (fr
om requests>=2.26->yfinance) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in /opt/conda/lib/python3.7/site-
packages (from requests>=2.26->yfinance) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packag
es (from requests>=2.26->yfinance) (2022.9.24)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.7/site-packages (from
cffi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
Installing collected packages: pytz, multitasking, yfinance
  Attempting uninstall: pytz
    Found existing installation: pytz 2022.1
    Uninstalling pytz-2022.1:
      Successfully uninstalled pytz-2022.1
ERROR: pip's dependency resolver does not currently take into account all the packages th
at are installed. This behaviour is the source of the following dependency conflicts.
dask-cudf 21.10.1 requires cupy-cuda114, which is not installed.
beatrix-jupyterlab 3.1.7 requires google-cloud-bigquery-storage, which is not installed.
pandas-profiling 3.1.0 requires markupsafe~=2.0.1, but you have markupsafe 2.1.1 which is
incompatible.
dask-cudf 21.10.1 requires dask==2021.09.1, but you have dask 2022.2.0 which is incompati
ble.
dask-cudf 21.10.1 requires distributed==2021.09.1, but you have distributed 2022.2.0 whic
h is incompatible.
apache-beam 2.40.0 requires dill<0.3.2,>=0.3.1.1, but you have dill 0.3.5.1 which is inco
```

```
mpatible.
Successfully installed multitasking-0.0.11 pytz-2022.7.1 yfinance-0.2.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

In [4]:

```
import yfinance as yf
```

1. Data Acquisition:

We will be extracting the data using the yfinance library.

You can read more about its documentation here: https://pypi.org/project/yfinance/

In [5]:

```
## Extracting data with the help of the yfinance library
## Here start parameter indicates the start time and end parameter indicates the end time
.
data = yf.download("AAPL", start="2012-01-01", end="2020-01-01")

### Displaying first 5 rows
data.head()
```

```
[**********************100%**********************]  1 of 1 completed
```

Out[5]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03 00:00:00-05:00	14.621429	14.732143	14.607143	14.686786	12.519277	302220800
2012-01-04 00:00:00-05:00	14.642857	14.810000	14.617143	14.765714	12.586557	260022000
2012-01-05 00:00:00-05:00	14.819643	14.948214	14.738214	14.929643	12.726295	271269600
2012-01-06 00:00:00-05:00	14.991786	15.098214	14.972143	15.085714	12.859331	318292800
2012-01-09 00:00:00-05:00	15.196429	15.276786	15.048214	15.061786	12.838936	394024400

As you can see, the data is clearly time series (index is the indication).

In [6]:

```
data.columns
```

Out[6]:

```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

The Dataset contains attributes like:

Open - Opening Price of the Stock,

High - Highest Trading Price of the day,

Low - Lowest Trading Price of the day,

Close - Closing Price of the Stock,

Adj Close - Adjusted Closing Price.

Plotting the OHLC(Open-High-Low-Close) Chart to get a sense of the data:

In [7]:

```
fig = make_subplots(rows=2, cols=1)

fig.add_trace(go.Ohlc(x=data.index,
                      open=data.Open,
                      high=data.High,
                      low=data.Low,
                      close=data.Close,
                      name='Price'), row=1, col=1)

fig.add_trace(go.Scatter(x=data.index, y=data.Volume, name='Volume'), row=2, col=1)

fig.update(layout_xaxis_rangeslider_visible=False)
fig.show()
```

In this Notebook, we will be focused on forecasting theClosing Price of the Stock.

In [8]:

```
data.shape
```

Out[8]:

```
(2012, 6)
```

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2012 entries, 2012-01-03 00:00:00-05:00 to 2019-12-31 00:00:00-05:00
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
```

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Open        2012 non-null   float64
 1   High        2012 non-null   float64
 2   Low         2012 non-null   float64
 3   Close       2012 non-null   float64
 4   Adj Close   2012 non-null   float64
 5   Volume      2012 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 174.6 KB
```

In [10]:

```
## Lets take a quick view of the Trend in the Closing Price
plt.figure(figsize=(16,8))
plt.title("Closing Price History")
plt.plot(data['Close'])
plt.xlabel("Data")
plt.ylabel("Close Price in USD")
plt.legend()
plt.show()
```

Apple has grown substantially since 2012.

2. Model Building

Using LSTMs:

Data Preprocessing

In [11]:

```
### Considering the Closing Price only
df=data["Close"].values

### Converting the array into a 2-D one
df=np.reshape(df,(-1,1))
```

There are two options for feature scaling — Standardisation and Normalisation. Both methods produce different results, so you can compare them and decide which one is best for you. I have used Normalisation in this notebook.

In [12]:

```
### Scaling the data
### sc=StandardScaler()
sc=MinMaxScaler(feature_range=(0,1))
df_scaled=sc.fit_transform(df)
df_scaled[:10]
```

Out[12]:

```
array([[0.01243228],
       [0.01375958],
       [0.01651631],
       [0.01914091],
       [0.01873851],
       [0.0196454 ],
       [0.01923099],
       [0.0185343 ],
       [0.01758536],
       [0.02052227]])
```

Preparing the train data:

In [13]:

```
### Using 80% as the Train data::
l=math.ceil(len(df_scaled)*0.8)
trained_data=df_scaled[0:l,:]
trained_data[:10]
```

Out[13]:

```
array([[0.01243228],
       [0.01375958],
       [0.01651631],
       [0.01914091],
       [0.01873851],
       [0.0196454 ],
       [0.01923099],
       [0.0185343 ],
       [0.01758536],
       [0.02052227]])
```

CRUX of the whole project lies in the way we feed input data to the model!

We define time step as a variable that can be tweaked to get better results.

Time step talks about the amount of temporal data used to train the model. In this notebook, I have considered the time step to be equal to 60, meaning that every row will contain 60 closing prices in order and the target will be the 61st closing price.

Feature	Target
Day 0, Day 1,....,Day 59	Day 60
Day 1, Day 2,....,Day 60	Day 61
Day 2, Day 3,....,Day 61	Day 62
.	
.	
.	
Day i-60, Day i-59,....., Day i-1	Day i

We preprocess the data in such a way that:

In [14]:

```
n_steps=60  ### time step variable
X_train=[]
y_train=[]

for i in range(n_steps,len(trained_data)):
    X_train.append(trained_data[i-n_steps:i,0])
    y_train.append(trained_data[i,0])
```

In [15]:

```
X_train=np.array(X_train)
y_train=np.array(y_train)
```

Preparing the test data:

In [16]:

