

Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

B.Tech. CE Semester – VI

Subject: (CE-621) System Design and Practice

Project Title:

Book Genre Prediction-

By:

Desai Priyal (Roll no: CE025 id: 17CEUOS073)

Ghata Saraiya (Roll no: CE035 id: 17CEUOS016)

Maliha Nan (Roll no: CE073 id: 17CEUOS109)

Guided by:

Prof. Jigar Pandya

Dr. Brijesh Bhatt



Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

**CERTIFICATE**

This is to certify that System Design Practice project entitled “Book Genre Prediction” is the bona fide report of work carried out by

1. **Desai Priyal (17CEUOS073)**

**2) Ghata Saraiya (17CEUOS016)**

**3) Maliha Nan (17CEUOS109)**

of Department of Computer Engineering, Semester VI, academic year 2019-20, under your supervision and guidance.

|  |  |  |
| --- | --- | --- |
| Guide | Guide | HOD |
|  |  |  |
| **Prof. Jigar Pandya** | **Dr. Brijesh Bhatt** | **Dr. C. K. Bhensdadia** |
| Assistant Professor | Associate Professor | Head of the Department |
| Department of Computer Engineering, | Department of Computer Engineering, | Department of Computer Engineering, |
| Dharmsinh Desai University, Nadiad. | Dharmsinh Desai University, Nadiad. | Dharmsinh Desai University, Nadiad. |

Contents

1. [Abstract 4](#_Toc38613864)

2. [Introduction 5](#_Toc38613865)

2.1 [Brief Introduction 5](#_Toc38613866)

2.2 [Problem Definition 5](#_Toc38613867)

2.3 [Libraries used 6](#_Toc38613868)

3. [Related Works/Literature Survey 7](#_Toc38613869)

4. [Proposed Approach 9](#_Toc38613870)

5. [Implementation 10](#_Toc38613871)

5.1. [Working with the CMU Book Summary Dataset 10](#_Toc38613872)

5.1.1. [Cleaning the data 11](#_Toc38613873)

5.1.2. [Data Augmentation 12](#_Toc38613874)

5.1.3. [Classification 16](#_Toc38613875)

5.1.4. [Testing and Accuracy 17](#_Toc38613876)

5.[2.Blurb Genre Collection 18](#_Toc38613877)

5.2.1. [Data distribution 18](#_Toc38613878)

5.2.2. [Data extraction and cleaning 19](#_Toc38613879)

5.2.3. [Multi Label Classification 21](#_Toc38613880)

5.2.4. [Multi Feature classification 26](#_Toc38613881)

6. [Experiments 29](#_Toc38613882)

7. [Conclusion 35](#_Toc38613883)

8. [Limitations and future extensions 36](#_Toc38613884)

9. [Bibliography 37](#_Toc38613885)

Abstract

This project examines automated genre classification in literature. The approach described uses text based comparison of book summaries as the primary feature to predict the genres of the book.

Genre means a type of art, literature, or music characterized by a specific form, content, and style. For example, literature has four main genres: poetry, drama, fiction, and non-fiction. All of these genres have particular features and functions that distinguish them from one another. Hence, it is necessary on the part of readers to know which category of genre they are reading in order to understand the message it conveys, as they may have certain expectations prior to the reading concerned.

This makes automatically generating genre labels a potentially useful tool in sorting unmarked text collections or searching the web. Our project deals with this problem by applying different text classification techniques and models to find the best solution for the same.

Introduction

## Brief Introduction

The aim of the project is to predict the genre of a book using python programming language. This project represents our study of the classification of books based upon their summaries. Our hypothesis is that it is possible to classify books based on the word content of their written summaries. Once the model has been trained using a dataset, it will be used to classify new books into predefined genres. One end goal is to enable easier classification of books, and let people know about possible genres, or the overlap of genres between books. This might allow books to be easily identified as more than one genre type.

Classification of literary works is significantly different from normal text classification. One big reason for this is length, as books are generally much longer than most other text mediums. That is why we will be working with the summaries of books in place of the entire text.

## Problem Definition

Often times, books fall under more than one genre. For instance, a book could be a romantic mystery in which a male detective gets together with the woman who hired him all the while trying to find her dead husbands killer. Should this book be marked under the primary genre of mystery, or romance? What qualifies the book to belong more to one genre than another? Without a clear metric to decide how much a book belongs to a specific genre, many books end up poorly classified or shoved under the super-genre heading of fiction. This is why it is important to find a way to classify books and their degree of relativity to a given genre.

So the main problem revolves around identifying the major genres to which a book belongs as well as to measure the accuracy of such predictions.

## Libraries used

**Nltk:**

A suite of libraries and programs for symbolic and statistical natural language processing for English.

**Pandas:**

A software library written for the Python programming language for data manipulation and analysis.

**Numpy:**

NumPy is a library for adding support for large, multi-dimensional arrays and matrices, along with high-level mathematical functions to operate on these arrays.

**Scikit-learn:**

Scikit-learn is a machine learning library. It features various classification, regression and clustering algorithms including support vector machines.

**Keras:**

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow.

**Tensorflow:**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks.

**Scikit-multilearn:**

Scikit-multilearn is a library for multi-label classification that is built on top of the well-known scikit-learn ecosystem.

**Matplotlib:**

Matplotlib is a plotting library for the Python programming language and NumPy.

**Seaborn:**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Related Works/Literature Survey

Before starting with our project we went through various works of researchers and programmers to find a suitable approach for the problem at hand. It was useful to read the work of those who approached a similar topic. One such source was the work of Emily Jordan on Automated Genre Classification in Literature. It provided us an in depth understanding of the complexity of predicting genres and how multiple genres may overlap. Another work we approached was an article by Susan Li on text classification model comparison and selection. It acted as our introductory guide to Natural Language processing, various classifiers and embedding that we could use.

We also went through the NLTK book, to gain better understanding of tools that we could use to process text. It provided us with ways to clean the text, and understand feature extraction.

Another such avenue was the paper Genre Identification and the Compositional Effect of Genre in Literature authored by Joseph Worsham and JugalKalita. Their study focused on specific genre classification such as romance or adventure stories. It was useful to read the work of those who approached a similar topic. When discussing features, Worsham noted that using word frequencies alone is usually inadequate for the purposes of genre classification. Word frequency is defined as how many times a word appears in a section, which is in this case, an entire book. This lead to us including additional features beyond a bag of words to our model.

We also considered a model based on predicting genre only by the title of the book. This model was created by a Github user by the name Akshay Bhatia. Like the paper by Worsham and Kalita, Bhatia’s model focused on classifying works into several genres, such as adventure and romance. However,

Bhatia’s success in predicting genres solely off book titles was useful to us when we began refining our model to increase our prediction accuracy.

We also approached various tutorials on text classification and the workings of different classifiers. We began with getting an understanding of NLP, followed by practising various libraries like nltk, pandas, scikit-learn, scikit-multilearn, matplotlib, seaborn, keras and tensorflow.

The articles available on *TowardsDataScience* acted as our guide. After experimenting with classifiers we decided to create our neural network model using keras. The article *Keras for Multi-label Text classification* by AmanSawarn helped us start our dealings with keras and understand the working of the Neural networks. We also followed courses on Udemy and Coursera for the same.

Proposed Approach

First, we found it necessary to understand Natural Language Processing and the process of creating classification models. Once we did that we broke down our approach to the problem in the following subtopics:

* Dataset identification and balancing.
* Dataset cleaning.
* Creating vectors of the book summaries.
* Training a model to use these vectors to predict genre of a new book.

We established that we will be working with summaries of book in place of the entire text owing to the increase in complexity of large texts and the resources required for the same.

We defined the features of a genre as words or phrases that are specific to that genre. For example, “travelling through time” may be used in science fiction and is specific to that genre. So in case a new book has this phrase, the chances of it belonging to science fiction will be higher.

It is also possible that the same feature may belong to multiple genres, for example “time stopped” could belong to romance as well as science fiction. This meant that features could also be overlapping.

Hence, it is important to maintain the context of words and not solely depend on individual word or phrase as features. Also the frequency of a word did not provide a suitable metric. We decided to use a TF-IDF technique to quantify the words in the summary. N-grams were also an important concept for preserving the context of words.

For training the model, we would approach various classifiers available through libraries like scikit-learn and how the accuracy was affected by changing various parameters.

To provide the interface, a simple flask page would be designed that makes calls to our predictor.

