

capstone_project_general_crime_modeling

July 29, 2021



Modeling and Forecasting Crime Rate in Colorado

Data Science Capstone Project, Part III; (modeling general crime rate) * Student name: Elena Kazakova * Student pace: Full-time * Cohort: DS02222021 * Scheduled project review date: 07/26/2021 * Instructor name: James Irving * Application url: TBD

TABLE OF CONTENTS

- **Section 1**
- **Section 2**
- **Section 3**

1 INTRODUCTION

This is part III of the Capstone Project, the previous parts can be found in the following notebooks:
1. [Part I](#), creation of SQLite database with the original data and preprocessing of the data in the tables of the databases and building DataFrames
2. [Part II](#), preprocessing of the data in DataFrames and EDA

2 Imports

If you are running this notebook without restarting the kernel replace ‘%load_ext autoreload’ in imports with ‘%reload_ext autoreload’

```
[1]: # Importing packages
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
import statsmodels
import statsmodels.tsa.api as tsa
import plotly.express as px
import plotly.io as pio
import plotly
import math
from math import sqrt
import holidays
import pmdarima as pm

from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error

import pickle
#import shutil
import os
import json

# from pathlib import Path
# import subprocess
# import io

import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)

from functions_all import *

%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

3 MODEL&iNTERPRET

3.1 Setting up data for modeling

3.1.1 Setting up the main DataFrame

```
[2]: with open('data/pickled_dataframes/df_full_clean.pickle', 'rb') as f:
      df_full=pickle.load(f)
```

```
[3]: # Setting up timestamp index
df_full_ts_full=df_full.copy()
df_full_ts=df_full_ts_full.loc[df_full_ts_full.timestamp >'2015']
df_full_ts.set_index('timestamp', drop=True, inplace=True)
df_full_ts.head()
```

```
[3]:      offense_id  incident_id  location_name \
timestamp
2015-09-13    90865054      83230679  Residence/Home
2015-09-27    90865110      83229845  Residence/Home
2015-09-26    90865082      83229813   Other/Unknown
2015-09-21    90865081      83230696    Field/Woods
2015-09-26    90865077      83229806  Residence/Home
```

```
      offense_name  crime_against \
timestamp
2015-09-13              Motor Vehicle Theft    Property
2015-09-27      Burglary/Breaking & Entering    Property
2015-09-26  Theft of Motor Vehicle Parts or Accessories    Property
2015-09-21              Motor Vehicle Theft    Property
2015-09-26      Theft From Motor Vehicle    Property
```

```
      offense_category_name  agency_id  incident_hour \
timestamp
2015-09-13      Motor Vehicle Theft      1971          25
2015-09-27  Burglary/Breaking & Entering      1971          16
2015-09-26    Larceny/Theft Offenses      1971          25
2015-09-21      Motor Vehicle Theft      1971          25
2015-09-26    Larceny/Theft Offenses      1971          25
```

```
      primary_county  icpsr_zip  bias_name  weapon
timestamp
2015-09-13    Kit Carson    80807      None    NA
2015-09-27    Kit Carson    80807      None    NA
2015-09-26    Kit Carson    80807      None    NA
2015-09-21    Kit Carson    80807      None    NA
2015-09-26    Kit Carson    80807      None    NA
```

```
[4]: len(df_full_ts_full)
```

```
[4]: 3201143
```

```
[5]: len(df_full_ts)
```

```
[5]: 1588675
```

```
[6]: df=df_full_ts.groupby('offense_category_name')['offense_id'].nunique().  
      ↪sort_values(ascending=False)  
df_
```

```
[6]: offense_category_name  
Larceny/Theft Offenses          537725  
Assault Offenses                216625  
Destruction/Damage/Vandalism of Property  215875  
Drug/Narcotic Offenses          156490  
Fraud Offenses                  114644  
Burglary/Breaking & Entering      107629  
Motor Vehicle Theft              99299  
Sex Offenses                     29977  
Weapon Law Violations            27470  
Counterfeiting/Forgery           25613  
Robbery                          17782  
Stolen Property Offenses         11268  
Kidnapping/Abduction             9220  
Arson                             4657  
Pornography/Obscene Material      3256  
Prostitution Offenses            2536  
Embezzlement                    2372  
Animal Cruelty                   2137  
Extortion/Blackmail              2108  
Homicide Offenses                1105  
Bribery                           671  
Human Trafficking                178  
Gambling Offenses                 38  
Name: offense_id, dtype: int64
```

```
[7]: pd.crosstab(index = df_full_ts['offense_name'], columns =  
      ↪df_full_ts['offense_category_name'])[:10]
```

```
[7]: offense_category_name      Animal Cruelty  Arson  \  
offense_name  
Aggravated Assault           0          0  
All Other Larceny             0          0  
Animal Cruelty               2137         0  
Arson                         0       4657  
Assisting or Promoting Prostitution  0          0  
Betting/Wagering              0          0
```

Bribery	0	0
Burglary/Breaking & Entering	0	0
Counterfeiting/Forgery	0	0
Credit Card/Automated Teller Machine Fraud	0	0

offense_category_name	Assault Offenses	Bribery \
offense_name		
Aggravated Assault	50396	0
All Other Larceny	0	0
Animal Cruelty	0	0
Arson	0	0
Assisting or Promoting Prostitution	0	0
Betting/Wagering	0	0
Bribery	0	671
Burglary/Breaking & Entering	0	0
Counterfeiting/Forgery	0	0
Credit Card/Automated Teller Machine Fraud	0	0

offense_category_name	Burglary/Breaking & Entering \
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	107629
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Counterfeiting/Forgery \
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	25613
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Destruction/Damage/Vandalism of Property \
offense_name	

Aggravated Assault
 0
 All Other Larceny
 0
 Animal Cruelty
 0
 Arson
 0
 Assisting or Promoting Prostitution
 0
 Betting/Wagering
 0
 Bribery
 0
 Burglary/Breaking & Entering
 0
 Counterfeiting/Forgery
 0
 Credit Card/Automated Teller Machine Fraud
 0

offense_category_name	Drug/Narcotic Offenses \
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Embezzlement	Extortion/Blackmail \
offense_name		
Aggravated Assault	0	0
All Other Larceny	0	0
Animal Cruelty	0	0
Arson	0	0
Assisting or Promoting Prostitution	0	0
Betting/Wagering	0	0
Bribery	0	0
Burglary/Breaking & Entering	0	0
Counterfeiting/Forgery	0	0
Credit Card/Automated Teller Machine Fraud	0	0

offense_category_name	...	Human Trafficking \
offense_name	...	
Aggravated Assault	...	0
All Other Larceny	...	0
Animal Cruelty	...	0
Arson	...	0
Assisting or Promoting Prostitution	...	0
Betting/Wagering	...	0
Bribery	...	0
Burglary/Breaking & Entering	...	0
Counterfeiting/Forgery	...	0
Credit Card/Automated Teller Machine Fraud	...	0

offense_category_name	Kidnapping/Abduction \
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Larceny/Theft Offenses \
offense_name	
Aggravated Assault	0
All Other Larceny	185716
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Motor Vehicle Theft \
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0

Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Pornography/Obscene Material	\
offense_name		
Aggravated Assault		0
All Other Larceny		0
Animal Cruelty		0
Arson		0
Assisting or Promoting Prostitution		0
Betting/Wagering		0
Bribery		0
Burglary/Breaking & Entering		0
Counterfeiting/Forgery		0
Credit Card/Automated Teller Machine Fraud		0

offense_category_name	Prostitution Offenses	Robbery	\
offense_name			
Aggravated Assault	0	0	
All Other Larceny	0	0	
Animal Cruelty	0	0	
Arson	0	0	
Assisting or Promoting Prostitution	561	0	
Betting/Wagering	0	0	
Bribery	0	0	
Burglary/Breaking & Entering	0	0	
Counterfeiting/Forgery	0	0	
Credit Card/Automated Teller Machine Fraud	0	0	

offense_category_name	Sex Offenses	\
offense_name		
Aggravated Assault	0	
All Other Larceny	0	
Animal Cruelty	0	
Arson	0	
Assisting or Promoting Prostitution	0	
Betting/Wagering	0	
Bribery	0	
Burglary/Breaking & Entering	0	
Counterfeiting/Forgery	0	
Credit Card/Automated Teller Machine Fraud	0	

offense_category_name	Stolen Property Offenses	\
offense_name		
Aggravated Assault		0

All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

offense_category_name	Weapon Law Violations
offense_name	
Aggravated Assault	0
All Other Larceny	0
Animal Cruelty	0
Arson	0
Assisting or Promoting Prostitution	0
Betting/Wagering	0
Bribery	0
Burglary/Breaking & Entering	0
Counterfeiting/Forgery	0
Credit Card/Automated Teller Machine Fraud	0

[10 rows x 23 columns]

```
[8]: TS_crime_category=create_ts_dict('offense_category_name', df_full_ts)
```

```
TS_crime_against=create_ts_dict('crime_against', df_full_ts)
```

```
TS_crime_location=create_ts_dict('location_name', df_full_ts)
```

```
[9]: with open('data/pickled_ts/TS_crime_category.pickle', 'wb') as f:
      pickle.dump(TS_crime_category, f)
```

```
with open('data/pickled_ts/TS_crime_against.pickle', 'wb') as f:
    pickle.dump(TS_crime_against, f)
```

```
with open('data/pickled_ts/TS_crime_location.pickle', 'wb') as f:
    pickle.dump(TS_crime_location, f)
```

```
[10]: TS_crime_category.keys()
```

```
[10]: dict_keys(['Motor Vehicle Theft', 'Burglary/Breaking & Entering', 'Larceny/Theft
Offenses', 'Fraud Offenses', 'Counterfeiting/Forgery', 'Assault Offenses',
'Destruction/Damage/Vandalism of Property', 'Arson', 'Drug/Narcotic Offenses',
'Weapon Law Violations', 'Sex Offenses', 'Stolen Property Offenses',
'Kidnapping/Abduction', 'Robbery', 'Extortion/Blackmail', 'Pornography/Obscene
```

```
Material', 'Prostitution Offenses', 'Bribery', 'Embezzlement', 'Homicide
Offenses', 'Human Trafficking', 'Gambling Offenses', 'Animal Cruelty']])
```

```
[11]: df_crime_against=pd.concat(TS_crime_against,axis=1)
df_crime_against.loc[(df_crime_against['Not a Crime'].isna()),'Not a Crime']=0
df_crime_against=df_crime_against.astype({'Not a Crime': 'int64'})
df_crime_against.head()
```

```
[11]:
```

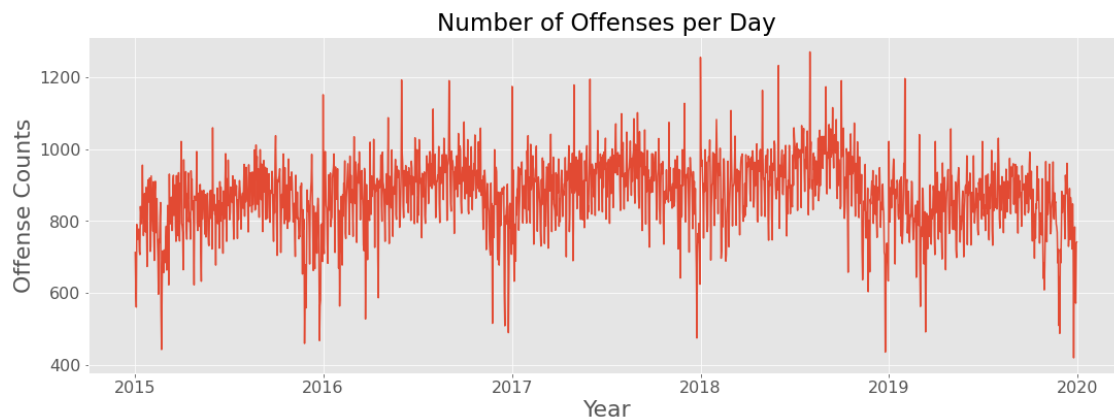
	Property	Person	Society	Not a Crime
timestamp				
2015-01-04	1369	326	190	0
2015-01-11	3954	779	548	0
2015-01-18	4288	839	711	1
2015-01-25	4040	792	761	0
2015-02-01	4331	871	690	1

3.1.2 Exploring time-series plots

