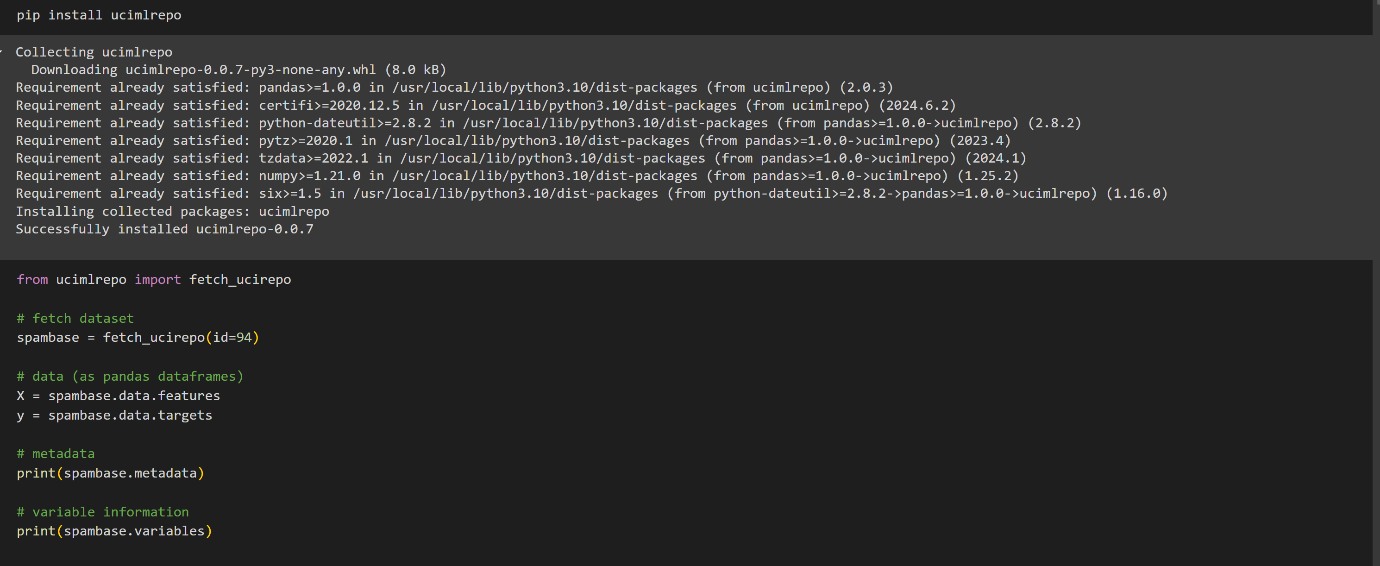
**Project Title: Spam Email Classifier**

**Project Description**: Create a machine learning model that can classify emails as spam or not spam (ham). This project will introduce you to text classification and binary classification tasks, commonly used in **natural language processing (NLP).**

* The data was collected from UCI Repository and was directed to Google Colab.The dependent and independent variables were assigned before beginning of the further processes
* In the next stage necessary libraries were added and it included. The libraries involved

**Numpy**: Provides support for large multi-dimensional arrays and matrices.

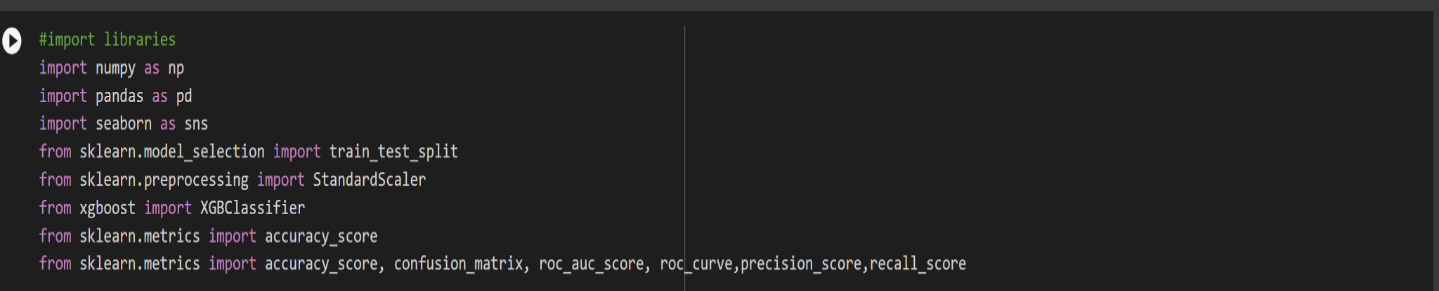
**Pandas**: Offers data structures and data analysis tools.

