**User manual:**

**Enhancing the School Bus Breakdown and Delay Predictions**

Contents

[1. Introduction 3](#_Toc160809049)

[1.1 Purpose 3](#_Toc160809050)

[1.2 Scope 3](#_Toc160809051)

[1.3 Features 3](#_Toc160809052)

[2. System Requirements 3](#_Toc160809053)

[2.1 Software Requirements 3](#_Toc160809054)

[2.2 Hardware Requirements 3](#_Toc160809055)

[3. Installation Guide 3](#_Toc160809056)

[3.1 Setting Up the Development Environment 3](#_Toc160809057)

[3.2 Installing Required Libraries 4](#_Toc160809058)

[4. Data Loading 5](#_Toc160809059)

[4.1 Obtaining the Dataset 5](#_Toc160809060)

[4.2 Loading the Dataset into Pandas DataFrame 5](#_Toc160809061)

[5. Data Preprocessing 5](#_Toc160809062)

[5.1 Handling Duplicates 9](#_Toc160809063)

[5.2 Balancing the Dataset using SMOTE 9](#_Toc160809064)

[5.3 Feature Engineering 9](#_Toc160809065)

[6. Exploratory Data Analysis (EDA): 10](#_Toc160809066)

[7. Splitting the Dataset into Training and Testing Sets: 11](#_Toc160809067)

[8. Model Training and Evaluation 12](#_Toc160809068)

[8.1 Decision Tree Classifier 12](#_Toc160809069)

[Train and evaluate a decision tree classifier for predicting delays. 12](#_Toc160809070)

[8.2 Decision Tree Classifier (Balanced Class) 13](#_Toc160809071)

[8.3 Random Forest Classifier 14](#_Toc160809072)

[Apply and evaluate the performance of a random forest classifier. 14](#_Toc160809073)

[8.4 Random Forest Classifier (Balanced Class) 14](#_Toc160809074)

[8.5 XGBoost Classifier 16](#_Toc160809075)

[8.6 XGBoost Classifier (Balanced Class) 16](#_Toc160809076)

[9. Results and Analysis 18](#_Toc160809077)

# 1. Introduction

## 1.1 Purpose

Enhancing the School Bus Breakdown and Delay Predictions aims to predict and classify bus delays using machine learning techniques. It provides a comprehensive analysis of various classifiers, including Decision Tree, Random Forest, and XGBoost.

## 1.2 Scope

The system focuses on handling imbalanced datasets through the application of the Synthetic Minority Over-sampling Technique (SMOTE). It targets users interested in analyzing and improving the classification of bus delays in Intelligent Transportation Systems.

## 1.3 Features

* Data loading and preprocessing
* SMOTE for balancing imbalanced datasets
* Training and evaluating Decision Tree, Random Forest, and XGBoost classifiers
* Visualizing and analyzing performance metrics

# 2. System Requirements

## 2.1 Software Requirements

Ensure the following software is installed on your system:

* Python (3.x recommended)
* Jupyter Notebook

## 2.2 Hardware Requirements

The system is designed to run on standard computer hardware. No specific hardware requirements are necessary

# 3. Installation Guide

## 3.1 Setting Up the Development Environment

1. Install Anaconda Navigator:

* Download Anaconda Navigator from the official website: [Anaconda Navigator Download](https://www.anaconda.com/anaconda-navigator)
* Follow the installation instructions for your operating system.

2. Launch Anaconda Navigator:

* Open Anaconda Navigator after the installation is complete.

3. Create a New Environment:

* Click on the "Environments" tab in the Anaconda Navigator.
* Create a new environment by clicking the "Create" button.
* Provide a name for your environment and select the Python version.

4. Install Required Libraries:

* Switch to the "Home" tab in Anaconda Navigator.
* Choose the environment you created.
* Open the "Home" tab, select the environment, and click on "Install" under Jupyter Notebook.

5. Launch Jupyter Notebook:

* In Anaconda Navigator, go to the "Home" tab.
* Select your environment.
* Launch Jupyter Notebook by clicking the "Launch" button.

