# PREDICTING MOVIE GENRE USING PLOT SUMMARY

**Kelvin Gwari**

Submitted to the
Graduation Projects Examination Jury
in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software Engineering

European University of Lefke
January 2020

Approval of the Software Engineering Department

_____

Assoc. Prof. Dr. Hüseyin Ademgil
Chairman/Director

This is to certify that we have read this graduation project and that in our opinion it is fully adequate, in cope and quality, as an Undergraduate Project.

Assist. Prof. Dr. Vesile Evrim
Supervisor

Members of the examining committee

| | Name | Signature |
|---|---|---|
| 1. | Assoc. Prof. Dr. Hüseyin Ademgil | ..... |
| 2. | Assist. Prof. Dr. Vesile Evrim | ..... |
| 3. | Assist. Prof. Dr. Cem B Kalyoncu | ..... |
| 4. | Dr. Ersin Çağlar | ..... |

Date: .....

# Abstract

## PREDICTING MOVIE GENRE USING PLOT SUMMARY

by

Kelvin Gwari

Software Engineering Department
European University of Lefke

Supervisor: Assist. Prof. Dr. Vesile Evrim

The project uses the supervised text classification tools and machine learning methods to predict movie genres based on their movie descriptions. The movies and their corresponding data which is the plot summaries, the titles and genres will be taken from TMDB with over 10 000 movie data. As a movie can belong to several genres, this project will use the multi label classification. The main aim is to process natural languages, therefore only the movie descriptions will be used.

Since movies are all over the internet and they are attracting a large portion of the population, it is with great importance that the information found about a movie which is mostly the plot summaries can be used to predict the genre of a movie. This will be achieved by text classification and machine learning techniques. We will also be including the emotions each plot carries to help with the predictions. Movie genres are associated with emotions like for horror there is fear.

**Keywords:** Machine Learning, Cross validation, Multi-label classification, Emotions, GUI (graphical user interface), Database, Python, MySQLetc.

## Acknowledgments

I would like to thank my supervisor Asst. Prof. Dr. Vesile Evrim, Department of Computer Engineering for her support, intellectual supervision and useful feedback regarding my final year Graduation project.

I also want to extent my utmost gratitude to Asst. Prof Dr. Fehun Yogancioglu, Asst. Prof. Dr. Cem Kalyoncu for guiding me throughout the years and also the entire Computer and Software department staff.

Lastly, I want to thank my family and friends for their valuable support and encouragement and above all the almighty for making this project a success.

## Table of Contents

**Contents** **Page**

**List of Figures** **Page**

**List of Equations** **Page**

**List of Code Implementation** **Page**

# 1. INTRODUCTION

## 1.1 Definition

In a world where everything is being digitalized and the Internet of Things (IoT ) is taking inroads to our day to day activities, it would be nice to have the entertainment industry to follow suit as well. With the invention of the internet, everything is now found online from even grocery shopping.

Therefore since movies are all over the internet and they are attracting a large portion of the population, it is with great importance that the information found about a movie which is mostly the plot summaries can be used to predict the genre of a movie. This will be achieved by text classification and machine learning techniques. The movie information will be taken from The Movie Database (TMDB) through a corpus [1] and will also be stored into our own genre database for ease of access and to speed up the process. The data will be placed into a file with each file holding one movie occurrence. The data will be processed and will provide a decision boundary that will decide whether the genre of a movie is correct or incorrect.

The text processing will be done in a supervised learning way where a group of words will be put in sets and the output produced and grouped according to the sets. The plot summaries are mostly made by users in the Internet Movie Database (IMDB). Sometimes the plot summaries won't be exactly matching with the genres of the movie. Therefore the system will give a better guidance for better plot summaries which will be matching with the genres.

## 1.2 Problem Statement and Motivation

Time, accuracy and consistency are the most wanted resources in today's day to day living. With this prediction model, i aim to optimize these three. Time is the most important resource, it is one we can't make more of, therefore if we can make a model that addresses time management by reducing the time that is used to come up with the categories of the movies. These movies involved watching the movies and one would then just give it a genre that he thinks suits the movie. The problem with this method is that it has a lot of inconsistencies involved.

As we want our movies to be categorized for parents to sort the type of movies they can watch as a family, consistency is of paramount importance. This is beacuse if a movie has been wrongfully catergorized kids would watch harmful movies. Therefore

this model will increase the level of consistency and make each movie be in sync with its scenes and genre categorization. This will in turn adds accuracy to the model and the main problems with the time, accuracy and consistency will be solved.

Since perfection has become one of the important attrribute in this twenty first century with people automating all aspects of life from sewing clothes in the factories to watering their gardens. As a young man growing up in this world, i would expect everything to be flawless since its automated.

Myself as a movie loving person have seen some movies that have been wrongfully categorised and i would play a movie hoping for some action packed scenes and then to my dissapointment, it will be all drama and maybe one action scene while you were hoping for the action scenes to dominate the whole movie.

This then made me to think, well there is machine learning at our dispossal, why not use it to train a machine to categorize the movies into genres which will be flawless than waiting for some people to categorize it for us which will lead to a whole lot of biased categorization.

### 1.2 Ethics

With the diversity of movies currently present in the world of entertainment. It is of great importance to present the ethics and morals of how the system will work to protect and respect cultures and different religions. Any malice use of the system to violet the safety and health of the public, the developer i.e. Kelvin Gwari, will not be held responsible for the action. The system will and must be used with human culture and conservation of the human rights in mind. It will not be implemented to get movies from the adult entertainment industry ever. The sets of the data used in the supervised training will not include any genre that is harmful or inappropriate.

# 2. BACKGROUND INFORMATION

## 2.1 Python

Python is a scripting programming language which is fast growing in the tech industry. Python isbeing used in the data analysis industry, used by scientist, students, software engineers etc. Python as a general purpose language can work quickly and gives you the time to test your product espeacially for machine learning purposes.

The most common features that makes python a very good and powerful programming language are its dynamic typing and binding. This means it does not know the type of a variable until the code is executed. The next feature is its object oriented, interpretation and high level programming capabilities. İt also has an easy syntax which can be understood by anyone.

A greatest advantage with python is that it is an open source, you can read and modify the source code. Existing deep learning and machine learning libraries run efficiently in python which makes it a very good programming language to use in the fields of machine learning and Artificial Intelligence. The libraies include but not limited to:

i. **SciPy :** Provides efficient routines for numerical integration and optimization through the use of numpy arrays.
ii. **NumPy:** It supports large, multi-dimensional arrays and matrices and includes many high level mathematical functions to operate the arrays.
iii. **Matplotlib:** is a 2D plotting library that can generate data visualizations as histograms, power spectra, bar charts, and scatterplots with just a few lines of code.
iv. **Scikit-learn:** For machine learning algorithms which will reduce the time and help to develop the product faster.

Since it has many advantages in the machine learnig department, python is not limited to these features only but has many more and one can notice that when using it.

## 2.2 MySQL

At the moment MySQL is the most popular open source database management with its reliability and ease of use. It is the database managent that even the world’s largest and fastest growing organizations use it to save high volume websites and business

critical. MySQL is a relational database management system based on SQL. SQL means Structured Query Language.

Relational database stores information in a centralised place and it comes with orderly and optimized files to maximise speed. Users can convert SQL statements into an encrypted code or hiding SQL sysntax the any language's API.

MySQL database provides features like:

i. Self-healing replication clusters to improve scalability, perfomance and availability.
ii. High performance and scalability to meet the demands of exponentially growing data loads and users.
iii. Platform independence giving you the flexibility to develop and developon multiple operating systems.
iv. Big data inteoperability using MySQL as the operational data store for Hadoop amd Cassandra.

### 2.3 Machine Learning

Machine learning is the scientific study of algorithms and statistical model that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen s a subset of artificial inteligence. It is based on the idea that systems can learn from data, identify patterns and make decisons with little to no human intervention.

The resurging of machine learning is because of the factors tha made data mining and bayesian analysis more popular than ever. These developments means that it is now possible to quickly and automatically produce models that can analyze bigger and more complex data and deliver faster and more accurate results [13].

Machine learning follows a process of preparing data, training an algorithm and generating a machine learning model, and then making and refining the predictions. Judith Hurwitz and Daniel Kirsch advise that machine learning needs the correct set of data that can be applied to a learning process. Data preparation involves:

**i. Selcting a sample data set:** In this technique you just select a valid data from a large population and make it your subset which you will apply the various machine learning tools to it.

ii. **Clean the data:** This is achieved by removing and or replacing missing items from the data set. The goal is to make that data is complete and relevant.

iii. **Normalize the Data**: When you have data to tarin the data might be different and might be scored numerically and some using percentage, therefore to compare data, the values must be normalized to a common scale.

