ILLINOIS INSTITUTE OF TECHNOLOGY

CSP 571 - Data Preparation and Analysis

Final Project

Sentiment Analysis to Detect Book Genres

Shahrukh Sohail
ssohail3@hawk.iit.edu

Prof. Jawahar Panchal

Dec 4, 2022

Table of Contents

1. Abstract

Sentiment analysis is not new. It has been used in multiple different fields outside of pure data analysis for decades and has repeatedly been used to solve big problems. The aim of this project is not to solve any big problems, but to instead analyze the different models that are commonly used for sentiment analysis in order to attempt to gain some insight on the strengths and weaknesses of each model.

This will be done through the use of a toy dataset. More specifically, a dataset that contains books, their genres, and their synopsis. With this dataset, the aim is to be able to decipher the book's genre given its synopsis. This, of course, will be accomplished through sentiment analysis.

While the main focus is to understand the performances the different models have, asking and attempting to answer certain questions about the dataset are not only interesting, but will serve to aid in the model design as well.

1.1. Specific Questions

- Which words or phrases posses specifically strong correlation with certain genres
- What words, if any, can be removed to either remove complexity or error

2. Overview

Books are a massive part of our life, whether it be directly or indirectly, a lot of our day to day knowledge comes from books. Since the world is now getting more and more digitized, optimizing the navigation experience has become crucial to keeping people's attention. Genres have been one of the main ways people have been navigating through books and we wanted to create a machine learning model that can predict the genre of books as accurately as possible.

Machine learning has numerous objectives. Regression, classification, and clustering are the most prevalent. A machine learning system delivers continuous output in regression, such as fitting an equation to a point plot. The machine learning method divides items into groups based on their resemblance to one another in clustering. Classification is similar to clustering in that the machine learning algorithm seeks to categorize objects based on previously stated criteria supplied by training data. Classification and clustering are examples of supervised and unsupervised machine learning, respectively.

It is also important to introduce **parametric** and **non-parametric** machine learning models here. Parametric models require specification of certain parameters in order for the model to train and produce results, such as the form of the hyperplane equation. On the other hand non-parametric models do not require any kind of parametric specifications and can train on the model 'out of the box'. In other words, parametric models assume the model function and as a result only the coefficients have to be determined, on the other hand non-parametric models do not assume the function of the model and thus have to calculate model function as well as the coefficients of the model. This project uses both parametric and non-parametric models to perform sentiment analysis.

This project applied four supervised learning techniques to predict the genre of book based on the synopsis provided, the following are the models that we will be using.

Table 1: Prediction Models

Models
Bernoulli Naive Bayes
Categorical Naive Bayes
Logistic Regression
Random Forest

3. Data Source

3.1. Data Properties

The publisher of the dataset mentioned that it has been webscraped from Goodreads website using Selenium and uploaded it to Kaggle. Our dataset has 1539 unique books, each having nine columns of information, or features. To bring this dataset into a usable format, we will perform exploratory data analysis (EDA) to look for missing values and skewness within the columns. The goal is to determine the ***genre*** variable, classify a book to a given genre.

The 9 columns of the dataset and what each column represents:

1. *Unnamed: 0*: Order of the books in the dataset; the index
2. *title*: the title of the book
3. *rating*: the rating of the book (out of 5 stars, user-rated)
4. *name*: the name of the author
5. *num_ratings*: the number of users who have rated the book
6. *num_reviews*: the number of users who have reviewed the book
7. *num_followers*: the number of followers of the author
8. *synopsis*: synopsis/summary/description of the book
9. *genre*: the genre/type/category of the book

Table 2: Data types of columns

Column	Type	Data Type
Unnamed 0	Integer	int64
title	Double	object
rating	Float	float64
name	Object	object
num_ratings	Object	object
num_reviews	Object	object
num_followers	Object	object
synopsis	Object	object
genre	Object	object

4. Data Pre-Processing

In order for us to build our machine learning models, we have to first try to pre-process our data so that we do not have inaccuracies stemming from the data itself.

