MOVIE GENRE CLASSIFICATION

USING NLP TECHNIQUES

PROJECT REPORT

UNIVERSITY OF NORTH TEXAS

DEPARTMENT OF COMPUTER SCIENCE

INFORMATION RETRIEVAL AND WEB SEARCH

Movie Genre Classification

using Natural Language Processing Techniques

by

Group No. 14

Lakshmi Keerthi Gopireddy (Student ID: 11653876)
LakshmiKeerthiGopireddy@my.unt.edu
Major: Artificial Intelligence

Raghunandan Reddy Tekulapally (Student ID: 11717166)
raghunandanreddytekulapally@my.unt.edu
Major: Computer Science

Dinesh Reddy Bollampally (Student ID: 11706437)
DineshReddyBollampally@my.unt.edu
Major: Computer Science

Pavan Kumar Allam (Student ID: 11706438)
pavankumarallam@my.unt.edu
Major: Computer Science

Individual Roles and Responsibilities

Task	Keerthi	Raghunandan	Dinesh	Pavan
Python Programming	Yes	Yes	Yes	Yes
Dataset selection	-	-	Yes	Yes
Project Proposal	Yes	Yes	-	-
Project Proposal PPT	-	Yes	-	Yes
Domain and Data Understanding	Yes	-	-	Yes
Initial Data Preprocessing	-	Yes	-	Yes
Initial Exploratory Data Analysis	Yes	-	-	-
Text Pre-processing	Yes	-	Yes	Yes
Exploratory Data Analysis	-	-	Yes	Yes
Feature Extraction	Yes	Yes	Yes	-
ML Algorithm Selection	Yes	-	Yes	-
Model Training	Yes	Yes	-	Yes
Model Evaluation	-	Yes	Yes	-
Model Saving and Deployment	-	Yes	Yes	-
Project Documentation	Yes	Yes	Yes	Yes
Presentation Preparation	Yes	Yes	Yes	Yes

***Details of Individual Roles*:** Apart from the combined effort in programming or coding, documentation and preparation of presentations, here are the individual responsibilities:

Keerthi:

- **Domain and Data Understanding:** Understanding the data structure and the meaning of each genre in the data.
- **Initial Exploratory Data Analysis:** Data Analysis of the initial data extracted from the text file. This includes the number of data points in each genre, the distribution of genres across years etc.
- **Text Pre-processing:** This is a crucial step where Natural Language Processing Techniques are applied and I will be responsible for data cleaning followed by segmentation and removal of stop words.
- **Feature Engineering:** This includes application of novel techniques to convert the text to vectors.
- **ML Algorithm Selection:** Researching on ML Algorithms which suit the text classification-based tasks and short-listing algorithms which can yield better results
- **Model Training:** Splitting the data into train and test data. Training the data on selected machine learning models

Pavan:

- **Dataset Selection:** I played a major role in selecting the appropriate dataset for the project
- **Domain and Data Understanding:** Understanding the data structure and the meaning of each genre in the data.
- **Initial Data Preprocessing:** Initial data preprocessing includes to delimit the data and save the data in a data frame for further processing.
- **Text Pre-processing:** In this crucial step of applying NLP techniques, I am responsible for implementing novel NLP techniques like POS tagging and NER techniques etc. and explore more possibilities in the field of NLP.
- **Exploratory Data Analysis:** Perform exploratory data analysis after the NLP techniques have been applied to understand the important terms in the data, frequency for terms in the data etc.
- **Model Training:** Responsible for determining the ratio of split to be done in order to achieve better accuracy and run train the data on Machine learning Algorithms. If time permits, also explore on different deep learning algorithms used for text classification and implement the same.

Dinesh:

- **Dataset selection:** I was responsible for short listing the datasets that were apt for the project considering the number of datapoints, etc.
- **Text Pre-processing:** Responsible for implementing Tokenization and Lemmatization / Stemming. Analysing its impact on the final results
- **Exploratory Data Analysis:** Perform exploratory data analysis after the NLP techniques have been applied to understand the important terms in the data, frequency for terms in the data etc.
- **Feature Engineering:** Responsible for converting the text to vectors using NLP techniques
- **ML Algorithm Selection:** Responsible for shortlisting machine learning algorithms which work well for multi class classification.
- **Model Evaluation:** Considering the KPI and other metrics to evaluate the performance of the model on the test data and compare the model with other models.
- **Model Saving and Deployment:** Saving the final model based on the evaluation metrics and using it on unseen data to determine the genre of new movies.

Raghunandhan:

- **Initial Data Preprocessing:** Initial data preprocessing includes to delimit the data and save the data in a data frame for further processing.
- **Feature Engineering:** Responsible for converting the text to vectors using NLP techniques
- **Model Training:** Responsible for implementing shortlisted machine learning models and training the models with the train data and responsible for hyperparameter tuning where ever required.
- **Model Evaluation:** Considering the KPI and other metrics to evaluate the performance of the model on the test data and compare the model with other models.
- **Model Saving and Deployment:** Saving the final model based on the evaluation metrics and using it on unseen data to determine the genre of new movies.

Collaboration Meetings

We are a team of 4 members from difference Majors having different class schedules. Planning the meeting timings, place of meet and means of communication becomes a key factor in maintaining team harmony. We have collaboratively decided to work for an hour post the class schedule (i.e., every Tuesday after 2 pm) in the Discovery Park library or the Willis Library to discuss the status of the project and to identify the need for help. This meeting will also be helpful to understand what and how other team members are working and consistently guide each other. As the roles and responsibilities are different for different members of the team, it is important for us to transfer knowledge from one person to another on almost daily basis.

