Analyzing Air Quality Data: Unveiling India's Pollution Patterns

Data Science Bootcamp - Spring 2024

Group 4
Jesse
Noppe-Brandon
Mithuna Grandhe
Yixin Xu
Kripa Shankar S R

Introduction

Air quality refers to **the degree to which the air is suitable or clean enough for humans or the environment**.

India is currently one of the major countries of rapid development but it is also very high prone to low air quality levels which cause a lot of diseases like lung cancer as well as even death.

Air Quality Monitoring and Importance

0
50
100
150
200
250
300
350
400
450
500
550
AIR QUALITY INDEX

Air Quality Index (AQI) is **used for reporting daily air quality**. It tells how clean or polluted your air is, and what associated health effects might be a concern for you. AQI focuses on health effects you may experience within a few hours or days after breathing polluted air.

1

Public Health Impact

Poor air quality significantly impacts public health, leading to increased respiratory issues, cardiovascular problems, and reduced life expectancy, especially for vulnerable populations.

2

Environmental Regulations

Comprehensive air quality monitoring is essential for enforcing environmental regulations and ensuring compliance with air pollution standards set by government agencies.

3

Pollution Source Identification

Air quality data can help identify the sources of pollution, enabling targeted interventions and policies to address the root causes of poor air quality.

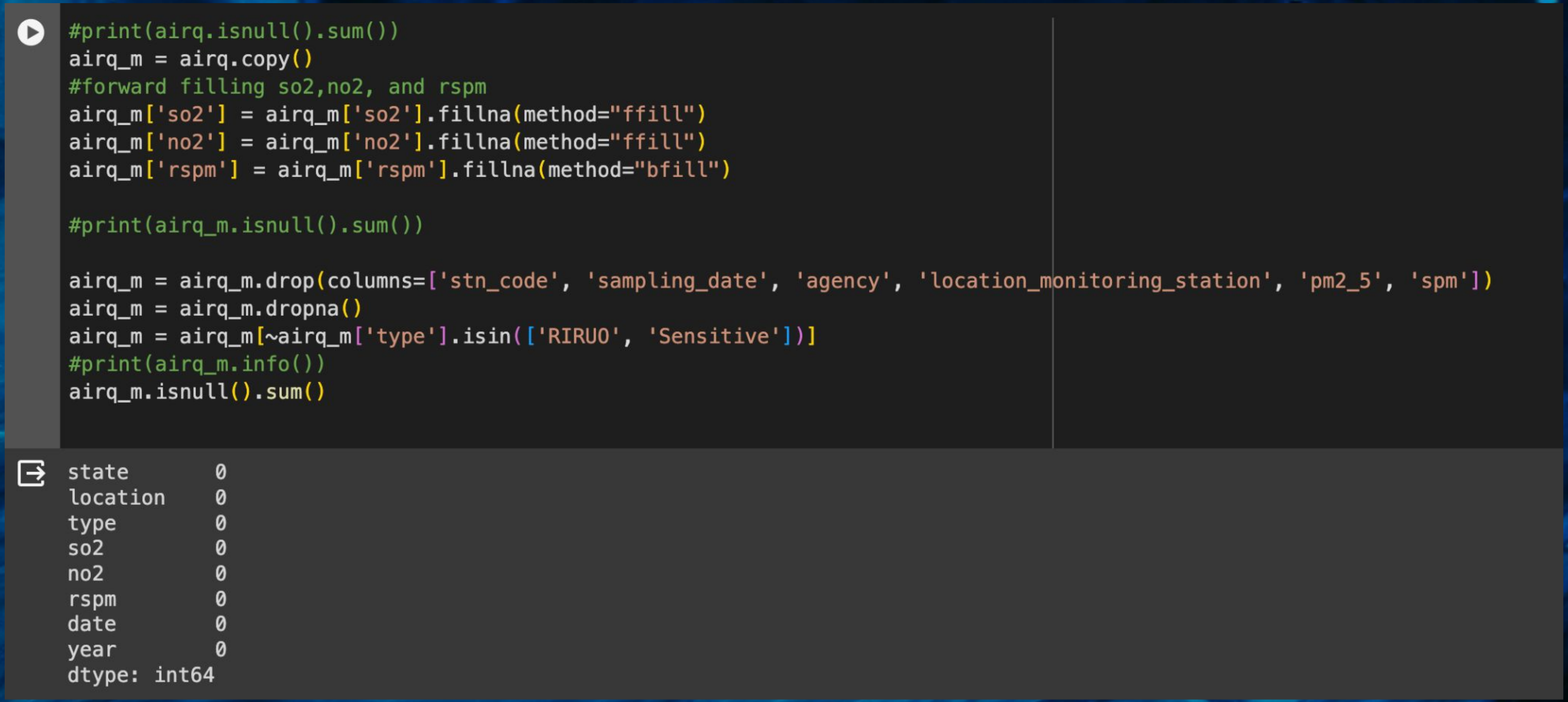
4

Informed Decision-Making

Real-time air quality data empowers citizens, policymakers, and urban planners to make informed decisions about transportation, industrial activities, and other factors affecting air pollution.

Data Collection and Preprocessing

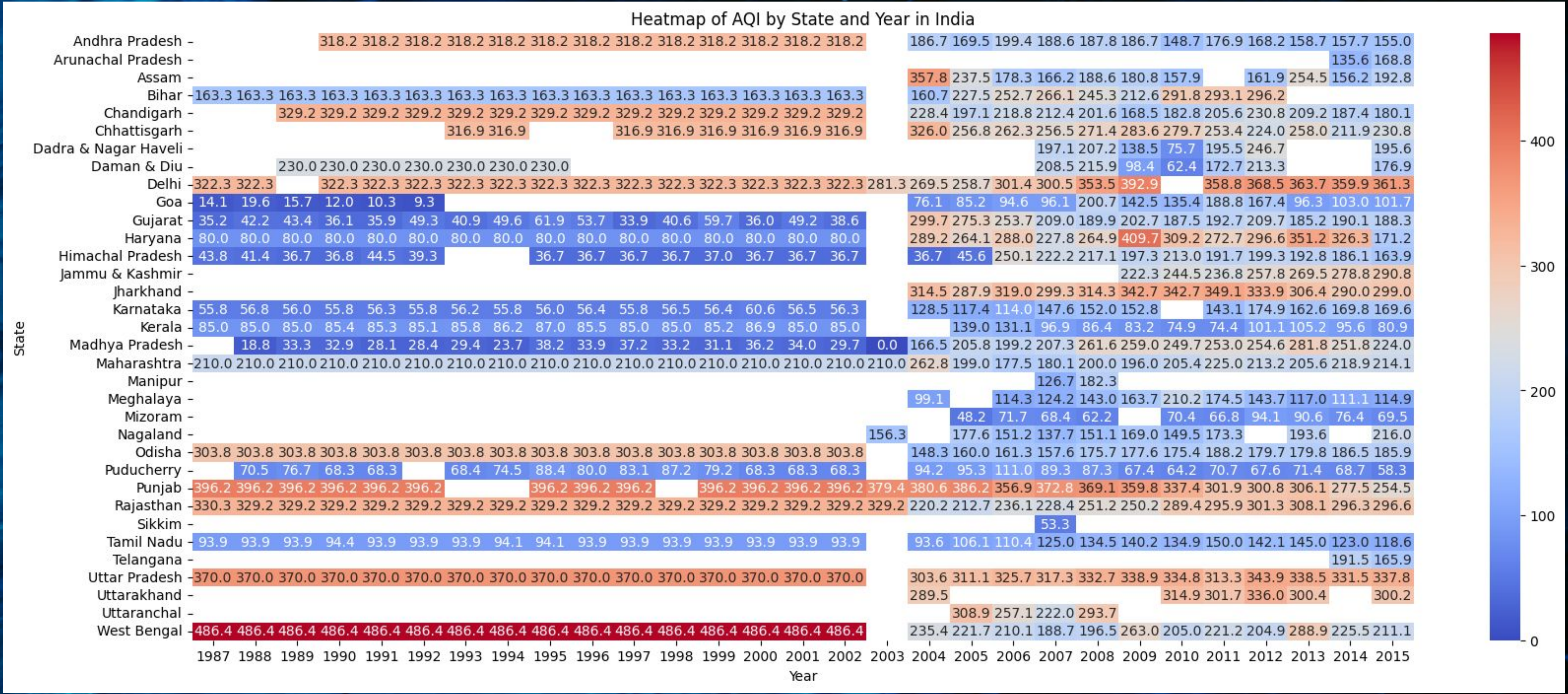
Effective air quality control requires robust data collection strategies. Networked IoT sensors capture real-time measurements of pollutants, particulates, and environmental factors across urban and industrial areas. Preprocessing this raw sensor data is crucial, involving techniques like outlier removal, data imputation, and feature scaling to ensure data integrity.



```
#print(airq.isnull().sum())
airq_m = airq.copy()
#forward filling so2,no2, and rspm
airq_m['so2'] = airq_m['so2'].fillna(method="ffill")
airq_m['no2'] = airq_m['no2'].fillna(method="ffill")
airq_m['rspm'] = airq_m['rspm'].fillna(method="bfill")

#print(airq_m.isnull().sum())

airq_m = airq_m.drop(columns=['stn_code', 'sampling_date', 'agency', 'location_monitoring_station', 'pm2_5', 'spm'])
airq_m = airq_m.dropna()
airq_m = airq_m[~airq_m['type'].isin(['RIRUO', 'Sensitive'])]
#print(airq_m.info())
airq_m.isnull().sum()
```

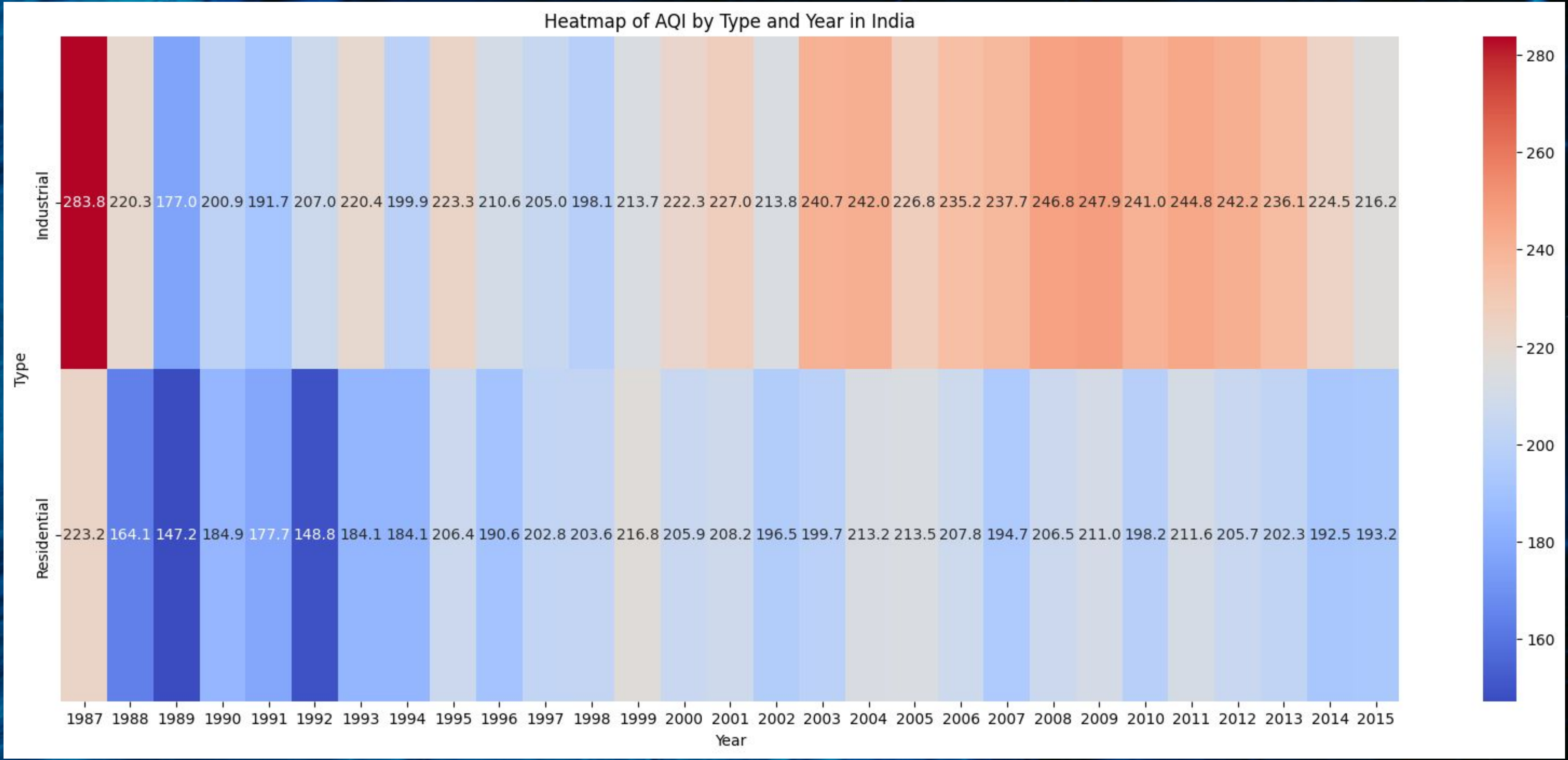
```
state       0
location    0
type        0
so2         0
no2         0
rspm        0
date        0
year        0
dtype: int64
```

