

**LLM MODEL PERFORMANCE**

**COMPARISON ANALYSIS**

**Report Milestone 3**

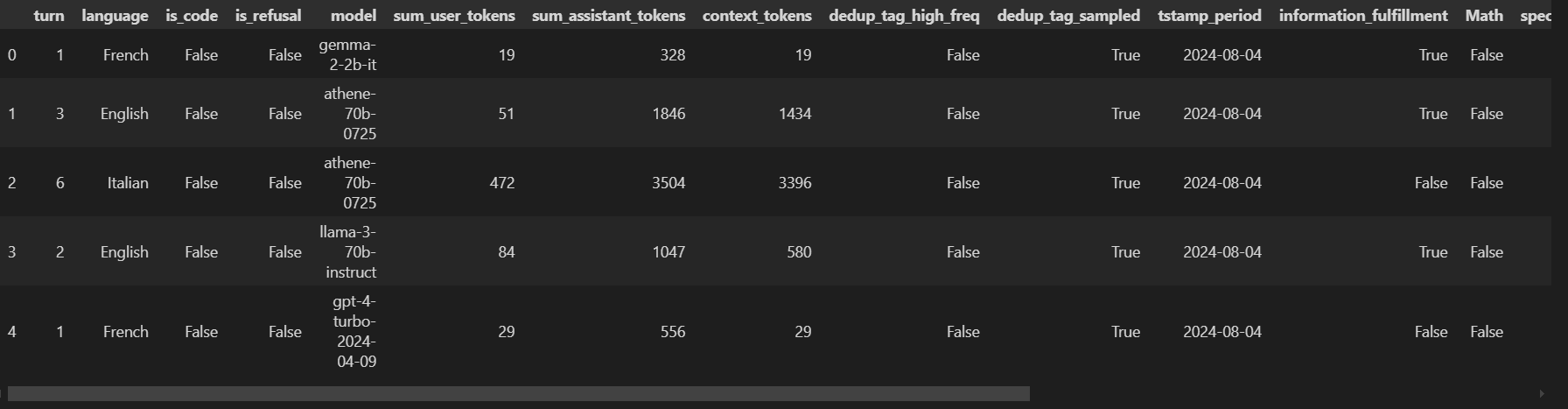
**Report by:**

**Akshara Sri Lakshmipathy -** [**akla8196@colorado.edu**](mailto:akla8196@colorado.edu)

**Harish Nandhan Shanmugam -** [**hash1366@colorado.edu**](mailto:hash1366@colorado.edu)

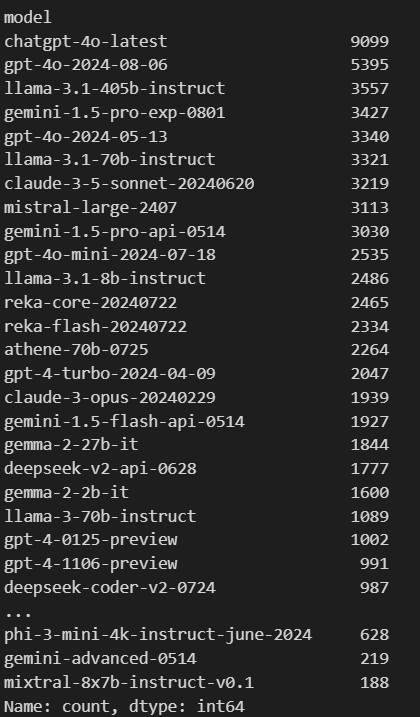
**Shivaraj Senthil Rajan -** [**shse1502@colorado.edu**](mailto:shse1502@colorado.edu)

In this we imported a CSV file (battles\_data\_cleaned.csv) into a DataFrame and then displayed the first few rows.



**HANDLING CLASS IMBALANCE**

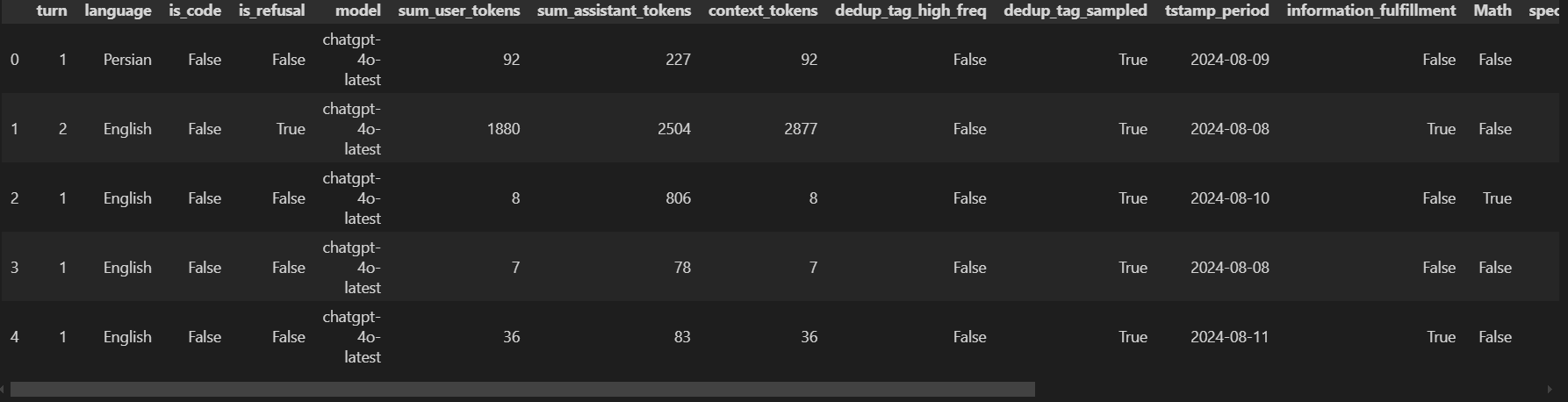
It calculates the frequency of models in the dataset, filter them and keep only with at least 1000 occurrences and create a new filtered DataFrame containing only these models.



