databrickslatest

(https://databricks.com)

# SF crime data analysis and modeling

## Import package

```
from csv import reader
from pyspark.sql import Row
from pyspark.sql import SparkSession
from pyspark.sql.types import *
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import warnings

import os
os.environ["PYSPARK_PYTHON"] = "python3"
```

```
# 从SF gov 官网读取下载数据
import urllib.request
urllib.request.urlretrieve("https://data.sfgov.org/api/views/tmnf-yvry/rows.csv?accessType=DOWNLOAD", "/tmp/myxxxx.csv")
dbutils.fs.mv("file:/tmp/myxxxx.csv", "dbfs:/data/sf_03_18.csv")
display(dbutils.fs.ls("dbfs:/data/"))
## 自己下载
# https://data.sfgov.org/api/views/tmnf-yvry/rows.csv?accessType=DOWNLOAD
```

Table

| | path | name | size | modificationTime |
|---|---|---|---|---|
| 1 | dbfs:/laioffer/spark_hw1/data/la_crime.csv | la_crime.csv | 536364291 | 1674353616000 |
| 2 | dbfs:/laioffer/spark_hw1/data/sf_03_18.csv | sf_03_18.csv | 550945238 | 1674706441000 |

2 rows

```
data_path = "dbfs:/data/sf_03_18.csv"
# use this file name later
```

## Get dataframe and sql

```
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("crime analysis") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

df_opt1 = spark.read.format("csv").option("header", "true").load(data_path)
display(df_opt1)
df_opt1.createOrReplaceTempView("sf_crime")
```

Table

| | PdId | IncidntNum | Incident Code | Category | Descript |
|---|---|---|---|---|---|
| 1 | 4133422003074 | 041334220 | 03074 | ROBBERY | ROBBERY, BODILY FORCE |
| 2 | 5118535807021 | 051185358 | 07021 | VEHICLE THEFT | STOLEN AUTOMOBILE |
| 3 | 4018830907021 | 040188309 | 07021 | VEHICLE THEFT | STOLEN AUTOMOBILE |
| 4 | 11014543126030 | 110145431 | 26030 | ARSON | ARSON |
| 5 | 10108108004134 | 101081080 | 04134 | ASSAULT | BATTERY |
| 6 | 13027069804134 | 130270698 | 04134 | ASSAULT | BATTERY |

1,000 rows | Truncated data

```
df_opt2 = df_opt1[['IncidntNum', 'Category', 'Descript', 'DayOfWeek', 'Date', 'Time', 'PdDistrict', 'Resolution', 'Address', 'X', 'Y', 'Locati
display(df_opt2)
df_opt2.createOrReplaceTempView("sf_crime")
```

Table

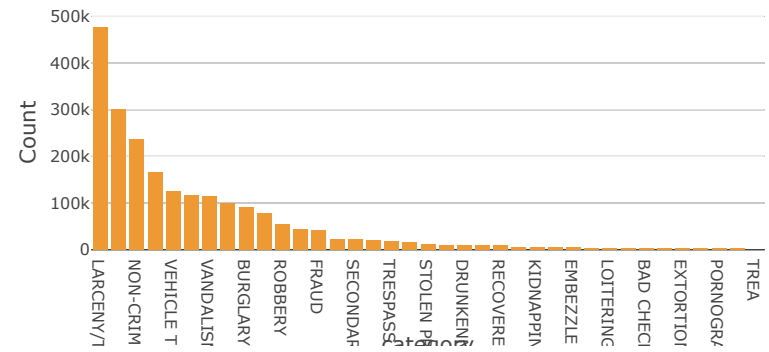
| | IncidntNum | Category | Descript | DayOfWeek | Date | Ti |
|---|---|---|---|---|---|---|
| 1 | 041334220 | ROBBERY | ROBBERY, BODILY FORCE | Monday | 11/22/2004 | 17 |
| 2 | 051185358 | VEHICLE THEFT | STOLEN AUTOMOBILE | Tuesday | 10/18/2005 | 20 |
| 3 | 040188309 | VEHICLE THEFT | STOLEN AUTOMOBILE | Sunday | 02/15/2004 | 02 |
| 4 | 110145431 | ARSON | ARSON | Friday | 02/18/2011 | 05 |
| 5 | 101081080 | ASSAULT | BATTERY | Sunday | 11/21/2010 | 17 |
| 6 | 130270698 | ASSAULT | BATTERY | Tuesday | 04/02/2013 | 15 |

1,000 rows | Truncated data

## The number of crimes for different category

```
spark_sql_q1 = spark.sql("SELECT  category, COUNT(*) AS Count FROM sf_crime GROUP BY category ORDER BY Count DESC")
display(spark_sql_q1)
```

Visualization




37 rows

## panda dataframe

```
crimes_pd_df = crimeCategory.toPandas()
```

```
#number of crimes for different district
spark_sql_q2 = spark.sql("SELECT PdDistrict, COUNT(*) AS Count FROM sf_crime GROUP BY 1 ORDER BY 2")
display(spark_sql_q2)
```

Table Visualization 1

| | PdDistrict | Count |
|---|---|---|
| 1 | NA | 1 |
| 2 | RICHMOND | 112804 |
| 3 | PARK | 119698 |
| 4 | TARAVAL | 155461 |
| 5 | INGLESIDE | 181092 |
| 6 | TENDERLOIN | 186954 |

11 rows

The number of crimes each "Sunday" at "SF downtown".

```
from pyspark.sql.functions import to_date, to_timestamp
df_opt2_new = df_opt2.withColumn('Date_o', to_date(df_opt2.Date, "M/d/y"))
display(df_opt2_new)
df_opt2_new.createOrReplaceTempView("sf_crime")
```

Table

| | IncidntNum | Category | Descript | DayOfWeek | Date | Ti |
|---|---|---|---|---|---|---|
| 1 | 041334220 | ROBBERY | ROBBERY, BODILY FORCE | Monday | 11/22/2004 | 17 |
| 2 | 051185358 | VEHICLE THEFT | STOLEN AUTOMOBILE | Tuesday | 10/18/2005 | 20 |
| 3 | 040188309 | VEHICLE THEFT | STOLEN AUTOMOBILE | Sunday | 02/15/2004 | 02 |
| 4 | 110145431 | ARSON | ARSON | Friday | 02/18/2011 | 08 |
| 5 | 101081080 | ASSAULT | BATTERY | Sunday | 11/21/2010 | 17 |
| 6 | 130270698 | ASSAULT | BATTERY | Tuesday | 04/02/2013 | 15 |

1,000 rows | Truncated data

```
spark_sql_q3 = spark.sql("""
                        SELECT Date_o, COUNT(*)
                        FROM sf_crime
                        WHERE DayOfWeek = 'Sunday'
                              and x > -122.4313
                              and x < -122.4213
                              and y > 37.7540
                              and y < 37.7740
                        Group by 1
                        order by 1
                        """)
display(spark_sql_q3)
```

Table Visualization 1

| | Date_o | count(1) |
|---|---|---|
| 1 | 2003-01-05 | 13 |
| 2 | 2003-01-12 | 20 |
| 3 | 2003-01-19 | 17 |
| 4 | 2003-01-26 | 13 |
| 5 | 2003-02-02 | 14 |
| 6 | 2003-02-09 | 22 |