```
[12]: # Creating a time-series
ts=df_full_ts.resample('D').count()['offense_id']
```

```
[13]: with plt.style.context('ggplot'):
fig, ax = plt.subplots(figsize=(18,6))

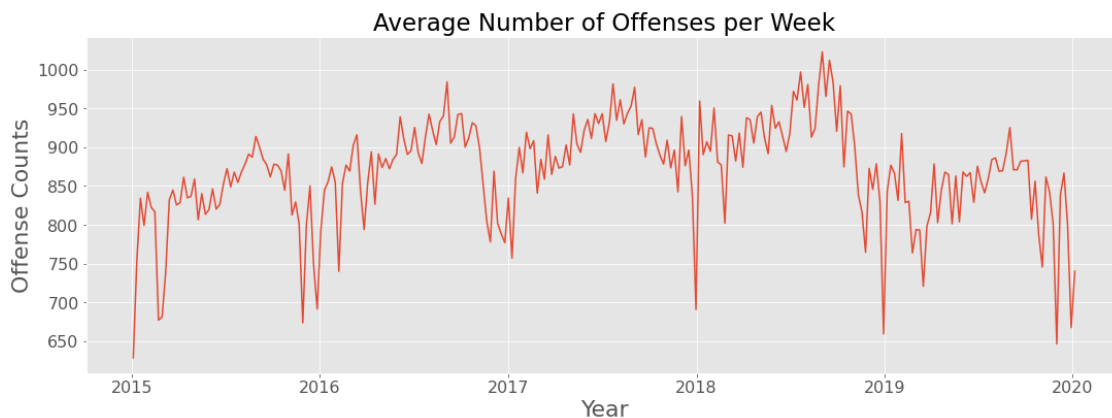
ax.plot(ts.index, ts.values)
ax.set_title('Number of Offenses per Day', fontsize=23);
ax.set_ylabel('Offense Counts', fontsize=22);
ax.set_xlabel('Year', fontsize=22);
ax.tick_params(axis='y', labels=16)
ax.tick_params(axis='x', labels=16)
plt.show()
```



```
[14]: ts_weekly=ts.resample('W').mean()
```

```
[15]: with open('data/pickled_ts/ts_weekly.pickle', 'wb') as f:  
      pickle.dump(ts_weekly, f)
```

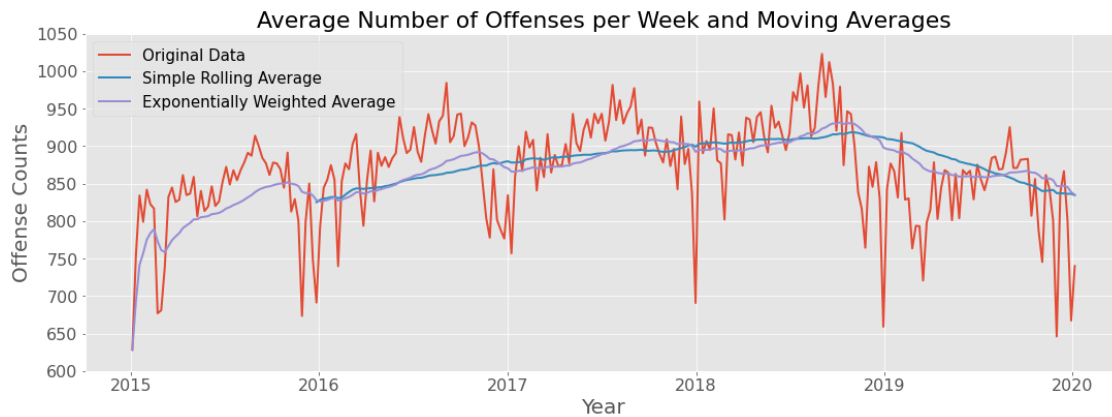
```
[16]: with plt.style.context('ggplot'):  
      fig, ax = plt.subplots(figsize=(18,6))  
  
      ax.plot(ts_weekly.index, ts_weekly.values)  
      ax.set_title('Average Number of Offenses per Week', fontsize=23);  
      ax.set_ylabel('Offense Counts', fontsize=22);  
      ax.set_xlabel('Year', fontsize=22);  
      ax.tick_params(axis='y', labelsize=16)  
      ax.tick_params(axis='x', labelsize=16)  
      plt.show()
```



```
[17]: ts_ma=ts_weekly.rolling(52).mean()
```

```
[18]: ts_ewm=ts_weekly.ewm(span=52).mean()
```

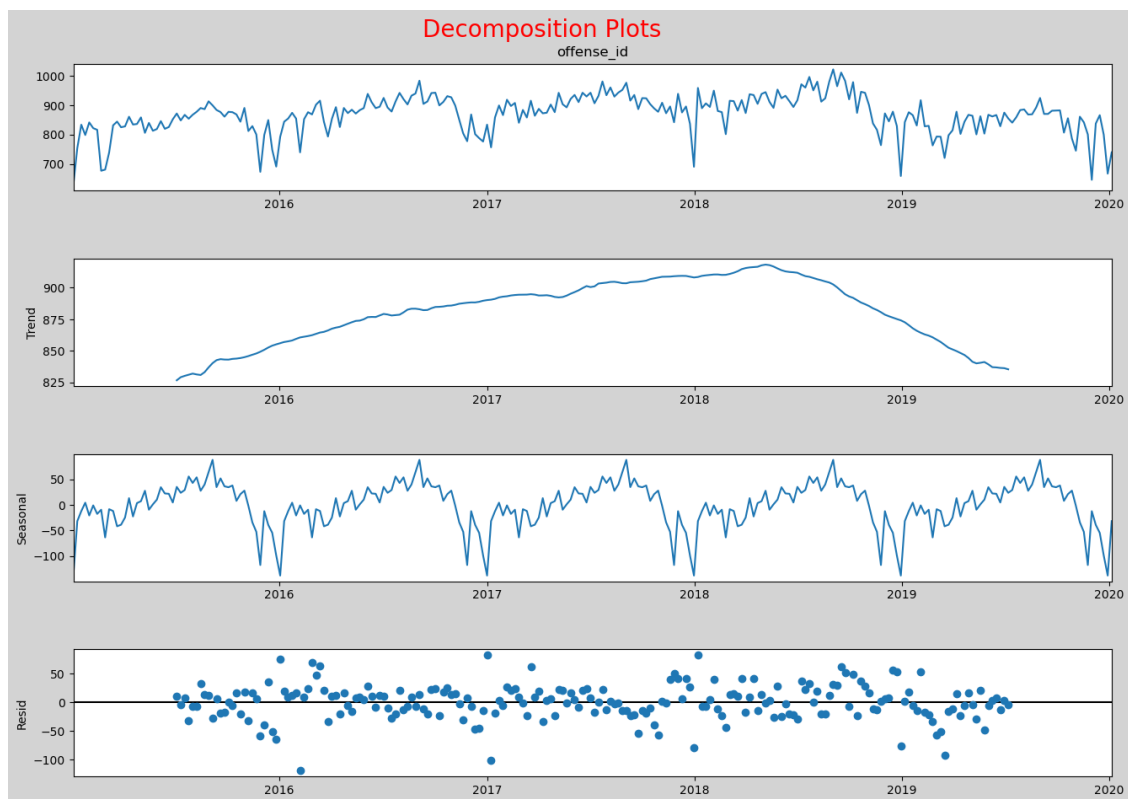
```
[19]: fig=display_figure_w_TSS(ts_weekly, ts_ma, 'Original Data', 'Simple Rolling_  
      ↳Average',  
                                'Average Number of Offenses per Week and Moving Averages',  
                                ↳n=3,  
                                ts3=ts_ewm, ts4=None, label3='Exponentially Weighted_  
      ↳Average', label4=None)
```



EWA displays a clear upward trend with a seasonality while SRA does not pick up the seasonal fluctuation tendency. The seasonality is quite pronounced and is of an additive nature. The problematic range of dates is a period from the late 2018 till the end of 2019 when the trend changes to a downward trend. Unfortunately, cutting off a test set with the latest dates will make it impossible to predict the overall trend correctly.

```
[20]: decomposing(ts_weekly);
```

<Figure size 640x480 with 0 Axes>

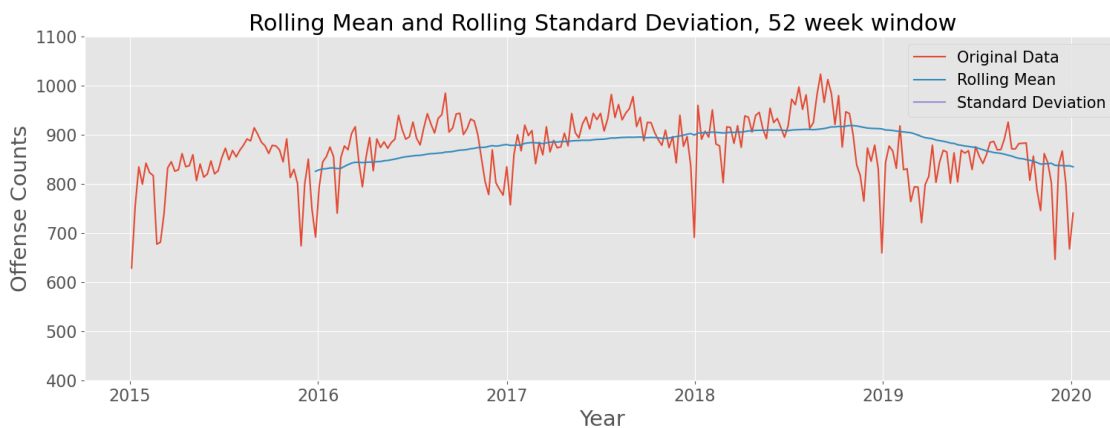


```
[21]: # with open('images/pickled_figs/decomposition_plot_ts_weekly.pickle', 'wb') as f:
# pickle.dump(decomposing(ts_weekly),f)
```

The time series displays a clear trend along with seasonal fluctuations. Seasonality is comparable with the overall trend values (>10%).

3.1.3 Testing for Stationarity

```
[22]: check_stationarity(ts_weekly, 'Original Data', min_=400, max_=1100)
```



```
[22]:
          T_value  P_value  Lags  Observations  \
Dickey-Fuller test results -3.224157  0.018628    3         258

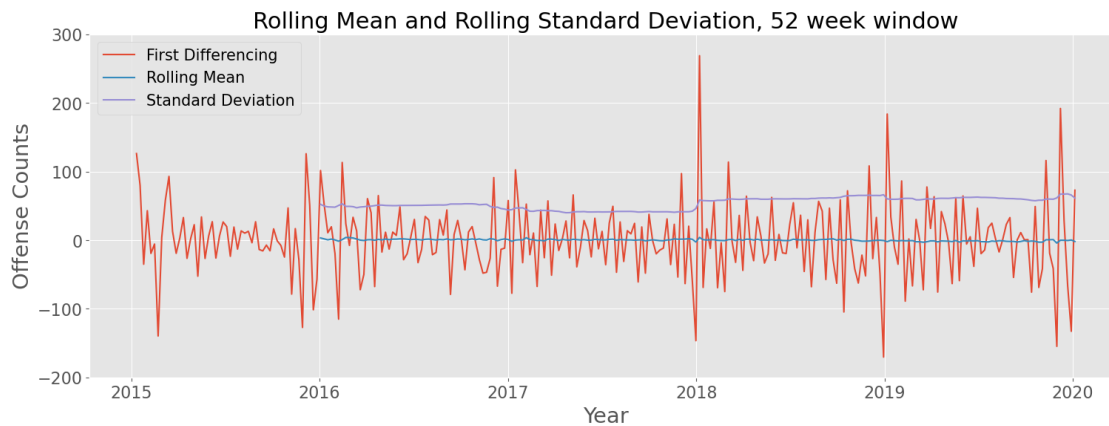
          Critical value, 1%  Critical value, 5%  \
Dickey-Fuller test results      -3.455953      -2.872809