It calculates the value counts of the 'model' column in the filtered data and prints the number of models.



To balance the dataset we can control the number of samples for each unique model. It filters models that have at least 1000 samples, and then resamples each model to ensure an equal representation of samples either by undersampling or oversampling. The code first saves the balanced dataset (balanced\_data) to a CSV file and printed the first few rows of the data are displayed to confirm the dataset's successful import in resampled\_data.head()

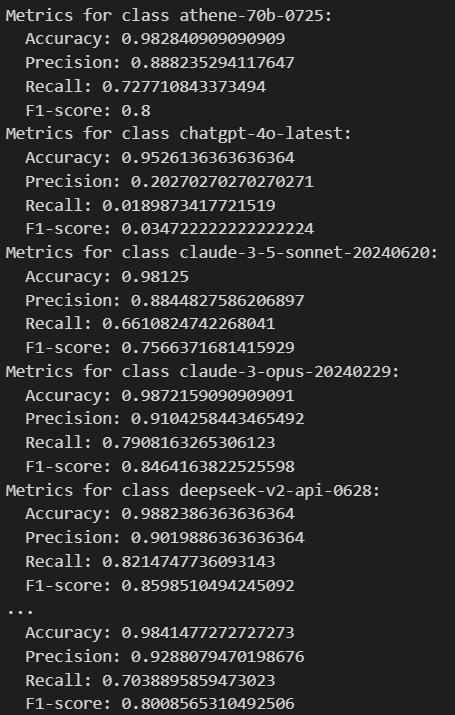


After this the code initializes and applies a LabelEncoder to the 'language' column from categorical text to numeric values. A LabelBinarizer is used to binarize the target labels, which helps transform the target variable into a format compatible with the multiclass classification. And then code splits the dataset into training and testing sets (80% training, 20% testing).

**MODEL IMPLEMENTATION**

**1. RANDOM FOREST**

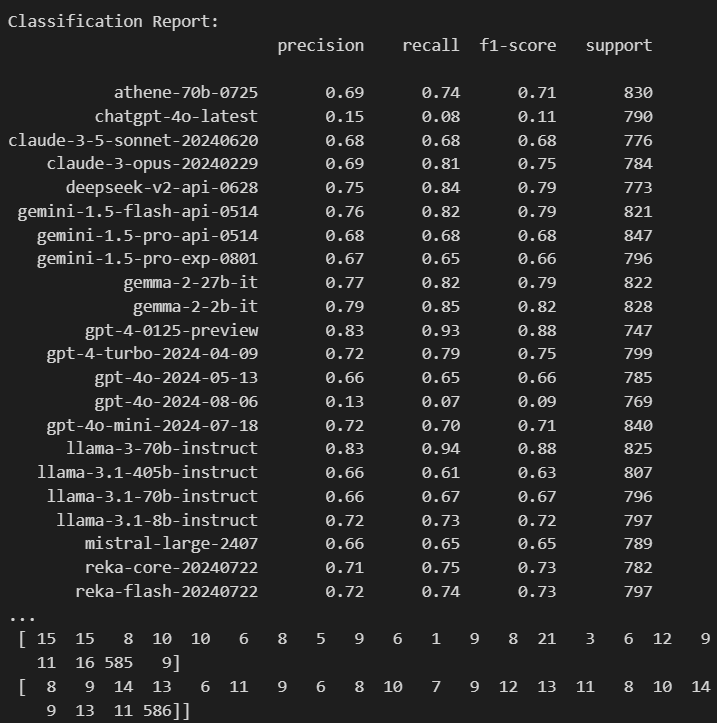
A **Random Forest** binary classifier is trained, resulting in multiple classifiers. After training, the classifiers are used to collect the probability scores of each instance in the test set for each class and the class with the highest probability is then selected for each test instance where it calculates and displays metrics—accuracy, precision, recall, and F1-score—for each of the binary classifiers on the test set



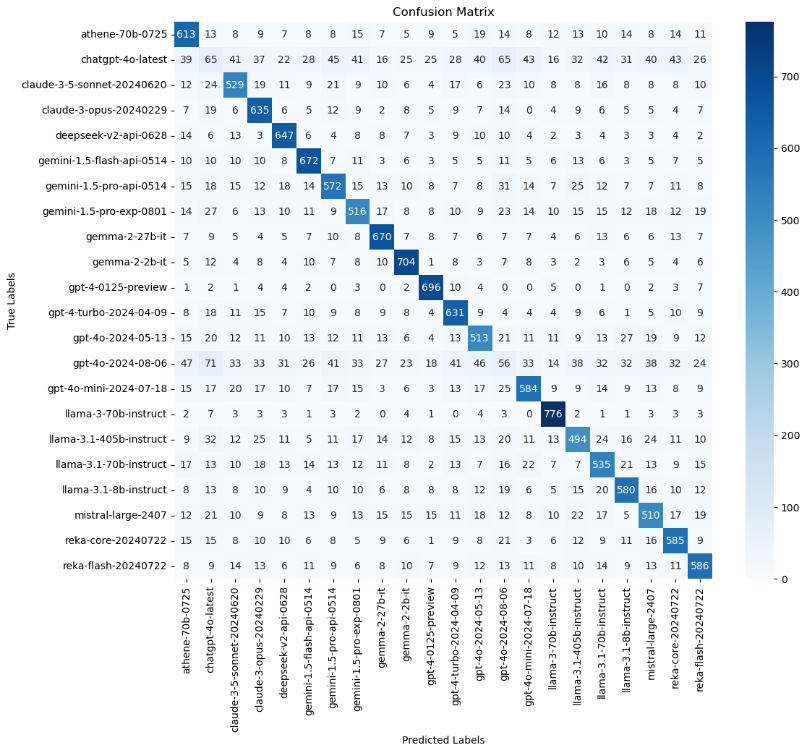
Printed the overall accuracy of the Random Forest classifier across all classes