4.1. Software & Programming Languages

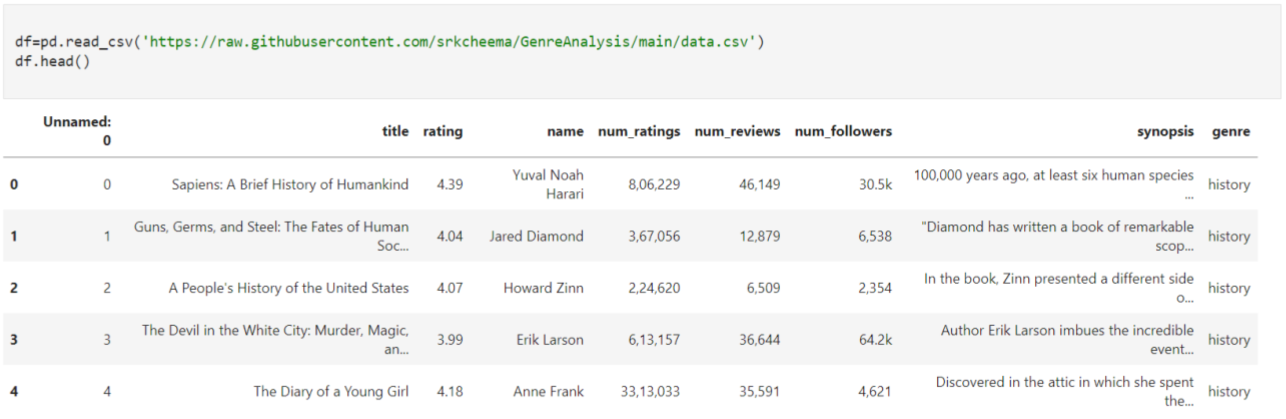
1. Python - Google Colab
2. Github

4.2. Data Acquisition

1. Download data
2. Upload data in Python environment

4.3. Data Cleaning

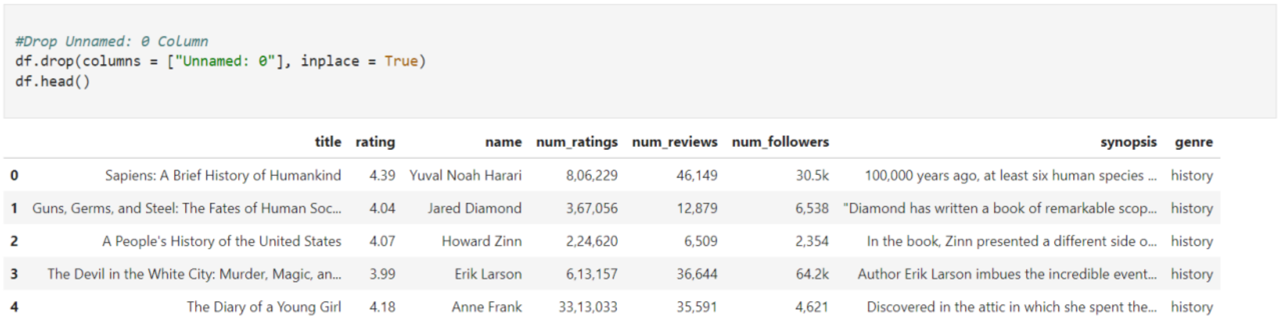
We start our cleaning process by getting as much information about the data itself.



```
df=pd.read_csv('https://raw.githubusercontent.com/srkcheema/GenreAnalysis/main/data.csv')
df.head()
```

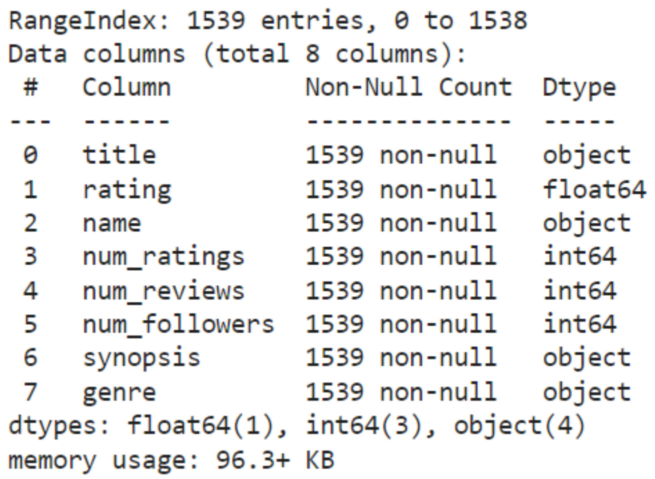
	Unnamed: 0	title	rating	name	num_ratings	num_reviews	num_followers	synopsis	genre
0	0	Sapiens: A Brief History of Humankind	4.39	Yuval Noah Harari	8,06,229	46,149	30.5k	100,000 years ago, at least six human species ...	history
1	1	Guns, Germs, and Steel: The Fates of Human Soc...	4.04	Jared Diamond	3,67,056	12,879	6,538	"Diamond has written a book of remarkable scop...	history
2	2	A People's History of the United States	4.07	Howard Zinn	2,24,620	6,509	2,354	In the book, Zinn presented a different side o...	history
3	3	The Devil in the White City: Murder, Magic, an...	3.99	Erik Larson	6,13,157	36,644	64.2k	Author Erik Larson imbues the incredible event...	history
4	4	The Diary of a Young Girl	4.18	Anne Frank	33,13,033	35,591	4,621	Discovered in the attic in which she spent the...	history

We noticed that the zeroth column was an unnamed column and of not much use as pandas will auto-index the data. We dropped the column from our dataframe and we also changed the data type of the number of ratings, reviews and followers columns to integers so that we can use it for analysis in the future.



```
#Drop Unnamed: 0 Column
df.drop(columns = ["Unnamed: 0"], inplace = True)
df.head()
```

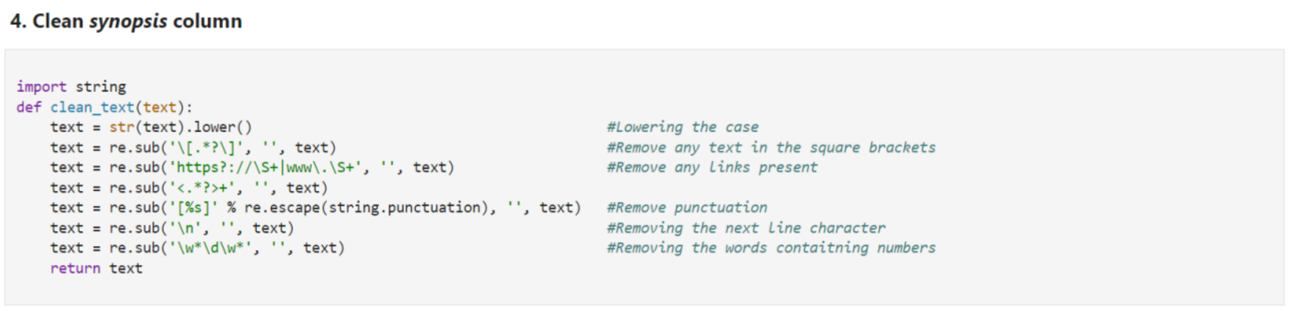
	title	rating	name	num_ratings	num_reviews	num_followers	synopsis	genre
0	Sapiens: A Brief History of Humankind	4.39	Yuval Noah Harari	8,06,229	46,149	30.5k	100,000 years ago, at least six human species ...	history
1	Guns, Germs, and Steel: The Fates of Human Soc...	4.04	Jared Diamond	3,67,056	12,879	6,538	"Diamond has written a book of remarkable scop...	history
2	A People's History of the United States	4.07	Howard Zinn	2,24,620	6,509	2,354	In the book, Zinn presented a different side o...	history
3	The Devil in the White City: Murder, Magic, an...	3.99	Erik Larson	6,13,157	36,644	64.2k	Author Erik Larson imbues the incredible event...	history
4	The Diary of a Young Girl	4.18	Anne Frank	33,13,033	35,591	4,621	Discovered in the attic in which she spent the...	history



```
RangeIndex: 1539 entries, 0 to 1538
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   title          1539 non-null   object
 1   rating         1539 non-null   float64
 2   name           1539 non-null   object
 3   num_ratings    1539 non-null   int64
 4   num_reviews    1539 non-null   int64
 5   num_followers  1539 non-null   int64
 6   synopsis       1539 non-null   object
 7   genre          1539 non-null   object
dtypes: float64(1), int64(3), object(4)
memory usage: 96.3+ KB
```