With this plan in mind we went forward with the implementation.

Implementation

**Dataset:**

The first task was to find a suitable dataset. We found two datasets:

* The CMU Book Summary Dataset
  + This dataset contains plot summaries for 16,559 books extracted from Wikipedia, along with aligned metadata from Freebase, including book author, title, and genre.
* The Blurb Genre Collection
  + This dataset contains blurbs (advertising descriptions of books) for 91,982 books extracted from Penguin Random House along with aligned metadata.

1. **Working with the CMU Book Summary Dataset**

There is a total of 179 unique genres in the dataset with a very uneven distribution. In order to balance the distribution, we only keep the mainstream genres, and remove books belonging to genres like “Anti-war”.

The distribution of number of books per each genre still seems to be imbalanced.

We will be using Data augmentation is further steps to make the distribution more even.

The count of the retained genres and their distribution is as follows.

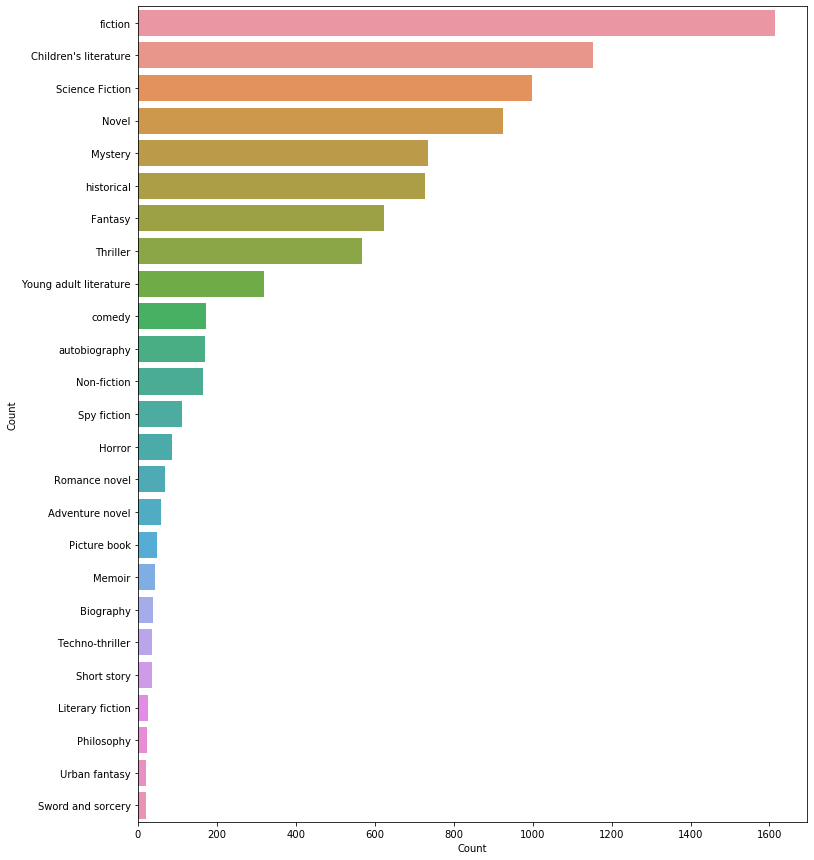


Figure 1. Distribution of CMU Book Summary Dataset

### Cleaning the data

We use the Natural Language Toolkit to clean our summaries. Steps followed for pre-processing:

1. Converting to lowercase
2. Removing the punctuation marks
3. Removing Stop words
4. Converting numbers to words
5. Stemming

Example sentence before preprocessing:

*Hazel Grace Lancaster, a 16-year-old with thyroid cancer that has spread to her lungs, attends a cancer patient support group at her mother's behest.*

After preprocessing:

*hazel grace lancastsixteen year old thyroid cancer spread lung attend cancer patient support group mother behest*

Psuedo code:



### Data Augmentation

Data augmentation is commonly used in computer vision. In vision, you can almost certainly flip, rotate, or mirror an image without risk of changing the original label.

But there is some difference when we are working with text, especially summaries. We use one simple operation for data augmentation, such that it does not change the genre of the book.

**Synonym Replacement:** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.



Based on the distribution, we will increase records of those genres that have a lower number of records.



Data frame:



The following is the frequency distribution after augmentation, note that it is more balanced than before.

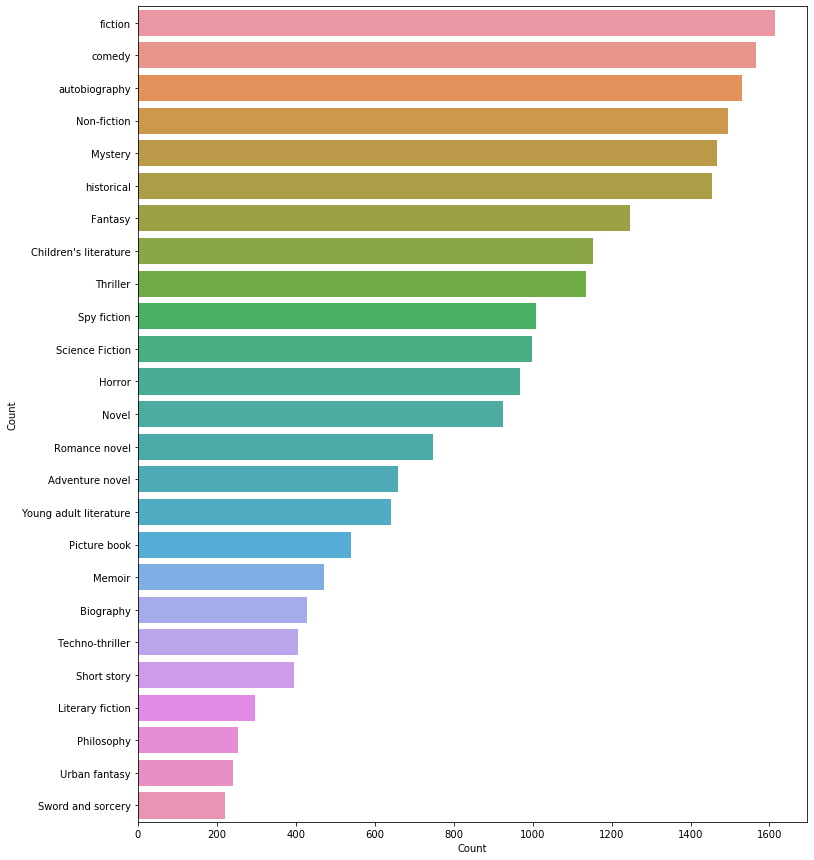
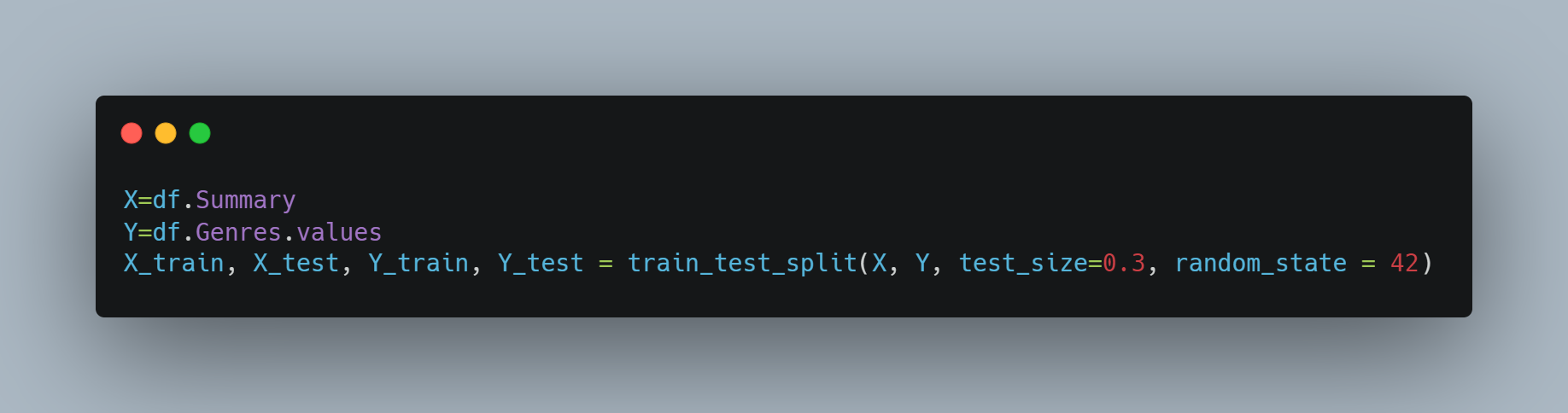


