Air Quality Classification Using Perceptron Models and Advanced Neural Techniques

Introduction

Air pollution is a major environmental issue affecting public health and climate. This study focuses on classifying air quality based on different environmental parameters using a combination of perceptron-based models, ensemble learning, and clustering. The dataset consists of various air quality indicators, and the goal is to compare a hardcoded perceptron model, a traditional MLP classifier, and an MLP model implemented using Keras with Adam optimizer and Dropout. An ensemble method is used to combine predictions from both MLP models. Additionally, clustering is applied on all four target classes with dimensionality reduction via PCA for visualization. Finally, a Gradio interface is created for model deployment.

Preprocessing Techniques Used

Data Exploration

- Dataset loaded and analyzed using **pandas**.
- Initial inspection using head(), tail(), info(), and describe().

Handling Missing Values

- Missing values detected using isna().sum().

Outlier Detection and Removal

- **Z-score normalization** used to detect outliers.
- Records with Z-score > 3 removed.

Feature Scaling

- **Min-Max scaling** applied to normalize values in the range [0, 1].

Encoding Categorical Variables

- LabelEncoder used to convert air quality categories into numerical form.

Feature Selection Using Correlation Analysis

- Correlation matrix visualized using **seaborn**.
- Features with correlation > 0.5 selected.

Model Comparison and Enhancements

1. Multi-Layer Perceptron (MLP) Classifier (Scikit-learn)

The Multi-Layer Perceptron (MLP) classifier from Scikit-learn is a feedforward neural network that consists of an input layer, one or more hidden layers, and an output layer. In this project, it was configured with one hidden layer of 5 neurons and trained over 150 iterations using backpropagation. The model learns patterns by adjusting weights based on the error between predicted and actual values. It performs significantly better than the hardcoded perceptron, as it can learn non-linear relationships within the data.

- **Library:** sklearn.neural_network
- **Hidden Layers:** One (5 neurons)
- **Iterations:** 150
- **Training:** Automatic weight learning via backpropagation
- **Evaluation:** Train-test split (80–20), accuracy_score, and confusion matrix

2. Hardcoded Perceptron Model

The hardcoded perceptron is a simple, single-layer neural network where the weights and bias are manually set and do not change. It uses a step function as the activation function to classify data points based on whether the weighted sum of inputs exceeds a threshold. This model is fast and interpretable but lacks adaptability. It cannot learn from data or improve performance over time, which limits its accuracy, especially for complex classification problems like air quality.

- **Implementation:** Manual weights (e.g., [0.3, -0.4, 0.2, 0.5, -0.4]), bias -0.1
- **Activation:** Step function
- **Training:** None (weights fixed)
- **Evaluation:** Direct comparison of predictions vs. ground truth

3. MLP Using Keras with Adam Optimizer and Dropout (Tensorflow)

This model extends the capabilities of the MLP classifier by using the Keras deep learning library. It incorporates advanced features such as the **Adam optimizer**, which provides adaptive learning rates, and **Dropout**, a regularization technique that randomly disables neurons during training to prevent overfitting. This MLP model has more hidden layers and neurons than the Scikit-learn version, allowing it to model more complex relationships in the data. It also benefits from Keras's flexibility in designing and visualizing the architecture.

- **Library:** tensorflow.keras
- **Architecture:** Input layer → Hidden layers (ReLU + Dropout) → Output (Softmax)
- **Optimizer:** Adam
- **Regularization:** Dropout added to prevent overfitting
- **Evaluation:** Accuracy, loss convergence, and validation performance

4. Ensemble Model

The ensemble model combines the predictions of both the Scikit-learn MLP and the Keras-based MLP. By aggregating predictions through **majority voting** or averaging probabilities, the ensemble method enhances robustness and compensates for weaknesses in individual models. This approach typically leads to higher accuracy and stability in prediction results. It is especially useful when models have comparable but slightly different strengths.