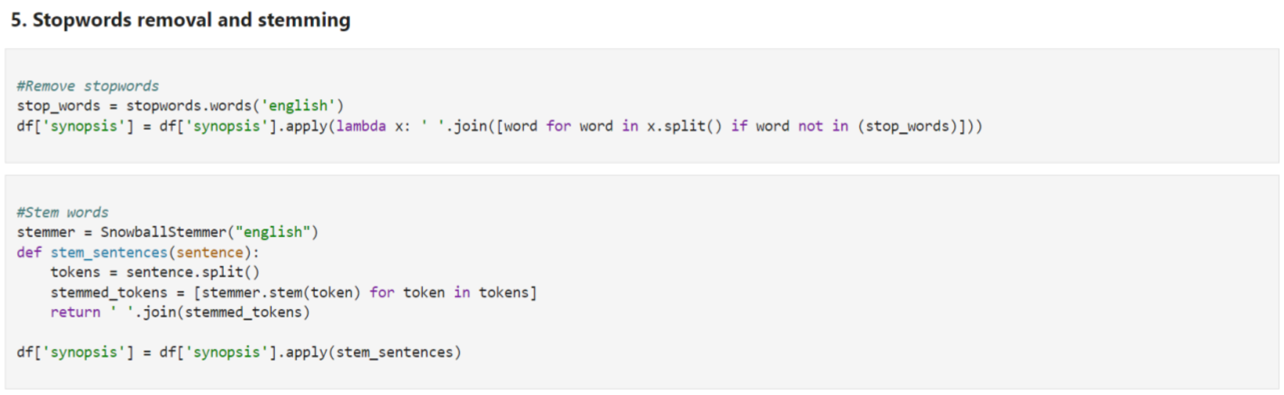
The next plan of our preprocessing phase was to process the synopsis column as it is difficult for the computer to understand the semantics of texts in its pure form. Some of pre-processing steps we did with the synopsis were

1. Convert all the text to lowercase
2. Remove any text in square brackets, links present, punctuations, next line characters and words containing numbers
3. Remove stopwords
4. Stemming the words



4. Clean *synopsis* column

```
import string
def clean_text(text):
    text = str(text).lower()                                              #Lowering the case
    text = re.sub('\[.*?\]', '', text)                                    #Remove any text in the square brackets
    text = re.sub('https?://\S+|www\.\S+', '', text)                      #Remove any links present
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)       #Remove punctuation
    text = re.sub('\n', '', text)                                         #Removing the next line character
    text = re.sub('\w*\d\w*', '', text)                                   #Removing the words contaitning numbers
    return text
```



5. Stopwords removal and stemming

```
#Remove stopwords
stop_words = stopwords.words('english')
df['synopsis'] = df['synopsis'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
```

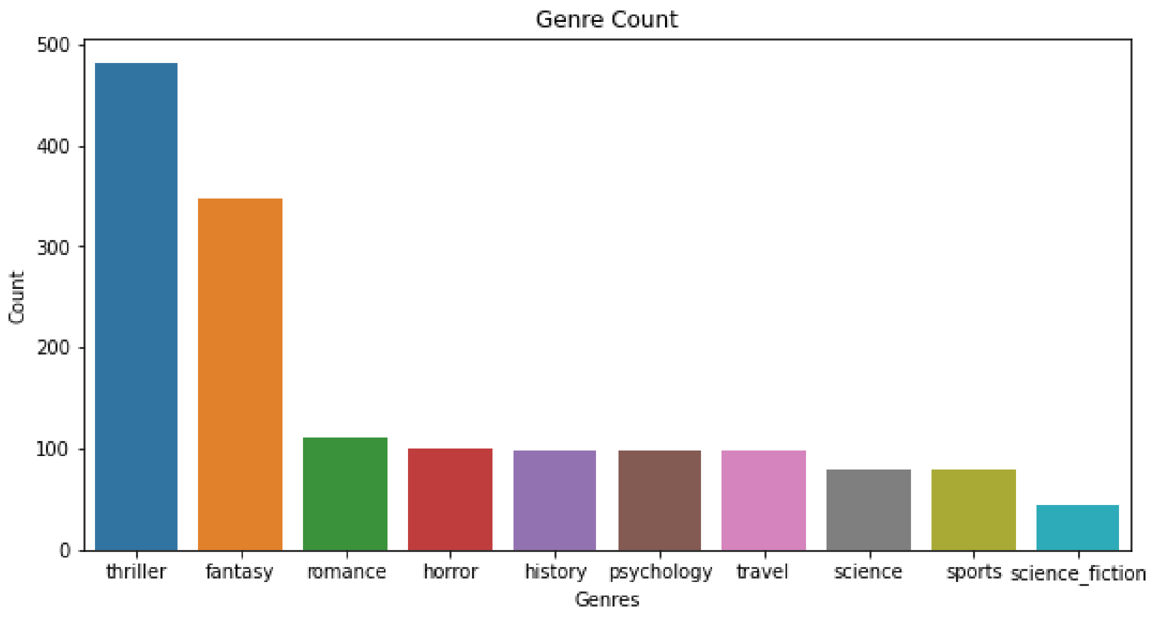
```
#Stem words
stemmer = SnowballStemmer("english")
def stem_sentences(sentence):
    tokens = sentence.split()
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return ' '.join(stemmed_tokens)

df['synopsis'] = df['synopsis'].apply(stem_sentences)
```