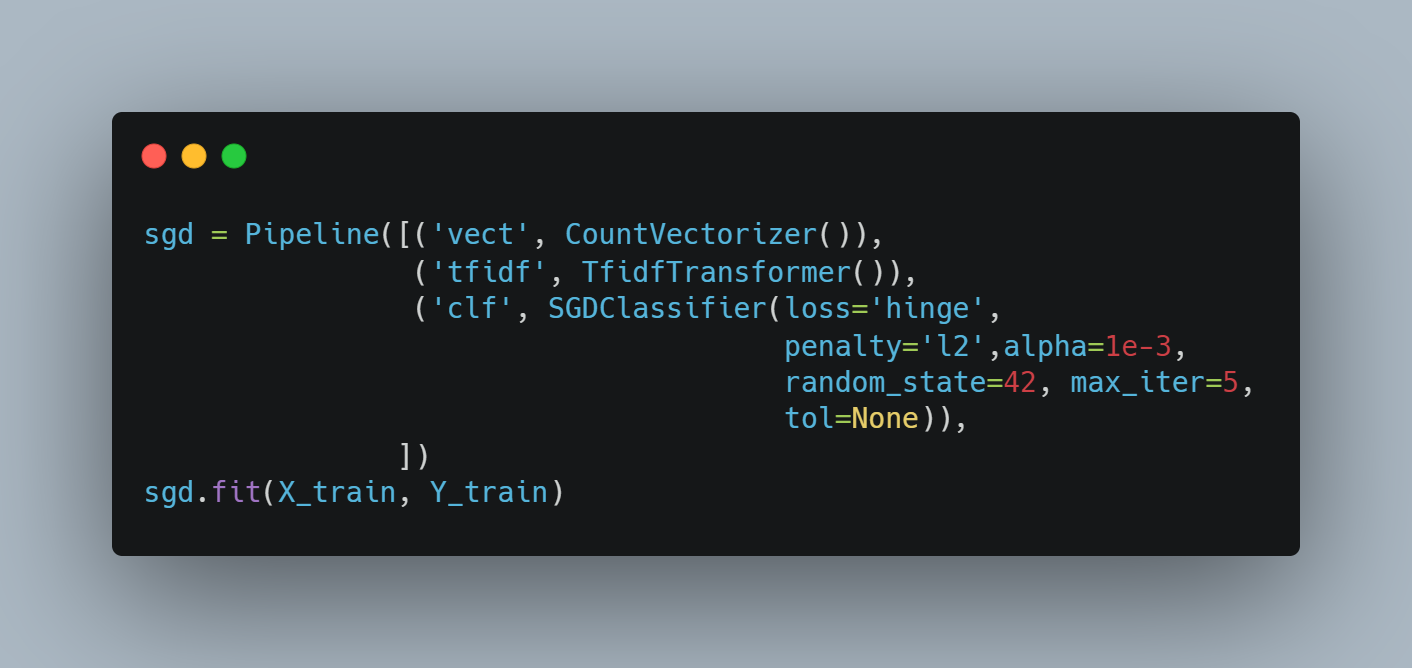
Figure 2. Distribution of CMU dataset after data augmentation

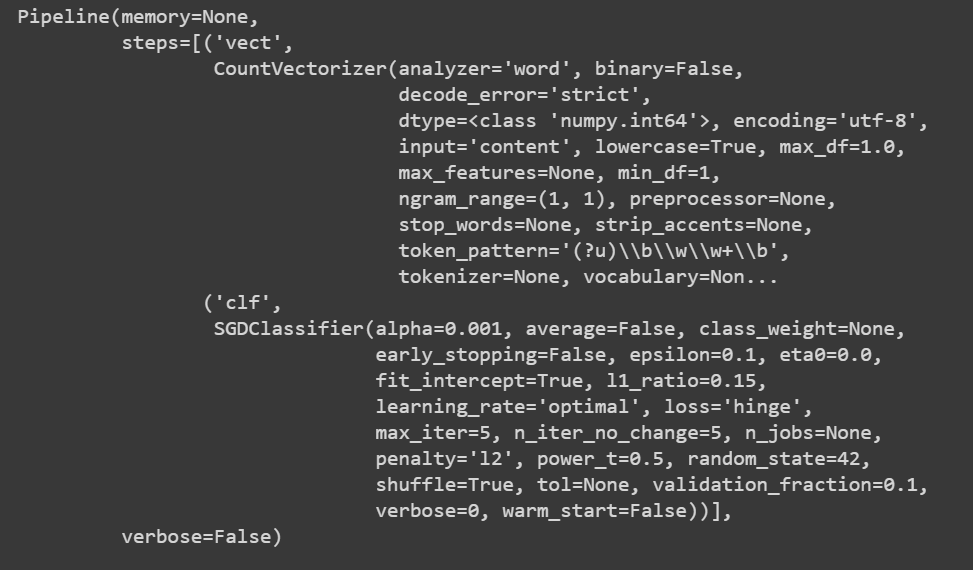
Splitting the training and test data



**Classification**

Creating a classifier and fitting our dataset. We have used a pipeline which vectorises the data before it is fed to the classifier.

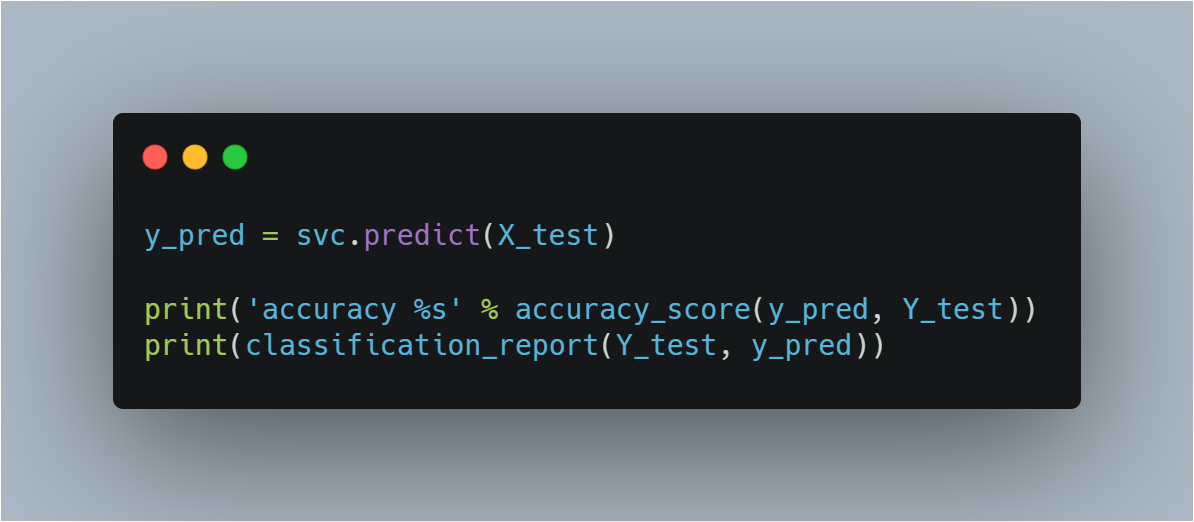




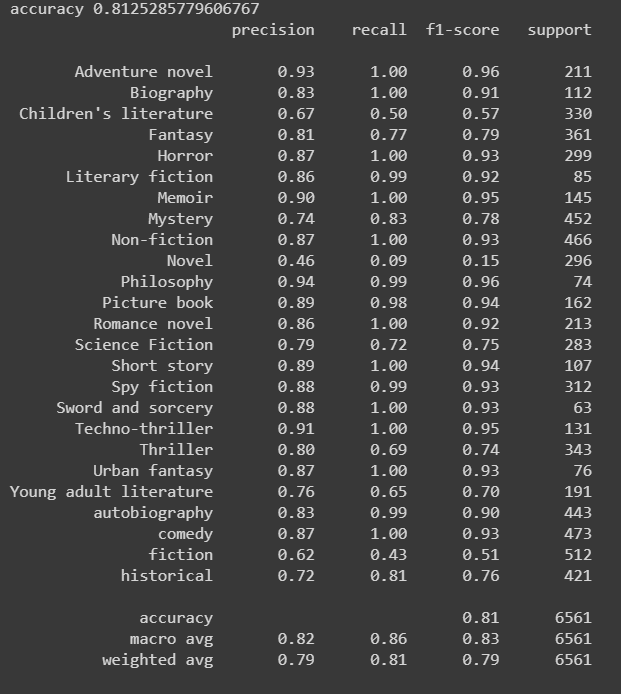
### Testing and Accuracy

Using predict:

Predict function predicts multi-class targets using underlying estimators.



