University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Assignment Title: Environmental Impact Analysis of Air Quality Data

Programme Title: Information Technology Management

Name: Safa Orhan

Number: Q1065727

Year: 2025

Contents

Introduction **3**
Data Collection and Preprocessing **3**
Outlier Analysis 6
Model Selection **7**
Random Forest 7
Gradient Boosting Regression 10
Data Analysis and Visualization of Air Quality **11**
World Map - Pollution of PM2.5 12
Season Trends of Air Quality 13
Temperature and Air Quality 14
PM2.5 Level per Hour 15
Top Countries with Average PM2.5 16
Weather Condition and Average PM2.5 17
PM2.5 and PM10 Relation 18
Top 5 Polluted Countries 19
Chile - Air Quality per Month 20
Chile - Temperature vs. PM2.5 21
Chile - PM2.5 and Weather Count 22
Dashboard 23
Evaluation Metrics 24
Conclusion **25**
References **26**
Appendix **27**

Introduction

Air pollution is an important problem of the world. It is impacting over 90% of the global population according to the World Health Organization[1]. With the increasing numbers of industrial factories, nitrogen dioxide (NO_2), carbon monoxide (CO), particulate matter (PM2.5 and PM10) and ozone (O_3) are released into the atmosphere in large amounts.

Air pollution happens when harmful materials like PM2.5, CO, and NO_2 are released into the air. The dataset we have consists of country, location, last updated date and hour, temperature, weather and values of air quality (NO_2, CO, PM2.5, PM10, O_3). According to Nature Communications[2], world's most important health factor is PM2.5. When PM2.5 level is high it will most likely be the reason for diseases like asthma, heart problems or lung cancer.

As this dataset is a continuous data with many relationships between different sources of air quality, temperature and weather, predicting PM2.5 value should be the entry point. PM2.5 is dependent on different variables and should be predicted from past data. Since there will be complex relationships like location, time and weather regression algorithms can be used to predict output value. Regression models can be trained to identify patterns using past air quality data and provide insights about air pollution to precaution and prevent catastrophic problems in advance.

The primary aim of this project is to train a machine learning regression model to predict PM2.5 air quality based on historical air quality, location, weather and temperature data, identify the hotspots and predict future air quality levels.

Data Collection and Preprocessing

Dataset consist different types of data as following:

- Location based data: country, location_name, latitude, longitude, timezone
- Time based data: last-updated, sunrise, sunset
- Weather data: temperature_celsius, condition text
- Air quality data: air_quality_Carbon_Monoxide, air_quality_Ozone, air_quality_Nitrogen_dioxide, air_quality_Sulphur_dioxide, air_quality_PM2.5, air_quality_PM10

To achieve highest accuracy, following columns will be added to dataset:

- Time based data: Extracting last-updated column for each hour, day, month, year will be important to train the model.

---

[1] World Health Organization, 2016. *Ambient air pollution: A global assessment of exposure and burden of disease*. World Health Organization. Available at: https://iris.who.int/bitstream/handle/10665/250141/9789241511353-eng.pdf?sequence=1 [Accessed 26 January 2025].

[2] McDuffie, E.E., Martin, R.V., Spadaro, J.V., Burnett, R., Smith, S.J., O'Rourke, P., Hammer, M.S., van Donkelaar, A., Bindle, L., Shah, V., Jaeglé, L., Luo, G., Yu, F., Adeniran, J.A., Lin, J. and Brauer, M., 2021. *Source sector and fuel contributions to ambient PM2.5 and attributable mortality across multiple spatial scales*. Nature Communications, [online] 12(1), pp.1. Available at: https://www.nature.com/articles/s41467-021-23853-y [Accessed 26 January 2025].

- Daylight duration: With sunrise and sunset data, total daylight duration will be calculated for the saved air quality day.
- Air pollution severity: Classifying in severity based air pollution will help for visualization.

To ensure that the dataset does not have any missing values, air pollution data will be checked and imported with estimated data depending on the location. The nearest located data point (KNN imputer[3]) can be adjusted for missing values because location is the most relevant relation between air pollution. For other types of data, the mean of total data can be adjusted.

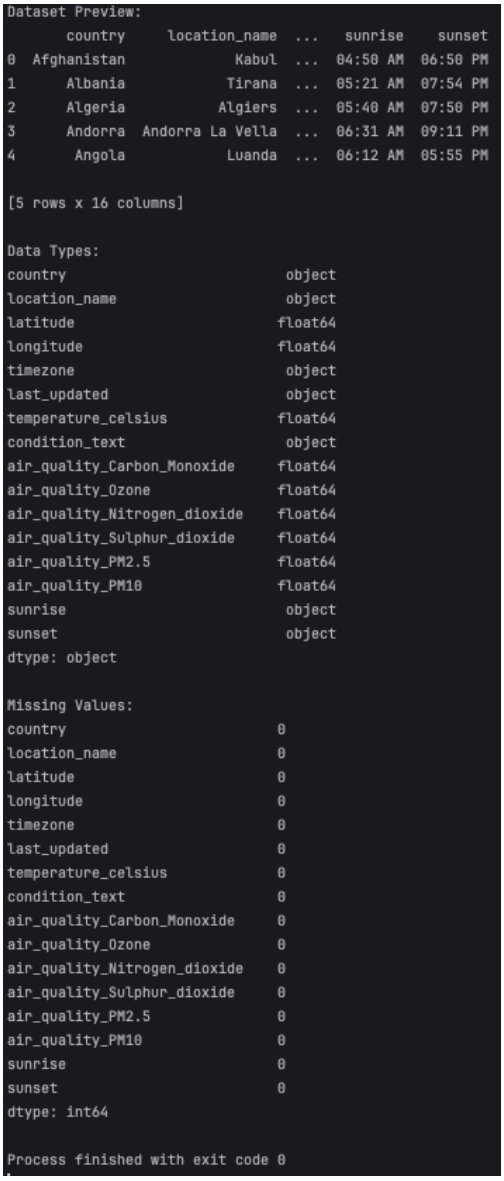
Weather data column will be encoded with one hot encoding in order to process this information in the model.

```python
def load_and_preprocess_data(file_path): # Step 1 -> Read the csv and pre process the dataset

    df = pd.read_csv(file_path, delimiter=';')

    df_original = df[['country', 'location_name']].copy() # Save original dataset for merging later

    def convert_to_seconds(time_str): # Convert AM and PM times to seconds to calculate daylight duration
        if pd.isna(time_str) or time_str.strip() == "":
            return np.nan  # Handle missing values
        dt = pd.to_datetime(time_str, format="%I:%M %p")
        return dt.hour * 3600 + dt.minute * 60 + dt.second


    df["sunrise"] = df["sunrise"].astype(str).apply(convert_to_seconds) # Convert sunrise & sunset to total seconds
    df["sunset"] = df["sunset"].astype(str).apply(convert_to_seconds)

    df['last_updated'] = pd.to_datetime(df['last_updated'], errors='coerce') # Convert last_updated column to datetime
    df["time_of_day"] = df["last_updated"].dt.hour * 3600 + df["last_updated"].dt.minute * 60 + df["last_updated"].dt.second
    df["daylight_duration"] = np.round((df["sunset"] - df["sunrise"]) / 3600, decimals: 1) # Compute daylight duration in hours


    df["is_daylight"] = (df["time_of_day"] >= df["sunrise"]) & (df["time_of_day"] <= df["sunset"]) # If daylight:1 if not 0
    df["is_daylight"] = df["is_daylight"].astype(int)

    # Extracting time based values
    df['year'] = df['last_updated'].dt.year
    df['month'] = df['last_updated'].dt.month
    df['day'] = df['last_updated'].dt.day
    df['hour'] = df['last_updated'].dt.hour

    df["condition_text"] = df["condition_text"].astype('category').cat.codes # Encode categorical weather conditions

    df = df.drop( labels: ["country", "location_name", "timezone", "last_updated"], axis=1, errors='ignore') # Drop all non numeric columns for model trainng

    print("\nData Types After Preprocessing:") # Print data types to verify all is numeric
    print(df.dtypes)

    return df, df_original
```

