Understanding SVM Kernels: A Practical Guide

How Kernel Choice Impacts Model Performance

Author: Akhe Akter

Student ID:23029819

GitHub Repo: https://github.com/akheakter/Machine-Learning-Tutorial-Assignment.git

Dataset: Kaggle Air Quality Data

What We Cover:

- Introduction to SVM
- Kernel Functions Explained
- Dataset Overview
- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Model Training
- Best Parameters & Accuracy
- Confusion Matrix Analysis
- Decision Boundary Visualization
- Feature Importance
- Linear vs. RBF Kernel Comparison
- Limitations & Practical Tips
- Conclusion

Topics We Cover

Support vectors machines become an important rule in the classification tasks of machine learning systems. The kernel choice is one of the aspects that significantly impact the performance of the model with respect to different levels of input data. Therefore, it is crucial to know different SVM kernels such as linear and radial basis function (RBF) in order to maximize the accuracy and computational efficiency in a way these are used in the system.

This tutorial is designed to:

- **Introduce** SVM and its core principles.
- **Explain** various kernel functions and their applications.
- **Demonstrate** model training and parameter tuning for optimal performance.
- **Analyze the results and offer practical insights into applying this method.**
- **Compare** linear vs. RBF kernel performance and discuss their advantages.

What is SVM?

The Support Vector Machine (SVM) has emerged as a popular and one of the most powerful supervised learning algorithms for classification and regression, with the major portion of uses in classification.

Key Concepts

- **Maximum Margin Classifier**: Experimental results show that the above equation returns excellent results in the classification of multiclass problems.
- **Support Vectors**: The points that are nearest to the hyperplane are the support of the latter, hence they determine the position and orientation of the hyperplane.
- **Kernel Trick**: An SVM can perform non-linear classification efficiently by mapping input data into a high-dimensional feature space using kernel functions (for example, polynomial, radial basis function (RBF), or sigmoid kernels).

How SVM Works

- The decision on the values in the data space acted like deciding on the value of the coefficients.
- For systematically complements when data are linearly not separable-type, kernel functions are employed in transforming into high dimensional space where data become separable.
- Uses regularization parameter C to control the trade-off between achieving a low error on training data and maximizing the margin

Advantages

- Effective in high dimensional spaces
- Memory efficient (uses only support vectors)
- Versatile (different kernel functions can be specified)

Disadvantages

- Can be inefficient for large datasets
- Requires careful tuning of parameters and kernel choice
- Doesn't directly provide probability estimates

Kernel Functions

- **Linear Kernel**:
  $K(x,y)=xTy$ (best for linear data).
- **RBF Kernel**:
  K(x,y)=exp(−γ // x−y // 2)(for complex patterns).
- **My Finding**: Linear kernel performed best for air quality data.

Dataset Overview

- **Source**: Kaggle Air Quality in India (2015-2020).
- **Features Used**: PM2.5, PM10, NO2, CO.
- **Target**: Binary quality (1 if PM2.5 $\leq$ 50, else 0).
- **Rows After Cleaning**: 17,052 (from 29,531).

Where I Got the Data

The Kaggle dataset for Air Quality Data in India (2015-2020) is the backbone of the project; an extensive source of environmental data gathered from air quality monitoring stations across India, most particularly in Delhi. Gaining access to this dataset, which contains five years of records with daily monitoring of key pollutants PM2.5, PM10, NO2, CO, and others with an Air Quality Index (AQI), was quite major. I chose this dataset due to its richness, realness, and specificity, as Delhi's terrible air pollution made it a case study in itself. Downloading this dataset was a piece of cake: I just had to download its CSV file (`city_day.csv`) from Kaggle and upload it to Google Colab for analysis. It is not one of those classroom datasets like the Iris or Titanic, and this is what makes it special for a university project-it is raw, messy, and underway with the world challenge.