Machine learning alorithms are programs tha adjust themselves to perform better as they are exposed to more data. Just like humans change how they process data by learning, so a machine-learning algorithm is a program with a specific way to adjust its own parameters, given responses on its previous perfomance making predictions about a dataset[15].

The differnt types of algorithms are Regression algorithms, Desion trees, Instance-based algorithms and clustering algorithms.

### 2.4 Cross Validation

Cross Validation [16] is a statistical method used to estimate the skill of machine learning models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited.

To tackle the issue of overfitting, we can split our initial dataset into separate training and test subsets. We can achieve cross validation in a number of ways. These ways include k-fold cross validation, leave one out cross validation and stratified k-fold cross validation. In k-fold we significantly reduces underfitting as we are using most of the data for fitting, and also reduces overfitting because most of the dat is used in validation set[14].

The basic concept is partitioning the data into subsets. These sets might be split in a ratio of 90:10 where we have 90% as the training set and the 10% as the testing set. The 90:10 is mostly done on a small data to achieve a good perfomance score.

The most commonly used split ration is 80:20. In this ratio there are 5 splits of training and validation sets. When we split the data set into two, we hold the one set which will be the 20% of the ovearal data and train the model using the remaining set. This will be done 5 times if we are using the 80:20 ratio. The other set that has been kept will be used to test the data and calculate precision, recall and eventually fscore of the model.

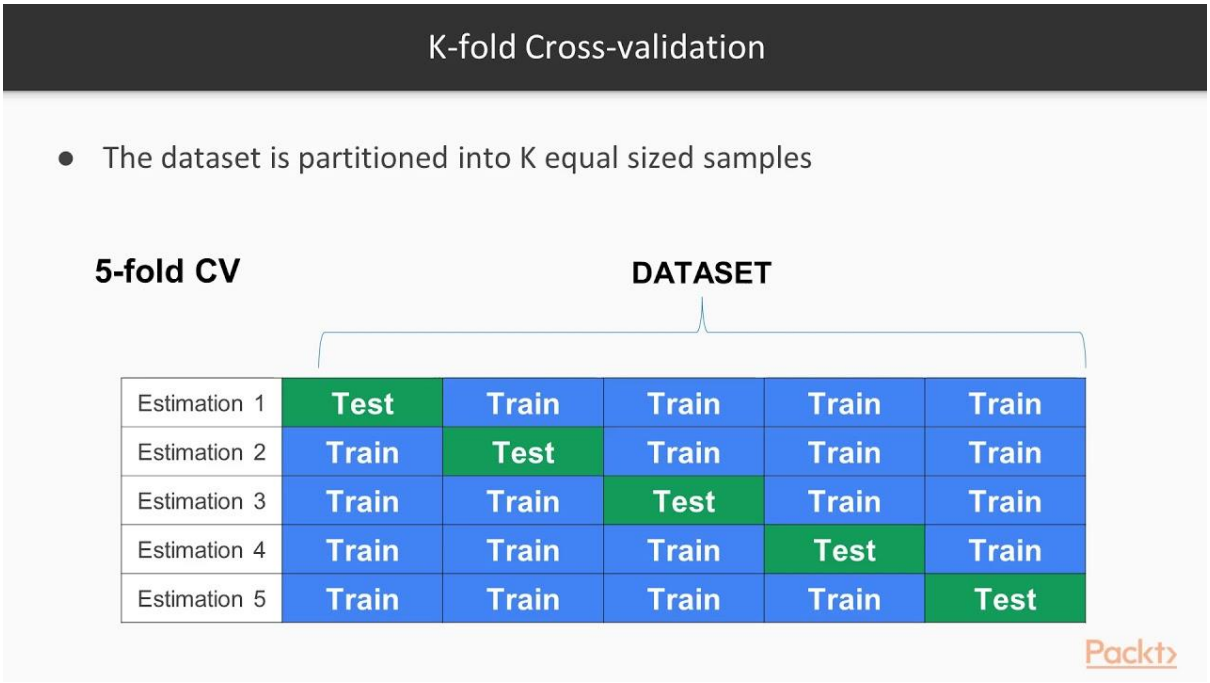


Figure 2.1 **Cross validation**

### 2.5 Precision

The main idea of precision [17] is that it has to tell us which of the positive identifications was actually correct which means the percentage of your results that is relevant.

$$PRECISION = \frac{TRUE\ POSITIVES}{TRUE\ POSITIVES + FALSE\ POSITIVES}$$

Equatıon 1.1 **Precision**

Precision and recall are trade offs of one another depending on the type of data one is predicting. If one is looking to predict people with a disease like cancer. We would like the model to be precise that is only to identify the relevant data points. If you want to minimize the number of false positives, that is the number of items that the model predicted as positive where as in actual fact they are negative, then we have to prioritize precision over recall.

### 2.6 Recall

Recall works when the data we are predicting is a sensitive one. This means that if the cost of getting high false negatives from our prediction model is high [17]. Recall is the fraction of the total amount of the relevant instances that were actually retrieved.

$$RECALL = \frac{TRUE\ POSITIVES}{TRUE\ POSITIVE + FALSE\ NEGATIVES}$$

Equation 1.2 **Recall**

Although recall and precision are trade offs, they are all based from an understanding and measure of relevance. If you want to minimize the number of false negative from the model then recall is the most appropriate metrix to use in your model.

### 2.7 F-score

Some models need a very good recall score and some need a very good precision score. There are other models however that need the best of both. Precision and recall are two important model evaluation metrics, unfortunately, we cannot get the best of both at the same time. This is because one comes at the expense of the other.

However in cases where we want the best blend of the two we intoduce another metrics evaluation called the f1-score [18]. F1-score is the measure of test's accuracy. It is the harmonic mean of precision and recall.

$$F1 - SCORE = 2 * \frac{PRECISION * RECALL}{PRECISION + RECALL}$$

Equation 1.3 **F-measure**

There are other metrics for combining precision and recall but the f-measure is the most commonly used. Therefore, if we want to create a balanced classification model with the optimal balance of recall and precision, we then sort out to find the maximum f1-score of the model. A good f1 score means that you have low false positives and low false negatives which means you are correctly identifying real threats and you are not disturbed by the false alarms. The harmonic mean is used because it pushes extreme values rather than the simple mean. An f-score is perfect if its 1 and a total failure if it is 0.[19]

### 2.8 Multi-Label classification

Today, the world is not binary anymore. We have a lot of fusions going on for example in music, a song does not belong to only one genre anymore but rather two or three categories. Multi label classification of textual data is an important problem. We find the problem from news articles to emails, from music genres to movie

genres. Multi label classifications are increasingly required by modern applications. There is multi class classification and multi label classification, the differences between the two is that in multi label problems each label stands for a different classification task, but the tasks are to some extend related while in multi class problems the classes are very different from each other that it is impossible for them to exist together.

We have two categories for multi label classification, the first is problem transformation methods [21] and the other is the algorithm adaptation methods [20].

**i. Transformation Problems**

This method maps the multi label tasks into one or more single label learning tasks. The data will be broken down into binary classification problems one for each label which participates in the multi label problem. This method is called binary relevance. However the multi label can also be transformed to one single label problem using a target values for class attribute all the distinct subsets of labels. This method is called label power set.

| X | y1 | y2 | y3 | y4 |
|---|---|---|---|---|
| x1 | 0 | 1 | 1 | 0 |
| x2 | 1 | 0 | 0 | 0 |
| x3 | 0 | 1 | 0 | 0 |
| x4 | 0 | 1 | 1 | 0 |
| x5 | 1 | 1 | 1 | 1 |
| x6 | 0 | 1 | 0 | 0 |

Figure 2.5 **Transformation Problem1**

The above table shows a multi label problem which will be changed to one single problem as shown below.

| X | y1 |
|---|---|
| x1 | 1 |
| x2 | 2 |
| x3 | 3 |
| x4 | 1 |
| x5 | 4 |
| x6 | 3 |

Figure 2.6 **Transformation Problem2**

The label power sset gave each label that is present in the training set a unique class. Therefore for problem transformation, adapts the data to the algorithm.

### ii. Algorithm Adaptation

This method adapts a single label algorithm to produce multi label outputs. It extends specific learning algorithms in order to handle multi label data directly.

