Project no :- 4

Prediction of genre of books

(Natural language processing)

By

Ankit G Dhanore

Content

- Problem statement
- Objective
- Introduction
- Importing libraries
- Data summary
- Preprocessing
- EDA
- Modelling
- Model comparison

Problem Statement

- Problem Statement: There is one library “Kitaab” in the city, they have stored tons of books since 1998, and they are getting lots of readers from everywhere. To enhance the readers, they entered into the digital world. They have launched an online book reading platform where they are uploading all their physical books. Along with that, they want to add some more features which

are possible to achieve by data science methods & techniques.

- Right now, they are working manually to create a genre of a book and tag or group those books together on their platform but it's very expensive and time-consuming work. Now they want to leverage data science power to automatically tag the genre to the books.

Objective

- The task is to develop a model that can predict the genres of books in a library collection. Given the title and plot summary of each book, the model should accurately classify the book into one or multiple genres such as science fiction, romance, mystery, fantasy, and horror, among others. This will help the library in categorizing and organizing its collection, as well as assist patrons in finding books that match their reading preferences. The solution should be able to handle a diverse range of genres and be robust enough to handle variations in writing styles and language used in the book summaries.

Introduction

- Books have been a source of entertainment and knowledge for centuries, and with the increasing number of books being published every year, it has become difficult to keep track of them all. Categorizing books based on their genre is one way of organizing this vast collection, making it easier for readers to find books that they are interested in. With the advancements in natural language processing and machine learning, it is now possible to automate this process by developing a model that can predict the genre of a book based on its title and plot summary.
- In this project, we aim to develop a model that can accurately predict the genre of books and make it easier for libraries and bookstores to categorize their collection.

Importing Libraries

- Here we import all the libraries which are required for the project like sklearn for modelling and preprocessing , matplotlib and seaborn for data visualization etc.
- We import nlp libraries also like nltk,langdetect, googletrans etc.
- Then import the data set and see some basic information regarding the dataset like no of rows and columns and their data type.

Data summary

- The data set is from online website kaggle which provide different datasets for machine learning projects.
- This data contains 1539 rows and 9 columns.
- **Title :-** The title of a book is the name given to it by the author or publisher, which typically reflects the subject or main idea of the work. It is often a brief and catchy phrase designed to capture the reader's attention and interest. This is important feature for deployment of model as well.
- **Rating :-** Rating of books refers to the process of evaluating and assigning a score or ranking to a book based on various criteria, such as the quality of the writing, plot, characters, themes, and overall impact. Ratings can be expressed using different scales, such as star ratings or numerical scores. These ratings can help readers quickly assess the perceived value or quality of a book and make informed decisions about whether to read it or not.
- **Name :-** As the name suggest this column tells us about the name of the author. Knowing the author's name can help readers identify other works they may have written, as well as provide context about the author's background, perspective, and literary style.

- **Num_ratings :-** The number of ratings of a book typically refers to the total count of ratings or reviews that a book has received from readers or critics. A higher number of ratings may suggest that the book has been read by many people, while a lower number of ratings may indicate that the book is less well-known or has not yet been widely read
- **Num_reviews :-** The number of reviews of a book refers to the total count of written evaluations or opinions that readers or critics have posted about the book. A higher number of reviews can indicate that the book has been read and discussed by many people, and may give potential readers a better understanding of the book's content, strengths, and weaknesses..
- **Num_followers :-** The number of followers refers to the count of individuals or accounts who have chosen to receive updates or content from a particular person or entity, such as an author or a book.
- **Synopsis :-** A synopsis of a book is a brief summary of the main plot, characters, and themes of the book. It typically provides an overview of the key events, conflicts, and resolutions in the story, as well as the major characters and their motivations. Synopses are often used by publishers, booksellers, and review sites to give readers a quick overview of the book's content and to help them decide whether to read the book.
- **Genre** :- Genre of books refers to the categories or types of books that share common characteristics, themes, or style. There are many other genres of books as well, including horror, thriller, comedy, and more. Understanding the genre of a book can help readers choose books that fit their interests and preferences.

Preprocessing

- First we checked for missing values but there are 0 missing values present in the dataset.

