# A STUDY OF TIME SERIES PROCESS FOR ESTIMATING THE PRODUCT SELLING


**THANAPOL PUNJAROJANAKUL**

**SCIM 6005700**

**DR. TANAPON TANTISRIPREECHA**

**PROJECT ADVISOR**

**BACHELOR OF INDUSTRIAL MATHEMATICS**

**FACULTY OF SCIENCE, MAHIDOL UNIVERSITY**



**SCIM407 INDUSTRIAL PROJECT SEMESTER 2, 2020**

**ABSTRACT**

The purpose of this project is to study a time series process for estimating product selling. time series data is the valuable information in the various fields and predict the value that can be happened in the future is a benefit for everyone who uses the data. This project is the study of how can we use the time series data process to predict value or in the short word we called Data Analysis.

The value that we can get from predicted will be the data to make a plan for the future and an easier to make decision prepare for the future. We can predict the future value by using the time series model. However, other than the model that I used in this project there has so many models that can be used and adapt to specific data but it can also provide how the time series model process work.

**INTRODUCTION**

Time series data is everywhere in this world it is used in a wide variety of fields. There are many datasets in every field which we would lie to predict the future. Knowing the future of data could be a benefit for everyone who uses the data, for examples if we can forecast how disease process team medic can predict how to take care of patient step by step; predicting consumer electricity energy demand could help power plant run more efficiently, and predicting how the frequency of earthquake happens can help MET. to warning people to decrease the risk of people will be affected by the earthquake. This project will look at the data of the quantity of an item that can be sold in the mart from 2011 to the middle of 2016 and forecast the future of quantity that may be sold in the future.

**Data Analysis**[2][3]

Data Analysis is the process of cleaning, transforming, and modeling data to find useful data and make use of it for business decision-making for example in this project we want to predict the future item will be sold in the future, of course, its accuracy can't be 100 percent but if we can know how it should be in the future we can make a plan to handle it better than not knowing it.
A simple example that can definite the Data analysis is whenever we decide our day-to-day life by thinking about what happened last time or what will happen by choosing a particular decision.

In short words, it is the process of analyzing past or future data to make a decision based on it.

Data Analysis has many tools to make it easier for users to process and manipulate data, analyze the relationships and correlations between data sets to identify patterns and trends for interpretation. In this project, I will use the Python programming language.

**Time Series[2]**

Time Series data is the data that form structured data in many fields such as finances, stocks, economics, physics, etc. Anything data that need to observe or measure many points in time forms. The most of time series will be "fixed frequency" for example time series contain the data of items that can be sales during the day the frequency of this example is 1 day or 24 hours. The frequency can be every 10 seconds, once per month or once per year, but some time series can be irregular one that not fixed unit of time or offset between the unit of time (something like the frequency is 10 minute but some part is 5 minute instead 10 minutes)

The point that we can tell this is the time series data is

1.Time in the data we will call it Timestamps, specific instants time

2.Fixed periods, such as the entire months or the entire years

3.Intervals of time, indicated by a start and end of timestamps.

4.Elapsed time; each timestamp is a measure of time relative to a particular start time

**THE DATA[1]**

The Raw Data of this project is from the website "Kaggle" one of the websites that contain the database. The website opens freely for everyone who wants to use it so. The data is about the quantity of an item that can be sold in Walmart. Data will contain the historical daily unit sales data per product and store and the dates of data. I will use this data to forecast the quantity that will be sales in the future by using the time series model AR, MA,ARMA, and ARIMA.

| | date | wm_yr_wk | weekday | wday | month | year | d | event_name_1 | event_type_1 | event_name_2 | event_type_2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2011-01-29 | 11101 | Saturday | 1 | 1 | 2011 | d_1 | NaN | NaN | NaN | NaN |
| 1 | 2011-01-30 | 11101 | Sunday | 2 | 1 | 2011 | d_2 | NaN | NaN | NaN | NaN |
| 2 | 2011-01-31 | 11101 | Monday | 3 | 1 | 2011 | d_3 | NaN | NaN | NaN | NaN |
| 3 | 2011-02-01 | 11101 | Tuesday | 4 | 2 | 2011 | d_4 | NaN | NaN | NaN | NaN |

**(Calendar files from raw data)**

| | id | item_id | dept_id | cat_id | store_id | state_id | d_1 | d_2 | d_3 | d_4 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | HOBBIES_1_001_CA_1_validation | HOBBIES_1_001 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 1 | HOBBIES_1_002_CA_1_validation | HOBBIES_1_002 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 2 | HOBBIES_1_003_CA_1_validation | HOBBIES_1_003 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 3 | HOBBIES_1_004_CA_1_validation | HOBBIES_1_004 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 4 | HOBBIES_1_005_CA_1_validation | HOBBIES_1_005 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |

