Tesla Stock Price Prediction using Keras LSTM Model

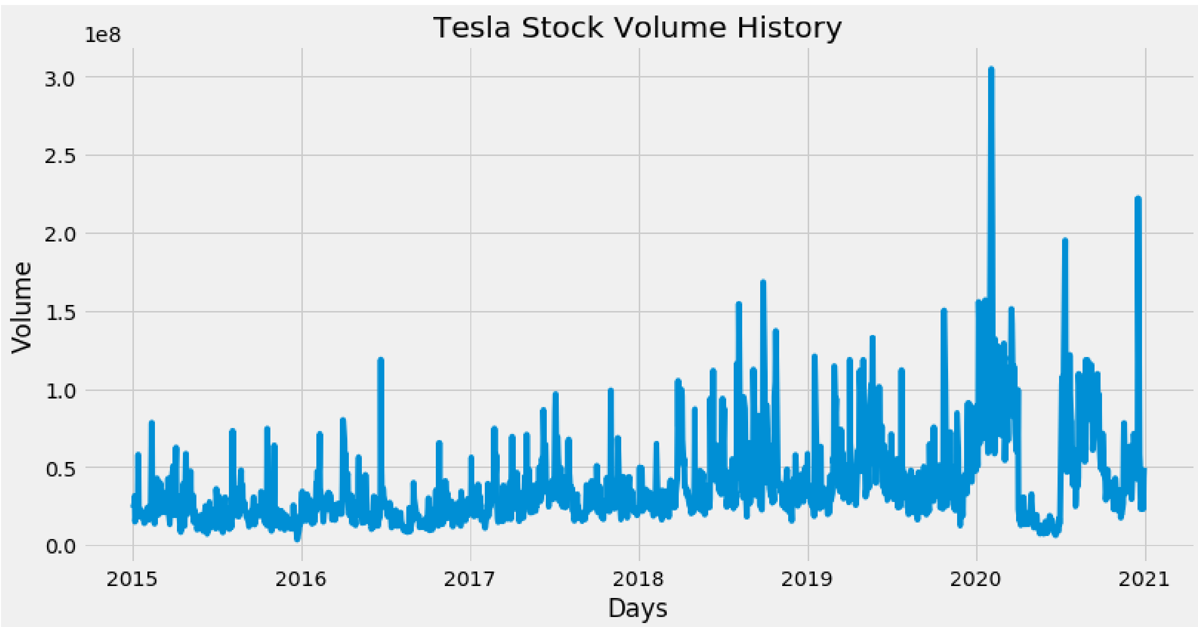
In [15]:
```
import math
import pandas_datareader as web
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import *
from sklearn.preprocessing import MinMaxScaler
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

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

In [16]:
```
# Get the stock quote from July 2015 to December 2020
# Pulled data from Yahoo Finance
df = web.DataReader('TSLA', data_source = 'yahoo', start = '2015-01-01',
end = '2020-12-31' )
print('Number of rows and columns: ', df.shape)
print(df.head(5))
print("checking if any null values are present\n", df.isna().sum())
```

```
Number of rows and columns:  (1511, 6)
                 High        Low       Open      Close      Volume   Adj
Close
Date
2015-01-02  44.650002  42.652000  44.574001  43.862000  23822000.0  43.
862000
2015-01-05  43.299999  41.431999  42.910000  42.018002  26842500.0  42.
018002
2015-01-06  42.840000  40.841999  42.012001  42.256001  31309500.0  42.
256001
2015-01-07  42.956001  41.956001  42.669998  42.189999  14842000.0  42.
189999
2015-01-08  42.759998  42.001999  42.562000  42.124001  17212500.0  42.
124001
checking if any null values are present
 High         0
Low          0
Open         0
Close        0
Volume       0
Adj Close    0
dtype: int64
```

In [18]:
```
plt.figure(figsize = (12,6))
plt.plot(df["Open"])
plt.plot(df["High"])
plt.plot(df["Low"])
plt.plot(df["Close"])
plt.title('Tesla Stock Price History')
plt.ylabel('Price (USD)')
plt.xlabel('Days')
plt.legend(['Open','High','Low','Close'], loc='upper left')
plt.show()
```

In [19]:
```
plt.figure(figsize = (12,6))
plt.plot(df["Volume"])
plt.title('Tesla Stock Volume History')
plt.ylabel('Volume')
plt.xlabel('Days')
plt.show()
```



In [54]:
```
# Create a dataframe with only the Close Stock Price Column
# Target Variable: Close stock price value
data_target = df.filter(['Close'])

