Machine Learning Nanodegree

Capstone Project

Store Item Demand Forecasting Challenge

Chandra Nayak
September 30, 2018

I. Definition

Project Overview

The goal of this project is to develop an algorithm that can analyze past sales at a store chain , and forecast the sales in a future quarter.
Sales forecasting is essential intelligence in today's world that every business needs to have with high degree of confidence. Predictive Sales forecasting within an acceptable threshold of error helps businesses make decisions around *budgeting, inventory & supply chain management, and managing workforce in otherwise a dynamic world with many independent variables.*

In this project we will explore supervised learning techniques to develop models for prediction based on time series data.

Supervised learning is the machine **learning** task of **learning** a function that maps an input to an output based on example input-output pairs. Image Recognition, Recommendations and Time Series forecasting have been wildly popular applications that employ supervised learning algorithms; these applications of supervised learning have been been adopted in the industry and have contributed massively to the success of e-commerce sites like amazon.com and Netflix. Deep Learning is a recent technique that has been gaining rapid adoption as a Supervised learning algorithm.

This effort here focuses on using ARIMA technique to analyze time-series data from store sales across a store chain. We may explore other advanced techniques like HYBRID ARIMA , LSTM and GRU (Gated Recurrent Unit) to analyze the time series.

The project "Store Item Demand Forecasting Challenge" is picked from kaggle competitions, which requires us to forecast future sales based on the existing sales data provided.

Software Install:

Numpy, Pandas, Scikit, scikit-learn, Statsmodels

The model is developed in Jupyter Notebook and is available as store_sales_2.0.ipynb my github location https://github.com/mcnayak/capstone
### Problem Statement

The project "Store Item Demand Forecasting Challenge" deals with developing a time-series machine learning model that can forecast future sales. We shall develop and train the model on daily sales data from a store chain for the past 5 years, and use it to predict sales of a quarter in future for the chain. The training dataset provided by the competition includes daily sales data from 10 different stores owned by the store chain selling 50 different items; training dataset has transactions for 5 years between 01-01-2013 and 12-31-2017 while the test dataset includes all sales data between 01-01-2018 and 31-03-2018.

Training data has Date, Item , Store and Sales fields populated with the count of items sold; whereas the test dataset does not have the sales field populated. The forecasting model we develop should be able to predict the sales count for the items on the recorded dates in the test data.

Ideal solution for this problem will be a model that can predict sales for all the stores aggregated grouped by item, which then can easily be used to arrive at the total revenue the store chain is likely to make in the test quarter.

Training dataset provided has about 913000 records of sales data that can be used to train the model. I have followed the approach below to arrive at a model.

a. Perform Initial statistical exploratory analysis of data.
b. Visualize the sales data against time.
c. Use GroupBy to get the mean of store sales by item per quarter.
d. We shall consider the output of step c) as our base model, which the machine learning model has to outdo.
e. We will use data between 01-01-2013 to 09-30-2017 to train the machine learning model, while holding back data between 10-01-2017 and 12-31-2017 for validation of the model.
f. In order to simplify the problem, we will solve the problem for one item and store, which we will refer as Store=1 and Item=1.
g. We will then use the ACF/PACF method to analyze the time series data.
h. Once an appropriate model is identified use it to forecast sales on the last quarter of training data which was held back.
i. Measure the variance between model prediction and test data to measure model efficiency.
j. If the model efficiency is satisfactory you can then apply that model to do sales prediction for the test data between 01-01-2018 and 03-31-2018.

### Metrics
In this section, you will need to clearly define the metrics or calculations you will use to measure performance of a model or result in your project. These calculations and metrics should be justified based on the characteristics of the problem and problem domain. Questions to ask yourself when writing this section:
- _Are the metrics you've chosen to measure the performance of your models clearly discussed and defined?_
- _Have you provided reasonable justification for the metrics chosen based on the problem and solution?_

The metric used to measure model effectiveness will be the deviance of predicted sales for an item to the actual sales for the item.
In order for us compare the predicted and actual sales, we shall hold back last quarter of sales data from the training data and call it as Validation data. Sales forecasting is a regression problem, with time as an additional factor, however, the time factor is removed in model development by making the training data stationary.
Once we develop a model for forecasting sales, model Efficiency will be measured by calculating the variance/Mean-Squared-Error between the sales predicted by the model on the data that was held back and the actual sales for the item during that quarter.

The model that leads to the least sum of mean-squared-errors , and which completes within an acceptable prediction time will be considered as the best model to use.