Figure 1.1: Preprocess data method

After creating a method for preprocessing data, output values are observed and no missing value exists in the dataset. (See Figure 1.2)

[3] Singh, B. (n.d.) 'Handling Missing Data with KNN Imputer', Medium. Available at: https://medium.com/@bhanupsingh484/handling-missing-data-with-knn-imputer-927d49b09015 (Accessed: 31 January 2025).



```
Dataset Preview:
       country     location_name  ...   sunrise    sunset
0  Afghanistan             Kabul  ...  04:50 AM  06:50 PM
1      Albania            Tirana  ...  05:21 AM  07:54 PM
2      Algeria           Algiers  ...  05:40 AM  07:50 PM
3      Andorra  Andorra La Vella  ...  06:31 AM  09:11 PM
4       Angola            Luanda  ...  06:12 AM  05:55 PM

[5 rows x 16 columns]

Data Types:
country                         object
location_name                   object
latitude                       float64
longitude                      float64
timezone                        object
last_updated                    object
temperature_celsius            float64
condition_text                  object
air_quality_Carbon_Monoxide    float64
air_quality_Ozone              float64
air_quality_Nitrogen_dioxide   float64
air_quality_Sulphur_dioxide    float64
air_quality_PM2.5              float64
air_quality_PM10               float64
sunrise                         object
sunset                          object
dtype: object

Missing Values:
country                        0
location_name                  0
latitude                       0
longitude                      0
timezone                       0
last_updated                   0
temperature_celsius            0
condition_text                 0
air_quality_Carbon_Monoxide    0
air_quality_Ozone              0
air_quality_Nitrogen_dioxide   0
air_quality_Sulphur_dioxide    0
air_quality_PM2.5              0
air_quality_PM10               0
sunrise                        0
sunset                         0
dtype: int64

Process finished with exit code 0
```

Figure 1.2: Output of data preprocessing method

As can be seen from Figure 1.2, float64 data typed values will be fed for model training. Object data types should be dropped or changed to a numerical value in order to be processed.

Outlier Analysis

To ensure that the dataset does not consist of extreme values for the air quality part, outlier analysis is implemented. Outliers might exist because of natural disasters or some errors of the tool that calculates air quality. If the outlier percentage is above %10, some of the outliers need to be handled in the dataset to train the model better.

```python
def check_outlier_severity(file_path): # Control Step -> Outlier severity check. If below 10% then do nothing.
    # Load dataset
    df = pd.read_csv(file_path, delimiter=';')

    # Air quaility columns
    pollution_cols = [
        "air_quality_Carbon_Monoxide",
        "air_quality_Ozone",
        "air_quality_Nitrogen_dioxide",
        "air_quality_Sulphur_dioxide",
        "air_quality_PM2.5",
        "air_quality_PM10"
    ]

    outlier_counts = {}

    # Count outliers with IQR method
    for col in pollution_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
        outlier_counts[col] = len(outliers)

    # Calculate outlier percentage
    total_rows = len(df)
    total_outliers = sum(outlier_counts.values())
    outlier_percentage = (total_outliers / (total_rows * len(pollution_cols))) * 100

    # Print results
    print("\n*Outlier Summary")
    for col, count in outlier_counts.items():
        print(f"{col}: {count} outliers")

    print(f"\nTotal Outliers: {total_outliers}")
    print(f"Outlier Percentage: {outlier_percentage:.2f}% of the dataset")

    # Determine severity level
    if outlier_percentage > 10:
        print("\nCritical Outlier Level! More than 10% of data points are outliers.")
    else:
        print("\nOutliers are within an acceptable range. No critical issue detected")
```

Figure 1.3: Outlier analysis

IQR method is used to evaluate outlier results. Outliers are defined when compared to the middle value. If it is 1.5 times smaller or bigger then marked as outlier and counted in Q1-Q3 variables. Total outlier percentage is calculated as 9.93% which is below %10 but at the edge.

Model Selection

As it is stated in the introduction section, the dataset has continuous data of air quality from historical records of different times and locations. Regression models are able to predict continuous data with complex environmental relationships like temperature, season or daylights in this case.

To decide model selection, different regression algorithms are compared. Linear regression can not handle complex relationships therefore eliminated. Random forest can be a good option in terms of required time for model and non linear relationships. Long short term memory (LSTM) networks can be used but require more time, data and might be too advanced for this problem. Alternative to LSTM, gradient boosting can be used for the same reasons of random forest model.

Random Forest

Random forest algorithm uses decision trees and compares with predicted results to improve accuracy and overcome the issue of overfitting.

```
# Step 2: Train test Split
def split_data(df, target_column):
    X = df.drop([target_column], axis=1)
    y = df[target_column]
    return train_test_split( *arrays: X, y, test_size=0.2, random_state=42)

# Step 3: Important features selection
def select_important_features(rf_model, X_train):
    feature_importances = pd.DataFrame({
        'Feature': X_train.columns,
        'Importance': rf_model.feature_importances_
    }).sort_values(by='Importance', ascending=False)

    print("--- Feature Importance ---")
    print(feature_importances)

    # Adjust threshold for feature selection
    important_features = feature_importances[feature_importances['Importance'] > 0.005]['Feature']
    return important_features

```

Figure 2.1: Split and select features before training

20% of the test data is extracted and 80% is fed for model training in the split_data method. Most important columns' coefficients are calculated and printed in the select_important_features method. If feature importance is below 0.005 then ignored as training attribute.

```
--- Feature Importance ---
                          Feature  Importance
8                 air_quality_PM10    0.628852
0                         latitude    0.164079
4      air_quality_Carbon_Monoxide    0.104328
2             temperature_celsius    0.012502
1                        longitude    0.011093
7      air_quality_Sulphur_dioxide    0.010071
5                air_quality_Ozone    0.009995
9                          sunrise    0.008261
15                           month    0.008174
6    air_quality_Nitrogen_dioxide    0.007832
16                             day    0.007227
10                          sunset    0.007069
11                  time_of_day    0.006964
12              daylight_duration    0.005197
3                  condition_text    0.003658
13                    is_daylight    0.002360
17                           hour    0.002055
14                           year    0.000283
```

Figure 2.2: Output of feature importance

As can be seen from Figure 2.2, some columns feature importance value below 0.005 therefore they are not included in the model training. Most important information for training is PM10 air quality value because it is the most related substance to PM2.5.

```python
def train_and_evaluate_random_forest(X_train, X_test, y_train, y_test): # Step 4 -> Train and Evaluate Random Forest
    rf_model = RandomForestRegressor(
        n_estimators=300,
        max_depth=None,
        min_samples_split=5,
        random_state=42,
        n_jobs=-1
    )

    rf_model.fit(X_train, y_train) # Train the model

    # Feature selection implementation
    important_features = select_important_features(rf_model, X_train)
    X_train_important = X_train[important_features]
    X_test_important = X_test[important_features]

    # Retrain the model on selected features
    rf_model.fit(X_train_important, y_train)
    y_pred = rf_model.predict(X_test_important)

    # Evaluate the model
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print("\n--- Random Forest Model Performance ---")
    print(f"RMSE: {rmse:.4f}")
    print(f"MAE: {mae:.4f}")
    print(f"R²: {r2:.4f}")

    return rf_model, y_pred, rmse, mae, r2
```

Figure 2.3: Random forest training

Random forest model training method is created in Figure 2.3. 300 decision trees and at least 5 samples per node is required to prevent overfitting. After training, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R^2 Score is calculated:

RMSE: 14.2248
MAE: 2.8967
R^2: 0.9179

RMSE shows that big mistakes exist and predictions are far off 14.2 while MAE is about 2.8 which is better. R^2 explains how the model is fit with predictions and it has a strong predict accuracy.

```python
def save_output_predictions(y_test, y_pred, rmse, mae, r2, output_file): # Step 6 -> Save output predictions csv file


    output_df = pd.DataFrame({  # Create DataFrame for actual vs predicted values
        "Actual_PM2.5": y_test.values,
        "Predicted_PM2.5": y_pred
    })


    metrics_df = pd.DataFrame({ # Create DataFrame for model metrics
        "Actual_PM2.5": ["Evaluation Metrics"],
        "Predicted_PM2.5": [""],
        "RMSE": [rmse],
        "MAE": [mae],
        "R²": [r2]
    })

    output_df = pd.concat( objs: [output_df, metrics_df], ignore_index=True) # Append metrics to the dataset

    output_df.to_csv(output_file, index=False) # Save to CSV

    print(f"\nOutput predictions and metrics saved to: {output_file}")

def save_edited_dataset(df, original_columns, output_file_edited_dataset): # Step 7 -> Save processed dataset

    edited_df = pd.concat( objs: [original_columns, df], axis=1) # Restore country & location_name to processed dataset

    edited_df.to_csv(output_file_edited_dataset, index=False)     # Save to CSV

    print(f"\nEdited dataset saved to: {output_file_edited_dataset}")
```

Figure 2.4: Saving edited data and predictions csv files

For predictions and edited dataset to be saved 2 different methods are created and saved to local files in order to import csv files to tableau dashboard for visualizations.

