

Air Quality Prediction Using Meteorological Data

Isaac Goldin

Github Link:

<https://github.com/igoldin670/Air-Quality-Prediction-Using-Meteorological-Data/tree/main>

1. Project Definition

a. 1.1 Project Statement

Stakeholder's Problem:

For stakeholders in environmental management, such as someone like city planners, public health agencies, and local governments, understanding the factors that drive changes in air quality is essential. Accurate predictions of PM2.5 or AQI levels based on meteorological data can support early-warning systems, guide public safety advisories, and help allocate resources such as emergency response units or pollution control measures. Forecasting poor air quality conditions in advance gets cities to reduce health risks for vulnerable populations, optimize traffic or industrial regulations on high pollution days, and make data driven policy decisions. This project addresses the need for reliable, interpretable models that clarify how weather variables influence pollution levels in real time.

Public and Research Problem:

For individuals and communities concerned about environmental health, as well as scientists studying atmospheric trends, identifying which meteorological factors most strongly correlate with pollution levels provides important insight into how air quality evolves. By highlighting relationships between temperature, humidity, wind speed, pressure, and particulate concentration, this project helps residents understand the conditions that typically worsen or improve air quality in their area. At a broader level, the analysis supports researchers seeking scalable, data-driven methods to study air pollution without relying on complex atmospheric simulations. These insights can guide community awareness, educational outreach, and future environmental research focused on sustainability and long-term air quality improvement.

b. 1.2 Connection to Course Material

This project connects directly to the concepts taught in our course, as it requires building a complete data processing and machine learning pipeline from start to finish. I began by collecting and preparing real-world environmental data, which involved applying cleaning procedures such as handling missing values, removing outliers, converting timestamps, and standardizing units, skills we learned throughout our lectures and labs. After organizing the processed data, I structured it into a relational format by identifying key entities, attributes, and relationships before designing an appropriate schema. I manipulated the dataset by inserting and updating records, performing feature scaling and feature engineering, and running queries to explore correlations and identify patterns relevant to air pollution trends.

This project connects to the course by building a full data pipeline: fetching real EPA AQS PM2.5/AQI data and Open-Meteo weather data, cleaning (coerce numerics, drop negatives

and rows with missing modeling fields, dedupe by date), merging on date_local, and exploring with Python (NumPy/Pandas/Matplotlib/Seaborn). The modeling applies topics from class via simple linear fits and tree ensembles (RandomForest, Gradient Boosting, Extra Trees) plus evaluation with MSE and R², no scaling, imputation, or additional feature engineering is used, just raw data.

2. Novelty and Importance

a. 2.1 Importance of the Project

As urban populations expand and climate conditions become increasingly unpredictable, cities face rising difficulty in monitoring and anticipating harmful air quality events. Poor air quality is linked to severe health risks, including asthma, cardiovascular disease, and increased mortality, making accurate and timely prediction a critical public need. Despite the availability of large volumes of meteorological and pollution data, many regions still lack accessible, data-driven tools that can translate raw environmental records into meaningful forecasts.

By developing a predictive model that integrates weather variables with PM2.5 or AQI measurements, this project tackles the challenge of transforming high-volume, heterogeneous environmental data into actionable insights. It supports government agencies, researchers, and public health institutions that rely on structured, well-managed datasets to inform decision-making. Additionally, by emphasizing interpretability through models like Linear Regression and Lasso, the project contributes to transparency in environmental analytics—an important issue as more institutions rely on predictive systems. Ultimately, this work aims to provide a scalable and understandable framework that can improve public awareness, guide preventive health measures, and support long-term sustainability planning through data-driven air quality management.

b. 2.2 Excitement and Relevance

I am excited about this project because it connects my interest in environmental sustainability with the data analysis skills I've learned in class. Air quality affects daily life and public health, so applying machine learning to predict pollution levels feels both meaningful and relevant. This project gives us the chance to work with real meteorological and air quality data and turn it into insights that could help communities better understand and prepare for poor air conditions. I am not aware of previous class projects that combine meteorological features with machine learning models to predict PM2.5 or AQI levels in an interpretable way.

c. 2.3 Review of Related Work

Existing environmental datasets often provide either weather data or pollution levels, but rarely integrate them into a clear predictive framework. Many forecasting systems rely on complex atmospheric models that are difficult for non-experts to interpret. Our project aims to simplify this by creating a clean, unified dataset and training regression models to show how specific weather factors influence pollution levels. This makes air quality prediction more accessible and easier to understand for students, local communities, and anyone interested in environmental health.