801 rows

## The number of crime in each month of 2015, 2016, 2017, 2018.

```
from pyspark.sql.functions import to_date, month, year
df_opt2_new = df_opt2_new.withColumn('Month',month(df_opt2_new['Date_o']))
df_opt2_new = df_opt2_new.withColumn('Year', year(df_opt2_new['Date_o']))
display(df_opt2_new)
df_opt2_new.createOrReplaceTempView("sf_crime")
```

Table

| | IncidntNum | Category | Descript | DayOfWeek | Date | Ti |
|---|---|---|---|---|---|---|
| 1 | 041334220 | ROBBERY | ROBBERY, BODILY FORCE | Monday | 11/22/2004 | 17 |
| 2 | 051185358 | VEHICLE THEFT | STOLEN AUTOMOBILE | Tuesday | 10/18/2005 | 20 |
| 3 | 040188309 | VEHICLE THEFT | STOLEN AUTOMOBILE | Sunday | 02/15/2004 | 02 |
| 4 | 110145431 | ARSON | ARSON | Friday | 02/18/2011 | 08 |
| 5 | 101081080 | ASSAULT | BATTERY | Sunday | 11/21/2010 | 17 |
| 6 | 130270698 | ASSAULT | BATTERY | Tuesday | 04/02/2013 | 18 |

1,000 rows | Truncated data

```
spark_sql_q4 = spark.sql("""
                        SELECT Year,Month, Count(*)
                        FROM sf_crime
                        where Year in (2015,2016,2017,2018)
                        Group by 1,2
                        order by 1,2
                        """)
display(spark_sql_q4)
```

Table Visualization 1

| | Year | Month | count(1) |
|---|---|---|---|
| 1 | 2015 | 1 | 13181 |
| 2 | 2015 | 2 | 11882 |
| 3 | 2015 | 3 | 13463 |
| 4 | 2015 | 4 | 12526 |
| 5 | 2015 | 5 | 13318 |
| 6 | 2015 | 6 | 12853 |

41 rows

```
spark_sql_q4 = spark.sql("""
                        SELECT Year,Month, Count(*)
                        FROM sf_crime
                        where Year in (2015,2016,2017,2018)
                        Group by 1,2
                        order by 1,2
                        """)
display(spark_sql_q4)
```

Table Visualization 2

| | Year | Month | count(1) |
|---|---|---|---|
| 1 | 2015 | 1 | 13181 |
| 2 | 2015 | 2 | 11882 |
| 3 | 2015 | 3 | 13463 |
| 4 | 2015 | 4 | 12526 |
| 5 | 2015 | 5 | 13318 |
| 6 | 2015 | 6 | 12853 |

41 rows

# insight

1. from visualization 1, the crome data might not be collected completely in May of 2018.
2. from visualization 1, although we only have 5-month crime data for 2018, we can observe that the number of crime decreases suddently.
3. from visualization 1, crime rate seems to be high in January
4. from visualization 2, overall, through the year, the crime number in 2016 is less than the ones in 2015 and 2017.

```
spark_sql_q4 = spark.sql("""

                        select Year, avg(cnt)
                        from
                        (SELECT Year,Month, Count(*) cnt
                        FROM sf_crime
                        where Year in (2015,2016,2017,2018)
                        Group by 1,2
                        order by 1,2) a
                        group by 1
                        order by 1
                        """)
```

```
display(spark_sql_q4)
```

Table

| | Year | avg(cnt) |
|---|---|---|
| 1 | 2015 | 12621.583333333334 |
| 2 | 2016 | 12166.166666666666 |
| 3 | 2017 | 12457.25 |
| 4 | 2018 | 9011.8 |

4 rows

```
# did not use 2018 since it is exception
spark_sql_q4 = spark.sql("""

                        select Month, avg(cnt)
                        from
                        (SELECT Year,Month, Count(*) cnt
                        FROM sf_crime
                        where Year in (2015,2016,2017)
                        Group by 1,2
                        order by 1,2) a
                        group by 1
                        order by 1
                        """)
```

```
display(spark_sql_q4)
```

Table

| | Month | avg(cnt) |
|---|---|---|
| 1 | 1 | 12800 |
| 2 | 2 | 11801.666666666666 |
| 3 | 3 | 12887.666666666666 |
| 4 | 4 | 12294.333333333334 |
| 5 | 5 | 12796.333333333334 |
| 6 | 6 | 12255.333333333334 |

12 rows

The means of years and the means of months proved the observation

### The number of crime with respsect to the hour in certian day like 2015/12/15, 2016/12/15, 2017/12/15.

```
from pyspark.sql.functions import to_timestamp, hour
df_opt2_new = df_opt2_new.withColumn('Time', to_timestamp(df_opt2_new.Time, "HH:mm"))
df_opt2_new = df_opt2_new.withColumn('Hour', hour(df_opt2_new['Time']))
display(df_opt2_new)
df_opt2_new.createOrReplaceTempView("sf_crime")
```

Table

| | IncidntNum | Category | Descript | DayOfWeek | Date | Ti |
|---|---|---|---|---|---|---|
| 1 | 041334220 | ROBBERY | ROBBERY, BODILY FORCE | Monday | 11/22/2004 | 19 |
| 2 | 051185358 | VEHICLE THEFT | STOLEN AUTOMOBILE | Tuesday | 10/18/2005 | 19 |
| 3 | 040188309 | VEHICLE THEFT | STOLEN AUTOMOBILE | Sunday | 02/15/2004 | 19 |
| 4 | 110145431 | ARSON | ARSON | Friday | 02/18/2011 | 19 |
| 5 | 101081080 | ASSAULT | BATTERY | Sunday | 11/21/2010 | 19 |
| 6 | 130270698 | ASSAULT | BATTERY | Tuesday | 04/02/2013 | 19 |

1,000 rows | Truncated data

```
spark_sql_q5 = spark.sql("""
                        select Hour, count(*)
                        from sf_crime
                        where Date like '12/15/%'
                        group by 1
                        order by 1
                        """)
display(spark_sql_q5)
```

Table Visualization 1

| | Hour | count(1) |
|---|---|---|
| 1 | 0 | 312 |
| 2 | 1 | 131 |
| 3 | 2 | 143 |
| 4 | 3 | 71 |
| 5 | 4 | 74 |
| 6 | 5 | 60 |

24 rows

We can observe that there is less crime occur from 1 am to 7 am. And there is crime peak around 12 pm; meanwhile, there is a increasing crime trend toward 18 pm, which are lunch and dinner time. Therefore, i would suggest travellers keep alert when they are having food.

