GENRE CLASSIFICATION OF MOVIES USING THEIR SYNOPSIS

Inspiring Excellence

SUBMISSION DATE: 14.12.17

Submitted by:

Ramiz Ihteshamur Rahman (17241007)
Department of Computer Science and Engineering

Sk. Sajidul Kadir (17241006)
Department of Computer Science and Engineering

Superviser:

Amitabha Chakrabarty, Ph.D
Assistant Professor
Department of Computer Science and Engineering

DECLARATION

We, hereby declare that this thesis is based on results we have found ourselves. Materials of work from researchers conducted by others are mentioned in references.

Signature of Supervisor

Amitabha Chakrabarty, Ph.D
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Signature of Authors

Ramiz Ihteshamur Rahman(17241007)

Sk. Sajidul Kadir (17241006)

ABSTRACT

Movies have been classified into genres since the inception of the medium. However, even till this day, the process of classifying movies into genres has been a manual, time consuming task requiring human expertise. Some work has been done in trying to automatically classify movies into genres using machine learning techniques and classifiers, and some success has been achieved. However, little work has been done in this field with regards to Indian movies specifically. In this paper, multiple supervised learning algorithms including Naïve Bayes, Logistic Regression, K Nearest Neighbor, Decision Tree and Linear SVM were used to classify a set of Indian movies including but not limited to Bollywood and South Indian movies. Naïve Bayes and Logistic Regression were found to be the better performers and K-Nearest Neighbors was the worst performer. Genres with high positive examples such as ‘Drama’ were classified correctly more often and 0.7 for precision and 0.7 for recall scores was obtained. Performance degraded drastically as the number of positive examples fell with the ‘Musical’ genre having precision scores close to 0.1 and recall scores nearing 0.

ACKNOWLEDGEMENT

This thesis would not have been possible without the help and mentorship of our thesis supervisor Dr. Amitabha Chakrabarty. We would like to offer our sincere gratitude towards him for guiding us throughout this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Classifying bodies of text into pre-defined groups has been very useful for lots of different purposes e.g. spam detection, sentiment analysis etc. The goal of our investigation was to collect a dataset of Indian movies from IMDb [5] and use it to teach text classification algorithms how to automate the process of genre classification of movies.

We want to see if limiting our dataset to just Indian movies improves the performance of traditional supervised classification algorithms. As we all know, visual media has become one of the most prominent forms of art after the introduction of television, followed by online video services like Youtube and Netflix. Most countries have their own unique cultures and customs that are heavily reflected in their works of art. The focus of our investigation will be to see if this uniqueness can be used to detect movie genres from this region with acceptable accuracy.

In 2014, Tanenbaum. tried to classify movies into genres by using a decision tree classifier and plot keywords as the input feature. He managed to achieve about 31% accuracy using a training set of 9,643 movies and concluded that there is a correlation between certain narrative features and film genre[1]. Earlier in 2011, KaWing Ho trained 4

different classifiers — neural networks, SVM, K-nearest neighbors and the Parametric Mixture Model — on 80% of a dataset acquired from IMDB consisting of 16,000 unique movies and TV shows. The remaining 20% of the dataset was used to test the trained models; a precision of 0.695, and a recall of 0.736 as obtained for ‘Drama’ using SVM[2].

All of the researchers used large datasets consisting of movies from all over the world and did not make use of two simple text classification algorithms — Naïve Bayes and Logistic Regression. We are diverging from this trend by restricting our dataset to one specific geographical location, India and using default implementations of the aforementioned simple algorithms.

1.1 Goals

For our project, we have decided to keep our dataset limited to include only Indian movies. The key motivation behind this project was to answer the following questions:

- Does the IMDb database contain enough data about Indian movies to train machine learning algorithms to detect their genres?
- Which machine learning technique or algorithm can yield the best results?

1.2 Motivation

Most online repositories of movies require movies to be manually tagged by its users or editors. This task is quite difficult because of the sheer number of movies released all over the world. The problem is compounded even further because of the variety of languages of the movies.

The objective of our research was to create a system that can automatically tag these movies with the proper genres so that it does not require any manual inputs.

1.3 Methodology

The previous works done in this field have never focused on movies originating from India, and therefore that has been the primary focus of the thesis. To conduct the research, similar works done by Tanenbaum[1] and Ka-Wing Ho[2] were reviewed along with other papers that used the same algorithms for different classification tasks[6][7][8][9][10][11] and drawing inspiration from these previous works, a dataset comprised exclusively of movies originating from India was gathered, cleaned, and processed. Then, the classic supervised machine learning classification algorithms were used to classify an unknown movie from India into well-defined genres using only the

movie's textual description. The 'Scikit-Learn' library of the Python programming language contains implementations for all of the classification and algorithms and therefore, the aforementioned library and programming language was used to carry out the research. The results were then visualized with the help of another Python library, 'Matplotlib' and visualized results were then explained in the paper.

1.4 Thesis Outline

Chapter 1 is the introduction of the thesis where the motivation and objectives are described.

Chapter 2 is the background study which covers the literature review. Existing work that has implemented similar ideas are discussed in this section.

Chapter 3 is dedicated to the algorithms that have been used for the analysis. Equations, pseudo code, and explanation of the algorithms are presented along with key terms that are useful for understanding the results of the experiments.

Chapter 4 is the implementation section where the procedure of work is discussed at length. The analysis and exploration of the dataset is presented along with a flowchart describing the steps taken to acquire the results.

Chapter 5 is dedicated to the presentation of the results using graphs and tables, attained after training and testing the classification algorithms.

Chapter 6 provides a conclusion to the thesis, and discusses future work that can be carried out.

Chapter 2

LITERATURE REVIEW

Utilizing different machine learning classification algorithms to classify text has previously been looked at by quite a few researchers. However, classifying movies using their synopsis data has had very little attention from the Machine Learning community so far. For this thesis we have looked at a number of papers that have been published in this topic.

Ka-Wing Ho collected information from IMDb about 16,000 unique titles and ran a number of binary classification machine learning algorithms on it - for detecting their genres based solely on their synopsis - with various degrees of success [2]. The data was split into 80%-20% sets for training and testing respectively. The data for his paper was processed using the NLTK library. NLTK, short for "Natural Language Toolkit" is a set of libraries for natural language processing[4]. The data processing step began by removing the stop words using the NLTK stop words list and all the numerical words were mapped to the same index because they hardly ever influence the results in these kinds of problems. The data was then filtered even further by putting it through the Python stemmer library. This resulted in a library of 63,840 words. Out of them,

only the words which appeared more than 15 times throughout the dataset were kept, the rest were ignored. As a result, only 10,656 words remained. These words were finally used to generate representation matrix, each synopsis was described by a vector generated by the tf-idf vectorizer[2]. Tf-idf is short for “term frequency–inverse document frequency”, is a numerical statistic that measures how important a word is to a document by using the frequency of its occurrences.[13]

A number of algorithms were used to classify synopsis text data. Four different approaches were evaluated, One-Vs-All approach with SVM, Multi-label K-nearest neighbor (KNN), Parametric mixture model (PMM) and Neural Networks. The most accurate among these algorithms were SVM and Neural Network. They performed almost equally in terms of average precision, recall and f-measure scores. What is notable is that the scores varied wildly for different genres, even for the same algorithm. In this paper[2], SVMs had 0.66 precision, 0.39 recall and 0.47 F-measure. kNN provided 0.41 precision and 0.73 recall with a 0.52 F-measure score.

But these movies were not filtered based on geographical location or language. For our project we wish to see if limiting our dataset to movies from a certain geographical area makes any noticeable

improvements to the results that he received because specific geographical areas often have unique cultures and customs that are often reflected by the movies from those locations.

Joshua Tanenbaum exclusively used Decision Trees to detect genres of movies [1]. Data on 516,980 films in total belonging to 28 genres were collected from IMDb for this experiment.

The author found quite a few issues with the initial dataset acquired from IMDb. The most prominent of these were the fact that most of the movies were tagged with multiple genres but there was no mention of the hierarchy of the genres. Usually movies have a primary genre with some elements from multiple other genres. However, the IMDb database treats every genre assigned to a movie equally. The other problem was that the database also contained data on video games, adult and pornographic content. To get rid of these unnecessary garbage data the author decided to prune off 9 of the 28 genres, leaving him with 19 genres.

This dataset was then run through the decision tree algorithm. However, the initial run resulted in an accuracy of less than 1%. To improve the results each movie was assigned just one genre, selected randomly from the list of genres assigned to each movie. This took boosted the accuracy rate to 12%. Lastly, to improve the performance of

the decision even further, the author decided to handpick the features used to generate the input matrix. Out of the 2,227 features that were available initially, the author picked 168, guided by his understanding of movies and their primary genres. This finally resulted in an accuracy rate of 32%.

Even though the results of this experiment were not particularly successful, the key take away from it is that, careful pruning of the feature set by getting rid of redundant features can improve the results significantly.

Thorsten Joachims, on his paper, mainly focused on text categorization with 'Support Vector Machines (SVM)' [6]. He claims was that SVM can provide substantial improvements over the then prevalent machine learning based text classification methods. It was mentioned in the paper that SVMs can perform well in text classification tasks because:

1. They can work well with high dimensional input space. Text classification problems usually have tens of thousands of features. So, SVMs are suitable for it.
2. Text classification algorithms have to work with sparse document vectors. SVMs are quite well suited for this kind of tasks.

3. Most text categorization problems are linearly separable, another strong suit of SVMs.

The experiment compares the performance of the Polynomial and RBF kernels of SVMs. Two datasets have been used for this experiment. The Reuters-21578 dataset compiled by David Lewis and the Ohsumed corpus compiled by William Hersh.

The “ModApte” collection consists of 9603 training documents and 3299 test documents. From the second dataset, the first 10000 documents are used for training and the second 10000 documents are used for testing. The experiments conducted for this paper show SVMs performing significantly better than Naive Bayes, Rocchio, C4.5 and k-NN algorithms for some categories.

However, when comparing training time, SVMs were much slower than Naive Bayes, Rocchio and k-NN. [6]

Chapter 3

MODELS AND ALGORITHMS

This chapter contains the theoretical introductions to the Machine Learning algorithms used for our thesis.

3.1 Machine Learning

Machine Learning is the process through which machines can learn from data and act according to this training data.