**Seaborn:** A visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**train\_test\_split**: A function to split the dataset into training and testing sets.

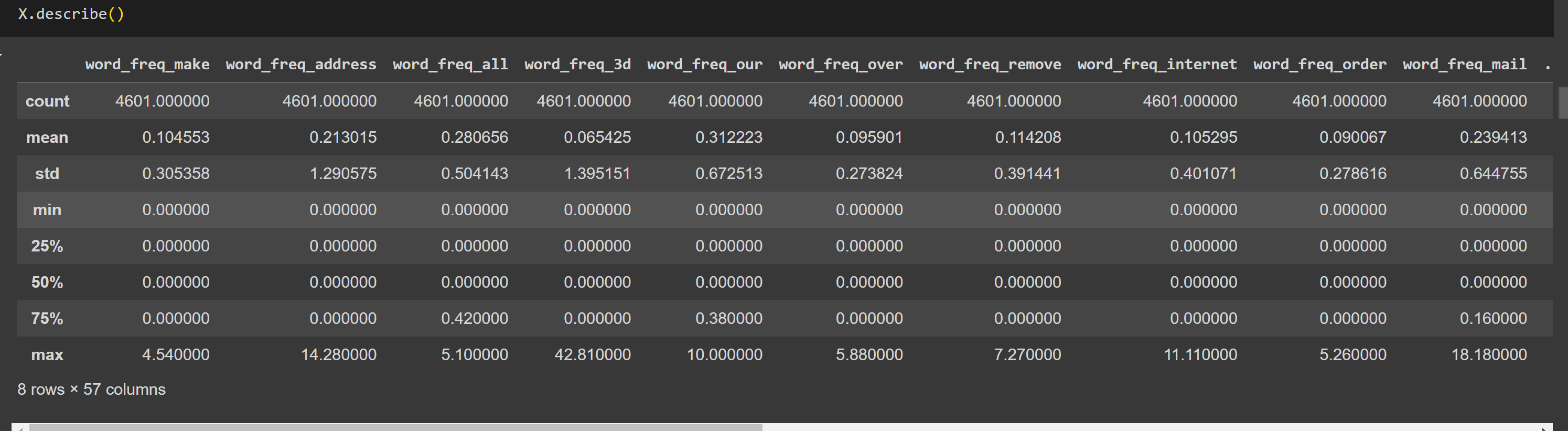
**StandardScaler**: Standardizes features by removing the mean and scaling to unit variance.

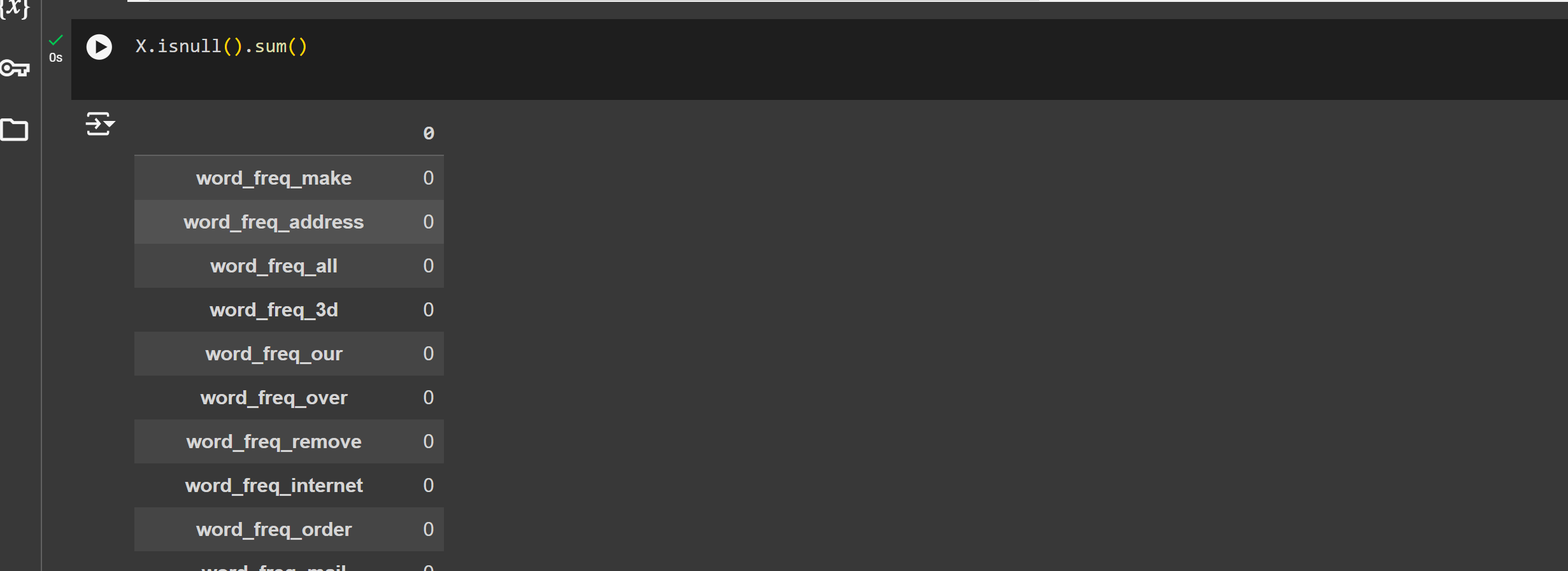
**accuracy\_score**: Evaluates the accuracy of the model.

