PORTFOLIO

By

Enock HABIMANA

Professional Background

I am a motivated Data Science student at World Quant University, set to graduate in May 2024. I am deeply passionate about employing data-driven insights to address complex challenges and am actively seeking internship opportunities to enhance my skills in Python, SQL, data wrangling, machine learning, and data visualization.

Education Background

World Quant University: Pursuing a degree in Data Science, anticipated completion in May 2024.

Bootcamp Training: Completed a two-month intensive bootcamp in Google sheet, Descriptive Statistics, and Tableau. At EntryLevel platform

Self-Taught Expertise: Gained foundational knowledge in Python and SQL through dedicated self-study at W3Schools.

Technical Skills

- **Programming Languages**: Python, SQL
- **Python Libraries**: NumPy, Matplotlib, Seaborn, Pandas
- **Machine Learning**: Proficient in techniques such as linear regression, ridge regression, and ensemble methods like random forest and gradient boosting classifiers. Skilled in time series analysis with ARIMA modeling.
- **Data Management**: Experienced in SQL database querying, managing data with MongoDB, and utilizing Jupyter Notebook for project development.
- **Data Visualization Tools**: Proficient with Excel, Google Sheets, and Tableau for insightful data presentations. Demonstrated the ability to derive meaningful insights from data and effectively communicate findings to non-technical audiences.

KEY PROJECTS

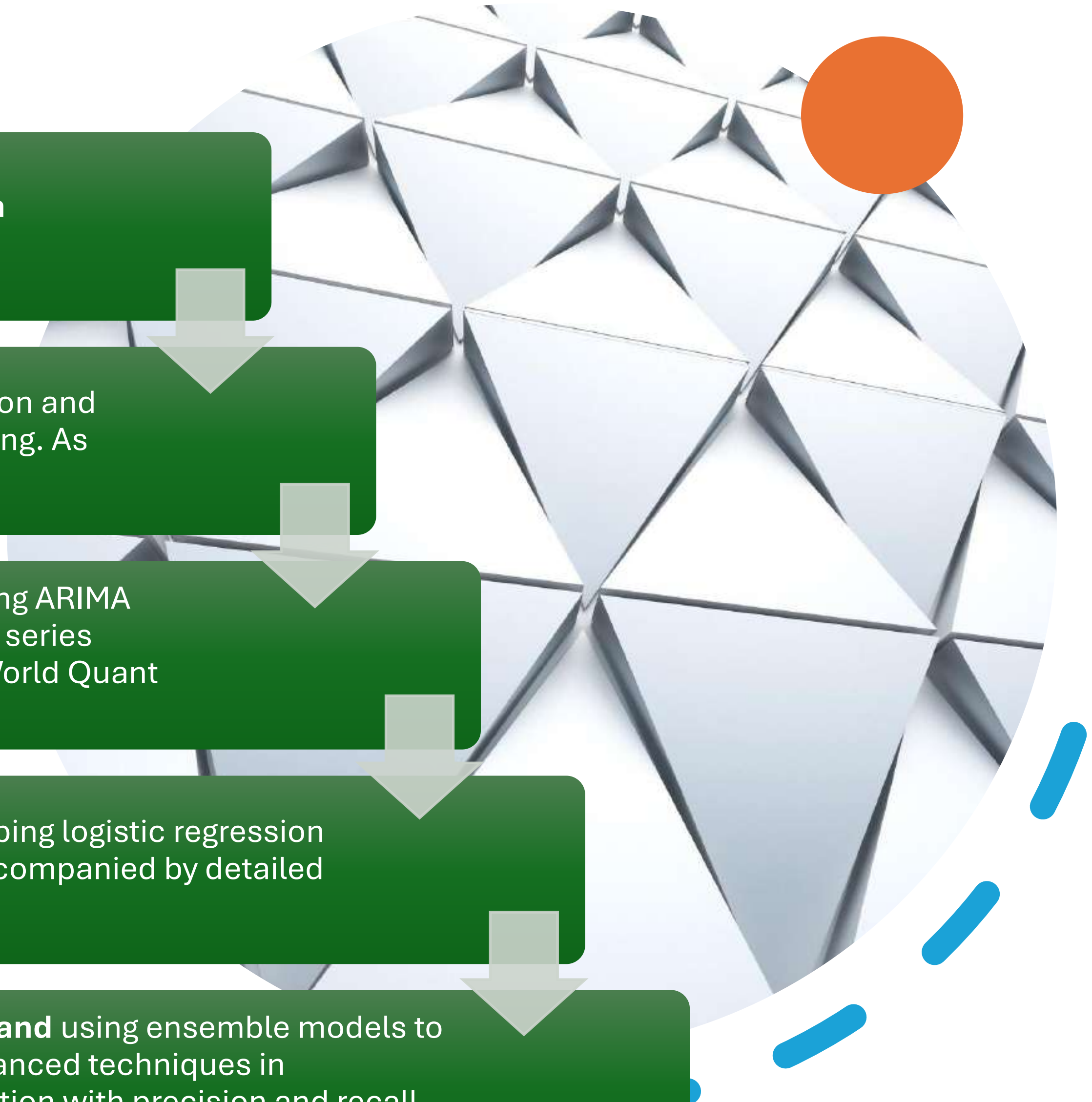
1. Independently I have been able to build a model to predict air quality in Abuja/ Nigeria

2. Predicting Apartment Prices in Latin America using linear regression and machine learning, with extensive data wrangling and feature engineering. As part of curriculum in data science at World Quant university.

3.Air Quality Analysis in Nairobi, Dar es Salaam, Lagos, utilizing ARIMA models and MongoDB for data manipulation, focusing on time series predictive analytics. As part of curriculum in data science at World Quant university.

4. Earthquake Damage Assessment in Nepal, developing logistic regression and decision tree models to classify damage levels, accompanied by detailed exploratory data analysis.

5.Bankruptcy Prediction for companies in Poland using ensemble models to handle imbalanced data sets, enhanced by advanced techniques in hyperparameter tuning and performance evaluation with precision and recall metrics. As part of curriculum in data science at World Quant university.



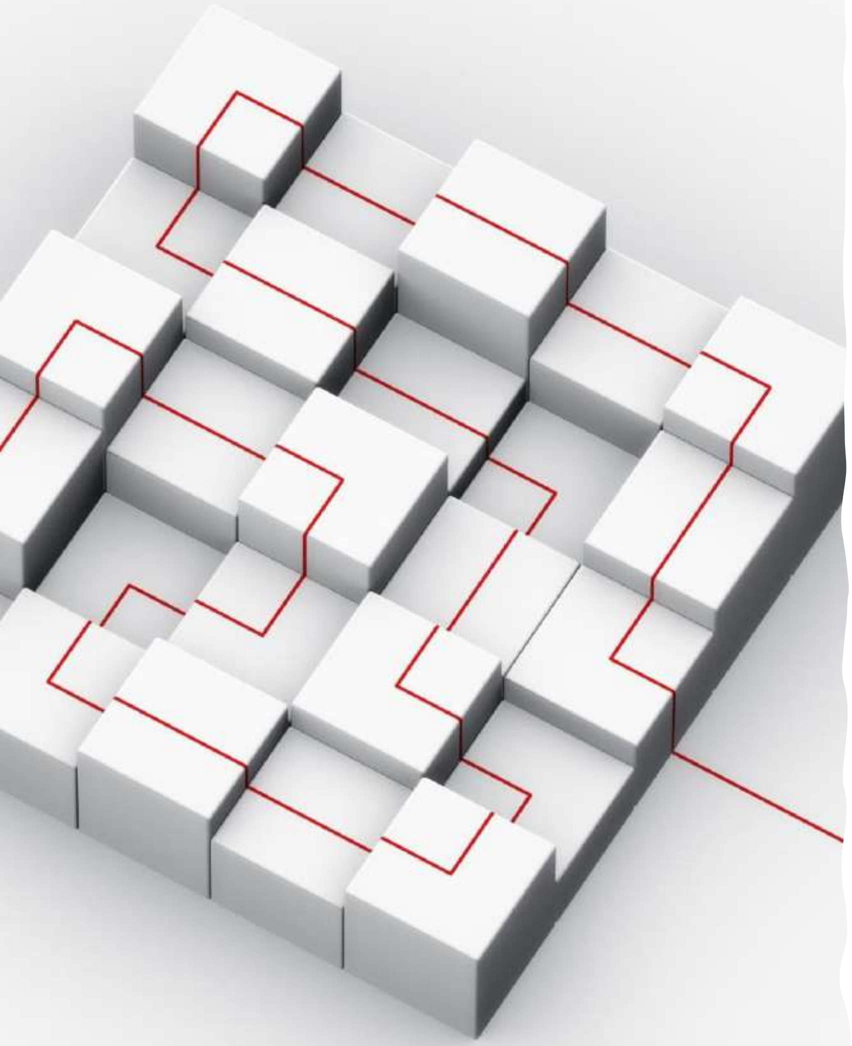
PROJECT

Predictive Model of Air Quality in Abuja Using PM2.5 Sensors

Table of content

Introduction

We're embarking on an insightful project that constructs and evaluates two predictive models: one using **Linear Regression** and another using **Autoregression** (AR). Our objective is to harness these models to accurately forecast air quality, comparing their effectiveness to determine the optimal approach for real-time environmental monitoring.



Project description

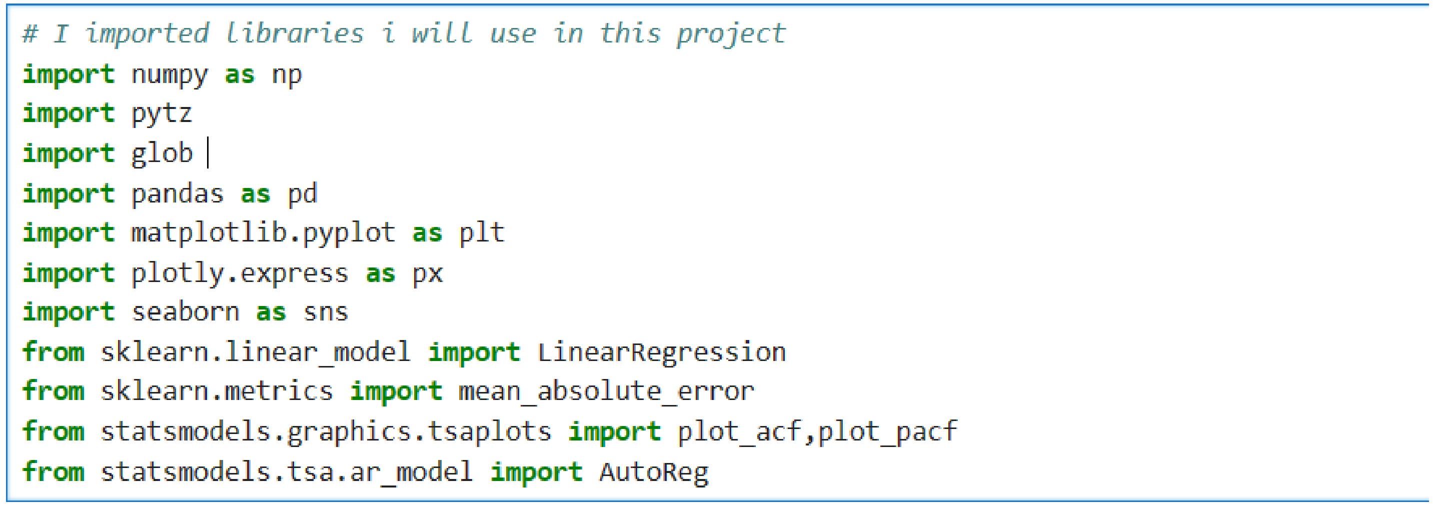
In this project, I focused on predicting air quality in Abuja using data from PM2.5 sensors, employing two statistical methods: Linear Regression and Autoregressive (AR) Modeling. The goal was to determine which model more accurately forecasts PM2.5 levels. Using Python for data processing and model building, I analyzed sensor data stored in CSV format. This analysis not only compared the efficacy of each modeling approach but also aimed to enhance air quality monitoring and public health interventions in urban settings. **P2 are reading from PM2.5 sensors**

