

# Time Series Analysis for Forecasting Sales for First Quarter of 2018

Presentation by  
Sidhant Ahluwalia





# **Part 1:**

## **Project Description**





# Store Item Demand Forecasting

## Background and Problem Introduction

- The business has ten stores selling fifty kinds of different items
- It has a five-year whole stores inventory sales dataset with 913000 records
- It needs to forecast the next three months of sales based on this historical records (From 2018: Jan to 2018: Mar)



# Dataset Description

In the provided dataset there are four variables `date`, `store_id`, `Item_id` and `sales` :

**Date:** The date variable stores every calendar date when transactions happen. It starts on Jan 1, 2013, and ends on Dec 31, 2017. Since every store and item has its own record of sales, each individual date appears multiple times in the dataset.

**Store:** The store variable stores the store id. There is ten unique id for each store which is labeled from 1 to 10.

**Item:** The Item variable stores a unique id for each item. There is fifty unique id for each item labeled from 1 to 50.

**Sales:** The sale variable stores the total number of items sold on the respective store at the given date.



## Data Description (Cont.)

Variable Names	Data Type	Description
Date	Date	Date of the sale data. There are no holiday effects or store closures
Store	Factor	Store ID
Item	Factor	Item ID
Sales	Int	Number of items sold at a particular store on a particular date

1	date	store	item	sales
2	1/1/2013	1	1	13
3	1/2/2013	1	1	11
4	1/3/2013	1	1	14
5	1/4/2013	1	1	13
6	1/5/2013	1	1	10
7	1/6/2013	1	1	12

\* The first six rows of the dataset as an example of how the data is structured



# Project Work Plan

- **Tool**
  - Python platform for overall forecasting of future sales
- **Purpose**
  - Pattern detection within time series of sales data
  - Sales forecasting with factors of location and category
- **Expectation**
  - Find the best structure to manipulate the data
    - Dimensions of aggregate data
  - Find the best model to forecast
    - Naive
    - ARIMA



# **Part 2:** **Exploring the dataset in** **Python**



# Exploring Dataset

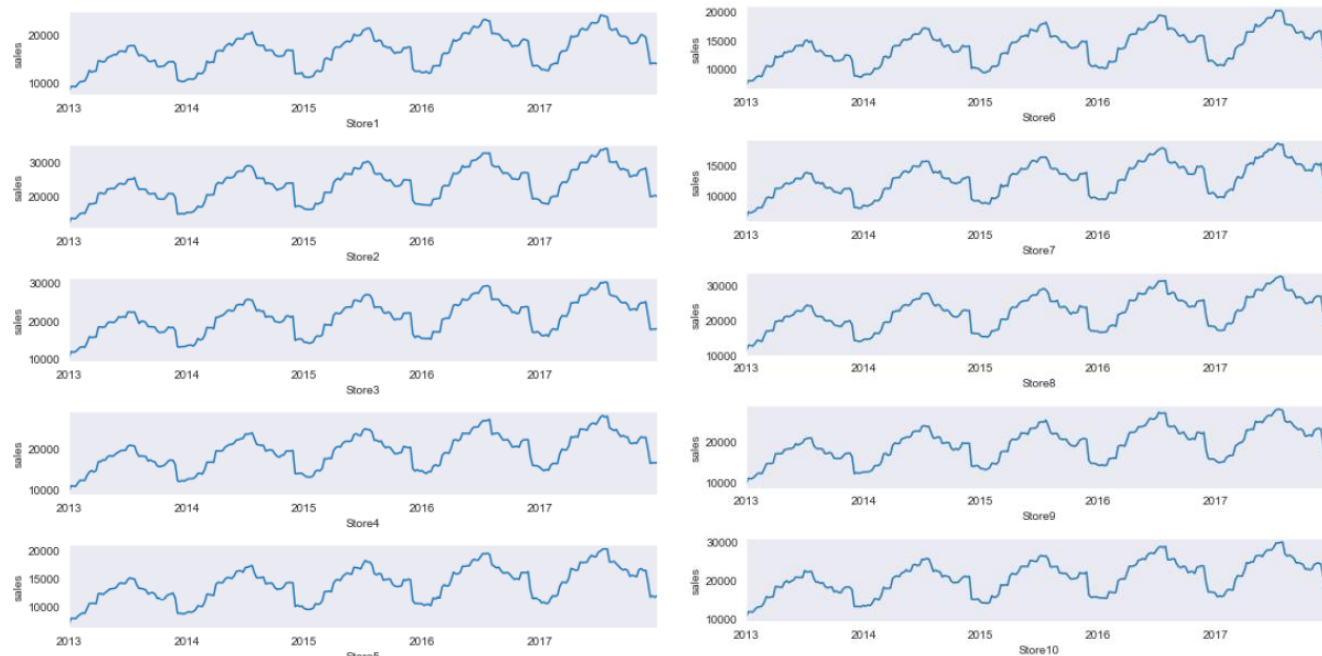


Figure 1: Daily weekly sales of all stores from 2013 to 2017.

- The figure displays the Time series graph of consolidated weekly sales of all the items in all the 10 stores.
- All Stores shows similar patterns.



# Exploring Store 1 and Item 1

- The time series displays similar sales pattern with sales increasing till mid yr and then decreasing with the spike at the end of the year.
- The scale of the sales can be observed to be increasing representing a trend in the timeseries.
- The similar repeating patterns represent seasonality.