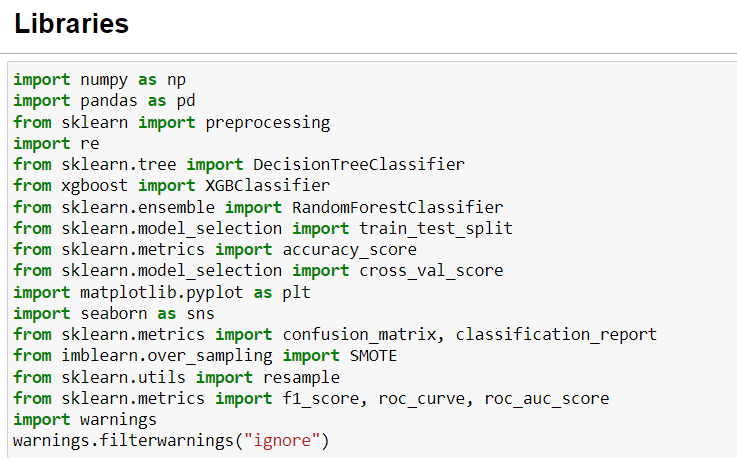
6. Open a Jupyter Notebook:

* Once Jupyter Notebook starts, navigate to the folder where you have your project files.
* Create a new Jupyter Notebook or open an existing one.

## 3.2 Installing Required Libraries

Open a terminal to install following libraries by run the following commands:

* NumPy
* Pandas
* Scikit-learn
* Matplotlib
* Seaborn
* Imbalanced-learn



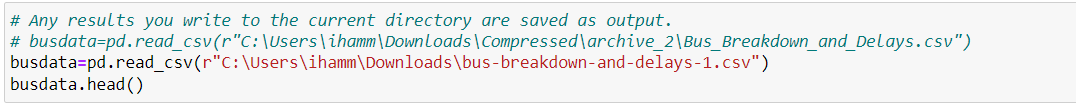
# 4. Data Loading

## 4.1 Obtaining the Dataset

Download the dataset (Bus\_Breakdown\_and\_Delays.csv) from [Data.World.](https://data.world/city-of-ny/ez4e-fazm)

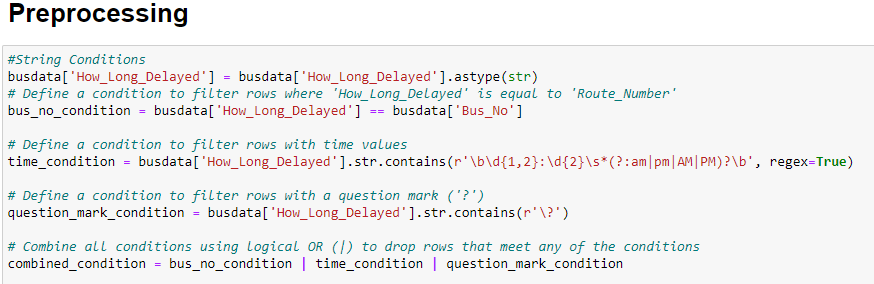
## 4.2 Loading the Dataset into Pandas DataFrame

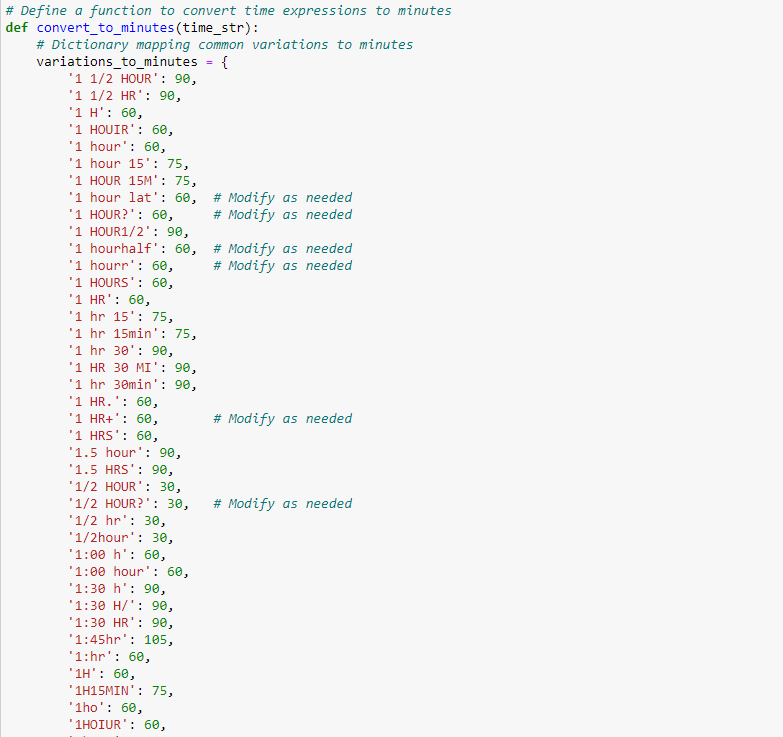
Use the provided code to load the dataset into a Pandas DataFrame using Jupyter Notebook.