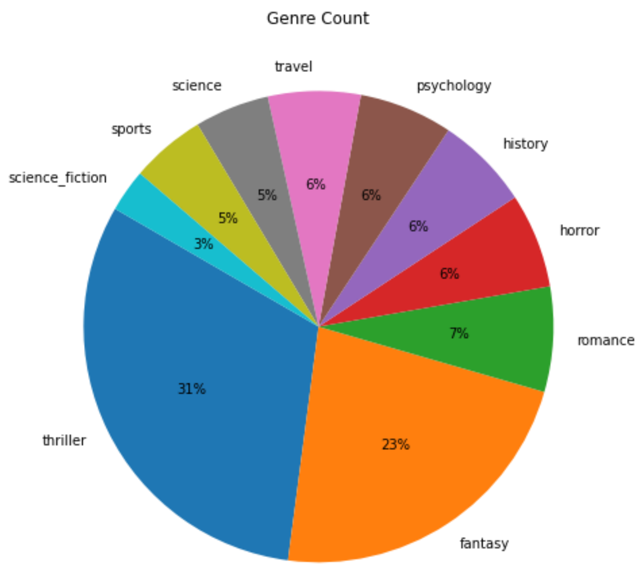
5. Data Visualization

We perform data visualization in order to highlight new insights about the underlying patterns and relationships contained within the data. We first start by getting a count of the number of unique genres in our dataset and the relative and absolute count of which genres there are more of in our dataset.

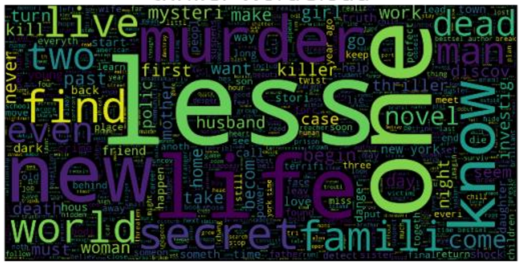
```python
print(df['genre'].unique())
print(df['genre'].nunique())
```

```
['history' 'horror' 'psychology' 'romance' 'science' 'science_fiction'
 'sports' 'thriller' 'travel' 'fantasy']
10
```

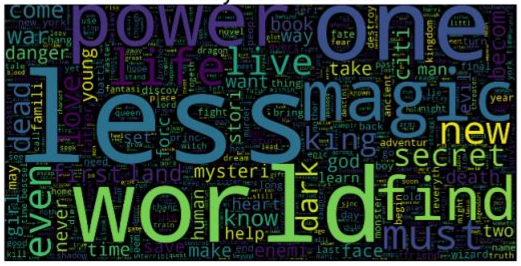




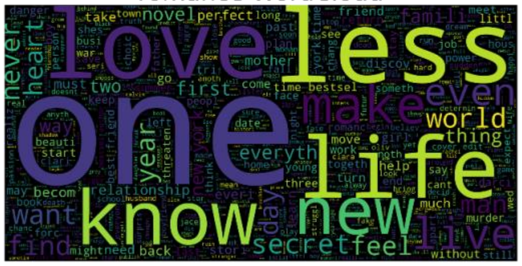
We also looked at the word cloud of all the different genres to see which words are used the most.



