MANHATTAN DATA - LSTM - Multi-Step Forecast - Vector Output Model

Here I have done the following:

1. Followed steps from this website: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ (https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/)
2. Import necessary modules
3. Fixed the parameters of the code to take in input of previous 60 days and output the next 30 days
    - n_steps_in = 60
    - n_steps_out = 30
4. Define the model and predict 30 days of data
5. Note any observations

In [2]:

```
# Imports
import numpy as np
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
import pandas as pd
#Supress default INFO logging
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)
import logging, sys
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [3]:

```
df = pd.read_csv('datasets/rollingsales_manhattan.xls_prepped_bare.csv', usecols
```

In [4]:

```
df = df.dropna()
df = df.reset_index(drop=True)
```

In [5]:

```
df = df.rename(columns={'SALE DATE':'ts', 'SALE PRICE': 'y'})
df.columns = df.columns.astype(str)
df = df.set_index(['ts'], drop=True)
df.index= pd.to_datetime(df.index)
```

In [6]:

```
# df
```

In [7]:

```
df = df.resample('D').mean()
df = df.reset_index()
```

In [8]:
```
df.dropna(inplace=True)
df
```

Out[8]:

	ts	y
0	2020-04-01	2.651838e+06
1	2020-04-02	1.899093e+06
2	2020-04-03	2.315087e+06
3	2020-04-04	1.369242e+06
5	2020-04-06	7.843903e+06
...	...	...
358	2021-03-25	3.024555e+06
359	2021-03-26	2.232141e+06
362	2021-03-29	1.530709e+06
363	2021-03-30	1.889714e+06
364	2021-03-31	6.265608e+06

270 rows × 2 columns

In [9]:
```
raw_input_test = list(df['y'])
raw_input_test
np.shape(df.index)
```

Out[9]:
```
(270,)
```

Below steps are taken from:

https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ (https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/)

In [10]:
```
# split a univariate sequence into samples
def split_sequence(sequence, n_steps_in, n_steps_out):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out
        # check if we are beyond the sequence
        if out_end_ix > len(sequence):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix:out_end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

# define input sequence
raw_seq = raw_input_test

# choose a number of time steps
n_steps_in, n_steps_out = 60, 30

# split into samples
X, y = split_sequence(raw_seq, n_steps_in, n_steps_out)


# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))


# define model
model = Sequential()
model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(n_ste
model.add(LSTM(100, activation='relu'))
model.add(Dense(n_steps_out))
model.compile(optimizer='adam', loss='mse')
# fit model
model.fit(X, y, epochs=100, verbose=0)
```

Out[10]:
```
<tensorflow.python.keras.callbacks.History at 0x227c59ebbe0>
```

In [12]:
```
# demonstrate prediction
# x_input = array([70, 80, 90])
x_input = array(raw_input_test[210:270])
x_input = x_input.reshape((1, n_steps_in, n_features))
yhat = model.predict(x_input, verbose=0)
print(yhat)
```

```
[[ -80600.555 2118380.2   1838946.9   2679223.    1019938.4   2295004.5
   1782356.6    701355.3   3179098.8   2589158.5   2484051.8   2258199.2
   2397421.    2851734.2   1166693.6   1669791.    1569022.4   2876600.2
   2034599.8    795425.5   1614458.    2220594.2   2334062.    2004019.5
   2298593.8   1247987.5   2583888.8   1872226.6   2086669.5    613778.1  ]]
```

In [13]:
```
np.shape(list(yhat))
```

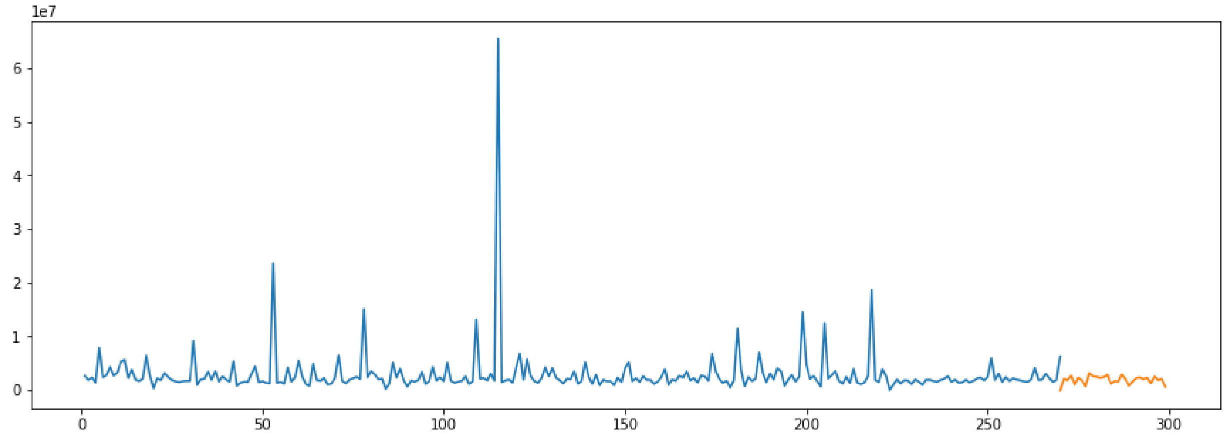
Out[13]:
```
(1, 30)
```

In [18]:
```
y_hat1 = np.reshape(yhat, (30,1))
np.shape(y_hat1)
```

Out[18]:
```
(30, 1)
```

In [19]:
```
# I increased the epochs and the predictions went higher.
x_list = list(range(1,300))
```

In [23]:
```
plt.figure(figsize=(15,5))
fig =plt.plot(x_list[0:270], df['y'][0:270])
ax = plt.plot(x_list[269:300], y_hat1)
```



Observations:

1. Predictions seem lower than the origional data for Manhattan