(1) The top-3 danger disrict
(2) The crime event w.r.t category and time (hour) from the result of 1

```
spark_sql_q61 = spark.sql("""
                        select PdDistrict, count(*)
                        from sf_crime
                        group by 1
                        order by 2 desc
                        limit 3
                        """)
```

```
display(spark_sql_q61)
```

Table

| | PdDistrict | count(1) |
|---|---|---|
| 1 | SOUTHERN | 390692 |
| 2 | MISSION | 288985 |
| 3 | NORTHERN | 266435 |

3 rows

```
spark_sql_q62 = spark.sql("""
                          select PdDistrict, Category, Hour, count(*)
                          from sf_crime
                          where PdDistrict in (
                         select PdDistrict
                         from sf_crime
                         group by 1
                         order by count(*) desc
                         limit 3)
                         group by 1,3,2
                         order by 1,2,3
                         """)
```

```
display(spark_sql_q62)
```

Table Visualization 1

| | PdDistrict | Category | Hour | count(1) |
|---|---|---|---|---|
| 1 | MISSION | ARSON | 0 | 35 |
| 2 | MISSION | ARSON | 1 | 23 |
| 3 | MISSION | ARSON | 2 | 28 |
| 4 | MISSION | ARSON | 3 | 36 |
| 5 | MISSION | ARSON | 4 | 27 |
| 6 | MISSION | ARSON | 5 | 26 |

1,000 rows | Truncated data

```
spark_sql_q62 = spark.sql("""

                          select PdDistrict, Category, Hour, count(*)
                          from sf_crime
                          where PdDistrict in (
                         select PdDistrict
                         from sf_crime
                         group by 1
                         order by count(*) desc
                         limit 3)
                         group by 1,3,2
                         order by 1,2,3
                         """)
```

```
display(spark_sql_q62)
```

Table Visualization 1

| | PdDistrict | Category | Hour | count(1) |
|---|---|---|---|---|
| 1 | MISSION | ARSON | 0 | 35 |
| 2 | MISSION | ARSON | 1 | 23 |
| 3 | MISSION | ARSON | 2 | 28 |
| 4 | MISSION | ARSON | 3 | 36 |
| 5 | MISSION | ARSON | 4 | 27 |
| 6 | MISSION | ARSON | 5 | 26 |

1,000 rows | Truncated data

From step1, top-3 danger districts are SOUTHERN, MISSION and NORTHERN. From step2, assault is more possible to happen around 12 am and most of burglary might happen in the ealry morning. There is more possibility of theft happen at night. The police should focus on different type of crime during different time. At the same time, we can also notice that the crime number from 1 to 7 are much lesser, so we can reduce police power during these time and focus on the other time

the percentage of resolution for different category of crime.

```
spark_sql_q7 = spark.sql("""
select a.category,resolution, (a.cnt_a*100)/b.cnt_b percentage
from (
select category, resolution, count(*) cnt_a
from sf_crime
group by 1,2)a
left join (select category, count(*) cnt_b
from sf_crime
group by 1) b
on a.category =b.category
order by 2 desc
                        """)
display(spark_sql_q7)
```

Table Visualization 1

| | category | resolution | percentage |
|---|---|---|---|
| 1 | BAD CHECKS | UNFOUNDED | 1.2987012987012987 |
| 2 | DISORDERLY CONDUCT | UNFOUNDED | 0.6041079339508659 |
| 3 | VANDALISM | UNFOUNDED | 0.3783190083509127 |
| 4 | BURGLARY | UNFOUNDED | 0.5984604741563904 |
| 5 | DRUG/NARCOTIC | UNFOUNDED | 0.18587518354113444 |
| 6 | PROSTITUTION | UNFOUNDED | 0.1515059693351918 |

382 rows

```
spark_sql_q72 = spark.sql("""
select a.category, (a.cnt_a*100)/b.cnt_b percentage
from (
select category, count(*) cnt_a
from sf_crime
where resolution != 'NONE'
group by 1)a
left join (select category, count(*) cnt_b
from sf_crime
group by 1) b
on a.category =b.category
order by 2 desc
                        """)
display(spark_sql_q72)
```

Table

| | category | percentage |
|---|---|---|
| 1 | PROSTITUTION | 94.84879704260348 |
| 2 | WARRANTS | 94.50816962362629 |
| 3 | DRIVING UNDER THE INFLUENCE | 94.39136588818117 |
| 4 | DRUG/NARCOTIC | 91.2952699433887 |
| 5 | LIQUOR LAWS | 88.97887323943662 |
| 6 | LOITERING | 87.55203996669442 |

37 rows

Most of the crime are unresolved. The policy should increase police system. Observing the percentage of the cases solved, the police are good at dealing with crime like prostitution, warrant, driving under the influence, which have harsh penalty by law, however, theft cases and recovered vehicle are the crime that is hard for the police to handle.

Analysis the new columns of the data

```
df_opt1.createOrReplaceTempView("sf_crime_1")
display(df_opt1)
```

Table

| | PdId | IncidntNum | Incident Code | Category | Descript |
|---|---|---|---|---|---|
| 1 | 4133422003074 | 041334220 | 03074 | ROBBERY | ROBBERY, BODILY FORCE |
| 2 | 5118535807021 | 051185358 | 07021 | VEHICLE THEFT | STOLEN AUTOMOBILE |
| 3 | 4018830907021 | 040188309 | 07021 | VEHICLE THEFT | STOLEN AUTOMOBILE |
| 4 | 11014543126030 | 110145431 | 26030 | ARSON | ARSON |
| 5 | 10108108004134 | 101081080 | 04134 | ASSAULT | BATTERY |
| 6 | 13027069804134 | 130270698 | 04134 | ASSAULT | BATTERY |

1,000 rows | Truncated data

```
spark_sql_q72 = spark.sql("""
select `Current Police Districts 2 2`, count(*) cnt
from sf_crime_1
where `Current Police Districts 2 2` is not Null
group by 1
order by 2 desc
                    """)
display(spark_sql_q72)
```

Table

| | Current Police Districts 2 2 | cnt |
|---|---|---|
| 1 | 7 | 103800 |
| 2 | 8 | 106412 |
| 3 | 10 | 156045 |
| 4 | 9 | 180294 |
| 5 | 2 | 195479 |
| 6 | 4 | 260521 |

10 rows

Higher current police district number tend to have less crime cases

## Conclusion.

There are more and more crime happened these day. And safety is one of the most important issue citizens care so much. SF is one of the famous cities in USA, but there are countless crime happened in SF every month; therefore, I hope this research is to analyze the present crime data so that police and policy maker can improve the society by making changes. Through the reasearch the crime numbers hold steady until 2018, I believe that there is a huge improvement in 2018. The series districts are southern mission and northern. And the cime peak are around 12 pm and 18 pm, which are the lunch and dinner time. We also find out that most of the cases are unsolved, which should be brought to the mind. We must impove our police system so that citizen can live in a better envirnment.