**(Sales file from raw data)**

**Re-Arrange Data**

In real life, the data will not be arranged to can use to train in the model immediately. The first thing that we need to know the answer to “What do you want to do with this data” in this project I want to forecast how many quantities of the item can be a sale by martin future so my answer is “How many quantities of an item can be sale in one-year later”. So my data will need “Date” and “Quantity of item that sale during a date”

After we know what data and what attributes that we need to use we will make DataFrame that contains all attributes and data that we need to use in the single DataFrame to make it easier to use. The CSV files that I need to use is “Calendar” that contains the date of data and “Sales_train” that contain all about the quantity of the item that can be sales during the da

| | date | wm_yr_wk | weekday | wday | month | year | d | event_name_1 | event_type_1 | event_name_2 | event_type_2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2011-01-29 | 11101 | Saturday | 1 | 1 | 2011 | d_1 | NaN | NaN | NaN | NaN |
| 1 | 2011-01-30 | 11101 | Sunday | 2 | 1 | 2011 | d_2 | NaN | NaN | NaN | NaN |
| 2 | 2011-01-31 | 11101 | Monday | 3 | 1 | 2011 | d_3 | NaN | NaN | NaN | NaN |
| 3 | 2011-02-01 | 11101 | Tuesday | 4 | 2 | 2011 | d_4 | NaN | NaN | NaN | NaN |

**(Calendar files from raw data)**

| | id | item_id | dept_id | cat_id | store_id | state_id | d_1 | d_2 | d_3 | d_4 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | HOBBIES_1_001_CA_1_validation | HOBBIES_1_001 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 1 | HOBBIES_1_002_CA_1_validation | HOBBIES_1_002 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 2 | HOBBIES_1_003_CA_1_validation | HOBBIES_1_003 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 3 | HOBBIES_1_004_CA_1_validation | HOBBIES_1_004 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |
| 4 | HOBBIES_1_005_CA_1_validation | HOBBIES_1_005 | HOBBIES_1 | HOBBIES | CA_1 | CA | 0 | 0 | 0 | 0 | ... |

**(Sales file from raw data)**

We will combine the two files into one DataFrame that contains only “Date” and “sum of quantity item during day”

| | date | sum |
|---|---|---|
| 0 | 2011-01-29 | 14195.0 |
| 1 | 2011-01-30 | 13805.0 |
| 2 | 2011-01-31 | 10108.0 |
| 3 | 2011-02-01 | 11047.0 |
| 4 | 2011-02-02 | 9925.0 |
| 5 | 2011-02-03 | 11322.0 |
| 6 | 2011-02-04 | 12251.0 |
| 7 | 2011-02-05 | 16610.0 |
| 8 | 2011-02-06 | 14696.0 |
| 9 | 2011-02-07 | 11822.0 |

**(After re-arrange CSV files)**

Next is because we want to use Time series model so we need to change date into timestamps .

```
raw_data.date=pd.to_datetime(raw_data.date) #Change Column into Timestamp
raw_data.set_index('date',inplace=True) #change timestamps to index of DataFrame
```

**(change date into Timestamps and change timestamps to be index of DataFrame)**

| | sum |
|---|---|
| **date** | |
| **2011-01-29** | 14195.0 |
| **2011-01-30** | 13805.0 |
| **2011-01-31** | 10108.0 |
| **2011-02-01** | 11047.0 |
| **2011-02-02** | 9925.0 |
| **2011-02-03** | 11322.0 |
| **2011-02-04** | 12251.0 |
| **2011-02-05** | 16610.0 |
| **2011-02-06** | 14696.0 |
| **2011-02-07** | 11822.0 |

**(The DataFrame that prepare completely to train in time series model)**

That the first step before training in the time series model.

Next, we need to check the data is stationary or not?

**What Is Stationarity?**[2][13]

A seasonal time series has patterns that repeat at regular intervals, for example, high sales every weekend. We need to make data into a stationary form that the distribution of data doesn't change with time.

We can check it stationary or not in many ways but in this project I will use the most common test for identifying whether a time series is non-stationary is called ADF or Augmented Dickey-Fuller.

**What is ADF?**[12][13]

ADF is a statistical test, where the null hypothesis is that your time series is non-stationary due to trend.

The augmented Dickey-Fuller (ADF) statistic, used in the test, is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.

I will use the ADF method to find the p-values of this data

```
#Compute and print ADF p-value to check it stationary or not
result=adfuller(df['sum'])
print("The p-value for the ADF test is",result[1],'')
```

```
The p-value for the ADF test is 0.39303545897323716
```

**(Compute Data by using ADF method and print p-values)**

**What is the p-value?**[12]

In short word, the p-value is the probability of the results of a statistical hypothesis test, by assuming that the null hypothesis is correct. The p-value is used to reject points to provide the smallest level of significance at which null hypothesis would be rejected, in other words, **if p-values are small that means it is the stronger evidence in favor alternative hypothesis.**

A p-value is a measure of the probability that an observed difference could have occurred just by random chance.