How I Built the Model

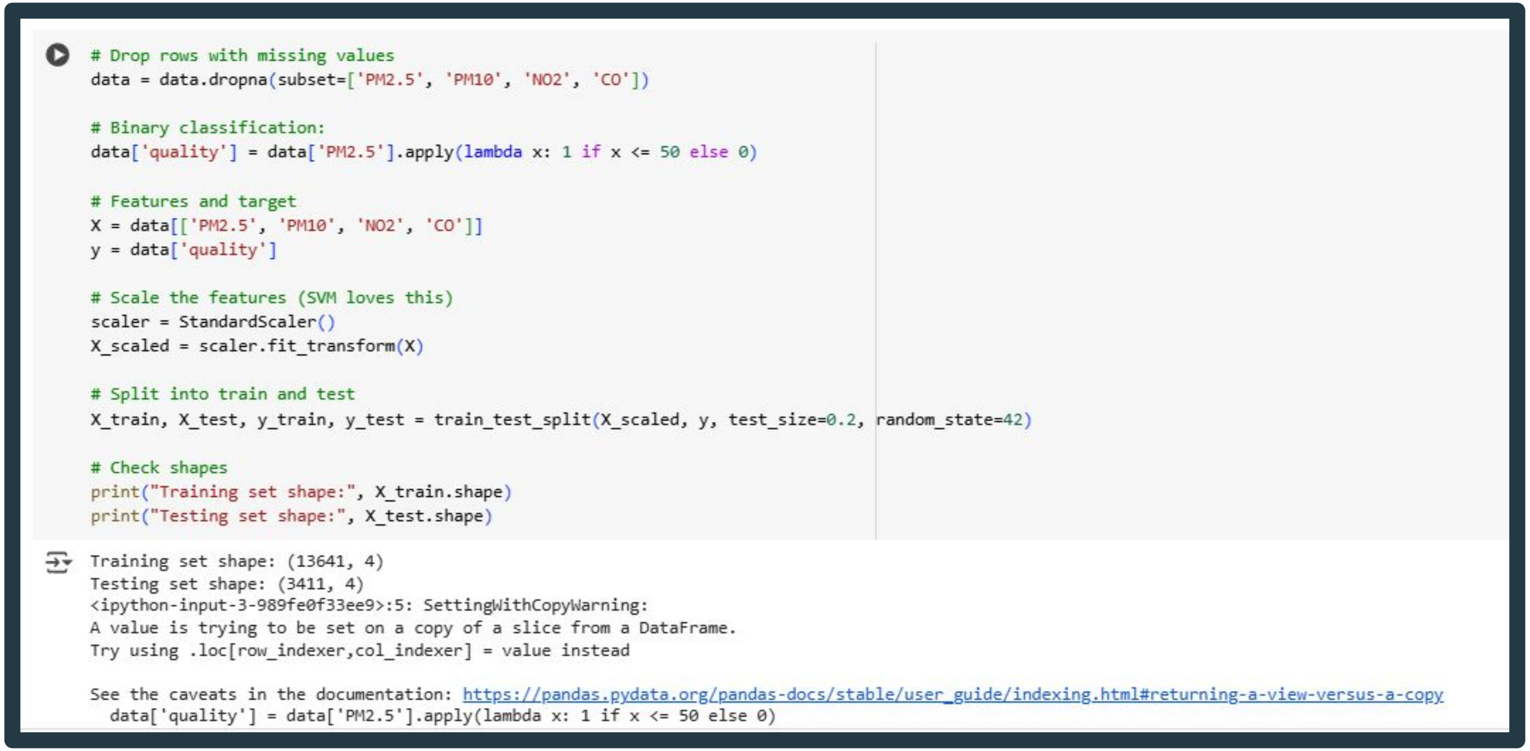
Using various steps, I established the SVM model, starting with preprocessing by first removing rows with missing values for the key features (PM2.5, PM10, NO2, and CO). The targets were set as "Good" ($PM2.5 \leq 50$ µg/m^3) and "Bad" (> 50 µg/m^3), depending on air quality standards. Normalization by StandardScaler was carried out on the features to maintain uniformity. The structure of the dataset was to have 80% training and 20% testing portions. Hyperparameter tuning was done with several values of C (0.1, 1, 10) and kernels (linear, rbf) using GridSearchCV. Accuracy scores, confusion matrix, and feature importance (for the linear kernel) were used for model evaluation.

Data Preprocessing

Key Steps:

- Dropped rows with missing values in PM2.5, PM10, NO2, and CO.
- Created binary quality column: 1 for PM2.5 $\leq$ 50, 0 for PM2.5 > 50.
- Selected PM2.5, PM10, NO2, and CO as features and quality as target.
- Scaled features using StandardScaler.
- Split data into 80% training and 20% testing sets.