One way of categorizing Machine Learning systems is by how much supervision they get during training. There are four noteworthy categories:

1. Supervised Learning
2. Unsupervised Learning
3. Semi supervised Learning
4. Reinforcement Learning

We are going to be dealing with Supervised Learning algorithms for our problem, as our training dataset is already labeled.

3.1.1 Supervised Learning

Supervised Learning algorithms learn from training datasets that are already labeled. Meaning, the training data fed to the program already

contains the correct results, called labels, the algorithm's job is to learn from the examples in the training data and make proper predictions for future unseen data.

The supervised learning task we will do is *classification.*

3.1.2 Classification Algorithms

The aim of Classification algorithms is to try to draw conclusions on an unseen input based on previously seen inputs (training data). Given inputs it will try to estimate the value of outputs. These outputs are labels that can be to the dataset.

Classification algorithms used for this thesis are:

- Naive Bayes
- Logistic Regression
- Support Vector Machines
- Decision Trees
- K Nearest Neighbors

3.2 Classifiers

The classifiers that were used in the thesis are described below.

3.2.1 Naive Bayes

Naive Bayes algorithms are some of the easiest and fastest classification algorithms around. This algorithm classifies new inputs

based on both the prior and the likelihood, to form a posterior probability using the Bayes' Rule.

Naive Bayes is the simplest among the Simple Naive Classifiers. All the attributes of the examples are considered as independent under this model. [7]

The pseudo code for the Naïve Bayes algorithm from, "Crime analysis and prediction using data mining" by S. Shiju and G. Surya, [7] is given in the next page.

Algorithm for Naïve Bayes

1. Let D = [$document_1$, $document_2$, …, $document_n$] be the training dataset
2. Let C = { set of classes }
3. Let P_r = [d_1: P_{r1}...d_i: P_{ri}] be a dictionary prior probabilities of the documents
4. Let F = [c_1: F_1...c_i: F_i] be a dictionary of word frequencies
5. Let P_c = [k_1: P_{c1}...k_i: P_{ci}] be a dictionary of conditional probabilities of the keywords
6. **for** c_i in C **do**
7. compute prior probability, P[d_i] = len(c_i) / len(n)
8. compute word frequency, F[d_i] = wordFrequency(c_i)
9. **end for**
10. **for** c_i in C **do**
11. **for** k_i in c_i **do**
12. P[k_i] = F[c_i] / len(c_i)
13. **end for**
14. **end for**
15. **for** c_i in C **do**
16. **for** k_i in c_i **do**
17. compute conditional probability, P[c_i] = F[c_i] /, P[k_i]
18. **end for**
19. **end for**
20. Output highest P_c

3.2.2 Logistic Regression

Logistic Regression determines the likelihood of a particular instance belonging to a particular class (e.g. what is the likelihood that a movie is a comedy movie?). If the returned likelihood is above 50%, then

the classifier decides that the instance belongs to that specific class, or else it predicts that it does not. So, it is a binary classifier [3].

A logistic Regression model calculates the weighted sum of the input features, it outputs the *logistic* of the result described by the following equation [3]:

$$\hat{p} = h_\theta(x) = \sigma(\theta^T x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(i)[3]}$$

The logistic is a sigmoid function that outputs a number between 0 and 1. It is defined as [3]:

$$\sigma(t) = \frac{1}{1+\exp(-t)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(ii)[3]}$$

If $\hat{p}$ is positive then it can make a prediction about $\hat{y}$ [3].

$$\hat{y} = \begin{cases} 0, \hat{p} < 0.5 \\ 1, \hat{p} \geq 0.5 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iii)[3]}$$

Training and Cost Function: The goal of training a Logistic Regression classifier is to properly decide the value of a parameter vector θ so that the model estimates high probabilities for positive instances and low probabilities for negative instances. This is expressed by the cost function[3]:

$$c(\theta) = \begin{cases} -\log(\hat{p}), y = 1 \\ -\log(1-\hat{p}), y = 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iv)[3]}$$

The cost function becomes very large when t approaches 0, so cost is very large when the probability of a positive outcome reaches close to 0. It will also be very large when the probability of a negative outcome reaches close to 1.

The pseudo code for the Logistic Regression algorithm from "Linear and Logistic Regression" by M. Arias [8] is given below:

Algorithm for Logistic Regression

1. Initialize $x = \{1, .., 1\}^T$
2. Scale the features of the attributes
3. **for** each j = 0, .., n **do:**
4. $\quad x_j' = x_j + \alpha\sum_i(y^i - h_a(x^i))x_j^i$
5. **end for**
6. **for each** j = 0, .., n **do:**
7. $\quad x_j = x$
8. **end for**
9. Output x

3.2.3 Support Vector Machines

Support Vector Machines (SVM) are supervised machine learning algorithm that are primarily used for classification of data. The examples fed into support vector machines are marked as a member of one or the other of two categories. And the classifier can mark new inputs as belonging to one category or the other.

SVMs maximize the distance between the decision boundary and the closest instances of both categories.

SVMs can be mainly grouped into two groups.

i. Linear SVMs

ii. Non-Linear SVMs

Linear SVMs: Linear SVMs have linear decision boundaries and are usually faster than Non-Linear SVMs.

Non-Linear SVMs: Despite Linear SVMs being fast and often being surprisingly good, a lot of datasets are not linearly distinguishable. So, in those cases SVMs with non-linear decision boundaries are utilized. They are called Non-Linear SVMs**.**

For our purposes we found that using a Linear SVM model gives us the best results. In our tests non-linear SVMs were very susceptible to imbalanced classes. Producing results with high precision (0.6) and recall (1.0) and F1-score (0.75) for the genre with the highest support. But produced very poor results (0 for precision, recall and F1-score) for all the other genres.

Hard Margin Classification: The hardness or softness of margins in an SVM means how accepting of errors the model is. Hard Margin Classification does not allow any errors. It is quite sensitive to outlier values. One outlier can entirely change the nature of the decision

boundary and in some cases the model can fail to produce any acceptable results at all [3]. Most real-world problems require some tolerance for errors.

Soft Margin Classification: Soft Margin Classification, as its name suggests, are more tolerant of errors. It allows for some errors to occur to make the model more generalized which makes more accurate.

The pseudo code for the Support Vector Machine algorithm from "Support vector machines for multiple-instance learning" by S. Andrews[9] is shown below.

Algorithm for Support Vector Machines

```
1. Compute y_i = Y_I for i ∈ I // I denotes the pattern
   selected as the positive "witness"
2. do
3.     compute SVM solution w, b for data set with
   labels
4.     f_i = {w, x_i} + b for all x_i in positive bags //
   outputs
5.     y_i = sgn(f_i) for every i ∈ I, Y_I = 1
6.     for every positive bag B_I do:
7.         if ∑_i∈I (1 + y_i) / 2 == 0 then
8.             i* = argmax_i∈I f_i
9.             y_i*= 1
10.        end if
11.     end for
12.     while (imputed labels have changed)
13.        output(w, b)
```

3.2.4 Decision Trees

Decision Trees can perform both classification and regression tasks. In fact, in many ways, they are more powerful than some other algorithms. They require very little data preparation and no feature scaling or centering at all. Each node in the decision tree has a "gini attribute" associated with it. The gini attribute (G_i) measures the node's impurity [3]. A node is pure if all the instances it applies to belong to the same class.

G_i is calculated using,

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2 \quad \text{.....(v)[3]}$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

A decision tree can also guess the likelihood of an example belonging to a particular class k.

Complexity: Traversing the tree has O(log(m)) complexity. But, training takes O(n*mlog(m)) time. [3]

The pseudocode for the Decision Tree Learning algorithm from "Decision tree-based learning to predict patient controlled analgesia consumption and readjustment" by Y.J. Hu, T.H. Ku, R.H. jan, K. Wang, Y.C. Tseng, and S.-F. Yang, [10] is given below.

Algorithm for Decision Tree Learning

```
1. function Tree-Learning(TR, Target t_i, Attr)
2.     Let TR[1…n_1] be an array of the training examples
3.     Let t_i be the target attribute
4.     Let Attr be a set of descriptive attributes
5.     Let R = root node for the tree
6.     if TR contains t_i do
7.        return R
8.     else if Attr is empty do
9.        return R // single node tree with most common
   value of target
10.     else
11.       Let A = select_from_Attr(entropy)
12.       R.A = A
13.       for v_i of values(A) do
14.          Add_branch(R)
15.          Let TR_vi ϵ TR that has A = v_i
16.          if  TR_vi is empty do
17.             leaf = add_leaf(R, TR.target)
18.          else
19.             Tree-Learning(TRvi, Target, Attr-{A})
20.          return R
```

3.2.5 K-Nearest Neighbors

KNN is another fairly popular classification algorithm. When training data is fed into it, it plots them on an n-dimensional space based

on their features. As a consequence, each example in the training set becomes a point in an n-dimensional matrix.

When it encounters an unknown input, it plots that input in the same n-dimensional space and looks for the k-nearest points, based on some distance measure. It goes through the entire training dataset for each new test input. The distance measure depends on the scope and type of the problem.

The pseudo code for K Nearest Neighbors algorithm from "A Machine Learning Approach for Specification of Spinal Cord Injuries Using Fractional Anisotropy Values Obtained from Diffusion Tensor Images" by B. Tay, J. K. Hyun and S. Oh [11] is given below.

Algorithm for K Nearest Neighbors

```
1. Classify (X, Y, x) // X: training data, Y: class labels
   of X, x unknown sample
2. for i = 1 to m do
3.     compute_distance(Xi ,x)
4. end for
5. Let I = { indices for the k smallest distances d(Xi, x)
   }
6. Let L = majority label for {Yi where i ∈ I}
7. return L
```

3.3 Measuring Accuracy

Accuracy is simply a proportion of correct results that the classifier has managed to achieve, but this metric is a bad measure of performance.

This is because, whenever there is a greater number of positive or negative examples, then simply always guessing one class in a binary classification problem will result in a decent number for accuracy, but it fails to correctly judge a model's true performance.

In a binary classification problem, precision is the ratio of true positives to all values classified as positive. On the other hand, recall is the ratio of true positives to all positive values in the dataset. The F-score is a the harmonic mean of precision and recall, and as this metric takes the other two metric into account, the F-score has been chosen as the measure of performance for this paper.

