OPINION MINING ON TWITTER DATA USING MACHINE LEARNING

BATCH-07

1.Introduction

Opinion mining, also known as sentiment analysis, is a technique used to analyze the subjective opinions expressed in text data. In recent years, Twitter has become a popular platform for expressing opinions on a variety of topics, ranging from politics to entertainment. As a result, there has been an increasing interest in using Twitter data for sentiment analysis. Machine learning is a subfield of artificial intelligence that focuses on building algorithms that can learn from data and make predictions or decisions. In the context of opinion mining on Twitter data, machine learning algorithms can be trained on a large corpus of annotated tweets to predict the sentiment of new, unseen tweets. The goal of opinion mining on Twitter data using machine learning is to automatically classify tweets into positive, negative, or neutral categories based on the sentiment expressed in the tweet. This can be useful for a variety of applications, such as monitoring public opinion on a particular topic, identifying customer satisfaction or dissatisfaction with a product or service, or predicting the outcome of an election. To implement opinion mining on Twitter data using machine learning, a typical workflow involves several steps. First, a dataset of tweets is collected and preprocessed to remove noise and irrelevant information. Next, the dataset is labeled with sentiment annotations using VADER. We collected a corpus of text posts from Twitter evenly split automatically between three sets of texts:

1. Texts containing positive emotions, such as happiness, amusement, or joy
2. Texts containing negative emotions, such as sadness, anger, or disappointment
3. Objective texts that only state a fact or do not express any emotions.

Then, a machine learning model is trained on the labeled dataset using a variety of algorithms, such as support vector machines, decision trees, and random forests. Finally, the trained model is used to classify new, unseen tweets into the appropriate sentiment category.

2. PROPOSED METHODOLOGY

Real time data is analyzed using TextBlob library. From our Literature Survey, we noticed that in preprocessing phase emojis had been removed. We, therefore, converted the emojis to text format using emot module and the resultant text was preprocessed by NLP. We are using different machine learning algorithms like SVM, Decision Tree, and Random Forest.

Data collection:

Static Data :

The dataset of 14,110 tweets is used. we collected the static data from DATA.NASA.GOV. The collected static data doesn't have class labels. Adding labels to a dataset is very important before you can use it to solve a problem. One of those problems where adding labels to a dataset is very important is sentiment analysis, where you get the data as reviews or comments from users, and you need to add labels to it to prepare it for sentiment analysis.

Image of Static Dataset:

	Tweet ID	Tweet
0	0	Here's to you, Oppy. 🥂 \n\nBefore you say #Good...
1	1	Are there rivers and lakes on other worlds? Yo...
2	2	We want to hear from you!\n\nJoin our series o...
3	3	The @NASAExoplanets data hint that WASP-39 b, ...
4	4	.@NASAWebb just scored another first: a full p...
...	...	...
14105	14105	The supermoon is here! Be sure to bundle up th...
14106	14106	Ever wonder how we track supermoons 🟡 and othe...
14107	14107	A supermoon is coming! Tonight, the full Moon ...
14108	14108	Happy New Year from space! Astronauts aboard t...
14109	14109	□ Send a robot to Mars\n□ Launch @NASA_Astrona...

14110 rows × 2 columns

Fig 1: NASA dataset of tweets collected through DATA.NASA.GOV.

Real-Time Data

We used the Twitter API to fetch real-time data from Twitter, which allows developers to programmatically access Twitter data in real time. To use the Twitter API, we need to create a Twitter developer account. We followed the instructions on the Twitter developer website to create an account. Once we created a Twitter account, we needed to authenticate requests using OAuth 1.0a or OAuth 2.0. We find the information about authentication on the Twitter developer website. Twitter provides different API endpoints for accessing different types of data. For real-time data, we used the Streaming API, which provides a continuous stream of tweets that match your search criteria, or the Premium or Enterprise APIs, which provide historical and real-time access to a larger volume of tweets. Depending on the API endpoint we choose, we write code to make API requests and handle the data returned by the API. Twitter provides documentation and code examples to help you get started. Once we fetched the real-time data, you can process it as per requirement. It's important to note that there are rate limits and other restrictions on the Twitter API, so be sure to check the documentation for details on usage limits and other guidelines

```
b'_The_Goat is it like adidas or'
b"RT Missed out on the last giveaway We've got another one coming at ya. Make the Beehive State Kit y
ours by liking,\xe2\x80\xa6"
b'#AdidasPH HYPERGLAM TRAINING TECHFIT SHORT LEGGINGS involved https://t.co/0UHLPfewvh Ecomobi: http
s://t.co/d2QnnYCgKx'
b'Adidas TOPANGA WHITE/BLACK Available in: Sizes EU 40-45 US 6-11 Price tag is Ugx 150,000 Call/Whats
App +256 https://t.co/8MYe6fiLvZ'
b'RT First Look adidas Crazy IIInfinity https://t.co/661xgI3gJx'
b'RT From doing high school color commentary at a near empty gym to covering All Star Weekend with ad
idas Never give up \xf0\x9f\x99\x8f htt\xe2\x80\xa6'
b'I don\xe2\x80\x99t suffer from bpd but my fianc does and it\xe2\x80\x99s stressful some days but we
get through it'
b'GRAILED https://t.co/17MOcuyO6y KICKS CREW https://t.co/yGTxHY8yep KLEKT https://t.co/2xFOHg8WEw YA
NKEEKICKS https://t.co/aYEAgtK5kI'
b"Move over status symbol 'cause #sneakers just took over https://t.co/JT6QnRCPkS"
b"SHOP adidas YEEZY 450 'Resin' StockX https://t.co/OMmkwaPYYz GOAT https://t.co/Ew7UC6yFJ8 ebay http
```