 **confusion\_matrix, roc\_auc\_score, roc\_curve, precision\_score, recall\_score**: Metrics for evaluating the performance of classification models.

* **Exploratory Data Analysis**

In this stage,the data was visualized to get insights from the data and understand the main features in the dataset.The processes involved checking null values, removing duplicate data ,descrptive statistics,distribution of data.Also did various data visualizations to understand the trends in data.



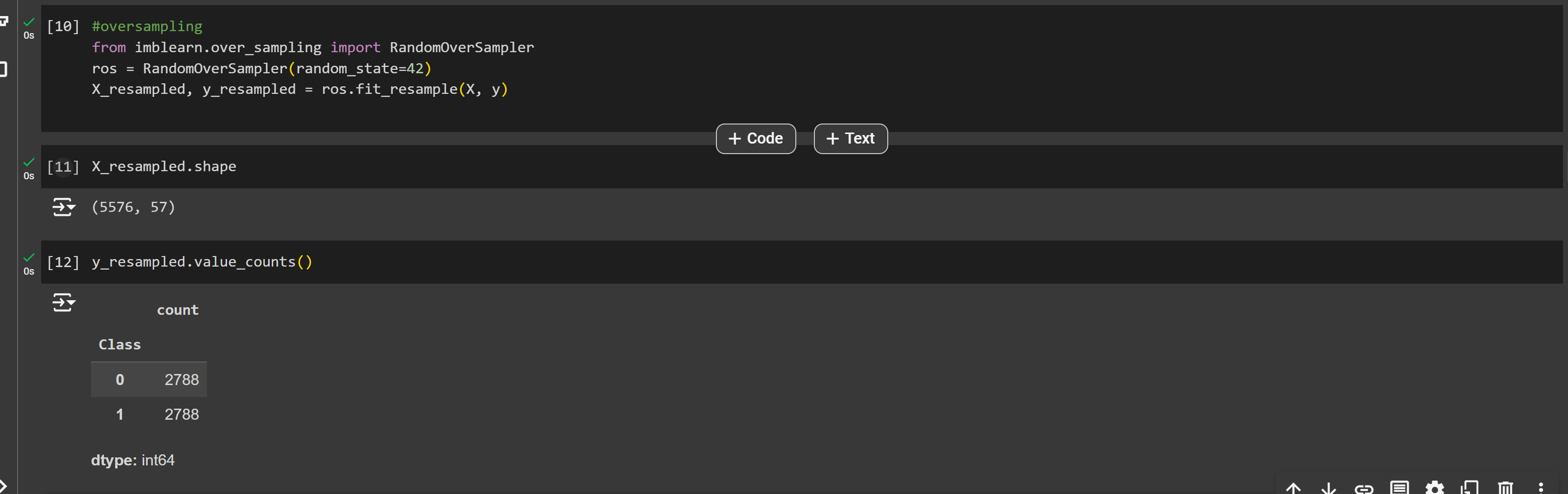


* **Handling Imbalanced data**

I used an oversampling technique to address the issue of imbalanced data in my machine learning project. Imbalanced datasets, where one class significantly outnumbers others, can result in models that are biased towards the majority class and fail to accurately predict the minority class. By employing oversampling, I generated additional samples for the minority class, effectively balancing the dataset.