## 2.9 Multi-class classification

Multiclass classification is a classification method where by each training point belongs to one of N different classes. In multiclass classification, a sample can only belong to one class. An example of a multiclass classification is to classify a set of images of fruits that may be lemons, oranges or apples. A fruit can only belong to one class that is a fruit can be an orange only and not an orange and an apple at the same time.

Within the domain of natural language processing and text classification, a machine learning based classifier, the Naïve Bayes model is the most popular. The model makes an assumption that all the input features to the model are mutually independent. That is the presence of one particular feature does not affect the other.

When solving the multiclass problem, it is recommended that we use One vs All classifiers (OVA) or All vs All classifier (AVA), and not worry about anything else. The choice of OVA and AVA is mostly computational. Classification problems that have multi problems with imbalanced dataset presents a different challenge than a binary problem. The misrepresent distribution makes machine learning algorithms less effective, mostly in the predictions of the less represented class problems.

The existing multiclass classification methods can be classified into batch and online learning. The online learning algorithms models are built in sequential iterations. In batch learning algorithms, the data samples are required to be available before hand.

# 3. LITERATURE SURVEY

Online movies are often registered with their genres and plot summaries. These movie genres are sometimes inconsistent with the real story line of the movie. This is because the classification of these genres is done manually and involves the collection of users' suggestions sent to Internet Movie Database. In the past only a handful of people have done this automation process of the movie genre prediction using plot summaries.

Ka-Wing Ho investigated different ways to classify movie genres by synopsis [8]. Since a movie can belong to multiple genres, [8] used the SVM method to do the multi label classification. One group of movies that belong to one genre will be the positive sample and the rest will be the negative sample and then train the classifier with the two disjoint sets. Multi-label Knearest neighbor was also used. In this method, the knowledge of movies that belong to the same genre should share common keywords was used.

If one were to consider a movie 3 synopsis y as a point in the hyperspace, movies that have similar genres combination of y will be close to it. Mo Velayati [9] published a paper where he also focused on the movie genre classification. A Naïve Bayes classifier with multinomial model was used to achieve this task. In order to extract features the plot summary was converted into a "bag of words" where individual words were the features. Words were assigned term frequencies the number of times the term occurred in the entire corpus and also document frequency that showed how many documents have the word in their plot text. They then train the system, the purpose of the training was basically to calculate the threshold value and use it to decide whether a genre belonged with the movie or not.

Alex Blackstock and Matt Spitz [10] investigated Classifying Movie Scripts by Genre with a MEMM Using NLP-Based Features. Two evaluation metrics to analyze the performance of two separate classifiers, Naïve Bayes Classifier and a Maximum Entropy Markov Model Classifier were devised. During the investigation they faced challenge of inconsistent format of movie scripts and multiway classification presented by the fact that each movie script they had was labelled with several genres.

Sivaraman K S and Gautam Somappa of the university of Virginia [26] identified the genre of movies just by analyzing the visual features of its trailer. To achieve this, they used deep neural networks. [26] used visual geometry group neural network

(VGGNET) and took the output of the fully connected layer (FC layer) that then resulted in a 4096 dimensional vector. [26] model classified frames that had multiple faces or persons as drama and those that were dark and shadowy as horror.

A good poster attracts many people when used in advertising and in and other media. A good poster in the movie domain is that which communicate important aspects of a movie such as the cast, elements of the plot and the theme. Gabriel Barney and Kris Kaya [27] investigated if we could train a model to learn features of a poster that could successfully predict the genres of the movies it represents. They used the multi-label K-Nearest Neighbors. The KNN was implemented using humming loss as the main metric to evaluate the model's performance. They also implemented the one vs many classifier, which creates separate binary classifiers for each genre. When using the classifier for training, where the given genre is fitted against the other genres and the genre that gives the maximum score will be predicted.

Emotions are an important factor in the prediction of movies. Since emotions [28] play an important part in the classification of text, the inclusion of the six basic emotions will help in the processing of the plot summaries. [28] found the emptions that are associated with the genre of a movie.

| ***EMOTION*** | ***MOVIE GENRE*** |
|---|---|
| SAD | DRAMA |
| DISGUST | HORROR |
| ANGER | FAMILY |
| FEAR | SPORT, HORROR |
| SURPISE | DRAMA, FANTASY |
| JOY | THRILER |
| TRUST | WESTERN |
| ANTICIPATION | THRILER |

Figure 3.1 **emotion with corresponding genre**

In this project I will be using text to classify whether a movie belongs to a certain class or not. the classifier will be the one vs rest classifier with logistic regression. To solve the challenge of movies having several genres assigned to them, I will use the multi-label classifier [3].

# 4. DESIGN MODEL

In this project execution, an iterative design model is used. This is where the software application is split into smaller parts and the original implementation is simplified. The iterative model is a particular implementation of software development life cycle which changes continually and gains more complexity producing a wider feature set until the final system is deployed or completed.

Every milestone of the model is achieved in a specific time period and the output data of the previous release is used as the input of the next deliverable. This model is suitable for the project because the requirements are transparently defined and noted. The concepts of incremental will be used interchangeably.

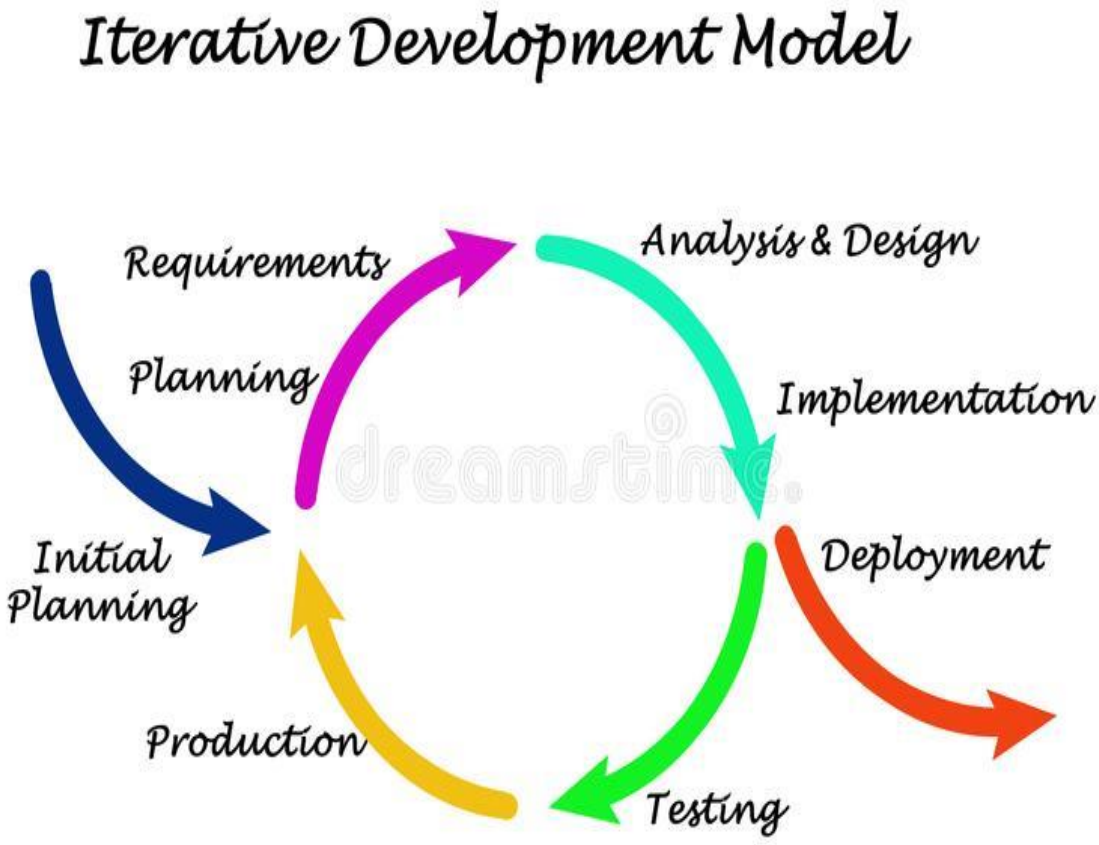


Figure 3.1 **The iterative model framework [25]**

# 5. IMPLEMEMTATION DETAILS

## 5.1 Corpus

The data for the corpus will be collected from [1] and will be comprising of the movie titles, plot summaries, genres, year etc. I will be using the data from this database especially the plot summaries, titles, year and also genres. The genres will be used to compare our results from our system with the actual results that the movie is said to belong with.