The lower the p-value, the greater the statistical significance of the observed difference.

P-value can be used as an alternative to or in addition to pre-selected confidence levels for hypothesis testing.

From the picture of the ADF test above the p-values of this data is 0.39303545897323716 so we can't reject the hypothesis. if we want to reject this hypothesis the p-values should less than 0.05 so it's mean this data is not stationary

**How can we make data to stationary?**

We can transform this data to make it stationary. The simplest way and normally use it take a differential of this data.

```
#Take first difference of the data to make it stationary
chg_quantity=df.diff()
#after take differential the some row of data will turn to be NaN so we need to drop NaN column to prevent error
chg_quantity=chg_quantity.dropna()
```

**(Take differential of this Data)**

**After take differential we will check p-value again**

```
result=adfuller(chg_quantity['sum'])
print("The p-value for the ADF test is",result[1],'')
```

```
The p-value for the ADF test is 0.0
```

**(P-value of data after take differential)**

As you can see the p-value of this data is 0.0 now it's less than 0.05 so we can reject the hypothesis. After we make data to stationary next is to train this data in the model.

**AR Model**[13][14]

AR or Autoregressive model in this model we regress the values of the time series against previous values of this same time series. We only use past data to the model. The process is a linear regression of the data in the current series against one or most past values in the same series.

AR(p) models are autoregressive model-specific lagged values of are used as predictor variables. Lags are where results from one time period affect the following periods. AR(p) models try to explain the momentum and mean reversion effects often observed in trading markets (market participant effects).

"p" called the order is the number of time lags used. For example, an AR(1) would be called a "first-order autoregressive model" The outcome variable in the first-order AR process at some point in time "***t***" is related to only one time period(i.e. the values at "***t-1"***) same as second or three order that will be related to only two or three periods apart.

AR(1) or first-order AR Model be like:

$$y_t = c + \emptyset_1 y_{t-1} + \in_t$$

$y_t = The\ value\ of\ time\ series\ at\ time\ t$

$a_1 = the\ autoregressive\ coefficient\ at\ lag\ one$

$y_{t-1} = The\ value\ of\ time\ series\ at\ previous\ step$

$\in_t$ = shock term, the shock term is white noise(time series that show no autocorrelation), meaning each shock is random not related to the other shocks in the series.

C = constant

For an AR(1) model:

When $\emptyset_1 = 0$, $y_t$ is equivalent to white noise;

When $\emptyset_1 = 1$ and c = 0, $y_t$ is equivalent to a random walk

When $\emptyset_1 = 1$ and c $\neq$ 0, $y_t$ is equivalent to a random walk with drift;

When $\emptyset_1 < 0$, $y_t$ tends to oscillate around the mean

The coefficient $a_1$ is just the slope of the line and the shocks are the residuals to the line.

AR Model can be extended to include more lagged values and more phi parameters

AR(2) Model be like:

$$y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \in_t$$

**AR(p) Model be like:**

$$\boldsymbol{y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \cdots + \emptyset_p y_{t-p} + \in_t}$$

**We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required.**

**For an AR(1) model: -1 < $\emptyset_1$ < 1**

**For an AR(2) model: -1 < $\emptyset_2$ < 1, $\emptyset_1 + \emptyset_2 < 1$, $\emptyset_1 - \emptyset_2 < 1$.**

**When p $\geq$ 3, the restrictions are much more complicated that why we need to use program to take care of complicated things.**

**MA Model**[13][15]

MA models or Moving Average models we regress the values of the time series against the previous shock values of this same time series**.** MA(q) models try to capture the shock effects observed in the white noise terms. These shock effects could be thought of as unexpected events affecting the observation process e.g. Surprise earnings, wars, attacks, etc.

"q" called the order is the number of time lags used. For example, an MA(1) would be called a "first-order moving average model" The outcome variable in the first-order AR process at some point in time "***t***" is related to only one time period(i.e. the values at "***t-1"***) same as second or three order that will be related to only two or three periods apart.

MA(1) model or first order of MA model be like:

$$y_t = c + m_1 \in_{t-1} + \in_t$$

$y_t = the\ values\ of\ the\ time\ series$

$m_1 = parameter\ of\ the\ model$

$\in_t = the\ values\ of\ shock\ for\ the\ current\ step$

$\in_{t-1}$= *the values of shock at the previous step*

*C = the mean of the series*

MA(2) model be like:

$$y_t = c + m_1 \in_{t-1} + m_2 \in_{t-2} + \in_t$$

**MA(q) model be like:**

$$\boldsymbol{y_t = c + m_1 \in_{t-1} + m_2 \in_{t-2} + \cdots + + m_q \in_{t-q} + \in_t}$$