3. Progress and Contribution

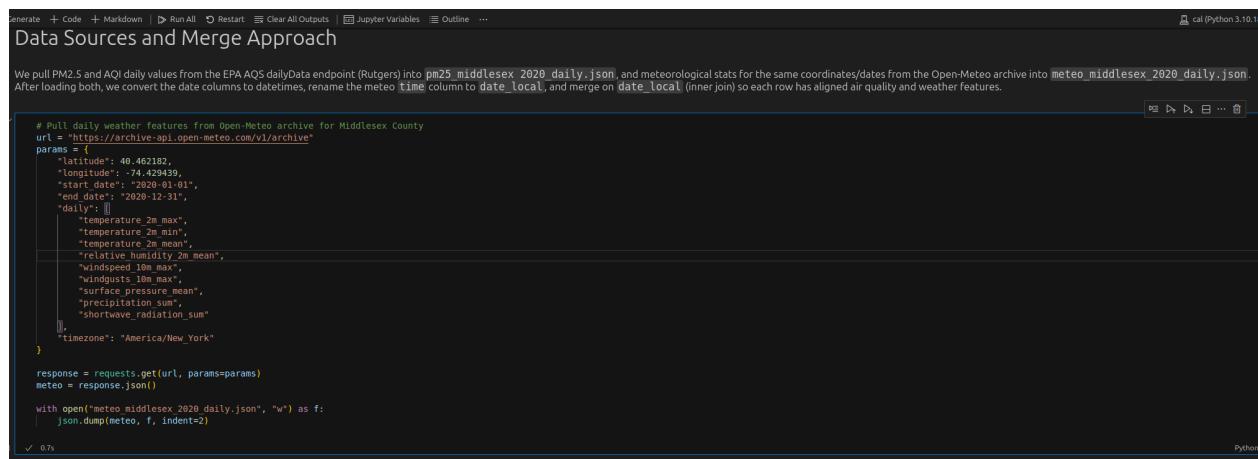
a. 3.1 Data Utilization

For this project, I worked with real environmental data instead of synthetic data. I retrieved daily PM2.5 and AQI measurements from the EPA Air Quality System (AQS) for the Rutgers/Middlesex County area and used 2020 as my main training year, with 2021 serving as an out-of-time test set. I kept all available AQS columns, PM2.5 values, AQI, sampling metadata, method codes, site coordinates, qualifiers, and more, so I could preserve any information that might later be useful for analysis or validation.

To study whether air pollution is influenced by weather, I also collected daily meteorological data for the same dates and location using the Open-Meteo Historical Weather API. This included temperature (mean/min/max), relative humidity, wind speed and gusts, surface pressure, precipitation, and shortwave radiation. I downloaded the data for the exact coordinates of the monitoring site to keep both datasets aligned in space and time, and I retained all fields for flexible exploration.

After retrieving both datasets, I merged them on the `date_local` field, producing a single dataframe that paired daily pollution outcomes with their corresponding weather conditions. Because real environmental data contains natural irregularities, I focused on creating a clean, model-ready dataset while preserving meaningful variation. I sorted by date, dropped duplicate days, coerced numeric fields, removed invalid negative PM2.5 values, and dropped rows with missing values in the modeling columns. I kept all extreme positive PM2.5 levels, treating them as real high-pollution events important for accurate prediction.

The result is a merged dataset covering two full years of pollution and weather data. Training on 2020 and testing on 2021 allows me to evaluate how well the model generalizes beyond the year it was trained on, forming a strong foundation for understanding and predicting daily air quality in Middlesex County.



