MAJOR PROJECT II

A Report on

Sentiment Analysis On Amazon Product Dataset Using Machine Learning Algorithms

Submitted in the partial fulfillment of the requirements
for the Award of Degree
OF
Master of Technology
In
INFORMATION TECHNOLOGY

Submitted by:
KRISHAN KUMAR
2K21/ISY/13

Under the supervision of
Dr. RITU AGARWAL

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

June 2023

CANDIDATE'S DECLARATION

I, Krishan Kumar, 2K21/ISY/13 student of MTech (IT-ISY), hereby declare that the major project-1 report titled "Sentiment Analysis On Amazon Product Dataset Using Machine Learning Algorithms" which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Masterof Technology is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi Krishan Kumar

Date: 31/05/2023

CERTIFICATE

I hereby certify that the Project Dissertation titled "Sentiment Analysis On Amazon Product Dataset Using Machine Learning Algorithms" which is submitted by Krishan Kumar, 2K21/ISY/13 Information Technology, Delhi Technological University, Delhi in partial fulfilment of the requirement forthe award of the degree of Master of Technology, is a record of the project work carried out bythe students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this university or elsewhere.

Place: Delhi

Date: 31/05/2023

DR. RITU AGARWAL

SUPERVISOR

888

ACKNOWLEDGMENT

I would like to express my gratitude to my guide to my guide Dr. Ritu Agarwal without whose valuable inputs and guidance this report wouldn't come to fruition. I would also like tothank my parents for their constant support throughout the process. Also, I'd like to thank myclassmates who kept my spirits up when I was down and helped me whenever I was stuck somewhere.

ABSTRACT

In this report, we will discuss "Sentiment Analysis on Amazon Product Dataset Using Machine Learning Algorithms." Due to the substantial influence that opinions have on people's behaviors and decision-making processes, sentiment analysis has extensive applications across many economic and social areas. When looking for information or making decisions, people and organizations frequently rely on the opinions of others.

The sentiment analysis of a dataset made up of Amazon reviews and ratings is the main topic of this research. To improve the analysis, various pre-processing strategies and vectorization approaches will be used. Machine learning algorithms will be used for classification, and the outcomes of each approach will be compared. Deep learning techniques will also be used to further assess and compare the results of the sentiment analysis.

TABLE OF CONTENTS

888

888

List of Figures

CHAPTER-1

INTRODUCTION

SENTIMENT ANALYSIS

Human behavior is heavily influenced by opinions, which have an impact on our beliefs, perceptions, and decision-making processes. When making decisions, both as people and as organizations, we frequently look to the perspectives of others. The fields of sentiment analysis and opinion mining have emerged as a result of this understanding of the importance of views.

With the growth of social media platforms like reviews, forums, blogs, microblogs (like Twitter), and social networks, the discipline of sentiment analysis acquired pace. These sites have made available an unheard-of amount of opinionated data that has been digitally recorded. The study of sentiment analysis has grown significantly in the fields of natural language processing, data mining, web mining, and text mining since the early 2000s.[4]

User-generated material, including comments regarding goods, services, and public policies, has significantly increased as a result of the growth of social media platforms like Twitter, Facebook, and Trip Advisor. With its large user base of 336 million active users per month, Twitter has established itself as a key platform for feedback from governing bodies, businesses, and service providers. Each day, around 500 million tweets are created on Twitter alone, producing a huge amount of unstructured text data [1].

The rapid digitization in the last decade has caused people to use the internet for completing all the basic tasks that they used to do manually. Since e-commerce websites display the products using images, it is hard to figure out the quality of the product's quality. As a result of this, there is a feature on internet that allows the user to write a review and rate a product. Rating of the product help us to understand the sentiment of people whether they are liking the product or not and review writing gives us the idea of positives and negatives of the product. Due to increase in usage of e-commerce number of sellers selling their product has increased asking customer to review their product. Large

number of reviews are available for a most used product. Because of this, it is difficult for an individual to read them and decide whether to purchase the product or not. To handle this sentiment analysis is used. Using sentiment analysis one can easily understand the emotion of the content.

Reviews by user for product evaluations on websites like Amazon, has a big impact on what people decide to buy. The recommendations and experiences given by other shoppers are a major source of motivation for consumers. Therefore, it is essential to create systematic techniques for understanding the important knowledge included inside user-generated material, attitude analysis, which entails detecting whether a review reflects a positive or negative attitude, is a common technique for drawing insights from text evaluations of products written by customers. However, human monitoring and sentiment extraction are made difficult by the enormous amount of user-generated content repositories and their ongoing expansion. As a result, automatic text categorization has emerged as a useful and effective technique for effective data analysis and understanding.[2]

Financial market players continually watch financial and economic news. The efficient market theory states that stock prices represent all historical knowledge, and that new information is instantly included when establishing future stock values. Therefore, quick extraction of positive or negative feelings from news is crucial for traders, portfolio managers, and investors making investment decisions. Unactionable signals from the news may be efficiently extracted using sentiment analysis methods. However, the absence of big labelled datasets and the use of terminology peculiar to the financial industry make financial sentiment research difficult[3].

Sentiment analysis may be used in a wide range of businesses and is an effective technique for deciphering online discussions and public opinion. Businesses use sentiment analysis to do market research and learn more about client preferences for particular goods or services. Organizations can ascertain whether their clients have a favorable or unfavorable sentiment towards their offerings by analyzing sentiment. The business can then make the required adjustments to improve its goods or services based on the sentiment analysis results, leading to improved results. This data's analysis has the power to completely alter how we do our jobs. Nevertheless, it might be difficult to draw

useful conclusions from this abundance of data.[7].

So, Recent developments have increased the importance of sentiment analysis in the domains of text mining and natural language processing (NLP).[6]

Preprocessing of text data, which entails converting web-based unstructured data into a format appropriate for classification, is the first step in the sentiment classification process. Tokenization, stop word removal, text conversion to lowercase, stemming, and number removal are examples of preprocessing tasks. The procedure of extracting different kinds of text characteristics from the preprocessed data is the following phase. Count vectors, bag of words, TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings, and NLP-based approaches are examples of common feature extraction techniques.

The process of selecting the most pertinent characteristics for categorization comes next, after feature extraction. Common methods used in this selection process include mutual information, information gain, chi-square, and Gini index.[1]

The chosen features are then subjected to machine learning techniques for sentiment categorization. Decision trees, naive Bayes, logistic regression, xgboot, random forest etc and artificial neural networks are examples of popular algorithms. These algorithms categorise fresh text input into categories of positive, negative, or neutral sentiment as they learn from the chosen characteristics.[1], Alternative expressions for the output value include very good, good, satisfactory, bad, or extremely bad on an n-point scale.[10]

OBJECTIVE

Safeguarding the company's financial well-being involves enhancing customer satisfaction and placing significant value on their feedback. By prioritizing customer contentment and actively valuing their input, the company can mitigate the risk of financial losses.

This entails striving to meet customer expectations, addressing their concerns promptly, and continuously improving products, services, and overall customer experience. What benefits will these solutions bring to the company? By prioritizing customer satisfaction and valuing their feedback, companies can safeguard themselves from financial losses.