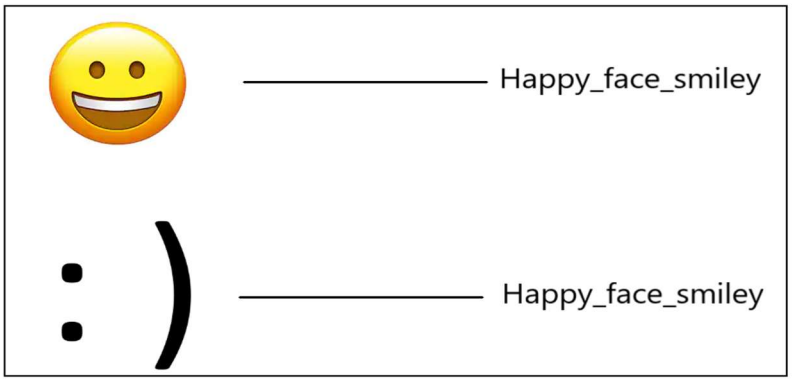
Fig 2: Real time data for adidas tweets

Data Preprocessing :

Text pre-processing is one of the mandatory steps we will perform while creating an NLP application. As humans, the text we usually write contains lots of spelling errors, short words, special symbols, emojis, etc, which we can understand but we need to preprocess this text if we want the computer to understand it.

1. **Conversion of emoji to text**

   Emot is a python library to extract emojis and emoticons from a text(string). All the emojis and emoticons are taken from a reliable source i.e. Wikipedia.org. The Emot library takes a string as input and returns a list of dictionaries. You can add/delete/modify that file under the library to create another dynamic pattern. The python 3.0 version provides more options such as bulk processing. It is useful when you have a long list of "sentences or words" and want to use multiprocessing power to speed up the process. It means you can dynamically create the pattern for the emoji or emoticons and run it in multiprocessing to get maximum performance from multiple cores.



2. **Conversion to lowercase**

   It is the process of converting a word into lowercase. If a particular word (let's say Book) appears at the beginning of the sentence with a capital letter and another word (book) appears later in the sentence without a capital letter, our model will treat these 2 words differently. The process of lowercasing is usually very simple, we can use the .lower()method (e.g. 'Canada' to 'canada')

3. Removal of HTML tags and URLs

URLs in a text reference a location to the web but just like HTML tags, it doesn't provide any useful information. We can remove URLs by using regular expressions. These tags won't add any value to the text data.

Before removing the URL: My profile link: https://www.twitter.com/vinay1234
Text after removing URL: My profile link:

4. Removal of Punctuations

The reason of removing punctuations is pretty simiar to lowercasing, in certain cases, we want the word hello and hello! to be treated in the exact same way. Although, be careful while using punctuation, the word can't can be converted to cant and can t depending upon what you set in the parameter .period, comma, apostrophe, quotation, question, exclamation, brackets, braces, parenthesis, dash, hyphen, ellipsis, colon, and semicolon are the puncuations which need to remove from text.

(e.g. 'hello!' vs 'hello')

5. Removal of stopwords

Stop words are words such as the, an, so, and which are present in abundance in every text but they don't provide much uselful information to the model, so by removing these words, we can focus on the more important information in the text. Although in some cases, we don't remove these stop words, one example being sentiment analysis.

6. Stemming

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem or root form. In the context of sentiment analysis, stemming can be useful because it allows us to normalize words that have different forms but the same meaning, which can help to improve the accuracy of sentiment analysis.

For example, consider the following sentence: "I love running, but I hate runners." If we

were to apply sentiment analysis without stemming, the word "running" and "runners" would be treated as separate words, even though they have the same root form. This could result in a less accurate analysis of the sentiment expressed in the sentence.

However, if we apply to stem to the sentence, the words "running" and "runners" would both be reduced to the stem "run", which would help to ensure that the sentiment analysis accurately reflects the sentiment expressed in the sentence.

7. **Lemmatization**

Lemmatization is a linguistic process of reducing words to their base or dictionary form, which is called a lemma. In the context of sentiment analysis, lemmatization can be useful because it can help to improve the accuracy of sentiment analysis by reducing the number of unique words in a text while still preserving their semantic meaning.

