

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:

Priyal Desai (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. **Priyal Desai (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 | Professor and  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. |

Table of Contents

1. [Abstract 4](#_Toc39392000)

2. [Introduction 5](#_Toc39392001)

2.1 [Brief Introduction 5](#_Toc39392002)

2.2 [Problem Definition 5](#_Toc39392003)

2.3 [Libraries used 7](#_Toc39392004)

2.4 [Timeline diagram for the project 8](#_Toc39392005)

3. [Related Work/Literature Survey 10](#_Toc39392006)

4. [Proposed Approach 11](#_Toc39392007)

5. [Prerequisites 12](#_Toc39392008)

5.1 [Natural Language Processing: 12](#_Toc39392009)

5.2 [Text Processing 12](#_Toc39392010)

5.2.1 [Data Preprocessing 12](#_Toc39392011)

5.2.2 [Feature extraction 13](#_Toc39392012)

5.2.3 [Choosing ML Algorithms 15](#_Toc39392013)

6. [Implementation 19](#_Toc39392014)

6.1 [Datasets 19](#_Toc39392015)

6.2 [Working with the CMU Book Summary Dataset 19](#_Toc39392016)

6.2.1 [Cleaning the data 21](#_Toc39392017)

6.2.2 [Data Augmentation 22](#_Toc39392018)

6.2.3 [Classification 25](#_Toc39392019)

6.2.4 [Testing and Accuracy 27](#_Toc39392020)

6.3 [Blurb Genre Collection 31](#_Toc39392021)

6.3.1 [Multi Label Classification 35](#_Toc39392022)

6.3.2 [Multi Feature classification 40](#_Toc39392023)

7. [Software Version Deployment Document 43](#_Toc39392024)

8. [Experiments 50](#_Toc39392025)

9. [Conclusion 56](#_Toc39392026)

10. [Limitations and future extensions 57](#_Toc39392027)

11. [Bibliography 58](#_Toc39392028)

Abstract

This project attempts to predict the genre of a book. The approach described uses text based comparison of book summaries as the primary feature to predict the genres of the book. We have also explored the effect of multiple features like name and author.

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 genre prediction a potentially useful tool in classification of unmarked books. 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 called The Book Thief, is a historical fiction where the plot is set in Nazi Germany but the story of the young girl is fictional. Should this book be marked under the primary genre of history, or fiction? 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. This is why it is important to find a way to classify books and their degree of belonging 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.

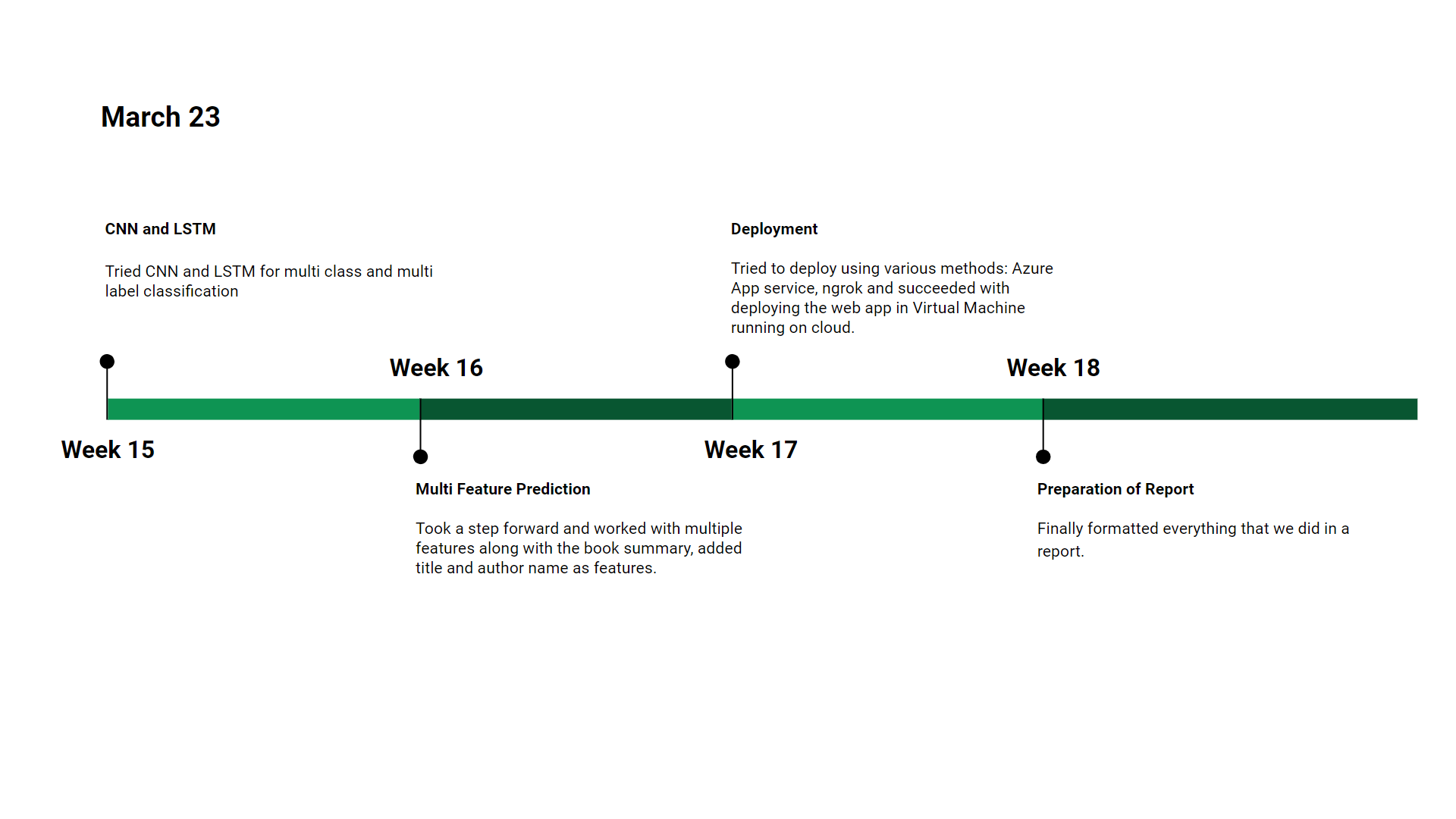
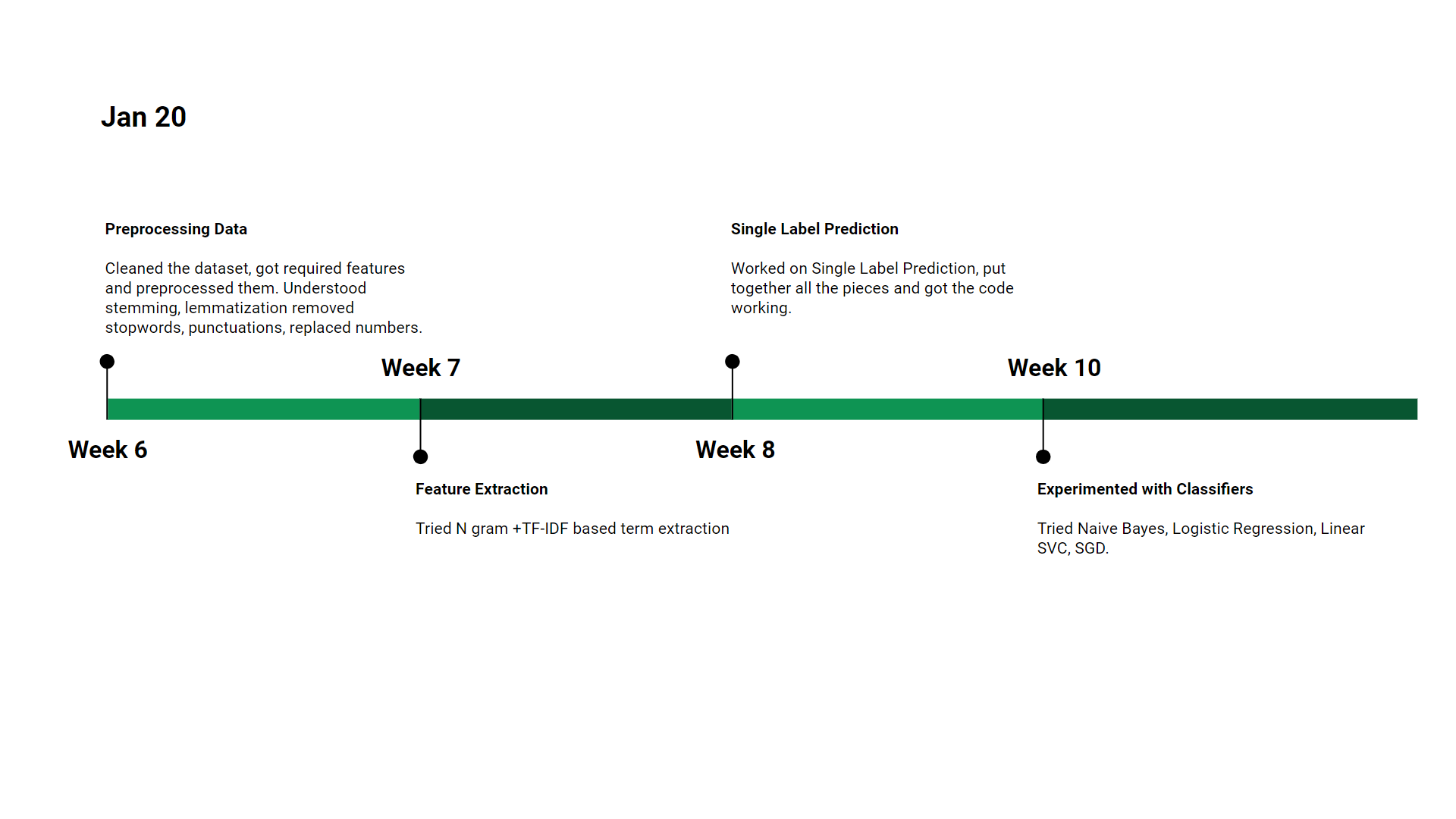
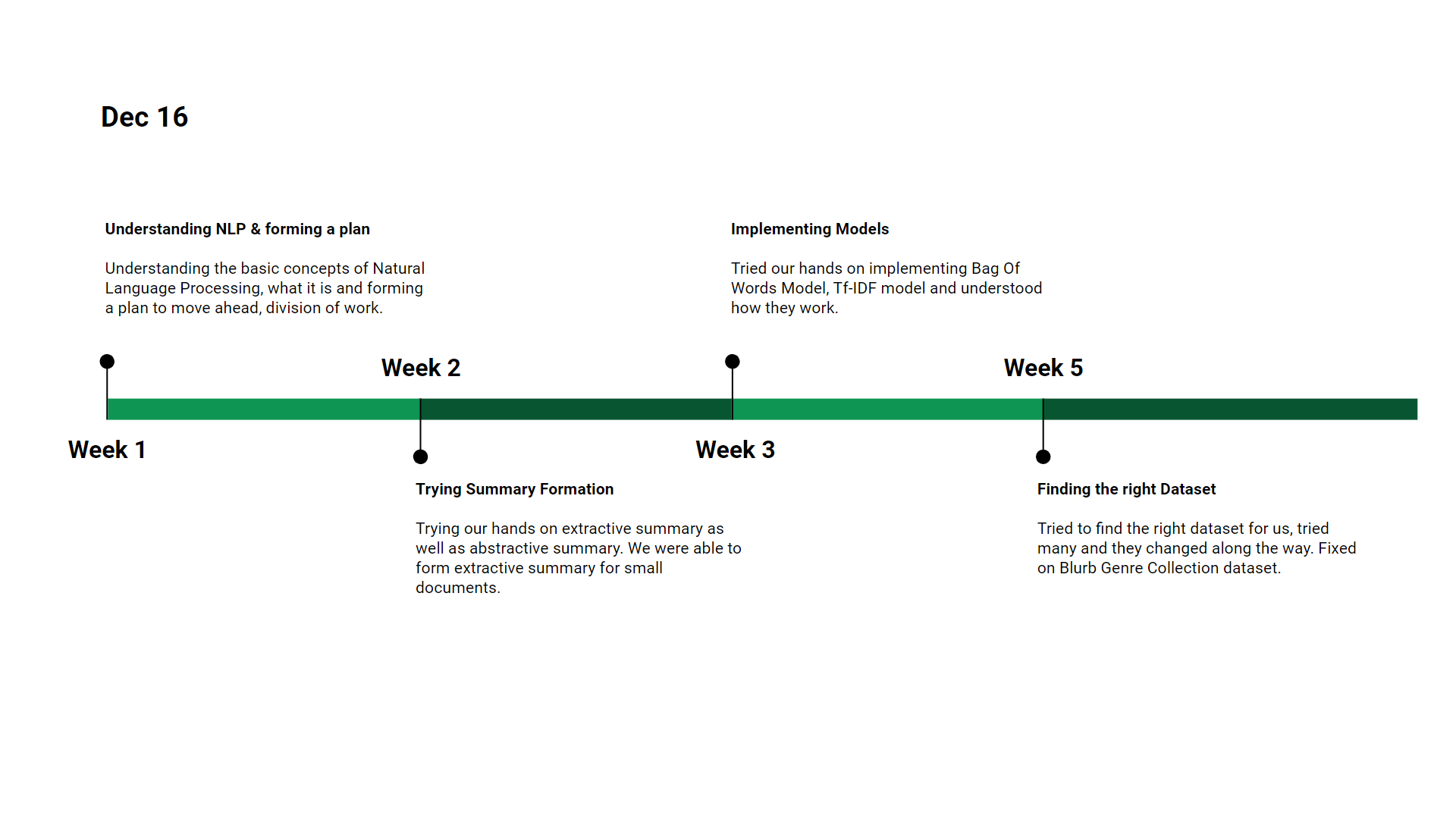
**Matplotlib:**