When customers are satisfied with their overall experience, they are more likely to continue purchasing products or services from the company. This leads to increased customer loyalty and higher retention rates, which, in turn, positively impact revenue generation. Actively seeking and considering customer feedback allows companies to identify areas for improvement and address potential issues promptly. By addressing customer concerns, companies can enhance their offerings, refine their strategies, and provide a more tailored and satisfactory experience to their customers.

This proactive approach helps prevent customer dissatisfaction and potential negative word-of-mouth, mitigating the risk of losing customers and revenue. 5 Additionally, giving importance to customer feedback helps companies stay in tune with evolving customer preferences, market trends, and industry dynamics. This enables them to adapt their products, services, and business strategies accordingly, ensuring their offerings remain relevant and competitive imarketplace.

By continuously meeting and exceeding customer expectations, companies establish a positive brand reputation, attracting new customers and fostering long-term business growth.

Moreover, valuing customer feedback fosters a sense of customer-centricity within the organization. When employees recognize the importance of customer satisfaction and understand the impact of their actions on the overall customer

experience, they become more attentive, responsive, and committed to delivering exceptional service.

This customer-centric culture permeates the entire company, leading to improved customer interactions, higher customer satisfaction levels, and ultimately, a more profitable and sustainable business.

CHAPTER 2

IMPLEMENTATION

In this project I am doing sentiment analysis of amazon dataset. So, I am using Amazon dataset (beauty-products) reviews for sentiment analysis and for classification purpose I'm using machine learning algorithms with different vectorization techniques, and then I am using some optimization techniques to improve the accuracy of these models and at last I will compare the accuracy of all algorithms.

DATASET USED

The data is obtained from github.io page of UC San Diego Computer Science and Engineering Department academic staff .Amazon beauty products reviews dataset is used with following attributes – asin, helpful, overall, reviewText, reviewTime, reviewerID, reviewerName, summary, unixReviewTime. Dataset have 198502 rows , 9 columns. Dataset is in the form of json object format.

```
reviews.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A1YJEY40YUW4SE	7806397051	Andrea	[3, 4]	Very oily and creamy. Not at all what I expect...	1	Don't waste your money	1391040000	01 30, 2014
1	A60XNB876KYML	7806397051	Jessica H.	[1, 1]	This palette was a decent price and I was look...	3	OK Palette!	1397779200	04 18, 2014
2	A3G6XNM240RMWA	7806397051	Karen	[0, 1]	The texture of this concealer pallet is fantas...	4	great quality	1378425600	09 6, 2013
3	A1PQFP6SAJ6D80	7806397051	Norah	[2, 2]	I really can't tell what exactly this thing is...	2	Do not work on my face	1386460800	12 8, 2013
4	A38FVHZTNQ271F	7806397051	Nova Amor	[0, 0]	It was a little smaller than I expected, but t...	3	It's okay.	1382140800	10 19, 2013

METHODOLOGY

The study employed a dataset of Amazon beauty product reviews to concentrate on sentiment analysis. The dataset underwent pre-processing methods, which included punctuation removal, null value handling, and data cleaning. The important information from the text was extracted using a variety of feature extraction techniques once the dataset had been produced, including CountVectorizer, TF-IDF vectorizer, n-grams (bi-tri), word-2-vec, and fastText.

A variety of machine learning classifiers, including Random Forest, Logistic Regression, Decision Tree, XtraTree classifier, XG Boost classifier, and

LightGBM, were used during the classification step. Each classifier underwent training and evaluation to see how well it handled sentiment analysis. To determine the most efficient strategy, the models' accuracy was evaluated.

We used additional methods to optimize the models. The classifiers' performance was enhanced by fine-tuning their hyperparameters with Grid Search. The prediction ability of the models was increased by using ensemble learning techniques. Weight balancing and SGD classifier techniques were used to overcome class imbalance concerns and boost overall accuracy.

For sentiment analysis, a Deep Learning Transformer Model dubbed BERT was used in addition to conventional machine learning techniques. BERT is a cutting-edge model that excels at efficiently capturing contextual data. To determine BERT's eligibility for sentiment analysis tasks, its performance was compared to that of the other models.

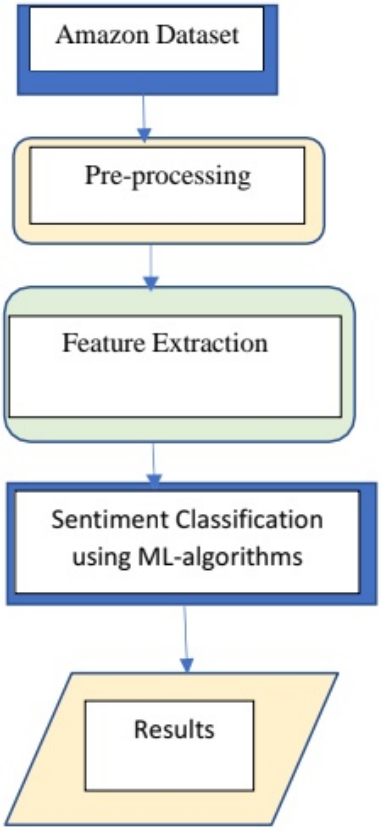

Fig 1. Flowchart

DATA PRE-PROCESSING

VISUALIZATION

Dataset have ratings from 1 to 5 and here below shows the total number of reviews belongs to each ratings.

```
[5]  reviews['overall'].value_counts()

5    114531
4     39741
3     22248
2     11456
1     10526
Name: overall, dtype: int64
```

```
plt.figure(figsize=(10,8))
ax = sns.countplot(x='overall', data=reviews)
```

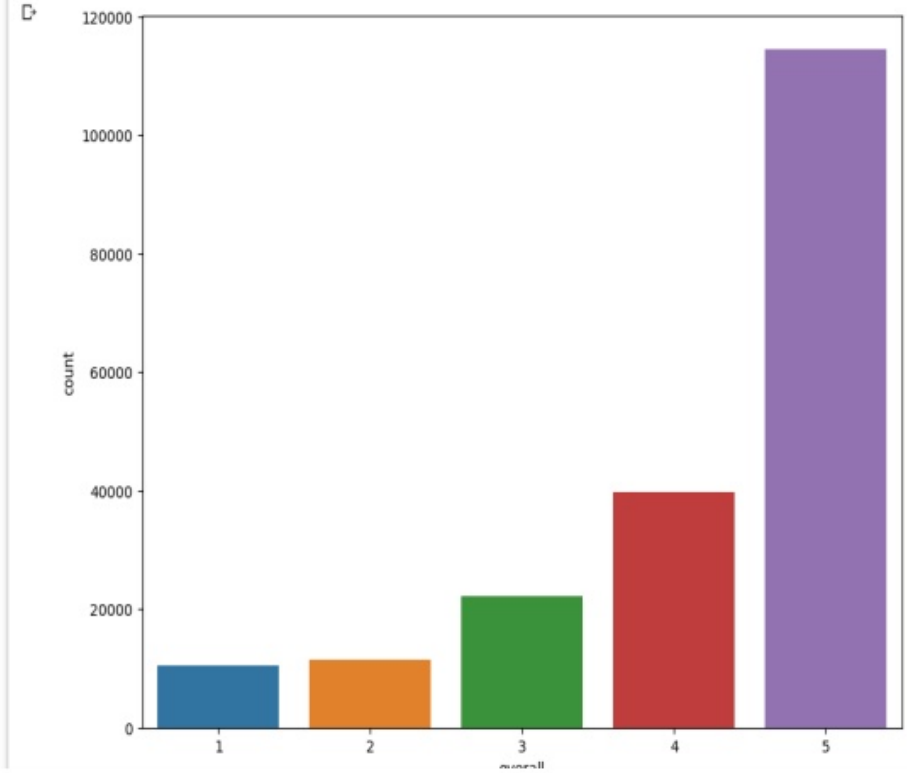

Fig 2. Visualization Of Data

Removing duplicate values-