## Time series analysis

process:
1.visualize time series
2.plot ACF and find optimal parameter
3.Train ARIMA
4.Prediction

```
# Importing libraries
from statsmodels.tsa.stattools import adfuller
import os
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

from pylab import rcParams

import statsmodels.api as sm
from numpy.random import normal, seed
from scipy.stats import norm
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.tsa.arima_model import ARIMA
import math
from sklearn.metrics import mean_squared_error

from plotly import tools

from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.figure_factory as ff
```

```
spark_sql_ts = spark.sql("""
                         SELECT Date_o, COUNT(*) num_crime
                         FROM sf_crime
                         where Date_o < '2018-01-01'
                         Group by 1
                         order by 1
                         """)
```

```
display(spark_sql_ts)
```

Table Visualization 1

| | Date_o | num_crime |
|---|---|---|
| 1 | 2003-01-01 | 582 |
| 2 | 2003-01-02 | 384 |
| 3 | 2003-01-03 | 418 |
| 4 | 2003-01-04 | 328 |
| 5 | 2003-01-05 | 367 |
| 6 | 2003-01-06 | 389 |

1,000 rows | Truncated data

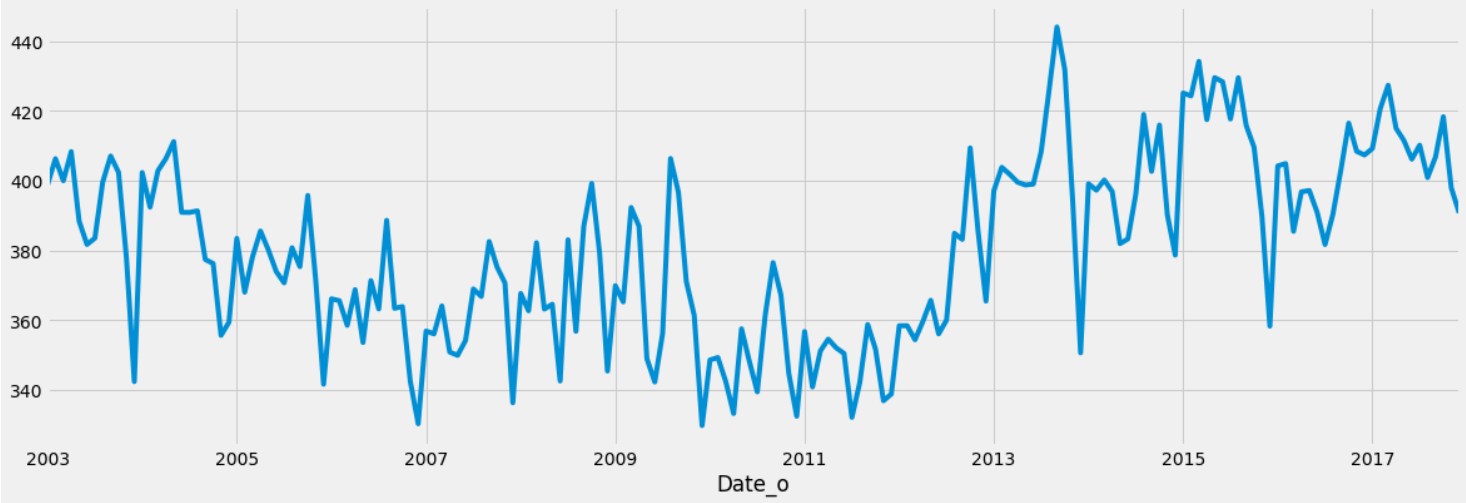
```
crimes_df = spark_sql_ts.toPandas()
crimes_df.Date_o=pd.to_datetime(crimes_df.Date_o)
```

```
crimes_df=crimes_df.set_index(['Date_o'])
crimes_df
```

| Date_o | num_crime |
|---|---|
| 2003-01-01 | 582 |
| 2003-01-02 | 384 |
| 2003-01-03 | 418 |
| 2003-01-04 | 328 |
| 2003-01-05 | 367 |
| ... | ... |
| 2017-12-27 | 374 |
| 2017-12-28 | 408 |
| 2017-12-29 | 401 |
| 2017-12-30 | 418 |
| 2017-12-31 | 413 |

5478 rows × 1 columns

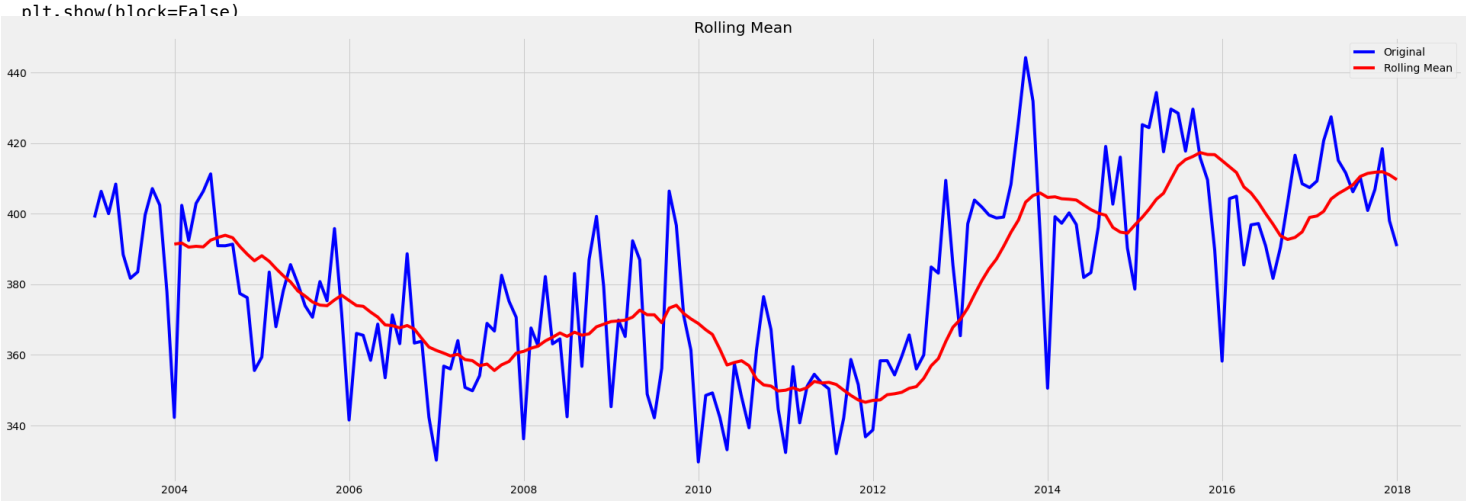
```
y = crimes_df['num_crime'].resample('M').mean()
y.plot(figsize=(18, 6))
plt.show()
```



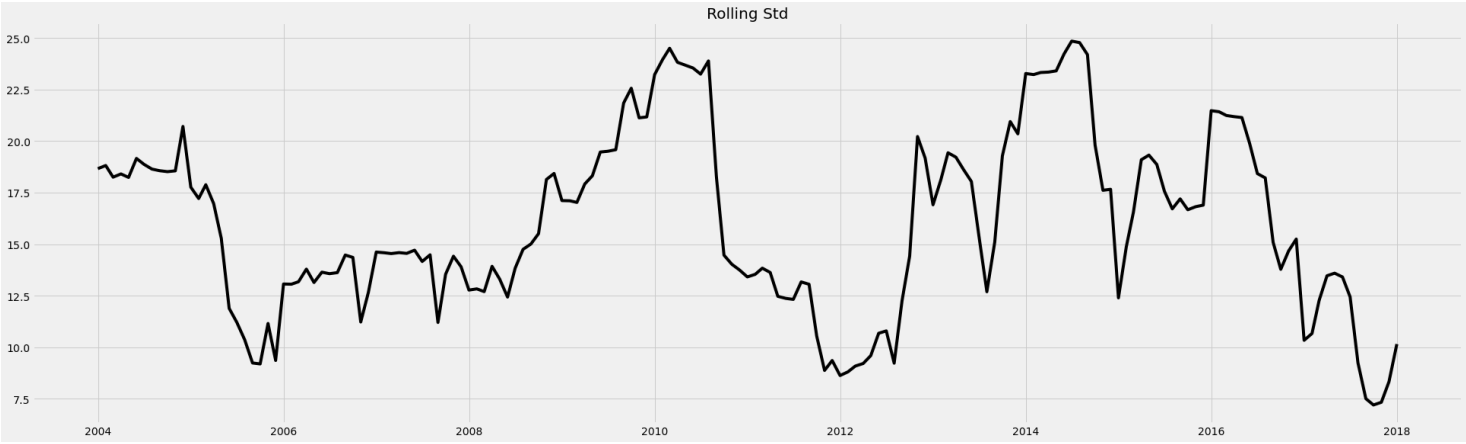

```
y.describe()
```

```
Out[33]: count    180.000000
mean     380.517255
std       26.156167
min      329.709677
25%      358.457731
50%      381.817742
75%      400.395161
max      444.166667
Name: num_crime, dtype: float64
```

