Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

Air Quality Predictor

Supervised By:

Mrs. Rishu Taneja

Submitted By:

Tannishtha Jain

2210990903

Group -13

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

Air pollution remains a pressing global issue with significant implications for public health and environmental sustainability. To address this challenge, this project focuses on developing a robust predictive model for forecasting Air Quality Index (AQI) levels, leveraging the power of Machine Learning (ML) techniques.

The project adopts a data-driven approach, utilizing a diverse dataset encompassing meteorological parameters, pollutant concentrations, geographical features, and historical AQI measurements. Through meticulous data preprocessing and feature engineering, informative features are extracted to capture the complex interplay between various environmental factors and air quality.

Several ML algorithms are employed to train and validate the predictive model, including ensemble methods such as Random Forest, Gradient Boosting, and Support Vector Regression, as well as deep learning techniques like Convolutional Neural Networks (CNNs) for spatial data analysis.

Performance evaluation of the model is conducted using established metrics such as mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE). Additionally, cross-validation techniques and hyperparameter tuning are utilized to ensure the model's robustness and generalizability across different regions and time periods.

The developed AQI prediction model holds immense potential for informing policy decisions, urban planning initiatives, and public health interventions aimed at mitigating the adverse effects of air pollution. By harnessing ML techniques, this project contributes to advancing environmental monitoring capabilities and promoting sustainable development practices.

Table of Contents

Introduction

Background

Air quality index (AQI) prediction is a crucial aspect of environmental monitoring and public health management, providing insights into the quality of the air we breathe and its potential impact on human health and the environment. Traditionally, air quality assessment has relied on monitoring various pollutants such as particulate matter (PM), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), and ozone (O3) at monitoring stations distributed across different regions.

However, the interpretation of raw pollutant concentration data alone can be challenging for policymakers, urban planners, and the general public. To address this challenge, the concept of the Air Quality Index (AQI) was introduced. The AQI is a standardized metric that transforms complex pollutant concentration data into a single numerical value or categorical rating, allowing for easier interpretation and communication of air quality information to the public.

Historically, air quality prediction has relied on statistical models and deterministic approaches, which often struggle to capture the nonlinear relationships and complex interactions between various meteorological factors, geographical features, and pollutant concentrations. As a result, there has been a growing interest in leveraging Artificial Intelligence (AI) and Machine Learning (ML) techniques to enhance the accuracy and reliability of AQI predictions.

In recent years, AI and ML algorithms, such as time-series forecasting models, ensemble methods, and deep learning architectures, have shown promising results in predicting AQI levels with higher precision and efficiency. These techniques can analyze large volumes of historical air quality data, meteorological parameters, satellite imagery, and other environmental factors to identify patterns, trends, and correlations that traditional models may overlook.

By harnessing the power of AI and ML, researchers and environmental scientists aim to develop robust predictive models that not only provide accurate real-time AQI forecasts but also enable proactive decision-making, policy formulation, and public awareness campaigns to mitigate the adverse effects of air pollution on human health and the environment.

Objectives

1. Develop a comprehensive AI-driven system capable of accurately predicting air quality index (AQI) levels based on diverse environmental and meteorological factors.

2. Investigate the efficacy of machine learning algorithms, including ensemble methods and deep learning techniques, in forecasting AQI levels with high precision and efficiency.

3. Establish a robust data preprocessing pipeline to handle data from multiple sources, including pollutant concentrations, meteorological parameters, geographical features, and historical AQI measurements.

4. Evaluate the performance of the prediction model using established metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) to ensure robustness and generalizability across different regions and time periods.

5. Collaborate with stakeholders, including environmental agencies, public health authorities, and urban planning departments, to integrate the AI-driven AQI prediction system into existing monitoring frameworks and decision-making processes.

6. Continuously refine and update the prediction model using real-time data and feedback from stakeholders to enhance its accuracy and reliability over time.

Significance

1. Informed Environmental Policy: Accurate predictions of air quality index (AQI) levels empower policymakers, environmental agencies, and urban planners with critical information to formulate evidence-based policies and regulations aimed at improving air quality and public health.

2. Early Warning Systems: By automating the prediction of AQI levels using machine learning algorithms, stakeholders can develop early warning systems to alert communities and vulnerable populations about potential air quality hazards, allowing for timely interventions and mitigating health risks.

3. Public Health Protection: Accurate AQI predictions enable public health authorities to implement targeted interventions, such as advisories to reduce outdoor activities during periods of poor air quality, thereby safeguarding the health and well-being of the population, particularly those with respiratory conditions.

4. Environmental Monitoring and Management: Leveraging advanced technologies for AQI prediction enhances our ability to monitor and manage environmental pollution, facilitating the identification of pollution hotspots, sources of emissions, and areas requiring remediation measures to promote environmental sustainability.

5. Sustainable Urban Planning: Insights gained from predicted AQI trends and their correlation with urban development factors can inform sustainable urban planning practices, including the design of green spaces, transportation infrastructure, and zoning regulations to minimize air pollution and enhance livability in cities.

6. Technological Advancement: The adoption of AI-driven predictive modeling techniques for air quality forecasting represents a significant technological advancement in environmental monitoring and management, demonstrating the potential of machine learning to address complex environmental challenges and drive innovation in sustainability efforts.

Problem Definition and Requirements

Problem Definition

The problem of stock price prediction entails accurately forecasting future stock prices based on historical data and relevant market indicators. This task is critical for investors, traders, and financial institutions to make informed decisions regarding buying, selling, or holding stocks. However, predicting stock prices is challenging due to the complex and dynamic nature of financial markets, where prices are influenced by a multitude of factors such as economic indicators, company performance, geopolitical events, and investor sentiment. Traditional methods of stock price prediction often rely on simplistic models or manual analysis, which may not effectively capture the underlying patterns and relationships in the data, leading to inaccurate forecasts.

