ASSIGNMENT 2:
Submission by Group_60

AIM: To develop a machine learning-based model on the training dataset, where the training dataset has an expression of genes and labels (1 for high risk and 0 for low risk)

DATA PROVIDED: Train, Test and Sample Data

Approach:

The provided code is a crucial initial setup for a machine learning project, with a focus on importing necessary libraries and configuring the environment for data preprocessing, feature selection, and model evaluation. This readme file containing a summary will provide an in-depth explanation of the code's components and their roles.

1. Library Imports

The code commences with a set of library imports, an integral step in establishing a machine-learning environment. Each imported library plays a distinct role:

pandas: This versatile library is fundamental for data manipulation and handling. It facilitates reading data from CSV files and storing it in data structures like DataFrames, offering a robust platform for various data preprocessing tasks.

NumPy: As a core library for numerical operations, NumPy is indispensable for mathematical computations and array manipulation, providing the groundwork for many machine learning algorithms.

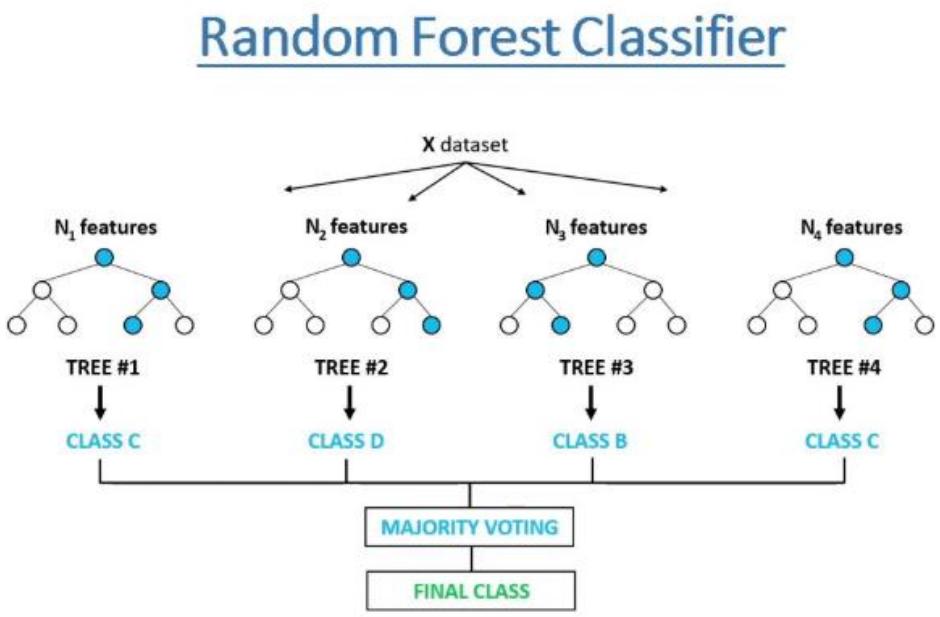
sklearn.model_selection: A part of scikit-learn, this module offers utilities for splitting datasets into training and testing sets. The code employs the train_test_split function from this module to partition the data accurately.

sklearn.ensemble: The ensemble module in scikit-learn is a treasure trove of ensemble-based machine learning algorithms. In this code, the focus is on the RandomForestClassifier, a popular choice for classification tasks.

sklearn.metrics: Scikit-learn's metrics module encompasses a plethora of functions for evaluating machine learning models. Within this code, the roc_auc_score metric is harnessed to assess the model's performance in terms of area under the receiver operating characteristic curve (AUC-ROC).

sklearn.preprocessing: This module is dedicated to data preprocessing and transformation. In the provided code, the StandardScaler is employed to standardize the input data, ensuring that features are on the same scale, which is crucial for various machine learning algorithms.

sklearn.feature_selection: Feature selection is a critical step in machine learning, aimed at identifying and retaining the most relevant features for model training. This code leverages the Recursive Feature Elimination (RFE) method, a powerful technique for feature selection.



2. Data Loading

After importing the essential libraries, the code proceeds to load the training and test data from CSV files. It assumes the existence of two files: "kaggle_train.csv" for training data and "kaggle_test.csv" for test data. The data is loaded into pandas DataFrames, a pivotal step that simplifies data handling and manipulation**.**

3. Extracting Test Data 'ID' Column

A specific operation is performed to extract the 'ID' column from the test data. This column, often included for tracking and submission purposes, does not contain predictive information. Following extraction, the 'ID' column is separated from the test data to ensure that it does not influence the modelling process.

4. Data Preparation

Data preparation is an indispensable phase in any machine-learning project. It entails the separation of data into features and labels. In this code, the features are stored in the variable X_train, and the corresponding labels are placed in y_train. This demarcation is essential for subsequent stages, including model training and evaluation.

5. Standardization

To guarantee that features are on a consistent and comparable scale, the code applies standardization to both the training data and the test data. This standardization process is executed through the StandardScaler, a component from the sklearn.preprocessing module. It is particularly vital when dealing with machine learning algorithms sensitive to feature scaling.