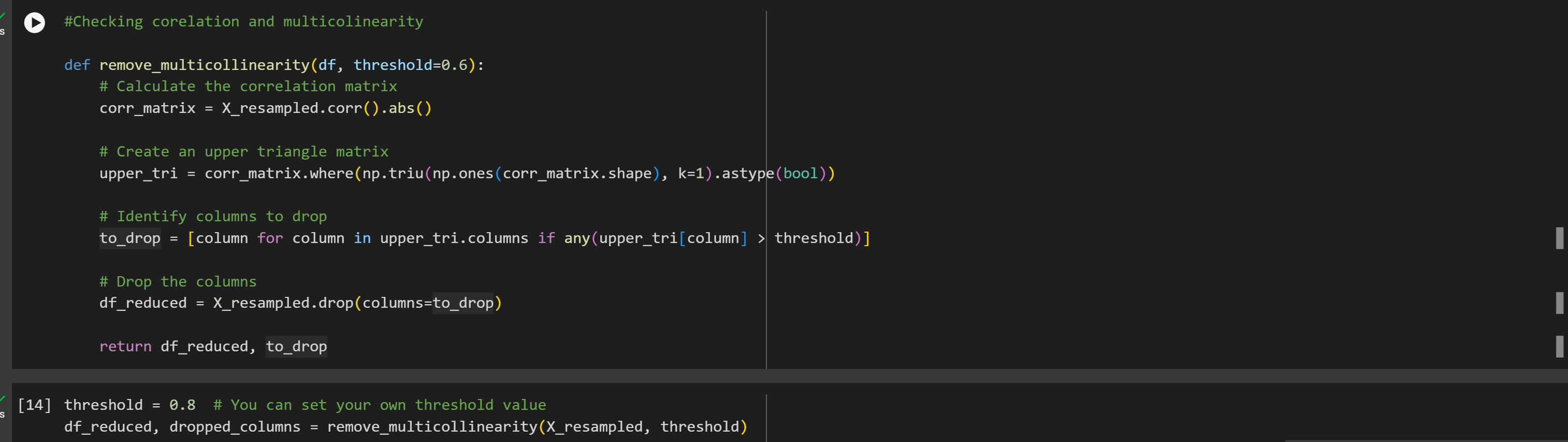
I applied the SMOTE (Synthetic Minority Over-sampling Technique) method to address the issue of imbalanced data in my machine learning project. Imbalanced datasets, where one class significantly outnumbers others, can lead to biased models that perform poorly on minority classes. By using SMOTE, I generated synthetic samples for the minority class, balancing the dataset and ensuring that the model receives an equal representation of all classes during training.

****

****

* **Check for Multicolinearity**

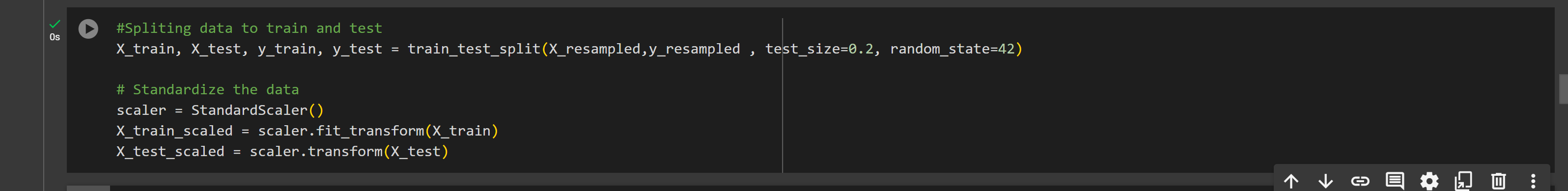
I checked for multicollinearity in my dataset as part of the data preprocessing steps in my machine learning project. Multicollinearity occurs when two or more predictor variables are highly correlated, leading to redundancy and instability in the model's coefficients. To address this, I calculated the Variance Inflation Factor (VIF) for each feature. High VIF values indicate multicollinearity, suggesting that those features should be removed or combined. By identifying and mitigating multicollinearity, I ensured that my model's estimates are reliable, and the results are interpretable, leading to better model performance and more accurate predictions.