For daily or alternate day meet ups, we will be either using Microsoft Teams or Zoom Meetings or other easier online ways. As most of the project milestones are assigned to two or more team members, it is important that we work collaboratively and update each other regularly in the project group. We are also planning to setup a zoom call on every Sunday to track the status of the project if necessary.

Project Abstract

With the vast number of movies in the collection till date and new movies getting released weekly, the number of movies is continuously increasing leaving the viewers confused on which movie to watch next. Genre of a movie is basically the category or theme of a movie which helps the viewers to understand what to expect from a movie. The interest of each viewer is different and genre helps them to identify the movies they like.

How do we know the genre of a newly released or to-be released movie ? Every movie releases a summary of the movie in the initial days of the pre-production stage to pitch the movie idea to production houses or cast. This short summary can be used for prediction or classification of the genre of the movie. The aim of this project is to simplify the classification process using Natural Language Processing techniques and Machine Learning models by using a dataset from IMDb. By this, we are building a machine learning model which can classify which genre the movie falls into.

The main goal is to help viewers discover movies such that the probability of them liking the movie is high. By giving the short summary of a movie, the viewers can get the genre of the movie like Drama, Crime, Thriller, Comedy etc. and from this the viewers can decide whether to watch that movie or not. This model aims to give accurate and reliable results by using the Machine Learning Algorithm which fetches more accuracy and appropriate Natural Langue Processing Techniques.

Data Abstract

Dataset Link: https://www.kaggle.com/datasets/hijest/genre-classification-dataset-imdb

Dataset file Type: .txt files separated by :::

The dataset used for Movie Genre Classification is a huge collection of movie descriptions from IMDb. We are considering a dataset with movie names along with the release year and a short description of the movie. These are used to predict or classify the genre of a movie. Dataset contains train and test data in separate .txt files. Each column is separated by :::.

Structure of Train dataset: ID ::: TITLE ::: GENRE ::: DESCRIPTION

Structure of Test dataset: ID ::: TITLE ::: DESCRIPTION

ID: Represents the sequence number of the movie in the list.

Datatype: Integer

Example: 1, 2, 3 .. etc.

TITLE: Represents the movie title along with the year of release in brackets ().

Datatype: String

Example: Oscar et la dame rose (2009), Cupid (1997), Young, Wild and Wonderful (1980)

GENRE: Represents the genre of the movie which is basically the target variable.

Datatype: String

Example: drama, thriller, adult, documentary, crime etc.

DESCRIPTION: Contains a short description of the movie.

Datatype: String

Example: Listening in to a conversation between his doctor and parents, 10-year-old Oscar learns what nobody has the courage to tell him. He only has a few weeks to live. Furious, he refuses to speak to anyone except straight-talking Rose, the lady in pink he meets on the hospital stairs. As Christmas approaches, Rose uses her fantastical experiences as a professional wrestler, her imagination, wit and charm to allow Oscar to live life and love to the full, in the company of his friends Pop Corn, Einstein, Bacon and childhood sweetheart Peggy Blue.

The actual genres of the movies in test data are available in a separate text file.

Number of Datapoints in Train Dataset: 54214

Number of Datapoints in Test Dataset: 54200

For our classification model, we will be combining the train and test dataset and the total number of datapoints will be 108414. We will be randomly splitting the data into train and test data by 70:30 or 60:40.

Total number of datapoints: 108414

Project Design

Technology Stack:

- **Programming Language:**
  - *Python Programming:* Python programming language is the most used programming language for Machine Learning models. It has libraries used for multiple Natural Language processing techniques as well as Machine Learning models
- **Development Environment:**
  - *Google Colab:* Google Colab is an online hosting service used to run Python codes. It takes data from cloud and helps us to run the code from any device.
- **Libraries:**
  - *NLTK (Natural Language Toolkit):* This library is known for Natural Langue Processing based functions and has multiple in-built functions to perform various NLP based tasks
  - *Scikit-learn:* This is used for implementing machine learning algorithms and model evaluation
  - *Pandas:* This is used for data manipulation using Data frames
  - *Numpy:* This is used for array handing and numerical operations
  - *Matplotlib:* This is used for data visualization and plotting graphs
- **Data Requirements:** A dataset which contains short descriptions of each movie and the target variable i.e., the genre of the movie. Here it is an IMDb dataset.

Programming Logic

1. Import all necessary libraries
2. Import the whole dataset
3. Perform basic delimitation and store the data in a Data Frame
4. Perform initial Exploratory Data Analysis
5. Perform pre-processing of the data
   a. Segmentation
   b. Stop-word removal
   c. Tokenization
   d. Lemmatization/Stemming
   e. Part-of-Speech tagging
6. Exploratory Data Analysis of pre-processed data
7. Text Vectorization using TF-IDF
8. Split the Data into Train and Test
9. Train different Machine Learning models
10. Make predictions using test data and evaluate the model performances
11. Save the best model and use it on unseen data

Project Workflow:

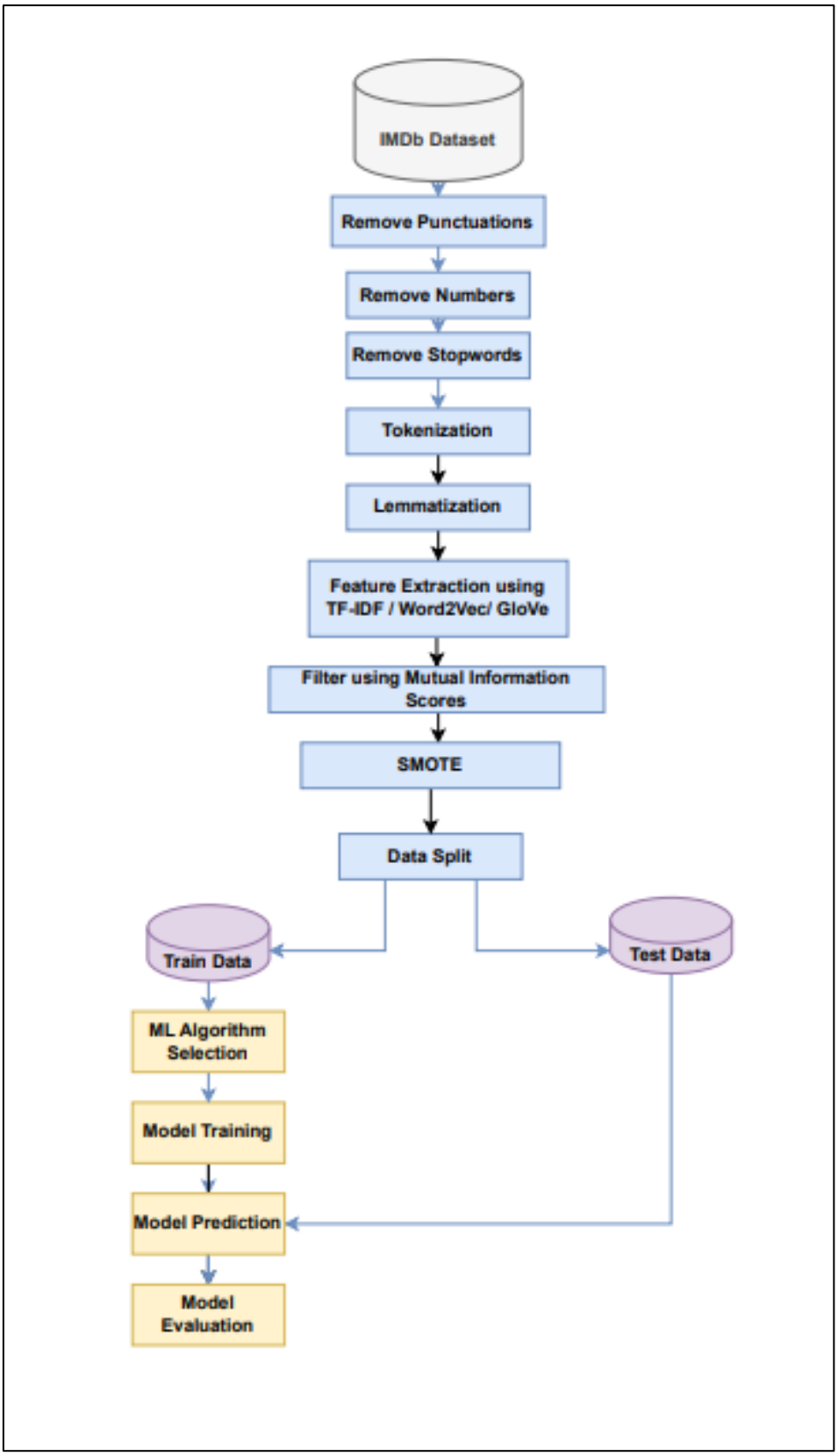

Figure 1: Flow Chart of Genre Classification

State of Art

The genre classification is one of the important pieces of information that is needed in order to recommend a movie to a viewer. This lays as an initial step to getting information about the movie which helps in further recommendation of the system. Previously, the genre of the movie was decided by the creators of the crew of the movie or was watched by multiple sources and the genre of the movie was decided and published accordingly. This happens at the later stages of the movie and post the release of the movie; many might not know the genre of the movie till they watch it. As the movie title might suggest something else than the plot of the movie.

In order to avoid these, recommendation systems were introduced where the movie is categorized into a particular genre and the viewers can decide whether they want to watch the movie or not. This can be done by considering the movie summary. It initially started with considering a few words from the summary and depending on the keywords, the genre of the movie was decided. From the key-word based systems, further advancement in the field of Artificial Intelligence lead to introduction of NLP techniques for feature extraction from the short descriptions of the movie. This helps the viewers to gain quite accurate results and laid a foundation to genre classification using Natural Language Processing Techniques. With the advancement in the NLP techniques like POS tagging [3], n-grams [4] and Named entity recognition [5], etc., we aim to build a machine learning model using the novel Natural Language Processing techniques.