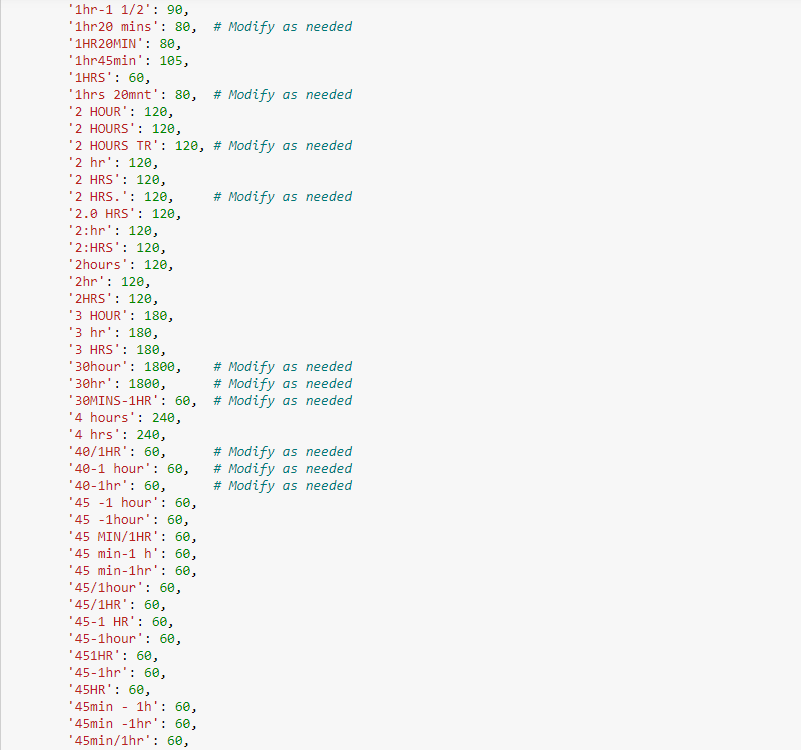
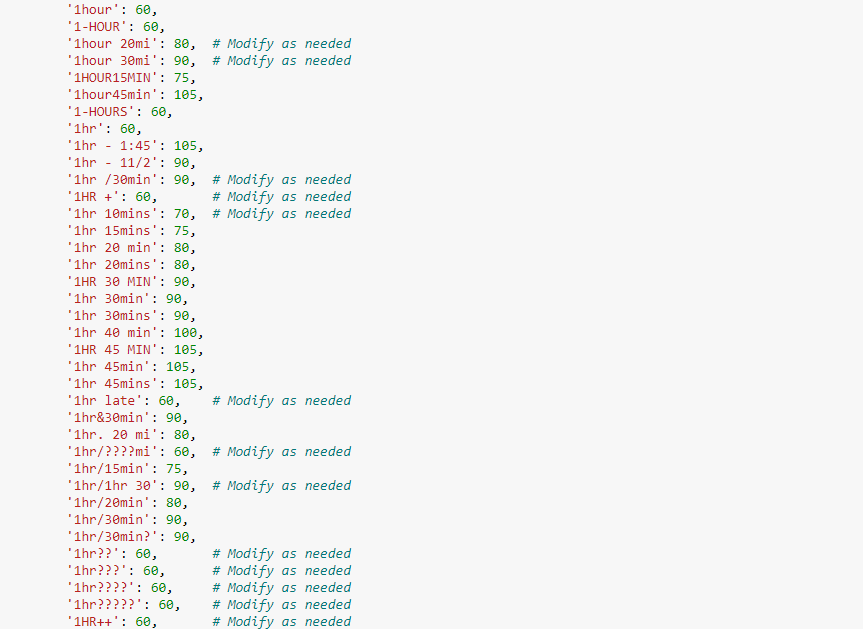