# Convert the dataframe to a numpy array to train the LSTM model
target = data_target.values

# Training set has 75% of the data
training_data = math.ceil(len(target)* 0.75)
training_data
```

Out[54]: 1134

In [55]:
```
# Normalizing data before model fitting using MinMaxScaler
# Feature Scaling

sc = MinMaxScaler(feature_range=(0,1))
training_scaled_data = sc.fit_transform(target)
training_scaled_data

# Create a training dataset containing the last 180-day closing price va
lues we want to use to estimate the 181st closing price value.
train_data = training_scaled_data[0:training_data  , : ]

X_train = []
y_train = []
for i in range(180, len(train_data)):
    X_train.append(train_data[i-180:i, 0])
    y_train.append(train_data[i, 0])

X_train, y_train = np.array(X_train), np.array(y_train) # converting int
o numpy sequences to train the LSTM model

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

print('Number of rows and columns: ', X_train.shape)  #(954 values, 180
 time-steps, 1 output)
```

```
Number of rows and columns:  (954, 180, 1)
```

Building the LSTM Model

In [31]:
```
# We add the LSTM layer and later add a few Dropout layers to prevent ov
erfitting.
# Building a LTSM model with 50 neurons and 4 hidden layers. We add the
 LSTM layer with the following arguments:
# 50 units which is the dimensionality of the output space
# return_sequences=True which determines whether to return the last outp
ut in the output sequence, or the full sequence input_shape as the shape
of our training set.
# When defining the Dropout layers, we specify 0.2, meaning that 20% of
 the layers will be dropped.
# Thereafter, we add the Dense layer that specifies the output of 1 uni
t.
# After this, we compile our model using the popular adam optimizer and
 set the loss as the mean_squarred_error.

model = Sequential()

#Adding the first LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_tra
in.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')

print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 180, 50)           10400     
_________________________________________________________________
dropout (Dropout)            (None, 180, 50)           0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 180, 50)           20200     
_________________________________________________________________
dropout_1 (Dropout)          (None, 180, 50)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 180, 50)           20200     
_________________________________________________________________
dropout_2 (Dropout)          (None, 180, 50)           0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                20200     
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0         
_________________________________________________________________
dense (Dense)                (None, 1)                 51        
=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
None
```

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

```
Epoch 1/100
30/30 [==============================] - 14s 264ms/step - loss: 5.2260e
-04
Epoch 2/100
30/30 [==============================] - 8s 259ms/step - loss: 1.0762e-
04
Epoch 3/100
30/30 [==============================] - 8s 262ms/step - loss: 7.1276e-
05
Epoch 4/100
30/30 [==============================] - 8s 253ms/step - loss: 7.0829e-
05
Epoch 5/100
30/30 [==============================] - 9s 315ms/step - loss: 6.0678e-
05
Epoch 6/100
30/30 [==============================] - 13s 431ms/step - loss: 5.2245e
-05
Epoch 7/100
30/30 [==============================] - 12s 403ms/step - loss: 5.0562e
-05
Epoch 8/100
30/30 [==============================] - 11s 356ms/step - loss: 4.7700e
-05
Epoch 9/100
30/30 [==============================] - 10s 337ms/step - loss: 5.9467e
-05
Epoch 10/100
30/30 [==============================] - 8s 276ms/step - loss: 4.4363e-
05
Epoch 11/100
30/30 [==============================] - 8s 260ms/step - loss: 4.8537e-
05
Epoch 12/100
30/30 [==============================] - 8s 263ms/step - loss: 5.0357e-
05
Epoch 13/100
30/30 [==============================] - 8s 258ms/step - loss: 4.3941e-
05
Epoch 14/100
30/30 [==============================] - 8s 269ms/step - loss: 4.6094e-
05
Epoch 15/100
30/30 [==============================] - 8s 258ms/step - loss: 4.5770e-
05
Epoch 16/100
30/30 [==============================] - 8s 259ms/step - loss: 3.7223e-
05
Epoch 17/100
30/30 [==============================] - 8s 258ms/step - loss: 3.7279e-
05
Epoch 18/100
30/30 [==============================] - 8s 257ms/step - loss: 3.5425e-
05
Epoch 19/100
30/30 [==============================] - 8s 258ms/step - loss: 3.5626e-
05
```