The objective is to develop an advanced stock price prediction system that leverages state-of-the-art Artificial Intelligence and Machine Learning techniques, particularly deep learning models like Long Short-Term Memory (LSTM) networks. The system should be capable of processing large volumes of historical stock market data, extracting relevant features, training predictive models, and generating actionable insights for stakeholders. By accurately forecasting stock price movements, the system aims to empower investors and traders with valuable information to optimize their investment strategies and mitigate risk.

Software Requirements

- **Programming Language:** Python - Python is a versatile and widely-used programming language with extensive support for data analysis, machine learning, and web development.

- **Libraries and Frameworks:**
    1. **NumPy:** NumPy is a powerful library for numerical computing in Python, providing support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

    2. **Pandas:** Pandas is a data manipulation library that offers data structures like DataFrame and Series, making it easy to manipulate and analyze structured data. It provides functionalities for data cleaning, reshaping, and aggregation.

    3. **Matplotlib:** Matplotlib is a plotting library that enables the creation of various types of static, interactive, and animated visualizations in Python. It offers a wide range of customization options for creating publication-quality plots.

    4. **Seaborn:** Seaborn is built on top of Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics. It simplifies the process of creating complex visualizations like heatmaps, violin plots, and pair plots.

    5. **Scikit-learn:** Scikit-learn is a comprehensive machine learning library that provides simple and efficient tools for data mining and data analysis. It offers algorithms for classification, regression, clustering, dimensionality reduction, and model selection.

    6. **Prophet:** Developed by Facebook, Prophet is a forecasting tool designed for time series analysis, particularly well suited for predicting AQI levels.

- **Development Environment:** Jupyter Notebook or any Python IDE (e.g., PyCharm, Visual Studio Code) - Jupyter Notebook provides an interactive computing environment for writing and executing Python code, making it ideal for exploratory data analysis and prototyping machine learning models. Python IDEs offer integrated development environments with features like code editing, debugging, and version control integration.

- **Version Control Git** - Git is a distributed version control system that allows multiple developers to collaborate on a project efficiently. It enables tracking changes to the codebase, managing different versions of the project, and facilitating collaboration through features like branching and merging.

Dataset

The dataset utilized in this project is sourced from the UCI Machine Learning Repository, comprising responses from a gas multisensor device deployed in an Italian city. The dataset contains hourly averaged sensor readings recorded over a one-year period, from March 2004 to February 2005, making it one of the longest freely available recordings of on-field air quality chemical sensor responses.

Dataset Characteristics:

- Multivariate: The dataset consists of multiple variables or features.
- Time-Series: The data is recorded over time, with hourly intervals.
- Subject Area: The dataset falls within the domain of Computer Science.
- Associated Tasks: The primary task associated with this dataset is regression.

Dataset Information:

- The dataset comprises 9358 instances, each representing hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device.
- The device was deployed in a significantly polluted area within an Italian city, at road level, to monitor air quality.
- Ground truth hourly averaged concentrations for various pollutants, including CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx), and Nitrogen Dioxide (NO2), are provided by a co-located reference certified analyzer.
- The dataset includes evidence of cross-sensitivities, concept drifts, and sensor drifts, which may impact the estimation capabilities of sensors.
- Missing values in the dataset are tagged with a value of -200.

Additional Information:

- The dataset includes 15 features, including the date, time, sensor responses, and true hourly averaged concentrations of various pollutants.
- Features such as CO(GT), NMHC(GT), C6H6(GT), NOx(GT), and NO2(GT) represent true hourly averaged concentrations of different pollutants.
- Sensor responses, such as PT08.S1(CO), PT08.S2(NMHC), PT08.S3(NOx), PT08.S4(NO2), and PT08.S5(O3), represent hourly averaged sensor responses targeted at specific pollutants.
- The dataset contains additional variables such as temperature, relative humidity, and absolute humidity.

Proposed Design/ Methodology

Design and Working

1. **Data Retrieval and Preprocessing:**
   • Air quality data is obtained from the UCI Machine Learning Repository, specifically the gas multisensor device responses dataset.
   • Data preprocessing involves handling missing values, removing duplicates, and scaling features to ensure data cleanliness and numerical stability.
   • Features such as Date, Time, and various sensor responses are extracted and formatted for further analysis and model training.

2. **Data Analysis and Visualization:**
   • Descriptive statistics of the air quality data are computed and displayed to provide insights into the dataset's characteristics.
   • Visualizations are generated using Matplotlib to illustrate trends, patterns, and relationships within the data.
   • Charts, histograms, and scatter plots are utilized to visualize the distribution of pollutants, sensor responses, and other relevant variables.

3. **Model Training and Testing:**
   • The project employs machine learning techniques, including the Prophet model, for predicting air quality index (AQI) values.
   • The Prophet model is trained separately in a Jupyter Notebook or Python script dedicated to model development.
   • Training data is processed to include relevant features and labels, such as pollutant concentrations and corresponding AQI values.
   • After training, the Prophet model is saved for future use in predicting AQI values based on new input data.

4. **Prediction and Evaluation:**
   • Testing data, representing new observations or future time points, is used to evaluate the trained Prophet model's performance.
   • The trained model is applied to the testing data to generate predictions of AQI values.
   • Predictions are compared against actual AQI values to assess the model's accuracy and reliability.
   • Evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are calculated to quantify the model's performance.
   • Visualizations, including time series plots of predicted vs. actual AQI values, are created to visualize model performance and identify any discrepancies or trends.

Machine Learning Technique

Air quality index (AQI) prediction also falls under supervised learning, similar to stock price prediction, as it involves using historical data (input) with corresponding AQI values (output) to train a model. In this context:

• **Labeled Data:** The historical data comprises various features such as pollutant concentrations, meteorological variables, and other relevant information. Each data point (e.g., each hour) is associated with the corresponding AQI value (output label).

• **Training Process:** During the training phase, the algorithm learns the patterns and relationships between the input features and the corresponding AQI values. It iteratively adjusts its parameters to minimize the discrepancy between its predicted AQI values and the actual observed values from the historical data.

• **Regression Task:** AQI prediction is typically formulated as a regression problem where the goal is to predict a continuous value (the AQI) based on the input features. The algorithm aims to approximate the underlying mapping function between the input variables (historical data) and the output variable (AQI).