```
### Test data to see the model performance::
test_data=df_scaled[l-n_steps:,:]

X_test=[]
y_test=df_scaled[l:,:]
for i in range(n_steps,len(test_data)):
    X_test.append(test_data[i-n_steps:i,0])

X_test=np.array(X_test)
```

Model Building and Evaluation:

The input into the LSTM and GRU layers requires to be of the form:

(n_rows,step_size,1)

In [17]:

```
### Reshaping the input data into (n_rows,window_size,1)
### This is the standard format to LSTMs and GRUs:
X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))  ### Reshaping the Test da
ta as well to satisfy input requirements:
```

In order to evaluate the model, we will be mainly focusing on using MAPE(Mean Absolute Percentage Error) as the metric, as well other metrics like RMSE and MSE.

MAPE is calculated as follows:

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

M = mean absolute percentage error

n = number of times the summation iteration happens

A_t = actual value

F_t = forecast value

The reason that this metric is preferred over the traditional RMSE is because:

MAPE is much more understandable than RMSE. Therefore, if you need to convey model performance to end users, especially to those who aren't data professionals, then MAPE would be the better choice as this is calculated as an easy to understand percentage.

In [18]:

```
### we will evaluate the model using a custom loss function which will calculate MAPE:
def mape(y_true, y_pred):
    # calculate loss, using y_pred
    loss=np.mean(np.abs((y_pred/y_true)-1))
    return loss
```

In [19]:

```
## Architecture of RNN using stacked LSTMs:
model1=Sequential([
                LSTM(1000,return_sequences=True,input_shape=(X_train.shape[1],1)),
                LSTM(500,return_sequences=True),
                LSTM(100,return_sequences=False),
                Dense(25),
                Dense(1)
])

model1.compile(optimizer="adam",loss="mean_squared_error")

## Architecture of RNN using stacked GRUs:
model2=Sequential([
                GRU(1000,return_sequences=True,input_shape=(X_train.shape[1],1)),
                GRU(500,return_sequences=True),
                GRU(100,return_sequences=False),
                Dense(25),
                Dense(1)
])

model2.compile(optimizer="adam",loss="mean_squared_error")
```

```
2023-01-24 12:08:17.925546: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:18.118189: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:18.119071: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:18.120861: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Ten
sorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the f
ollowing CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flag
s.
2023-01-24 12:08:18.121208: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:18.122000: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:18.122684: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:20.398348: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:20.399277: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:20.400042: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] s
uccessful NUMA node read from SysFS had negative value (-1), but there must be at least o
ne NUMA node, so returning NUMA node zero
2023-01-24 12:08:20.400716: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Crea
ted device /job:localhost/replica:0/task:0/device:GPU:0 with 15401 MB memory:  -> device:
0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

It is possible to use RMSprop or Adam as the optimizer. Both optimizers yield similar results, but RMS consumes a lot of memory, so I chose Adam instead.

In [20]:

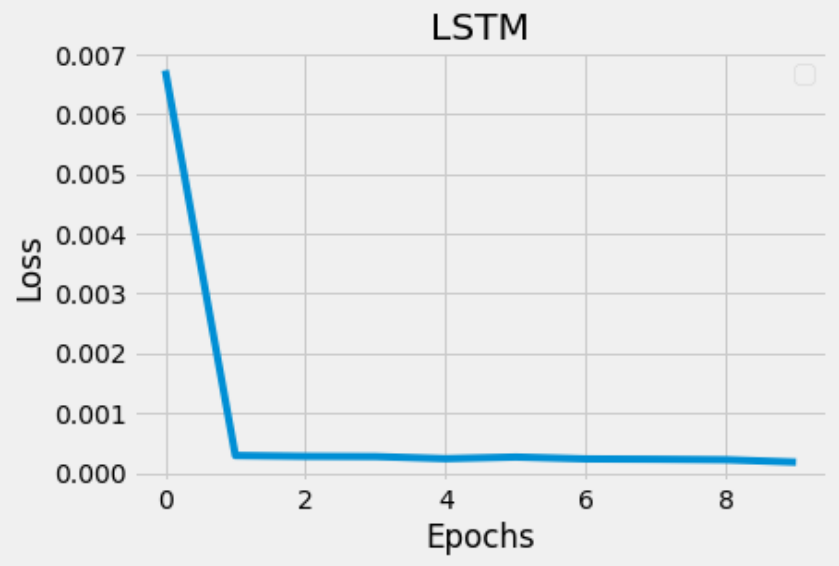
```
models={"LSTM":model1,"GRU":model2}
for i,j in models.items():
    ### Training the model
    j.fit(X_train,y_train,epochs=10)
    ### Visualising the loss vs epoch curve
    plt.title(i)
    plt.plot(j.history.history["loss"])
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend()
    plt.show()
    print("-----------------------------------")
```

```
2023-01-24 12:08:23.732425: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
```

```
Epoch 1/10
```

