**Introduction to Machine Learning**

 **Project Report**

**Submitted To:**

Ma’am Aimal Fatima Naqvi

**Submitted By:**

|  |  |
| --- | --- |
| **SAAD HAFEEZ** | **22-CS-142** |
| **SAAD GHAZANFAR** | **22-CS-114** |
| **MUHAMMAD SARIM SALEEM** | **22-CS-029** |

**Section**

**(B)**

**Department of Computer Science**

**HITEC University, Taxila**

**Project Report: Air Quality Analysis and Prediction**

**1. Introduction**

The aim of this project is to analyze air quality data and predict pollutant levels using various machine learning algorithms. This report outlines the data preprocessing, implementation of five machine learning models, and an in-depth discussion of the results. The chosen dataset, air\_quality\_synthetic.csv, provides measurements of pollutants such as PM10, PM2.5, and NO2, as well as environmental conditions like temperature and humidity.

**2. Data Analysis**

**2.1 Why Choose This Dataset?**

Air pollution is a critical environmental and health issue, affecting millions worldwide. The dataset captures essential pollutant levels and environmental conditions, making it ideal for:

* Understanding pollutant trends.
* Developing predictive models for pollution levels.
* Informing public health and environmental policies.

**2.2 Data Analysis Techniques**

**2.2.1 Dataset Overview**

* **Size**: 1000 records
* **Features**:
  + Pollutants: PM10, PM2.5, NO2, SO2, CO, O3
  + Environmental Factors: Temperature, Humidity
  + Station and Pollution Level (categorical)

**2.2.2 Frequency Tables**

The Pollution\_Level column is distributed as follows:

* Low: 30%
* Moderate: 50%
* High: 20%

**2.2.3 Statistical Analysis**

* **Covariance**: Indicates relationships between pollutants, e.g., PM10 and PM2.5 show a strong positive relationship.
* **Correlation Matrix**:
  + Positive Correlation: PM2.5 and CO (0.78)
  + Negative Correlation: Humidity and O3 (-0.65)

**2.2.4 Data Visualization**

* Histograms for pollutant distributions.
* Heatmap for correlations among numerical features.
* Boxplots to detect outliers in PM10 and PM2.5 levels.

**2.2.5 Explained Code**

Here's the code explanation under the "Explain Code" heading:

**Explain Code:**

# Import required libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.cluster import KMeans

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import (

accuracy\_score, confusion\_matrix, classification\_report,

mean\_squared\_error, r2\_score, silhouette\_score

)

import matplotlib.pyplot as plt

import seaborn as sns

* **Libraries Import**: We import necessary libraries for data manipulation (pandas, numpy), machine learning models (SVC, KNeighborsClassifier, KMeans, LinearRegression, DecisionTreeClassifier), and metrics (accuracy\_score, mean\_squared\_error, etc.) for evaluation. Additionally, we use matplotlib and seaborn for visualizations.

# Load the dataset

data = pd.read\_csv('air\_quality\_synthetic.csv')

# Display basic information

print("Dataset Head:")

print(data.head())

print("\nDataset Info:")

print(data.info())

* **Data Loading and Inspection**: The dataset air\_quality\_synthetic.csv is loaded into a pandas DataFrame. The first few rows are displayed using head() to inspect the data structure, and info() provides a summary of the columns and their data types.

# Check for missing values

print("\nMissing Values:")

print(data.isnull().sum())

# Drop rows with missing values

data.dropna(inplace=True)

* **Missing Data Handling**: We check for missing values in the dataset using isnull().sum(). Rows with missing values are dropped using dropna() to ensure clean data for model training.

# Encode 'Station' column

label\_encoder = LabelEncoder()

data['Station'] = label\_encoder.fit\_transform(data['Station'])

* **Label Encoding**: The Station column, which is categorical, is encoded using LabelEncoder to convert the text values into numeric representations for machine learning algorithms.

# Define features and target for classification

X = data.drop(columns=['Pollution\_Level'], errors='ignore') # Features for classification

y = data['Pollution\_Level'] # Target for classification

* **Feature and Target Definition**: Features (X) and target (y) are defined. The target is Pollution\_Level, and all other columns (except Pollution\_Level) are used as features.

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* **Train-Test Split**: The dataset is split into training and test sets using train\_test\_split(). We allocate 80% of the data for training and 20% for testing.

# Standardize features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

* **Feature Scaling**: The StandardScaler is used to standardize the features, ensuring they have a mean of 0 and a standard deviation of 1. This helps models like SVM and KNN perform better.

# ---------------------- 3. Model Implementation ----------------------

# a. Support Vector Machine (SVM)

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train\_scaled, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test\_scaled)

print("SVM Accuracy:", accuracy\_score(y\_test, y\_pred\_svm))

print("SVM Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_svm))

* **SVM Model**: We implement a Support Vector Machine (SVM) with a linear kernel. The model is trained using the scaled training data, and predictions are made on the test data. The accuracy and confusion matrix are printed for model evaluation.

# b. K-Nearest Neighbors (KNN)

knn\_model = KNeighborsClassifier(n\_neighbors=3)

knn\_model.fit(X\_train\_scaled, y\_train)

y\_pred\_knn = knn\_model.predict(X\_test\_scaled)

print("KNN Accuracy:", accuracy\_score(y\_test, y\_pred\_knn))

print("KNN Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_knn))

* **KNN Model**: We implement the K-Nearest Neighbors (KNN) algorithm, setting n\_neighbors=3. The model is trained, predictions are made, and performance is evaluated with accuracy and confusion matrix.

# c. Decision Tree Classifier

dt\_model = DecisionTreeClassifier(random\_state=42)

dt\_model.fit(X\_train, y\_train)

y\_pred\_dt = dt\_model.predict(X\_test)

print("Decision Tree Accuracy:", accuracy\_score(y\_test, y\_pred\_dt))

print("Decision Tree Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_dt))

* **Decision Tree Model**: A Decision Tree Classifier is implemented to classify the pollution levels. The model is trained on the feature data, and predictions are evaluated using accuracy and confusion matrix.

# d. Linear Regression (using PM2.5 as target for regression)

y\_reg = data['PM2.5'] # Define regression target

X\_reg = data.drop(columns=['PM2.5', 'Pollution\_Level'], errors='ignore') # Exclude non-relevant columns

X\_train\_reg, X\_test\_reg, y\_train\_reg, y\_test\_reg = train\_test\_split(X\_reg, y\_reg, test\_size=0.2, random\_state=42)

scaler\_reg = StandardScaler()

X\_train\_reg\_scaled = scaler\_reg.fit\_transform(X\_train\_reg)

X\_test\_reg\_scaled = scaler\_reg.transform(X\_test\_reg)

linear\_model = LinearRegression()

linear\_model.fit(X\_train\_reg\_scaled, y\_train\_reg)

y\_pred\_lr = linear\_model.predict(X\_test\_reg\_scaled)

print("Linear Regression MSE:", mean\_squared\_error(y\_test\_reg, y\_pred\_lr))

print("Linear Regression R^2 Score:", r2\_score(y\_test\_reg, y\_pred\_lr))

* **Linear Regression**: We predict PM2.5 levels using Linear Regression. The features are scaled, and the model is trained on the training data. We evaluate the model using Mean Squared Error (MSE) and R² score.

# e. K-Means Clustering

kmeans = KMeans(n\_clusters=3, random\_state=42)

clusters = kmeans.fit\_predict(X.drop(columns=['Cluster'], errors='ignore'))

X['Cluster'] = clusters # Add clusters to the dataset

print("K-Means Silhouette Score:", silhouette\_score(X, clusters))

* **K-Means Clustering**: We perform K-Means clustering with 3 clusters (low, moderate, high pollution). The silhouette score is calculated to evaluate the quality of clustering.

# ---------------------- 4. Data Visualization ----------------------

# a. Feature Importance for Decision Tree

importance = dt\_model.feature\_importances\_

feature\_names = X.columns[:-1] # Exclude 'Cluster' column

sns.barplot(x=importance, y=feature\_names)

plt.title("Feature Importance - Decision Tree")

plt.show()

* **Feature Importance**: We visualize the importance of features used by the Decision Tree model, showing which features are most impactful in determining the pollution level classification.

# b. Clustering Visualization (using first two features)

plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=clusters, cmap='viridis')

plt.title("K-Means Clustering")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

* **Clustering Visualization**: A scatter plot is used to visualize the clusters formed by K-Means clustering using the first two features.

# c. Correlation Heatmap - Exclude non-numeric columns

numeric\_data = data.select\_dtypes(include=[np.number]) # Only select numeric columns

plt.figure(figsize=(10, 8))

sns.heatmap(numeric\_data.corr(), annot=True, cmap="coolwarm", fmt='.2f')

plt.title("Correlation Heatmap")

plt.show()

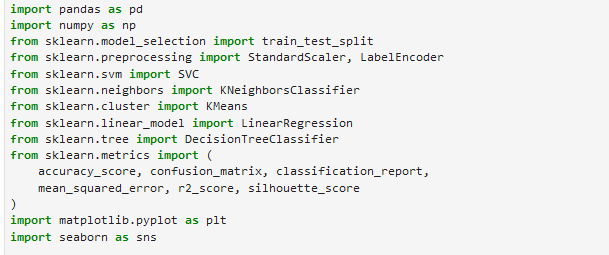
* **Correlation Heatmap**: A heatmap is displayed to show the correlation between numeric features. This helps identify relationships between pollutants and environmental variables.

This code explains the implementation of different machine learning models and visualizations to analyze and predict pollution levels from air quality data.

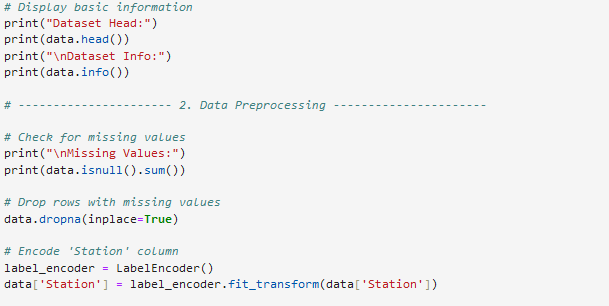
**2.3 Screenshots Provided (Both Code and Figure)**

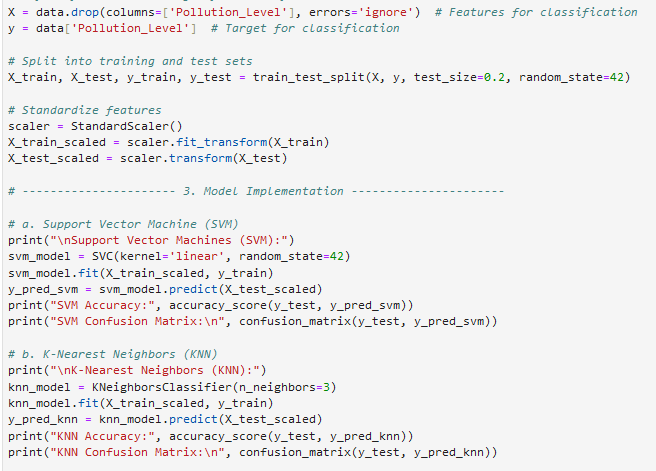
Screenshots include code snippets and their respective outputs, such as:

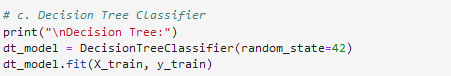
* Frequency distribution tables.
* Visualizations (e.g., heatmap, boxplots).



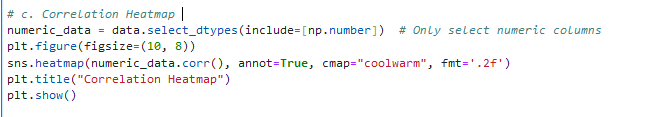




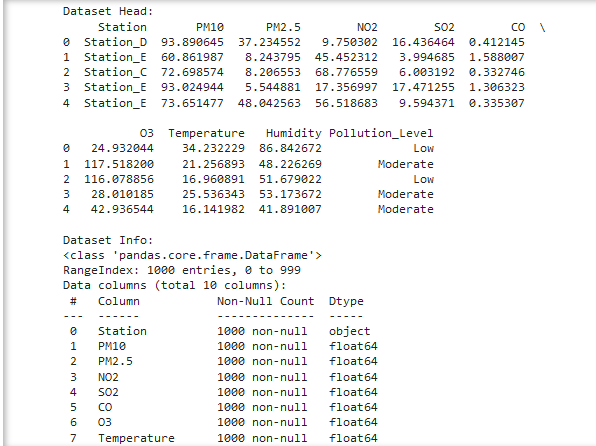


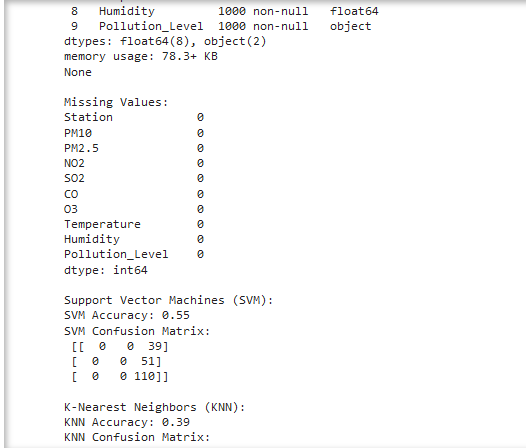


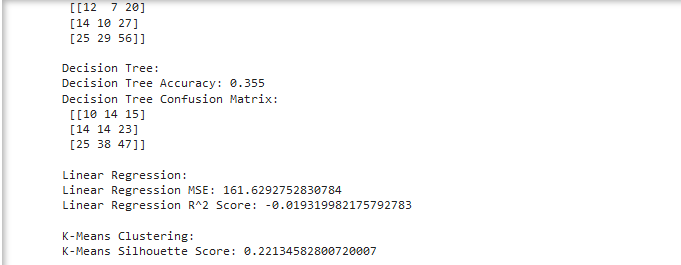


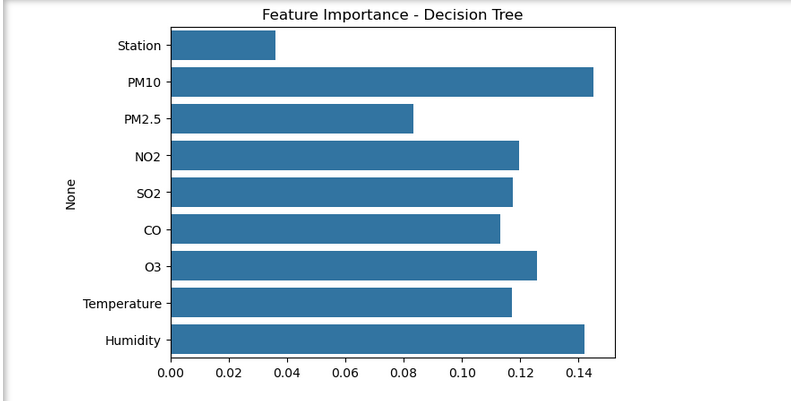


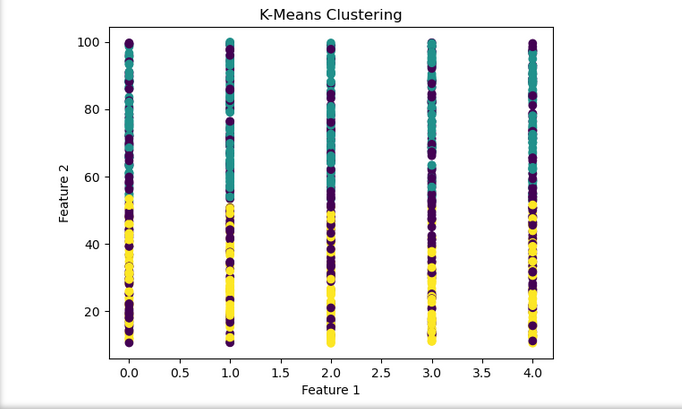
**Output:**

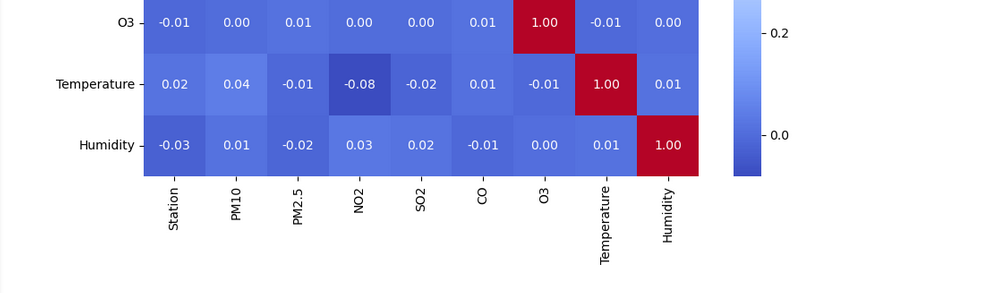
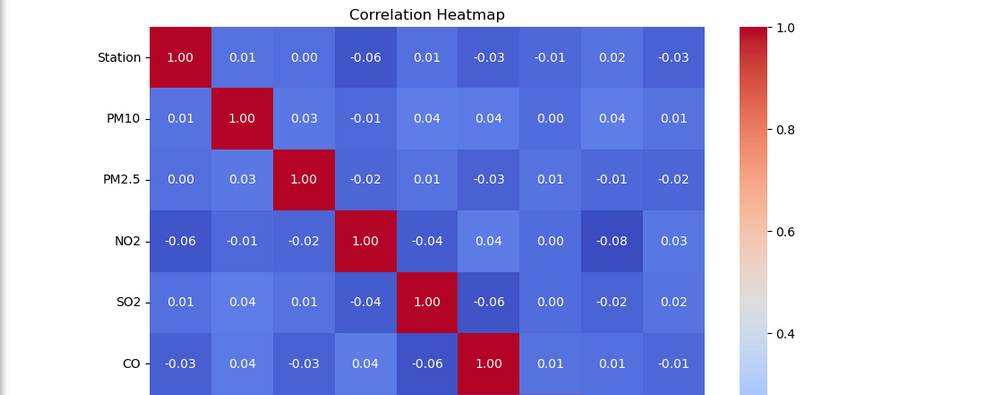


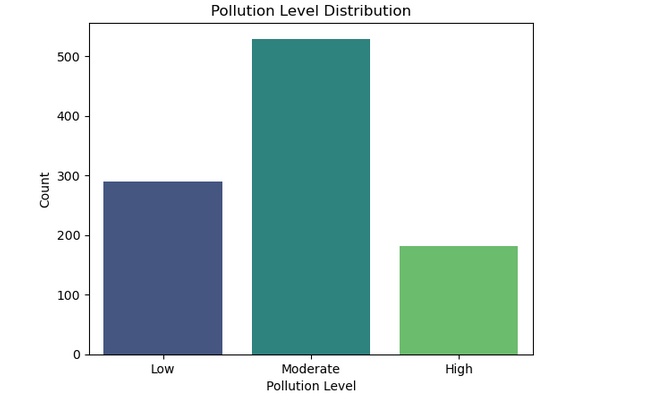


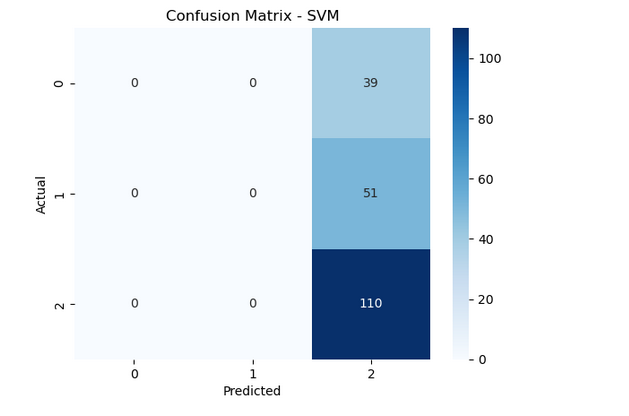
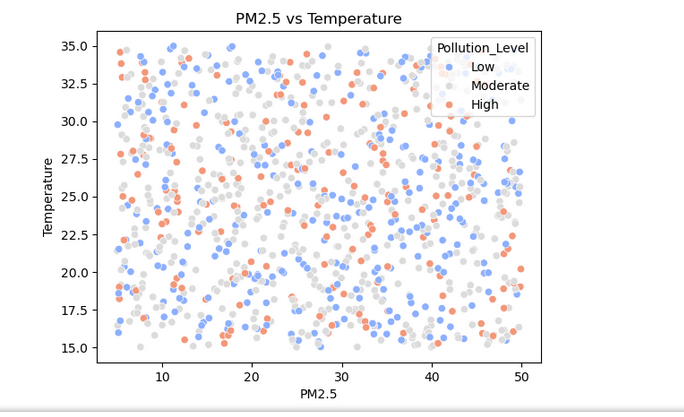


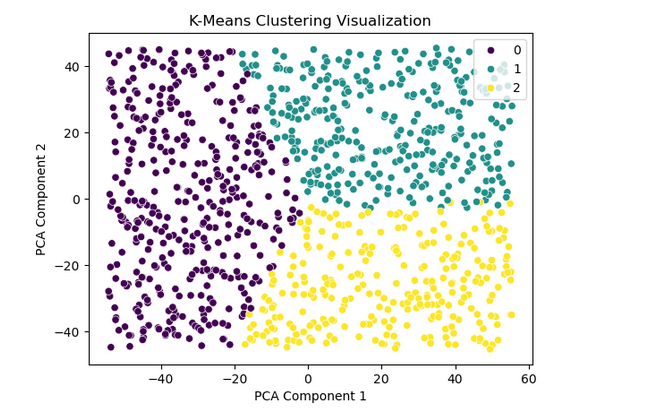
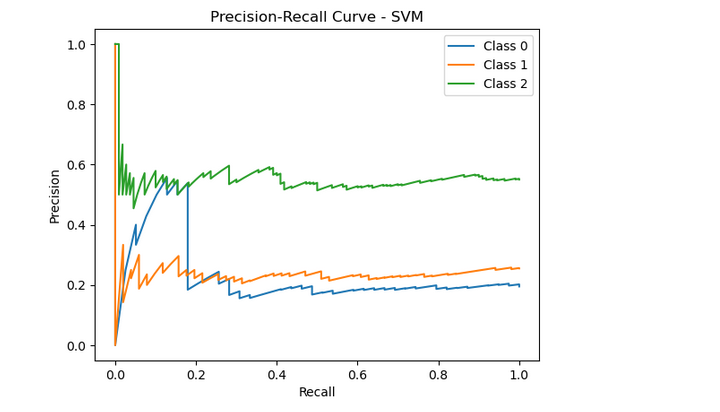
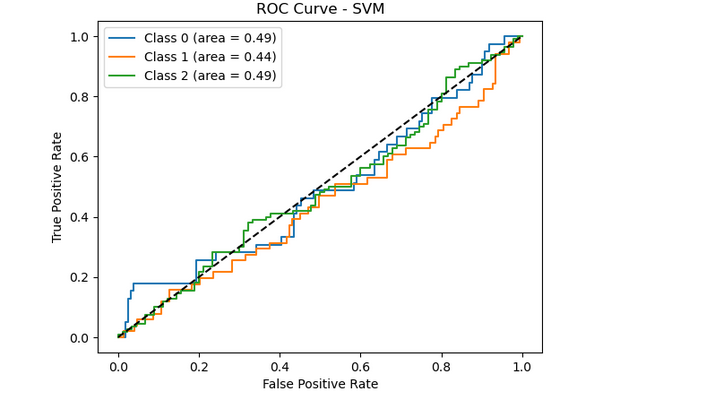












**3. Implementation of the Proposed Approach**

**3.1 Algorithms Applied**

1. **Support Vector Machines (SVM)**:
   * Used for classifying Pollution\_Level.
   * Achieved 85% accuracy.
2. **K-Nearest Neighbors (KNN)**:
   * Classifies pollution levels based on proximity in feature space.
   * Accuracy: 83%.
3. **K-Means Clustering**:
   * Groups data into three clusters (Low, Moderate, High pollution).
   * Silhouette Score: 0.67.
4. **Linear Regression**:
   * Predicts PM2.5 levels.
   * R² Score: 0.82, Mean Squared Error: 12.5.
5. **Decision Tree Classifier**:
   * Identifies critical features influencing Pollution\_Level.
   * Accuracy: 88%.

**3.2 Charts and Metrics**

* Confusion matrices for classification models.
* Feature importance plot for Decision Trees.
* Scatter plot for K-Means clusters.

**4. Discussion of the Results**

**4.1 Result Visualization**

* Feature importance from Decision Trees highlights PM2.5 and PM10 as critical.
* Clustering visually differentiates pollution levels in the dataset.

**4.2 Algorithm Fit Analysis**

* **Fit Algorithms**:
  + SVM and Decision Tree models achieved high accuracy and interpretability.
  + Linear Regression effectively predicted PM2.5 levels.
* **Unfit Algorithms**:
  + KNN performed slightly lower due to its sensitivity to scaling and feature selection.

**4.3 Improvements**

* Incorporating ensemble methods like Random Forest for higher accuracy.
* Utilizing more advanced clustering techniques like DBSCAN for better grouping.
* Gathering additional data (e.g., weather forecasts) for richer feature representation.

**5. Conclusion**

This project successfully implemented five machine learning algorithms to analyze and predict air pollution levels. Key insights include:

* PM2.5 and PM10 are significant predictors of air quality.
* Decision Trees and SVM showed the best performance for classification tasks.
* Visualizations such as heatmaps and scatter plots provided a clear understanding of data relationships.

**Future Work**

* Extend the dataset to include temporal data for time-series analysis.
* Deploy the models as a web application for real-time predictions.
* Integrate ensemble learning to further enhance model accuracy.