```
reviews.drop_duplicates(subset=['reviewerID', 'asin','unixReviewTime'],inplace=True)
```

Removing null values from important parameters-

```
reviews.dropna(subset=['reviewText'], inplace=True)
```

Removing Punctuations-

```
import string
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
def remove_punctuation(text):
    text_no_punc = "".join([c for c in text if c not in string.punctuation])
    return text_no_punc
```

```
reviews['review_clean'] = reviews['reviewText'].apply(lambda x: remove_punctuation(x))
reviews.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime	review_clean
0	A1YJEY40YUW4SE	7806397051	Andrea	[3, 4]	Very oily and creamy. Not at all what I expect...	1	Don't waste your money	1391040000	01 30, 2014	Very oily and creamy Not at all what I expecte...
1	A60XNB876KYML	7806397051	Jessica H.	[1, 1]	This palette was a decent price and I was look...	3	OK Palette!	1397779200	04 18, 2014	This palette was a decent price and I was look...
2	A3G6XNM240RMWA	7806397051	Karen	[0, 1]	The texture of this concealer pallet is fantas...	4	great quality	1378425600	09 6, 2013	The texture of this concealer pallet is fantas...
3	A1PQFP5SAJ6D80	7806397051	Norah	[2, 2]	I really can't tell what exactly this thing is...	2	Do not work on my face	1386460800	12 8, 2013	I really cant tell what exactly this thing is ...
4	A38FVHZTNQ271F	7806397051	Nova Amor	[0, 0]	It was a little smaller than I expected, but t...	3	It's okay.	1382140800	10 19, 2013	It was a little smaller than I expected but th...

Now I have divided dataset in different classes.

firstly I have divide the reviews which have ratings(overall) 1,2 as bad review(0) , 3 as neutral review(1), 4,5 as good review(2).

```
df =  reviews
X = df['reviewText']
y_dict = {1:0, 2:0, 3:1, 4:2, 5:2}
y = df['overall'].map(y_dict)
```

In second category I have divided the reviews which have rating(overall) 1,2 as bad(0)3,4,5 as good(1).

```
df =  reviews
X = df['reviewText']
y_dict = {1:0, 2:0,3:1, 4:1, 5:1}
y = df['overall'].map(y_dict)
```

In third category I have divided the reviews which have rating(overall) 1,2 as bad (0)

,4,5 as good(1) and I have dropped the reviews which have rated 3.

```
df =  reviews[reviews['overall'] != 3]
X = df['reviewText']
y_dict = {1:0, 2:0, 4:1, 5:1}
y = df['overall'].map(y_dict)
```

In fourth category I have divided the reviews which have rating(overall) 1 as very bad(0) ,2 as bad(1), 3 as neutral(2) , 4 as good(3) and 5 as very good(4).

```
df =  reviews
X = df['reviewText']
y_dict = {1:0, 2:1, 3:2, 4:3, 5:4}
y = df['overall'].map(y_dict)
```

I have done train and test split of dataset in the ratio of 80:20.

So, these are some pre-processing step , after pre- processing we use ML-based classifier forclassification.

CLASSIFIERS USED

1. LOGISTIC REGRESSION

Logistic regression stands out as a prominent Machine Learning algorithm within the framework of Supervised Learning. This algorithm serves as a powerful tool for predicting categorical dependent variables based on a given set of independent variables..

A single multinomial logistic regression model with a single estimator is used in the classification technique. It establishes the line dividing the classes and then assigns probability based on how far away the line is. Due to its ability to be fitted in many ways and ability to produce precise predictions, logistic regression is more than just a classifier. These confident forecasts might not always come true, though. For prediction purposes, applied statistics and discrete data analysis frequently employ logistic regression.[11]

It can output 0 or 1, i.e. a binary value[9]. Furthermore, logistic regression requires no linear relationship between inputs and outputs, unlike linear regression.

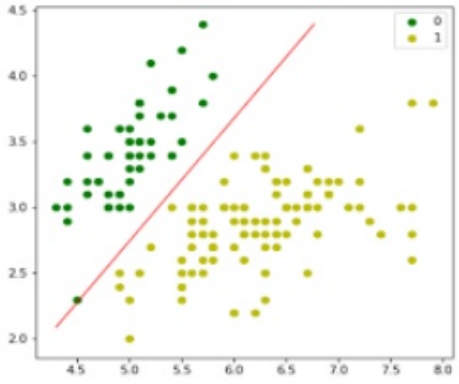

Fig.3 Logistic Regression

```
tfidf = TfidfVectorizer(stop_words = 'english')
text_fit(X, y, tfidf, LogisticRegression())
```

```
# features: 72694
# train records: 132190
# test records: 44064
Model Accuracy: 0.9281726579520697
```

2. DECISION TREE

One of the most effective methods for categorization and prediction, decision trees have grown significantly in popularity. These tree-like structures follow a syntax similar to a flowchart, with internal nodes denoting attribute tests, branching denoting test results, and leaf nodes denoting associated class names. The decision is performed using on the basis of attributes of the given dataset. There are three main algorithms used to build a decision tree which are: CART, ID3 and C4.5. Among these three algorithms CART is most commonly used.

Decision Trees (DT) are classification models that classify instances according to the values of their features using a hierarchical structure. The tree assigns each instance to a certain class by beginning at the root node and branching out based on several feature values. Because it enables the mapping of observations to goal values, decision tree learning is a widely used approach in data mining and machine learning. By assessing the tree's performance and removing pointless nodes, post-pruning approaches can further improve decision tree classifiers. This makes the model easier to understand and increases its accuracy.[11]

The internal nodes denote a test on an attribute, the branches represent the outcomes of the test, and the leaf nodes represent the attributes of the classes.

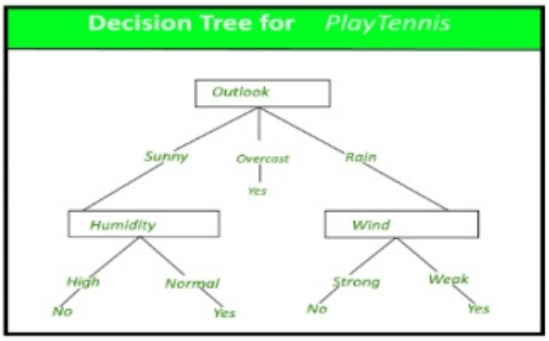

Fig 4. Decision Tree for playing tennis

3. EXTRA TREE CLASSIFIER

Extremely Randomized Trees Classifier (Extra Trees Classifier)-

In essence, it is an ensemble learning technique that synthesizes a classification from the outputs of various de-correlated decision trees gathered in a "forest" of trees. The sole distinction between the algorithm and Random Forest Classifiers is the methodology used to build the decision trees in the forest. In place of looking for the best split between two groups of samples, randomly selected features are split into splits and the best split is chosen from each[12].