The following equations were taken from "Movies' Genres Classification by Synopsis" by K.-W. Ho, [2] and "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" by D.Powers [12]

True Positives (TP) – positive values correctly classified as positive

False Positives (FP) – negative values incorrectly classified as negative

True Negatives (TN) – negative values correctly classified as negative

False Negatives (FN) – positive values incorrectly classified as negative

$$precision = \frac{TP}{TP + FP} \quad \text{..... (vi)}$$

$$recall = \frac{TP}{TP + FN} \quad \text{.....} \quad \text{(vii)}$$

$$f - measure = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad \text{.....} \quad \text{(ix)}$$

Chapter 4

THE DATASET

This section contains the details of the dataset and the procedure of work.

4.1 Acquiring the Dataset

Acquiring a proper dataset is the first step of every machine learning process. Our main source of data was the “Internet Movie Database (IMDb)” [5]. We used IMDb’s filtering functionality combined with python scripts to scrap their database. We downloaded synopsis and genre information for more than 10,000 movies released in India.

We utilized the very popular “Scrapy” python library to do this. Some of the initial complications were that the structure of HTML page sent from the IMDb servers made it very difficult to extract data from the webpages. So, we had to use some fairly complex regular expressions to collect the datapoints. Afterwards, the script saved the genre and synopsis information in a CSV file.

4.2 Analysis and Cleaning of Dataset

The *Python* libraries *NumPy* and *Pandas* were used for wrangling the data and the *Matplotlib* library was used for producing the charts. The

following table shows the first five rows of the raw dataset that was acquired.

Table-4.1 First five rows of the raw dataset

Title	Description	Genre
Bol Bachchan	When a child falls into the well of the temple...	Action, Comedy, Drama, Romance
Action Hero Biju	The story of Action Hero Biju revolves around ...	Action, Comedy, Thriller
Vivegam	A celebrated covert operation specialist Ajay ...	Action, Thriller
Welcome Back	Uday Shetty and Majnu Bhai have left the under...	Action, Comedy, Crime, Drama
Ek Aur Ek Gyarah: By Hook or by Crook	Two petty thieves Tara and Sitara (Govinda & S…	Action, Comedy, Crime, Musical

The dataset contained 13,875 unique titles, however, for some of the titles, the ‘Genre’ column was blank. We could not use those data points and thus they were removed, and we were left with 13,868 unique movies.

A custom python function was used to extract the unique genres from the dataset and there was a total of 26 different genres. The frequency of the 15 most popular genres are shown in the following table:

Table-4.2. Top 15 genres and their frequency

	Genre	Frequency
1	Drama	8314
2	Comedy	3470
3	Romance	3413
4	Action	3056
5	Thriller	2518
6	Short	2031
7	Family	1765
8	Crime	1610
9	Music	1113
10	Musical	995
11	Mystery	905
12	Adventure	808
13	Documentary	513
14	Fantasy	473
15	History	321

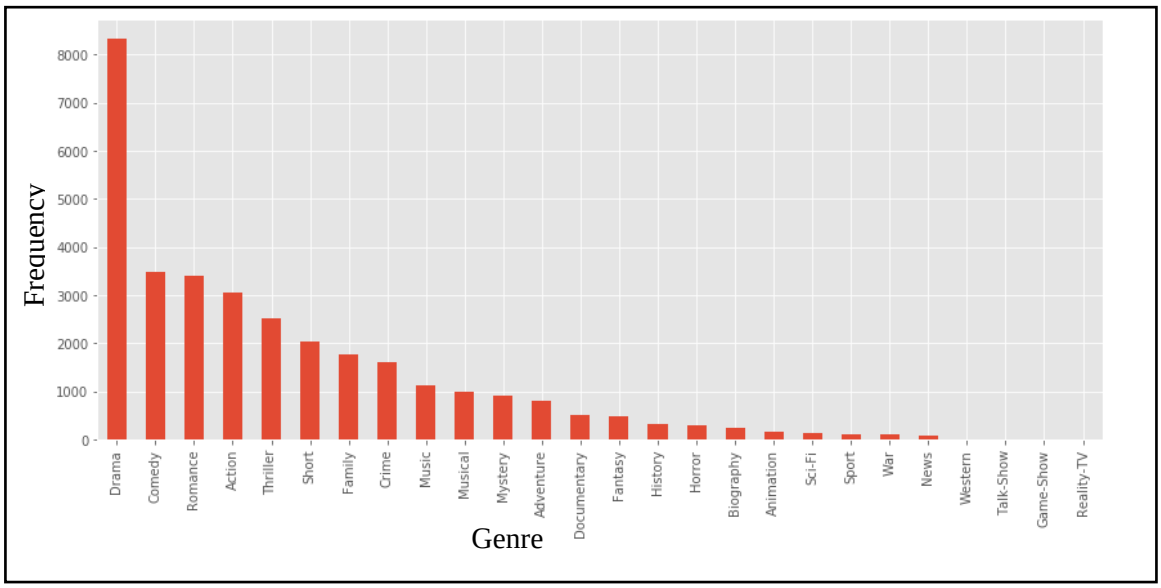

Figure 4.1. Frequency of genres

The most popular genre by a large margin was ‘Drama’ as 8314 movies in the dataset included the label in the genre column. Only four of the genre labels occurred more than 3000 times whereas 17 of the 26 unique genres occurred less than 1000 times. This shows that the genres are far from being distributed evenly.

4.3 Training and Test Dataset

The Scikit-Learn (sklearn) python library, which was used for the machine learning tasks contained the “test_train_split” function within its “model_selection” module. This function was used to divide the processed dataset into two parts – a training set containing 80% of the data and a test set containing 20% of the data. Two datasets are used in order to escape the problem of over fitting. This can arise if the same

dataset is used to both classify and test the model. The classification algorithm will try to fit the data as strongly as possible and the results will show a very high accuracy. However, this is very misleading as the model was fit too closely to the training data and on a different set of data, the model will perform poorly. The goal of our experiment was to create a general model that can be used to classify new Indian movies into genres. If over fitted models were used, then new movies would be far more likely to be misclassified.

4.4 Analysis of the Features

The ‘Description’ of each movie was used to predict the genre. The ‘Description’ itself was a chunk of text that needed to be preprocessed and converted into a numerical form as input to the classification algorithms. In the training set, the mean length of each description was 94.7 words out of which 62.1 were unique.

4.4.1 Exploring Parts of Speech

Python has a library named NLTK that can be used for natural language processing. This pos_tag function from the tag package of the NLTK library can be used to assign a parts of speech label to each word. We used a custom wrapper function and used it to count the number of occurrences of words within each tag. This function was applied on to

each and every description present in the training set and total count of words in each tag is presented in the following figure:

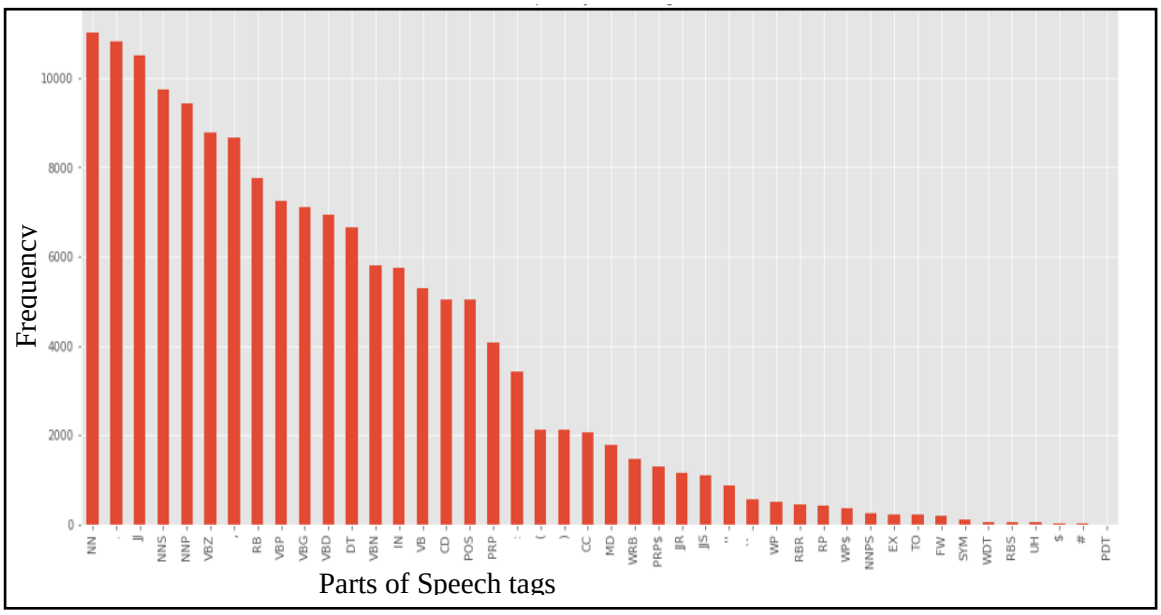

Figure 4.2. Frequency of POS tags

The graph clearly depicts that words with the ‘NN’ tag, which stands for common nouns were the most popular, followed closely by the ‘.’ punctuation mark and the ‘JJ’ tag which represents ordinal adjectives such as ‘third’. The other popular tags were ‘NNS’ and ‘NNP’ which stand for common plural noun and singular proper noun respectively. Selectively removing words with certain tags can dramatically decrease the number of features and can lead to improvements in accuracy.

4.4.2 Processing description features for classification

Every human written text contains a set of words, such as ‘the’ which are used very frequently. Due to their high frequency of use, these words are found in every single description and thus they add no meaningful value when it comes identifying genres. This set of words are called stop words. By removing this set of words from both the training and the test sets, the classification can be done in a shorter time and the results are also more accurate. To remove these stop words the NLTK library is used. The library’s corpus package contains a list of stop words and every instance of words of the aforementioned list that occur in the move descriptions are removed with a custom Python function.

Simply removing the stop words is not enough though. For grammatical reasons, many of the words used in the descriptions would be the same word, written in a different form. For example, ‘kill’, ‘killed’ and ‘killing’ are different forms of the same word. The three individual words can be reduced down to a single word – ‘kill’. This process is known as stemming and it can significantly reduce the number of features and thus positively affect the result of the classifier. To perform the stemming, the “SnowballStemmer” class from the “stem.snowball” package of the “NLTK” library is used. A custom wrapper function uses

a “SnowballStemmer” object to stem each word in each of the descriptions in our dataset.