For example, consider the sentence: "She went to the bank to deposit her money." If we were to apply sentiment analysis without lemmatization, the words "went" and "deposit" would be treated as separate words, even though they have the same root form as "go" and "deposit", respectively. This could result in a less accurate analysis of the sentiment expressed in the sentence. However, if we apply lemmatization to the sentence, the words "went" and "deposit" would both be reduced to their base form, which would help to ensure that the sentiment analysis accurately reflects the sentiment expressed in the sentence.

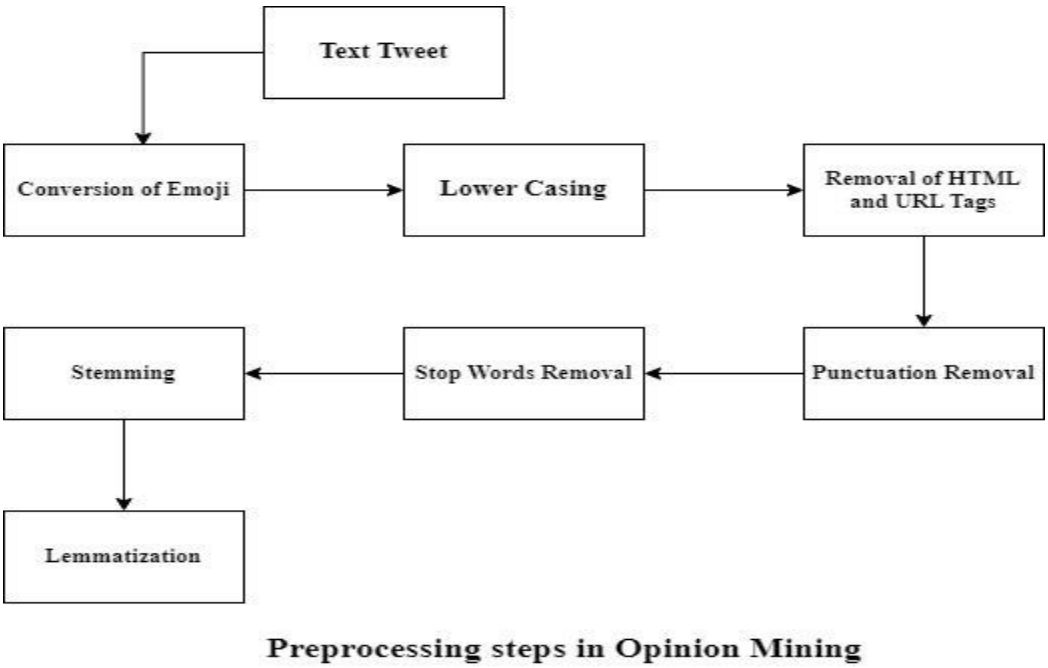


Fig 3: Phases in preprocessing

TextBlob

TextBlob is a Python library used for natural languages processing tasks such as sentiment analysis, part-of-speech tagging, and noun phrase extraction. TextBlob provides an easy-to-use interface for performing sentiment analysis on text data. In TextBlob, sentiment analysis is performed using a machine learning-based approach where a pre-trained sentiment classifier is used to classify the polarity (positive, negative, or neutral) of a given text. The classifier is trained on a dataset of movie reviews, where each review is labeled as positive or negative.

To perform sentiment analysis using TextBlob, first, you need to create a TextBlob object for the text you want to analyze. Then, you can use the sentiment property of the TextBlob object to get a tuple containing the polarity and subjectivity scores for the text. The polarity score is a float value between -1 and 1, where negative values indicate negative sentiment, positive values indicate positive sentiment, and 0 indicates neutral sentiment. The

subjectivity score is a float value between 0 and 1, where 0 indicates an objective statement and 1 indicates a subjective statement.

Adding Class Labels:

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a rule-based sentiment analysis tool that is widely used for analyzing the sentiment of social media content. In VADER, sentiment analysis is performed by assigning a sentiment score to each word in a given text. The score ranges from -1 to 1, where -1 represents a highly negative sentiment, 0 represents a neutral sentiment, and 1 represents a highly positive sentiment. The overall sentiment of the text is calculated by aggregating the individual scores of the words in the text.

To add class labels using VADER, using threshold value to classify the sentiment scores into positive, negative, or neutral classes. For example, consider scores greater than 0.5 as positive, scores less than -0.5 as negative, and scores between -0.5 and 0.5 as neutral.

Feature extraction:

Feature extraction is a critical step in sentiment analysis because it involves identifying and selecting the most relevant features or attributes from text data that can help in predicting the sentiment of the given text. The success of any sentiment analysis system depends on the quality and appropriateness of the features that it uses.

CountVectorizer is a Python library that is commonly used in natural language processing tasks such as sentiment analysis, topic modeling, and document classification. It is used to convert text data into a matrix of token counts, which is then used as input to machine learning algorithms.