- Then we checked for duplicate values but there also duplicate values are present.
- Since there are no null or duplicate values are present the dataset we jump to next which is EDA and preprocessing.

```
In [5]: df.isna().sum()
Out[5]: Unnamed: 0      0
        title           0
        rating          0
        name            0
        num_ratings     0
        num_reviews     0
        num_followers   0
        synopsis        0
        genre           0
        dtype: int64

In [6]: df.duplicated().sum()
Out[6]: 0
```

- First we select the features which are requied for our project which is title, synopsis and genre.

```
df = df[["title","synopsis","genre"]]  ## considering columns which are required for our model.
df.head(5)
```

	title	synopsis	genre
0	Sapiens: A Brief History of Humankind	100,000 years ago, at least six human species ...	history
1	Guns, Germs, and Steel: The Fates of Human Soc...	"Diamond has written a book of remarkable scop...	history
2	A People's History of the United States	In the book, Zinn presented a different side o...	history
3	The Devil in the White City: Murder, Magic, an...	Author Erik Larson imbues the incredible event...	history
4	The Diary of a Young Girl	Discovered in the attic in which she spent the...	history

- As the book can be in any language we first detect the language using langdetect library.
- We can see that there are many different languages present in the synopsis.
- By default langdetect support 55 languages.

```
df["language_detected"].value_counts()

en    1523
es       4
fr       3
de       3
nl       1
vi       1
pt       1
pl       1
hu       1
lt       1
Name: language_detected, dtype: int64
```

- Now we translate this launguages into English with the help of google translate. By default this translate it into English.
- We create dependent and independent dataset before applying other preprocessing to avoid data leakage.

❖ Removing new line

- In NLP removing new lines from data is a common data cleaning technique that involves removing line breaks, carriage returns, and other special characters that indicate the end of a line or paragraph. This is often done to prepare text data for further processing.

```
text = "This is a \n sample text \n with new lines."

# Use replace() to replace new lines with spaces
clean_text = text.replace('\n', ' ')

print(clean_text)
# Output: "This is a  sample text  with new lines."
```

❖ Removing white space

- In NLP, removing white spaces is a common data cleaning technique used to preprocess text data. Extra white spaces, such as multiple spaces, tabs, or line breaks, can negatively affect NLP models' performance, as these models may interpret these spaces as separate tokens or words.

```
text = "  This is a   sample text   with extra  spaces.  "

# Use re.sub() to remove extra spaces with a regular expression
clean_text = re.sub('\s+', ' ', text).strip()

print(clean_text)
# Output: "This is a sample text with extra spaces."
```

❖ Removing accented characters

- Removing accented characters in NLP refers to the process of eliminating diacritics, which are marks added to letters to indicate changes in pronunciation or stress. This is often done to normalize text, reduce noise, and improve the performance of text-based models . Removing accented characters can be accomplished using various techniques, such as regular expressions, libraries, or customized code, depending on the specific needs of the NLP project

```
from unidecode import unidecode

text = "café"
text_no_accents = unidecode(text)

print(text_no_accents) # Output: "cafe"
```

❖ Doing contraction mapping

- In NLP, a contraction is a shortened form of a word or phrase that is created by combining two words and replacing one or more letters with an apostrophe. Contractions are commonly used in spoken and informal written language to save time and effort, but they can present a challenge for NLP models that are designed to process and understand text.
- The approach to deal with contractions is to expand them into their full form, using a pre-defined list of contractions and

```
text = "I don't think I can make it to the party. I'd like to go, but I'm too busy.

# Use contractions.fix() to expand contractions
expanded_text = contractions.fix(text)

print(expanded_text)
# Output: "I do not think I can make it to the party. I would like to go, but I am
```

their corresponding expanded forms. This process is called contraction expansion.

❖ Removing stopwords and Punctuations

- In NLP, stopwords are words that are commonly used in a language but do not carry significant meaning or context in a sentence. Examples of stopwords in English include "the," "and," "of," "a," and "in." Removing stopwords is a common preprocessing step in NLP because it can help to reduce the size of text data and improve the performance of NLP models.

```
def stop_words(data):
    tokens = RegexpTokenizer(r"[\w']+").tokenize(data)
    clean_text = [a for a in tokens if a not in stop]
    final_text = []
    for word in clean_text:
        if (len(word)>=2) or word in punctuation:
            pass
    else:
        final_text.append(word)
    text = " ".join(final_text)
    return text
```

- Punctuations are characters that are not part of words and are often irrelevant for NLP tasks. Removing punctuation can help to reduce the size of text data and improve the performance of NLP models.

❖ Stemming and lemmatization

- In NLP, stemming and lemmatization are two techniques used to reduce words to their base or root form. The goal of

```
def lemmatization(data):                                ## tokenizing the data.
    tokens = word_tokenize(data)
    tokens = [word.lower() for word in tokens if word.isalpha()]
    lemma = WordNetLemmatizer()
    final_text =[]
    for i in tokens:
        lemmatized_word = lemma.lemmatize(i)
        final_text.append(lemmatized_word)
    return " ".join(final_text)
```

stemming and lemmatization is to transform words into a common base form, so that different forms of the same word can be treated as a single word.