          Critical value, 10%  Stationary?
Dickey-Fuller test results      -2.572775      True
```

The time-series is more or less **stationary**, p-value is 0.02 (<0.05). Visually it is not very stationary, the trend is somewhat visible. Since critical value -3.22 > -3.46, but < -2.87 (t-values at 1% and 5% confidence intervals), null hypothesis is rejected. However, the TS might benefit from stationarization

```
[23]: ts_diff1=ts_weekly.diff().dropna()

check_stationarity(ts_diff1, 'First Differencing', min_=-200, max_=300)
```



```
[23]:
```

	T_value	P_value	Lags	Observations	\
Dickey-Fuller test results	-4.872453	0.000039	12	248	

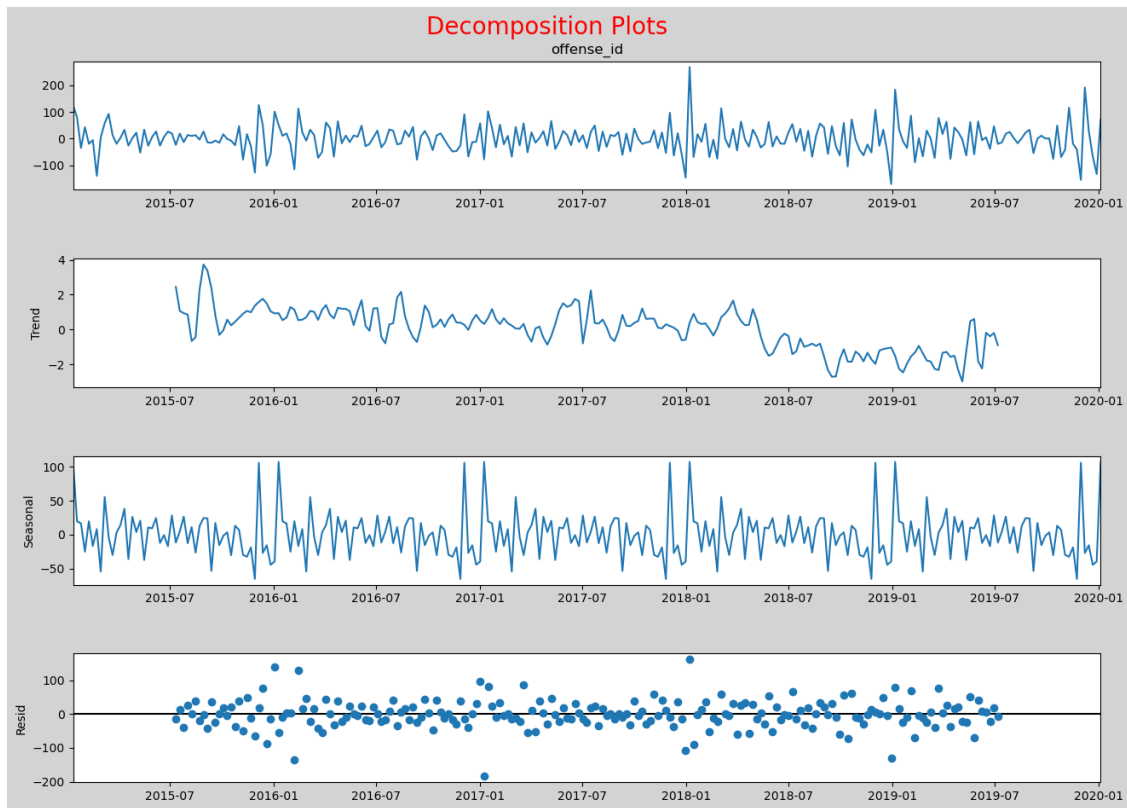
	Critical value, 1%	Critical value, 5%	\
Dickey-Fuller test results	-3.456996	-2.873266	

	Critical value, 10%	Stationary?
Dickey-Fuller test results	-2.573019	True

The first differencing time-series is **stationary**, p-value is $3.9e-5$ (well below 0.05). Also the critical value $-4.87 < -3.46, -2.87$ (t-values at 1% and 5% confidence intervals); null hypothesis is rejected.

```
[24]: decomposing(ts_diff1);
```

<Figure size 640x480 with 0 Axes>



The first differencing time-series decomposition displays clear seasonality.

3.2 General Crime Rate Modeling

3.2.1 Splitting into a training and a test sets

I am cutting off a ~10% tail of my data to create a test set because I want the downswing of the data in the last year to be included in the training dataset

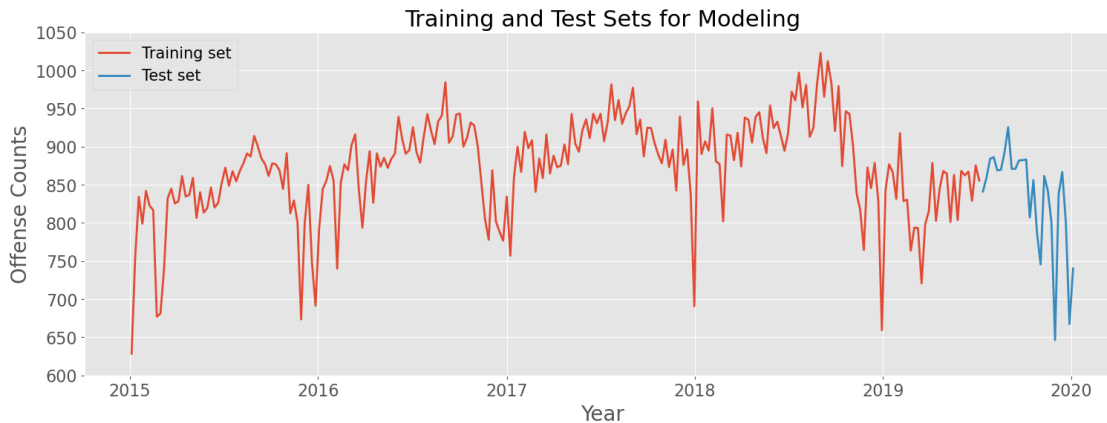
```
[25]: train_size = round(len(ts_weekly) * 0.90)
      ts_train, ts_test = ts_weekly[:train_size], ts_weekly[train_size:]
      print('Observations: %d weeks' % (len(ts_weekly)))
      print('Training Observations: %d weeks' % (len(ts_train)))
      print('Testing Observations: %d weeks' % (len(ts_test)))
```

Observations: 262 weeks

Training Observations: 236 weeks

Testing Observations: 26 weeks

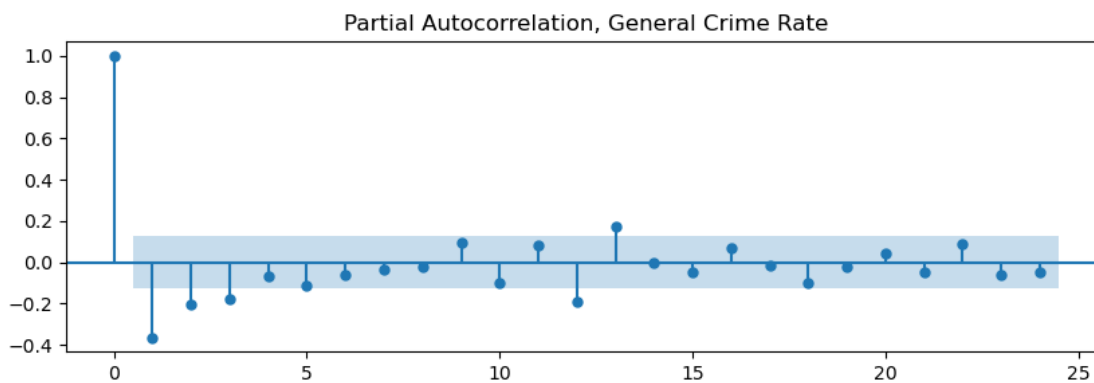
```
[26]: fig=display_figure_w_TSs(ts_train, ts_test, 'Training set', 'Test set',
      ↪ 'Training and Test Sets for Modeling')
```



```
[27]: with open('images/pickled_figs/ts_weekly_train_test.pickle', 'wb') as f:
      pickle.dump(fig,f)
```

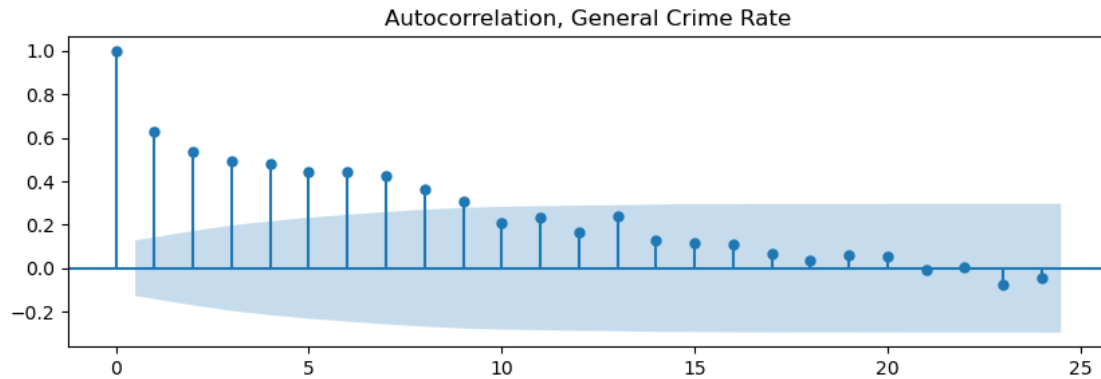
3.2.2 Partial Autocorrelation and Autocorrelation Functions

```
[28]: matplotlib.rc_file_defaults()
      plt.rc("figure", figsize=(10,3))
      plot_pacf(ts_train.diff().dropna(), title='Partial Autocorrelation, General_
      ↳Crime Rate');
```



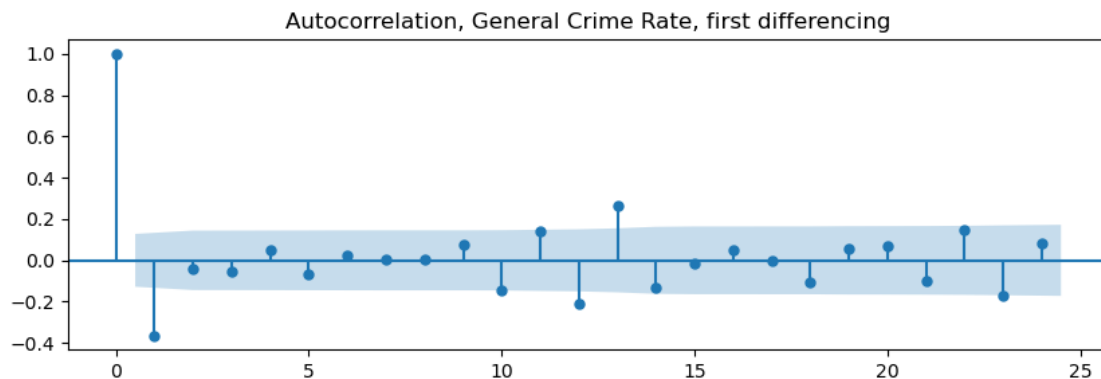
Partial autocorrelation function of the first ts differencing indicates the importance of the first 3 lags.

```
[29]: matplotlib.rc_file_defaults()
      plt.rc("figure", figsize=(10,3))
      plot_acf(ts_train, title='Autocorrelation, General Crime Rate');
```

The ACF shows a long persistent autocorrelation up to the 9th lag. That is a strong indicator that the differencing should be taken to stationarize the TS.