**Output**

****

Hence, an accuracy of 0.81 is achieved for single label genre prediction using SGD classifier (Stochastic Gradient Descent). We get balanced values for precision and recall for all genres except those genres with lower number of records.

But a single genre is often insufficient for books, we require multiple genre predictions for the model to be useful.

1. **Blurb Genre Collection**

**Data distribution**

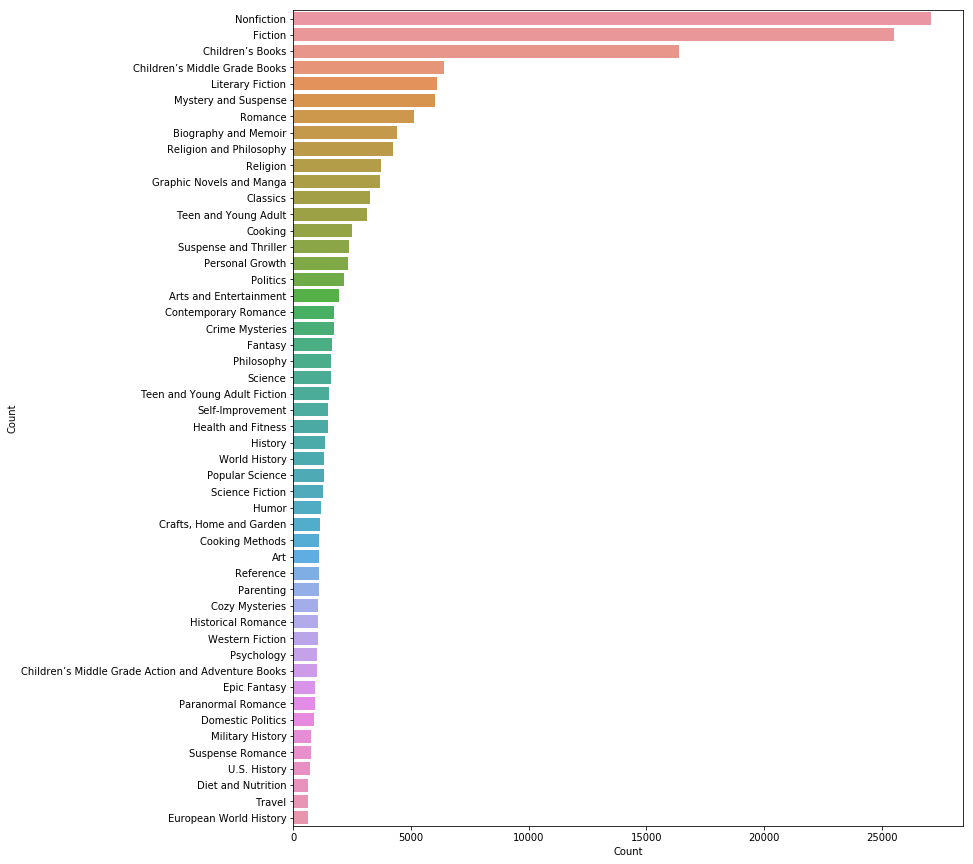
Distribution of data:

Figure 3. Distribution of Blurb Genre Collection

Each summary is assigned multiple genres, following is the distribution for the same.

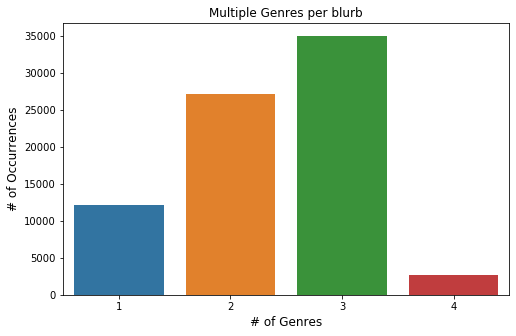
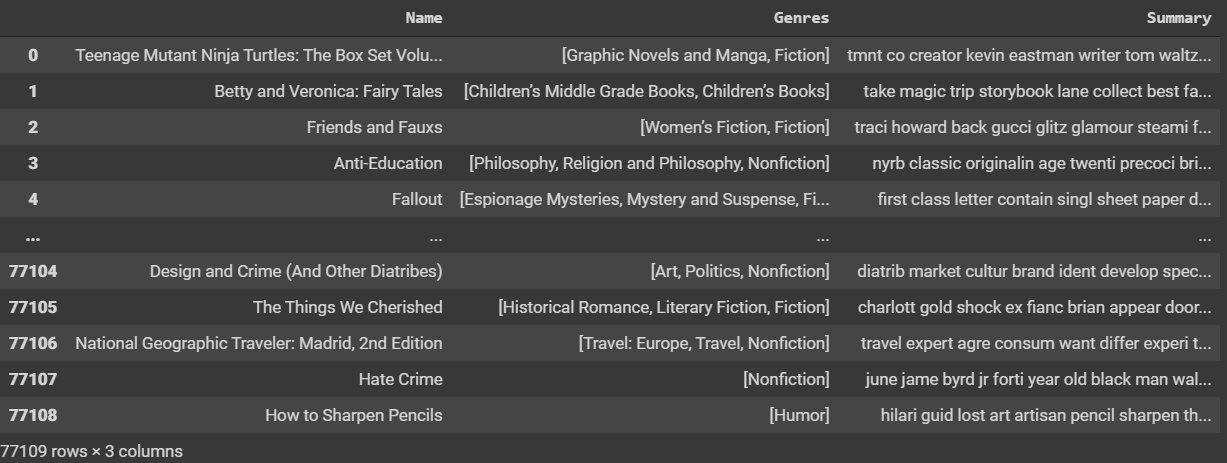


Figure 4. Blurb Genre Collection: Number of genres per summary

**Data extraction and cleaning**

The data is available in XML format so the first task was to extract the metadata from the file to create a data frame. Each blurb is categorized into one or multiple categories. The categories are structured hierarchically. The **minimum code policy**requires the assignment of at least one category to each document of the collection. The **hierarchy policy**ensures that every ancestor of a document's label is assigned as well. To make the access to data easier we created lists of genres to which each blurb belongs.



****

### Multi Label Classification

[Multi-label](https://en.wikipedia.org/wiki/Multi-label_classification) classification is a generalization of multi-class classification which is the single-label problem of categorizing instances into precisely one of more than two classes, in the multi-label problem there is no constraint on how many of the classes the instance can be assigned to i.e. there could be one, two or many labels in the output data used for training.

Metric used:

[**F1 Score**](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html): F1 score is calculated using the harmonic mean of precision and recall.

F1 Score = 2 \* (precision \* recall) / (precision + recall)

This F1 score is micro averaged to use it as a metric for multi-class classification. It is calculated by counting the value of true positives, false positives, true negatives, and false negatives. All the predicted outputs, in this case, are column indices and are used in sorted order by default.

**Modifying the dataset for Multi label:**

Although a list of sets or tuples is a very intuitive format for multilabel data, it is difficult to process. We modify the dataset to create a binary matrix, such that each genre is a separate column. There are 139 unique genres. If a blurb belongs to a genre the value for that column will be 1 otherwise 0. Another approach uses the multilabel\_binarizer provided by scikit-learn. This transformer converts between the intuitive format and the supported multilabel format: a (samples x classes) binary matrix indicating the presence of a class label.

We have followed both approaches to get similar outputs.

Now we use a TF-IDF vectorizer from the scikit-learn library to vectorise our summaries. We can also specify the ngram length. A TfidfVectorizerconverts a collection of raw documents to a matrix of TF-IDF features which can be fed to classifiers.



After vectorising, we will use a One vs. Rest classifier for multi label prediction. Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n\_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.

This strategy can also be used for multilabel learning, where a classifier is used to predict multiple labels for instance, by fitting on a 2-d matrix in which cell [i, j] is 1 if sample i has label j and 0 otherwise.

In the multilabel learning literature, OvR is also known as the binary relevance method.