- Stemming is the process of removing suffixes from words to obtain their base form, known as the stem.
- Lemmatization, on the other hand, is the process of reducing words to their base form, known as the lemma. Unlike stemming, lemmatization considers the context of the word and its part of speech (POS), and produces real words as its output.

❖ Spelling correction

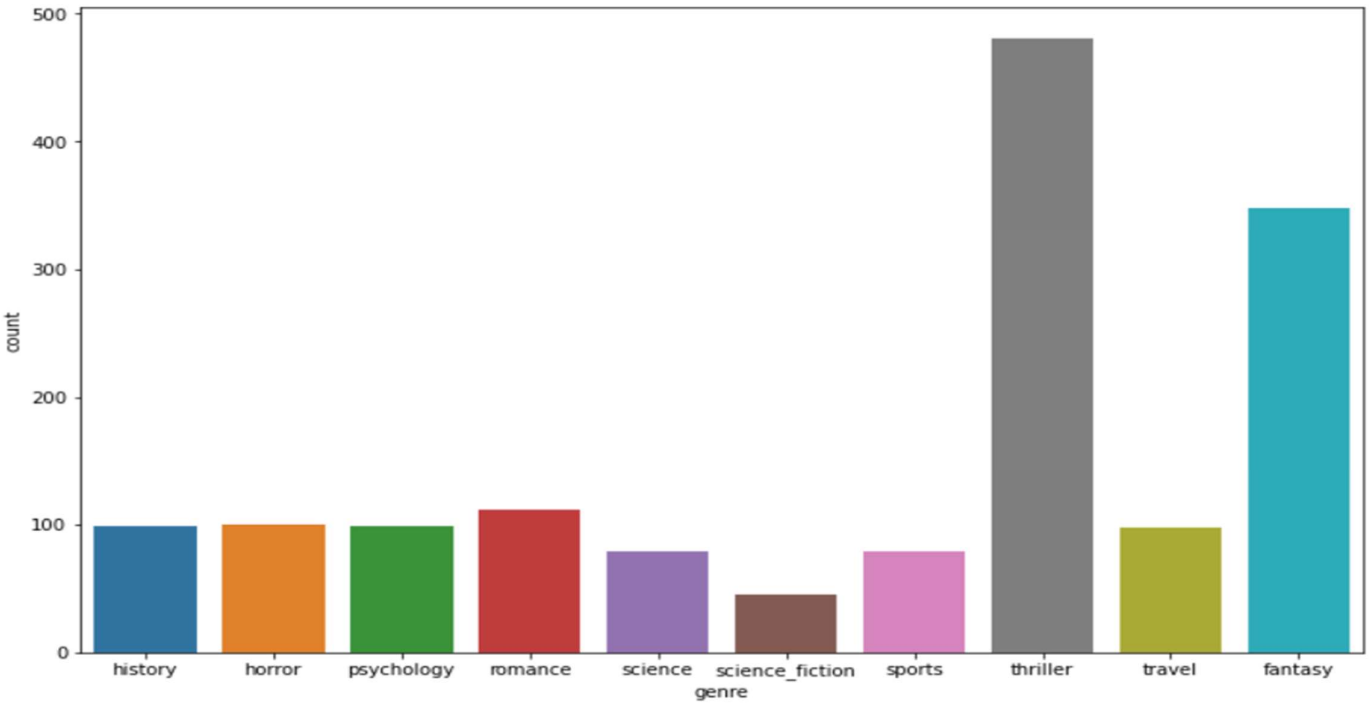
- Spelling correction in NLP involves the automatic detection and correction of misspelled words in text using various techniques such as rule-based methods, statistical models, and machine learning algorithms. These methods can help improve the accuracy and readability of text data in a wide range of applications, from search engines and chatbots to document processing and speech recognition systems.

```
def spelling_correction(data):
    spell = Speller(lang='en')
    corrrected_text = spell(data)
    return corrrected_text
```

