Crimes in Boston

By Dvir Hachoen and Yuval amar

In kohelet written “What has been is what will be, and what has been done is what will be done, and there is nothing new under the sun”(kohelet 9,1)

In this article we will try to see if this that said thousands of years from now is true.

This dataset took from “Kaggle” (link: https://www.kaggle.com/AnalyzeBoston/crimes-in-boston) The dataset contains crime records from June 2015 to September 2018, the collected data provided by the “Boston Police Department” (BPD). The dataset contains 319073 rows and 17 columns.

Data description:

- INCIDENT_NUMBER – (Object) unique incident number
- OFFENSE_CODE- (Int64) unique police offense code.
- OFFENSE_CODE_GROUP-(Object) unique police offense code group.
- OFFENSE_DESCRIPTION – (object)short offense description of the crime.
- DISTRICT- (Object) unique value of the district.
- REPORTING_AREA- (Object) unique value of the reporting area.
- SHOOTING- (Object) contains ‘Y’ if fire was shoot.hat
- OCCURRED_ON_DATE-(Object) contains full data with hour
- YEAR- (Int64) the year that the incident happened.
- MONTH-(Int64) the month that the incident happened.
- DAY_OF_WEEK-(Object) the day in the week that the incident happened.
- HOUR- (Int64) the hour that the incident happened (without the minutes).
- UCR_PART-(Object) in which part the incident appears in the UCR.
- STREET-(Object) the street that incident was happened.
- Lat-(Float64) the latitude coordinate where the incident happened.
- Long-(Float64) the longitude coordinate where the incident happened.
- Location-(Object) full GPS coordinate where the incident happened.

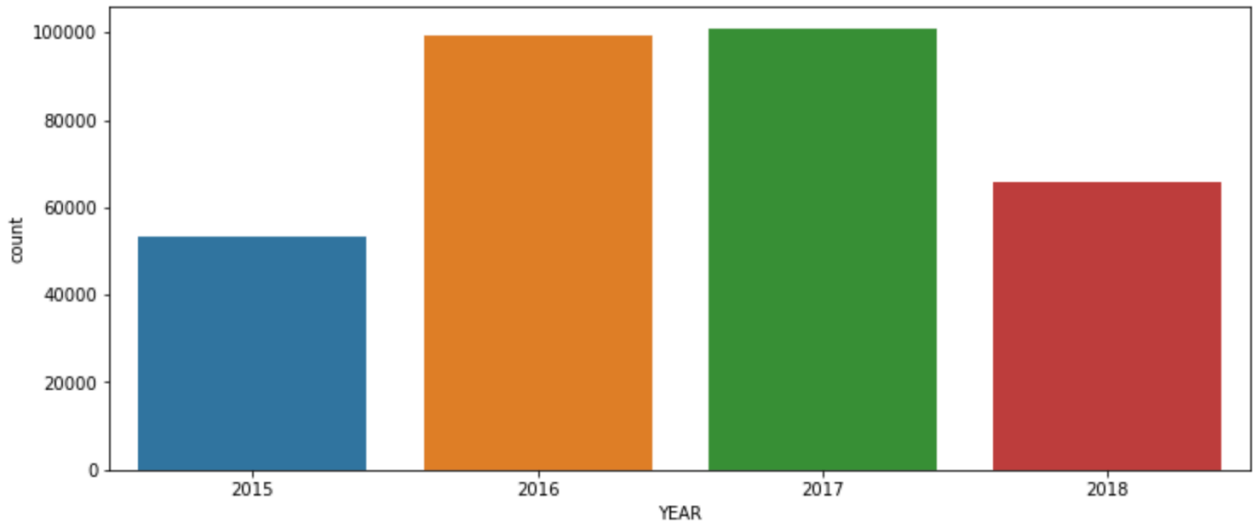
Those are the article parts:

1. Visualisations.
2 Understanding the data
3. Algorithms:
   3.1 Auto Regressive.
   3.2 Moving Average.
   3.3. ARIMA.
   3.4. SARIMA.
4. Conclusion.

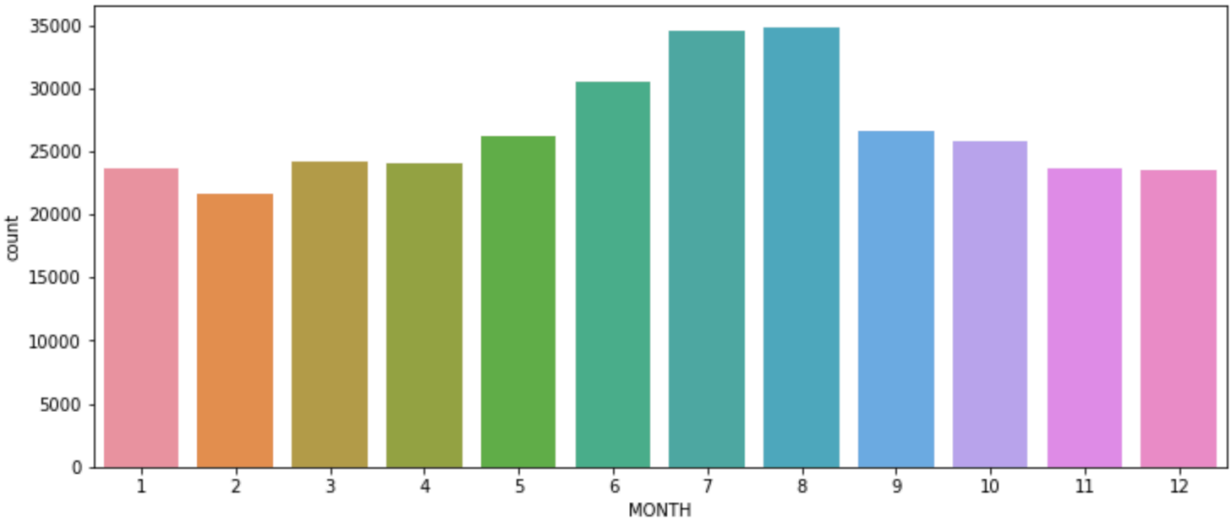
With those algorithms we will try to predict the number of crime incidents in Boston.