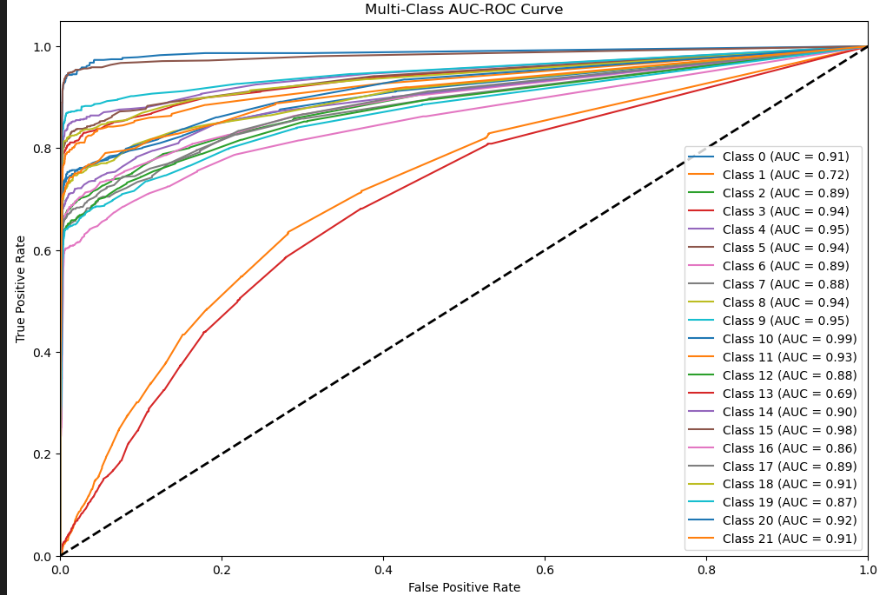
gpt-4-0125-preview and llama-3-70b-instruct showing high precision, recall, and F1-scores, while models like chatgpt-4o-latest underperformed. Most of the models have moderate performance, with scores ranging from 0.60 to 0.80.



The matrix provides a view of how well each model is performing to the others in predicting different classes.

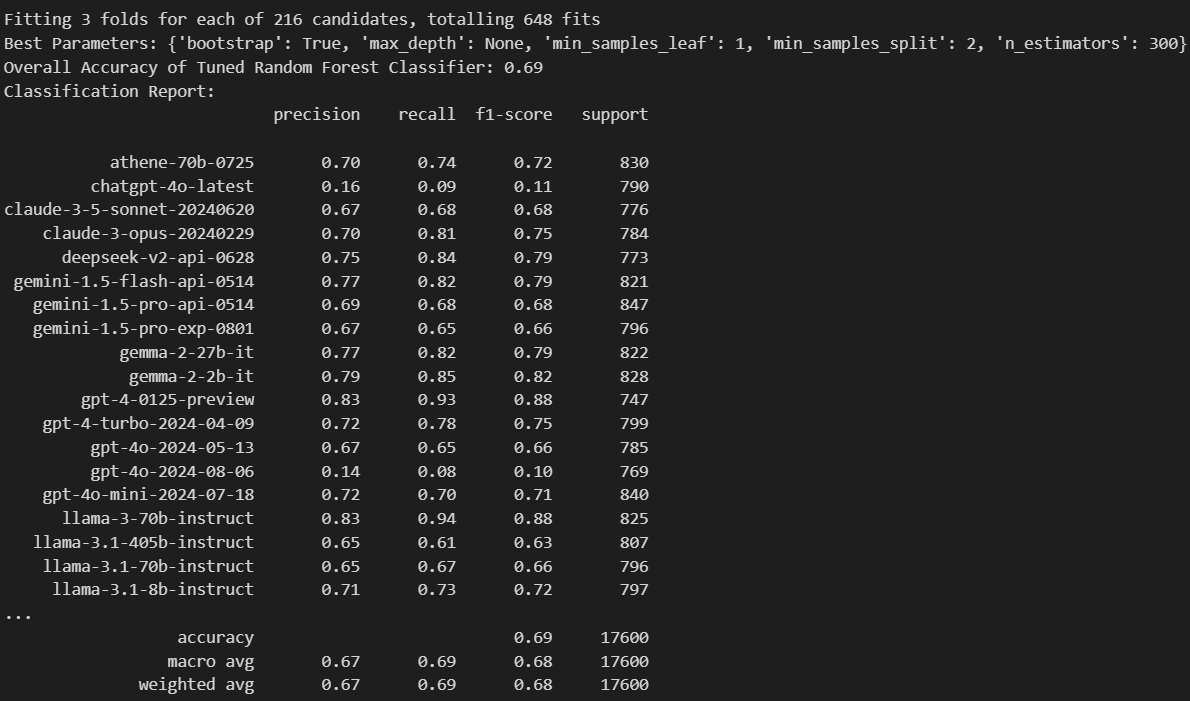


The area under each curve (AUC) is an indicator of how well the model can distinguish between classes, with values closer to 1 indicating better performance.