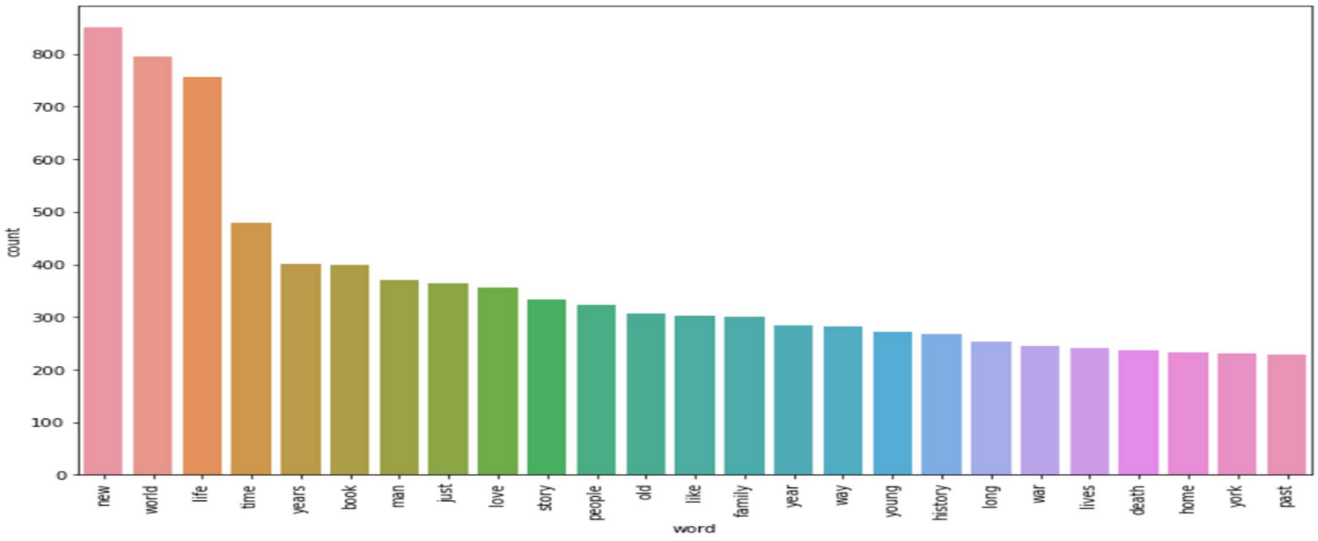
EDA (exploratory data analysis)

❖ For Genres

- From the above data it is clear that the thriller and fantasy have highest no of count.



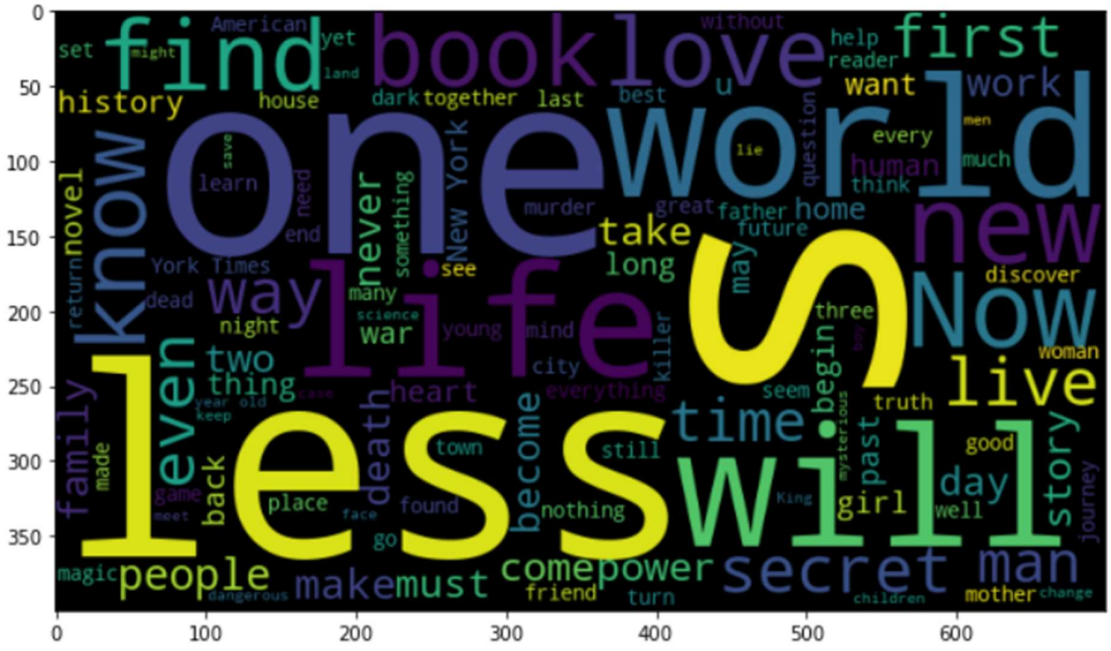
❖ Shows top 25 ngrams which are most frequent



- This are the top 25 ngrams(1,1) which are most repeated in the data.

❖ Word cloud

- The wordcloud is showing the words which are most repeated in the data.