Once the stop words have been removed and the remaining words stemmed, certain POS tags could be removed to improve result. Through analysis of the dataset, it was discovered that lots of proper nouns, such as names of directors, producers, actors and other cast members were present some of the descriptions. Also present were the names of the characters and locations. This information did not in any way help the classifiers and therefore these words could be removed to reduce the number of features and improve accuracy. The task was simple, as only the words with ‘NNP’ tag needed to be removed.

With the description stripped off of stop words and proper nouns, and with each word reduced to its stem, the text was ready to be transformed into a form suitable to work as an input to the classification algorithms. This was done with the help of the “CountVectorizer” class from “sklearn”. Once the transformation process was complete, a feature matrix consisting of 12,304 unique features was ready to be used for training the algorithms.

4.5 Implementation steps

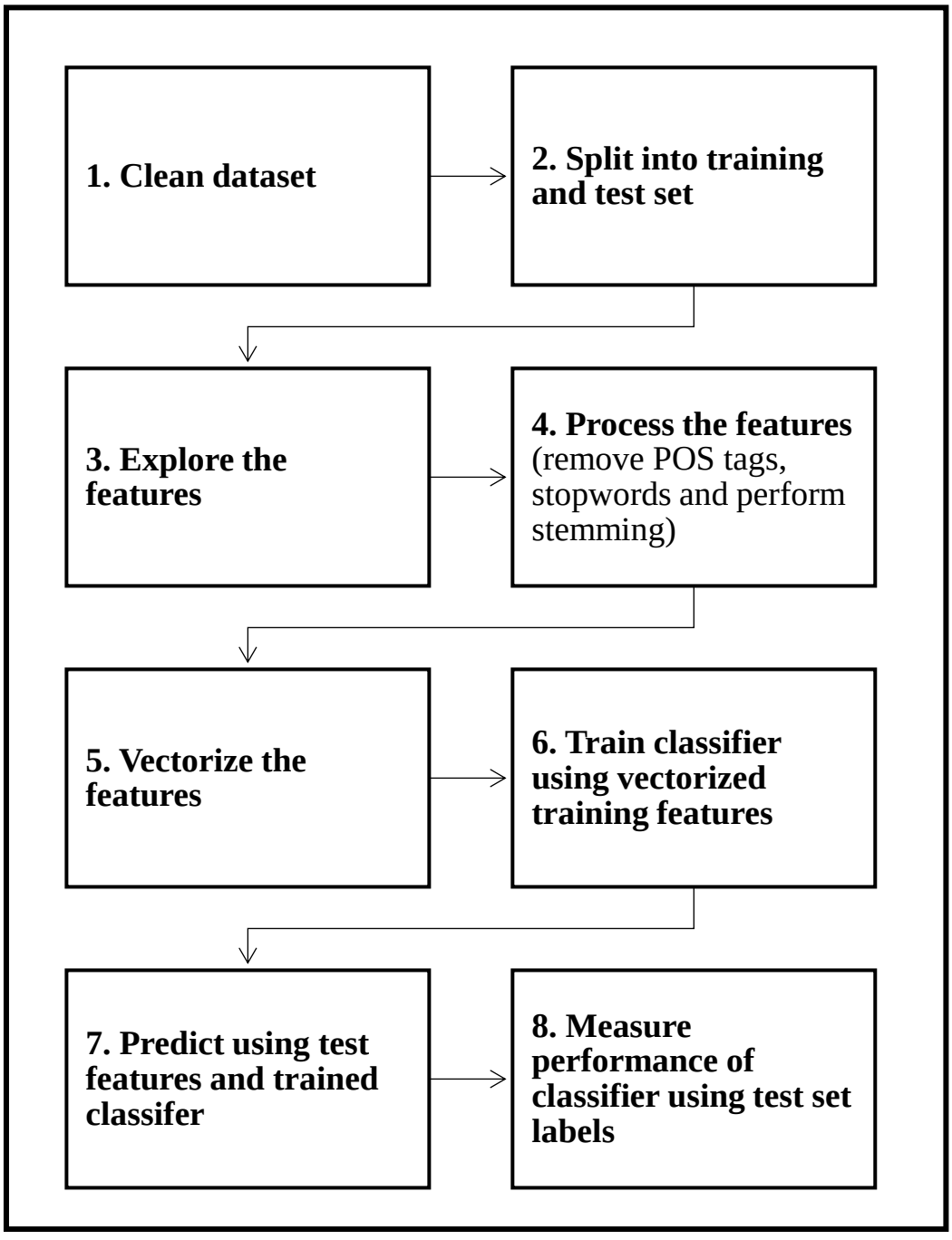

Figure 4.3. Flow chart for classification project

Chapter 5

RESULTS

This chapter contains the results, which are explained with the help of tables and charts.

Each movie can be assigned to more than one genre and therefore it is a multi-label classification problem. However, deciding whether a movie belongs to a certain genre or not is essentially a binary classification problem. To make the process more uniform and easier, each supervised learning classifier is wrapped inside of the "OneVsRestClassifier" that "sklearn" makes available. Then, each of the supervised classifier is trained using the processed training set and parameters are set according to the ones determined by "GridSearchCV" from "sklearn.model_selection". Then the trained models are tested using the test set and the results are presented below.

5.1 Multinomial Naïve Bayes

"Sklearn.naive_bayes module" provides "MultinomialNB" class. The table describes the precision, recall, F1-score and support for each genre in the dataset that used to train and test the model.

- Precision is the fraction of correctly classified movies over all movies classified by algorithm into the genre. It has a minimum value of 0 and a maximum value of 1.
- Recall is the fraction of correctly classified movies over all movies that actually belong to the genre. . It has a minimum value of 0 and a maximum value of 1.
- F1-score is the performance measure used in the experiment. It is the harmonic mean of the precision and recall. . It has a minimum value of 0 and a maximum value of 1.
- Support is the number of positive samples in the test set. There is no maximum value for support and the minimum value is 0.

The results of the classifier are displayed in the table below:

Table-5.1 Classification report for Naïve Bayes

	Precision	Recall	F1-score	Support
Drama	0.70	0.70	0.70	1665
Comedy	0.59	0.39	0.47	707
Romance	0.54	0.53	0.53	683
Action	0.61	0.60	0.60	623
Thriller	0.48	0.41	0.44	517
Short	0.41	0.41	0.41	398
Family	0.25	0.10	0.14	320
Crime	0.46	0.38	0.42	354
Music	0.26	0.05	0.09	225
Musical	0.19	0.03	0.06	201
Average / Total	0.421	0.36	0.386	5693

The classifier performed well on genres with a lot of support, but very poorly on ‘Music’, ‘Musical’ and ‘Family’. This could be because of the lack of support for these movies, or because there are no features in the description that correlate to movies being a part of these genres.

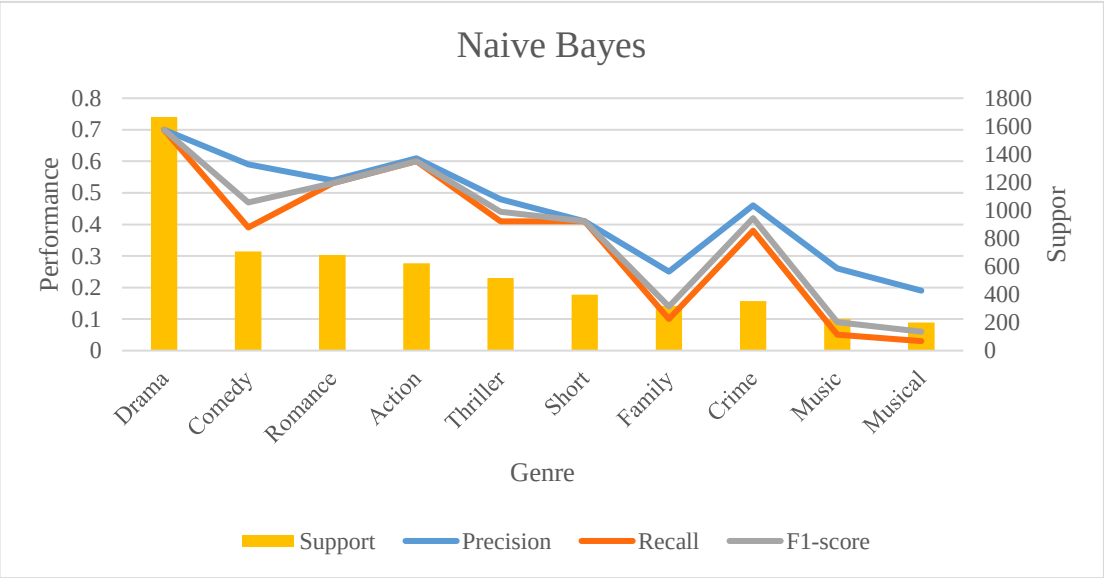

Figure 5.1. Chart for Naïve Bayes

The left-hand y-axis shows the value for ‘Precision’, ‘Recall’ and ‘F1-score’, and the right-hand y-axis shows the value for ‘Support’. The bars refer to the support, and the lines correspond to the performance measures. There appears to be a negative correlation between support and the performance measures. Interestingly, the model performs worse when classifying movies labeled as ‘Comedy’, compared to those labeled as ‘Romance’ or ‘Action’ even though ‘Comedy’ has greater support than the other two genres. The performance for ‘Crime’ is also much better than that for ‘Family’, and this could be attributed to slightly higher support. The performance of the model goes down significantly for the final two genres, ‘Music’ and ‘Musical’.

5.2 Logistic Regression

Sklearn.linear_model provides the Logistic Regression class.

Table-5.2 Classification report for Logistic Regression

	Precision	Recall	F1-score	Support
Drama	0.66	0.69	0.68	1665
Comedy	0.53	0.39	0.45	707
Romance	0.53	0.40	0.46	683
Action	0.62	0.48	0.54	623
Thriller	0.50	0.34	0.41	517
Short	0.49	0.35	0.41	398
Family	0.28	0.17	0.21	320
Crime	0.43	0.23	0.30	354
Music	0.26	0.10	0.14	225
Musical	0.20	0.07	0.11	201
Average / Total	0.45	0.322	0.371	5693

The logistic regression classifier shows similar results with good performance on genres with lots of support but performing very poorly on ‘Music’, ‘Musical’ and ‘Family’. ‘Music’ and ‘Musical’ and ‘Family’ have relatively low support (i.e. positive examples) and thus it becomes

clear that the performance of the algorithms is very much dependent on the number of examples.