Gradient Boosting Regression

Gradient boosting regression (GBR) is another decision tree machine learning algorithm. The difference with random forest is that GBR creates multiple trees in parallel and learns from mistakes. The aim is to move accuracy even higher than random forest evaluation results.

```python
def train_and_evaluate_gradient_boosting(X_train, X_test, y_train, y_test): # Step 8 -> Train and Evaluate Gradient Boosting
    gbr_model = GradientBoostingRegressor(
        n_estimators=300,
        learning_rate=0.1,
        max_depth=5,
        random_state=42
    )

    #Train the model
    gbr_model.fit(X_train, y_train)

    # Extract important features
    important_features = select_important_features_gbr(gbr_model, X_train)
    X_train_important = X_train[important_features]
    X_test_important = X_test[important_features]

    # Train the model again using only selected features
    gbr_model.fit(X_train_important, y_train)

    # Make predictions
    y_pred = gbr_model.predict(X_test_important)

    # Evaluate model performance
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print("\n--- Optimized Gradient Boosting Performance ---")
    print(f"RMSE: {rmse:.4f}")
    print(f"MAE: {mae:.4f}")
    print(f"R²: {r2:.4f}")

    return gbr_model, y_pred, rmse, mae, r2

```

Figure 2.5: Gradient boosting regression training

Similar to random forest, 300 decision trees are used to train the model with feature selection method. After training, RMSE, MAE, and R^2 Score is calculated:

RMSE: 17.7490
MAE: 3.2976
R^2: 0.8721

When compared to random forest evaluation metrics, GBR is trained slightly worse than random forest algorithms. Possible reason might be GBR learns from mistakes and the dataset is not large enough to train the model properly. Another reason could be related to GBR being more sensitive to extreme values.

Data Analysis and Visualization of Air Quality

Tableau tool is used to create visualizations. Categorical data was transformed to numerical data to train the model previously. Now Categorical data is decrypted to text format in Tableau by creating calculated fields. To minimize visual details, some of the categorical data combine together, for instance in the dataset 44 types of weather were indicated. At the end it is reduced to 5:

Category	**Numerical Values**
Clear/Sunny	2, 30, 40
Cloudy/Overcast	3, 28, 29
Rain/Thunderstorms	7, 8, 10, 12, 13, 18, 19, 20, 21, 25, 26, 32, 33, 34, 35, 38, 39, 41, 42, 43
Snow/Blizzards	0, 1, 5, 6, 9, 11, 14, 15, 16, 17, 22, 23, 24, 27, 31, 36, 37
Fog/Mist	4, 18

Table 1: Tableau weather calculated field

Similar to weather to have color visuals according to pollution level, pollution category is created in Tableau.

World Map - Pollution of PM2.5

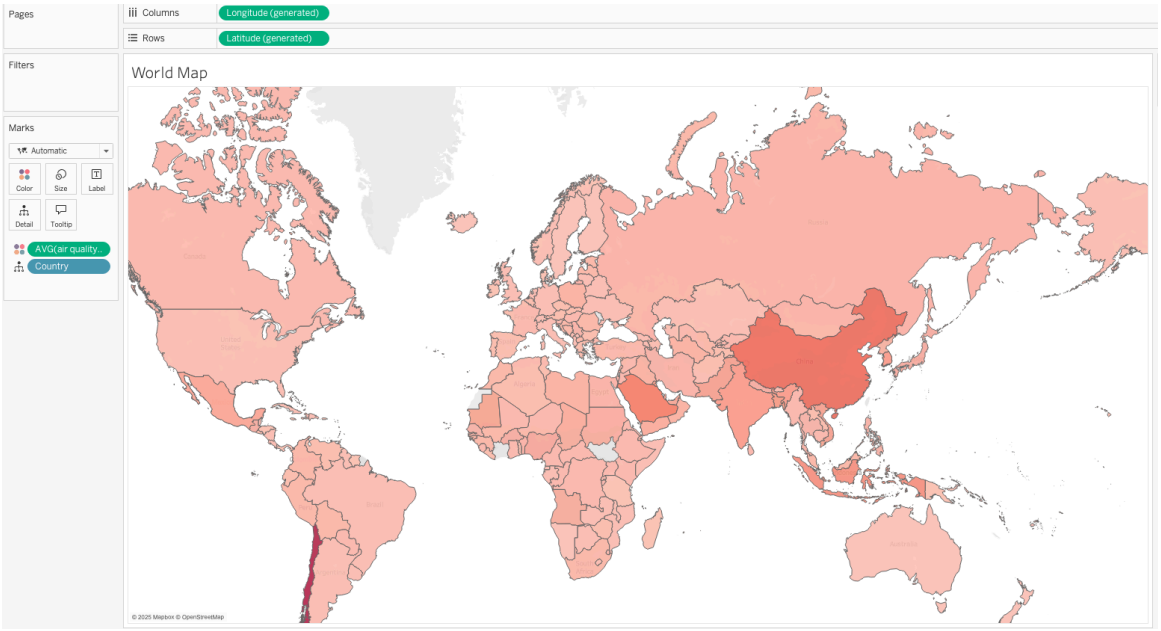


Figure 3.1: World map

World map is created to see which locations have the highest air pollution. As can be seen from figure 3.1, Chile has the highest average PM2.5 air quality.

Season Trends of Air Quality

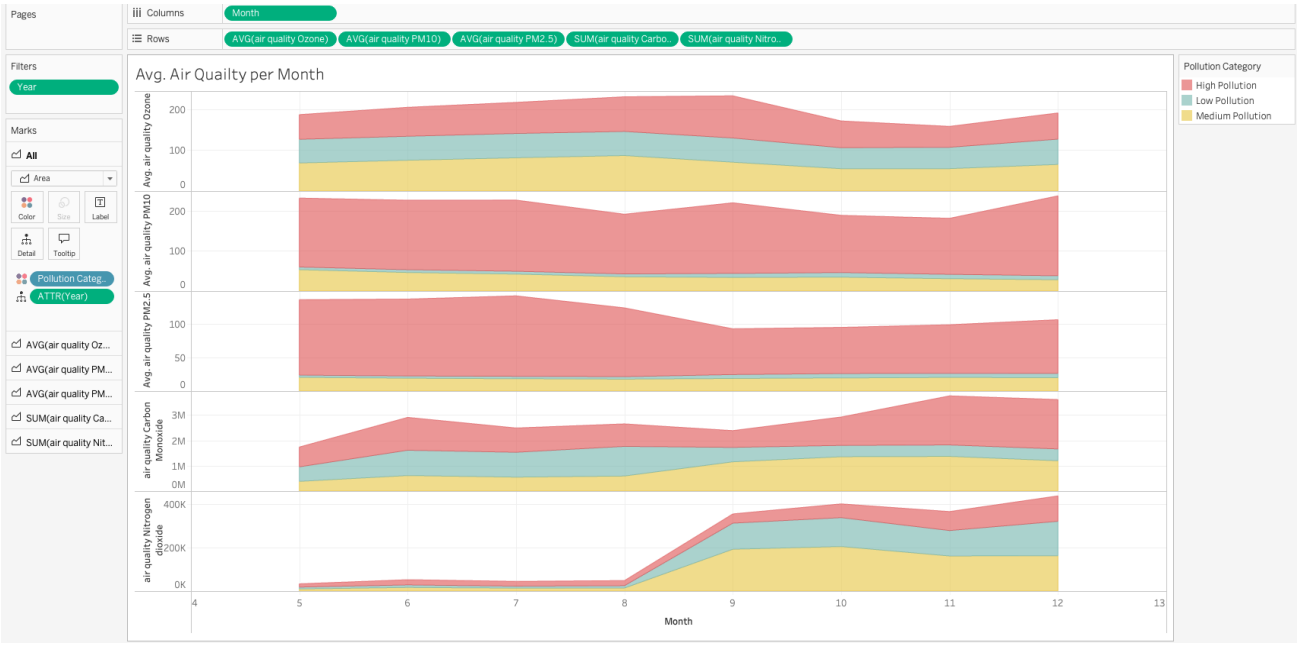


Figure 3.2: Air quality per month

When season trends are visualized with all types of air qualities, there are some key trends depending on the season:

- NO_2 : After August it is having a huge increase throughout the whole autumn and winter season which might be related with human heating system behaviors.
- CO: Shows a small pattern of increase in autumn.
- PM2.5 - PM10: Shows similar patterns in terms of monthly increase and especially peaking during summer. PM2.5 is dependent on various variables such as air quality types, weather conditions, wind, daylight, time of the day, industries etc.

Temperature and Air Quality

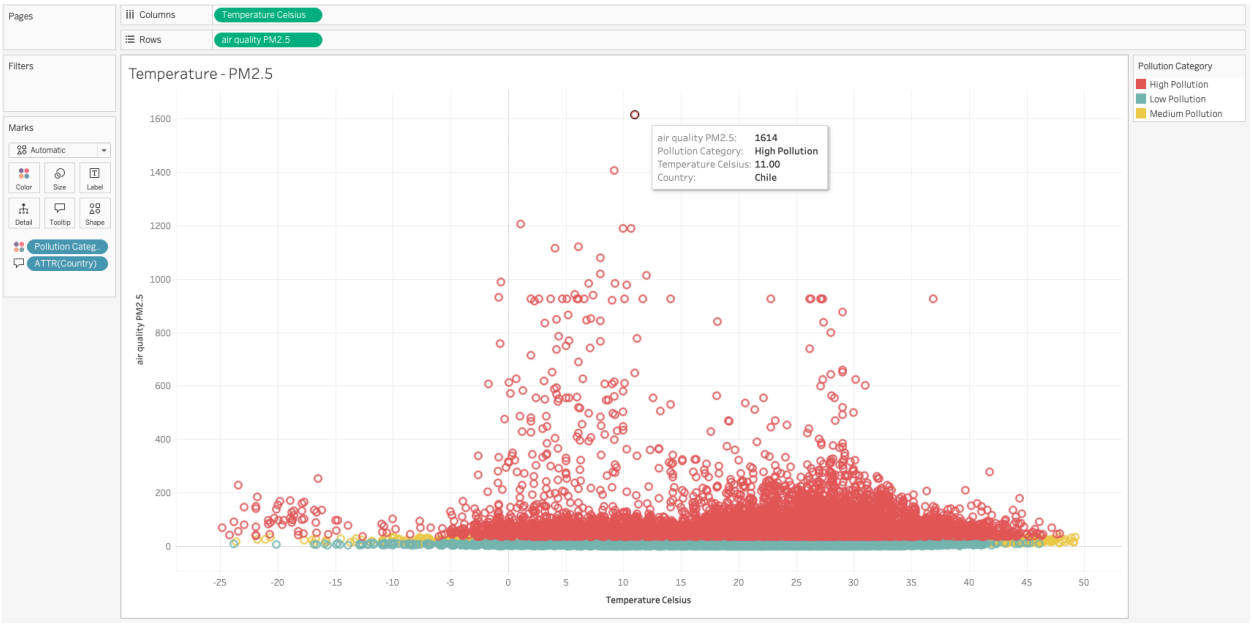


Figure 3.3: Temperature and PM2.5

In this scatter plot, temperature and PM2.5 relation can not be seen. Although there are some patterns increasing in the mid ranged area, there is not a specific correlation between temperature and PM2.5.

PM2.5 Level per Hour

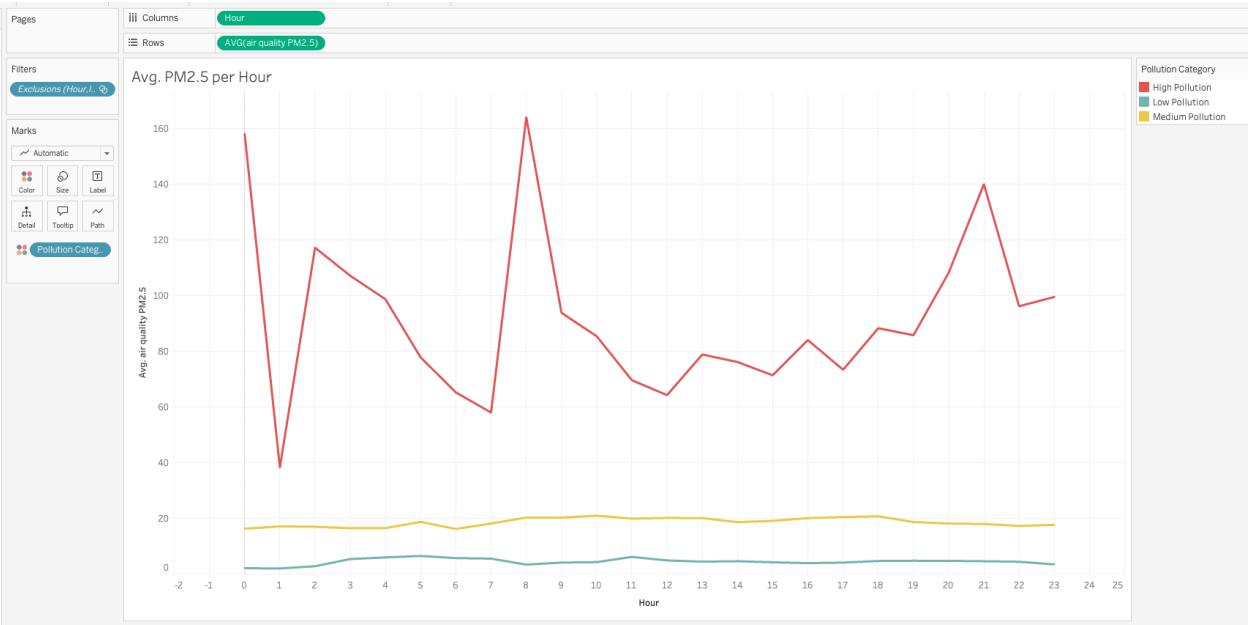


Figure 3.4: Average PM2.5 per hour

In this line chart, the peak hours of the pollution spikes can be observed which are 0, 8 and 21. This might be suggesting due to traffic and work hours of humans pollution level increasing. There is also a significant drop from 2 to 6 in the morning which might show that it is decreasing when human activity is at minimal.

