Applying LSTM on Log Transformed data

In [63]:
```
import pandas as pd
import itertools
import pylab as pl
import os
import itertools
import numpy as np
from matplotlib import pyplot as plt
import matplotlib
import os
from datetime import datetime
%matplotlib inline
import warnings
import statsmodels.api as sm
warnings.filterwarnings('ignore')
from matplotlib import dates
from statsmodels.tsa.stattools import adfuller
from sklearn.preprocessing import MinMaxScaler
import pickle
from datetime import datetime
```

In [17]:
```
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.models import Sequential
from keras.layers import Dense,Flatten
from keras.layers import LSTM
from keras.models import load_model
```

In [22]:
```
def get_files_from_dir(dir):
    '''
        This function checks if files are atleast having 2 years of data
    '''
    print(get_files_from_dir.__doc__)
    ls = os.listdir(dir)
    print('\nTotal files in folder: ', len(ls))
    pairs = []
    good_data = []
    good_data_names = []
    locations = []
    count = 0
    for file_n in ls:
        count += 1
        if dir == './unique_df/':
            location = dir + file_n + '/' + file_n + '.pkl'
        else:
            location = dir + file_n
        a = pd.read_pickle(location)
        good_data.append(a)
        good_data_names.append(file_n)
        locations.append(location)
        pairs.append((file_n, location))

    return good_data, good_data_names, locations, pairs
```

In [65]:
```
start = datetime.now()
good_data_log, good_names_log, good_paths_log, good_pairs_log = get_files_from_dir("./log_df/")
print('Time taken: ', datetime.now() - start)
```

```
        This function checks if files are atleast having 2 years of data


Total files in folder:  208
Time taken:  0:00:00.242113
```

In [30]:
```
def mape(y_true, y_pred):
    '''
        Mean Absolute Percentage Error
    '''
    if len(y_true) != len(y_pred):
        print('true and predicted values are not of same length')

    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

In [4]:
```
def load_models(dir_path):
    directory = dir_path

    list = os.listdir(directory)
    Locations=[]

    for file in list:
        # Use join to get full file path.
        location = os.path.join(directory, file)
        Locations.append(location)

    return Locations
```

In [1]:
```
def build_lstm(n_input, n_features,generator,path):
    '''
        This function contains our model Architecture and trains our model
        it returns our model which will later be used to predict our test data
    '''
    model = Sequential()
    model.add(LSTM(100, activation='relu', input_shape=(n_input, n_features), return_sequences=True))
    model.add(Flatten())
    model.add(Dense(1)) #output layer
    model.compile(optimizer='adam', loss='mape') # our Performance Metric has been set to MAPE
    model.fit_generator(generator, epochs=30, verbose=0)

    return model
```

Observations

- If we look at our model architecture, we have 100 LSTM units
- I have used a simple Sequential Architecture since we have no use of Functional API Architecture for this problem
- Output dense layer just consists of 1 unit as we are only taking one output which is our Predcitive Sales value
- I am running my model for 30 epochs and verbose is set to 0 as there will be a clutter of printing done

In [43]:
```
def predict(model, path, scaled_train, test, scaler):
    '''
        In this function we are predicting the values for our test data and
        storing it in a predictions folder so that we can calculate their MAPE
        at once in the future
    '''
    test_predictions = []
    n_input = 12
    n_features = 1
    first_eval_batch = scaled_train[-n_input:]
    current_batch = first_eval_batch.reshape((1, n_input, n_features))
    for i in range(len(test)):
        # get prediction 1 time stamp ahead ([0] is for grabbing just the number instead of [array])
        current_pred = model.predict(current_batch)[0]