Data collection, Data Cleaning and Preparation

Data was extracted from

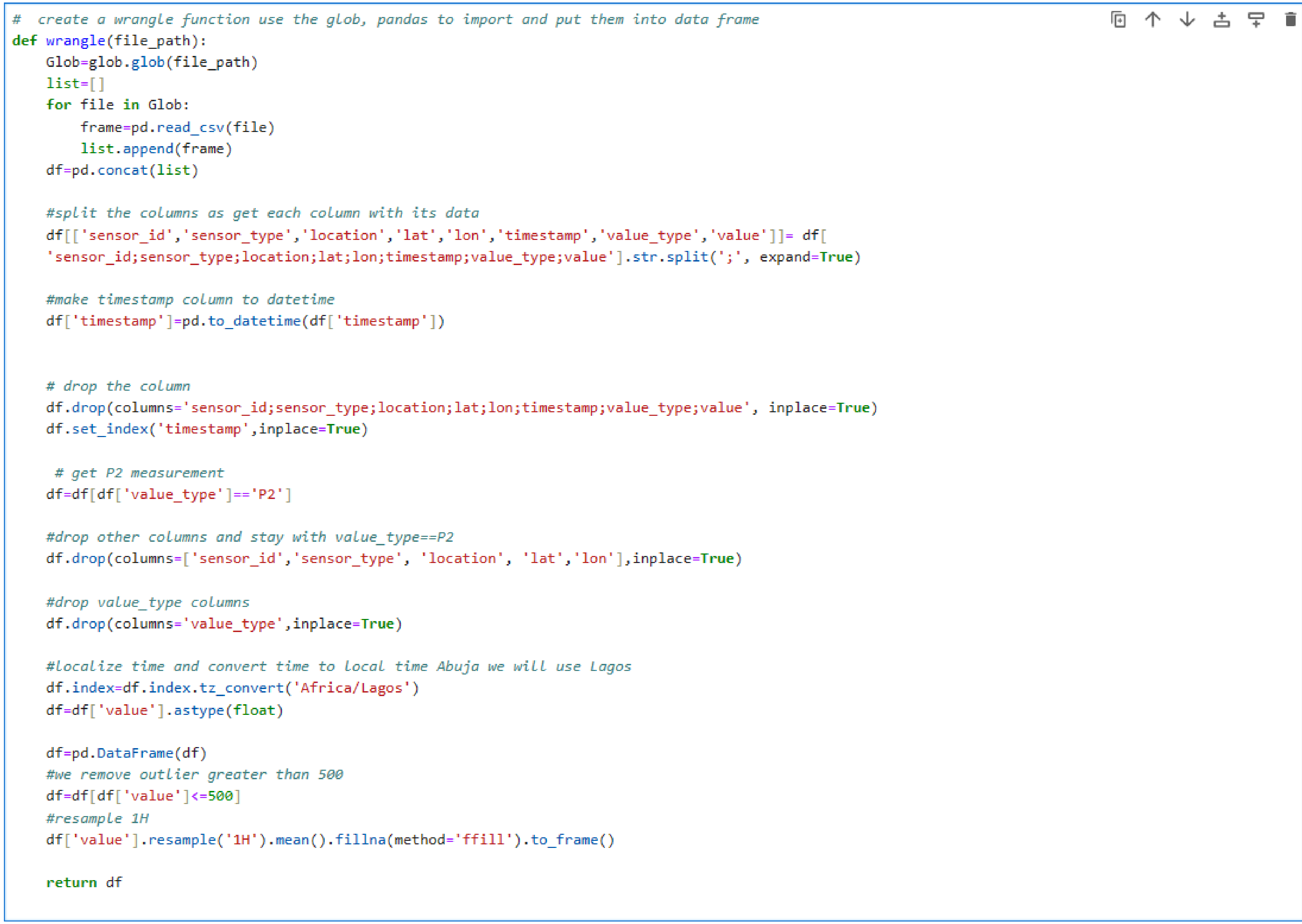
https://open.africa/dataset/sensorsafrica-airquality-archive-abuja

Libraries I used photo from jupyter notebook cell



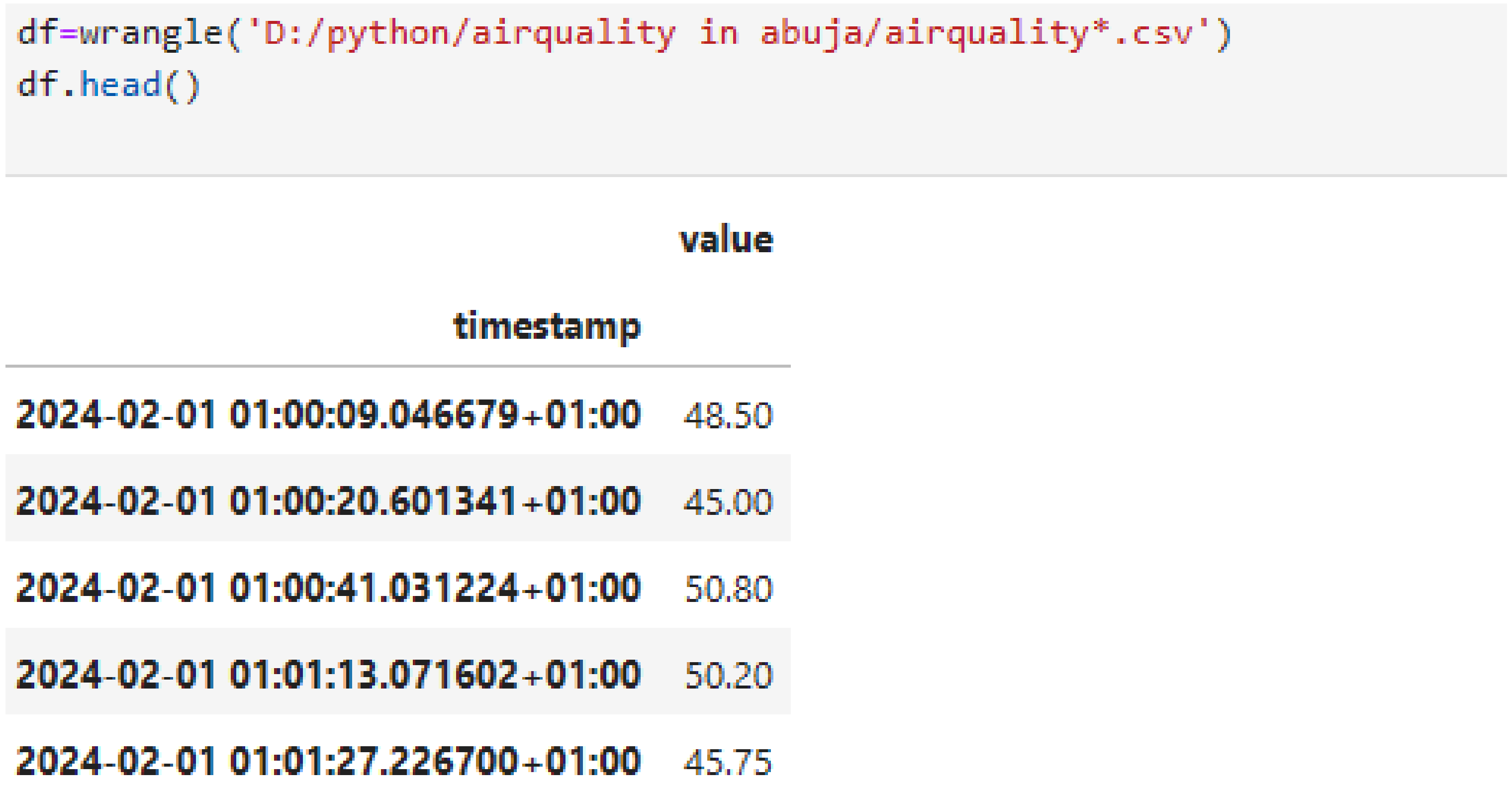
```
# I imported libraries i will use in this project
import numpy as np
import pytz
import glob
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.ar_model import AutoReg
```

Data cleaning photo from jupyter notebook cell



```python
#  create a wrangle function use the glob, pandas to import and put them into data frame
def wrangle(file_path):
    Glob=glob.glob(file_path)
    list=[]
    for file in Glob:
        frame=pd.read_csv(file)
        list.append(frame)
    df=pd.concat(list)

    #split the columns as get each column with its data
    df[['sensor_id','sensor_type','location','lat','lon','timestamp','value_type','value']]= df[
    'sensor_id;sensor_type;location;lat;lon;timestamp;value_type;value'].str.split(';', expand=True)

    #make timestamp column to datetime
    df['timestamp']=pd.to_datetime(df['timestamp'])


    # drop the column
    df.drop(columns='sensor_id;sensor_type;location;lat;lon;timestamp;value_type;value', inplace=True)
    df.set_index('timestamp',inplace=True)

     # get P2 measurement
    df=df[df['value_type']=='P2']

    #drop other columns and stay with value_type==P2
    df.drop(columns=['sensor_id','sensor_type', 'location', 'lat','lon'],inplace=True)

    #drop value_type columns
    df.drop(columns='value_type',inplace=True)

    #localize time and convert time to local time Abuja we will use Lagos
    df.index=df.index.tz_convert('Africa/Lagos')
    df=df['value'].astype(float)

    df=pd.DataFrame(df)
    #we remove outlier greater than 500
    df=df[df['value']<=500]
    #resample 1H
    df['value'].resample('1H').mean().fillna(method='ffill').to_frame()

    return df
```

We call our function and assign to it df



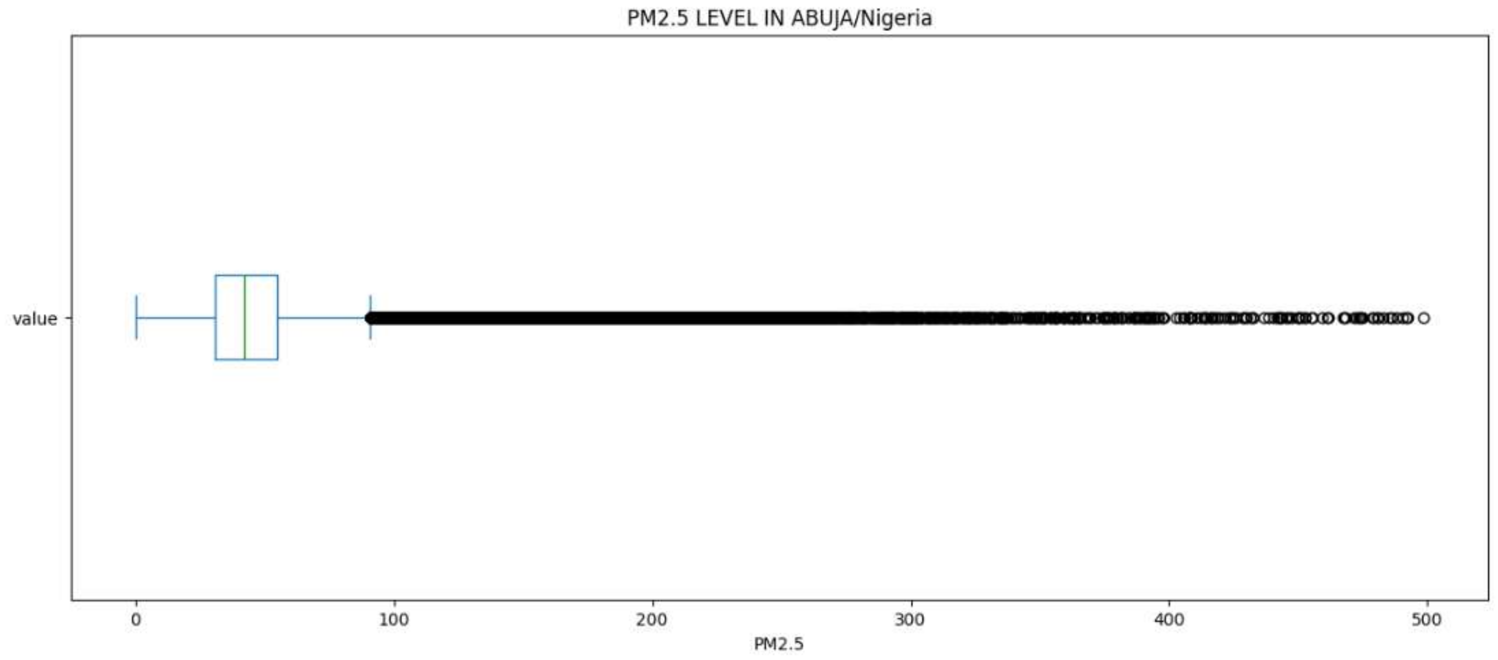
```
df=wrangle('D:/python/airquality in abuja/airquality*.csv')
df.head()
```

```
                                    value
timestamp
2024-02-01 01:00:09.046679+01:00    48.50
2024-02-01 01:00:20.601341+01:00    45.00
2024-02-01 01:00:41.031224+01:00    50.80
2024-02-01 01:01:13.071602+01:00    50.20
2024-02-01 01:01:27.226700+01:00    45.75
```

