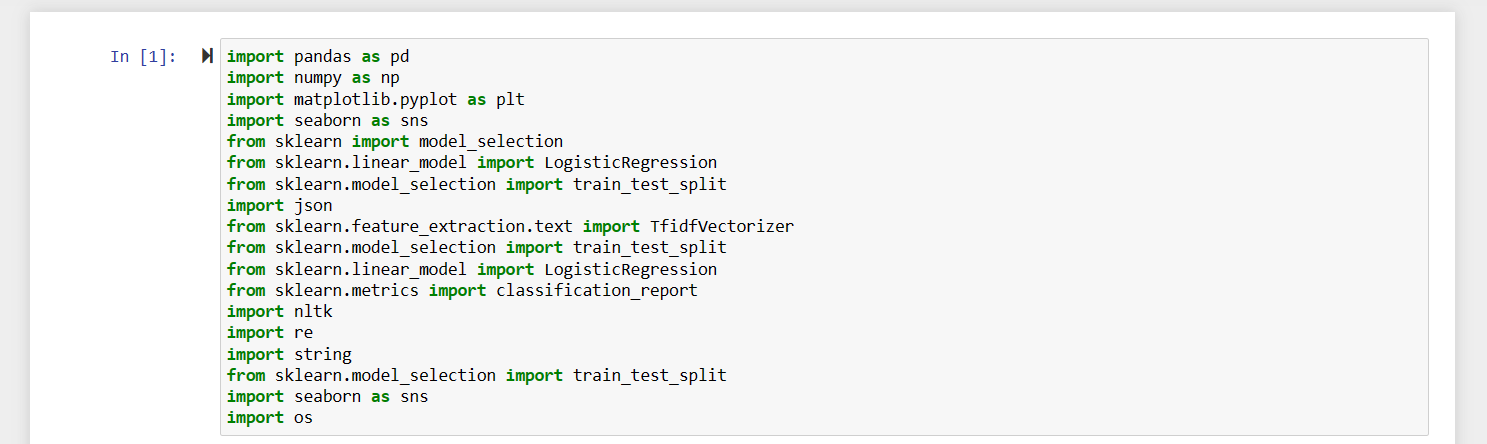
Model Documentation/Explanation

Import the required libraries and modules for the code as shown below.

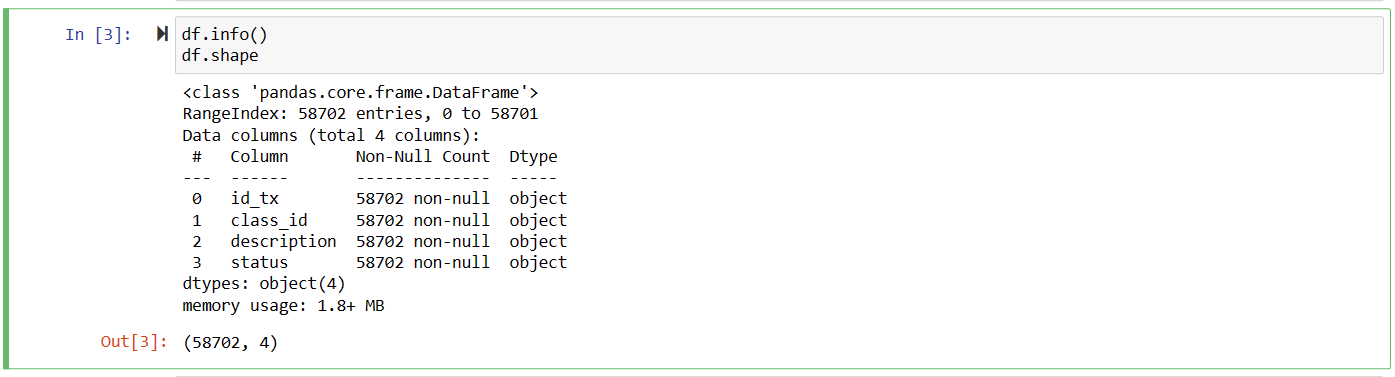


The following code snippet reads a JSON file named "idmanual.json" into a pandas DataFrame called df.

To display the first few rows of the DataFrame, we use the head() function without any parameter, like this:



The info() method in pandas DataFrame provides a concise summary of the DataFrame, including information about the columns, data types, and memory usage. The shape attribute of a DataFrame returns a tuple representing the dimensions of the DataFrame (number of rows, number of columns).