Using the original training data, the Extra Trees Forest constructs each decision tree. At each test node in a decision tree, a random sample of k features is presented, and the tree must select the optimal feature from this subset based on a mathematical criterion, commonly the Gini Index. This process of sampling features results in the creation of multiple decision trees that are decorrelated from each other

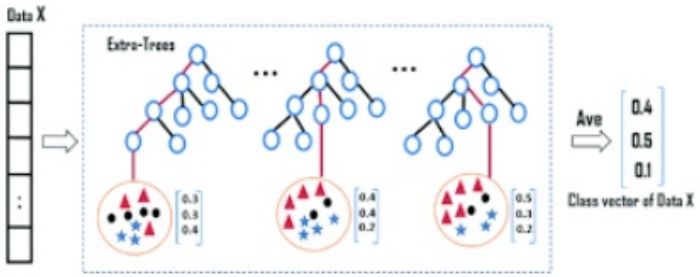

Fig 5 ExtraTree Claissifier

```
rf_extra = ExtraTreesClassifier(max_depth=5, criterion= 'entropy', min_samples_leaf=3, min_samples_split=18,
                        random_state=42, n_estimators = 100, class_weight='balanced', n_jobs = -1)
text_fit(X, y, tfidf_n,rf_extra)
```

```
# features: 2383953
# train records: 148876
# test records: 49626
Model Accuracy: 0.6496191512513602
```

4. RANDOM FOREST CLASSFIER

As part of an ensemble learning approach known as the Random Forest Classifier, predictions are made based on the mode of the class outputs after several decision trees have been built during training. Random Forest Classifier, as opposed to individual decision trees, builds a large number of trees to increase the predicted accuracy and resilience overall.[12]

- • The supervised learning method known as random forest can be applied to both classification and regression problems.
- • Random forest classifiers are a type of ensemble learning technique that we may use to address complex issues and enhance the performance of our models.

  • In random forest classifier we take various subsets of a dataset and make different decision trees on these subsets. Based on majority votes from different trees random forest predicts thefinal output.

  • It also improves the predictive accuracy of the decision tree classifier.

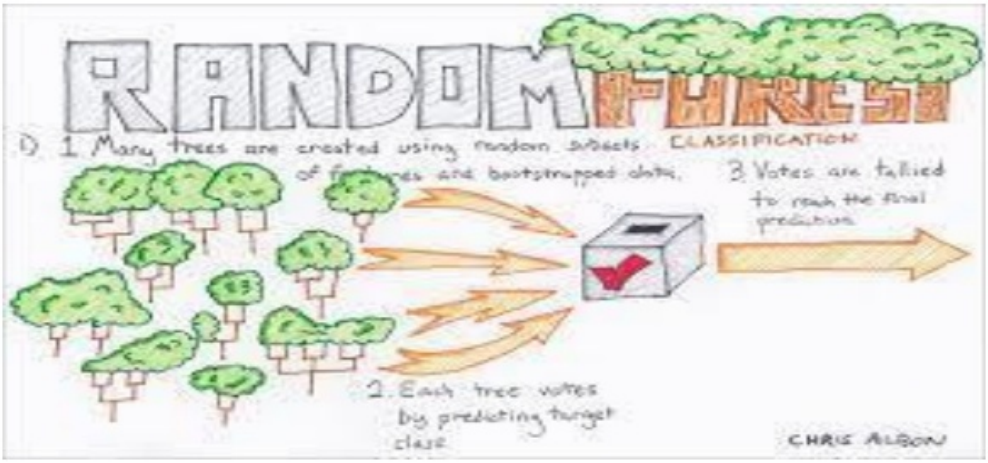

Fig 6. Random Forest Classifier

```
rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1,class_weight='balanced',
                            criterion = 'entropy',max_features = 'sqrt',min_samples_split = 5)
text_fit(X, y, tfidf_n,rf)
```

```
# features: 2383953
# train records: 148876
# test records: 49626
Model Accuracy: 0.7959537339297948
```

5. XG BOOST

The widely used gradient boosting framework XGBoost is now accessible in well-known data analysis languages. It has strong skills for both classification and regression problems.[14]

- It stands for Extreme Gradient Boosting classifier.
- It is based on ensemble learning algorithm, it uses a gradient boosting framework.
- It uses parallel processing, tree pruning and handles the missing values too.

```
xgb = XGBClassifier(objective = 'multi:softmax', booster = 'gbtree', nrounds = 'min.error.idx',
                    num_class = 3, maximize = False, eval_metric = 'merror',early_stopping_rounds=10,
                      eta = .1,max_depth = 12, colsample_bytree = .4, learning_rate = 0.1,
                      max_delta_step=1)
text_fit(X, y, tfidf_n, xgb)
```

```
# features: 2383953
# train records: 148876
# test records: 49626
Model Accuracy: 0.8145327046306372
```

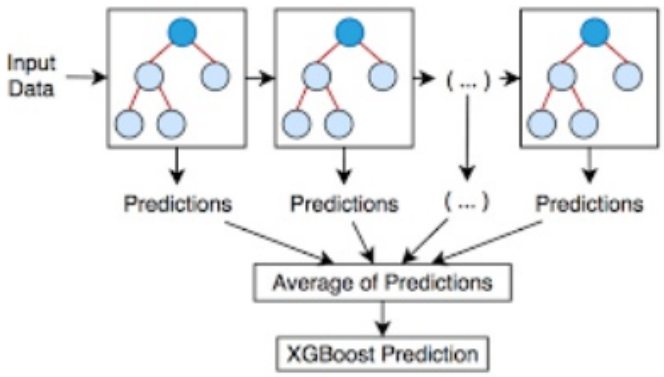

Fig 7. XGBoost

6. LIGHT BGM

The LGBM method, which refers to Light Gradient Boosting Machine, is built on two important concepts: GOSS (Gradient-based One-Sided Sampling) and GBDT (Gradient Boosting Decision Tree). In several research investigations, these approaches are generally used for tasks requiring prediction.[15]

It was produced by Microsoft and is open source and free. It is based on decision tree techniques and is used for classification, ranking, and other machine learning applications. The principal development priorities are performance and scalability.

```
lgbm = LGBMClassifier(booster = 'gbtree', nrounds = 'min.error.idx', maximize = False,eta = .1,max_depth = 10,
                      colsample_bytree = .4,learning_rate = 0.1,max_delta_step=1)
text_fit(X, y, tfidf_n,lgbm)
```

```
# features: 2383953
# train records: 148876
# test records: 49626
Model Accuracy: 0.8117720549711844
```

