In [100…

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pmdarima as pm
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
```

Read and prepare the data

In [103…

```python
df = pd.read_csv("C:\\Users\\user\\OneDrive\\Desktop\\Project\\city_day.csv")
delhi = df[df['City'] == 'Delhi']
```

In [105…

```python
delhi['Date'] = pd.to_datetime(delhi['Date'])
delhi.set_index('Date', inplace = True)
```

In [107…

```python
delhi.head()
```

Out[107…

Date	City	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Tol
2015-01-01	Delhi	313.22	607.98	69.16	36.39	110.59	33.85	15.20	9.25	41.68	14.36	
2015-01-02	Delhi	186.18	269.55	62.09	32.87	88.14	31.83	9.54	6.65	29.97	10.55	
2015-01-03	Delhi	87.18	131.90	25.73	30.31	47.95	69.55	10.61	2.65	19.71	3.91	
2015-01-04	Delhi	151.84	241.84	25.01	36.91	48.62	130.36	11.54	4.63	25.36	4.26	
2015-01-05	Delhi	146.60	219.13	14.01	34.92	38.25	122.88	9.20	3.33	23.20	2.80	

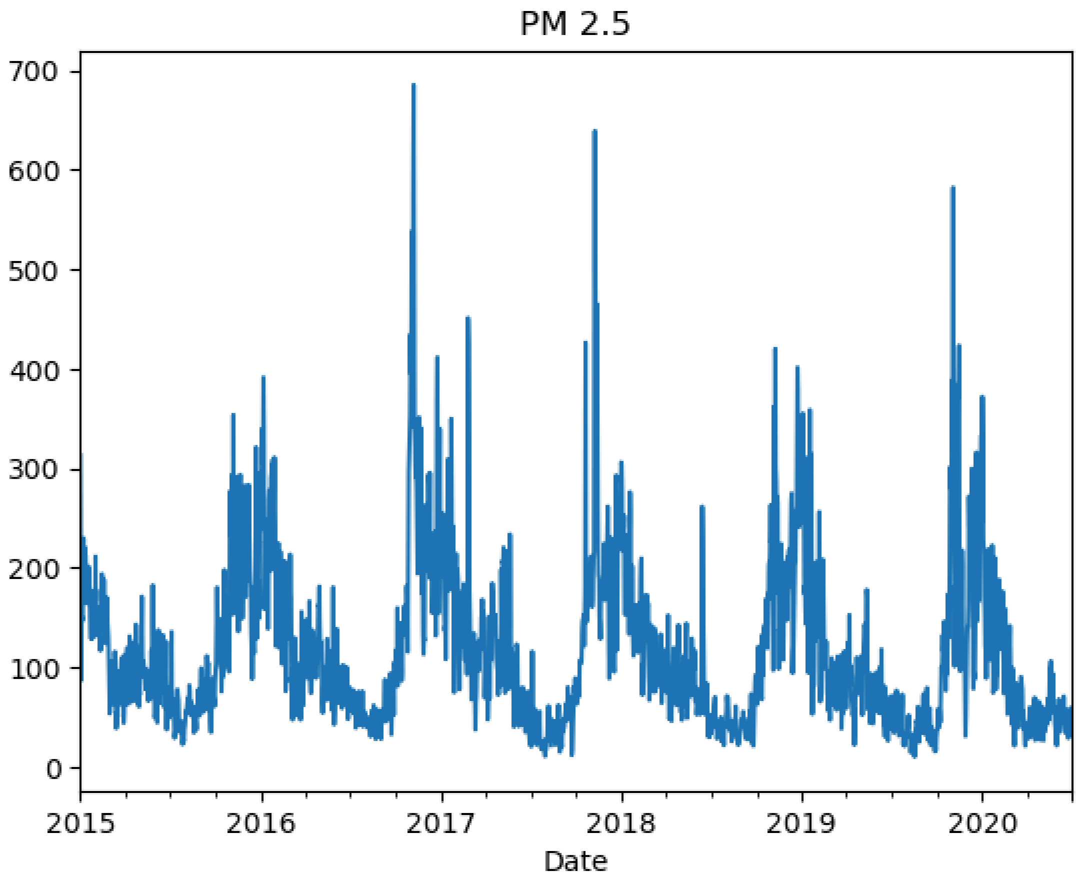
PM 2.5

In [110…

```python
delhi['PM2.5'].plot(title = 'PM 2.5')
```

Out[110…

```
<Axes: title={'center': 'PM 2.5'}, xlabel='Date'>
```



Converting the daily data into a monthly data for easier analysis

In [113…