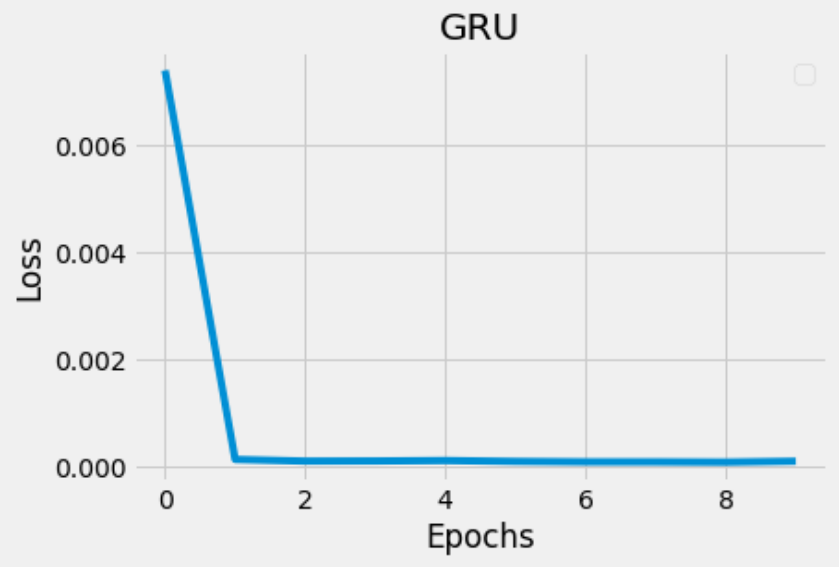
```
2023-01-24 12:08:28.002487: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN version 8005
```

```
49/49 [==============================] - 7s 37ms/step - loss: 0.0067
Epoch 2/10
49/49 [==============================] - 2s 36ms/step - loss: 2.9044e-04
Epoch 3/10
49/49 [==============================] - 2s 36ms/step - loss: 2.7742e-04
Epoch 4/10
49/49 [==============================] - 2s 36ms/step - loss: 2.7314e-04
Epoch 5/10
49/49 [==============================] - 2s 36ms/step - loss: 2.3787e-04
Epoch 6/10
49/49 [==============================] - 2s 37ms/step - loss: 2.6537e-04
Epoch 7/10
49/49 [==============================] - 2s 36ms/step - loss: 2.3388e-04
Epoch 8/10
49/49 [==============================] - 2s 36ms/step - loss: 2.2855e-04
Epoch 9/10
49/49 [==============================] - 2s 36ms/step - loss: 2.1980e-04
Epoch 10/10
49/49 [==============================] - 2s 36ms/step - loss: 1.8016e-04
```




```
-----------------------------------
Epoch 1/10
49/49 [==============================] - 5s 30ms/step - loss: 0.0074
Epoch 2/10
49/49 [==============================] - 1s 30ms/step - loss: 1.4353e-04
Epoch 3/10
```

```
49/49 [==============================] - 1s 30ms/step - loss: 1.1255e-04
Epoch 4/10
49/49 [==============================] - 1s 30ms/step - loss: 1.1505e-04
Epoch 5/10
49/49 [==============================] - 1s 30ms/step - loss: 1.2324e-04
Epoch 6/10
49/49 [==============================] - 1s 30ms/step - loss: 1.0671e-04
Epoch 7/10
49/49 [==============================] - 1s 30ms/step - loss: 9.8754e-05
Epoch 8/10
49/49 [==============================] - 1s 30ms/step - loss: 9.8441e-05
Epoch 9/10
49/49 [==============================] - 1s 30ms/step - loss: 9.3399e-05
Epoch 10/10
49/49 [==============================] - 1s 30ms/step - loss: 1.1023e-04
```



In [21]:

```
X_train.shape,X_test.shape
```

Out[21]:

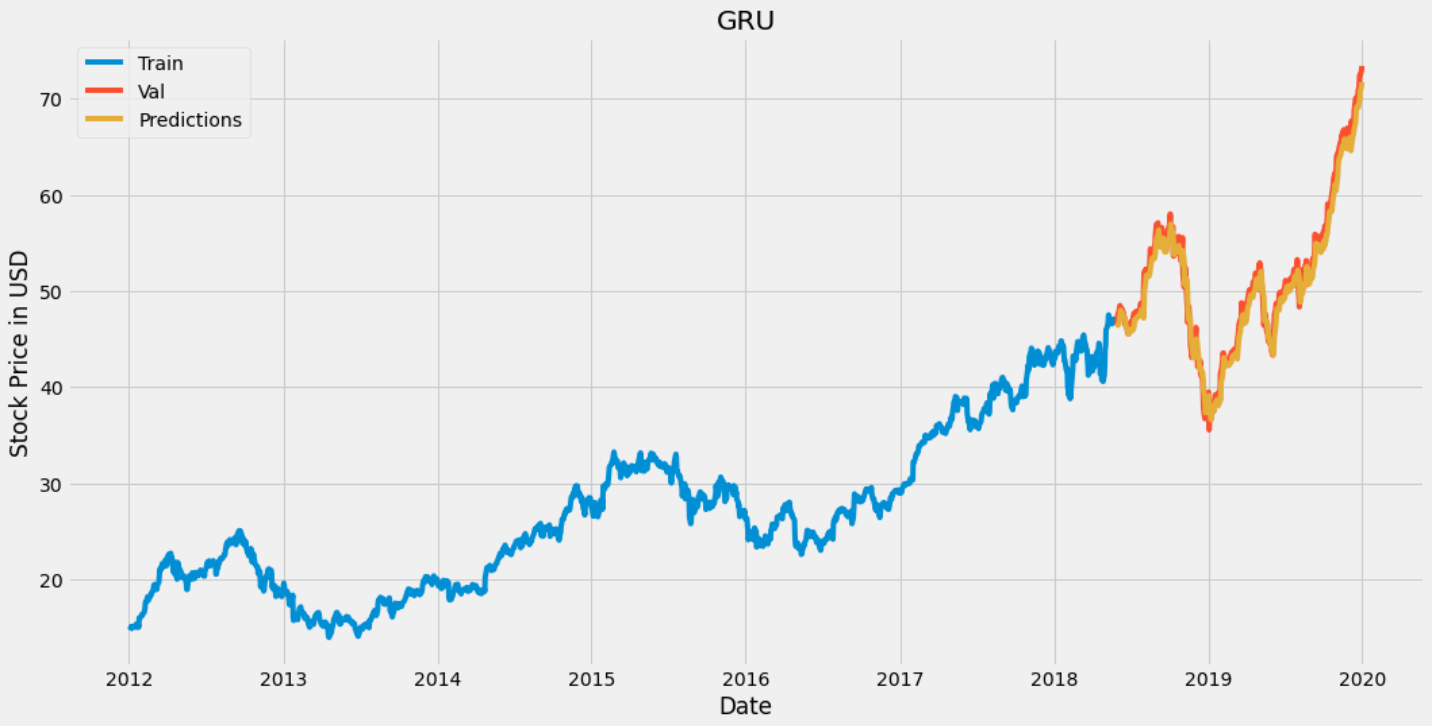
```
((1550, 60, 1), (402, 60, 1))
```

In [22]:

```
pred_={}

train=data[0:l] ### selecting the training data
valid=data[l:]  ### selecting the validation data

for i,j in models.items():
    ### Training the model
    pred=j.predict(X_test)
    pred=sc.inverse_transform(pred)    #### Rescaling the predictions

    valid["Predictions"]=pred

    plt.figure(figsize=(16,8))
    plt.title(f"{i}")
    plt.xlabel("Date")
    plt.ylabel("Stock Price in USD")
    plt.plot(train["Close"])
    plt.plot(valid[["Close","Predictions"]])
    plt.legend(["Train","Val","Predictions"])
    plt.show()

    print("\n")
    print(f"The MSE loss produced by {i} model is:",np.mean(((pred- y_test)**2)))
    print("\n")
```

```
    print(f"The RMSE loss produced by {i} model is:",np.sqrt(np.mean(((pred- y_test)**2)
)))
    print("\n")
    print(f"The MAPE loss produced by {i} model is:",mape(y_test, pred))
    print("\n")
```


```
The MSE loss produced by LSTM model is: 2628.981183604213

The RMSE loss produced by LSTM model is: 51.27359148337683

The MAPE loss produced by LSTM model is: 82.13285656769469
```




```
The MSE loss produced by GRU model is: 2549.3427967682574

The RMSE loss produced by GRU model is: 50.49101699082974
```

```
The MAPE loss produced by GRU model is: 80.74097398505018
```

Using the Classical Machine Learning Models:

Moving Averages

I'm calculating few moving averages to be used as features: SMA5, SMA10, SMA15, SMA30, and EMA9.