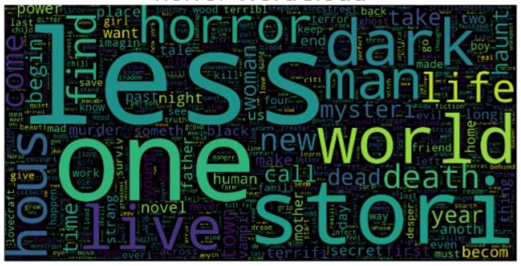
thriller WordCloud



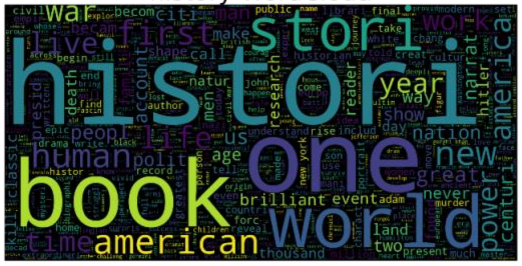
fantasy WordCloud



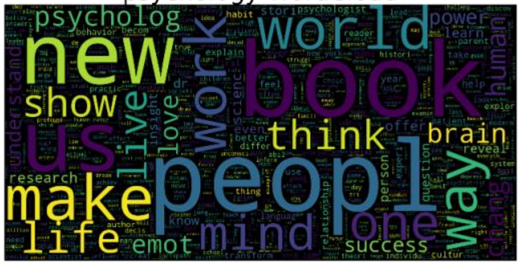
romance WordCloud



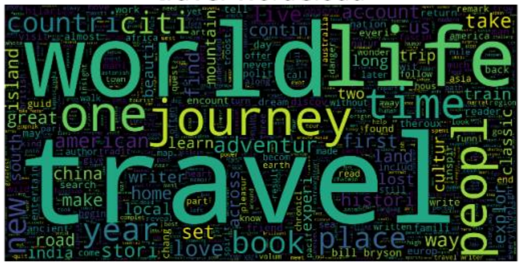
horror WordCloud



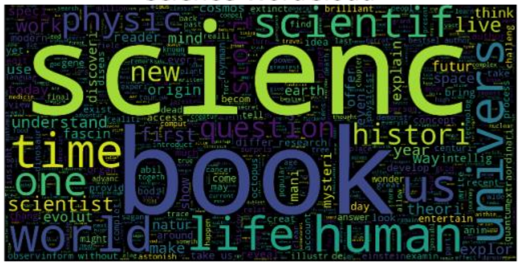
history WordCloud



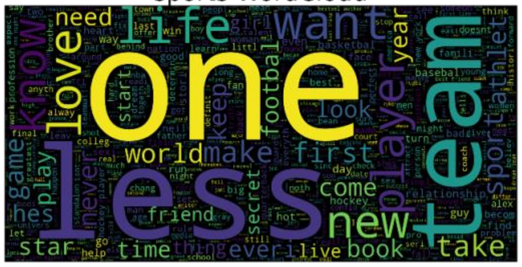
psychology WordCloud



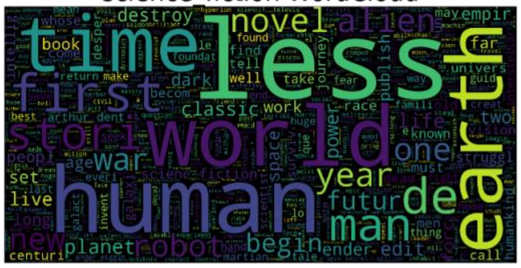
travel WordCloud



science WordCloud



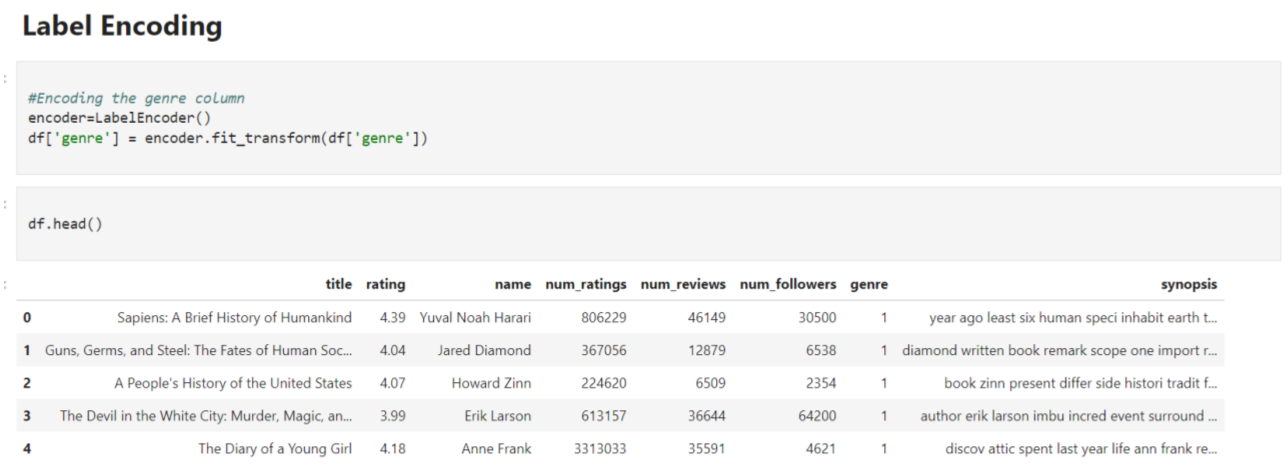
sports WordCloud



science_fiction WordCloud

6. Model Training

Before we start with our model training, we will encode our genre data into factorized values.



Label Encoding

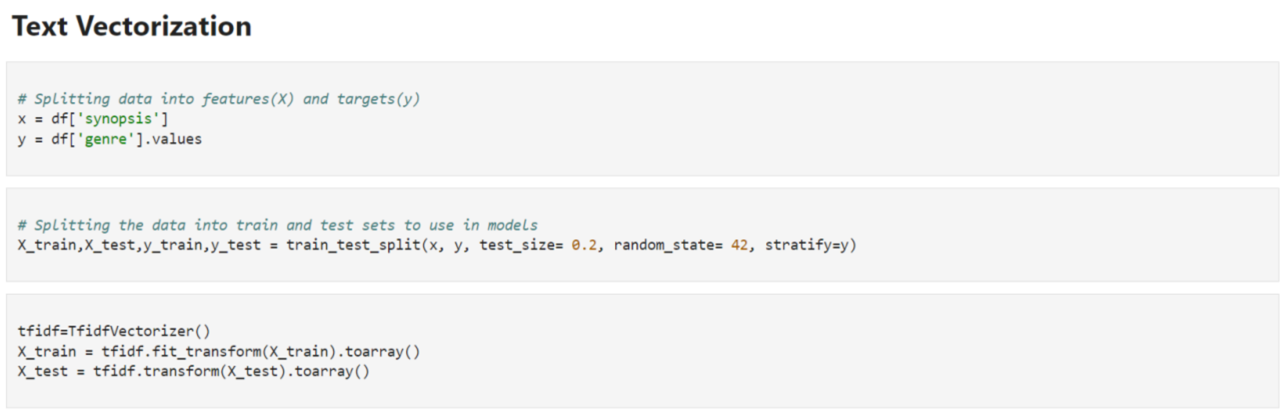
```
#Encoding the genre column
encoder=LabelEncoder()
df['genre'] = encoder.fit_transform(df['genre'])
```

```
df.head()
```

	title	rating	name	num_ratings	num_reviews	num_followers	genre	synopsis
0	Sapiens: A Brief History of Humankind	4.39	Yuval Noah Harari	806229	46149	30500	1	year ago least six human speci inhabit earth t...
1	Guns, Germs, and Steel: The Fates of Human Soc...	4.04	Jared Diamond	367056	12879	6538	1	diamond written book remark scope one import r...
2	A People's History of the United States	4.07	Howard Zinn	224620	6509	2354	1	book zinn present differ side histori tradit f...
3	The Devil in the White City: Murder, Magic, an...	3.99	Erik Larson	613157	36644	64200	1	author erik larson imbu incred event surround ...
4	The Diary of a Young Girl	4.18	Anne Frank	3313033	35591	4621	1	discov attic spent last year life ann frank re...

We then split the data into features and target values, split the data into 80% training set and 20% test set and also vectorize the data by performing TF-IDF since in order to analyze text we need to convert them into a numerical format where each word is represented by a matrix.

