

PROJECT REPORT

GROUP MEMBERS:

UMER ABBAS(SP23-BAI-053)

MUHMMMAD ZAIMAL ZIA(SP23-BAI-043)

SUBMITTED TO: DR. AKSAM IFTIKHAR

SUBMISSION DATE: 12-12-24

DEPARTMENT OF COMPUTER SCIENCE

COMSATS UNIVERSITY ISLAMABAD, LAHORE CAMPUS

**1. Introduction**

* Problem Description: This project applies machine learning to classify air quality levels based on environmental parameters.
* Dataset Overview:  
  The dataset, sourced from Kaggle, contains 5000 samples and 9 features total 10 columns with one representing target values. Features include various pollutant levels and meteorological factors. The target variable, "Air Quality," indicates air quality categories. The air quality is classified into four classes (Good, Poor, Moderate, Hazardous). Two features have some missing values. All the features are numerical, only the target column was categorical.

**2. Methodology**

* Preprocessing:
* Encoded categorical target variable using Label Encoder.
* Handled missing values by replacing them with mean of column.
* Split data into training (80%) and testing (20%) sets.
* Balanced classes using SMOTE to address class imbalance.
* Standardized features using StandardScaler.
* Reduced dimensionality with PCA (95% explained variance).
* Algorithms Applied:
  + Random Forest: Ensemble learning model tuned for n\_estimators, depth, min\_max\_split, max\_features.
  + Support Vector Machine (SVM): Tuned for kernel type, regularization (C), and gamma.
  + XGBoost: Gradient-boosting model with hyperparameter tuning for depth, learning rate, and subsampling.
  + K-Nearest Neighbors (KNN): Optimized for number of neighbors, distance metric, and weight function.
* Optimization Techniques:
  + Used Grid Search and Randomized Search CV for hyperparameter tuning.
  + Evaluated models using cross-validation to ensure robust performance.

**3. Results**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | Accuracy | Precision | Recall | F1 score | Execution time in seconds |
| Random Forest | 0.88 | 0.99 | 0.91 | 0.95 | 3.4s |
| SVM | 0.83 | 1 | 0.76 | 0.86 | 0.4s |
| XGBoost | 0.88 | 0.99 | 0.91 | 0.95 | 1.2s |
| Knn | 0.88 | 0.99 | 0.94 | 0.96 | 0.1s |

Best Hyperparameters:

1) Random Forest:

n estimatots:200, max depth: none, min max split:2, max features: sqrt

2) SVM:

C:10, gama:auto, kernel:rbf

3) XGBoost:

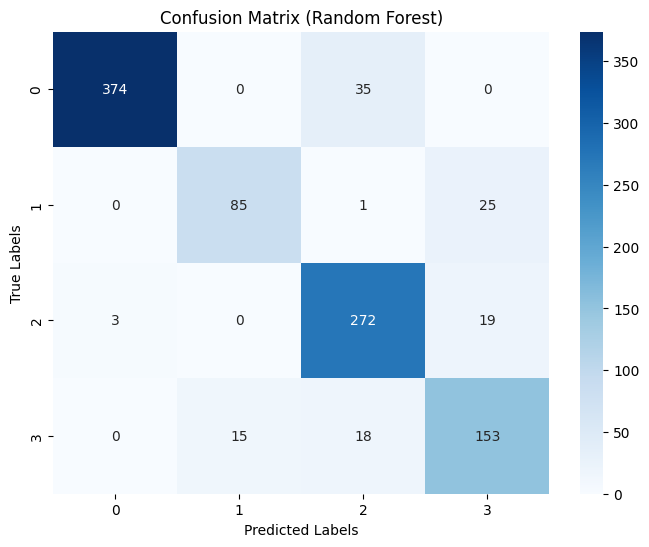
Learning rate :0.2,max depth:9,n estimators: 200, subsample:0.8

4) Knn:

Metric : manthan ,n neighbours: 3 , weights : distance

Confusion Metrices:

Random Forest:



SVM:

A diagram of a graph

Description automatically generated with medium confidence

XGBoost:

A graph of blue squares with numbers and labels

Description automatically generated

Knn:

A graph of a confusion matrix

Description automatically generated with medium confidence

**4. Analysis:**

1. XGBoost:
   * Maintains its position as the best overall performer.
   * With a reasonable execution time of 1.2 seconds, it balances accuracy and efficiency effectively.
2. KNN:
   * Fastest algorithm with an execution time of just 0.1 seconds.
   * Exceptionally high recall makes it suitable for scenarios where missing positive cases (e.g., poor air quality) is critical.
   * Its lightweight nature makes it well-suited for real-time applications.
3. Random Forest:
   * Despite strong performance metrics, it is the slowest algorithm (3.4 seconds), due to the ensemble's computational complexity.
   * Still a solid choice when speed is not a priority and interpretability of feature importance is valuable.
4. SVM:
   * Execution time of 0.4 seconds is relatively fast but still lags in accuracy, precision, and F1-Score compared to XGBoost and KNN.
   * Best used for moderately sized datasets where computation isn't a issue.

Final Insights:

* For high performance we prefer XGBoost
* For speed we prefer KNN
* For balanced needs and feature importance analysis we prefer random forest
* For scenarios requiring moderate speed and performance we prefer SVM

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_