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.

## Timeline diagram for the project



Related Work/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 (Jordan, 2012). 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 (Li, 2018). 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 (Natural Language Processing with Python, n.d.), 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 (Kalita, n.d.). 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, they 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 exploring different embedding and models.

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 (Sawarn, 2019). 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. Another thing we decided was using a supervised model for classification.

We defined the features of a genre as words or phrases that are specific to that genre. For example, “time-travel” 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.

Prerequisites

## Natural Language Processing:

Natural language processing (NLP) is a field of artificial intelligence in which computers analyze, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

Apart from common word processor operations that treat text like a mere sequence of symbols, NLP considers the hierarchical structure of language: several words make a phrase, several phrases make a sentence and, ultimately, sentences convey ideas. By analyzing language for its meaning, NLP system can be used to analyze the genre of a text.

## Text Processing

Text processing is the process of analyzing and manipulating textual information. This includes extracting smaller bits of information from text which means text extraction, assign values or tags depending on its content which means text classification, or performing calculations that depend on the textual information.

Text Processing is one of the most common task in many ML applications. Steps required for text processing are:

### Data Preprocessing

Steps for data preprocessing are:

* Tokenization — convert sentences to words.
* Removing unnecessary punctuation, tags.
* Removing stop words — frequent words such as “the”, “is”, etc. that do not have specific semantic.
* Stemming — words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
* Lemmatization — another approach to remove inflection by determining the part of speech and utilizing detailed database of the language.

For example, stemming & lemmatization helps in reducing words like ‘studies’, ‘studying’ to a common base form or root word ‘study’.

### Feature extraction

In text processing, words of the text represent discrete, categorical features. So we require to encode such data in a way which is ready to be used by the algorithms. The mapping from textual data to real valued vectors is called feature extraction. The techniques to numerically represent text are:

**Bag of Words (BOW):** A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.

2. A measure of the presence of known words.

It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Since we need to map the context of the data this model is not useful for us.

**Term Frequency-Inverse Document Frequency (TF-IDF):** TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

 It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don’t mean much to that document in particular.

However, if the word imagination appears many times in a document, while not appearing many times in others, it probably means that it’s very relevant. For example, if what we’re doing is trying to find out which genre some summaries belong to, the word imagination would probably end up being tied to the genre Fantasy, since most summaries containing that word would be about that genre.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

The **term frequency** of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.

The **inverse document frequency** of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:

TF-IDF formula

Where:

TF-IDF formula

TF-IDF formula

**Word Embedding:** Itis a representation of text where words that have the same meaning have a similar representation. In other words, it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together. The well-known model of word embedding is Word2Vec.

### Choosing ML Algorithms

There are various approaches to building ML models for various text based applications depending on what is the problem space and data available.

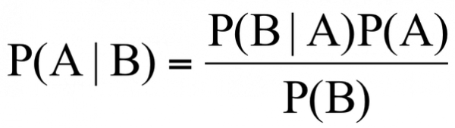
Classical ML approaches like ‘Naive Bayes’ or ‘Support Vector Machines’ for genre predictions has been widely used. Deep learning techniques are giving better results for NLP problems like sentiment analysis and language translation. Deep learning models are very slow to train and it has been seen that for simple text classification problems classical ML approaches as well give similar results with quicker training time.

We examine the following classifiers:

**Naive Bayes Classifier:** A Naive Bayes Classifier is a supervised machine-learning algorithm that uses the Bayes’ Theorem, which assumes that features are statistically independent. The theorem relies on the naive assumption that input variables are independent of each other, i.e. there is no way to know anything about other variables when given an additional variable. Regardless of this assumption, it has proven itself to be a classifier with good results.

Bayes’ theorem is based on conditional probability or in simple terms, the likelihood that an event (A) will happen given that another event (B) has already happened. Essentially, the theorem allows a hypothesis to be updated each time new evidence is introduced.

The equation below expresses Bayes’ Theorem in the language of probability:



* “P” is the symbol to denote probability.
* P (A | B) = The probability of event A (hypothesis) occurring given that B (evidence) has occurred.
* P (B | A) = The probability of the event B (evidence) occurring given that A (hypothesis) has occurred.
* P (A) = The probability of event B (hypothesis) occurring.
* P (B) = The probability of event A (evidence) occurring.

Given a vector **x** of features, Naive Bayes calculates the probability that the vector belongs to each class as **P (Ck | x1, x2, … xn)**.

**Linear Regression**: Regression is a method of modelling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.

**Support Vector Machine:** SVM algorithm is a simple yet powerful Supervised Machine Learning algorithm that can be used for building both regression and classification models. SVM algorithm can perform really well with both linearly separable and non-linearly separable datasets. Even with a limited amount of data, the support vector machine algorithm does not fail to show its magic.

Support vector machine or SVM algorithm is based on the concept of ‘decision planes’, where hyperplanes are used to classify a set of given objects.

The datasets can be separated easily with the help of a line, called a **decision boundary**.

The nearest points from the optimal decision boundary that maximize the distance are called **support vectors**.

In other words, here’s how a support vector machine algorithm model works:

* First, it finds lines or boundaries that correctly classify the training dataset.
* Then, from those lines or boundaries, it picks the one that has the maximum distance from the closest data points.

The following image represents the decision boundary along with the support vectors.

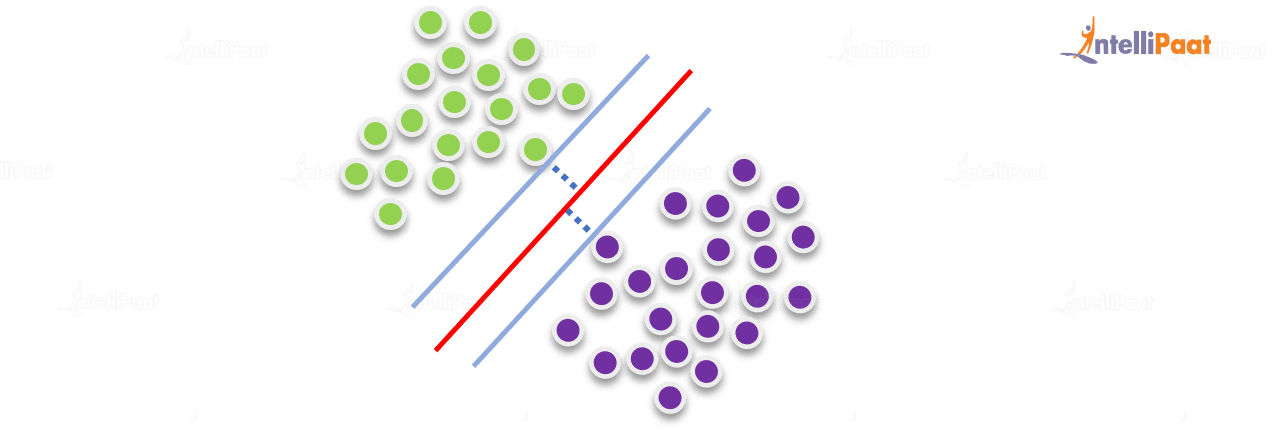


Figure Support Vector Machine (Image source: svm-algorithm-in-python)

**Stochastic Gradient Descent:** SGD is a very popular and common algorithm used in various Machine Learning algorithms, most importantly forms the basis of Neural Networks.

Gradient, in plain terms means slope or slant of a surface. So gradient descent literally means descending a slope to reach the lowest point on that surface.

The major advantage of SGD is its efficiency, which is basically linear in the number of training examples. If X is a matrix of size (n, p) training has a cost of O(kn), where k is the number of iterations (epochs) and is the average number of non-zero attributes per sample.

Mathematical Formulation:

Given a set of training examples **(x1, y1) …, (xn, yn)** where **xi∈ Rm** and **yi∈ {−1,1}**, our goal is to learn a linear scoring function ***f*(x)=wTx+b** with model parameters w∈Rm and intercept b∈R. In order to make predictions, we simply look at the sign of *f*(x). A common choice to find the model parameters is by minimizing the regularized training error given by

where ***L*** is a loss function that measures model (mis)fit and ***R*** is a regularization term (aka penalty) that penalizes model complexity; α>0 is a non-negative hyper-parameter. Different choices for L entail different classifiers.

**Logistic Regression:** Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function which is normally a sigmoid function. If we use linear regression for classification problem, there is a need for setting up a threshold based on which classification can be done. For example, while classification of tumours, if the actual class is malignant with predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Implementation

## Datasets

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

* The Carnegie Mellon University (CMU) Book Summary Dataset (CMU Book Summary Dataset, n.d.)
  + 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 (Blurb Genre Collection, n.d.)
  + This dataset contains blurbs (advertising descriptions of books) for 91,982 books extracted from Penguin Random House along with aligned metadata.

