Internship Report: Air Quality Data Analysis

Student Name: Nikhil Barman

University: Guru Ghasidas Vishwavidyalaya, Bilaspur,Chhattisgarh

Major: Computer Science and Application

Internship Duration: February 1st, 2025- Februaryt 28th, 2025

Company: ShadowFox

Domain: Data Science

Mentor: Mr. Hariharan

Coordinator: Mr. Aakash

Objectives My primary objectives for this internship were to:

1. Develop a comprehensive understanding of data science methodologies and practices, specifically in air quality analysis.
2. Gain hands-on experience in data analysis, statistical modeling, and visualization of air pollution data.
3. Enhance my skills in using data science tools and techniques to extract insights from complex air quality datasets.

Tasks and Responsibilities During my internship, I was involved in the following key tasks related to air quality analysis:

- **Data Collection and Cleaning**: Collected and preprocessed raw air quality data to ensure its accuracy and usability. This involved handling missing values, removing outliers, and standardizing data formats.

- **Exploratory Data Analysis (EDA)**: Conducted EDA to identify trends and patterns in pollutants such as PM2.5, PM10, NO2, SO2, and O3.

Utilized descriptive statistics and visualizations to understand pollution variations over time.

- **Statistical Modeling**: Developed and evaluated statistical models to predict air quality index (AQI) levels based on various pollutant concentrations. Applied regression and classification techniques to understand pollutant relationships.

- **Data Visualization**: Created interactive dashboards and visual reports using Matplotlib and Tableau to communicate findings effectively. Used time-series graphs and heatmaps to showcase pollution trends.

- **Machine Learning Implementation**: Implemented predictive modeling using machine learning algorithms to forecast air pollution levels. Evaluated model performance with metrics like Mean Absolute Error (MAE) and R-squared value.

- **Report Generation**: Compiled comprehensive reports detailing the air quality analysis, insights, and recommendations for mitigating pollution impacts.

Learning Outcomes Through this internship, I gained valuable experience and skills, including:

- **Technical Proficiency**: Developed expertise in data science tools such as Pandas, Matplotlib, Scikit-learn, and ARIMA for time-series forecasting.
- **Understanding of Air Quality Analysis**: Acquired in-depth knowledge of air pollutants, their sources, and their impact on health and the environment.
- **Analytical Skills**: Enhanced my ability to analyze large air quality datasets, identify key pollution trends, and apply appropriate statistical techniques.
- **Professional Development**: Improved my ability to communicate technical findings, collaborate with team members, and manage multiple tasks in a data-driven project environment.

Challenges and Solutions

- **Handling Large Datasets**: Processing vast amounts of air quality data was challenging. I optimized workflows by using efficient data handling techniques and parallel processing.
- **Model Accuracy and Validation**: Ensuring reliable AQI predictions required iterative model refinement. I employed cross-validation techniques and fine-tuned model parameters to improve accuracy.

Conclusion My internship in air quality data analysis provided invaluable hands-on experience in data science. The exposure to various analytical techniques, statistical modeling, and machine learning applications has significantly enhanced my skills. This experience has strengthened my interest in pursuing a career in data science and prepared me for real-world data challenges.

Acknowledgments I extend my gratitude to ShadowFox, my mentor, Mr. Hariharan, and my coordinator, Mr. Aakash, for their guidance throughout this internship. I also thank Amrita Vishwa Vidyapeetham for providing this opportunity, which has been instrumental in my academic and professional growth.