Project Milestone

1. Initial Data Preprocessing
2. Initial Exploratory Data Analysis
3. Text Pre-processing
    a. Segmentation
    b. Stop-word removal
    c. Tokenization
    d. Lemmatization/Stemming
    e. Part-of-Speech tagging
    f. Other new NLP Techniques
4. Exploratory Data Analysis on Pre-processed data
5. Feature Extraction
    a. TF-IDF
    b. n-grams
    c. GloVe
6. ML Algorithm Selection
    a. Naïve Bayes Model
    b. Vector Machine based models
    c. Tree Based Models
    d. Deep Learning Models (if time permits)
7. Model Training
8. Model Evaluation
9. Model Optimization
10. Model Saving

Expected Results

The expected result from the project is to create a machine learning model which can predict the genre of the movie from the short description of the movie provided. The Key Performance Indicator of the project will the average F1 score of the model considering all the genres available in the dataset. The final expectation is to take a description as input and give the genre from the description provided using a model. The final saved model is expected to predict the genre of a new movie from its short description.

Project Functions

The movie genre classification model design mainly depends on the Natural Language Processing techniques used for the preprocessing of the data and the model used for classification of the movie genre. We have used multiple techniques to identify and increase the capability of the model to accurately predict the genre of any new movie. We started with understanding the data and performing Exploratory data analysis on the raw data to understand the distribution across, genres, years and other attributes. The distribution was visualized using seaborn and matplotlib libraries.

Proceeding to the preprocessing stage using the NLP techniques, below are the key functions used:

The main code used for text pre-processing is as in Figure 2:

```
[16] import nltk
     from nltk.corpus import stopwords
     from nltk.stem import PorterStemmer, WordNetLemmatizer

     nltk.download('punkt')
     nltk.download('stopwords')
     nltk.download('wordnet')
     nltk.download('punkt_tab')

     stemmer = PorterStemmer()
     lemmatizer = WordNetLemmatizer()

     def remove_punctuation(text):
         return text.translate(str.maketrans('', '', string.punctuation))

     def remove_numbers(text):
         return ''.join([char for char in text if not char.isdigit()])

     def remove_stopwords(text):
         stop_words = set(stopwords.words('english'))
         return [word for word in text if word.lower() not in stop_words]

     def stem_text(text):
         return [stemmer.stem(word) for word in text]

     def lemmatize_text(text):
         return [lemmatizer.lemmatize(word) for word in text]
```

Show hidden output

```
[17] movie_genre_df['DESCRIPTION_CLEAN'] = movie_genre_df['DESCRIPTION'].apply(remove_punctuation)
     movie_genre_df['DESCRIPTION_CLEAN'] = movie_genre_df['DESCRIPTION_CLEAN'].apply(remove_numbers)
     movie_genre_df['TOKENS'] = movie_genre_df['DESCRIPTION_CLEAN'].apply(nltk.word_tokenize)
     movie_genre_df['DESCRIPTION_NO_STOPWORDS'] = movie_genre_df['TOKENS'].apply(remove_stopwords)
     movie_genre_df['STEMMED_WORDS'] = movie_genre_df['DESCRIPTION_NO_STOPWORDS'].apply(stem_text)
     movie_genre_df['LEMMATIZED_WORDS'] = movie_genre_df['DESCRIPTION_NO_STOPWORDS'].apply(lemmatize_text)
     movie_genre_df['LEMMATIZED_DESCRIPTION'] = movie_genre_df['LEMMATIZED_WORDS'].apply(lambda x: ' '.join(x))
```

Figure 2: Code for Text Pre-processing

Removal of Punctuations: As the punctuations like commas, exclamatory marks, quotations etc do not consist of semantic meanings, removing them would make the classification task easier. For this, we are using the punctuations available in the string library and replacing them with blank.

Removal of Numerical values: Similar to punctuations, numerical values also do not have any semantic meaning and will not be useful for predicting the genre, so to remove this we are using isdigit() method to check if each character is a numerical value and replace that with blank.

Tokenization: Splitting the description into multiple tokens i.e. words is important for further analysis of the data, enables feature extraction and maintaining structed data. For this, we are using the word_tokenize method in the NLTK (Natural Language Toolkit) library.

Removal of stop words: Stop words, do not add much meaning to the sentence and would not be a good feature to retain for text classification. This will enable us to decrease the size of features. To perform this, we have again used the stopwords method in the NLTK library and selecting the language as English.

Lemmatization: Lemmatization is used to convert each word to its base or dictionary form. This is to understand the variations in the words and consider it as same if they have the same base word. For applying this, we are using the WordNetLemmatizer class available in the NLTK library. We have also performed stemming on the description but we see that the meaning of the word is lost in multiple cases. For example, in the figure below, we can see that the word bus in the description was converted to 'bu' when stemming was applied and has no meaning. To avoid this we are using the lemmatization.

	DESCRIPTION_NO_STOPWORDS	STEMMED_WORDS	LEMMATIZED_WORDS
0	[Listening, conversation, doctor, parents, yea...	[listen, convers, doctor, parent, yearold, osc...	[Listening, conversation, doctor, parent, year...
1	[brother, sister, past, incestuous, relationsh...	[brother, sister, past, incestu, relationship,...	[brother, sister, past, incestuous, relationsh...
2	[bus, empties, students, field, trip, Museum, ...	[bu, empti, student, field, trip, museum, natu...	[bus, empty, student, field, trip, Museum, Nat...
3	[help, unemployed, father, make, ends, meet, E...	[help, unemploy, father, make, end, meet, edit...	[help, unemployed, father, make, end, meet, Ed...
4	[films, title, refers, unrecovered, bodies, gr...	[film, titl, refer, unrecov, bodi, ground, zer...	[film, title, refers, unrecovered, body, groun...
5	[Quality, Control, consists, series, mm, singl...	[qualiti, control, consist, seri, mm, singl, t...	[Quality, Control, consists, series, mm, singl...
6	[tough, economic, times, Max, Joey, run, ideas...	[tough, econom, time, max, joey, run, idea, di...	[tough, economic, time, Max, Joey, run, idea, ...
7	[Ron, Petrie, Keanu, Reeves, troubled, teen, w...	[ron, petri, keanu, reev, troubl, teen, whose,...	[Ron, Petrie, Keanu, Reeves, troubled, teen, w...
8	[sudden, calamitous, event, causing, great, lo...	[sudden, calamit, event, caus, great, loss, li...	[sudden, calamitous, event, causing, great, lo...
9	[Four, high, school, students, embark, terrify...	[four, high, school, student, embark, terrifi,...	[Four, high, school, student, embark, terrifyi...