## 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Genre | Count | Genre | Count |
| Fiction | 1615 | **Horror** | 88 |
| Children's literature | 1152 | **Romance novel** | 68 |
| Science Fiction | 999 | **Adventure novel** | 60 |
| Novel | 925 | **Picture book** | 49 |
| Mystery | 734 | **Memoir** | 43 |
| Historical | 727 | **Biography** | 39 |
| Fantasy | 624 | **Techno thriller** | 37 |
| Thriller | 568 | **Short story** | 36 |
| Young adult literature | 321 | **Literary fiction** | 27 |
| Comedy | 174 | **Philosophy** | 23 |
| Autobiography | 170 | **Urban fantasy** | 22 |
| Nonfiction | 166 | **Sword and sorcery** | 20 |
| Spy fiction | 112 |  |  |

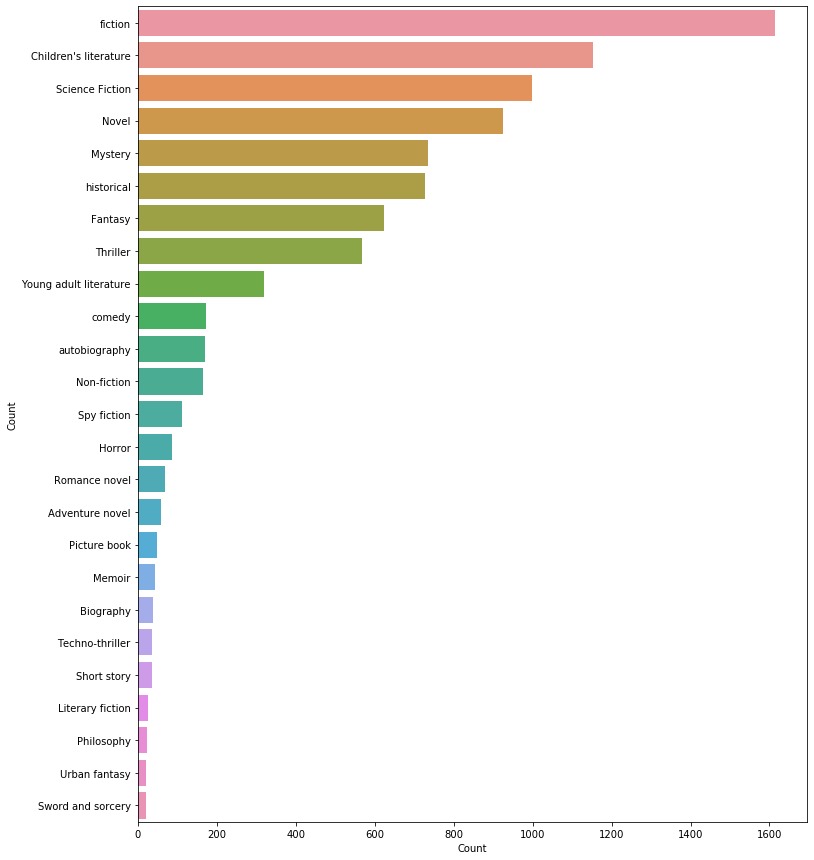


Figure Distribution of CMU dataset

Here is an example data entry for Don DeLillo's White Noise:

Book metadata  
Wikipedia ID 1166383  
Freebase ID /m/04cvx9  
Book title White Noise  
Book author Don DeLillo  
Publication date 1985-01-21  
Genres Novel, Postmodernism, Speculative fiction, Fiction

Plot summary  
Set at a bucolic Midwestern college known only as The-College-on-the-Hill, White Noise follows a year in the life of Jack Gladney, a professor who has made his name by pioneering the field of Hitler Studies (though he hasn't taken German language lessons until this year)…[continued]

### 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 lancast sixteen 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 those are not stop words. Replace each of these words with one of its synonyms chosen at random (Wei).



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.

|  |  |  |  |
| --- | --- | --- | --- |
| Genre | Count | Genre | Count |
| Fiction | 1615 | **Horror** | 968 |
| Children's literature | 1152 | **Romance novel** | 748 |
| Science Fiction | 999 | **Adventure novel** | 660 |
| Novel | 925 | **Picture book** | 539 |
| Mystery | 1468 | **Memoir** | 473 |
| Historical | 1454 | **Biography** | 429 |
| Fantasy | 1248 | **Techno thriller** | 407 |
| Thriller | 1136 | **Short story** | 396 |
| Young adult literature | 642 | **Literary fiction** | 297 |
| Comedy | 1566 | **Philosophy** | 253 |
| Autobiography | 1539 | **Urban fantasy** | 242 |
| Nonfiction | 1494 | **Sword and sorcery** | 220 |
| Spy fiction | 1008 |  |  |

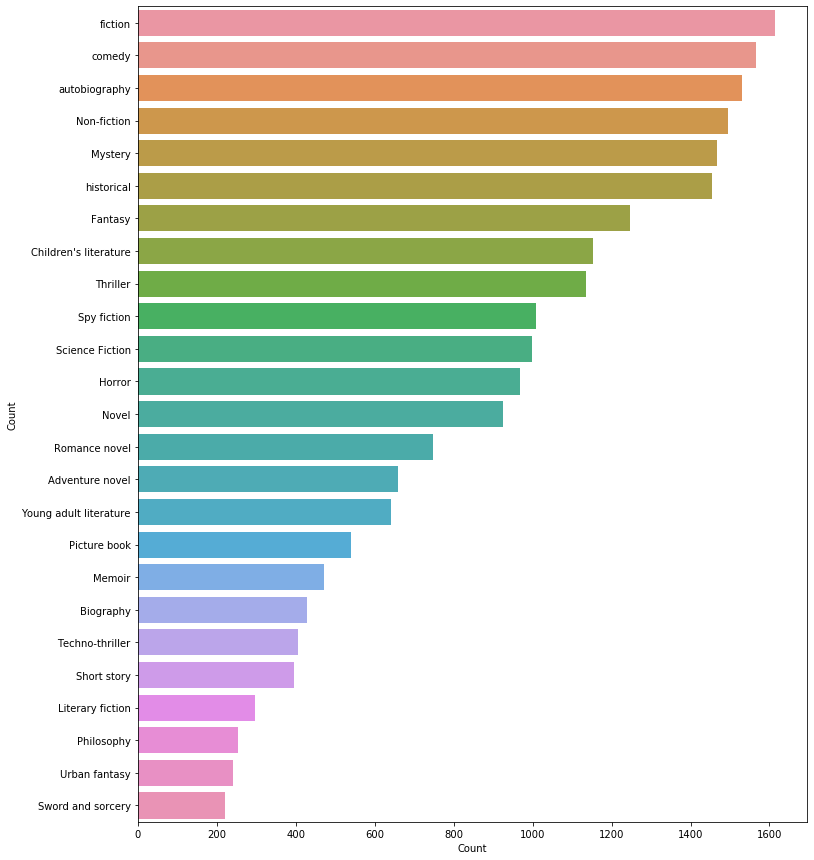
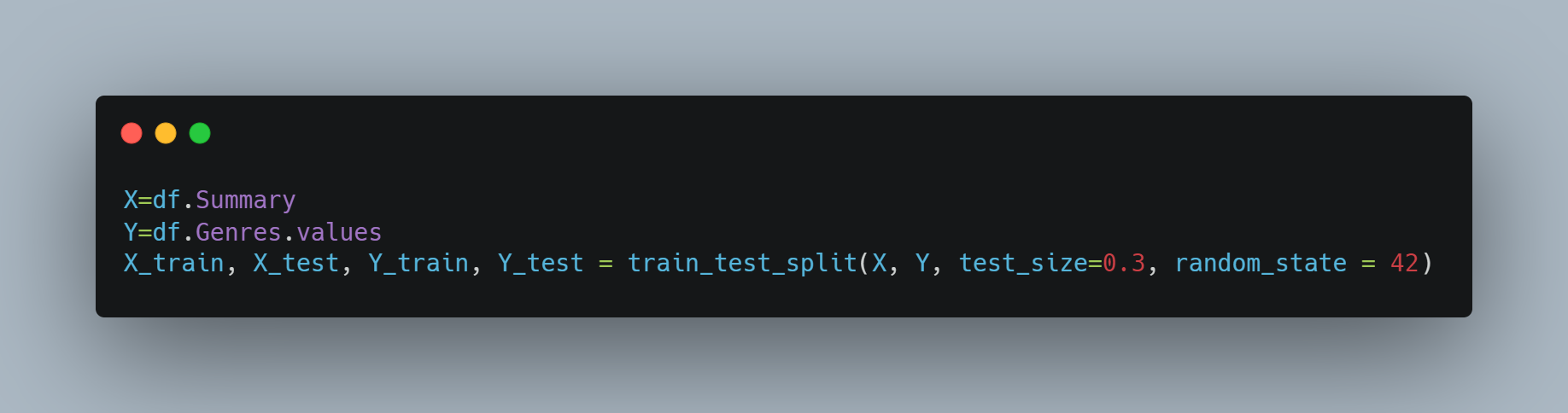


Figure Distribution of CMU dataset after data augmentation

We spilt the data into two sets, a training set and a test set. The training set will be used to fit the model, whereas the test set will be kept aside for checking the accuracy of model on unknown data.



### Classification

 A classifier is a function that assigns a class label to a data point. We use classifiers provided by scikit-learn to train our model and fit the training data. We have used a pipeline which vectorises the data before it is fed to the classifier. The Stochastic Gradient Descent (SGD) classifier provides the best output.

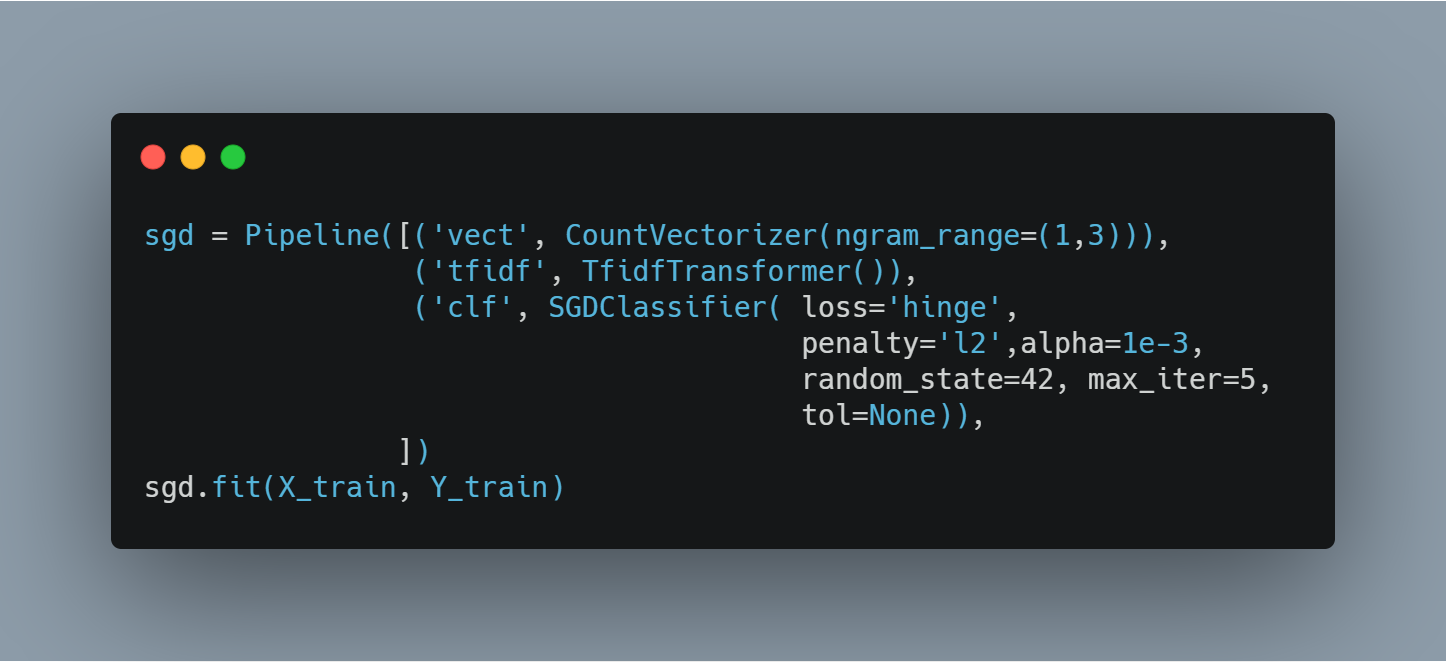
The class SGDClassifier implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification. As other classifiers, SGD has to be fitted with two arrays: an array X of size [n\_samples, n\_features] holding the training samples, and an array Y of size [n\_samples] holding the target values (class labels). Our X contains the summary data and Y contains genre tags.

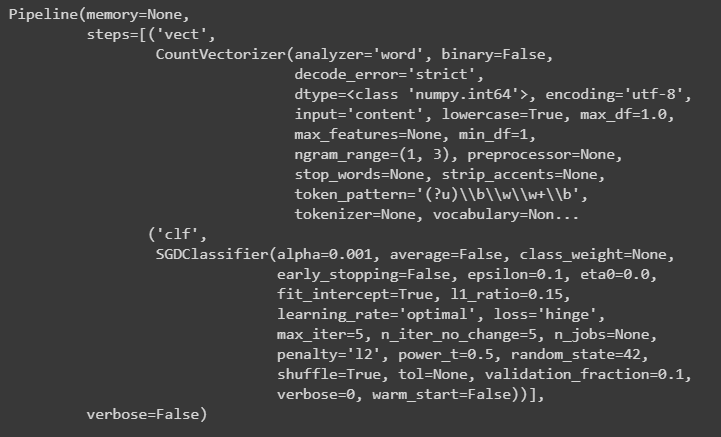
SGD fits a linear model to the training data. The concrete loss function can be set via the loss parameter. SGDClassifier supports the following loss functions:

* loss="hinge": (soft-margin) linear Support Vector Machine,
* loss="modified\_huber": smoothed hinge loss,
* loss="log": logistic regression,
* and all regression losses below.