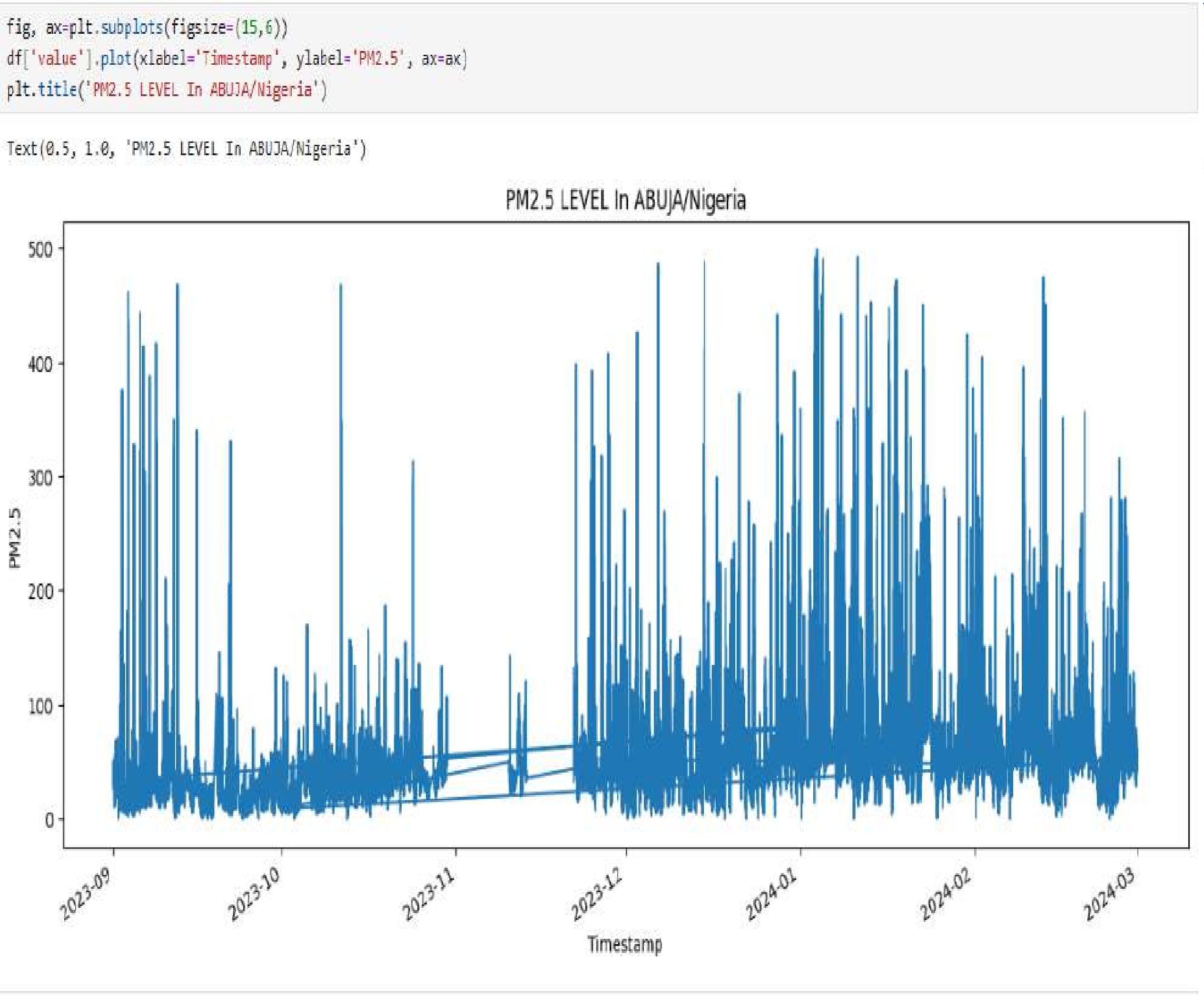
Using box plot

```
fig, ax=plt.subplots(figsize=(15,6))
df['value'].plot(kind='box',vert=False, ax=ax)
plt.xlabel('PM2.5')
plt.title('PM2.5 LEVEL IN ABUJA/Nigeria')
```

```
Text(0.5, 1.0, 'PM2.5 LEVEL IN ABUJA/Nigeria')
```

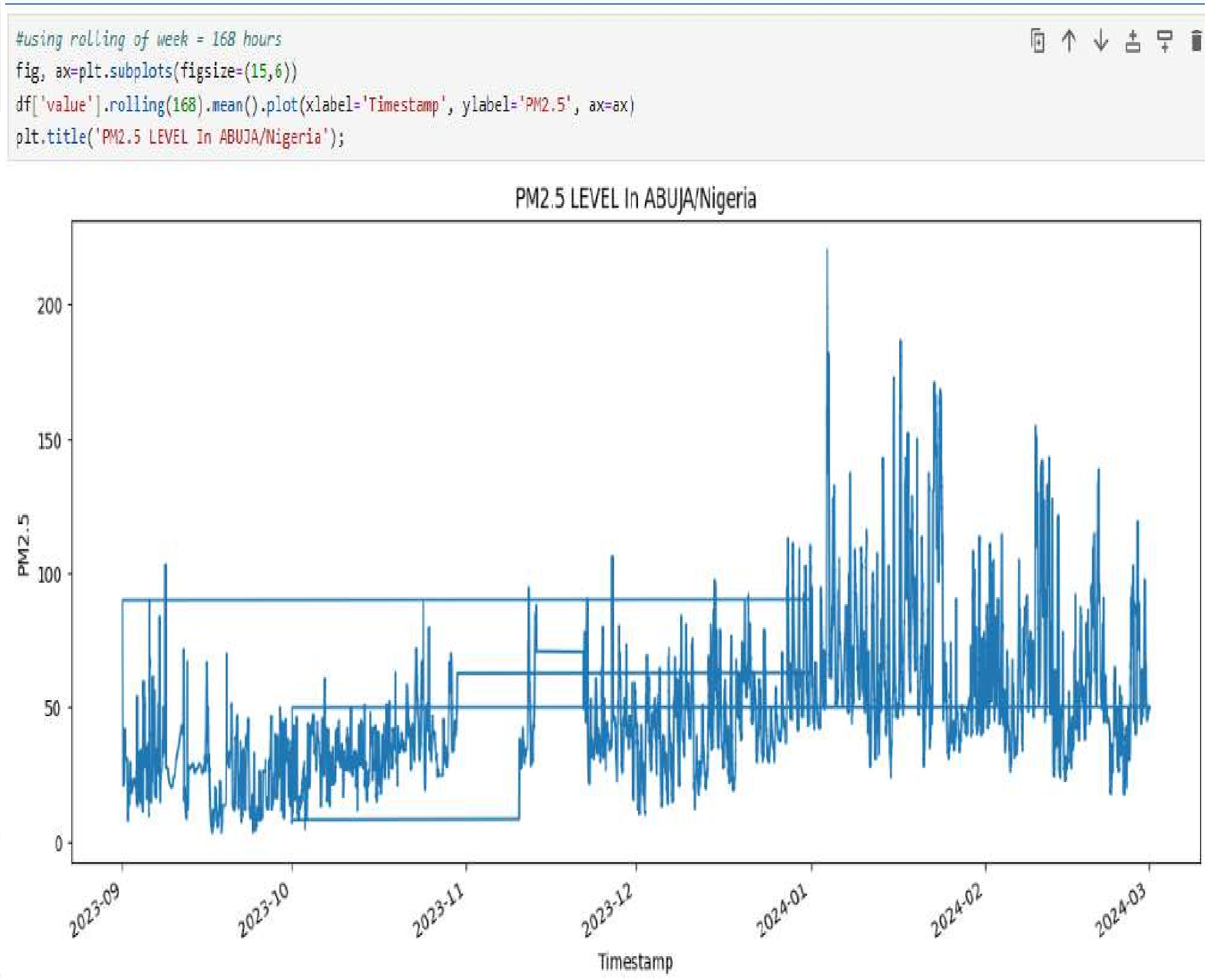


Comparison, before and after rolling average= 1 week(168 hours)



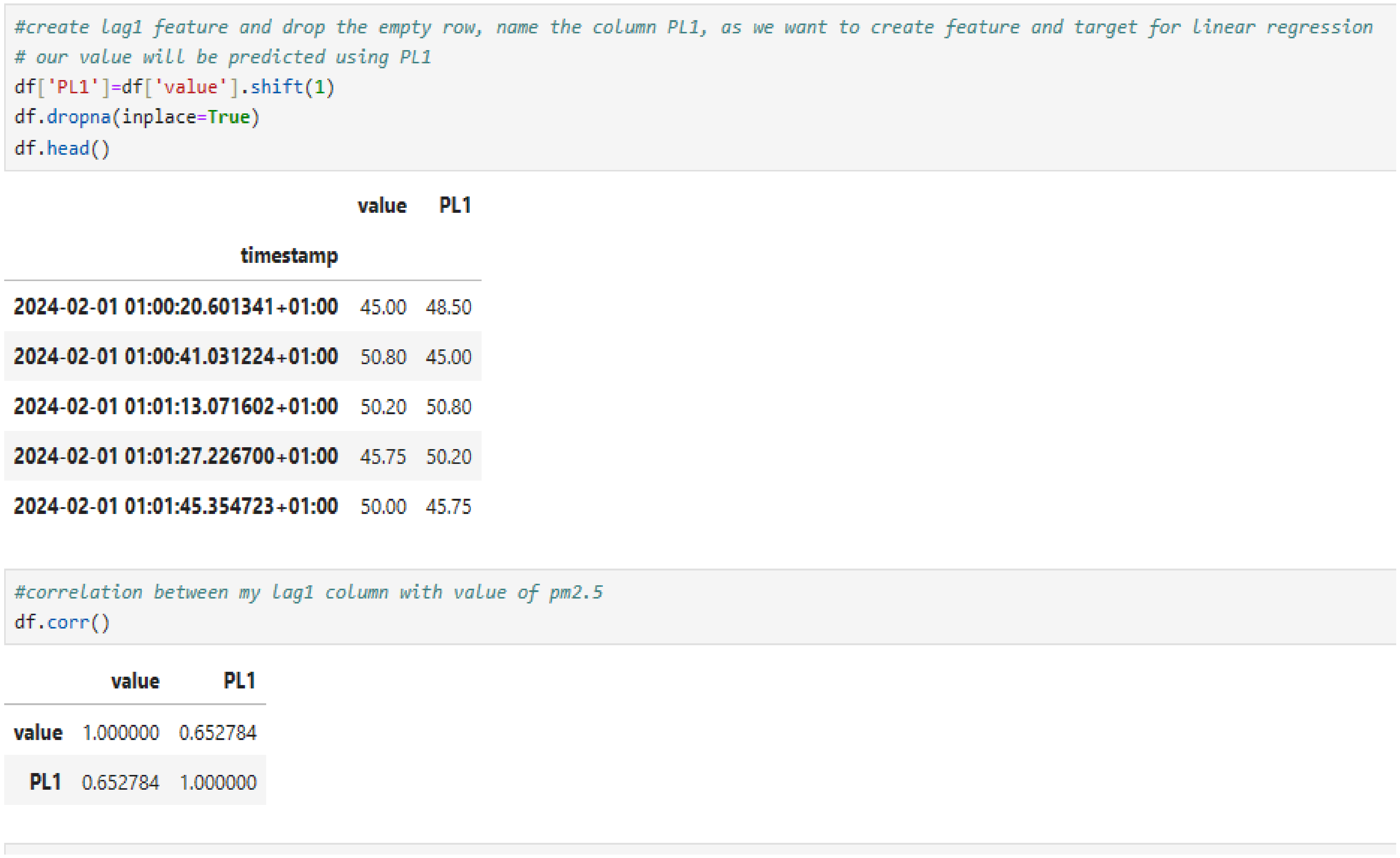
```
fig, ax=plt.subplots(figsize=(15,6))
df['value'].plot(xlabel='Timestamp', ylabel='PM2.5', ax=ax)
plt.title('PM2.5 LEVEL In ABUJA/Nigeria')
```

```
Text(0.5, 1.0, 'PM2.5 LEVEL In ABUJA/Nigeria')
```



```
#using rolling of week = 168 hours
fig, ax=plt.subplots(figsize=(15,6))
df['value'].rolling(168).mean().plot(xlabel='Timestamp', ylabel='PM2.5', ax=ax)
plt.title('PM2.5 LEVEL In ABUJA/Nigeria');
```