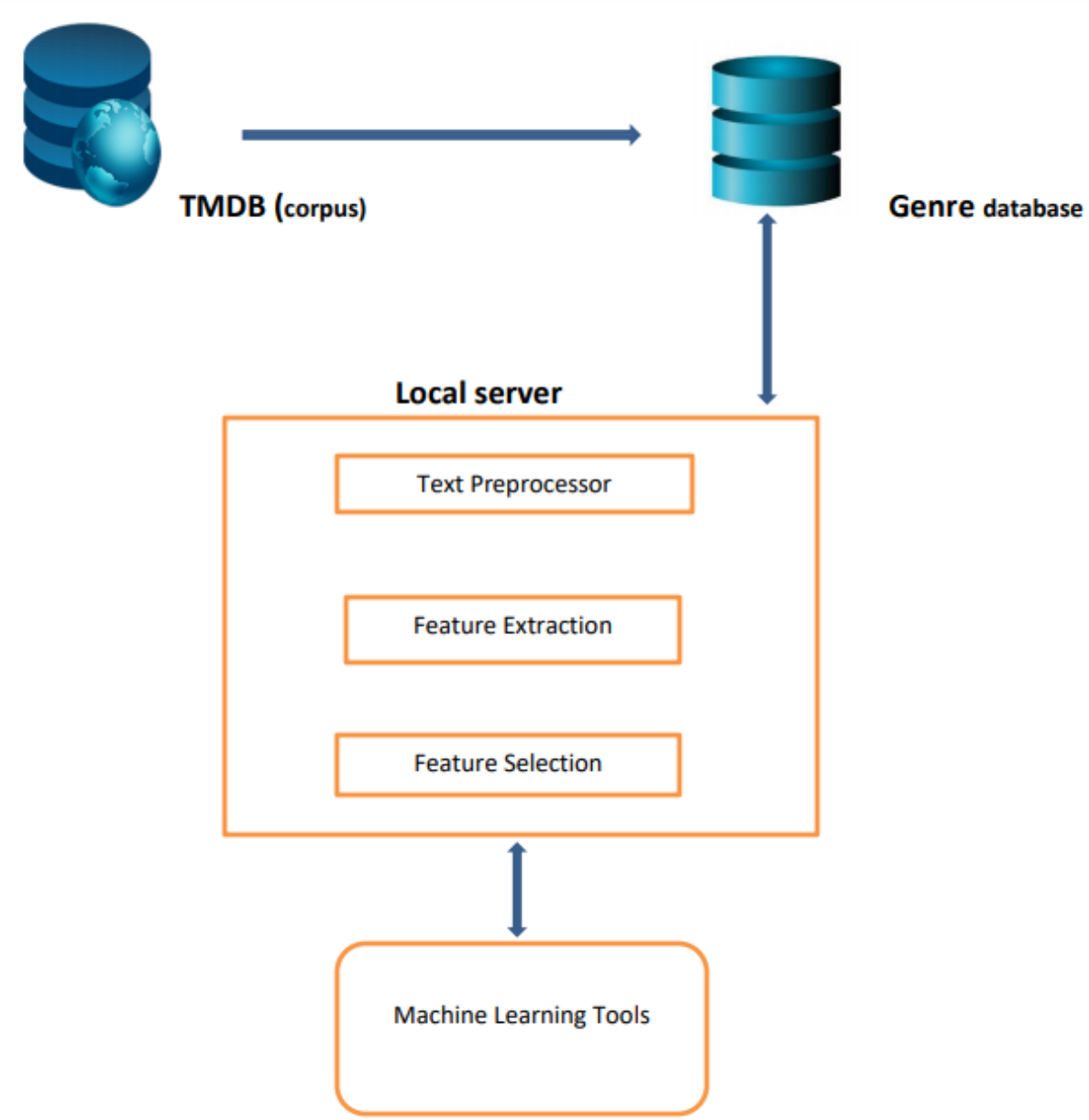


Figure 1.1 **Module Framework**

The data i used in this model training and testing is a data that i took from the movie database (TMDB) [2]. This data includes plot summaries, movie genres and movie cast. TMDB has movies that are over 10 000 and the API will allow you to take movies from one page which is 20 000 movies, hence our movie data will be limited and imbalanced. We chose to take data from TMDB because internet movie database IMDB had blocked its API service.

The data from [2] was retrieved using an Application Programming Interface (API)(https://api.themoviedb.org/3/discover/movie?api_key=<<api_key>>&language=enUS&sort_by=popularity.desc&include_adult=false&include_video=false&page=1) that is provided by the website for users to get movie data and also to edit some data like the genres of the movies, ratings etc. The movies was then stored in a json file awaiting to be processed and stored in a database.

### 5.2 Plots Database Module

In this module, the data will be accessed on a local machine which will make the natural language processing faster. For us not to be inconvinienced when the api from [2] has been cut, i decided to store the data in local database. The database is called plots and has all the data from the movie plots/summaries, the genres, the tags, the cast and also the emotions which will be used in the predictions of the movie genres taking into account the emotions in the plots. All this data will be in separate tables for ease of access and they will be related through index keys like primary keys and foreign keys since we are using a relational database management system [6].

```
def moviedb():
    conn = pymysql.connect(
        "localhost",
        "root",
        "1234",
        "plots")
    mycursor = conn.cursor()

    with open('MOVE12345.json', 'rb') as f:
        data = json.load(f)
```

```
  genre1 = genre2 = genre3 = genre4 = ""

  for number in range(1, 5):

      for t in data['movies']:

          if len(t['genre_ids']) == number:

              count = 0
              for r in t['genre_ids']:

                  if count == 0:
                      mycursor.execute("select  Genre_Name from genre where
Imdb_Genre_Id=%s" % r)
                      name = mycursor.fetchone()
                      genre1 = name
                      # print(name)
                      count = count + 1
                  elif count == 1:
                      mycursor.execute("select  Genre_Name from genre where
Imdb_Genre_Id=%s" % r)
                      name = mycursor.fetchone()
                      genre2 = name
                      # print(name)
                      count = count + 1
                  elif count == 2:
                      mycursor.execute("select  Genre_Name from genre where
Imdb_Genre_Id=%s" % r)
                      name = mycursor.fetchone()
                      genre3 = name
                      # print(name)
                      count = count + 1
                  elif count == 3:
```

```
                    mycursor.execute("select  Genre_Name from genre where
Imdb_Genre_Id=%s" % r)
                    name = mycursor.fetchone()
                    genre4 = name
                    # print(name)
                    count = count + 1


                print(t['id'], "   ", t['vote_count'], "    ", t['title'], "   ", genre4, genre3, genre2,
genre1,
                      t['overview'], "   ", t['release_date'])
                try:
                    sql = "INSERT INTO
movie_genre(Imdb_Id,Vote_count,Title,Genre_ids0,Overview,Release_date)
VALUES(%s,%s,%s,%s,%s,%s)"
                    val = (t['id'], t['vote_count'], t['title'], genre1, t['overview'],
t['release_date'])
                    mycursor.execute(sql, val)
                    conn.commit()
                    print(t['title'] + " has been commited")
                except:
                    conn.rollback()
                    print(t['title'] + " has been rolled back")
```

**Populating the movie_genre database table**

The above code shows how the table movie_genre was populated with the data that was taken from [2].

Next to be populated was the cast table. The cast table holds the data about the cast crew of the movies. This data was also taken from the movie database [2] using another API for cast that is provided by [2] also. This data includes the cast id, the character name, the person's real name and gender.

```
def get_cast(mid):
    # num = 4108
```

```
    try:
        response = requests.get(

'https://api.themoviedb.org/3/movie/%d/credits?api_key=bc96a52a6cb35498e2fe6f3e
6dffeaec'
                                % mid)
        data = json.loads(response.text)
        print(data['cast'])
        castdb(data, int(mid))
    except:
        data = []
        castdb(data, int(mid))
```

**Api to get the cast of a particular movie**

After acquiring the data for cast from [2] I then added it the cast table which is in the plots database on the local machine for ease of access.

```
def castdb(data, mid):

    conn = pymysql.connect(
        "localhost",
        "root",
        "1234",
        "plots"
    )
    mycursor = conn.cursor()

    # print('printin %d ' % len(data['cast']))

    try:
        for t in data['cast']:
```

```
            cast = t['cast_id']
            pid = t['id']
            character = t['character']
            name = t['name']
            gender = t['gender']


            if gender == 1:
                gender = 'female'
            elif gender == 2:
                gender = 'male'

            print(name, cast, gender, character, mid, pid)

            try:
                sql = "INSERT INTO movie_cast( tmdb_id, cast_id, actor_id, movie_name,
real_name, gender ) VALUES(%s, %s,%s,%s,%s,%s)"
                val = (mid, cast, pid, character, name, gender)
                mycursor.execute(sql, val)
                conn.commit()
                print("%d and %s has been commited\n" % (mid, name))

            except:
                print("%d and %s has been rolled back\n" % (mid, name))
                conn.rollback()
    except:
        pass

    conn.close()
```

**populating the movie_cast table**

### 5.3 Entity Relational Diagram

An entity relationship diagram (ERD) displays the entity sets relationships stored in a database. In this system MYSQL database was used. The figure above describes how entities relate to one another. The above design is for the movies and their genres with some additional features like tags and emotion words in the emotion entity. It is the plots database with the data that was used to train the model.