```
pm25 = delhi['PM2.5'].resample(rule = 'MS').mean()
pm25
```

Out[113…

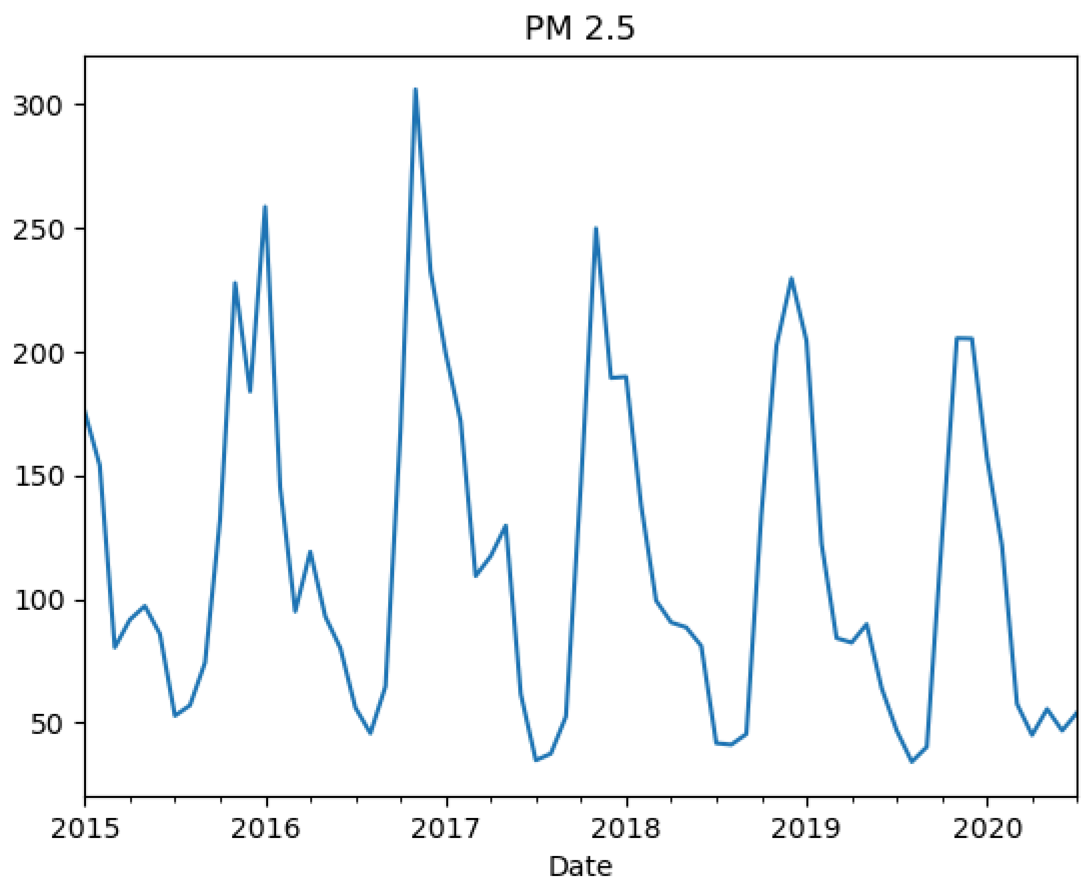
```
Date
2015-01-01    175.690645
2015-02-01    153.920357
2015-03-01     80.338065
2015-04-01     91.562333
2015-05-01     97.109355
                 ...
2020-03-01     57.506452
2020-04-01     44.940000
2020-05-01     55.448710
2020-06-01     46.694667
2020-07-01     54.010000
Freq: MS, Name: PM2.5, Length: 67, dtype: float64
```

In [115…

```
pm25.plot(title = 'PM 2.5')
```

Out[115…

```
<Axes: title={'center': 'PM 2.5'}, xlabel='Date'>
```



The data clearly doesn't show any trend but indeed has a visible seasonality. So, we use the **adfuller test** to check for stationarity of the series.

Check for stationarity

In [119…

```
pm25_result = adfuller(pm25)
print('Test Statistic: {}'.format(pm25_result[0]))
print('p value: {}'.format(pm25_result[1]))
```

```
Test Statistic: 0.1051175256024009
p value: 0.966408759417222
```

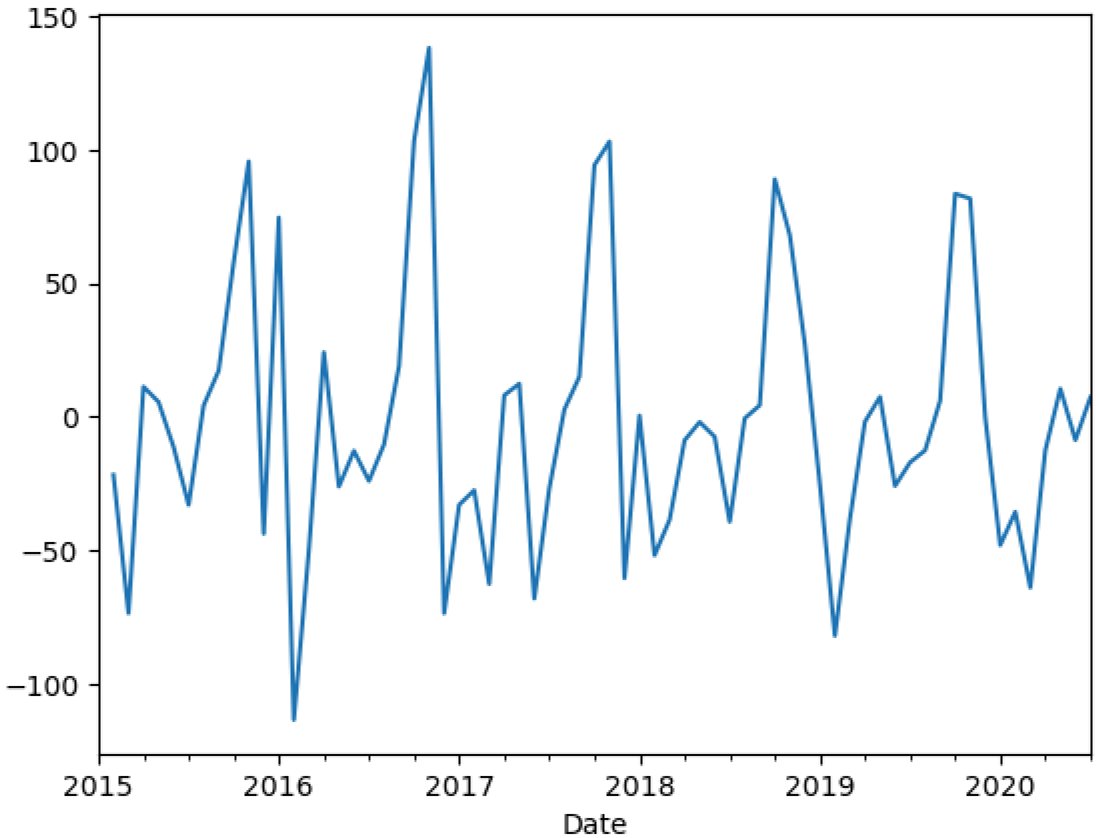
The p value is 0.966 > 0.05, which indicates that there is very weak evidence to reject the null hypothesis and the series is not stationary. So we compute the first order differences of the data and check again for stationarity.

In [122…