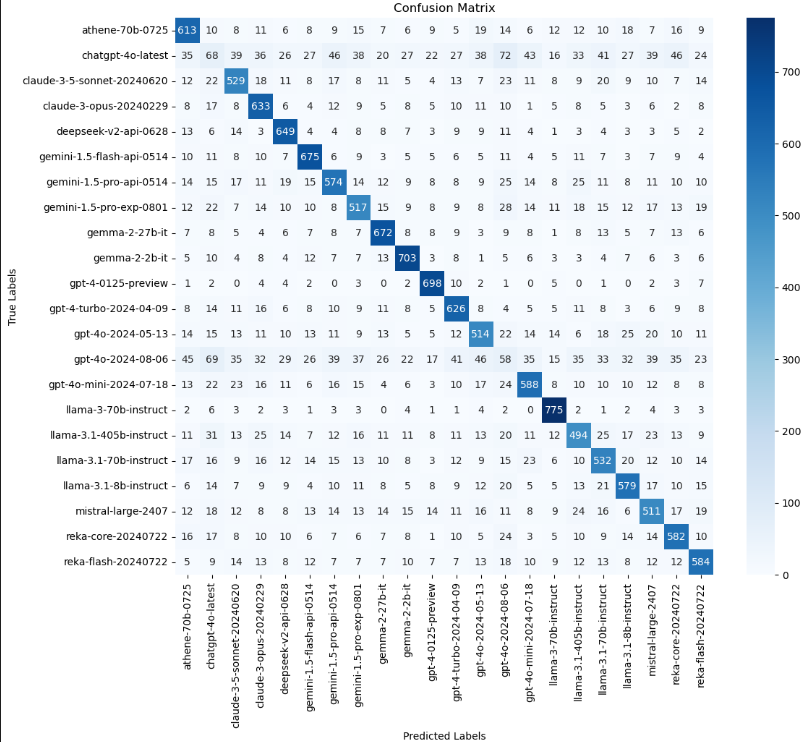


The Random Forest Classifier undergoes hyperparameter tuning using GridSearchCV to find the optimal parameters, which are used to train a One-vs-Rest classification model. After training, predictions are made by selecting the class with the highest probability for each instance. The evaluation involves calculating overall accuracy, classification report for precision, recall, and F1-score, and visualizing the confusion matrix. Additionally, AUC-ROC curves are plotted.

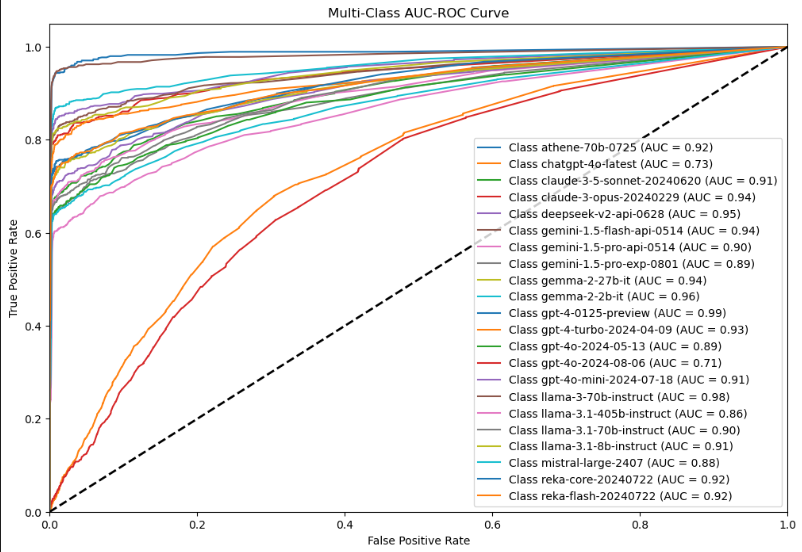
The Random Forest model achieved an overall accuracy of 0.69, with gpt-4-0125-preview and llama-3-70b-instruct performing F1-scores above 0.88. Models like chatgpt-4o-latest had low F1-scores around 0.11.



High values on the diagonal elements indicate a high number of correct predictions, while off-diagonal elements show the misclassifications. For example, model 'deepseek-v2-api-0628' has 649 correct predictions for one of its classes, while it has misclassified other classes a few times as shown by the non-diagonal elements.