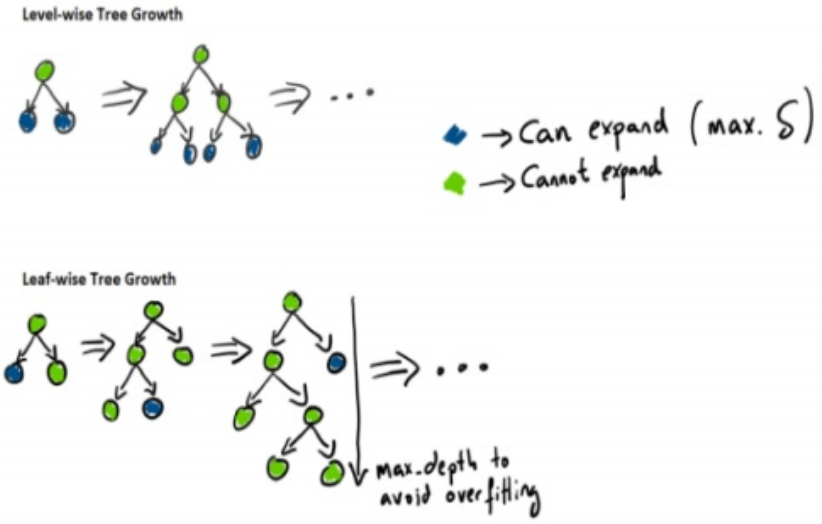

Fig 8. Light BGM

7. BERT-

BERT (Bidirectional Encoder Representations from Transformers), a revolutionary language representation model introduced by Devlin in 2018, leveraged the transformer architecture to bring about a significant advancement in natural language processing (NLP). BERT's state-of-the-art performance on various NLP tasks marked a pivotal moment in the field. Two innovative pre-training objectives, were used to achieve this. BERT overcame the drawbacks of earlier language models that only captured unidirectional word representations by combining both goals. Instead, BERT used a bidirectional masked language model that predicted randomly masked words inside sentences and so was able to capture more contextual information.[3]

BERT encompasses two variations: BERT-base and BERT-large. The BERT-base model consists of 110 million parameters, with 12 encoder layers, a hidden size of 768, and 12 multi-head attention heads. On the other hand, the BERT-large model comprises 340 million parameters, 24 encoder layers, 16 multi-head attention heads, and a hidden size of 1024. Both models were trained using the English Wikipedia and BookCorpus datasets. These extensive training efforts contributed to the remarkable language representation capabilities exhibited by BERT in various natural language processing tasks.[3].

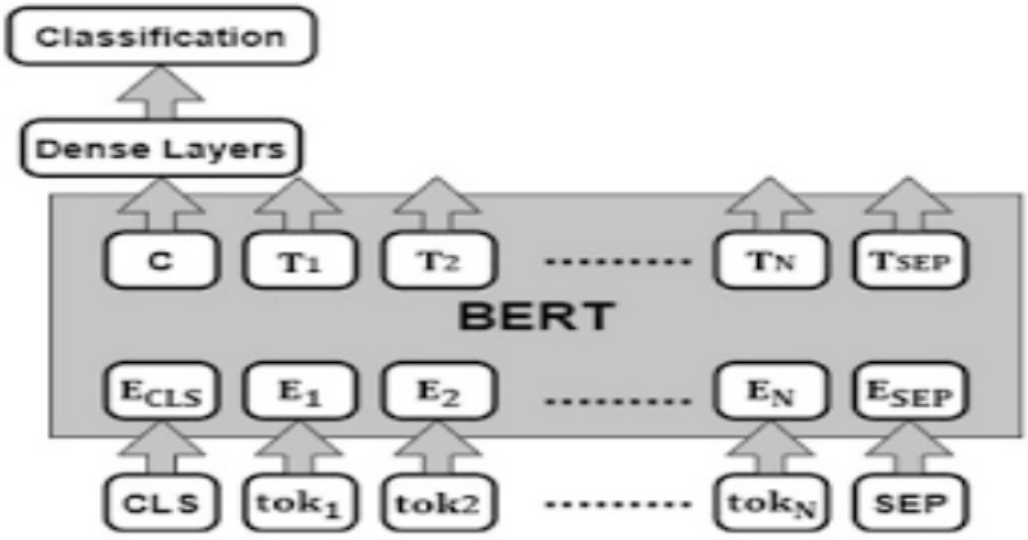

Fig 9. Bert

So, I have used these classifier for classification with different vectorizers i-e countVectorizer,TF-IDF, n-gram, Word-2-vec, FastText

VECTORIZERS

1. CountVectorizer

By simply counting the number of times each word appears in the full text, a python programme offered by the scikit-learn module can transform text to vectors. This method is useful when we need to convert each word in a corpus into a different vector.

In this vectorization technique we built a vector for each word by countingthat word in each document and just writing its frequency i-e let us suppose we have two documents and a word “life”. It occurs 4 times in first document and 3 times in second then thevector for this word can simply be return as [4 3].

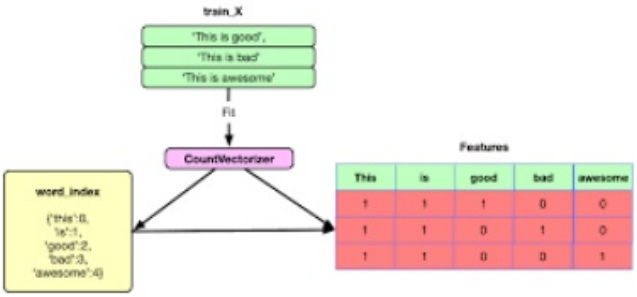

Fig 10. CountVectorizer

2. TF-IDF

In modelling texts, the well-known TF-IDF (Term Frequency - Inverse Document Frequency) measure is frequently utilised.[8] .By dividing the term's occurrences inside a text by the total number of words within, one may get the term frequency (TF) of a given term (t). This aids in determining the term's relative usage frequency across the text.

The relevance of a word in the entire corpus is instead determined by its inverse document frequency (IDF), which is calculated as IDF(t) = log(N/DF), where N denotes the total number of documents in the corpus and DF is the number of documents that include the phrase t. IDF gives phrases that appear in fewer documents a larger weight by calculating the logarithm of the inverse ratio,

showing their relative relevance.

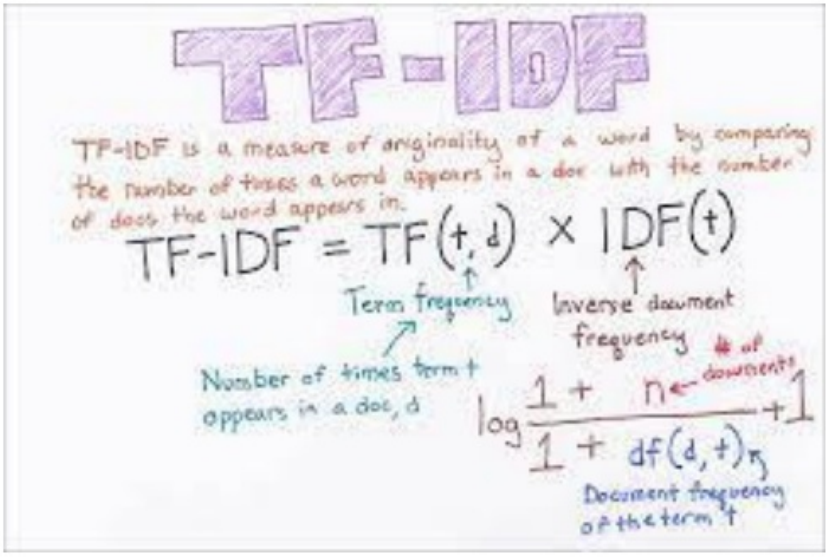

Fig 11. TF-IDF

3. N-gram

It can be used with combination of countVectorizer or with TF-IDF to get better results in this we defined a range from 1 to n then it will be consider as unigram,bi-gram…..n-gram. It simply takes a parameter n_gram in which we can give a range. For example if we are giving a range (1,2), then it will consider unigrams and bigrams. Suppose we have a sentence "No one is here". Then it will take unigrams as: "No", "one", "is", "here" and bigrams as "No one", "one is", "is here".

```
tfidf_n = TfidfVectorizer(ngram_range=(1,2),stop_words = 'english')
```

In our code we are using n-gram with TF-IDF vectorizer.