I went one step a back and correlation between current observations and lag1



```python
#create lag1 feature and drop the empty row, name the column PL1, as we want to create feature and target for linear regression
# our value will be predicted using PL1
df['PL1']=df['value'].shift(1)
df.dropna(inplace=True)
df.head()
```

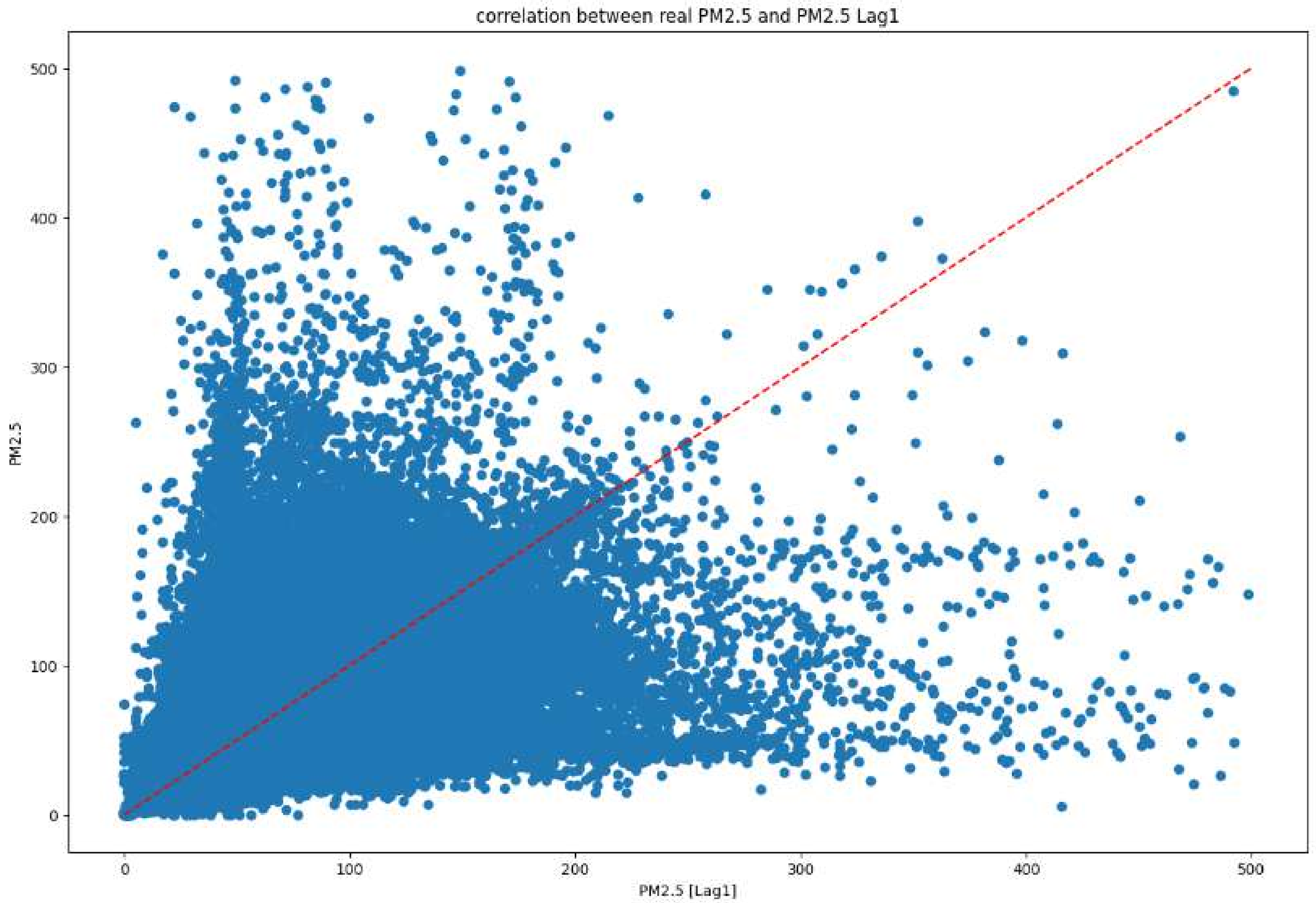
	value	PL1
timestamp		
2024-02-01 01:00:20.601341+01:00	45.00	48.50
2024-02-01 01:00:41.031224+01:00	50.80	45.00
2024-02-01 01:01:13.071602+01:00	50.20	50.80
2024-02-01 01:01:27.226700+01:00	45.75	50.20
2024-02-01 01:01:45.354723+01:00	50.00	45.75

```python
#correlation between my lag1 column with value of pm2.5
df.corr()
```

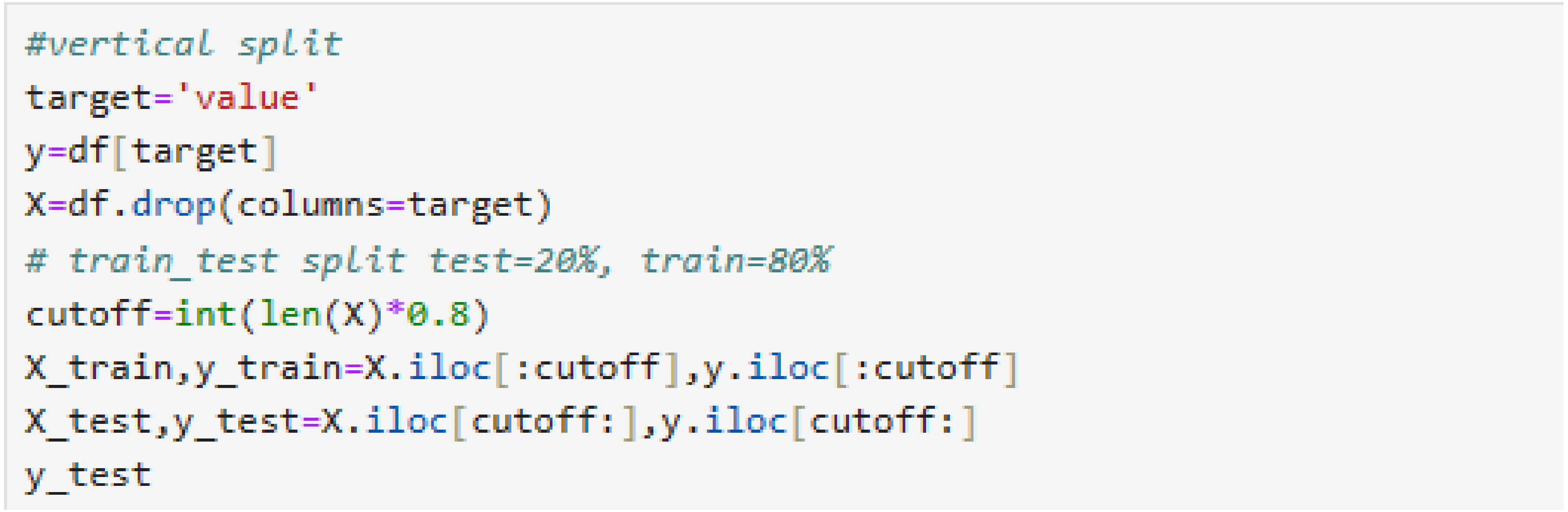
	value	PL1
value	1.000000	0.652784
PL1	0.652784	1.000000

Scatter plot showing correlation with red line as an overall linear correlation

```
fig, ax=plt.subplots(figsize=(15,10))
plt.scatter(x=df['PL1'], y=df['value'])
plt.plot((0,500),(0,500), linestyle='--',color='red')
plt.xlabel('PM2.5 [Lag1]')
plt.ylabel('PM2.5')
plt.title('correlation between real PM2.5 and PM2.5 Lag1')
plt.show()
```