For easier understanding of deviation of actual and forecasting sales, we shall use appropriate charts from matplotlib library. We will first plot the actual sales data for item 1 and store 1 between 01-01-2013 and 12-31-2017, and then overlay the predicted sales data for item 1 and store 1 for dates between 10-01-2017 and 12-31-2017. If the prediction plot is within an acceptable tolerance then the model prediction is acceptable.

In order to manage the complexity of the problem the result charts are all plotted for ITEM 1 from Store 1 only.

## II. Analysis
(approx. 2-4 pages)

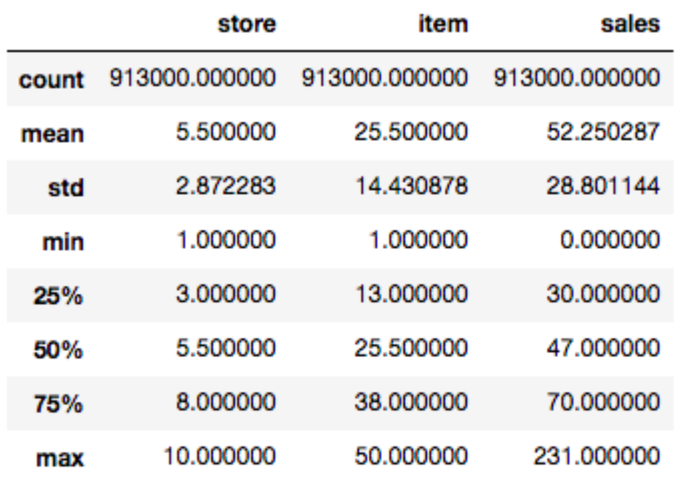
### Data Exploration

The dataset for this problem is provided on kaggle.com, and is distributed as two csv files : train.csv and test.csv.

train.csv - has sales data for the store chain across the 10 stores it owns, and the 50 different items it sells across those stores, and has those transactions recorded between 01-01-2013 and 12-31-2017.

Characteristics of sales data in train.csv: Has 913K rows of sales data with 4 fields/attributes(date, store, item and sales quantity). The 50 different items sold by the chain are enumerated with as numbers starting from 1 to 50 and the stores themself are numbered 1 to 10.
Statistical description of the training data is available in the picture below.

train_df.describe()



	store	item	sales
count	913000.000000	913000.000000	913000.000000
mean	5.500000	25.500000	52.250287
std	2.872283	14.430878	28.801144
min	1.000000	1.000000	0.000000
25%	3.000000	13.000000	30.000000
50%	5.500000	25.500000	47.000000
75%	8.000000	38.000000	70.000000
max	10.000000	50.000000	231.000000

From the statistical information above , I did not see any outliers that I thought would need additional consideration.

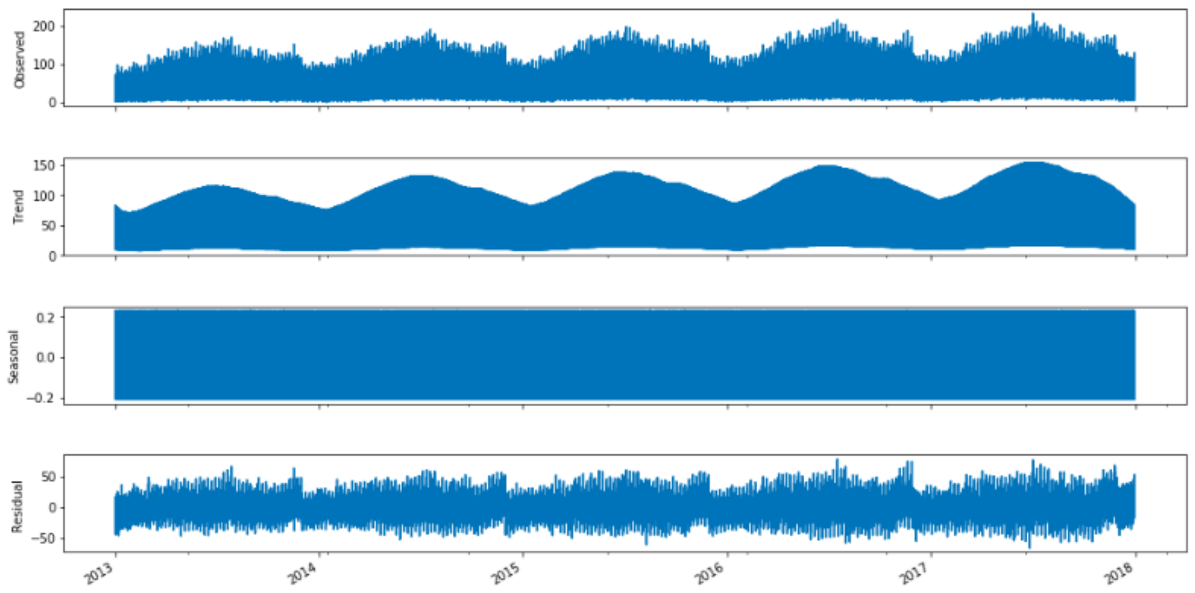
Characteristics of Sales data in test.csv: Test Data has a total of 45K rows of daily sales data between 01-01-2018 and 12-31-2018, with 10 stores enumerated as 1 to 10 which sells 50 different items enumerated as 1 to 50. In case of the testing data the Sales Column is not populated as already stated earlier.

