.

**Movie Genre Prediction using Word Embedding and Naive Bayes**

**Objective**

To classify movies into their respective genres using the plot summaries by applying a **Bag-of-Words (CountVectorizer) + Naive Bayes** pipeline with oversampling to handle class imbalance.

**Dataset Description**

We used three text files:

* train\_data.txt: Contains labeled training data (id, title, genre, plot)
* test\_data.txt: Contains unlabeled test data (id, title, plot)
* test\_data\_solution.txt: Contains true genres for test data (used only for evaluation)

**Implementation Steps**

**1. Data Loading**

python

CopyEdit

train\_df = pd.read\_csv('train\_data.txt', sep=' ::: ', engine='python', names=['id', 'title', 'genre', 'plot'])

test\_df = pd.read\_csv('test\_data.txt', sep=' ::: ', engine='python', names=['id', 'title', 'plot'])

test\_solutions = pd.read\_csv('test\_data\_solution.txt', sep=' ::: ', engine='python', names=['id', 'title', 'genre', 'plot'])

We read all three datasets using custom separators (:::).

**2. Text Cleaning**

All text is converted to lowercase and non-alphabetic characters are removed:

python

CopyEdit

def clean\_text(text):

text = str(text).lower()

text = re.sub(r'[^a-z\s]', '', text)

return text

This ensures consistency in word representation before vectorization.

**3. Feature Extraction – Bag-of-Words**

We applied the CountVectorizer to convert plots into numerical vectors.

python

CopyEdit

vectorizer = CountVectorizer(max\_features=5000)

X\_train = vectorizer.fit\_transform(train\_df['clean\_plot'])

* This generates a sparse matrix of token counts.
* We limited the vocabulary to the top 5000 words.

**4. Handling Class Imbalance**

We used RandomOverSampler to balance the class distribution:

python

CopyEdit

ros = RandomOverSampler()

X\_train\_resampled, y\_train\_resampled = ros.fit\_resample(X\_train, y\_train)

This replicates minority class samples to prevent model bias toward dominant genres.

**5. Model Training – Multinomial Naive Bayes**

python

CopyEdit

nb\_model = MultinomialNB()

nb\_model.fit(X\_train\_resampled, y\_train\_resampled)

The Naive Bayes classifier is trained using the resampled data.

**6. Testing & Evaluation**

* The test data is cleaned and merged with true labels.
* Predictions are made and evaluated using classification\_report.

**Model Performance**

**Accuracy:** 0.49  
**Macro Average F1 Score:** 0.33  
**Weighted Average F1 Score:** 0.50

**Sample Classification Report:**

markdown

CopyEdit

precision recall f1-score support

action 0.30 0.47 0.36 1314

adult 0.37 0.48 0.42 590

adventure 0.19 0.21 0.20 775

animation 0.22 0.23 0.23 498

...

western 0.70 0.85 0.76 1032

accuracy 0.49 54200

macro avg 0.32 0.39 0.33 54200

weighted avg 0.54 0.49 0.50 54200

**Insights:**

* High precision and recall for frequent genres like **documentary**, **drama**, and **comedy**.
* Reasonable recall for **western**, **music**, and **reality-tv**.
* Low performance on underrepresented genres like **biography**, **history**, and **news**.

**Original Training Genre Distribution**

| **Genre** | **Count** |
| --- | --- |
| drama | 13613 |
| documentary | 13096 |
| comedy | 7447 |
| short | 5073 |
| horror | 2204 |
| thriller | 1591 |
| action | 1315 |
| western | 1032 |
| reality-tv | 884 |
| family | 784 |
| ... | ... |

This imbalance justifies the use of oversampling for fairness across genres.

**Conclusion**

* Using **Bag-of-Words with Naive Bayes** performs moderately well for text-based genre classification.
* **Oversampling** improves performance on underrepresented genres.
* Future improvements could include:
  + Using **TF-IDF or word embeddings like Word2Vec or GloVe**
  + Applying **Logistic Regression or SVM**
  + Implementing **multi-label classification** (as movies often have multiple genres)