• **Supervision**: The historical AQI values act as supervision for the learning process. The algorithm learns from past data, guided by the known outcomes, and aims to generalize this knowledge to make predictions on unseen data (future AQI values).

Usage for AQI Prediction:

• **Sequence Modeling:** AQI data can be treated as a sequence, where each data point represents the AQI value at a specific time. LSTMs are well-suited for modeling such sequential data due to their ability to capture temporal dependencies.

• **Temporal Dynamics:** AQI values exhibit complex temporal dynamics influenced by various factors such as pollutant emissions, weather conditions, and geographical location. LSTMs are capable of capturing these dynamics and making predictions based on historical patterns.

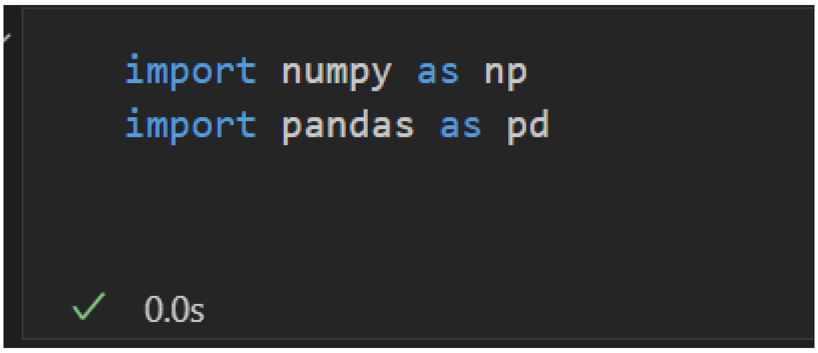
Implementation in the Project:

• **Model Training:** In the project, an LSTM model is trained using historical air quality data. The model learns from the sequential patterns in the data to make predictions about future AQI values.

• **Temporal Dependencies:** The model captures temporal dependencies in the air quality data, allowing it to learn from past AQI values and make predictions based on historical patterns.

• **Predictive Performance**: Metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) may be used to quantify the accuracy of the predictions.

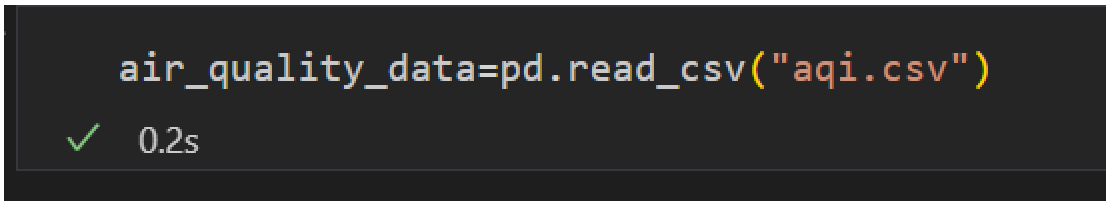
Results



```
import numpy as np
import pandas as pd
```

```
✓ 0.0s
```

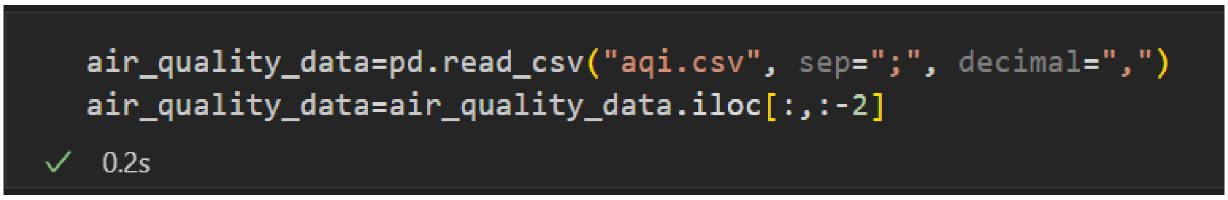
Figure 1 Importing Libraries



```
air_quality_data=pd.read_csv("aqi.csv")
```

```
✓ 0.2s
```

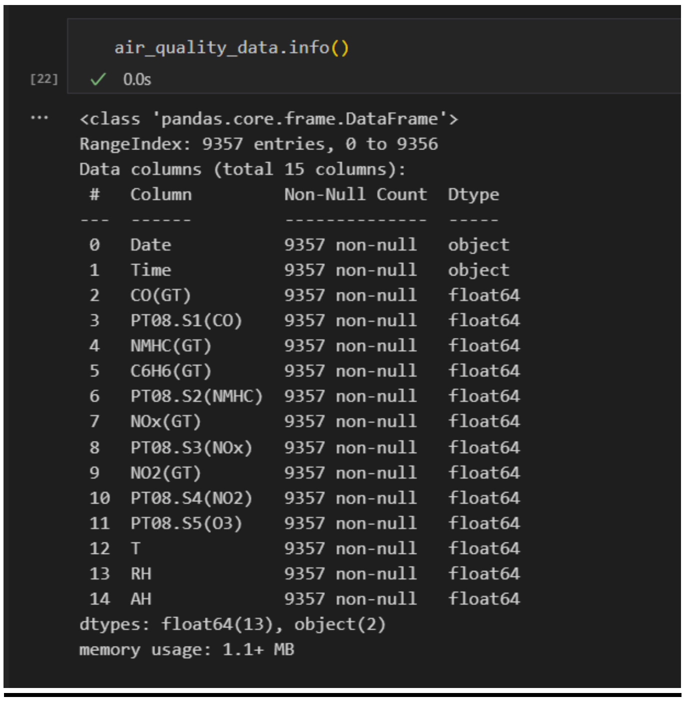
Figure 2 Downloading Dataset



```
air_quality_data=pd.read_csv("aqi.csv", sep=";", decimal=",")
air_quality_data=air_quality_data.iloc[:,:-2]
```

```
✓ 0.2s
```