	id	store	item
count	45000.000000	45000.000000	45000.00000
mean	22499.500000	5.500000	25.50000
std	12990.525394	2.872313	14.43103
min	0.000000	1.000000	1.00000
25%	11249.750000	3.000000	13.00000
50%	22499.500000	5.500000	25.50000
75%	33749.250000	8.000000	38.00000
max	44999.000000	10.000000	50.00000

### Exploratory Visualization
In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section:
- _Have you visualized a relevant characteristic or feature about the dataset or input data?_
- _Is the visualization thoroughly analyzed and discussed?_
- _If a plot is provided, are the axes, title, and datum clearly defined?_

We will use statsmodel package to visualize the time series data using the steps below

```
residuals = sm.tsa.seasonal_decompose(total_df.sales.dropna(),freq=90)
fig = residuals.plot()
fig.set_figheight(8)
fig.set_figwidth(15)
plt.show()
```



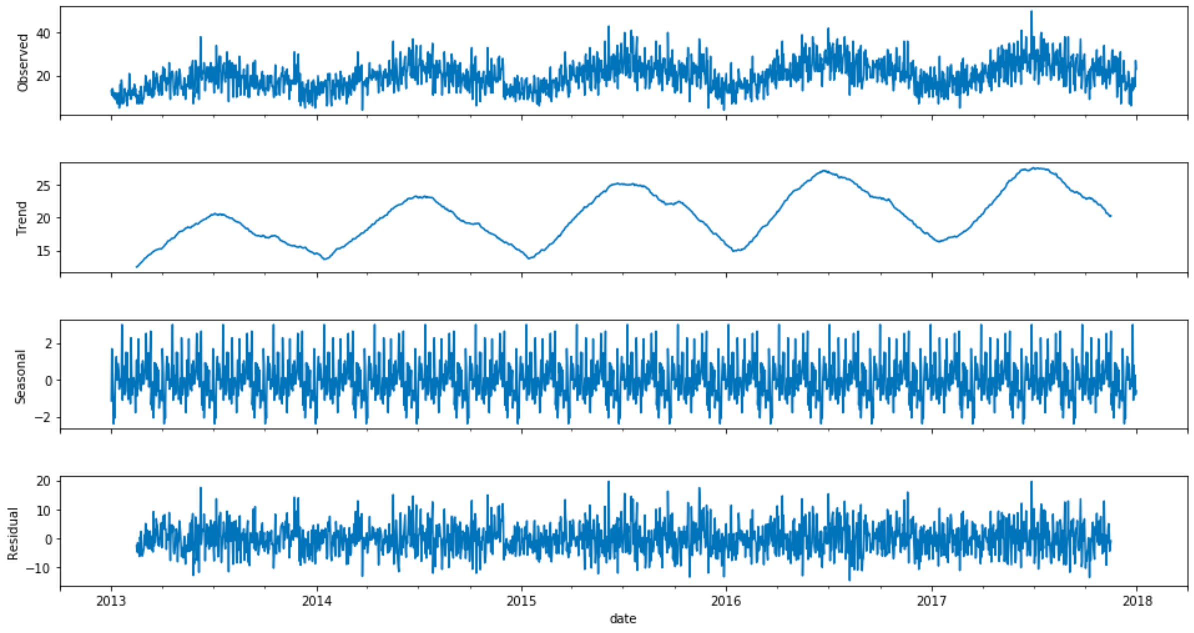
As we can see deciphering the above chart is very complicated because there are too many overlapping sales charts from across 50 different items and from 10 stores.

Therefore, in order to reduce the complexity of solving the problem, we will approach it by solving it for a store numbered 1 and item numbered 1, and try and find ways to generalize the solution. For the rest of this notes all the steps in solving the problem are executed against this store and item

.

