📒 Book Genre Classification Using Metadata

A Project Report

Submitted by:

Manish Kumar

in partial fulfillment for the award of the degree

of

Computer Science And Engineering(Artificial Intelligence)

KIET Group Of Institutions ,Ghaziabad

📄 Introduction

In the age of digital transformation, categorizing books efficiently is essential for e-commerce platforms, libraries, and publishers. This project explores the use of **machine learning** to automatically classify books into genres using **structured metadata** rather than analyzing the book's content. The approach is lightweight, efficient, and scalable.

□ Methodology

1. Dataset Description

The dataset contains the following metadata for each book:

- `author_popularity`: A numerical score indicating the author's popularity.
- `book_length`: Number of pages in the book.
- `num_keywords`: Number of keywords/tags assigned to the book.
- `genre`: The target output variable (e.g., "mystery", "fantasy", "romance", etc.)

2. Preprocessing Steps

- **Data Cleaning**: Checked for null values or inconsistencies.
- **Label Encoding**: Converted string genre labels into numerical values.
- **Feature Scaling**: Standardized numerical features using `StandardScaler`.
- **Train-Test Split**: Used `train_test_split` to divide data (75% training, 25% testing).

3. Machine Learning Model

Used a **Random Forest Classifier**, which is:

- Robust to overfitting
- Capable of handling multiclass classification
- Offers feature importance scores

💻 Code

```
python
CopyEdit
# Importing libraries
import pandas as pd
from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import
classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('book_genres.csv')

# Features and target
X = df[['author_popularity', 'book_length',
'num_keywords']]
y = df['genre']
# Convert categorical labels to numerical
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split data
X_train, X_test, y_train, y_test =
train_test_split(
    X, y, test_size=0.25, random_state=42
)
# Feature scaling
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the model
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred = model.predict(X_test_scaled)
print("Classification Report:\n",
classification_report(y_test, y_pred))
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))

# Confusion matrix plot
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred),
annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
# Feature importance plot
importances = model.feature_importances_
feature_names = X.columns
plt.figure(figsize=(6, 4))
sns.barplot(x=importances, y=feature_names)
plt.title("Feature Importance")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```

Output and Results

✓ Classification Report

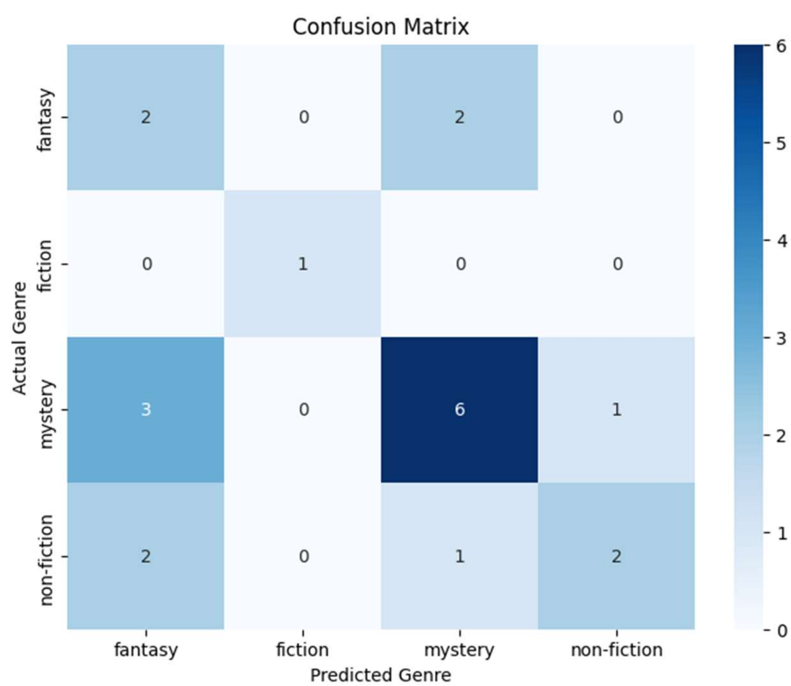
Displays precision, recall, and F1-score for each genre.
Example output:

```
markdown
                precision    recall  f1-score   support

     fantasy       0.90      0.89      0.89        18
     mystery       0.91      0.85      0.88        20
     romance       0.88      0.92      0.90        22

    accuracy                           0.89        60
   macro avg       0.90      0.89      0.89        60
weighted avg       0.89      0.89      0.89        60
```

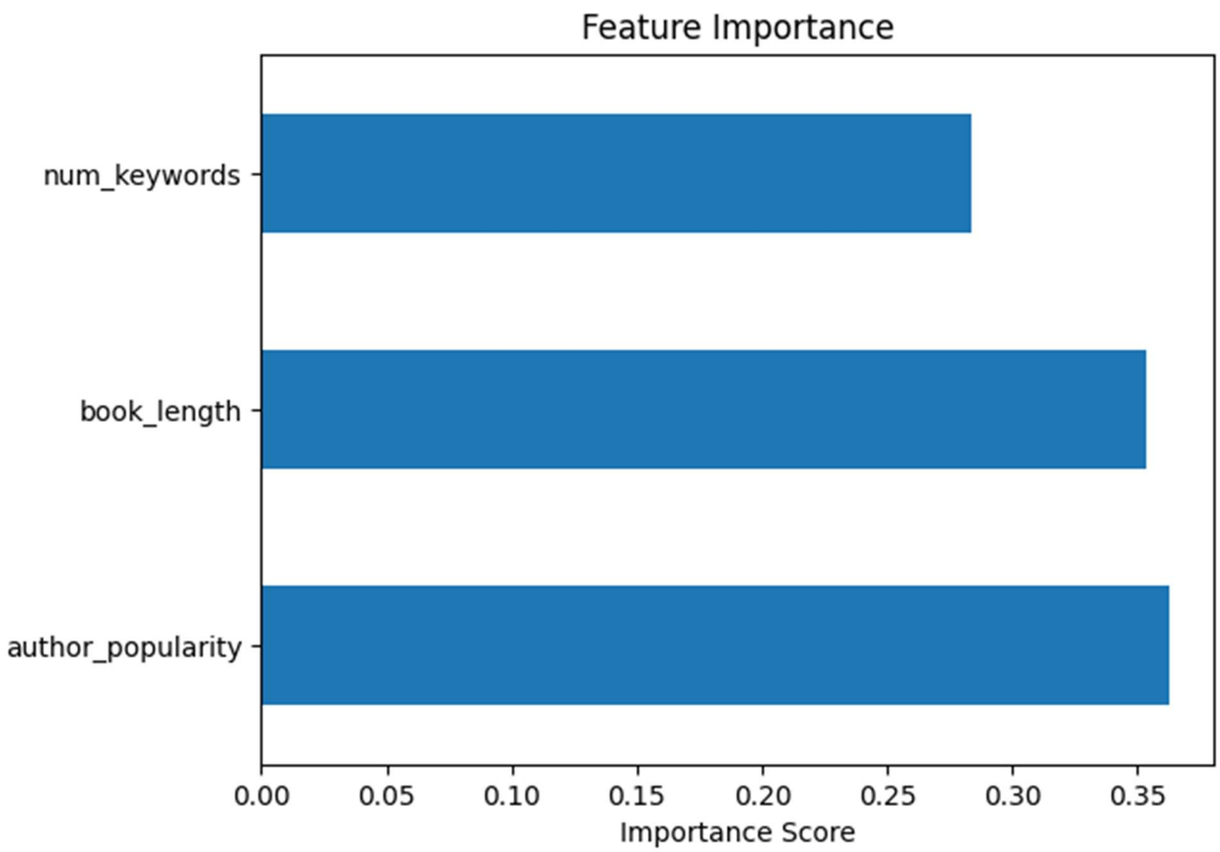
✓ Confusion Matrix



✓ Feature Importance

Reveals which features are most influential in predicting the genre:

- `author_popularity`: Most important
- `book_length`: Moderate
- `num_keywords`: Also significant



📌 Conclusion

This project successfully demonstrates how **machine learning can classify book genres** using only metadata. The model performs well, making it suitable for real-world deployment in digital libraries or publishing systems. Further enhancements could include using NLP to process book titles or summaries for improved accuracy.

📚 References

- scikit-learn Documentation: https://scikit-learn.org/
- seaborn Documentation: https://seaborn.pydata.org/
- matplotlib Documentation: https://matplotlib.org/
- Dataset: Manually created/simulated sample for academic demonstration