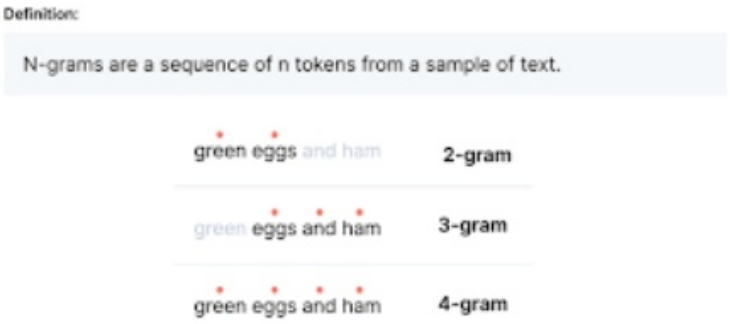

Fig 12 n-gram

4. Word2Vec

In 2013, Tomas Mikolov created the Word2Vec algorithm, a word embedding method that converts textual words into vectors that capture their semantic meaning. It is a neural network-based unsupervised learning strategy.

Word2Vec is made up of two layers: a projection layer, which is hidden, and a fully connected layer, which was trained using stochastic gradient descent and backpropagation. The mapping of words in the context of N-grams into continuous vectors is done by the projection layer. There is a link between words that regularly occur together in an N-gram context because their weights tend to be similar.[16]

Continuous Bag-of-Words (CBOW) and Skip-Gram are two architectural models offered by the word embedding method Word2Vec.In the CBOW model, Word2Vec predicts the target word by using the words that come before and after it within a constrained frame. In contrast, the Skip-Gram model, limited by the window size, uses a word to forecast the words that come before and after it. For gathering input and target words, the window functions as the kernel. It moves from the start of the text to the finish.

```python
# train word2vec model on the text data
sentences = [review.split() for review in X]
model = Word2Vec(sentences, min_count=1)

# extract features using word2vec
X_vectors = []
for sentence in sentences:
    vector = []
    for word in sentence:
        if word in model.wv:
            vector.append(model.wv[word])
    if vector:
        X_vectors.append(sum(vector) / len(vector))
    else:
        X_vectors.append([0] * model.vector_size)

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.2, random_state=42)

# train logistic regression model on the extracted features
clf = LogisticRegression(max_iter=1300)
clf.fit(X_train, y_train)
acc = clf.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.8002317321981814
```

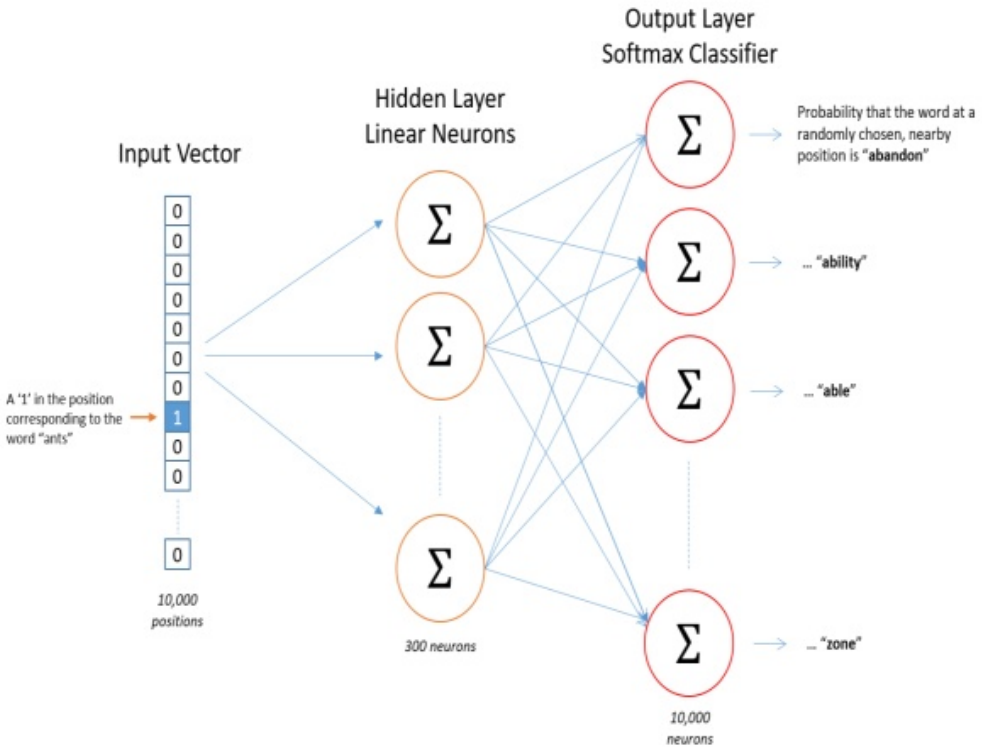

Fig 13. Word2Vec

5. FastText

FastText is a word embedding method that builds on the word2vec strategy. It was created by Facebook's AI research team and was influenced by Mikolov et al. FastText showed that it can train quickly on huge datasets with billions of words, outperforming other models in terms of performance by a wide margin.

The Continuous Bag-of-Words (CBOW) architecture of Word2Vec and the FastText model's design are similar. The representation of words as dense vectors and the introduction of a hierarchical structural contrast. Between the input layer and the output layer, FastText uses a hidden layer.
FastText enhances the word2vec framework by adding character n-grams, subword representations, and a hierarchical design, which results in effective training and better performance.[16]

```
# Train FastText model
sentences = [review.split() for review in X]
model = FastText(sentences, window=5, min_count=5, workers=4)

# Transform text into vectors using FastText model
X_vectors = []
for sentence in sentences:
    vector = []
    for word in sentence:
        if word in model.wv:
            vector.append(model.wv[word])
    if vector:
        X_vectors.append(sum(vector) / len(vector))
    else:
        X_vectors.append([0] * model.vector_size)

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.2, random_state=42)

# train logistic regression model on the extracted features
clf = LogisticRegression(max_iter=1300)
clf.fit(X_train, y_train)
acc = clf.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.7983174227349437
```

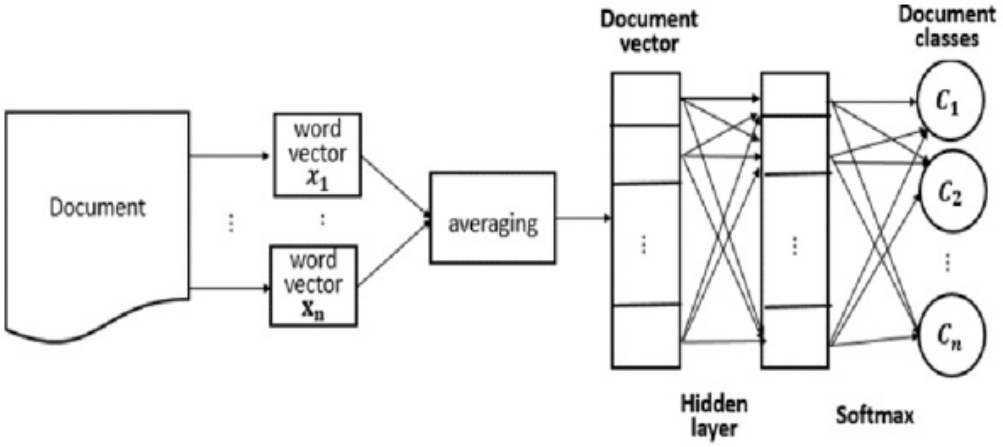

Fig 14. FastText

CHAPTER 3

RESULTS

1. For dataset, when it is divided in 3-classes, bad(1,2 rating(overall)), neutral(3rating(overall), good(4,5 rating(overall)).