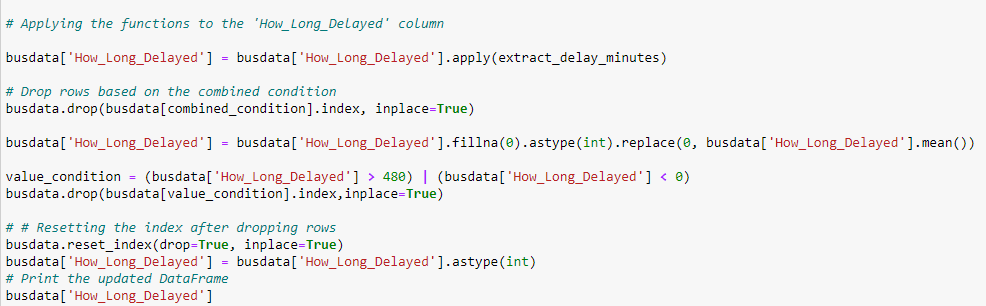
# 5. Data Preprocessing

Load the dataset into the project and perform initial preprocessing steps. This may involve handling missing values, converting data types, and organizing the data for further analysis.





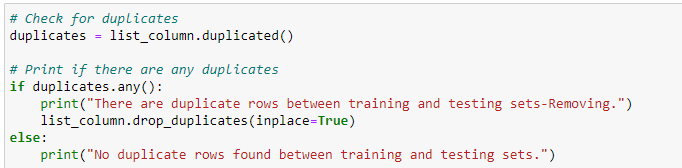


Converting Data Types for Tree Models: Adjust data types, if necessary, especially for tree-based machine learning models.



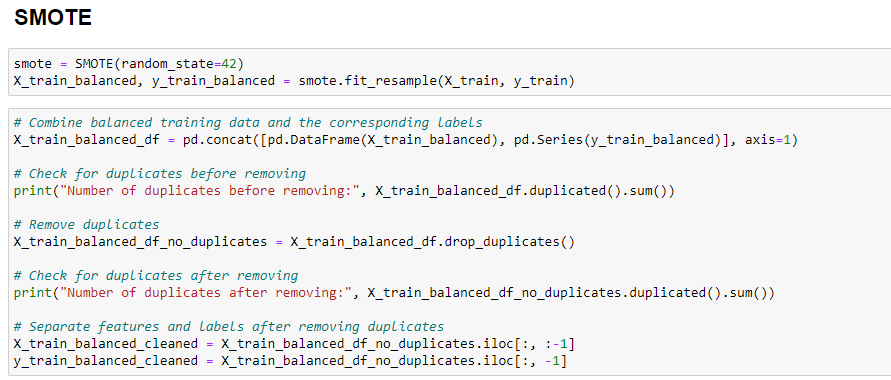
## 5.1 Handling Duplicates

The system includes functionality to handle duplicates in the dataset.



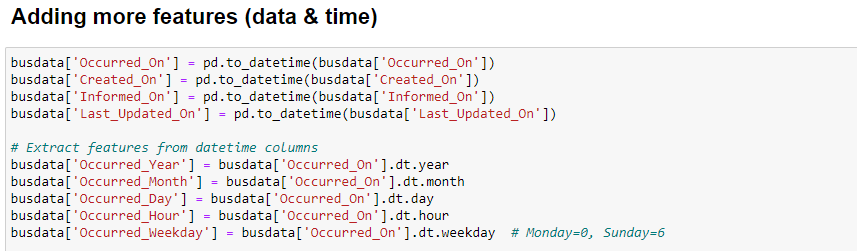
## 5.2 Balancing the Dataset using SMOTE