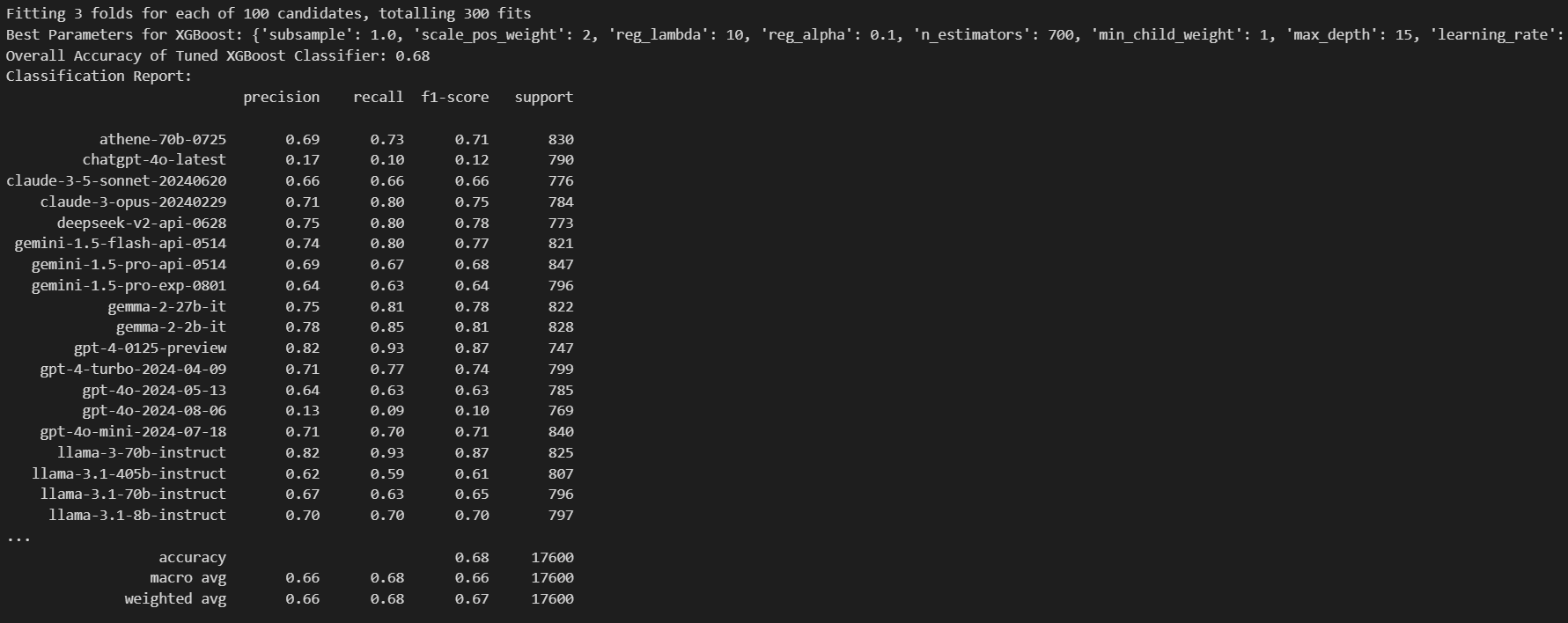
Higher AUC values, closer to 1, indicate better model performance.



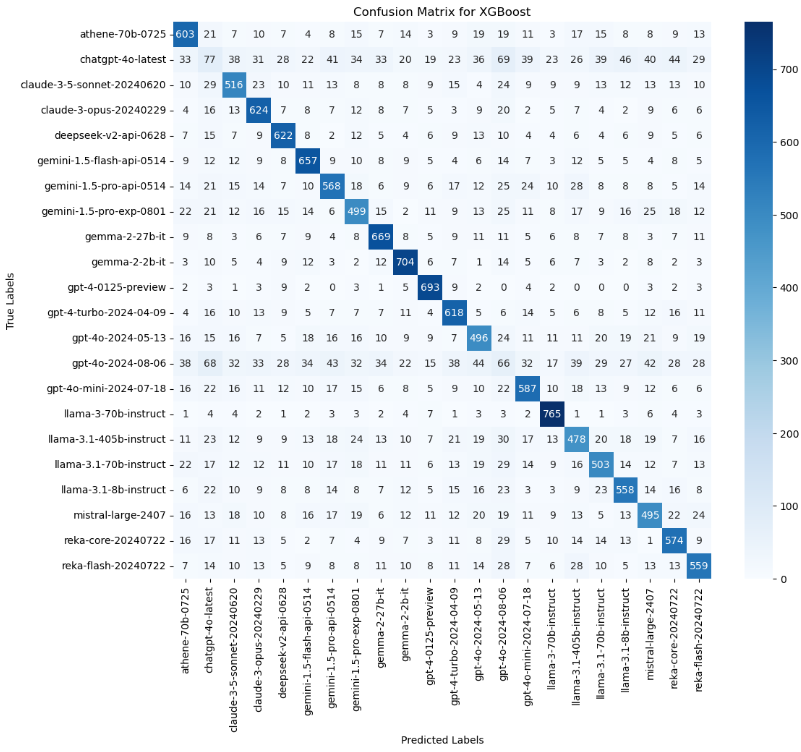
**2. XG BOOST**

The **XGBoost** classifier achieved an overall accuracy of 0.68.

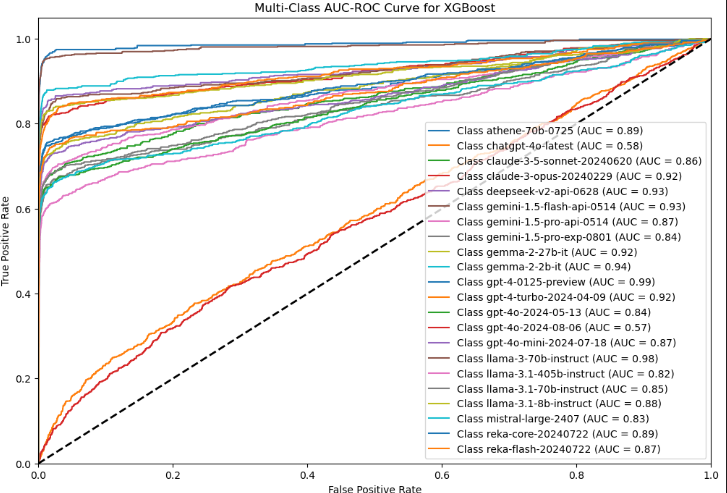
We have used XGBoost for a multi-class classification using a One-vs-Rest approach and involves the following steps: Data Loading and Preprocessing, Hyperparameter Tuning using RandomizedSearchCV to tune hyperparameters and selecting the best parameters. Then it trains separate One-vs-Rest XGBoost classifiers for each class. Making predictions which involves calculating overall accuracy, classification report for precision, recall, and F1-score, and visualizing the confusion matrix. Additionally, AUC-ROC curves are plotted.



The diagonal values, which show the highest numbers (e.g., 603 for the first class), indicate correct predictions.