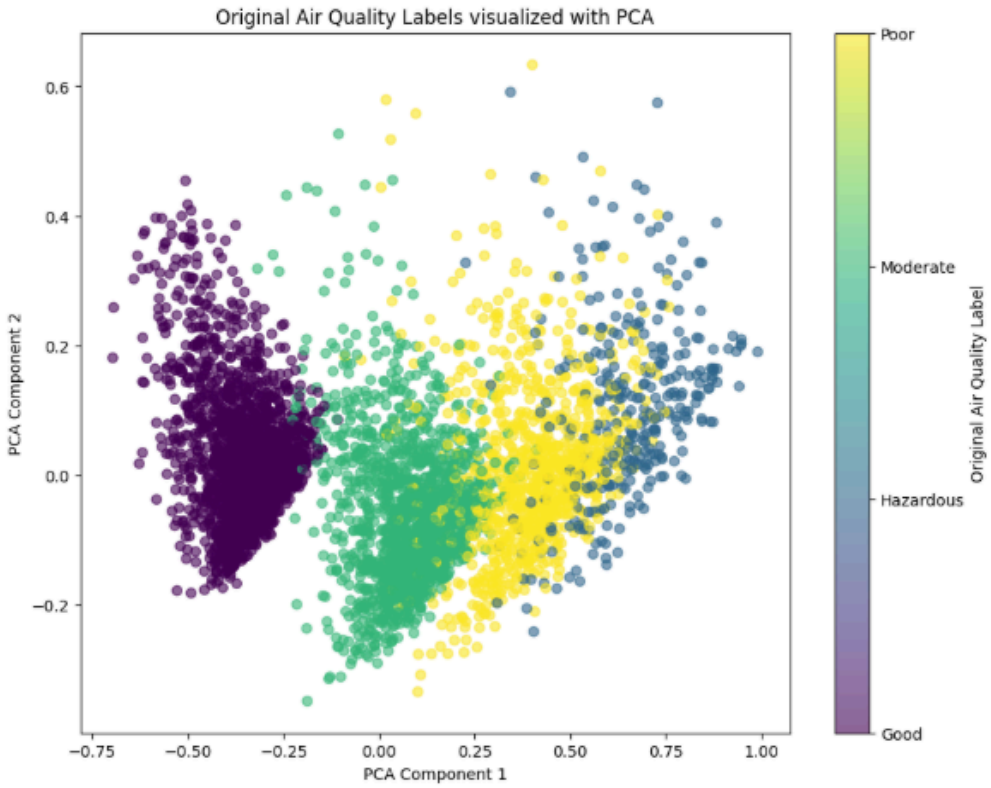
- Combined predictions from:
    - MLPClassifier (sklearn)

    - Keras-based MLP (tensorflow)

- Final output decided using **majority voting**
- Ensemble boosted classification robustness and stability

Clustering and Visualization

- **Clustering** applied on features to group data based on similarity for all four target classes (poor, hazardous, moderate, good).
- **Algorithm Used:** KMeans
- **Visualization:**
    - **PCA (Principal Component Analysis)** reduced dimensionality to 2D.
    - Scatter plots revealed distinct cluster separations.