In sentiment analysis, CountVectorizer is used to convert a collection of text documents into a matrix of token counts, which can then be used to train a machine learning model to classify the sentiment of a given text as positive, negative, or neutral. CountVectorizer

works by first tokenizing the input text into individual words or n-grams, and then counting the occurrence of each token in each document.

frequency/count [Word] = Total number of times a word appeared in the text

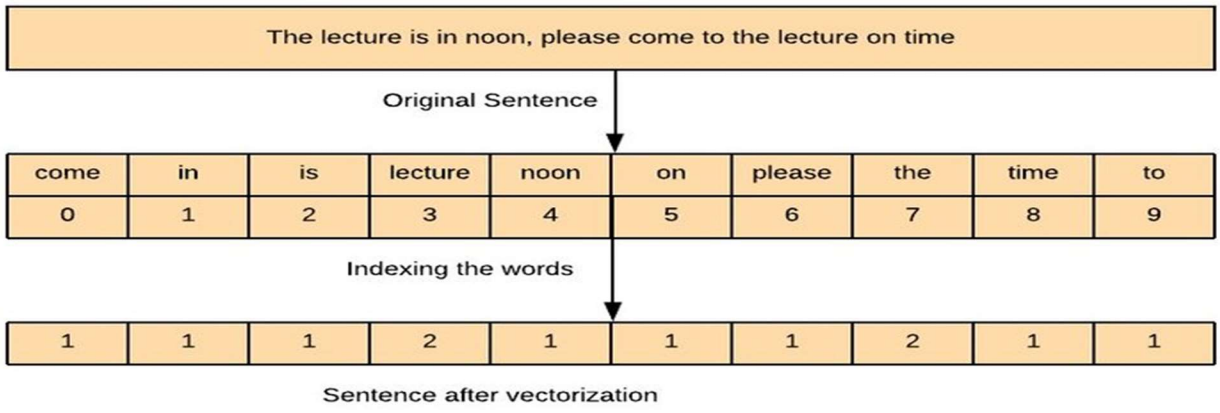


Fig 4: CountVectorizer on a tweet

Data splitting:

It is a process of splitting the dataset into training and testing sets. The datasets consist of 14,110 rows. Training datasets are used to train the model and testing sets are used for validation and testing. Here dataset is split in the basis of 70% and 30% as train and test sets respectively.

Model Selection:

Selecting a machine learning model for sentiment analysis involves considering several factors, such as the size of dataset, the nature of text data, the complexity of the sentiment analysis task, and the required level of accuracy. Before selecting a model, it is important to define the problem you are trying to solve. For our use-case, the problem is multiclass classification , we used several models like support vector machines (SVM), Decision tree, and Random Forest. It is

important to note that selecting a suitable model for sentiment analysis can be a trial-and-error process, and it may take several iterations to find the best model for your specific problem.

Model Training:

The preferred dataset is collected and loaded to the model by splitting in the ration of 7:3 where 7 proportions is training data and 3 proportion is testing data. These trainings sets are loaded to the model.. By using a training set the selected models learn the process and get trained.

Model Evaluation:

Models get trained by extracting features of tweets and based on their characteristics they are. Evaluating the performance of the models on the Testing data set using metrics such as accuracy, precision, recall, and F1 score

By using the classification report.

Model Testing:

Machine learning model testing is a crucial step in sentiment analysis to ensure that the model is performing accurately and reliably. We prepared a set of test data that is separate from the training data. This ensures that the model is not overfitting to the training data and can generalize well to new data. Evaluate the performance of the model using metrics such as accuracy, precision, recall, and F1 score. Accuracy measures the overall percentage of correct predictions, while precision measures the percentage of correct positive predictions, and recall measures the percentage of actual positive instances correctly predicted. F1 score is a harmonic mean of precision and recall

Model Architecture:

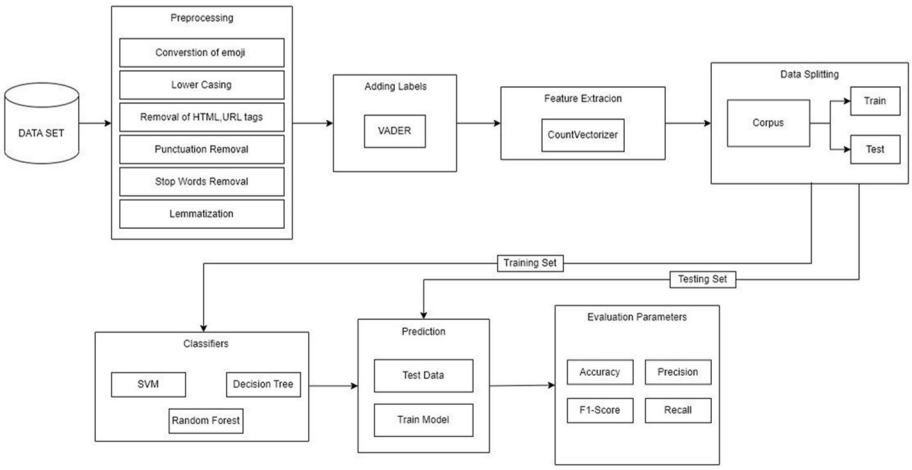


Fig 5: Model Architecture of sentiment analysis

ALGORITHMS USED TO OVERCOME THE IDENTIFIED PROBLEM

1 Support vector machine:

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
- However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
- This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