In [23]:

```
df=data.copy()
df.reset_index(inplace=True)

df['EMA_9'] = df['Close'].ewm(9).mean().shift()
df['SMA_5'] = df['Close'].rolling(5).mean().shift()
df['SMA_10'] = df['Close'].rolling(10).mean().shift()
df['SMA_15'] = df['Close'].rolling(15).mean().shift()
df['SMA_30'] = df['Close'].rolling(30).mean().shift()

fig = go.Figure()
fig.add_trace(go.Scatter(x=df.Date, y=df.EMA_9, name='EMA 9'))
fig.add_trace(go.Scatter(x=df.Date, y=df.SMA_5, name='SMA 5'))
fig.add_trace(go.Scatter(x=df.Date, y=df.SMA_10, name='SMA 10'))
fig.add_trace(go.Scatter(x=df.Date, y=df.SMA_15, name='SMA 15'))
fig.add_trace(go.Scatter(x=df.Date, y=df.SMA_30, name='SMA 30'))
fig.add_trace(go.Scatter(x=df.Date, y=df.Close, name='Close', opacity=0.2))
fig.show()
```

In [24]:

```
df['Close'] = df['Close'].shift(-1)
```

Drop invalid samples

Drop invalid samples

Because of calculating moving averages and shifting label column, few rows will have invalid values i.e. we haven't calculated SMA10 for the first 10 days. Moreover, after shifting Close price column, last row price is equal to 0 which is not true. Removing these samples should help.

In [25]:

```
df = df.iloc[33:] # Because of moving averages
df = df[:-1]      # Because of shifting close price

df.index = range(len(df))
```

In [26]:

```
### Here I the split stock dataframe into three subsets: training (70%), validation (15%)
and test (15%) sets.
test_size  = 0.15
valid_size = 0.15

test_split_idx  = int(df.shape[0] * (1-test_size))
valid_split_idx = int(df.shape[0] * (1-(valid_size+test_size)))

train_df  = df.loc[:valid_split_idx].copy()
valid_df  = df.loc[valid_split_idx+1:test_split_idx].copy()
test_df   = df.loc[test_split_idx+1:].copy()

fig = go.Figure()
fig.add_trace(go.Scatter(x=train_df.Date, y=train_df.Close, name='Training'))
fig.add_trace(go.Scatter(x=valid_df.Date, y=valid_df.Close, name='Validation'))
fig.add_trace(go.Scatter(x=test_df.Date,  y=test_df.Close,  name='Test'))
fig.show()
```

Drop unnecessary columns

In [27]:

```
drop_cols = ['Date', 'Volume', 'Open', 'Low', 'High', 'Adj Close']
```

```
train_df = train_df.drop(drop_cols, 1)
valid_df = valid_df.drop(drop_cols, 1)
test_df  = test_df.drop(drop_cols, 1)
```

In [28]:

```
y_train = train_df['Close'].copy()
X_train = train_df.drop(['Close'], 1)

y_valid = valid_df['Close'].copy()
X_valid = valid_df.drop(['Close'], 1)

y_test  = test_df['Close'].copy()
X_test  = test_df.drop(['Close'], 1)

X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1385 entries, 0 to 1384
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   EMA_9   1385 non-null   float64
 1   SMA_5   1385 non-null   float64
 2   SMA_10  1385 non-null   float64
 3   SMA_15  1385 non-null   float64
 4   SMA_30  1385 non-null   float64
dtypes: float64(5)
memory usage: 54.2 KB
```

In [29]:

```
models={"Random Forest":RandomForestRegressor(),"Linear Regression":LinearRegression(),"S
upport Vector":SVR(),"XGBoost Regressor":XGBRegressor()}
for i,j in models.items():
    ### Training the model::
    print(f"Fitting into {i} Model::")
    model = j
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(f'Mean Squared Error given by model is = {mean_squared_error(y_test, y_pred)}\n
')
```

```
Fitting into Random Forest Model::
Mean Squared Error given by model is = 204.92846315242025

Fitting into Linear Regression Model::
Mean Squared Error given by model is = 2.0283838322756225

Fitting into Support Vector Model::
Mean Squared Error given by model is = 609.1563994291051

Fitting into XGBoost Regressor Model::
Mean Squared Error given by model is = 198.13717285497052
```

Considering XGBoost as an example to improve its performance::

In [30]:

```
### Fine Tuning using Grid Search to find the best hyperparameters for our XGB Model:
parameters = {
    'n_estimators': [100, 200, 300, 400],
    'learning_rate': [0.001, 0.005, 0.01, 0.05],
    'max_depth': [8, 10, 12, 15],
    'gamma': [0.001, 0.005, 0.01, 0.02],
    'random_state': [42]
}
```

```python
eval_set = [(X_train, y_train), (X_valid, y_valid)]
model = XGBRegressor(eval_set=eval_set, objective='reg:squarederror', verbose=False)
clf = RandomizedSearchCV(model, parameters)

clf.fit(X_train, y_train)

print(f'Best params: {clf.best_params_}')
print(f'Best validation score = {clf.best_score_}')
```

```
[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:14] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:15] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:15] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:15] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:16] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:16] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:17] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:17] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:17] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:18] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:18] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:19] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:20] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:20] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:21] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:22] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:22] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:22] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:22] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:22] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:23] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:23] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:23] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:23] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:23] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:24] WARNING: ../src/learner.cc:627: 
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:24] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:24] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:25] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:25] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:25] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:26] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:26] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:26] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:27] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:27] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:27] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:28] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:28] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:28] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:29] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:29] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

```
[12:09:29] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:30] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:30] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


[12:09:31] WARNING: ../src/learner.cc:627:
Parameters: { "eval_set", "verbose" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually being use
d
  but getting flagged wrongly here. Please open an issue if you find any such cases.


