Sentiment Analysis on Movie Reviews Using Different Machine Learning Approaches.

Amirush Javare
California State University, Sacramento
amirushjavare@gmail.com

Neha Maiya
California State University, Sacramento
maiya.neha@gmail.com

Abstract: From the last 10 years, social media popularity has increased at an alarming rate. Everyone is utilizing technology at higher rates than earlier. People are now sharing their emotions and opinions on social media sites allowing others to know what their opinions are about a particular thing. Many companies are utilizing the data from the website to generate meaningful information out of it which can be further used for business purposes. Huge textual data is available on sites like Amazon, IMDB, Rotten Tomatoes on movies and analyzing such massive data manually is a tedious task. So, to speed up the process we are using sentiment analysis. Sentiment analysis is a submodule of opinion mining where the analysis focuses on the extraction of text and opinions of the people on a particular topic. Analyzing the words, the user writes and predict the result of the movies. We are making use of IMDB reviews on movies to predict how the user rated the movies and predict the movies have a positive or negative review. We proposed a model that includes different sentiment analysis methods which will help us to extract useful information from the data and predict which is the most suitable classifier for this particular domain by looking at accuracy, precision-recall, and many other factors. Models like Naïve Bayes, Support Vector Machine, Logistic regressions and random forest. Also, for presentation we are using Confusion matrix which will show us precision, recall, error and F-1 score. Due to the lack of strong grammatical formats in movie reviews which is an informal jargon we also take into account the N-Grams and count vectorizer and TF-IDF approach as well. Tokenization is used to transfer the input string into a word vector, stemming is used for extracting the root of the words, while feature selection fetches the essential word and lastly classification is used to classify the movie as positive or negative.

Keywords—Sentiment Analysis, precision-recall, N-Grams, Count Vectorizer, Tokenization, stemming, Random Forest, Naïve Bayes, Logistic Regression, Support Vector Machine, accuracy, TF-IDF.

1. INTRODUCTION

Movies are the most convenient ways for the people for entertainment. But only few movies are success and are rated high. There are many ratings websites that will help the movie goers to decide which movie they should watch which not. Websites likes IMDB, Rotten tomatoes are the leading ones amongst those. The rating on this website determine the success of the movie. So, there isn't any formula which can provide the exact and correct prediction of this reviews. So, sentiment analysis comes into picture. Sentiment analysis has become hot topic now a days and many big companies are investing a lot of time of time to predict the results. Sentiment Analysis is a technology that will play a pivotal role in the coming years.

The working principle of such techniques includes tokenization, word filtering, stemming and classifications. In Tokenization text needs to be segmented into units such as words or numbers or punctuations before we can do any operations on the data. Next comes stemming which is the process of removing prefixes and affixes to convert a particular word into its stem. We are also performing TF-IDF which will help us to determine the frequency of word in our dataset and how important a particular word is. After the preprocessing is done, we will analyze our model by performing Naïve Bayes, Support Vector Machine, Logistic regressions and random forest**.** So, we find the best model based on different factors like accuracy, rmse**,** factors of confusion matrix like sensitivity, error, precision. We analyze and study the features that affect the scores of our review text**.** Also, we study the approaches and also get a deeper understanding of the problem domain.

2. RELATED WORK

1. Mais Yasen, Sara Tedmori: In this paper authors have implemented a model that includes tokenization's, stemming on the model and on some words, feature extractions also done. Later they have evaluated models on 8 different classifiers and for better comparison they have also utilised 5 different metrics as well. After comparison random forest gives the best result of accuracy of 96% based on the metrics [1].

2. Tirath Prasad Sahu, Sanjeev Ahuja: In this paper authors are utilizing sentiment expression to classify the polarity of the

movie reviews and furthermore perform feature extractions, ranking. This feature is the used to train multilabel classifier to classify the correct label for the movie. So, in this papers authors concludes that random forest classification attains the best accuracy amongst the stated models in the paper which is 88.95%. [2].

3. Unggul Widodo Wijayanto, Riyanarto Sarno: This paper focuses on supervised methods and also TF-IDF is used to convert words to features. To, improve the quality authors have also utilised CHI2 and stop words. Models like K-folds, cross validation to get results. The authors conclude that context-based stop words enrich the number of stop words that removes bias features. [3].