We need to provide an estimator as a parameter to the OneVsRestClassifier, an estimator is a object implementing [fit](https://scikit-learn.org/stable/glossary.html#term-fit) and one of [decision\_function](https://scikit-learn.org/stable/glossary.html#term-decision-function) or [predict\_proba](https://scikit-learn.org/stable/glossary.html#term-predict-proba). Two estimators can be used in our context:

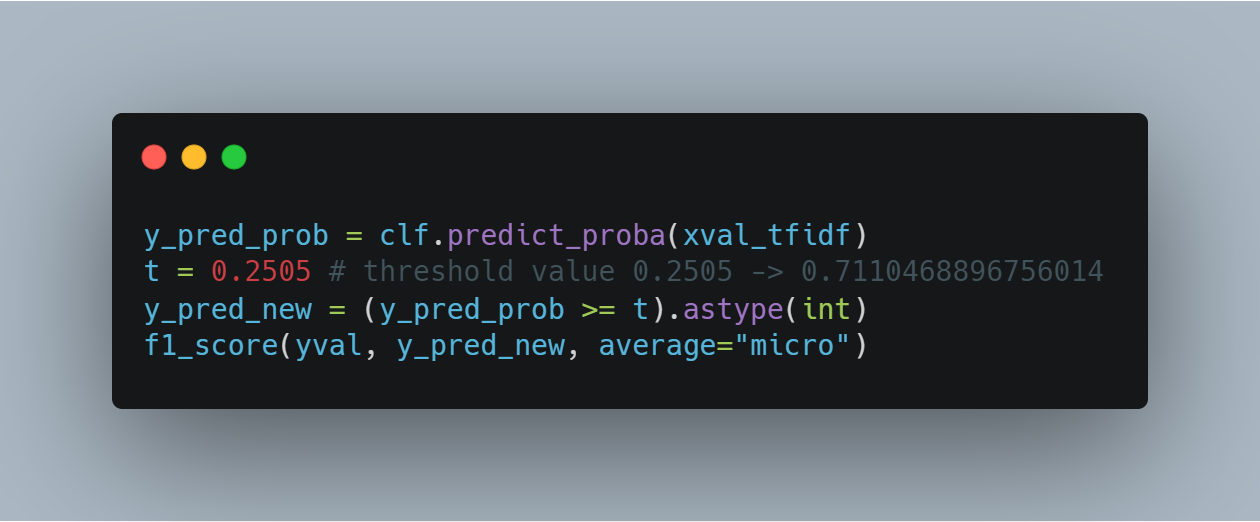
* Linear Support Vector Machine
* Logistic Regression

Now we train the classifier using the fit function.



**Predicting the genres**

Using predict\_proba: Probability estimates.



Output: 0.7136495754642689

We achieve an F1 score of 0.713 using this model.

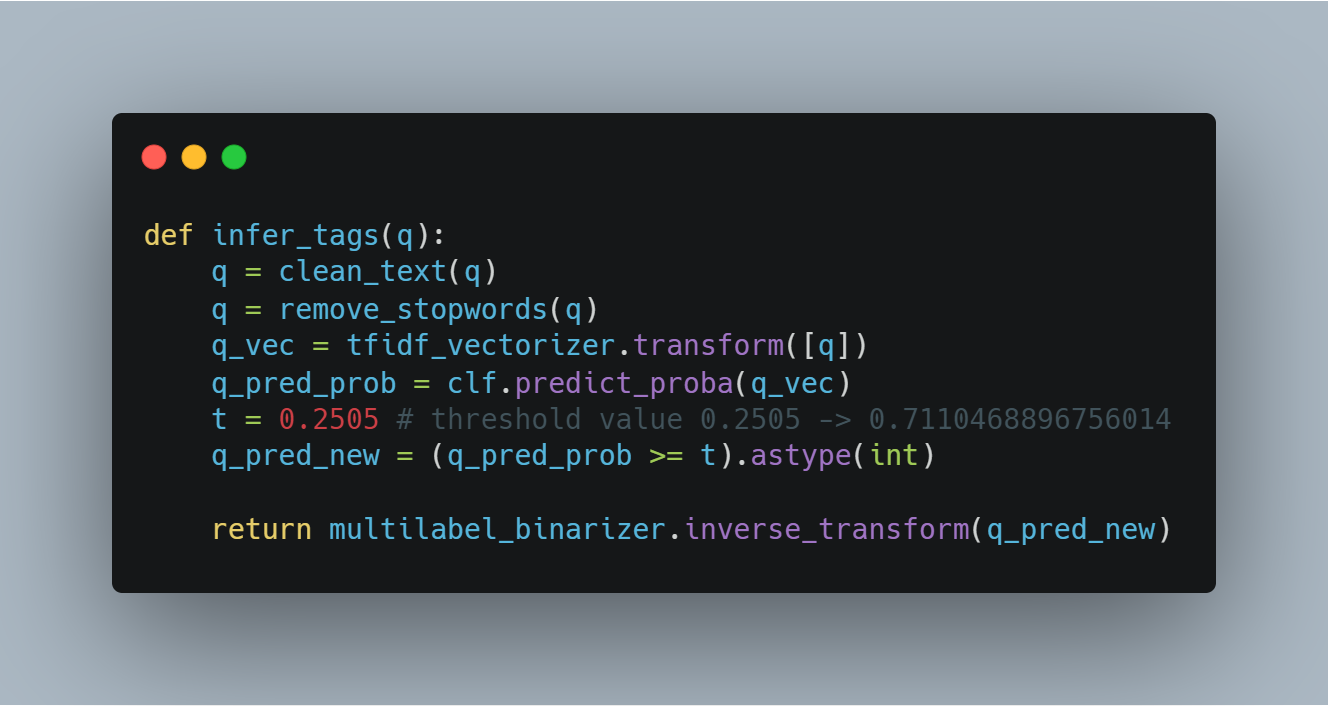
The returned estimates for all classes are ordered by label of classes.

Note that in the multilabel case, each sample can have any number of labels. This returns the marginal probability that the given sample has the label in question. For example, it is entirely consistent that two labels both have a 90% probability of applying to a given sample.

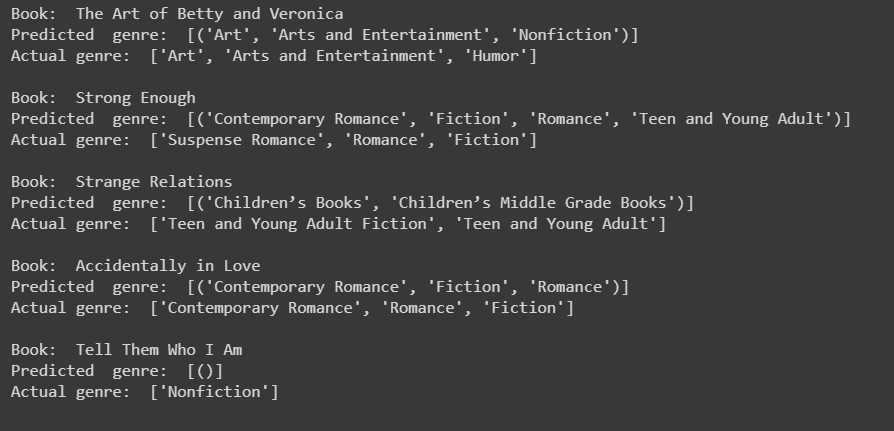
Printing the predicted genre classes:

We use a threshold value to determine whether or not a book belongs to that genre. Applying different thresholds to the same prediction helps us identify the optimal value which is 0.25 in our case. A very large or a very small value of threshold gives a lower value of F1 metric score because when tags are chosen based on a lower threshold value, too many tags get chosen which reduce the F1 metric score, while when the threshold value gets very large, almost no tags get chosen and thus reducing the performance metric.

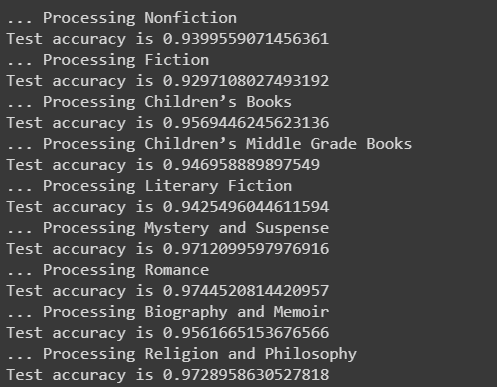
We then use inverse\_transform to get string values of the classes from the binarizer.