```
Epoch 20/100
30/30 [==============================] - 8s 255ms/step - loss: 3.2626e-
05
Epoch 21/100
30/30 [==============================] - 8s 267ms/step - loss: 3.3728e-
05
Epoch 22/100
30/30 [==============================] - 8s 257ms/step - loss: 3.3163e-
05
Epoch 23/100
30/30 [==============================] - 8s 258ms/step - loss: 3.1376e-
05
Epoch 24/100
30/30 [==============================] - 8s 256ms/step - loss: 2.9796e-
05
Epoch 25/100
30/30 [==============================] - 8s 256ms/step - loss: 3.2208e-
05
Epoch 26/100
30/30 [==============================] - 8s 254ms/step - loss: 2.8164e-
05
Epoch 27/100
30/30 [==============================] - 8s 269ms/step - loss: 2.8135e-
05
Epoch 28/100
30/30 [==============================] - 8s 258ms/step - loss: 2.7968e-
05
Epoch 29/100
30/30 [==============================] - 11s 353ms/step - loss: 3.0517e
-05
Epoch 30/100
30/30 [==============================] - 10s 337ms/step - loss: 2.6992e
-05
Epoch 31/100
30/30 [==============================] - 9s 308ms/step - loss: 2.6708e-
05
Epoch 32/100
30/30 [==============================] - 8s 253ms/step - loss: 2.4510e-
05
Epoch 33/100
30/30 [==============================] - 9s 295ms/step - loss: 2.3994e-
05
Epoch 34/100
30/30 [==============================] - 8s 255ms/step - loss: 2.2424e-
05
Epoch 35/100
30/30 [==============================] - 8s 261ms/step - loss: 2.2365e-
05
Epoch 36/100
30/30 [==============================] - 8s 261ms/step - loss: 2.4429e-
05
Epoch 37/100
30/30 [==============================] - 8s 255ms/step - loss: 2.7228e-
05
Epoch 38/100
30/30 [==============================] - 8s 254ms/step - loss: 2.1792e-
05
```

```
Epoch 39/100
30/30 [==============================] - 8s 252ms/step - loss: 2.4017e-
05
Epoch 40/100
30/30 [==============================] - 8s 252ms/step - loss: 2.1893e-
05
Epoch 41/100
30/30 [==============================] - 8s 253ms/step - loss: 2.7295e-
05
Epoch 42/100
30/30 [==============================] - 8s 253ms/step - loss: 2.5664e-
05
Epoch 43/100
30/30 [==============================] - 8s 253ms/step - loss: 2.1950e-
05
Epoch 44/100
30/30 [==============================] - 8s 256ms/step - loss: 2.7168e-
05
Epoch 45/100
30/30 [==============================] - 8s 253ms/step - loss: 2.2807e-
05
Epoch 46/100
30/30 [==============================] - 8s 255ms/step - loss: 2.2683e-
05
Epoch 47/100
30/30 [==============================] - 8s 254ms/step - loss: 1.8586e-
05
Epoch 48/100
30/30 [==============================] - 8s 252ms/step - loss: 1.9996e-
05
Epoch 49/100
30/30 [==============================] - 8s 253ms/step - loss: 2.2624e-
05
Epoch 50/100
30/30 [==============================] - 8s 260ms/step - loss: 2.2477e-
05
Epoch 51/100
30/30 [==============================] - 8s 263ms/step - loss: 1.9122e-
05
Epoch 52/100
30/30 [==============================] - 8s 251ms/step - loss: 1.8267e-
05
Epoch 53/100
30/30 [==============================] - 8s 271ms/step - loss: 2.0033e-
05
Epoch 54/100
30/30 [==============================] - 9s 306ms/step - loss: 1.8415e-
05
Epoch 55/100
30/30 [==============================] - 9s 289ms/step - loss: 1.9451e-
05
Epoch 56/100
30/30 [==============================] - 8s 257ms/step - loss: 1.8010e-
05
Epoch 57/100
30/30 [==============================] - 8s 253ms/step - loss: 1.9404e-
05
```