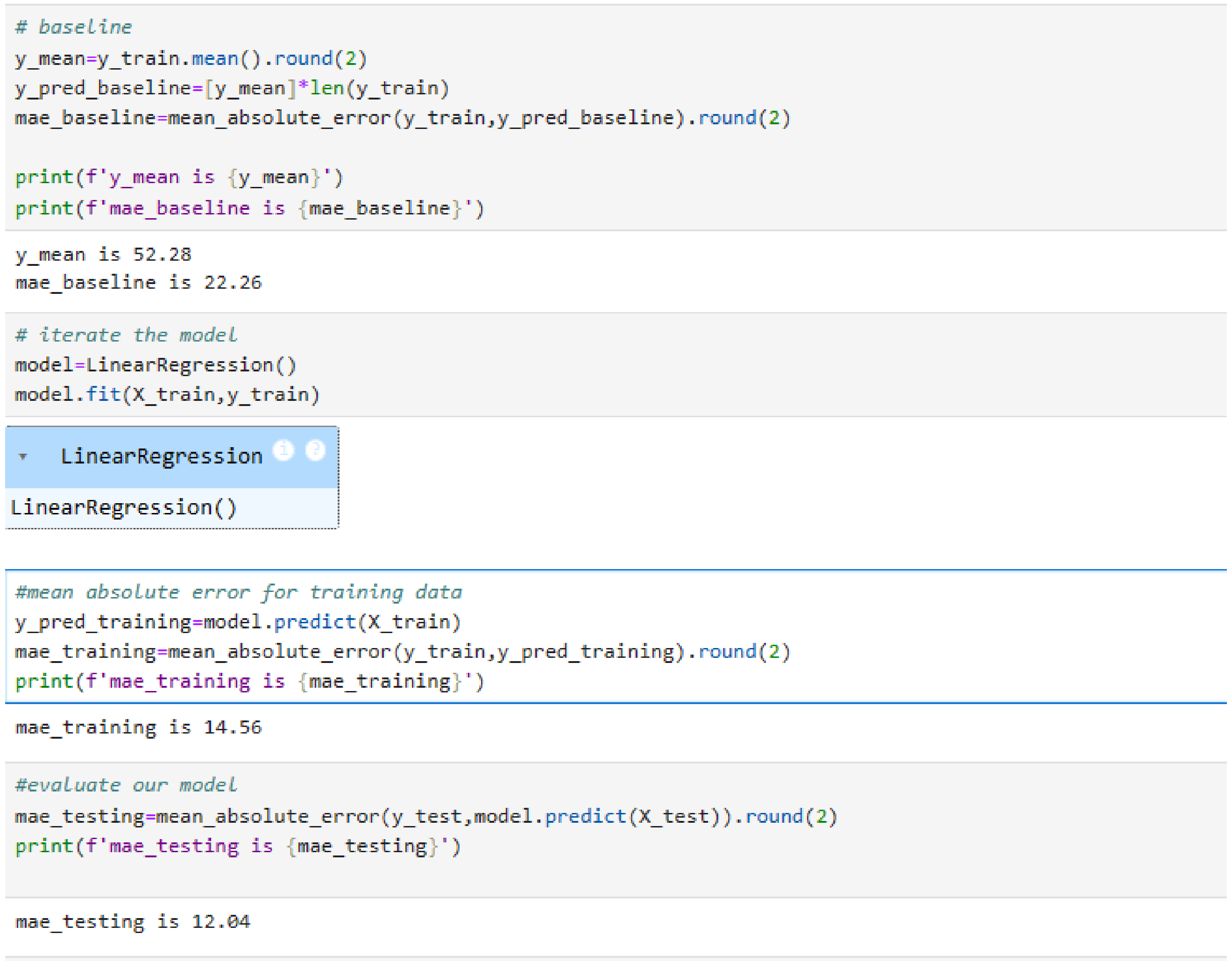


Split our data



```
#vertical split
target='value'
y=df[target]
X=df.drop(columns=target)
# train_test split test=20%, train=80%
cutoff=int(len(X)*0.8)
X_train,y_train=X.iloc[:cutoff],y.iloc[:cutoff]
X_test,y_test=X.iloc[cutoff:],y.iloc[cutoff:]
y_test
```

Baseline, iterate, train, evaluate model and calculate mean_absolute_error



```
# baseline
y_mean=y_train.mean().round(2)
y_pred_baseline=[y_mean]*len(y_train)
mae_baseline=mean_absolute_error(y_train,y_pred_baseline).round(2)

print(f'y_mean is {y_mean}')
print(f'mae_baseline is {mae_baseline}')
```

```
y_mean is 52.28
mae_baseline is 22.26
```

```
# iterate the model
model=LinearRegression()
model.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

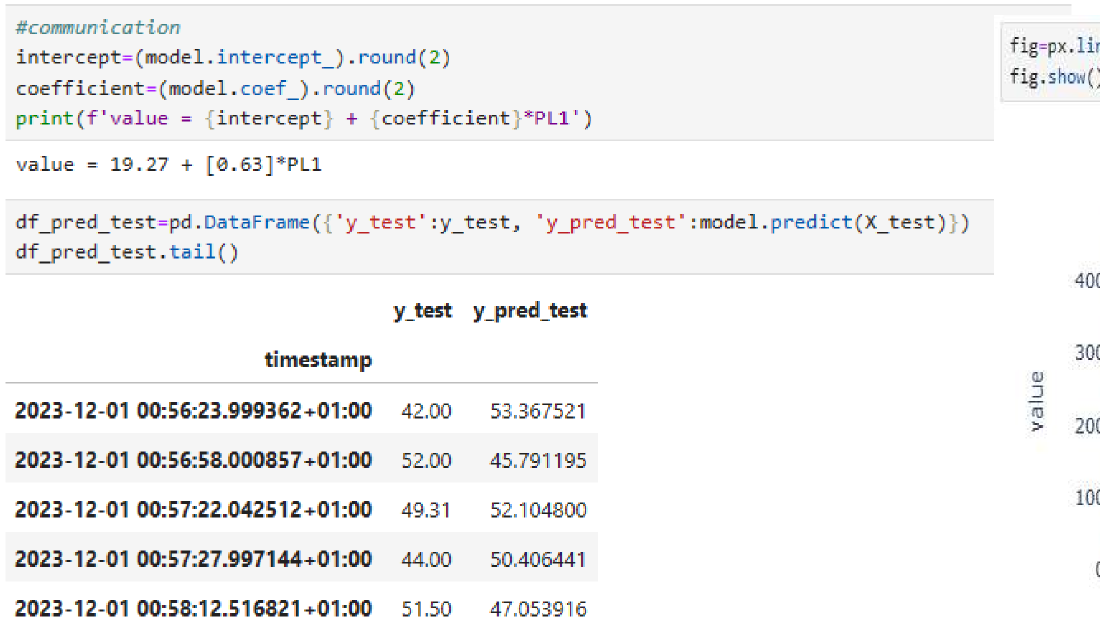
```
#mean absolute error for training data
y_pred_training=model.predict(X_train)
mae_training=mean_absolute_error(y_train,y_pred_training).round(2)
print(f'mae_training is {mae_training}')
```

```
mae_training is 14.56
```

```
#evaluate our model
mae_testing=mean_absolute_error(y_test,model.predict(X_test)).round(2)
print(f'mae_testing is {mae_testing}')
```

```
mae_testing is 12.04
```

Communication: we can see that there some correlation between current value and predicted values

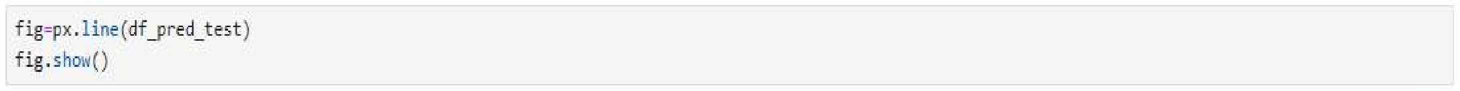


```
#communication
intercept=(model.intercept_).round(2)
coefficient=(model.coef_).round(2)
print(f'value = {intercept} + {coefficient}*PL1')
```

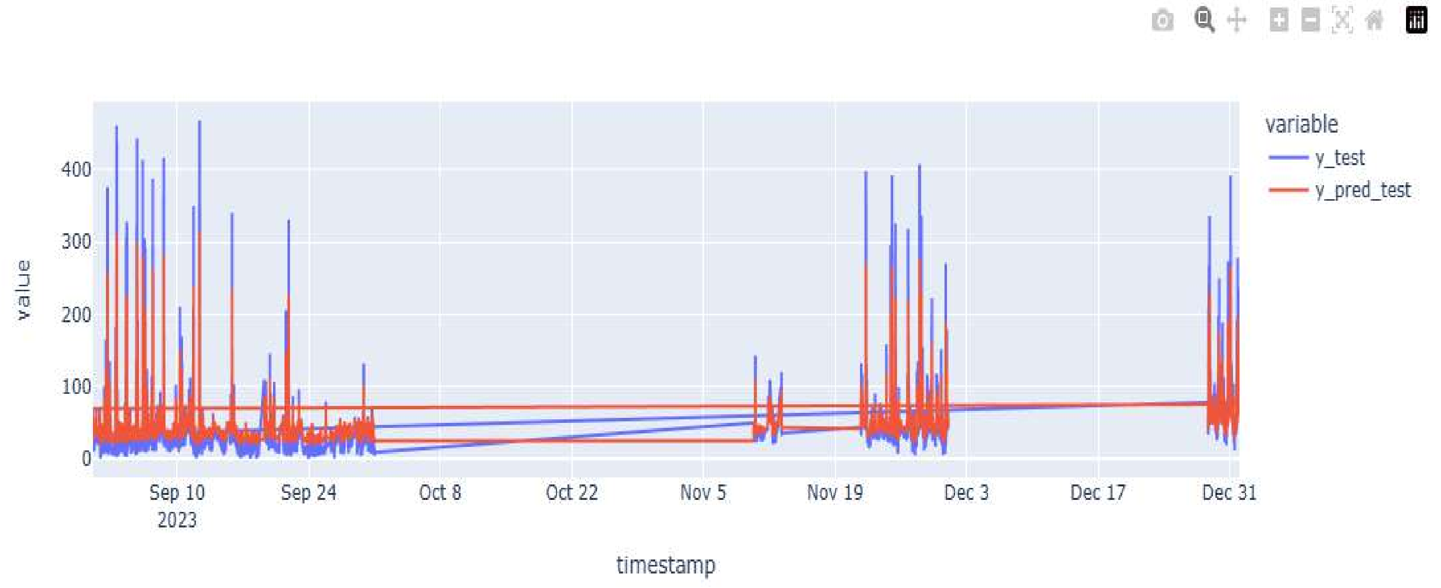
```
value = 19.27 + [0.63]*PL1
```