The actual label and predicted labels:



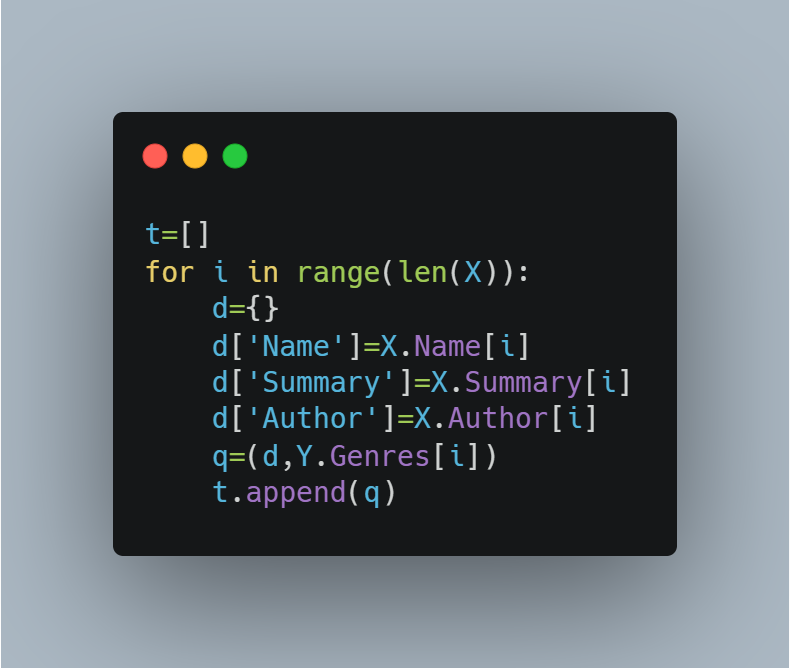
Here we also get the accuracy for correct predictions of individual genres.



### Multi Feature classification

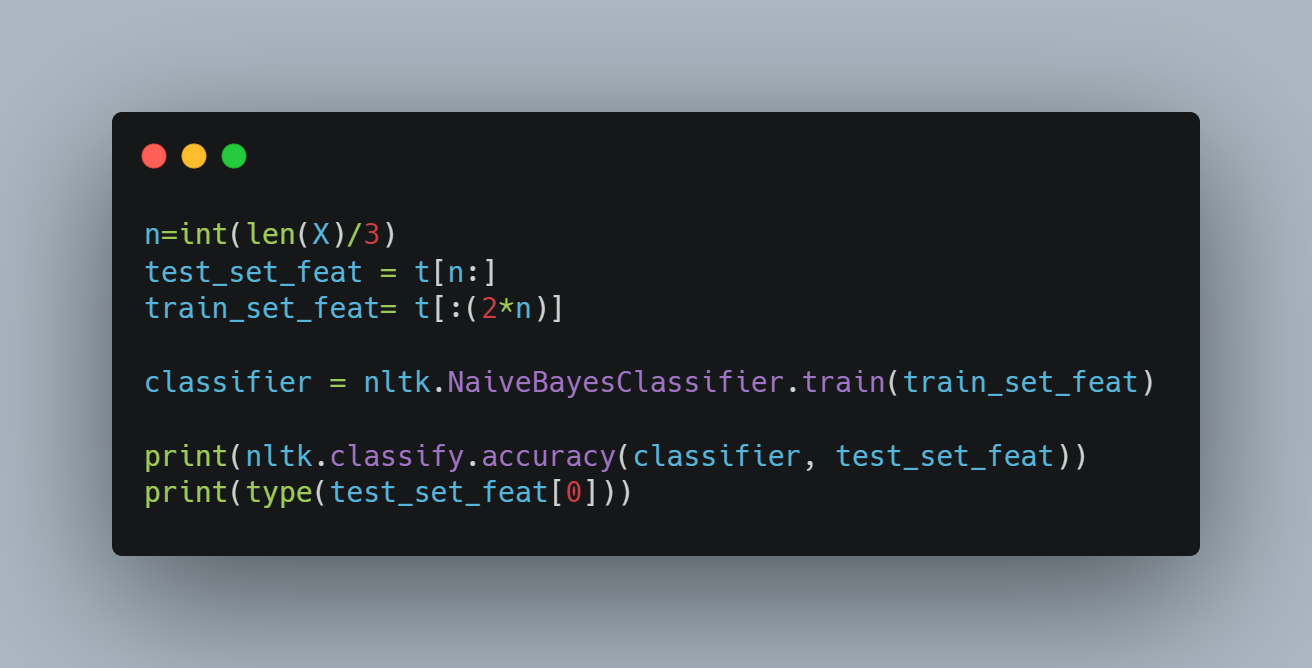
In our general model we have only used the summary of the book to predict its genre. But other features like name of the book and author may also affect the genre. Hence, we apply multiple features to the classification.

We first convert the rows to a dictionary.



Now we apply various classifiers.

The Naïve Bayes classifier gives the best accuracy. Naïve Bayes classifiers, a family of classifiers that are based on the popular Bayes’ probability theorem, are known for creating simple yet well performing models, especially in the fields of document classification. In our case as well Naïve Bayes proved to be the most efficient.



Output: 0.7204800995992686

Hence we achieve an accuracy of 0.72 using the multi feature model. We also found relevant predictions when we used this model to predict genres of random summaries outside the dataset.

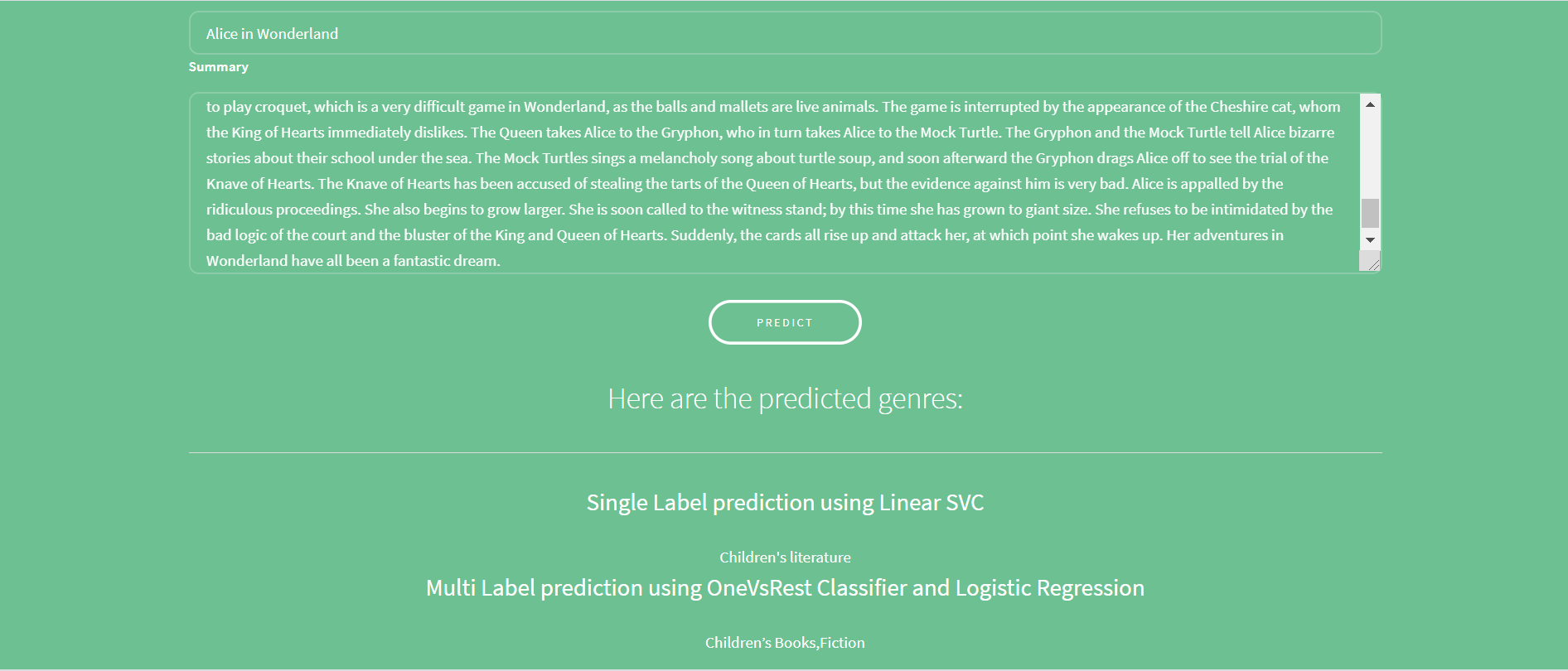
We store this model into pickle files for further use.