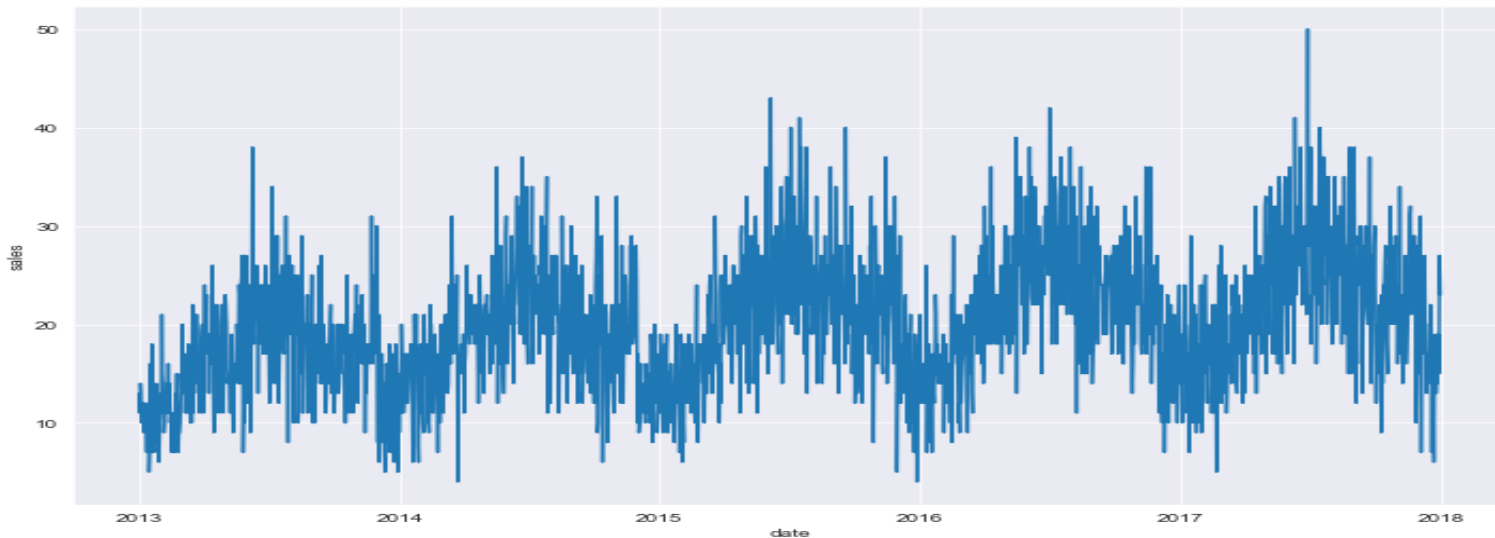


Figure 2: Daily sales of Item1 store 1 from 2013 to 2017.

# Exploring Store 1 and Item 1

- The time series displays sales pattern for the year 2013.
- Daily sales keeps on increasing for the first half of the year then decrease during fall with spike in beginning of December.
- The occasional spikes and a pattern indicate the the time series is not stationary.

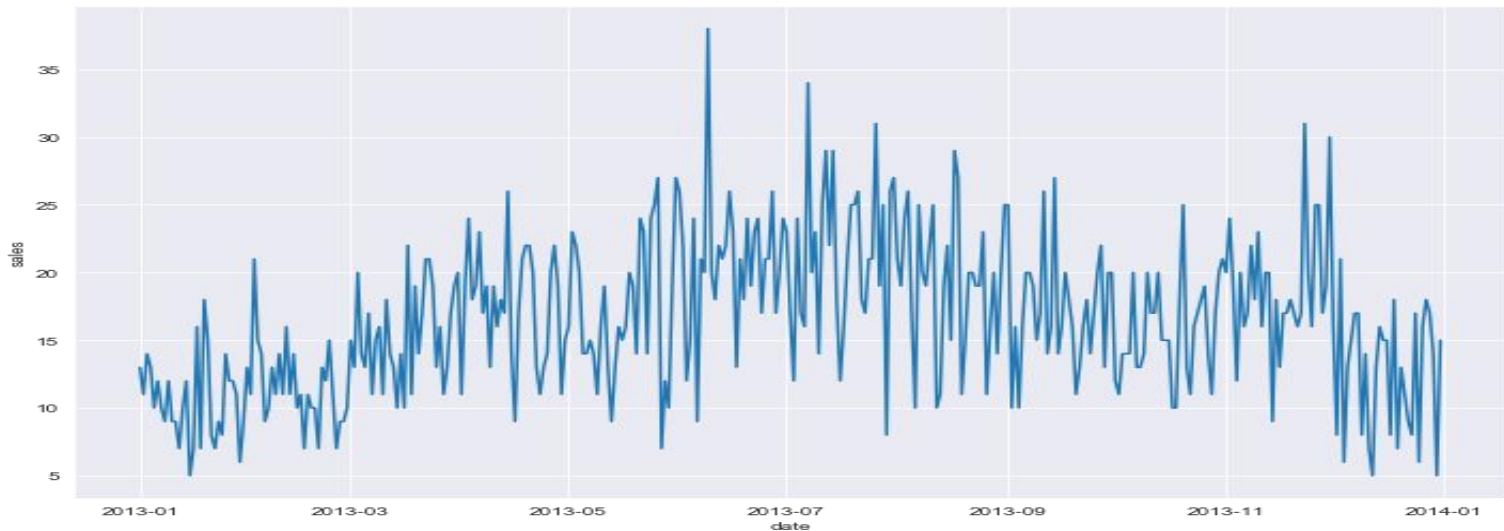


Figure 3: Daily sales of Item1 store 1 in 2013.

# Exploring Store 1 and Item 1

- The box-plot displays sales for each month.
- The months of July and November records the highest sales.
- Outliers represent special events such as holidays or store closures.