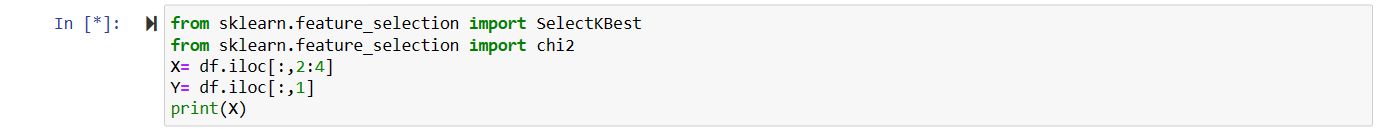


The describe() method in pandas DataFrame provides descriptive statistics for each numeric column in the DataFrame. It calculates various statistical measures such as count, mean, standard deviation, minimum, maximum, and quartiles as shown below.

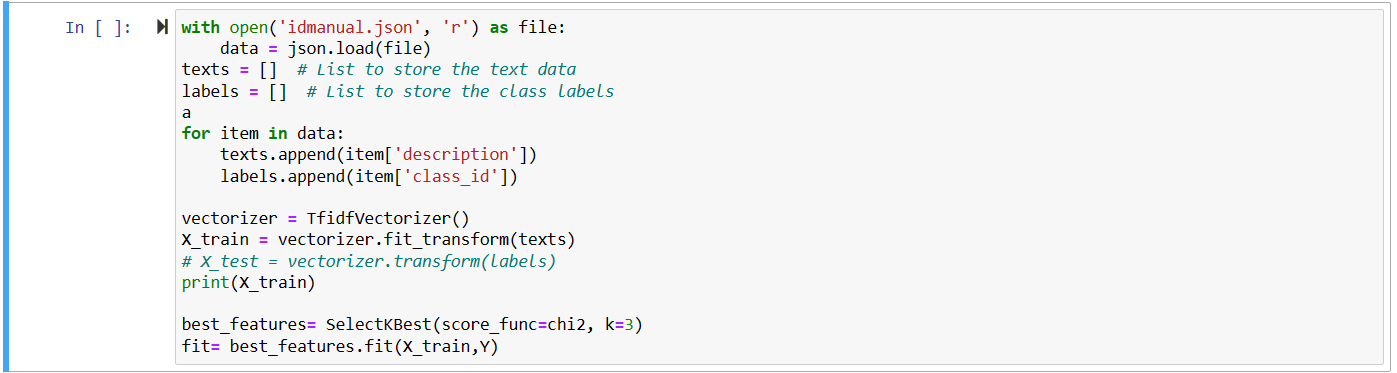


Use scikit-learn's SelectKBest and chi2 from the feature\_selection module to perform feature selection based on the chi-squared test.

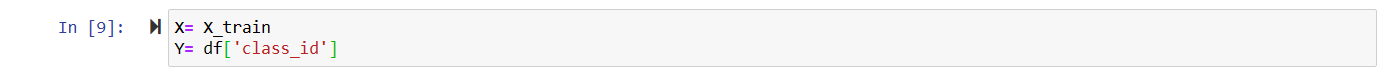
select the columns from the DataFrame df for X and Y variables, and then print the X variable.



Read the data from the "idmanual.json" file using the json.load() function and store the description texts in the texts list and the class labels in the labels list. Then use the TfidfVectorizer from scikit-learn to convert the text data into TF-IDF feature vectors.



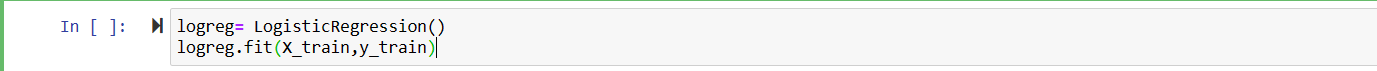
assign the transformed feature matrix X\_train to X and the 'class\_id' column of the DataFrame df to Y as shown below



Use the train\_test\_split function from scikit-learn to split the feature matrix X and the target variable Y into training and testing sets. It splits the data into a 60% training set (X\_train and y\_train) and a 40% testing set (X\_test and y\_test). The random\_state parameter is set to 100 to ensure reproducibility of the split.