**The constraints for other models are similar to the stationarity constraints.**

**For an MA(1) model: -1 < $\emptyset_1$ < 1**

**For an MA(2) model: -1 < $\emptyset_2$ < 1, $\emptyset_1 + \emptyset_2$ < 1, $\emptyset_1 - \emptyset_2$ < 1.**

**When q $\geq$ 3, the restrictions are much more complicated that why we need to use program to take care of complicated things.**

**ARMA Model**[13][16]

**ARMA model or Auto Regressive Moving Average(ARMA) models, is used to describe weakly stationary stochastic time series in terms of two polynomials first is AR(Autoregressive) and MA(Moving Average)**. In short words, the ARMA model is simply the merger between AR(p) and MA(q) models to attempts to capture both of these two aspects by regressed on the previous values and the previous shock terms when modeling financial time series.

ARMA(1,1) model is the combination between AR(1) and MA(1):

$$\boldsymbol{y_t = \emptyset_1 y_{t-1} + m_1 \in_{t-1} + \in_t}$$

**ARIMA Model**[13][17]

ARIMA, short for 'Auto-Regressive Integrated Moving Average' is similar to the ARMA model but what sets ARMA and ARIMA apart is **differencing**. ARMA model is the model for the data that already is stationary; if your data is not stationary we can make the data stationary by taking differencing to make it stationary. The "I" that stands in the ARIMA model is integrated; It is a measure of how many times you take your data to take differences to make data stationery. If you don't need to take differencing to make it stationary the ARIMA model becomes a simply ARMA model.

An ARIMA(p,d,q) model is characterized by 3 terms: p, d, q

where,

p is the order of the AR term(same as the ARMA model)

q is the order of the MA term(same as the ARMA model)

d is the number of differences required to make the time series data into stationary

**How can we know the correct model order?**[17]

The important part of these Models is the model order. The model order is very important to the quality of forecasts, naturally, we don't know what order is the best for our model that why we need to identify the correct model order.

One of the main ways to identify the correct model order is by using the autocorrelation function in another name the ACF, and the partial autocorrelation function in another name the PACF.

**ACF**

when we talk about the autocorrelation function we mean the set of correlation values for different lags

The autocorrelation function at lag-1 is the correlation between a time series and the same time series offset by one step

lag-1 autocorrelation → $corr(y_t, y_{t-1})$

lag-2 autocorrelation → $corr(y_t, y_{t-2})$

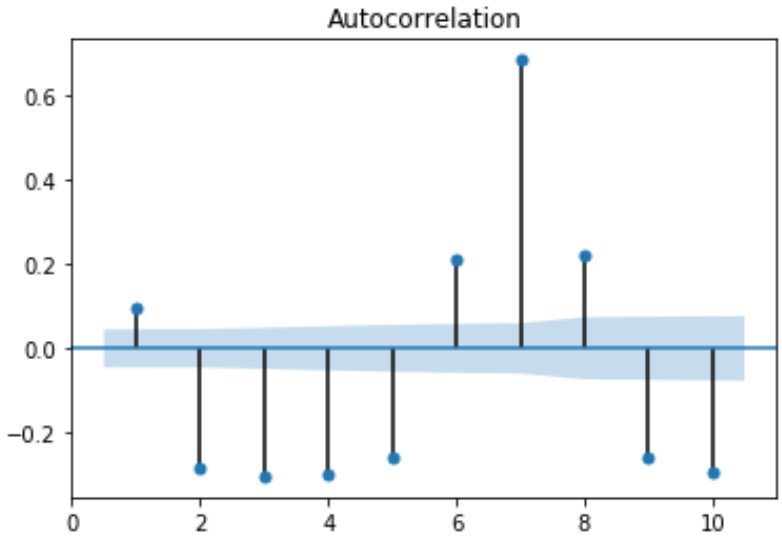
.

.

.