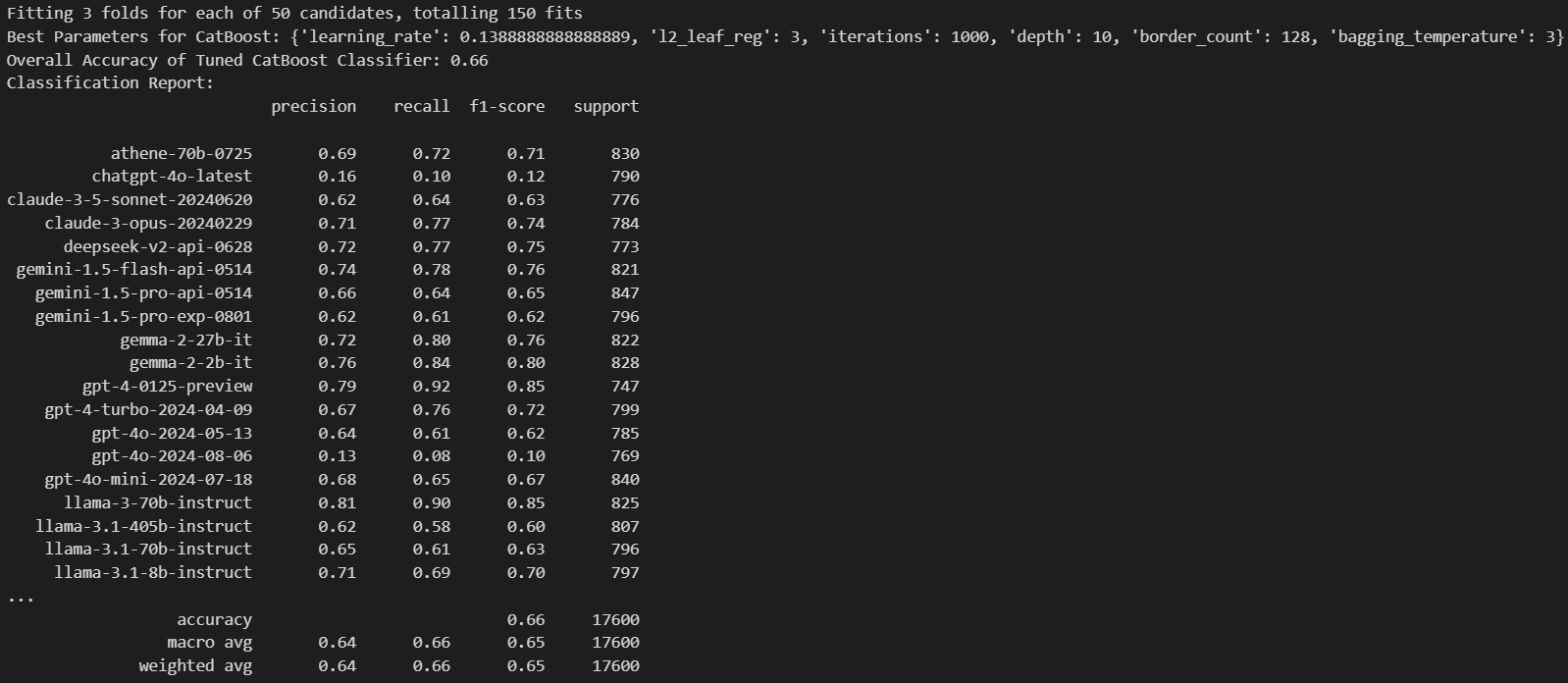
The curves closer to the top-left corner of the plot indicate better performance, with AUC values closer to 1.0