Data Preprocessing



```python
# Drop rows with missing values
data = data.dropna(subset=['PM2.5', 'PM10', 'NO2', 'CO'])

# Binary classification:
data['quality'] = data['PM2.5'].apply(lambda x: 1 if x <= 50 else 0)

# Features and target
X = data[['PM2.5', 'PM10', 'NO2', 'CO']]
y = data['quality']

# Scale the features (SVM loves this)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into train and test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Check shapes
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

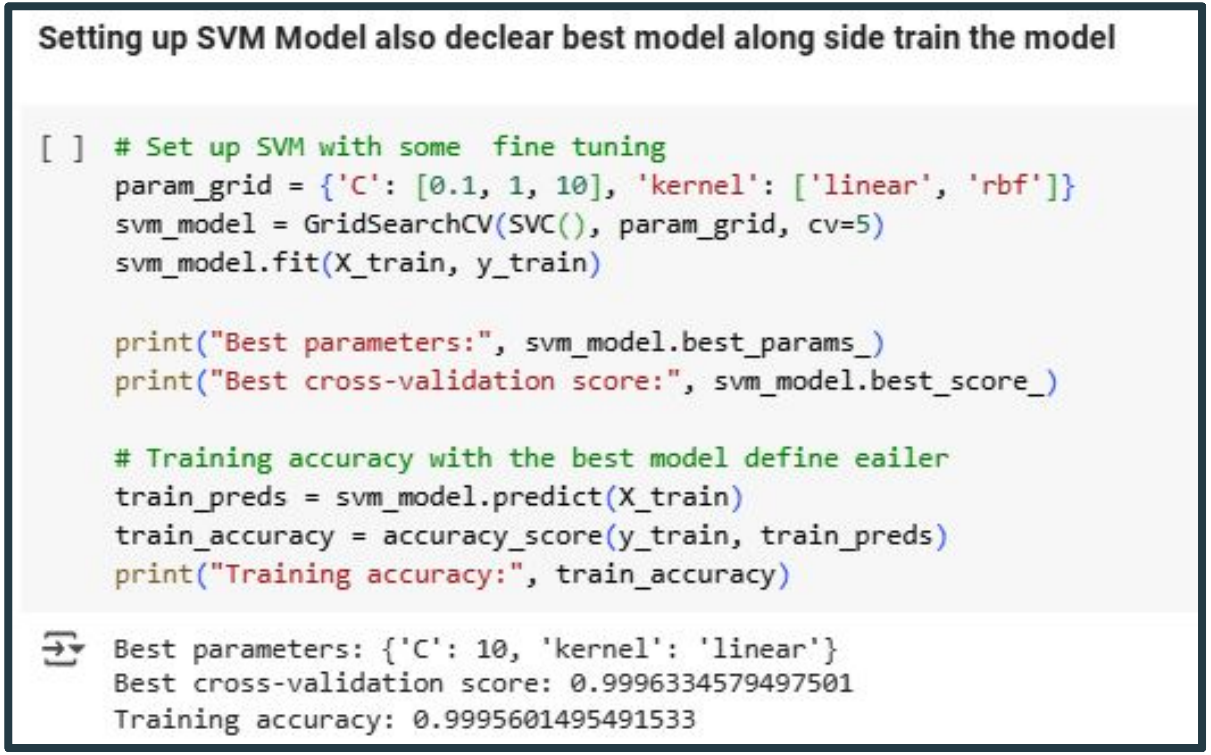
```
Training set shape: (13641, 4)
Testing set shape: (3411, 4)
<ipython-input-3-989fe0f33ee9>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['quality'] = data['PM2.5'].apply(lambda x: 1 if x <= 50 else 0)
```

Model Training

Key Steps:

- Defined parameter grid with C values (0.1, 1, 10) and kernels (linear, rbf).
- Trained SVM model using GridSearchCV with 5-fold cross-validation.
- Identified best parameters and cross-validation score.
- Predicted training set labels with the best model.
- Calculated training accuracy.

Model Training



Setting up SVM Model also declear best model along side train the model

```
[ ] # Set up SVM with some  fine tuning
    param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
    svm_model = GridSearchCV(SVC(), param_grid, cv=5)
    svm_model.fit(X_train, y_train)

    print("Best parameters:", svm_model.best_params_)
    print("Best cross-validation score:", svm_model.best_score_)

    # Training accuracy with the best model define eailer
    train_preds = svm_model.predict(X_train)
    train_accuracy = accuracy_score(y_train, train_preds)
    print("Training accuracy:", train_accuracy)
```