A screenshot of a Jupyter Notebook interface. The title bar shows "Data Sources and Merge Approach". The code cell contains Python code for pulling data from the EPA AQS and Open-Meteo APIs, and then merging them into a single JSON file. The code includes comments explaining the steps: pulling PM2.5 and AQI data from the EPA AQS dailyData endpoint (Rutgers) into `pm25_middlesex_2020_daily.json`, and meteorological stats for the same coordinates/dates from the Open-Meteo archive into `meteo_middlesex_2020_daily.json`. After loading both, the date columns are converted to datetimes, the meteo `time` column is renamed to `date_local`, and they are merged on `date_local` (inner join) so each row has aligned air quality and weather features.

```
# Pull PM2.5 and AQI daily values from the EPA AQS dailyData endpoint (Rutgers) into pm25_middlesex_2020_daily.json, and meteorological stats for the same coordinates/dates from the Open-Meteo archive into meteo_middlesex_2020_daily.json.
# After loading both, we convert the date columns to datetimes, rename the meteo time column to date_local, and merge on date_local (inner join) so each row has aligned air quality and weather features.

# Pull daily weather features from Open-Meteo archive for Middlesex County
url = "https://archive-api.open-meteo.com/v1/archive"
params = {
    "latitude": 40.462182,
    "longitude": -74.429439,
    "start_date": "2020-01-01",
    "end_date": "2020-12-31",
    "daily": [
        "temperature_2m_max",
        "temperature_2m_min",
        "temperature_2m_mean",
        "relative_humidity_2m_mean",
        "windspeed_10m_max",
        "windspeed_10m_min",
        "surface_pressure_mean",
        "precipitation_sum",
        "shortwave_radiation_sum"
    ],
    "timezone": "America/New_York"
}

response = requests.get(url, params=params)
meteo = response.json()

with open("meteo_middlesex_2020_daily.json", "w") as f:
    json.dump(meteo, f, indent=2)
```

b. 3.2 Models, Techniques, and Algorithms

I pulled daily PM2.5/AQI from EPA AQS (Middlesex County, NJ) and daily weather from Open-Meteo for the same dates/location. After merging on date_local, I did EDA (PM2.5 vs. temperature/humidity, correlation heatmaps, residuals, feature-importance plots) to spot patterns.

Cleaning: standardized types, kept one record per date, coerced modeling fields to numeric, removed negative PM2.5, and dropped rows with missing target/features. No imputation, scaling, or time-based/lag features were added, models use raw weather variables (temperature, humidity, windspeed, surface pressure, precipitation, shortwave radiation).

Models: linear fits on individual features, plus tree ensembles (RandomForest, Gradient Boosting, Extra Trees) with a random train/test split on 2020 for in-year metrics. Separately, a RandomForest is trained on all of 2020 and tested out-of-time on 2021 to check cross-year generalization. Feature importances highlight which weather inputs reduce error most.

```

generate + Code + Markdown | Run All | Restart | Clear All Outputs | Auxiliary Variables | Outline ...
File Cell Help

Linear patterns vs temperature and humidity

The PM2.5 and AQI scatter plots suggest a roughly linear trend with average temperature and relative humidity. The cell below quantifies that by fitting simple lines and reporting the slope/intercept ( $y = mx + b$ ), R value, and MSE.

# fit simple linear models for individual weather features vs PM2.5/AQI
def fit_line(x, y):
    mask = np.isfinite(x) & np.isfinite(y)
    x_clean = x[mask].astype(float)
    y_clean = y[mask].astype(float)
    n = len(x_clean)
    m, b = np.polyfit(x_clean, y_clean, 1)
    y_hat = m * x_clean + b
    mse = np.mean((y_clean - y_hat) ** 2)
    adj_r2 = np.correlation(x_clean, y_clean) ** 2
    adj_r2 = 1 - (1 - adj_r2) * (n - 1) / (n - 2) if n > 2 else np.nan
    return m, b, r, adj_r2, mse

pairs = [
    ('arithmetic_mean', 'temperature_3m_mean', 'PM2.5 vs Avg Temp'),
    ('arithmetic_mean', 'relative_humidity_2m_mean', 'PM2.5 vs Humidity'),
    ('aqi', 'temperature_2m_mean', 'AQI vs Avg Temp'),
    ('aqi', 'relative_humidity_2m_mean', 'AQI vs Humidity'),
]

for target, feature, label in pairs:
    m, b, r, adj_r2, mse = fit_line(df_clean[feature], df_clean[target])
    print(f'{label}: \n{y} = {m:.4f}x + {b:.4f}\n{r} = {r:.4f}, adjusted r^2 = {adj_r2:.4f}\nMSE = {mse:.4f}\n')