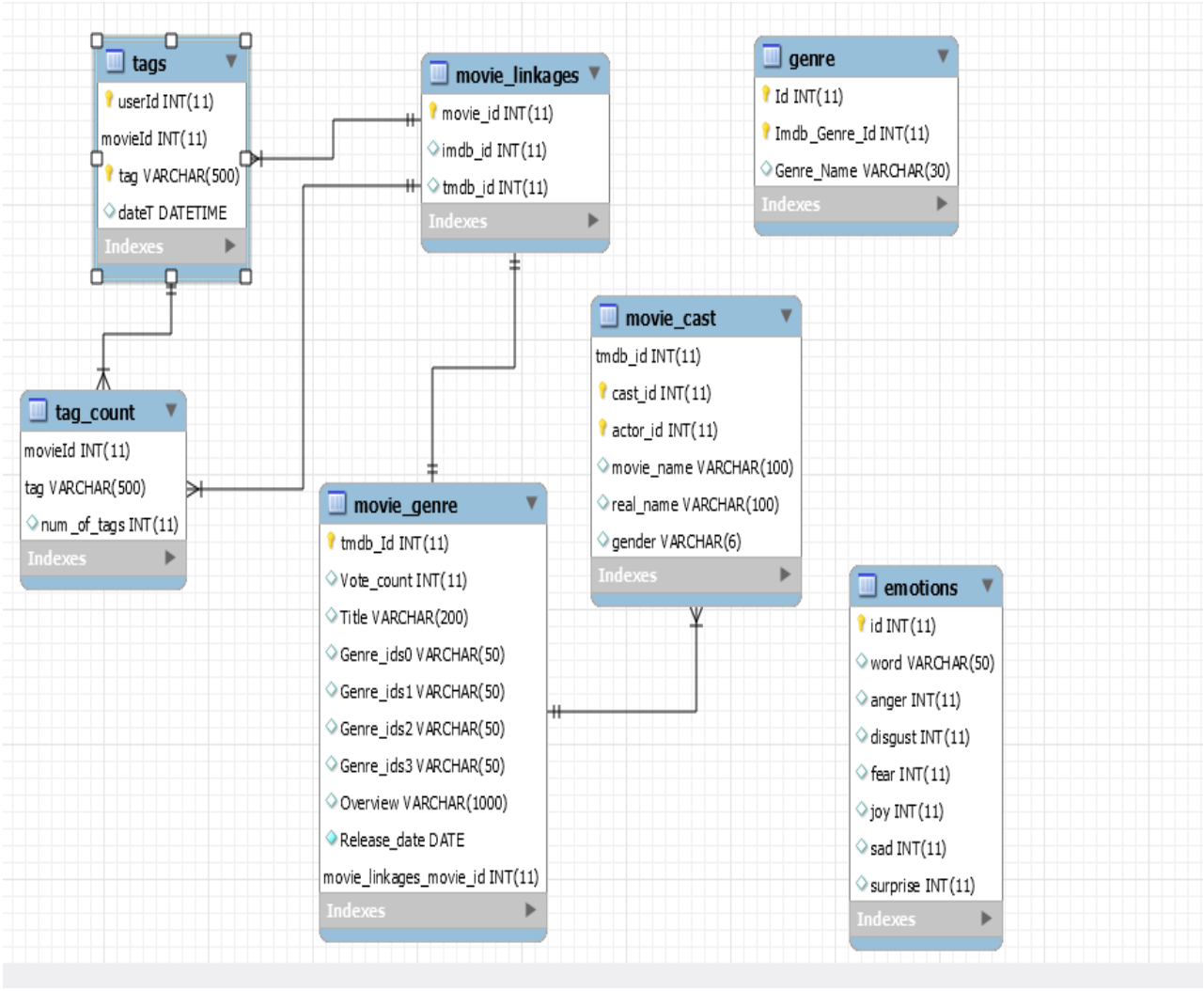


Figure 4.1 **Entity Relational Diagram**

### 5.4 Database Connection.

The database was developed using MySQL where it uses PyMySQL extension which is a python connector of MySQL. This project database was connected using PyMySQL because it is fast and is less complicated since it has straight forward documentation.

```
import pymysql

conn = pymysql.connect(
    "localhost",
    "root",
    "1234",
    "plots"
)
curr = conn.cursor()
```

**MySQL database connection code**

Database connection above was implemented in every function that require the connection of the database, and the query to be executed will follow suit as follows.

```
curr.execute(
        "select Genre_ids0, Genre_ids1, Genre_ids2, Genre_ids3 from movie_genre"
)
data = [{"genre1": col1, "genre2": col2, "genre3": col3, "genre4": col4} for (col1,
col2, col3, col4) in curr.fetchall()]
```

**MySQL Query Execution Code**

Database connection was done with pymysql which need four variables which are localhost, username, password and schema name.

### 5.5 Third Parties

The help of other libraries was needed in this project so that it makes implementation easier. The Libraries used will have the algorithms that are needed when calculating some mathematical processes of the project. These are as follows:

```
import pymysql
import json
import csv
```

```
import sys
import pandas as pd
from tqdm import tqdm
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import KFold
```

**Third parties libraries**

PyMySQL is the library that is responsible in the connection of the database.

Json helps with the creation and to read json objects to be used in the code.

Sys comes with python language itself and it help in halting the execution of the script. It is used as follows:

```
. . .
sys.exit(0)
```

**Sys usage**

### 5.6 Text Processors module

This module is responsible for processing the text from the plot summaries. The processing involves the elimination of stop words from the text to achieve a cleaner text for further processing stages.

The movie plots have to be consistent with genres for training to take place. To get this data to be in sync, i wrote the plots to a file with their corresponding movie ids

from [2]. I took the genres and placed them in a json object which i then convert to a list and place them an array.

```
for sample in data:
    sample = json.dumps(sample)
    genres.append(list(json.loads(sample).values()))

gent = []

for item in genres:
    # eliminating null values, that is the genre columns that are null
    gen = [n for n in item if n]
    gent.append(gen)

print(gent[1])
return gent
```

**creating array for genres**

In the above code, the final array for the genres is called gent. It is the gent which i will merge with the plot summaries to finalize the standardization part and pass to the next step.

```
movies = pd.DataFrame({'tmdb_id': movie_id, 'plot': plot, 'genre': gent})
```

**joining movie id, plot and genres**

### 5.7 Processing the plots

The plots have to be cleaned and stop words removed from them. Stop words are a set of commonly used words in any language, for eaxample english has words like the, a is, these word are considered as stop words beacause they are non-discriminative and are just additional memory overhaed.

For data to be fitted in a machine learning model we have to clean the data. Cleaning raw data includes handling punctuations and casing also spliting the text into words.

```
def c_plot(text):
    text = re.sub("\'", "", text)
    text = text.strip(".,")
    text = re.sub("[^a-zA-Z]", " ", text)
    text = ' '.join(text.split())
    text = text.lower()

    return text
```

**cleaning text and casing**

After the cleaning process our data will have to be further prepared for training and that is to remove all the non-informative, destructive words from the texts.

```
def stop_words_fn(word):
    stop_words = set(stopwords.words('english'))
    tokens = word_tokenize(word)
    # result = [r for r in tokens if not r in stop_words]
    # print(result)
    result = []

    for w in tokens:
        if w not in stop_words:
            result.append(w)

    word = ' '.join([str(t) for t in result])
    # print(tokens)
    return word
```

**removing stopwords**

In the above code, the text is split into words using the word_tokenize function and then each word compared to the list of stop words that are found in the English language. If the word is not in the list of stop words then it is a discriminative word

which will help in the training and classification of our model and will be placed in the result array. After checking for stop words, the words will be joined together to form a sentence again.

### 5.8 Feature Extraction module

This module is for creating a new set of features that captures most of the useful information that includes the keywords by combination of the existing features using various machine learning algorithms.