```
[30]: matplotlib.rc_file_defaults()
plt.rc("figure", figsize=(10,3))
plot_acf(ts_train.diff().dropna(), title='Autocorrelation, General Crime Rate,␣
↪first differencing');
```

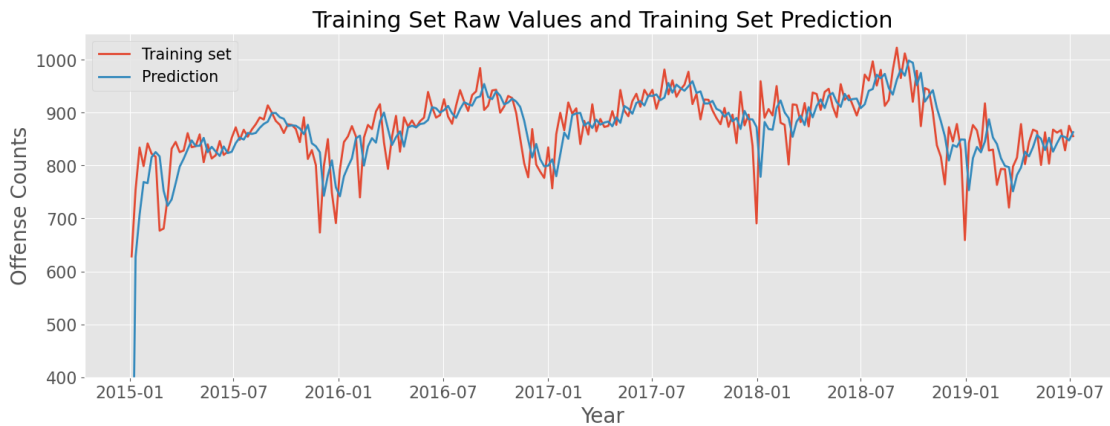


The ACF of the differenced series displays the sharp cut-off and the negative lag1 correlation and therefore one MA term could be added to the model.

3.2.3 Baseline Model

```
[31]: arima_1=ARIMA(ts_train, order=(3,1,0)).fit()
y_hat_train=arima_1.predict(typ='levels')

fig=display_figure_w_TSs(ts_train, y_hat_train, 'Training set', 'Prediction',
                          'Training Set Raw Values and Training Set Prediction',
                          min_=400)
```



```
[32]: diagnostics(arima_1)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

SARIMAX Results

```
=====
Dep. Variable:          offense_id    No. Observations:          236
Model:                ARIMA(3, 1, 0)  Log Likelihood             -1240.793
Date:                 Wed, 28 Jul 2021  AIC                        2489.586
Time:                 19:46:25         BIC                        2503.424
Sample:               01-04-2015      HQIC                       2495.165
                   - 07-07-2019
```

```
Covariance Type:          opg
```

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
ar.L1         -0.5013     0.046    -10.995     0.000     -0.591     -0.412
ar.L2         -0.3149     0.063     -4.999     0.000     -0.438     -0.191
ar.L3         -0.1955     0.059     -3.317     0.001     -0.311     -0.080
sigma2        2253.0732    148.340     15.189     0.000    1962.332    2543.815
=====
```

```
Ljung-Box (L1) (Q):                0.00    Jarque-Bera (JB):                67.77
Prob(Q):                          1.00    Prob(JB):                          0.00
Heteroskedasticity (H):            0.99    Skew:                             -0.53
```

Prob(H) (two-sided):

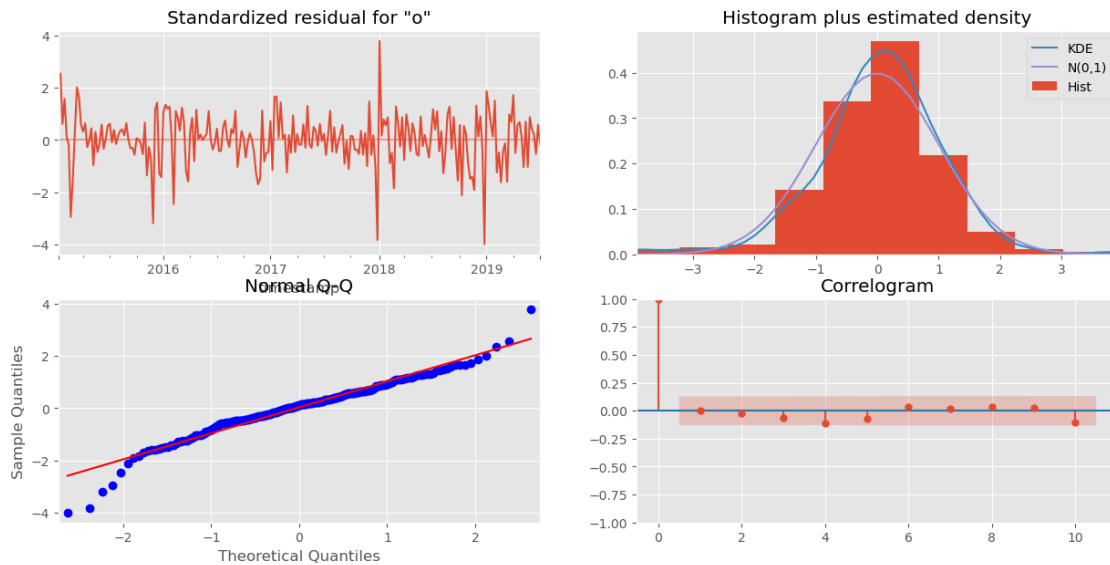
0.95

Kurtosis:

5.41

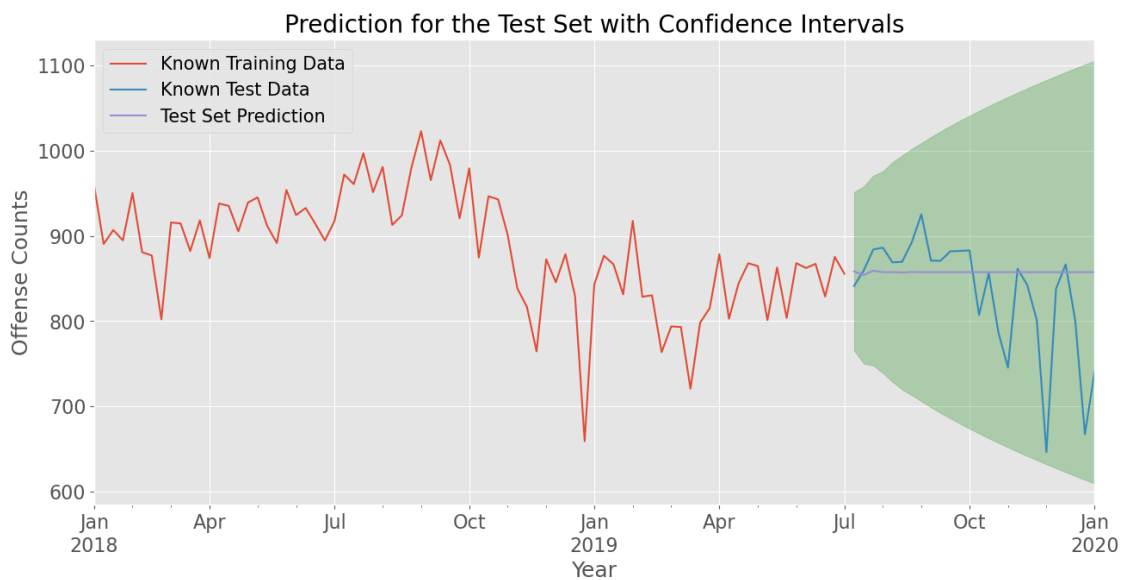
Warnings:

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



Testing the model

```
[33]: fig=predictions_testset(ts_train, ts_test, arima_1)
```



```
[34]: with open('images/pickled_figs/arima_train_test.pickle', 'wb') as f:
      pickle.dump(fig,f)
```

The ARIMA model above is our Baseline model

3.2.4 SARIMAX models

No exogenous regressors It's a manual grid search section. I tried several combinations of pdq/PDQs and it seems that the most appropriate tryout ranges for MA terms and for AR terms in both trend and seasonal parts of the models are 0-1 and 0-3 respectively.

```
[35]: p=range(0,4)
      q=range(0,2)
      pdq=list(itertools.product(p,[1],q))

      P=range(0,4)
      Q=range(0,2)
      seasonal_pdq=[(x[0], x[1], x[2], 52) for x in list(itertools.product(P,[1],Q))]

      for i in pdq:
          for s in seasonal_pdq:
              print('SARIMAX combination: {}'.format(i,s))
```

```
SARIMAX combination: (0, 1, 0)x(0, 1, 0, 52)
SARIMAX combination: (0, 1, 0)x(0, 1, 1, 52)
SARIMAX combination: (0, 1, 0)x(1, 1, 0, 52)
SARIMAX combination: (0, 1, 0)x(1, 1, 1, 52)
SARIMAX combination: (0, 1, 0)x(2, 1, 0, 52)
SARIMAX combination: (0, 1, 0)x(2, 1, 1, 52)
SARIMAX combination: (0, 1, 0)x(3, 1, 0, 52)
SARIMAX combination: (0, 1, 0)x(3, 1, 1, 52)
SARIMAX combination: (0, 1, 1)x(0, 1, 0, 52)
SARIMAX combination: (0, 1, 1)x(0, 1, 1, 52)
SARIMAX combination: (0, 1, 1)x(1, 1, 0, 52)
SARIMAX combination: (0, 1, 1)x(1, 1, 1, 52)
SARIMAX combination: (0, 1, 1)x(2, 1, 0, 52)
SARIMAX combination: (0, 1, 1)x(2, 1, 1, 52)
SARIMAX combination: (0, 1, 1)x(3, 1, 0, 52)
SARIMAX combination: (0, 1, 1)x(3, 1, 1, 52)
SARIMAX combination: (1, 1, 0)x(0, 1, 0, 52)
SARIMAX combination: (1, 1, 0)x(0, 1, 1, 52)
SARIMAX combination: (1, 1, 0)x(1, 1, 0, 52)
SARIMAX combination: (1, 1, 0)x(1, 1, 1, 52)
SARIMAX combination: (1, 1, 0)x(2, 1, 0, 52)
SARIMAX combination: (1, 1, 0)x(2, 1, 1, 52)
SARIMAX combination: (1, 1, 0)x(3, 1, 0, 52)
SARIMAX combination: (1, 1, 0)x(3, 1, 1, 52)
```

```

SARIMAX combination: (1, 1, 1)x(0, 1, 0, 52)
SARIMAX combination: (1, 1, 1)x(0, 1, 1, 52)
SARIMAX combination: (1, 1, 1)x(1, 1, 0, 52)
SARIMAX combination: (1, 1, 1)x(1, 1, 1, 52)
SARIMAX combination: (1, 1, 1)x(2, 1, 0, 52)
SARIMAX combination: (1, 1, 1)x(2, 1, 1, 52)
SARIMAX combination: (1, 1, 1)x(3, 1, 0, 52)
SARIMAX combination: (1, 1, 1)x(3, 1, 1, 52)
SARIMAX combination: (2, 1, 0)x(0, 1, 0, 52)
SARIMAX combination: (2, 1, 0)x(0, 1, 1, 52)
SARIMAX combination: (2, 1, 0)x(1, 1, 0, 52)
SARIMAX combination: (2, 1, 0)x(1, 1, 1, 52)
SARIMAX combination: (2, 1, 0)x(2, 1, 0, 52)
SARIMAX combination: (2, 1, 0)x(2, 1, 1, 52)
SARIMAX combination: (2, 1, 0)x(3, 1, 0, 52)
SARIMAX combination: (2, 1, 0)x(3, 1, 1, 52)
SARIMAX combination: (2, 1, 1)x(0, 1, 0, 52)
SARIMAX combination: (2, 1, 1)x(0, 1, 1, 52)
SARIMAX combination: (2, 1, 1)x(1, 1, 0, 52)
SARIMAX combination: (2, 1, 1)x(1, 1, 1, 52)
SARIMAX combination: (2, 1, 1)x(2, 1, 0, 52)
SARIMAX combination: (2, 1, 1)x(2, 1, 1, 52)
SARIMAX combination: (2, 1, 1)x(3, 1, 0, 52)
SARIMAX combination: (2, 1, 1)x(3, 1, 1, 52)
SARIMAX combination: (3, 1, 0)x(0, 1, 0, 52)
SARIMAX combination: (3, 1, 0)x(0, 1, 1, 52)
SARIMAX combination: (3, 1, 0)x(1, 1, 0, 52)
SARIMAX combination: (3, 1, 0)x(1, 1, 1, 52)
SARIMAX combination: (3, 1, 0)x(2, 1, 0, 52)
SARIMAX combination: (3, 1, 0)x(2, 1, 1, 52)
SARIMAX combination: (3, 1, 0)x(3, 1, 0, 52)
SARIMAX combination: (3, 1, 0)x(3, 1, 1, 52)
SARIMAX combination: (3, 1, 1)x(0, 1, 0, 52)
SARIMAX combination: (3, 1, 1)x(0, 1, 1, 52)
SARIMAX combination: (3, 1, 1)x(1, 1, 0, 52)
SARIMAX combination: (3, 1, 1)x(1, 1, 1, 52)
SARIMAX combination: (3, 1, 1)x(2, 1, 0, 52)
SARIMAX combination: (3, 1, 1)x(2, 1, 1, 52)
SARIMAX combination: (3, 1, 1)x(3, 1, 0, 52)
SARIMAX combination: (3, 1, 1)x(3, 1, 1, 52)

```

```

[36]: # for param in pdq:
#     for param_seasonal in seasonal_pdq:
#         try:
#             sarimax_mod=SARIMAX(ts_train,
#                                 order=param,
#                                 seasonal_order=param_seasonal,

```

```

#                                     enforce_invertibility=False)
#                                     results=sarimax_mod.fit()
#                                     print('ARIMA{x}-AIC:{:}'.format(param, param_seasonal, results.
→aic))
#                                     except:
#                                     print('Error!')
#                                     continue

```

ARIMA(3, 1, 0)x(3, 1, 0, 52)-AIC:256.74: is our best model. it took 55 minutes to complete this search. Therefore I am commenting out this snippet.