1. Visualisations

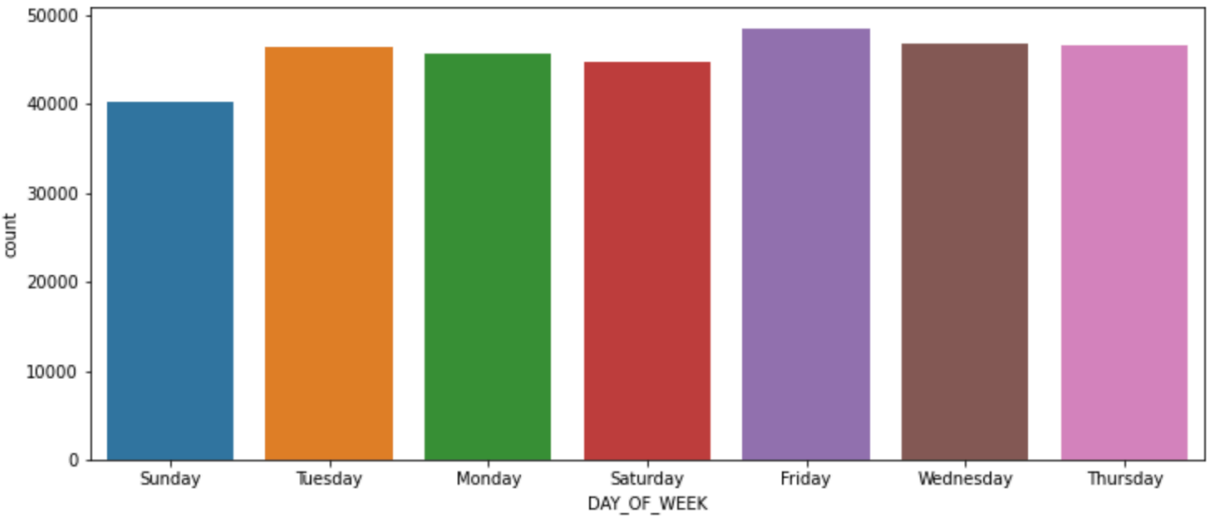
Crimes by year:



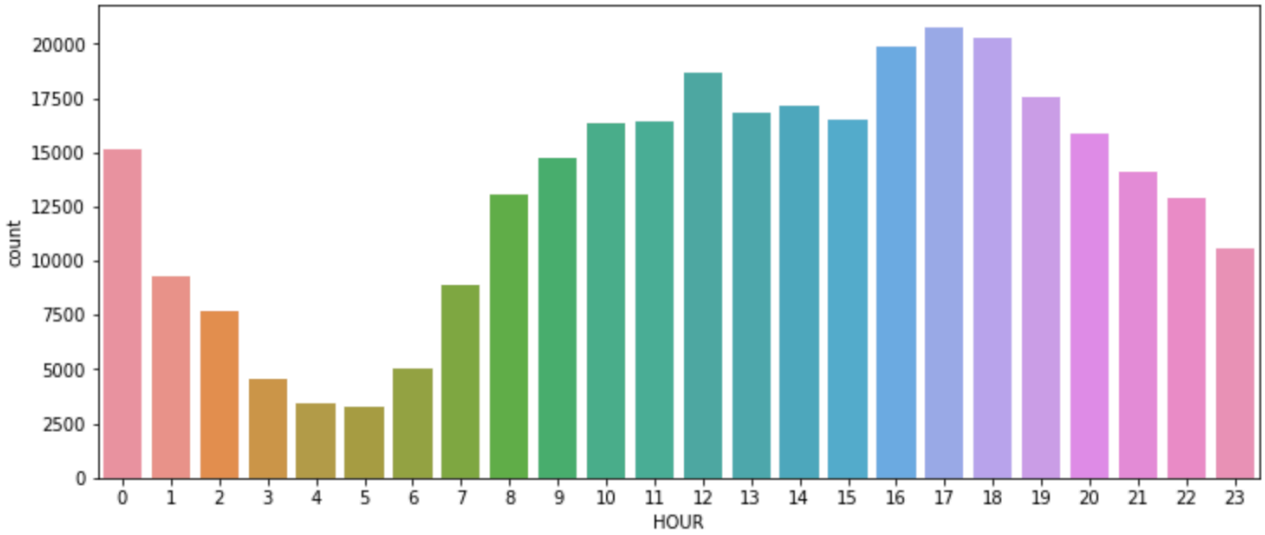
Crimes by month:



Crimes by day:

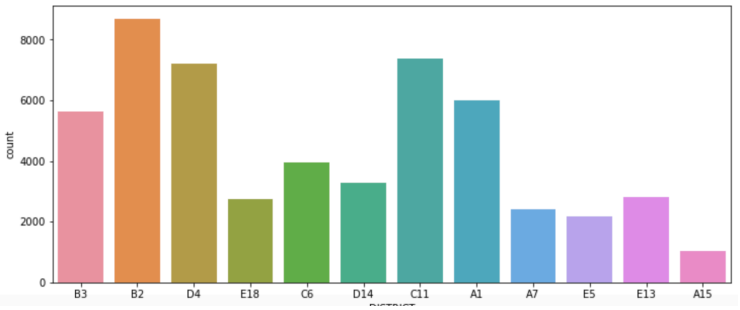


Crimes by hour:

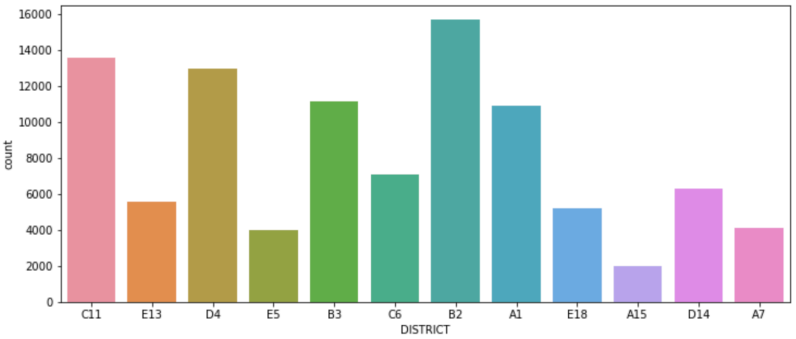


Crimes by district for each year:

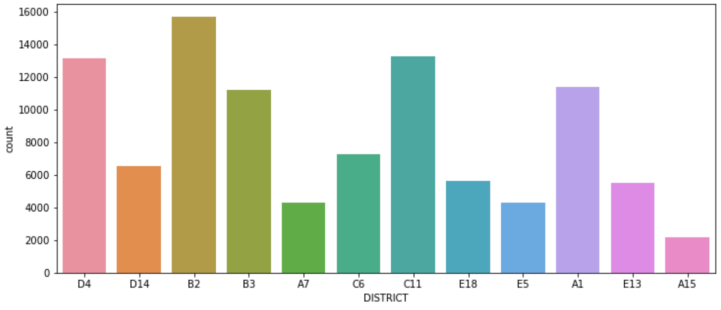
2015



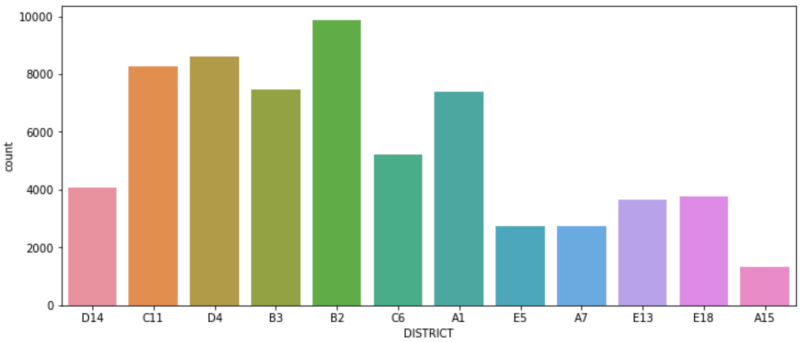
2016



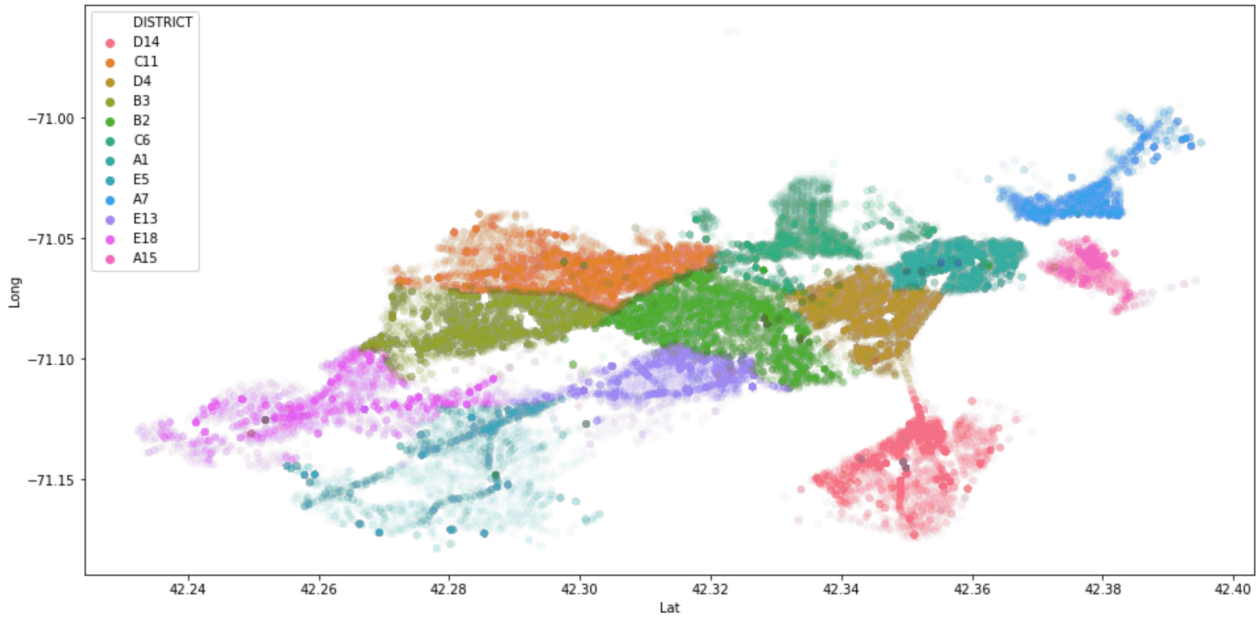
2017



2018



Crimes by districts on map:

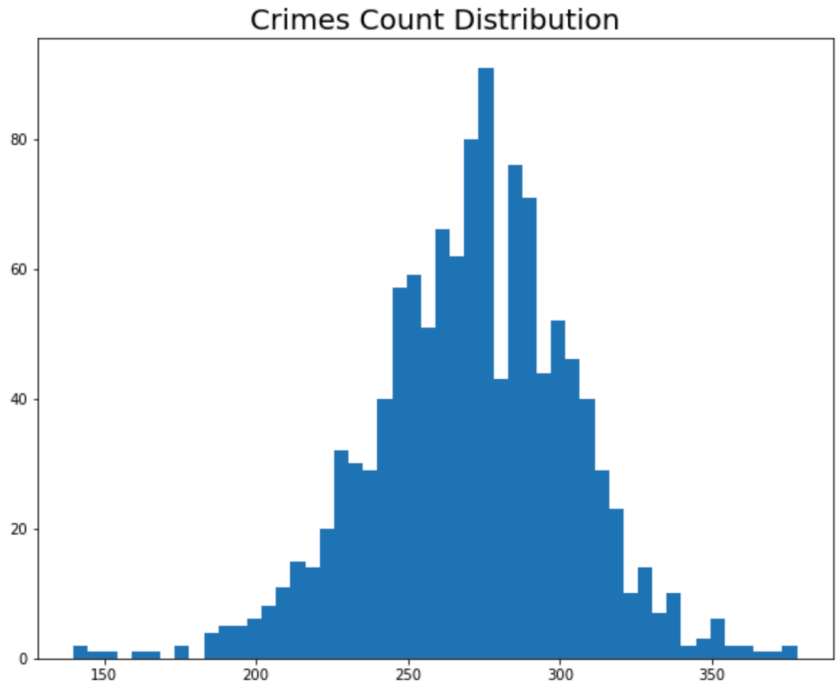


2 Understanding the data.

When we are trying to predict the number of the crimes incidents we first should create new DataFrame that include the total number of crimes for each day.
DataFrame for total number of crimes per day example:

Count	Date	
378	2016-09-01	**2016-09-01**
376	2017-09-01	**2017-09-01**
372	2018-06-15	**2018-06-15**
368	2017-09-22	**2017-09-22**
361	2017-08-04	**2017-08-04**

The distribution of the values in the data frame:

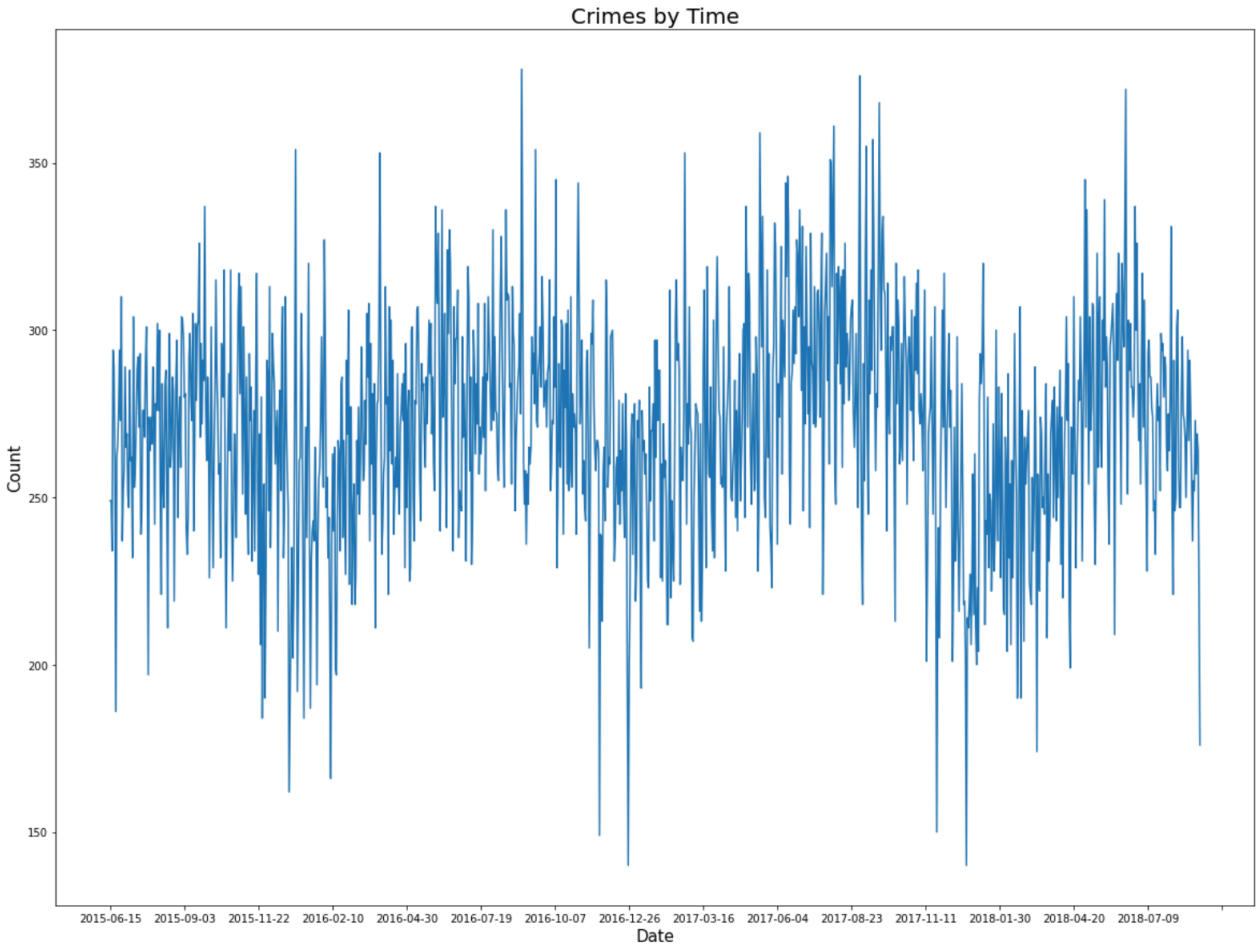


When we are performing methods to check the distribution we get:

- skewness is -0.23642018110351762 - our distribution is left skewed but not that much.
- kurtosis is 0.6916671057308545 - the kurtosis is greater than 0 means that the peak of the distribution is sharper than a stand normal distribution.

This distribution seems to be like normal distribution.

Now we will show the distribution of crimes for the whole data.



This graph looks like a “sin” function, which is a nice sign for us since it tells us that we have a good data to work with(maybe it has seasonality but we will talk about it later)

We split the data to train and test by specific date.

```
timeseries = crimes_count['Count']
train_start = datetime(2018,5,1)
train_end = datetime(2018,8,9)
train_data = timeseries[:train_end]
test_data = timeseries[train_end + timedelta(days=1):]
```

We will start by using 2 models:

1. Auto Regressive.
2. Moving Average.

Auto Regressive(AR): We try to predict C_t based on past values. It's a regression method.

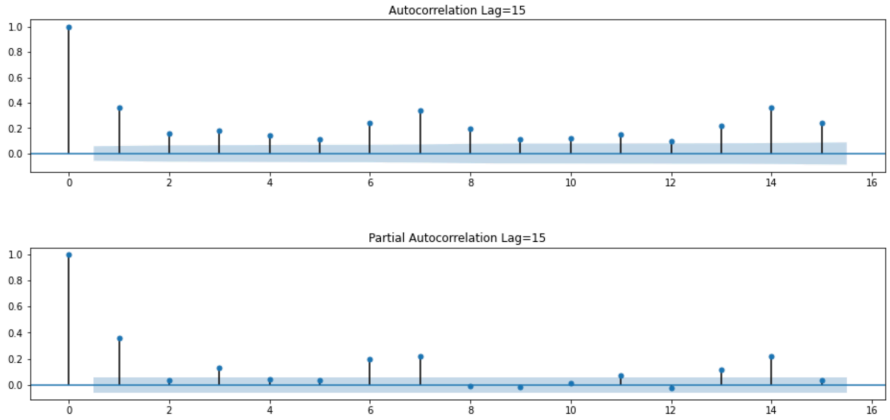
Moving Average(MA): We try to predict C_t by learning from errors of previous time.

ACF and PACF:

In order to see patterns inside our data we will use ACF and PACF.

1. ACF- The direct and indirect correlation between C_{t-k} and C_t.
2. PACF- Only the direct correlation between C_{t-k} and C_t.

ACF and PACF results:



3.1 Auto Regressive

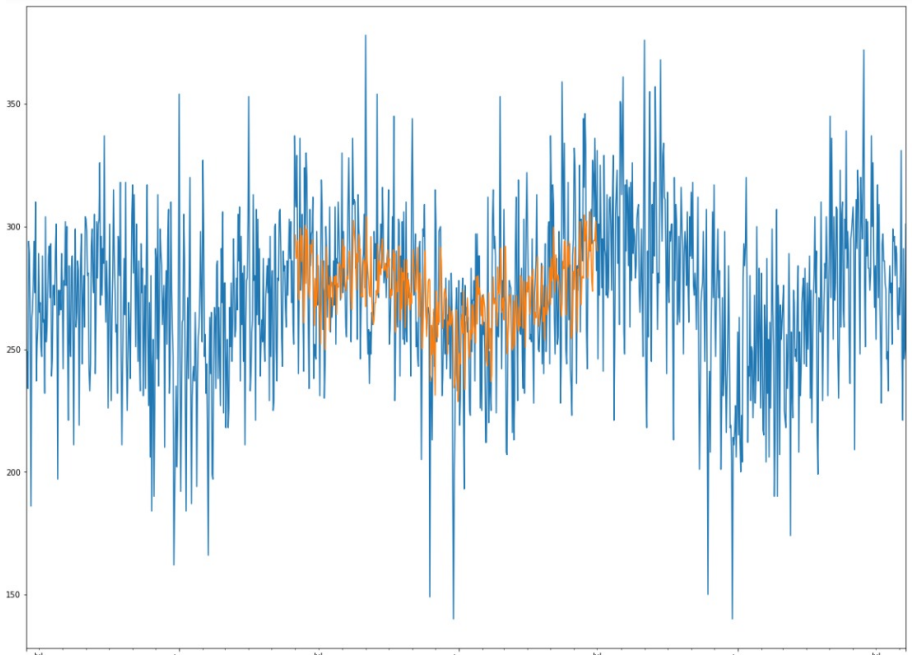
Looking at PACF plot, we can can see that lags 1, 6 ,7 are are more correlated with the current time. We will go with AR of order 7.

```
AR_model = ARMA(train_data,(7,0)).fit()
summary = (arma_mod.summary2(alpha=.05, float_format="%.8f"))
print(summary)
```