Figure 3: Stemming vs Lemmatization

After, text pre-processing, we have analysed the data and visualized the most important terms and the distribution of the tokens across genres. The below is a representation of most frequent words in the movie descriptions. The higher the frequency of the word, the bigger is the font as shown in Figure 4. This is done using wordclound function in the wordcloud library. The bar chart in Figure 5 represents the average number of tokens in each genre. This is basically useful to check the number of tokens in each genre.

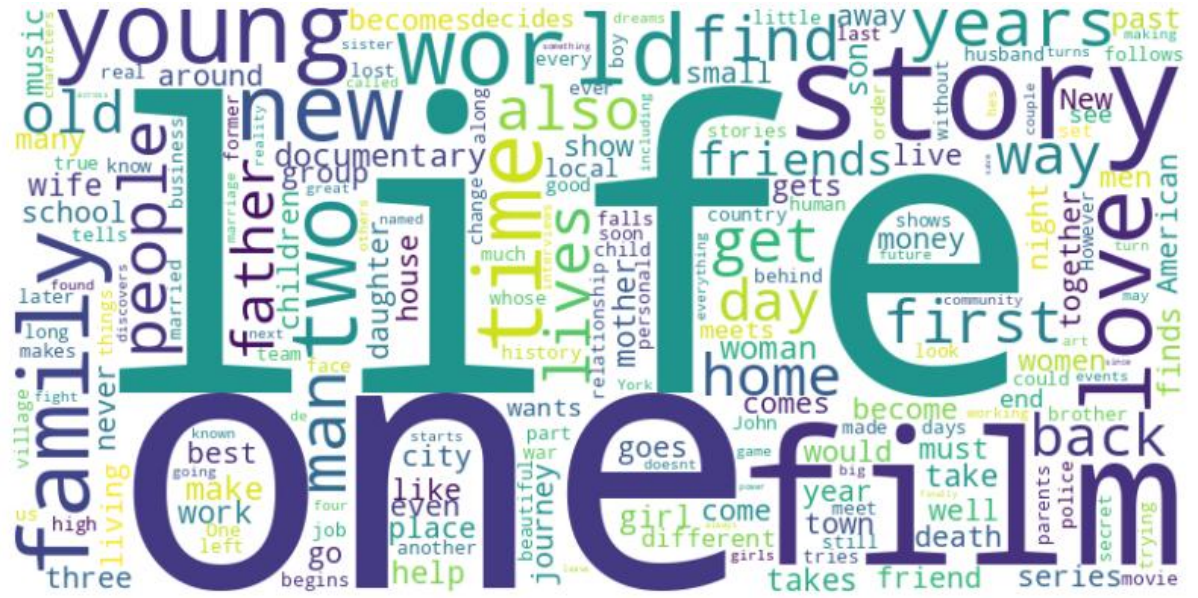

Figure 4: Most Frequent Words in Movie Descriptions

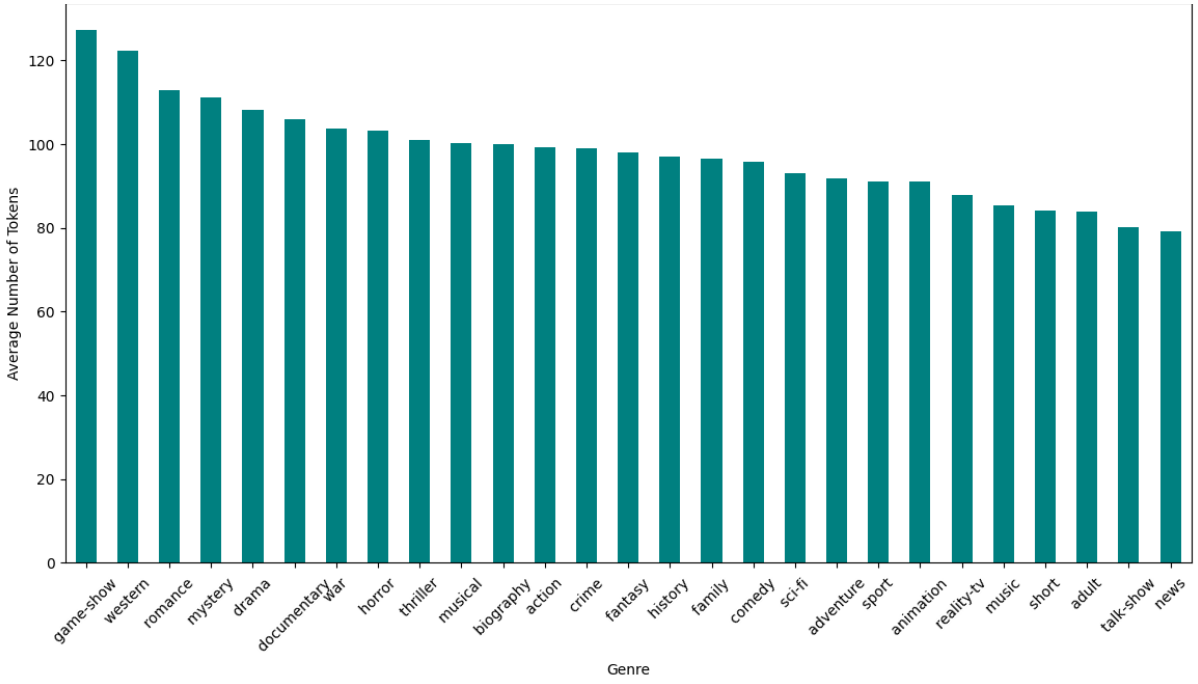

Figure 5: Average number of tokens per genre

Next step was to extract features from the tokenized and lemmatized data. For this process we have explored TD-IDF method, Word2vec method and Glove embeddings method. A simple TD-IDF based vectorization was not yielding good results as it does not capture semantic relationships and the meaning of the words. Word2Vec and GloVe are pretrained models and thus can capture the semantic meanings in the data. The word2vec pretrained model was working well on our data and thus we have selected Word2Vec for extracting features from the data.

Word2Vec: This pretrained model captures the semantic relationship between words and can be useful for text classification when the meaning of the words are important for identifying the genre. For this we are downloading the word2vec-google-news-300 pretrained model which produces 300-dimensional vectors for each datapoint. Each word is converted into its vector representation and the average of the vector is considered to be the feature of the vector. The vector is zero if the word is not available in the library. In this way, each datapoint has a 300 dimensional vector. Figure 6 represents the code used for converting words to vectors using Word2Vec library.

```
word2vec_model = api.load("word2vec-google-news-300")

def get_word2vec_vector(text, model, vector_size=300):
    """
    Given a text, computes the average of word vectors from Word2Vec embeddings.
    """