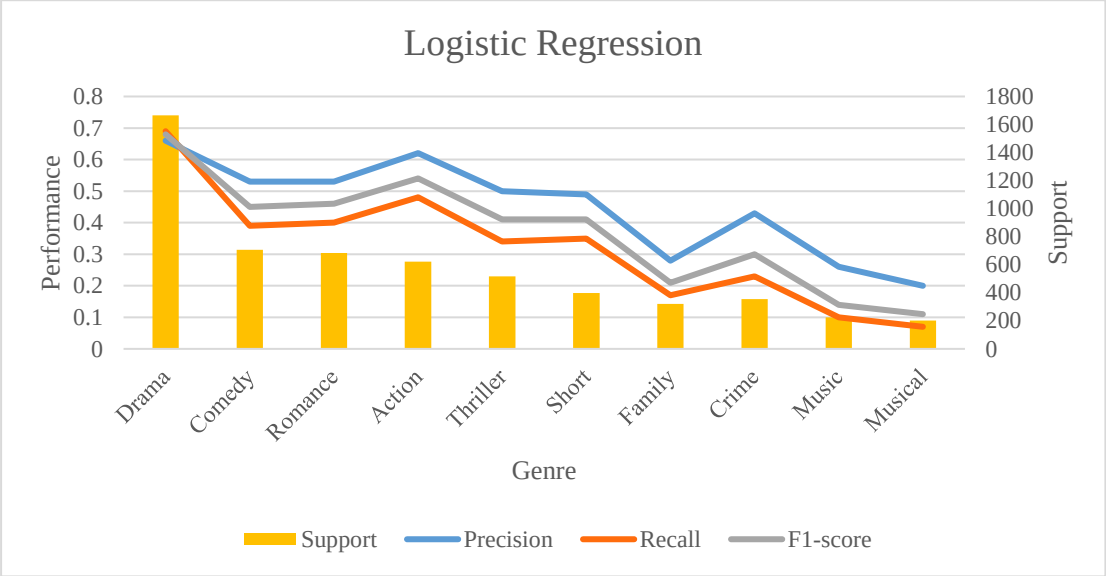

Figure 5.2 Chart for Logistic Regression

The graph shown on figure 5.2 is similar to the one for Naïve Bayes. There appears to be a negative correlation between support and the performance metrics. ‘Action’ performs surprisingly well compared to ‘Comedy’ and ‘Romance’ and again an increase in performance for ‘Crime’ is observed which is likely a result of the increased support. There is a sharp dip in all three performance measures when going from ‘Short’ to ‘Family’, a trend observed in figure 5.1 as well.

5.3 K Nearest Neighbors (KNN)

Sklearn.neighbors provides KNeighborsClassifier class.

Table-5.3 Classification report for K Nearest Neighbors

	Precision	Recall	F1-score	Support
Drama	0.60	0.64	0.62	1665
Comedy	0.31	0.09	0.14	707
Romance	0.49	0.25	0.33	683
Action	0.57	0.10	0.17	623
Thriller	0.33	0.05	0.09	517
Short	0.22	0.20	0.21	398
Family	0.19	0.06	0.09	320
Crime	0.38	0.02	0.04	354
Music	0.10	0.00	0.01	225
Musical	0.25	0.00	0.01	201
Average / Total	0.344	0.141	0.171	5693

The K Neighbors Classifier performs very poorly on genres with a low number of positive examples. Even on ‘Drama’, the genre with the highest number of supports, KNN still performs worse than other competing algorithms.

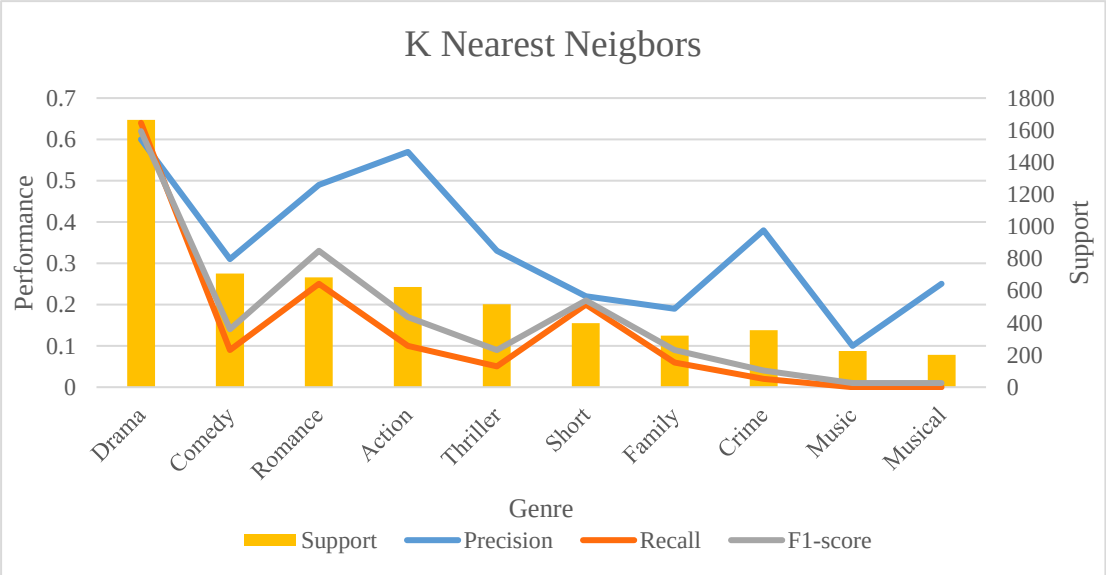

Figure 5.3 Chart for K Nearest Neighbors

The graph clearly depicts the performance of the K Nearest Neighbors algorithm. There is a very sharp fall in recall and therefore F1-score when the algorithm tries to classify ‘Comedy’ instead of ‘Drama’. Precision is much higher than recall for all genres. The recall of the algorithm, unlike its precision does not even increase when classifying ‘Crime’ and it approaches zero as support decreases further, as observed when classifying movies into ‘Music’ and ‘Musical’.

5.4 Decision Tree

Sklearn.tree provides DecisionTreeeClassifier class.

Table-5.4 Classification report for Decision Tree

	Precision	Recall	F1-score	Support
Drama	0.64	0.65	0.65	1665
Comedy	0.39	0.36	0.37	707
Romance	0.42	0.40	0.41	683
Action	0.43	0.40	0.42	623
Thriller	0.35	0.32	0.33	517
Short	0.32	0.33	0.33	398
Family	0.18	0.16	0.17	320
Crime	0.25	0.22	0.23	354
Music	0.16	0.14	0.15	225
Musical	0.08	0.08	0.08	201
Average / Total	0.322	0.306	0.314	5693

Again, a similar pattern emerges with the decision tree classifier. Genres with high support are predicted more accurately compared to those with lower support.

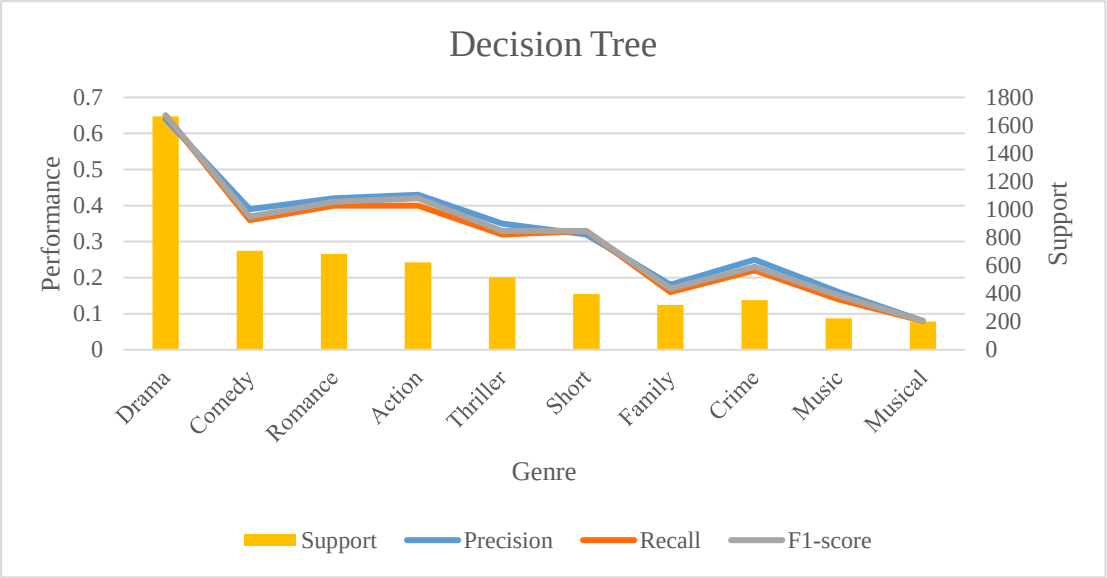

Figure 5.4 Chart for Decision Trees

The precision and recall values for the Decision Tree classifier remains close to equal unlike the other algorithms. A similar pattern for performance and movie genre emerges with support being negatively correlated to the performance measures. There is an increase in all three measures of performance for ‘Crime’, which is probably due to an increase in support.

5.5 Linear Support Vector Machine (SVM)

Sklearn.svm provides the SVC class.

Table-5.5. Classification report for linear SVM

	Precision	Recall	F1-score	Support
Drama	0.65	0.65	0.65	1665
Comedy	0.46	0.42	0.44	707
Romance	0.45	0.43	0.44	683
Action	0.52	0.48	0.50	623
Thriller	0.40	0.39	0.39	517
Short	0.40	0.39	0.40	398
Family	0.22	0.23	0.22	320
Crime	0.31	0.27	0.29	354
Music	0.15	0.13	0.14	225
Musical	0.10	0.09	0.09	201
Average / Total	0.366	0.348	0.356	5693

Linear SVM again shows similar output to the other algorithms. The precision and recall are close to one another. Performance is decent when trying to classify movies into genres with lots of support, but when support decreases, the performance falls sharply.