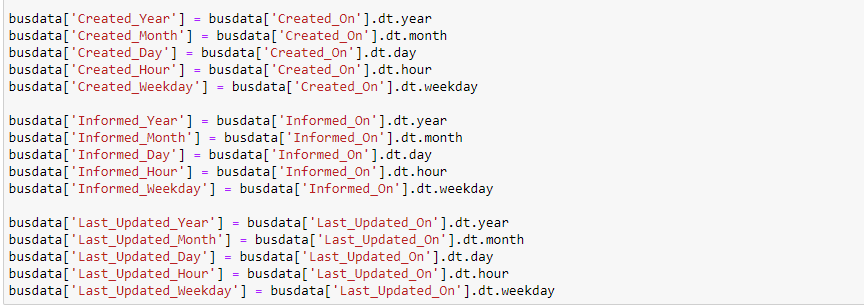
SMOTE is applied to balance the dataset, as shown in the code.



## 5.3 Feature Engineering

The system performs feature engineering to enhance the model's predictive capabilities.



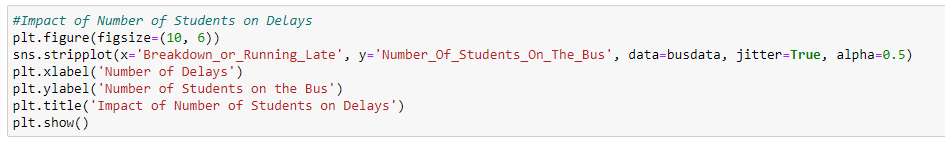




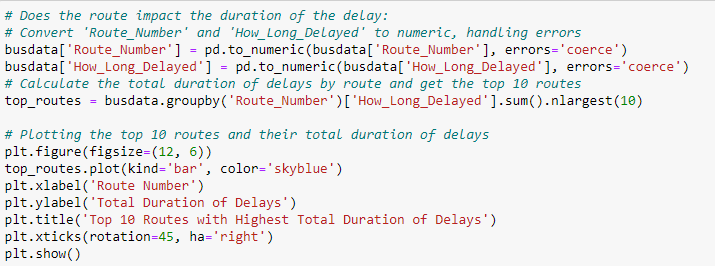
# 6. Exploratory Data Analysis (EDA):

Conduct exploratory data analysis to gain insights into the dataset. This step often involves visualizations and statistical analysis to understand the distribution and relationships within the data.



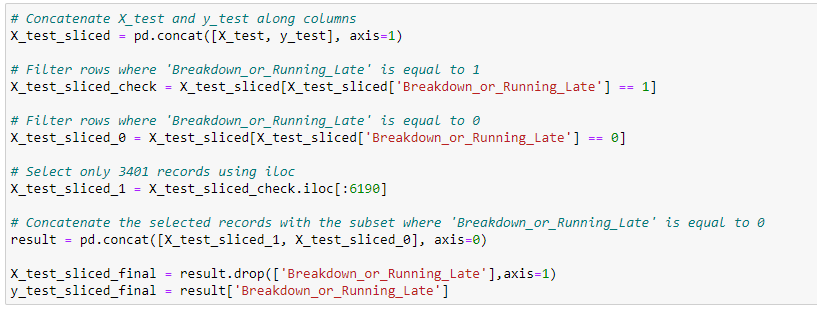






# 7. Splitting the Dataset into Training and Testing Sets:

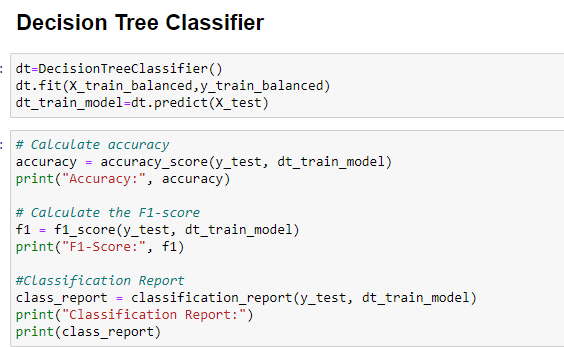
Split the dataset into training and testing sets to evaluate the model's performance.