For the training of the data for our model, we have to have two sets. The two sets will be split using cross validation for this model. These sets will be the training set and the validation set. The goal for this is to come up with a model that generalizes well to new data when exposed. Since we have limited data we have to prevent our model from overfitting. Therefore we will run the folds and calculate some estimations for each fold then get the average of them all.

```
kf = KFold(n_splits=5, random_state=12, shuffle=True)
# 5 splits = 80:20 split
for tr_index, ts_index in kf.split(movies['new_plot']):

    xtrain = movies['new_plot'][tr_index]
    xval = movies['new_plot'][ts_index]
    ytrain = y[tr_index]
    yval = y[ts_index]
```

**k-fold cross validation**

the k-fold class is instantiated with the number of splits which is 5 because the model is trained using 80:20 ratio of the training and validation sets. The random state ensures that when ever we execute the script with the random state assigned to the number 12 we will get the exact same outcome.

### 5.9 Feature Selection module

This module is responsible for dimensionality reduction, especially when dealing with a lot of features (words) there is need of dimensionality reduction. This will be achieved by selecting a subset of the existing features without a transformation.

### 5.9.1 Calculating the term frequency and inverse document frequency

Tf-idf is an abriviation for term frequency-inverse document frequency. This is a measure used in information retrieval and text mining. The statistical measure is used to get the importance of a word in a corpus. Term frequency [23] is a measssure to see how frequently a term occurs in a document. The term frequency is fount using the below formula.

$$TF = \frac{TERM\ COUNT}{DOCUMENT\ WORD\ COUNT}$$

Equation 2.1 **Term Frequency**

Inverse document frequency [23] is the measure of how important a term is. The more a term is used in the documents the less important it becomes. Idf is found as follows.

$$IDF = \log \frac{NUMBER\ OF\ DOCUMENTS}{NUMBER\ OF\ DOCUMENTS\ WITH\ TERM}$$

Equation 2.2 **IDF**

Therefore to get the tf-idf [23] values we multiply the tf with the idf as follows.

$$W_{i,j} = TF_{i,j} * \log\left(\frac{N}{DF_i}\right)$$

$TF_{i,j}$ = number of occurences of i in j

$DF_i$ = number of documents containing i

N = total number of documents

Equation 2.3 **Tf-Idf**

Using python the tf-idf is calculated using a tfidf vectorizer from scikit learn as follows.

```
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
###FOR THE VALIDATION DATA SET
xval_tfidf = tfidf_vectorizer.transform(xval)

# sys.exit(0)
```

**calculating tfidf for training and validation sets**

The fit transform will learn the vocabulary and then return term document matrix. The transform function will just transform documents to document-term matrix.

### 5.10 Relavance of emotion words in our plots

We have six basic emotions which are Anger, Joy, Sad, Disgust, Surprise and Fear. These emotions are seen from facial expressions to even the text we write. When writing texts, people often express how they will be feeling through how they will be picking their words.

In the movie industry these emotions are also categorised to each movie genres as shown belowWe will read the emotion words that belong to each emotion.

```
feature_names = tfidf_vectorizer.get_feature_names()

feature_names.append("anger")
feature_names.append("disgust")
feature_names.append("fear")
feature_names.append("joy")
feature_names.append("sad")
feature_names.append("surprise")

# include emotions
xtrain_tfidf_new = normalize(xtrain_tfidf, xtrain)
```

**adding emotion categories to the feature name vector**

After adding the categories to the tfidf vector, we then normalize the emotion words and merge them with the tfidf values for the training to begin.

```
def normalize(xtrain_tfidf, xtrain):
    xtrain_tfidf_new = []
    count = 0
    new_xtrain_tfidf = xtrain_tfidf.toarray()


    for i in xtrain:
```

```
        if count < len(xtrain):
            n = word_tokenize(i)
            print(i)
            print(n)
            a1 = additional_features(n)
            print(len(new_xtrain_tfidf[count]))
            new_xtrain_tfidf1 = np.append(new_xtrain_tfidf[count], a1)
            xtrain_tfidf_new.append(new_xtrain_tfidf1)


            print(count)
            print("\n")
            count = count + 1

# next work with more than 1000 feature names to see the maximum number of
feaures that it can accommodate !!!

    # xtrain_tfidf_new = np.asmatrix(xtrain_tfidf_new)
    xtrain_tfidf_new = sparse.csr_matrix(xtrain_tfidf_new)

    return xtrain_tfidf_new
```

**Normalizing the emotions**

For us to compare the words in the plots with the words that belongs to each emotion we use the additional_features function. In the normalization function returns a sparse matrix with new tfidf values. The words will be normalized by counting the number of each emotion words and then dividing that by the length of the plot.

```
def additional_features(array):
    feat = []
    add = []

    anger = disgust = fear = joy = sad = surprise = 0.00
```

```
    for i in array:
        conn = pymysql.connect(
            "localhost",
            "root",
            "1234",
            "plots"
        )
        curr = conn.cursor()
        curr.execute(
            "select anger, disgust, fear, joy, sad, surprise from emotions where word =
'%s'" % i
        )
        data = curr.fetchone()
        count = 1
        try:
            if data:
                for item in list(data):
                    if count == 1:
                        anger = item + anger
                        count = count + 1
                    elif count == 2:
                        disgust = item + disgust
                        count = count + 1


                    elif count == 3:
                        fear = fear + item
                        count = count + 1
                    elif count == 4:
                        joy = joy + item
                        count = count + 1
                    elif count == 5:
                        sad = sad + item
                        count = count + 1
```

```
            elif count == 6:
                surprise = surprise + item
                count = count + 1
    except:
        continue
add.append(anger)
add.append(disgust)
add.append(fear)
add.append(joy)
add.append(sad)
add.append(surprise)
for item in add:
    if len(array) > 0:
        item = round(item, 3) / len(array)
        feat.append(round(item, 3))
    else:
        item = round(item, 3)
        feat.append(item)

return feat
```

**additional features**

The additional features function will accept an array which will be the tokenized plot. Each word will then be checked if it exists in the emotion database. If it exist a tuple with either 0 or 1 under the appropriate emotion will be returned and will count the number of words that represents each emotion in the plot. When we divide by the length of the plot, the number is them rounded to 3 decimal places and appended to an array which will be returned for the conversion to sparse matrix to take place.

After we have the sparse matrix, classification will take place and the model's performance will be noted and if tweaking is required, it will take place at this stage as well.

### 5.11 Machine Learning Tools

In this module we will be using the machine learning tools provided by the python language from libraries like scikit learn, scipy and many more to predict and vectorize the data set from our corpus.

#### 5.11.1 Binarizing the genre classes

When binarizing the genres we will be taking all the genres we have in our corpus and then come up with only the distict classes that are in the data set the we transform the genres with placing a 1 when the movie belongs to the class and 0 otherwise. The multilabel binarizer from scikit learn will help with te binarization of the labels. The binarizer will be used to get the labels from the predicted movie matrix and be able to see which genre a movie belongs to.

```
multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit(movies['genre'])

# transform target variable
y = multilabel_binarizer.transform(movies['genre'])
```

**multi-label binarizer**

#### 5.11.2 Logistics Regression and classification

Logistic regression is the most efficient mechanism for calculating probabilities. I used the logistic regerssion as an estimator for the classification problem. On its own it is just a binary clasifier which cannot handle target vectors with more than two classes. The classifer that i used is the one vs rest classifier [24].

```
lr = LogisticRegression()
# clf = OneVsRestClassifier(lr)
clf = OneVsRestClassifier(lr)

# sys.exit(0)
# fit model on train data
```

```
clf.fit(xtrain_tfidf, ytrain)
```

**one vs rest logistic regression**

### 5.12 Predictions

For our model to start predicting we have to first classify our training data and then test it using the validation set which we have split in the initial stages.

```
y_pred = clf.predict(xval_tfidf)
print("this is y_pred", y_pred)
multilabel_binarizer.inverse_transform(y_pred)
# evaluate performance
print("First fscore: ", f1_score(yval, y_pred, average="micro"))
print("Precision score: ", precision_score(yval, y_pred, average="micro"))
print("Accuracy score: ", accuracy_score(yval, y_pred))
print("Recall score: ", recall_score(yval, y_pred, average="micro"))

# predict probabilities
y_pred_prob = clf.predict_proba(xval_tfidf)
print(y_pred_prob)
```

**predicting genres of the validation set**

On this stage, we will be executing our code to find the right parameters that will make the model efficient. This is where we will add or remove features to see the right amount that will work best. This is called model tweaking, it is defined as adjusting the model from the learning rate to addition of features.

