

**TITLE:** AirSense - AI-Powered Air Quality Monitoring & Forecasting

**Model Refinement Submission**

**Team Members**

1. Gelasa Jarso

2. Dawit Getachew

3. Helen Zelalem

4. Mukitar Seid

5. Tekleeyesus Munye

**Machine Learning Project Documentation**

**Model Refinement**

**1. Overview**

Model refinement is a pivotal phase in the machine learning (ML) pipeline. After initial training and evaluation, the model may demonstrate limitations such as overfitting, underfitting, low generalizability, or poor performance on key metrics. The goal of the refinement phase is to optimize the model's architecture, hyperparameters, features, and training strategies to improve its robustness, accuracy, and real-world applicability. This phase includes evaluating model performance using statistical techniques and refining it through an iterative process until an optimal configuration is achieved.

**2. Model Evaluation**

After initial model training, evaluation was conducted using training, validation, and test data partitions. The following key metrics were used to assess model quality:

**Accuracy:** Measures the ratio of correct predictions to total predictions.

**Precision:** Indicates the number of true positive predictions over all positive predictions made.

**Recall (Sensitivity):** Measures the ability of the model to identify all relevant cases.

**F1 Score:** Harmonic mean of precision and recall, useful for imbalanced datasets.

**ROC-AUC Score:** Measures the model’s capability of distinguishing between classes.

**Initial Results:**

| **Metric** | **Training** | **Validation** |
| --- | --- | --- |
| Accuracy | 95% | 84% |
| Precision | 93% | 82% |
| Recall | 90% | 80% |
| F1 Score | 91.5% | 81% |
| ROC-AUC | 0.96 | 0.87 |

These discrepancies between training and validation scores pointed to overfitting, necessitating improvements in regularization, feature selection, and model generalization.

**3. Refinement Techniques**

This may include adjusting The following model refinement techniques were systematically applied:

a. Hyperparameter Optimization

Employed Grid Search CV and Randomized Search CV to find optimal values.

* Tuned parameters like:
* Learning Rate
* Maximum Depth
* Min Samples Split / Leaf
* Number of Estimators
* Regularization Terms (L1, L2)
* Kernel Types (for SVM)
* Dropout Rate (for neural networks)

b. Ensemble Learning

* Bagging: Combined results from multiple decision trees using Random Forest.
* Boosting: Applied Gradient Boosting Machines (GBM), XGBoost, and LightGBM.
* Stacking: Trained base learners and combined their predictions using a meta-learner.

c. Regularization Techniques

Implemented Ridge (L2) and Lasso (L1) regularization.

Used ElasticNet when both L1 and L2 were beneficial.

Helped to reduce model complexity and mitigate overfitting.

d. Data Balancing

Applied SMOTE (Synthetic Minority Oversampling Technique) for balanced class distribution.

Tested undersampling methods to ensure all classes were equally represented.

e. Feature Engineering

Created polynomial features, interaction terms, and normalized inputs.

Applied PCA and t-SNE for feature reduction and visual exploration.

**4. Hyperparameter Tuning**

Tools Used:

* GridSearchCV for exhaustive search.
* RandomizedSearchCV for randomized subset sampling.
* Optuna for Bayesian optimization.

Example: Random Forest Tuning

param\_grid = {

'n\_estimators': [100, 200, 300],

'max\_depth': [10, 20, 30],

'min\_samples\_split': [2, 5],

'min\_samples\_leaf': [1, 2],

'bootstrap': [True, False]

}

Impact:

Improved validation accuracy by 7%.

Reduced training time by 20% by limiting tree depth.

Enhanced F1 Score consistency across folds.

**5. Cross-Validation**

Cross-validation was pivotal for reliable model assessment:

Initial Strategy: 5-Fold CV without stratification.

Refined Strategy: 10-Fold Stratified CV to preserve class balance in each fold.

Leave-One-Out CV (LOOCV): Used on small datasets to reduce bias.