```
Epoch 58/100
30/30 [==============================] - 8s 253ms/step - loss: 1.9353e-
05
Epoch 59/100
30/30 [==============================] - 8s 253ms/step - loss: 1.6728e-
05
Epoch 60/100
30/30 [==============================] - 8s 253ms/step - loss: 1.8122e-
05
Epoch 61/100
30/30 [==============================] - 8s 256ms/step - loss: 2.0174e-
05
Epoch 62/100
30/30 [==============================] - 8s 256ms/step - loss: 1.5961e-
05
Epoch 63/100
30/30 [==============================] - 8s 253ms/step - loss: 1.7559e-
05
Epoch 64/100
30/30 [==============================] - 8s 255ms/step - loss: 1.5870e-
05
Epoch 65/100
30/30 [==============================] - 9s 293ms/step - loss: 1.5623e-
05
Epoch 66/100
30/30 [==============================] - 8s 260ms/step - loss: 1.5225e-
05
Epoch 67/100
30/30 [==============================] - 8s 259ms/step - loss: 1.8071e-
05
Epoch 68/100
30/30 [==============================] - 10s 316ms/step - loss: 1.9109e
-05
Epoch 69/100
30/30 [==============================] - 8s 260ms/step - loss: 1.6036e-
05
Epoch 70/100
30/30 [==============================] - 7s 249ms/step - loss: 1.4698e-
05
Epoch 71/100
30/30 [==============================] - 7s 248ms/step - loss: 1.8109e-
05
Epoch 72/100
30/30 [==============================] - 7s 248ms/step - loss: 1.5596e-
05
Epoch 73/100
30/30 [==============================] - 7s 248ms/step - loss: 1.5694e-
05
Epoch 74/100
30/30 [==============================] - 8s 258ms/step - loss: 1.6740e-
05
Epoch 75/100
30/30 [==============================] - 8s 252ms/step - loss: 1.7133e-
05
Epoch 76/100
30/30 [==============================] - 8s 250ms/step - loss: 1.8114e-
05
```

```
Epoch 77/100
30/30 [==============================] - 7s 246ms/step - loss: 1.6072e-
05
Epoch 78/100
30/30 [==============================] - 7s 247ms/step - loss: 1.7967e-
05
Epoch 79/100
30/30 [==============================] - 7s 248ms/step - loss: 1.4216e-
05
Epoch 80/100
30/30 [==============================] - 8s 255ms/step - loss: 1.3535e-
05
Epoch 81/100
30/30 [==============================] - 7s 248ms/step - loss: 1.5954e-
05
Epoch 82/100
30/30 [==============================] - 8s 257ms/step - loss: 1.6682e-
05
Epoch 83/100
30/30 [==============================] - 7s 248ms/step - loss: 1.6367e-
05
Epoch 84/100
30/30 [==============================] - 8s 267ms/step - loss: 1.9696e-
05
Epoch 85/100
30/30 [==============================] - 7s 248ms/step - loss: 1.7110e-
05
Epoch 86/100
30/30 [==============================] - 7s 247ms/step - loss: 1.6747e-
05
Epoch 87/100
30/30 [==============================] - 7s 245ms/step - loss: 1.4419e-
05
Epoch 88/100
30/30 [==============================] - 8s 265ms/step - loss: 1.4058e-
05
Epoch 89/100
30/30 [==============================] - 8s 258ms/step - loss: 1.3385e-
05
Epoch 90/100
30/30 [==============================] - 8s 270ms/step - loss: 1.4339e-
05
Epoch 91/100
30/30 [==============================] - 8s 252ms/step - loss: 1.5357e-
05
Epoch 92/100
30/30 [==============================] - 8s 255ms/step - loss: 1.6098e-
05
Epoch 93/100
30/30 [==============================] - 8s 256ms/step - loss: 1.4547e-
05
Epoch 94/100
30/30 [==============================] - 7s 248ms/step - loss: 1.5370e-
05
Epoch 95/100
30/30 [==============================] - 8s 280ms/step - loss: 1.5169e-
05
```

```
Epoch 96/100
30/30 [==============================] - 8s 259ms/step - loss: 1.1573e-
05
Epoch 97/100
30/30 [==============================] - 8s 257ms/step - loss: 1.2481e-
05
Epoch 98/100
30/30 [==============================] - 8s 255ms/step - loss: 1.4114e-
05
Epoch 99/100
30/30 [==============================] - 8s 255ms/step - loss: 1.4337e-
05
Epoch 100/100
30/30 [==============================] - 8s 263ms/step - loss: 1.4722e-
05
```

Out[32]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff26efc6310>
```