Modelling

- Here we are going to use count vectorizer and TFIDF for word embedding

❖ Count vectorizer

- CountVectorizer is a popular technique in NLP that is used to transform a collection of text documents into a numerical matrix representing the frequency of each word in the document corpus. This technique is commonly used in tasks such as text classification, sentiment analysis, and information retrieval. The resulting matrix can be used as input to machine learning algorithms to build models for various NLP tasks. CountVectorizer is a part of the scikit-learn library in Python and provides a convenient way to convert text data into a usable format for NLP applications.

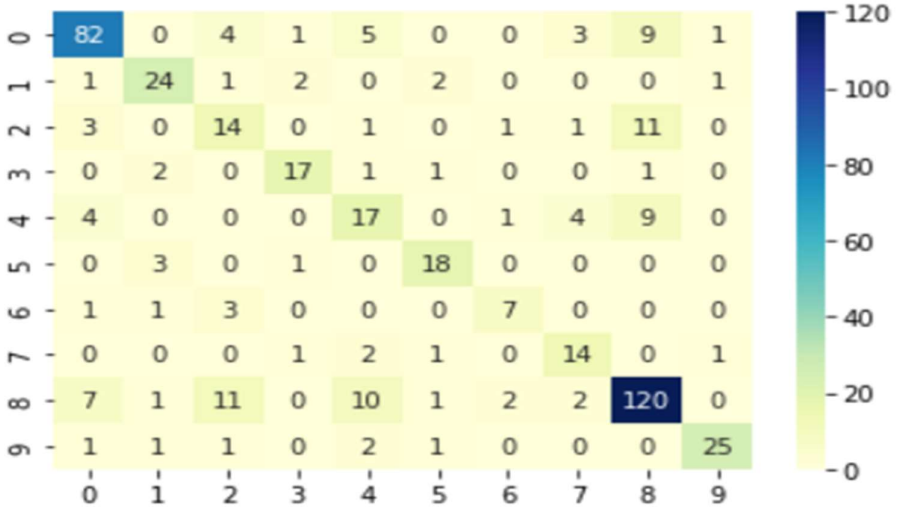
❖ TFIDF (Term Frequency-Inverse Document Frequency)

- TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used technique in natural language processing (NLP) for text feature extraction. It measures the importance of a term in a document or corpus by taking into account both its frequency in the document and its rarity in the corpus. TF-IDF assigns a high score to a term that is frequent in a document but rare in the corpus, and a low score to a term that is frequent in the corpus but less important in the document.

Model comparison

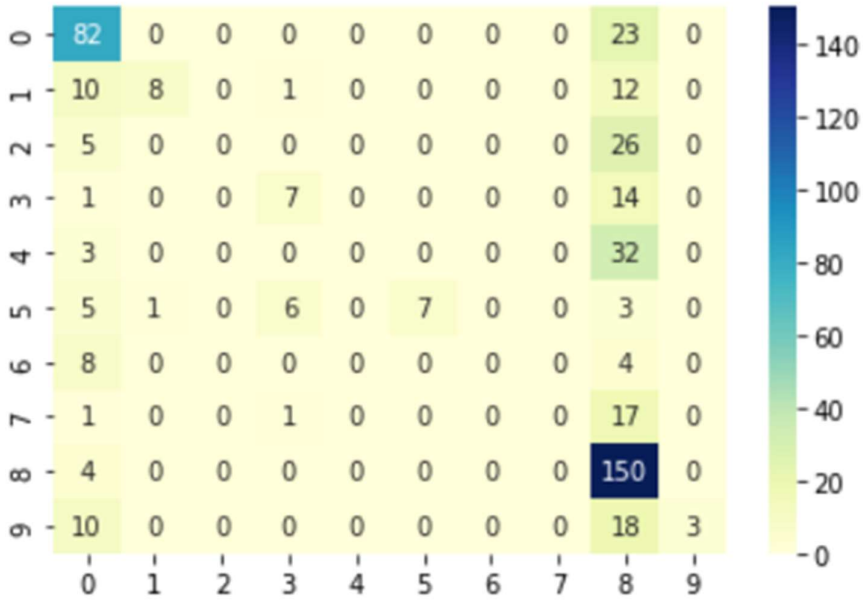
❖ Count vectorizer (evaluating using multinomial naive bayes algorithm)

- accuracy score is : 0.73
- precision is : 0.74
- recall is : 0.73
- f1 score is : 0.73
- confusion matrix is



❖ TFIDF (evaluating using multinomial naive bayes algorithm.)

- accuracy score is : 0.56
- precision is : 0.51
- recall is : 0.56
- f1 score is : 0.46
- confusion matrix is



Challenges faced

- Preprocessing.
- Doing feature engineering was difficult as the data was huge.

Thank you