        # store prediction
        test_predictions.append(current_pred)

        # update batch to now include prediction and drop first value
        current_batch = np.append(current_batch[:, 1:, :], [[current_pred]], axis=1)

    true_predictions = scaler.inverse_transform(test_predictions)
    test['Predictions'] = true_predictions

    path = path.replace('./final_data/', '')
    path = path.replace('.pkl', '')
    test.to_pickle("./predictions/"+path)
```

Observations

- I have created a separate predict function instead of creating a validation generator is due to the fact that i wanted to see my predicted values for each unique combination instead of just looking at the performance metric
- Each unique combination's predicted values have been stored in "predictions" folder
- I use data from that folder in the end to calculate my performance metric

In [67]:
```
def run_lstm(paths):
    '''
        This is the main function which trains our Architecture and predicts our test data
    '''
    cnt = 0
    for path in paths:
#         print(path)
        df = pd.read_pickle(path)
        #df['Actual_Sales'].fillna(0, inplace=True)
        df.dropna(inplace=True)
        df.drop(['Region', 'Division_Name','Facility_Name','Year','Month'], axis=1, inplace=True)
        len_df = len(df)

        # train test split
        train = df.iloc[:len_df-12]
        test = df.iloc[len_df-12:]

        scaler = MinMaxScaler()
        scaler.fit(train)

        scaled_train = scaler.transform(train)
        scaled_test = scaler.fit_transform(test)

        n_input = 12
        n_features = 1
        generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_size=1)
        model = build_lstm(n_input, n_features, generator, path)

        predict(model, path, scaled_train, test, scaler)
        cnt = cnt+1

    print('Total Files trained and predicted: ', cnt)
```

Observations

- As we can see the above function runs the whole workflow from training, prediction and storing our predictions

In [61]:
```
start = datetime.now()
run_lstm(good_paths_log)
print('\nTime taken: ', datetime.now() - start)
```

```
Total Files trained and predicted:  208

Time taken: 04:36:376648
```

In [68]:
```
start = datetime.now()
good_data_pred, good_names_pred, good_paths_pred, good_pairs_pred = get_files_from_dir("./predictions/")
print('Time taken: ', datetime.now() - start)
```

```
        This function checks if files are atleast having 2 years of data


Total files in folder:  208
Time taken:  0:00:00.207457
```

In [69]:
```
good_paths_pred[0]
```

Out[69]:
```
'./predictions/APG ATLANTA_Oldcastle Retail CMP999820_EZ Mix'
```

Now let's look at one of the predictions

In [74]:
```
# this is how our predicted data looks like for one file
df_pred = pd.read_pickle(good_paths_pred[0])
print('Name of the file: ', good_names_pred[0])
print('\nIt\'s predicted values: ')
df_pred
```

```
Name of the file:  APG ATLANTA_Oldcastle Retail CMP999820_EZ Mix

It's predicted values:
```

Out[74]:

Dates	Actual_Sales	Predictions
2018-12-01	10.459583	8.844612
2019-01-01	8.475538	9.360433
2019-02-01	10.455446	9.769975
2019-03-01	10.118962	9.874326
2019-04-01	11.002000	9.917007
2019-05-01	11.002000	9.883842
2019-06-01	10.154713	9.904720
2019-07-01	10.429133	9.974070
2019-08-01	10.415712	9.829557
2019-09-01	10.100616	9.555475
2019-10-01	10.442376	9.265079
2019-11-01	9.001962	8.704635

In [35]:
```
def cal_mape(names, paths):
    '''
        Calculates Mean Absolute Percentage Error of all predicted files
    '''
    di = {}
    for i in range(len(paths)):
        df = pd.read_pickle(paths[i])
        loss_val = mape(df.Actual_Sales, df.Predictions)
        di[names[i]] = loss_val