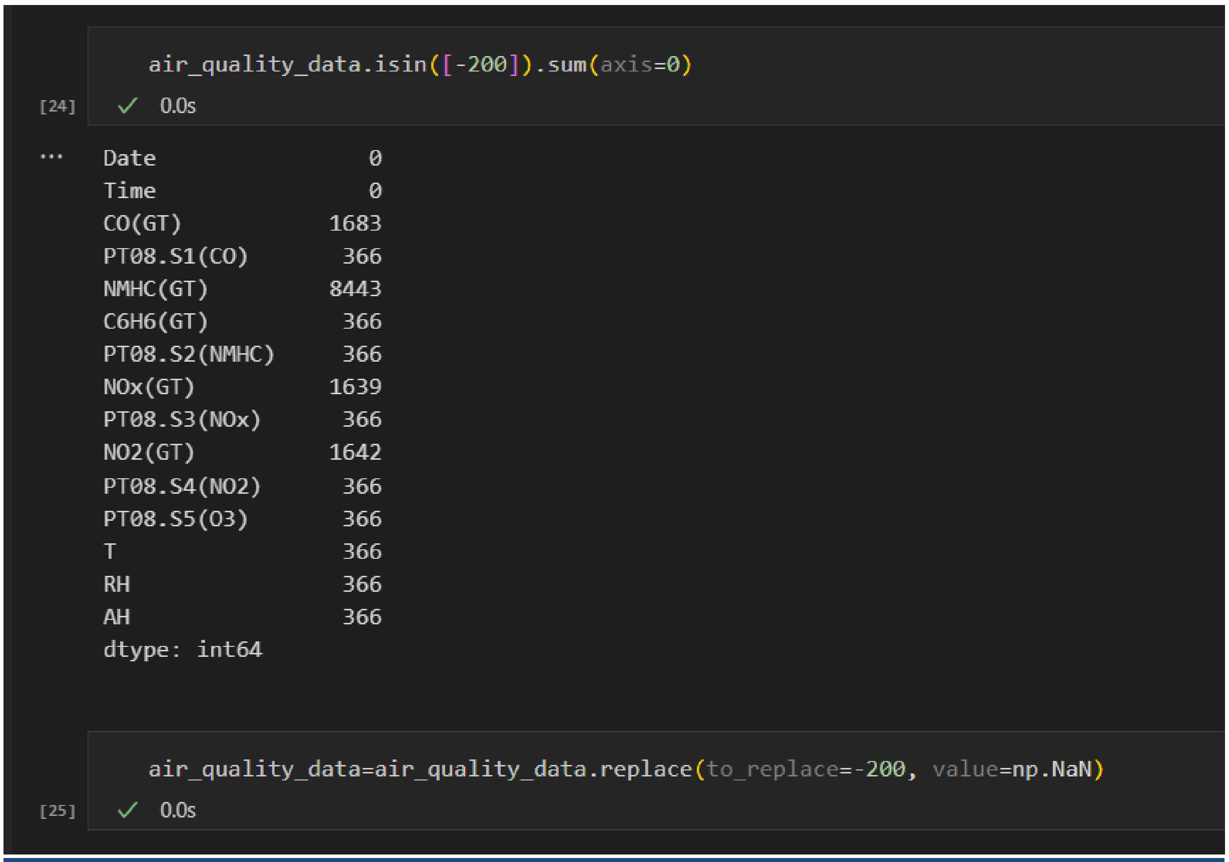
Figure 3 Making Desired Dataframe



```
air_quality_data.info()
```

```
[22]  ✓ 0.0s
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           9357 non-null   object
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   float64
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: float64(13), object(2)
memory usage: 1.1+ MB
```

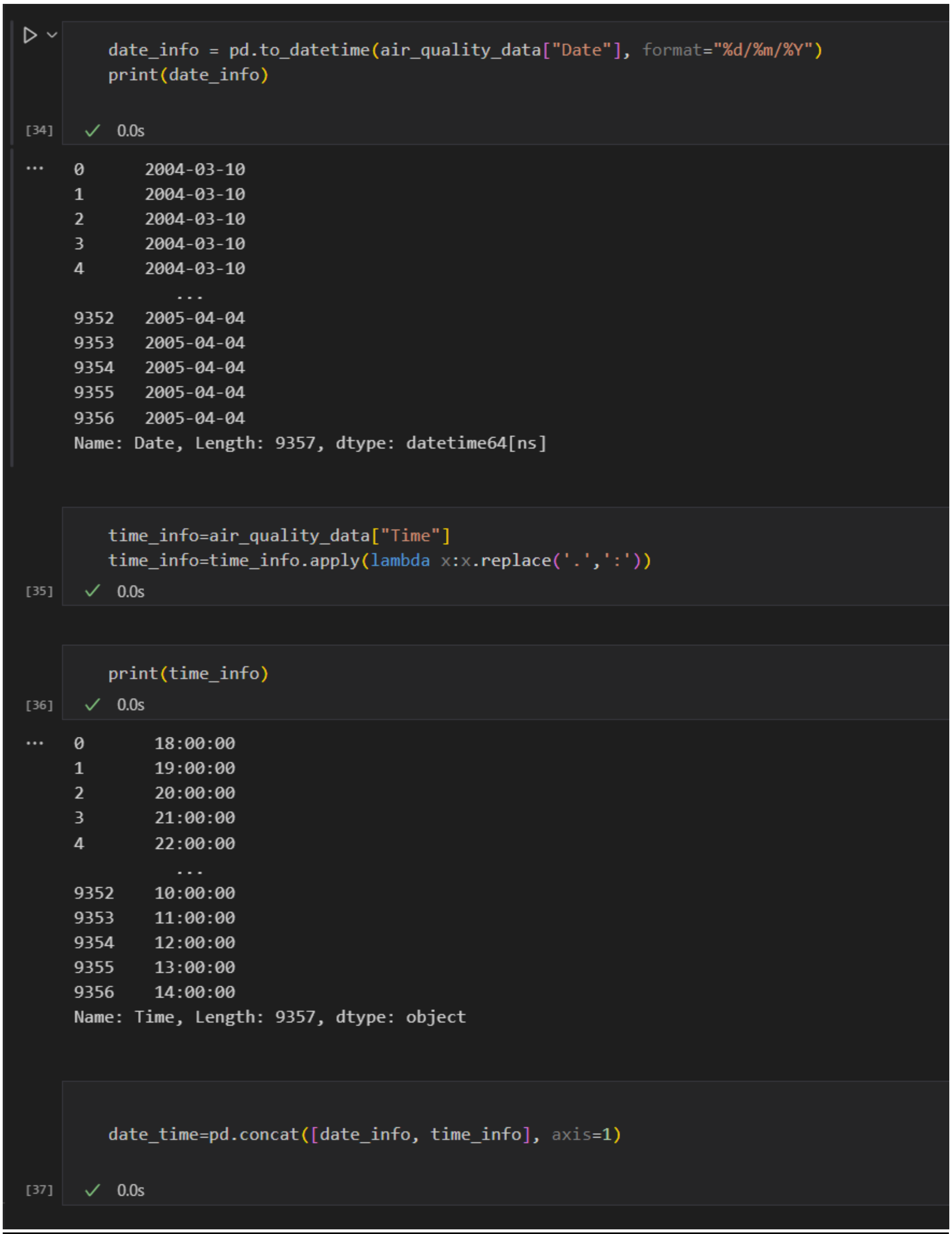


```
air_quality_data.isin([-200]).sum(axis=0)
```
[24] ✓ 0.0s

```
Date                0
Time                0
CO(GT)           1683
PT08.S1(CO)       366
NMHC(GT)         8443
C6H6(GT)          366
PT08.S2(NMHC)     366
NOx(GT)          1639
PT08.S3(NOx)      366
NO2(GT)          1642
PT08.S4(NO2)      366
PT08.S5(O3)       366
T                 366
RH                366
AH                366
dtype: int64
```

```
air_quality_data=air_quality_data.replace(to_replace=-200, value=np.NaN)
```
[25] ✓ 0.0s