```
pm25_diff1 = pm25 - pm25.shift(1)
pm25_diff1.plot()
```

Out[122…

```
<Axes: xlabel='Date'>
```



In [123…

```
pm25_result2 = adfuller(pm25_diff1.dropna())
print('Test Statistic: {}'.format(pm25_result2[0]))
print('p value: {}'.format(pm25_result2[1]))
```

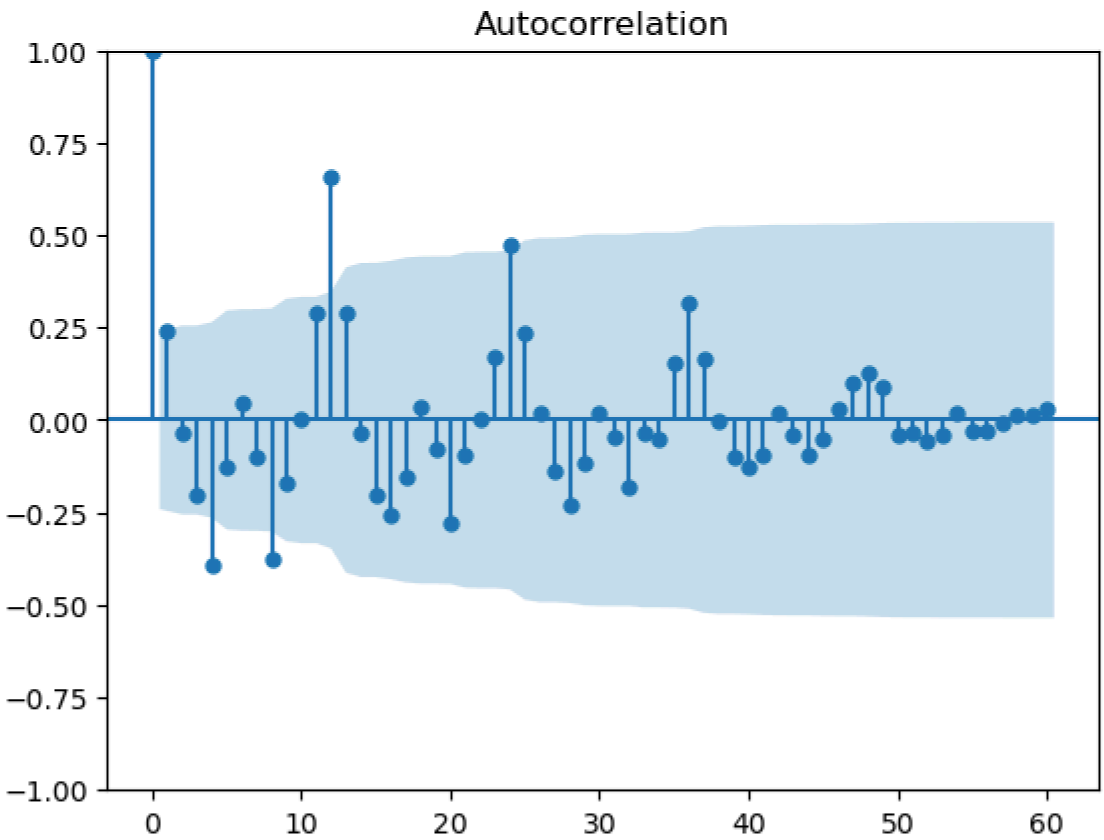
```
Test Statistic: -7.71642271845219
p value: 1.2248286849539635e-11
```

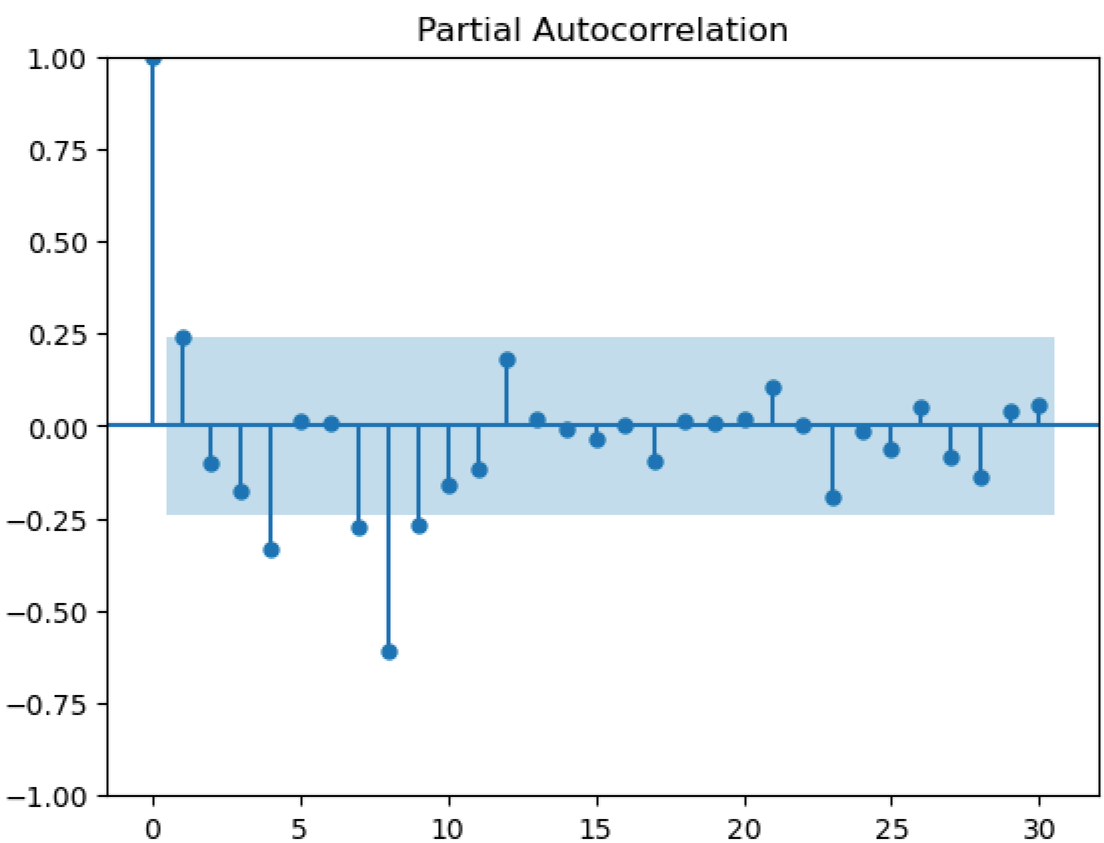
Here the p value is very very smaller than 0.05 indicating a very strong evidence against the null hypothesis. So we conclude that the first order differences are stationary.

ACF and PACF plot of seasonal data

In [128…