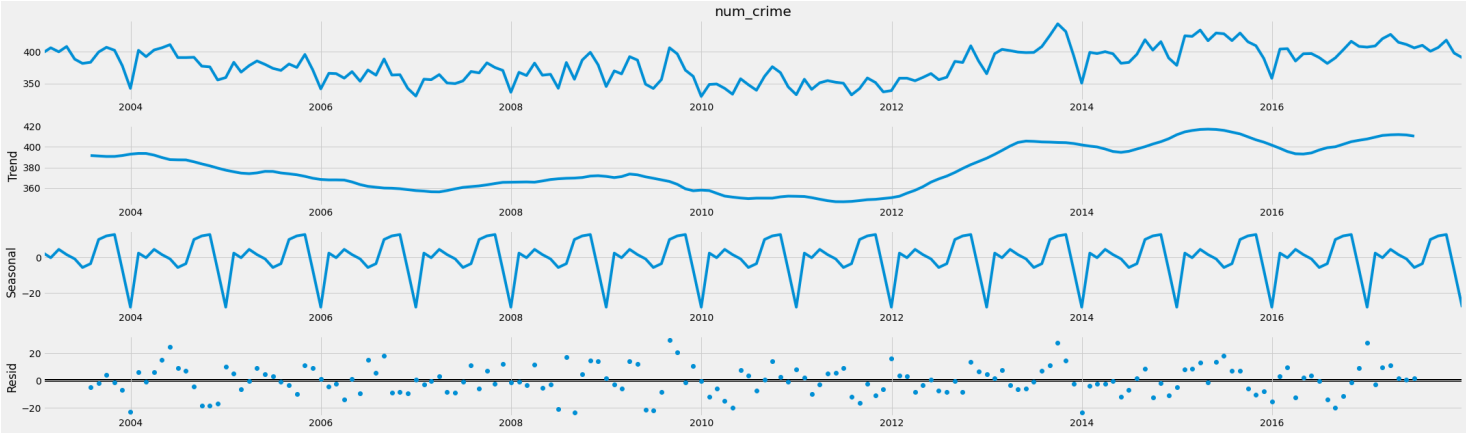
```
rollingmean = y.rolling(window=12).mean()
rollingstd = y.rolling(window=12).std()

orig = plt.plot(y, color='blue', label='Original')
mean = plt.plot(rollingmean , color='red', label='Rolling Mean')
#std = plt.plot(rollingstd, color='black', label='Rolling Std')
plt.rcParams["figure.figsize"] = (30,9)

plt.legend(loc='best')
plt.title('Rolling Mean ')
plt.show(block=False)
```




```
std = plt.plot(rollingstd, color='black', label='Rolling Std')
plt.title('Rolling Std')
plt.show(block=False)
```



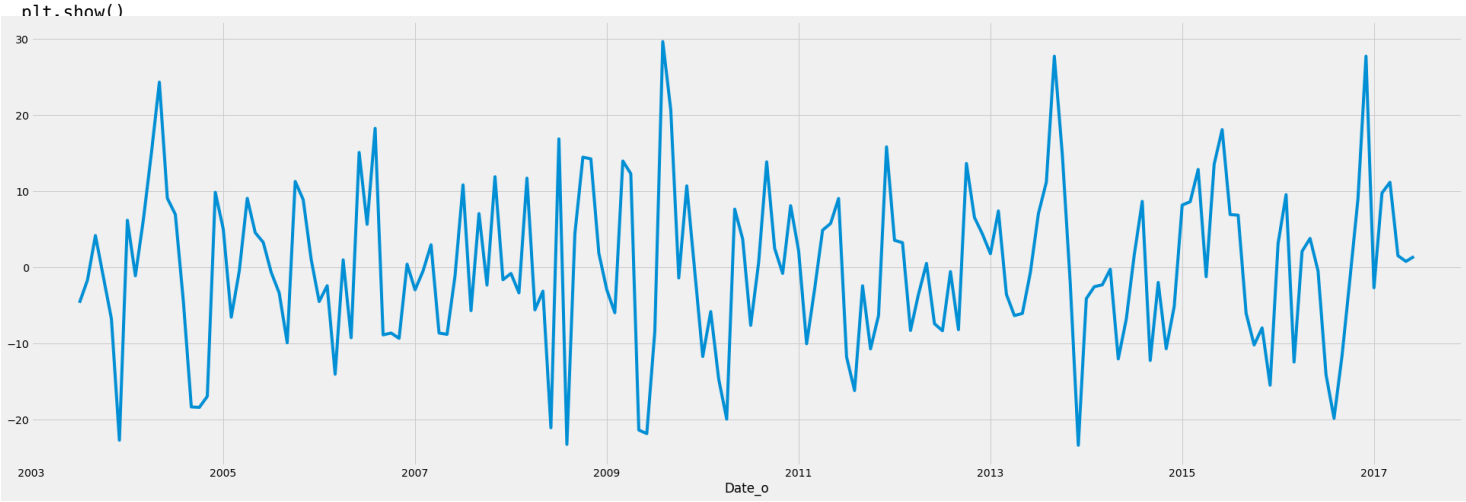

```
deco = sm.tsa.seasonal_decompose(y, model='additive')
fig = deco.plot()
plt.rcParams["figure.figsize"] = (30,10)

plt.show()
```