The first two loss functions are lazy, they only update the model parameters if an example violates the margin constraint, which makes training very efficient and may result in sparser models. (Stochastic Gradient Descent, n.d.)

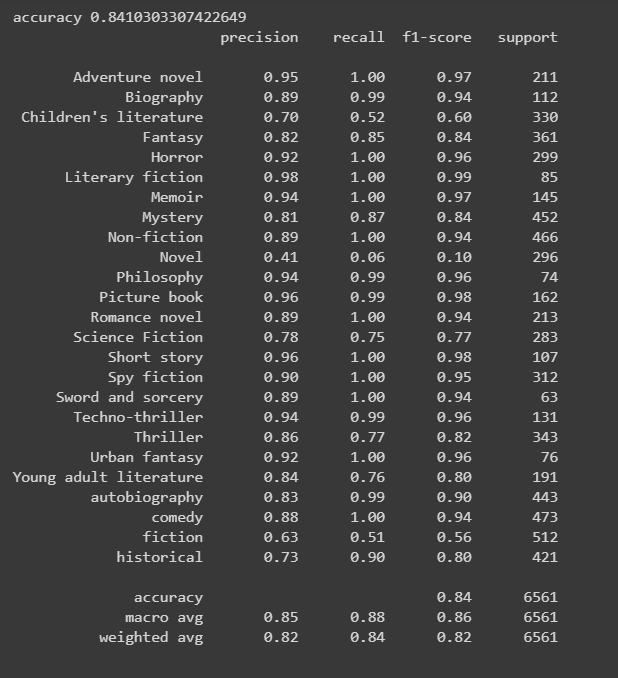
We use a loss="hinge", so our loss function uses linear Support Vector Machines.





### Testing and Accuracy

We use the predict function which predicts multi-class targets using underlying estimators. We then find a classification report for the predicted genres. It looks as follow:

****

Hence, an accuracy of **0.8410** is achieved for single label genre prediction using SGD classifier with a Ngram range of (1,3).

SGDClassifier supports multi-class classification by combining multiple binary classifiers in a “one versus all” (OVA) scheme. For each of the K classes, a binary classifier is learned that discriminates between that and all other K−1 classes. At testing time, we compute the confidence score (i.e. the signed distances to the hyperplane) for each classifier and choose the class with the highest confidence (Stochastic Gradient Descent, n.d.).

**Ensuring that the model is not over fitting or under fitting**

A key challenge with overfitting, is that we can’t know how well our model will perform on new data until we actually test it. We use Cross Validation to ensure that our model is not overfitting or under fitting (Cross Validation Explained: Evaluating estimator performance., 2018). Overall concept:

• To partition the data into a number of subsets

• Hold out a set at a time and train the model on remaining set

• Test model on hold out set

Repeat the process for each subset of the dataset

We use K-fold cross validation technique, using scikit-learn.

The resulting accuracies for each fold are:

|  |  |
| --- | --- |
| Fold | Accuracy |
| 1 | 0.728395 |
| 2 | 0.731139 |
| 3 | 0.756744 |
| 4 | 0.749886 |
| 5 | 0.853681 |
| 6 | 0.871513 |
| 7 | 0.862826 |
| 8 | 0.860997 |
| 9 | 0.863283 |
| 10 | 0.870997 |
| Average | **0.814946** |

On plotting the accuracy data, we observe the following graph:

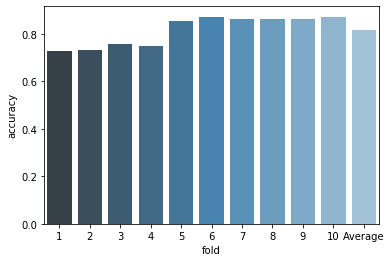


Figure Cross validation output for SGD classifier

We see that there is some variation in the accuracy with the lowest accuracy being 0.72. This is because there are still some genres having low counts and when a large proportion of those are placed in test data the trends and features for that genre may not be identified correctly. If we ignore this aspect and notice the similarities in the accuracies, we can imply that the model is not overfitting.

**Understanding the variation in precision and recall**

Let us focus on the metrics for fiction. We noticed that it has a low precision and recall, why does this happen? Upon analyzing the problem, we discovered the following.

A low recall implies a high rate of false negative, meaning that our classifier is missing many instances of summaries where it actually belonged to fiction, it is assigning it to some other group.

A low precision implies a high rate of false positives, meaning that our classifier is incorrectly assigning fiction as a genre to books that do not belong to fiction.

Notice that we observe a low precision, low recall only in the broader genres. A book may be science fiction and it may be classified as fiction, more such cases lead to low recall. On the other hand, a book belonging to fiction may be classified as science fiction, such instances lead to low precision. The same is observed for Children’s literature.

This problem is caused by overlaps between various genres, a single genre is often insufficient for classifying books, we require multiple genre predictions for the model to be useful. For this reason, we now evaluate another dataset for multi label and multi feature classification.

**Blurb Genre Collection**

**Data distribution**

A frequency table:

|  |  |  |  |
| --- | --- | --- | --- |
| Genre | Count | Genre | Count |
| Nonfiction | 27109 | **Health and Fitness** | 1485 |
| Fiction | 25547 | **History** | 1337 |
| Children’s Books | 16396 | **World History** | 1312 |
| Children’s Middle Grade Books | 6416 | **Popular Science** | 1302 |
| Literary Fiction | 6101 | **Science Fiction** | 1272 |
| Mystery and Suspense | 6028 | **Humor** | 1184 |
| Romance | 5140 | **Crafts, Home and Garden** | 1128 |
| Biography and Memoir | 4390 | **Cooking Methods** | 1101 |
| Religion and Philosophy | 4245 | **Art** | 1095 |
| Religion | 3711 | **Reference** | 1072 |
| Graphic Novels | 3694 | **Parenting** | 1072 |
| Classics | 3249 | **Cozy Mysteries** | 1057 |
| Teen and Young Adult | 3127 | **Historical Romance** | 1040 |
| Cooking | 2468 | **Western Fiction** | 1034 |
| Suspense and Thriller | 2355 | **Psychology** | 1008 |
| Personal Growth | 2322 | **Children’s Adventure Book** | 1003 |
| Politics | 2146 | **Epic Fantasy** | 924 |
| Arts and Entertainment | 1927 | **Paranormal Romance** | 904 |
| Contemporary Romance | 1725 | **Domestic Politics** | 892 |
| Crime Mysteries | 1715 | **Military History** | 760 |
| Fantasy | 1647 | **Suspense Romance** | 749 |
| Philosophy | 1580 | **U.S. History** | 687 |
| Science | 1579 | **Diet and Nutrition** | 637 |
| Teen and Young Adult Fiction | 1506 | **Travel** | 635 |
| Self-Improvement | 1488 | **European World History** | 633 |

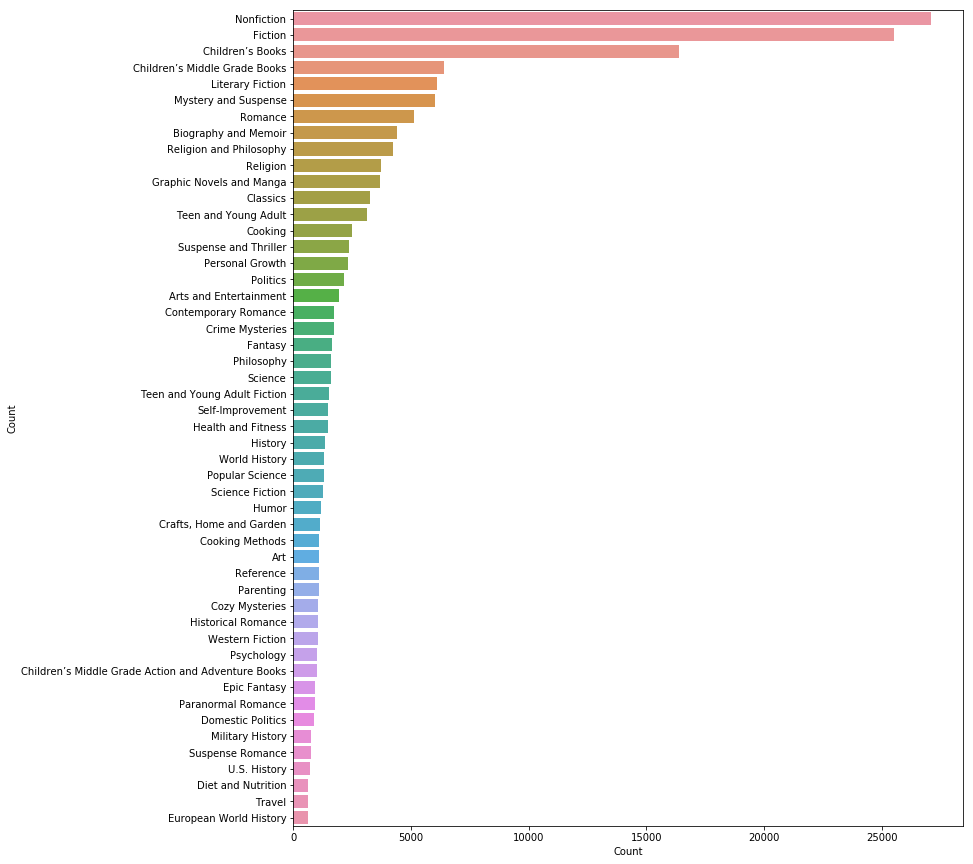


Figure Distribution of Blurb Genre Collection

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

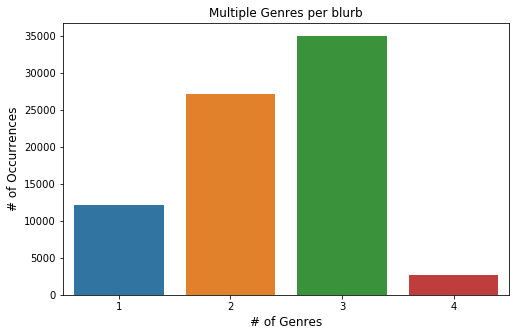


Figure Blurb Genre Collection: Number of genres per summary

A sample of “When Gods Die” looks as follows:

<book date="2018-08-18" xml:lang="en">

<title>When Gods Die</title>

<body>The young wife of an aging marquis is found murdered in the arms of the Prince Regent. Around her neck lies a necklace said to have been worn by Druid priestesses-that is, until it was lost at sea with its last owner, Sebastian St. Cyr’s mother. Now Sebastian is lured into a dangerous investigation of the marchioness’s death-and his mother’s uncertain fate. As he edges closer to the truth-and one murder follows another-he confronts a conspiracy that imperils those nearest him and threatens to bring down the monarchy.</body>

<metadata>

<topics>

<d0>Fiction</d0><d1>Mystery and Suspense</d1><d1>Historical Fiction</d1>

</topics>

<author>C. S. Harris</author>

<published>Nov 06, 2007 </published>

<page\_num> 400 Pages</page\_num>

<isbn>9780451222558</isbn>

<url>https://www.penguinrandomhouse.com/books/296121/when-gods-die-by-c-s-harris/</url>

</metadata>

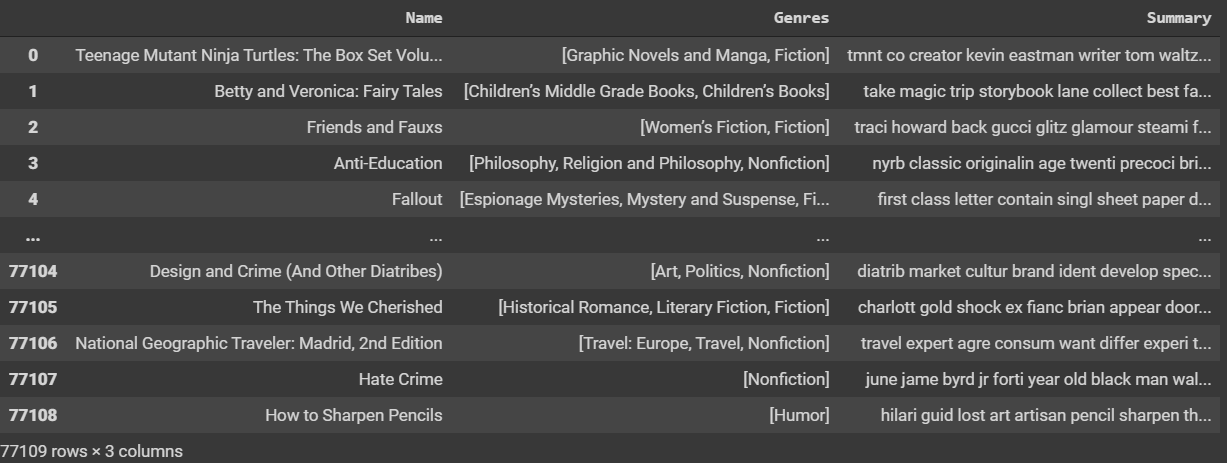
</book>

Notice that the genres of the book are split according to hierarchy.