```
pm25_diff1_acf = plot_acf(pm25_diff1.dropna(), lags = 60)
pm25_diff1_pacf = plot_pacf(pm25_diff1.dropna(), lags = 30)
```



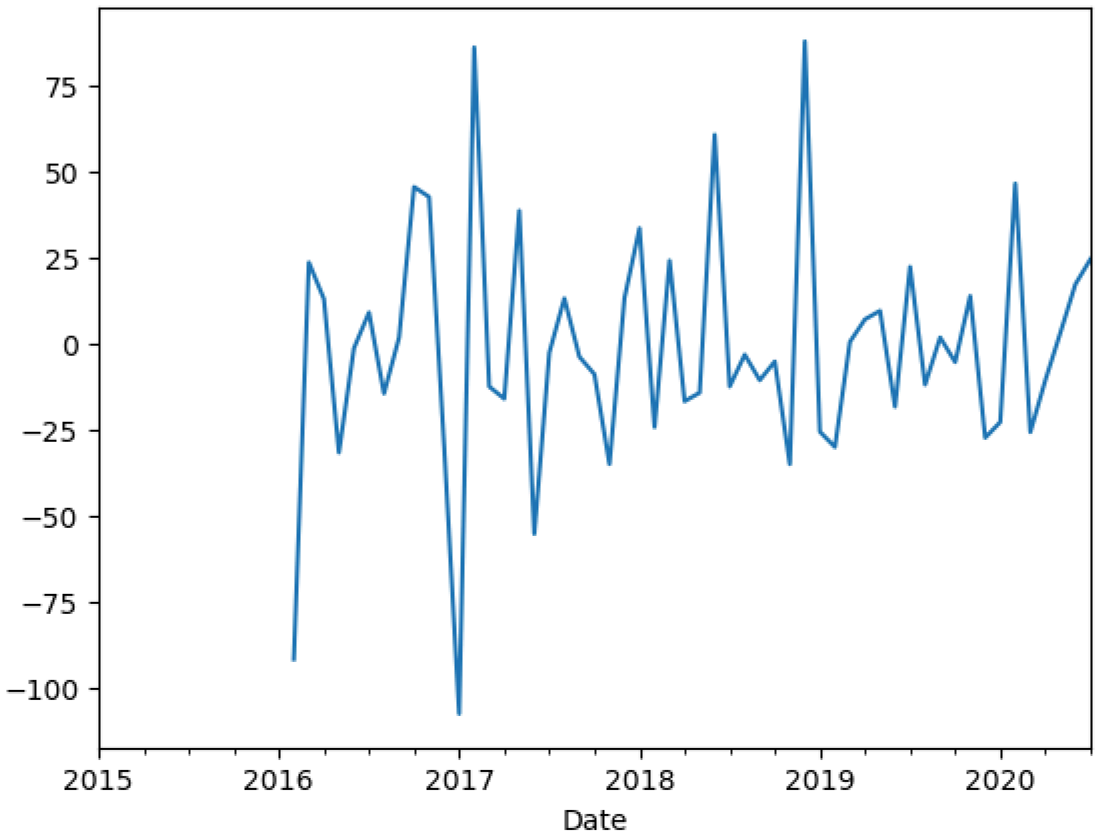


Seasonal difference

Now we take a difference of 12 steps to remove the seasonality from the the data

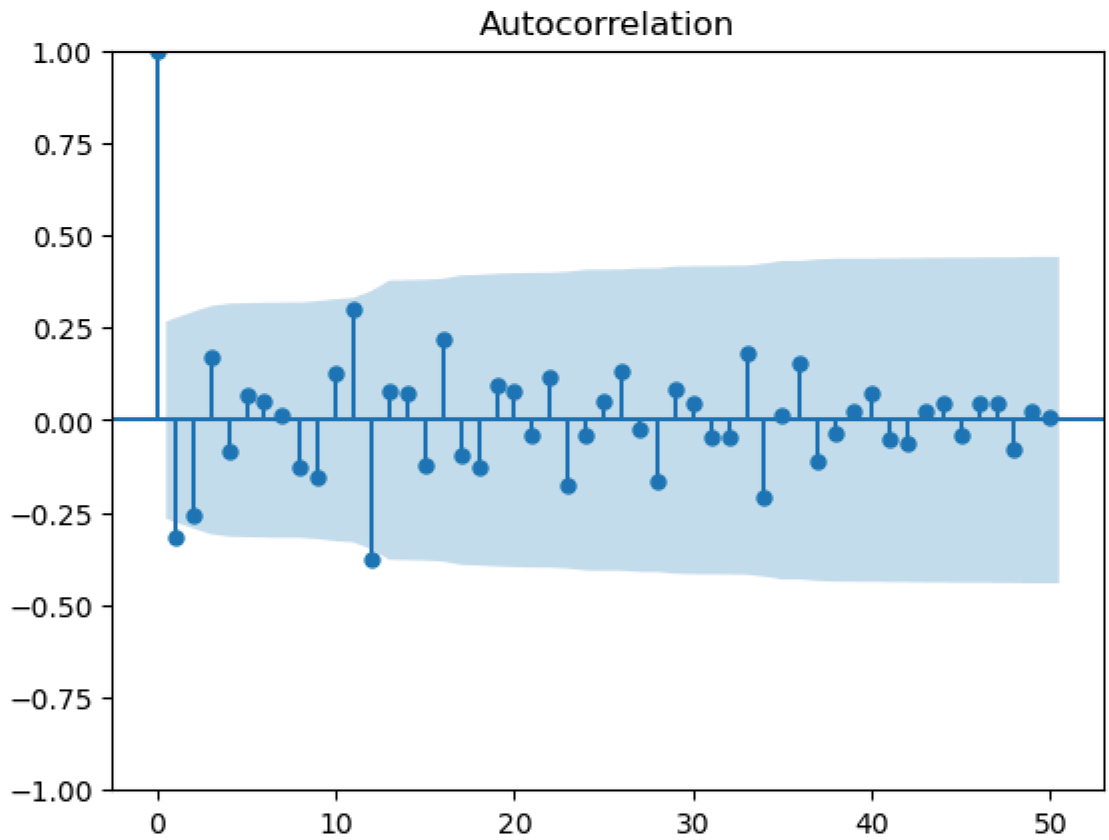
In [132…
```
pm25_sdiff = pm25_diff1 - pm25_diff1.shift(12)
pm25_sdiff.plot()
```