In the above code a predict fuction is being called from the classifier and the tfidf values of the validation set are passed fro the predictions to happen. The results will then be fit to the multi-label binarizer to find the classes each movie in the validation set belongs to. We then claculate the F measure of the model.

The F-score shows how our model is performing in terms of the precision and recall of the model.

### 5.13 Thresholding

Since logistic regression returns a probability and also our labels have been binarized, we have to find a way to map the logistic regeression value to a binary category. For me to achieve this, i then introduced a decision threshold also known as the classification threshold. A value that will be above this threshold value will be taken as the value that will make our model efficient. The predict function in the classifier has a threshold value of 0.5. threfore if we are to find a model that is efficient and has a better F-measure we will have to tune the threshold.

```
y_pred_prob = clf.predict_proba(xval_tfidf)
print(y_pred_prob)

threshold = 0.3

# threshold value 0.4_fscore = 0.533 ### 0.2_fscore = 0.583 ### 0.3_fscore = 0.574

new_pred = (y_pred_prob >= threshold).astype(int)


print(f1_score(yval, new_pred, average="micro"))
print("Precision score: ", precision_score(yval, new_pred, average="micro"))
print("Accuracy score: ", accuracy_score(yval, new_pred))
print("Recall score: ", recall_score(yval, new_pred, average="micro"))
```

**tuning the threshold**

When tunning the threshold, the value of F-measure will change pointing out the threshold that is effective. Tuning the threshold is different from tuning the hyperparameters like the learning rate.

### 5.14 Predicting incoming data

When our model is ready, with all the perfomance statistics noted, we will start to predict the genres from different movie plots in the external environment. For this to

happen we have to have a fuction that will go through the processing of the incoming raw data and then fit it to our classifier for our model to predict which label class it belongs to.

```
def predict(plot):
    plot_t = c_plot(plot)
    plot_t = stop_words_fn(plot_t)
    plot_vec = tfidf_vectorizer.transform([plot_t])
    q_pred = clf.predict(plot_vec)
    y_pred_prob = clf.predict_proba(plot_vec)
    qw_pred =  (y_pred_prob >= threshold).astype(int)
    print('\nin the inference\n')
    return multilabel_binarizer.inverse_transform(qw_pred)
```

**predicting incoming raw plot summary**

The raw plot summary will first go through the cleaning process. This is where punctuations are removed and casing taken care of. The next stage will be removing the stop words like prepositions and the non discriminatory words. We then calculate the term frequency inverse document frequency of the plot.

The plot will then have to be fit in our model for the predictions to take place and use the threshold that our model perfoms better at. We will then transform the binaries to classes and see which label the plot summary belongs to.

### 5.15 The user interface module

This is where one will input data, this data will be the plot summaries of a movie. The genres that will be the output will be produced on the same interface.

The person that wishes to get the genre of the movie will have to copy the plot summary of the movie he wants to know the genre and paste the text in the plot field and press predict button on the bottom left just above the genre field. The keywords of the plot and the predicted genre will then be shown in their respective fields


Figure 6.1 **Genre Prediction Model User Interface**

```
from tkinter import *
import test

root = Tk()
# root = Frame(master=root1, bg="blue")
root.configure(background='gray40')

root.title("App Beezy")

#root.pack_propagate(1)
# name = StringVar()

# create the widgets
header = Label(root, text="APP BEEZY", fg="gold", font="Courier 44 bold")
```

```
go_btn = Button(root, text="Predict", fg="blue", font="arial 14 bold", width=20,
command=lambda: mint())

title_label = Label(root, text="Title", fg="red")
title_label.config(font=("Arial", 20))
title_entry = Text(root, width=90, font='14', height=2)

plot_label = Label(root, text="Plot", fg="red")
plot_label.config(font=("Arial", 20))
plot_entry = Text(root, width=100, height=10)

keyword_label = Label(root, text="Keywords", fg="red")
keyword_label.config(font=("Arial", 20))
keyword_entry = Text(root, width=100, height=5)
genre_label = Label(root, text="Genre", fg="maroon")
genre_label.config(font=("Arial", 20))
genre_entry = Text(root, width=100, height=2)
# place the widgets
header.grid(row=1, column=30, pady=(10, 5))
title_label.grid(row=4, column=15, sticky=E,  padx=(10, 5))
title_entry.grid(row=4, column=30, pady=(12, 2), padx=(10, 5))
plot_entry.grid(row=7, column=30, pady=(12, 2), padx=(10, 5))
plot_label.grid(row=7, column=15, sticky=E, pady=(10, 5), padx=(10, 5))
keyword_entry.grid(row=10, column=30, pady=(12, 2), padx=(10, 5))
keyword_label.grid(row=10, column=15, sticky=E, pady=(10, 5), padx=(10, 5))
go_btn.grid(row=12, column=30, sticky=E, pady=(10, 5), padx=(5, 5))
genre_label.grid(row=15, column=15, sticky=E, pady=(15, 10), padx=(10, 5))
genre_entry.grid(row=15, column=30, pady=(15, 10), padx=(10, 5))
```

**The user interface code**

When the predict button is pressed, the mint function is called. The mint function is responsible for displaying information in the appropriate fields on our interface and the function will interact with the model functions to get data.

```
def mint():

    #name = ent2.get("1.0", END)
    title_entry.delete("1.0", END)
    genre_entry.delete("1.0", END)
    keyword_entry.delete("1.0", END)
    keyword_entry.insert(0.0,
test.word_tokenize(test.clean_text(test.remove_stopwords(plot_entry.get("1.0",
END)))))
    title_entry.insert(0.0, "COULDN'T FIND THE TITLE !!")
    genre_entry.insert(0.0, test.infer_tags(plot_entry.get("1.0", END)))

    print('gui')
    # return name
```

**Get Data and display**

### 5.16 Results and Discussion

Our model perfomed best when we used the cross validation technique unlike when we tested it with just spliting our data once. For the performance calculations, i tested with average micro, macro and weighted and the average micro gave the best results with the F measure ranging above 50% (0.50).

The highest F-measure reached was when we introduced thresholding and with different threshold values the F-measure values changed as well. I then tested the data we have using cross validation and tunining the threshold, the results were much better. While testing these we also changed the feature size varying it between the values of 5000, 10000 and 15000. We saw that with threshold of 0.4 and the maximum document frequency(max_df) of 0.1, we would get the best results as shown in the figures below.

The document frequency(_df) of 0.9 and 0.4 gave identical results in all the attributes which are f-score, precision, recall and accuracy, but when we used 0.1 as the max_df the results were much better. This symbolises that when we eliminate features that are greater than 0.9 and 0.4 respectively our model's performance will be negatively impacted. We also saw that the words that are in more than 0.4 are also above the 0.9 threshold since we got identical results when we tuned these threshold.

With the number of features being changed, the model was also reducing its perfomance with the increasing number of features. However, when we droped this number beyond the 5000 mark, the perfomance dropped as well. This observation showed that as we use too much features our model will then produce too many false positives than true positives and also that is the case with the fewer features.

| Doc_freq | Split | feature_si | fscore | precision_ | recall_sco | accuracy |
|---|---|---|---|---|---|---|
| 0.1 | 5 | 5000 | 0.320337 | 0.659886 | 0.25199 | 0.151873 |
| 0.1 | 5 | 10000 | 0.301502 | 0.664065 | 0.236038 | 0.153064 |
| 0.1 | 5 | 15000 | 0.291566 | 0.660236 | 0.228562 | 0.151061 |
| 0.4 | 5 | 5000 | 0.321401 | 0.661727 | 0.253149 | 0.151711 |
| 0.4 | 5 | 10000 | 0.302781 | 0.663766 | 0.237202 | 0.153876 |
| 0.4 | 5 | 15000 | 0.293475 | 0.665474 | 0.230012 | 0.151819 |
| 0.9 | 5 | 5000 | 0.321401 | 0.661727 | 0.253149 | 0.151711 |
| 0.9 | 5 | 10000 | 0.302781 | 0.663766 | 0.237202 | 0.153876 |
| 0.9 | 5 | 15000 | 0.293475 | 0.665474 | 0.230012 | 0.151819 |

Figure 7.1 **Macro**

The above figure shows the average macro calculations. As the figure shows, when features size increases the f-measure decreases.