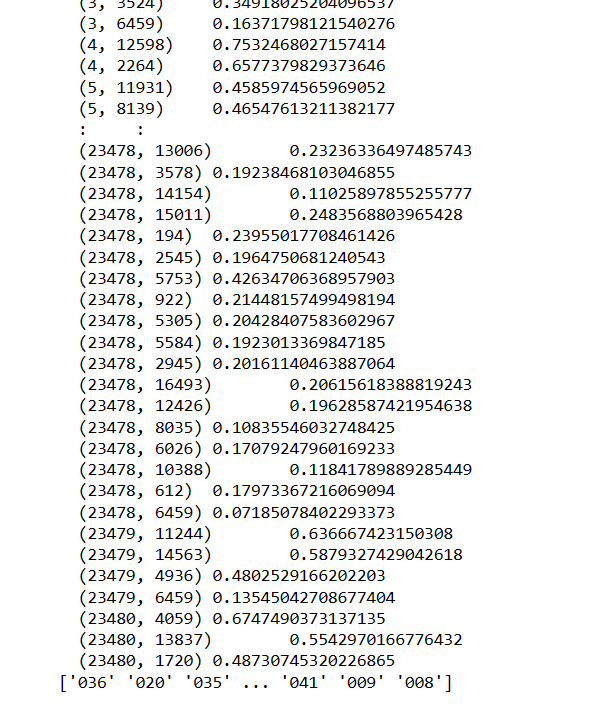
Use the Logistic Regression classifier from scikit-learn and fit the model on the training data. Logistic Regression model is created using LogisticRegression(), and the fit() method is called to train the model on the training data X\_train and y\_train. The model is now ready to make predictions on new data.



Predict the target variable y\_pred for the test dataset X\_test using the trained Logistic Regression model logreg. predict() method is used to predict the target variable for the test dataset X\_test using the trained Logistic Regression model logreg. The predicted labels are stored in the variable y\_pred.

Print the test dataset X\_test and the predicted labels y\_pred for inspection or further analysis.