	Model	features	accuracy
0	Logreg+c	77708	0.821303
1	Logreg+tfidf	77708	0.832064
2	Logreg+tfidf+n-gram	2383953	0.835409
3	Random Forest +tfidf+n-gram	2383953	0.795954
4	XGBM+tfidf+n-gram	2383953	0.814533
5	LGBM+tfidf+n-gram	2383953	0.811772
6	DecisonTree+tfidf+n-gram	2383953	0.443155
7	ExtraTreeClassifier+tfidf+n-gram	2383953	0.649619

2. For dataset, when it is divided in 2-classes, bad (1,2 rating(overall)), good (3,4,5rating(overall)).

	Model	features	accuracy
0	Logreg+c	77708	0.912727
1	Logreg+tfidf	77708	0.917684
2	Logreg+tfidf+n-gram	2383953	0.913150
3	Random Forest +tfidf+n-gram	2383953	0.899891
4	XGBM+tfidf+n-gram	2383953	0.907750
5	LGBM+tfidf+n-gram	2383953	0.904889
6	DecisonTree+tfidf+n-gram	2383953	0.455064
7	ExtraTreeClassifier+tfidf+n-gram	2383953	0.778140

3. For dataset, when it is divided in 2-classes, bad (1,2 rating(overall)), good (4,5rating(overall)) 1-e- rating 3- reviews are dropped.

	Model	features	accuracy
0	Logreg+c	72694	0.923952
1	Logreg+tfidf	72694	0.928173
2	Logreg+tfidf+n-gram	2177780	0.922204
3	Random Forest +tfidf+n-gram	2177780	0.898125
4	XGBM+tfidf+n-gram	2177780	0.912695
5	LGBM+tfidf+n-gram	2177780	0.907634
6	DecisonTree+tfidf+n-gram	2177780	0.481255
7	ExtraTreeClassifier+tfidf+n-gram	2177780	0.595611

4. For dataset, when it is divided in 5-classes,very bad (1 rating (overall)), bad(2 rating (overall)), neutral(3 rating(overall)),good (4 rating (overall)), very good(5 rating(overall)).

	Model	features	accuracy
0	Logreg+c	77708	0.644420
1	Logreg+tfidf	77708	0.652904
2	Logreg+tfidf+n-gram	2383953	0.661931
3	XGBM+tfidf+n-gram	2383953	0.633861
4	LGBM+tfidf+n-gram	2383953	0.631222
5	DecisonTree+tfidf+n-gram	2383953	0.305989
6	ExtraTreeClassifier+tfidf+n-gram	2383953	0.303530

IMPROVEMENTS

Here are some improvements made to the models' accuracy, and we also tested a deep learning model to see whether it worked with our dataset

So, I have chosen the dataset which was labelled in three classes i.e positive, negative and neutral and used following techniques on that dataset to get best results.

As we know our Logistic regression gives the best accuracy among all , now my aim to increase its accuracy and make it more efficient so, for first improvement I have implied "***Grid Search algorithm***" for hyper parameter tunning.

I observe that my dataset is little bit of unbalanced so, I gave weights to classes.

```
class_weights = {
    0: len(y) / np.sum(y == 0),  # Weight for class 0
    1: len(y) / np.sum(y == 1),  # Weight for class 1
    2: len(y) / np.sum(y == 2)   # Weight for class 2
}
```

```
class_weights
```

```
{0: 9.030206532617596, 1: 8.922240201366415, 2: 1.2867014104957477}
```

I have observed that the features extracted by feature extractors are very high numbered i-e 2383593 , so when I restrict the features to about 40000 and when I increased no. of iterations of model the accuracy of model increased from 83.54% to 84.052%

```
tfidf_n = TfidfVectorizer(ngram_range=(1,2),stop_words = 'english',max_features=40000)
text_fit(X, y, tfidf_n, LogisticRegression(max_iter=1300))
```

```
# features: 40000
# train records: 148876
# test records: 49626
Model Accuracy: 0.8405271430298634
```

I came to know about a classifier model named SGDclassifier. With loss :hinge it becomes Linear SVM and with loss: log_loss it becomes logistic regression. It is appropriate for situations where the number of features is significantly more than the number of samples

```
tfidf = TfidfVectorizer(ngram_range=(1,3),stop_words = 'english',max_features=40000)
text_fit(X, y, tfidf, SGDClassifier(alpha = 0.0001, average = False, class_weight = class_weights,
   early_stopping = True, epsilon = 0.1, eta0 = 0.0, fit_intercept = True,
   l1_ratio = 0.15, learning_rate = 'optimal', loss = 'hinge', max_iter = 1300, n_iter_no_change = 5, n_jobs = None, penalty = 'elasticnet',
   power_t = 0.5, random_state = None, shuffle = True, tol = 0.001,
   validation_fraction = 0.1, verbose = 0, warm_start = False))
```

```
# features: 40000
# train records: 148876
# test records: 49626
Model Accuracy: 0.8262201265465683
```

since it can handle high-dimensional datasets with many features effectively.

I have also tried some other feature extractors such as Word2Vec and FastText with our machine learning models.

```
# train word2vec model on the text data
sentences = [review.split() for review in X]
model = Word2Vec(sentences, min_count=1)

# extract features using word2vec
X_vectors = []
for sentence in sentences:
    vector = []
    for word in sentence:
        if word in model.wv:
            vector.append(model.wv[word])
    if vector:
        X_vectors.append(sum(vector) / len(vector))
    else:
        X_vectors.append([0] * model.vector_size)

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.2, random_state=42)

# train logistic regression model on the extracted features
clf = LogisticRegression(max_iter=1300)
clf.fit(X_train, y_train)
acc = clf.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.8002317321981814
```

At last I have applied deep learning transformer model BERT(Bidirectional Encoder

```
# Train FastText model
sentences = [review.split() for review in X]
model = FastText(sentences, window=5, min_count=5, workers=4)

# Transform text into vectors using FastText model
X_vectors = []
for sentence in sentences:
    vector = []
    for word in sentence:
        if word in model.wv:
            vector.append(model.wv[word])
    if vector:
        X_vectors.append(sum(vector) / len(vector))
    else:
        X_vectors.append([0] * model.vector_size)

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.2, random_state=42)

# train logistic regression model on the extracted features
clf = LogisticRegression(max_iter=1300)
clf.fit(X_train, y_train)
acc = clf.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.7983174227349437
```

Representations from Transformers). It gave highest accuracy of 89.48%.