Figure 4 Handling Null values



```
date_info = pd.to_datetime(air_quality_data["Date"], format="%d/%m/%Y")
print(date_info)
```
[34] ✓ 0.0s

```
0       2004-03-10
1       2004-03-10
2       2004-03-10
3       2004-03-10
4       2004-03-10
           ...
9352    2005-04-04
9353    2005-04-04
9354    2005-04-04
9355    2005-04-04
9356    2005-04-04
Name: Date, Length: 9357, dtype: datetime64[ns]
```

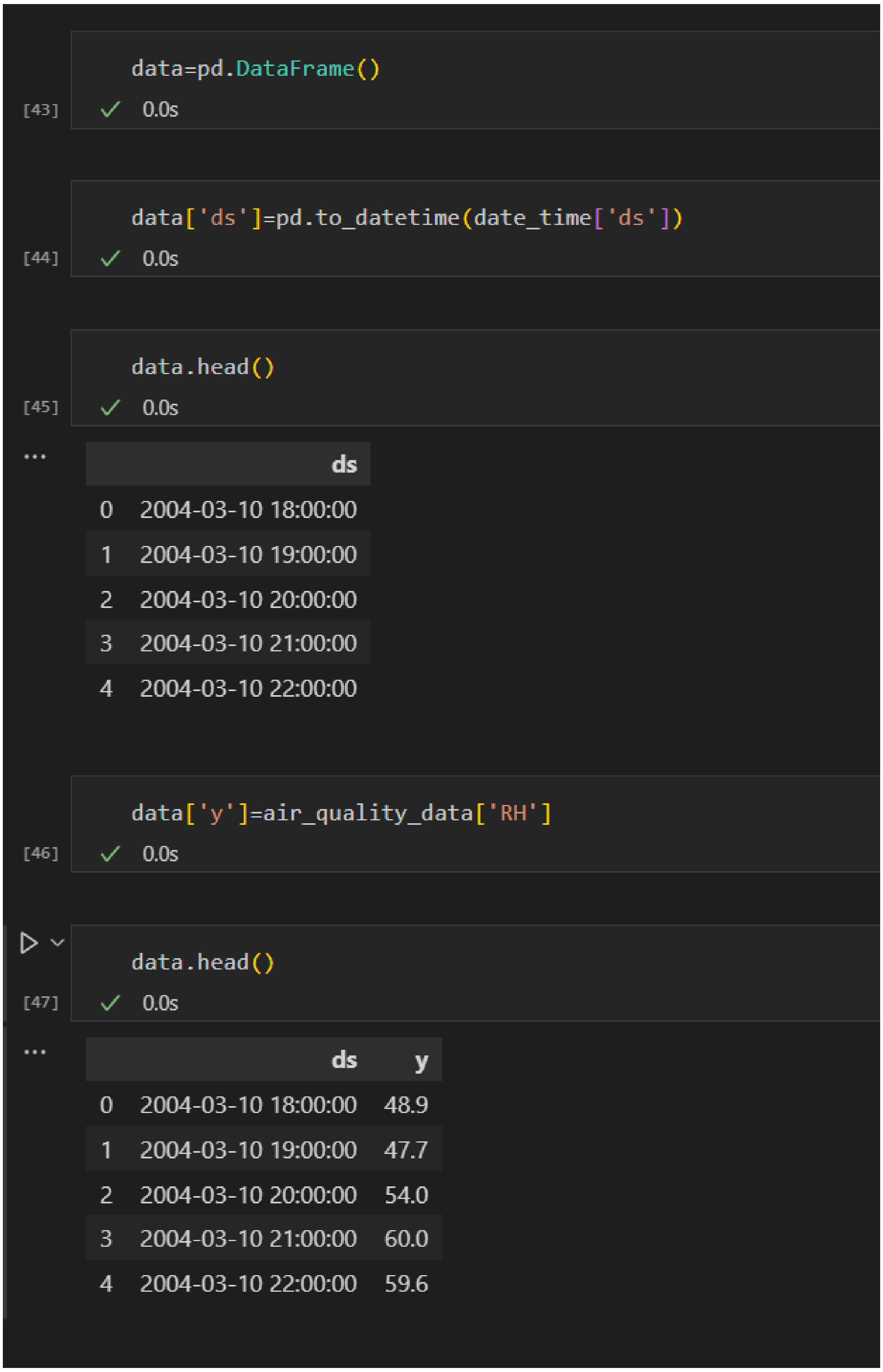
```
time_info=air_quality_data["Time"]
time_info=time_info.apply(lambda x:x.replace('.',':'))
```
[35] ✓ 0.0s

```
print(time_info)
```
[36] ✓ 0.0s

```
0        18:00:00
1        19:00:00
2        20:00:00
3        21:00:00
4        22:00:00
           ...
9352     10:00:00
9353     11:00:00
9354     12:00:00
9355     13:00:00
9356     14:00:00
Name: Time, Length: 9357, dtype: object
```

```
date_time=pd.concat([date_info, time_info], axis=1)
```
[37] ✓ 0.0s