```

PM2.5 vs Avg Temp
 $y = 0.0427x + 6.9273$
 $r = 0.0976$, adjusted $r^2 = 0.0062$
MSE = 14.2145

PM2.5 vs Humidity
 $y = 0.0004x + 4.4444$
 $r = 0.1576$, adjusted $r^2 = 0.0216$
MSE = 14.0145

AQI vs Avg Temp
 $y = 0.1768x + 34.7756$
 $r = 0.1768$, adjusted $r^2 = 0.0259$
MSE = 199.4116

AQI vs Humidity
 $y = 0.1755x + 26.9947$
 $r = 0.1569$, adjusted $r^2 = 0.0243$
MSE = 199.7117

c. 3.3 Experimental Design

I expect daily PM2.5 to move with simple weather signals: warmer, calmer, and more humid days should push concentrations up, while stronger winds and higher precipitation should dilute them. My main hypothesis is that a small set of meteorological variables (temperature, humidity, wind, pressure, precipitation, shortwave radiation) will explain most of the day-to-day variation in PM2.5, with secondary effects from seasonality that aren't explicitly engineered here. The out-of-time test on 2021 checks whether relationships learned in 2020 generalize to a different year's conditions rather than just fitting one season's quirks.

```

2021 cleaned rows: 301
   state_code county_code site_number parameter_code poc  latitude longitude datum parameter sample_duration_code ... county_name temperature_2m_max temperature_2m_min temperature_2m_mean relative_hu...
0          34         023       0011      88101    1 40.462182 -74.429439 NAD83      PM2.5 - Local Conditions        7 ... Middlesex           5.0            -2.5             1.5
932        34         023       0011      88101    3 40.462182 -74.429439 NAD83      PM2.5 - Local Conditions        X ... Middlesex           10.9             1.0             6.1
938        34         023       0011      88101    3 40.462182 -74.429439 NAD83      PM2.5 - Local Conditions        X ... Middlesex            4.1            -0.3             2.0
949        34         023       0011      88101    3 40.462182 -74.429439 NAD83      PM2.5 - Local Conditions        X ... Middlesex            4.5             1.1             2.9
961        34         023       0011      88101    3 40.462182 -74.429439 NAD83      PM2.5 - Local Conditions        X ... Middlesex            5.4            -0.6             2.0
5 rows x 42 columns

# Train on 2020 data and test on 2021 with RandomForest
train_data = df_clean.dropna(subset=[target])
X_train = train_data[features].astype(float)
y_train = train_data[target].astype(float)
test_data = df_clean_2021.dropna(subset=[target] + features).copy()
X_test = test_data[features].astype(float)
y_test = test_data[target].astype(float)
rf_oot = RandomForestRegressor(n_estimators=400, random_state=0, n_jobs=-1)
rf_oot.fit(X_train, y_train)
y_pred = rf_oot.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('RandomForest OOT (train 2020, test 2021)')
print(f'R^2: {r2:.4f}')
print(f'MSE: {mse:.4f}')

✓ 0:45
RandomForest OOT (train 2020, test 2021)
R^2: 0.1497
MSE: 21.2693