6. Feature Selection with RFE

Feature selection is a pivotal technique to identify and retain the most relevant features while discarding less informative ones. In this code, Recursive Feature Elimination (RFE) is utilized for this purpose. RFE operates iteratively, eliminating less important features from the dataset until a specified number of features remains. The parameter num_features_to_select governs the number of features to be retained. A RandomForestClassifier is engaged to assess the importance of features, and RFE systematically selects the most relevant ones.

7. Data Splitting

Data splitting is a critical step to ensure the integrity of the machine learning workflow. The training data is further divided into a training set and a validation set using the train_test_split function from sklearn.model_selection. This partitioning allows for model training on one portion of the data and evaluation on another. The split is performed with a test size of 20%, signifying that 80% of the data is allocated to the training set and 20% to the validation set. A random seed of 50 is applied to ensure reproducibility of the split.

8. Model Training

Having prepared the data, the code proceeds to train a machine learning model. The chosen algorithm for this task is the RandomForestClassifier. It is initialized with 100 decision trees (n_estimators=100) and a fixed random seed (random_state=42) to ensure that the model's behavior is consistent and reproducible.

9. Model Evaluation on Validation Set

The trained model is then employed to make predictions on the validation set. The predictions produced are in the form of probability scores, specifically the probability of belonging to class 1 in a binary classification task. These predicted probabilities are stored in the variable y_val_pred. To assess the performance of the model, the code computes the AUC-ROC score

using the roc_auc_score function. The AUC-ROC score is a widely used metric for evaluating binary classification models, measuring the area under the receiver operating characteristic curve.

10. Making Predictions on the Test Set

Once the model is trained and evaluated, it is applied to the test data to generate predictions for the target variable. These predictions are also in the form of probability scores, representing the likelihood of instances in the test set belonging to class 1. These predicted probabilities are stored for further processing**.**

11. Submission Preparation

A critical aspect of many machine learning projects is the preparation of results for submission, particularly in the context of data science competitions. In this code, a submission DataFrame is constructed to facilitate this process. The DataFrame comprises two columns:

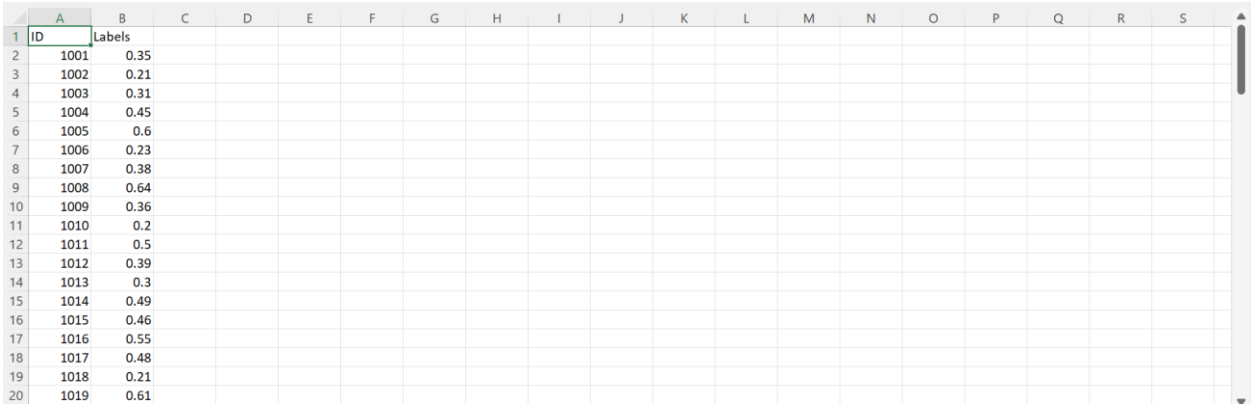
'ID': This column contains the 'ID' values extracted from the test data. It serves as a unique identifier for each instance in the test set.

'Labels': This column houses the model's predicted probabilities for class 1 for each test instance. These probabilities represent the model's predictions for the target variable.

The result is a structured DataFrame that simplifies the process of preparing and organising model predictions for submission.

12. Saving the Submission

In the final phase of the code, the submission DataFrame is saved to a CSV file. This file is named "group_60.csv." The purpose of saving the predictions in this format is to enable easy submission to a competition platform or to facilitate further analysis and reporting. The CSV format is a common choice for sharing results due to its simplicity and compatibility with various tools and platforms.



	A	B
1	ID	Labels
2	1001	0.35
3	1002	0.21
4	1003	0.31
5	1004	0.45
6	1005	0.6
7	1006	0.23
8	1007	0.38
9	1008	0.64
10	1009	0.36
11	1010	0.2
12	1011	0.5
13	1012	0.39
14	1013	0.3
15	1014	0.49
16	1015	0.46
17	1016	0.55
18	1017	0.48
19	1018	0.21
20	1019	0.61

This is the group_60.csv generated.