Top Countries with Average PM2.5

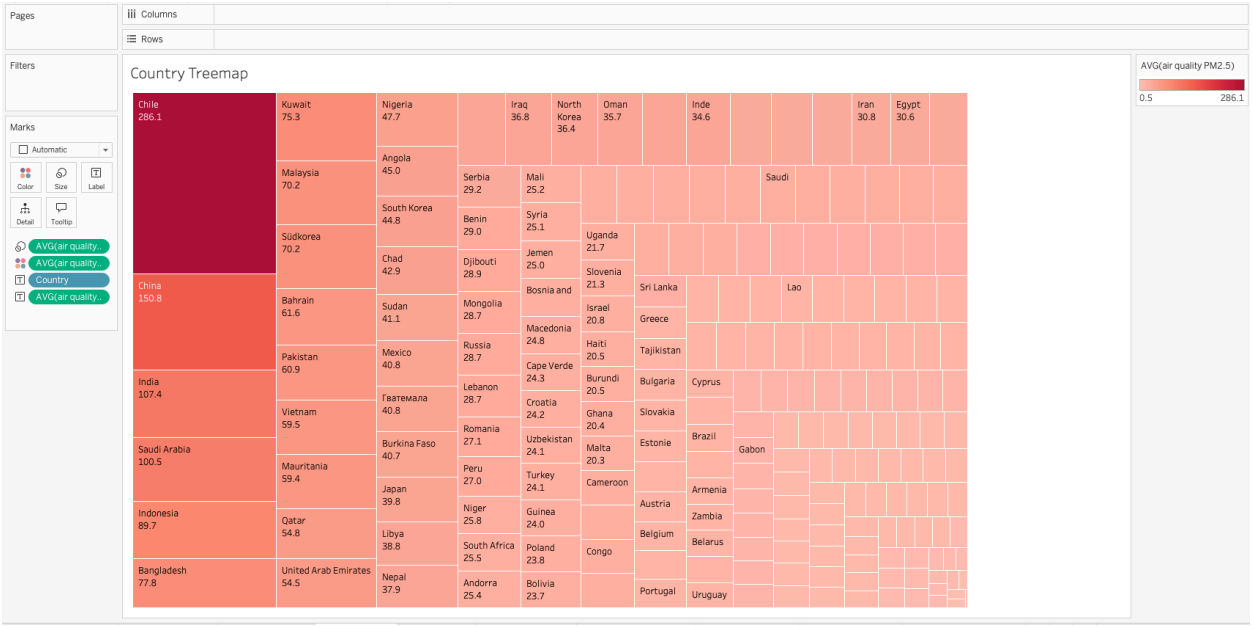


Figure 3.5: Top polluted countries

In this treemap, average PM2.5 air quality with country labels can be observed. Top 5 polluted level countries can be listed as following:

1. Chile - 286.1
2. China - 150.8
3. India - 107.4
4. Saudi Arabia - 100.5
5. Indonesia - 89.7

Weather Condition and Average PM2.5

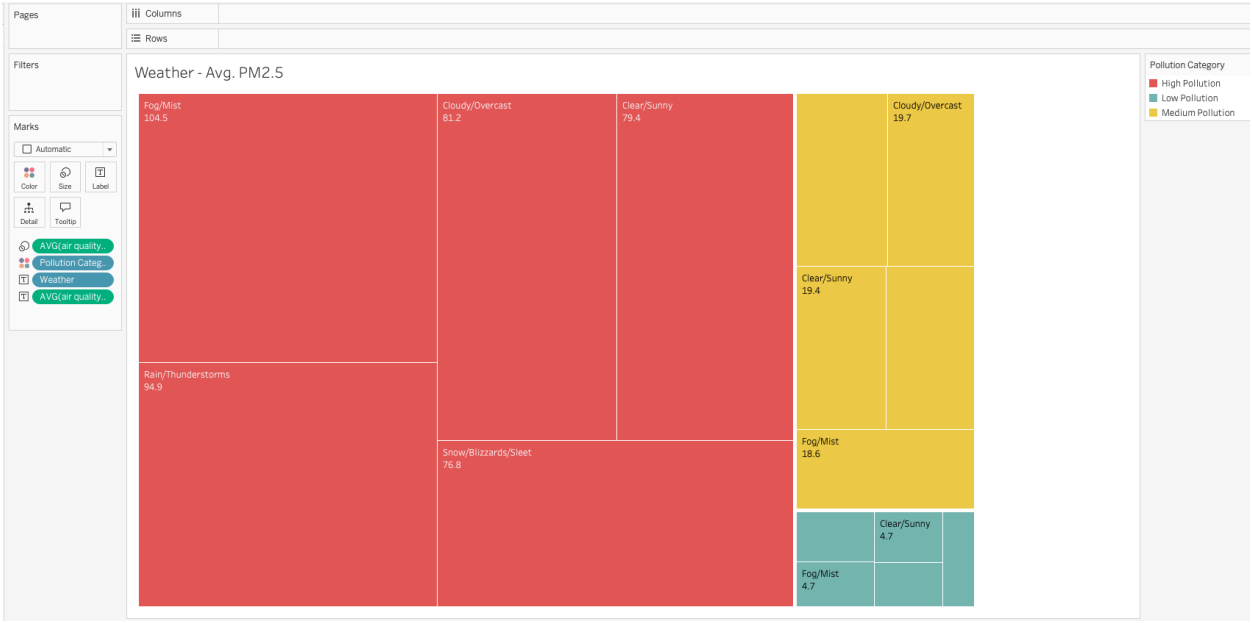


Figure 3.6: Weather and PM2.5

In Figure 3.6 treemap, weather condition patterns with PM2.5 air quality are aimed to be observed. Fogged weather has the highest PM2.5 value which could be related with air quality device sensors during foggy weather. When only compared with highly polluted areas, the least weather condition is sunny and snowy. Weather conditions can be related to air quality including other variables. But there is certainly not a direct relationship.

PM2.5 and PM10 Relation

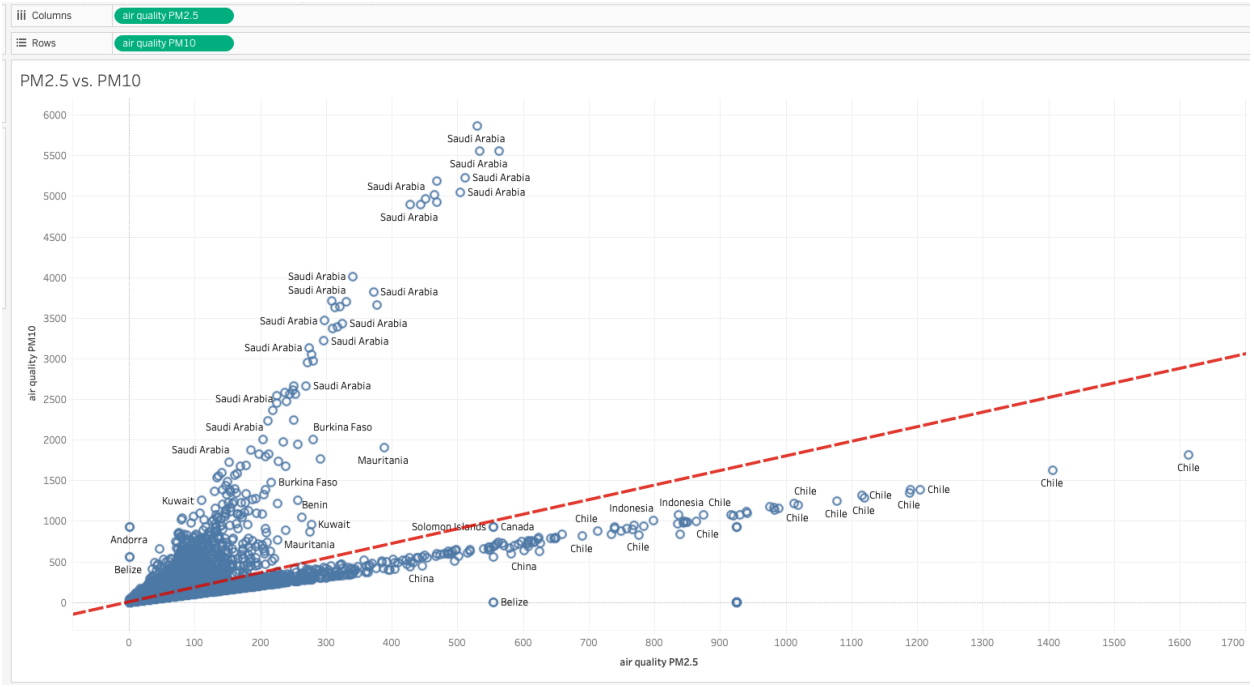


Figure 3.7: PM2.5 vs. PM10

In this scatter plot PM2.5 and PM10 air quality is compared. There is certainly a close relationship between these air qualities. For Saudi Arabia while the PM10 value is very high, PM2.5 is not that high as expected. This could be related to the dusty conditions in that location. The opposite relationship can be observed in Chile as well.