We shall visualize the training sales data to see if there is seasonality and trends in Sales for the specific item.

```
residuals = sm.tsa.seasonal_decompose(itemstore.sales.dropna(),freq=90)
fig = residuals.plot()
fig.set_figheight(8)
fig.set_figwidth(15)
plt.show()
```



Observed chart: shows the sales for item 1 across a period of 5 years , the x-axis indicating the date of sales, and the y-axis shows the quantity of sales.

Trend chart: shows if there is a time-based trending of sales for the item 1 at store 1. As we can see from the chart that although there is seasonal changes to the item sales, the crest and troughs are rising across time. The Mean sales for the item/year is also higher than the mean sales for the same item the past year.

Seasonal chart: The seasonal chart indicates there is seasonality to the sales, we can see the sales rising through the first half of every year and then sales trends downwards as the year closes out. The observed seasonality pattern seems to repeat every year for which we have the data available.

Residual chart: A visual analysis of the chart indicates residuals seem to have a zero mean, but can't say if they are normally distributed or have a constant variance. I believe the graph is showing there is correlation between residuals, but we will analyze these as we go further.

### Algorithms and Techniques

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

- _Are the algorithms you will use, including any default variables/parameters in the project clearly defined?_
- _Are the techniques to be used thoroughly discussed and justified?_
- _Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?_

Average of Sales per Item per Quarter: It is a very simplistic method of predicting sales for the quarter based on the average of sales for the past quarters. Using Pandas group by command as below we can identify the sales per quarter per item

```
data['month'] = data['date'].apply(dateutil.parser.parse, dayfirst=False)
data['quarter'] = pd.PeriodIndex(data['month'],freq='Q')
data.groupby(['quarter','item'])['sales'].sum()
```

Once we have that grouped by , we can easily do an average on the individual item for all the quarters and use that as the potential sales for the test quarter. However, this is an extremely simplistic model because it does not consider sales trends. Below command gives us average sales / day for item1 from store1 , by averaging across the entire training period of 5 years.

```
print('Item 1 sold at Store1 per day averaged across the entire period of training data =>',
    (item1_store1_avg.groupby(['date'])['sales'].sum()).mean())
```

Now, we will explore how we can use ARIMA which stands for **Autoregressive Integrated Moving Average** model for the filtered sales data of item1 from store1.
ARIMA is a model for forecasting a time series which can be made to be “stationary” by differencing (if necessary). A random variable that is a time series is stationary if its statistical properties are all constant over time, such a series will have no trend, its variations around its mean have a constant amplitude.

The ARIMA forecasting equation for a stationary time series is a linear equation in which the predictors consist of lags of the dependent variable and/or lags of the forecast errors. If the predictors consist only of lagged values of Y, it is pure autoregressive model, however, in most cases there is also a noise factor(error) which makes it a non-linear regression model. So, coefficients in ARIMA models that include lagged errors must be estimated by nonlinear optimization methods like hill-climbing.

Lags of the forecast errors are called “moving average” terms, and a time series which needs to be differenced to be made stationary is said to be an “integrated” version of a stationary series.

A non-seasonal ARIMA model is classified as an “ARIMA(p,d,q)” model, where:

- P: is the number of autoregressive terms,
- D: is the number of non-seasonal differences needed for stationarity, and
- Q: is the number of lagged forecast errors in the prediction equation.

We will identify the values for p,d,q use ACF and PACF charts.

### Benchmark

The benchmark for our model will be to do better than the average of sales per item across a store across the entire training period.

I have performed this calculation for store 1 and for item 1 using the formula below and the result is as below
Item 1 sold at Store1 per day averaged across the entire period of training data => 19.97 => ~20 items/day
Item 1 sold at Store1 per Quarter averaged across the entire period of training data => 1823.4 =>~1834 /quarter

This analysis was performed using the pandas group by function

```
item1_store1_avg = data[(data.item==1)&(data.store==1)].copy()
print('Item 1 sold at Store1 per day averaged across the entire period of training data =>',
    (item1_store1_avg.groupby(['date'])['sales'].sum()).mean())
print('Item 1 sold at Store1 per Quarter averaged across the entire period of training data =>',
    (item1_store1_avg.groupby(['quarter'])['sales'].sum()).mean())
```

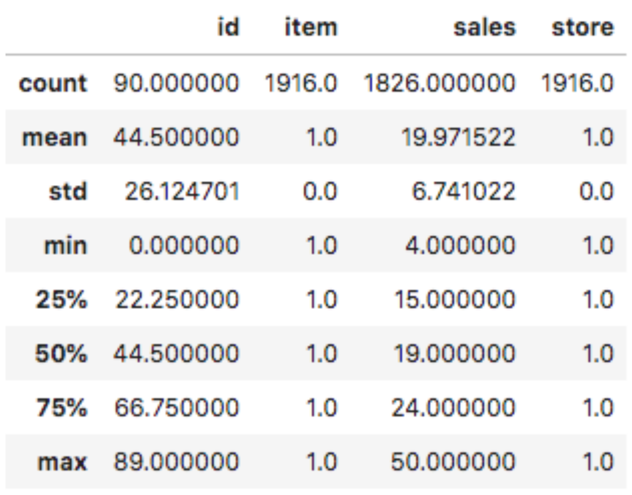
A better model will be able to forecast the sales for the item1 at Store1 which is closer to the actual sales than suggested by the average of sales across the training period.