**Benefits:**

Reduced variance in validation scores.

Enhanced confidence in generalization.

Provided insights into model stability across different data segments.

**6. Feature Selection**

If applicable, describe any feature selection methods employed during model refinement. **Techniques Applied:**

* Univariate Feature Selection (SelectKBest)
* Recursive Feature Elimination (RFE)
* Embedded Methods (Lasso, Tree Feature Importance)
* PCA for dimensionality reduction

**Outcomes:**

* Reduced feature count from 58 to 35.
* Boosted model interpretability.
* Maintained or improved accuracy.
* Reduced overfitting and training duration.

**Test Submission**

**1. Overview**

The test submission phase involved preparing the final version of the model for evaluation on unseen data. It simulated real-world deployment by applying the trained model on an isolated test dataset to assess its robustness and performance without retraining.

**2. Data Preparation for Testing**

Explain how the test dataset was prepared, and any specific considerations taken into account To ensure consistency and reliability:

Data Cleaning: Handled missing values using imputation strategies.

Standardization: Used StandardScaler and MinMaxScaler for feature scaling.

Encoding: Applied one-hot encoding and label encoding.

Feature Matching: Ensured feature order and transformation matched training pipeline using ColumnTransformer.

from sklearn.preprocessing import StandardScaler

from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer(

transformers=[('num', StandardScaler(), numeric\_features)]

)

**3. Model Application**

The final model was loaded and applied to the test dataset using the exact transformation pipeline:

import joblib

from sklearn.pipeline import Pipeline

model\_pipeline = joblib.load("best\_model\_pipeline.pkl")

test\_preds = model\_pipeline.predict(test\_data)

The pipeline included preprocessing, transformation, and prediction in a single flow for production-readiness.

**4. Test Metrics**

| **Metric** | **Test Set Value** |
| --- | --- |
| Accuracy | 90.5% |
| Precision | 88% |
| Recall | 89% |
| F1 Score | 88.5% |
| ROC AUC | 0.91 |

Performance was consistent with cross-validation scores, confirming good generalization.

Confusion matrix and ROC curves were analyzed to inspect misclassification behavior.

**5. Model Deployment**

Deployment Strategy:

API Creation: Using FastAPI for REST endpoints.

Model Serialization: Saved as .pkl using joblib.

Dockerization: Packaged with Docker for reproducibility and environment consistency.

CI/CD: Used GitHub Actions for continuous integration and deployment to cloud platforms like Heroku or AWS.

bash

docker build -t ml-api .

docker run -p 8000:8000 ml-api

**6. Code Implementation**

from fastapi import FastAPIimport joblibimport pandas as pd

app = FastAPI()

model = joblib.load("best\_model\_pipeline.pkl")

@app.post("/predict/")def predict(input\_data: dict):

df = pd.DataFrame([input\_data])

prediction = model.predict(df)

return {"prediction": prediction.tolist()}

**Conclusion**

The model refinement and test submission phases were instrumental in transforming a baseline model into a high-performing, production-ready solution. Through systematic experimentation, feature engineering, and validation strategies, we achieved strong performance with reliable generalization. The deployment pipeline ensures scalability and integration into real-world applications.

**Key Challenges Encountered:**

Imbalanced datasets

Feature collinearity

Overfitting in complex models

**Achievements:**

7–10% improvement in key performance metrics.

Reduced model complexity by 40%.

Deployed an end-to-end ML system via REST API.

**References**

1.Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Ed.

2.Scikit-learn Documentation – [https://scikit-learn.org/](https://scikit-learn.org/" \t "_new)

3.XGBoost Documentation – [https://xgboost.readthedocs.io/](https://xgboost.readthedocs.io/" \t "_new)

4.FastAPI Documentation – https://fastapi.tiangolo.com/

5.Optuna – [https://optuna.org/](https://optuna.org/" \t "_new)

6.SMOTE for Imbalanced Learning – [https://imbalanced-learn.org/](https://imbalanced-learn.org/" \t "_new)