* **Train Test Split & Standard Scalar**

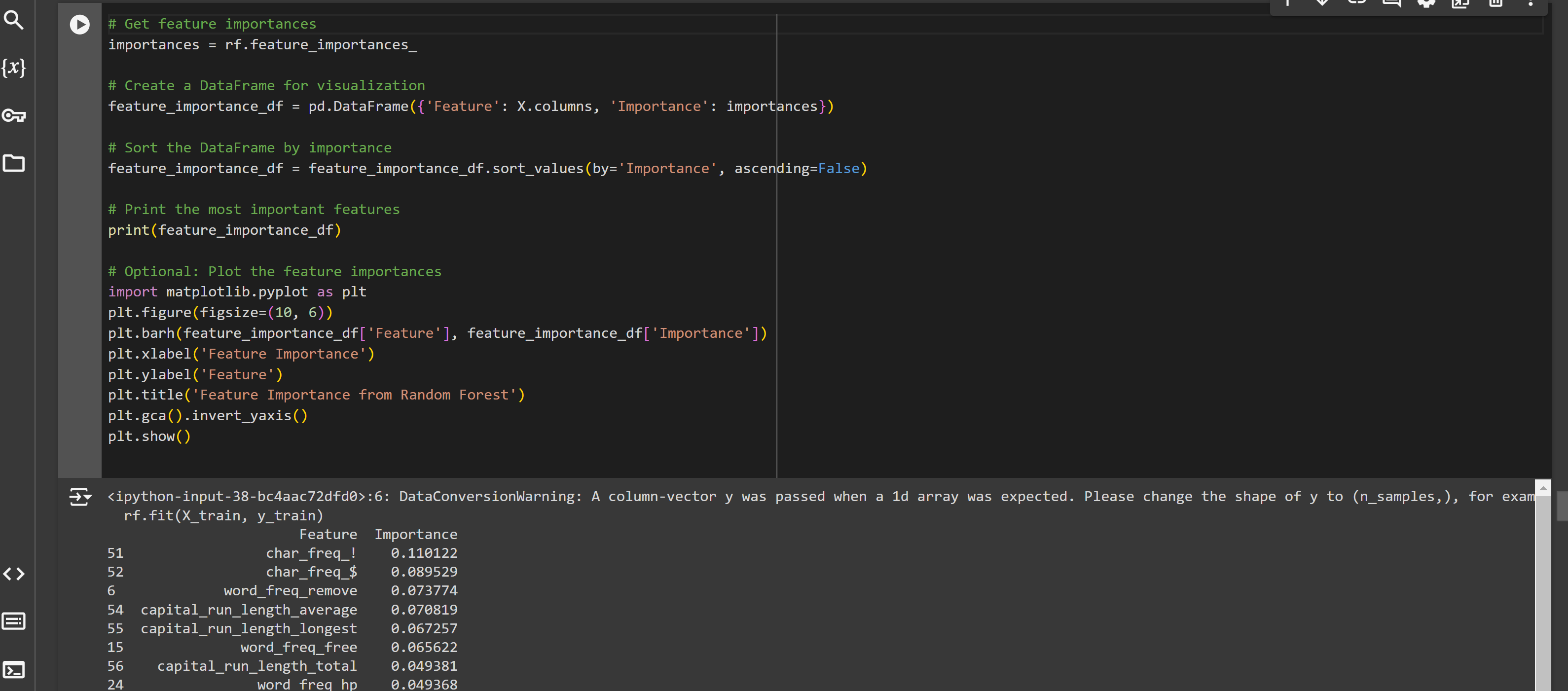
I split the data into training and test sets and applied scaling using the StandardScaler to ensure that my machine learning model performs optimally and yields reliable results. Splitting the data into training and test sets is essential to evaluate the model's performance on unseen data. It allows me to train the model on one subset of the data and test its generalization capability on another, thus preventing overfitting.

Scaling the data using **StandardScaler**, which standardizes features by removing the mean and scaling to unit variance, ensures that all features contribute equally to the model. Without scaling, features with larger ranges could disproportionately influence the model, leading to biased predictions. Standardizing the data improves the convergence rate of gradient-based optimizers and helps models like Support Vector Machines (SVM) and **k-Nearest Neighbors (k-NN)** perform better. Overall, these preprocessing steps enhance the model's accuracy and generalization ability.