The formula for TF-IDF is TF*IDF which if individually expanded is Term Frequency (TF) = (Frequency of a term in the document)/(Total number of terms in documents) Inverse Document Frequency(IDF) = log((total number of documents)/(number of documents with term t))



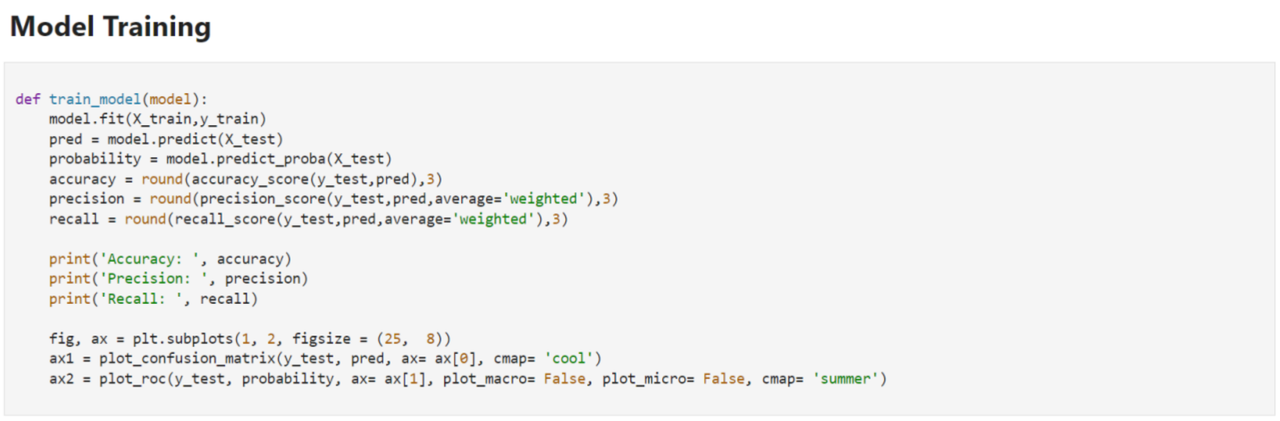
Text Vectorization

```
# Splitting data into features(X) and targets(y)
x = df['synopsis']
y = df['genre'].values
```

```
# Splitting the data into train and test sets to use in models
X_train,X_test,y_train,y_test = train_test_split(x, y, test_size= 0.2, random_state= 42, stratify=y)
```

```
tfidf=TfidfVectorizer()
X_train = tfidf.fit_transform(X_train).toarray()
X_test = tfidf.transform(X_test).toarray()
```

We then define a generic function that we can use to train and test our model with any machine leanring model so that we can then compare the models using various metrics



Model Training

```
def train_model(model):
    model.fit(X_train,y_train)
    pred = model.predict(X_test)
    probability = model.predict_proba(X_test)
    accuracy = round(accuracy_score(y_test,pred),3)
    precision = round(precision_score(y_test,pred,average='weighted'),3)
    recall = round(recall_score(y_test,pred,average='weighted'),3)

    print('Accuracy: ', accuracy)
    print('Precision: ', precision)
    print('Recall: ', recall)

    fig, ax = plt.subplots(1, 2, figsize = (25,  8))
    ax1 = plot_confusion_matrix(y_test, pred, ax= ax[0], cmap= 'cool')
    ax2 = plot_roc(y_test, probability, ax= ax[1], plot_macro= False, plot_micro= False, cmap= 'summer')
```

6.1. Machine Learning Models

We then deploy four machine learning models as stated earlier to classify the book genre.

6.1.1. Bernoulli Naive Bayes

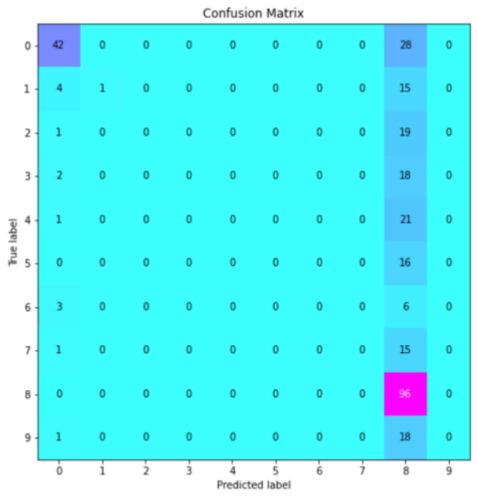
Bernoulli Naive Bayes is used for discrete data and it works on data that follows a Bernoulli distribution. The main feature of Bernoulli Naive Bayes is that it accepts features only as binary values like true or false, yes or no, success or failure, 0 or 1 and so on. So when the feature values are binary we know that we have to use Bernoulli Naive Bayes classifier. Bernoulli Naive Bayes Classifier is based on the following rule:

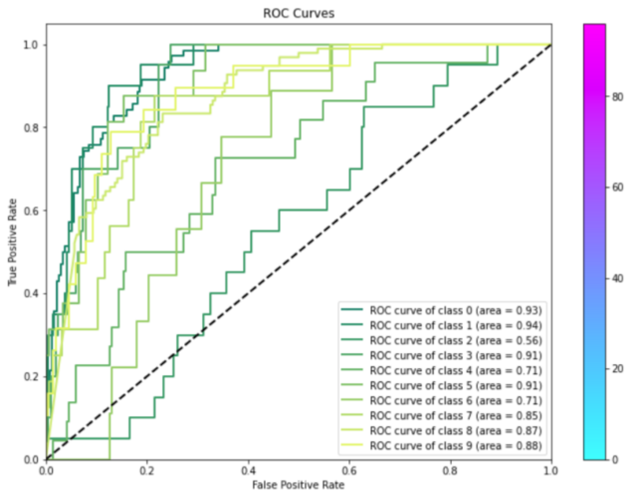
$$P(xi \mid y) = P(i \mid y)\, xi + (1 - P(i \mid y))(1 - xi)$$

The result of using Bernoulli Naive Bayes as our model was:

```python
from sklearn.naive_bayes import BernoulliNB

bernoulli_naive = BernoulliNB()
train_model(bernoulli_naive)
```

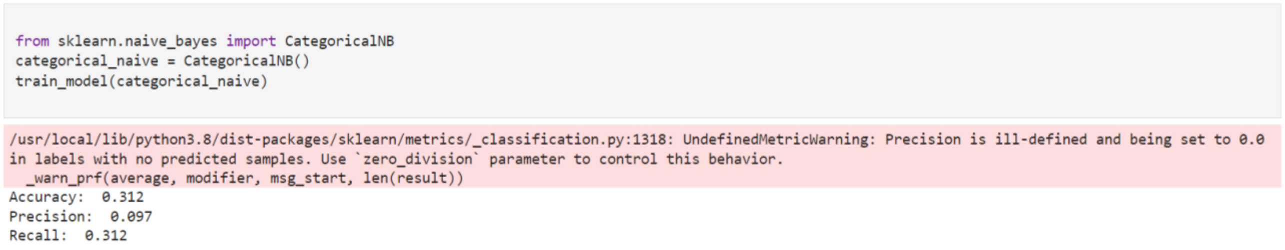
```
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy:  0.451
Precision:  0.357
Recall:  0.451
```