Heatmap of AQI by State and Year in India



State	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Andhra Pradesh				318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2	318.2		186.7	169.5	199.4	188.6	187.8	186.7	148.7	176.9	168.2	158.7	157.7	155.0
Arunachal Pradesh																												135.6	168.8
Assam																		357.8	237.5	178.3	166.2	188.6	180.8	157.9		161.9	254.5	156.2	192.8
Bihar	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3	163.3		160.7	227.5	252.7	266.1	245.3	212.6	291.8	293.1	296.2			
Chandigarh			329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2		228.4	197.1	218.8	212.4	201.6	168.5	182.8	205.6	230.8	209.2	187.4	180.1
Chhattisgarh							316.9	316.9			316.9	316.9	316.9	316.9	316.9	316.9		326.0	256.8	262.3	256.5	271.4	283.6	279.7	253.4	224.0	258.0	211.9	230.8
Dadra & Nagar Haveli																					197.1	207.2	138.5	75.7	195.5	246.7			195.6
Daman & Diu			230.0	230.0	230.0	230.0	230.0	230.0	230.0												208.5	215.9	98.4	62.4	172.7	213.3			176.9
Delhi	322.3	322.3		322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	322.3	281.3	269.5	258.7	301.4	300.5	353.5	392.9		358.8	368.5	363.7	359.9	361.3
Goa	14.1	19.6	15.7	12.0	10.3	9.3												76.1	85.2	94.6	96.1	200.7	142.5	135.4	188.8	167.4	96.3	103.0	101.7
Gujarat	35.2	42.2	43.4	36.1	35.9	49.3	40.9	49.6	61.9	53.7	33.9	40.6	59.7	36.0	49.2	38.6		299.7	275.3	253.7	209.0	189.9	202.7	187.5	192.7	209.7	185.2	190.1	188.3
Haryana	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0	80.0		289.2	264.1	288.0	227.8	264.9	409.7	309.2	272.7	296.6	351.2	326.3	171.2
Himachal Pradesh	43.8	41.4	36.7	36.8	44.5	39.3			36.7	36.7	36.7	36.7	37.0	36.7	36.7	36.7		36.7	45.6	250.1	222.2	217.1	197.3	213.0	191.7	199.3	192.8	186.1	163.9
Jammu & Kashmir																							222.3	244.5	236.8	257.8	269.5	278.8	290.8
Jharkhand																		314.5	287.9	319.0	299.3	314.3	342.7	342.7	349.1	333.9	306.4	290.0	299.0
Karnataka	55.8	56.8	56.0	55.8	56.3	55.8	56.2	55.8	56.0	56.4	55.8	56.5	56.4	60.6	56.5	56.3		128.5	117.4	114.0	147.6	152.0	152.8		143.1	174.9	162.6	169.8	169.6
Kerala	85.0	85.0	85.0	85.4	85.3	85.1	85.8	86.2	87.0	85.5	85.0	85.0	85.2	86.9	85.0	85.0			139.0	131.1	96.9	86.4	83.2	74.9	74.4	101.1	105.2	95.6	80.9
Madhya Pradesh		18.8	33.3	32.9	28.1	28.4	29.4	23.7	38.2	33.9	37.2	33.2	31.1	36.2	34.0	29.7	0.0	166.5	205.8	199.2	207.3	261.6	259.0	249.7	253.0	254.6	281.8	251.8	224.0
Maharashtra	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	210.0	262.8	199.0	177.5	180.1	200.0	196.0	205.4	225.0	213.2	205.6	218.9	214.1
Manipur																					126.7	182.3							
Meghalaya																		99.1		114.3	124.2	143.0	163.7	210.2	174.5	143.7	117.0	111.1	114.9
Mizoram																			48.2	71.7	68.4	62.2		70.4	66.8	94.1	90.6	76.4	69.5
Nagaland																	156.3		177.6	151.2	137.7	151.1	169.0	149.5	173.3		193.6		216.0
Odisha	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8	303.8		148.3	160.0	161.3	157.6	175.7	177.6	175.4	188.2	179.7	179.8	186.5	185.9
Puducherry		70.5	76.7	68.3	68.3		68.4	74.5	88.4	80.0	83.1	87.2	79.2	68.3	68.3	68.3		94.2	95.3	111.0	89.3	87.3	67.4	64.2	70.7	67.6	71.4	68.7	58.3
Punjab	396.2	396.2	396.2	396.2	396.2	396.2			396.2	396.2	396.2		396.2	396.2	396.2	396.2	379.4	380.6	386.2	356.9	372.8	369.1	359.8	337.4	301.9	300.8	306.1	277.5	254.5
Rajasthan	330.3	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	329.2	220.2	212.7	236.1	228.4	251.2	250.2	289.4	295.9	301.3	308.1	296.3	296.6
Sikkim																					53.3								
Tamil Nadu	93.9	93.9	93.9	94.4	93.9	93.9	93.9	94.1	94.1	93.9	93.9	93.9	93.9	93.9	93.9	93.9		93.6	106.1	110.4	125.0	134.5	140.2	134.9	150.0	142.1	145.0	123.0	118.6
Telangana																												191.5	165.9
Uttar Pradesh	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0	370.0		303.6	311.1	325.7	317.3	332.7	338.9	334.8	313.3	343.9	338.5	331.5	337.8
Uttarakhand																		289.5						314.9	301.7	336.0	300.4		300.2
Uttaranchal																			308.9	257.1	222.0	293.7							
West Bengal	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4	486.4		235.4	221.7	210.1	188.7	196.5	263.0	205.0	221.2	204.9	288.9	225.5	211.1

Air Quality Index (AQI) trends for India based on types- industrial and residential.



These policies aim to regulate emissions and improve air quality over the years:

Industrial Pollution Policies

- National Emission Standards: Set limits for pollutants from industries, requiring pollution control technologies.
- Perform, Achieve, and Trade (PAT) Scheme: Encourages energy efficiency in industries through a trading system of energy savings certificates.

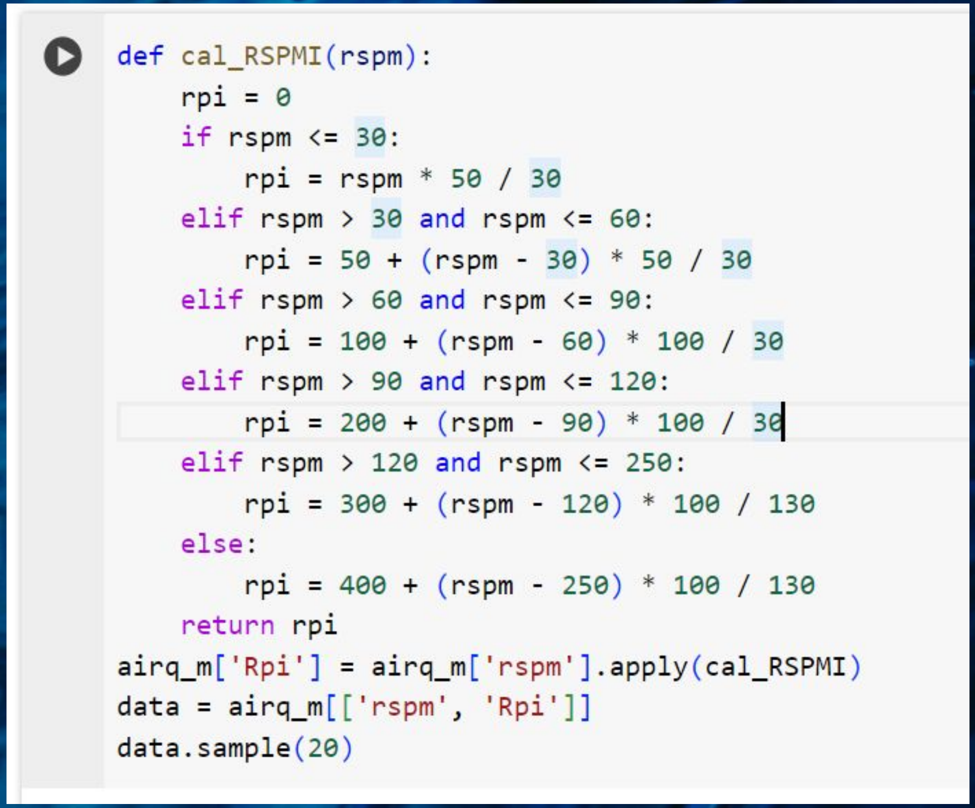
Residential Pollution Policies

1. Pradhan Mantri Ujjwala Yojana (PMUY): Provides LPG connections to households to replace polluting fuels.
2. National Biomass Cookstoves Initiative: Promotes cleaner cookstoves to reduce indoor air pollution.

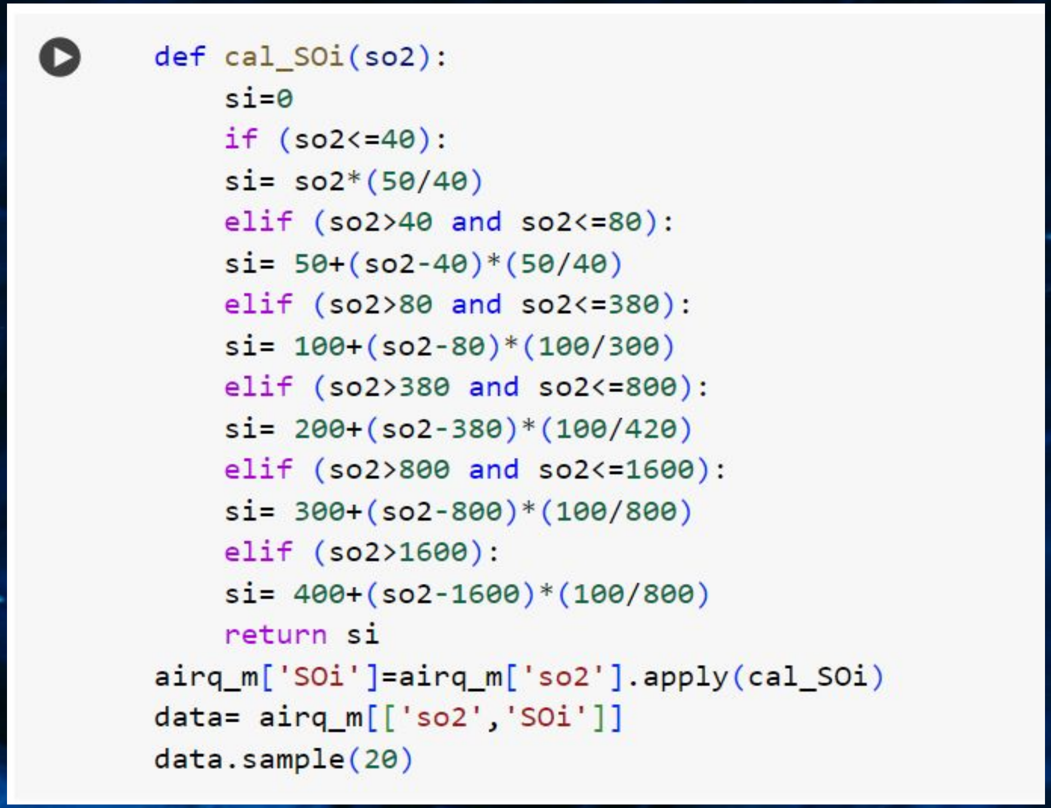
Overarching Policies Affecting Both Sectors

- National Clean Air Programme (NCAP): Introduced in 2019, the NCAP aims to reduce particulate matter (PM10 and PM2.5) levels by 20-30% by 2024 compared to 2017 levels in more than 100 non-attainment cities identified for poor air quality.
- Smart Cities Mission: This initiative indirectly contributes to reducing urban pollution by promoting sustainable and smart solutions in urban development, including waste management and sustainable urban transport