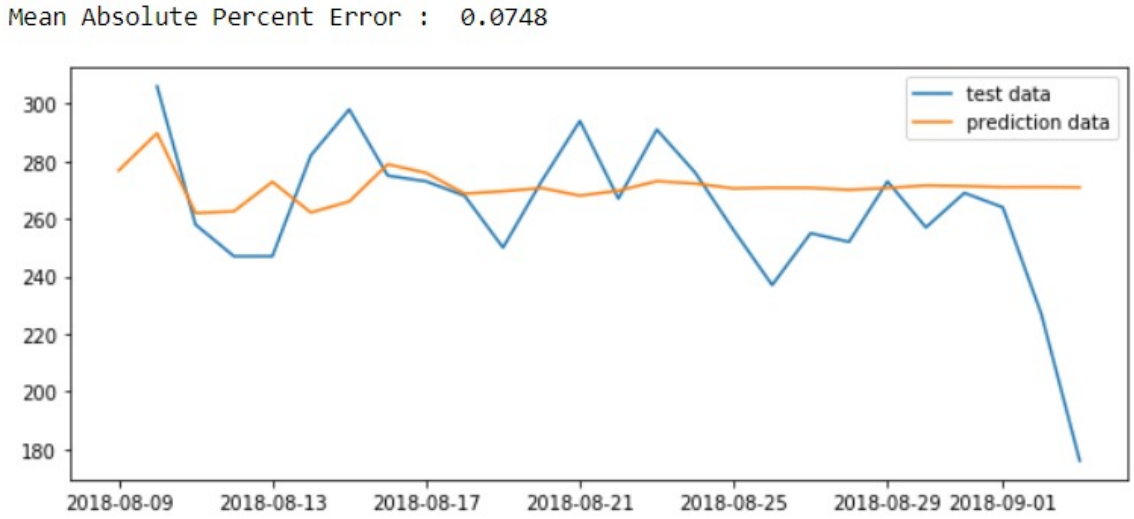
```
                         Results: ARMA
=================================================================
Model:              ARMA             BIC:                11127.5125
Dependent Variable: Count            Log-Likelihood:     -5542.6
Date:               2020-08-11 17:41 Scale:              1.0000
No. Observations:   1152             Method:             css-mle
Df Model:           5                Sample:             06-15-2015
Df Residuals:       1147                                 08-09-2018
Converged:          1.0000           S.D. of innovations: 29.726
No. Iterations:     24.0000          HQIC:               11108.652
AIC:                11097.2169
-----------------------------------------------------------------
              Coef.    Std.Err.     t      P>|t|    [0.025    0.975]
-----------------------------------------------------------------
const        270.9899    5.4823   49.4299  0.0000  260.2448  281.7351
ar.L1.Count    1.2012    0.0318   37.7774  0.0000    1.1389    1.2636
ar.L2.Count   -0.3020    0.0452   -6.6868  0.0000   -0.3905   -0.2134
ar.L3.Count    0.0907    0.0303    2.9948  0.0028    0.0313    0.1500
ma.L1.Count   -0.9332    0.0130  -71.6853  0.0000   -0.9587   -0.9077
-----------------------------------------------------------------------------
                 Real           Imaginary          Modulus          Frequency
-----------------------------------------------------------------------------
AR.1            1.0116            -0.0000           1.0116            -0.0000
AR.2            1.1596            -3.0919           3.3022            -0.1929
AR.3            1.1596             3.0919           3.3022             0.1929
MA.1            1.0716             0.0000           1.0716             0.0000
=============================================================================
```

We will talk about the results of the summary in the future (but we chose wisely.)

Prediction of AR:



Prediction of the test:



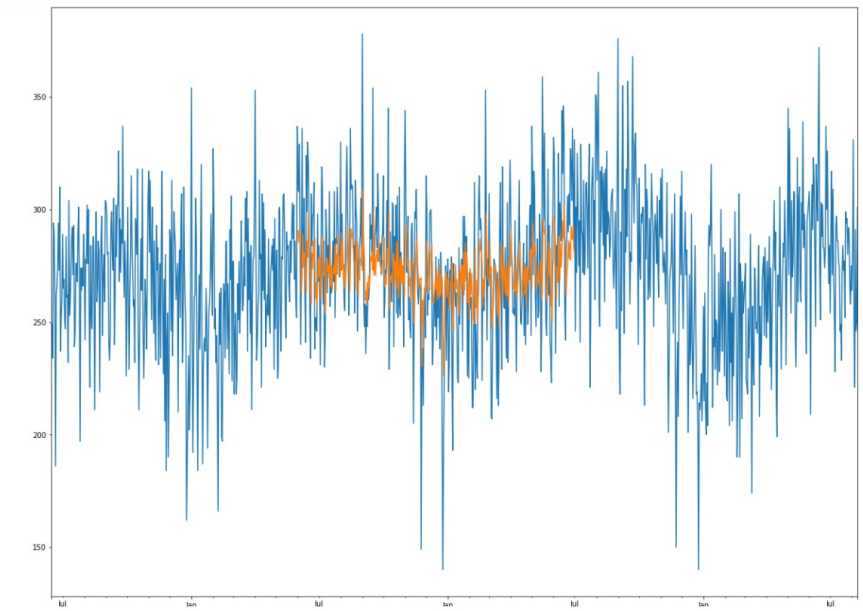
3.2. Moving Average

Looking at ACF plot, We can see that's lags 1,2,3 are high, so we will go with MA of order 3.

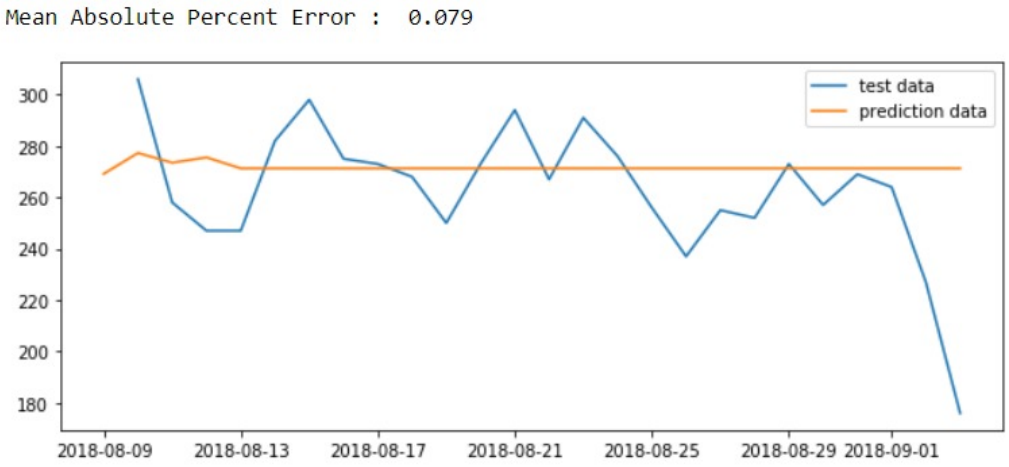
Prediction of MA:

```
MA_model = ARMA(train_data,(0,3)).fit()
summary = (MA_model.summary2(alpha=.05, float_format="%.8f"))
print(summary)
```

```
                         Results: ARMA
=================================================================
Model:              ARMA             BIC:                11207.4234
Dependent Variable: Count            Log-Likelihood:     -5586.1
Date:               2020-08-11 16:46 Scale:              1.0000
No. Observations:   1152             Method:             css-mle
Df Model:           4                Sample:             06-15-2015
Df Residuals:       1148                                 08-09-2018
Converged:          1.0000           S.D. of innovations: 30.877
No. Iterations:     14.0000          HQIC:               11191.706
AIC:                11182.1771
-----------------------------------------------------------------
               Coef.    Std.Err.     t      P>|t|    [0.025    0.975]
-----------------------------------------------------------------
const        271.2635    1.4373  188.7299  0.0000  268.4464  274.0806
ma.L1.Count    0.3322    0.0290   11.4560  0.0000    0.2753    0.3890
ma.L2.Count    0.1142    0.0299    3.8257  0.0001    0.0557    0.1727
ma.L3.Count    0.1344    0.0319    4.2148  0.0000    0.0719    0.1968
-----------------------------------------------------------------------------
                 Real           Imaginary          Modulus          Frequency
-----------------------------------------------------------------------------
MA.1           -1.7907            -0.0000           1.7907            -0.5000
MA.2            0.4703            -1.9838           2.0388            -0.2130
MA.3            0.4703             1.9838           2.0388             0.2130
=================================================================
```