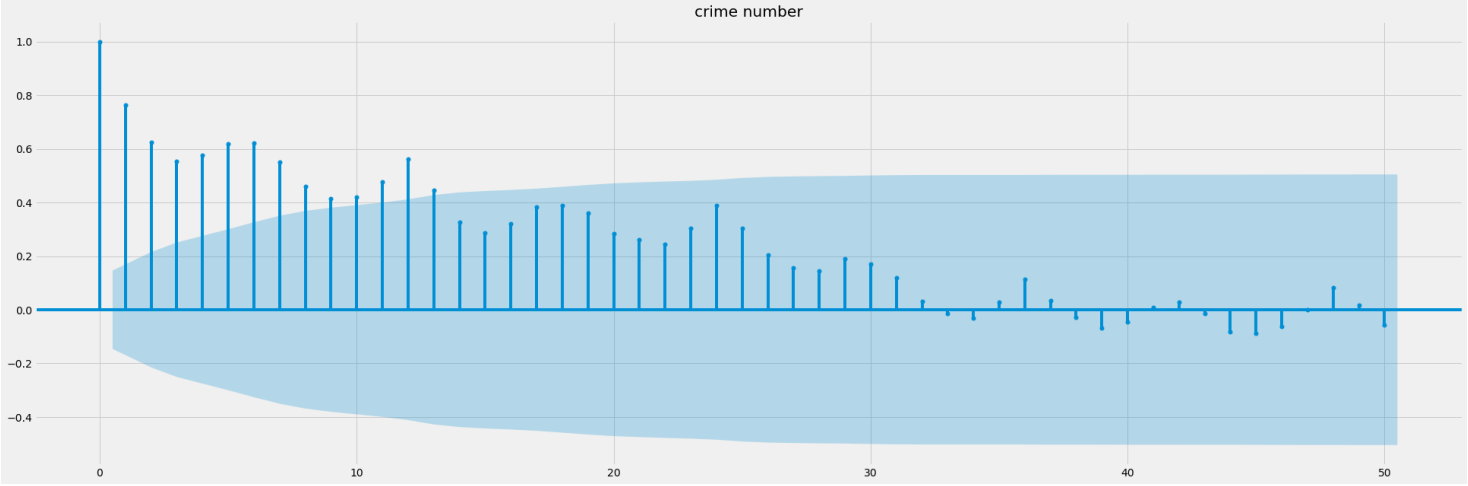

```
deco.resid.describe()
```

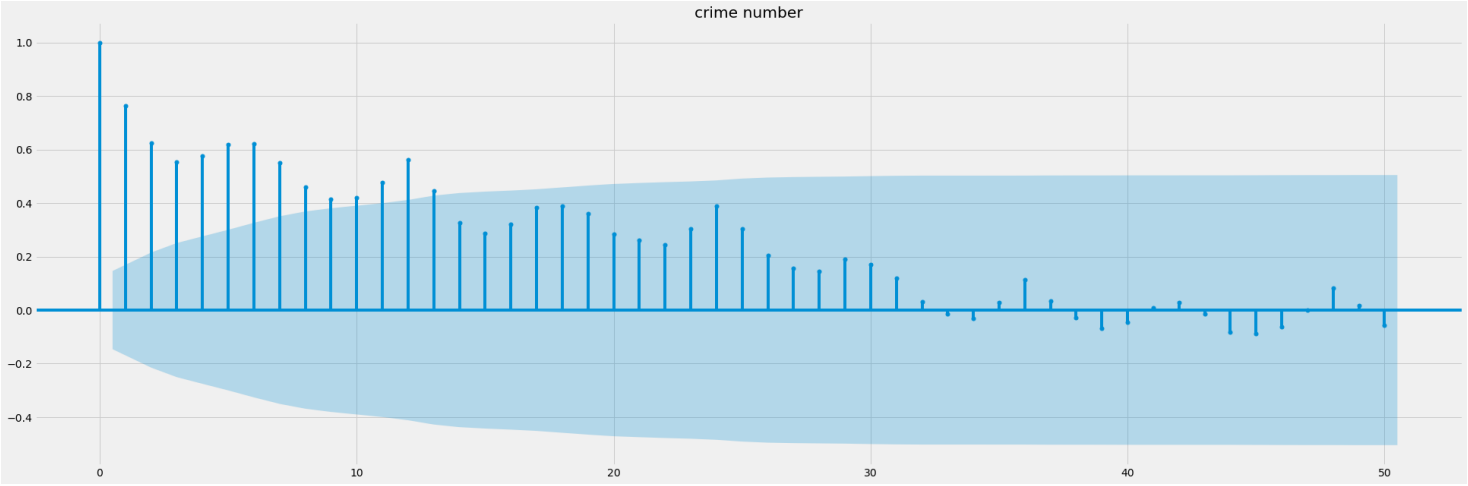
```
Out[57]: count    108.000000
mean       0.055538
std       10.458618
min      -23.333856
25%       -6.564380
50%       -0.580532
75%        7.068715
max       29.683243
Name: resid, dtype: float64
```

```
Residual = deco.resid
fig = Residual.plot()
plt.show()
```