lag-n autocorrelation → $corr(y_t, y_{t-n})$

We can plot autocorrelation function to get an overview of the data



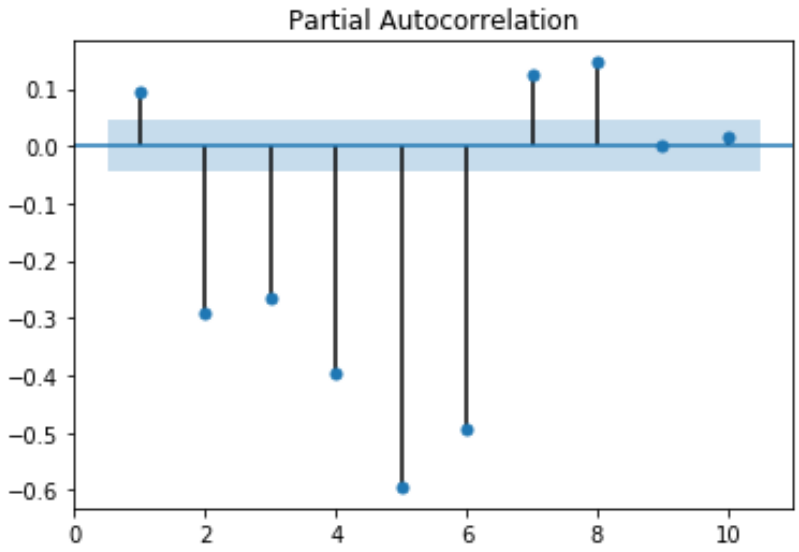

**(Autocorrelation plot)**

**(If these values are small and inside the blue shade region, then they are not statistically significant.)**

**PACF**

The partial autocorrelation is the correlation between a time series and the lagged version of itself after we subtract the effect of correlation at smaller lags. In short words, it is the correlation associated with a particular lag.

The partial autocorrelation function is series of values and we can plot it to get another view of data.




**(Partial Autocorrelation function plot)**

By comparing between the ACF and PACF we can deduce the model

How to identify

| | ACF | PACF |
|---|---|---|
| AR(p) | Tails off | Cuts off after lag p |
| MA(q) | Cuts off after lag q | Tails off |
| ARMA(p, q) | Tails off | Tails off |

The figure above it shows ACF and PACF tails off immediately after one lag. So the model that suitable to use is the ARMA model.

If ACF and PACF are hard to identify there is another way to identify the right order of ARMA model that is called Information Criteria.

**What are Information Criteria?** [13]

Information Criteria: adjusts goodness-of-fit for number of parameters

Two popular adjusted goodness-0f-fit measure are

1.AIC(Akaike Information Criteria)

2.BIC(Bayesian Information Criteria)

**What is AIC?**

The Akaike information criterion or AIC is a metric that tells us how good a model is. A model which makes better predictions is given a 'lower AIC score'.

The AIC also penalizes models which have lots of parameters. If we set the order too high compared to the data, we will get a high AIC value and will stop getting overfitting training data.

**What is BIC?**

The Bayesian information criterion or BIC this method is very similar to the AIC. Models which fit the data better have lower BIC and the BIC penalizes overly complex models.

The BIC will penalize additional model orders.

```
order_aic_bic=[]
#loop over AR order
for p in range(10):
    #Loop over MA order
    for q in range(10):
        #fit model
        model=SARIMAX(chg_quantity,order=(p,1,q))
        results=model.fit()
        #add order and score to list
        order_aic_bic.append((p,q,results.aic,results.bic))
```

**(The loop to find p and q in range(0 – 9) the loop will continuously change value of p and q and will be record in the "order_aic_bic")**

```
#make DataFrame of model order(p,q ,AIC,BIC)
order_df=pd.DataFrame(order_aic_bic,columns=['p','q','AIC','BIC'])
```

**(Make it do DataFrame for easy to use)**

The AIC and BIC will often choose the same model, but if they choose different model it time for us to choose model.

```
#sort by AIC
print(order_df.sort_values('AIC').head())
```

```
   p  q         AIC         BIC
89  8  9  33191.605134  33291.602009
69  6  9  33191.684134  33280.570245
99  9  9  33196.498817  33302.051074
78  7  8  33234.684004  33323.570115
98  9  8  33245.720766  33345.717641
```

**(Sort values of AIC's values from smallest to largest)**

```
#sort by BIC
print(order_df.sort_values('BIC').head())
```

```
   p  q         AIC         BIC
69  6  9  33191.684134  33280.570245
89  8  9  33191.605134  33291.602009
99  9  9  33196.498817  33302.051074
78  7  8  33234.684004  33323.570115
68  6  8  33255.008242  33338.338971
```

**(Sort values of BIC's values from smallest to largest)**

From the picture above we can see if we base on AIC's values the best order of p and q will be ARMA(8,9) but if we base on the BIC's values the best order of p and q will be ARMA(6,9). Then what order we need to choose.

We will choose the order based on what we need to use this model to do.

**AIC is better at choosing predictive models.**

**BIC is better at choosing a good explanatory model.**

So we need to use this model to forecast the values of the future so we will choose based on the AIC's values.

After we check the model by ACF and PACF method and choose order by using Information criteria the best model that suitable for this data will be ARMA(8,9) but because this data is not stationary from the beginning so we need to take differencing to make it stationary so that means we need to use ARIMA model instead of ARMA model. We take differencing 1 time so the model will be ARIMA(8,1,9).