4. Tejaswini M. Untawale, Prof. G. Choudhari: The authors are proposed evaluated model like naïve Bayes, Random Forest. Also, they have performed feature extractions as well. Lastly the conclude by stating that Naïve Bayes requires more memory as they are considering memory and time as their pivotal part for evaluation. Random Forest is best amongst the two. [4].

5. Sourav Mehra, Tanupriya Choudhary: In this paper authors have implemented SVM and Naïve Bayes and comparision is done between by observing the accuracies of the model They have taken data of IMDB movie reviews which possess of 25000 each for positive and negative provided by the Cornell University The authors concludes by stating that SVM has better accuracy over Naïve Bayes 87.33%. [5].

6. Nisha Rathee, Nikita Joshi, Jaspreet Kaur: In this paper author are trying to derive the polarity of reviews of airline dataset as good, bad or average. The authors have implemented models like Random Forest, SVM, k-nearest, decision tree, decision tree. Finally, authors conclude that there is no algorithm which is better because some may word better in one dataset while other wont better as compared. According to the observations the authors states that Random Forest is best in Classification of IMDB reviews.

3. DESCRIPTION OF DATA

We have gathered data from [7] which includes a dataset which has 50000 reviews from IMDB which is equally divided into 25000 for training and testing. There are only 30 reviews per movie as reviews for the same movie tend to have corelated ratings. Furthermore, the train and test sets contain a disjoints set of movies so memorizing a particular movie terms and their associated labels would have no significant. A negative review is given a score of <=4 out of 10 while a positive one holds a score of >=7 and a neutral review has scores from >4 and <7. There is also dataset which is unlabeled for unsupervised learning. In that reviews of any rating are included and there are an even number of reviews>5 and <=5. In the figure 3.1 below we can see the head of data for the first 5 rows.

Figure 3.1

```
#  concatenating positive and negative files together in reviews_train
reviews_train = pd.concat([
    pd.DataFrame({"review":P_train, "Label":1, "file":posFiles}),
    pd.DataFrame({"review":N_train, "Label":-1, "file":negFiles})
], ignore_index=True).sample(frac=1, random_state=1)
reviews_train.head()
```

	review	Label	file
21492	I have copy of this on VHS, I think they (The ...	-1	6844_1.txt
9488	After several extremely well ratings to the po...	1	7290_10.txt
16933	I still don't know why I forced myself to sit ...	-1	2740_1.txt
12604	Mt little sister and I are self-proclaimed hor...	-1	10094_1.txt
8222	I have personally seen many Disney movies in m...	1	6150_7.txt

4. FILE DESCRIPTION

There are 2 top level directories train and test which corresponds to the training and test sets. Each of them has a separate directory for positive and negative which has labels as positive and negatives. Within These the reviews are stored in the following formats:

[test/pos/200_8.txt]
This tells us that it's a positive review which has rating 8/10 and belongs to test set with ID 200.

The directory train/unsup has 0 for all ratings because the rating is omitted for this portion of dataset. For Simplicity there's also a link for the reviews from the IMDB website as well.

We also have bag of words feature which can be found in .feat file in train and test directories which is an LIBSVM format file. The feature indices in these files start from 0, and the text tokens corresponding to a feature index is found in [imdb. vocab]. So, a line with 0:7 in a .feat file means the first word in [imdb. vocab] appears 7 times in that review.[7]

5. PROPOSED FRAMEWORK

The proposed framework for our model includes data cleaning, data pre-processing, applying algorithm's and then lastly comparing the results from the expected ones.

A. Data Pre-Processing

In order to improve the performance of our model we have done some operations on the data that we have collected. We have removed the unnecessary noise which will help to in classification of our model. It includes the following steps.

- **Removing HTML tags:** As the data is collected from IMDB reviews it consist of some of the html tags which are not required by our model as it doesn't have any impact in performance. The below figure 5.1 shows the code snippet for removing html tags and URLs that are available in the reviews.