Top 5 Polluted Countries

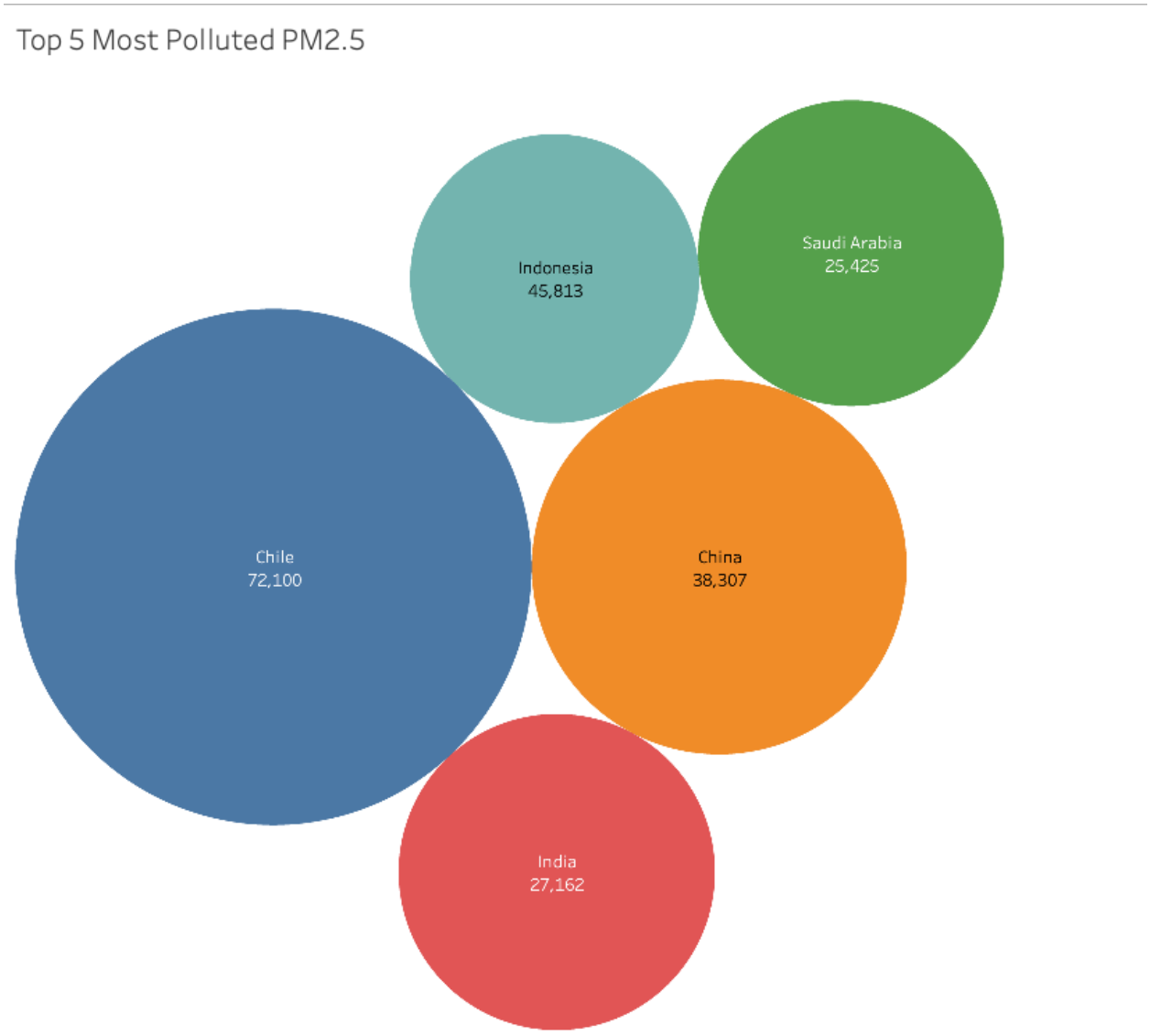


Figure 3.8: Top 5 polluted countries

As it was observed in Figure 3.8, most polluted countries are compared in packed bubbles visually. Chile has the extreme value of 72 PM2.5 level therefore Chile visualizations for deeper analysis will be created.

Chile - Air Quality per Month

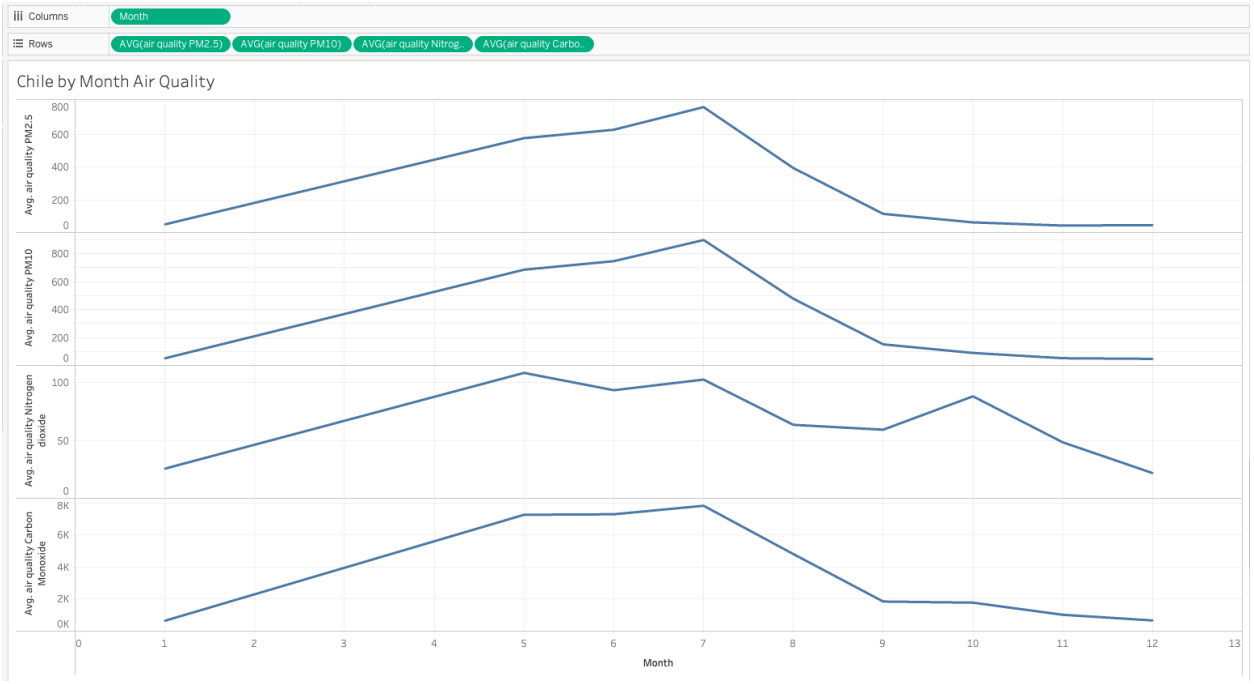


Figure 3.9: Chile - air quality per month

Country Chile is on the south side of the equator therefore during May, June, July and August is cold weather for Chile. In this air quality line chart, polluted air increase trend can be observed starting from May to September. This might be explained with possible cold weather conditions in Chile.

