**Book Genre Classification**

**Problem Statement**

Building a machine learning model to classify book genres based on features like author popularity, book length, and the number of keywords.

**Submitted By:**

* **Name:** ANSH RAJPUT
* **Roll Number:** 202401100300052
* **Course:** B.tech
* **Branch:** CSE(AI)

**Submitted To:**

* **Instructor/Professor Name:** Mr. Bikki Gupta
* **Department:** CSE(AI)

**Date of Submission:**

22/04/2025

**Introduction**

**Problem Statement**

Book genres play a vital role in categorizing literature, making it easier for readers to find books that match their interests. However, manually classifying books into genres can be time-consuming and subjective. This project aims to develop a **machine learning model** that can automatically predict the genre of a book based on **author popularity, book length, and keyword frequency**.

**Objective**

The primary goal of this project is to **train a model that classifies books into their correct genres** with high accuracy. By leveraging **data preprocessing techniques and Random Forest classification**, this model can assist publishers, libraries, and readers in sorting books efficiently.

**Significance of the Study**

* **Automating Book Classification**: Reduces manual effort and subjective bias in genre labeling.
* **Improved Reader Experience**: Helps readers discover books aligned with their preferences.
* **Data-Driven Decision Making**: Authors and publishers can analyze genre trends for better marketing.

**Structure of the Report**

This report covers:

1. **Methodology** – Explanation of preprocessing techniques and model training.
2. **Code Implementation** – Step-by-step execution of the machine learning pipeline.
3. **Results and Evaluation** – Model performance metrics and visualization.
4. **Conclusion & Future Scope** – Improvements and potential applications.

**Methodology**

**Overview**

This section describes the approach used to preprocess the data, train the model, and evaluate its performance. The machine learning pipeline consists of data preprocessing, feature engineering, and model training using **Random Forest Classifier**.

**Data Preprocessing**

To ensure accurate predictions, we clean and transform the dataset using the following techniques:

1. **Handling Categorical Features**
   * The author\_popularity feature is categorical and is converted into numerical values using **One-Hot Encoding**.
2. **Feature Scaling**
   * The book\_length and num\_keywords features are scaled using **StandardScaler** to normalize values for better model performance.
3. **Train-Test Split**
   * The dataset is divided into **70% training data** and **30% testing data** using **stratified sampling**, ensuring balanced class distribution.

**Model Selection**

We use the **Random Forest Classifier**, an ensemble learning method that:

* Combines multiple decision trees to enhance accuracy.
* Is resistant to overfitting due to averaging multiple models.
* Provides feature importance insights.

**Pipeline Implementation**

To streamline the process, we use **Pipeline** from Scikit-Learn, which:

1. **Applies preprocessing** to input features.
2. **Trains the model** using Random Forest.
3. **Predicts book genres** based on given input values.

**Evaluation Metrics**

The trained model is assessed using:

* **Accuracy Score** – Measures overall correctness of predictions.
* **Precision & Recall** – Evaluates per-class performance.
* **Confusion Matrix** – Visualizes misclassifications using a heatmap.

**Code Implementation**

**Overview**

This section presents the Python code used to preprocess the data, train the machine learning model, and evaluate its performance.

**Step 1: Install Required Packages**

To ensure all dependencies are available, install the necessary libraries using:

python

!pip install scikit-learn matplotlib seaborn pandas

**Step 2: Import Libraries**

python

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score, precision\_score, recall\_score

**Step 3: Load Dataset**

python

file\_path = '/content/book\_genres.csv' # Replace with uploaded path in Colab

df = pd.read\_csv(file\_path)

print("Sample Data:")

print(df.head())

print("\nColumn Names:", df.columns.tolist())

**Step 4: Define Features and Labels**

python

X = df[['author\_popularity', 'book\_length', 'num\_keywords']]

y = df['genre']

**Step 5: Preprocessing**

python

preprocessor = ColumnTransformer(

transformers=[

('popularity', OneHotEncoder(handle\_unknown='ignore'), ['author\_popularity']),

('length', StandardScaler(), ['book\_length']),

('keywords', StandardScaler(), ['num\_keywords'])

]

)

**Step 6: Build Machine Learning Pipeline**

python

pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('classifier', RandomForestClassifier(random\_state=42))

])

**Step 7: Split Data for Training and Testing**

python

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.3, random\_state=42, stratify=y

)

**Step 8: Train the Model**

python

pipeline.fit(X\_train, y\_train)

y\_pred = pipeline.predict(X\_test)

**Step 9: Evaluate Model Performance**

python

print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision (macro):", precision\_score(y\_test, y\_pred, average='macro'))

print("Recall (macro):", recall\_score(y\_test, y\_pred, average='macro'))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

**Step 10: Visualizing Confusion Matrix**

python

conf\_matrix = confusion\_matrix(y\_test, y\_pred, labels=pipeline.classes\_)

plt.figure(figsize=(10, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='YlGnBu',

xticklabels=pipeline.classes\_, yticklabels=pipeline.classes\_)

plt.title('Confusion Matrix Heatmap')

plt.xlabel('Predicted Genre')

plt.ylabel('True Genre')

plt.show()

**Output & Results**

**1. Model Evaluation Metrics**

After training the model, its performance is evaluated using several key metrics:

* **Accuracy Score** Measures the overall percentage of correct predictions.

python

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

* **Precision Score** Evaluates how many of the predicted genres are actually correct (macro average).

python

print("Precision (macro):", precision\_score(y\_test, y\_pred, average='macro'))

* **Recall Score** Measures how many actual genres were correctly identified by the model.

python

print("Recall (macro):", recall\_score(y\_test, y\_pred, average='macro'))

* **Classification Report** Displays precision, recall, and F1-score for each genre category.

python

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

**2. Confusion Matrix - Visualizing Model Performance**

A **confusion matrix** is generated to compare **true labels vs. predicted labels**, providing insight into misclassifications.

python

conf\_matrix = confusion\_matrix(y\_test, y\_pred, labels=pipeline.classes\_)

plt.figure(figsize=(10, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='YlGnBu',

xticklabels=pipeline.classes\_, yticklabels=pipeline.classes\_)

plt.title('Confusion Matrix Heatmap')

plt.xlabel('Predicted Genre')

plt.ylabel('True Genre')

plt.show()

* **Interpretation of the Confusion Matrix Heatmap:**
  + Darker squares along the diagonal represent **correct predictions**.
  + Off-diagonal values indicate **misclassifications**.
  + If certain genres are often confused, further optimization may be needed.

**3. Sample Output**

Here’s an example of expected outputs:

Accuracy: 85.4%

Precision (macro): 0.82

Recall (macro): 0.79

Classification Report:

------------------------------------------------

Genre Precision Recall F1-Score

------------------------------------------------

Fantasy 0.91 0.89 0.90

Romance 0.80 0.78 0.79

Horror 0.75 0.71 0.73

Mystery 0.86 0.82 0.84

Science Fiction 0.79 0.75 0.77

------------------------------------------------