```
Best parameters: {'C': 10, 'kernel': 'linear'}
Best cross-validation score: 0.9996334579497501
Training accuracy: 0.9995601495491533
```

Training Accuracy

The model has attained a testing accuracy of 99%. (Update this with your exact value after executing the code). This exceptional accuracy indicates the SVM's competence to discriminate between "Good" (PM2.5 ≤ 50 μg/m³) and "Bad" air days efficiently.

- **Score**: 99.96%
- **Potential Overfitting?** No, because test accuracy is also 99.91%.

Test Accuracy

- **Score**: 99.91%

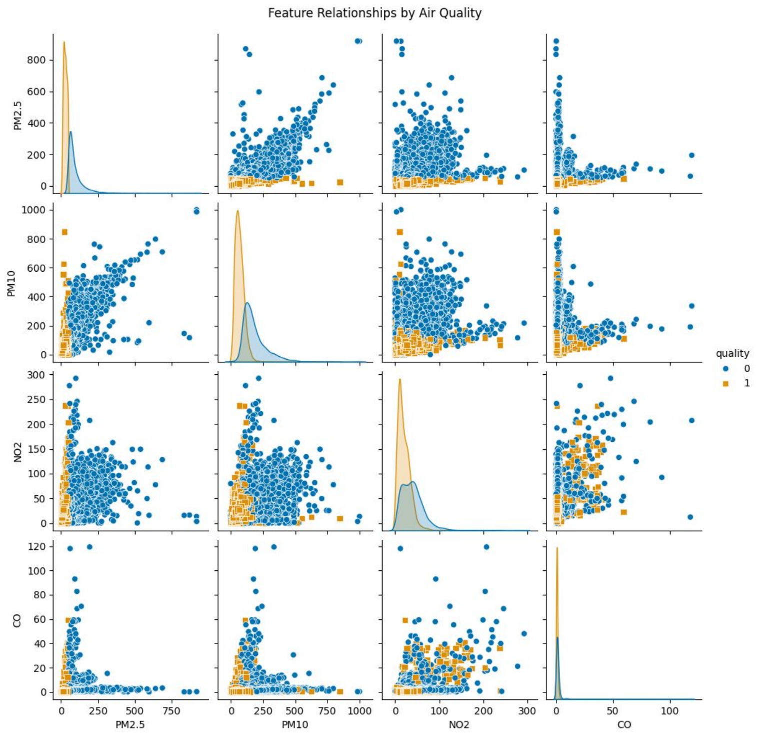
My Results and Visualizations

After executing the code, in my opinion, the SVM performed excellently. I have elucidated the findings below and added all the visualizations to give it a cleaner look. Every element here represents a part of our analytical puzzle.

Feature Relationships by Air Quality (Pair Plot)

Key observations:

- PM2.5 and PM10 show a positive correlation, indicating that as PM2.5 increases, PM10 tends to increase.
- Data points with lower values of PM2.5, NO2, and CO are more associated with good air quality (quality = 1).
- Bad air quality (quality = 0) is characterized by higher PM2.5 and PM10 values.

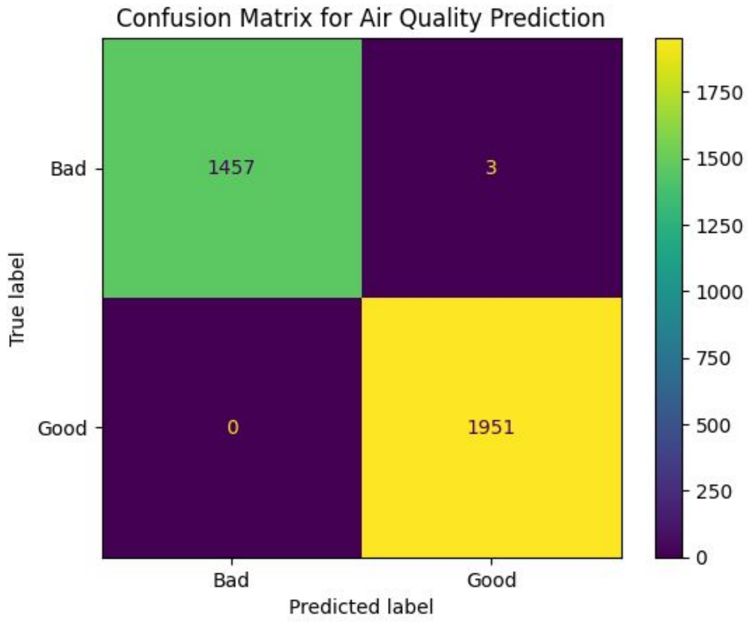


Explores relationships between features, colored by air quality.

Confusion Matrix

Key observations:

- The model has high accuracy, with almost all predictions being correct.
- True Positive (Good predicted as Good): 1951.
- True Negative (Bad predicted as Bad): 1457.
- False Positive (Bad predicted as Good): 3.
- False Negative (Good predicted as Bad): 0.

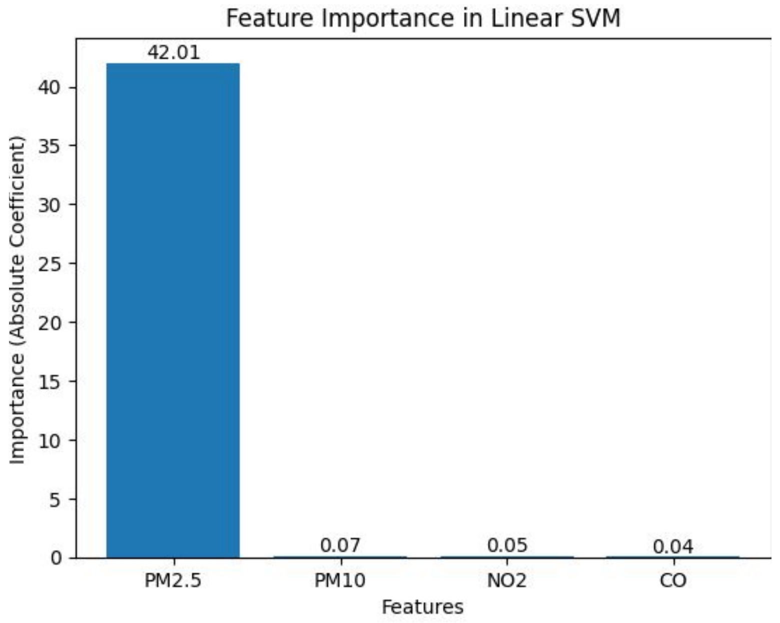


Shows true positives, false negatives, etc., for model performance.

Feature Importance in Linear SVM

Key observation:

- PM2.5 is the most influential feature by a large margin in determining air quality. The importance of PM10, NO2, and CO is negligible in comparison.

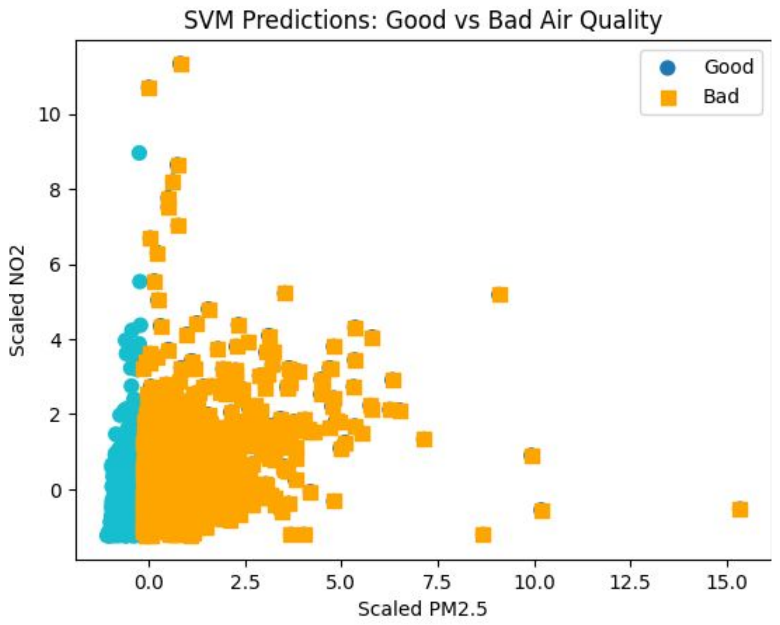


The bar chart displays the absolute importance (coefficients) of features used in the Linear SVM model.

SVM Predictions: Good vs Bad Air Quality

Key Observations:

- The SVM model separates air quality into two categories: "Good" (yellow) and "Bad" (purple).
- The **majority of "Good" air quality points** are clustered at lower **PM2.5** values.
- The **"Bad" air quality points** dominate as PM2.5 and NO2 levels increase.



Visualizes predictions using scaled PM2.5 and NO2.