In [59]:
```
# Getting the predicted stock price
# Create the x_test and y_test data sets

test_data = training_scaled_data[training_data - 180: , : ]

X_test = []
y_test =  target[training_data : , : ]
for i in range(180,len(test_data)):
    X_test.append(test_data[i-180:i,0])

# Convert x_test to a numpy array
X_test = np.array(X_test)

#Reshape the data into the shape accepted by the LSTM
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))

print('Number of rows and columns: ', X_test.shape)
```

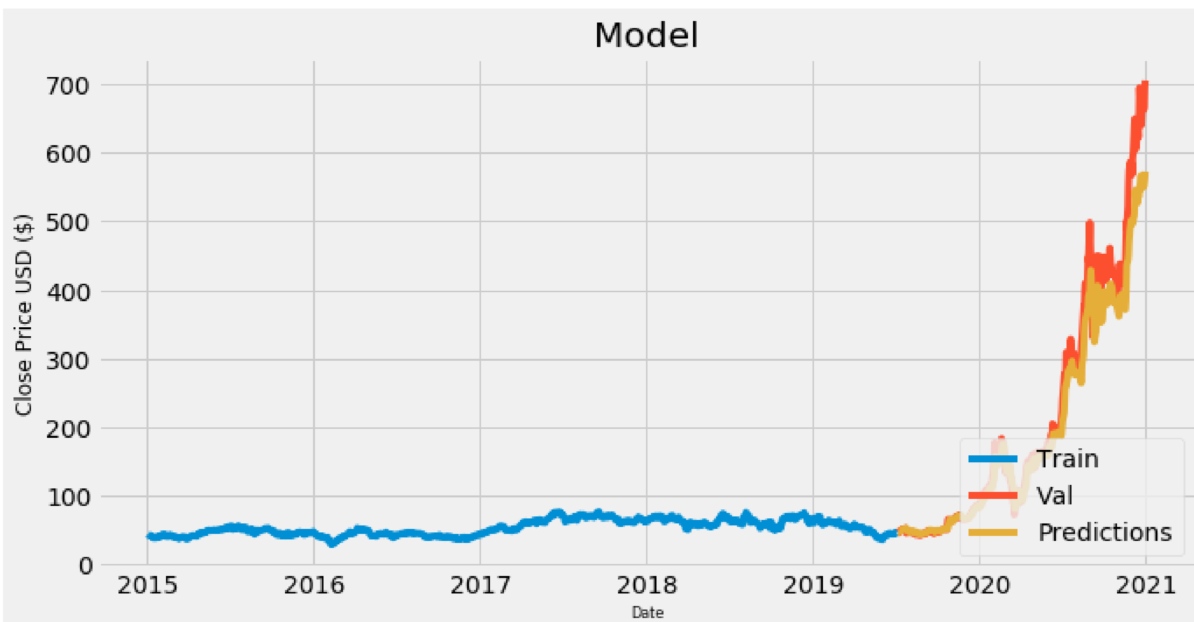
```
Number of rows and columns:  (377, 180, 1)
```

Predictions

In [69]:
```
# Making predictions using the test dataset
predicted_stock_price = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

In [72]:
```
#Create the data for the graph
train = data_target[:training_data]
valid = data_target[training_data:]
valid['Predictions'] = predicted_stock_price

#Visualize the data
plt.figure(figsize=(10,5))
plt.title('Model')
plt.xlabel('Date', fontsize=8)
plt.ylabel('Close Price USD ($)', fontsize=12)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show();
```

```
/Users/shimonyagrawal/opt/anaconda3/lib/python3.7/site-packages/ipykern
el_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```



In [73]:
```
# Show valid and predicted prices

valid
```

Out[73]:

Date	Close	Predictions
2019-07-08	46.068001	47.431995
2019-07-09	46.012001	47.384583
2019-07-10	47.784000	47.103020
2019-07-11	47.720001	47.618355
2019-07-12	49.015999	48.234806
...	...	...
2020-12-24	661.770020	548.692810
2020-12-28	663.690002	551.229858
2020-12-29	665.989990	556.942139
2020-12-30	694.780029	561.622437
2020-12-31	705.669983	572.838196

377 rows × 2 columns