## III. Methodology
(approx. 3-5 pages)

### Data Preprocessing

Initial visual data analysis of all sales data made me feel that I wasn’t getting it detailed enough to understand the underlying characteristics. It wasn’t easy to figure if the sales is autoregressive, so the preprocessing step I used was to do visual analysis on Item 1 from Store 1. Thereafter I have done the model development and forecasting model using that filter. The idea was to develop the model, and then find ways to generalize the model for all the items.

```
itemstore = total_df[(total_df.item==1)&(total_df.store==1)].copy()
itemstore.describe()
```



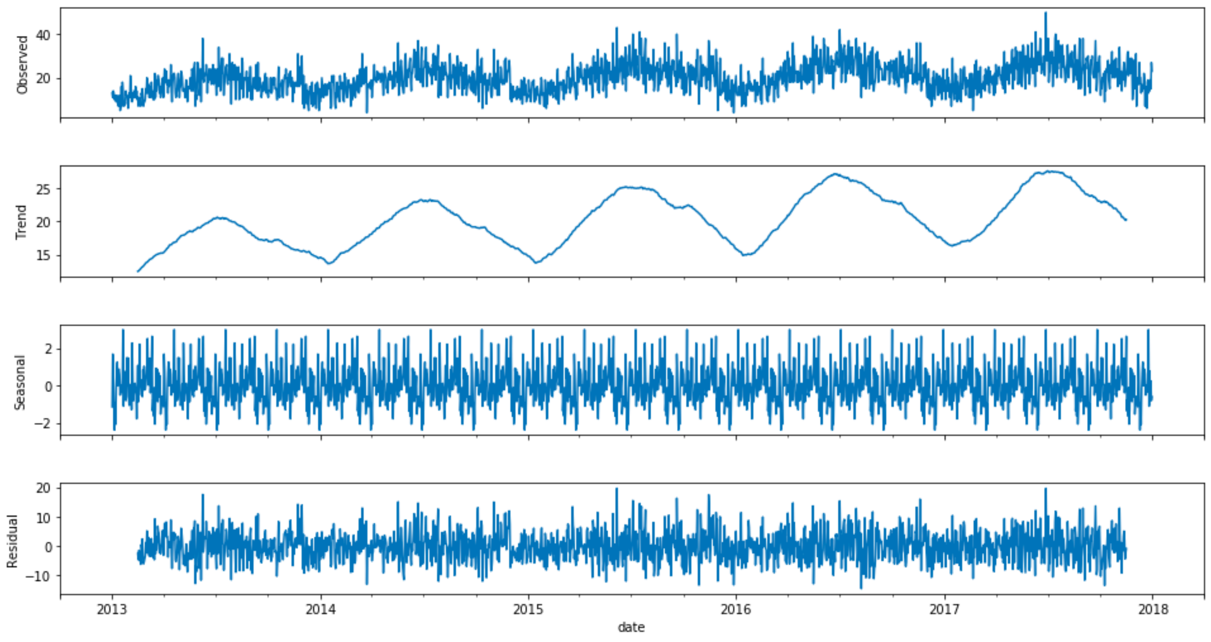
	id	item	sales	store
count	90.000000	1916.0	1826.000000	1916.0
mean	44.500000	1.0	19.971522	1.0
std	26.124701	0.0	6.741022	0.0
min	0.000000	1.0	4.000000	1.0
25%	22.250000	1.0	15.000000	1.0
50%	44.500000	1.0	19.000000	1.0
75%	66.750000	1.0	24.000000	1.0
max	89.000000	1.0	50.000000	1.0

### Implementation

Based on research around time-series forecasting , during the capstone proposal development phase I had identified ARIMA was the appropriate algorithm to be used.

ARIMA - Autoregressive Integrated Moving Average model can be applied on a stationary time-series. Also, i already stated in the pre-processing section that I decided to simplify the problem by implementing the algorithm to a specific item (Item 1) and the store 1. (We will refer to this dataframe as itemstore)

Now, we will visualize the data for itemstore dataframe using the seasonal_decompose method from the library statsmodels.api.