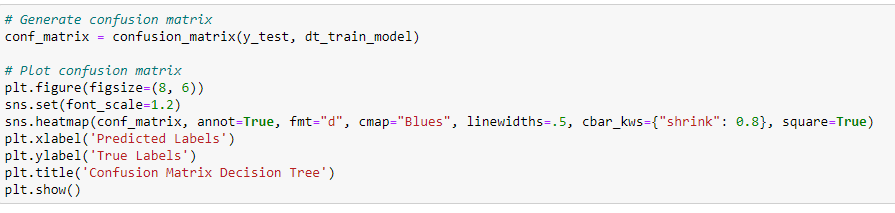
# 8. Model Training and Evaluation

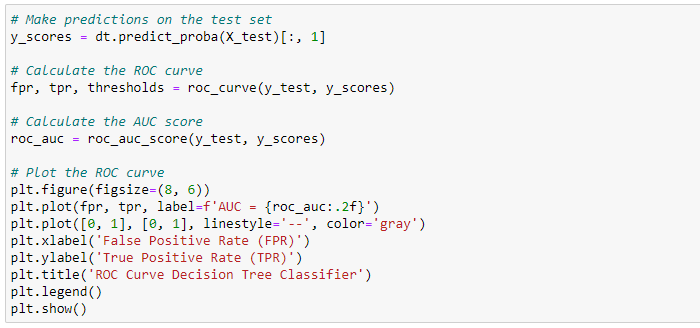
## 8.1 Decision Tree Classifier

## Train and evaluate a decision tree classifier for predicting delays.



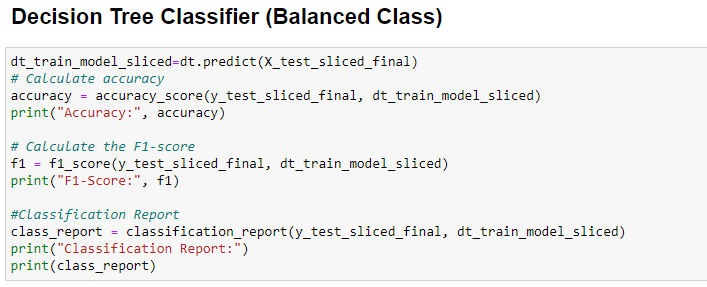
Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.



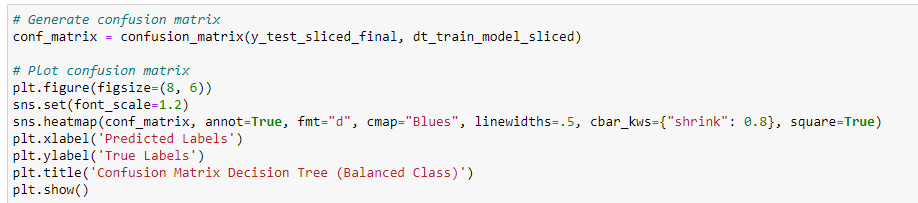


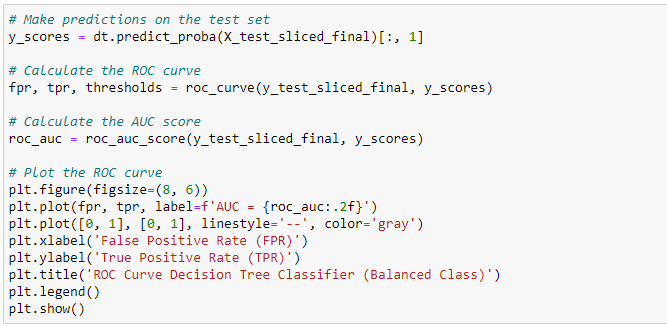
## 8.2 Decision Tree Classifier (Balanced Class)

Balancing Classes for Model Evaluation: Address class imbalances during the model evaluation phase.



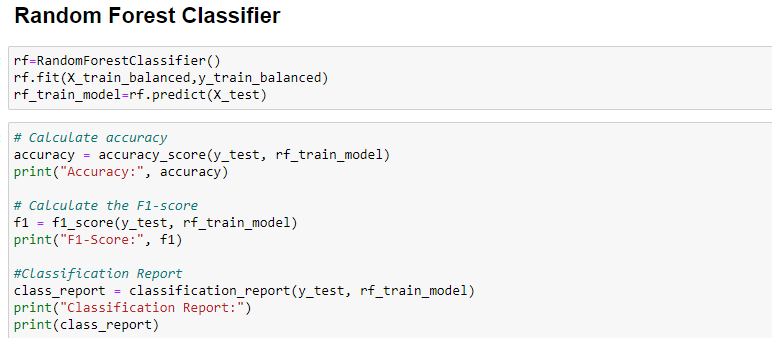
Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.