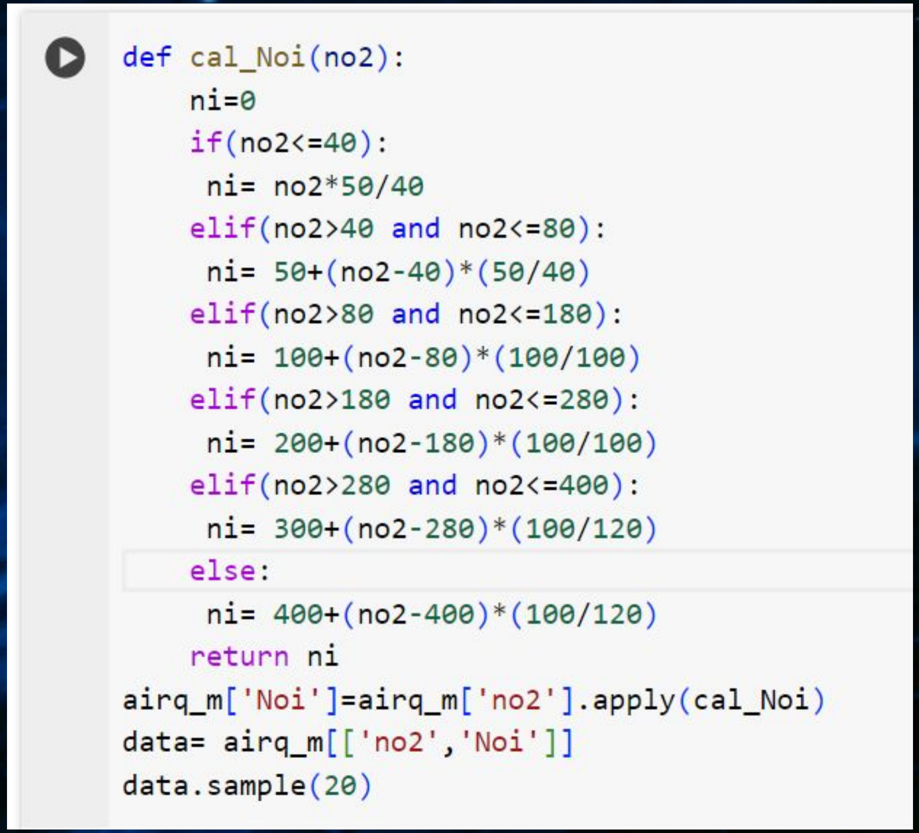
Introduction to SOi, NOi, RSPMi



```
def cal_RSPMI(rspm):
    rpi = 0
    if rspm <= 30:
        rpi = rspm * 50 / 30
    elif rspm > 30 and rspm <= 60:
        rpi = 50 + (rspm - 30) * 50 / 30
    elif rspm > 60 and rspm <= 90:
        rpi = 100 + (rspm - 60) * 100 / 30
    elif rspm > 90 and rspm <= 120:
        rpi = 200 + (rspm - 90) * 100 / 30
    elif rspm > 120 and rspm <= 250:
        rpi = 300 + (rspm - 120) * 100 / 130
    else:
        rpi = 400 + (rspm - 250) * 100 / 130
    return rpi
airq_m['Rpi'] = airq_m['rspm'].apply(cal_RSPMI)
data = airq_m[['rspm', 'Rpi']]
data.sample(20)
```

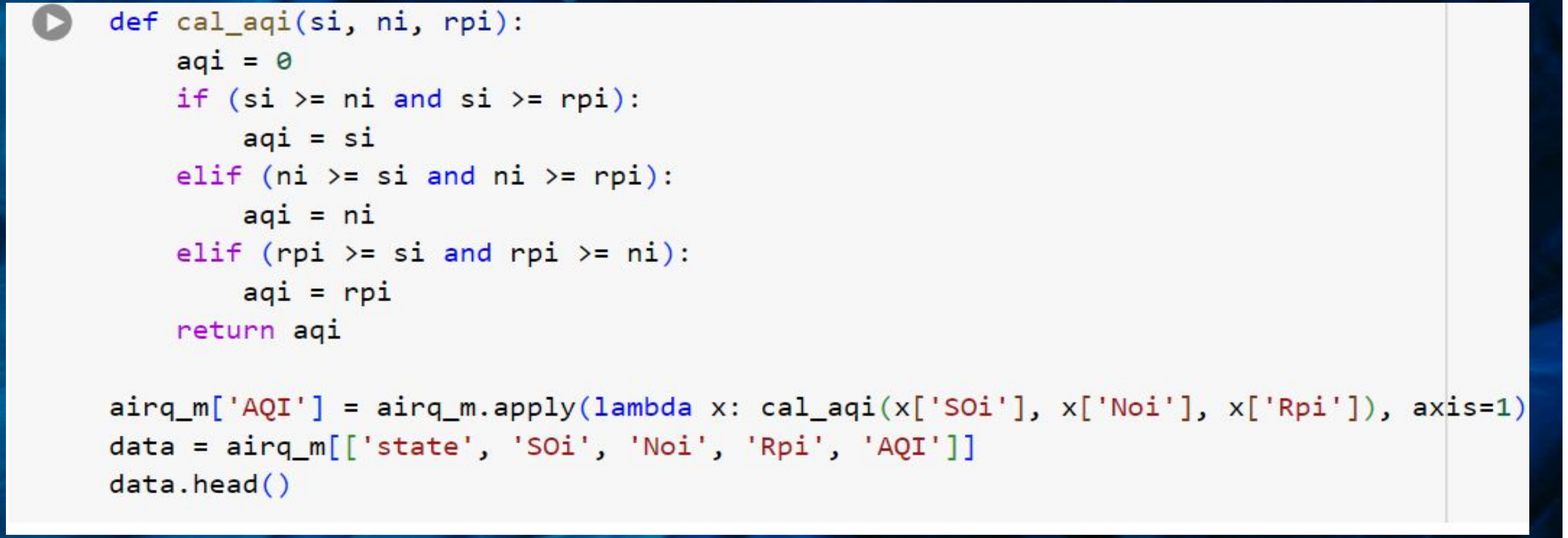


```
def cal_SOi(so2):
    si=0
    if (so2<=40):
    si= so2*(50/40)
    elif (so2>40 and so2<=80):
    si= 50+(so2-40)*(50/40)
    elif (so2>80 and so2<=380):
    si= 100+(so2-80)*(100/300)
    elif (so2>380 and so2<=800):
    si= 200+(so2-380)*(100/420)
    elif (so2>800 and so2<=1600):
    si= 300+(so2-800)*(100/800)
    elif (so2>1600):
    si= 400+(so2-1600)*(100/800)
    return si
airq_m['SOi']=airq_m['so2'].apply(cal_SOi)
data= airq_m[['so2','SOi']]
data.sample(20)
```

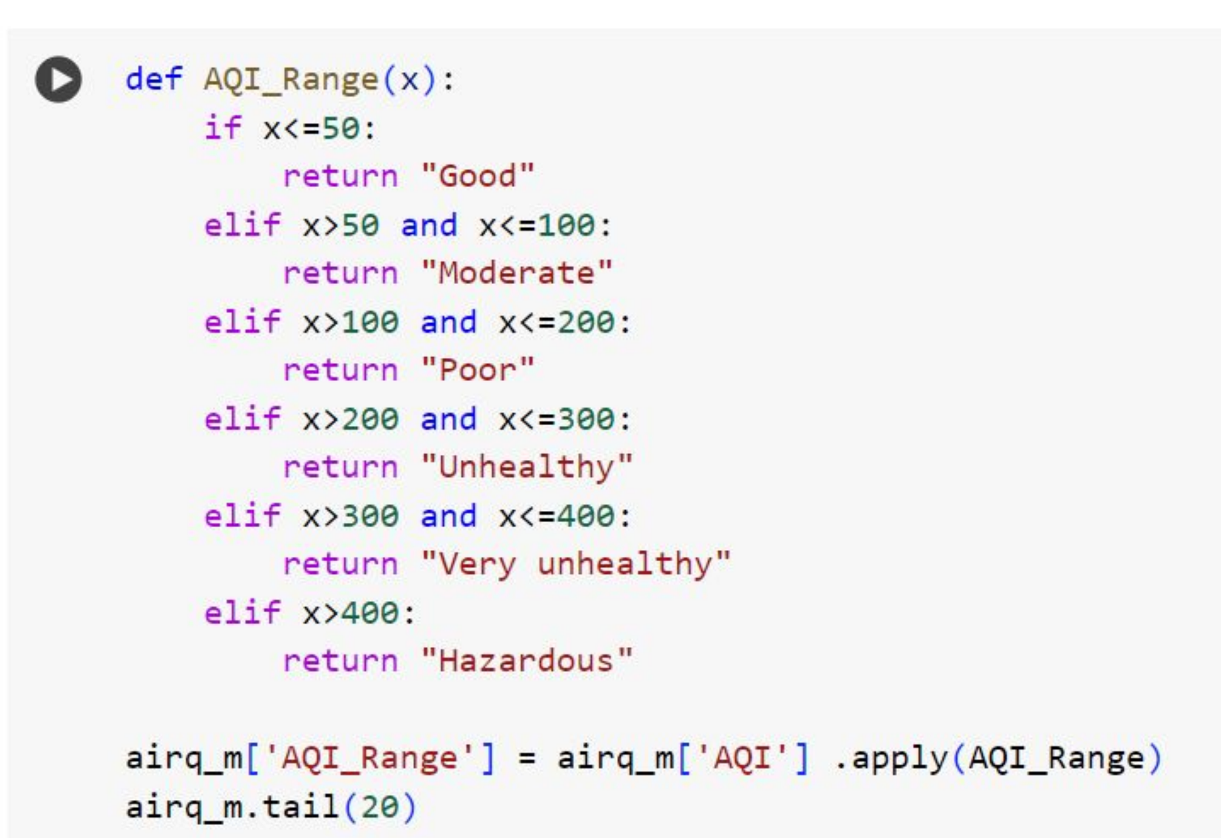


```
def cal_Noi(no2):
    ni=0
    if(no2<=40):
     ni= no2*50/40
    elif(no2>40 and no2<=80):
     ni= 50+(no2-40)*(50/40)
    elif(no2>80 and no2<=180):
     ni= 100+(no2-80)*(100/100)
    elif(no2>180 and no2<=280):
     ni= 200+(no2-180)*(100/100)
    elif(no2>280 and no2<=400):
     ni= 300+(no2-280)*(100/120)
    else:
     ni= 400+(no2-400)*(100/120)
    return ni
airq_m['Noi']=airq_m['no2'].apply(cal_Noi)
data= airq_m[['no2','Noi']]
data.sample(20)
```

Calculation of AQI - Air Quality Index

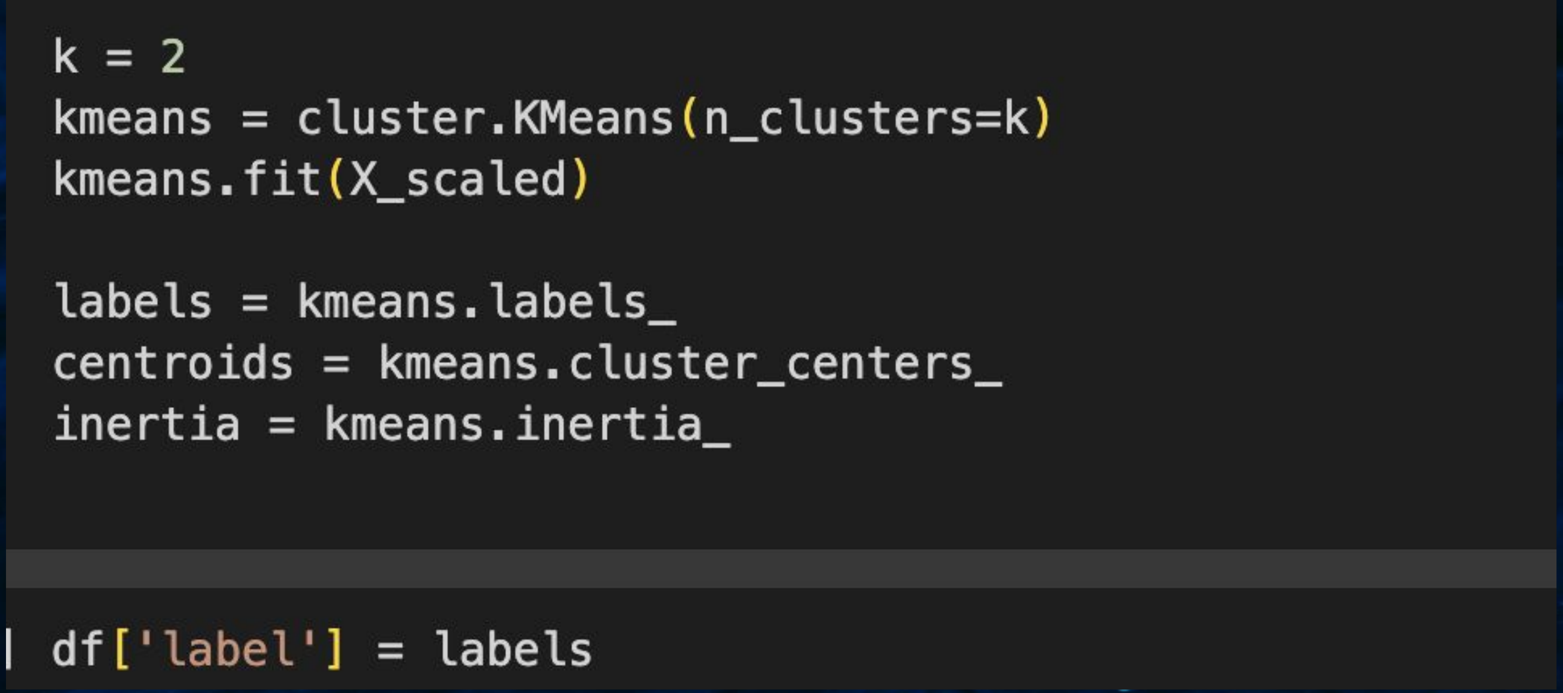


```python
def cal_aqi(si, ni, rpi):
    aqi = 0
    if (si >= ni and si >= rpi):
        aqi = si
    elif (ni >= si and ni >= rpi):
        aqi = ni
    elif (rpi >= si and rpi >= ni):
        aqi = rpi
    return aqi

airq_m['AQI'] = airq_m.apply(lambda x: cal_aqi(x['SOi'], x['Noi'], x['Rpi']), axis=1)
data = airq_m[['state', 'SOi', 'Noi', 'Rpi', 'AQI']]
data.head()
```