    words = text.split()
    word_vectors = []

    for word in words:
        if word in model:
            word_vectors.append(model[word])

    if word_vectors:
        return np.mean(word_vectors, axis=0)
    else:
        return np.zeros(vector_size)


movie_genre_df['WORD2VEC_VECTOR'] = movie_genre_df['LEMMATIZED_DESCRIPTION'].apply(lambda x: get_word2vec_vector(x, word2vec_model))
X_word2vec = np.vstack(movie_genre_df['WORD2VEC_VECTOR'])

y = movie_genre_df['GENRE_ENCODED']
```

Figure 6: Code for converting words to vectors using Word2Vec

Mutual Information: The 300-dimensional vector obtained may contain information that may add noise to the model. Thus, we are calculating the mutual information score and removing the features which have less information about the target. For this we are setting a threshold of 0.2 and only considering the features above it. This helps in avoiding overfitting of the model, reducing the dimensionality of the features and retaining the features with high importance across all datapoints. Mutual Information is calculated using the mutual_info_classif function available in the sklearn.feature_selection library and the selected features are stored in a dataframe. Figure 7 contains the code used to calculate MI and filtering it according to the threshold.

```
y = movie_genre_df['GENRE_ENCODED']

mi_scores = mutual_info_classif(X_word2vec, y)

mi_threshold = 0.02
selected_features = mi_scores >= mi_threshold

X_word2vec_selected = X_word2vec[:, selected_features]
```

Figure 7: Code for calculation of Mutual Information

SMOTE: In the Figure 8, we can see that distribution of classes i.e. genres is unequal and due to this the model can tend to predict only the label that has more data or overfit to few genres. SMOTE enables us to oversample and produce synthetic data for the genres which have less data when compared to others and ensure that the number of datapoints for each genre is equal or almost equal. SMOTE is available in the imblearn.over_sampling library and can be implemented by selecting a random value and resampling the data. This gives resampled balanced data as an output.