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| Doc_freq | Split | feature_si | fscore | precision_ | recall_sco | accuracy | |
| 0.9 | 5 | 5000 | 0.487247 | 0.663227 | 0.446778 | 0.151711 | |
| 0.9 | 5 | 10000 | 0.478296 | 0.674485 | 0.43684 | 0.153876 | |
| 0.9 | 5 | 15000 | 0.472877 | 0.676392 | 0.432152 | 0.151819 | |
| 0.4 | 5 | 5000 | 0.487247 | 0.663227 | 0.446778 | 0.151711 | |
| 0.4 | 5 | 10000 | 0.478296 | 0.674485 | 0.43684 | 0.153876 | |
| 0.4 | 5 | 15000 | 0.472877 | 0.676392 | 0.432152 | 0.151819 | |
| 0.1 | 5 | 5000 | 0.485821 | 0.6622 | 0.44513 | 0.151873 | |
| 0.1 | 5 | 10000 | 0.476698 | 0.673449 | 0.435245 | 0.153064 | |
| 0.1 | 5 | 15000 | 0.470924 | 0.674224 | 0.430438 | 0.151061 | |

Figure 7.2 **Weighted**

| | | | TO DEMONSTRATE THE IMPORTANCE OF EMOTIONS | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Doc_freq | threshold | Split | feature_si | fscore | precision_ | recall_sco | accuracy |
| TEST 1 | 0.9 | 0.4 | 5 | 15000 | 0.522928 | 0.662004 | 0.432152 | 0.151819 |
| TEST 2 | 0.4 | 0.4 | 5 | 15000 | 0.522928 | 0.662004 | 0.432152 | 0.151819 |
| TEST 3 | 0.1 | 0.4 | 5 | 15000 | 0.521135 | 0.660291 | 0.430438 | 0.151061 |
| TEST 4 | 0.01 | 0.4 | 5 | 15000 | 0.476465 | 0.644704 | 0.377898 | 0.130656 |
| | | | | | | | | |
| TEST 1 | 0.9 | 0.4 | 5 | 10000 | 0.526122 | 0.661313 | 0.43684 | 0.153876 |
| TEST 2 | 0.4 | 0.4 | 5 | 10000 | 0.526122 | 0.661313 | 0.43684 | 0.153876 |
| TEST 3 | 0.1 | 0.4 | 5 | 10000 | 0.524544 | 0.659985 | 0.435245 | 0.153064 |
| TEST 4 | 0.01 | 0.4 | 5 | 10000 | 0.481051 | 0.642689 | 0.384418 | 0.130927 |
| TEST 5 | 0.1 | 0.4 | 5 | 10006 | 0.526908 | 0.656571 | 0.44004 | 0.152144 |
| | | | | | | | | |
| TEST 1 | 0.9 | 0.4 | 5 | 5000 | 0.530393 | 0.652572 | 0.446778 | 0.151711 |
| TEST 2 | 0.4 | 0.4 | 5 | 5000 | 0.530393 | 0.652572 | 0.446778 | 0.151711 |
| TEST 3 | 0.1 | 0.4 | 5 | 5000 | 0.528945 | 0.6517 | 0.44513 | 0.151873 |
| TEST 4 | 0.01 | 0.4 | 5 | 5000 | 0.4852 | 0.632321 | 0.39366 | 0.131468 |
| TEST 5 | 0.1 | 0.4 | 5 | 5006 | 0.531739 | 0.650379 | 0.449744 | 0.151549 |
| | | | | | | | | |
| TEST 1 | 0.9 | 0.4 | 5 | 1000 | 0.509837 | 0.609005 | 0.438465 | 0.140723 |
| TEST 2 | 0.4 | 0.4 | 5 | 1000 | 0.509837 | 0.609005 | 0.438465 | 0.140723 |
| TEST 3 | 0.1 | 0.4 | 5 | 1000 | 0.508866 | 0.607622 | 0.437747 | 0.138937 |
| TEST 4 | 0.01 | 0.4 | 5 | 1000 | 0.45704 | 0.573467 | 0.379946 | 0.112254 |
| TEST 5 | 0.1 | 0.4 | 5 | 1006 | 0.510592 | 0.606139 | 0.441095 | 0.139154 |

Figure 7.3 **Micro**

Since the results showed that micro was the best average to use, we then tested the rest of the data using the same average parameter. We then looked at the relevance of the emotions in the plot summary since words can also signify if the writer was angry, happy etc.

The results showed that if we take emotions into consideration, our model will perform much better than when we just use the frequencies of the words in different documents. The features with an additional 6 to them are the ones that we included the emotions.

# 6. CONCLUSION AND FUTURE RECOMMENDATION

## 6.1. Benefits

Predicting movie genres based on plot summaries is a project that will see many movie lovers in the world stop wasting time looking for the genres while watching the movies but rather they will use the system to decide which genre they want using the plot summaries provided. This will also enable the people who set genres to set accurate and precise movie genres for the viewers. Since I will be using python, a language I am starting to learn to write such big programs, it will help me intensify my knowledge on the language. My knowledge of algorithms 9 will be taken to a whole new level, this is because during the course of the project, I will be using algorithms of different types and testing their time complexity on different levels of development. The natural language processing field requires one to have an extensive knowledge of machine learning techniques which I will also acquire during the course of the project. Many organizations in the world are now using MySQL as their preferred database management system. With this knowledge, I decided to prepare myself for the future by just using MySQL database on this project.

## 6.2. Future works

The movie genre prediction based on plot summary system can be expanded in the future by adding the movie success predictor. The success of the movie can be predicted using information from the Internet Movie Database (IMDB). This data includes the casting crew, the director, main actors in the movie etc. We can also add the rating system which will rate movies based on the user comments. This will take the comments from users of the same IMDB and based on what they will be commenting whether negative or positive will then be used with natural language processing and provide the ratings of the movie.

# 7. References


[1]: IMDB data. ftp://ftp.fu-berlin.de/pub/misc/movies/database/frozendata/

[2]: The Movie Database (TMDB). Retrieved from (TMDB API) https://www.themoviedb.org/documentation/api /

[3]: Dive into multi-label classification...! Kartik Nooney (June 7, 2018) https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification384c40229bff/

[4]: Why Is Python So Good for AI, Machine Learning and Deep Learning? Jakub Protasiewicz (Aug 31, 2018) https://www.netguru.com/blog/why-is-python-so-good-for-ai-machine-learningand-deep-learning/

[5]: Python Programming Language official page. https://www.python.org/

[6]: MySQL Database Management System. https://dev.mysql.com/

[7]: PyCharm Ide. https://www.jetbrains.com/

[8]: Movies genres classification by synopsis. Ko-Wing Ho, 2011.

[9]: Movie genre classification. Mo Velayati, 2017. https://mvelayati.com/2017/07/19/moviegenre-classification/

[10]: Classifying Movie Scripts by Genre with a MEMM Using NLP-Based Features. Alex Blackstock Matt Spitz, 2008.

[11]: https://hackr.io/blog/python-programming-language

[12]: https://hackernoon.com/why-python-used-for-machine-learning-u13f922ug

[13]: Machine learning importance.

https://www.sas.com/en_us/insights/analytics/machine-learning.html

[14]: Overfitting problems

https://medium.com/@george.drakos62/cross-validation-70289113a072

https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

[15]: https://www.ibm.com/topics/machine-learning

[16]: Cross validation https://stats.stackexchange.com

[17]: Recall and Precision https://pathmind.com/images/wiki/precision.png

[18]: F1-score
https://miro.medium.com/max/376/1*UJxVqLnbSj42eRhasKeLOA.png

[19]: Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined. Chris Nicholson, (2019)

https://pathmind.com/wiki/accuracy-precision-recall-f1

[20]: Multi label classification. Grigorios Tsoumakas, Ioannis Katakis
http://dml.cs.byu.edu/~cgc/docs/atdm/Readings/MLM-Overview.pdf

[21]: Term frequency and inverse document frequency http://www.tfidf.com/

[22]: H. Wu and R. Luk and K. Wong and K. Kwok. "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, 26 (3). 2008.

[23]: Term frequency https://surferseo.com/blog/tf-idf-seo-prominent-words-phrases/

[24]: One vs Rest Logistic Regression
https://chrisalbon.com/machine_learning/logistic_regression/one-vs-rest_logistic_regression/. 2017.

[25]: Iterative model https://www.dreamstime.com/illustration/iterative-model.html

[26]: Movie trailer classification using Deep Neural Networks. Sivaraman K S & Gautam Somappa Dept of Computer Science University of Virginia.

[27]: Predicting Genre from Movie Posters. Gabriel Barney (barneyga) and Kris Kaya (kkaya23)

[28]: Emotions associated with the genre of a movie.
https://www.geeksforgeeks.org/movie-recommendation-based-emotion-python/