AUTOMATED_RATING_REVIEW_SYSTEM

Model Documentation

NAÏVE BAYES

1. Model Used

The model used throughout this project is the Multinomial Naive Bayes (MNB) classifier. It is a probabilistic machine learning model based on Bayes' Theorem and is particularly well-suited for text classification tasks where features represent word counts or TF-IDF scores.

2. Why Multinomial Naive Bayes?

- Efficient for Text Data: Works very well with high-dimensional sparse features produced by TF-IDFvectorization.
- Computationally Fast: Requires less training time compared to models like SVM or RandomForest.
- Good Baseline Model: Provides reliable performance for sentiment and rating prediction tasks.- Handles Imbalanced Data: Performs reasonably well even when some star ratings are less frequent.

3. Working Principle

Based on Bayes' Theorem:
P(Class|Words) = [P(Words|Class) * P(Class)] / P(Words)

It assumes that the features (words) are independent of each other given the class. For each review, the model calculates the probability of it belonging to each rating class (1★ to 5★) and assigns the class with the highest probability.

4. Why Not Other Models?

- Random Forest / Gradient Boosting: Very powerful for structured/numeric datasets but notefficient for sparse text data.
- Support Vector Machine (SVM): Can achieve high accuracy but computationally expensive forlarge datasets and requires tuning.

5. Strengths of Naive Bayes in This Project

- Fast to train and predict.
- Works effectively with TF-IDF features.
- Requires minimal parameter tuning.
- Provides interpretable results (class probabilities).

6. Limitations

- Independence assumption (words treated as unrelated) is not always realistic.
- May struggle when features are highly correlated.
- Usually outperformed by deep learning models for very complex tasks.

7. Conclusion

In this project, Multinomial Naive Bayes was chosen because it is simple yet powerful for text-based problems, scalable for large datasets, efficient in computation, and well-suited for review rating prediction tasks. Thus, all model training, evaluation, and inference steps were carried out using Naive Bayes as the core model.

Codes :

1. Balance the dataset

```
# Balance the dataset to have equal number of reviews per rating
balanced_df = balance_dataset(review, target_col="Rating", n_samples=5000)
```

2. Train-Test Split & TF-IDF Vectorization

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

X = balanced_df['Review_text']
y = balanced_df['Rating']

# Stratified train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2), stop_words='english')
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

3. Train Naïve Bayes Model

```
from sklearn.naive_bayes import MultinomialNB

# Initialize and train the model
nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)
```

4. Evaluate the Model (Optional for Documentation)

```
from sklearn.metrics import accuracy_score, classification_report

# Predict on test set
y_pred = nb_model.predict(X_test_tfidf)

# Accuracy & classification report
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```