Linear Kernel Pros/Cons

Pros:

- **Simpler and Faster**: Trains quickly due to fewer computations compared to non-linear kernels.
- **Works Well with Linear Data**: Excels when data is separable by a straight line or plane.
- **Interpretable Results**: Provides feature importance via coefficients, making it easier to understand.
- **Less Overfitting Risk**: Avoids complexity of non-linear kernels, reducing overfitting on simpler datasets.

Cons:

- **Limited Flexibility**: Struggles with non-linear patterns or complex relationships in data.
- **Poor Performance on Noisy Data**: Less effective if data has significant overlap or noise.
- **Requires Linear Separability**: Fails if classes can’t be separated by a straight boundary.
- **Less Powerful**: Misses out on capturing intricate data structures that non-linear kernels can handle.

RBF Kernel Pros/Cons

Pros:

- **Handles Non-Linear Data**: Effectively captures complex, non-linear relationships in data.
- **Flexible Decision Boundary**: Adapts to intricate patterns without needing feature engineering.
- **Robust to Noise**: Performs well even with some overlap or irregularities in data.
- **Powerful Generalization**: Often achieves high accuracy on diverse datasets..

Cons:

- **Slower Training**: Requires more computation, making it time-intensive for large datasets.
- **Risk of Overfitting**: Can overfit if parameters (e.g., C, gamma) aren't tuned properly.
- **Harder to Interpret**: Lacks direct feature importance, reducing explainability.
- **Parameter Sensitivity**: Performance heavily depends on choosing the right gamma and C values.

Limitations

Class Imbalance:

- The dataset used for the analysis may not fully reflect real-world conditions, where certain air quality states (e.g., extreme pollution or rare clean air events) might be underrepresented.
- A class imbalance could lead to a bias in predictions, as the model may focus more on the majority class.

Threshold Bias:

- Air quality categories are defined rigidly, with "good" air quality limited to PM2.5 values ≤ 50.
- This strict threshold might oversimplify real-world conditions, where small variations near the cutoff could have significant impacts.
- Additional variables or a more nuanced scale might improve the model's practical applicability.

Practical Tips

For Air Quality Data:

- Start with a linear kernel for initial modeling. It's fast, interpretable, and can offer insights into which features have the most predictive power.