Chile - Temperature vs. PM2.5

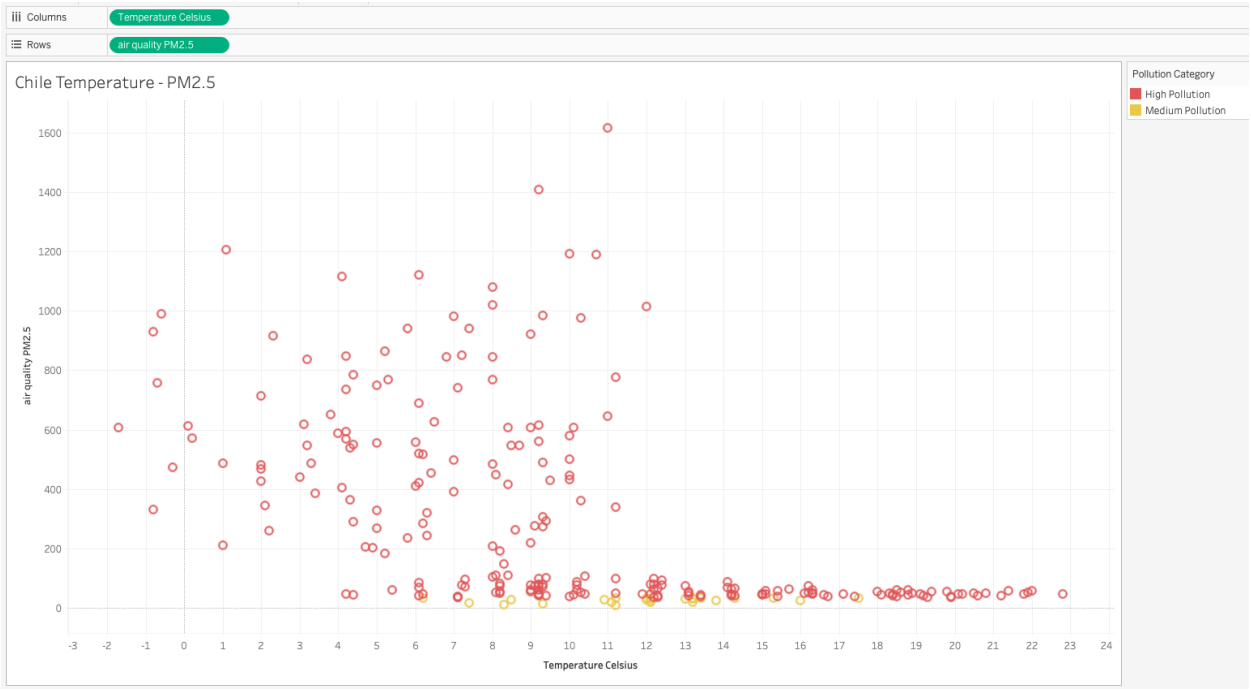


Figure 3.10: Chile - temperature comparison

As it is stated in Figure 3.9, temperature decrease might be affecting bad pollution in Chile. To be able to understand this temperature - PM2.5 scatter plot is drawn. It is suggesting that temperature decrease has significant importance in Chile which results in an extreme amount of PM2.5.

Chile - PM2.5 and Weather Count

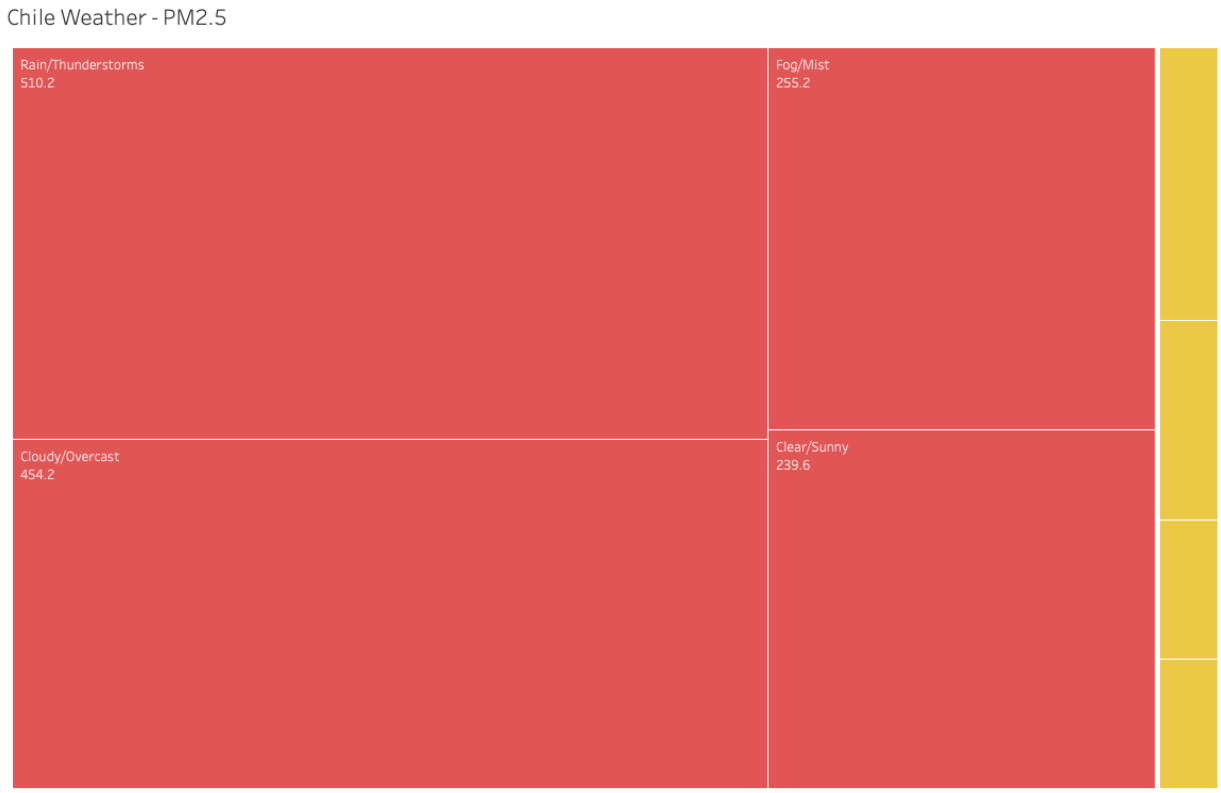


Figure 3.11: Chile - weather and PM2.5

When weather is compared in Chile, most polluted data is when rainy or overcasted days which again could be related with the bad weather conditions during winter.