**Train the model and forecast**

We already know the best model to use in this data will be ARIMA(8,1,9) so the next step is we need to train the model based on this order.

```
#Created Model for ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
model=SARIMAX(chg_quantity['sum'],order=(8,1,9))
#fit model
results=model.fit()
#make forecast
diff_forecast=results.get_forecast(steps=365).predicted_mean
```

**(we fit the model and forecast future steps=365 means predicts next 365 values)**

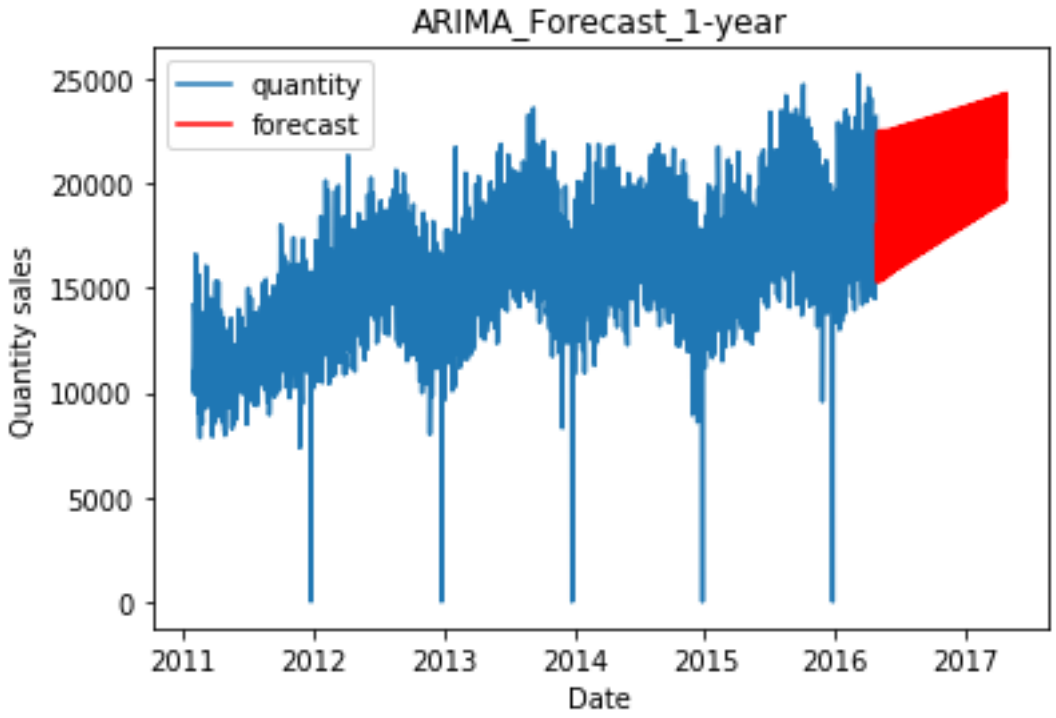
After we finish predict values the other thing that needs to remind is the data that the model used to train is the data after take differencing so the values that we predicted were the values we predicted from differencing of data. So we can't use this value.

That's why we need to revert the data to become the same as before data change after take differencing

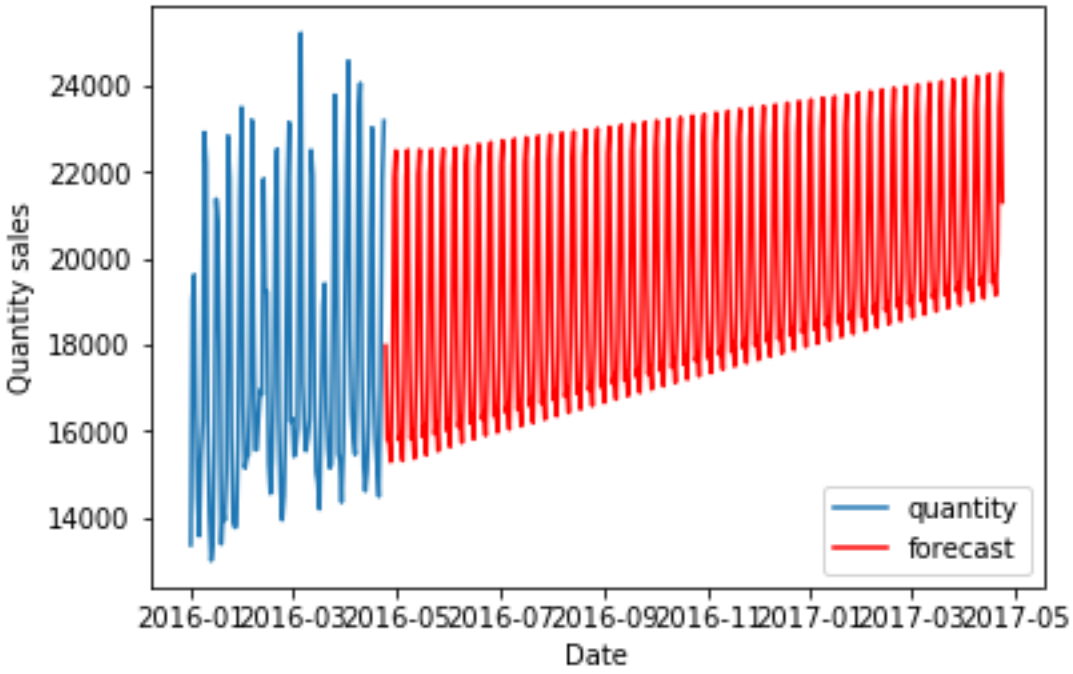
```
#reconstruct data
from numpy import cumsum
mean_forecast=cumsum(diff_forecast)+df.iloc[-1,0]
```

**(we will use cumsum command to revert data back before differencing values the values that we predicted will return as well)**

After we revert it back completely we will plot it in graph to make it easily to see.