```
df_pred_test=pd.DataFrame({'y_test':y_test, 'y_pred_test':model.predict(X_test)})
df_pred_test.tail()
```

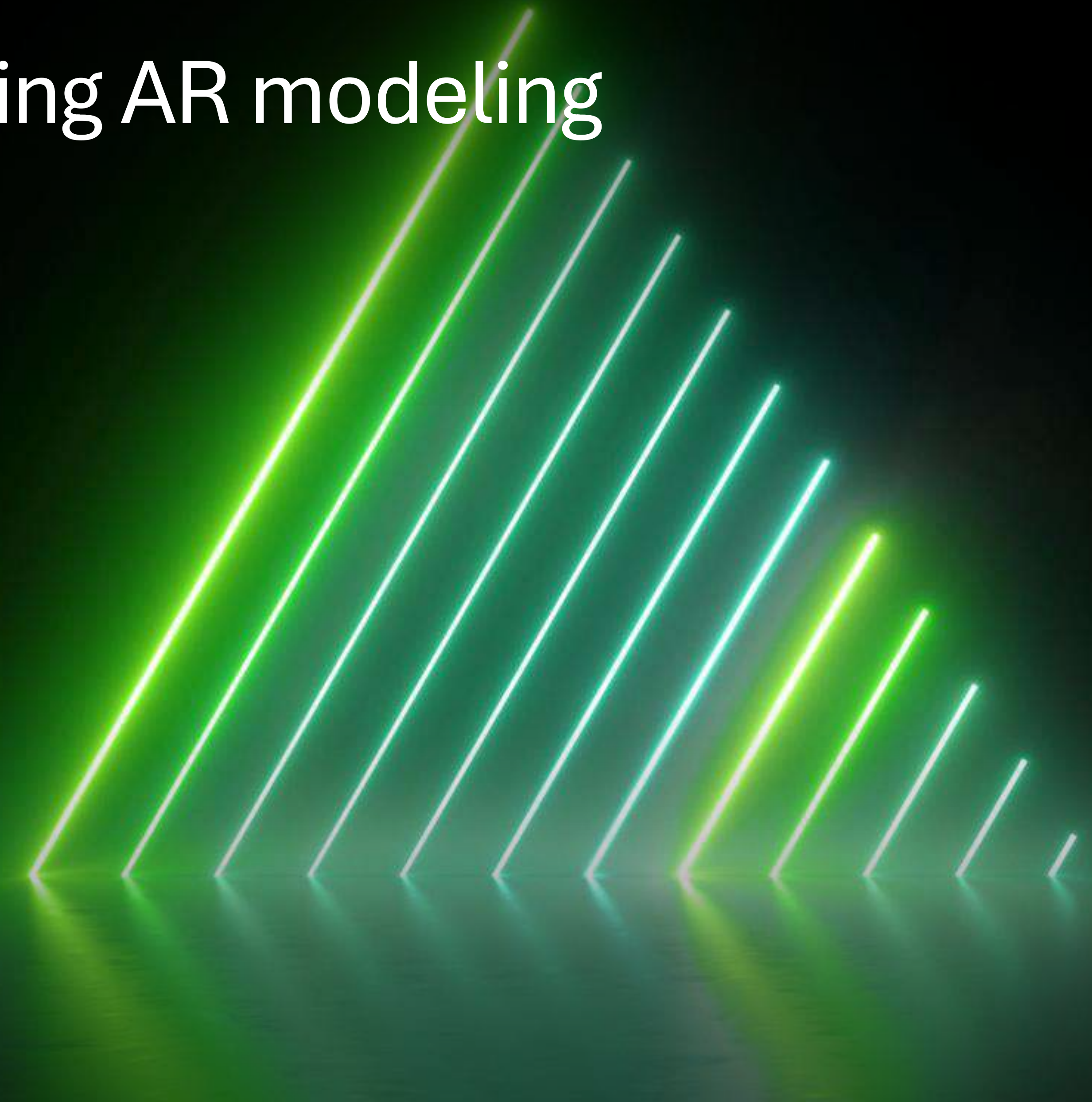
	y_test	y_pred_test
timestamp		
2023-12-01 00:56:23.999362+01:00	42.00	53.367521
2023-12-01 00:56:58.000857+01:00	52.00	45.791195
2023-12-01 00:57:22.042512+01:00	49.31	52.104800
2023-12-01 00:57:27.997144+01:00	44.00	50.406441
2023-12-01 00:58:12.516821+01:00	51.50	47.053916



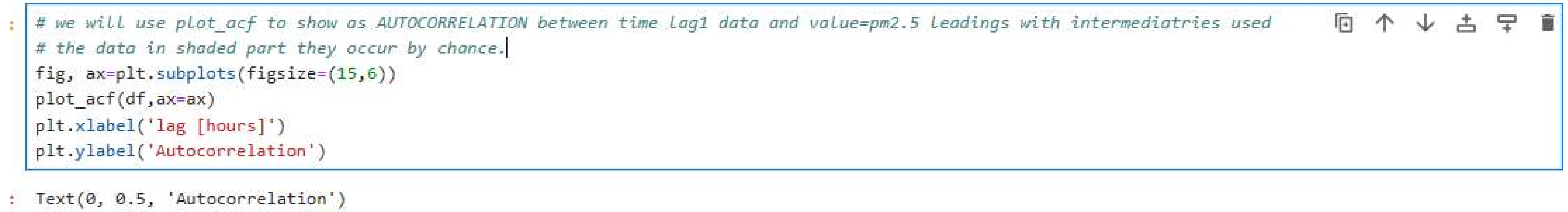
```
fig=px.line(df_pred_test)
fig.show()
```



Let’s see using AR modeling

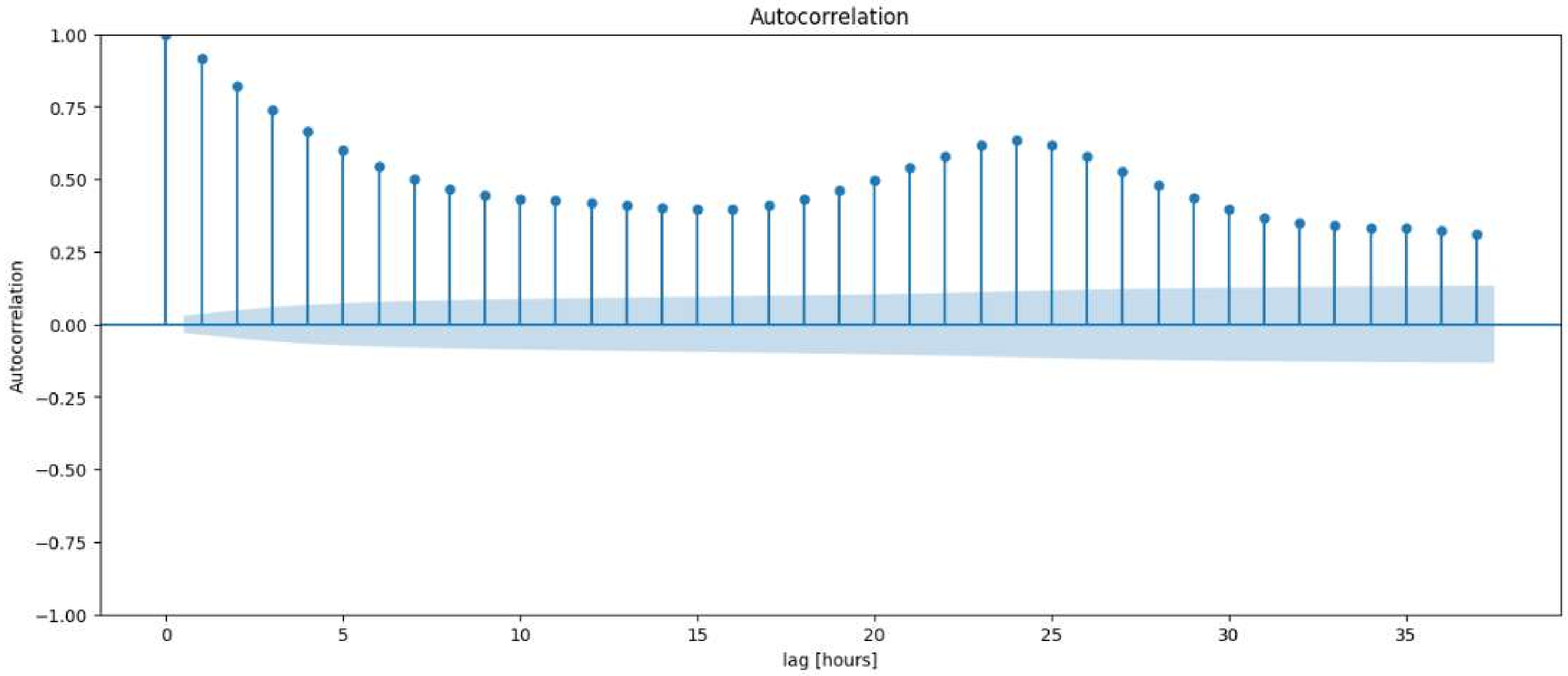


Plot_acf

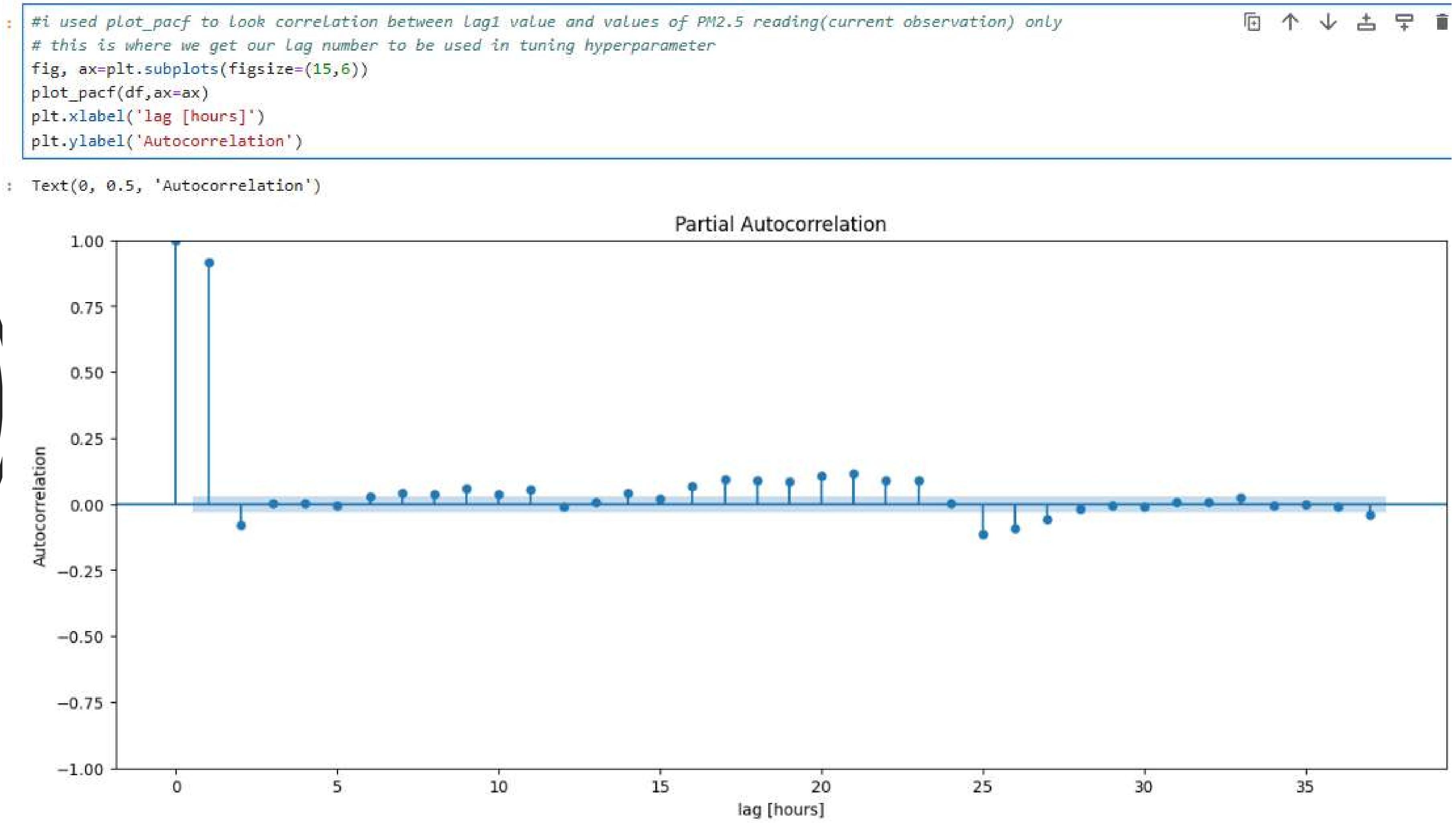


```
# we will use plot_acf to show as AUTOCORRELATION between time lag1 data and value=pm2.5 leadings with intermediatries used
# the data in shaded part they occur by chance.
fig, ax=plt.subplots(figsize=(15,6))
plot_acf(df,ax=ax)
plt.xlabel('lag [hours]')
plt.ylabel('Autocorrelation')
```