Gradio Interface for Deployment

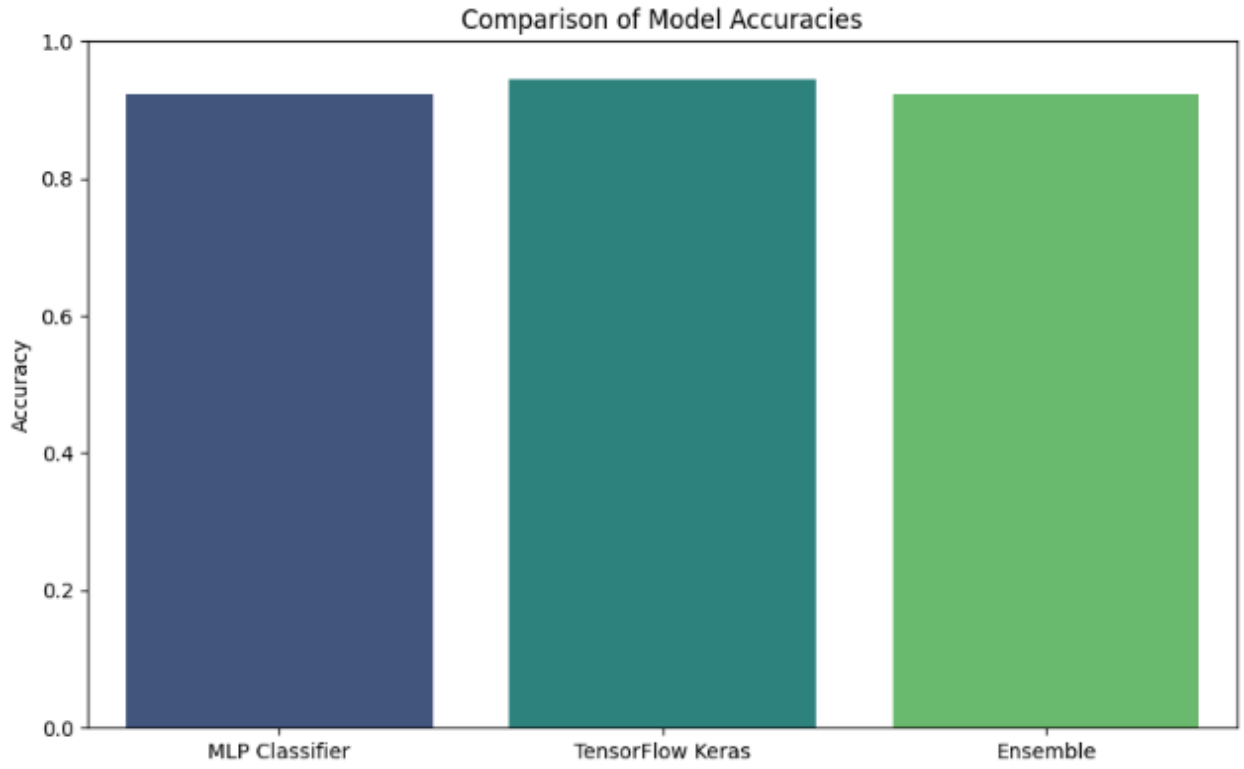
- Built a **Gradio-based interactive web app** to:
    - Take user input for environmental parameters
    - Allow model selection (Sklearn MLP, Keras MLP, Ensemble)
    - Output predicted air quality category
- User-friendly, responsive UI with clear labels and submit button

Results & Conclusion

📊 Model Comparison Table

Aspect	**Hardcoded Perceptron**	**MLP Classifier (Sklearn)**	**MLP (Keras + Adam + Dropout)**	**Ensemble Model**
Learning Method	Manual weights (no training)	Supervised learning (backpropagation)	Deep learning with optimizer	Combination of MLP Sklearn and Keras
Complexity	Very simple	Moderate	High	High
Layers	Single layer	One hidden layer	Multiple hidden layers	Combines outputs from two models
Activation Function	Step function	ReLU, Logistic	ReLU, Softmax	Depends on underlying models
Regularization	None	None	Dropout	Implicit via model diversity
Optimizer	N/A	SGD (default)	Adam	N/A (uses outputs from others)

Adaptability	Not adaptable	Moderately adaptable	Highly adaptable	Highly adaptable
Accuracy (Project Result)	0.48	0.92	**0.94**	0.92
Training Time	Instant	Fast	Moderate to High	Moderate
Use Case	Educational/demo purposes	Standard ML applications	Complex data and deep learning tasks	Enhanced performance via ensemble



Observations

- **Keras MLP** outperformed Scikit-learn MLP and Ensemble method due to deeper architecture and dropout.

- **Hardcoded perceptron** lacked the flexibility to model complex relationships.

- PCA-based clustering visualization helped uncover data structure and class separation.

Potential Improvements

- **Tuning:** Hyperparameter optimization (e.g., learning rate, dropout rate)
- **Additional Models:** Try tree-based models (Random Forest, XGBoost)
- **Temporal Analysis:** Explore time series trends in air quality
- **API Integration:** Extend Gradio interface to REST API for deployment

References

- **Python Libraries:** pandas, NumPy, matplotlib, seaborn, sklearn, keras, tensorflow, gradio
- **Algorithms:** MLP Classifier, Hardcoded Perceptron, Keras MLP, Ensemble Voting, KMeans, PCA