For providing the interface to users we have created a flask web application. When the user enters a summary an AJAX request is sent to the server and the response is displayed by parsing the JSON data received.



**Preview:**



Experiments

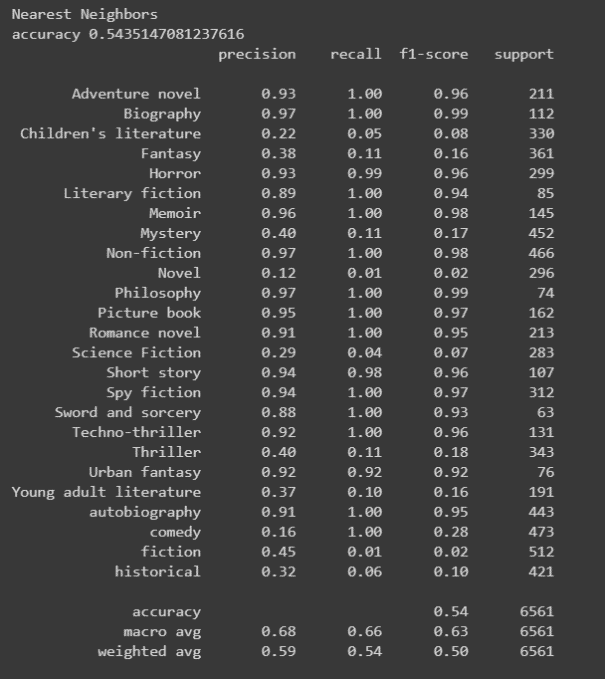
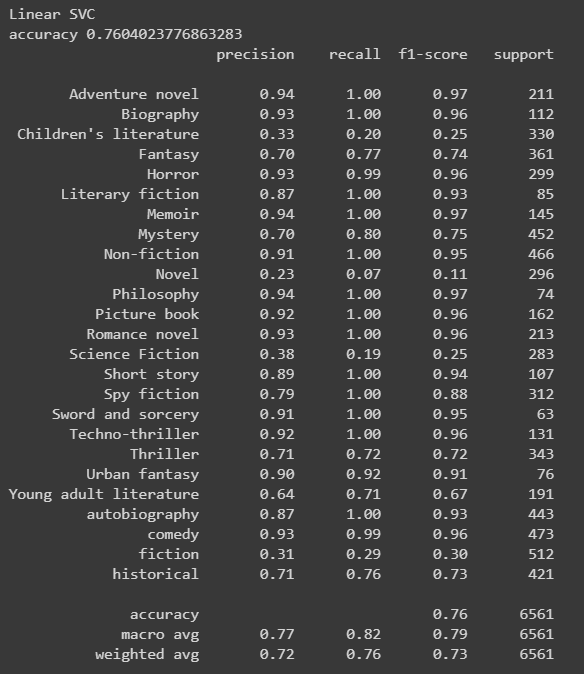
We tried using different classifiers on different datasets to get the best results.

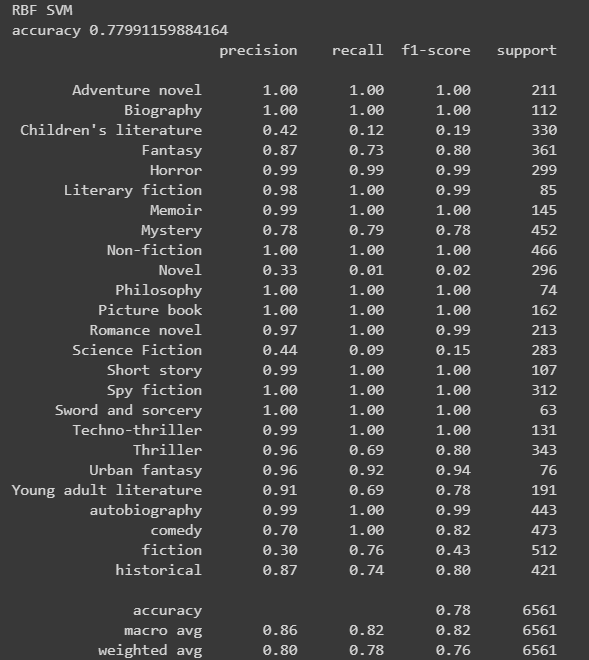
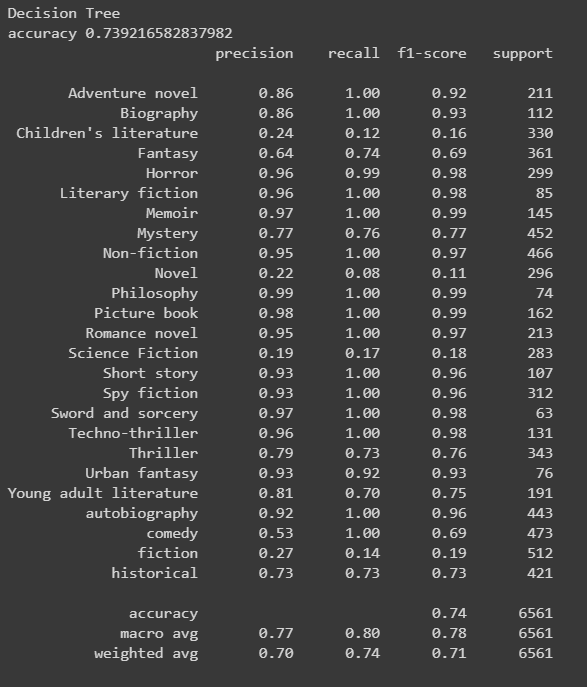
CMU Book Summary Dataset

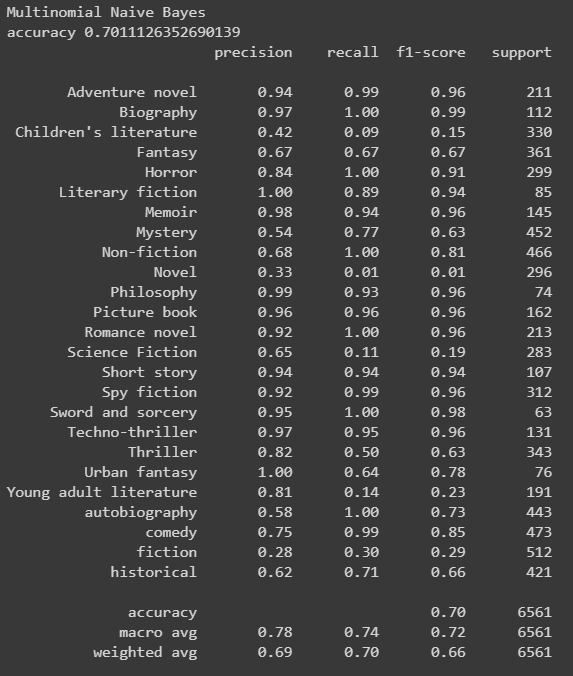
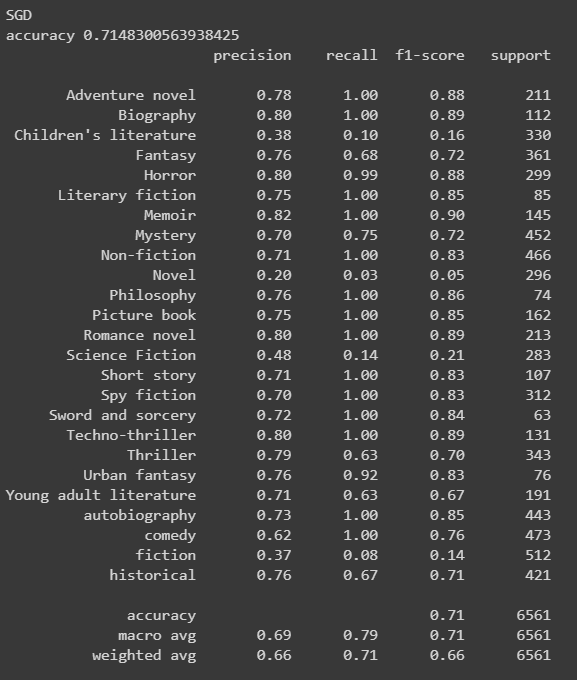
* Applying different classifiers on augmented data



Classification Reports:

Here is a boxplot comparing the accuracies of different classifiers.