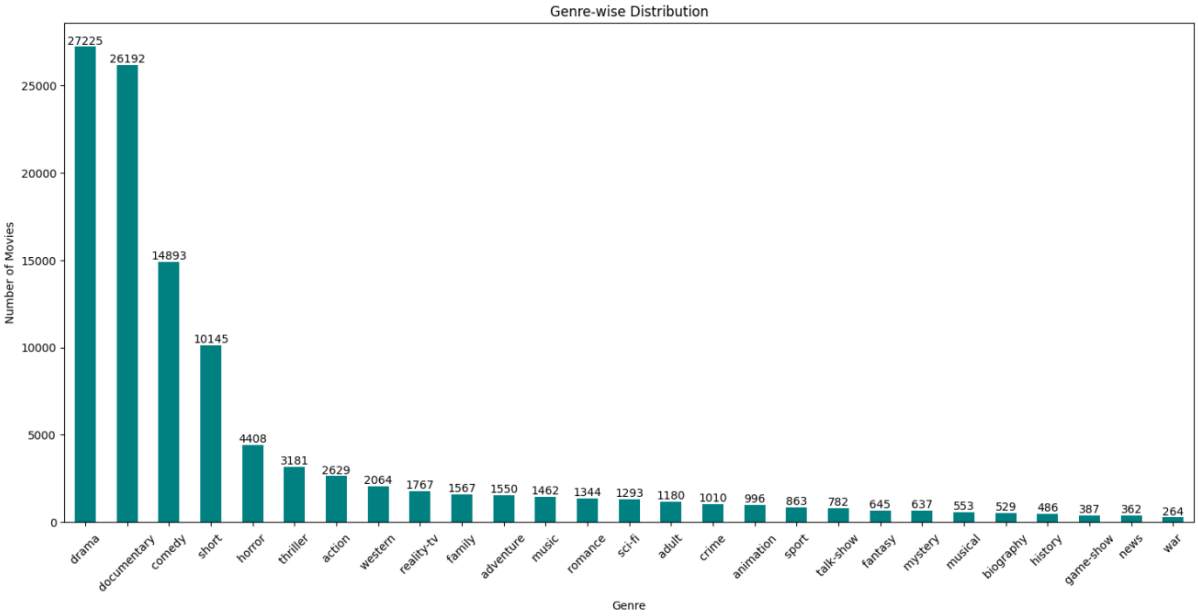

Figure 8: Genre-wise Distribution before SMOTE

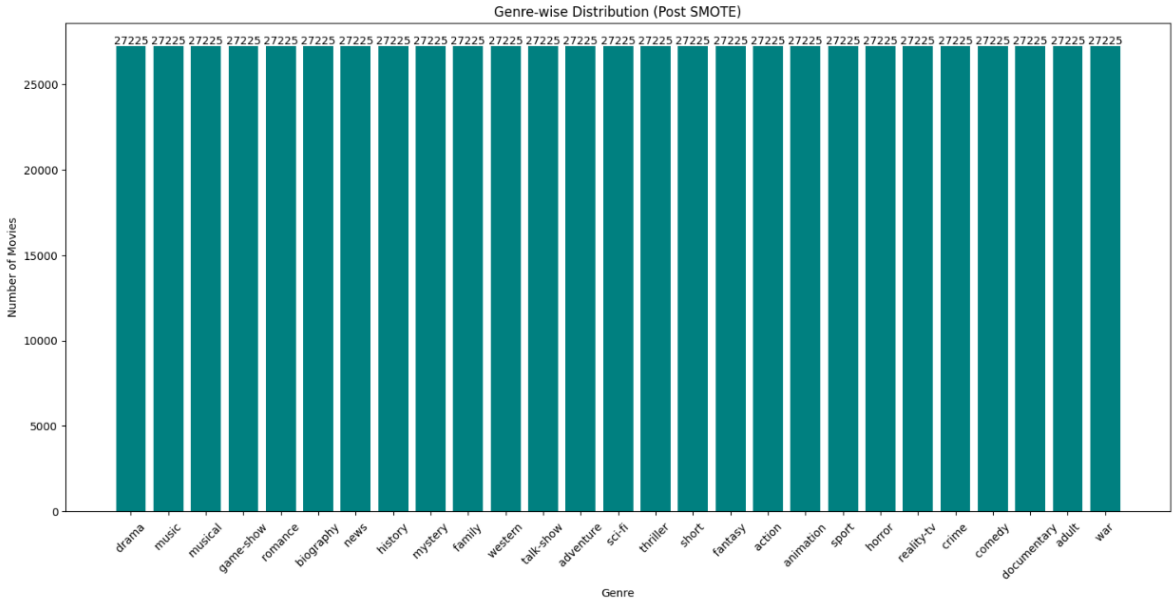

Figure 9: Genre-wise Distribution after SMOTE

Post balancing the classes in the data as we can see in Figure 9, we will split the data into train and test considering 80% of the data as train and 20% of the data as test data. We have trained the model using Logistic Regression, Multinomial Naïve Bayes and XG Boost. Out of all the models, the boosting technique was yielding the expected results and thus we have used this model to classify the genre.

XG Boost Classifier: Using the resampled data, we are we are training a model using a powerful gradient boosting algorithm known as XGBoost. We are using SoftMax function as the objective function and logloss as the evaluation metric to reduce the error. Softmax considers the genre with highest probability as the predicted genre. Code for the XGBoost Classifier is available in Figure 10.

```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=16)

model = XGBClassifier(objective='multi:softmax', eval_metric='mlogloss', use_label_encoder=False)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))
```

Figure 10: Code for training XGBoost Classifier

We are saving all the necessary pickle files for further use on unseen data. This is done using pickle library. We are saving the label encoder, selected features, model in separate files.

Project Result

Our goal was to train a model which can classify a movie description into a particular genre without overfitting the model. Given, any description, the model can predict the genre of the movie and this prediction does not tend towards majority class. The model works well for all the genres available in the dataset. This shows that the model is not overfitted and works well on unseen data. The results achieved are closely aligned to the expectations mentioned in the early stages of the project proposal.

The word2vec model along with mutual information-based filtering, SMOTE and XGBoost Classifier enabled us to achieve the results. Word2Vec helped us to convert the words to vectors along with meaningfully combining words enabling higher accuracy and f1 scores. Mutual Information scoring helped us filter the features that were important which reduced the dimensions of the features enabling faster and more reliable results. SMOTE handled the class imbalance problem and made sure that the model does not overfit to a particular genre. The performance metrics varied a little across genres and the results are available in the below table.