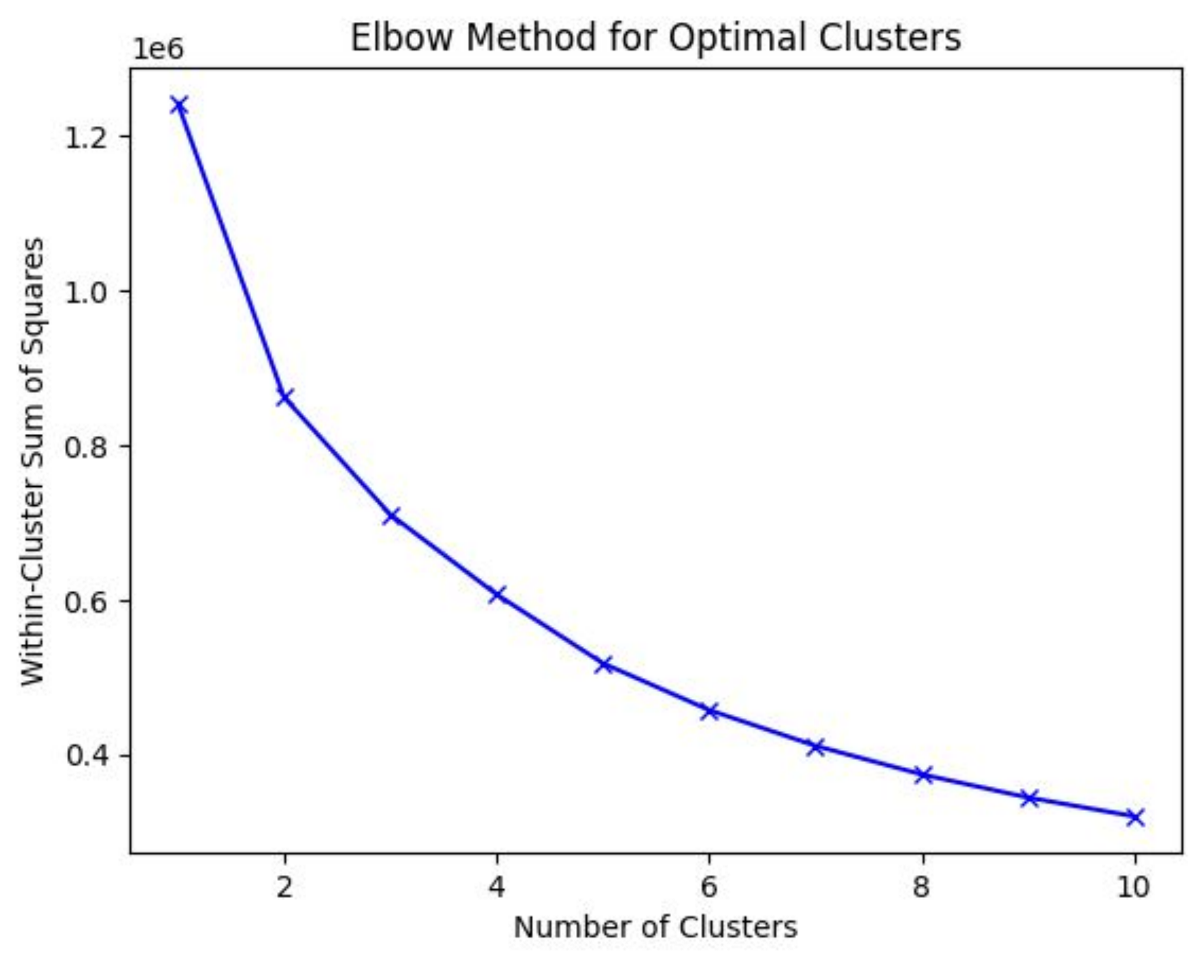
```python
def AQI_Range(x):
    if x<=50:
        return "Good"
    elif x>50 and x<=100:
        return "Moderate"
    elif x>100 and x<=200:
        return "Poor"
    elif x>200 and x<=300:
        return "Unhealthy"
    elif x>300 and x<=400:
        return "Very unhealthy"
    elif x>400:
        return "Hazardous"

airq_m['AQI_Range'] = airq_m['AQI'] .apply(AQI_Range)
airq_m.tail(20)
```

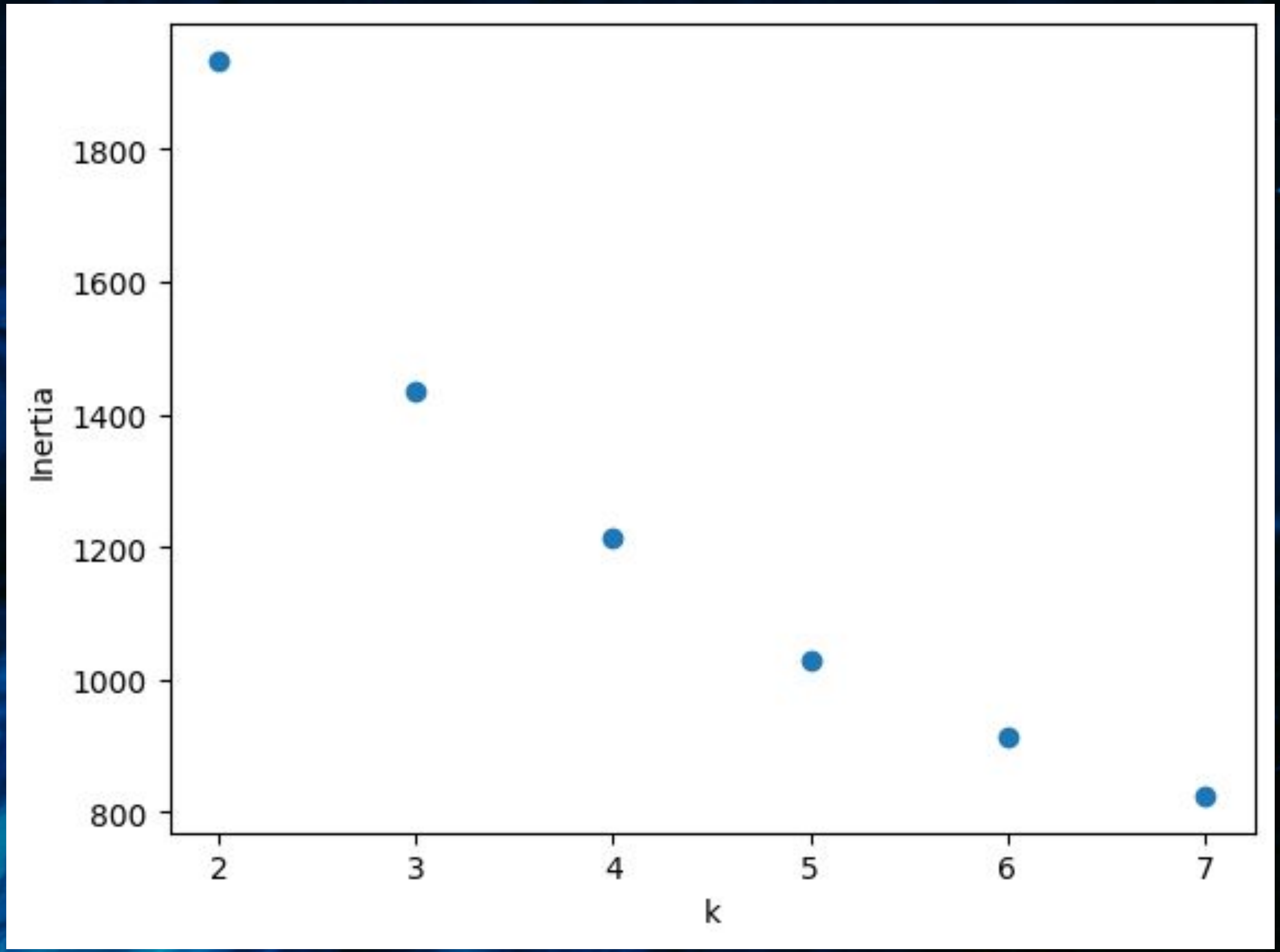
K-means Clustering:



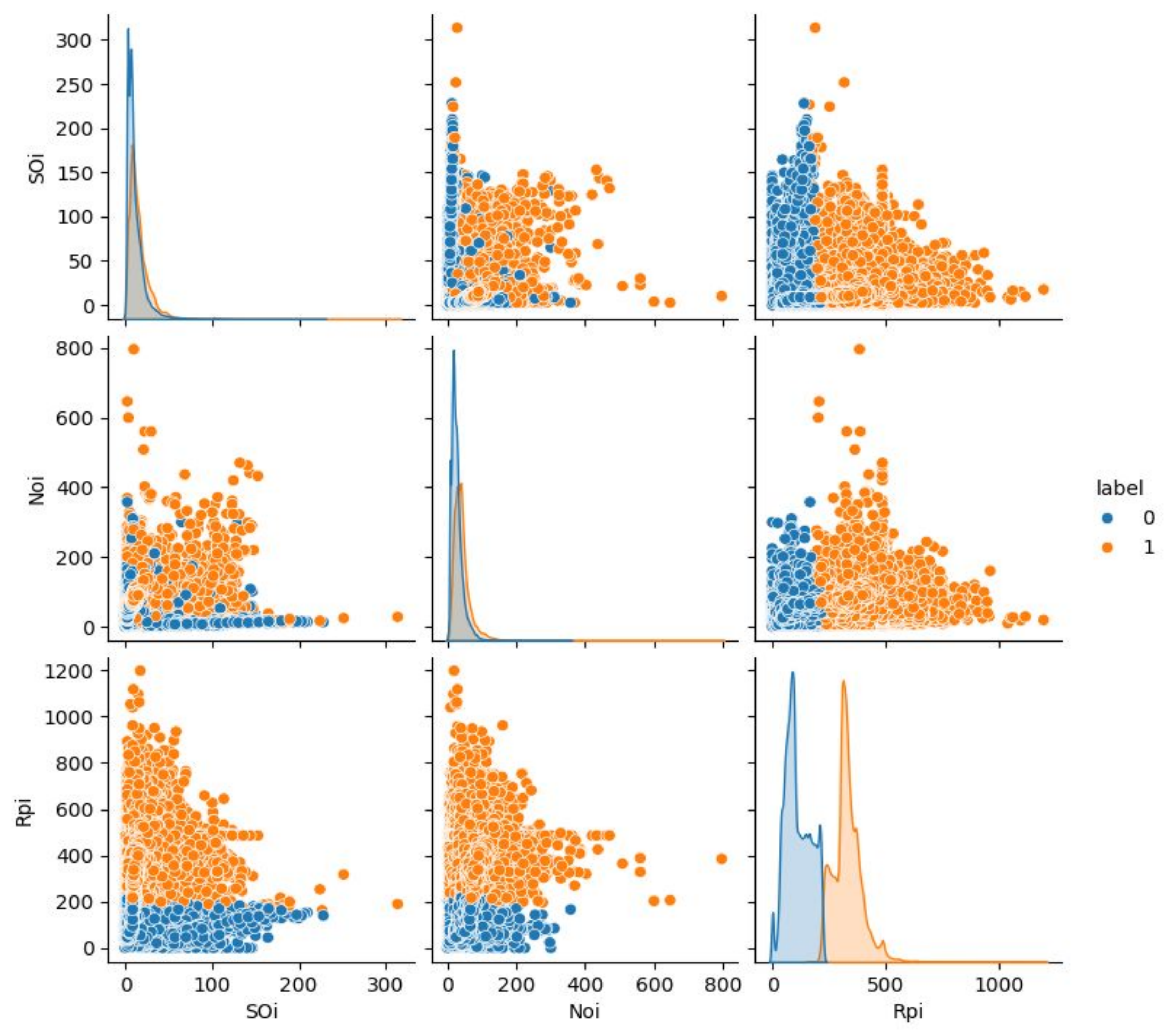
```
k = 2
kmeans = cluster.KMeans(n_clusters=k)
kmeans.fit(X_scaled)

labels = kmeans.labels_
centroids = kmeans.cluster_centers_
inertia = kmeans.inertia_
```

```
df['label'] = labels
```