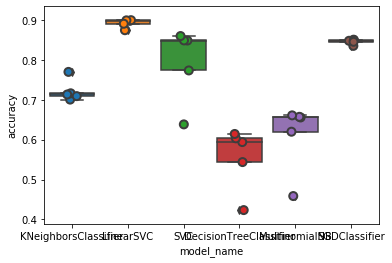


Figure . Accuracy vs. Classifier

Following is the accuracy data:

**Model name Accuracy**

DecisionTreeClassifier 0.555763

KNeighborsClassifier 0.721846

LinearSVC 0.892405

MultinomialNB 0.610499

SGDClassifier 0.845809

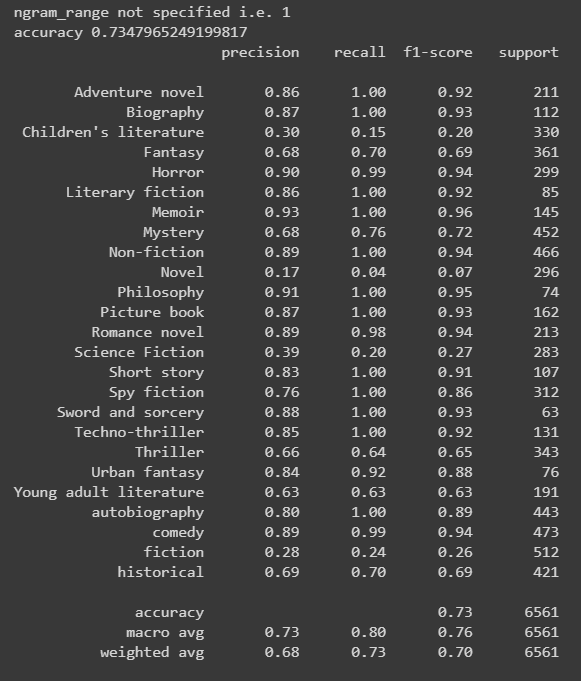
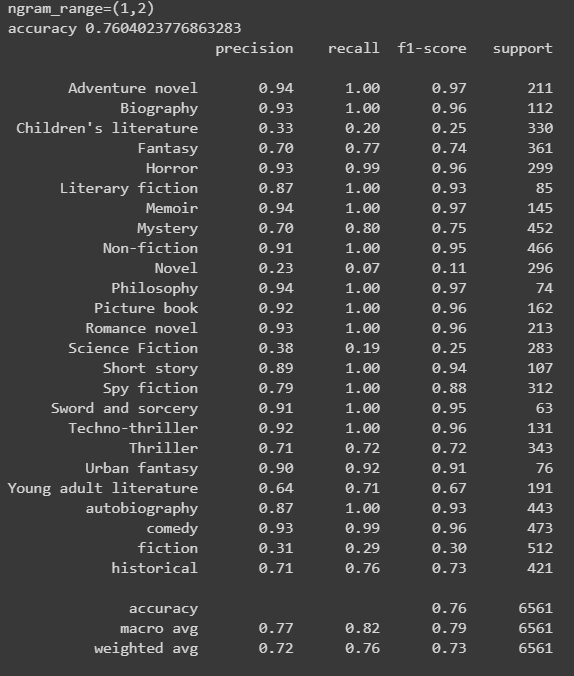
SVC 0.793863

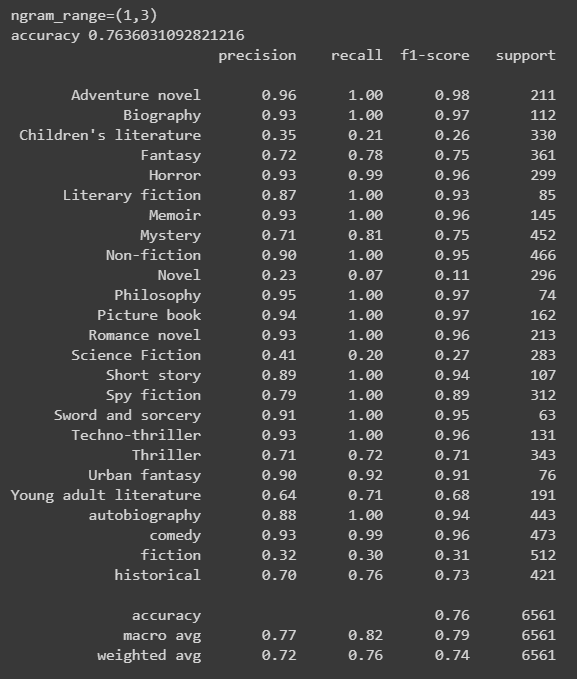
* Applying different Ngram ranges to the vectorizer

We apply ranges: 1, (1,2) and (1,3). On increasing the range further, we encounter memory error due to usage of all available memory. Hence we limit ourselves to maximum 3 word long ngram. We now analyse the impact of ngrams.



Classification Reports:



Other experiments conducted:

* Application of different classifiers to the original dataset. We have noted the positive impact of data augmentation.

Accuracy of linear SVC on original dataset: 0.41

Accuracy of linear SVC on augmented dataset: 0.76

* We also tried data augmentation with the Blurb collection and Multi label classification using One vs Rest classifier with Logistic Regression as an estimator.

Accuracy of linear SVC on original dataset: 0.68

Accuracy of linear SVC on augmented dataset: 0.77

* Applying multiple features to the blurb genre collection.

Accuracy using the features Name and Summary: 0.52

Accuracy using the features Author and Summary: 0.47

Accuracy using the features Name, Author and Summary: 0.72

Conclusion

We were able to train efficient models that had high accuracy. The model performed well on random inputs outside the dataset as well. We tried pursuing different approaches to solve the same problem, including single label classification, multi label classification and multi feature classification.

We have come to the understanding that it is important to maintain the context of the words in the summary as well as to ensure that we are accounting for overlaps between various genres. This was achieved using the multi label model which also used Ngrams.

Classification of books into genres is a complex task, and our model works well.

Limitations and future extensions

Our model is unable to predict “Non-fiction” books, because it does not have information related to the real life events on which the book may be based. To the model, the story of Mahatma Gandhi may just be another fictional work on peace, as it does not know about the Indian history. In this manner there are several other genres that face this limitation.

The current model can further be extended in future by using word2vec embedding on the summary and analysing it’s effect. Another possibility is to use Convolutional Neural Networks in an efficient manner to optimize the predictions.

Another possibility is to use the text of the book and generate the respective summary using abstractive summarization. This will overcome the need to supply summary of the book, we can simply use the entire text to predict the genre.

There are various other enhancements that may be possible.

Bibliography

[CMU Book Summary Dataset | Kaggle](https://www.kaggle.com/ymaricar/cmu-book-summary-dataset)

[Book-Summary-Genre-Prediction | Kaggle](https://www.kaggle.com/iamhungundji/book-summary-genre-prediction)

[TF-IDF from scratch in python on real world dataset.](https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089)

[Multi-Class Text Classification Model Comparison and Selection](https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568)

[BOOK GENRE PREDICTOR – mc.ai](https://mc.ai/week-ii-book-genre-predictor/)

[Genre Identification and the Compositional Effect of Genre in Literature](https://www.aclweb.org/anthology/C18-1167.pdf)

[Data Augmentation Techniques in Natural Language Processing](https://towardsdatascience.com/these-are-the-easiest-data-augmentation-techniques-in-natural-language-processing-you-can-think-of-88e393fd610)

[Beyond Accuracy: Precision and Recall - Towards Data Science](https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c)

[scikit-learn Tutorials — scikit-learn 0.22.1 documentation](https://scikit-learn.org/stable/tutorial/index.html)

[An Introduction to N-grams: What Are They and Why Do We Need Them?](https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/)

[Simple Text Classification using Keras Deep Learning Python Library](https://www.opencodez.com/python/text-classification-using-keras.htm)

[Guide to the Sequential model - Keras Documentation](https://keras.io/getting-started/sequential-model-guide/)

[Multi-Label Classification(Blog Tags Prediction)using NLP](https://medium.com/coinmonks/multi-label-classification-blog-tags-prediction-using-nlp-b0b5ee6686fc)