Figure 5.1

```
#Data preprocessing
#Here we remove html tags, urls, special characters,Lemmanatize-
#which is better than stemmng as it gives a proper word after cutting
def rmvhtmltags(text):
    remreg = re.compile('<.*?>')
    cleartext = re.sub(remreg, '', text)
    return text

def remove_urls (vTEXT):
    vTEXT = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', vTEXT, flags=re.MULTILINE)
    return(vTEXT)
```

- **Lemmatization:** It's a process of converting the given word to it root or base format. The main objective of lemmatization is to get proper morphological meaning of words by referring to the dictionary which is incorporated in the library. We are making use of wordnet and porter stemmer lemmatization type. Wordnet is one of the mostly used lemmatizer. It consists of a public database of lexical words of English in it making it easy to establish semantic connections between words. While porter stemmer is known for its simplicity and speed of execution. In the figure 5.2 we have developed a function for lemmatization.

Figure 5.2

```
#Lemmantize
def lemmatize_words(text):
    lemmatized_words = [lemmatizer.lemmatize(word, 'v') for word in text.split()]
    return(' '.join(lemmatized_words))
```

- **Removing Stop words and special characters:** The data consist of stop words like "a", "I", "you", and". This word mostly appears a lot of time in reviews and are not important. The code snippet for stop words and special character is shown in figure 5.3

Figure 5.3

```
def rmvspclcharacter(text):
    clearspcl = re.sub(r'[^A-Za-z0-9\s.]', r'', str(text).lower())
    clearspcl = re.sub(r'\n', r' ', text)
    clearspcl = " ".join([word for word in text.split() if word not in stopWords])
    return text
```

- **Text Tokenization:** Partitioning text into sentences and words by addressing basic linguistic words, punctuations and numbers. In language like English words are separated by white spaces. Sop character are indication of sentence completion. But sometimes stop character and also be used as abbreviations by the user who types the sentence. To prevent such kind of problems sentence tokenization is used using NLTK. It basically consists of many languages like German, English, Spanish, French and many more languages trained with NLTK. This training comprises of identifying the punctuations and characters that the algorithm encounters at the end and beginning of sentences. In NLTK word tokenization is a wrapper function that utilizes tree bank tokenize and splits the punctuations other than periods.

So, after all the data pre-processing steps are completed and we do execute the function specified above we get the following output. Figure 5.4 shows the review which consist of all the html, special character, and how the data is before tokenization and stemming.

Figure 5.5 shows the output which is after all the data pre-processing is done on that particular review.

Figure 5.4

```
#before data preprocessing for training datasets
reviews_train['review'][7]
```

```
'In this "critically acclaimed psychological thriller based on true events, Gabriel (Robin Williams), a celebrated writer and late-night talk show host, becomes captivated by the harrowing story of a young listener and his adoptive mother (Toni Collette). When troubling questions arise about this boy\'s (story), however, Gabriel finds himself drawn into a widening mystery that hides a deadly secret\x85" according to film\'s official synopsis.<br /><br />You really should STOP reading these comments, and watch the film NOW...<br /><br />The "How did he lose his leg?" ending, with Ms. Collette planning her new life, should be chopped off, and sent to "deleted scenes" land. It\'s overkill. The true nature of her physical and mental ailments should be obvious, by the time Mr. Williams returns to New York. Possibly, her blindness could be in question - but a revelation could have be made certain in either the "highway" or "video tape" scenes. The film would benefit from a re-editing - how about a "director\'s cut"? <br /><br />Williams and Bobby Cannavale (as Jess) don\'t seem, initially, believable as a
```

Figure 5.5

```
#after data preprocessing for training datasets
reviews_train['review'][7]
```

```
'critically acclaim psychological thriller base true events gabriel robin williams celebrate writer late night talk host captivate harrow story young listener adoptive mother toni collette trouble question arise boy s story however gabriel find draw widen mystery hide deadly secret accord film s official synopsis br br you stop read comment watch film now br br the how lose leg end ms collette plan new life chop off send delete scenes land it s overkill true nature physical mental ailments obvious time mr williams return new york possibly blindness question revelation certain highway video tape scenes film benefit re edit director s cut br br williams bobby cannavale as jess don t seem initially believable couple scene establish relationship help set stage otherwise cast exemplary williams offer exceptionally strong characterization gay impersonation sandra oh as anna joe morton as ashe rory culkin pete logand perfect br br best all collette s donna belong creepy hall fame ms oh correct say collette be you know like guy psycho years organizations give act award reach women slighter dispersion roles certainly notice collette award consideration good and director patrick stettner definitely evok
```

The data pre-processing is done for testing datasets.

