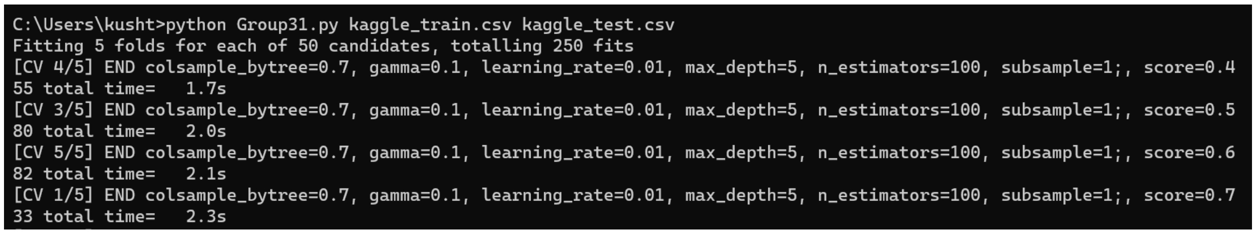
MLBA ASSIGNMENT-2

How to run the code?



```
C:\Users\kusht>python Group31.py kaggle_train.csv kaggle_test.csv
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV 4/5] END colsample_bytree=0.7, gamma=0.1, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.4
55 total time=   1.7s
[CV 3/5] END colsample_bytree=0.7, gamma=0.1, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.5
80 total time=   2.0s
[CV 5/5] END colsample_bytree=0.7, gamma=0.1, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.6
82 total time=   2.1s
[CV 1/5] END colsample_bytree=0.7, gamma=0.1, learning_rate=0.01, max_depth=5, n_estimators=100, subsample=1;, score=0.7
33 total time=   2.3s
```

Code explanation-

It uses the XGBoost algorithm, a popular and efficient gradient boosting framework.

1. **Import Libraries**

The script starts by importing the necessary libraries for data handling (pandas and numpy), machine learning (joblib, sklearn, and xgboost).

2. **Function Definitions**

Several functions are defined to handle different parts of the machine learning workflow:

load_data(train_path, test_path): Loads the training and test datasets from CSV files.

prepare_data(train_data, test_data, feature_columns, label_column, id_column): Prepares the data for training by selecting the features, the target variable, and the identifiers from the datasets.

feature_selection_and_standardization(X, y, X_test): Standardizes the feature columns and selects the most important features using an XGBClassifier as the base model and SelectFromModel for feature selection.

tune_hyperparameters(X, y): Performs hyperparameter tuning using RandomizedSearchCV with a specified parameter grid, cross-validation strategy, and scoring metric (ROC AUC).

evaluate_model(model, X, y): Evaluates the trained model using the ROC AUC score.

save_model(model, filename) and load_model(filename): Save and load a model using joblib.

3. Script Execution

This is the main part of the script where the functions are called to execute the machine learning pipeline:

Load datasets: The load_data function is called to load the train and test datasets.

Prepare data: If data loading is successful, the datasets are prepared using the prepare_data function. Feature columns, labels, and IDs are separated for training and test data.

Feature selection and standardization: The features are scaled and selected using the feature_selection_and_standardization function. This might help in improving the model performance and reducing overfitting.

Splitting data for validation: The training data is split into training and validation sets using train_test_split. This allows for an internal evaluation before making predictions on the test set.

Model training and hyperparameter tuning: The tune_hyperparameters function is called to find the best hyperparameters for the XGBClassifier and fit it on the training data.

Model evaluation: The best model obtained from hyperparameter tuning is evaluated on the validation set using the evaluate_model function.

Save the best model: The best model is saved to the file system using joblib for later use.

Predict on test data: The best model is used to make predictions on the test data.

Save the predictions: The predictions are saved into a CSV file formatted for submission (typically to a Kaggle competition).

If there is any error in loading the data, an error message is printed.

The script assumes that the input CSV files have a specific format, with a label column named 'Labels' and an ID column named 'ID'.
The feature selection is based on the median importance of the features as determined by the initial XGBoost model.
Hyperparameter tuning could take a long time since it uses RandomizedSearchCV with 50 iterations and a 5-fold stratified cross-validation strategy.
The model's performance is evaluated using the ROC AUC score, which is a common metric for binary classification problems.
The final output is a CSV file containing test IDs and their corresponding predicted probabilities.The provided Python script is designed to train a neural network model for binary classification using TensorFlow and Keras. The script is structured as a command-line utility, which takes input arguments for the training data file path, test data file path, and an optional submission file path.

Why i choose this model?
XGBoost is chosen for its strong performance in classification tasks, especially in competitive machine learning environments. It is efficient, provides a robust way to handle a variety of data types, and includes regularization to avoid overfitting. Additionally, XGBoost offers a powerful way to do feature selection and has built-in methods for handling missing data. Its hyperparameters are also highly tunable, allowing for optimization to achieve the best possible model performance.