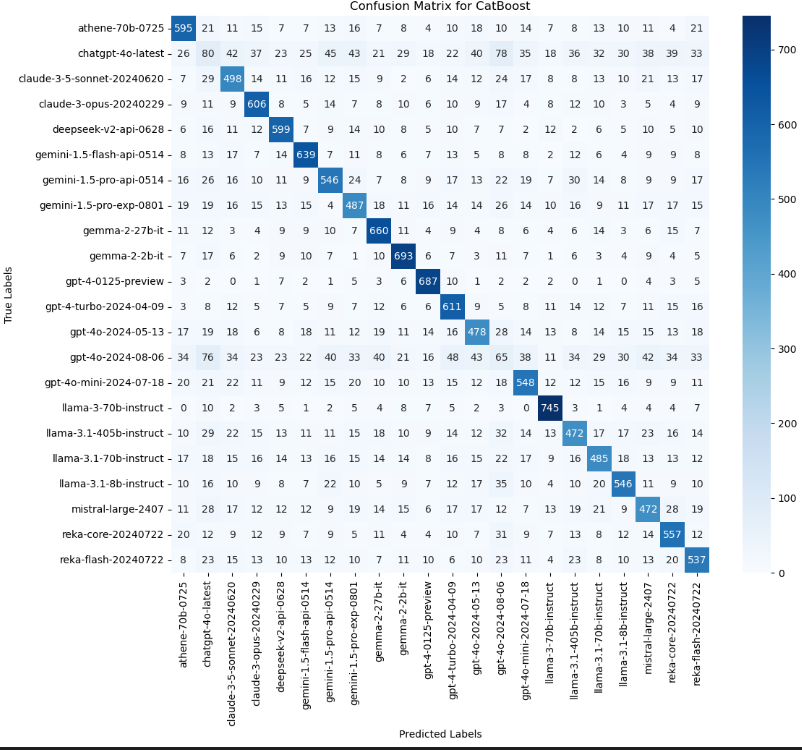


**3. CATBOOST**

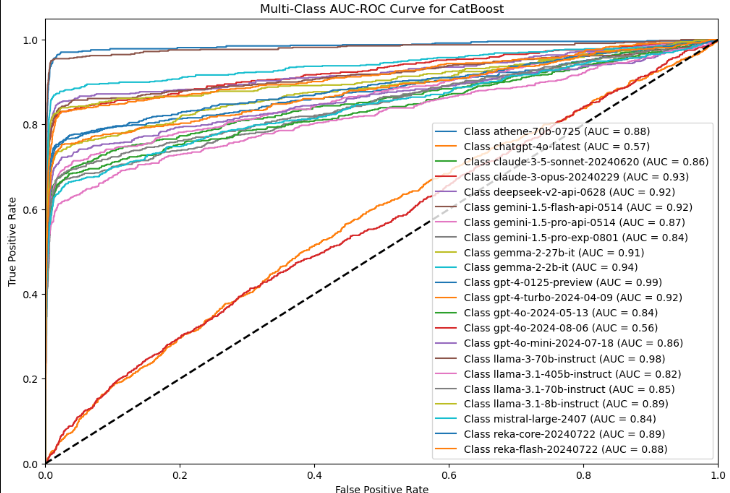
The **CatBoost** classifier achieved an overall accuracy of 0.66.

Here we have used CatBoost and involve the following steps: Data Loading and Preprocessing, Hyperparameter Tuning using RandomizedSearchCV to tune hyperparameters and selecting the best parameters. Then it trains separate CatBoost classifiers One-vs-Rest approach. Making predictions which involves calculating overall accuracy, classification report for precision, recall, and F1-score, and visualizing the confusion matrix. Additionally, AUC-ROC curves are plotted.

****

****

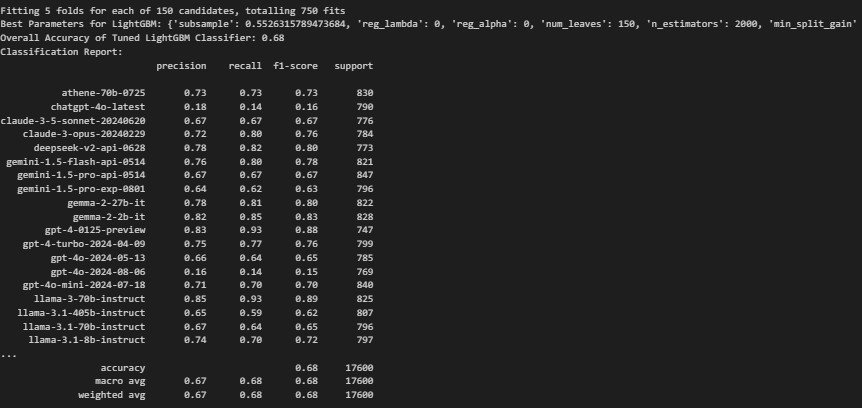
The Area Under the Curve (AUC) where values close to 1 indicate high ability and values closer to 0.5 suggest no better performance.

****

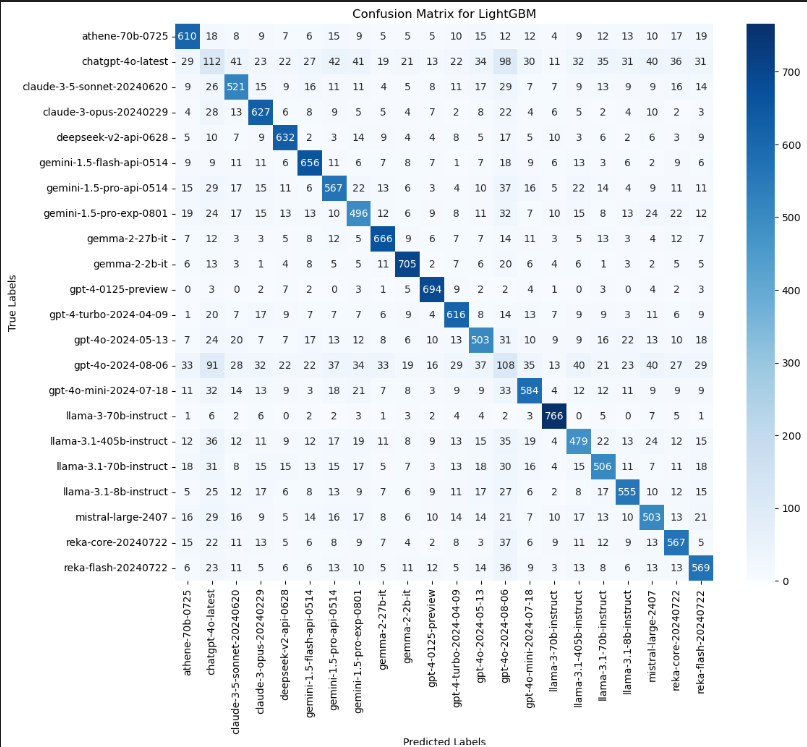
**4. LIGHTGBM**

The **LightGBM** classifier achieved an overall accuracy of 0.68.

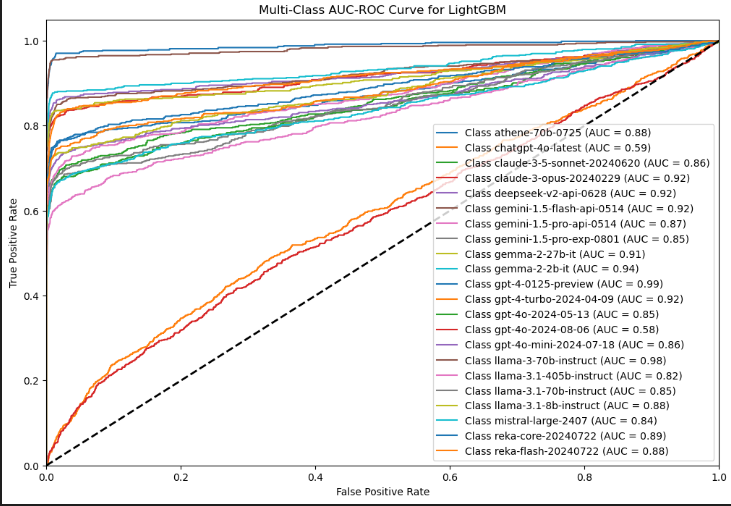
The code usesLightGBM for a multi-class classification through a One-vs-Rest approach and involves the following steps: Data Loading and Preprocessing, Hyperparameter Tuning using RandomizedSearchCV with 150 iterations and 5-fold cross-validation to tune hyperparameters and selecting the best parameters. Then using the One-vs-Rest method, separate LightGBM classifiers and train it. Evaluating the results with confusion matrices, classification reports, and AUC-ROC curves.