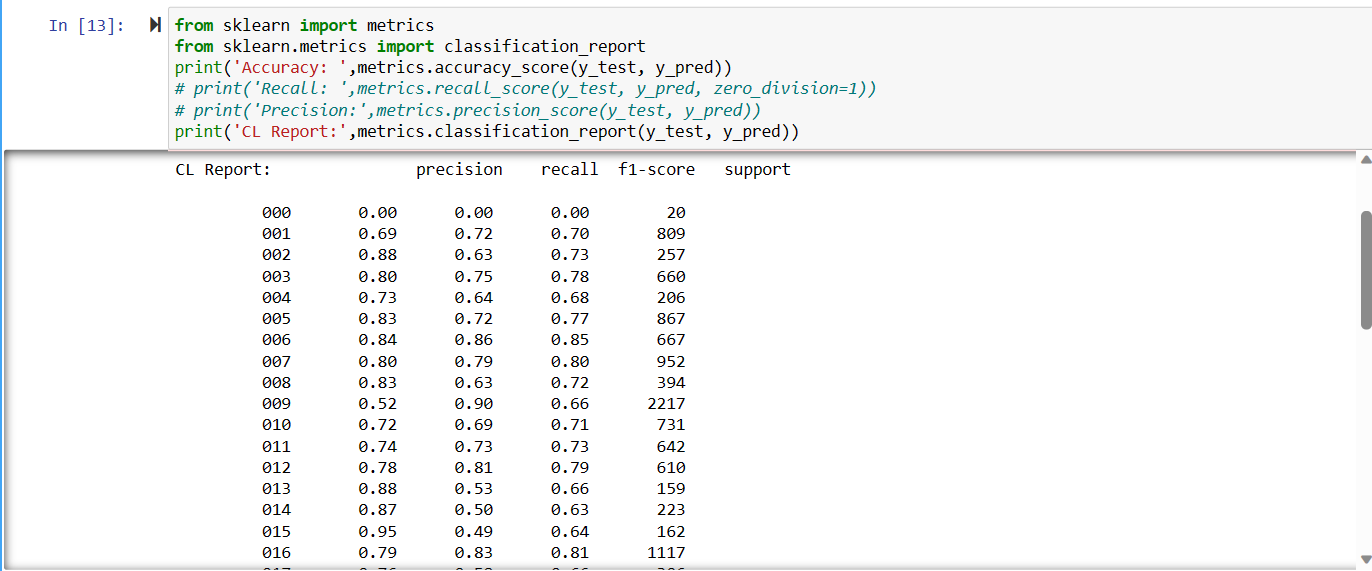


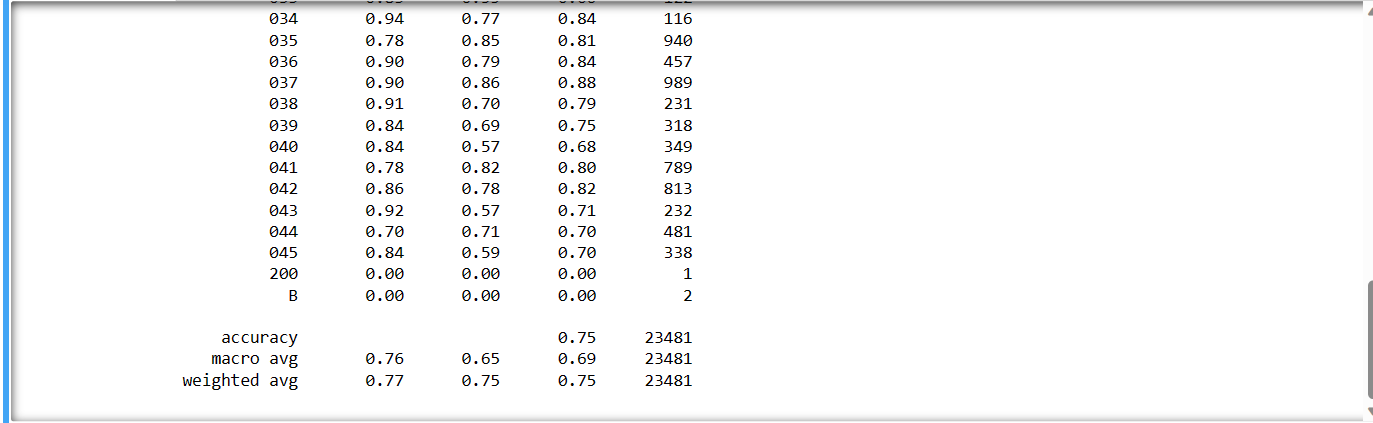


Calculate and print various evaluation metrics for the predicted labels y\_pred compared to the actual labels y\_test.

Accuracy: The accuracy of the predicted labels compared to the actual labels is calculated using metrics.accuracy\_score(). It represents the proportion of correctly predicted labels.

Classification Report: The classification\_report() function from scikit-learn's metrics module is used to generate a detailed classification report. It includes metrics such as precision, recall, F1-score, and support for each class.