****

* **Feature Importance**

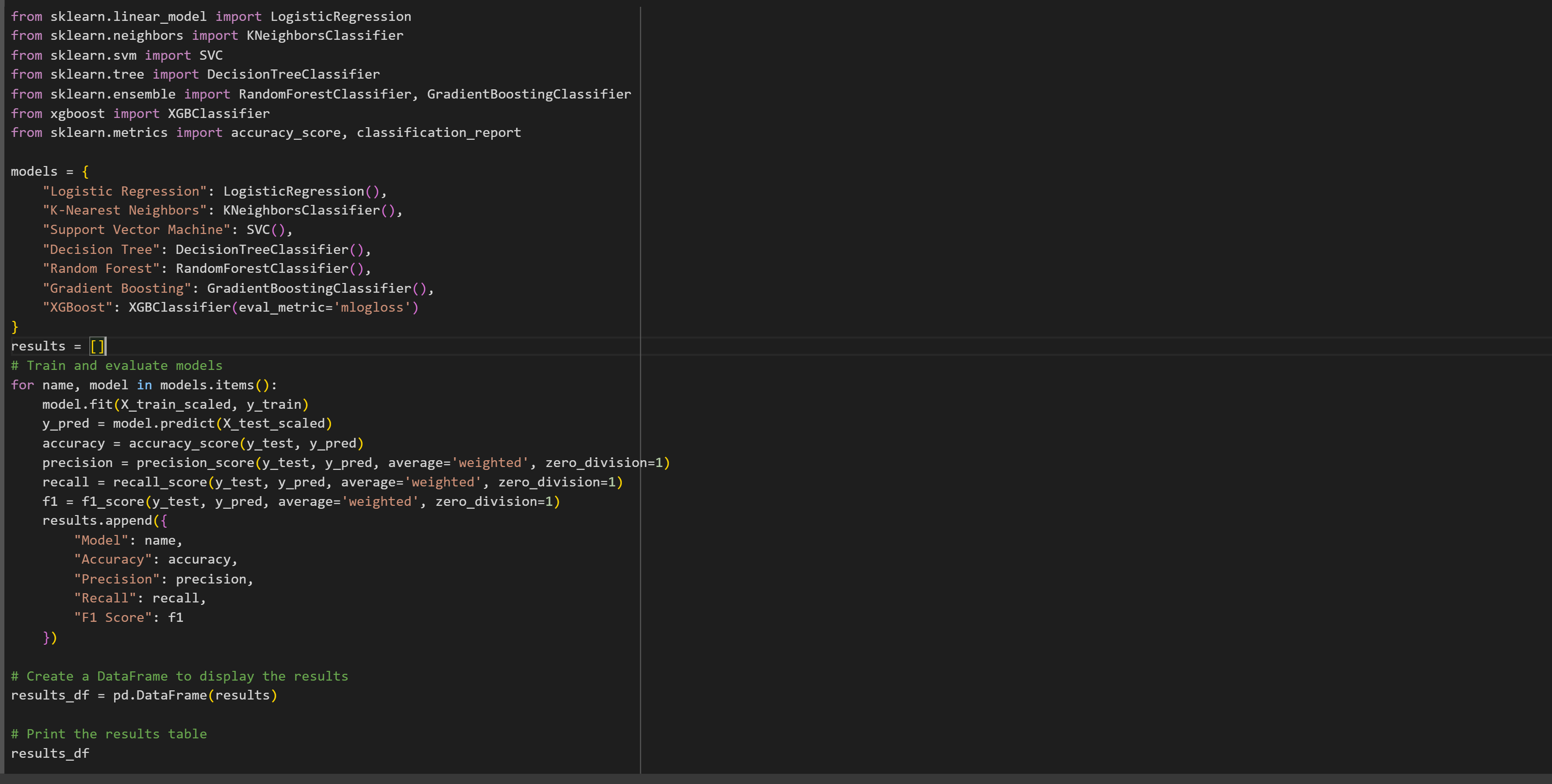
I performed feature importance analysis to identify which features have the most significant impact on my machine learning model's predictions. Understanding feature importance is crucial for several reasons. It helps in selecting the most relevant features, thereby reducing the model's complexity and improving its performance.