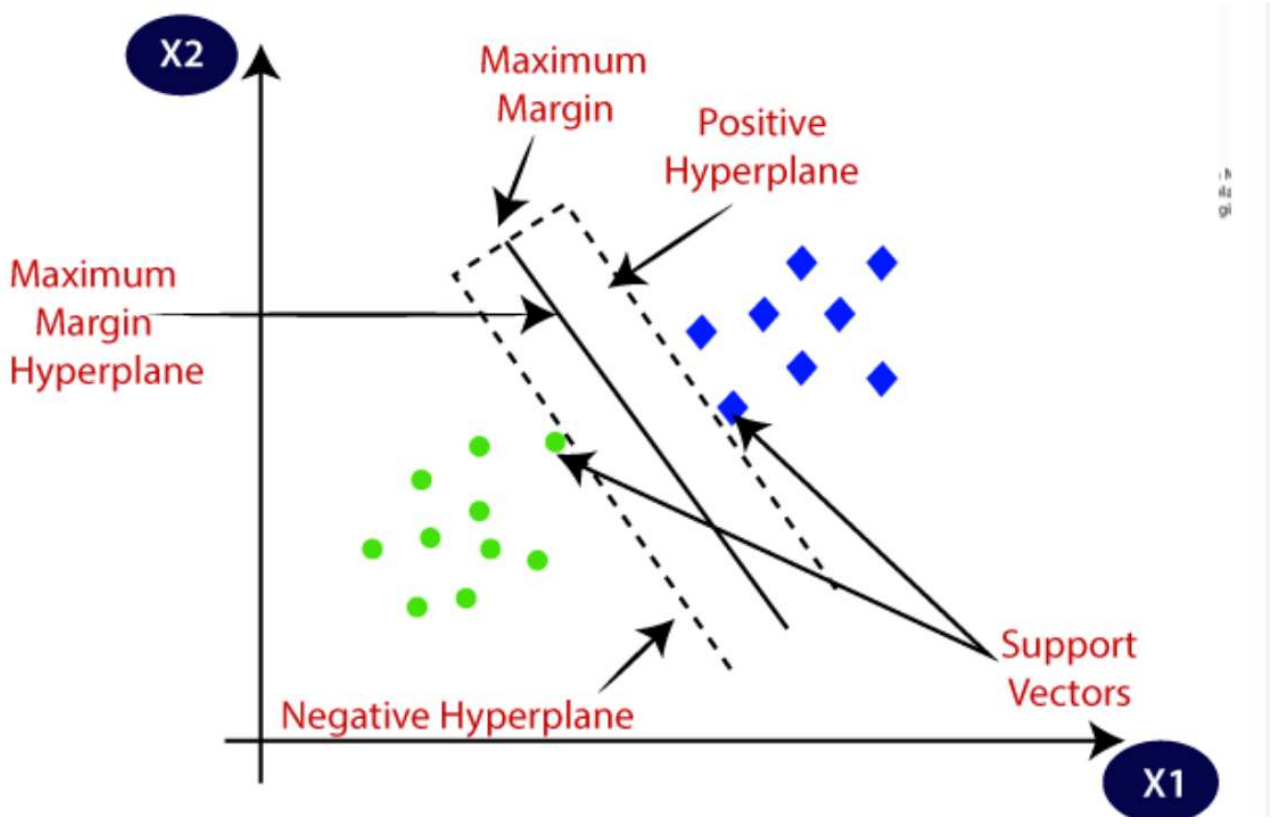


Fig 6: SVM Hyperplane

PSEUDO CODE FOR SVM

Input: Training dataset (X_train, y_train), Testing dataset (X_test)
Output: Predictions for the testing dataset

Step 1: Preprocess the training and testing dataset
- Convert the text into numerical vectors using techniques like CountVectorizer
- Normalize the data
- Remove stop words and perform stemming or lemmatization if necessary

Step 2: Train the SVM model on the training dataset
- Define the SVM algorithm and its hyperparameters
- Fit the model to the training dataset using the SVM algorithm

Step 3: Test the SVM model on the testing dataset
- Predict the sentiment labels for the testing dataset using the trained SVM model
- Calculate the accuracy, precision, recall, and F1 score of the model on the testing dataset

Step 4: Optimize the SVM model
- Tune the hyperparameters of the SVM model using techniques like cross-validation
- Repeat steps 2 and 3 with the optimized hyperparameters

2.Decision Tree

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.

- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:

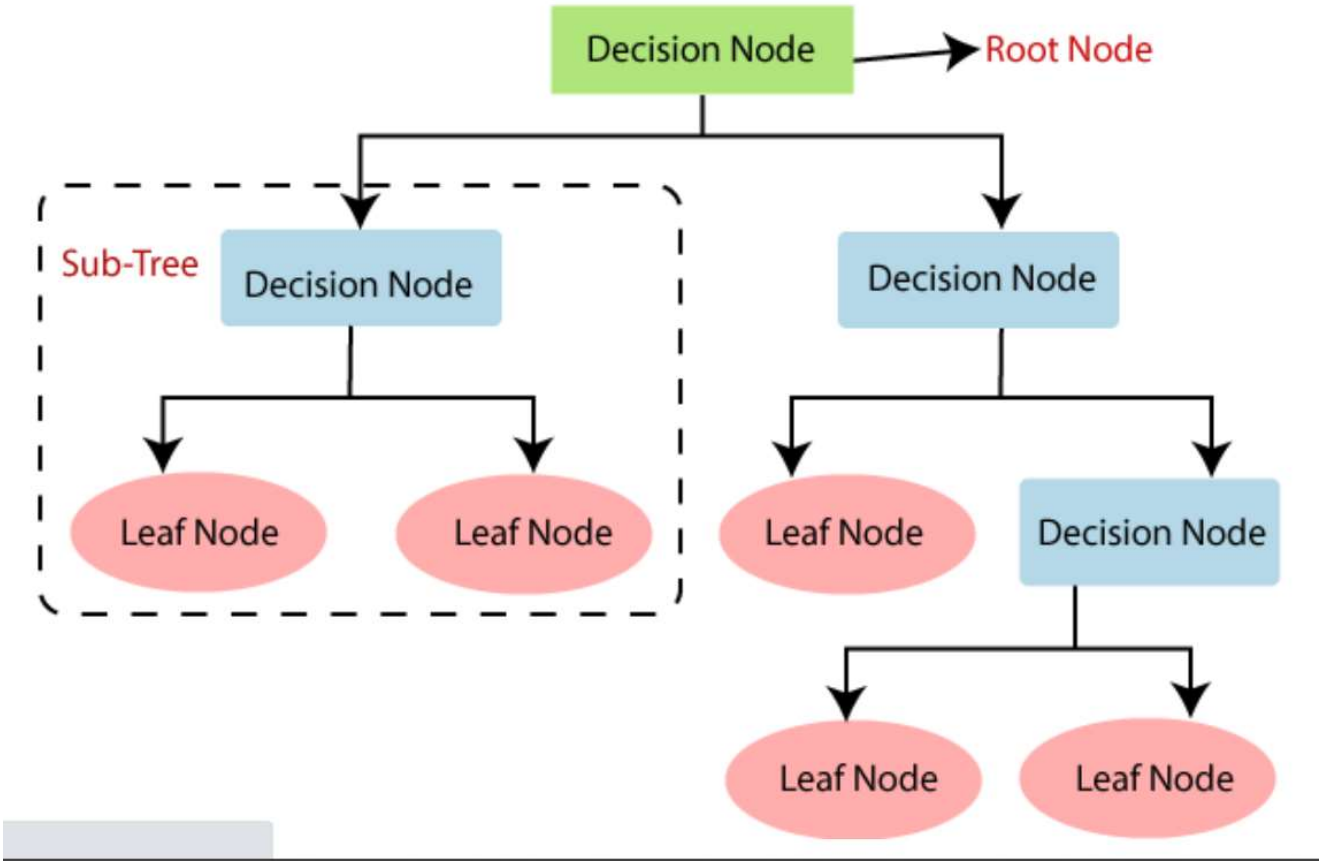


Fig 7: Decision Tree

PSEUDO CODE FOR RANDOM FOREST

Input: Training dataset (X_train, y_train), Testing dataset (X_test)

Output: Predictions for the testing dataset

Step 1: Preprocess the data

- Convert the text into numerical vectors using techniques like CountVectorizer
- Normalize the data
- Remove stop words and perform stemming or lemmatization if necessary

Step 2: Build the Decision Tree

- Define a function to calculate the information gain or other splitting criterion for each feature
- Define the stopping criteria for the tree, such as maximum depth or minimum number instances per node
- Recursively build the tree by selecting the feature with the highest information gain and splitting the dataset
- Assign class labels to the leaf nodes based on the majority class of the instances

Step 3: Test the Decision Tree

- Traverse the Decision Tree for each instance in the testing dataset and predict the class label based on the leaf node reached
- Calculate the accuracy, precision, recall, and F1 score of the model on the testing dataset

3.RANDOM FOREST

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML.
- It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