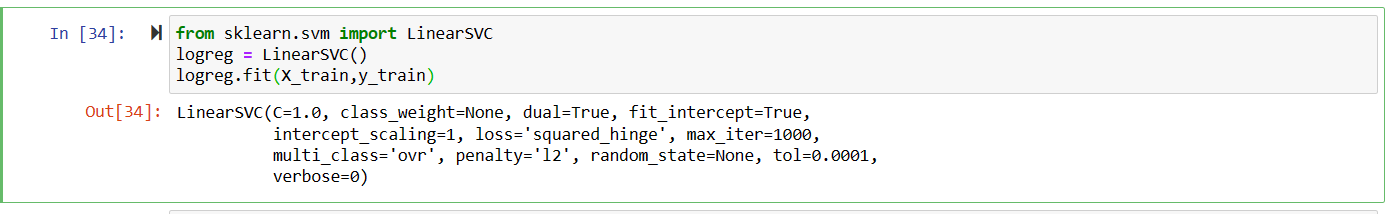


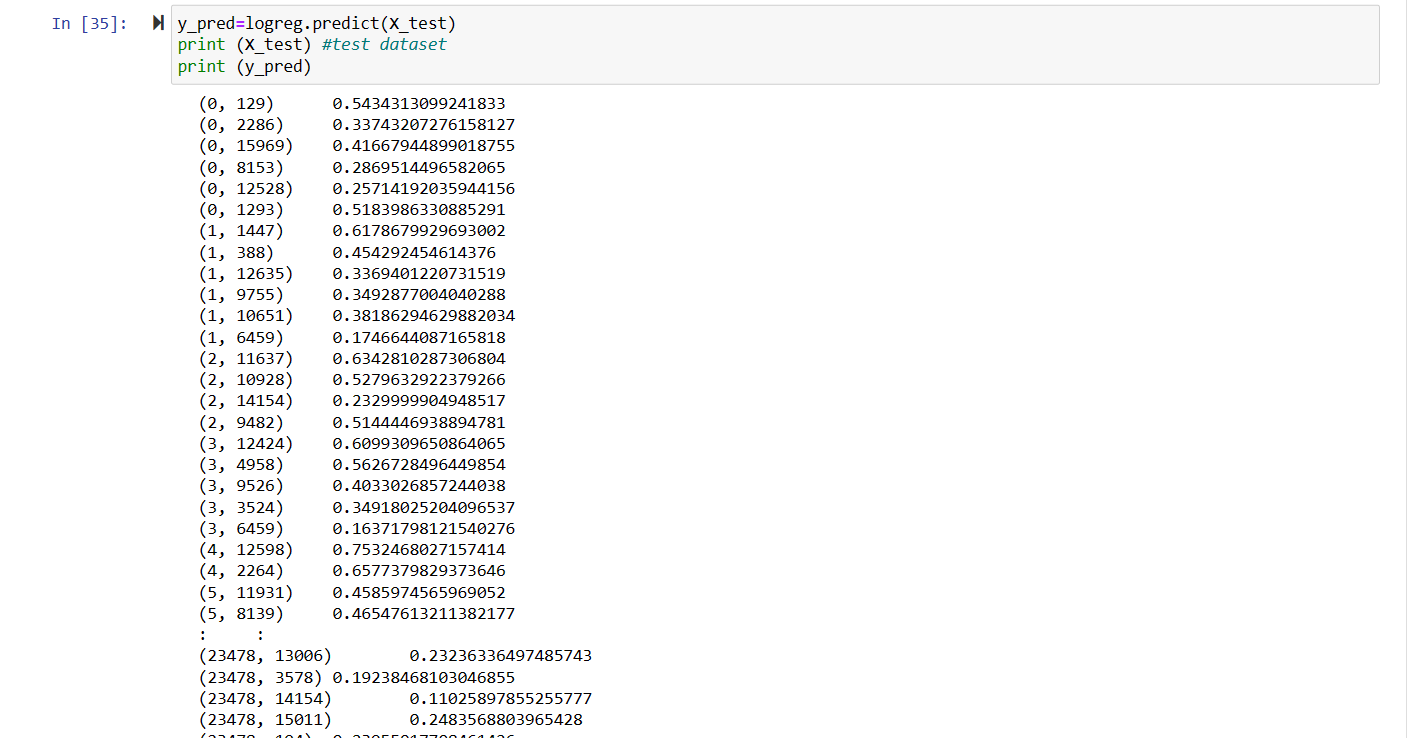


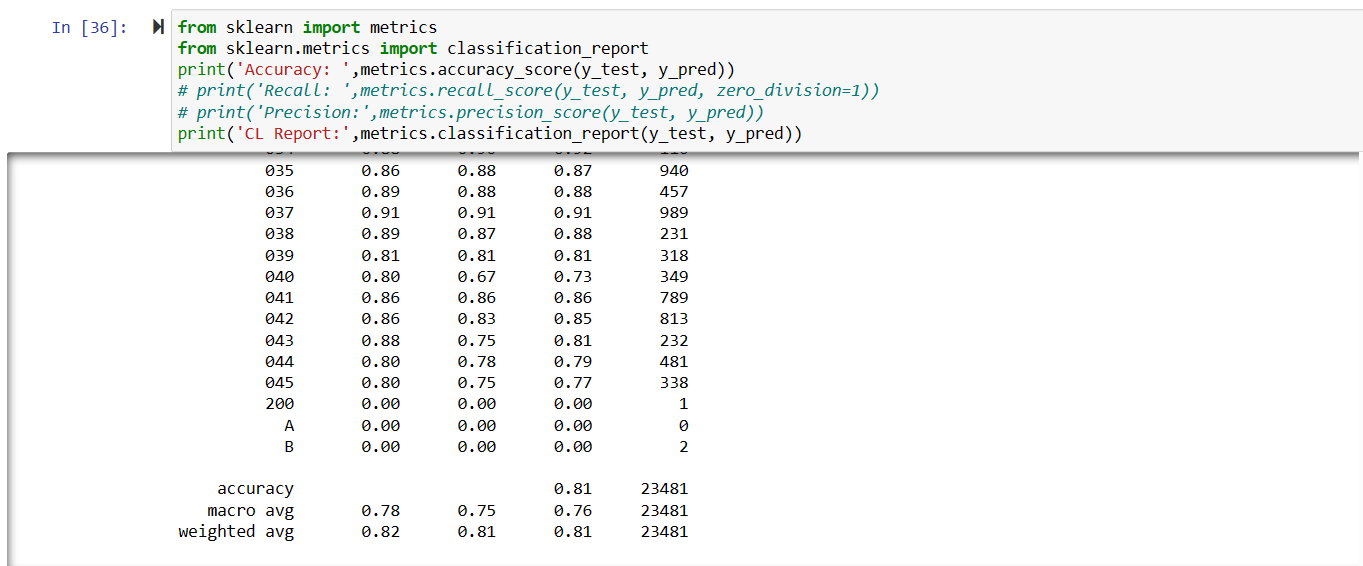
In order to increase accuracy,

Use the LinearSVC (Linear Support Vector Classification) classifier from scikit-learn and fit the model on the training data.

The LinearSVC model is created using LinearSVC(), and the fit() method is called to train the model on the training data X\_train and y\_train.







SVM gave an accuracy of 81% while logistic regression gave 75%.