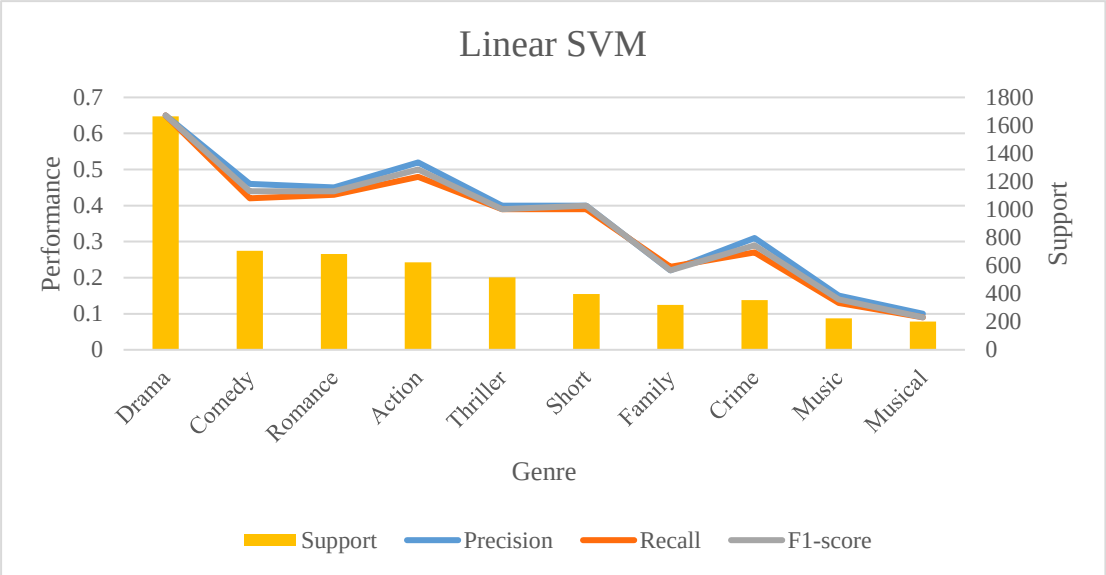

Figure 5.5 Chart for Linear SVM

The graph shows that the dip for the performance measures are not as sharp when Genre is changed from ‘Drama’ to ‘Comedy’ is not as sharp as the other algorithms. When genre is changed from ‘Comedy’ to ‘Romance’, there is a slight increase in performance, which increase further when genre changes to ‘Action’, From then onwards, there is a steady dip in performance as support decrease, but there is a sharp rise between ‘Family’ and ‘Crime’ due to an increase in support.

5.6 Comparison of Algorithms

The measure of performance is given by precision, recall and f1-score. To compare the performance of each algorithm on each genre is presented in the following tables.

The genre ‘Drama’ has the highest support, i.e. the greatest number of positive examples in the test set. All algorithms perform relatively well, with Naïve Bayes coming out on top. The precision and recall scores are almost the same for every algorithm with recall scores being slightly higher than precision scores for Logistic Regression, K Nearest Neighbors and Decision Trees.

Table 5.6.1 Results for Drama

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.7	0.7	0.7	1665
Logistic Regression	0.66	0.69	0.68	1665
K Nearest Neighbors	0.6	0.64	0.62	1665
Decision Trees	0.64	0.65	0.65	1665
Linear SVM	0.65	0.65	0.65	1665

‘Comedy’, the second most popular genre in the dataset has less than half the number of supports compared to ‘Drama’ and therefore all algorithms perform much more poorly with recall and precision scores going down by a large margin. Recall scores are much lower compared to precision scores for all algorithms, with K Nearest Neighbors faring the

worst with a very low recall score of 0.09. Naïve Bayes once again tops the chart with Logistic Regression and Linear SVM close behind.

Table 5.6.2 Results for Comedy

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.59	0.39	0.47	707
Logistic Regression	0.53	0.39	0.45	707
K Nearest Neighbors	0.31	0.09	0.14	707
Decision Trees	0.39	0.36	0.37	707
Linear SVM	0.46	0.42	0.44	707

An interesting phenomenon is observed when trying to classify the third most populous genre. The F-score for each algorithm is higher when compared to ‘Comedy’ and no algorithm performs exceptionally poor relative to others. The recall scores are slightly lower than precision scores for all algorithms, with K Nearest Neighbors having a far worse recall score compared to precision, which resulted in the lowest F-score in comparison to all other classifiers.

Table 5.6.3 Results for Romance

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.54	0.53	0.53	683
Logistic Regression	0.53	0.4	0.46	683
K Nearest Neighbors	0.49	0.25	0.33	683
Decision Trees	0.42	0.4	0.41	683
Linear SVM	0.45	0.43	0.44	683

The precision scores are greater than recall scores with K Nearest Neighbors once again posting a very low recall score in comparison to its precision score.

Table 5.6.4 Results for Action

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.61	0.6	0.6	623
Logistic Regression	0.62	0.48	0.54	623
K Nearest Neighbors	0.57	0.1	0.17	623
Decision Trees	0.43	0.4	0.42	623
Linear SVM	0.52	0.48	0.5	623

The same pattern emerges for the ‘Thriller’ genre with recall scores slightly worse than precision. A very low recall score is observed when

using K Nearest Neighbors is used to classify thriller movies in the dataset.

Table 5.6.5 Results for Thriller

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.48	0.41	0.44	517
Logistic Regression	0.5	0.34	0.41	517
K Nearest Neighbors	0.33	0.05	0.09	517
Decision Trees	0.35	0.32	0.33	517
Linear SVM	0.4	0.39	0.39	517

The ‘Short’ genre has only 398 positive samples in the test set which is significantly lower than other genres. Logistic Regression has the best precision but its overall performance as measured by F-score is equal to Naïve Bayes as the recall for Logistic Regression is low. K Nearest Neighbors performs terribly posting the lowest F-score, although when classifying this genre, the difference between precision and recall is low for the algorithm.

Table 5.6.6 Results for Short

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.41	0.41	0.41	398
Logistic Regression	0.49	0.35	0.41	398
K Nearest Neighbors	0.22	0.2	0.21	398
Decision Trees	0.32	0.33	0.33	398
Linear SVM	0.4	0.39	0.4	398

All of the algorithms perform poorly when trying to correctly classify movies into the ‘Family’ genre. This can perhaps be attributed to the low support value. However, the value is not significantly lower than it was for ‘Short’ genre but the F-scores are much lower. Perhaps there is little information in the textual description of a movie to predict whether the movie can be classified as family or not. All of the algorithms perform poorly but ‘Linear SVM’, followed closely by ‘Logistic Regression’ perform better than the others. Surprisingly Naïve Bayes is much worse than the other algorithms, although it is better than K Nearest Neighbors when trying to correctly classify ‘Family’ movies.

Table 5.6.7 Results for Family

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.25	0.1	0.14	320
Logistic Regression	0.28	0.17	0.21	320
K Nearest Neighbors	0.19	0.06	0.09	320
Decision Trees	0.18	0.16	0.17	320
Linear SVM	0.22	0.23	0.22	320

When trying to accurately classify ‘Crime’ movies, Naïve Bayes performs much better than other algorithms. The recall scores are lower in comparison to precision scores for all algorithms and the problems is particularly acute, as noticed on several previous occasions, for K Nearest Neighbors.

Table 5.6.8 Results for Crime

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.46	0.38	0.42	354
Logistic Regression	0.43	0.23	0.3	354
K Nearest Neighbors	0.38	0.02	0.04	354
Decision Trees	0.25	0.22	0.23	354
Linear SVM	0.31	0.27	0.29	354

The F-scores for each and every algorithm was very poor for ‘Music’. Even though the difference in support between music and family is present, it is not extremely big, but scores are much lower. This could be attributed to two potential causes. One is that once support goes below 350, the performance of every algorithm goes down. The second explanation is that ‘Music’ and ‘Family’ are genres for which there does not exist a correlation between narrative elements found in the textual description and these genres.

Table 5.6.9 Results for Music

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.26	0.05	0.09	225
Logistic Regression	0.26	0.1	0.14	225
K Nearest Neighbors	0.1	0	0.01	225
Decision Trees	0.16	0.14	0.15	225
Linear SVM	0.15	0.13	0.14	225

The ‘Musical’ genre is the genre with the lowest support and similarly has the lowest performance as measured by the F-score for each and every single algorithm that has been used during the process of writing this paper. It appears that Naïve Bayes performs poorly in comparison to other algorithms when the support is lower than 350, but

performs better relative to the other classifiers when support is greater than 350. This trend was noticed for genres ‘Family’ and ‘Music’ and the same phenomenon is observed when trying to correctly classify movies having the ‘Musical’ genre label. All scores are very low and amongst the classifiers for this category, Logistic Regression performs the best followed by Linear SVM.

Table 5.6.10 Results for Musical

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.19	0.03	0.06	201
Logistic Regression	0.2	0.07	0.11	201
K Nearest Neighbors	0.25	0	0.01	201
Decision Trees	0.08	0.08	0.08	201
Linear SVM	0.1	0.09	0.09	201

The average of precision, recall and F-score for each algorithm is shown on the following the table. It is observed that performance of all the algorithms are somewhat comparable. Naïve Bayes is the best overall performer followed by Logistic Regression, Linear SVM, and Decision Trees. K Nearest Neighbors has the lowest F-score by a large margin and this is due to the classifier posting a very low average recall score. The

precision score of the aforementioned algorithm is still on the lower end of the spectrum but is higher than that of Decision Trees.