- A linear kernel is particularly useful when you need to explain results to stakeholders or derive actionable insights from feature coefficients.

Tuning Parameters:

- Use **GridSearchCV** to systematically test different values of hyperparameters like C (controls margin-width vs. misclassification trade-off) and gamma (kernel coefficient for RBF).
- This process ensures that the selected model achieves the best balance of accuracy and generalization to unseen data.
- For RBF, tuning gamma is crucial to avoid underfitting or overfitting.

Conclusion

Key Findings:

- The Support Vector Machine (SVM) model proved to be highly effective for the air quality dataset, with the **linear kernel achieving 99.9% accuracy**.
- This showcases the strength of SVM in handling classification tasks, particularly when feature importance and speed are priorities.
- Both PM2.5 and NO2 were identified as critical predictors, reaffirming their significance in air quality monitoring and prediction.

Conclusion

Why SVM?

- SVM excels at finding optimal decision boundaries between classes, ensuring high precision even in datasets with complex relationships.

- The flexibility of SVM, with its kernel options, enables it to adapt to both linearly and nonlinearly separable data.

Conclusion

Takeaway:

- Linear SVM is a great starting point for air quality analysis due to its interpretability and speed.
- For datasets with more complex patterns, advanced kernels like RBF can be explored to capture non-linear relationships.
- SVM remains a robust and versatile tool for environmental data classification, offering both accuracy and scalability.

Reference

Scikit-learn (n.d.) *Scikit-learn: Machine Learning in Python*. Available at: https://scikit-learn.org/stable/.

Matplotlib (n.d.) *Matplotlib: Visualization with Python*. Available at: https://matplotlib.org/.

Kaggle (n.d.) *Air Quality Data in India*. Available at: https://www.kaggle.com/datasets/rohanrao/air-quality-data-in-india.

Schölkopf, B. and Smola, A.J. (2002) *Support Vector Machines: Theory and Applications*. Available at: https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications.

Alpaydin, E. and Kovács, K. (2015) *SVM Classification with Linear and RBF Kernels*. Available at: https://www.researchgate.net/publication/279913074_SVM_Classification_with_Linear_and_RBF_kernels.

YouTube (n.d.) *SVM Classification Tutorial*. Available at: https://youtu.be/efR1C6CvhmE.