* **Machine learning**

I applied machine learning techniques to train the model and make predictions on the data. By training the model on a representative dataset, the algorithm learns the underlying patterns and relationships within the data, enabling it to make accurate predictions on new, unseen data. This process involves selecting an appropriate model, tuning its parameters, and validating its performance through metrics such as **accuracy, precision, recall, and F1-score.** The ultimate goal of applying machine learning is to develop a robust and reliable model that can generalize well to real-world data, providing valuable insights and aiding in decision-making processes.

I applied entire machine learning algorithms to train the data and testing.This allowed me to compare the performance of each models.



* **Comparing the results**

**Metric Analysis**

Accuracy measures the overall correct predictions. XGBoost and Random Forest excel in this.

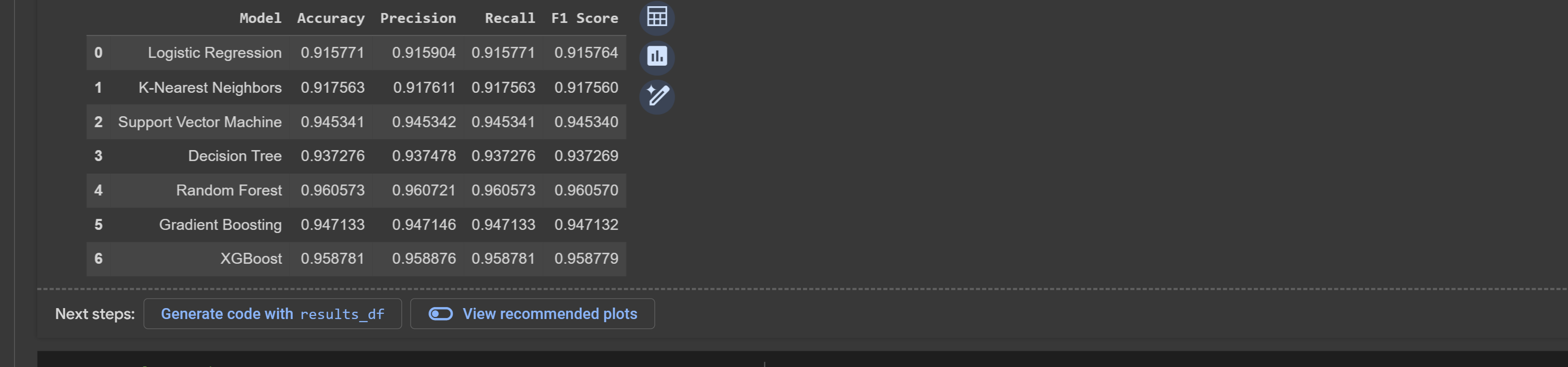
Precision focuses on the proportion of positive predictions that are truly positive. Again, XGBoost and Random Forest lead.

Recall measures the ability to find all positive cases. XGBoost and Random Forest demonstrate good recall.

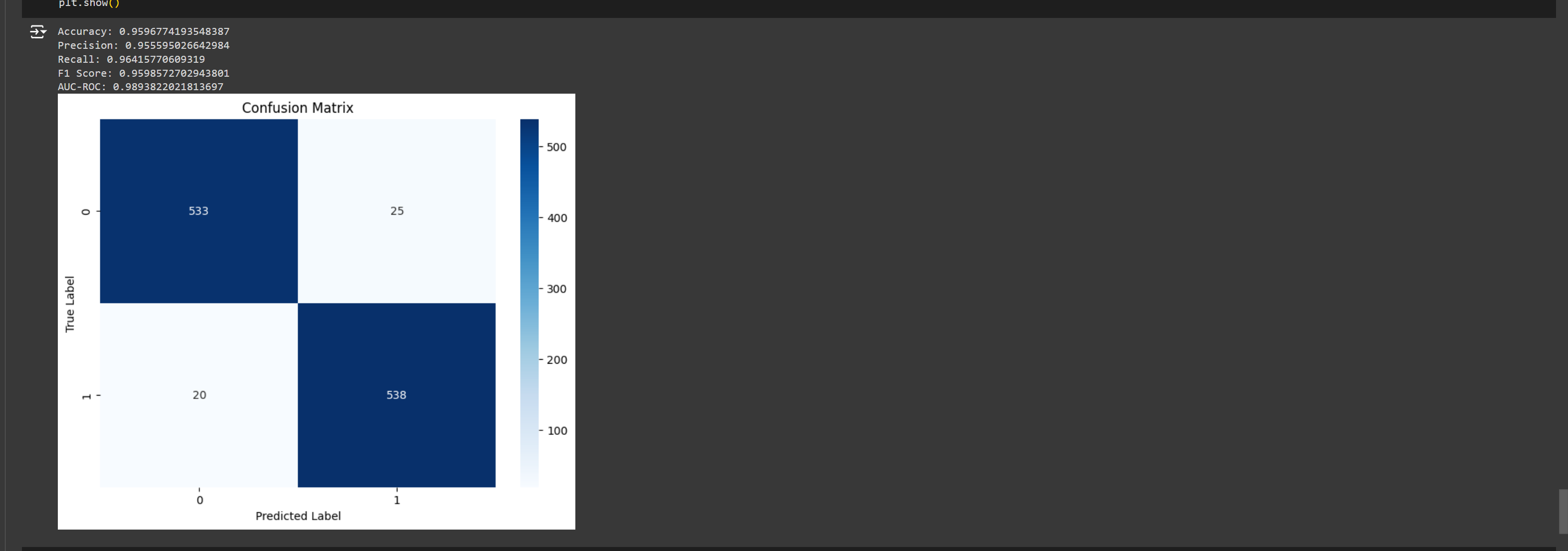
F1-Score balances precision and recall. XGBoost and Random Forest achieve the highest F1-Scores.