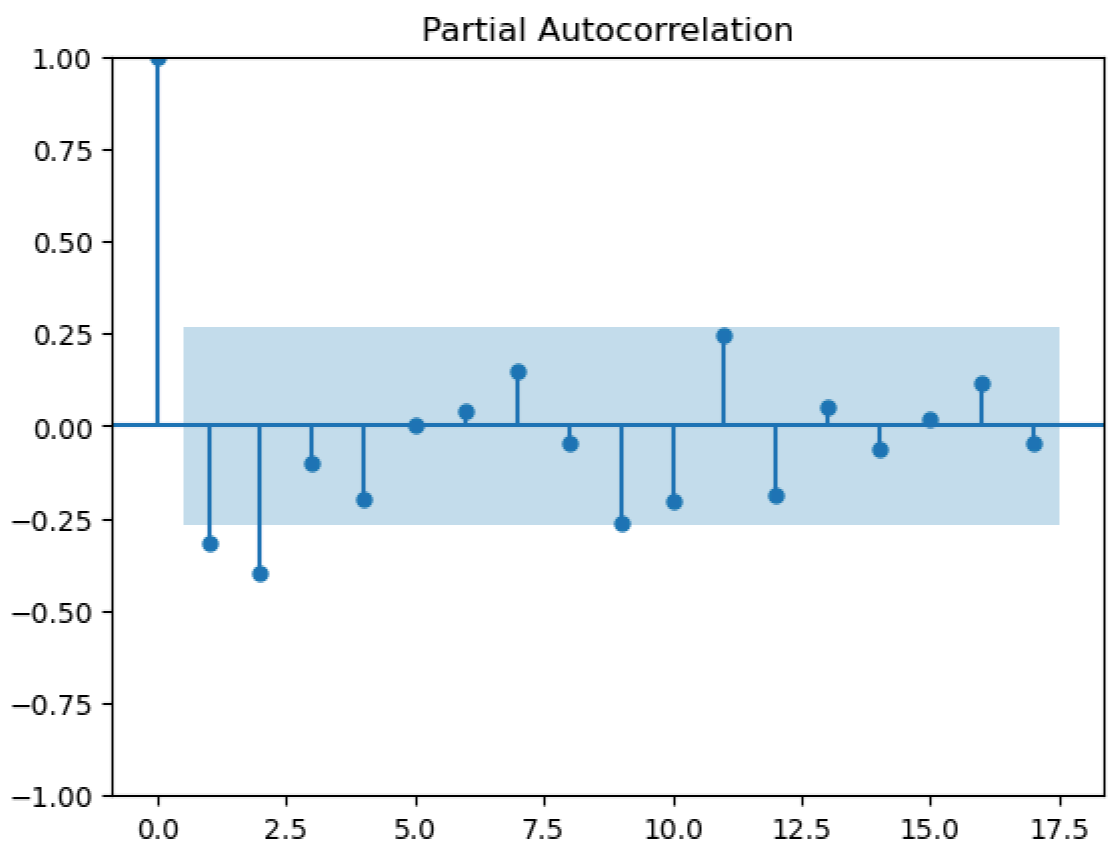
Out[132…
```
<Axes: xlabel='Date'>
```



ACF and PACF plot after seasonal differencing

In [135…
```
pm25_sdiff_acf = plot_acf(pm25_sdiff.dropna(), lags = 50)
pm25_sdiff_pacf = plot_pacf(pm25_sdiff.dropna(), lags = 17)
```





From the ACF and PACF plot we should take values of p,q and P,Q as -

p = 1,2

q = 1,2

P = 1

Q = 1,2

Data Splitting

In [139…