Dashboard

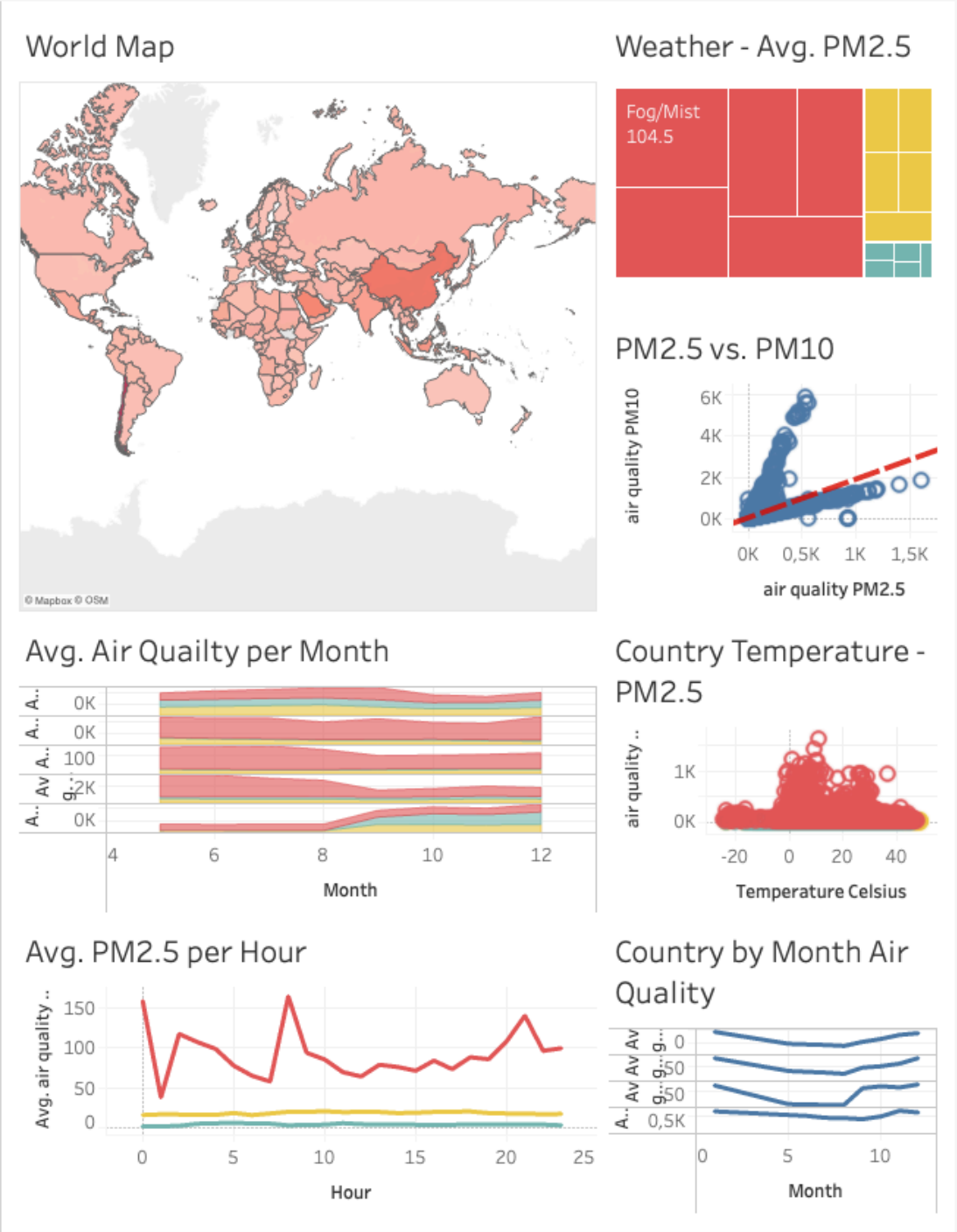


Figure 3.12: Interactive dashboard

In the created dashboard, instead of using filters, world map visuals can be used as filters directly. When any country is selected, the other visuals will be filtered for that country and changes can be observed.

Evaluation Metrics

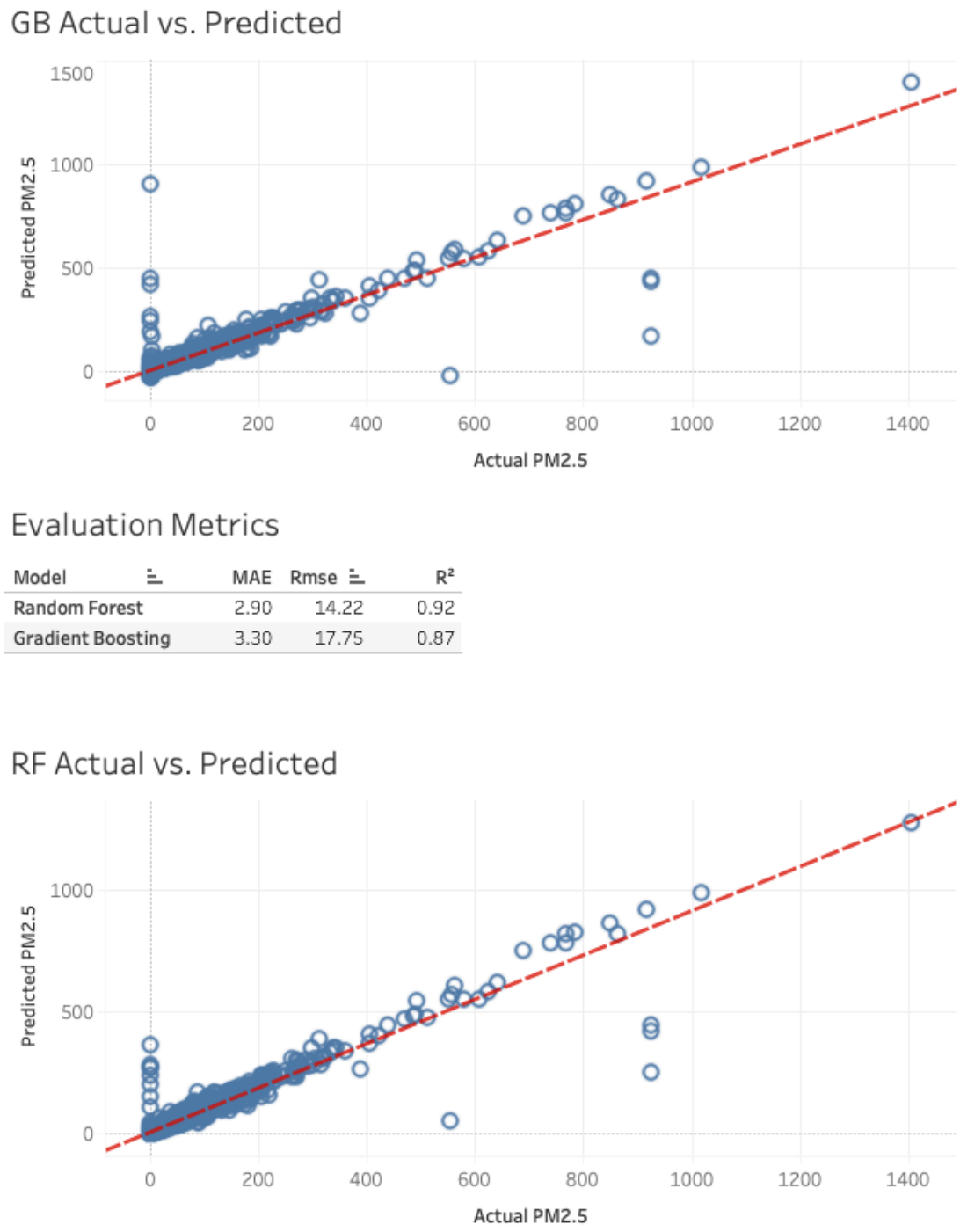


Figure 3.13: Evaluation metrics

Actual vs. predicted PM2.5 values with best fit line can be observed in tableau dashboard. Best fit line is used from the trend line feature and it is showing a smaller angle. It could be internal calculations of tableau. Outliers can be observed here as well.

Conclusion

Air quality data analysis project aimed to analyze pollution in the world with machine learning techniques and visualize findings. In this project PM2.5 air quality is selected as target variable and the model is trained with random forest and gradient boosting. For data analysis visualization, tableau tool is used with csv datasets.

Some of the key findings related with model training suggest that for this particular dataset random forest succeeds better than gradient boosting algorithm with higher R^2 and lower RMSE values. One of the most important impacts to increase the evaluation metrics of the model was using feature importance for random forest. It has increased R^2 value ~4.5%. There were challenges while preprocessing the dataset as well especially with date & time information. Because of data types of hour, date is not the same there were a lot of faced issues while calculating daylight duration.

After visualization in Tableau some of the trends were observed related with air quality in the world. World map showed some of the hotspots like Chile, China, India, Saudi Arabia. Pollution levels have strong patterns with seasonal activities of humans. While weather is slightly related to pollution, no direct relation was able to be found unlike temperature. Temperature showed significant patterns with air quality in the world. One of the obvious patterns was that PM2.5 and PM10 levels.

In order to develop further this study, a larger with more specific information is needed. Additional information like traffic result, industry emission, wind speed, population data can be imported. In that way the model will be able to predict higher accuracy scores.

In the high risk polluted spots governments and public communities should define a strategy to reduce pollution and act accordingly.

References

1. World Health Organization, 2016. Ambient air pollution: A global assessment of exposure and burden of disease. World Health Organization. Available at: https://iris.who.int/bitstream/handle/10665/250141/9789241511353-eng.pdf?sequence=1 [Accessed 26 January 2025].

2. McDuffie, E.E., Martin, R.V., Spadaro, J.V., Burnett, R., Smith, S.J., O'Rourke, P., Hammer, M.S., van Donkelaar, A., Bindle, L., Shah, V., Jaeglé, L., Luo, G., Yu, F., Adeniran, J.A., Lin, J. and Brauer, M., 2021. Source sector and fuel contributions to ambient PM2.5 and attributable mortality across multiple spatial scales. Nature Communications, [online] 12(1), pp.1. Available at: https://www.nature.com/articles/s41467-021-23853-y [Accessed 26 January 2025].

3. Singh, B. (n.d.) 'Handling Missing Data with KNN Imputer', Medium. Available at: https://medium.com/@bhanupsingh484/handling-missing-data-with-knn-imputer-927d49b09015 [Accessed: 31 January 2025].

Appendix

https://github.com/Q1065727/AirQuality