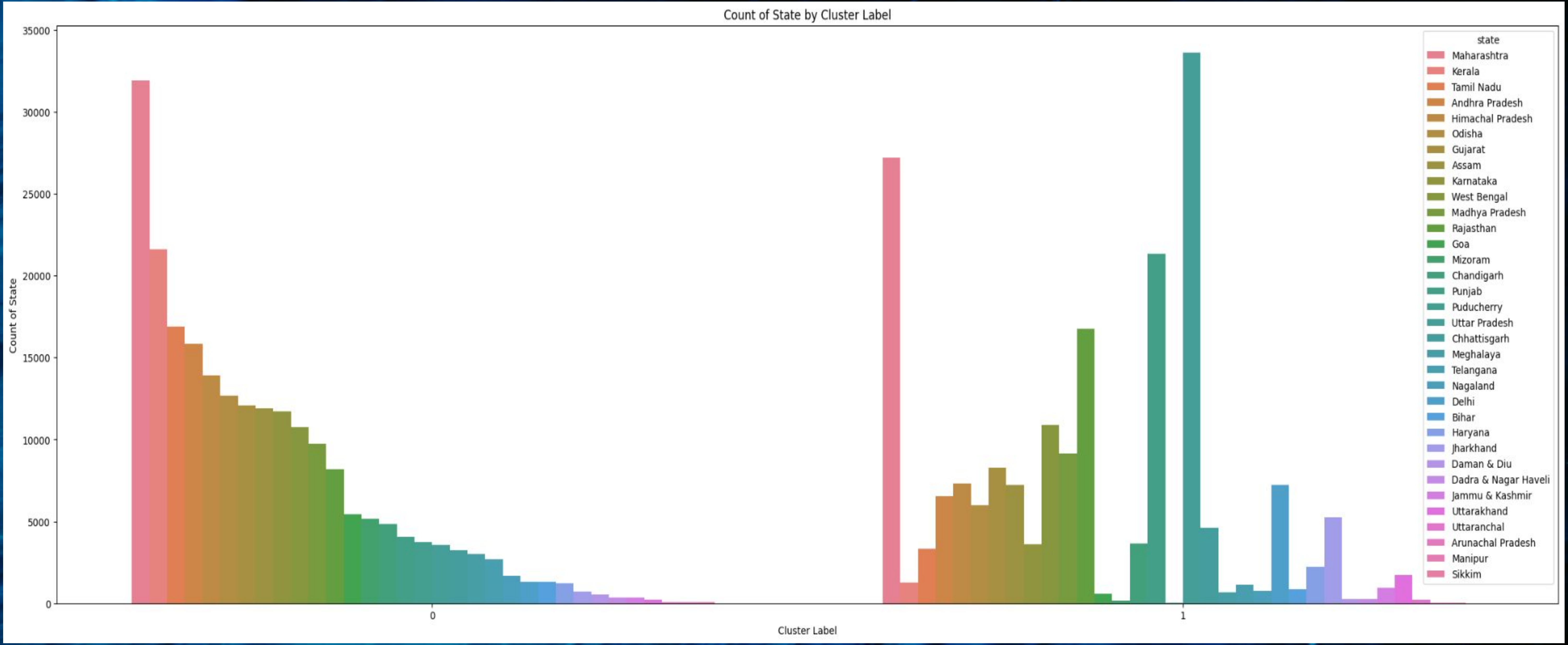


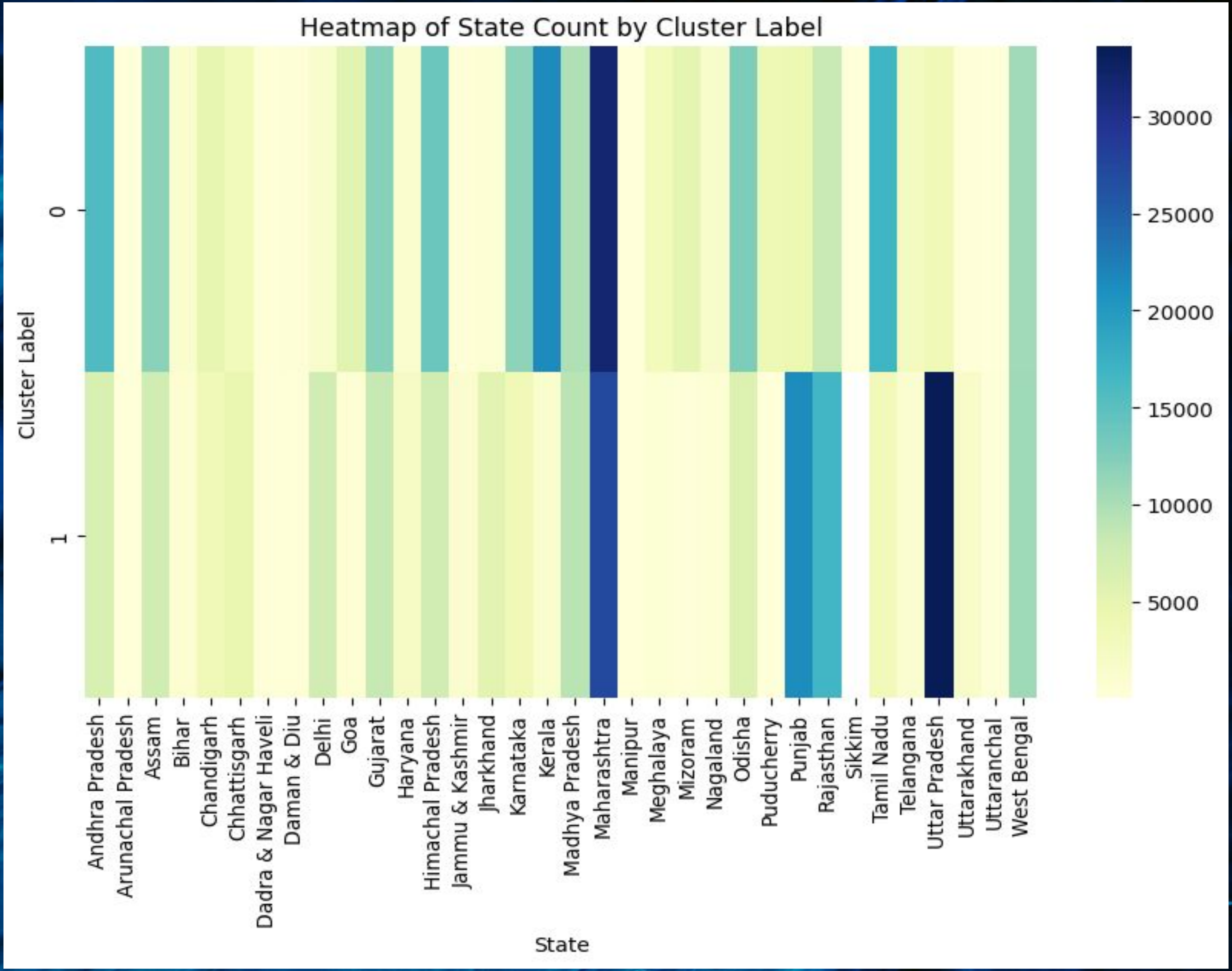


K-means clustering on SOI, NOI, RPI using two labels:

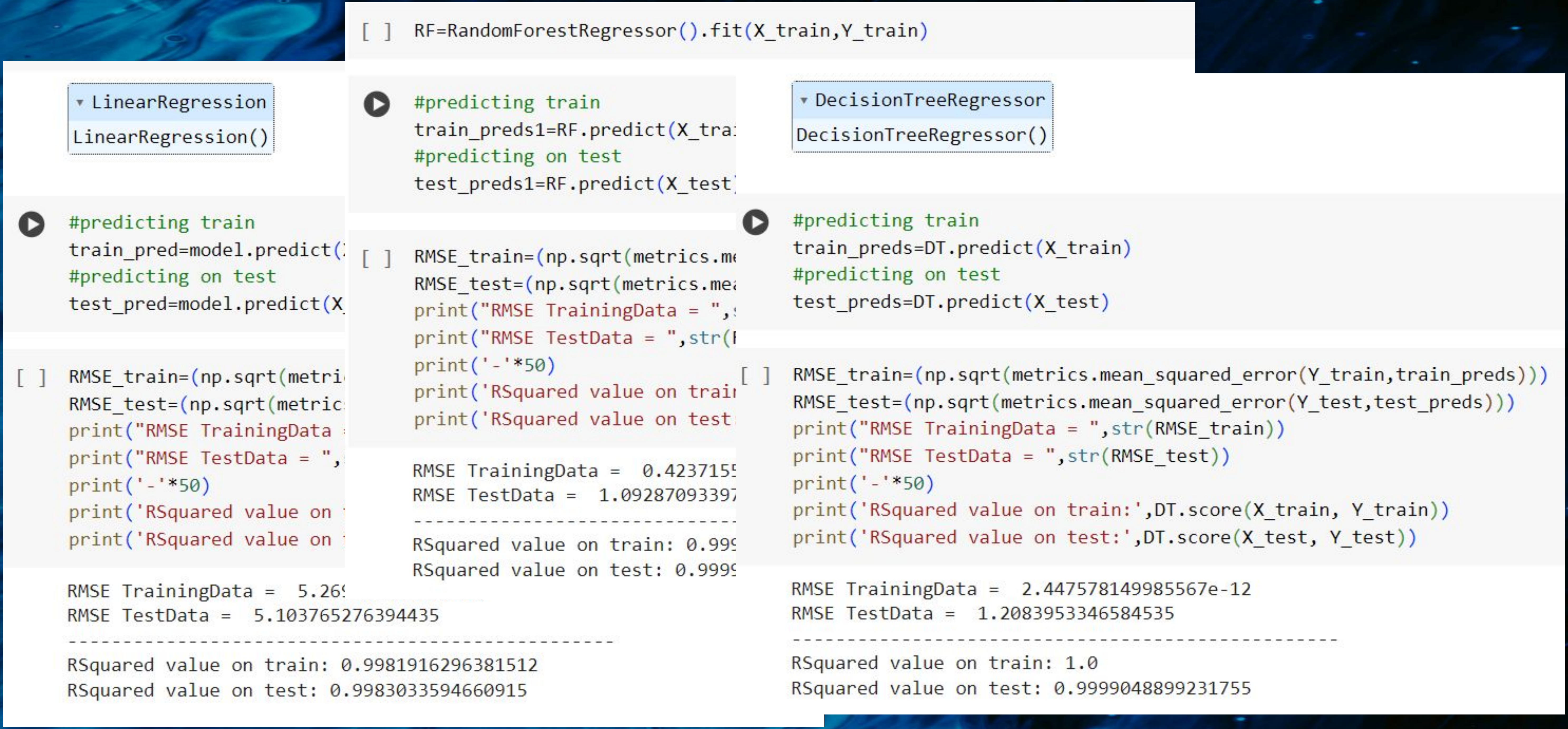


Count of state by Cluster Label Graph:




Heatmap of State Count by Cluster Label
Cluster Label
0
1
State
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chandigarh
Chhattisgarh
Dadra & Nagar Haveli
Daman & Diu
Delhi
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu & Kashmir
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Puducherry
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Uttar Pradesh
Uttarakhand
Uttaranchal
West Bengal
30000
25000
20000
15000
10000
5000

ML models used to Train



```
▾ LinearRegression
LinearRegression()
```

```
#predicting train
train_pred=model.predict(
#predicting on test
test_pred=model.predict(X
```

```
[ ]  RMSE_train=(np.sqrt(metri
     RMSE_test=(np.sqrt(metric
     print("RMSE TrainingData
     print("RMSE TestData = ",
     print('-'*50)
     print('RSquared value on
     print('RSquared value on
```

```
RMSE TrainingData =  5.26
RMSE TestData =  5.103765276394435
--------------------------------------------------
RSquared value on train: 0.9981916296381512
RSquared value on test: 0.9983033594660915
```

```
[ ]  RF=RandomForestRegressor().fit(X_train,Y_train)
```

```
#predicting train
train_preds1=RF.predict(X_tra
#predicting on test
test_preds1=RF.predict(X_test)
```

```
[ ]  RMSE_train=(np.sqrt(metrics.me
     RMSE_test=(np.sqrt(metrics.mea
     print("RMSE TrainingData = ",
     print("RMSE TestData = ",str(
     print('-'*50)
     print('RSquared value on trai
     print('RSquared value on test
```

```
RMSE TrainingData =  0.4237155
RMSE TestData =  1.09287093397
------------------------------
RSquared value on train: 0.999
RSquared value on test: 0.9999
```

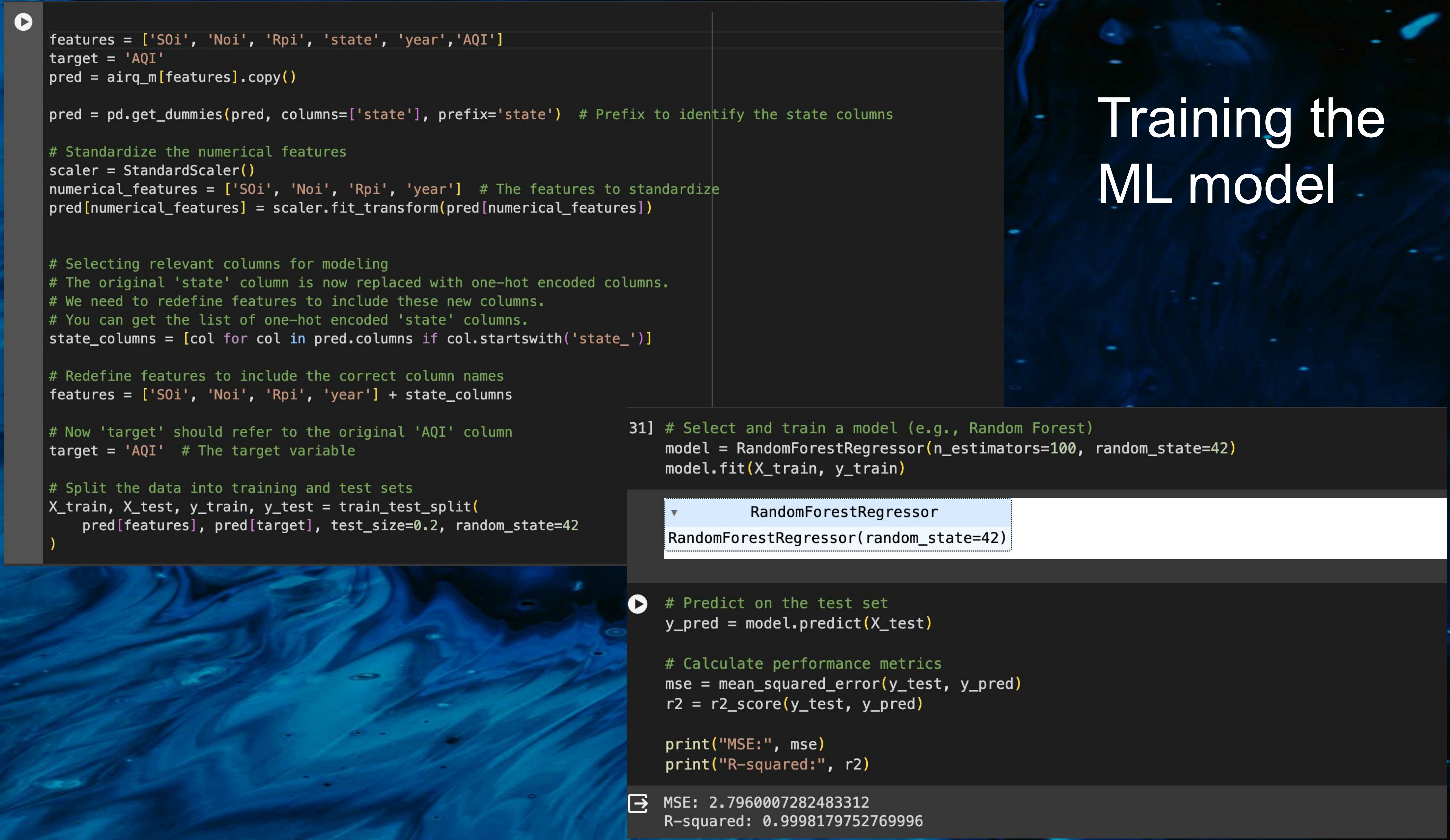
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
#predicting train
train_preds=DT.predict(X_train)
#predicting on test
test_preds=DT.predict(X_test)
```