This report highlights my learning journey, the integration of academic knowledge with practical experience, and my contributions to air quality data analysis during my internship.

```
import pandas as pd

# Load the dataset
file_path = "delhiaqi.csv"
df = pd.read_csv(file_path)

# Display basic information and the first few rows
df.info(), df.head()
```

Output

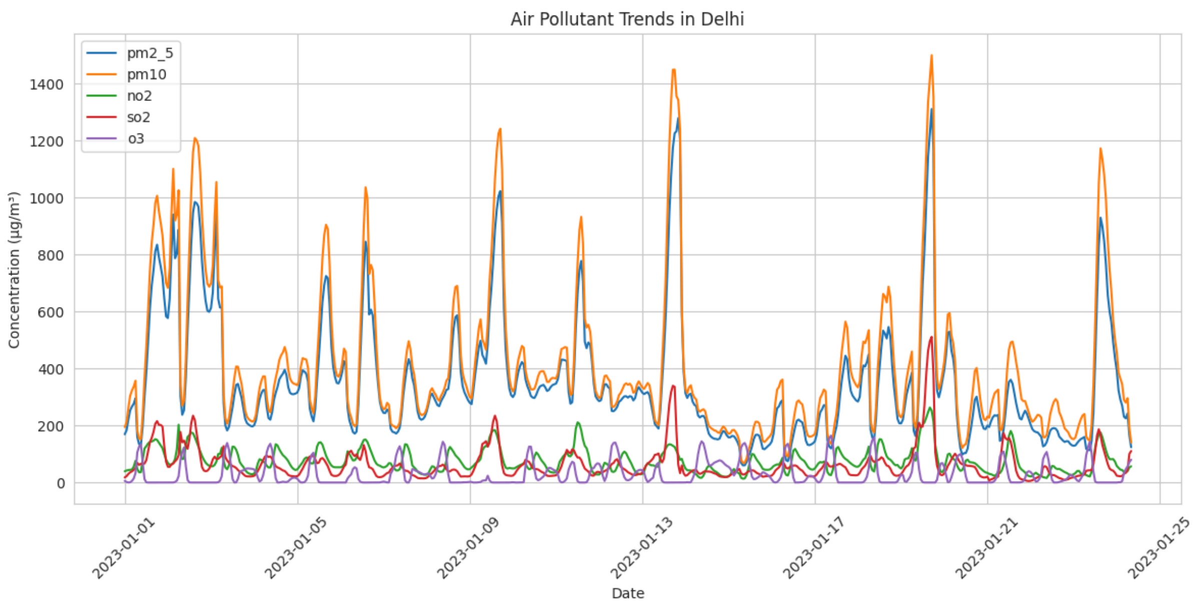
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 561 entries, 0 to 560
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    561 non-null    object
 1   co      561 non-null    float64
 2   no      561 non-null    float64
 3   no2     561 non-null    float64
 4   o3      561 non-null    float64
 5   so2     561 non-null    float64
 6   pm2_5   561 non-null    float64
 7   pm10    561 non-null    float64
 8   nh3     561 non-null    float64
dtypes: float64(8), object(1)
memory usage: 39.6+ KB
(None,
                   date       co     no    no2    o3    so2   pm2_5    pm10  \
 0  2023-01-01 00:00:00  1655.58   1.66  39.41  5.90  17.88  169.29  194.64
 1  2023-01-01 01:00:00  1869.20   6.82  42.16  1.99  22.17  182.84  211.08
 2  2023-01-01 02:00:00  2510.07  27.72  43.87  0.02  30.04  220.25  260.68
 3  2023-01-01 03:00:00  3150.94  55.43  44.55  0.85  35.76  252.90  304.12
 4  2023-01-01 04:00:00  3471.37  68.84  45.24  5.45  39.10  266.36  322.80

      nh3
 0   5.83
 1   7.66
 2  11.40
 3  13.55
 4  14.19  )
```

```
# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Check for missing values
missing_values = df.isnull().sum()

# Display the updated dataset info and missing values
df.info(), missing_values
```

Output

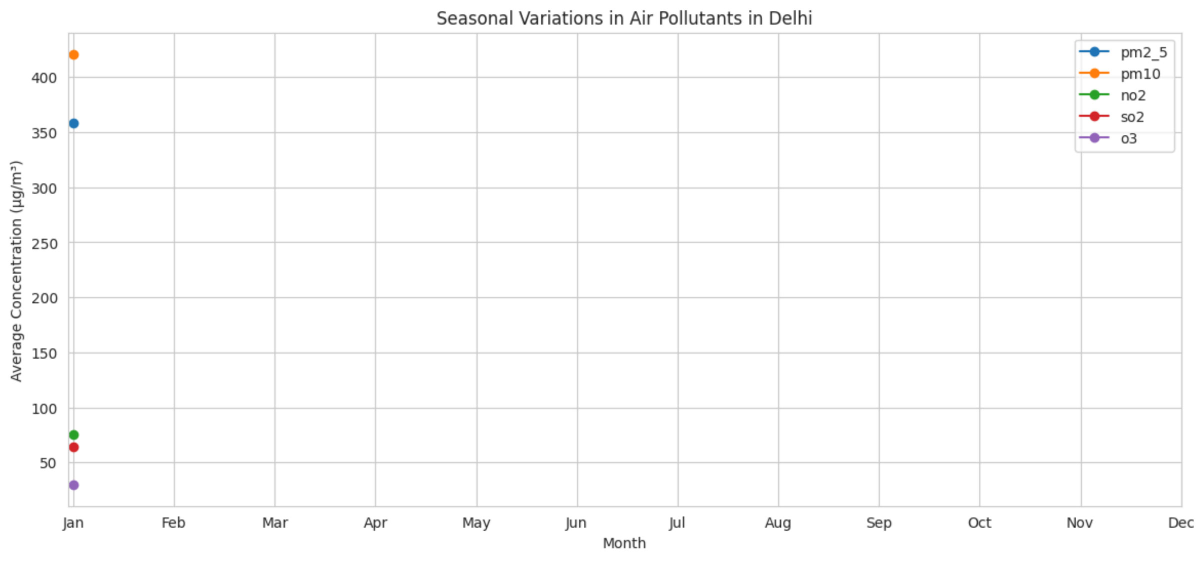
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 561 entries, 0 to 560
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    561 non-null    datetime64[ns]
 1   co      561 non-null    float64
 2   no      561 non-null    float64
 3   no2     561 non-null    float64
 4   o3      561 non-null    float64
 5   so2     561 non-null    float64
 6   pm2_5   561 non-null    float64
 7   pm10    561 non-null    float64
 8   nh3     561 non-null    float64
dtypes: datetime64[ns](1), float64(8)
memory usage: 39.6 KB
(None,
 date     0
 co       0
 no       0
 no2      0
 o3       0
 so2      0
 pm2_5    0
 pm10     0
 nh3      0
 dtype: int64)
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Summary statistics for pollutants
summary_stats = df.describe()

# Set plot style
sns.set_style("whitegrid")

# Plot time series of key pollutants
plt.figure(figsize=(14, 6))
for pollutant in ['pm2_5', 'pm10', 'no2', 'so2', 'o3']:
    plt.plot(df['date'], df[pollutant], label=pollutant)

plt.xlabel('Date')
plt.ylabel('Concentration (µg/m³)')
plt.title('Air Pollutant Trends in Delhi')
plt.legend()
plt.xticks(rotation=45)
plt.show()

# Display summary statistics
summary_stats
```