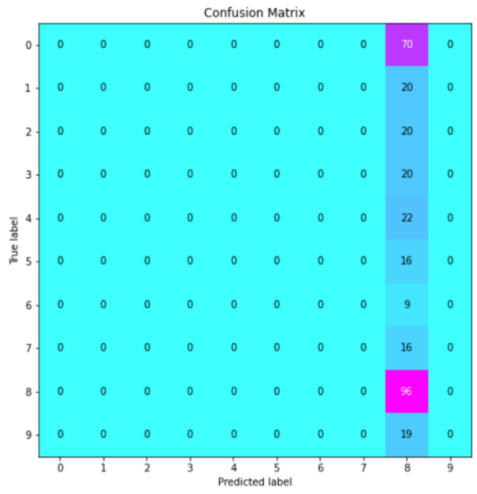
6.1.2. Categorical Naive Bayes

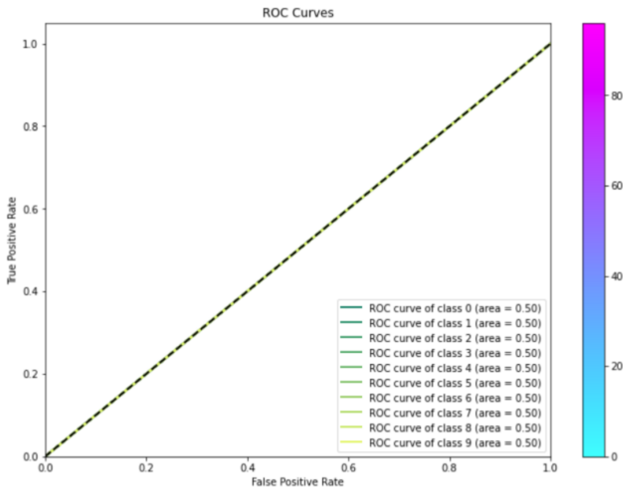
The categorical Naive Bayes classifier is suitable for classifying observations with discrete features. The categories of each feature are drawn from a categorical distribution. The result of using categorical Naive Bayes as our model was:



```
from sklearn.naive_bayes import CategoricalNB
categorical_naive = CategoricalNB()
train_model(categorical_naive)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy:  0.312
Precision:  0.097
Recall:  0.312
```

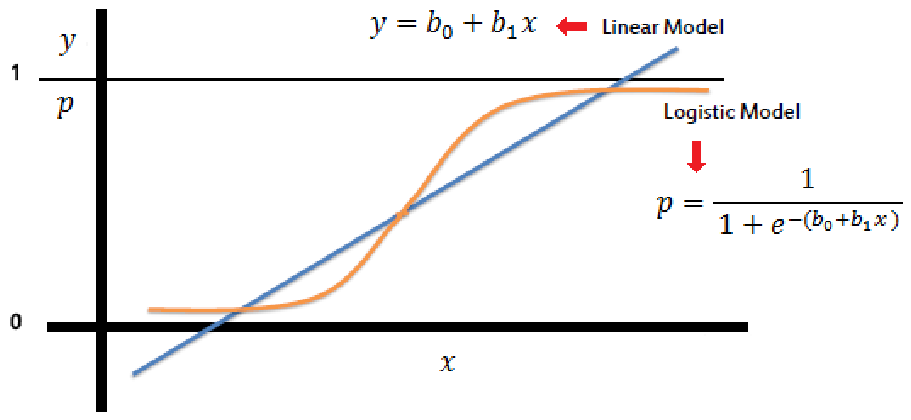




.

6.1.3. Logistic Regression

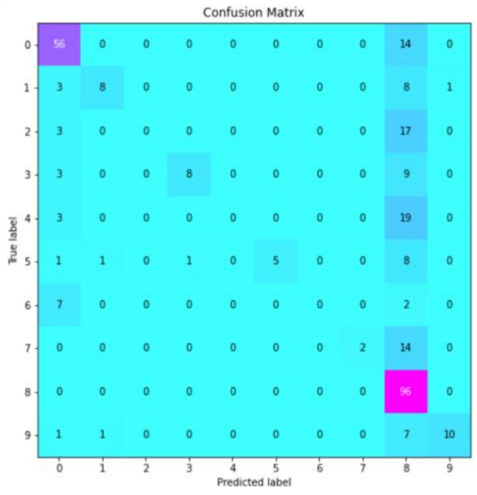
Logistic regression is a classification procedure that is used to forecast the likelihood of a target variable. The target or dependent variable has a dichotomous character, which means there are only two potential classes. In this case, we used one-against-all-others. The representation for logistic regression is an equation. To anticipate an output value, input data are linearly mixed with coefficient values. The output value is represented as a binary value, which distinguishes it from linear regression.