```
[ ]  RMSE_train=(np.sqrt(metrics.mean_squared_error(Y_train,train_preds)))
     RMSE_test=(np.sqrt(metrics.mean_squared_error(Y_test,test_preds)))
     print("RMSE TrainingData = ",str(RMSE_train))
     print("RMSE TestData = ",str(RMSE_test))
     print('-'*50)
     print('RSquared value on train:',DT.score(X_train, Y_train))
     print('RSquared value on test:',DT.score(X_test, Y_test))
```

```
RMSE TrainingData =  2.4475781499855567e-12
RMSE TestData =  1.2083953346584535
--------------------------------------------------
RSquared value on train: 1.0
RSquared value on test: 0.9999048899231755
```

Training the ML model



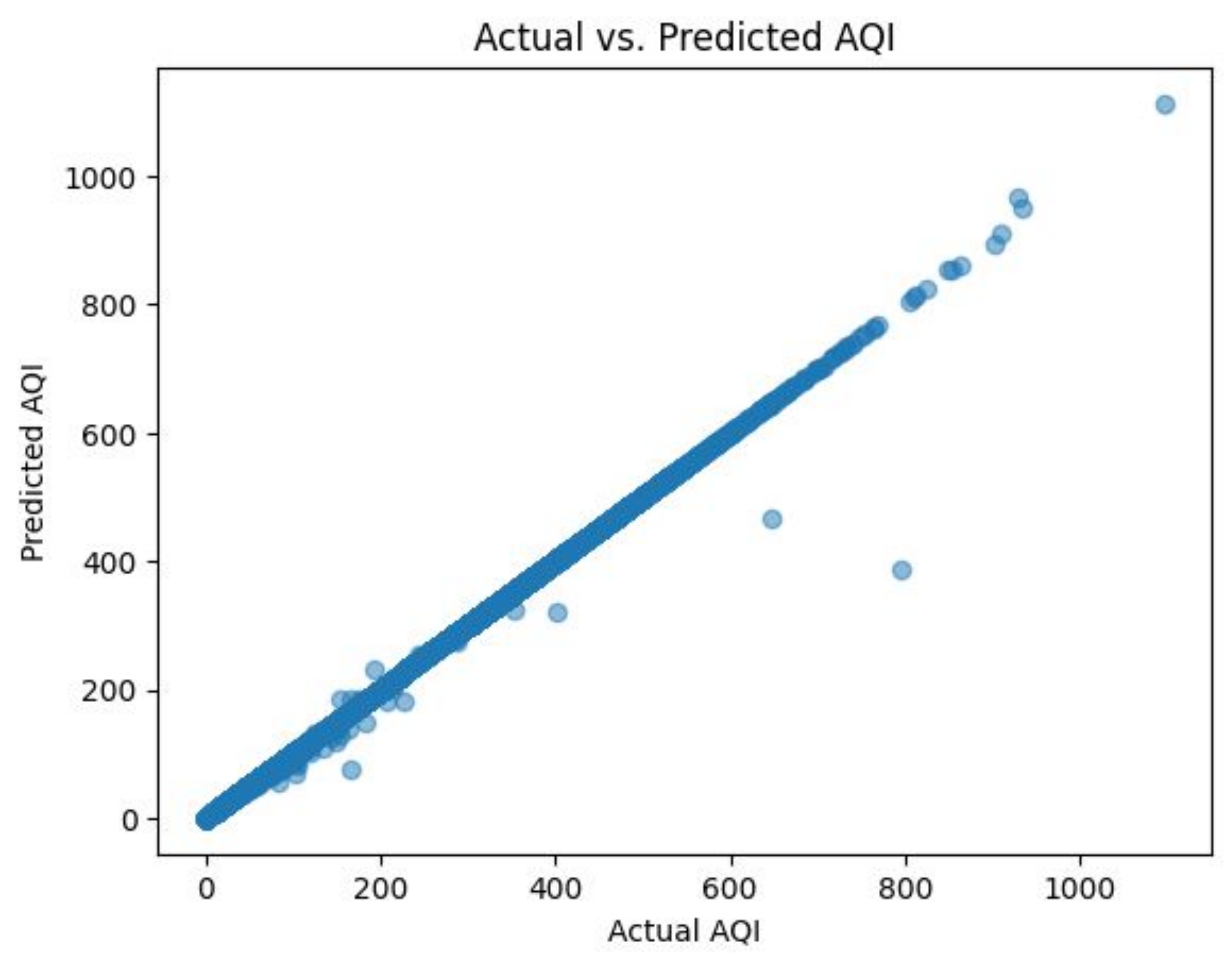
```
features = ['SOi', 'Noi', 'Rpi', 'state', 'year','AQI']
target = 'AQI'
pred = airq_m[features].copy()

pred = pd.get_dummies(pred, columns=['state'], prefix='state')  # Prefix to identify the state columns

# Standardize the numerical features
scaler = StandardScaler()
numerical_features = ['SOi', 'Noi', 'Rpi', 'year']  # The features to standardize
pred[numerical_features] = scaler.fit_transform(pred[numerical_features])


# Selecting relevant columns for modeling
# The original 'state' column is now replaced with one-hot encoded columns.
# We need to redefine features to include these new columns.
# You can get the list of one-hot encoded 'state' columns.
state_columns = [col for col in pred.columns if col.startswith('state_')]

# Redefine features to include the correct column names
features = ['SOi', 'Noi', 'Rpi', 'year'] + state_columns

# Now 'target' should refer to the original 'AQI' column
target = 'AQI'  # The target variable

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    pred[features], pred[target], test_size=0.2, random_state=42
)
```

```
31] # Select and train a model (e.g., Random Forest)
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
```

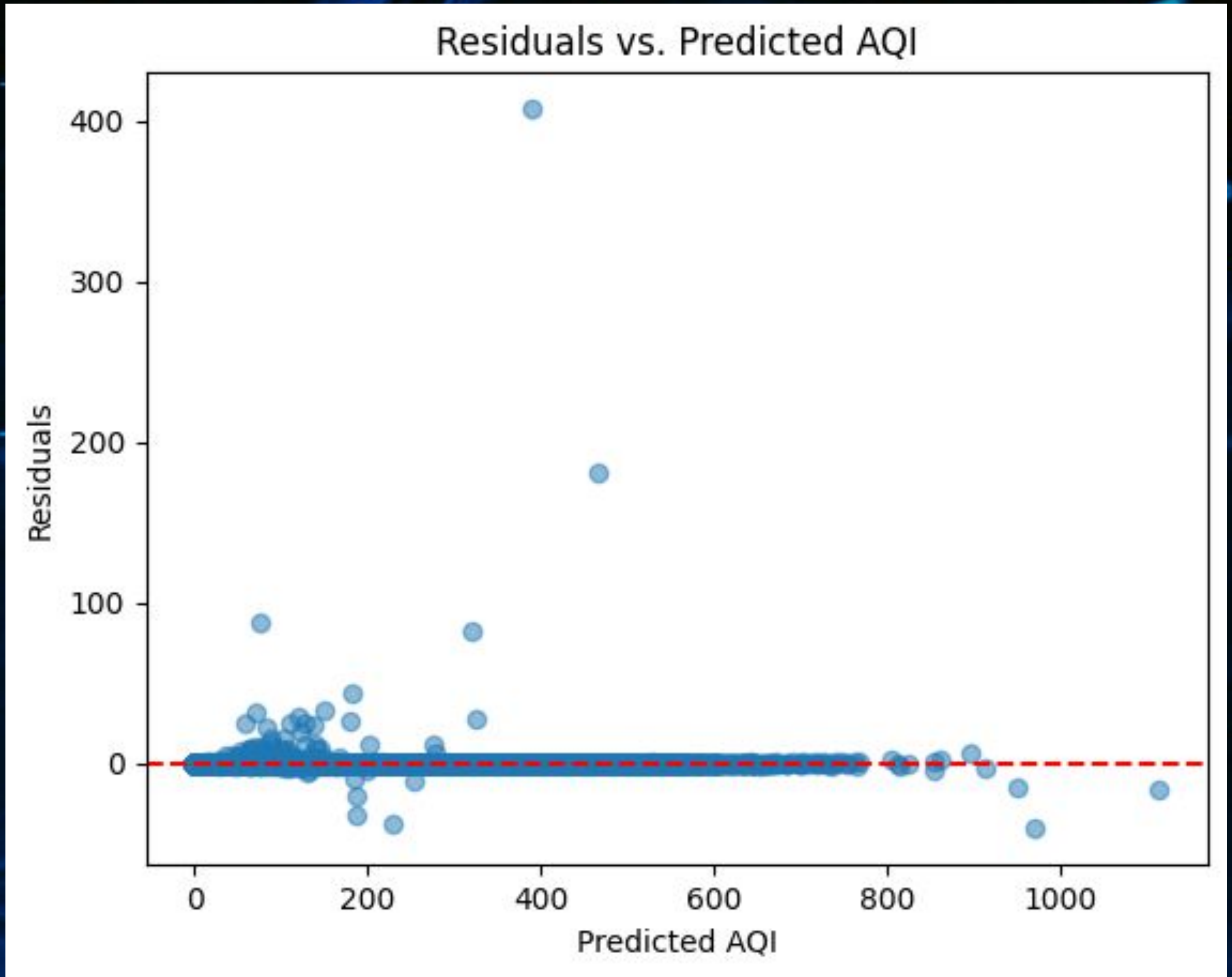
```
▾        RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
# Predict on the test set
y_pred = model.predict(X_test)

# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("MSE:", mse)
print("R-squared:", r2)
```

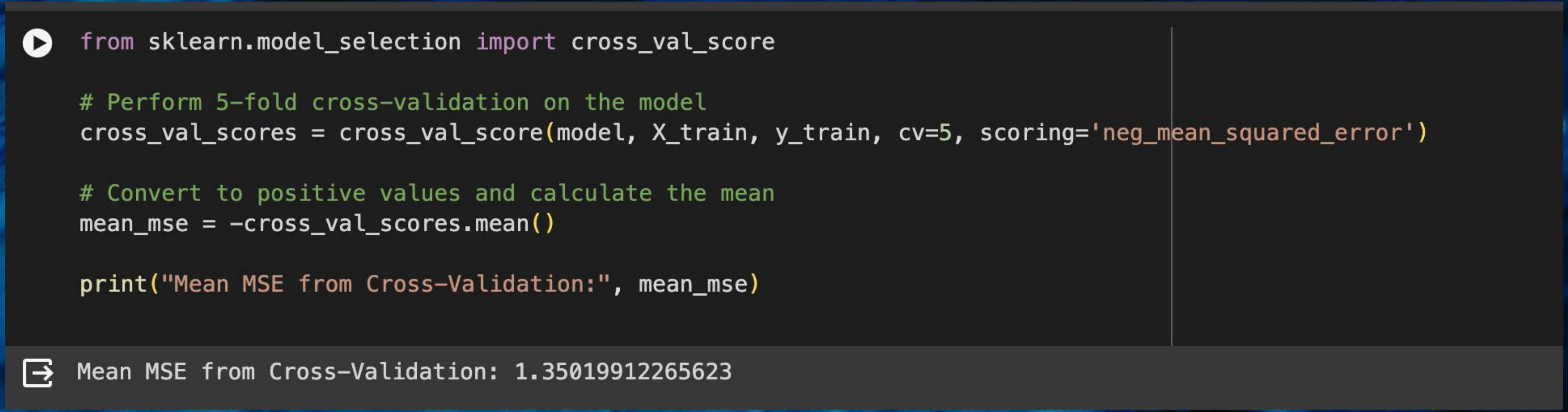
```
MSE: 2.7960007282483312
R-squared: 0.9998179752769996
```



Actual vs. Predicted AQI



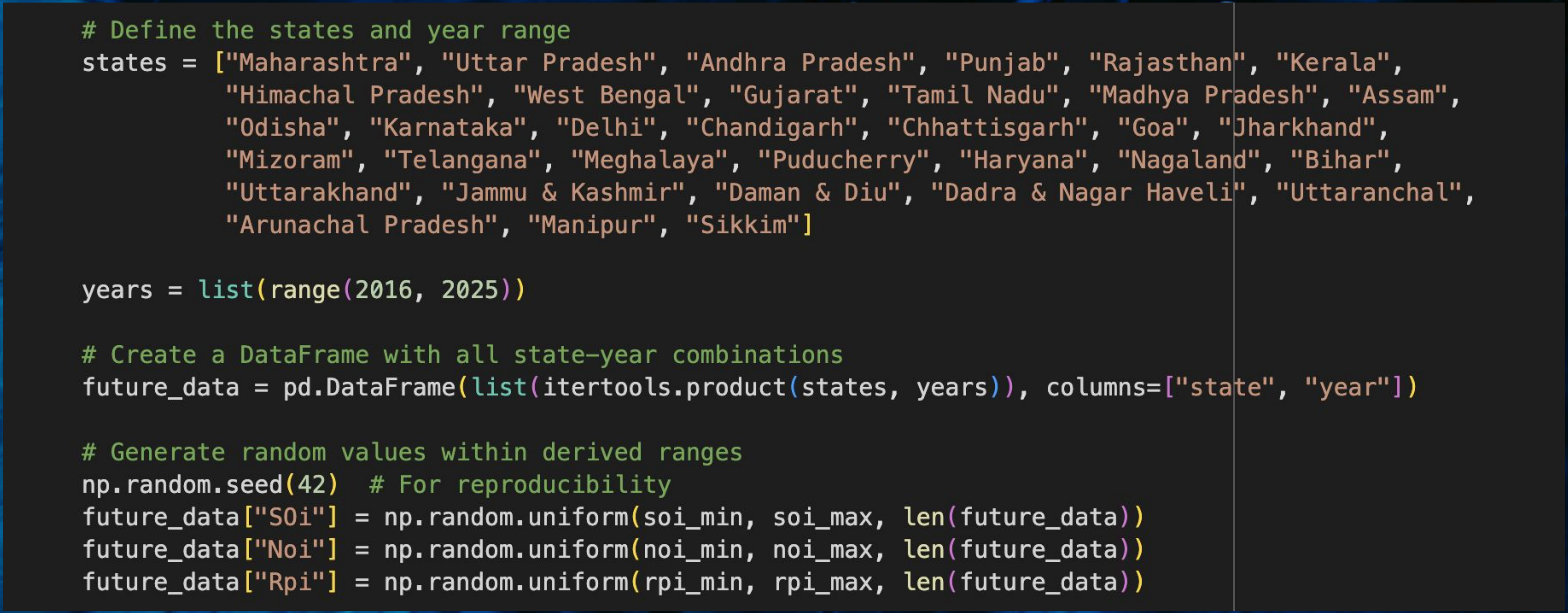
Residuals vs. Predicted AQI



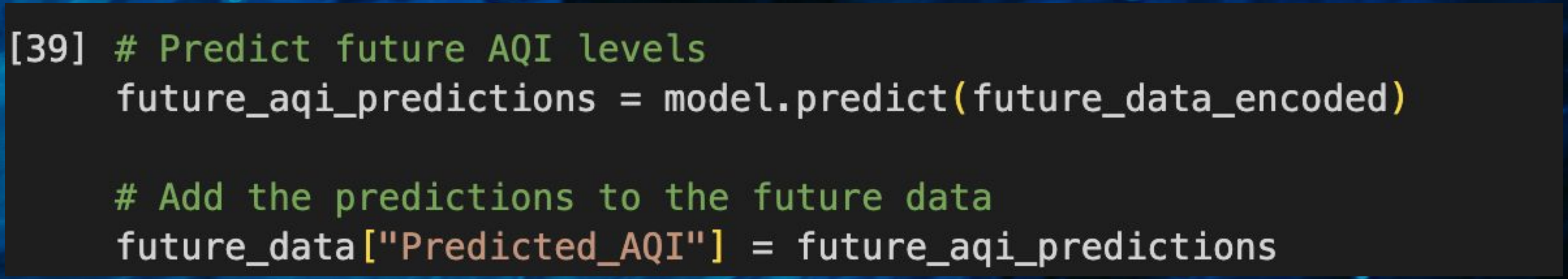
```
from sklearn.model_selection import cross_val_score

# Perform 5-fold cross-validation on the model
cross_val_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_squared_error')

# Convert to positive values and calculate the mean
mean_mse = -cross_val_scores.mean()

print("Mean MSE from Cross-Validation:", mean_mse)
```

```
Mean MSE from Cross-Validation: 1.35019912265623
```

Developing the future data for the model to work on:

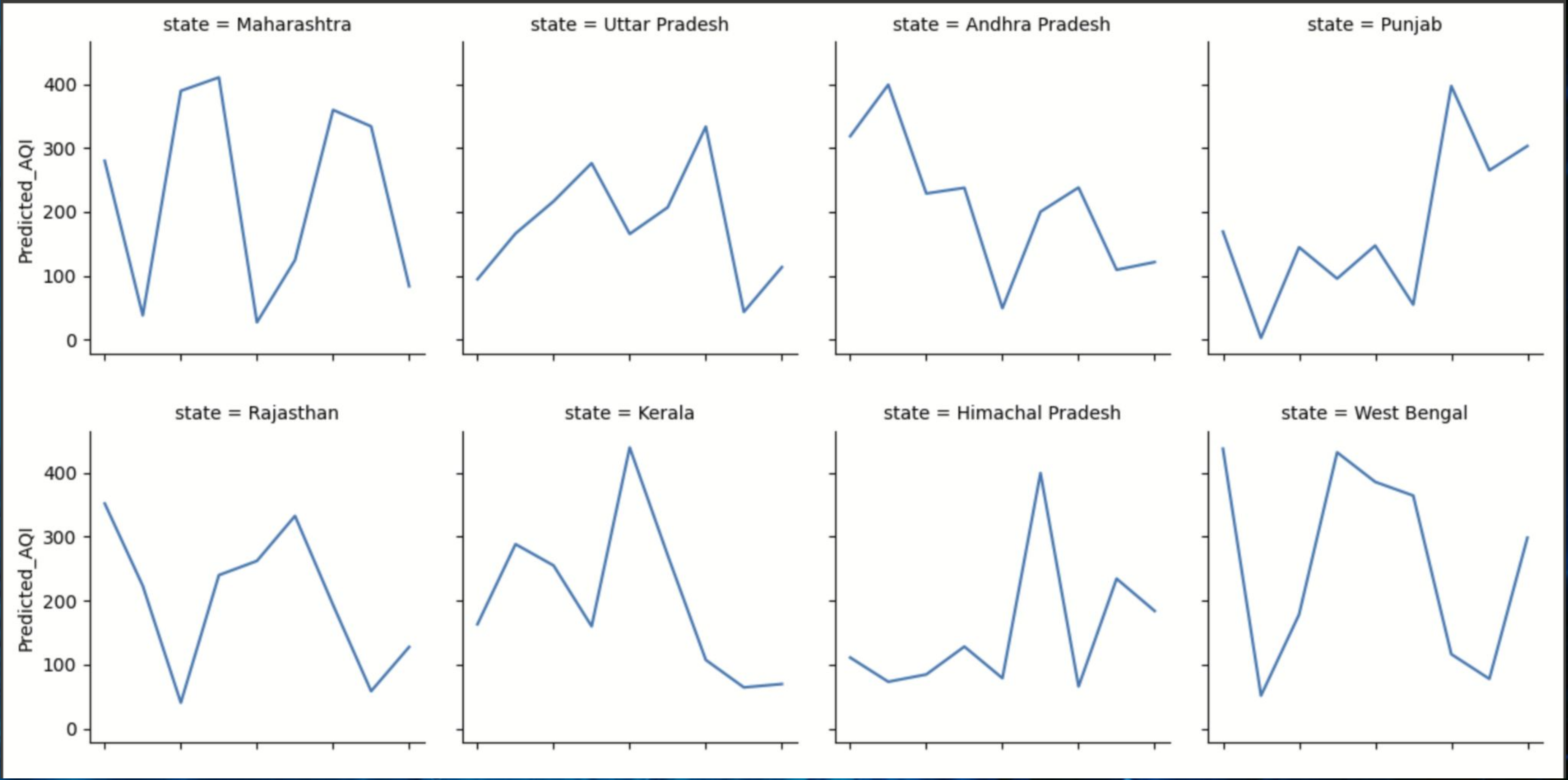


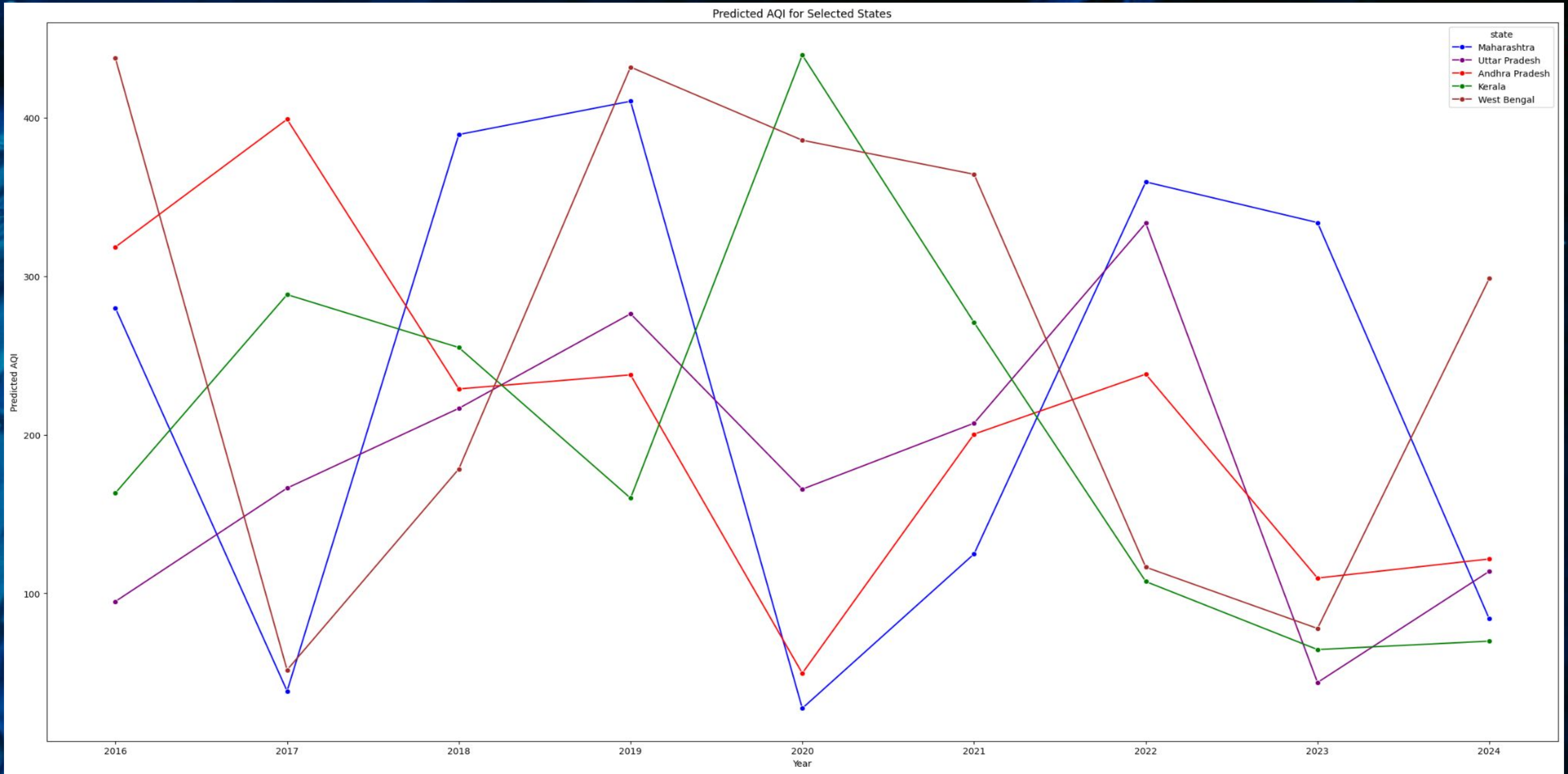
```
# Define the states and year range
states = ["Maharashtra", "Uttar Pradesh", "Andhra Pradesh", "Punjab", "Rajasthan", "Kerala",
          "Himachal Pradesh", "West Bengal", "Gujarat", "Tamil Nadu", "Madhya Pradesh", "Assam",
          "Odisha", "Karnataka", "Delhi", "Chandigarh", "Chhattisgarh", "Goa", "Jharkhand",
          "Mizoram", "Telangana", "Meghalaya", "Puducherry", "Haryana", "Nagaland", "Bihar",
          "Uttarakhand", "Jammu & Kashmir", "Daman & Diu", "Dadra & Nagar Haveli", "Uttaranchal",
          "Arunachal Pradesh", "Manipur", "Sikkim"]

years = list(range(2016, 2025))

# Create a DataFrame with all state-year combinations
future_data = pd.DataFrame(list(itertools.product(states, years)), columns=["state", "year"])