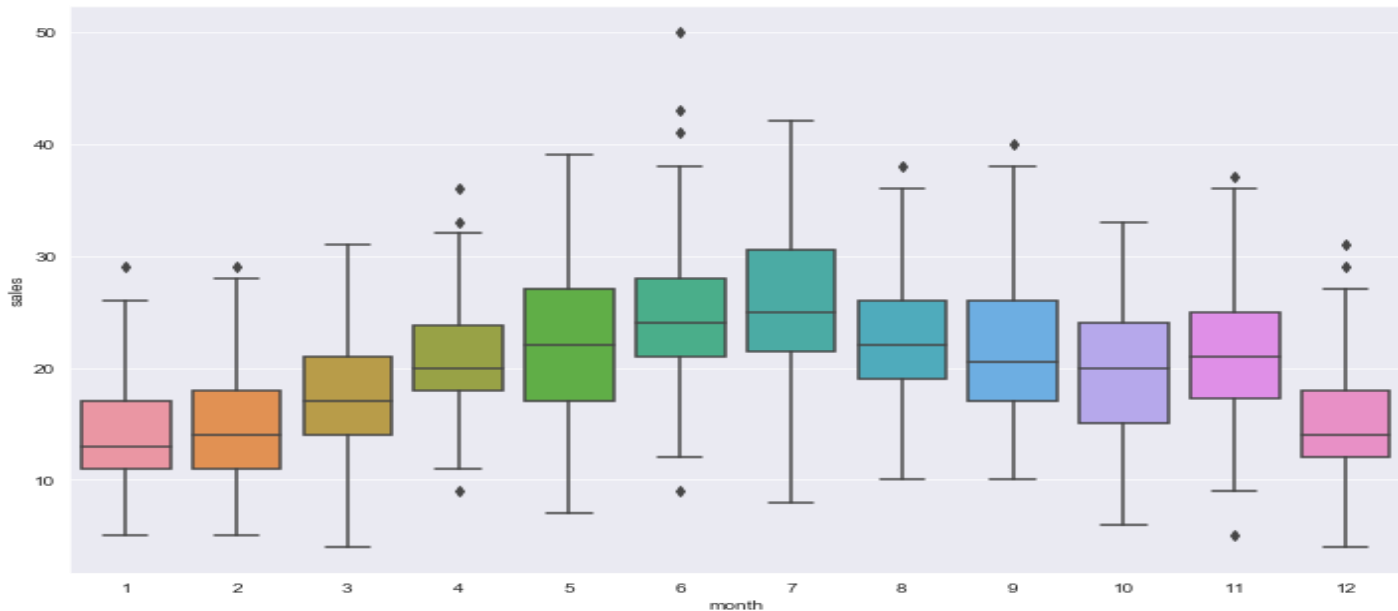


Figure 5: Sales by each month for Item1 store 1.

# Exploring Store 1 and Item 1

- The box-plot displays sales on each day of the week from monday to sunday. (monday = 0)
- Maximum sales happen on the weekend.
- Outliers represent special events such as holidays or store closures.

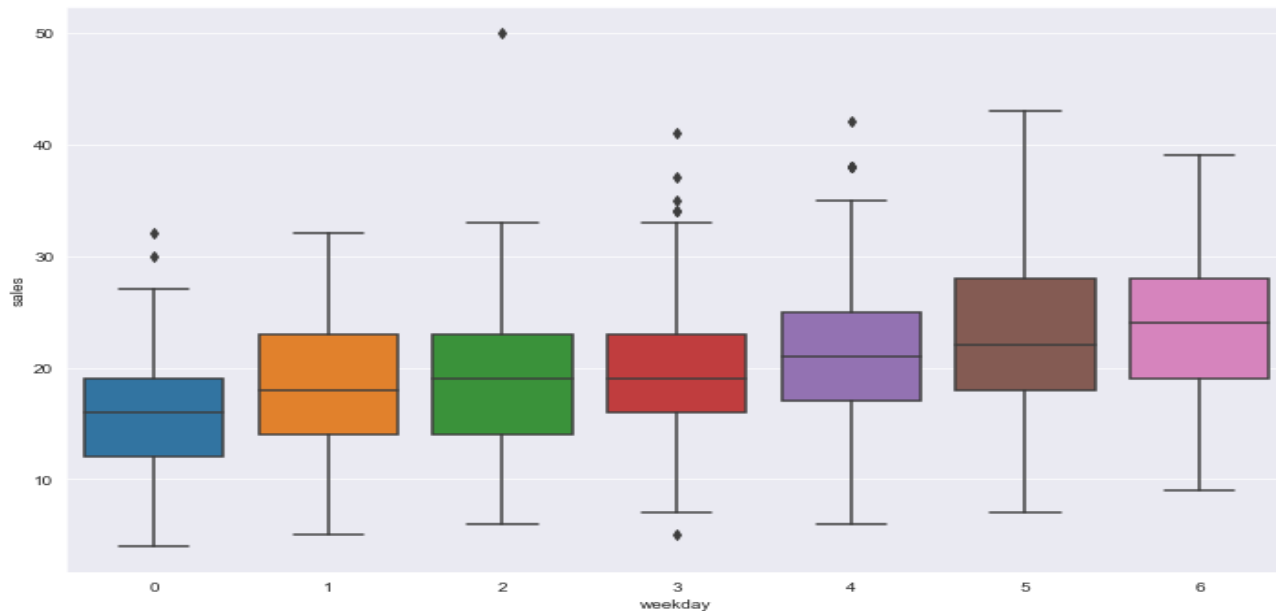
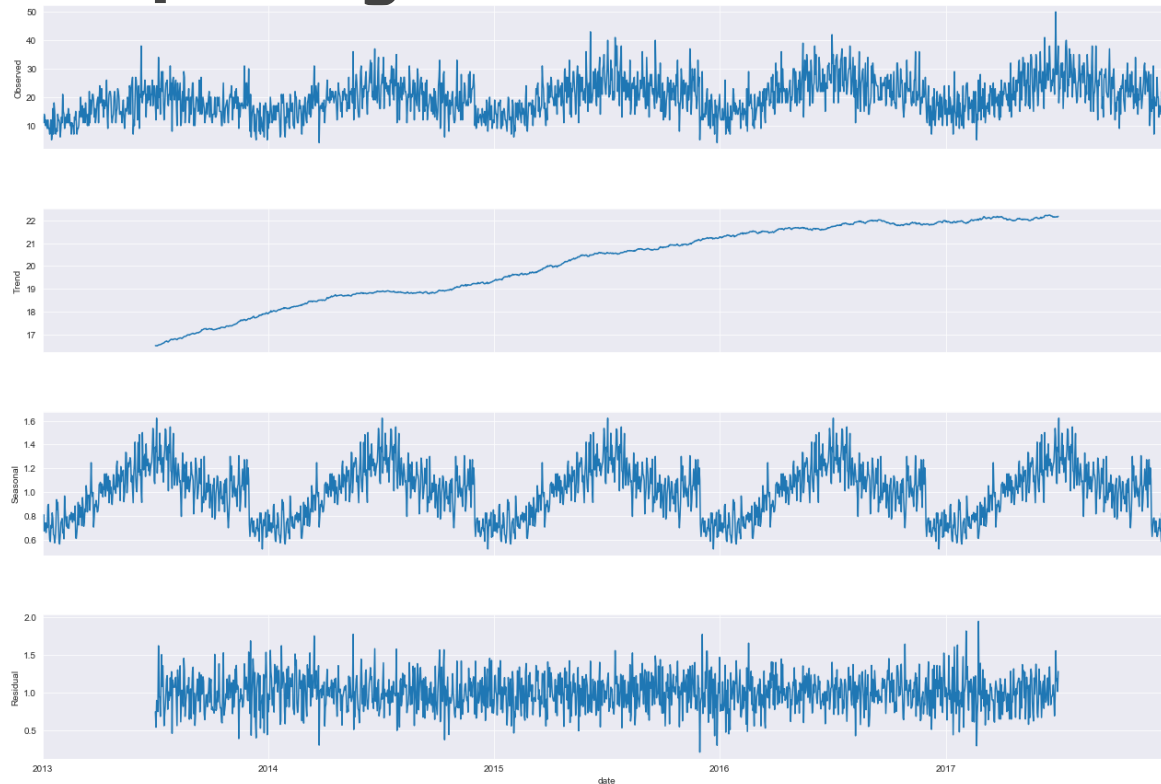