The average weighted f1 across all genres is 0.91 when XG Boost classifier along with Word2Vec was used. The weighted average f1 score and accuracy when TF-IDF representation was used was 0.79 and 0.78 respectively but the classifier was not able to classify documentary, comedy, drama, shorts genres efficiently. The weighted average f1 score and accuracy when GloVe representation was used was 0.87 and 0.86 respectively but the classifier was not able to classify drama, shorts and thriller genres efficiently.

The table below represents the different method tried and their results:

ML Algorithm	SMOTE	FE Method	Accuracy	Precision	Recall	F1-Score
Logistic Regression	No	TF-IDF	0.55	0.52	0.55	0.51
MultinomialNB	No	TF-IDF	0.49	0.49	0.49	0.41
SVM	No	TF-IDF	0.56	0.53	0.56	0.52
XGBoost	Yes	TF-IDF	0.79	0.78	0.79	0.78
XGBoost	Yes	GloVe	0.87	0.86	0.87	0.86
XGBoost	Yes	Word2Vec	0.91	0.91	0.91	0.91

Below table represents the genre-wise precision, recall and f1 score for Word2Vec Representation + XG Boost Classifier:

genre	precision	recall	f1-score	Count
action	0.89	0.90	0.89	5456
adult	0.97	0.99	0.98	5502
adventure	0.94	0.93	0.94	5469
animation	0.96	0.97	0.96	5516
biography	0.97	0.99	0.98	5520
comedy	0.70	0.65	0.67	5508
crime	0.94	0.97	0.96	5366
documentary	0.70	0.65	0.68	5391
drama	0.58	0.50	0.53	5535
family	0.92	0.94	0.93	5412
fantasy	0.96	0.99	0.98	5381
game-show	1.00	1.00	1.00	5362
history	0.97	0.99	0.98	5510
horror	0.90	0.90	0.90	5440
music	0.96	0.99	0.98	5447
musical	0.97	0.99	0.98	5390
mystery	0.97	0.99	0.98	5514
news	0.99	1.00	1.00	5396
reality-tv	0.93	0.96	0.94	5462
romance	0.91	0.97	0.94	5419
sci-fi	0.95	0.97	0.96	5533
short	0.70	0.60	0.65	5358
sport	0.98	1.00	0.99	5540
talk-show	0.97	0.99	0.98	5332
thriller	0.85	0.83	0.84	5480
war	0.99	1.00	1.00	5403
western	0.99	0.99	0.99	5373

We can see that the f1 score is stable across most of the genres and is above 0.9 which indicates that the classifier is able to classify all genres efficiently. Further, we can also observe the trade-off between recall and precision is stable and this indicates that the model is not overfitted. Further we have also used unseen descriptions from Google and checked if the model is able to classify. We have used the short description for Interstellar Movie – "*In a near future ravaged by drought and dirt storms, a group of space explorers travel through a wormhole to another galaxy to try and find a suitable new home for humankind*" and it was classified under Sci-Fi which is correct.

```
[ ] new_input = input("Enter a movie description: ")

    predicted_class = predict_genre(new_input)
    print(f"Predicted Genre: {predicted_class}")
```

```
Enter a movie description: In a near future ravaged by drought and dirt storms, a group of space explorers travel through a wormhole to another galaxy
Predicted Genre:  sci-fi
```

Figure 11: Model Performance on Unseen Data

Result Evaluation

The project achieved better results than expected mainly due to the SMOTE technique and XG Boosting algorithm.

***Accuracy of the Model*: 91%.**
This indicates that the model is able to classify the majority of the movies to its corresponding genres.

***Precision*: 91%**
***Recall*: 91%**
***F1-Score*: 91%**
All the 3 scores above indicate that the model is performing well across all genres.
High performing Genres: game-show, news, war have achieved almost perfect score i.e., it was able to classify most of the descriptions correctly
Low performing Genres: comedy, documentary, drama, short have comparatively low f1 scores but is acceptable to an extent considering the scores are not too bad.
Overall, the performance of the model is more than expected as per the model expectation results mentioned in the project proposal.

References

[1] https://www.kaggle.com/datasets/hijest/genre-classification-dataset-imdb
[2] https://www.turing.com/kb/natural-language-processing-function-in-ai
[3] https://medium.com/@sujathamudadla1213/what-is-parts-of-speech-pos-tagging-natural-language-processing-in-2b8f4b07b186
[4] https://medium.com/@abhishekjainindore24/n-grams-in-nlp-a7c05c1aff12
[5] https://www.analyticsvidhya.com/blog/2021/11/a-beginners-introduction-to-ner-named-entity-recognition/
[6] https://neptune.ai/blog/vectorization-techniques-in-nlp-guide
[7] https://www.analyticsvidhya.com/blog/2019/04/predicting-movie-genres-nlp-multi-label-classification/
[8] https://towardsai.net/p/nlp/natural-language-processing-concepts-and-workflow-48083d2e3ce7
[9] https://towardsai.net/p/nlp/natural-language-processing-concepts-and-workflow-48083d2e3ce7
[10] https://www.sciencedirect.com/science/article/abs/pii/S1568494617305112
[11] https://sunscrapers.com/blog/9-best-python-natural-language-processing-nlp/
[12] Code Repo: https://colab.research.google.com/drive/1tLaNTAiT9_8yDj0UEABjGLdlvbPnZ-sz?usp=sharing