# Generate random values within derived ranges
np.random.seed(42)  # For reproducibility
future_data["SOi"] = np.random.uniform(soi_min, soi_max, len(future_data))
future_data["Noi"] = np.random.uniform(noi_min, noi_max, len(future_data))
future_data["Rpi"] = np.random.uniform(rpi_min, rpi_max, len(future_data))
```

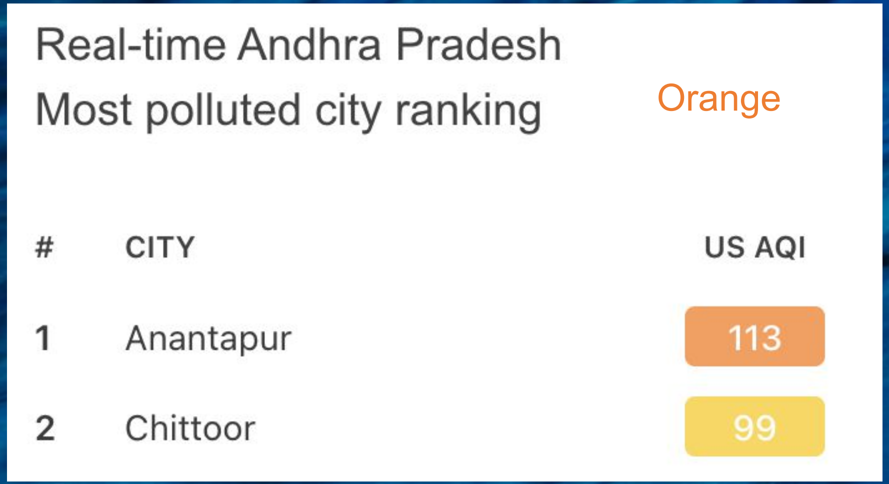


```
[39] # Predict future AQI levels
     future_aqi_predictions = model.predict(future_data_encoded)

     # Add the predictions to the future data
     future_data["Predicted_AQI"] = future_aqi_predictions
```

Predicted Values for Individual States from 2016-2024



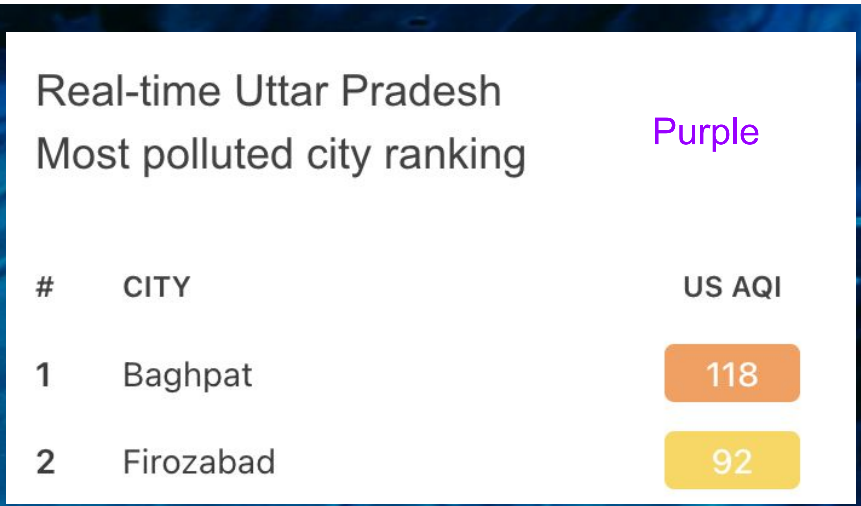




Real-time Andhra Pradesh Most polluted city ranking

Orange

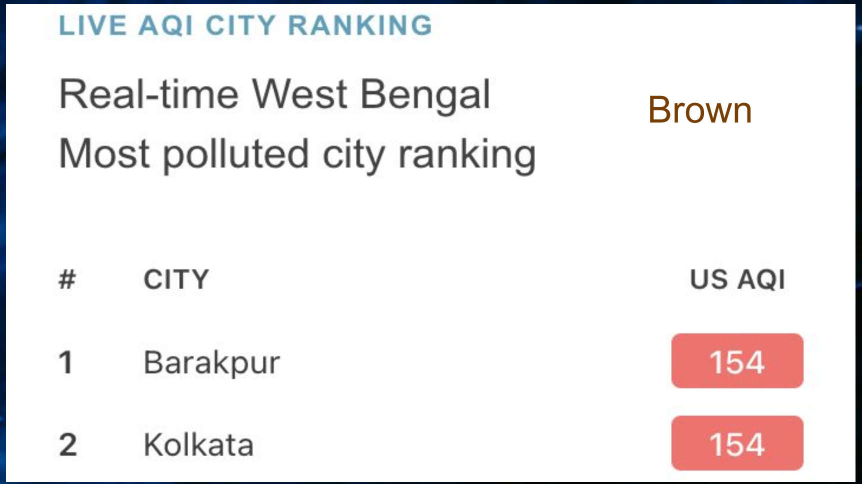
#	CITY	US AQI
1	Anantapur	113
2	Chittoor	99



Real-time Uttar Pradesh Most polluted city ranking

Purple

#	CITY	US AQI
1	Baghpat	118
2	Firozabad	92

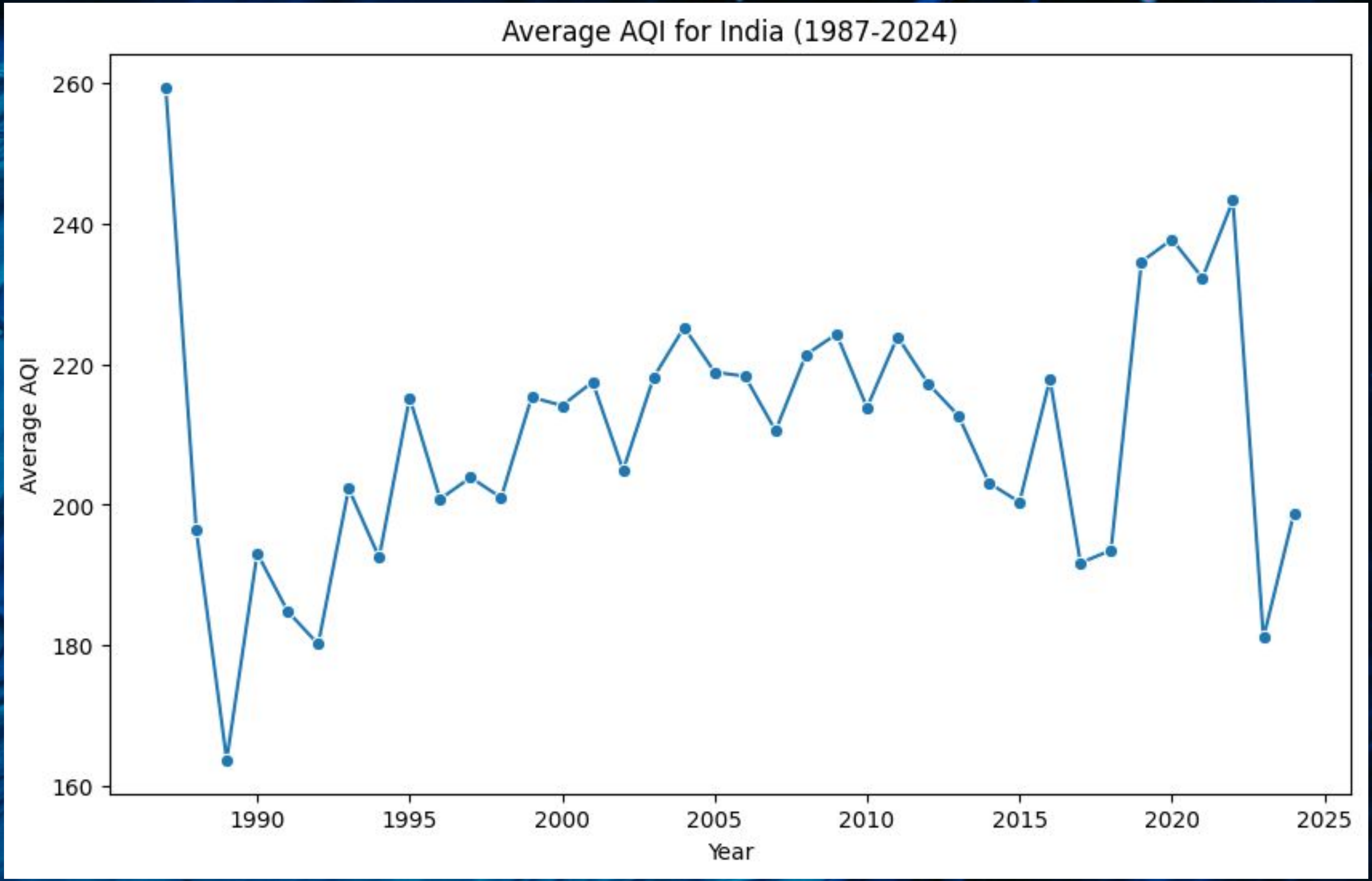


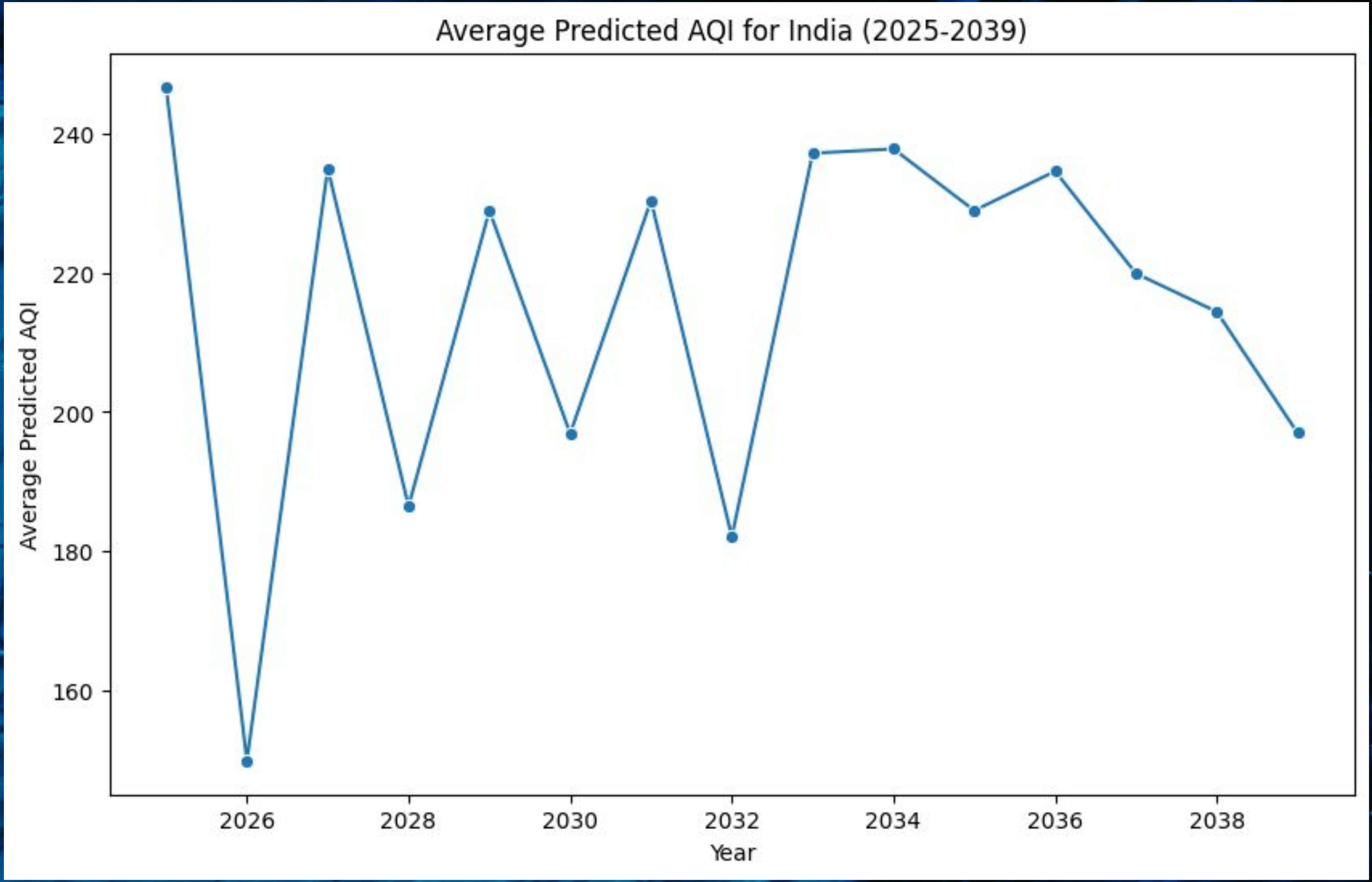
LIVE AQI CITY RANKING

Real-time West Bengal Most polluted city ranking

Brown

#	CITY	US AQI
1	Barakpur	154
2	Kolkata	154


Average AQI for India (1987-2024)
Average AQI
260
240
220
200
180
160
1990
1995
2000
2005
2010
2015
2020
2025
Year



Average Predicted AQI for India (2025-2039)

Conclusion and Future Directions

In conclusion, the application of data science and machine learning algorithms has proven to be a powerful approach for effective air quality control. By leveraging predictive modeling, anomaly detection, and optimization techniques, we can better monitor, forecast, and optimize air quality. **Looking ahead**, integrating air quality data with IoT sensor networks will enable real-time, hyper-local insights to drive even more precision in air quality management.

Thank You