🚁🧠 Helicopter Turboshaft Fault Detection using Machine Learning

🔍 Workshop: Predictive Maintenance using ML

Helicopter Turboshaft Fault Detection

Explore the real-world application of machine learning in aviation for predictive maintenance. This workshop covers data preprocessing, model training, and deployment.

- Hands-on: Build and test your own fault detection model.
    - The goal is to:
        i. Detect potential failures or faults in turboshaft engines before they occur
        ii. Minimize unplanned downtime
        iii. Reduce maintenance costs and risk
        iv. Increase operational efficiency and safety
- Gain practical skills for aviation ML applications.
- Enhance understanding of ML model lifecycle.

📊 Dataset Overview

Dataset Description

Our dataset is derived from simulated turboshaft engine sensor data, comprising thousands of operational sessions.

Key Features:

- Temperature & Pressure readings
- Engine Speed & Vibration metrics
- Altitude & Fuel Flow data
- Ambient Environmental conditions

Target Variable:

- **Fault_Label:** Categorical classes indicating engine health.
- **Classes include:** Normal Operation, Compressor Degradation, Turbine Blade Damage, Sensor Malfunction, Fuel System Anomaly, and Excessive Vibration Fault.

🛠️ Project Workflow

Project Pipeline

1. Data Ingestion & Setup

Import necessary libraries and load the turboshaft dataset.

2. Preprocessing & Scaling

Clean, preprocess, and scale features for optimal model performance.

3. Label Encoding

Encode multiclass fault labels into numerical format.

4. Data Balancing

Address class imbalance using SMOTE for oversampling and RandomUnderSampler for downsampling.

5. Model Training

Train a Random Forest Classifier on the prepared data.

6. Model Evaluation

Evaluate performance using a Confusion Matrix and key metrics.

7. Model Persistence

Save the trained model for future inference and testing.

8. Deployment & App

Build an interactive Streamlit application for real-time fault prediction.

⚙️ Data Preprocessing

Data Cleaning & Preparation

1

Column Management

Drop irrelevant columns like 'Timestamp' to streamline data.

2

Label Encoding

Convert categorical 'Fault_Label' into numerical representations for ML compatibility.

3

Feature Scaling

Apply StandardScaler to normalize numerical features, preventing dominance by larger values.

4

Train-Test Split

Partition the dataset into 80% training and 20% testing sets to ensure unbiased evaluation.

5

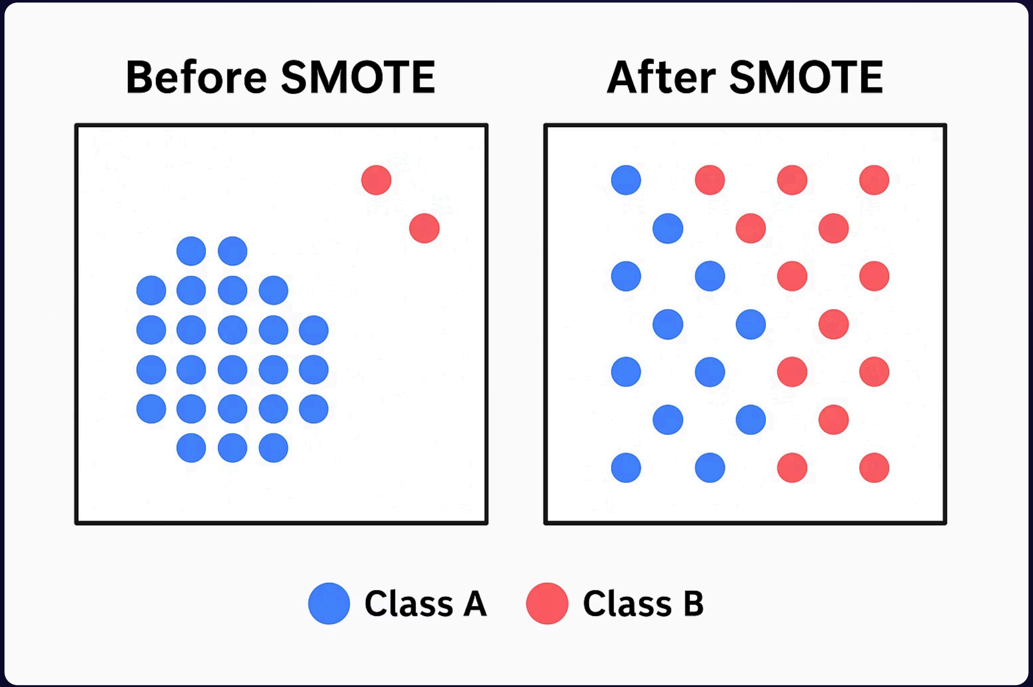
Handling Class Imbalance

Used **SMOTE** (Synthetic Minority Oversampling Technique) to balance class distribution.