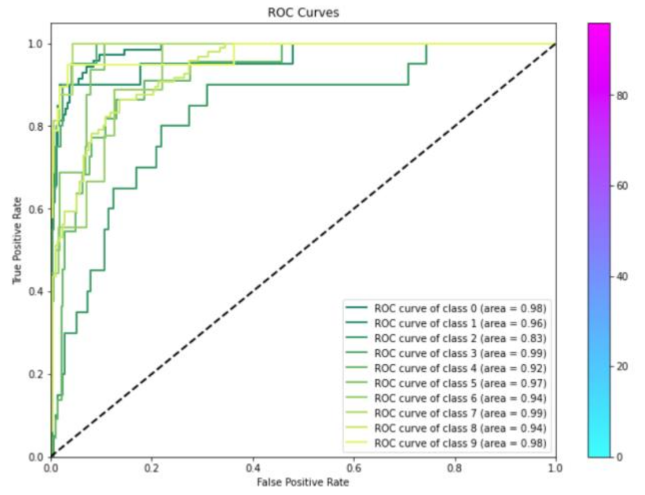
The result of using Linear Regression as our model was:

```
from sklearn.linear_model import LogisticRegression

logistic = LogisticRegression()
train_model(logistic)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy:  0.601
Precision:  0.589
Recall:  0.601
```



	0	1	2	3	4	5	6	7	8	9
0	56	0	0	0	0	0	0	0	14	0
1	3	8	0	0	0	0	0	0	8	1
2	3	0	0	0	0	0	0	0	17	0
3	3	0	0	8	0	0	0	0	9	0
4	3	0	0	0	0	0	0	0	19	0
5	1	1	0	1	0	5	0	0	8	0
6	7	0	0	0	0	0	0	0	2	0
7	0	0	0	0	0	0	0	2	14	0
8	0	0	0	0	0	0	0	0	96	0
9	1	1	0	0	0	0	0	0	7	10

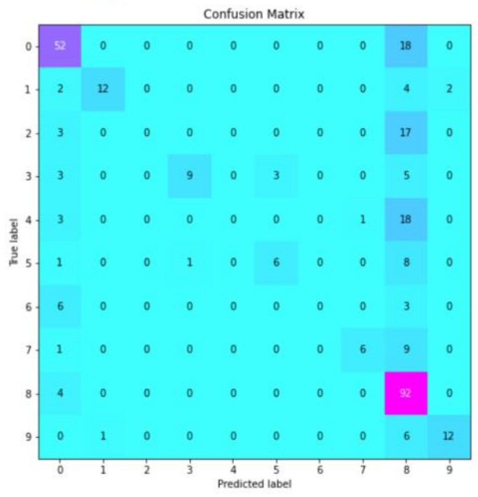


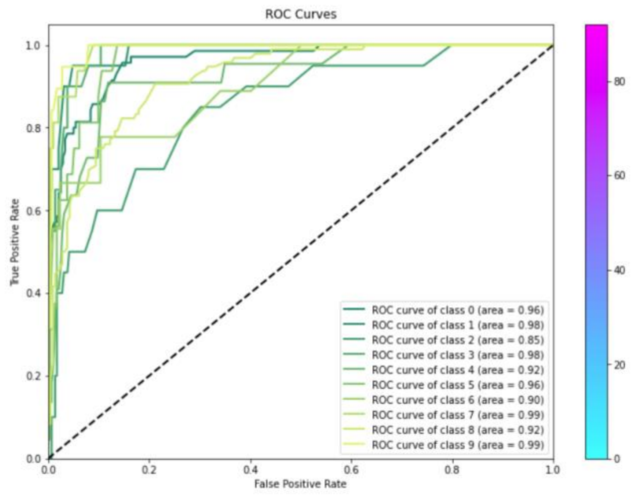
6.1.4. Random Forest

Random Forest is widely used for classification problems. It builds decision trees on different samples and takes their majority vote for classification. One of the important features of random forest is that it can handle continuous and categorical variables, which in our case, the focus would be more on the it's ability to handle categorical data. The result of using random forest as our model was:

```python
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier(n_estimators=300)
train_model(random_forest)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy:  0.614
Precision:  0.567
Recall:  0.614
```





7. Model Validation

Our goal was to compare different models and see which model yielded the highest accuracy. We can see from the table below which two models stood out from the rest.

Table 3: Prediction Results

Model	**Accuracy**
Bernoulli Naive Bayes	45.1%
Categorical Naive Bayes	31.2%
Logistic Regression	60.1%
Random Forest	61.4%

Random Forest Model had the greatest prediction accuracy of 61.4%, followed by Logistic Regression with an accuracy score of 60.1%.

8. Conclusions and Future Scope

In this project, we have investigated analyzing texts from books in the form of synopsis and try to predict the genre of the book through that information. For our preprocessing step, we heavily emphasized on the synopsis to make sure it is easy to work with our machine learning models. We used 4 different models at the end and analyzed which model yielded the highest accuracy.

So far the highest accuracy that we were able to achieve with our model is around 61.4% which is not reliable enough to be used as an actual predictive tool so for future plan we look towards optimizing our model by taking more parameters like rating, followers and reviews into consideration when building our model. We can also look into using a larger dataset and experimenting with smaller or larger scope of genres in order to see whether we would be able to build a more reliable model, used for mass tagging books on digital platforms. Finally, tackling more than one language could also be an ambitious goal for this project.

a.

9. Data Sources

[1] https://www.kaggle.com/datasets/athu1105/tagmybook

[2] https://raw.githubusercontent.com/srkcheema/GenreAnalysis/main/data.csv

10. Source Code

https://github.com/srkcheema/GenreAnalysis

11. Bibliography and Credits

[1] *Book Genre Categorization Using Machine Learning Algorithms (K-Nearest ...* https://www.researchgate.net/publication/350102357_Book_Genre_Categorization_Using_Machine_Learning_Algorithms_K-Nearest_Neighbor_Support_Vector_Machine_and_Logistic_Regression_using_Customized_Dataset.

[2] *Classification of Book Genres by Cover and Title - cs229.Stanford.edu*. https://cs229.stanford.edu/proj2015/127_report.pdf.

[3] Ng, Ernest. "Keras, Tell Me the Genre of My Book." *Medium*, Towards Data Science, 27 May 2020, https://towardsdatascience.com/keras-tell-me-the-genre-of-my-book-a417d213e3a1.

[4] *Deepgenre: Deep Neural Networks for Genre Classification ... - Scott Liu*. https://scott-liu.com/assets/pdfs/deepgenre.pdf.

[5] "Diva." *Simple Search*, https://www.diva-portal.org/smash/.

[6] "Classifying 19th Century British Library Books Using Crowdsourcing and Machine Learning." *Training Our First Book Genre Classification Model - Classifying 19th Century British Library Books Using Crowdsourcing and Machine Learning*, https://living-with-machines.github.io/genre-classification/01_BL_fiction_non_fiction.html.