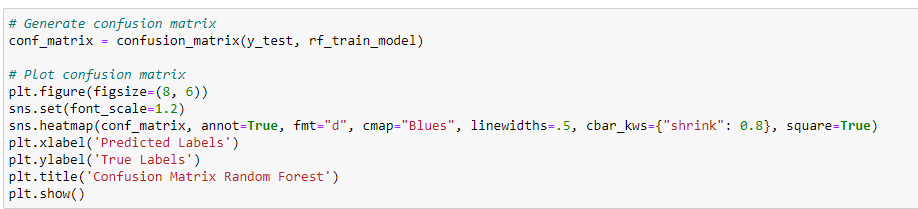


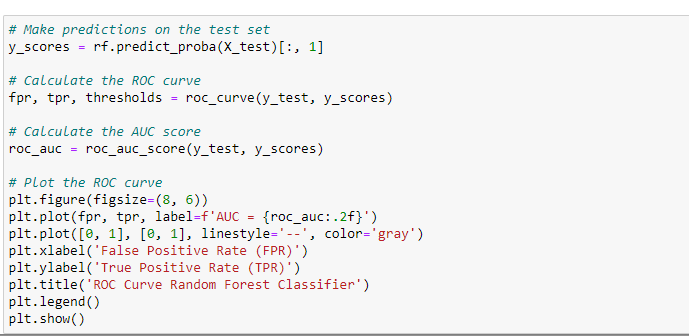
## 8.3 Random Forest Classifier

## Apply and evaluate the performance of a random forest classifier.



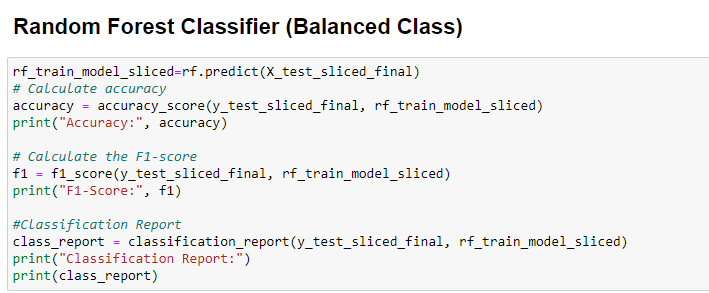
Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.



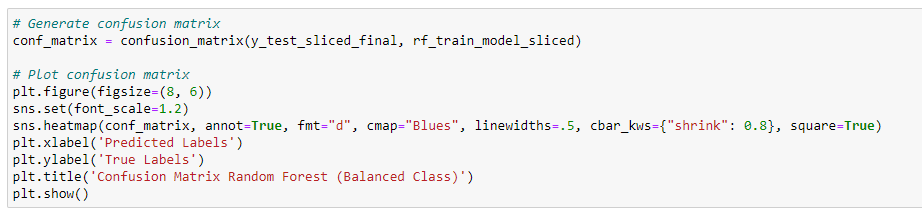


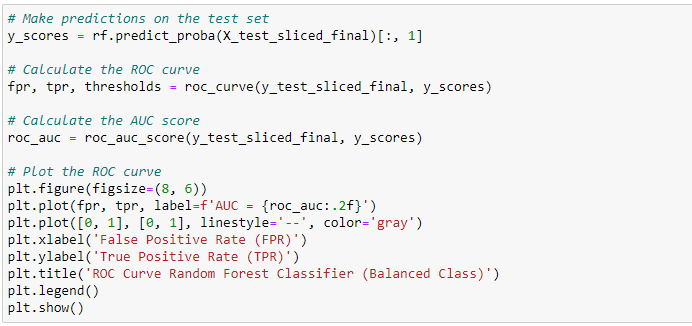
## 8.4 Random Forest Classifier (Balanced Class)

Address class imbalances during the model evaluation phase.