B. Features Extractions

Feature extraction is the process of converting a word into a matrix form. For doing that we used following approach:

- **Bag of Words Approach:** In this method a sentence is considered a bag and each word is called as gram. It is a method of extracting features from the data. After extracting the data, the features can be used in the algorithm. It works by taking every word and finds the word which is unique. Then for each word it finds the frequency of word appearing and marks it as 0 if absent and 1 if present which a vector. So now for creating a vector we have 2 ways one is count vectorizer and another is TF-IDF which stands for Term Frequency Inverse Document Frequency.
  - **TF-IDF:** It's an efficient approach that will help to determine how frequently a particular word appears in our document or dataset. We are using TF-IDF from sklearn library in python which allows us to utilize this function.

    TF – IDF (term) = TF (term in document) * IDF (term) where

    TF (term) = Number of times the term appears in document / total number of terms in the document.

    IDF (term) = Log (total number of documents/ numbers of document with term in it
  - **N-Grams Bi-Grams:** It's a task of finding a probability of a word occurring in a sentence given the probability of some word from previous history. As this method works good for larger data sets but sometimes it may also end up getting bad probabilities as content from the web is increasing and getting a very good probabilities is a difficult task. For N-grams approach we have made use of pipeline which is a tool in python which is used in feature extraction and evaluation. Bi-Gram uses the probability of the previous word that it knows to predict the probability of the current word.
  - **Count Vectorizer:** The process includes a blend of tokenizing a collection of documents from the datasets and then build a set of vocabulary for those words. The result of this is a length of vocabulary words and an integer value assigned for words according to how many times they appear. Words that do not occur may posses of zeros as value and are defined as sparse.

C. Word Cloud Representation

Word Cloud is done after the data pre-processing is done. So, all the noise interrupts like HTML tags, repeating words, special characters are removed and then word cloud is implemented. It's just a visualization technique which gets invoked by generate () function. We can show in the below figure 5.6 the word cloud for training datasets after data pre-processing.

Figure 5.6

D. Classification Model

In our experiment we have made used of Naïve Bayes, Logistic Regression, Support Vector Machine. We have trained our model on the above classifiers to predict the movie as positive or negative and this model are also examined on the test data to verify that the model performs in the similar way for the new dataset on I unknown.

- **Naïve Bayes:** Its one of the most popular algorithm of supervised machine learning. Its main objective is to find the probability of a class and predict the future objects when a bunch of objects are given in the datasets. This algorithm utilizes Bayes Theorem. In simpler terms we can say that the algorithm assumes that a feature from a particular class should not affect features present in the same class. This algorithm is easy to implement and is suitable for larger datasets. The formula for naïve Bayes is given below: [5]

  P(Z|Y) = P(Y|Z) P(Z) ÷ P(Y)

  The terms are explained below:

  Z and Y: This are two events

  P(Z): It's the probability of event Z happening.

  P(Y): It's the probability of event Y happening with condition (P(Y) ≠ 0)

  P(Y|Z): It's the probability of event Y happening given that event Z happens.

  P(Z|Y): It's the probability of event Z happening given that event Y happens.

We have implemented naïve Bayes in 2 versions one with count vectorizer, one with TF-IDF using Bi-Gram for both. The code snippet for those are attached below in figure 5.7 shows the naïve Bayes with TF-IDF where we get accuracy of 87.90% and with count Vectorizer we get accuracy of 87.36%. which is shown in figure 5.8

Figure 5.7

```
Best parameters:  {'clf_alpha': 1, 'tfidf_max_df': 0.4, 'tfidf_min_df': 2, 'tfidf_ngram_range': (1,
Accuracy score:  0.8790666666666667
Precision score:  0.8912510220768601
Recall score:  0.8655373213340392
F1 score:  0.8782059889888545
-------------------------
Confusion Matrix:  [[3323  399]
 [ 508 3270]]
```

Figure 5.8

```
Best parameters:  {'clf_alpha': 1, 'vect_max_df': 0.4, 'vect_min_df': 2, 'vect_ngram_range': (1, 2)}
Accuracy score:  0.8736
Precision score:  0.8838849701573521
Recall score:  0.8623610375860243
F1 score:  0.8729903536977492
-------------------------
Confusion Matrix:  [[3294  428]
 [ 520 3258]]
```

- **Logistic Regression:** Logistic regression is one of the mostly used statistical method for analysing the datasets which consist of one or more variables which are independent. It can be considered an improved version of linear regression. In lay mans terms we can say its just gives a probability of event to be occurring by fitting the data to a logit named function.