    return di
```

In [38]:
```
lstm_di_log = cal_mape(good_names_pred, good_paths_pred)
```

In [39]:
```
lstm_di_log
```

Out[39]:
```
{'APG ATLANTA_Oldcastle Retail CMP999820_EZ Mix': 7.275538072449193,
 'Canada_Expocrete_Acheson': 6.295071619496996,
 'Canada_Expocrete_Balzac': 11.786179965515238,
 'Canada_Expocrete_Edmonton': 5.447160279887716,
 'Canada_Expocrete_Richmond': 2.3374634844182616,
 'Canada_Expocrete_Saskatoon': 42.793731040844435,
 'Canada_Expocrete_Winnipeg': 7.140858879124003,
 'Canada_Permacon_Milton ON': 3.531503070920213,
 'Canada_Permacon_Montreal QC': 5.375592566376957,
 'Canada_Permacon_Woodstock Ontario': 67.65861321722801,
 'Central_Ash Grove MPC_Fort Smith, AR': 3.1288950720221997,
 'Central_Ash Grove MPC_Fremont, NE': 2.4756086312135666,
 'Central_Ash Grove MPC_Harrisonville, MO': 6.409477537179509,
 'Central_Ash Grove MPC_Jackson, MS': 5.18030676160578,
 'Central_Ash Grove MPC_Memphis, TN': 2.4707007109319745,
 'Central_Ash Grove MPC_Muskogee, OK': 3.8464527445800263,
 'Central_Ash Grove MPC_North Little Rock, AR': 7.677955296471382,
 'Central_Ash Grove MPC_Oklahoma City, OK': 6.597278276278649,
 'Central_Jewell_Austin TX (S)': 6.817809018107406,
 'Central_Jewell_Brittmoore': 2.8330044912285754,
 'Central_Jewell_Dallas TX (S)': 7.369895314637757,
 'Central_Jewell_Frisco TX (S)': 1.8844430447638387,
 'Central_Jewell_Houston TX - West Hardy (M)': 6.4987501919327615,
 'Central_Jewell_Houston TX-N Garden': 5.969998308891916,
 'Central_Jewell_Hurst TX-SAK': 2.568928706367802,
 'Central_Jewell_IBC TX-SAK': 5.170537784991266,
 'Central_Jewell_Katy TX-SAK': 1.5069374227456795,
 'Central_Jewell_Keller TX (S)': 9.709101686380658,
 'Central_Jewell_Marble Falls (SAK)': 4.859562783836571,
 'Central_Jewell_Rosenberg TX': 5.166902660501991,
 'Central_Jewell_Waco TX': 10.368502408151354,
 'Central_Northfield_Bridgeport MI': 6.267926589901061,
 'Central_Northfield_Cincinnati OH-SAK': 1.2791445840056972,
 'Central_Northfield_Forest View IL': 2.4698562893767746,
 'Central_Northfield_Franklin Park IL-SAK': 1.2235364580366024,
 'Central_Northfield_Indianapolis IN': 8.432557149588483,
 'Central_Northfield_Miller Materials KC Plant': 5.868451197653956,
 'Central_Northfield_Miller MaterialsBonner Springs': 3.3279169322657793,
 'Central_Northfield_Morris IL': 2.680668582722849,
 'Central_Northfield_Mundelein IL': 6.989100217148724,
 'Central_Northfield_Shakopee': 9.785718472425355,
```