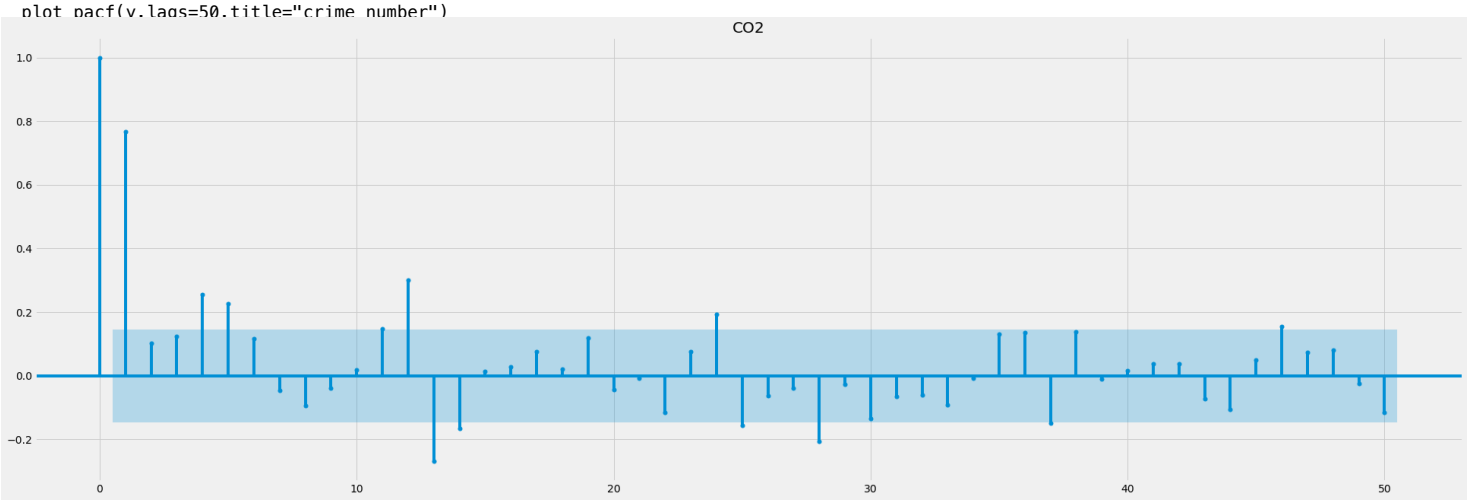


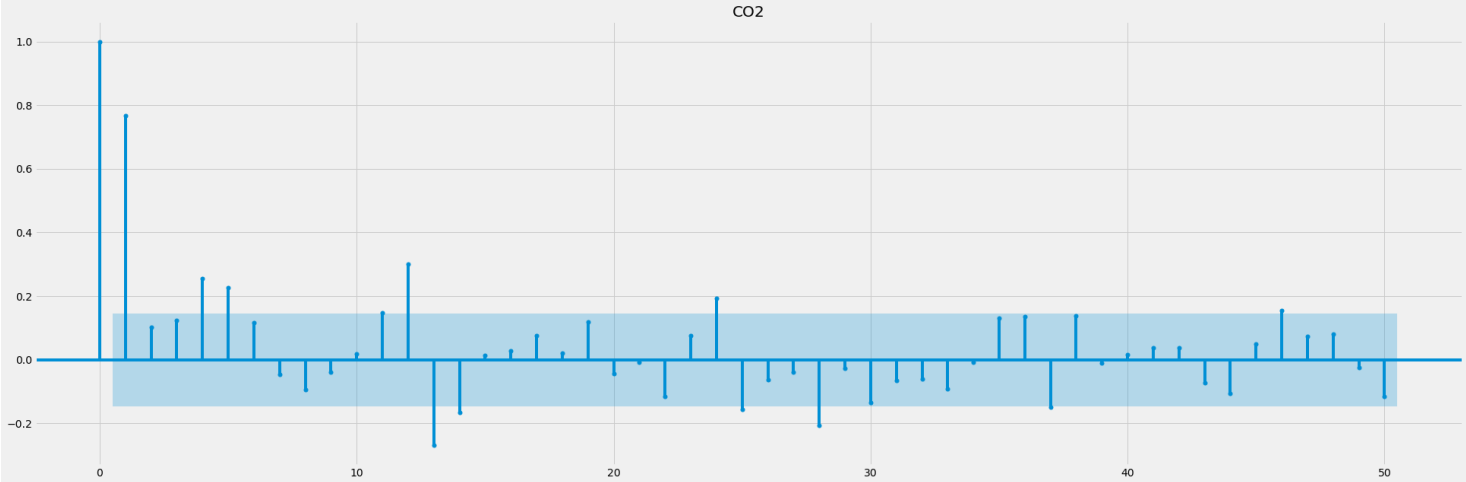

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(y,lags=50,title="crime number")
```



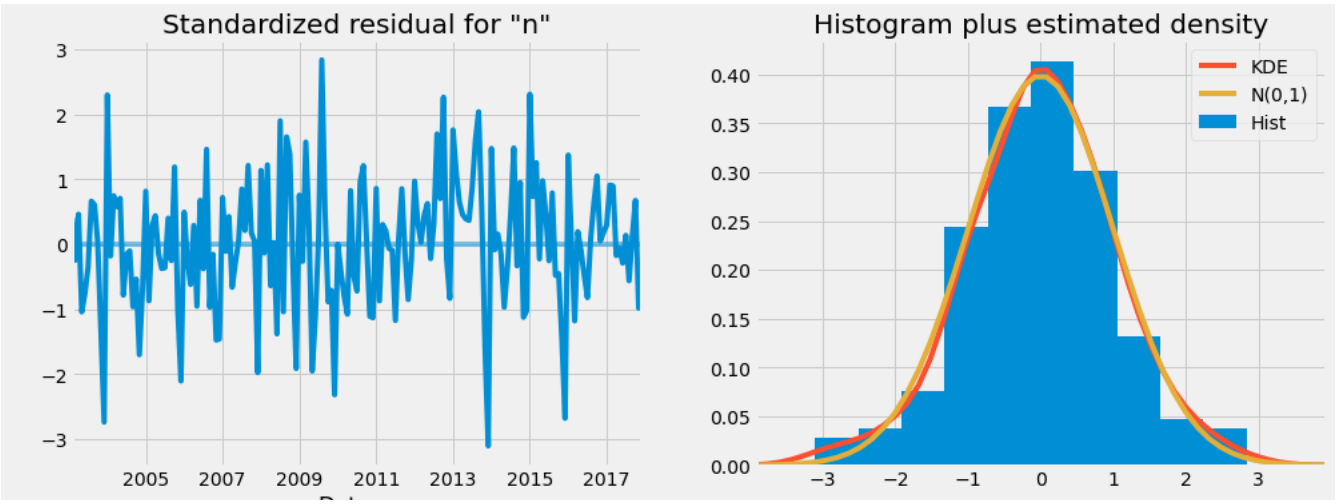



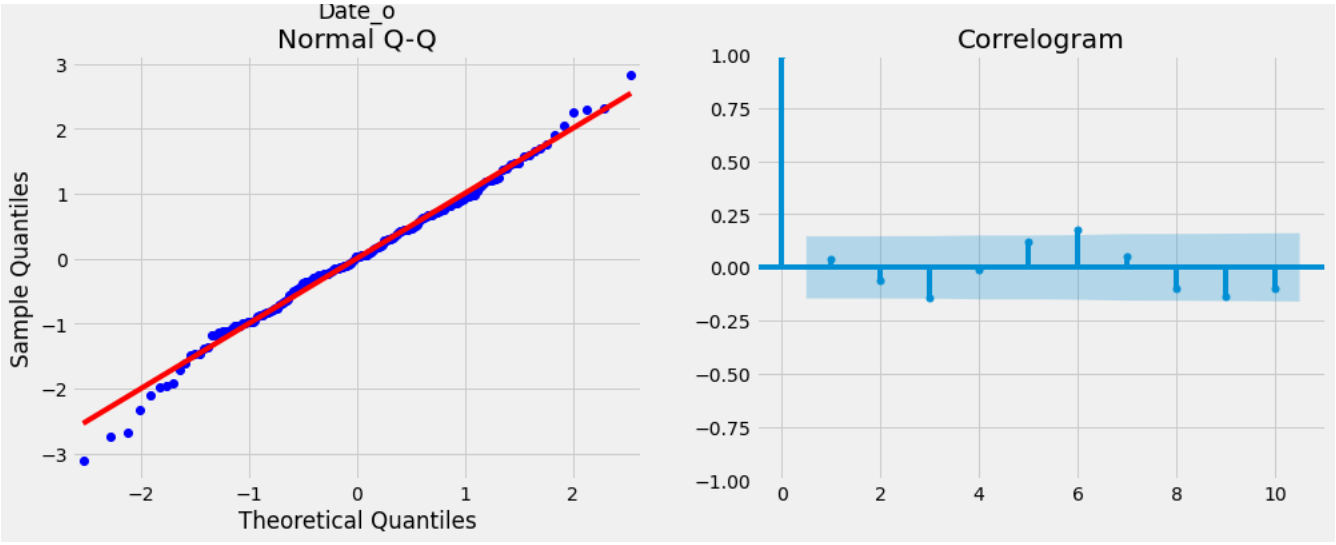