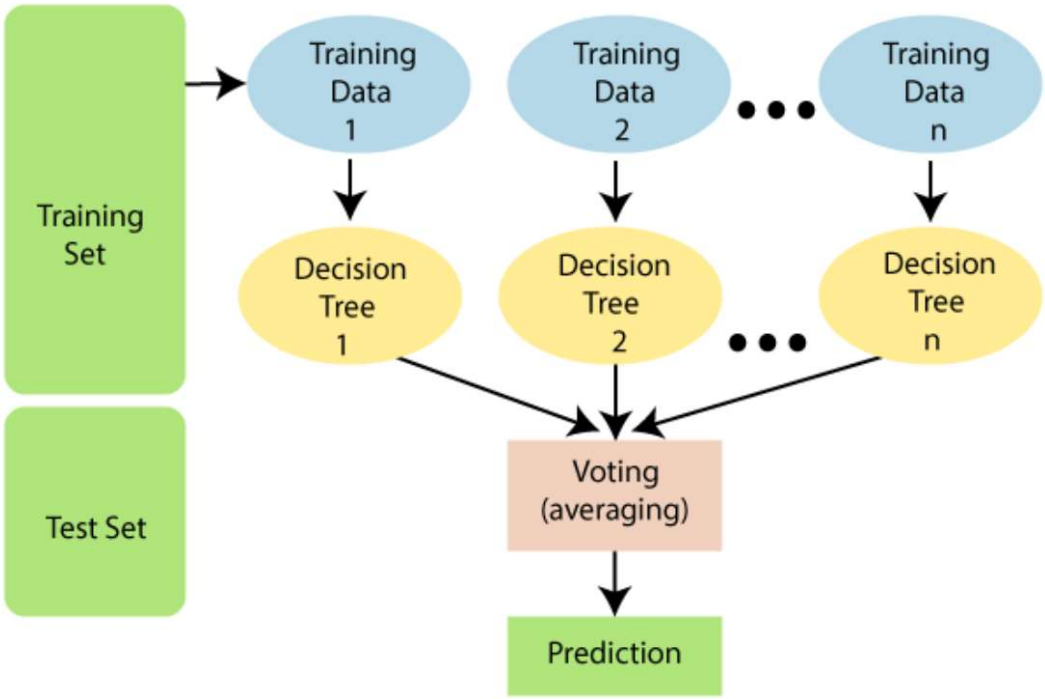


Fig 8: Random Forest

PSUEDO CODE FOR RANDOM FOREST

Input: Training dataset (X_train, y_train), Testing dataset (X_test)

Output: Predictions for the testing dataset

Step 1: Preprocess the data

- Convert the text into numerical vectors using techniques like CountVectorizer
- Normalize the data
- Remove stop words and perform stemming or lemmatization if necessary

Step 2: Build the Random Forest

- Define the number of decision trees (n_estimators) and the maximum depth of the trees (max_depth)

- Define the stopping criteria for the tree, such as minimum number of instances per leaf node
- For each decision tree:
  - Randomly select a subset of the features (max_features) to use for splitting the data
  - Sample the training dataset with replacement (bootstrapping) to create a new dataset for training the tree
  - Build the Decision Tree using the new dataset and the selected features

Step 3: Test the Random Forest

- For each instance in the testing dataset:
  - Predict the class label by aggregating the predictions of all decision trees
  - Calculate the accuracy, precision, recall, and F1 score of the model on the testing dataset

MODULES USED

To implement our project some inbuilt libraries or modules are used.

Matplotlib:

Matplotlib is a popular Python library for creating high-quality data visualizations. It provides a wide range of 2D and 3D plotting functionalities and is widely used for data exploration and presentation. Matplotlib is an open-source library and is used for data analysis, including line plots, scatter plots, bar plots, histograms, and more. Here matplotlib is used to visualize the graphs of different diseases and to plot confusion matrix. Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB

pyplot:

Pyplot is a module within the Matplotlib library that provides a convenient interface for creating plots using Matplotlib. It provides a high-level interface for creating a wide range of 2D plots, such as bar plots.

Seaborn:

Seaborn is a Python data visualization library that is built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn provides a wide range of data visualization capabilities, including categorical plots. Seaborn is designed to work well with Pandas data frames.

Sklearn:

Scikit-learn (sklearn) is a popular Python machine learning library that provides a wide range of tools for data analysis. Scikit-learn provide a simple and efficient interface for performing common machine learning tasks such as classification. The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

4. CONCLUSION AND RESULTS

In the context of this work we present an approach that aims to extract Twitter sentiment analysis by converting emojis to text. Experimental results indicate that Random Forest and Decision Tree have an almost similar accuracy which is better than that of the support vector machine(SVM). However, the Random Forest gives the highest accuracy at 85% followed by Decision Tree at 84%, and Support vector machine(SVM) at 83%

SVM Performance:

```
[Text(0.5, 15.0, 'Predicted Labels'), Text(33.0, 0.5, 'Actual labels')]
```