Figure 5 *Converting Date Time Format into Desired Format*



```
[43]  data=pd.DataFrame()
      ✓ 0.0s
```

```
[44]  data['ds']=pd.to_datetime(date_time['ds'])
      ✓ 0.0s
```

```
[45]  data.head()
      ✓ 0.0s
```

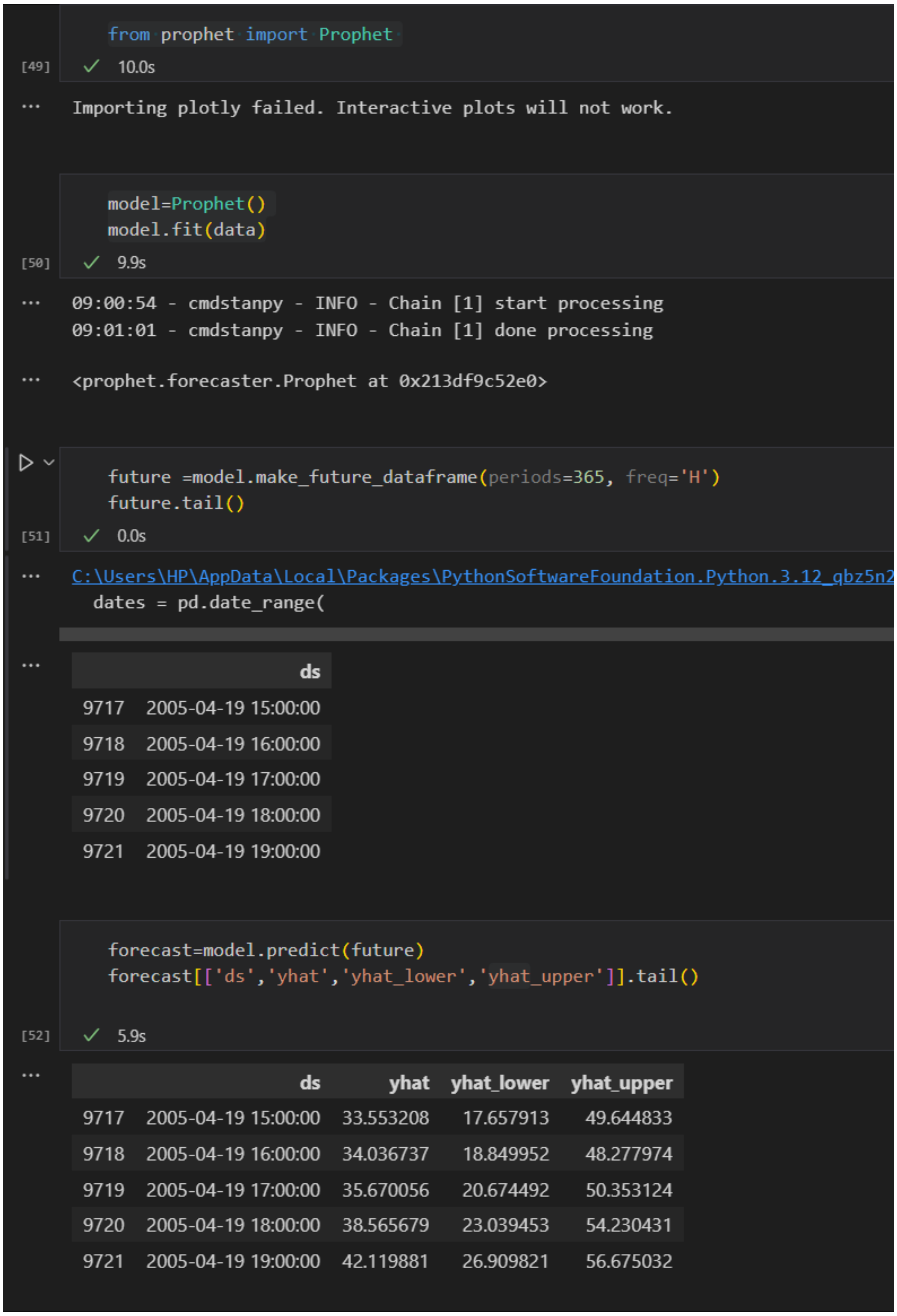
```
                    ds
0  2004-03-10 18:00:00
1  2004-03-10 19:00:00
2  2004-03-10 20:00:00
3  2004-03-10 21:00:00
4  2004-03-10 22:00:00
```

```
[46]  data['y']=air_quality_data['RH']
      ✓ 0.0s
```

```
[47]  data.head()
      ✓ 0.0s
```

```
                    ds     y
0  2004-03-10 18:00:00  48.9
1  2004-03-10 19:00:00  47.7
2  2004-03-10 20:00:00  54.0
3  2004-03-10 21:00:00  60.0
4  2004-03-10 22:00:00  59.6
```

Figure 6 *Creating the DataFrame*



```
from prophet import Prophet
[49] ✓ 10.0s
```

```
Importing plotly failed. Interactive plots will not work.
```

```
model=Prophet()
model.fit(data)
[50] ✓ 9.9s
```

```
09:00:54 - cmdstanpy - INFO - Chain [1] start processing
09:01:01 - cmdstanpy - INFO - Chain [1] done processing
```

```
<prophet.forecaster.Prophet at 0x213df9c52e0>
```

```
future =model.make_future_dataframe(periods=365, freq='H')
future.tail()
[51] ✓ 0.0s
```