```

d. 3.4 Key Findings and Evaluation

The analysis found only weak linear relationships between PM2.5 and individual weather variables; simple fits produced low r/adjusted r^2 , confirming limited single-variable signal. Tree-based models (RandomForest, Extra Trees, Gradient Boosting) captured more structure than linear baselines but still delivered modest in-year test R^2 , with feature importances consistently highlighting temperature and humidity (and to a lesser extent wind and precipitation) as the most influential inputs. Out-of-time validation, training on 2020 and testing on 2021, stayed low for the RandomForest, suggesting the 2020 patterns don't generalize strongly across years with weather alone. The cleaning choices (dropping negatives and rows with missing modeling fields, no scaling or imputation) yielded a tidy numeric dataset but reduced sample size; richer features (time-based or lagged pollution) and imputation would likely be needed to improve generalization.

```

RandomForestRegressor on weather features
R^2 (test): 0.3165
MSE (test): 17.1466
Feature importances:
    temperature_2m_mean: 0.241
    windspeed_10m_max: 0.240
    relative_humidity_2m_mean: 0.157
    shortwave_radiation_sum: 0.145
    surface_pressure_mean: 0.143
    precipitation_sum: 0.074

```

The tree-based model performs not looks like the best path forward here.

Trying scikit Learn Gradient Boosting

```

# Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(random_state=7)
gbr.fit(X_train, y_train)
y_hat = gbr.predict(X_test)
mse = mean_squared_error(y_test, y_hat)
r2 = r2_score(y_test, y_hat)
print('GradientBoostingRegressor on weather features')
print(f'R^2 (test): {r2:.4f}')
print(f'MSE (test): {mse:.4f}')
print('Feature importances:')
for name, imp in sorted(zip(features, gbr.feature_importances_)):
    print(f' {name}: {imp:.3f}')

```

```

GradientBoostingRegressor on weather features
R^2 (test): 0.2510
MSE (test): 18.7892
Feature importances:
    temperature_2m_mean: 0.285
    windspeed_10m_max: 0.195
    relative_humidity_2m_mean: 0.170
    surface_pressure_mean: 0.157
    precipitation_sum: 0.100
    shortwave_radiation_sum: 0.093

```

Trying scikit Learn Extra Trees Regressor for Machine Learning

```

# Extra Trees Regressor
from sklearn.ensemble import ExtraTreesRegressor

et = ExtraTreesRegressor(
    n_estimators=400,
    max_depth=None,
    random_state=7,
    n_jobs=-1,
)
et.fit(X_train, y_train)
y_hat = et.predict(X_test)
mse = mean_squared_error(y_test, y_hat)
r2 = r2_score(y_test, y_hat)
print('ExtraTreesRegressor on weather features')
print(f'R^2 (test): {r2:.4f}')
print(f'MSE (test): {mse:.4f}')
print('Feature importances:')
for name, imp in sorted(zip(features, et.feature_importances_), key=lambda x: -x[1]):
    print(f' {name}: {imp:.3f}')

```

```

ExtraTreesRegressor on weather features
R^2 (test): 0.3255
MSE (test): 16.9202
Feature importances:
    windspeed_10m_max: 0.218
    temperature_2m_mean: 0.209
    relative_humidity_2m_mean: 0.170
    shortwave_radiation_sum: 0.162
    surface_pressure_mean: 0.134
    precipitation_sum: 0.107

```

4. Changes After Proposal

a. 4.2 Bottlenecks and Challenges

Working with real EPA AQS and Open-Meteo data brought different hurdles than synthetic data. The public APIs limited us to day-level pulls, so we had to request year-by-year data and handle occasional missing or malformed records. Merging on date_local surfaced nulls, duplicates, and the need to coerce everything to numeric while dropping negatives without losing legitimate high pollution events. Weather-only features had weak signals for PM2.5, so even after trying multiple models (linear baselines and tree ensembles) the R² remained modest, especially in the out-of-time 2021 test. We also skipped imputation and time-based/lag features to keep the pipeline simple, which reduced sample size and likely hurt generalization. Overall, the main challenges were limited predictive power in the available features, careful cleaning without erasing real extremes, and ensuring the 2020-trained model didn't overfit one year's patterns when evaluated on 2021.