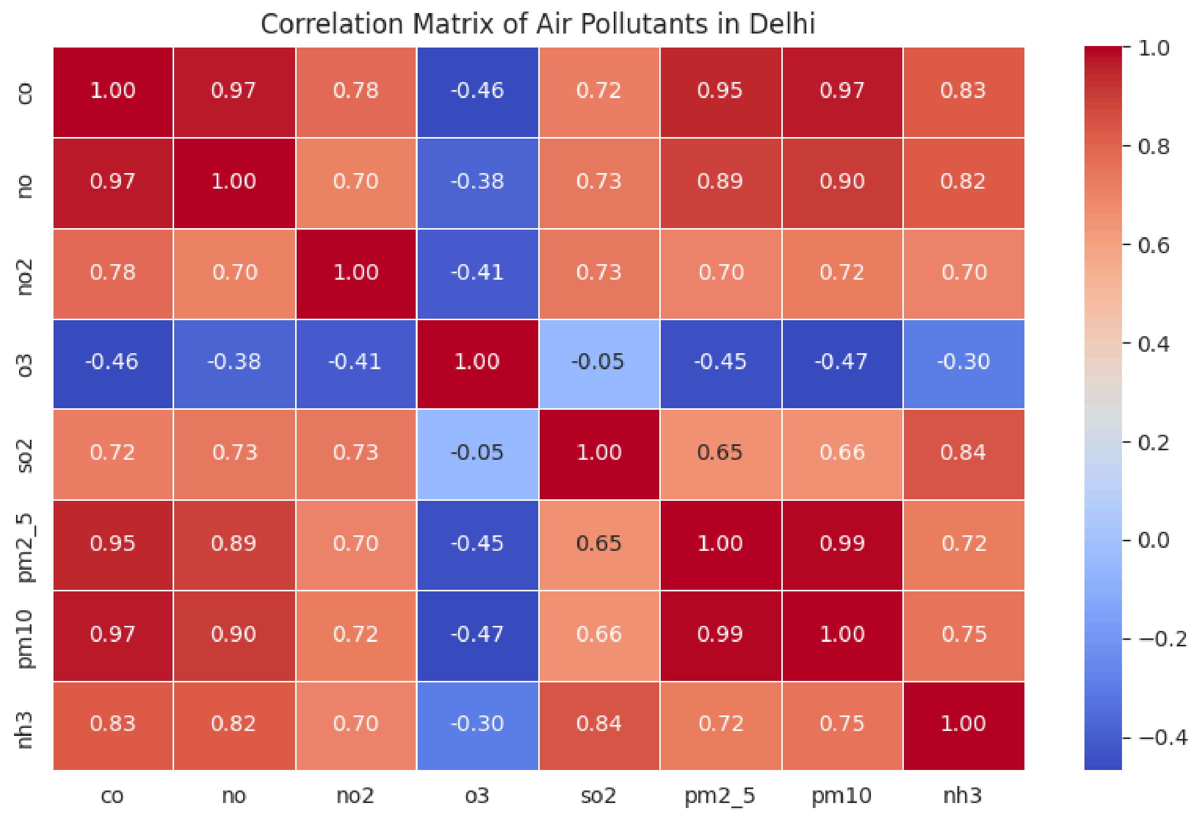


```
# Extract month and year for seasonal analysis
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year

# Group by month to find seasonal trends
monthly_avg = df.groupby('month').mean()

# Plot seasonal variations
plt.figure(figsize=(14, 6))
for pollutant in ['pm2_5', 'pm10', 'no2', 'so2', 'o3']:
    plt.plot(monthly_avg.index, monthly_avg[pollutant], marker='o', label=pollutant)

plt.xlabel('Month')
plt.ylabel('Average Concentration (µg/m³)')
plt.title('Seasonal Variations in Air Pollutants in Delhi')
plt.legend()
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
'Dec'])
plt.grid(True)
plt.show()
```