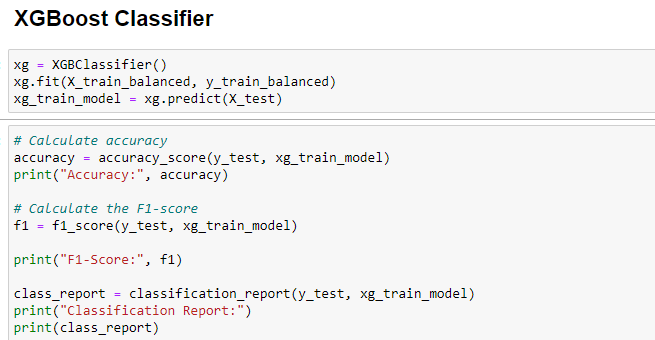
Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.





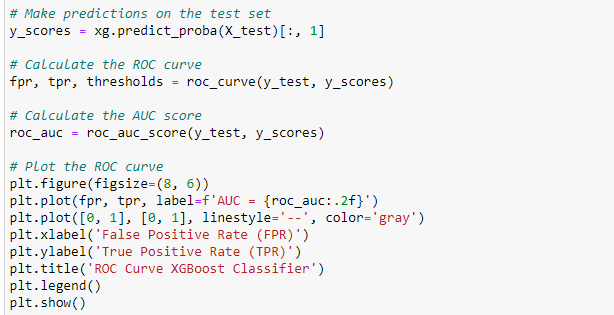
## 8.5 XGBoost Classifier

Implement and evaluate an XGBoost classifier for predicting delays.



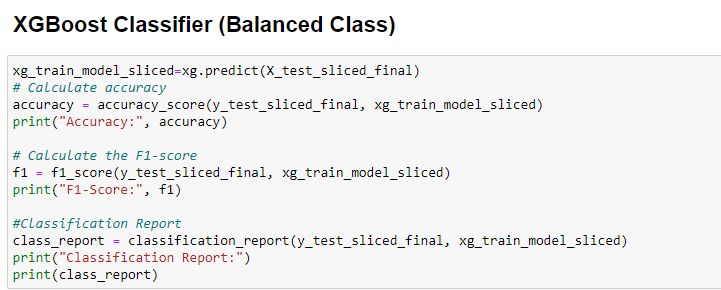
Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.



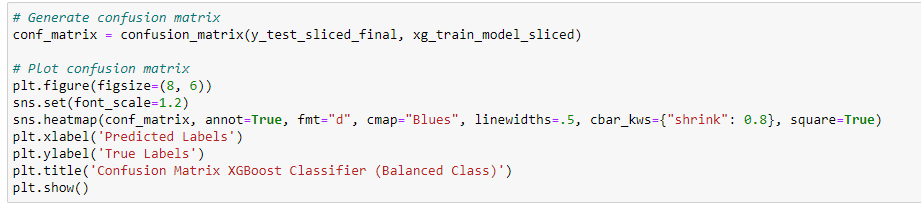


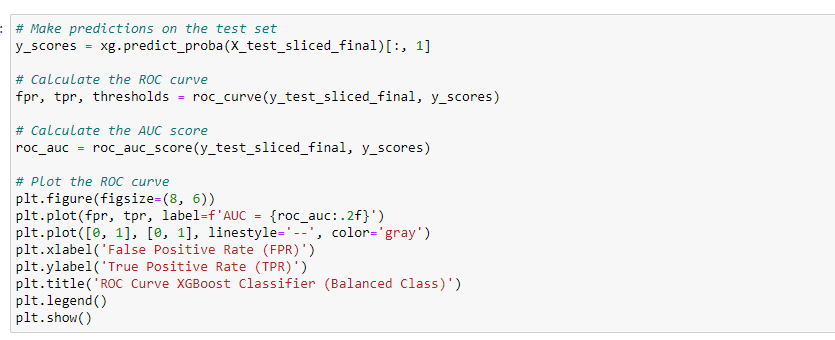
## 8.6 XGBoost Classifier (Balanced Class)

Address class imbalances during the model evaluation phase.



Confusion Matrix and ROC Curve: Utilize confusion matrices and ROC curves to evaluate and visualize model performance.





# 9. Results and Analysis

The Performance Metrics Before and After Balancing for Decision Tree, Random Forest, and XGBoost classifiers is shown in below table.