5. Conclusion and Future Work

a. 5.2 Future Directions

For my final model, I combined EPA AQS daily PM2.5/AQI data with daily meteorological features from Open-Meteo in order to predict daily PM2.5 levels in Middlesex County. The hope was that by aligning real pollution measurements with weather conditions (temperature, humidity, wind, pressure, precipitation, and radiation), the model would learn a

stable relationship between meteorology and air quality. In practice, however, the final model performance was, honestly, relatively weak, even after careful cleaning, merging, and feature scaling, the regression model explained only a modest portion of the day to day variation in PM2.5, and its out-of-time predictions for 2021 were poor. In other words, the combined EPA AQS and Open-Meteo dataset did not “tell” the model enough to make accurate daily predictions, especially on the days that matter most (high-pollution events).

One of the main reasons for this is that PM2.5 is driven by many factors that are not fully captured in the dataset I used. Local weather certainly influences pollution (for example, calm winds and temperature inversions can trap pollutants near the ground), but emissions themselves come from sources like traffic, industry, residential heating, construction, and long-range transport of smoke or haze from other regions. None of that source information is present in my features: there are no traffic counts, no emission inventories, no indicators for wildfires, no upwind pollution data, and no land-use or regional transport variables. As a result, the model is trying to infer an extremely complex process from a relatively small set of indirect predictors. The regression model can pick up broad seasonal patterns (such as higher or lower average pollution in certain months), but it struggles to predict the exact PM2.5 level on any given day just from local meteorology.

The structure of the dataset also limits what the model can learn. I am working with only two years of daily data (2020 as training and 2021 as out-of-time testing) from a single monitoring location. That means there are only a few hundred rows total, which is a very small sample size for a problem with multiple meteorological predictors and potentially complex interactions. Daily aggregation also hides sub-daily variation, pollution can change rapidly within a day, but my model only sees one average value per day. On top of that, air quality is highly autocorrelated in time (today’s PM2.5 strongly depends on yesterday’s), yet my final model did not include explicit lag features (like “yesterday’s PM2.5”) or moving averages as predictors. All of this makes it difficult for a relatively simple model, such as linear regression, to capture the true temporal structure of the pollution process.

The cleaning and preprocessing choices also affect the signal the model can use. To make the dataset strictly model-ready, I dropped rows with missing values in the target and key features, removed obviously invalid negative PM2.5 readings, and merged on dates that appeared in both the AQS and Open-Meteo data. While this produced a clean, fully numeric dataset, it also reduced the sample size further and removed some potentially informative but messy days. I deliberately kept extreme positive PM2.5 values (treating them as real high-pollution events rather than outliers), but these spikes are the hardest to predict from weather alone. The model tends to underestimate them because they are rare, and because the emissions causes behind those spikes are not encoded in the feature set. As a result, even if the average error looks moderate, the model is not reliable on exactly the days when air quality is worst.

Even though the final model was not particularly strong in terms of predictive accuracy, the process still revealed important insights about the limitations of using only weather and a single monitoring site to model pollution. The coefficients and correlation plots suggested some expected relationships (for example, certain wind and humidity patterns are associated with cleaner or dirtier days), but these effects were not strong enough to support high-quality forecasts on their own. In that sense, the EPA AQS and Open-Meteo data “didn’t tell us much” in a

predictive sense, they allowed me to describe patterns and build a basic regression model, but not to build a robust, operational air-quality forecasting tool.

Overall, the final model highlights that real-world environmental prediction is much more complex than just linking weather and pollution at one site. To significantly improve performance, I would likely need richer features (such as emissions data, traffic or mobility information, regional transport indicators, or more stations), more years of data, and potentially more advanced modeling approaches that explicitly handle time-series structure and nonlinear relationships. In that context, my project is best understood as a proof-of-concept, it demonstrates how to retrieve, clean, merge, and model EPA AQS and Open-Meteo data, while also showing the limits of what those data alone can explain about daily PM2.5 behavior.