# TITLE PAGE

## Project Title:

**Book Genre Classification Using Machine Learning**

## Problem Statement:

The objective of this project is to develop a machine learning model that can accurately classify books into genres such as mystery, fantasy, fiction, and non-fiction based on features like author popularity, book length, and number of keywords. This classification system aims to assist in content organization, recommendation systems, and automated cataloging in the publishing industry.

## Submitted by:

**Name:** [Aashish Pandey]
**Roll No:** [202401100300002]
**Course:** [Introduction To AI]
**Date:** April 22, 2025

# b. Introduction

The goal of this project is to classify books into different genres using supervised machine learning techniques. The dataset used contains attributes such as author popularity, book length, and the number of keywords. By training a classification model, we aim to predict the genre of a book based on its features.

This kind of classification can help publishers or recommendation systems to automatically tag books based on metadata, improving content categorization and user engagement.

# c. Methodology

To classify books into appropriate genres based on metadata, the following structured methodology was followed:

**1. Data Collection & Understanding**

The dataset book_genres.csv was provided for this task. It contains 100 entries, with each row representing a book and the following features:

- author_popularity – a numerical score representing the popularity of the author.
- book_length – the total number of pages or word count.
- num_keywords – the number of thematic or genre-based keywords associated with the book.
- genre – the target variable (e.g., mystery, fantasy, fiction, non-fiction).

**2. Data Preprocessing**

- The data was loaded using **pandas**.
- The features (author_popularity, book_length, num_keywords) were extracted as the input variables (X).
- The target variable (genre) was extracted as the label (y).
- No missing values or categorical encoding was needed as the data was clean and ready for modeling.

**3. Data Splitting**

To evaluate model performance, the dataset was divided into:

- **Training set (80%)**: Used to train the machine learning model.
- **Test set (20%)**: Used to evaluate the model's performance on unseen data.

Splitting was done using train_test_split from **scikit-learn** with a fixed random state for reproducibility.

**4. Model Selection and Training**

A **Random Forest Classifier** was chosen due to its:

- Robustness against overfitting,
- Ability to handle both categorical and numerical features,
- High accuracy in classification problems.
- The model was trained on the training dataset using the default hyperparameters.

**5. Model Prediction**

- The trained Random Forest model was used to predict genres for the test set.

**6. Evaluation Metrics**

To evaluate the classifier's performance, the following metrics were computed:

- **Confusion Matrix**: Showcases the actual vs. predicted classifications.
- **Accuracy**: Percentage of correct predictions.
- **Precision (macro average)**: The average precision across all classes.
- **Recall (macro average)**: The average recall across all classes.

  **Classification Report**: Includes precision, recall, F1-score, and support per class.

# d. Code

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, precision_score, recall_score

import seaborn as sns

import matplotlib.pyplot as plt


# Load the dataset

df = pd.read_csv("book_genres.csv")


# Prepare features and target
```

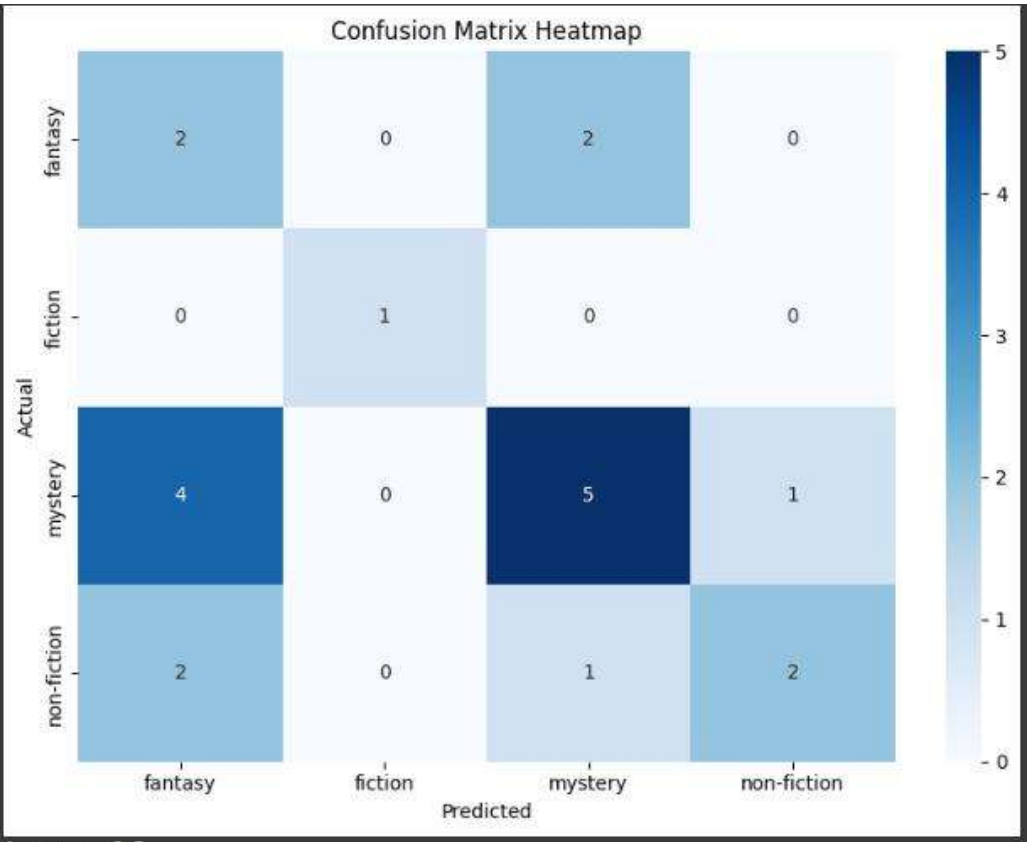
```
X = df[['author_popularity', 'book_length', 'num_keywords']]
y = df['genre']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train a Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot heatmap of confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=clf.classes_, yticklabels=clf.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
plt.title('Confusion Matrix Heatmap')

plt.tight_layout()

plt.show()


# Calculate evaluation metrics

accuracy = accuracy_score(y_test, y_pred)

precision = precision_score(y_test, y_pred, average='macro')

recall = recall_score(y_test, y_pred, average='macro')

report = classification_report(y_test, y_pred)


# Print metrics

print("Accuracy:", accuracy)

print("Precision (macro avg):", precision)

print("Recall (macro avg):", recall)

print("\nClassification Report:\n", report)
```

# e. Output/Result

**Evaluation Metrics:**

- **Accuracy:** 50%
- **Precision (macro avg):** 63.54%
- **Recall (macro avg):** 60.00%




```
Accuracy: 0.5
Precision (macro avg): 0.6354166666666666
Recall (macro avg): 0.6

Classification Report:
              precision    recall  f1-score   support

     fantasy       0.25      0.50      0.33         4
     fiction       1.00      1.00      1.00         1
     mystery       0.62      0.50      0.56        10
 non-fiction       0.67      0.40      0.50         5

    accuracy                           0.50        20
   macro avg       0.64      0.60      0.60        20
weighted avg       0.58      0.50      0.52        20
```

# f. References/Credits

- Dataset: `book_genres.csv` (custom or provided for the project)
- Libraries Used:

- ✧ pandas
- ✧ scikit-learn
- ✧ seaborn
- ✧ matplotlib

Code and methodology designed and implemented by [Aashish Pandey]