Random Forest Classifier: Model Summary

Why Random Forest?

Random Forest is an **ensemble learning method** that combines multiple decision trees to improve classification accuracy and reduce overfitting. It works especially well in high-dimensional spaces like text classification (e.g., TF-IDF-based vectorized text).

We chose Random Forest because:

- It's **robust to noise and overfitting** due to averaging of multiple trees.
- It handles **multiclass problems** (like review ratings: 1–5) efficiently.
- It can model **nonlinear relationships** between text features and ratings.
- It supports **class balancing** via `class_weight='balanced'`, which is helpful when data is imbalanced.

Model Training Logic

We trained the Random Forest using `GridSearchCV` to **tune hyperparameters** and find the best configuration.

Training pipeline:

1. Preprocessed the review text (punctuation removal, lowercase, stopword removal, lemmatization).
2. Converted text to numerical features using **TF-IDF vectorization**.
3. Split the dataset into train/test using **stratified sampling** to preserve class balance.
4. Applied `GridSearchCV` with 3-fold cross-validation to search the best combination of:
    - `n_estimators`: number of trees
    - `max_depth`: how deep each tree grows
    - `min_samples_split` and `min_samples_leaf`: tree pruning controls
    - `max_features`: how many features are considered when splitting
    - `class_weight`: automatic balancing of class weights

```
grid = GridSearchCV(rf, param_grid, cv=3, scoring='f1_macro', verbose=2, n_jobs=-1)
grid.fit(X_train_tfidf, y_train)
```

📊 Model Evaluation and Results

After training the model with the best parameters, we evaluated it using the test set:

```
best_model = grid.best_estimator_
y_pred = best_model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.48      0.58      0.53       400
           2       0.35      0.32      0.33       400
           3       0.36      0.30      0.33       400
           4       0.39      0.37      0.38       400
           5       0.52      0.56      0.54       400

    accuracy                           0.43      2000
   macro avg       0.42      0.43      0.42      2000
weighted avg       0.42      0.43      0.42      2000
```

Interpretation of Results

- **Accuracy: ~43%**, better than random guessing (which would be 20% in a 5-class problem).
- Best performance was seen on **1-star and 5-star ratings**, likely because those reviews contain more polarized language.
- **Lower precision/recall on middle ratings (2, 3, 4)**, which is common in subjective review analysis.

Summary

Feature	Value
Algorithm	Random Forest
Vectorization	TF-IDF
Hyperparameter Tuning	Yes (GridSearchCV)
Class Handling	Stratification + `class_weight=balanced`
Accuracy	~43%
Strengths	Robust, easy to interpret, good for tabular & text data
Limitations	Slower for large grid searches, limited performance on fuzzy boundaries (e.g., rating 3 vs 4)