**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 amulticlass 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.a

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:

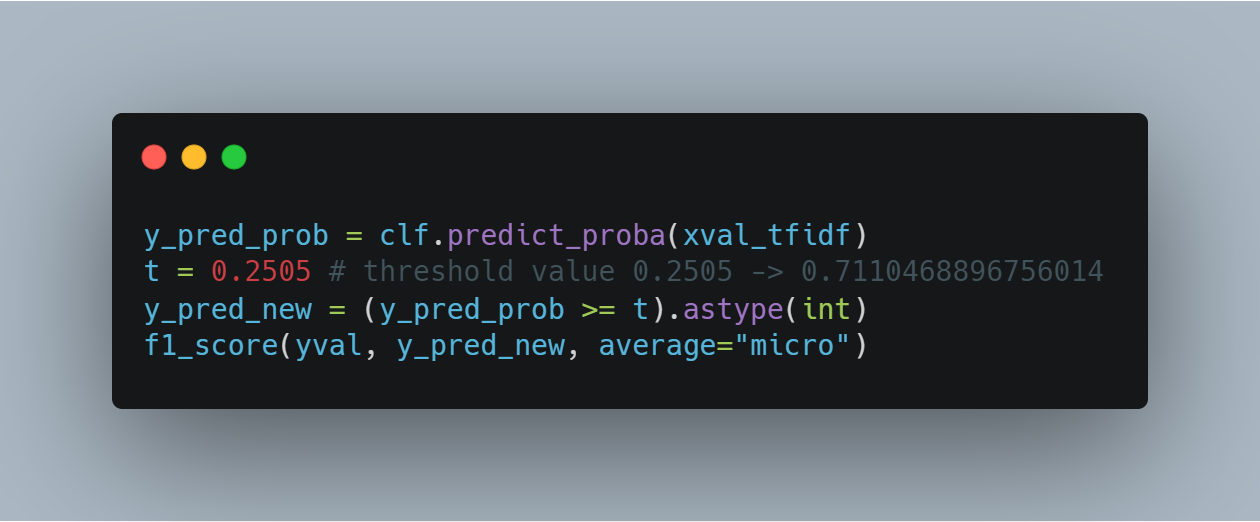
* Linear Support Vector Machine
* Logistic Regression

Now we train the classifier using the fit function.



**Predicting the genres**

Using predict\_proba we get the probability estimates for each genre.



Output: 0.7136495754642689

We achieve an F1 score of 0.713 using this model.

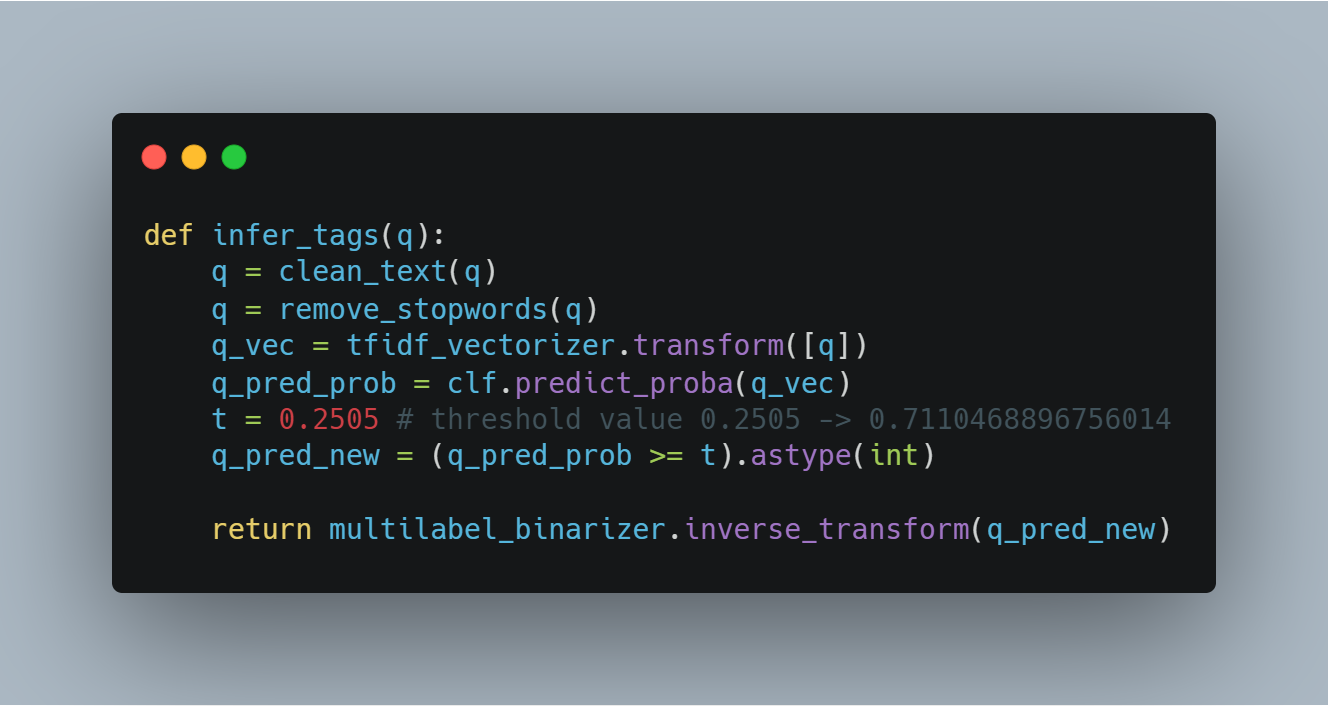
The returned estimates for all classes are ordered by label of classes.

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.

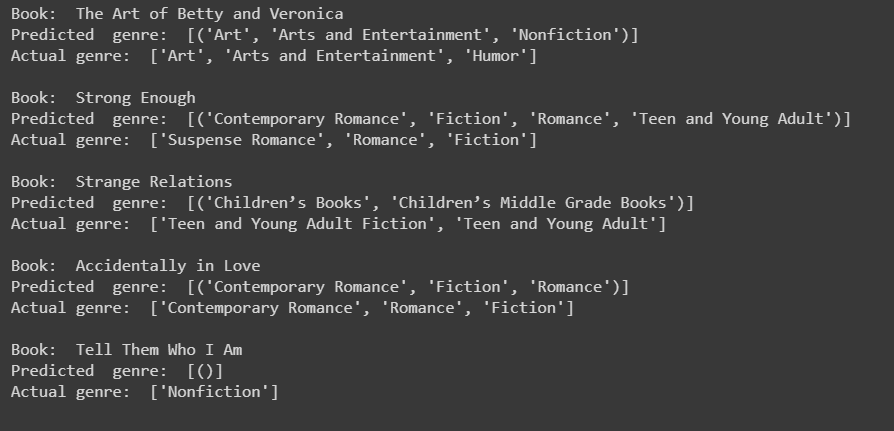
**Printing the predicted genre classes**:

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.

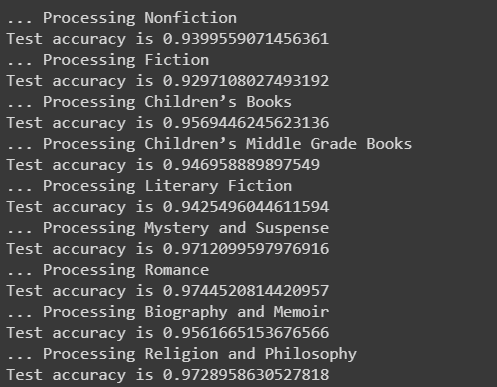
We then use inverse\_transform to get string values of the classes from the binarizer.



The actual label and predicted labels:



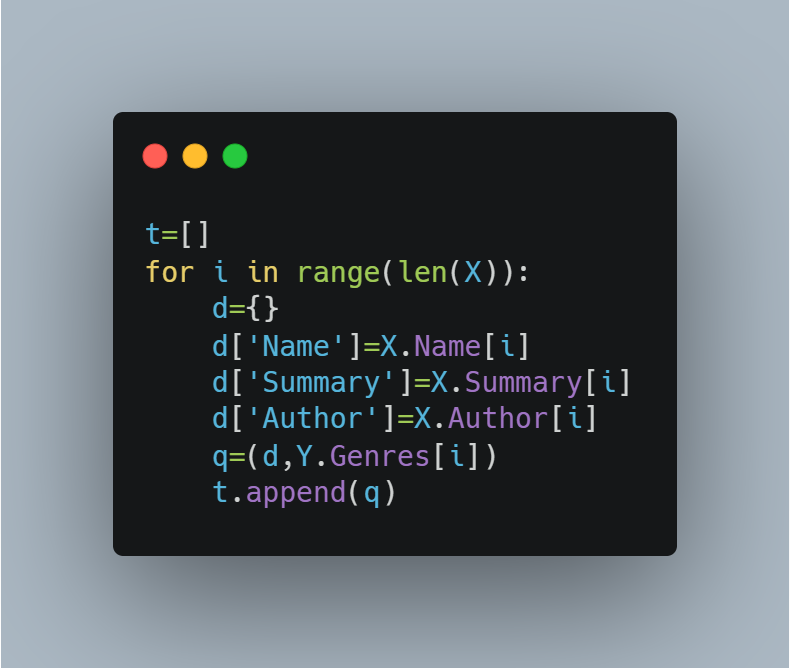
Here we also get the accuracy for correct predictions of individual genres.