I have use Voting Classifier from ensemble learning technique to create ensemble models to get better results so, I have used different classifiers for ensemble models such as- Deceison tree + random forest,+ logistic regression, random forest+xgboost +logistic regression , logistic regression +xgboost, logistic regression +sgd classifier, logistic regression+random forest. I got maximum accuracy with random forest and logistic regression of 84.30%

```
tfidf = TfidfVectorizer(ngram_range=(1,3),stop_words = 'english',max_features=40000)
tfid=tfidf.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(tfid, y, test_size=0.2, random_state=42)
ensemble_model = VotingClassifier(estimators=[ ('sgd',sgd),('lgr',lg)], voting='hard')
ensemble_model.fit(X_train,y_train)
acc = ensemble_model.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.8336565829576081
```

```
tfidf = TfidfVectorizer(ngram_range=(1,3),stop_words = 'english',max_features=40000)
tfid=tfidf.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(tfid, y, test_size=0.2, random_state=42)
ensemble_model = VotingClassifier(estimators=[ ('rf',rf),('lgr',LogisticRegression(max_iter=1300))], voting='hard')
ensemble_model.fit(X_train,y_train)
acc = ensemble_model.score(X_test, y_test)
print ('Model Accuracy: {}'.format(acc))
```

```
Model Accuracy: 0.8430014357320974
```

CONCLUSION

In my thorough examination of sentiment categorization, I looked into the effectiveness of numerous machine learning algorithms and how they worked together with different vectorization techniques. I uncovered significant model performance improvements through diligent experimentation and provided insightful insights.

Implementing particular tweaks proven to be essential in boosting accuracy by optimising the Tf-idf + n-gram + Logistic Regression model. Specifically, the model demonstrated higher classification abilities by limiting the feature space to 40000 and dramatically increasing the number of training iterations to roughly 1300. Better feature selection and more robust learning were made possible by this optimisation approach, which eventually produced sentiment predictions that were more correct.

Expanding on my research, I looked at the possible advantages of ensemble learning by combining the capabilities of random forest and logistic regression. Overall accuracy was increased by building an ensemble model out of the predictions from both methods. Diverse viewpoints were made possible by the ensemble approach, which also increased the robustness of sentiment analysis.

Additionally, I explored the world of deep learning by using the cutting-edge language model BERT. With an amazing accuracy score of 89.743%, BERT outperformed all other techniques and shown exceptional performance. BERT demonstrated to be a highly successful tool for sentiment categorization by utilising its sophisticated contextual comprehension and semantic representation capabilities.

The need of investigating and fine-tuning various machine learning algorithms, vectorization strategies, and deep learning models to obtain superior sentiment analysis performance is highlighted by the findings taken together. The never-ending search to improve and adapt these techniques has enormous potential for bringing out more nuanced insights from textual data and influencing decision-making.

CHAPTER – 4

References-

[1] Ahuja, Ravinder, Aakarsha Chug, Shruti Kohli, Shaurya Gupta, and Pratyush Ahuja. "The impact of features extraction on the sentiment analysis." *Procedia Computer Science* 152 (2019): 341-348.

[2] Nguyen, Heidi, Aravind Veluchamy, Mamadou Diop, and Rashed Iqbal. "Comparative study of sentiment analysis with product reviews using machine learning and lexicon-based approaches." SMU Data Science Review 1, no. 4 (2018): 7.

[3] Mishev, Kostadin, Ana Gjorgjevikj, Irena Vodenska, Lubomir T. Chitkushev, and Dimitar Trajanov. "Evaluation of sentiment analysis in finance: from lexicons to transformers." IEEE access 8 (2020): 131662-131682.

[4] Cardie, Claire. "Sentiment Analysis and Opinion Mining Bing Liu (University of Illinois at Chicago) Morgan & Claypool (Synthesis Lectures on Human Language Technologies, edited by Graeme Hirst, 5 (1)), 2012, 167 pp; paperbound, ISBN 978-1-60845-884-4." (2014): 511-513.

[5] Gonçalves, Pollyanna, Matheus Araújo, Fabrício Benevenuto, and Meeyoung Cha. "Comparing and combining sentiment analysis methods." In Proceedings of the first ACM conference on Online social networks, pp. 27-38. 2013.

[6] Neethu, M. S., and R. Rajasree. "Sentiment analysis in twitter using machine learning techniques." In 2013 fourth international conference on computing, communications and networking technologies (ICCCNT), pp. 1-5. IEEE, 2013.

[7] Saberi, Bilal, and Saidah Saad. "Sentiment analysis or opinion mining: A review." Int. J. Adv. Sci. Eng. Inf. Technol 7, no. 5 (2017): 1660-1666.

[8] Yelena, Mejova. "Sentiment Analysis: An Overview Comprehensive Exam Paper." Computer Science Department, University of Iowa (2009).

[9] Tyagi, Abhilasha, and Naresh Sharma. "Sentiment analysis using logistic regression and effective word score heuristic." International Journal of Engineering and Technology (UAE) 7, no. 2.24 (2018): 20-23.

[10] Vohra, Aarushi, and Ritu Garg. "Deep learning based sentiment analysis of public perception of working from home through tweets." Journal of Intelligent Information Systems 60, no. 1 (2023): 255-274.

[11] Osisanwo, F. Y., J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi. "Supervised machine learning algorithms: classification

and comparison." International Journal of Computer Trends and Technology (IJCTT) 48, no. 3 (2017): 128-138.

[12] Abhishek, L. "Optical character recognition using ensemble of SVM, MLP and extra trees classifier." In 2020 International Conference for Emerging Technology (INCET), pp. 1-4. IEEE, 2020.

[13] Zhang, Dahai, Liyang Qian, Baijin Mao, Can Huang, Bin Huang, and Yulin Si. "A data-driven design for fault detection of wind turbines using random forests and XGboost." Ieee Access 6 (2018): 21020-21031.

[14] Zvonarev, Andrey, and Andrey Bilyi. "A Comparison of Machine Learning Methods of Sentiment Analysis Based on Russian Language Twitter Data." In MICSECS. 2019.

[15] Ahamed, B. Shamreen. "Prediction of type-2 diabetes using the LGBM classifier methods and techniques." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, no. 12 (2021): 223-231.

[16] Dharma, EDDY MUNTINA, F. Lumban Gaol, H. L. H. S. Warnars, and B. E. N. F. A. N. O. Soewito. "The accuracy comparison among Word2vec, Glove, and Fasttext towards convolution neural network (CNN) text classification." Journal of Theoretical and Applied Information Technology 100, no. 2 (2022): 31.

turnitin

Similarity Report ID: oid:27535:36428198

PAPER NAME	AUTHOR
krishan report (1) (1) (1).docx	Krishan Kumar

WORD COUNT	CHARACTER COUNT
4673 Words	27008 Characters
PAGE COUNT	**FILE SIZE**
37 Pages	1.1MB
SUBMISSION DATE	**REPORT DATE**
May 29, 2023 4:02 PM GMT+5:30	May 29, 2023 4:03 PM GMT+5:30

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- 3% Publications database
- Crossref database
- Crossref Posted Content database
- 7% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Small Matches (Less then 10 words)