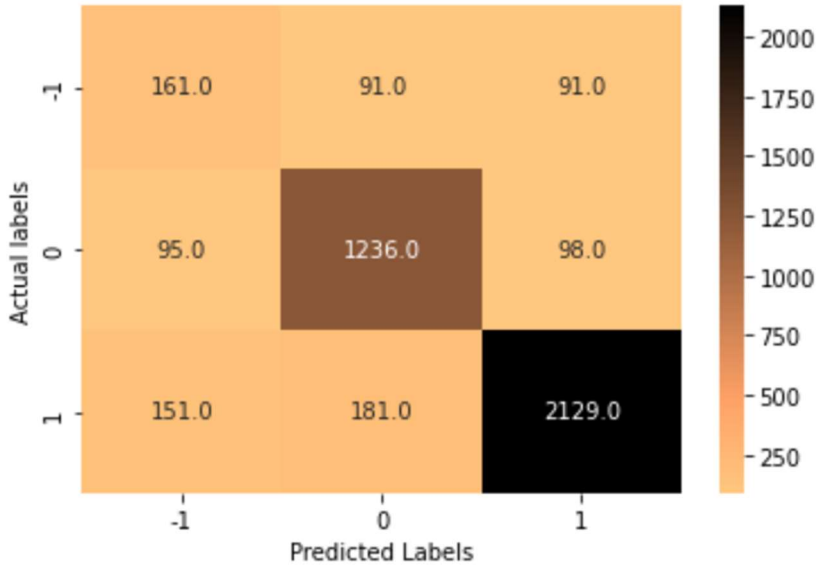


Fig 9: Confusion matrix of SVM

```
accuracy_score 0.8445546893456177
              precision    recall  f1-score   support

          -1       0.45      0.49      0.47       355
           0       0.84      0.87      0.85      1462
           1       0.92      0.88      0.90      2416

    accuracy                           0.84      4233
   macro avg       0.73      0.75      0.74      4233
weighted avg       0.85      0.84      0.85      4233
```

Fig 10: Classification report of SVM

Decision Tree Performance:

```
Out[72]: [Text(0.5, 15.0, 'Predicted Labels'), Text(33.0, 0.5, 'Actual labels')]
```

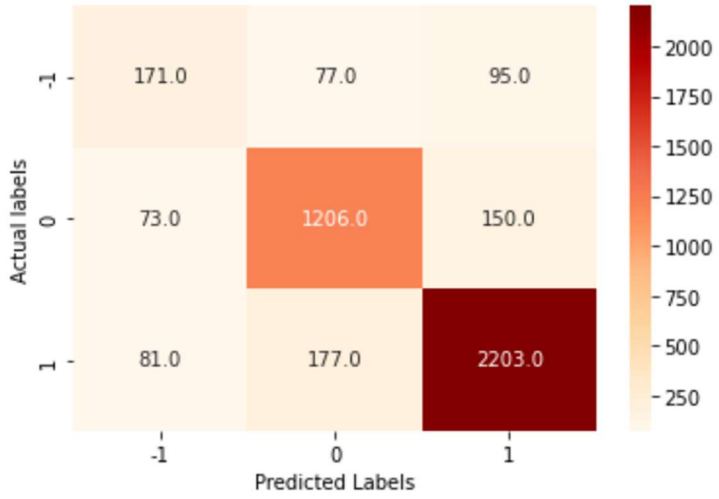


Fig 11: Confusion matrix of Decision Tree

```
accuracy_score 0.8412473423104181
              precision    recall  f1-score   support

          -1       0.52      0.43      0.47       355
           0       0.83      0.85      0.84      1462
           1       0.89      0.90      0.89      2416

    accuracy                           0.84      4233
   macro avg       0.74      0.72      0.73      4233
weighted avg       0.84      0.84      0.84      4233
```

Fig 12: Classification report of Decision Tree

Random Forest Performance:

```
Out[70]: [Text(0.5, 15.0, 'Predicted Labels'), Text(33.0, 0.5, 'Actual labels')]
```

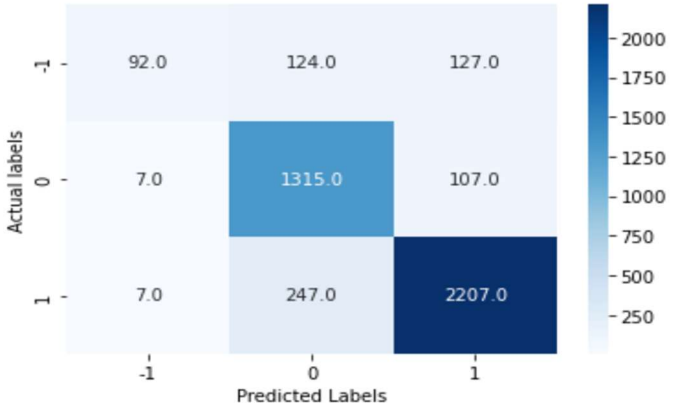


Fig 13: Confusion matrix of Random Forest

accuracy_score 0.8452634065674463

	precision	recall	f1-score	support
-1	0.85	0.23	0.36	355
0	0.80	0.90	0.85	1462
1	0.88	0.90	0.89	2416
accuracy			0.85	4233
macro avg	0.84	0.68	0.70	4233
weighted avg	0.85	0.85	0.83	4233

Fig 14: Classification report of Random Forest

PROJECT MEMBERS:

R. Vinay -19BQ1A05I9,

P. Manjunath -19BQ1A05I2,

P. Srinivas -19BQ1A05H6,

D.Niharika -19BQ1A05D2.

PROJECT GUIDE:

M. Kishore Babu,

Assistant Professor,

Dept of CSE.

Signature of Guide