Table 5.6.11 Average/ Total for different algorithms

Algorithm	Precision	Recall	F1-score	Support
Naïve Bayes	0.421	0.36	0.386	5693
Logistic Regression	0.45	0.322	0.371	5693
K Nearest Neighbors	0.344	0.141	0.171	5693
Decision Trees	0.322	0.306	0.314	5693
Linear SVM	0.366	0.348	0.356	5693

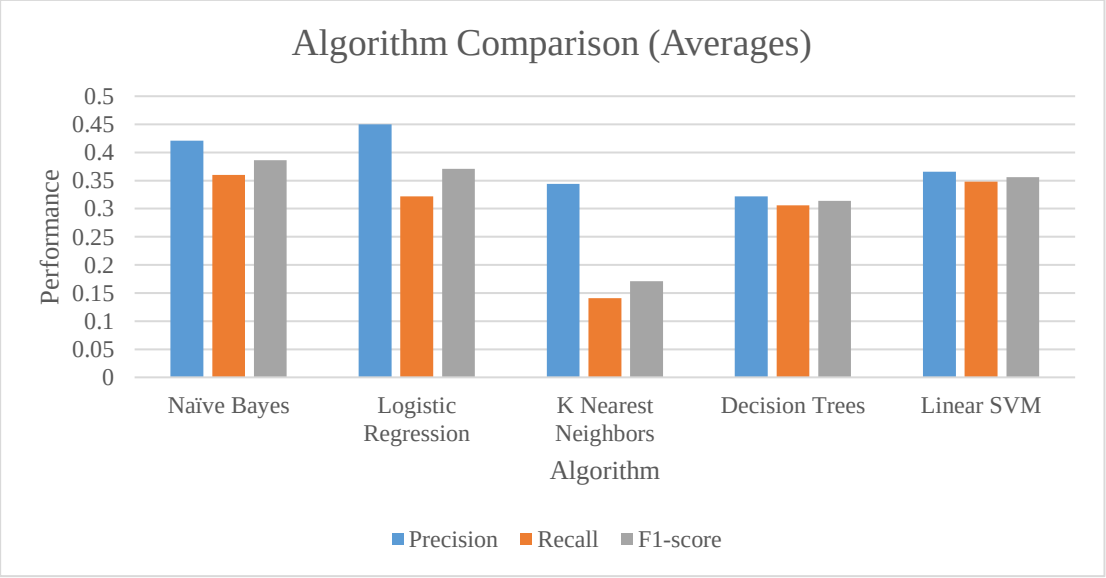

Figure 5.6.1 Algorithm Comparison (Averages)

The results are visualized in fig 5.6.1. Logistic Regression has the best precision but a worse recall brings the F-score down resulting in

Naïve Bayes being the algorithm with the best F-score. All of the algorithms are close to each other save for K Nearest Neighbor as its terrible recall score brings down its F-score.

5.7 Comparison against existing work

As different datasets and methods were used, a direct comparison is not possible. Nevertheless, we have tried to compare the results of this project against the one done by Ka-Wing Ho[2] as that is the one which matches most closely.

The following table shows the average precision, average recall and average F-measure for all genres as was found by Ka-Wing Ho[2].

Table 5.7.1 Average results over all genres

	Precision	**Recall**	**F Measure**
SVM(Default)	0.66141	0.39572	0.47151
SVM(weighted)	0.47689	0.62533	0.53785
SVM(1:1)	0.51205	0.61631	0.54999
KNN(k=97)(MLE)	0.40987	0.73400	0.51580
KNN(k=88)(MAP)	0.64093	0.33261	0.40060
PMM(MLE)	0.46371	0.54234	0.48550
PMM(MAP)	0.53624	0.45876	0.47375
NN(λ = 1, P C = 1000)	0.67630	0.41513	0.49896
NN(λ = 1, P C = 4000)	0.65444	0.43225	0.50849
NN(λ = 1, P C = 8000)	0.62493	0.45708	0.52044
NN(λ = 1, No PCA)	0.64453	0.43666 0	.50938
NN(λ = 0, No PCA)	0.66886	0.41655	0.49786

For the sake of comparison, the values of 5.7.1 and 5.6.1 are charted together and shown in the following figure.

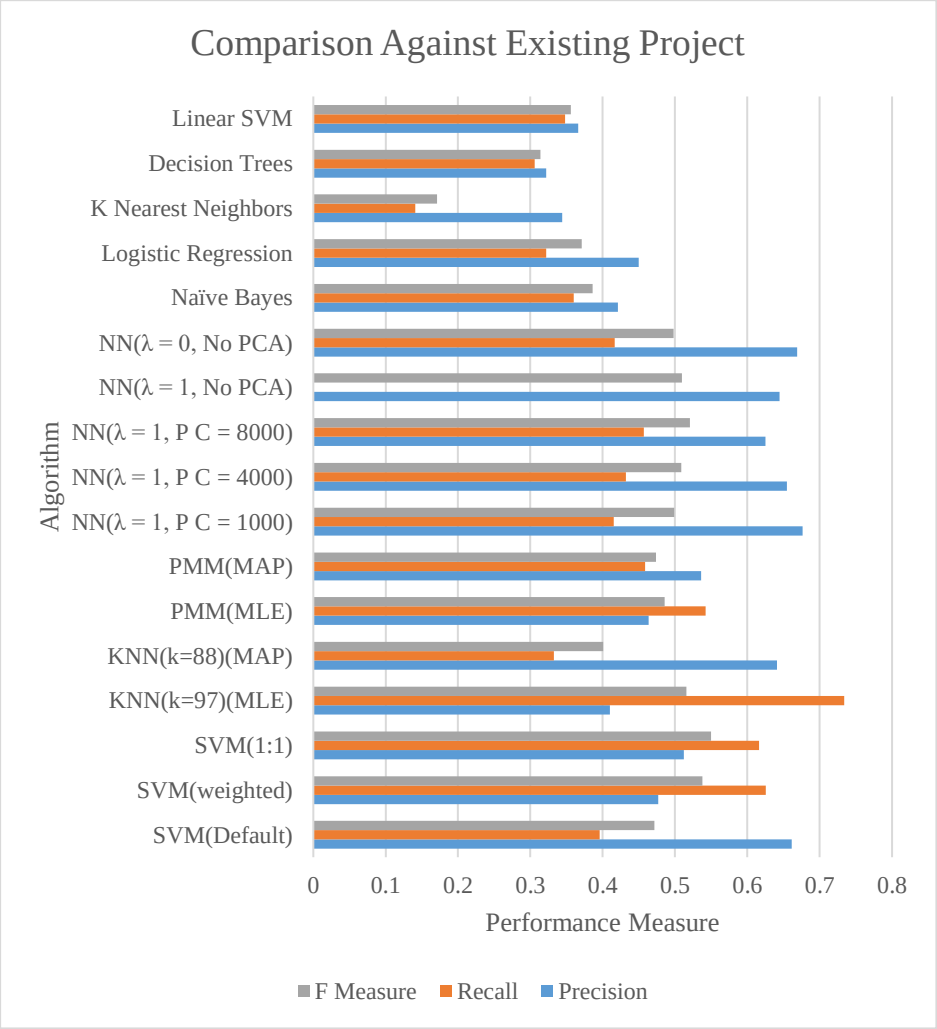

Figure 5.7.1 Comparison against existing project

In figure 5.7.1, the first five algorithms are the ones used for this project whereas the rest are from [2]. This figure is only valuable as a rough illustration because not only were the algorithms used different, but also, the dataset and distribution of genres were different.

For further illustrations, the following figures compare the value of precision and recall achieved for each genre by Ho using SVM. Note the support and data used in this project are different and discrepancies in values might be attributed to the aforementioned differences.

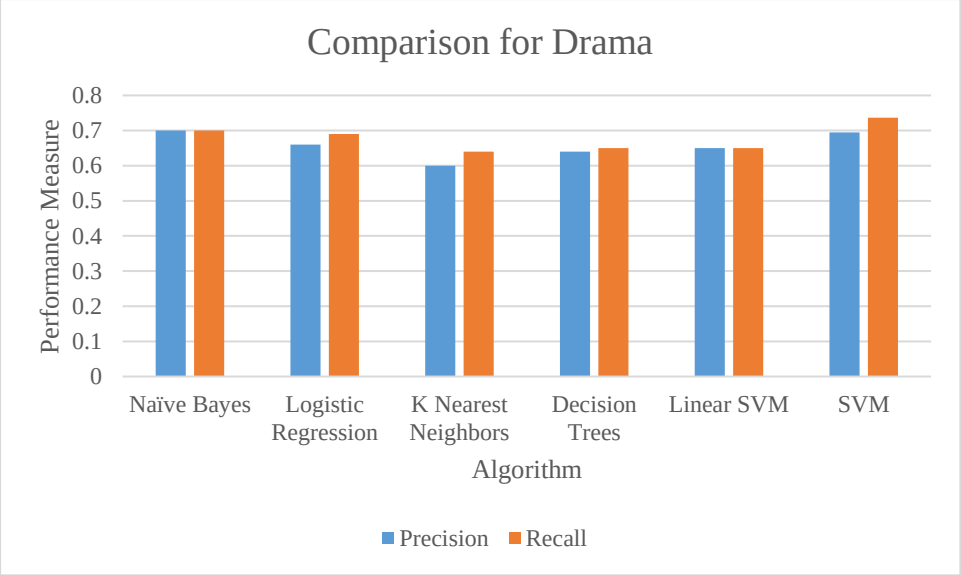

Figure 5.7.2 Comparison for Drama

The above figure shows the comparison of algorithms for Drama, whose values are taken from table 5.6.1. The SVM column comes from [2]. Accurate comparisons cannot be made due to the difference in support and the dataset itself, but it appears that for ‘Drama’, a genre with the highest support in both this paper and in [2], the results obtained in this experiment are comparable to the ones obtained by Ka-Wing Ho[2]

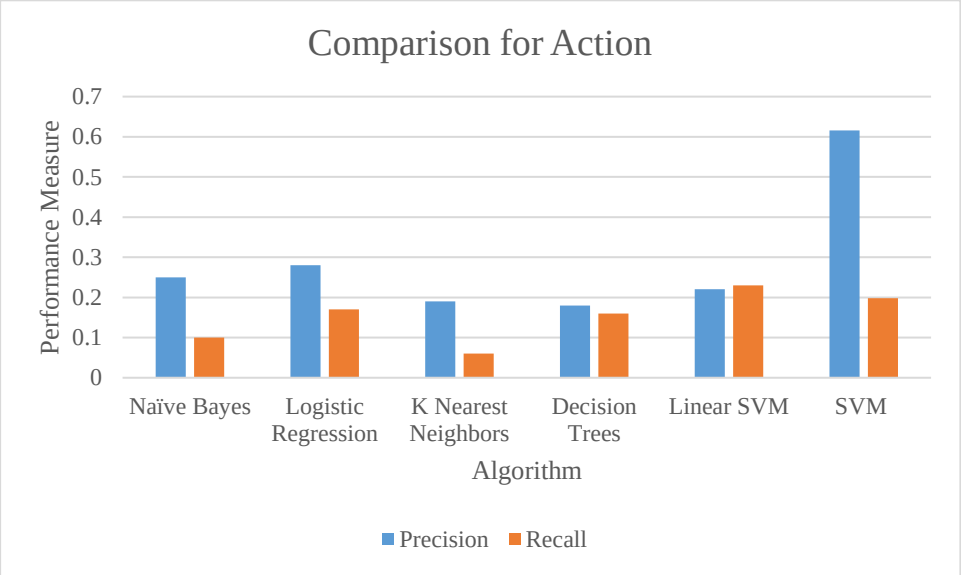

Figure 5.7.3 Comparison for Action

The above figure compares the precision and recall scores for each algorithm plus the SVM as trained and tested by Ka-Wing Ho[2]. The values are for the 'Action' genre and the exact numbers are displayed in table 5.6.4. 'Action' is a genre with medium low support both in our dataset and the one used by Ho[2]. It appears that of the algorithms used in this paper, Logistic Regression has the best precision whereas Naïve Bayes has the best recall. SVM from [2] has better precision than all of the other algorithms but the second lowest recall. However direct comparisons cannot be made as the datasets are different and the distribution of support is also different.