### Multi Feature classification

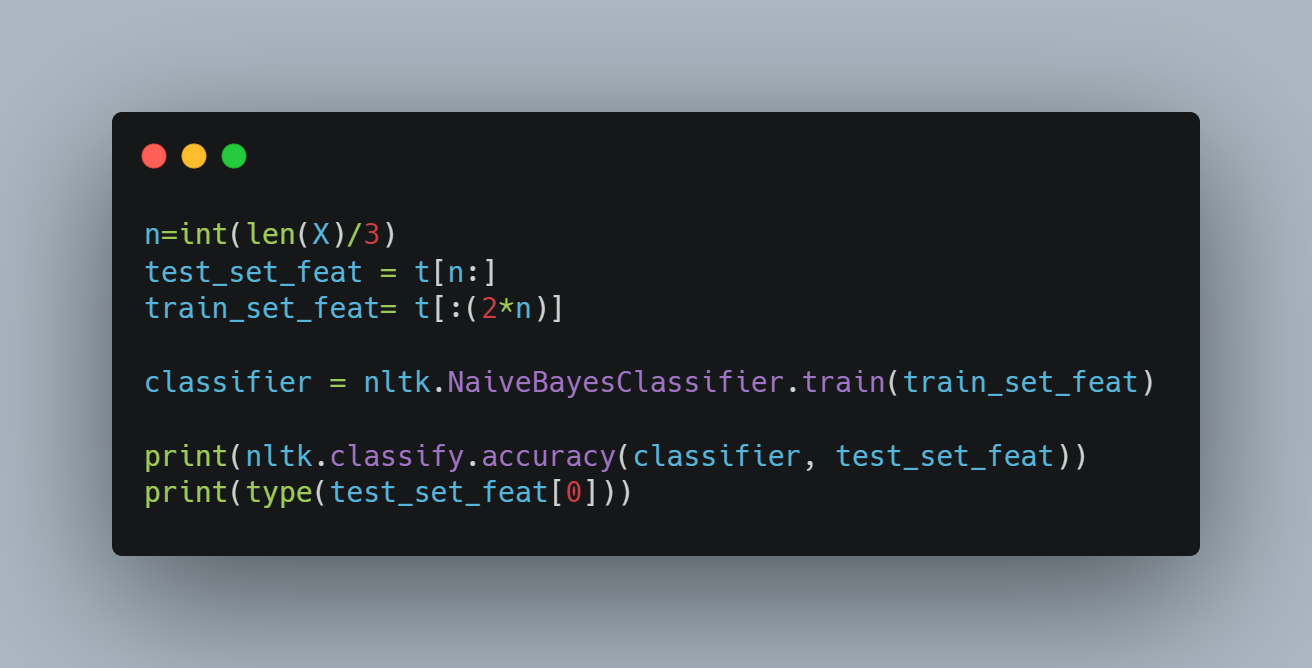
In our general model we have only used the summary of the book to predict its genre. But other features like name of the book and author may also affect the genre. Hence, we apply multiple features to the classification.

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.

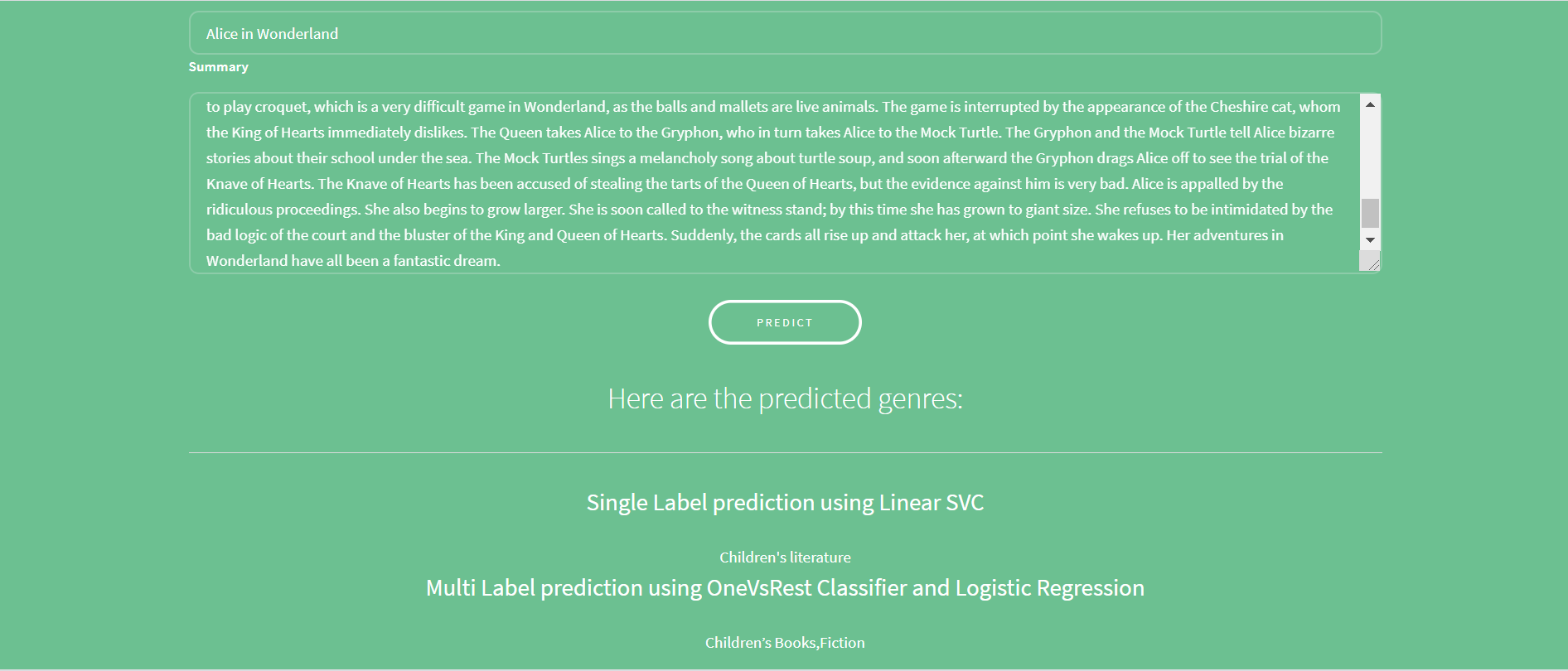
We store this model into pickle files for further use.



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:**

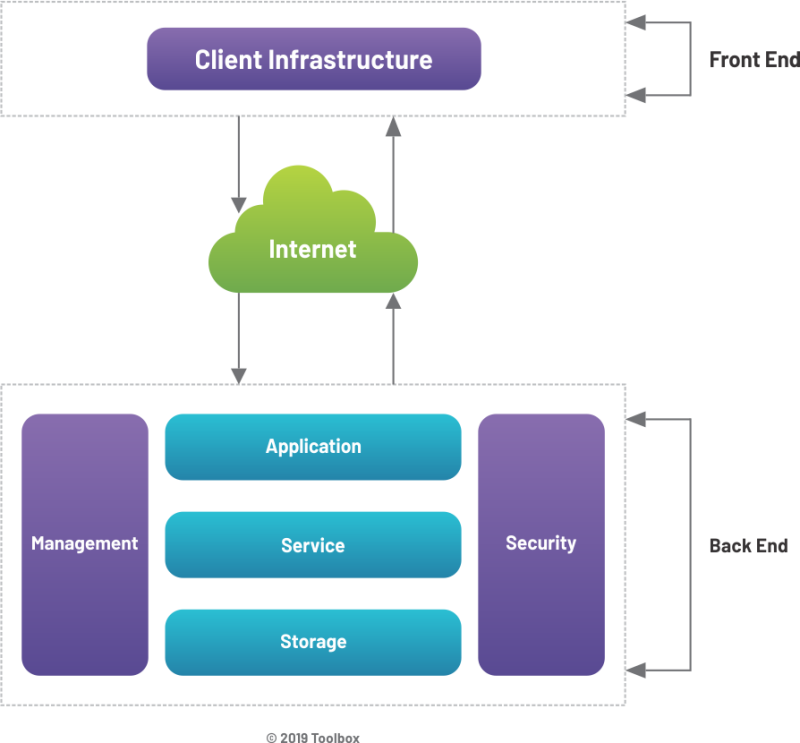


Software Version Deployment Document

Our web app is deployed on a Virtual Machine (VM) running on cloud and a Domain Name Server (DNS) has been associated with the VM IP address letting the VM serve requests from the Internet.

**What is Cloud Computing?**

Cloud computing is when you access computing services—like servers, storage, networking, software—over the internet (“the cloud”) from a provider like Azure. Cloud computing platforms, like Azure, tend to be less expensive and more secure, reliable, and flexible than on-premises servers. With the cloud, equipment downtime due to maintenance, theft, or damage is almost non-existent. We can scale our compute and storage resources—up or down—almost instantly when our needs change.



(Img ref: <https://it.toolbox.com/tech-101/what-is-cloud-computing-architecture-front-end-back-end-explained>)

Steps for deployment:

* Create an account and log in to <https://portal.azure.com>
* Create a resource group for managing resources used for deployment.

A resource group is a container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that we want to manage as a group. Generally, we add resources that share the same lifecycle to the same resource group so we can easily deploy, update, and delete them as a group.

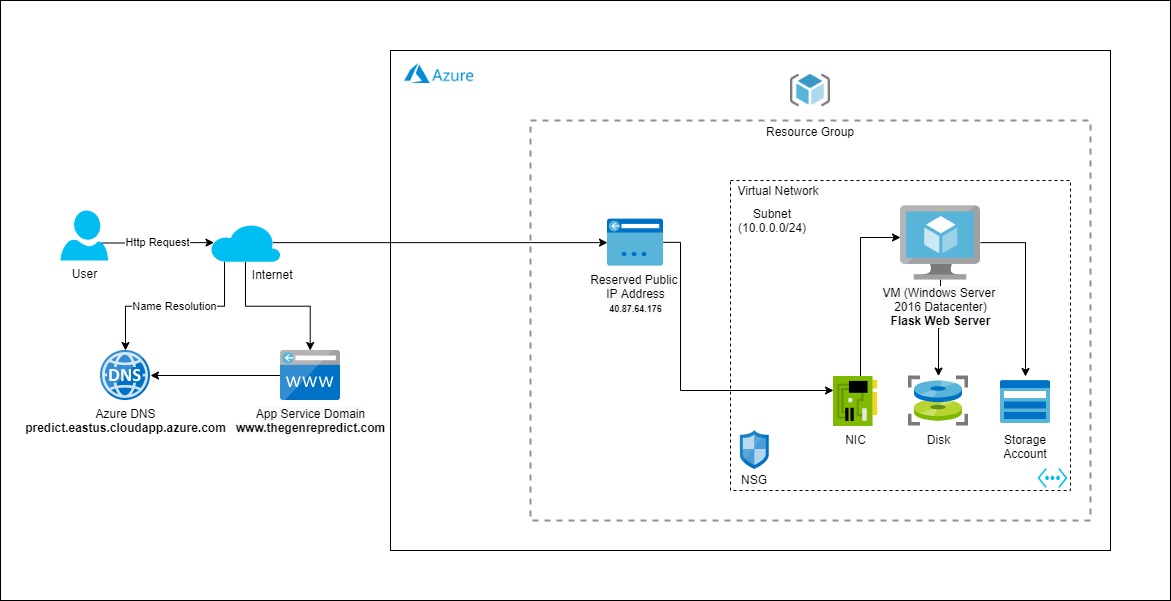
For our project, we’ll add the App Service domain & Virtual Machine in the same resource group.

* Create a Virtual Machine in the created resource group.

In computing, a virtual machine is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer.

With support for Linux, Windows Server, SQL Server, Oracle, IBM, and SAP, Azure Virtual Machines give the flexibility of virtualization for a wide range of computing solutions—development and testing, running applications, and extending datacenter.

VM architecture for our project:



VM details:

* Operating System: Windows (Windows Server 2016 Datacenter)
* Size: Standard F2s\_v2 (2 vcpus, 4 GiB memory)
* Location: East US
* Set inbound and outbound security rules for the VM
* By default, inside the Virtual Network, all requests are allowed and outside requests denied.
* Allow TCP requests from all sources to all destinations for port 80 (http), port 443 (https) and the port that the VM is running on.
* For Cross-origin resource sharing (CORS) allow requests from origin 0.0.0.0/0
* Give a DNS alias to the public IP address of the VM (predict.eastus.cloudapp.azure.com in our case)
* Create an App Service Domain (ASD) in the resource group. (thegenrepredict.com in our case)

App Service Domains allow you toPurchase and manage domains and hostnames for use in your Azure apps.