```
Text(0, 0.5, 'Autocorrelation')
```



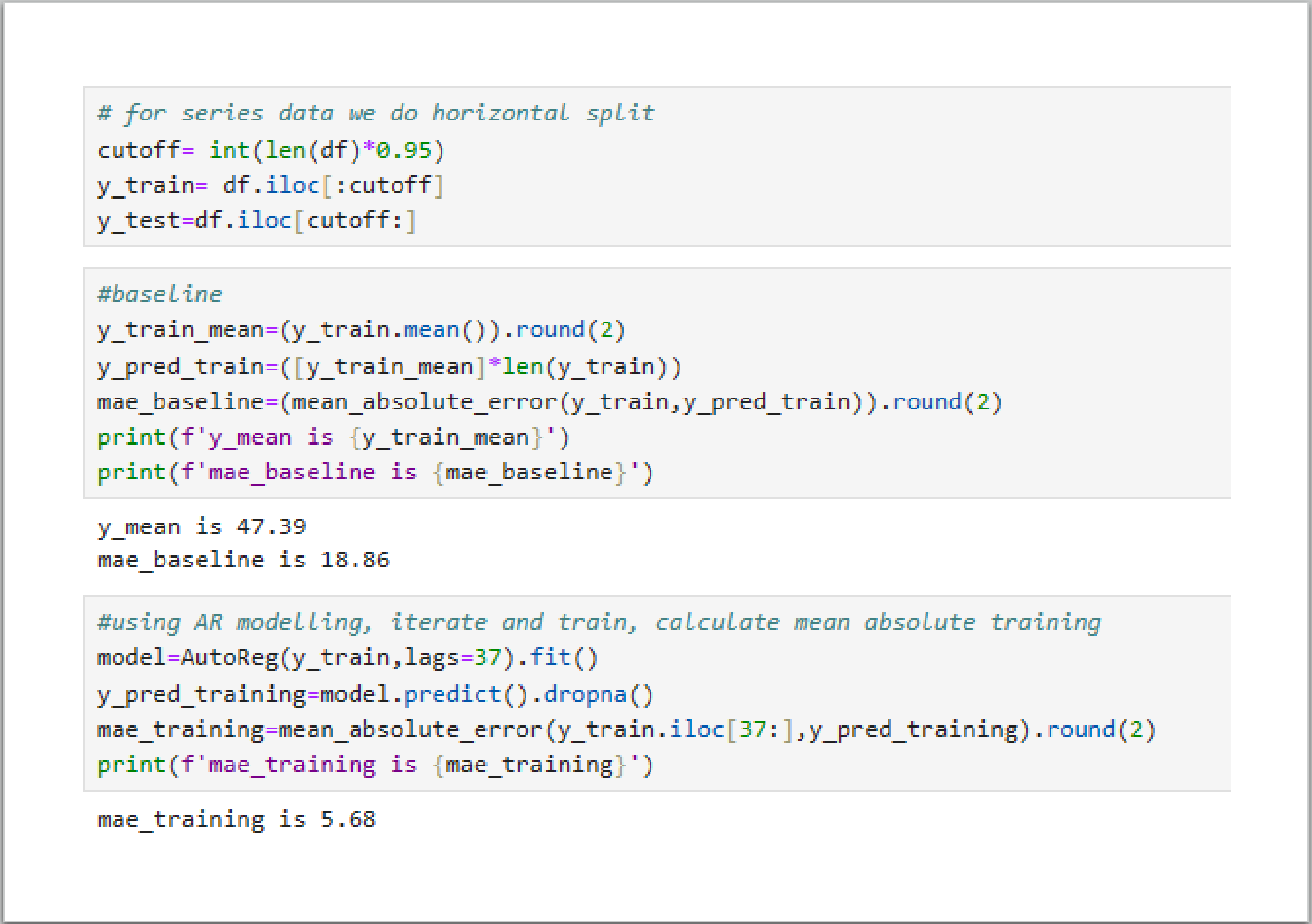
Plot_pacf, during tuning our model we will use lag=37, as we can see it performs well even on 36



```
#i used plot_pacf to look correlation between lag1 value and values of PM2.5 reading(current observation) only
# this is where we get our lag number to be used in tuning hyperparameter
fig, ax=plt.subplots(figsize=(15,6))
plot_pacf(df,ax=ax)
plt.xlabel('lag [hours]')
plt.ylabel('Autocorrelation')
```

```
Text(0, 0.5, 'Autocorrelation')
```

Data split, baseline, iterate, train model and use lag=37



```
# for series data we do horizontal split
cutoff= int(len(df)*0.95)
y_train= df.iloc[:cutoff]
y_test=df.iloc[cutoff:]
```

```
#baseline
y_train_mean=(y_train.mean()).round(2)
y_pred_train=([y_train_mean]*len(y_train))
mae_baseline=(mean_absolute_error(y_train,y_pred_train)).round(2)
print(f'y_mean is {y_train_mean}')
print(f'mae_baseline is {mae_baseline}')
```

```
y_mean is 47.39
mae_baseline is 18.86
```

```
#using AR modelling, iterate and train, calculate mean absolute training
model=AutoReg(y_train,lags=37).fit()
y_pred_training=model.predict().dropna()
mae_training=mean_absolute_error(y_train.iloc[37:],y_pred_training).round(2)
print(f'mae_training is {mae_training}')
```

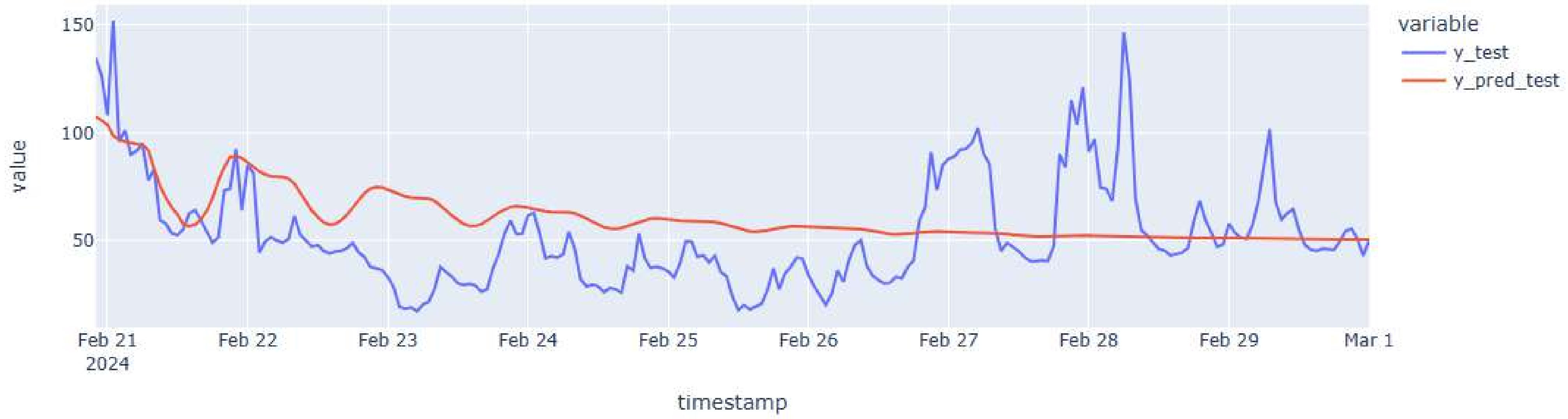
```
mae_training is 5.68
```

We can see that our model is performing poorly, as it is giving *mae_testing*=*20.38*, while our model trained well at *mae_training*=*5.68*, so we need to use walk forward validation

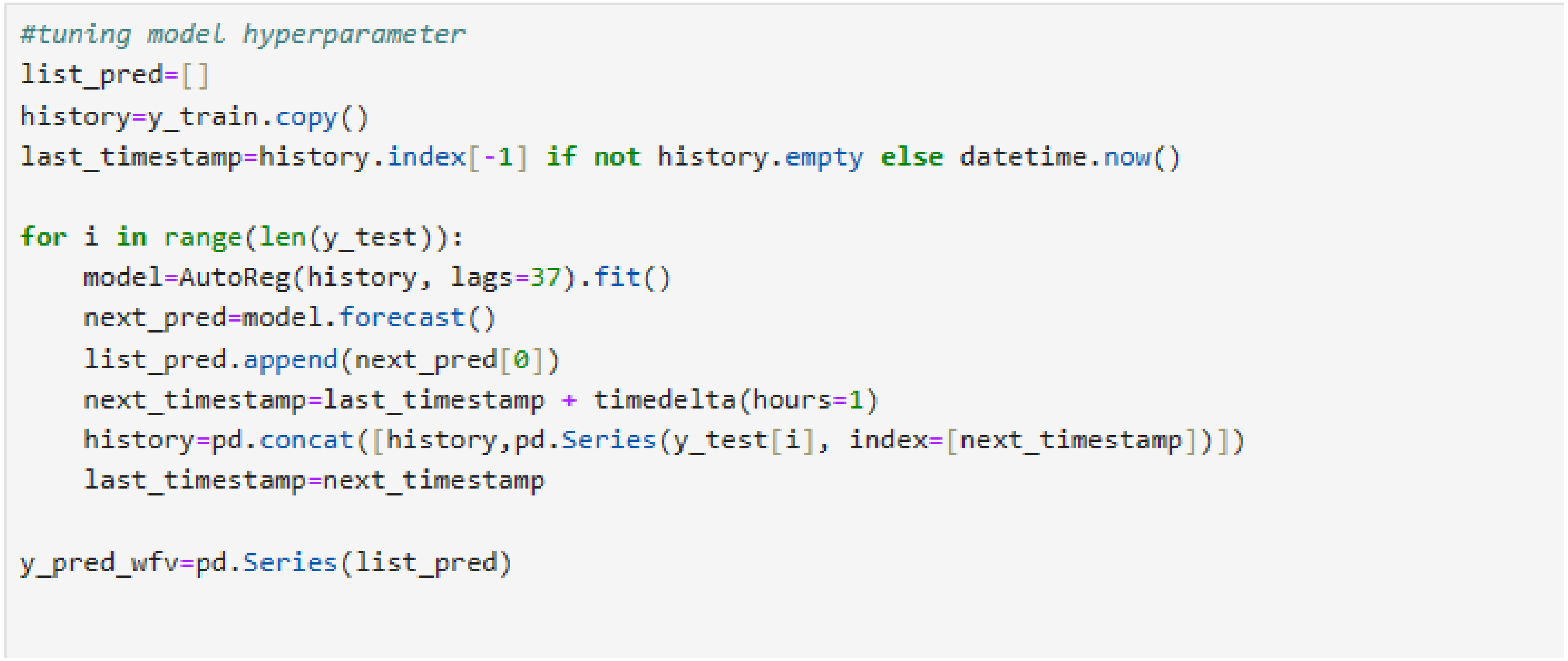
```
y_pred_test= model.predict(y_test.index.min(),y_test.index.max())
mae_test=mean_absolute_error(y_test,y_pred_test).round(2)
print(f'mae_testing is {mae_test}')
```

```
mae_testing is 20.38
```

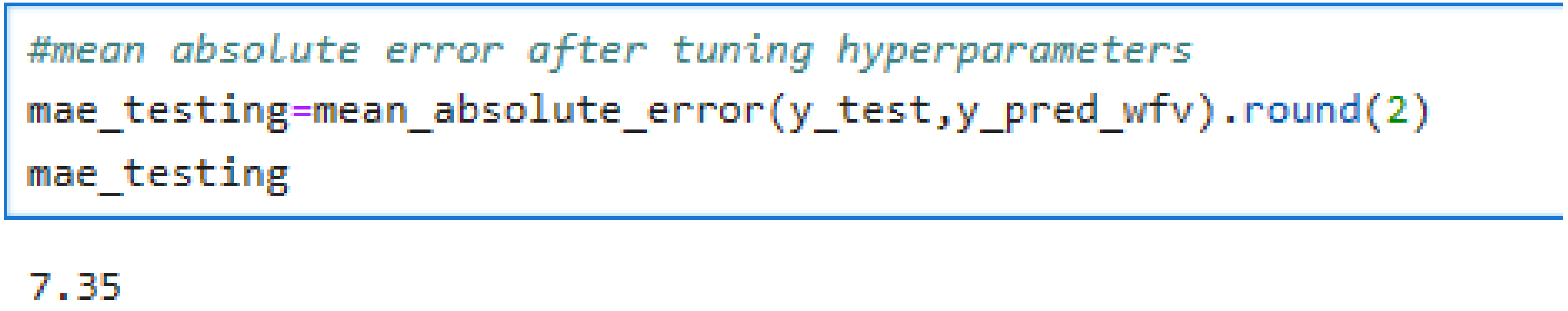
```
df_test_pred=pd.DataFrame({'y_test':y_test,'y_pred_test':y_pred_test},index=y_test.index)
fig=px.line(df_test_pred)
fig.show()
```



lag=37 and use *walk forward validation*