```
plot_pacf(y,lags=50,title="crime number")
```



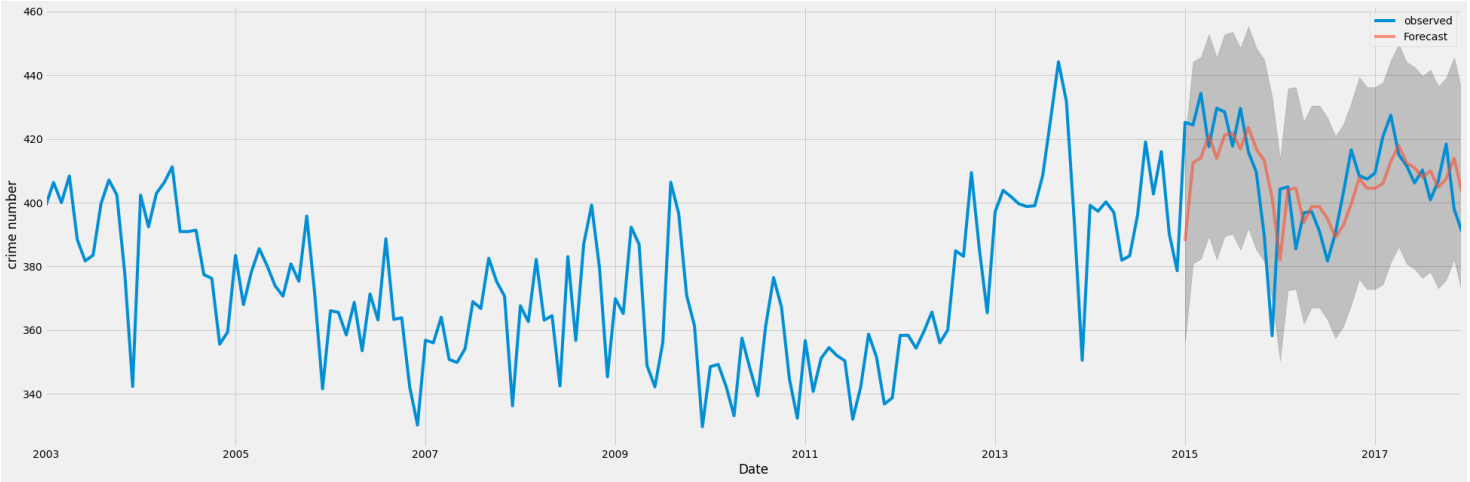



```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                )

TSresults = mod.fit()

#print(TSresults.summary().tables[1])

TSresults.plot_diagnostics(figsize=(15, 12))
plt.show()
```






```
pred = TSresults.get_prediction(start=pd.to_datetime('2015-01-31'), dynamic=False)
pred_ci = pred.conf_int()
#Returns the confidence interval of the fitted parameters.

ax = y['2003':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='Forecast', alpha=.6)

ax.fill_between(pred_ci.index, pred_ci.iloc[:, 0], pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel(' crime number')
plt.legend()

plt.show()
```

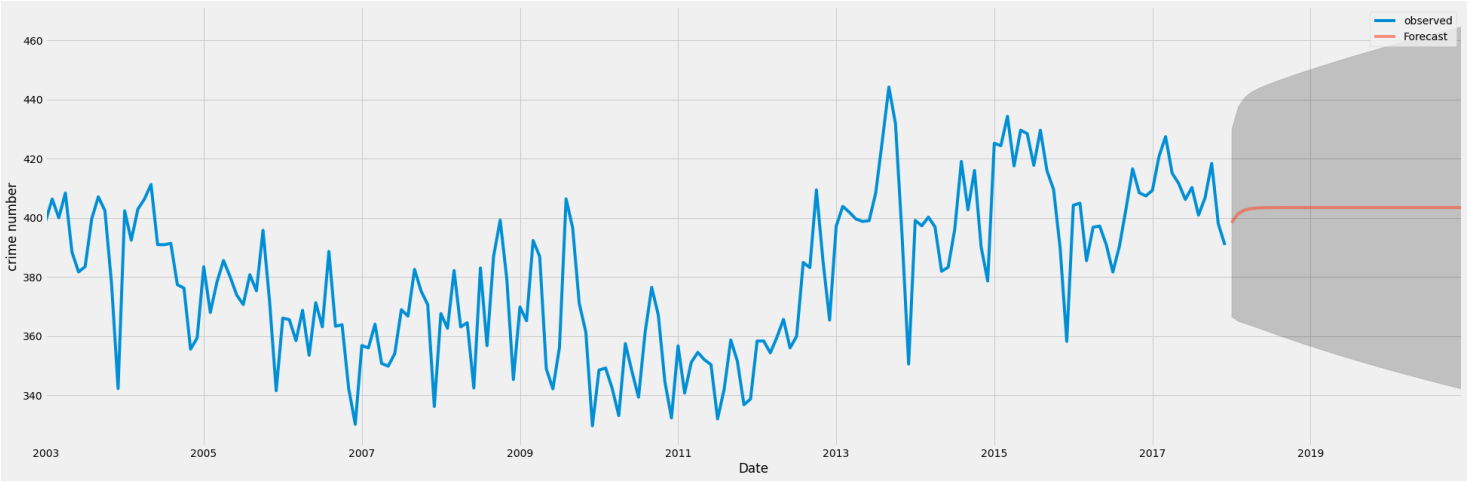



```
pred_ci
```

| Date_o | lower num_crime | upper num_crime |
|---|---|---|
| **2015-01-31** | 356.240993 | 419.564597 |
| **2015-02-28** | 380.851757 | 444.175362 |
| **2015-03-31** | 382.349508 | 445.673112 |
| **2015-04-30** | 389.511475 | 452.835080 |
| **2015-05-31** | 382.168684 | 445.492289 |
| **2015-06-30** | 389.501804 | 452.825409 |
| **2015-07-31** | 390.151661 | 453.475266 |
| **2015-08-31** | 385.177551 | 448.501155 |
| **2015-09-30** | 391.954827 | 455.278432 |
| **2015-10-31** | 385.205818 | 448.529423 |
| **2015-11-30** | 381.533324 | 444.856929 |
| **2015-12-31** | 369.848843 | 433.172448 |
| **2016-01-31** | 350.424651 | 413.748256 |
| **2016-02-29** | 372.469688 | 435.793293 |
| **2016-03-31** | 372.865868 | 436.189473 |
| **2016-04-30** | 362.046820 | 425.370425 |
| **2016-05-31** | 367.122296 | 430.445901 |
| **2016-06-30** | 367.021429 | 430.345034 |
| **2016-07-31** | 363.268956 | 426.592561 |
| **2016-08-31** | 357.483617 | 420.807222 |
| **2016-09-30** | 361.235430 | 424.559034 |
| **2016-10-31** | 367.914991 | 431.238596 |
| **2016-11-30** | 375.999662 | 439.323267 |
| **2016-12-31** | 372.855293 | 436.178898 |
| **2017-01-31** | 372.863520 | 436.187125 |
| **2017-02-28** | 374.354068 | 437.677673 |
| **2017-03-31** | 381.266982 | 444.590587 |
| **2017-04-30** | 386.225750 | 449.549354 |
| **2017-05-31** | 380.791553 | 444.115158 |
| **2017-06-30** | 379.228051 | 442.551656 |
| **2017-07-31** | 376.337474 | 439.661079 |
| **2017-08-31** | 378.311159 | 441.634764 |
| **2017-09-30** | 373.133945 | 436.457550 |
| **2017-10-31** | 375.830010 | 439.153615 |
| **2017-11-30** | 382.206272 | 445.529877 |
| **2017-12-31** | 371.499332 | 434.822937 |

```
y_forecasted = pred.predicted_mean
y_truth = y['2015-01-31':]

# Compute the mean square error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
```

```
The Mean Squared Error of our forecasts is 210.32
```


observed
Forecast
460
440
420
400
380
360
340
crime number
2003
2005
2007
2009
2011
2013
2015
2017
2019
Date