High diagonal values suggest that the classifier performs well in predicting those classes, whereas high off-diagonal values indicate areas where the classifier confuses one class for another

****

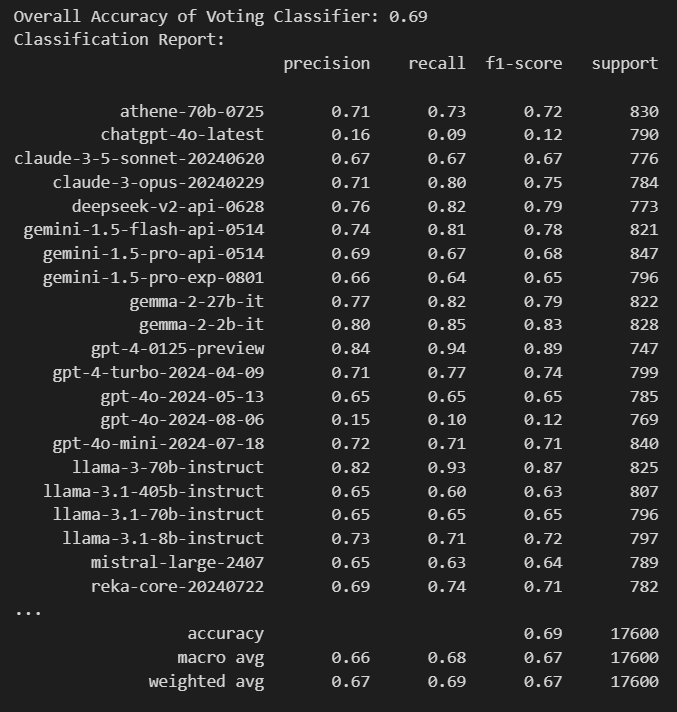
Values close to 1 indicating excellent ability and values near 0.5 suggesting no better accuracy.

****

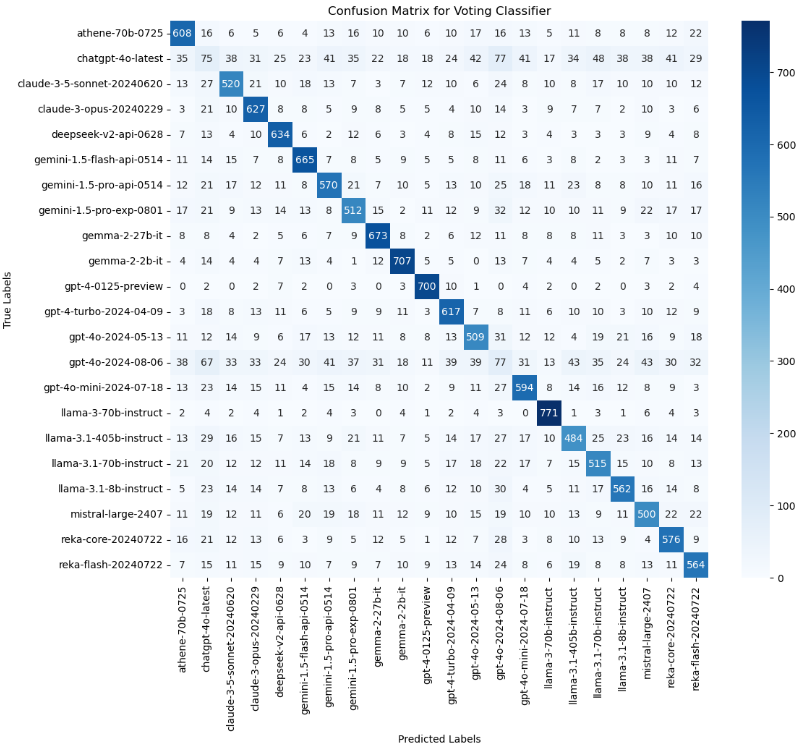
**5. VOTING**

The **Voting** classifier achieved an overall accuracy of 0.69.

We handled a multi-class classification by utilizing a **Voting Classifier** by combining LightGBM, CatBoost, and XGBoost through a One-vs-Rest (OvR) strategy. and involves the following steps: Data Loading and Preprocessing, Model Initialization: where it initializes three models LightGBM, CatBoost, and XGBoost with hyperparameters which constructs a Voting Classifier using soft voting from all three models. Then Train the One-vs-Rest Model with multiple Voting Classifiers in a One-vs-Rest. Evaluating the results with confusion matrices and classification reports.

****

The numbers on the matrix diagonal (e.g., 608 for the first class) indicate correct predictions for each class, while off-diagonal numbers represent misclassifications.

****

**ACCURACY ACHIEVED BY EACH MODEL**

| **MODEL NAME** | **ACCURACY** |
| --- | --- |
| RANDOM FOREST | 0.69 |
| XG BOOST | 0.68 |
| CATBOOST | 0.66 |
| LIGHTGBM | 0.68 |
| VOTING | 0.69 |