* Keep the default nameservers provided by azure
* To the ASD, add an alias record set of type A (Address Record) with the alias target of the VM-IP

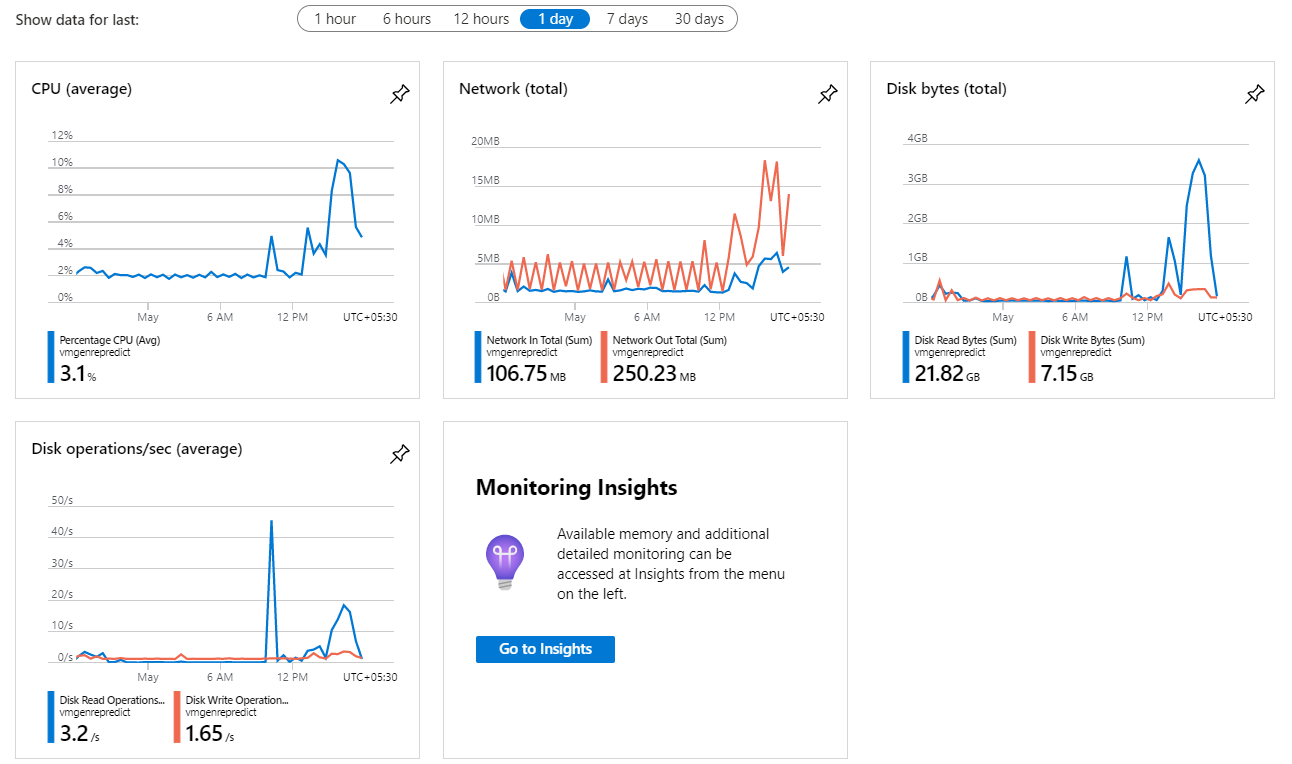
A record type A returns a 32-bit [IPv4](https://en.wikipedia.org/wiki/IPv4) address, most commonly used to map hostnames to an IP address of the host.

* Download the Remote Desktop Protocol (RDP) file of the VM and run it.
* In the VM, install the dependencies required to run the web app.
* Download and run the web app locally in the Virtual Machine
  + Host: 0.0.0.0
  + Port: 80

**What happens when a user tries to request at the DNS?**

The DNS has an address record for the VM hence the address translation to the VM’s public IP occurs. As the request is an http request, it will be sent to port 80 of the VM where the web app is running, allowing requests from all addresses (host: 0.0.0.0). Hence a response will be generated and returned.

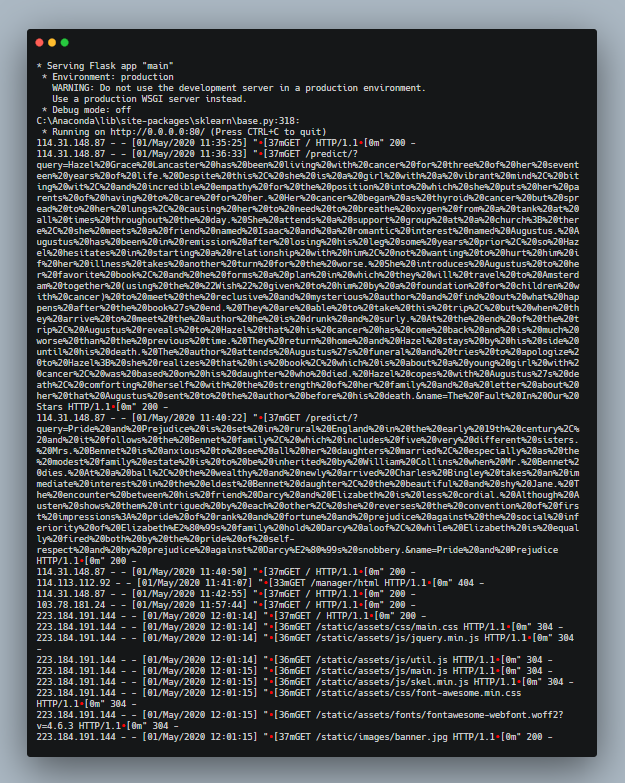
VM Metrics for a day:



Log file generated for tracking requests for future enhancement of model:

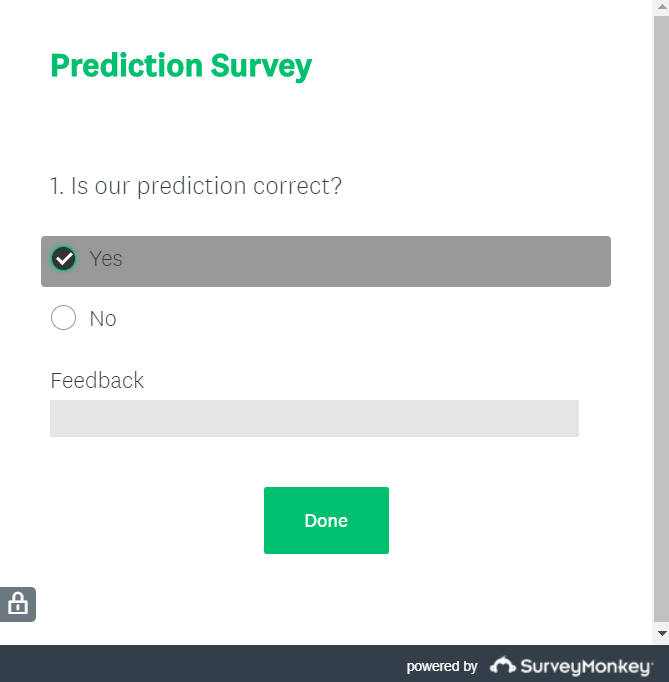


Server Log:

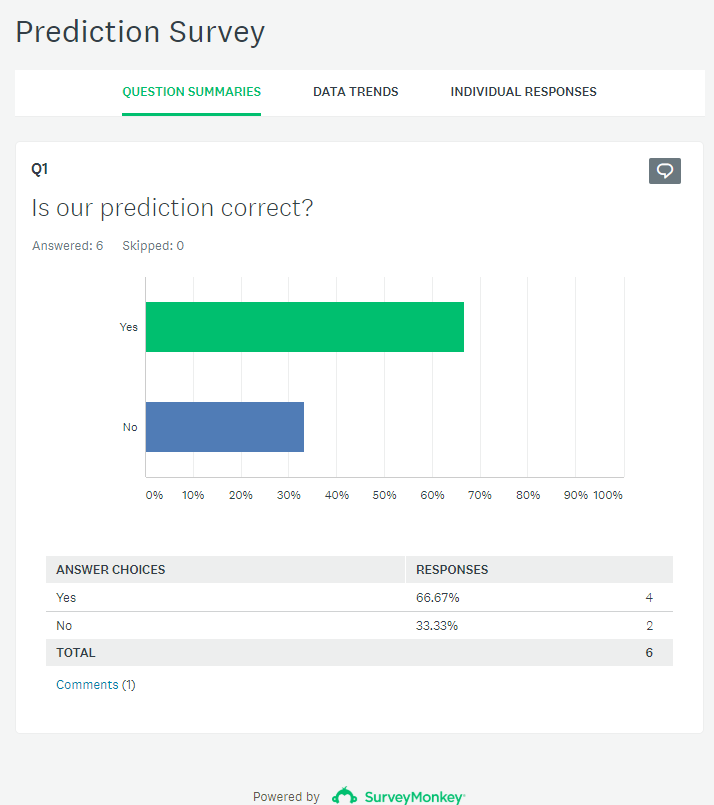


As part of deployment, we have also added a survey form for checking real time accuracy of the model. The form is powered by [www.surveymonkey.com](https://www.surveymonkey.com).

Here is a snapshot of how it looks:



Here is the feedback we got from real users:



Experiments

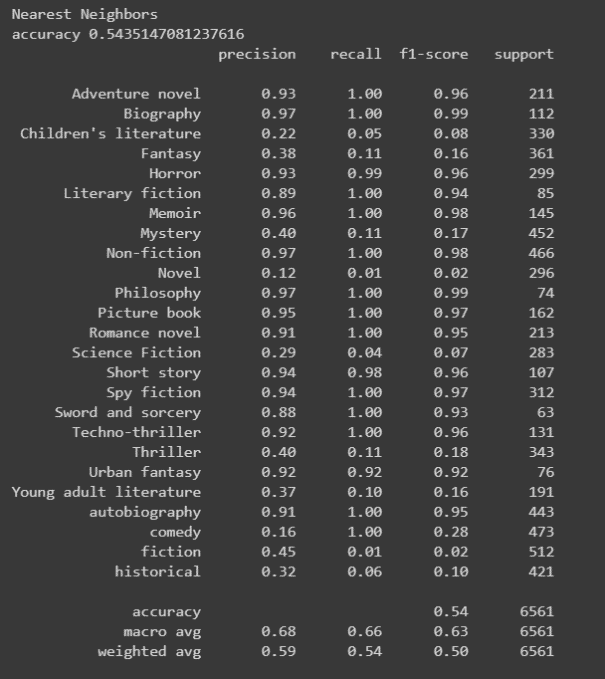
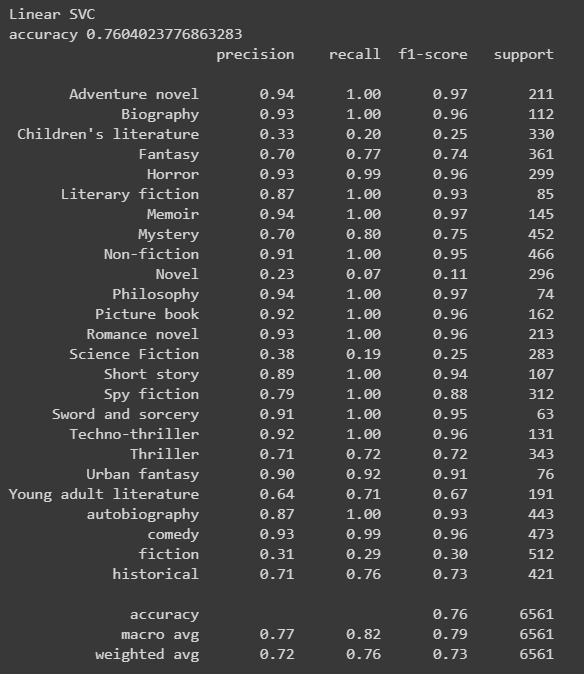
We tried using different classifiers on different datasets to get the best results.