XGBoost seems to be the top-performing model based on Accuracy, Precision, Recall, and F1-Score.

Random Forest also shows strong performance across all metrics, closely following XGBoost.



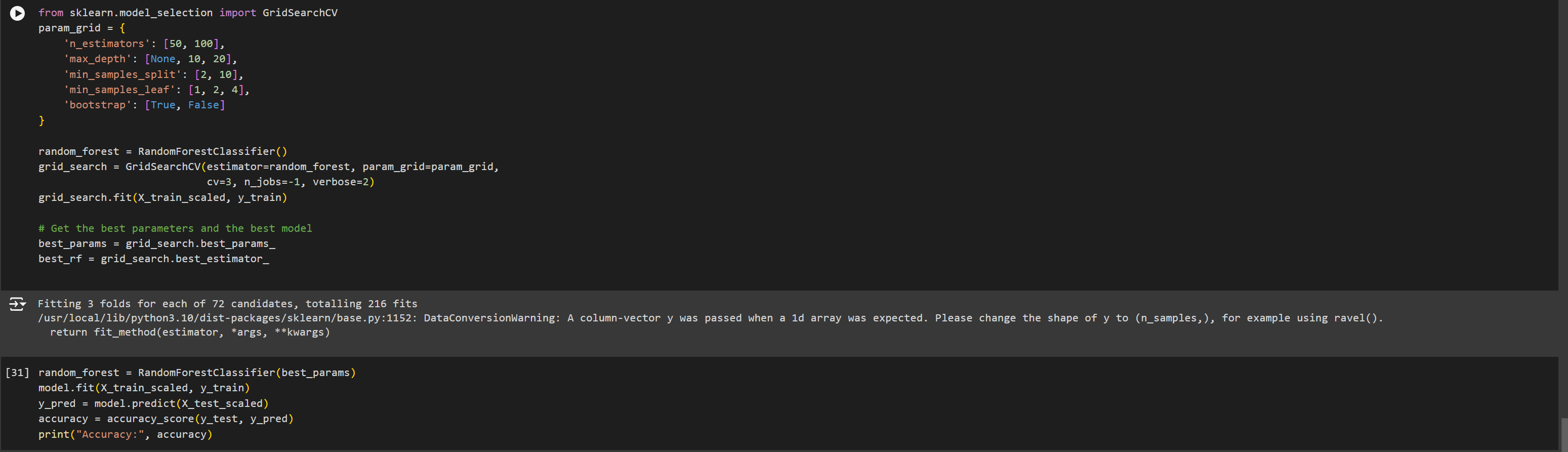
From the metrics XG-Boost and RandomForest are the best performing models in terms of accuracy and other metrices .



Since Random Forest performed well in terms of metrics this is the best model. I chose this model for the classifier .

* **Hyperparameter Tuning**

Hyperparameters are settings that influence how your model learns from data. They can significantly impact the performance of your model. By tuning these hyperparameters, you can find the combination that yields the best performance. This reduced the overfitting of the model.



* **Saving the model**

The resultant model is saved as pickle format for further use . Saving models in pickle format allows you to store them for long-term use. You can load the model whenever needed without worrying about the original training data or environment setup.