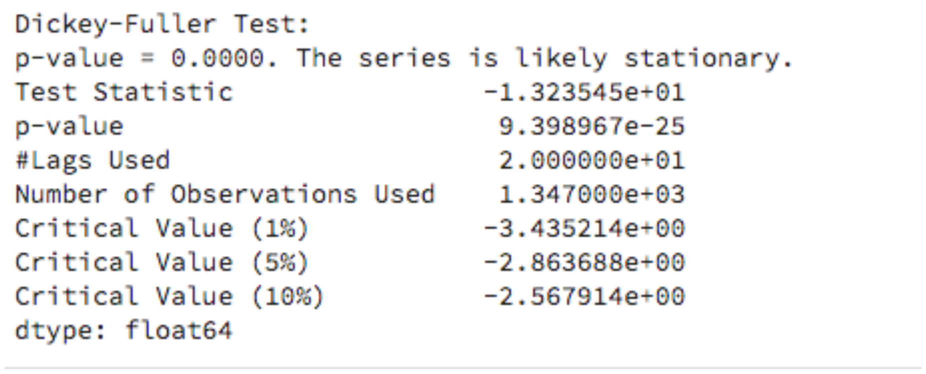


We can learn from the Trend chart that the sales data for itemstore has a positive trend and has no “stationarity” , and the seasonal chart indicates that there is a seasonality. We will further verify if the dataframe is stationary by using the Dickey-Fuller test in the function test_stationarity(timeseries_df,window).

Now, that the function confirms itemstore is non-stationary, we will perform steps to make the series stationary. In order to make the series stationary we will perform differencing. Differencing in statistics is a transformation applied to time-series data in order to make it stationary. A stationary time series' properties do not depend on the time at which the series is observed.

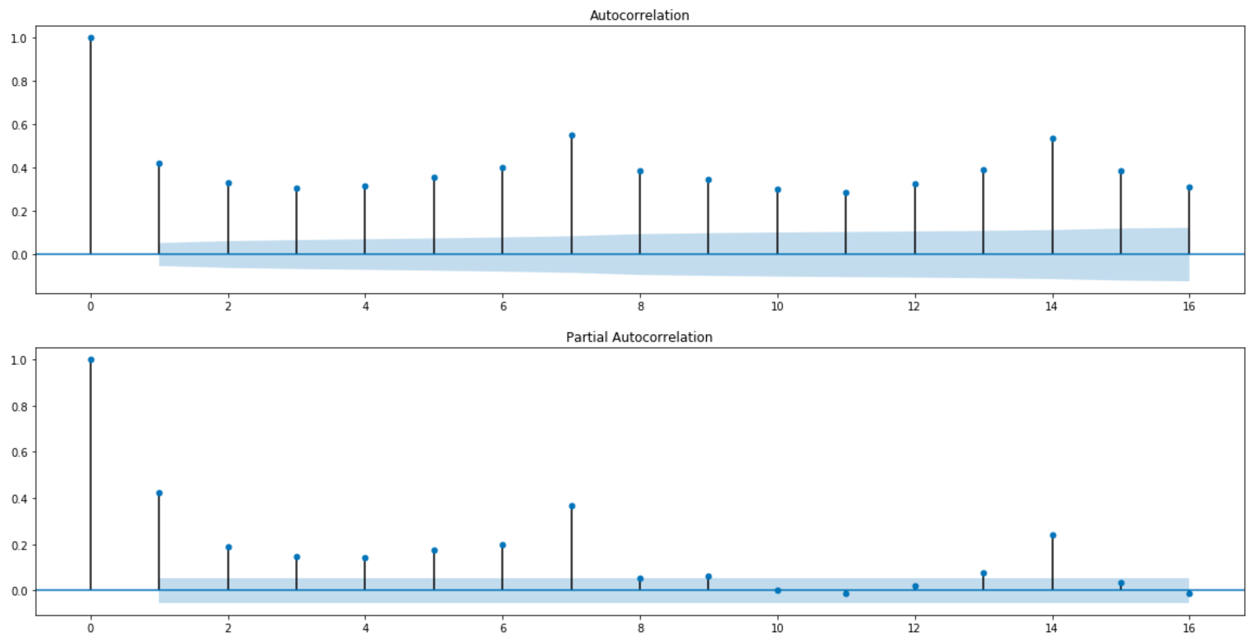
Once we perform differencing on itemstore , let us do ACF and PACF plots along with testing for stationarity.