```
'Central_Northfield_Sheffield OH': 10.979355528004834,
'Central_Northfield_West Des Moines IA': 6.716706641324592,
'Central_Northfield_Wisconsin Distribution Yard': 5.03637601485567,
'East_Adams Products_Anderson SC': 2.66631508846665,
'East_Adams Products_Asheville NC': 6.624120289044518,
'East_Adams Products_Charlotte NC': 1.644846589783523,
'East_Adams Products_Charlotte NC Plant': 1.51000999647456,
'East_Adams Products_Clarksville TN': 1.629907685078737,
'East_Adams Products_Colfax NC': 2.363831960565878,
'East_Adams Products_Cowpens SC': 3.4066712479713037,
'East_Adams Products_Dunn NC': 4.76823567228056,
'East_Adams Products_Durham NC': 3.1607524095812445,
'East_Adams Products_Fayetteville NC': 7.019244250900701,
'East_Adams Products_Franklin NC': 1.424433765271397,
'East_Adams Products_Franklin TN-SAK': 1.3754190629005374,
'East_Adams Products_Goldsboro NC': 2.982916336268882,
'East_Adams Products_Greensboro NC': 6.813821141008471,
'East_Adams Products_Greenville NC': 2.117731943616705,
'East_Adams Products_Greenville SC': 1.627094957866457,
'East_Adams Products_Hickory NC': 2.5592644757285883,
'East_Adams Products_HollyHill SC': 3.6497739173907155,
'East_Adams Products_Inman SC': 1.5349429297007424,
'East_Adams Products_Jacksonville NC': 1.6454902689371183,
'East_Adams Products_Lilesville NC-SAK': 1.161121748387979,
'East_Adams Products_Morehead NC': 4.166771082299547,
'East_Adams Products_Morrisville NC': 1.1001355762899963,
'East_Adams Products_Myrtle Beach SC': 6.203358135066357,
'East_Adams Products_Nashville TN': 15.671204316844348,
'East_Adams Products_Rockwood TN': 2.1923489276181307,
'East_Adams Products_Rocky Mount NC': 6.535002671860965,
'East_Adams Products_Stallings NC': 4.229156143796898,
'East_Adams Products_Wilmington NC': 6.12485272249106,
'East_Adams Products_Youngsville NC': 1.390912373663109,
'East_Anchor_Anchor South Region': 23.625472687097517,
'East_Anchor_Batavia NY-SAK': 1.451940726639264,
'East_Anchor_Brick NJ': 2.313175141207682,
'East_Anchor_Bristol PA-SAK': 1.5856898527604608,
'East_Anchor_Calverton NY-SAK': 1.4752221067622329,
'East_Anchor_Canaan CT-SAK': 0.9133103825690279,
'East_Anchor_Cranston RI': 3.3672176181072766,
'East_Anchor_Crofton MD': 2.9149084748221488,
'East_Anchor_East Petersburg PA': 1.9341646660903822,
```

'East_Anchor_Easton PA': 7.951432277450903,
'East_Anchor_Emigsville PA': 3.8455784234721726,
'East_Anchor_Farmingdale NJ': 7.546647035733116,
'East_Anchor_Fishers NY': 2.1050759663401584,
'East_Anchor_Fredonia PA-SAK': 8.81542360839262,
'East_Anchor_Holbrook MA': 3.2306945437260293,
'East_Anchor_Keene NH': 11.77405331516453,
'East_Anchor_Lebanon NH': 7.6476854578754025,
'East_Anchor_Lyndhurst NJ': 5.443153209404852,
'East_Anchor_Milford VA-SAK': 1.3416324374995194,
'East_Anchor_Oxford MA-SAK': 1.758702065959847,
'East_Anchor_White Marsh MD-SAK': 1.3364843085563063,
'East_Anchor_Winchester VA': 1.5158459571022542,
'East_Georgia Masonry Supply_Cartersville GA BLOCK': 2.9688468534050414,
'East_Georgia Masonry Supply_Conley GA-SAK': 1.614670965024469,
'East_Georgia Masonry Supply_Florence AL': 2.256688199246868,
'East_Georgia Masonry Supply_Jasper AL': 28.98324985717983,
'East_Georgia Masonry Supply_Jonesboro GA': 8.50860208916466,
'East_Georgia Masonry Supply_Lawrenceville GA DIST': 6.451679469422748,
'East_Georgia Masonry Supply_Lawrenceville GA MANF': 4.002954553450485,
'East_Georgia Masonry Supply_Macon GA': 1.7995014224402381,
'East_Georgia Masonry Supply_Montgomery AL': 5.3541351793415775,
'East_Georgia Masonry Supply_Pelham AL': 1.855514485937065,
'East_Georgia Masonry Supply_Tyrone GA': 1.4536635795623756,
'East_OldcastleCoastal_Auburndale FL-SAK': 5.814374670975762,
'East_OldcastleCoastal_Defuniak': 2.451803722307917,
'East_OldcastleCoastal_Fort Myers FL': 5.684742937170217,
'East_OldcastleCoastal_Fort Pierce FL': 6.342945506708631,
'East_OldcastleCoastal_Gainesville FL': 1.7121015577986378,
'East_OldcastleCoastal_Gulfport MS': 2.198668341464816,
'East_OldcastleCoastal_Haines City FL Hardscapes': 2.413726675070585,
'East_OldcastleCoastal_Jacksonville FL': 2.1978331575075027,
'East_OldcastleCoastal_Jacksonville FL-SAK': 1.3433622106215979,
'East_OldcastleCoastal_Lehigh Acres FL': 3.229307135493494,
'East_OldcastleCoastal_Longwood FL': 1.9189846879762762,
'East_OldcastleCoastal_Orlando FL': 5.15517802331461,
'East_OldcastleCoastal_Pensacola FL-SAK': 7.473568710450375,
'East_OldcastleCoastal_Pompano FL Hardscapes': 1.5528692129305113,
'East_OldcastleCoastal_Pompano FL-SAK': 4.5422209734606405,
'East_OldcastleCoastal_Sarasota FL': 6.216036153222893,
'East_OldcastleCoastal_Tampa FL - Anderson Rd': 2.1841462654292125,
'East_OldcastleCoastal_Tampa FL - Busch Blvd': 1.8405637211634944,