The sigmoid function is similar to that of linear regression. Sigmoid function is as below:[6]

$$Y = e^{(b0 + b1 * x)} / (1 + e^{(b0 + b1 * x)})$$

Here b0 is the bias and b1 is the coefficient for the value x and y. This algorithm is used by invoking the logisticRegression () from the sklearn library

$$Odds = e^{\beta 0 + \beta 1X1 + \beta 2X2 + ... + \beta nXn}$$ **applying odds** $\longrightarrow$

$$Log\ (Odds) = \beta 0 + \beta 1X1 + \beta 2X2 + ... + \beta nXn$$

So, this logit is log of odds and this odd is function of P which can be defined as:

$$Logit\ (P) = a + bX$$

In logistic Regression we have done 2 variations one Bi-Gram with Count Vectorizer and the other Bi-Gram with TF-IDF. Figure 5.9 shows the output with Count Vectorizer and Figure 5.10 includes TF-IDF. With Count Vectorizer we get an accuracy of 88.30 % while doing TF-IDF and Bi-Gram we get an accuracy of 89.33%.

Figure 5.9

```
Best parameters:  {'clf_C': 0.1, 'vect_max_df': 0.5, 'vect_min_df': 2, 'vect_ngram_range': (1, 2)}
Accuracy score:  0.8830666666666667
Precision score:  0.8838317015083356
Recall score:  0.884065643197459
F1 score:  0.883948656874421
-------------------------
Confusion Matrix:  [[3283  439]
 [ 438 3340]]
```

In the above we can see that logistic regression with tfidf(0.893) has greater accuracy than logistic regression with count vectorizer(0.883)

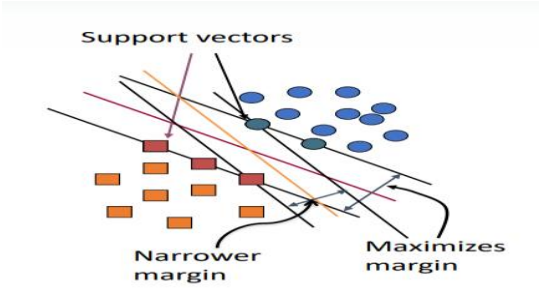
Figure 5.10

```
Best parameters:  {'clf_C': 10, 'tfidf_max_df': 0.25, 'tfidf_min_df': 2, 'tfidf_ngram_range': (1, 2)}
Accuracy score:  0.8933333333333333
Precision score:  0.8924617817606747
Recall score:  0.8962413975648491
F1 score:  0.8943475964078182
-------------------------
Confusion Matrix:  [[3314  408]
 [ 392 3386]]
```

- **Support Vector Machine:** This algorithm is used to solve regression problems. It utilizes a hyper plane to do classifications. It results is an optimal hyper plane which is generated when we give training data as input. Then later uses this hyperplane to perform classification on the dataset. SVM is known for its good performance. It's working is that it finds the distance between the two given observations which is then followed by search for a decision boundary in order to get distance between the closest members of separate class. SVM are robust in case of overfitting of the model. This algorithm is used by calling the LinearSVC() method of sklearn as its more flexible and better to scale large number of samples. [6].

We can consider the below figure in which there are blue circles and orange squares as our classes. Both this are separated by a hyperplane. So, for choosing the hyperplane we must select those points as our SVM which are equidistance from the hyperplane which we have taken. Doing so correctly we will be getting margin between the 2 chosen SVM maximum. The below figure 5.11 shows the proper example of SVM.

Figure 5. 11



In SVM also we have done the same thing that we have done in logistic regression one with TF-IDF where we get an accuracy score of 88.62% in figure 5.12 and one with Count Vectorizer where we get an accuracy score of 87.50 which is lesser than SVM Naïve Bayes with TF-IDF in figure 5.13

Figure 5.12

```
Best parameters:  {'clf__C': 10, 'tfidf__max_df': 0.5, 'tfidf__min_df': 2, 'tfidf__ngram_range': (1, 2)}
Accuracy score:  0.8862666666666666
Precision score:  0.8865979381443299
Recall score:  0.8877713075701429
F1 score:  0.8871842348895648
--------------------------
Confusion Matrix:  [[3293  429]
 [ 424 3354]]
```