The plots and output from Dickey Fuller test shown below indicates that the timeseries is stationary after first level differencing



```
Dickey-Fuller Test:
p-value = 0.0000. The series is likely stationary.
Test Statistic                 -1.323545e+01
p-value                         9.398967e-25
#Lags Used                      2.000000e+01
Number of Observations Used     1.347000e+03
Critical Value (1%)            -3.435214e+00
Critical Value (5%)            -2.863688e+00
Critical Value (10%)           -2.567914e+00
dtype: float64
```

Also, the ACF plot and PACF plot indicate that the seasonality for the sales data is 6 quarters, and the value of q-parameter is 7



Now , as we have identified the values of p,d,q we can run the ARIMA model.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
predictions = list()
model_sarimax =
sm.tsa.statespace.SARIMAX(train_store_sales,order=(6,1,7),enforce_invertibility=False,enforce_stat
ionarity=False)
model_fit = model_sarimax.fit()
```

Once, we fit the model on the training data , we can use it to forecast the sales for the testing data. I have performed forecasting for testing period of itemstore using the following steps. Because I had merged the training and testing data into a single dataframe for item1, I am identifying the beginning date from where I need to start forecasting

```
# predict using model.
start_forecast = 1349
end_forecast = 1441
train_store_sales['forecast'] = results_sarimax.predict(start = start_forecast,end = end_forecast,dynamic=True)
```

### Refinement

Based on the analysis of ACF and PACF plots , i decided to use 7,1,7 for parameters p,d,q respectively for the seasonal ARIMA model(SARIMAX).

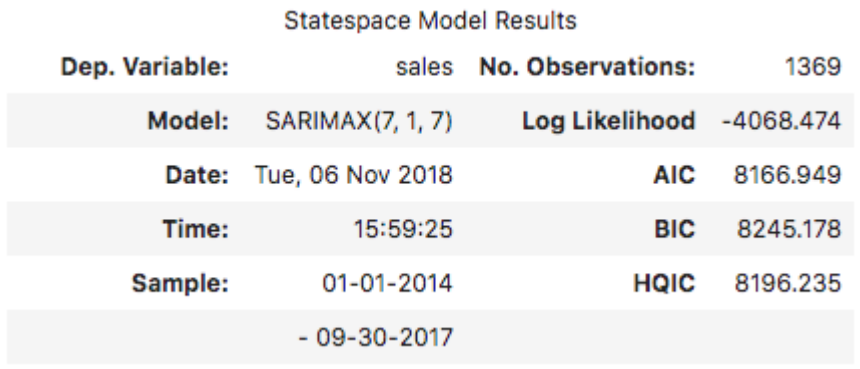
Based on further reading to understand the summary, I realized that Akaike Information Criteria (AIC) is a widely used measure of statistical model; and that the model with the lower AIC was a better model.
So, I ran the model fit again with 6,1,7 as p,d,q values and used that as the final model because it generated a lower AIC value.

## IV. Results
(approx. 2-3 pages)

### Model Evaluation and Validation

Based on the analysis of ACF and PACF plots , i decided to use 7,1,7 for parameters p,d,q respectively for the seasonal ARIMA model(SARIMAX).

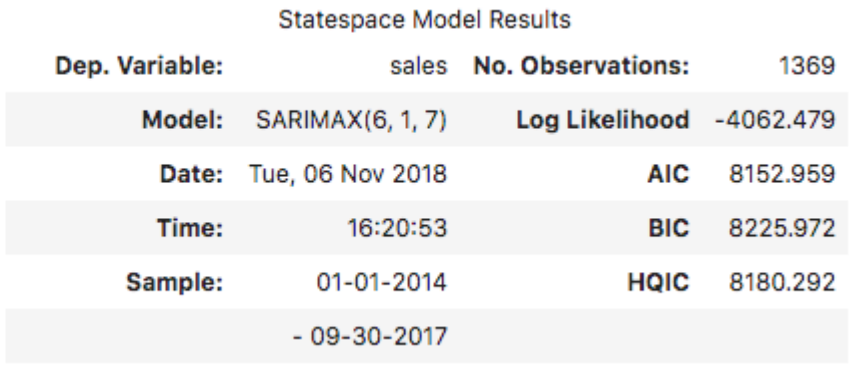
The output for SARIMAX.summary() with the above values for p,d,q is as below.



Statespace Model Results

Dep. Variable:	sales	**No. Observations:**	1369
Model:	SARIMAX(7, 1, 7)	**Log Likelihood**	-4068.474
Date:	Tue, 06 Nov 2018	**AIC**	8166.949
Time:	15:59:25	**BIC**	8245.178
Sample:	01-01-2014	**HQIC**	8196.235
	- 09-30-2017		

Based on further reading to understand the summary, I realized that Akaike Information Criteria (AIC) is a widely used measure of statistical model; and that the model with the lower AIC was a better model.
So, I ran the model fit again with 6,1,7 as p,d,q values and got the following summary



Statespace Model Results

Dep. Variable:	sales	**No. Observations:**	1369
Model:	SARIMAX(6, 1, 7)	**Log Likelihood**	-4062.479
Date:	Tue, 06 Nov 2018	**AIC**	8152.959
Time:	16:20:53	**BIC**	8225.972
Sample:	01-01-2014	**HQIC**	8180.292
	- 09-30-2017		

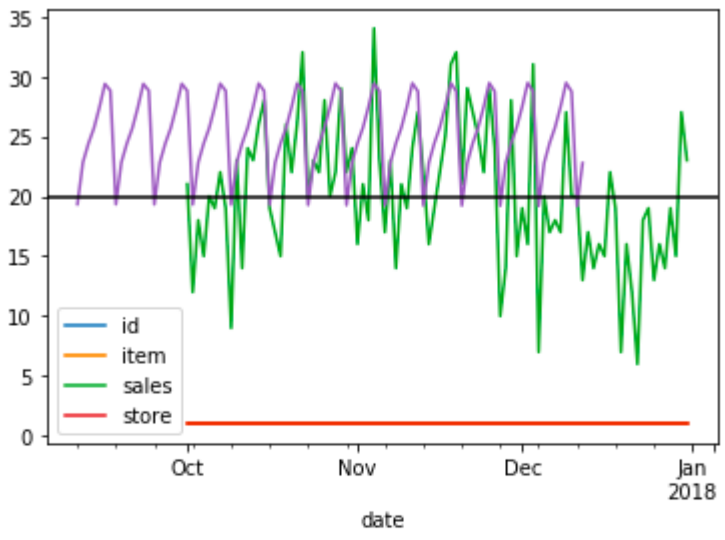
Decided to use the later, as values (6,1,7) for (p,d,q) generate lower AIC value.

Now, that we have done model fit with final parameters, we can do a forecast for the test period