```
'East_OldcastleCoastal_Tampa FL-SAK': 2.128627558330544,
'East_OldcastleCoastal_Theodore AL': 3.8981662056185242,
'East_OldcastleCoastal_West Palm Beach FL': 5.228303446519645,
'East_OldcastleCoastal_Zephyrhills FL': 5.724668812326115,
'Lawn and Garden_L&G Central_Aliceville AL': 19.055441406558693,
'Lawn and Garden_L&G Central_Amherst Junction WI': 2.910586683616632,
'Lawn and Garden_L&G Central_Bridgeport MI': 9.705544191983433,
'Lawn and Garden_L&G Central_Cleveland, TX': 2.2804321157915313,
'Lawn and Garden_L&G Central_Dallas TX': 3.7559335047302875,
'Lawn and Garden_L&G Central_Del Valle, TX': 3.3176018667292793,
'Lawn and Garden_L&G Central_Harrah OK': 3.2883109370573815,
'Lawn and Garden_L&G Central_Hope AR': 1.606890085700167,
'Lawn and Garden_L&G Central_Livingston TX': 2.947177287512787,
'Lawn and Garden_L&G Central_Marseilles IL': 2.4746540429685897,
'Lawn and Garden_L&G Central_Miami OK': 2.5649318420235687,
'Lawn and Garden_L&G Central_Paola KS': 15.529639746338923,
'Lawn and Garden_L&G Central_Powderly TX': 2.8828720336468923,
'Lawn and Garden_L&G Central_Sauget IL': 4.5574774496409844,
'Lawn and Garden_L&G Central_Tulia TX': 8.201885312063176,
'Lawn and Garden_L&G Central_Tylertown MS': 3.8629671949876347,
'Lawn and Garden_L&G Central_Waterloo IN': 3.520362956158099,
'Lawn and Garden_L&G Northeast_Berlin NY': 21.28842134855701,
'Lawn and Garden_L&G Northeast_Carey OH': 7.407415303982493,
'Lawn and Garden_L&G Northeast_Castlewood VA': 7.299583596095226,
'Lawn and Garden_L&G Northeast_Chatsworth GA': 10.04319375584352,
'Lawn and Garden_L&G Northeast_Historical-Co-Packer': 20.524461296501535,
'Lawn and Garden_L&G Northeast_Hooksett NH': 21.368904161647652,
'Lawn and Garden_L&G Northeast_Lee MA': 9.42509140262139,
'Lawn and Garden_L&G Northeast_Manchester NY': 23.400512460480538,
'Lawn and Garden_L&G Northeast_Mount Hope NJ': 1.8576014125912614,
'Lawn and Garden_L&G Northeast_Poland Spring ME': 10.409371777152439,
'Lawn and Garden_L&G Northeast_Quakertown PA': 11.88117279720865,
'Lawn and Garden_L&G Northeast_Thomasville PA': 9.682649625366468,
'Lawn and Garden_L&G Northeast_Wyoming RI': 16.954476292789725,
'Lawn and Garden_L&G Southeast_Aberdeen NC': 3.469225200913554,
'Lawn and Garden_L&G Southeast_Bostwick FL': 6.646401280874997,
'Lawn and Garden_L&G Southeast_Cross City FL': 8.355268334394506,
'Lawn and Garden_L&G Southeast_Davenport FL': 5.683579189219197,
'Lawn and Garden_L&G Southeast_Fort Green FL': 2.3889451919945213,
'Lawn and Garden_L&G Southeast_Gaffney SC': 2.2803481488775406,
'Lawn and Garden_L&G Southeast_Louisburg NC': 2.3857531039749014,
'Lawn and Garden_L&G Southeast_Moore Haven FL': 3.4845041342679304,
```