Figure 5.13

```
Best parameters:  {'clf__C': 10, 'vect__max_df': 0.5, 'vect__min_df': 2, 'vect__ngram_range': (1, 2)}
Accuracy score:  0.8750666666666667
Precision score:  0.8772908366533865
Recall score:  0.8742721016410799
F1 score:  0.875778867824473
--------------------------
Confusion Matrix:  [[3260  462]
 [ 475 3303]]
```

- **Random Forest:** It's an algorithm based on decision tree. First some "n" numbers of trees are generated from which a single tree is formed. The Higher the number of trees we get the higher robustness in our model. The limitations of decision tree are overcome here in Random forest which is overfitting and also it reduces bias. The main motive behind random forest is to search for a pair of variable value in the training data and then split it in such a way that will yield two best child subnets. We are also using ensemble method with RF which get average outputs. So, it operates by taking weak learners to algorithm tries to use string learners. So, by doing we will get the average output of individual predictions by diversifying the predictors sets. Thereby resulting in lower variance. This algorithm can be used by calling the RandomForestClassifier() function from sklearn in python.

Here we consider Random forest with TF-IDF using Bi-Gram where we get an accuracy score of 86% in figure 5.14 and count vectorizer using Bi-Gram where we get an accuracy score of 86.28% which is slightly greater than Random Forest TF-IDF in figure 5.15.

Figure 5.14

```
Best parameters:  {'clf__max_depth': 100, 'clf__n_estimators': 200, 'tfidf__max_df': 0.5, 'tfidf__min_df': 2, 'tfidf__ngram_r
ange': (1, 2)}
Accuracy score:  0.86
Precision score:  0.8676549865229111
Recall score:  0.8520381154049762
F1 score:  0.859775641025641
--------------------------
Confusion Matrix:  [[3231  491]
 [ 559 3219]]
```

Figure 5.15

```
Best parameters:  {'clf__max_depth': 100, 'clf__n_estimators': 200, 'vect__max_df': 0.2, 'vect__min_df': 2, 'vect__ngram_rang
e': (1, 2)}
Accuracy score:  0.8628
Precision score:  0.861805738352198
Recall score:  0.8665960825833775
F1 score:  0.864194272139369 3
--------------------------
Confusion Matrix:  [[3197  525]
 [ 504 3274]]
```

E. Results Evaluation

The results of the models implemented are to be compared based on accuracy, precision, recall and f-1 score.

Accuracy: It is one of the commonly used performance measure parameters. It is simply the ratio of correctly predicted observations to the total number of observations. If we a get higher accuracy our model is good. Equation (1) shows the accuracy formula:

(1) Accuracy= TP + TN / TP + TN + FP + FN.

Precision: Precision is the ratio of correctly predicted positive observations divided by total predicted positive observations.

(2) Precision= TP / TP + FP

Recall: It's the ratio of correctly predicted positive observations to the total observations in that class.

(3) Recall = TP / TP + FN

F1-Score: It's a weighted average of precision and recall.

(4) F1-score= 2 *[(Recall * Precision) / (Recall + Precision)]

The analysis of accuracies of all the models is given below:

rf_tfidf = Random Forest with TF-IDF

rf_cv = Random Forest with Count Vectorizer

nb_tfidf = Naïve Bayes with TF-IDF

nb_cv = Naïve Bayes with Count Vectorizer

svm_tfidf = Support Vector Machine with TF-IDF

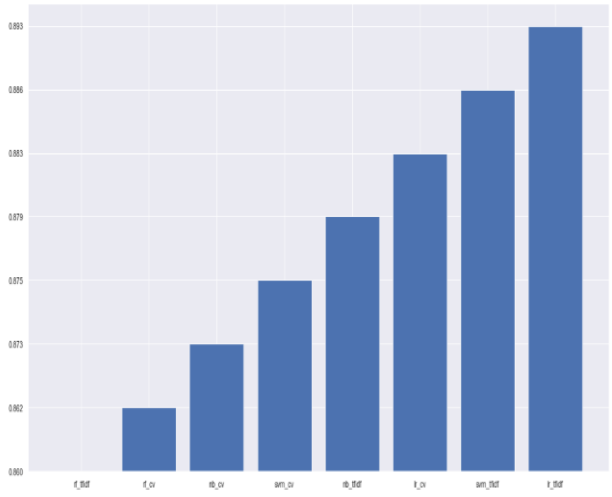
svm_cv = Support Vector Machine with Count Vectorizer