A comparison of algorithms for the lowest support cannot be made as 'Music' and 'Musical' are not present in the dataset used by Ho[2] and

‘Crime’ is proportionately more prominent than in the dataset used for this project. This leaves ‘Family’ as the comparable genre with the lowest support and the results are displayed in the following chart.

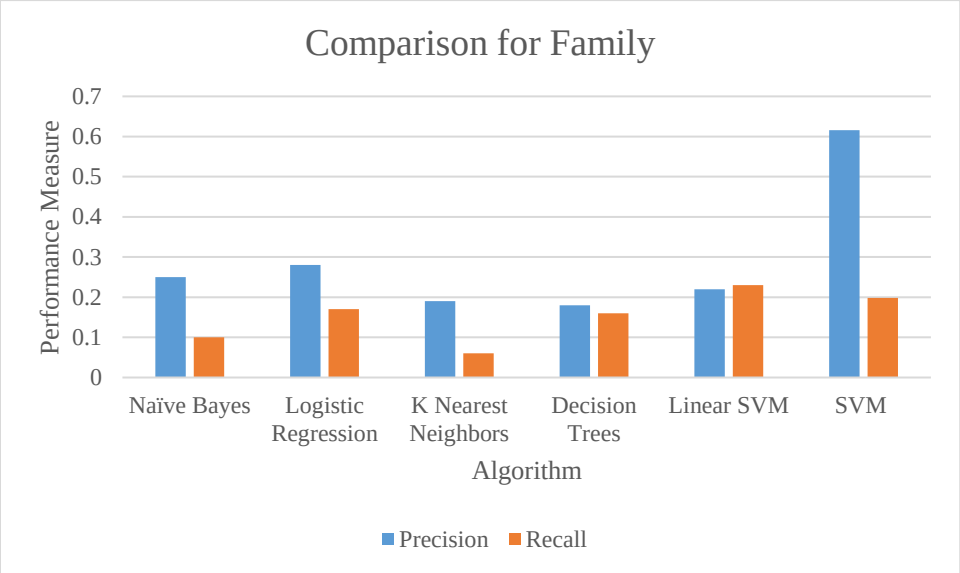

Figure 5.7.4 Comparison for Family

Figure 5.7.3 shows the values of precision and recall for ‘Family’ as shown in table 5.6.7 in addition to the rightmost SVM column taken from Ka-Wing Ho’s research[2]. Amongst the algorithms used in this paper, Logistic Regression shows the best performance as it has both greater precision and recall values. The SVM from [2] eclipses all other algorithms when it comes to precision but shows comparable recall scores. However, it is to be noted that the values are from a different dataset and therefore direct comparisons cannot be made.

Chapter 6

CONCLUSION

This section talks about the conclusion and future work of the thesis.

6.1 Conclusion

In conclusion, it can be ascertained that there is enough data in the IMDB database for Indian Movies that can used to classify them into genres. However, it was discovered that the performance of the algorithms was largely correlated with the number of positive samples of a particular genre and therefore if a movie was labelled as belonging to one of the more popular genres such as ‘Drama’, then the supervised learning classification algorithms performed reasonably but when the algorithm tried to classify a movie into a genre for which there were a small number of positive examples, then the results were generally very poor. The research was limited to only movies originating in India, which led to a dataset consisting of 13,868 movies, a far lower number than the total IMDB database which exceeds 150,000 movies. If the dataset was not restricted, then there would likely have been an increase in accuracy.

When the algorithms were compared to one another, a pattern emerged. The relatively simple classifiers, Naïve Bayes and Logistic

Regression performed better than the more complex models. Naïve Bayes was the best performing algorithm for all genres save for the three with the lowest positive samples, and in those cases, it was superseded by Logistic Regression. Linear SVM and Decision Trees performed slightly worse than the others but K Nearest Neighbors had the lowest performance by a large margin. Interestingly, K Nearest Neighbors had decent average precision but recall was far worse, which led to the F1-score — the performance measure that was used in the thesis — falling far below that of the other classifiers. Further research will be required to discover the root cause of the poor performance of K Nearest Neighbors. All of the algorithms perform poorly when number of positive examples for a label fall and while it cannot be stated with absolute certainty that the cause of the poor performance is simply due to a lack of data there does exist an undeniable correlation.

6.2 Future Work

The performance of any machine learning algorithm depends on the data that is being used to train the model and the parameters. In this experiment, the default parameters were used to try and predict the genre of a movie based solely on its textual description. The performance of the models could potentially be improved by fine tuning the parameters of

the algorithms and further processing the raw data before it is used to train the models.

Another major issue faced in this investigation was that some genres of movies simply did not have enough data points to properly train the algorithm. While the obvious remedy to the problem would be to gather more data, the task could prove challenging if the limitation of using a dataset comprised exclusively of movies originating from India was to remain. The second, although not far from perfect remedy would be to balance the labels by over sampling or under sampling.

In this thesis, classic supervised machine learning algorithms were used. While better processing of training data and fine tuning of the parameters of the models could lead to improved performance, a deep learning approach using neural networks and related algorithms could potentially result in greater performance.

Performance can potentially be improved even further by identifying the primary genres for the movies. As pointed out in J. J. Tanenbaum's paper[1], IMDb does not identify the primary genre of a movie. If we can find a data source to identify the primary genres of the movies in our database and remove the other less important genres from each movie results should improve.

Appendix A POS Tag List

CC	coordinating conjunction
CD	cardinal digit
DT	determiner
EX	existential there (like: "there is" ... think of it like "there exists")
FW	foreign word
IN	preposition/subordinating conjunction
JJ	adjective 'big'
JJR	adjective, comparative 'bigger'
JJS	adjective, superlative 'biggest'
LS	list marker 1)
MD	modal could, will
NN	noun, singular 'desk'
NNS	noun plural 'desks'
NNP	proper noun, singular 'Harrison'
NNPS	proper noun, plural 'Americans'
PDT	predeterminer 'all the kids'
POS	possessive ending parent's
PRP	personal pronoun I, he, she
PRP$	possessive pronoun my, his, hers
RB	adverb very, silently,
RBR	adverb, comparative better

RBS	adverb, superlative best
RP	particle give up
TO	to go 'to' the store.
UH	interjection errrrrrrrm
VB	verb, base form take
VBD	verb, past tense took
VBG	verb, gerund/present participle taking
VBN	verb, past participle taken
VBP	verb, sing. present, non 3d take
VBZ	verb, 3rd person sing. present takes
WDT	wh determiner which
WP	wh pronoun who, what
WP$	possessive wh pronoun whose
WRB	wh abverb where, when

Appendix B Software Packages Used

Scikit Learn: We have heavily utilized Scikit-learn library for our implementation. Scikit-learn is a free software library written mostly in the Python programming language. Contained within it, is a plethora of different classification, regression and clustering algorithms e.g. k-means cluster, Support Vector Machines, random forests etc. It is very well documented and well maintained. It is extremely popular for programming Machine Learning based systems. It is fully compatible with other data analysis libraries like Scipy and NumPy.

Scrapy: One of the most popular and most easy-to-use data scraping libraries written in Python. You can get it up and running and scraping data within minutes. But it also features advanced features for those who need it. Like many other popular python libraries, it too is well documented and maintained.

Natural Language Toolkit (NLTK): NLTK is a Natural Language processing library written in Python. It is presently one of the most popular NLP libraries in the world.

REFERENCES

[1] J. Tanenbaum, “Issues in Decision Tree Classification of Film Genre Using Plot Features”, Simon Fraser University, 2014.

[2] K.-W. Ho, “Movies’ Genres Classification by Synopsis”, Stanford University, 2011.

[3] A. C. A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2017.

[4] S. Bird, E. Klein and E. Loper, Natural language processing with Python. Beijing [etc.]: O'Reilly, 2011.

[5] "IMDb: Most Popular Titles", *IMDb*, 2017. [Online]. Available: http://www.imdb.com/search/title?country_of_origin=in. [Accessed: 13- Aug- 2017].

[6] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features”, in Machine learning: ECML-98, 1998, p. 137–142.

[7] S. Shiju and G. Surya, “Crime analysis and prediction using data mining”, in First International Conference on Networks & Soft Computing (ICNSC), Guntur, Andhra Pradesh, India, 2014, p. 406-412.

[8] M. Arias, “Linear and Logistic Regression”, Polytechnic University of Catalonia, 2012.

[9] S. Andrews, “Support vector machines for multiple-instance learning”, in Advances in neural information processing systems, 2003, p. 577-584.

[10] Y.-J. Hu, T.-H. Ku, R.-H. jan, K. Wang, Y.-C. Tseng, and S.-F. Yang, “Decision tree-based learning to predict patient controlled analgesia consumption and readjustment”, in BMC medical informatics and decision making, 2012.

[11] B. Tay, J. K. Hyun and S. Oh, “A Machine Learning Approach for Specification of Spinal Cord Injuries Using Fractional Anisotropy Values Obtained from Diffusion Tensor Images”, Dankook University, 2013.

[12] D. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation", Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37-63, 2011.

[13] A. Rajaraman, Mining of Massive Datasets. 2011, p. 1–17.