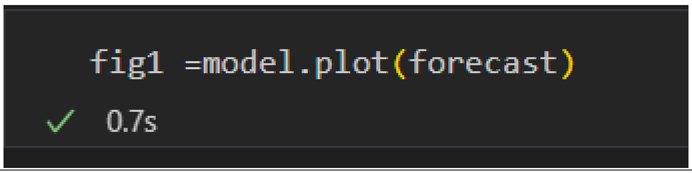
```
C:\Users\HP\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2
  dates = pd.date_range(
```

	ds
9717	2005-04-19 15:00:00
9718	2005-04-19 16:00:00
9719	2005-04-19 17:00:00
9720	2005-04-19 18:00:00
9721	2005-04-19 19:00:00

```
forecast=model.predict(future)
forecast[['ds','yhat','yhat_lower','yhat_upper']].tail()
[52] ✓ 5.9s
```

	ds	yhat	yhat_lower	yhat_upper
9717	2005-04-19 15:00:00	33.553208	17.657913	49.644833
9718	2005-04-19 16:00:00	34.036737	18.849952	48.277974
9719	2005-04-19 17:00:00	35.670056	20.674492	50.353124
9720	2005-04-19 18:00:00	38.565679	23.039453	54.230431
9721	2005-04-19 19:00:00	42.119881	26.909821	56.675032

Figure 7 *Predicting values*



```
fig1 =model.plot(forecast)
✓ 0.7s
```

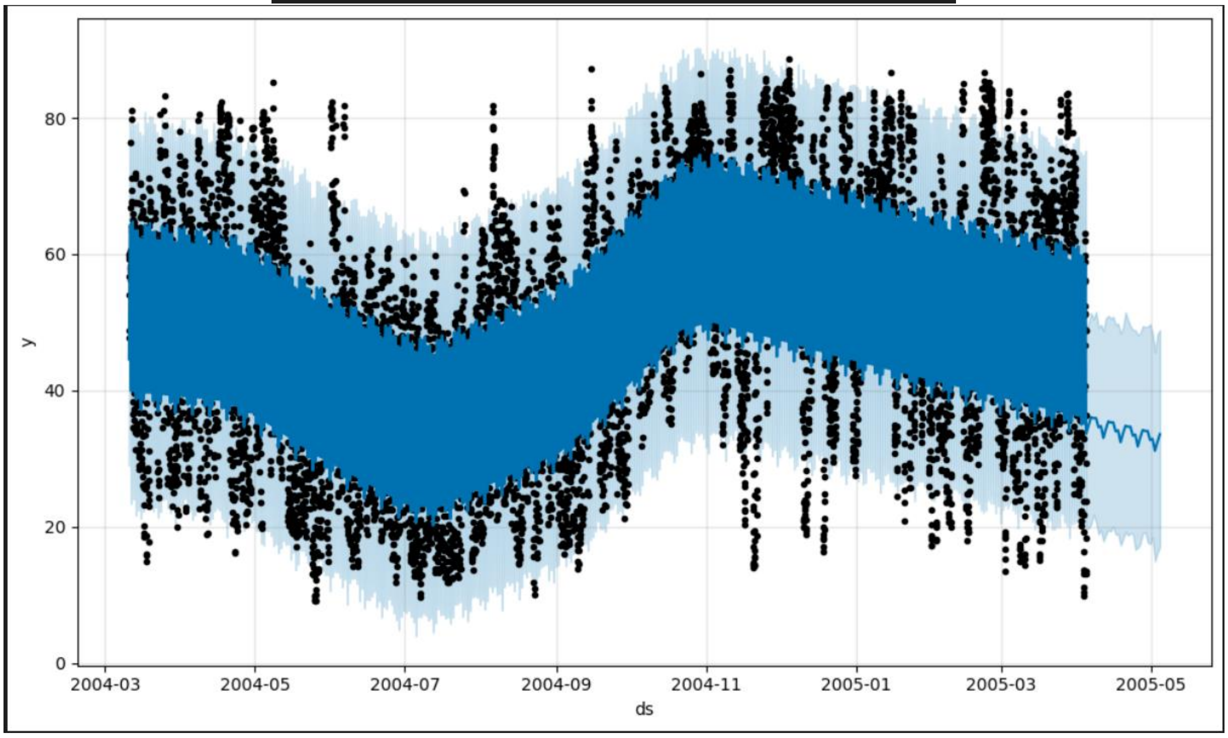
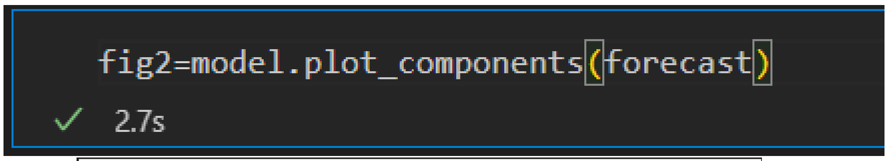


Figure 8 Visualise forecasted values



```
fig2=model.plot_components(forecast)
✓ 2.7s
```