```
'Lawn and Garden_L&G Southeast_Pageland SC': 3.264351028089385,
'Lawn and Garden_L&G Southeast_Shady Dale GA': 9.685963393062714,
'Lawn and Garden_L&G Southeast_Walterboro SC': 9.842614431491143,
'National_AMTC_Saint Paul MN': 6.88250736657649,
'National_Anchor Wall Systems_Minnetonka MN': 12.440410939783856,
'National_MoistureShield_Lowell AR': 2.452122990211984,
'National_MoistureShield_Springdale AR': 8.014306375152634,
'National_Oldcastle Sakerete Billing_Easy Mix': 9.803730359674056,
'National_Oldcastle Sakerete Billing_Roberts Concrete': 7.88827620286113,
'National_Oldcastle Sakerete Billing_US Mix': 3.492927733928138,
'National_Techniseal_Ash Grove Memphis': 10.067507588574049,
'National_Techniseal_Ash Grove Nebraska': 4.427190942433016,
'National_Techniseal_Candiac': 3.478638160502194,
'National_Techniseal_Coastal Tampa': 11.134726296224995,
'National_Techniseal_CPM Portland': 6.25033699366245,
'National_Techniseal_Ectra': 39.99498155779078,
'National_Techniseal_Eurl Leps Mehat': 35.28013802145186,
'National_Techniseal_EZ Mix': 7.966435911117332,
'National_Techniseal_Handy Concrete': 11.943071126166176,
'National_Techniseal_PTB  Compaktuna': 16.486136754963134,
'National_Techniseal_Ras': 8.490136620006826,
'National_Techniseal_Techmix': 11.51947724261132,
'National_Westile_Westile Roofing Products': 1.9487081831292374,
'West_Amcor_North Salt Lake UT': 6.751276974390394,
'West_CPM_Frederickson, WA': 4.212873725914967,
'West_CPM_Kent, WA': 5.792254526316904,
'West_CPM_Northstar Consignment': 14.290872252115838,
'West_CPM_Pasco WA': 9.993787280112668,
'West_CPM_Portland, OR': 1.3696375222917925,
'West_CPM_Spokane, WA': 7.050091145372224,
'West_Sierra_Fontana CA': 2.2823988875475774,
'West_Sierra_Reno NV': 5.130910764595873,
'West_Sierra_San Carlos CA': 15.53722173074278,
'West_Sierra_Stockton CA': 4.47300639520385,
'West_Superlite_Gilbert, AZ': 8.930738951426413,
'West_Superlite_Integra Product': 10.414481615909095,
'West_Superlite_Lone Butte': 6.6560644745332205,
'West_Superlite_North Las Vegas': 5.808336473116973,
'West_Superlite_Superlite - Western 19th Ave': 6.223471909361451,
'West_Superlite_Tucson, AZ - Gardner Ln': 5.581850925605079,
'West_Superlite_West Phoenix N. 42nd Ave, AZ': 2.057449853850422}
```

In [42]:
```
# # Storing our dictionary into a pickle file

# with open('lstm_di_log.pkl', 'wb') as f:
#     pickle.dump(lstm_di_log, f)
```

Observations

- As we can see our LSTM Model performance is good enough compared to our ML models
- We will see their comparisons in the results.ipynb notebook
- I have used "TimeSeries Augmentation" since we have less data and it will help our model learn better by having different perspectives of single unique data
- As you can see i am not running LSTM on differencing/Lag transformed data as through the ML solution we concluded that this transformation is not reaping good results as comapred to Log Transformation, since running LSTM was time consuming i decided to run it only on valuable Log Transformed data