```
train_pm25 = pm25[pm25.index < '2020-01-01']
test_pm25 = pm25[pm25.index >= '2020-01-01']
```

Model Fitting

In [142…

```
orders = [(1,1,1), (1,1,2), (2,1,1), (2,1,2)]
seasonal_orders = [(1,1,1,12), (1,1,2,12)]
```

In [144…

```
for order in orders:
    for seasonal_order in seasonal_orders:
        model = SARIMAX(train_pm25, order = order, seasonal_order = seasonal_order)
        print('AIC for Model {}{}:'.format(order, seasonal_order), model.aic)
```

```
AIC for Model (1, 1, 1)(1, 1, 1, 12): 443.92278033791234
AIC for Model (1, 1, 1)(1, 1, 2, 12): 445.65112619600853
AIC for Model (1, 1, 2)(1, 1, 1, 12): 445.323018694011
AIC for Model (1, 1, 2)(1, 1, 2, 12): 447.3117321890521
AIC for Model (2, 1, 1)(1, 1, 1, 12): 445.26622534220934
AIC for Model (2, 1, 1)(1, 1, 2, 12): 447.28166878063604
AIC for Model (2, 1, 2)(1, 1, 1, 12): 446.90842765267826
AIC for Model (2, 1, 2)(1, 1, 2, 12): 448.89604149386867
```

after checking the aic we conclude that it increases with the increase of complexity in the model. So we choose the simpler model that is SARIMAX(1,1,1)(1,1,1,12)

In [147…

```
pm25_model = SARIMAX(train_pm25, order = (1,1,1), seasonal_order = (1,1,1,12)).fit(
print(pm25_model.summary())
```