Prediction of the test:



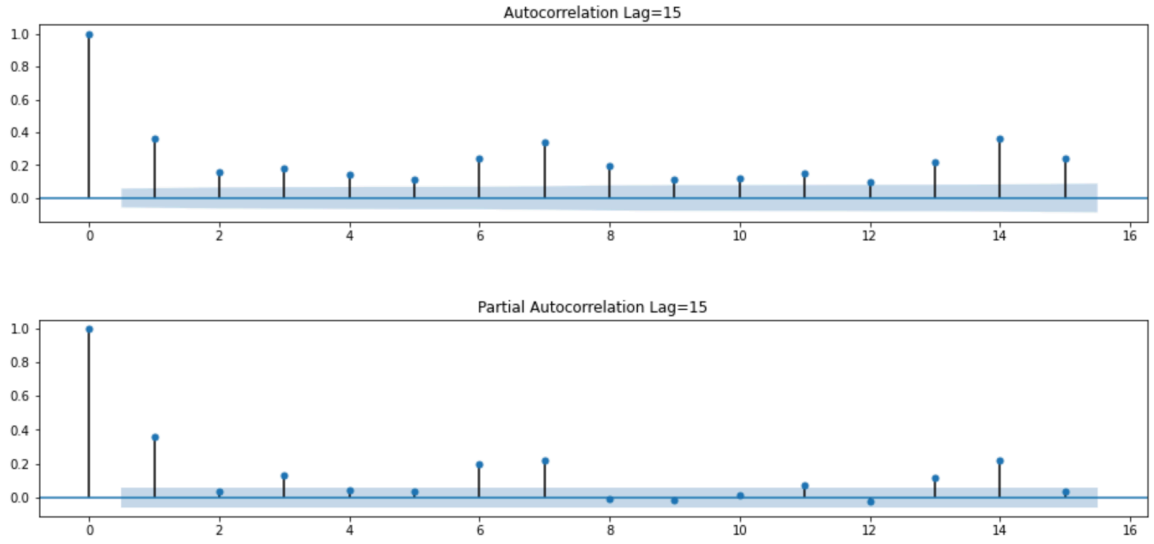
As we can see, those 2 models aren't good enough for us.
We will try to use the ARIMA model to get better predictions.

3.3 ARIMA- Auto Regressive Integrated Moving Average

The ARIMA model combines Auto Regressive with Moving average algorithms.
We will talk about Integrated later on when we will talk about differencing.

To use the ARIMA algorithm we need to choose (p, d, q) parameters.
"p" stands for the order of AR.
"q" stands for the order of MA.
How to choose d will be cleared when we will get into differencing.

Reminder of the ACF and PACF plots:



From those four diagrams we can conclude that when the lag is shorter, we can see in more details the correlation between days. As we can see the correlation is significant when lag 1 , lag 6 and lag 7, in other words the crimes correlated with yesterday and the same day in last week.

Another important thing is that the ARIMA model assume that the data is stationary.
In order to be stationary, data should satisfy 3 main attributes:

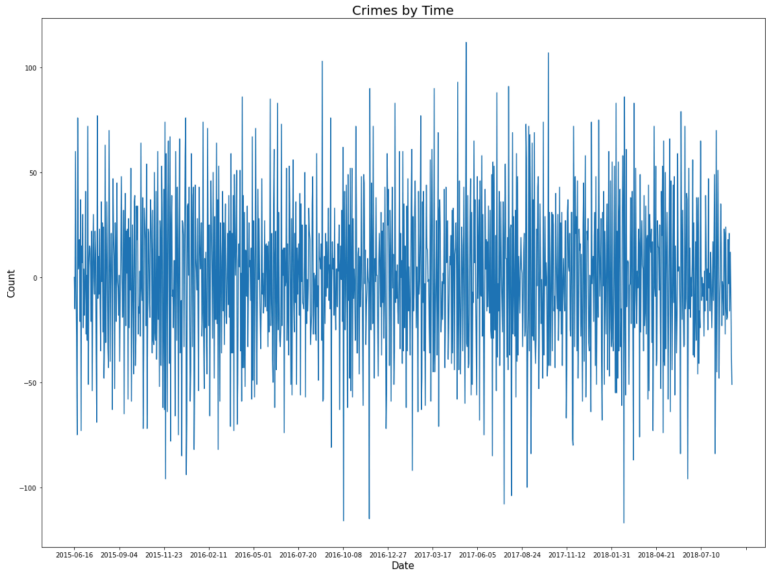
1. The mean needs to be constant.
2. The standard deviation needs to be constant.
3. There is no seasonality.

Looking in our data, it is hard to it is difficult to decide whether he is stationary or not.
So, we can run the ADF(Augmented Dickey–Fuller) test to do so.
After running the test these are the results:

```
ADF Test Result
Test Statistic                  -2.237273
p value                          0.192997
used lag                        34.000000
Number of observations used    1142.000000
Critical Value 1%               -3.436089
Critical Value 5%               -2.864074
Critical Value 10%              -2.568119
dtype: float64
```

The P value is greater than 0.05 - which indicates that our data isn't stationary.
So, we need to make our data stationary.
One popular way is to take the differential between two consecutive days.
This process is called "differencing".

After the 1-differencing process we get this result:

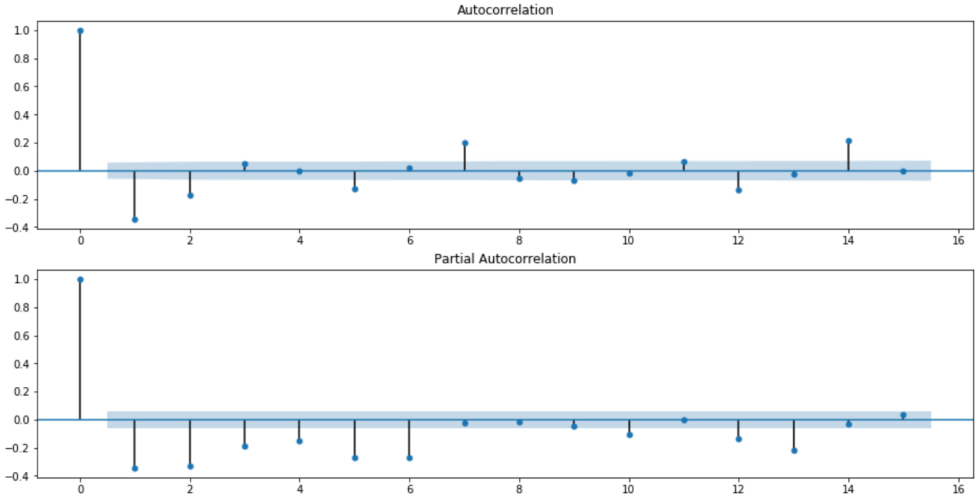


Now just by looking at the data it looks more stationary!
But we will run the ADF test again just to confirm it.

```
Average= -0.06207482993197279
Std= 37.349601593334036
SE= 1.0891364999989281
ADF Test Result
Test Statistic                -9.988185e+00
p value                        2.029010e-17
used lag                       3.300000e+01
Number of observations used    1.142000e+03
Critical Value 1%             -3.436089e+00
Critical Value 5%             -2.864074e+00
Critical Value 10%            -2.568119e+00
dtype: float64
None
```

The P value is now less than 0.05- which indicates us that our data is stationary and we are ready to continue with the ARIMA algorithm.

Lets calculate the ACF and PACF again:



From ACF we can see that lags 1 and 2 are high which indicate us we need to choose MA(2).
In addition, we can see seasonality correlation every 7 days, which is not great for us because we said earlier that our data needs to be stationary, but we will handle with it later on.

From PACF it is hard to choose ,but we will go with 3 because lag 4 is getting smaller.
So, we will go with AR(3)

Now we are ready to build our ARIMA model with parameters (p,d,q) = (3,1,2)
3- order of AR
1- number of differencing we made.
2- order of MA

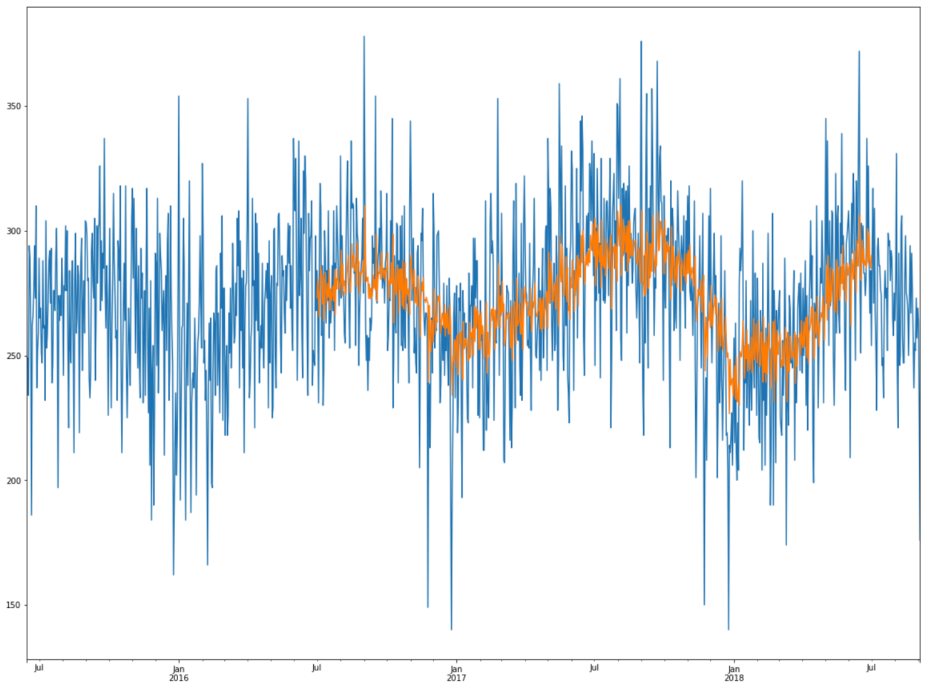
Building and fitting the model with our train:

```
arma_mod = ARMA(train_data,(3,1,2)).fit()
summary = (arma_mod.summary2(alpha=.05, float_format="%.8f"))
print(summary)
```

The results summery of the ARIMA algorithm:

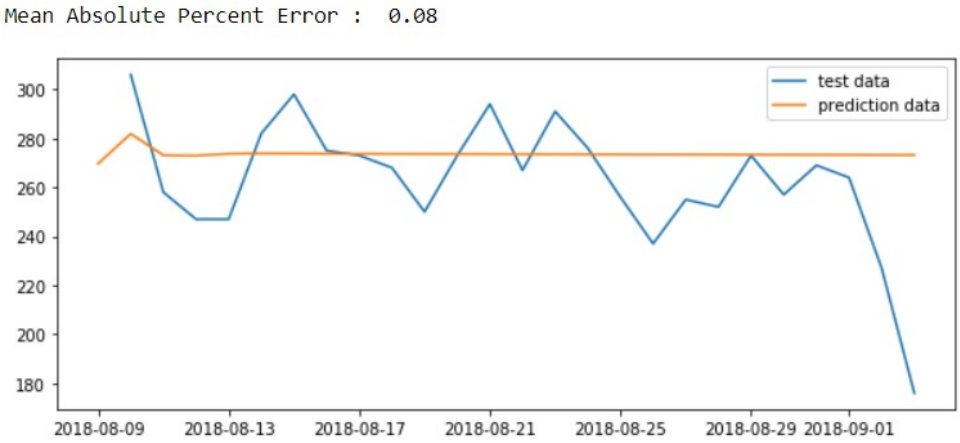
```
Results: ARMA
=================================================================
Model:              ARMA             BIC:                 11117.7713
   Dependent Variable: Count           Log-Likelihood:      -5537.7
    Date:               2020-08-09 17:22 Scale:               1.0000
   No. Observations:   1151             Method:              css-mle
Df Model:           5                Sample:              06-15-2015
Df Residuals:       1146                                  08-08-2018
    Converged:          1.0000           S.D. of innovations: 29.724
 No. Iterations:     28.0000          HQIC:                 11098.915
                                      AIC:                 11087.4810
-----------------------------------------------------------------------
 Coef.    Std.Err.     t       P>|t|    [0.025    0.975]
-----------------------------------------------------------------------
const        270.8313    5.4649   49.5584  0.0000  260.1203  281.5423
ar.L1.Count    1.2015    0.0318   37.7602  0.0000    1.1391    1.2639
ar.L2.Count   -0.3017    0.0452   -6.6820  0.0000   -0.3902   -0.2132
ar.L3.Count    0.0900    0.0303    2.9737  0.0029    0.0307    0.1494
ma.L1.Count   -0.9329    0.0131  -71.2285  0.0000   -0.9586   -0.9072
-------------------------------------------------------------------------------
Real           Imaginary          Modulus          Frequency
```

Prediction of 1 year Visualisations:



Its looks like we are in the right way but there is always lower estimated value. This maybe happened because there is seasonal correlation as we said before and we will deal with it later on.

Prediction with ARIMA:



When we try to predict the next 27 days, we get that the mean absolute percent error is 0.08.

Let's try to do better!

We will try another similar model called “SARIMA”.
Basically, it just like ARIMA model, with the addition of ’S’-seasonality.

3.4 SARIMA
The SARIMA algorithm is very similar to ARIMA with the addition of (P,D,Q,M)- four “new” parameters for the seasonal part.
In this situation, because we have many parameters to choose, we will try using brute force(in range [0,2]) and select the combination that minimise the AIC value.(another popular method we saw that many people doing so.)

We Choose 7 as the seasonal routine because of what we said earlier that we saw significant correlation every 7 days.

Those are the top five results:

	order	seasonal_order	AIC
11001.434324	(7 ,1 ,1 ,1)	(1 ,1 ,1)	63
11002.242899	(7 ,1 ,1 ,0)	(1 ,1 ,1)	59
11026.530669	(7 ,1 ,1 ,1)	(1 ,0 ,1)	47
11031.466461	(7 ,1 ,1 ,1)	(1 ,1 ,0)	31
11031.521687	(7 ,1 ,1 ,0)	(1 ,0 ,1)	43

hence we will Choose (1,1,1),(1,1,1,7)

Now we will run the SARIMA algorithm:

```
model=SARIMAX(train_data, order=(1,1,1), seasonal_order=(1,1,1,7)).fit()
summary = model.summary()

print(summary)
```

SARIMAX Results

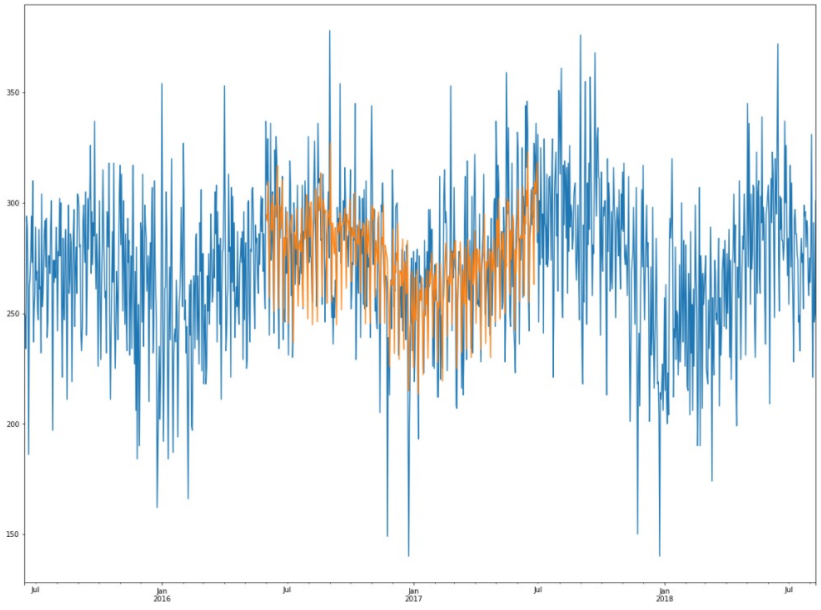
Dep. Variable:	Count	No. Observations:	1151
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 7)	Log Likelihood	-5377.400
Date:	Sun, 09 Aug 2020	AIC	10764.800
Time:	17:48:27	BIC	10790.007
Sample:	06-15-2015	HQIC	10774.318
	08-08-2018		
	Covariance Type:		opg

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.1898	0.031	6.111	0.000	0.129	0.251
ma.L1	-0.9304	0.013	-73.225	0.000	-0.955	-0.905
ar.S.L7	-0.0517	0.028	-1.838	0.066	-0.107	0.003
ma.S.L7	-0.9974	0.054	-18.601	0.000	-1.102	-0.892
sigma2	691.2939	41.910	16.495	0.000	609.151	773.436

Ljung-Box (Q):	51.60	Jarque-Bera (JB):	80.03
Prob(Q):	0.10	Prob(JB):	0.00
Heteroskedasticity (H):	1.06	Skew:	-0.20
Prob(H) (two-sided):	0.58	Kurtosis:	4.23

In the summery we can conclude that we chose the parameters wisely since their “P> |Z|” values are very small and near 0.

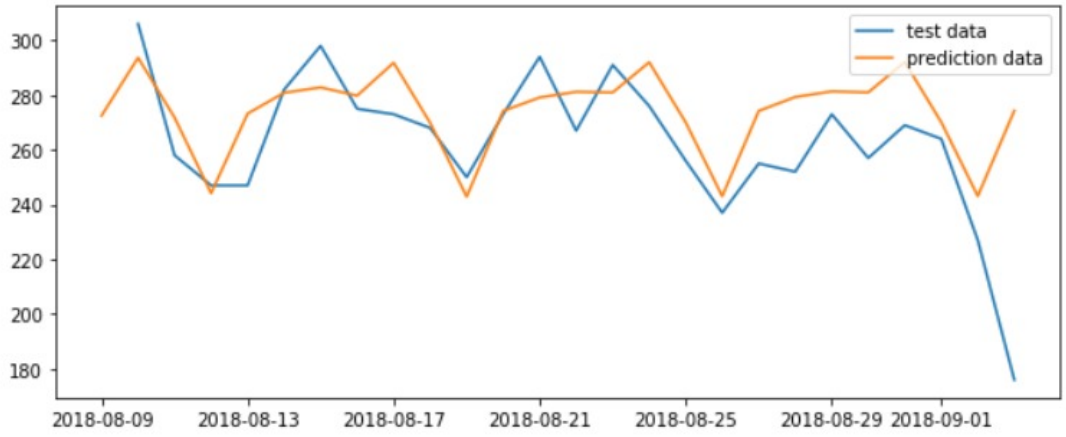
Lets see how this prediction was on the current data.



Clearly, It is looking better than the ARIMA.
The last thing we going to do is prediction for the next 27 days.

The mean absolute percent error is now 0.0686.
We indeed got better results than the ARIMA model.

```
Mean Absolute Percent Error :  0.0686
```



4. Conclusions and Impressions:

First we need to deal with a lot of statistics that we didn't have the knowledge for this, so we read a lot of new content, and watch YouTube videos, after reading and understanding this we had a better looking on those algorithm and what is needed to know.
The implementation of the algorithms was the easy part because there are a lot of tutorials that shows time series forecasting on other data sets.

As we said before the SARIMA gives us better results because the data include seasonality, therefore it is more suitable algorithm to this situation.