```
# predict using model.
start_forecast = 1349
end_forecast = 1441
train_store_sales['forecast'] = results_sarimax.predict(start = start_forecast,end = end_forecast,dynamic=True)
```

I think visualization of actual and forecast data would be the most intuitive way to understand how the model is performing.

So I have overlapped forecasted sales (train_store_sales['forecast']) on the actual sales data over the test period.



Green Line - Actual Sales Data during the test period
Purple Line - Forecasted Sales Data during the test period.
Black Line - The average of sales for item=1 at Store=1 for the entire training period

### Justification

ARIMA model does perform better than the benchmark , because the former is able to forecast the trend of the sales for item 1 from store 1. ARIMA is able to forecast the direction of sales trend , unlike the benchmark which is a horizontal line that represents the average sales/day.

The Seasonal ARIMA model that I have developed analyzes the seasonality of the past sales, and is able to forecast sales to a fairly acceptable degree of confidence. However, I believe the solution can be improved by exploring other techniques like Hybrid ARIMA and Deep Neural networks.
Many research papers I have come across while working on this project indicate that Hybrid models can reduce the MSE errors and arrive at better models.

## V. Conclusion

Seasonal ARIMA does perform fairly well in forecasting time-series problems as evidenced by this project. As I had simplified the problem to a single item and store, I was able to research and arrive at a model .
However, I find given the significant amount of manual involvement needed to analyze the ACF and PCF plots, and identifying the parameters p,d,q values make it hard for ARIMA to scale when there are many time series to analyze. I believe using Hyper parameter optimization technique could be used to arrive at optimal p,d,q values faster.

This method of exploring data and developing the model across all stores and items, especially if sales for different items show different patterns seems to be a herculean task. I believe

### Improvement

Seasonal ARIMA is a very popular algorithm to model time series because it provides easy interpretability , and realistic confidence intervals; but there are certain disadvantages with ARIMA as it’s restrictive and multiple assumptions need to be made. However, the main drawback that I experienced is that it’s hard to scale the ARIMA algorithm when there are many different time-series need be analyzed and they may not have similar trends.

Based on additional reading, i see that we can use HYBRID ARIMA with Neural Networks to model time-series. Additionally other techniques like Exponential smoothing and Dynamic Linear Modeling can help in addressing some of the short-falls identified with ARIMA.

Before submitting, ask yourself. . .

- Does the project report you’ve written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?