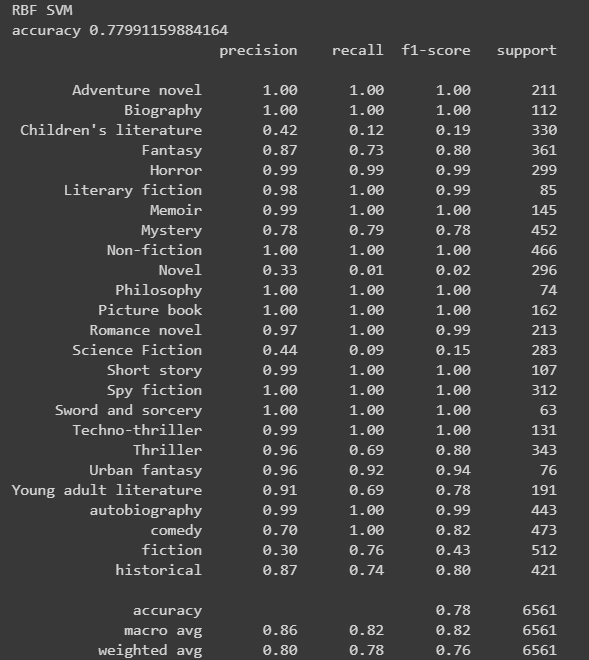
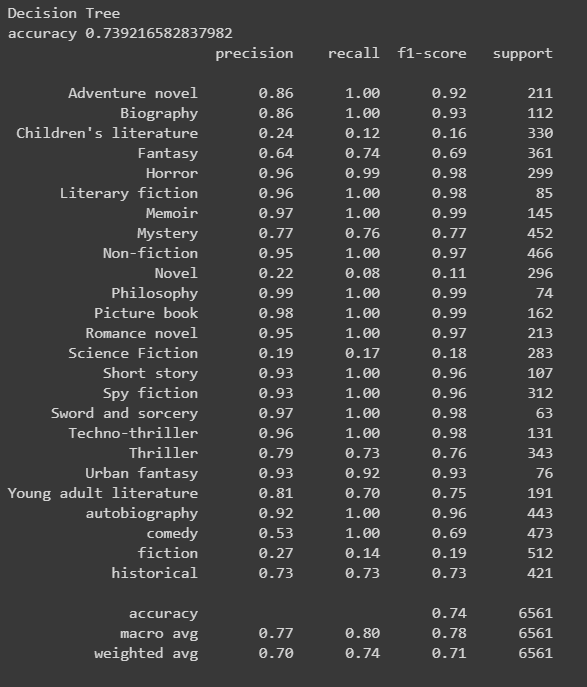
CMU Book Summary Dataset

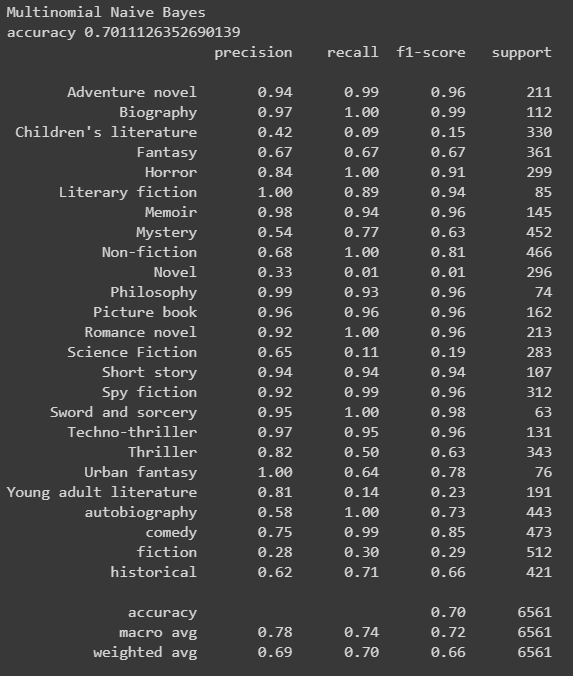
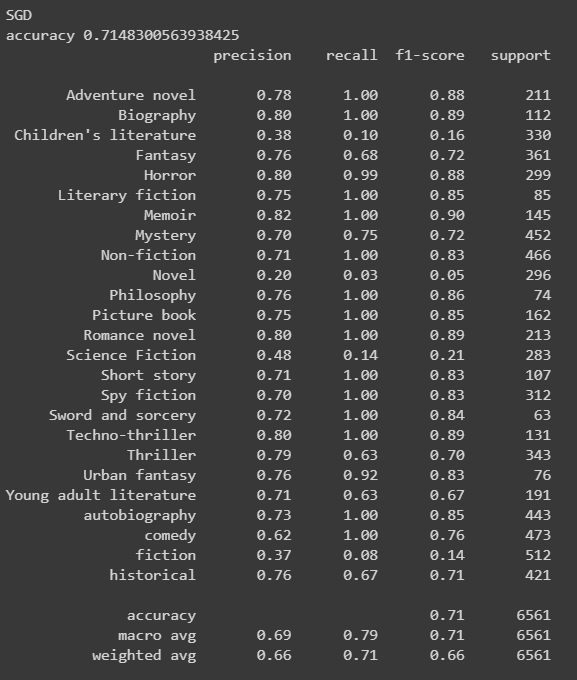
* Applying different classifiers on augmented data



Classification Reports:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1 score | Accuracy |
| Nearest Neighbours | 0.68 | 0.66 | 0.63 | 0.5435 |
| Linear SVC | 0.77 | 0.82 | 0.79 | 0.7604 |
| SVC | 0.86 | 0.82 | 0.82 | 0.7799 |
| Decision Tree | 0.77 | 0.80 | 0.78 | 0.7392 |
| Multinomial Naïve Bayes | 0.78 | 0.74 | 0.72 | 0.7011 |
| Stochastic Descent Gradient | 0.69 | 0.79 | 0.71 | 0.7148 |

Following is the accuracy data obtained using cross validation technique.

|  |  |
| --- | --- |
| Model name | Accuracy |
| DecisionTreeClassifier | 0.555763 |
| KNeighborsClassifier | 0.721846 |
| LinearSVC | 0.892405 |
| MultinomialNB | 0.610499 |
| SGDClassifier | 0.845809 |
| SVC | 0.793863 |

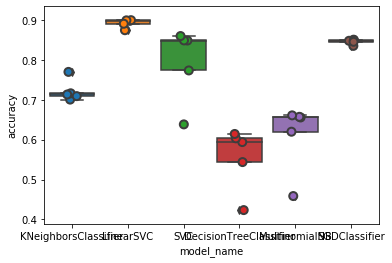


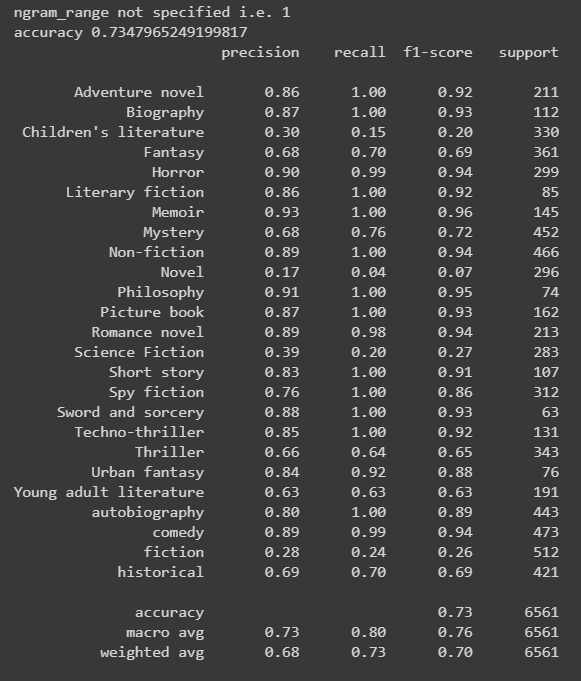
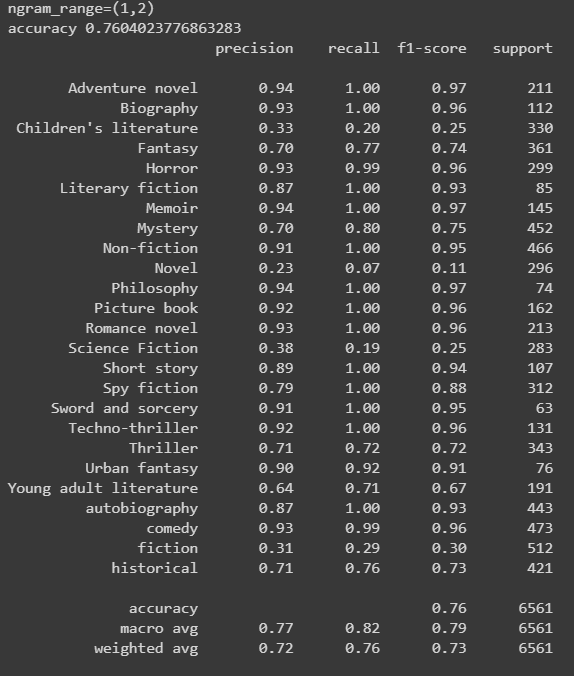
Figure Boxplot for accuracy vs. classifier

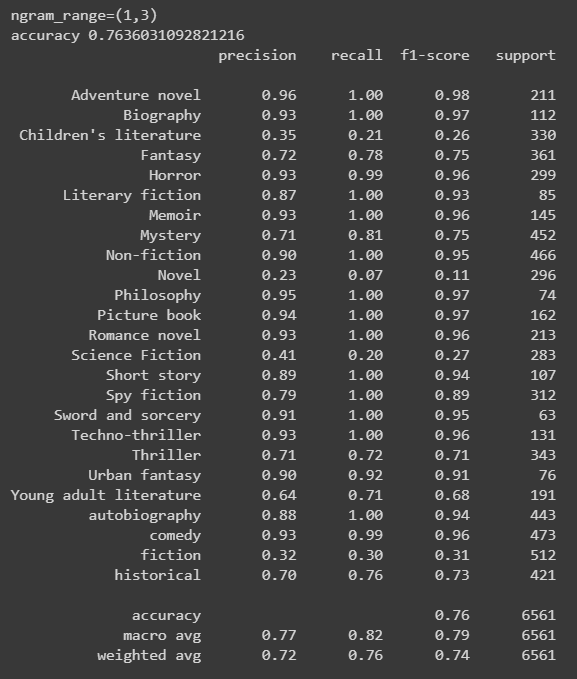
* Applying different Ngram ranges to the vectorizer

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 analyze the impact of Ngrams using a SVC model.

We observe that in accordance to our expectation the accuracy increases when we increase Ngram range. This is because Ngrams provide contextual information and phrases which help in improving the model.

Classification Reports:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ngram range | Precision | Recall | F1 Score | Accuracy |
| 1 | 0.73 | 0.80 | 0.76 | 0.7347 |
| 1-2 | 0.77 | 0.82 | 0.79 | 0.7604 |
| 1-3 | 0.77 | 0.82 | 0.79 | 0.7636 |

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.

F1 score of logistic regression on original dataset: 0.71

F1 score of logistic regression 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 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 analyzing its 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. We can also validate the accuracy of the generated summary and its relativity to the original book. This can be utilized to display highlights of the book along with the predicted genre.

There are various other enhancements that may be possible.

Bibliography

*Blurb Genre Collection*. (n.d.). Retrieved from University of Hamburg: https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/blurb-genre-collection.html

*CMU Book Summary Dataset*. (n.d.). Retrieved from https://www.kaggle.com/ymaricar/cmu-book-summary-dataset

*Cross Validation Explained: Evaluating estimator performance.* (2018). Retrieved from towardsdatascience.com: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

*Image source: svm-algorithm-in-python.* (n.d.). Retrieved from intellipaat: https://intellipaat.com/blog/tutorial/machine-learning-tutorial/svm-algorithm-in-python/

Jordan, E. (2012). *Automated Genre Classification in Literature.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.864.9862&rep=rep1&type=pdf

Kalita, J. W. (n.d.). *Genre Identification and the Compositional Effect of Genre in Literature*. Retrieved from https://www.aclweb.org/anthology/C18-1167.pdf

Li, S. (2018). *Multi-Class Text Classification Model Comparison and Selection*. Retrieved from towardsdatascience: https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568

*Natural Language Processing with Python*. (n.d.). Retrieved from https://www.nltk.org/book/

Sawarn, A. (2019). *Keras for Multi-label Text Classification*. Retrieved from https://medium.com/towards-artificial-intelligence/keras-for-multi-label-text-classification-86d194311d0e

*Stochastic Gradient Descent*. (n.d.). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/sgd.html

Wei, J. (n.d.). *Data Augmentation Techniques in Natural Language Processing .* Retrieved from towardsdatascience: https://towardsdatascience.com/these-are-the-easiest-data-augmentation-techniques-in-natural-language-processing-you-can-think-of-88e393fd610