```

[37]: # sarimax_mod1=SARIMAX(ts_train,
#                               order=(3, 1, 0),
#                               seasonal_order=(3, 1, 0, 52),
#                               enforce_invertibility=False).fit()

```

The model above took 44 seconds to fit but just in case it is pickled to be used forward.

```

[38]: # with open('data/pickled_models/sarimax_mod1.pickle', 'wb') as f:
#         pickle.dump(sarimax_mod1, f)

```

```

[39]: with open('data/pickled_models/sarimax_mod1.pickle', 'rb') as f:
        sarimax_mod1=pickle.load(f)

```

```

[40]: diagnostics(sarimax_mod1)

```

```

<class 'statsmodels.iolib.summary.Summary'>
"""

```

```

                                SARIMAX Results
=====
Dep. Variable:                offense_id    No. Observations:                236
Model:                SARIMAX(3, 1, 0)x(3, 1, 0, 52)    Log Likelihood                -971.969
Date:                Wed, 28 Jul 2021    AIC                1957.937
Time:                19:46:27    BIC                1980.403
Sample:                01-04-2015    HQIC                1967.044
                        - 07-07-2019

Covariance Type:                opg
=====

```

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.6356	0.056	-11.337	0.000	-0.745	-0.526
ar.L2	-0.3886	0.066	-5.877	0.000	-0.518	-0.259
ar.L3	-0.1715	0.075	-2.292	0.022	-0.318	-0.025
ar.S.L52	-0.6286	0.092	-6.800	0.000	-0.810	-0.447
ar.S.L104	-0.4472	0.151	-2.960	0.003	-0.743	-0.151
ar.S.L156	-0.2381	0.165	-1.439	0.150	-0.562	0.086
sigma2	2033.5387	269.943	7.533	0.000	1504.460	2562.617

```

=====
Ljung-Box (L1) (Q):                0.03    Jarque-Bera (JB):                39.08

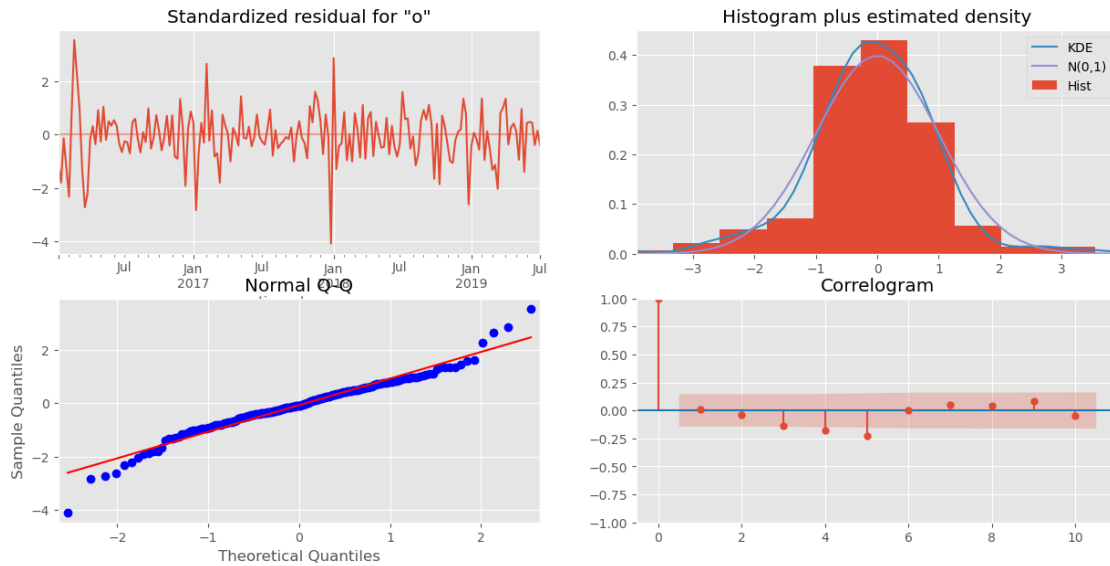
```

Prob(Q):	0.86	Prob(JB):	0.00
Heteroskedasticity (H):	0.61	Skew:	-0.24
Prob(H) (two-sided):	0.06	Kurtosis:	5.21

=====

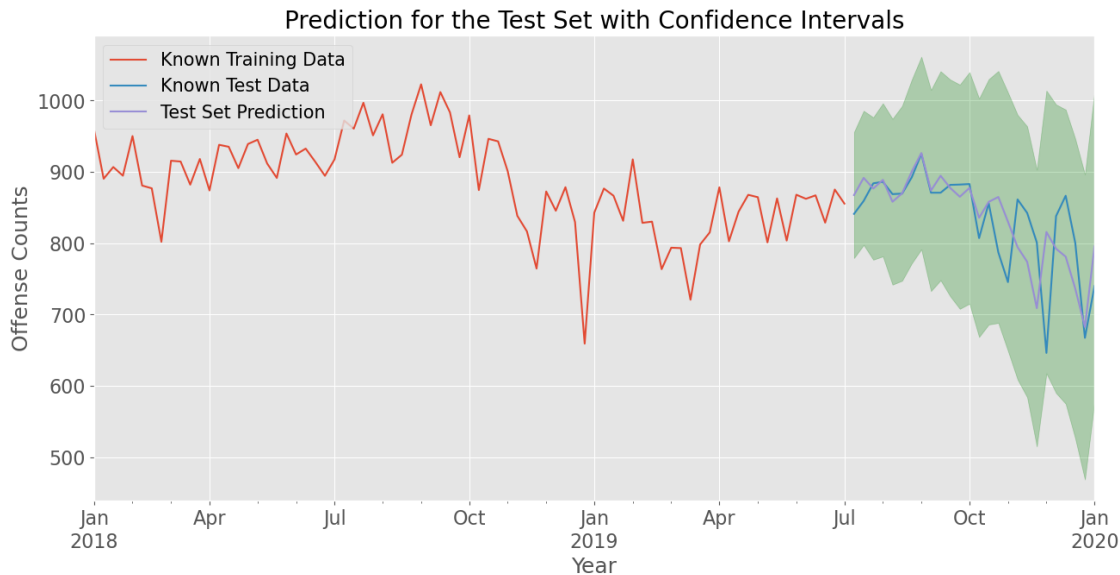
Warnings:

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



Testing

```
[41]: fig=predictions_testset(ts_train, ts_test, sarimax_mod1)
```



```
[42]: with open('images/pickled_figs/sarimax_mod1_train_test.pickle', 'wb') as f:
      pickle.dump(fig,f)
```

The RMSE value is significantly better than the RMSE for the Baseline model. Though visually the prediction for the train and test sets are not perfect.

Adding US holidays as exogenous regressors I will add holiday exogenous variables as an additional argument to the SARIMAX model to see if it helps improving the performance.

```
[43]: ts_holidays_weekly=us_holidays_predictors_TS(start='1/1/2015',
      end='12/31/2019',
      years=range(2015, 2020),
      freq='W')
```

```
[44]: # Splitting exogenous TS into the training and the test parts

train_size_holidays = round(len(ts_holidays_weekly) * 0.90)
ts_train_holiday, ts_test_holiday = ts_holidays_weekly[:train_size],
↳ts_holidays_weekly[train_size:]
```

```
[45]: # sarimax_mod2=SARIMAX(ts_train, exog=ts_train_holiday,
#                               order=(3, 1, 0),
#                               seasonal_order=(3, 1, 0, 52),
#                               enforce_invertibility=False).fit()
```

The model above took 1.5 minute to fit, therefore it is pickled to be used forward.

```
[46]: # with open('data/pickled_models/sarimax_mod2.pickle', 'wb') as f:
#       pickle.dump(sarimax_mod2, f)
```



```
[47]: with open('data/pickled_models/sarimax_mod2.pickle', 'rb') as f:
      sarimax_mod2=pickle.load(f)
```

```
[48]: diagnostics(sarimax_mod2)
```

```
<class 'statsmodels.iolib.summary.Summary'>
```

```
"""
```

SARIMAX Results

```
=====
Dep. Variable:          offense_id    No. Observations:          236
Model:                SARIMAX(3, 1, 0)x(3, 1, 0, 52)    Log Likelihood          -971.795
Date:                  Wed, 28 Jul 2021    AIC          1959.590
Time:                  19:46:29    BIC          1985.266
Sample:                01-04-2015    HQIC         1969.998
                  - 07-07-2019
```

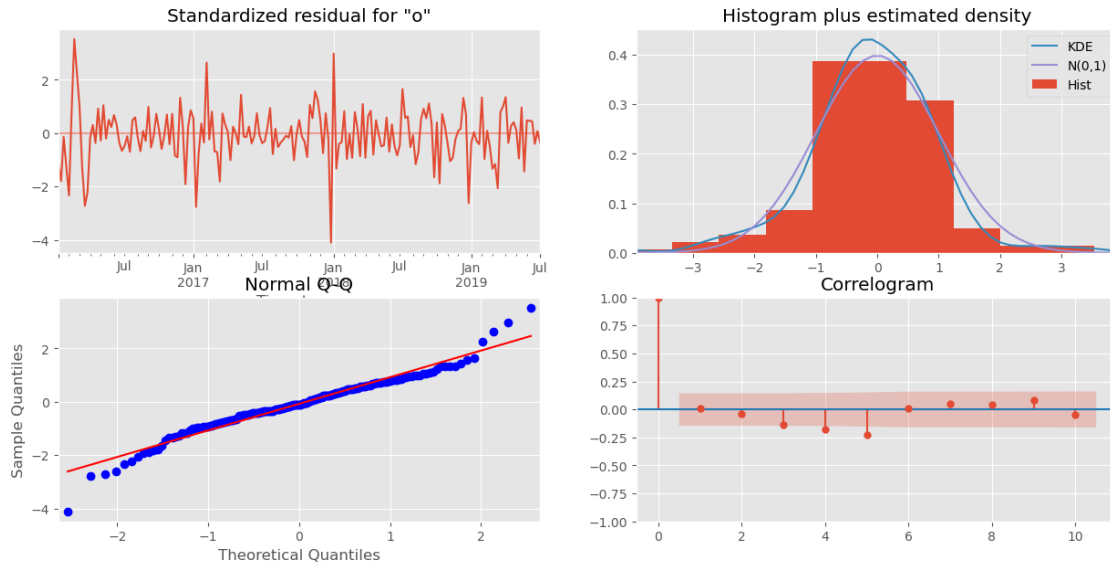
```
Covariance Type:                opg
```

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
Holiday         -5.5068     12.288     -0.448     0.654     -29.591     18.577
ar.L1            -0.6332      0.058    -10.933     0.000     -0.747     -0.520
ar.L2            -0.3928      0.067     -5.833     0.000     -0.525     -0.261
ar.L3            -0.1735      0.076     -2.276     0.023     -0.323     -0.024
ar.S.L52         -0.6326      0.092     -6.841     0.000     -0.814     -0.451
ar.S.L104        -0.4542      0.150     -3.020     0.003     -0.749     -0.159
ar.S.L156        -0.2450      0.163     -1.501     0.133     -0.565      0.075
sigma2          2021.2538    270.058      7.485     0.000    1491.949    2550.558
=====
```

```
Ljung-Box (L1) (Q):                0.03    Jarque-Bera (JB):                40.31
Prob(Q):                          0.87    Prob(JB):                  0.00
Heteroskedasticity (H):            0.61    Skew:                      -0.23
Prob(H) (two-sided):              0.06    Kurtosis:                   5.25
=====
```

```
Warnings:
```

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



This model performed very slightly worse than the one without exogenous regressors in terms of AIC value.