**(This is the graph of quantity of item that can be sales the red line is the one that we forecast)**




**(The graph plot to focus on 2016(the end of raw data) to 2017(the values that we forecast))**

that it's the end of the forecast part, but before we went to use the values that we predicted we need to confirm one thing. The next question is how could we confirm our model is behaving well. We will confirm it with the model diagnostics. After we have a final model we will check how good that model. This is the most important part of the building model to diagnose our model we focus on the residuals of the training data.

The residuals are different between our model's prediction and the real values of the time series data. If the model fits well the residuals should be uncorrelated white Gaussian noise centered on zero.

```
#Assign residuals to variable
residuals=results.resid #results from above
```

**(Use the "results(variable that we fit the model)" )**

**How large the residuals are and so how far our predictions are from the true values?**

We will answer this question by calculating the mean absolute error of the residuals. The mean absolute error(MAE) will help us to determine the accuracy of the forecast.

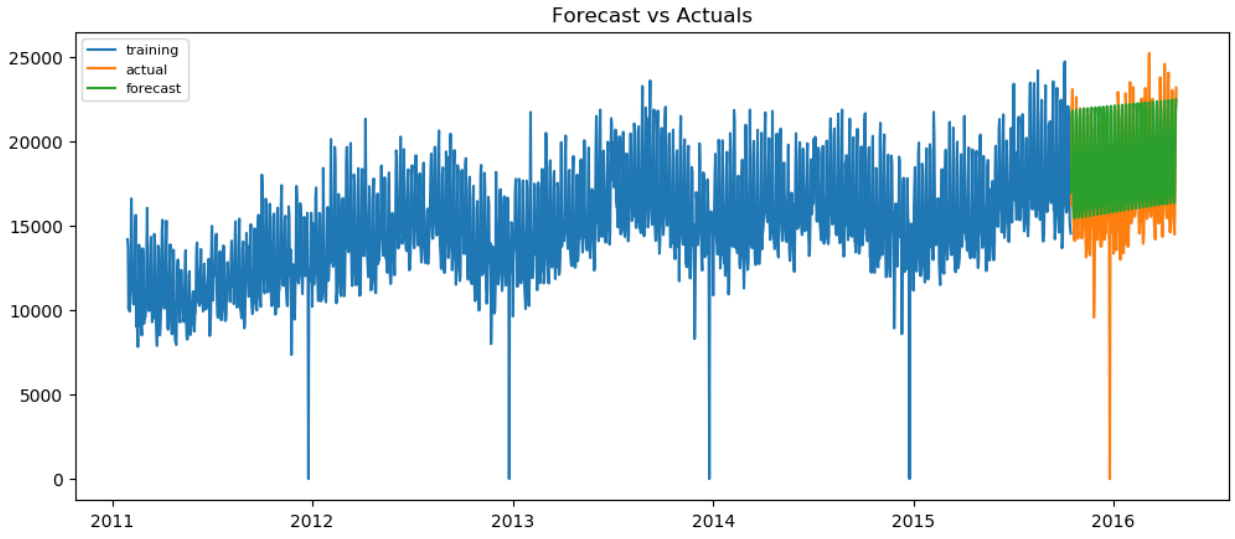
```
#calculate the mean absolute error
mae=np.mean(np.abs(residuals))
print(mae)
```

```
932.2817818611106
```

**(the mean absolute error of variable "residuals")**

and that's is the end of this project but before we went to the conclusion I want to show the other ways that we can check the error between our model's predicted values and the real values of the time series data.

**ERROR ANALYSIS**




**(The Graph of predicted value compare to real values by using ARIMA model)**

This graph shows how train-test split(Train 80% Test 20%). This graph compares the real data and the forecast data that we predict by using the ARIMA model that uses 80% of actual data to predict 20% of data. After we compare the forecast data and the actual data it's has a big chance of data this means my model can't predict it perfectly, but its location is between the actual data it's not far above or far below the actual data graph. The Mean absolute error of this graph is 1449.6321507882883.