Best params: {'random_state': 42, 'n_estimators': 100, 'max_depth': 15, 'learning_rate':
0.05, 'gamma': 0.005}
Best validation score = 0.7520246586769218
```

In [31]:

```
#### Creating a XGB Model with the best parameters obtained from above::
model = XGBRegressor(**clf.best_params_, objective='reg:squarederror')
model.fit(X_train, y_train, eval_set=eval_set, verbose=False)
```

Out[31]:

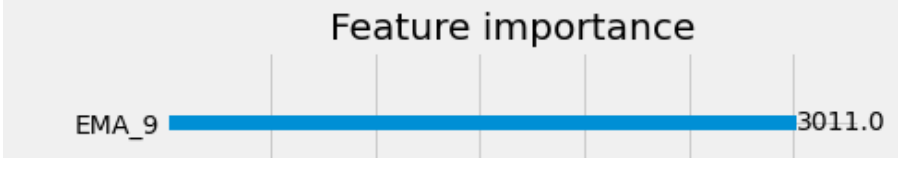
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0.005, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.05, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=15, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=42,
             reg_alpha=0, reg_lambda=1, ...)
```

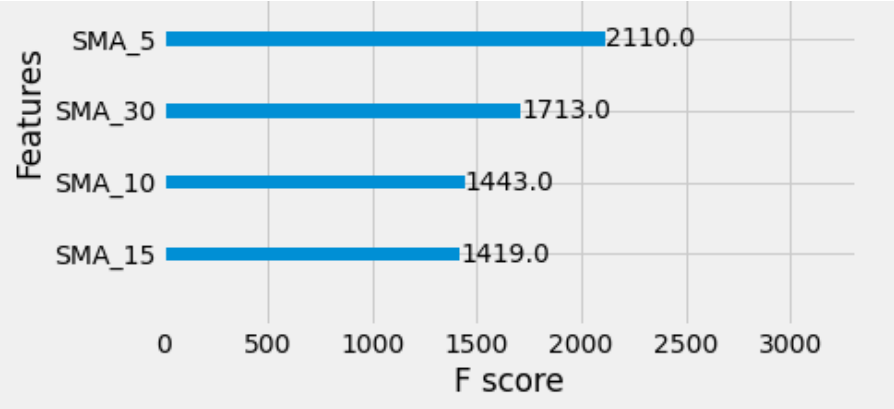
In [32]:

```
## Plotting Feature Importance::
plot_importance(model)
```

Out[32]:

```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>
```





In [33]:

```
y_pred = model.predict(X_test)
print(f'y_true = {np.array(y_test)[:5]}')
print(f'y_pred = {y_pred[:5]}')
```

```
y_true = [54.07500076 53.06000137 53.32500076 54.71500015 55.55500031]
y_pred = [39.152424 39.152424 39.152424 39.152424 39.152424]
```

In [34]:

```
print(f'mean_squared_error = {mean_squared_error(y_test, y_pred)}')
```

```
mean_squared_error = 215.0436023707764
```

In [35]:

```
predicted_prices = df.loc[test_split_idx+1:].copy()
predicted_prices['Close'] = y_pred

fig = make_subplots(rows=2, cols=1)
fig.add_trace(go.Scatter(x=df.Date, y=df.Close,
                         name='Truth',
                         marker_color='LightSkyBlue'), row=1, col=1)

fig.add_trace(go.Scatter(x=predicted_prices.Date,
                         y=predicted_prices.Close,
                         name='Prediction',
                         marker_color='MediumPurple'), row=1, col=1)

fig.add_trace(go.Scatter(x=predicted_prices.Date,
                         y=y_test,
                         name='Truth',
                         marker_color='LightSkyBlue',
                         showlegend=False), row=2, col=1)

fig.add_trace(go.Scatter(x=predicted_prices.Date,
                         y=y_pred,
                         name='Prediction',
                         marker_color='MediumPurple',
                         showlegend=False), row=2, col=1)

fig.show()
```

XGBoost didnt do well in forecasting, you can further try tweaking the parameters of Linear Regression.

Using Facebook's(Meta) Prophet Model:

Prophet (previously FbProphet), by Meta (previously Facebook), is a method for predicting time series data that uses an additive model to suit non-linear trends with seasonality that occurs annually, monthly, daily, and on holidays. Prophet typically manages outliers well and is robust to missing data and changes in the trend. The Prophet library can automatically manage seasonality parameters.

The mathematical equation behind the Prophet model is defined as:

y(t) = g(t) + s(t) + h(t) + e(t)

where,

g(t) representing the trend. Prophet uses a piecewise linear model for trend forecasting.

s(t) represents periodic changes (weekly, monthly, yearly).

h(t) represents the effects of holidays (recall: Holidays impact businesses).

e(t) is the error term.

The Prophet model fitting procedure is usually very fast (even for thousands of observations) and it does not require any data pre-processing. It deals also with missing data and outliers.

In [36]:

```
!pip install prophet
```

```
Requirement already satisfied: prophet in /opt/conda/lib/python3.7/site-packages (1.1.1)
Requirement already satisfied: matplotlib>=2.0.0 in /opt/conda/lib/python3.7/site-package
s (from prophet) (3.5.3)
Requirement already satisfied: python-dateutil>=2.8.0 in /opt/conda/lib/python3.7/site-pa
ckages (from prophet) (2.8.2)
Requirement already satisfied: setuptools-git>=1.2 in /opt/conda/lib/python3.7/site-packa
ges (from prophet) (1.2)
Requirement already satisfied: wheel>=0.37.0 in /opt/conda/lib/python3.7/site-packages (f
rom prophet) (0.37.1)
Requirement already satisfied: cmdstanpy>=1.0.4 in /opt/conda/lib/python3.7/site-packages
(from prophet) (1.0.7)
Requirement already satisfied: pandas>=1.0.4 in /opt/conda/lib/python3.7/site-packages (f
rom prophet) (1.3.5)
Requirement already satisfied: LunarCalendar>=0.0.9 in /opt/conda/lib/python3.7/site-pack
ages (from prophet) (0.0.9)
Requirement already satisfied: holidays>=0.14.2 in /opt/conda/lib/python3.7/site-packages
(from prophet) (0.16)
Requirement already satisfied: tqdm>=4.36.1 in /opt/conda/lib/python3.7/site-packages (fr
om prophet) (4.64.0)
```

```
Requirement already satisfied: convertdate>=2.1.2 in /opt/conda/lib/python3.7/site-packag
es (from prophet) (2.4.0)
Requirement already satisfied: setuptools>=42 in /opt/conda/lib/python3.7/site-packages (
from prophet) (59.8.0)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.7/site-packages (f
rom prophet) (1.21.6)
Requirement already satisfied: ujson in /opt/conda/lib/python3.7/site-packages (from cmds
tanpy>=1.0.4->prophet) (5.3.0)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /opt/conda/lib/python3.7/site-packa
ges (from convertdate>=2.1.2->prophet) (0.5.11)
Requirement already satisfied: korean-lunar-calendar in /opt/conda/lib/python3.7/site-pac
kages (from holidays>=0.14.2->prophet) (0.3.1)
Requirement already satisfied: hijri-converter in /opt/conda/lib/python3.7/site-packages
(from holidays>=0.14.2->prophet) (2.2.4)
Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.7/site-packages (
from LunarCalendar>=0.0.9->prophet) (4.1.3)
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (from Lunar
Calendar>=0.0.9->prophet) (2022.7.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-package
s (from matplotlib>=2.0.0->prophet) (1.4.3)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.0.0->prophet) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-package
s (from matplotlib>=2.0.0->prophet) (4.33.3)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages
(from matplotlib>=2.0.0->prophet) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (f
rom matplotlib>=2.0.0->prophet) (9.1.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (fr
om matplotlib>=2.0.0->prophet) (0.11.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from p
ython-dateutil>=2.8.0->prophet) (1.15.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/site-package
s (from kiwisolver>=1.0.1->matplotlib>=2.0.0->prophet) (4.1.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

In [37]:

```
from prophet import Prophet
```

In [38]:

```
#Selecting only the required features i.e. the date and closing price
df=data.reset_index()[['Date','Adj Close']]

## We must rename the features to "ds" and "y" for the model fitting
df = df.rename(columns = {"Date":"ds","Adj Close":"y"})
df["ds"]=df["ds"].apply(lambda x:pd.to_datetime(x).date())
df.head()
```

Out[38]:

	ds	y
0	2012-01-03	12.519277
1	2012-01-04	12.586557
2	2012-01-05	12.726295
3	2012-01-06	12.859331
4	2012-01-09	12.838936

In [39]:

```
train_data = df.sample(frac=0.8, random_state=0)
test_data = df.drop(train_data.index)

print(f'training data size : {train_data.shape}')
```

```
print(f'testing data size : {test_data.shape}')
```

```
training data size : (1610, 2)
testing data size : (402, 2)
```

In [40]:

```
model = Prophet(daily_seasonality=True)
model.fit(train_data)
```

```
12:09:44 - cmdstanpy - INFO - Chain [1] start processing
12:09:45 - cmdstanpy - INFO - Chain [1] done processing
```

Out[40]:

```
<prophet.forecaster.Prophet at 0x7fb9c77e8a90>
```

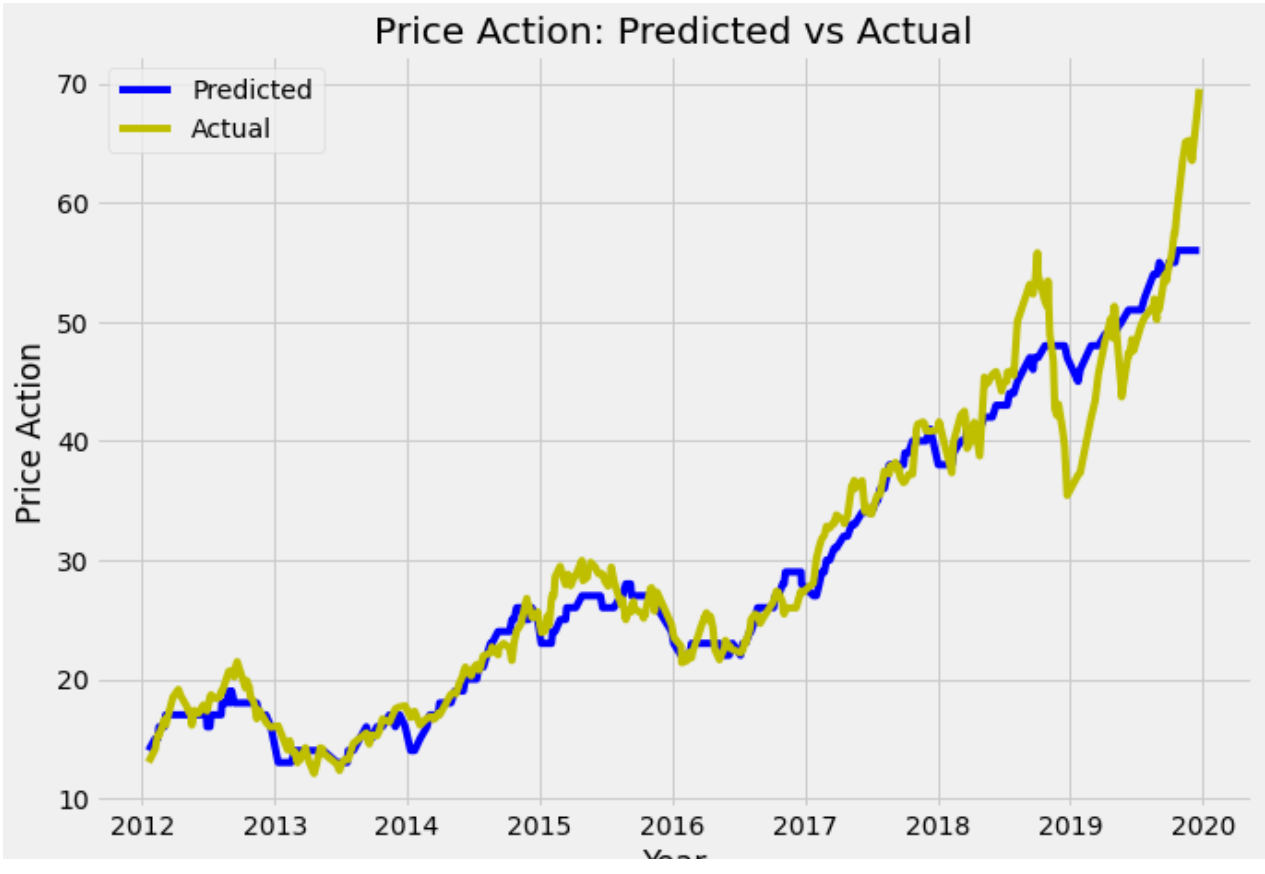
In [41]:

```
y_actual = test_data['y']
prediction = model.predict(pd.DataFrame({'ds':test_data['ds']}))
y_predicted = prediction['yhat']
y_predicted = y_predicted.astype(int)
print(f"The MSE obtained using the Prophet Model is: {mean_absolute_error(y_actual, y_predicted)}")
print("\n")
print(f"The RMSE obtained using the Prophet Model is: {mean_absolute_error(y_actual, y_predicted)**0.5}")
```

```
The MSE obtained using the Prophet Model is: 1.7987683092183735


The RMSE obtained using the Prophet Model is: 1.3411816838960982
```

In [42]:

```
plt.figure(figsize=(10,7))
plt.plot(test_data['ds'], y_predicted, 'b', label="Predicted")
plt.plot(test_data['ds'], y_actual, 'y', label="Actual")
plt.xlabel("Year")
plt.ylabel("Price Action")
plt.title("Price Action: Predicted vs Actual")
plt.legend()
plt.show()
```



Forecasting future prices::

In [43]:

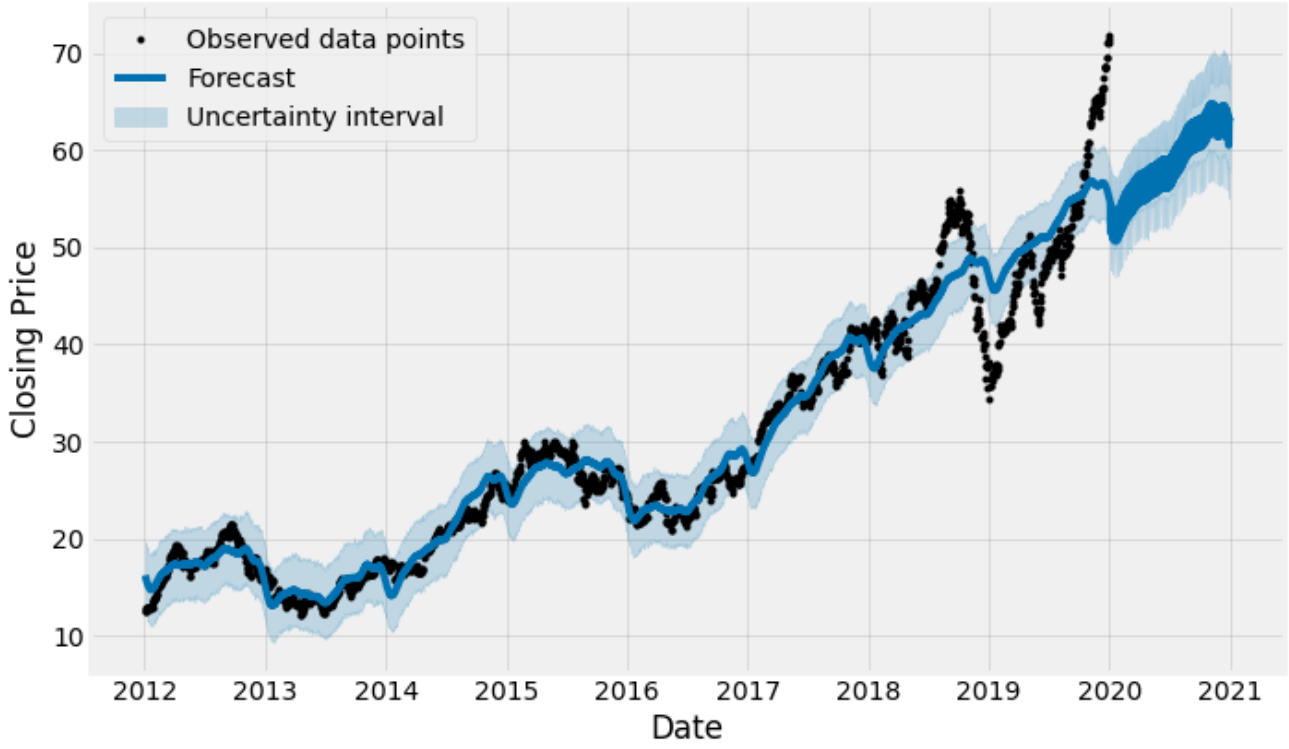
```
model = Prophet()
model.fit(df)
future = model.make_future_dataframe(365) #we need to specify the number of days in futur
e as parameter.
forecast = model.predict(future)
```

```
12:09:47 - cmdstanpy - INFO - Chain [1] start processing
12:09:48 - cmdstanpy - INFO - Chain [1] done processing
```

In [44]:

```
### Visualising the Forecasted Prices
plt.figure(figsize=(10,7))
model.plot(forecast)
plt.legend()
plt.xlabel("Date")
plt.ylabel("Closing Price")
plt.show()
```

```
<Figure size 720x504 with 0 Axes>
```



Using ARIMA (Autoregressive Integrated Moving Average) Model:

- AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.

- I: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.

- MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

- **p: The number of lag observations included in the model, also called the lag order.**
- **d: The number of times that the raw observations are differenced, also called the degree of differencing.**
- **q: The size of the moving average window, also called the order of moving average.**

Further these parameters can be tuned as per the reader's wish.

In [45]:

```
### Splitting the data
train, test = data["Close"][0:int(len(data)*0.8)], data["Close"][int(len(data)*0.8):]
```

In [46]:

```
history = [x for x in train]   ### making a list and updating it continously
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    y_hat = output[0]
    predictions.append(y_hat)
    obs = test[t]
    history.append(obs)
    #print('predicted=%f, expected=%f' % (y_hat, obs)) ## To cross verify our outputs
```

In [47]:

```
er1 = mean_squared_error(test, predictions)
print(f'\nMean Squared Error using ARIMA: {er1} \n')
er2 = mape(test, predictions)
print(f'Mean Absolute Percentage Error using ARIMA: {er2}\n')
```

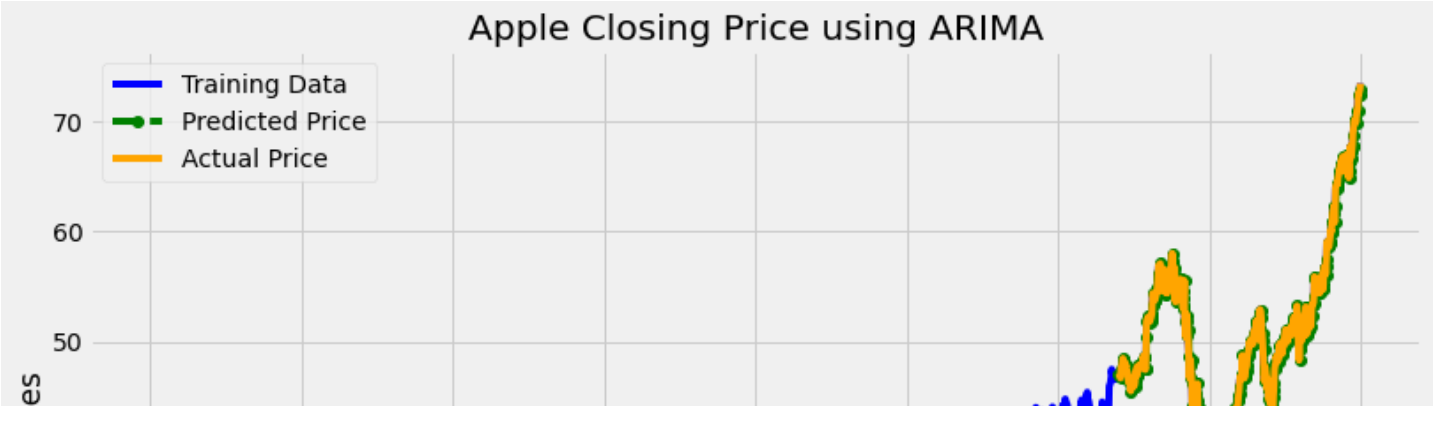
```
Mean Squared Error using ARIMA: 0.7415347245065128

Mean Absolute Percentage Error using ARIMA: 0.012482131557971584
```

In [48]:

```
plt.figure(figsize=(12,7))
plt.plot(data['Close'], 'green', color='blue', label='Training Data')
plt.plot(test.index, predictions, color='green', marker='o', linestyle='dashed',
         label='Predicted Price')
plt.plot(test.index, test.values, color='orange', label='Actual Price')
plt.title('Apple Closing Price using ARIMA')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
```

Out[48]:

```
<matplotlib.legend.Legend at 0x7fbe34564850>
```

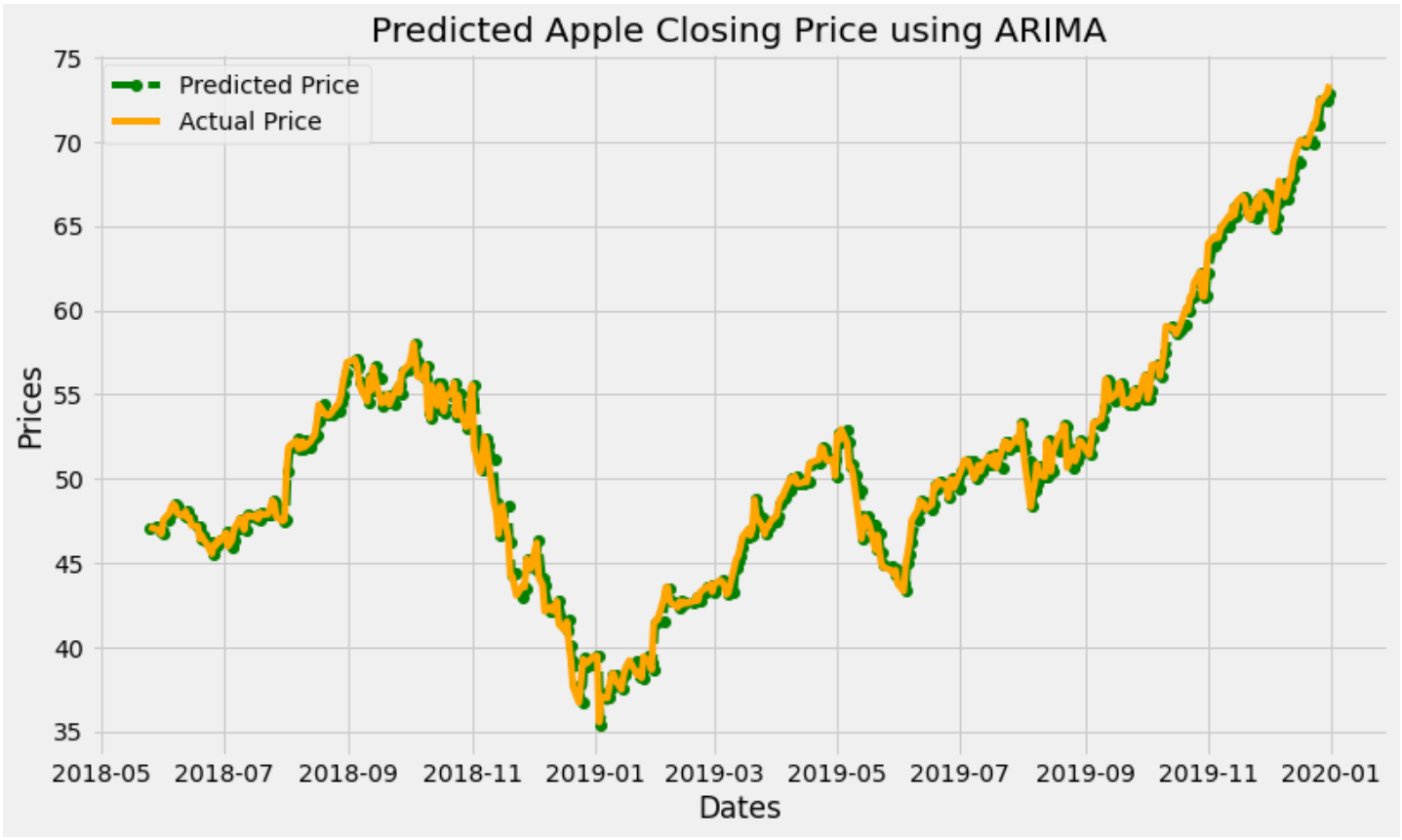


In [49]:

```
#### Here's a zoomed up version of the previous plot
plt.figure(figsize=(12,7))
plt.plot(test.index, predictions, color='green', marker='o', linestyle='dashed',
         label='Predicted Price')
plt.plot(test.index, test.values, color='orange', label='Actual Price')
plt.title('Predicted Apple Closing Price using ARIMA')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
```

Out[49]:

```
<matplotlib.legend.Legend at 0x7fb9c488ef10>
```



Looks like ARIMA performs well!

End Notes::

I enjoyed working on this mini forecasting experiment and hope the readers get a sense of what I have conveyed through my notebook. Happy experimenting!