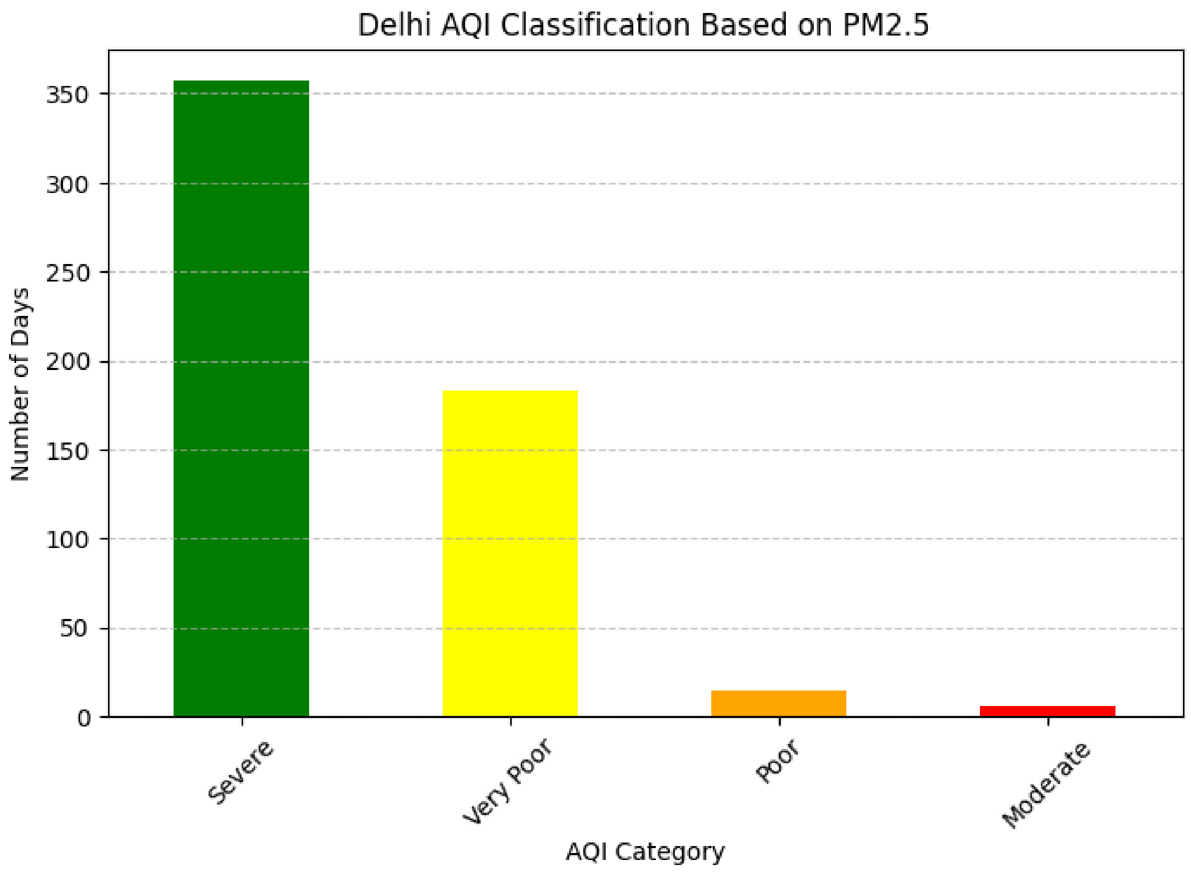
Output



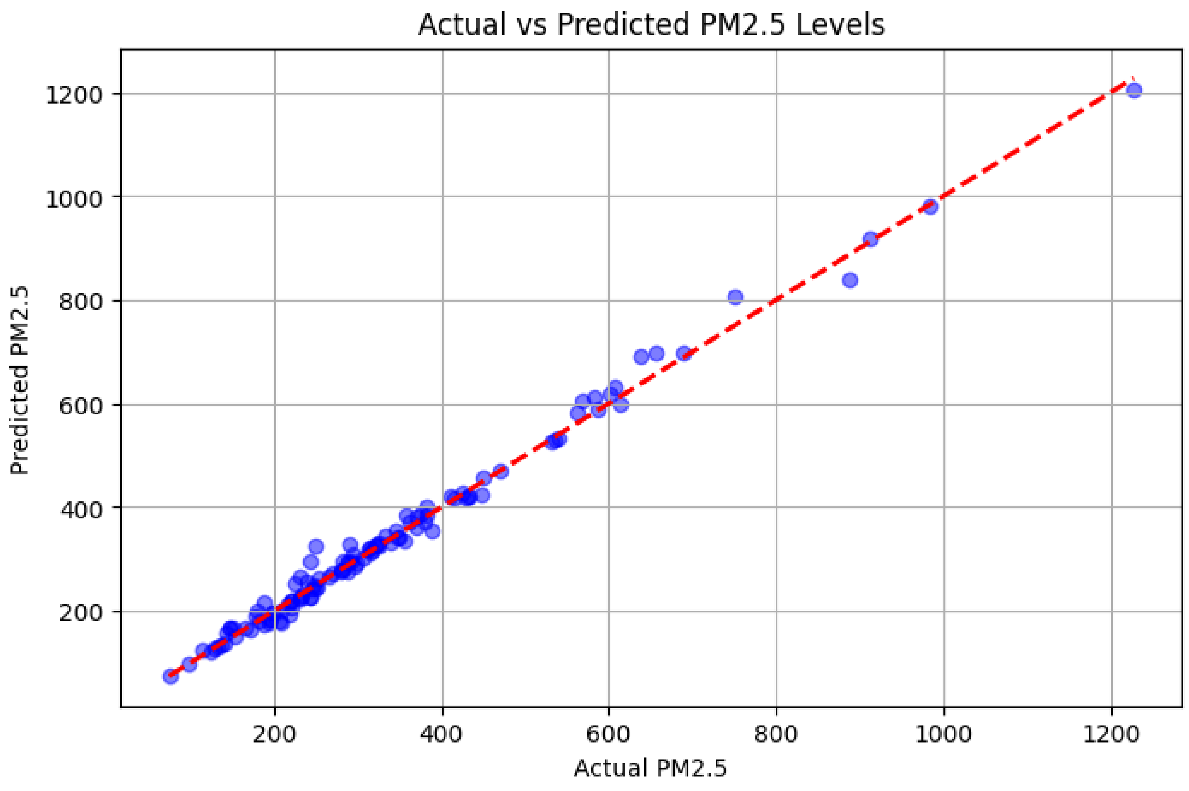
```
# Compute correlation matrix
corr_matrix = df.drop(columns=['date', 'month', 'year']).corr()

# Plot heatmap of correlations
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Air Pollutants in Delhi')
plt.show()
```



```
# Define AQI Classification
def classify_aqi(pm2_5):
    if pm2_5 <= 30:
        return "Good"
    elif pm2_5 <= 60:
        return "Satisfactory"
    elif pm2_5 <= 90:
        return "Moderate"
    elif pm2_5 <= 120:
        return "Poor"
    elif pm2_5 <= 250:
        return "Very Poor"
    else:
        return "Severe"

df['AQI_Category'] = df['pm2_5'].apply(classify_aqi)

# Plot AQI Distribution
plt.figure(figsize=(8, 5))
df['AQI_Category'].value_counts().plot(kind='bar', color=['green', 'yellow', 'orange', 'red',
'purple', 'brown'])
plt.xlabel("AQI Category")
plt.ylabel("Number of Days")
plt.title("Delhi AQI Classification Based on PM2.5")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
# Prepare Data for Prediction
features = ['no', 'no2', 'o3', 'so2', 'pm10', 'nh3']
target = 'pm2_5'
df_clean = df.dropna(subset=features + [target])
X_train, X_test, y_train, y_test = train_test_split(df_clean[features], df_clean[target],
test_size=0.2, random_state=42)

# Train Random Forest Model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict PM2.5
y_pred = model.predict(X_test)

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

print(f"Model Performance:\nMAE: {mae:.2f}\nRMSE: {rmse:.2f}\nR² Score: {r2:.2f}")

# Plot Actual vs Predicted
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.5, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], '--r', lw=2)
plt.xlabel("Actual PM2.5")
plt.ylabel("Predicted PM2.5")
plt.title("Actual vs Predicted PM2.5 Levels")
plt.grid(True)
plt.show()
```


Actual vs Predicted PM2.5 Levels
Predicted PM2.5
Actual PM2.5