**ROOM OF IMPROVEMENT**

To tell the truth my works is hard to describe the best forecast you can see that in the picture above in the error analysis by using train test split. After all my model is not a perfect model that can be used to forecast it completely it has a lot of errors the best thing that I can describe in my model is at least it's not far away from actual data. From my perspective Time Series data can be used in many model the ARIMA model that I used is one of the many models that can be used in the time series maybe the other model can predict this data better than the ARIMA model or maybe I need to improve the data I used more than this to make ARIMA model predict better than this.

**CONCLUSION**

After we look at collecting data and train it with the Time series model we can see an overall image of the ARIMA model and how time series work. In reality, almost everything in this world is a time series and can be used to forecast the future values to use the forecast values to make a plan to handle the situation before it's happened.

Data analysis is one of the most important parts of a company or institution. Because the values that we forecast can be used to decide for business or can warn us of something that can be happened in the future to make time for us to prepare it before that time come.

Of course, it's nearly impossible to can predict values that have 100% accuracy because in reality, we have many attributes and the random events can appear everywhere every time to make our predicted value change, but the best thing of the time series forecast is that we can have time to prepare ourselves before.

**Why choose Python programming language?** [2][4]

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is one of the popular programming languages because it is easy to use without a complication step. For data analysis and data visualization, Python compares to other programming languages, such as R, MATLAB, SAS, Stata, and others. In recent years, Python's improved support from libraries(such as pandas and NumPy) has made it a popular choice for doing data analysis tasks.

**What is NumPy?** [2][5][6]

NumPy or Numerical Python is a library for the Python programming language. It provides the data structures, algorithms, and library glue need for most scientific applications involves numerical data in Python.

One of its primary uses in data analysis is used to store the data to be passed between algorithms and libraries for numerical data NumPy arrays are more efficient for storing and manipulating data. The goods thing of NumPy is the low-level programming language, such as C and Fortran can operate in NumPy array without the need to copy data to other memory storage.

**What is Pandas?** [2][7]

Pandas is a software library written for the Python programming language for data manipulation and analysis. Pandas will provide data structures and functions designed to work with the tabular data that can make it is faster and easier to use. The thing that will be used mostly in this project is DataFrame, Series and the column-oriented data structure with both row and labels.

Pandas have a good performance with the array-computing ideas of NumPy with the flexible data manipulation capabilities of spreadsheets and rational database(such as SQL)

**Matplotlib**[2][8][9]

Matplotlib is the most popular Python library for creating static, animated, and interactive visualizations in Python. It designs for creating plots for publication. Compare to the other visualization libraries in Python libraries, matplotlib is the most widely used and become the default visualization tool when it comes to visualization Compare to MATLAB Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source.

**Statsmodels**[2][10][11]

Statsmodels is a Python module that provides classes and functions for the estimation of many different statsmodels, as well as for conducting statistical tests and statistical data exploration. Statsmodels contain algorithms for statistics and econometrics. Statsmodels is part of the Python scientific stack that is oriented towards data analysis, data science, and statistics.

In this project, we will use the module Time Series Analysis in the statsmodels.

Time Series Analysis: AR, MA,ARMA, ARIMA models

Statsmodels is focused on statistical inference, providing uncertainty estimates and **'p-value'** for parameters to use with Numpy and Pandas.

# REFERENCE


[1]: M5 forecasting(https://www.kaggle.com/c/m5-forecasting-accuracy/overview)

[2]: Python for Data Analysis by Wes McKinney published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472

[3]: What is Data Analysis(https://www.guru99.com/what-is-data-analysis.html)

[4]: What is Python? Executive Summary(https://www.python.org/doc/essays/blurb/)

[5]: NumPy Wikipedia(https://en.wikipedia.org/wiki/NumPy)

[6]: What is NumPy(https://numpy.org/doc/stable/user/whatisnumpy.html)

[7]: Pandas Wikipedea(https://en.wikipedia.org/wiki/Pandas_(software))

[8]: Matplotlib Wikipedea(https://en.wikipedia.org/wiki/Matplotlib)

[9]: Matplotlib:Visualization with the Python(https://matplotlib.org/)

[10]: Statsmodels(https://www.statsmodels.org/stable/index.html)

[11]: Statsmodels Wikipedea(https://en.wikipedia.org/wiki/Statsmodels)

[12]: p-value(https://www.investopedia.com/terms/p/p-value.asp)

[13]: Forecasting Principles and Practice by Rob J Hyndman& Feorge Athanasopoulas

[14]: What is an Autoregressive model?( https://www.statisticshowto.com/autoregressive-model/)

[15]: What is a Moving Average model?( https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/moving-average/)

[16]: What is ARMA model(https://www.statisticshowto.com/arma-model/)

[17]: Time series analysis: identifying AR and MA using ACF and PACF plots(https://towardsdatascience.com/identifying-ar-and-ma-terms-using-acf-and-pacf-plots-in-time-series-forecasting-ccb9fd073db8)

[18]: Using Mean Absolute Error for forecast accuracy(https://canworksmart.com/using-mean-absolute-error-forecast-accuracy/)