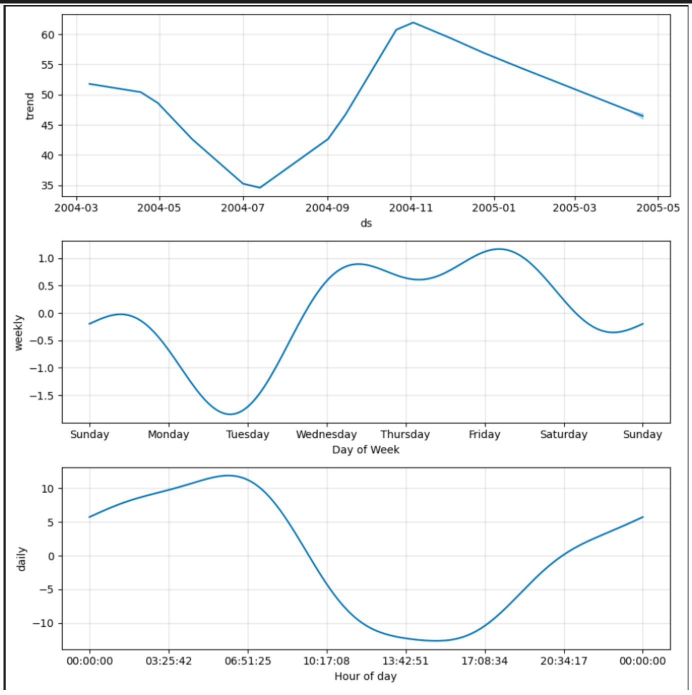
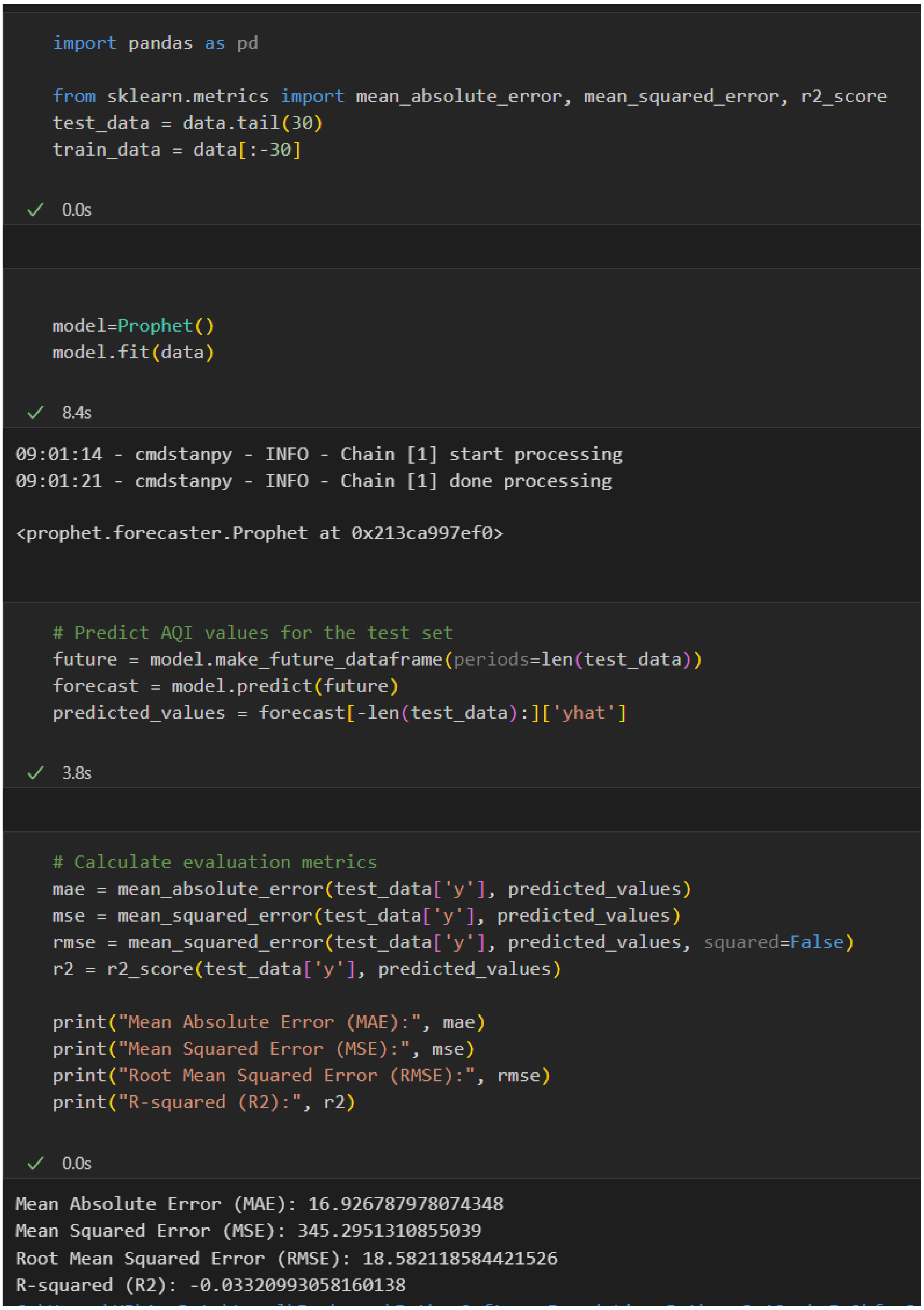


Figure 9 *Visualize time-wise breakdown*



```python
import pandas as pd

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
test_data = data.tail(30)
train_data = data[:-30]
```

✓ 0.0s

```python
model=Prophet()
model.fit(data)
```

✓ 8.4s

```
09:01:14 - cmdstanpy - INFO - Chain [1] start processing
09:01:21 - cmdstanpy - INFO - Chain [1] done processing

<prophet.forecaster.Prophet at 0x213ca997ef0>
```

```python
# Predict AQI values for the test set
future = model.make_future_dataframe(periods=len(test_data))
forecast = model.predict(future)
predicted_values = forecast[-len(test_data):]['yhat']
```

✓ 3.8s

```python
# Calculate evaluation metrics
mae = mean_absolute_error(test_data['y'], predicted_values)
mse = mean_squared_error(test_data['y'], predicted_values)
rmse = mean_squared_error(test_data['y'], predicted_values, squared=False)
r2 = r2_score(test_data['y'], predicted_values)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared (R2):", r2)
```

✓ 0.0s

```
Mean Absolute Error (MAE): 16.926787978074348
Mean Squared Error (MSE): 345.2951310855039
Root Mean Squared Error (RMSE): 18.582118584421526
R-squared (R2): -0.033209930581601338
```

Figure 10 Calculating MAE,MSE,RMSE and R-squared values

References

1. https://archive.ics.uci.edu/dataset/360/air+quality
2. https://stackoverflow.com/
3. https://www.geeksforgeeks.org/predicting-air-quality-index-using-python/