```
#tuning model hyperparameter
list_pred=[]
history=y_train.copy()
last_timestamp=history.index[-1] if not history.empty else datetime.now()

for i in range(len(y_test)):
    model=AutoReg(history, lags=37).fit()
    next_pred=model.forecast()
    list_pred.append(next_pred[0])
    next_timestamp=last_timestamp + timedelta(hours=1)
    history=pd.concat([history,pd.Series(y_test[i], index=[next_timestamp])])
    last_timestamp=next_timestamp

y_pred_wfv=pd.Series(list_pred)
```

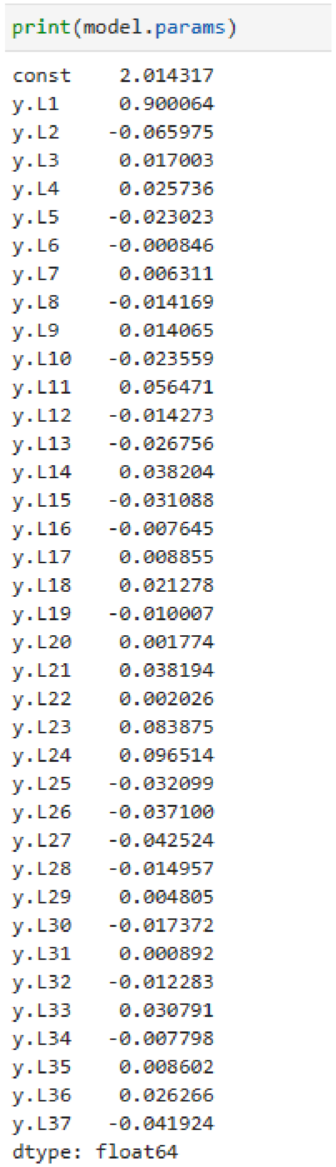
After using Walk forward validation



```
#mean absolute error after tuning hyperparameters
mae_testing=mean_absolute_error(y_test,y_pred_wfv).round(2)
mae_testing
```

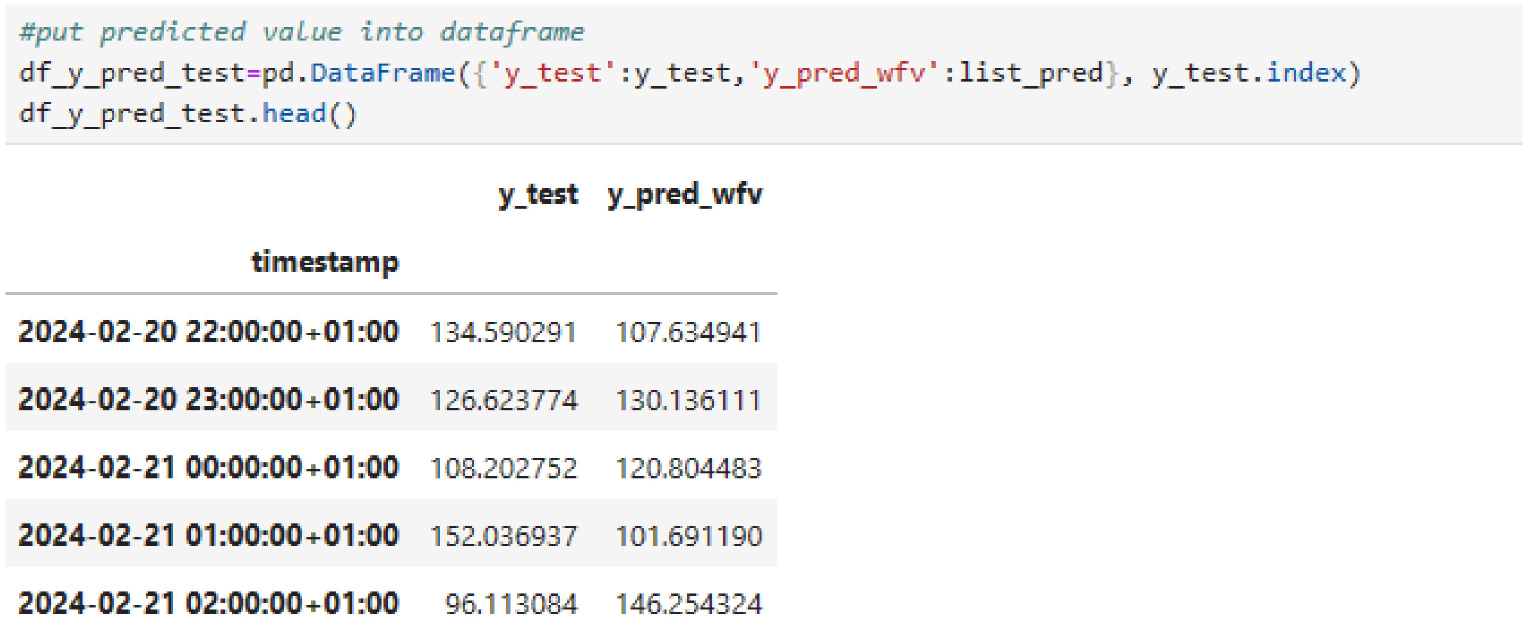
```
7.35
```

Params at each lag, and data frame of our new data



```
print(model.params)
```

```
const     2.014317
y.L1      0.900064
y.L2     -0.065975
y.L3      0.017003
y.L4      0.025736
y.L5     -0.023023
y.L6     -0.000846
y.L7      0.006311
y.L8     -0.014169
y.L9      0.014065
y.L10    -0.023559
y.L11     0.056471
y.L12    -0.014273
y.L13    -0.026756
y.L14     0.038204
y.L15    -0.031088
y.L16    -0.007645
y.L17     0.008855
y.L18     0.021278
y.L19    -0.010007
y.L20     0.001774
y.L21     0.038194
y.L22     0.002026
y.L23     0.083875
y.L24     0.096514
y.L25    -0.032099
y.L26    -0.037100
y.L27    -0.042524
y.L28    -0.014957
y.L29     0.004805
y.L30    -0.017372
y.L31     0.000892
y.L32    -0.012283
y.L33     0.030791
y.L34    -0.007798
y.L35     0.008602
y.L36     0.026266
y.L37    -0.041924
dtype: float64
```

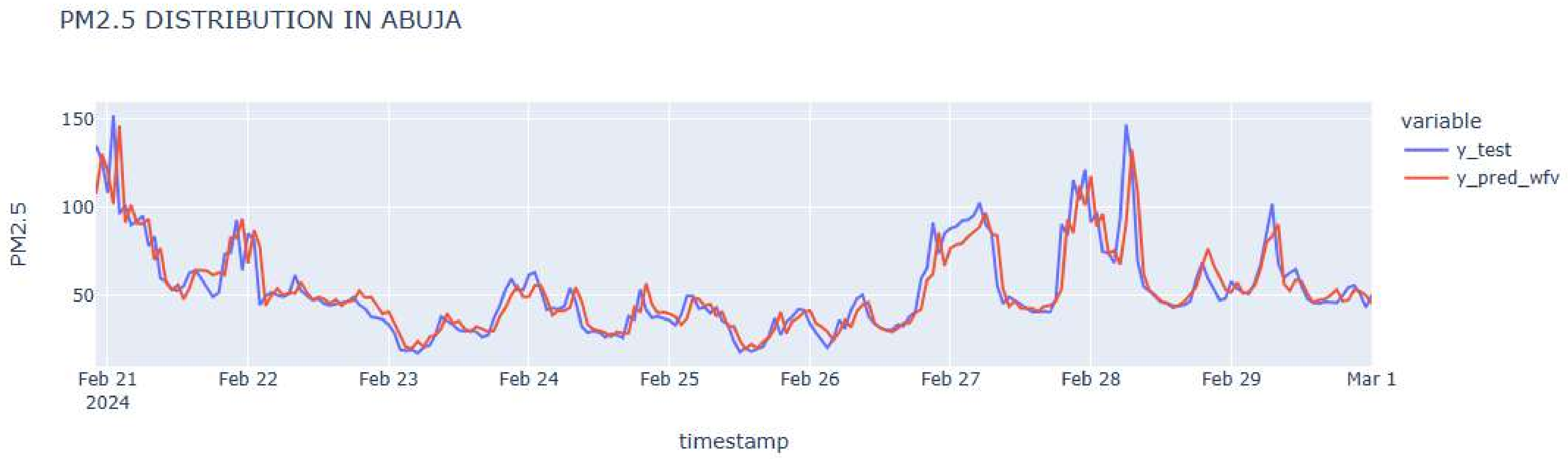


```
#put predicted value into dataframe
df_y_pred_test=pd.DataFrame({'y_test':y_test,'y_pred_wfv':list_pred}, y_test.index)
df_y_pred_test.head()
```

	y_test	y_pred_wfv
timestamp		
2024-02-20 22:00:00+01:00	134.590291	107.634941
2024-02-20 23:00:00+01:00	126.623774	130.136111
2024-02-21 00:00:00+01:00	108.202752	120.804483
2024-02-21 01:00:00+01:00	152.036937	101.691190
2024-02-21 02:00:00+01:00	96.113084	146.254324

We can see that there is trends in predicted value and current value and good model performance

```
#communication on how model performs
fig=px.line(df_y_pred_test,labels={'value':'PM2.5'}, title='PM2.5 DISTRIBUTION IN ABUJA')
fig.update_layout(title_text='PM2.5 DISTRIBUTION IN ABUJA')
fig.show()
```



Trends and insights

- In this project we have seen that using linear regression modelling gives poor performance in our model and this has showed as high mean_absolute_error, which indicate the difference between real value and predicted values have bigger difference.
- We use Plot_pacf to know how long we will go back, lag=n
- In model built used AR(autocorrelation regression) modelling we have seen that we have to use walk forward validation to increase performance of our model.
- Mean_absolute_error for AR model was less than that of linear regression
- AR modelling is best choice for time series analysis

Recommendations

Implement Real-Time Monitoring Systems

Utilize AR models within real-time air quality monitoring systems to provide accurate and timely forecasts. This can enable public health authorities to issue warnings and health advice to the public when poor air quality is predicted.

Inform Policy Decisions

Policymakers should rely on the more accurate AR modeling predictions to create and enforce air quality regulations. This can help in the design of interventions such as traffic restrictions or industrial emission controls during periods of predicted high pollution.

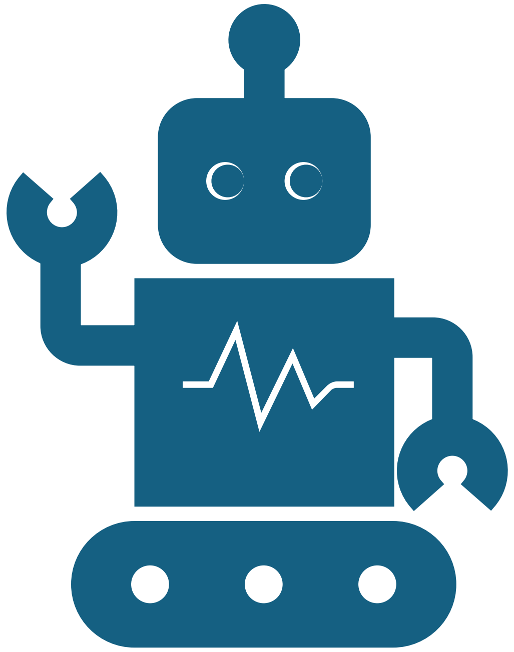
Enhance Public Awareness Campaigns

Develop public awareness campaigns that educate citizens about the potential health risks of poor air quality using the predictive insights from AR models. These campaigns can be targeted to times and locations where poor air quality is expected.

Target High-Risk Populations

Use AR model forecasts to provide guidance to vulnerable populations, such as children, the elderly, and those with pre-existing health conditions, to take preventative actions during times of anticipated poor air quality.

Recommendations



Resource Allocation

Health services should use the more accurate forecasts to allocate resources effectively. For instance, they can prepare for potential increases in hospital admissions related to respiratory problems when high pollution levels are forecasted.

Urban Planning

Urban planners should consider the insights gained from AR modeling to design city landscapes that can help mitigate pollution such as creating green belts or optimizing traffic flow to reduce congested areas known for high pollution levels.

Cross-Sector Collaboration

Encourage collaboration between meteorological services, environmental agencies, and public health organizations to develop comprehensive strategies for air quality management that leverage the predictive power of AR models.

Continuous Model Improvement

Invest in ongoing research to further refine AR models, ensuring they incorporate the latest data and respond to changing environmental conditions, thus maintaining their accuracy over time.

Conclusion

In the end, using the AR model to predict how clean the air will be has helped make important decisions to keep people safe and the environment clean. This project shows that by using this model, we can take steps before problems happen, leading to better health for everyone and less pollution.