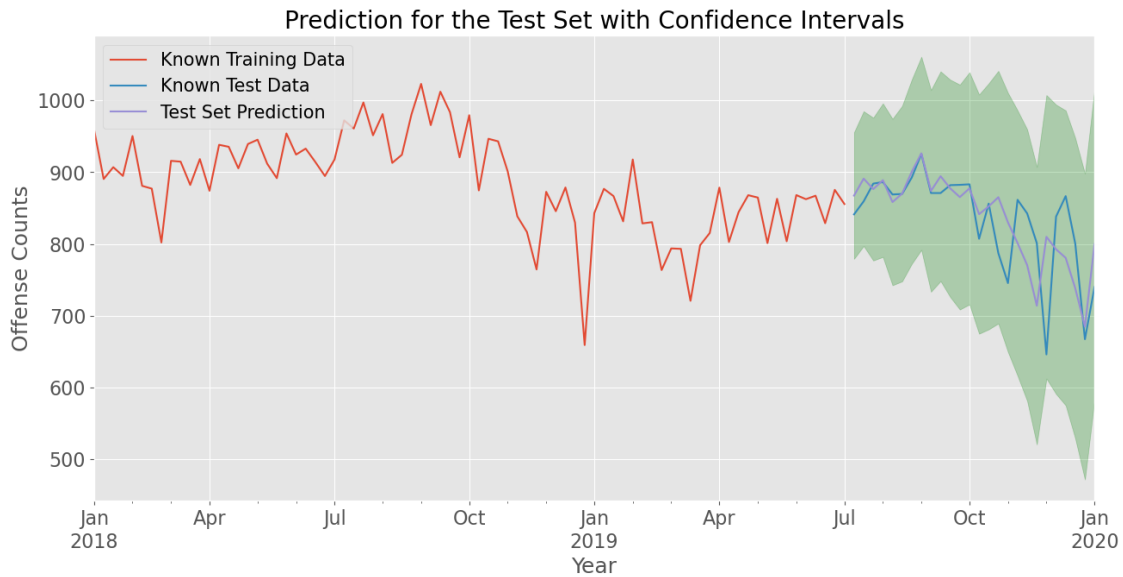
Testing

```
[49]: # plt.style.use('ggplot')
# y_hat_train=sarimax_mod2.predict(typ='levels')
# y_hat_test=sarimax_mod2.predict(start=ts_test.index[0], end=ts_test.
#     ↪ index[-1], exog=ts_test_holiday, typ='levels')

# rmse = np.sqrt(mean_squared_error(ts_test, y_hat_test))
# print('RMSE of the SARIMAX model (w US holidays) is {}'.format(round(rmse,2)))

# fig=display_figure_w_TSs(ts_train, ts_test, 'Train set', 'Test set',
# #     'Training and Test Sets Raw Values and Predictions'
#     ↪ (SARIMAX w US Holidays)',
# #         n=4, ts3=y_hat_test,
# #         ts4=y_hat_train, label3='Prediction for Test set',
#     ↪ label4='Prediction for Training set')

[50]: fig=predictions_testset(ts_train, ts_test, sarimax_mod2, egog_flag=True,
#     ↪ exog=ts_test_holiday)
```

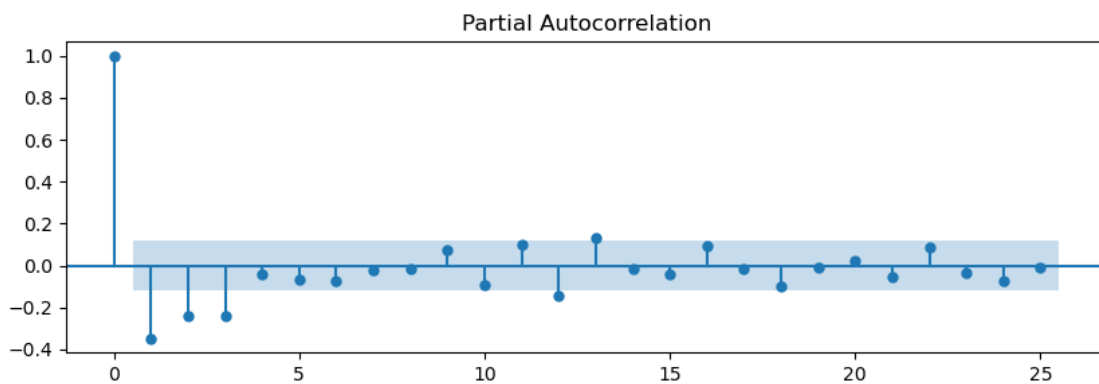


The RMSE value of this model is better than the one from the model w/o US holidays regressors.

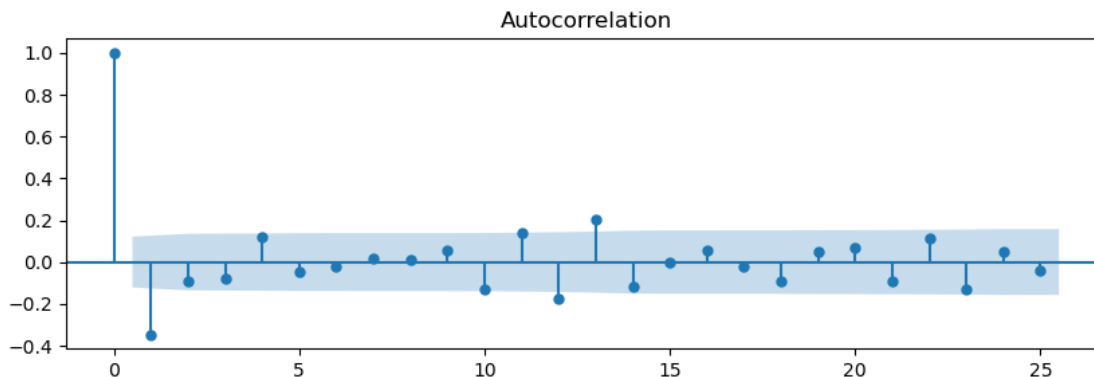
3.2.5 Forecasting

SARIMAX w/o US holidays ACF and PACF

```
[51]: matplotlib.rc_file_defaults()
plt.rc("figure", figsize=(10,3))
plot_pacf(ts_weekly.diff().dropna());
```



```
[52]: matplotlib.rc_file_defaults()
plt.rc("figure", figsize=(10,3))
plot_acf(ts_weekly.diff().dropna());
```



Based on PACF and ACF plots the same model would work best for the full dataset as well (not just the training subset).

Fitting the model to the full dataset

```
[53]: # sarimax_mod1_for=SARIMAX(ts_weekly,
#                               order=(3, 1, 0),
#                               seasonal_order=(3, 1, 0, 52),
#                               enforce_invertibility=False).fit()
```

The model above took 43 seconds to fit, therefore it is pickled to be used forward.

```
[54]: # with open('data/pickled_models/sarimax_mod1_for.pickle', 'wb') as f:
#         pickle.dump(sarimax_mod1_for, f)
```

```
[55]: with open('data/pickled_models/sarimax_mod1_for.pickle', 'rb') as f:
        sarimax_mod1_for=pickle.load(f)
```

```
[56]: diagnostics(sarimax_mod1_for)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

SARIMAX Results

```
=====
Dep. Variable:          offense_id    No. Observations:          262
Model:                SARIMAX(3, 1, 0)x(3, 1, 0, 52)    Log Likelihood          -1119.803
Date:                  Wed, 28 Jul 2021    AIC                    2253.605
Time:                  19:46:31    BIC                    2277.002
Sample:                01-04-2015    HQIC                   2263.065
                    - 01-05-2020

Covariance Type:                opg
=====
```

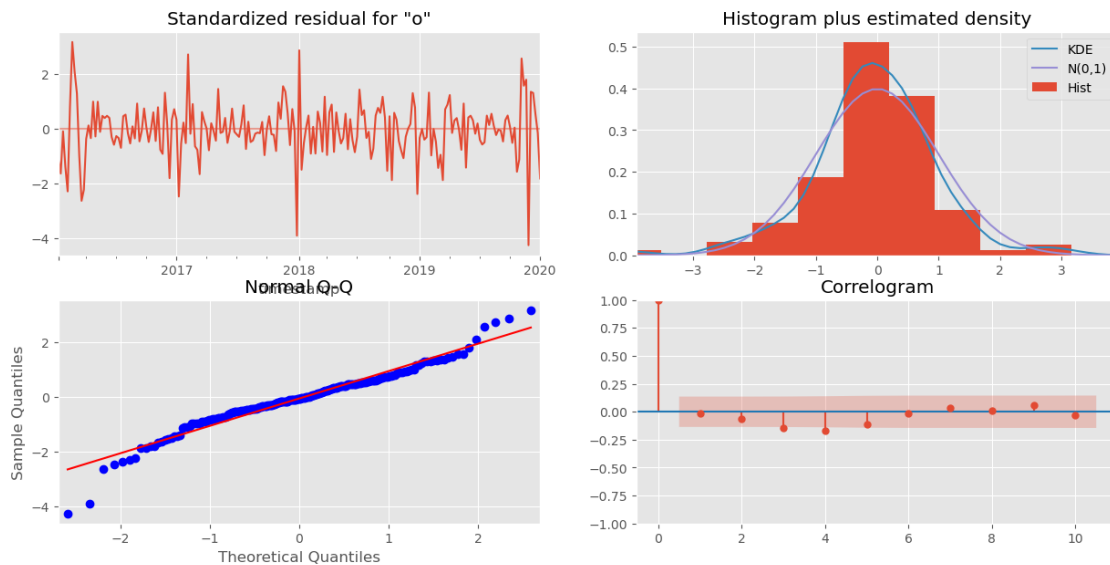
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.6327	0.050	-12.567	0.000	-0.731	-0.534

ar.L2	-0.4433	0.061	-7.311	0.000	-0.562	-0.324
ar.L3	-0.2734	0.058	-4.704	0.000	-0.387	-0.160
ar.S.L52	-0.5999	0.088	-6.819	0.000	-0.772	-0.427
ar.S.L104	-0.4110	0.124	-3.310	0.001	-0.654	-0.168
ar.S.L156	-0.1834	0.140	-1.312	0.189	-0.457	0.091
sigma2	2326.7001	217.935	10.676	0.000	1899.555	2753.846

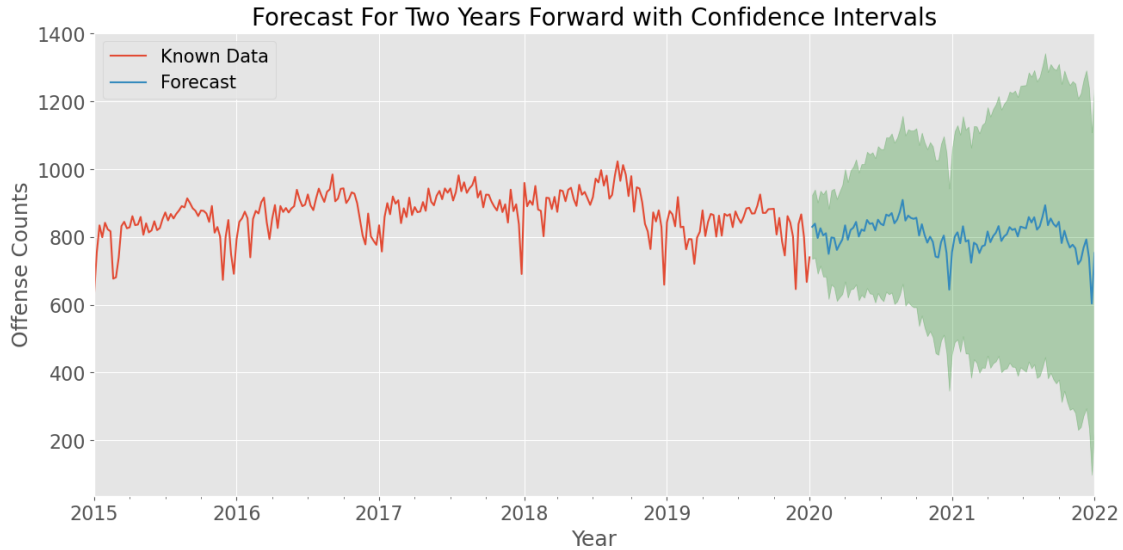
Ljung-Box (L1) (Q):	0.06	Jarque-Bera (JB):	65.24
Prob(Q):	0.81	Prob(JB):	0.00
Heteroskedasticity (H):	1.01	Skew:	-0.42
Prob(H) (two-sided):	0.97	Kurtosis:	5.60

Warnings:

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



```
[57]: fig=plot_predictions(ts_weekly, sarimax_mod1_for, 'Forecast For Two Years_
↳Forward with Confidence Intervals',
steps=104, xmin='2015')
```



```
[58]: with open('images/pickled_figs/sarimax_mod1_forecast.pickle', 'wb') as f:
      pickle.dump(fig,f)
```

SARIMAX with US holidays Fitting the model to the full dataset with full US holiday schedule

```
[59]: # sarimax_mod2_for=SARIMAX(ts_weekly, exog=ts_holidays_weekly,
#                               order=(3, 1, 0),
#                               seasonal_order=(3, 1, 0, 52),
#                               enforce_invertibility=False).fit()
```

The fitting of the model took 2.5 minutes therefore I am saving it to a pickle file.

```
[60]: # with open('data/pickled_models/sarimax_mod2_for.pickle', 'wb') as f:
#       pickle.dump(sarimax_mod2_for, f)
```

```
[61]: with open('data/pickled_models/sarimax_mod2_for.pickle', 'rb') as f:
      sarimax_mod2_for=pickle.load(f)
```

```
[62]: diagnostics(sarimax_mod2_for)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

SARIMAX Results

```
=====
Dep. Variable:          offense_id    No. Observations:          262
Model:                SARIMAX(3, 1, 0)x(3, 1, 0, 52)    Log Likelihood          -1117.886
Date:                  Wed, 28 Jul 2021    AIC                    2251.771
Time:                  19:46:33    BIC                    2278.510
Sample:                01-04-2015    HQIC                   2262.582
=====
```

- 01-05-2020

Covariance Type:

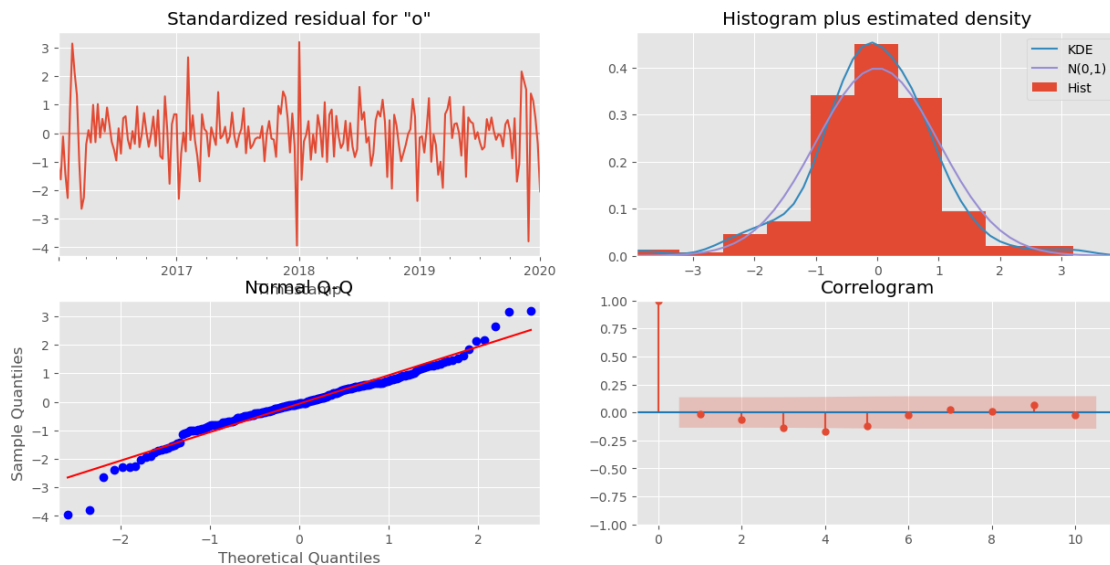
opg

	coef	std err	z	P> z	[0.025	0.975]
Holiday	-16.8103	6.897	-2.437	0.015	-30.328	-3.292
ar.L1	-0.6228	0.049	-12.602	0.000	-0.720	-0.526
ar.L2	-0.4621	0.061	-7.581	0.000	-0.582	-0.343
ar.L3	-0.2657	0.067	-3.936	0.000	-0.398	-0.133
ar.S.L52	-0.6029	0.091	-6.594	0.000	-0.782	-0.424
ar.S.L104	-0.4262	0.126	-3.372	0.001	-0.674	-0.178
ar.S.L156	-0.1967	0.144	-1.368	0.171	-0.478	0.085
sigma2	2271.3878	236.961	9.585	0.000	1806.952	2735.823

Ljung-Box (L1) (Q):	0.05	Jarque-Bera (JB):	45.46
Prob(Q):	0.83	Prob(JB):	0.00
Heteroskedasticity (H):	0.95	Skew:	-0.33
Prob(H) (two-sided):	0.84	Kurtosis:	5.19

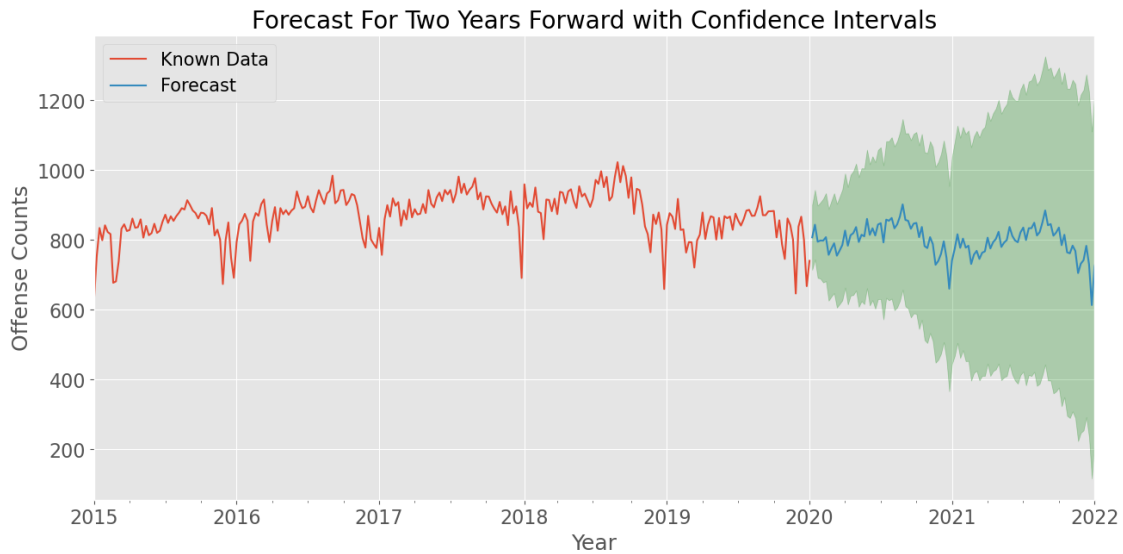
Warnings:

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



```
[63]: # I am extending the period to 105 weeks to cover all the span of the
      ↪ US_holidays TS above
      # (it includes 53 weeks+52 weeks next year)
```

```
fig=plot_predictions(ts_weekly, sarimax_mod2_for, 'Forecast For Two Years_
↳Forward with Confidence Intervals',
                    steps=105, xmin='2015',
                    egog_flag=True, exog=exog_reg_timeframe('1/1/2020', '1/1/
↳2022'))
```



Auto ARIMA search for the best parameters Training and testing

```
[64]: # auto_model_train = pmd.auto_arima(ts_train,
#                                         start_p=0, start_q=0, d=1,
#                                         max_p=3, max_q=1,
#                                         max_P=3, max_Q=1,
#                                         start_P=0, start_Q=0, D=1,
#                                         m=52, trace=True,
#                                         verbose=2)
```


Performing stepwise search to minimize aic

```

ARIMA(0,1,0)(0,1,0)[52] : AIC=2030.878, Time=0.14 sec
ARIMA(1,1,0)(1,1,0)[52] : AIC=1981.679, Time=2.04 sec
ARIMA(0,1,1)(0,1,1)[52] : AIC=inf, Time=4.15 sec
ARIMA(1,1,0)(0,1,0)[52] : AIC=2002.205, Time=0.30 sec
ARIMA(1,1,0)(2,1,0)[52] : AIC=1974.602, Time=8.17 sec
ARIMA(1,1,0)(3,1,0)[52] : AIC=inf, Time=31.73 sec
ARIMA(1,1,0)(2,1,1)[52] : AIC=inf, Time=18.88 sec
ARIMA(1,1,0)(1,1,1)[52] : AIC=inf, Time=12.71 sec
ARIMA(1,1,0)(3,1,1)[52] : AIC=1976.784, Time=44.05 sec
ARIMA(0,1,0)(2,1,0)[52] : AIC=2010.941, Time=6.07 sec
ARIMA(2,1,0)(2,1,0)[52] : AIC=1960.136, Time=11.07 sec
ARIMA(2,1,0)(1,1,0)[52] : AIC=1966.722, Time=3.15 sec
ARIMA(2,1,0)(3,1,0)[52] : AIC=inf, Time=38.62 sec
ARIMA(2,1,0)(2,1,1)[52] : AIC=inf, Time=23.66 sec
ARIMA(2,1,0)(1,1,1)[52] : AIC=inf, Time=7.88 sec
ARIMA(2,1,0)(3,1,1)[52] : AIC=1963.287, Time=53.10 sec
ARIMA(3,1,0)(2,1,0)[52] : AIC=1957.526, Time=14.56 sec
ARIMA(3,1,0)(1,1,0)[52] : AIC=1963.767, Time=3.88 sec
ARIMA(3,1,0)(3,1,0)[52] : AIC=inf, Time=45.00 sec
ARIMA(3,1,0)(2,1,1)[52] : AIC=inf, Time=30.04 sec
ARIMA(3,1,0)(1,1,1)[52] : AIC=inf, Time=19.36 sec
ARIMA(3,1,0)(3,1,1)[52] : AIC=1959.937, Time=62.02 sec
ARIMA(3,1,1)(2,1,0)[52] : AIC=1940.570, Time=29.35 sec
ARIMA(3,1,1)(1,1,0)[52] : AIC=1945.851, Time=8.29 sec
ARIMA(3,1,1)(3,1,0)[52] : AIC=1941.304, Time=78.07 sec
ARIMA(3,1,1)(2,1,1)[52] : AIC=inf, Time=62.33 sec
ARIMA(3,1,1)(1,1,1)[52] : AIC=inf, Time=21.50 sec
ARIMA(3,1,1)(3,1,1)[52] : AIC=1943.304, Time=87.79 sec
ARIMA(2,1,1)(2,1,0)[52] : AIC=1938.628, Time=23.60 sec
ARIMA(2,1,1)(1,1,0)[52] : AIC=1943.953, Time=6.60 sec
ARIMA(2,1,1)(3,1,0)[52] : AIC=1939.309, Time=62.31 sec
ARIMA(2,1,1)(2,1,1)[52] : AIC=inf, Time=32.64 sec
ARIMA(2,1,1)(1,1,1)[52] : AIC=inf, Time=21.80 sec
ARIMA(2,1,1)(3,1,1)[52] : AIC=1941.309, Time=65.30 sec
ARIMA(1,1,1)(2,1,0)[52] : AIC=1936.683, Time=15.17 sec
ARIMA(1,1,1)(1,1,0)[52] : AIC=1941.970, Time=4.47 sec
ARIMA(1,1,1)(3,1,0)[52] : AIC=1937.403, Time=47.51 sec
ARIMA(1,1,1)(2,1,1)[52] : AIC=inf, Time=24.39 sec
ARIMA(1,1,1)(1,1,1)[52] : AIC=inf, Time=19.71 sec
ARIMA(1,1,1)(3,1,1)[52] : AIC=1939.403, Time=56.63 sec
ARIMA(0,1,1)(2,1,0)[52] : AIC=1936.612, Time=10.39 sec
ARIMA(0,1,1)(1,1,0)[52] : AIC=1942.505, Time=2.69 sec
ARIMA(0,1,1)(3,1,0)[52] : AIC=1936.962, Time=34.36 sec
ARIMA(0,1,1)(2,1,1)[52] : AIC=inf, Time=17.49 sec
ARIMA(0,1,1)(1,1,1)[52] : AIC=inf, Time=13.34 sec
ARIMA(0,1,1)(3,1,1)[52] : AIC=1938.962, Time=44.74 sec
ARIMA(0,1,1)(2,1,0)[52] intercept : AIC=1933.412, Time=14.70 sec
ARIMA(0,1,1)(1,1,0)[52] intercept : AIC=1940.285, Time=5.96 sec
ARIMA(0,1,1)(3,1,0)[52] intercept : AIC=1933.673, Time=52.78 sec
ARIMA(0,1,1)(2,1,1)[52] intercept : AIC=inf, Time=53.25 sec
ARIMA(0,1,1)(1,1,1)[52] intercept : AIC=inf, Time=18.68 sec
ARIMA(0,1,1)(3,1,1)[52] intercept : AIC=inf, Time=65.89 sec
ARIMA(0,1,0)(2,1,0)[52] intercept : AIC=2012.806, Time=11.98 sec
ARIMA(1,1,1)(2,1,0)[52] intercept : AIC=1932.341, Time=28.83 sec
ARIMA(1,1,1)(1,1,0)[52] intercept : AIC=1938.748, Time=7.02 sec
ARIMA(1,1,1)(3,1,0)[52] intercept : AIC=1933.069, Time=65.45 sec
ARIMA(1,1,1)(2,1,1)[52] intercept : AIC=inf, Time=35.05 sec
ARIMA(1,1,1)(1,1,1)[52] intercept : AIC=inf, Time=21.06 sec
ARIMA(1,1,1)(3,1,1)[52] intercept : AIC=inf, Time=97.09 sec
ARIMA(1,1,0)(2,1,0)[52] intercept : AIC=1976.309, Time=18.49 sec
ARIMA(2,1,1)(2,1,0)[52] intercept : AIC=1934.090, Time=37.89 sec
ARIMA(2,1,0)(2,1,0)[52] intercept : AIC=1961.684, Time=28.25 sec

```

Best model: ARIMA(1,1,1)(2,1,0)[52] intercept

Total fit time: 1794.188 seconds

The search above took 29.5 minutes to run, therefore it is pickled to be used forward.

```
[65]: # with open('data/pickled_models/auto_model_train.pickle', 'wb') as f:
#       pickle.dump(auto_model_train, f)
```

```
[66]: with open('data/pickled_models/auto_model_train.pickle', 'rb') as f:
auto_model_train=pickle.load(f)
```

```
[67]: model_auto_train = tsa.SARIMAX(ts_train, order=auto_model_train.order,
seasonal_order=auto_model_train.seasonal_order,
enforce_invertibility=False, freq='W').fit()
```

```
[68]: with open('data/pickled_models/model_auto_train.pickle', 'wb') as f:
pickle.dump(model_auto_train, f)
```

```
[69]: diagnostics(model_auto_train)
```

```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

SARIMAX Results

```
=====
Dep. Variable:          offense_id    No. Observations:          236
Model:                SARIMAX(1, 1, 1)x(2, 1, [], 52)    Log Likelihood          -963.341
Date:                  Wed, 28 Jul 2021    AIC          1936.683
Time:                  19:46:52    BIC          1952.730
Sample:                01-04-2015    HQIC          1943.188
                    - 07-07-2019
```

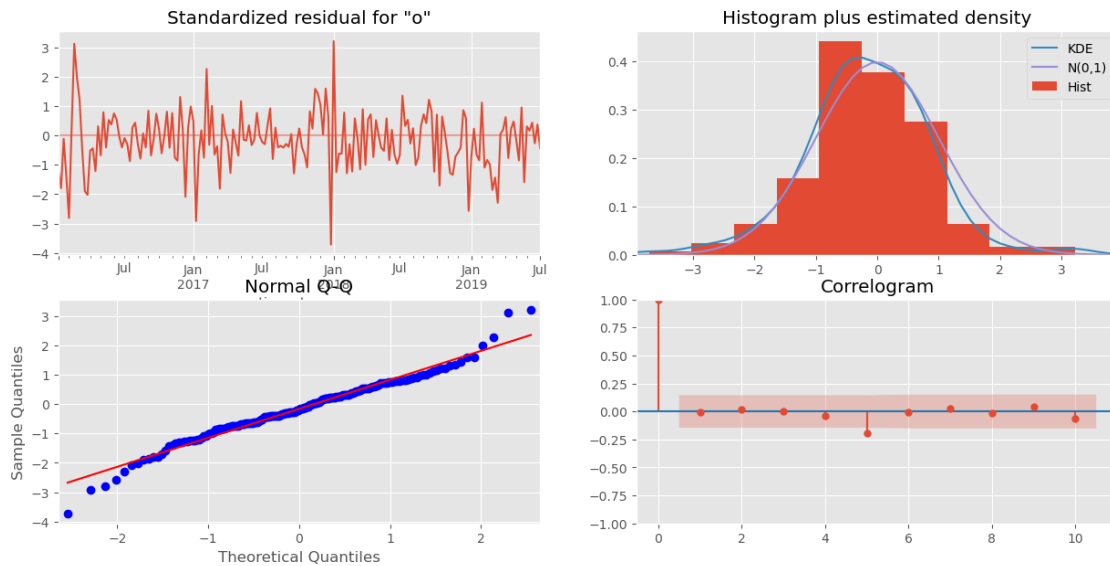
```
Covariance Type:          opg
```

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
ar.L1          0.1223     0.065      1.885     0.059     -0.005     0.249
ma.L1         -0.8601     0.041     -20.853     0.000     -0.941    -0.779
ar.S.L52       -0.4910     0.061     -8.081     0.000     -0.610    -0.372
ar.S.L104      -0.2871     0.093     -3.091     0.002     -0.469    -0.105
sigma2        1979.7726    194.117     10.199     0.000    1599.309    2360.236
=====
```

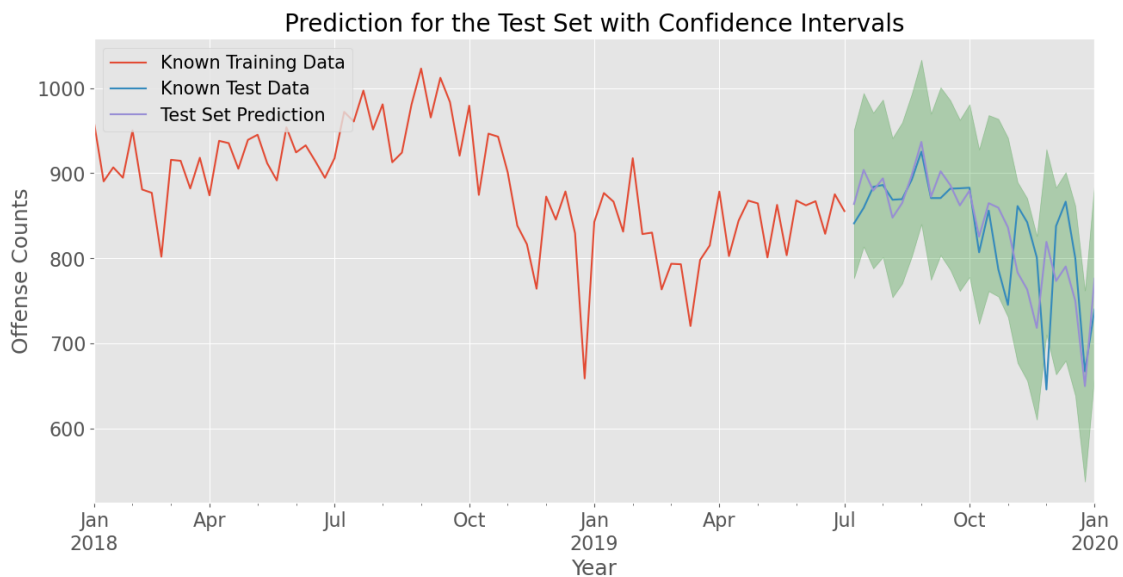
```
=====
Ljung-Box (L1) (Q):          0.00    Jarque-Bera (JB):          19.00
Prob(Q):                    0.97    Prob(JB):              0.00
Heteroskedasticity (H):      0.70    Skew:                  -0.10
Prob(H) (two-sided):         0.16    Kurtosis:              4.57
=====
```

Warnings:

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



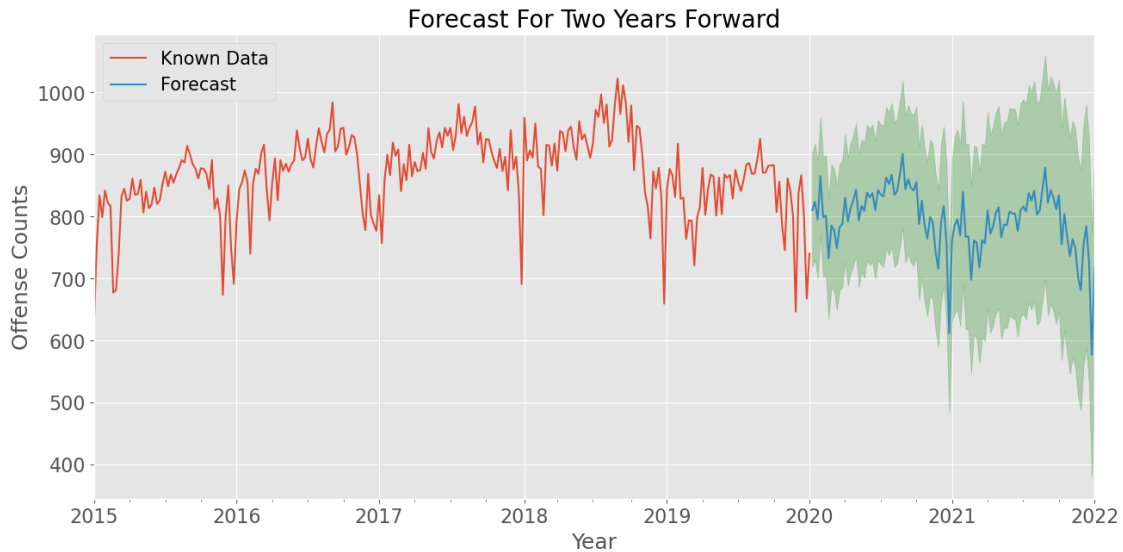
```
[74]: fig=predictions_testset(ts_train, ts_test, model_auto_train, xmin='2018')
```



```
[71]: model_auto_for = tsa.SARIMAX(ts_weekly, order=auto_model_train.order,
    seasonal_order=auto_model_train.seasonal_order,
    enforce_invertibility=False, freq='W').fit()
```

```
[72]: with open('data/pickled_models/model_auto_for.pickle', 'wb') as f:
    pickle.dump(model_auto_for, f)
```

```
[73]: fig=plot_predictions(ts_weekly, model_auto_for, 'Forecast For Two Years_
↳Forward', steps=104, xmin='2015')
```



```
[74]: with open('images/pickled_figs/auto_arima_forecast.pickle', 'wb') as f:
pickle.dump(fig,f)
```

```
[75]: plot_predictions_px(ts_weekly, model_auto_for, 'Crime Data and Forecast for Two_
↳Years', xmin='2015')
```

It takes ~3 minutes to run this notebook

All Crime rate modeling for various crime categories are located in [part IV notebook](#).

```
[ ]:
```