```
                                     SARIMAX Results
==========================================================================================
======
Dep. Variable:                                PM2.5   No. Observations:
60
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                    -2
16.961
Date:                              Mon, 17 Mar 2025   AIC                                4
43.923
Time:                                      01:18:46   BIC                                4
53.174
Sample:                                  01-01-2015   HQIC                               4
47.404
                                       - 12-01-2019
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0168      0.194     -0.087      0.931      -0.397       0.364
ma.L1         -0.8076      0.177     -4.560      0.000      -1.155      -0.460
ar.S.L12      -0.0419      0.343     -0.122      0.903      -0.714       0.630
ma.S.L12      -0.9990    255.998     -0.004      0.997    -502.746     500.748
sigma2       382.2499   9.78e+04      0.004      0.997   -1.91e+05    1.92e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.21   Jarque-Bera (JB):                 2.94
Prob(Q):                              0.65   Prob(JB):                         0.23
Heteroskedasticity (H):               0.60   Skew:                             0.31
Prob(H) (two-sided):                  0.31   Kurtosis:                         4.06
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
```

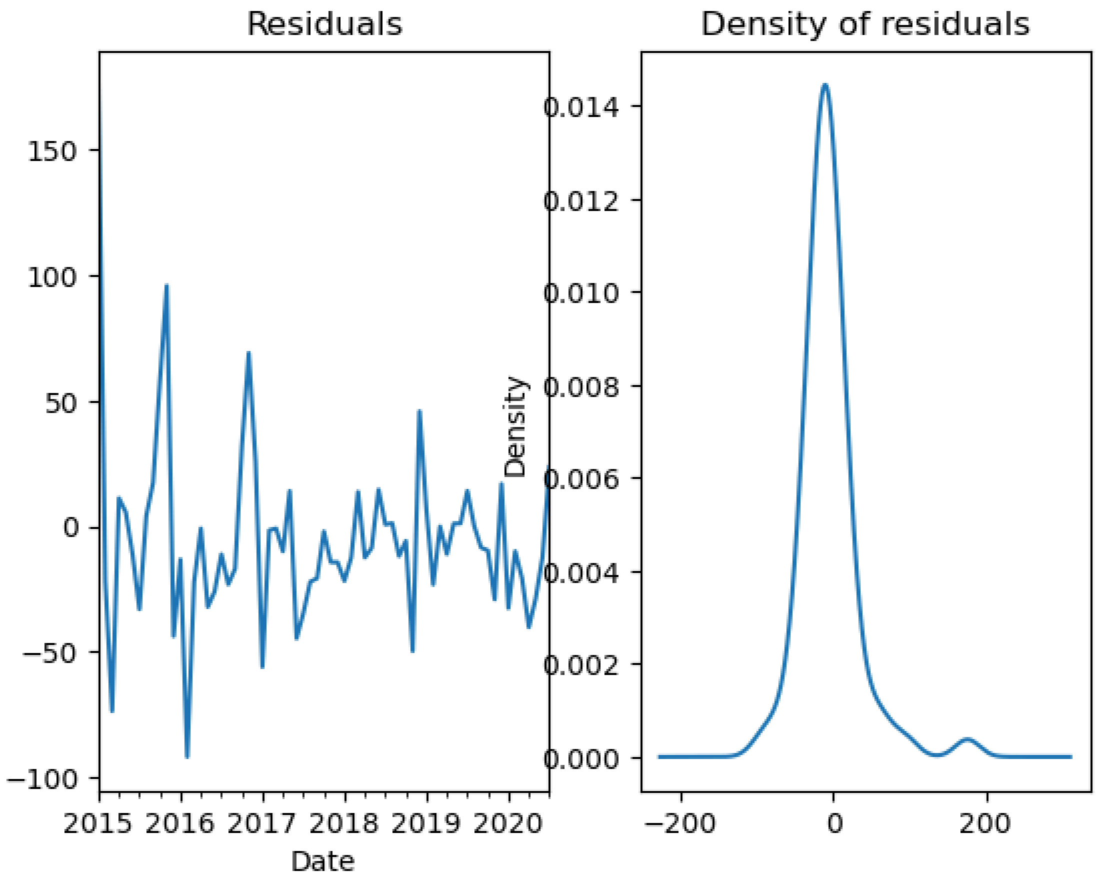
Residuals

In [150…

```python
pm25_res = pm25 - pm25_model.predict(start = pm25.index[0], end = pm25.index[-1])
fig, ax = plt.subplots(1,2)
pm25_res.plot(title = 'Residuals', ax = ax[0])
pm25_res.plot(title = 'Density of residuals', kind = 'kde', ax = ax[1])
```

Out[150…

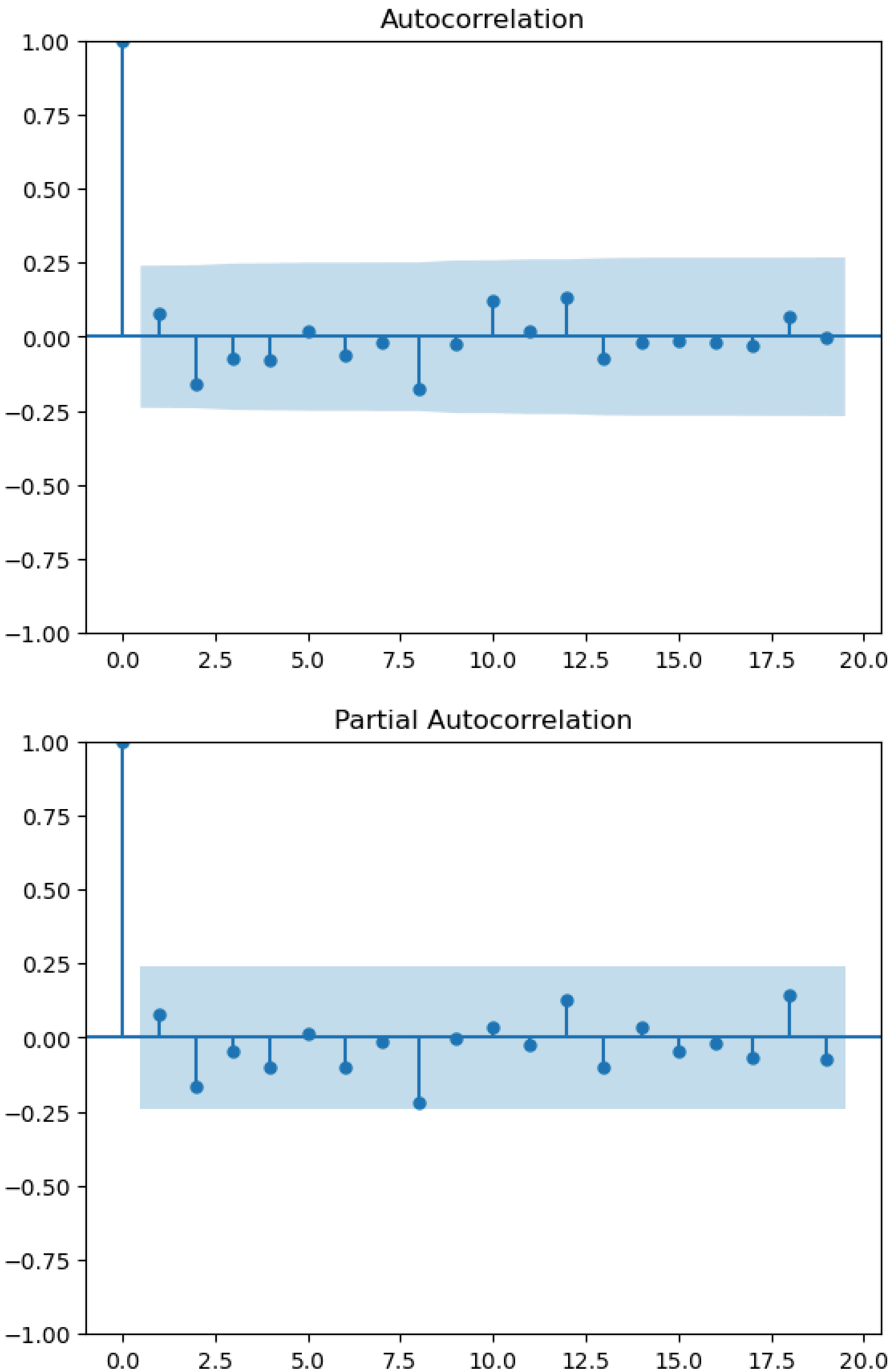
```
<Axes: title={'center': 'Density of residuals'}, ylabel='Density'>
```



ACF and PACF plot of residuals

In [153…