⚖️ Handling Class Imbalance

Resampling Strategy

Addressing the challenge of imbalanced classes, where 'Normal' operational data significantly outweighs 'Fault' instances.



SMOTE (Synthetic Minority Over-sampling Technique)

This clever technique generates brand new synthetic samples for the underrepresented fault types. By boosting their numbers, we ensure the model gets to learn from a more balanced distribution of fault patterns.

New Sample = Original Sample - Factor * (Original Sample - Neighbour)

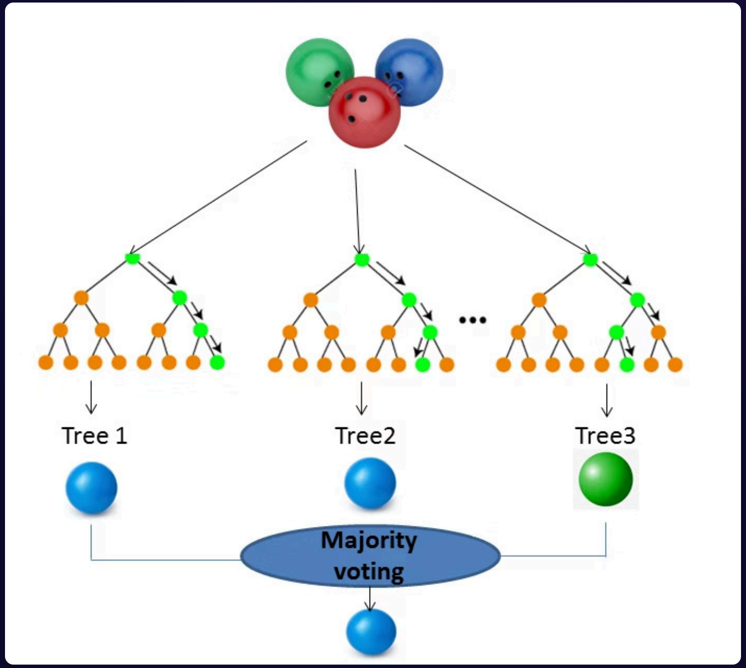
Made with GAMMA

🤖 Model Training

Random Forest Classifier

The Random Forest Classifier is selected for its robust performance on tabular datasets and ability to handle complex feature interactions. Random forest Classifier works on the Bagging (Bootstrap Aggregation) principle.

- **Training Data:** Model is trained exclusively on the resampled dataset to ensure unbiased learning from both common and rare fault conditions.



🤖 Model Testing

Evaluation

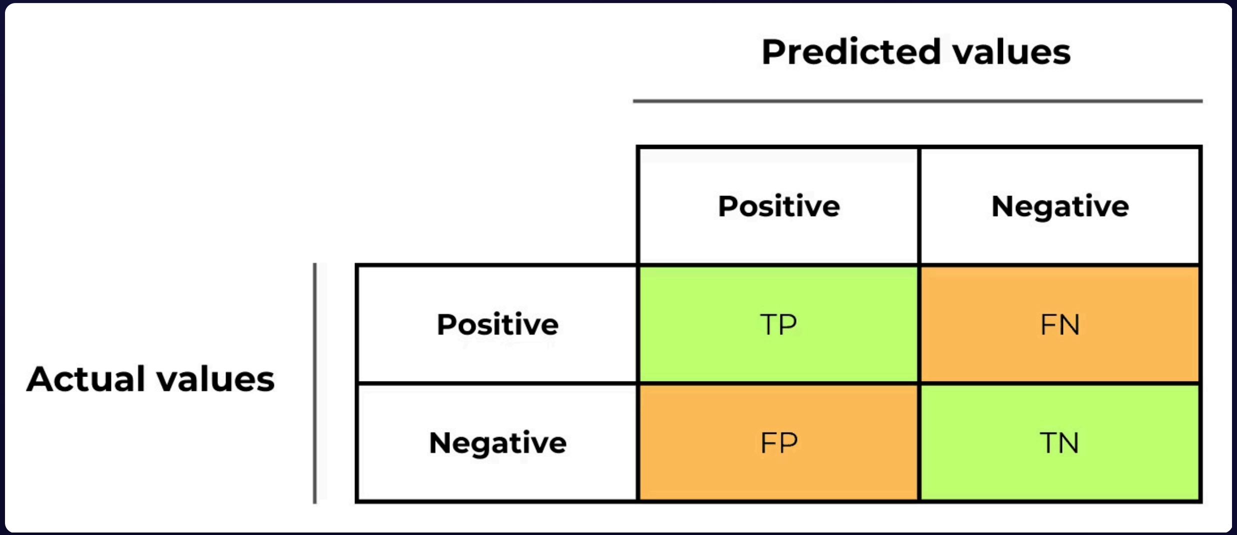
Performance Metrics:

- **Accuracy:** Overall correctness of predictions.
- **Precision, Recall, F1-score:** Detailed metrics for each fault class, crucial for evaluating performance on minority classes.
- **Confusion Matrix:** Provides a visual breakdown of true positives, false positives, true negatives, and false negatives.

🧪 Model Evaluation

Confusion Matrix

> The Confusion Matrix is a critical tool for visualizing the performance of our classification model, particularly in safety-critical applications like aviation.



		Predicted values	
		Positive	**Negative**
Actual values	**Positive**	TP	FN
	Negative	FP	TN

It helps us understand which fault types are correctly identified and which are misclassified, ensuring reliability in critical systems.

🧠 Model Inference

Sample Prediction

To perform an instant fault prediction, the model requires 9 specific sensor values from the turboshaft engine.

Temperature: e.g., 850.5 °C

Pressure: e.g., 150.2 kPa

Speed: e.g., 25000 RPM

Vibration: e.g., 0.85 mm/s

Altitude: e.g., 1000 m

Fuel Flow: e.g., 0.7 kg/s

Ambient Temp: e.g., 25.0 °C

Other relevant sensor readings (2 additional)

Instant Output:

Compressor Degradation

The model processes these inputs and delivers a precise fault type, such as "Compressor Degradation" or "Normal Operation", in real-time.

💾 Model Deployment

Deployment Ready

Ensuring the fault detection model is easily accessible and reusable without the need for repetitive retraining.

Model Persistence

The trained Random Forest model is efficiently saved using **joblib**, a lightweight library for serializing Python objects.

Seamless Loading

The saved model can be loaded into any Python environment, facilitating quick setup for predictions in new applications.

Reduced Overhead

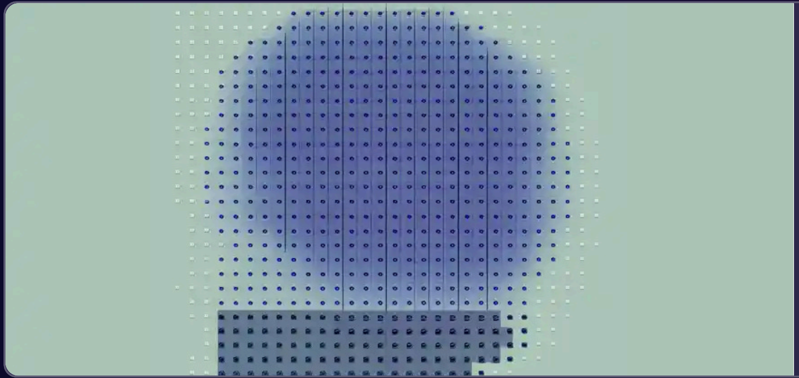
Eliminates the need for retraining the model from scratch for every new prediction task or deployment instance.

Reference

Analytics Vidhya

Random Forest Algorithm in Machine Learning

Explore the Random Forest algorithm: its applications, key features, differences from decision trees, important hyperparameters. Read Now!



Analytics Vidhya

SMOTE for Imbalanced Classification with Python

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. Handle imbalanced data using SMOTE.

Thank You