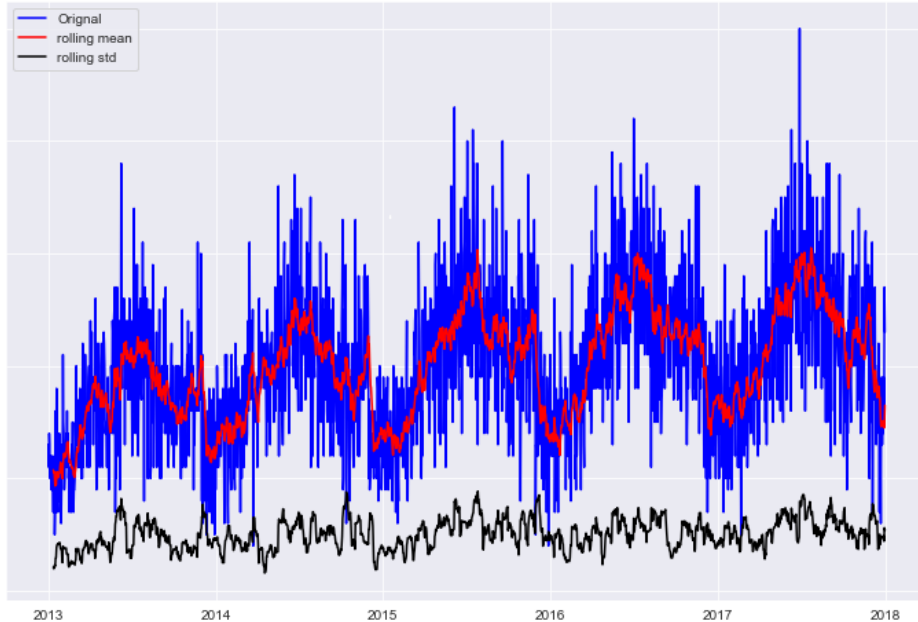
Figure 4: Sales by each day of Item1 store 1.

# Exploring Store 1 and Item 1



The decomposition graph depicts a similar pattern, rising trend and a seasonality in the time series. As a result we can confirm that the time series is not stationary.

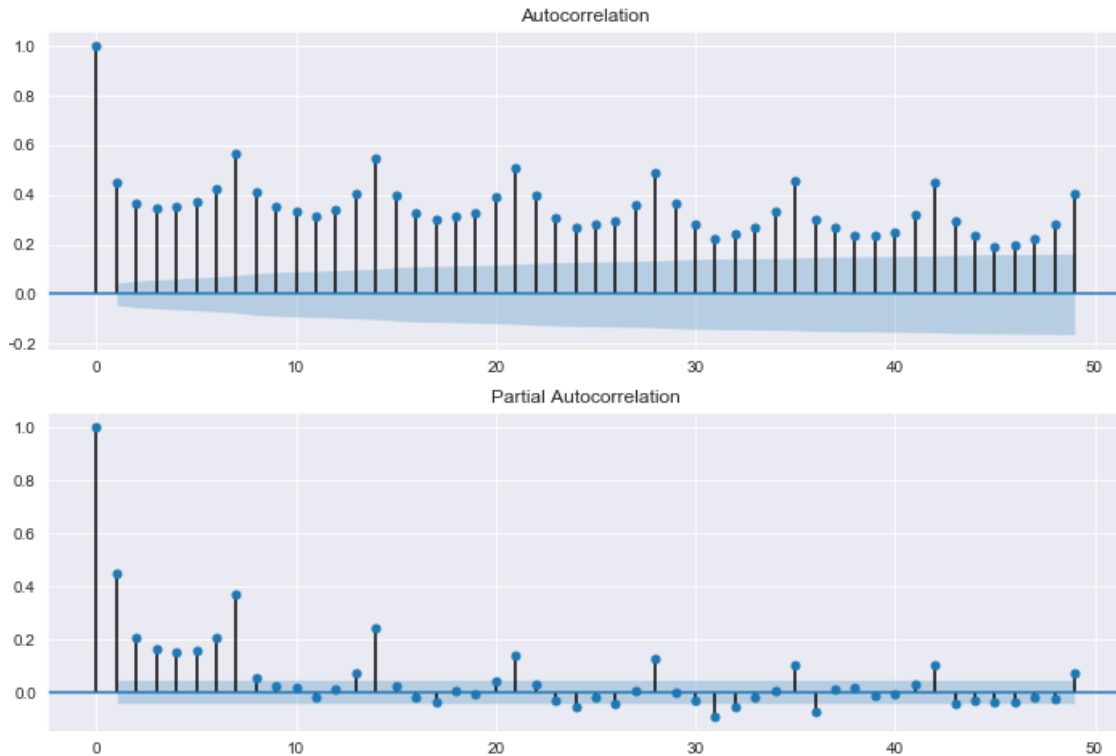
# Exploring Store 1 and Item 1




The results of the test are :  
p-value = 0.0361. The series is likely not stationary.  
Test statistic -2.987278  
p-value 0.036100  
No. of lags 20.000000  
No. of observations 1805.000000  
Critical value (1%) -3.433978  
Critical value (5%) -2.863143  
Critical value (10%) -2.567623  
dtype: float64

The variation of the mean in the time series graph further validates the point that the time series is not stationary .

# Exploring Store 1 and Item 1




The significant spikes in the acf and pacf plots also validates that there is a trend and seasonality in the time series.



# **Part 3:**

## **Arima Model Selection**







# Arima Model

- ARIMA is an acronym for AutoRegressive Integrated Moving Average
- It is a combination of Auto regressive(p), differencing(d) and Moving average(q)
- It is represented as Arima (p,d,q) (P,D,Q)m
- p & q represents order of autoregressive and moving average part.
- P & Q represents seasonal order of autoregressive and moving average part.
- D & d represents seasonal and simple differencing.
- m represents the number of observations per year.

# Check for Stationarity

```
seven_diff = train_1_1.sales - train_1_1.sales.shift(7)
seven_diff.head()
```



After seasonal differencing we see that the mean has become constant and the time series has become stationary.

The results of the test are :

p-value = 0.0000. The series is likely stationary.

Test statistic -1.260596e+01

p-value 1.695680e-23

No. of lags 2.000000e+01

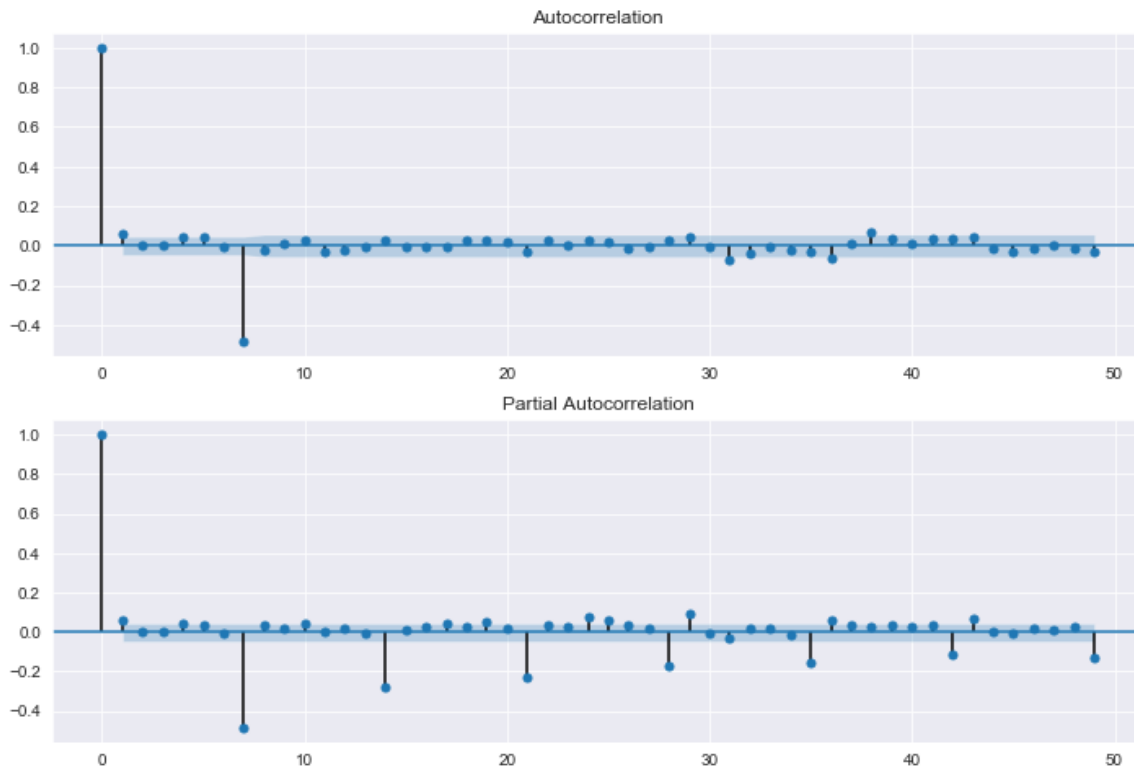
No. of observations 1.798000e+03

Critical value (1%) -3.433992e+00

Critical value (5%) -2.863149e+00

Critical value (10%) -2.567626e+00

# Check for Stationarity



- The significant spikes in the ACF and PACF plots of the differenced time series have reduced.
- ACF plot has significant spike at lag 7 and lag 1
- PACF graph has significant exponential decays at periodic intervals of (7,14,21..) and a significant spike at lag 1.



# Model Selection

Based on the findings in the ACF and PACF plots we can have following possible ARIMA models.

	<b>p</b>	<b>d</b>	<b>q</b>	<b>P</b>	<b>D</b>	<b>Q</b>
<b>Model_1</b>	0	0	0	0	1	1
<b>Model_2</b>	1	0	0	0	1	1
<b>Model_3</b>	0	0	1	0	1	1
<b>Model_1</b>	1	0	1	0	1	1

# Model Selection

```
cnt = 0
for p in p_values:
    for q in q_values:
        try:
            mod = sm.tsa.statespace.SARIMAX(y, order=(p,d,q),
                                             seasonal_order=(P,D,Q,s),
                                             enforce_stationarity=False,
                                             enforce_invertibility=False)

            results = mod.fit()
            cnt += 1
            if cnt % 50 :
                print('Current Iter - {}, ARIMA({}x{}x{}) & {}x{}x{}x7 - AIC:{}'.format(cnt, p, d, q, P, D, Q, results.aic))
        except:
            continue
```

Current Iter - 1, ARIMA0x0x0 & 0x1x1x7 - AIC:11041.103684371203  
Current Iter - 2, ARIMA0x0x1 & 0x1x1x7 - AIC:11003.387863193702  
Current Iter - 3, ARIMA1x0x0 & 0x1x1x7 - AIC:11001.229636140988  
Current Iter - 4, ARIMA1x0x1 & 0x1x1x7 - AIC:10744.26501132096

```
p_values = range(0,2)
q_values = range(0,2)
d = 0
s = 7
P = 0
D = 1
Q = 1
y = train_1_1.sales[:"2017-10-01"]
#x = train_1_1.sales["09-30-2017"]
```

# Model Selection

```
arima_model_1 = sm.tsa.SARIMAX(train_1_1.sales, order=(1,0,1), seasonal_order=(0,1,1,7),
                                enforce_stationarity=False, enforce_invertibility=False).fit(dispatch=False)
print(arima_model_1.summary())
```

## Statespace Model Results

```
=====
Dep. Variable:          sales    No. Observations:          1826
Model:                SARIMAX(1, 0, 1)x(0, 1, 1, 7)    Log Likelihood          -5368.133
Date:                  Thu, 25 Apr 2019    AIC          10744.265
Time:                  16:02:21    BIC          10766.269
Sample:                0    HQIC          10752.385
                        - 1826
```

Covariance Type: opg

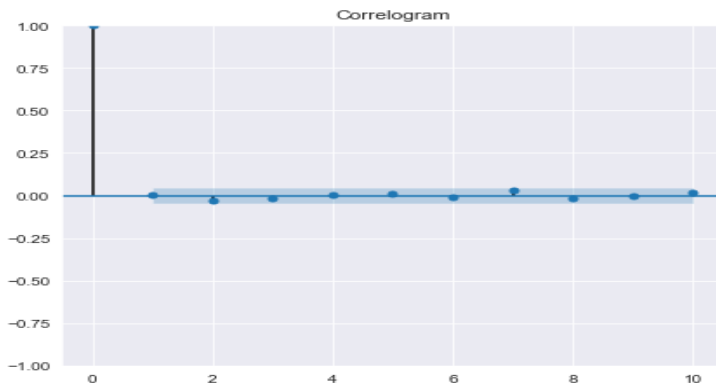
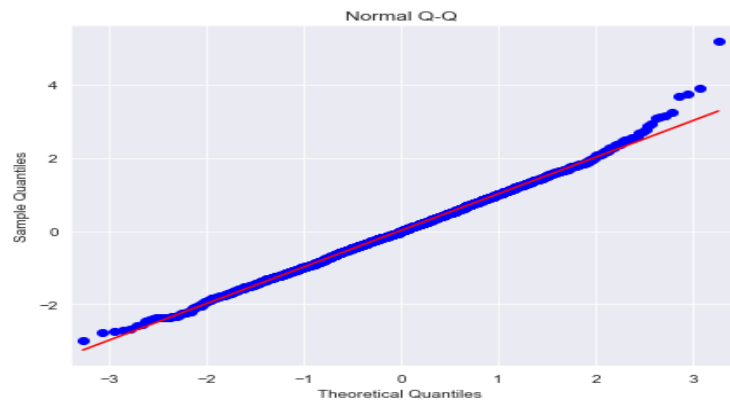
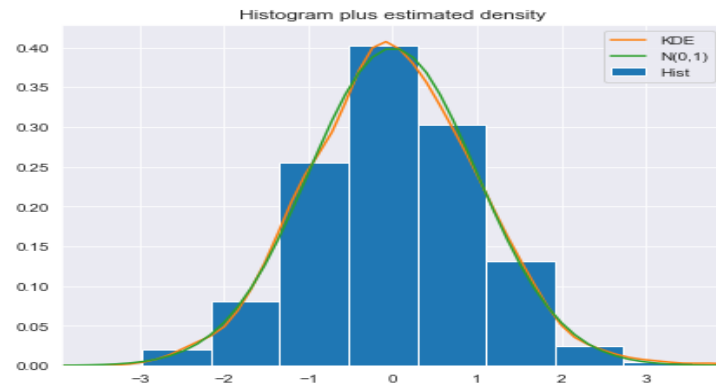
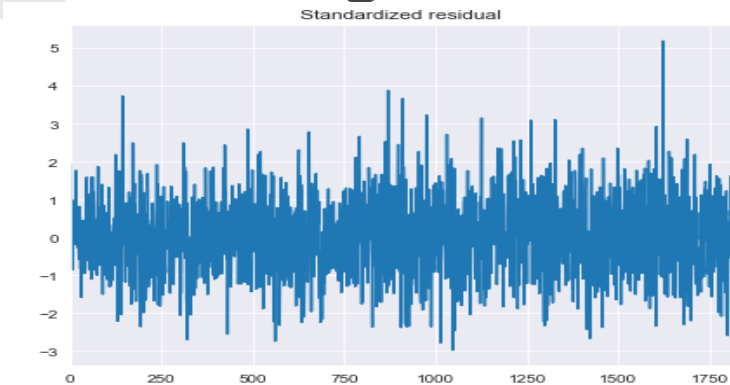
```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
ar.L1          0.9912      0.003     301.544      0.000      0.985      0.998
ma.L1         -0.8860      0.013    -69.930      0.000     -0.911     -0.861
ma.S.L7       -1.0071      0.007   -144.061      0.000     -1.021     -0.993
sigma2        21.4353      0.698     30.701      0.000     20.067     22.804
=====
```

```
=====
Ljung-Box (Q):          42.39    Jarque-Bera (JB):          29.28
Prob(Q):                0.37    Prob(JB):              0.00
Heteroskedasticity (H):  1.32    Skew:                  0.20
Prob(H) (two-sided):    0.00    Kurtosis:              3.48
=====
```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

# Testing Residuals



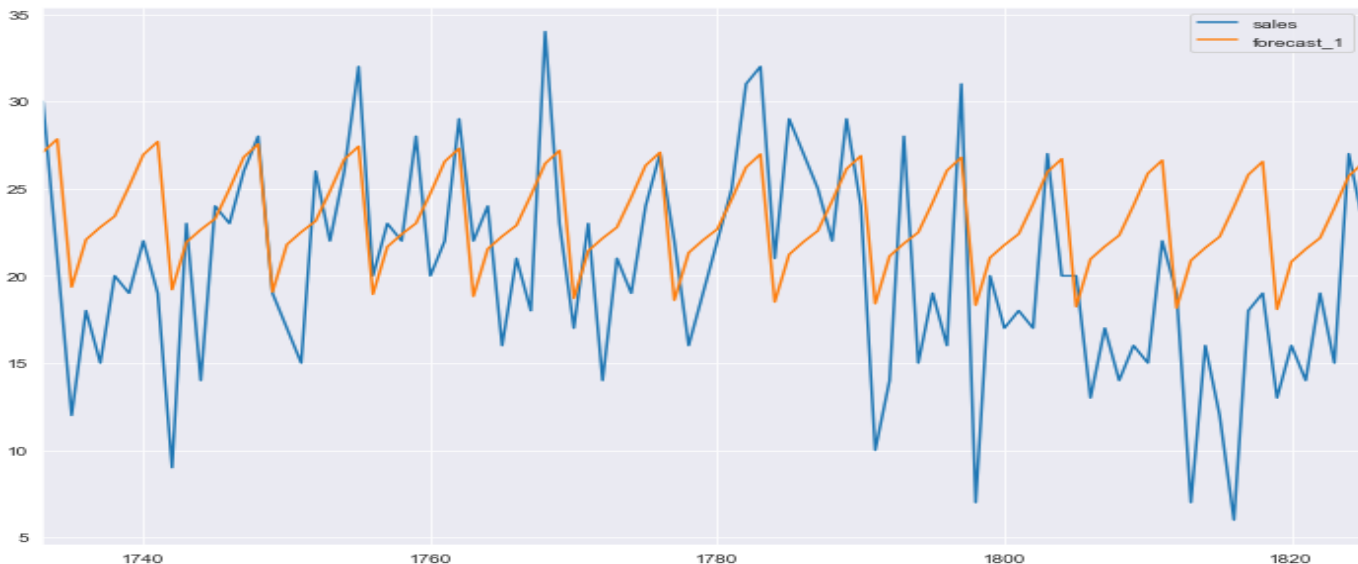
## Forecasting Using the selected Model

```
begin = 1733
stop = 1825
train_1_1['forecast_1'] = arima_model_1.predict(start = begin, end = stop, dynamic=True)
train_1_1.tail()
```

	date	store	item	sales	year	month	day	weekday	forecast_1
1821	2017-12-27	1	1	14	2017	12	27	2	21.532563
1822	2017-12-28	1	1	19	2017	12	28	3	22.169574
1823	2017-12-29	1	1	15	2017	12	29	4	23.887625
1824	2017-12-30	1	1	27	2017	12	30	5	25.724053
1825	2017-12-31	1	1	23	2017	12	31	6	26.480575



# Measuring Accuracy



```
mape_smape(train_1_1[1733:1826]['sales'],train_1_1[1733:1826]['forecast_1'])
```

MAPE : 32.25 %

SMAPE: 24.57 %



## Benchmark:Naive

Naive model is the benchmark of our model selection, every valid forecasting model should have a better accuracy rate than the Naive method.

Accuracy Summary - Item1@Store1 from 2013 to 2017

	MAPE	SMAPE
Naive Forecast	35.25 %	31.34 %
Arima_model1_Forecast	32.25 %	24.57 %



# Modifying Dataset to capture special events

holiday_bool	month_1	month_2	month_3	...	holiday_Presidents Day (Washingtons Birthday)	holiday_Thanksgiving Day	holiday_Veterans Day	weekday_0	weekday_1	weekday_2
1	1	0	0	...	0	0	0	0	1	0
0	1	0	0	...	0	0	0	0	0	1
0	1	0	0	...	0	0	0	0	0	0
0	1	0	0	...	0	0	0	0	0	0
0	1	0	0	...	0	0	0	0	0	0

# Model Evaluation

```
p_values = range(0,2)
q_values = range(0,2)
d = 0
s = 7
P = 0
D = 1
Q = 1
y = train_1_1_d.sales[:begin]
x = exog_data[:begin]
```

```
cnt = 0
for p in p_values:
    for q in q_values:
        try:
            mod = sm.tsa.statespace.SARIMAX(exog=x, endog=y, order=(p,d,q),
                                             seasonal_order=(P,D,Q,s),
                                             enforce_stationarity=False,
                                             enforce_invertibility=False)

            results = mod.fit()
            cnt += 1
            if cnt % 50 :
                print('Current Iter - {}, ARIMA{x}{x} & {x}{x}{x}x7 - AIC:{}'.format(cnt, p, d, q, P, D, Q, results.aic))
        except:
            continue
```

```
Current Iter - 1, ARIMA0x0X0 & 0x1x1x7 - AIC:10106.066690161462
Current Iter - 2, ARIMA0x0X1 & 0x1x1x7 - AIC:10101.798781855738
Current Iter - 3, ARIMA1x0X0 & 0x1x1x7 - AIC:10107.376011214357
Current Iter - 4, ARIMA1x0X1 & 0x1x1x7 - AIC:10104.078963320386
```



# Measuring Accuracy

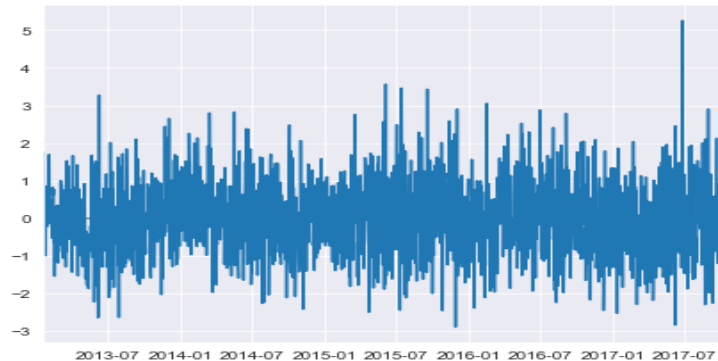
```
arima_model_2 = sm.tsa.statespace.SARIMAX(endog=train_1_1_d.sales[:begin], exog=exog_data[:begin], order=(0,0,1),
                                           seasonal_order=(0,1,1,7),enforce_stationarity=False,enforce_invertibility=False).fit()
print(arima_model_2.summary())
```

## Statespace Model Results

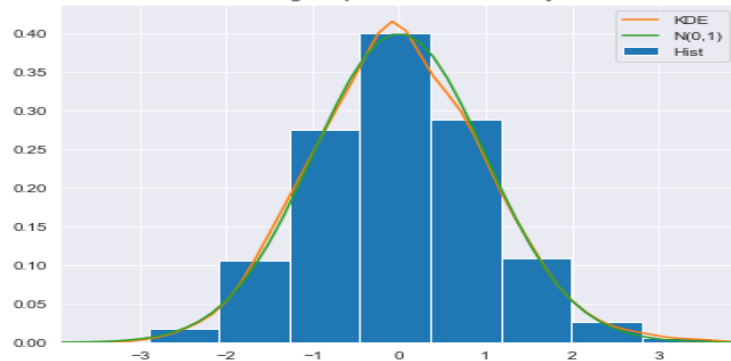
```
=====
Dep. Variable:          sales      No. Observations:          1735
Model:                SARIMAX(0, 0, 1)x(0, 1, 1, 7)    Log Likelihood          -5015.899
Date:                 Thu, 25 Apr 2019                AIC              10101.799
Time:                 19:55:04                       BIC              10292.531
Sample:               01-01-2013                      HQIC             10172.369
                   - 10-01-2017
```

# Testing Residuals

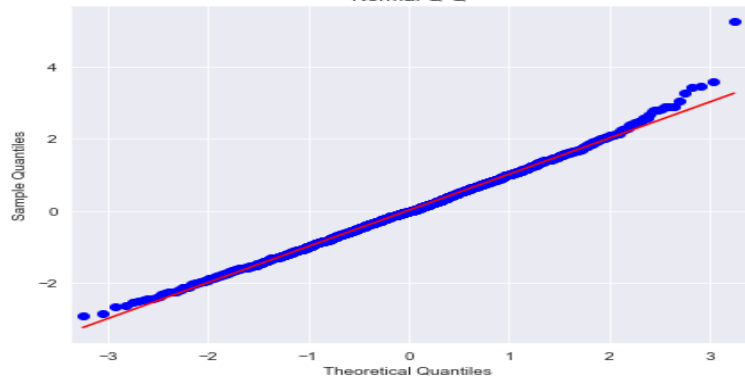
Standardized residual



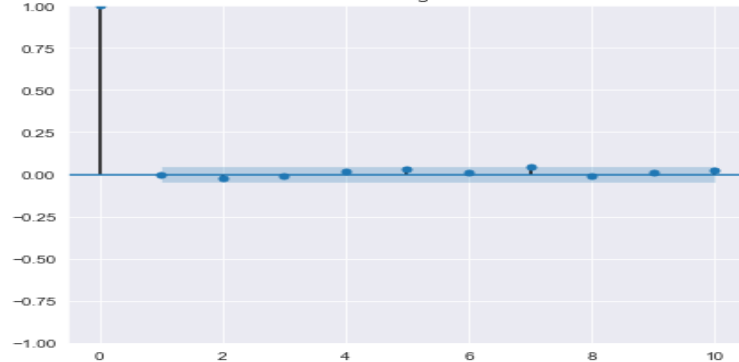
Histogram plus estimated density



Normal Q-Q



Correlogram



# Measuring Accuracy



# Measuring Accuracy

```
train_1_1_d['forecast_2'] = arima_model_2.predict(start=pd.to_datetime(begin), end = pd.to_datetime(stop1),  
                                                  exog=exog_data[begin:stop], dynamic=True)  
train_1_1_d[begin:stop]['forecast_2'].head()
```

```
date  
2017-10-01    26.604354  
2017-10-02    17.755507  
2017-10-03    20.669078  
2017-10-04    21.487374  
2017-10-05    22.385232
```

```
mape_smape(train_1_1_d[begin:stop1]['sales'],train_1_1_d[begin:stop1]['forecast_2'])
```

```
MAPE : 25.23 %  
SMAPE: 20.94 %
```





# Benchmark: Naive

Naive model is the benchmark of our model selection, every valid forecasting model should have a better accuracy rate than the Naive method.

Accuracy Summary - Item1@Store1 from 2013 to 2017

	MAPE	SMAPE
<b>Naive Forecast</b>	35.25 %	31.34 %
<b>Arima_model1_Forecast</b>	32.25 %	24.57 %
<b>Arima_model2_Forecast</b>	25.23 %	20.94 %



# **Part 4:**

## **Forecasting 3 Months of Item Sales**



```

forecast = []
for s in range(1,11):
    for i in range(1,51):
        storeid = s
        itemid = i
        train = train_df[train_df['store']==storeid]
        train = train[train['item']==itemid]
        test = test_df[test_df['store']==storeid]
        test = test[test['item']==itemid]
        test.set_index('id')
        test.drop('id',axis = 1, inplace=True)
        test['sales'] = np.NaN

    df = pd.concat([train,test], axis = 0, join='outer')
    df['date'] = pd.to_datetime(df['date'])

    df['year'] = df['date'].dt.year - 2012
    df['month'] = df['date'].dt.month
    df['day'] = df['date'].dt.dayofyear
    df['weekday'] = df['date'].dt.weekday
    df = df.merge(holiday, how='left', on='date')
    df['holiday_bool'] = pd.notnull(df['holiday']).astype(int)
    df = pd.get_dummies(df, columns = ['month','holiday','weekday'] , prefix = ['month','holiday','weekday'])
    exog_data_2 = df[column_list]
    exog_data_2 = exog_data_2.set_index('date')
    df = df.set_index('date')
    begin = '2018-01-01'
    stop = '2018-03-30'
    stop1 = '2018-03-31'
    arima_model_4 = sm.tsa.statespace.SARIMAX(endog=df.sales[:begin], exog=exog_data_2[:begin], order=(0,0,1),
                                                seasonal_order=(0,1,1,7),enforce_stationarity=False,enforce_invertibility=False).fit()
    df['forecast'] = arima_model_4.predict(start=pd.to_datetime(begin), end = pd.to_datetime(stop1),
                                          exog=exog_data_2[begin:stop], dynamic=True)

    forecast.append(df[begin:stop][['store','item','forecast']])
print('item:',i,'store:',s,'Finished.')

```



# Forecast

```
[89 rows x 3 columns],
```

date	store	item	forecast
2018-01-01	3	4	18.287411
2018-01-02	3	4	21.015472
2018-01-03	3	4	20.737005
2018-01-04	3	4	22.273677
2018-01-05	3	4	25.316212
2018-01-06	3	4	26.084417
2018-01-07	3	4	27.554290
2018-01-08	3	4	17.324563
2018-01-09	3	4	20.218420
2018-01-10	3	4	20.524243
2018-01-11	3	4	22.060915
2018-01-12	3	4	25.103449
2018-01-13	3	4	25.871654
2018-01-14	3	4	27.341527
2018-01-15	3	4	17.111801

```
[89 rows x 3 columns],
```

date	store	item	forecast
2018-01-01	6	1	12.137933
2018-01-02	6	1	14.536071
2018-01-03	6	1	12.642259
2018-01-04	6	1	14.152696
2018-01-05	6	1	15.064946
2018-01-06	6	1	16.453739
2018-01-07	6	1	17.164566
2018-01-08	6	1	10.703893
2018-01-09	6	1	13.089127
2018-01-10	6	1	12.543280
2018-01-11	6	1	14.053717
2018-01-12	6	1	14.965967
2018-01-13	6	1	16.354760
2018-01-14	6	1	17.065587
2018-01-15	6	1	10.604914
2018-01-16	6	1	14.327603

# Reference List

- Data Resource: <https://www.kaggle.com/c/demand-forecasting-kernels-only/>
- Forecasting: Principles and Practice: <https://otexts.com/fpp2/>
- Python