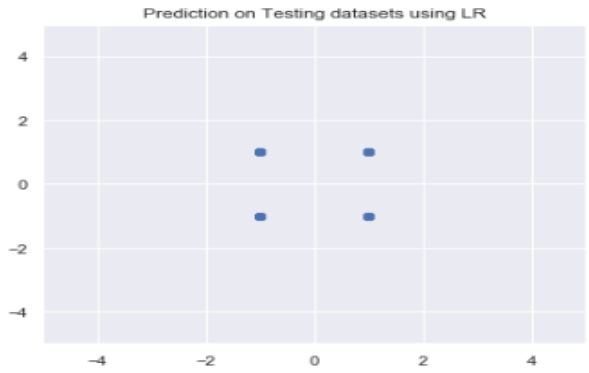
lr_tfidf = Logistic Regression with TF-IDF

lr_cv = Logistic Regression with Count Vectorizer

The above figure 5.16 displays the accuracies of all the models implemented.

Figure 5.16





6. CONCLUSION

The main motive behind this project was to construct a sentiment analysis model that will help us to get a better understanding of movie reviews that we have collected. Then later comparing the results on the 4 classifiers with two different version one with TF-IDF and other is count vectorizer. For Evaluation, we observed the accuracy, precision, recall, f-1 score and represented confusion matrix. for each model. By evaluating the models on TF-IDF and Count Vectorizer after hyperparameter tuning, we can observe that whenever we perform in SVM, Naïve Bayes and Logistic Regression models with TF-IDF it gives better accuracy as compared to Count Vectorizer for the similar model. But Random Forest with Count Vectorizer has a slightly higher accuracy compared to Random Forest with TF-IDF. From the observation we found out that Logistic Regression with TF-IDF has the highest accuracy score for the training model that is 89.33%.

We considered the best Logistic Regression with TF-IDF model to predict on test datasets and we got an accuracy score of 88.524%.

Figure 6.1

```
Accuracy score:  0.88524
Precision score:  0.8892571336189475
Recall score:  0.88008
F1 score:  0.8846447669977081
---------------------------
Confusion Matrix:  [[11130  1370]
 [ 1499 11001]]
```

Figure 6.2

7. REFERENCES

[1]. MaisYasen, Sara Tedmori. “Movies Reviews Sentiment Analysis and Classification”. IEEE Jordon International Joint Conference on Electrical Engineering and Information Technology (JEEIT). 978-1-5386-7942-5. “https://ieeexplore.ieee.org/document/8717422”.

[2]. Tirath Prasad Sahu, Sanjeev Ahuja. “Sentiment Analysis of movie reviews: A study on feature selection and classification algorithms”. International Conference on Microelectronics, Computing, and Communication (MicroCom).978-1-4673-6621-2. “https://ieeexplore.ieee.org/document/7522583”

[3]. Wijayanto**,** Unggul and Sarno, Ritanarto. “An Experimenal Study of Supervised Sentiment Analysis Using Gussai Naïve Bayes”. 476-481.10.1109/ISEMANTIC.2018.8549788.

“https://www.researchgate.net/publication/329322750_AnExperimental_Study_of_Supervised_Sentiment_Analysis_Using_Gaussian_Naive_Bayes” **.**

[4]. Tejaswini M. Untawale, G. Choudhari. “Implementation of Sentiment Classification of Movie Reviews by Supervised Machine Learning Approaches”. 978-1-5386-7808-4. “https://ieeexplore.ieee.org/document/8819800”.

[5]. Sourav Mehra, Tanupriya Choudhary. “Sentiment Analysis of User Entered Text”. International Confernce of Computational Techniques, Electronics and Mechanical Systems (CTEMS). 978-1-5386-7709-4.

“https://ieeexplore.ieee.org/document/8769136”.

[6]. Nisha Rathee, Nikita Joshi, Jaspreet Kaur. “Sentiment Analysis Using Machine Learning Techniques on Python”. 978-1-5386-2842-3 “https://ieeexplore.ieee.org/document/8663224”.

[7].“https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset”

[8]. https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624