```
pm25_res_acf = plot_acf(pm25_res)
pm25_res_pacf = plot_pacf(pm25_res)
```


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

The density and acf/pacf plot of the residuals shows that the residuals are random and normaly distributed around 0, which suggests our model has captured most of the components and patterns from the series.

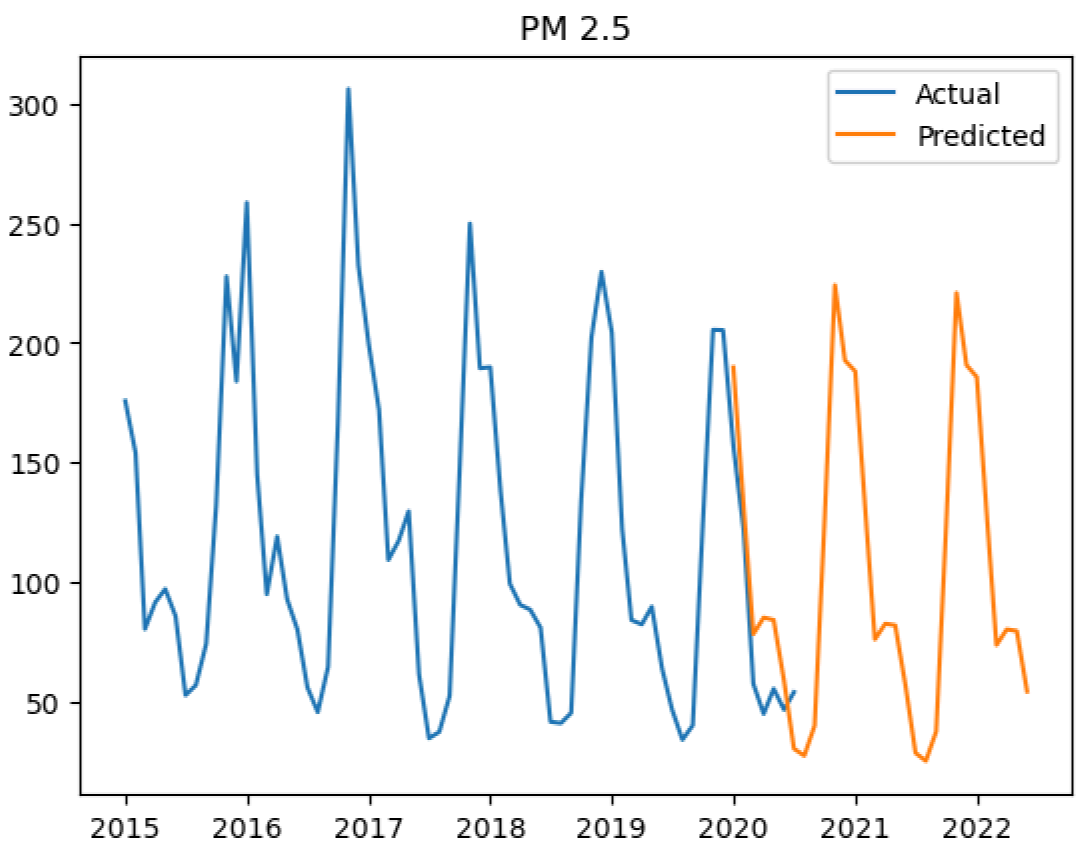
Forecasting and Comparing

In [157…

```
plt.plot(pm25, label = 'Actual')
plt.plot(pm25_model.forecast(steps = 30), label = 'Predicted')
plt.title('PM 2.5')
plt.legend()
```

Out[157…

```
<matplotlib.legend.Legend at 0x1f0416f4fe0>
```



In [ ]: