VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Social Network Analysis LAB

Assessment - 3 REPORT

Sentiment Analysis on Tweets

Faculty Name: Durgesh Kumar

Student Name: Sarthak Tripathi

Reg. No.: 22MCB0030

E-mail: sarthak.tripathi2022@vitstudent.ac.in

Mobile No.: 8630054046

Table of Contents

INTRODUCTION

Nowadays, the age of Internet has changed the way people express their views, opinions. It is now mainly done through blog posts, online forums, product review websites, social media, etc. Nowadays, millions of people are using social network sites like Facebook, Twitter, Google Plus, etc. to express their emotions, opinion and share views about their daily lives. Through the online communities, we get an interactive media where consumers inform and influence others through forums. Social media is generating a large volume of sentiment rich data in the form of tweets, status updates, blog posts, comments, reviews, etc. Moreover, social media provides an opportunity for businesses by giving a platform to connect with their customers for advertising. People mostly depend upon user generated content over online to a great extent for decision making. For e.g., if someone wants to buy a product or wants to use any service, then they firstly look up its reviews online, discuss about it on social media before taking a decision. The amount of content generated by users is too vast for a normal user to analyse. So, there is a need to automate this, various sentiment analysis techniques are widely used.

Sentiment analysis (SA)tells user whether the information about the product is satisfactory or not before they buy it. Marketers and firms use this analysis data to understand about their products or services in such a way that it can be offered as per the user's requirements. Textual Information retrieval techniques mainly focus on processing, searching or analysing the factual data present. Facts have an objective component but, there are some other textual contents which express subjective characteristics. These contents are mainly opinions, sentiments, appraisals, attitudes, and emotions, which form the core of Sentiment Analysis (SA). It offers many challenging opportunities to develop new applications, mainly due to the huge growth of available information on online sources like blogs and social networks. For example, recommendations of items proposed by a recommendation system can be predicted by taking into account considerations such as positive or negative opinions about those items by making use of SA.

Sentiment Analysis

Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service or idea. Sentiment analysis involves the use of data mining, machine learning (ML), artificial intelligence and computational linguistics to mine text for sentiment and subjective information such as whether it is expressing positive, negative or neutral feelings.

Sentiment analysis systems help organizations gather insights into real-time customer sentiment, customer experience and brand reputation. Generally, these tools use text analytics to analyze online sources such as emails, blog posts, online reviews, customer support tickets, news articles, survey responses, case studies, web chats, tweets, forums and comments. Algorithms are used to implement rule-based, automatic or hybrid methods of scoring whether the customer is expressing positive words, negative words or neutral ones.

In addition to identifying sentiment, sentiment analysis can extract the polarity or the amount of positivity and negativity, subject and opinion holder within the text. This approach is used to analyze various parts of text, such as a full document or a paragraph, sentence or subsentence.

Vendors that offer sentiment analysis platforms include Brandwatch, Critical Mention, Hootsuite, Lexalytics, Meltwater, MonkeyLearn, NetBase Quid, Sprout Social, Talkwalker and Zoho. Businesses that use these tools to analyze sentiment can review customer feedback more regularly and proactively respond to changes of opinion within the market.

Dataset Description

SMILE Twitter Emotion Dataset

This dataset is collected and annotated for the SMILE project http://www.culturesmile.org. This collection of tweets mentioning 13 Twitter handles associated with British museums was gathered between May 2013 and June 2015. It was created for the purpose of classifying emotions, expressed on Twitter towards arts and cultural experiences in museums.

Sentiment140 dataset with 1.6 million tweets

This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment.

Content:

It contains the following 6 fields:

target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)

ids: The id of the tweet (2087)

date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)

flag: The query (lyx). If there is no query, then this value is NO_QUERY.

user: the user that tweeted (robotickilldozr)

text: the text of the tweet (Lyx is cool)

Objective - 1

Two Classical Machine learning models such as Naive bayes, SVM, Decision tree etc.

Random Forest Algorithm

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.

CODE:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

model1 = RandomForestClassifier()
model1.fit(x_train, y_train)

y_pred = model1.predict(x_valid)

print("Training Accuracy :", model1.score(x_train, y_train))
print("Validation Accuracy :", model1.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("F1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

RESULT:

```
Training Accuracy : 0.9989987902048308
Validation Accuracy : 0.9534476285821549
F1 score : 0.6257545271629779
[[7308  100]
 [ 272  311]]
```

Decision Tree Classification Algorithm

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.

CODE:

```
from sklearn.tree import DecisionTreeClassifier

model3 = DecisionTreeClassifier()
model3.fit(x_train, y_train)

y_pred = model3.predict(x_valid)

print("Training Accuracy :", model3.score(x_train, y_train))
print("Validation Accuracy :", model3.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

RESULT:

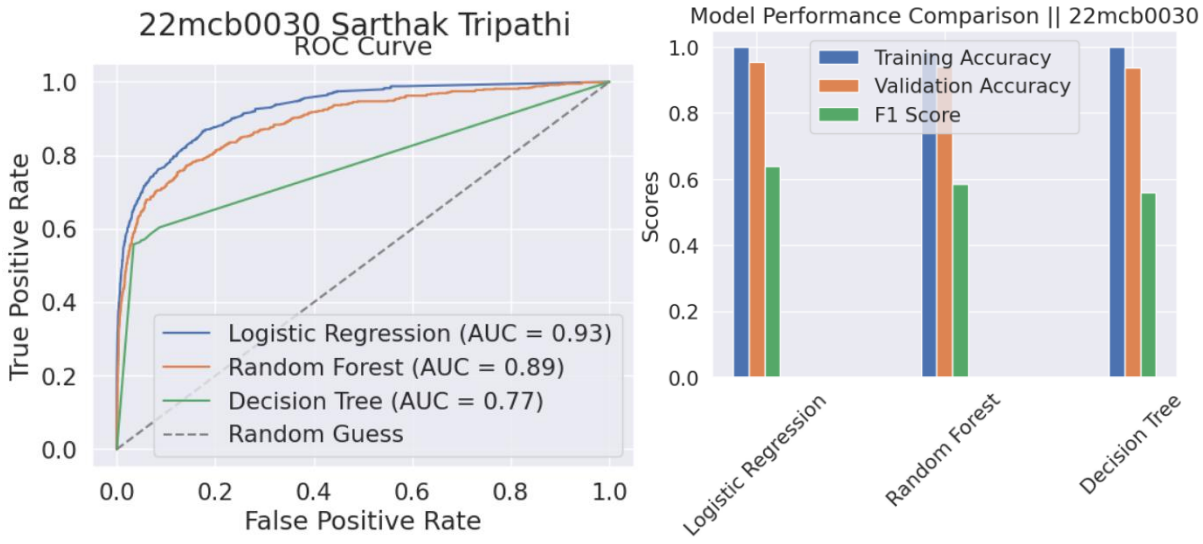
```
Training Accuracy : 0.9990405072796296
Validation Accuracy : 0.9365536228256789
f1 score : 0.5662959794696323
[[7153  255]
 [ 252  331]]
```

COMPARITIVE STUDY:

Here we perform comparative study of random forest classification and decision tree classification how they evaluated the results on the dataset.

```
# Calculate the ROC curve and AUC for each model
fpr1, tpr1, thresholds1 = roc_curve(y_valid,
model11.predict_proba(x_valid)[:, 1])
auc1 = auc(fpr1, tpr1)

fpr2, tpr2, thresholds2 = roc_curve(y_valid,
model2.predict_proba(x_valid)[:, 1])
auc2 = auc(fpr2, tpr2)

fpr3, tpr3, thresholds3 = roc_curve(y_valid,
model3.predict_proba(x_valid)[:, 1])
auc3 = auc(fpr3, tpr3)

# Plot the ROC curves and show the AUC values in the legend
plt.plot(fpr1, tpr1, label=f'{models[0]} (AUC = {auc1:.2f})')
plt.plot(fpr2, tpr2, label=f'{models[1]} (AUC = {auc2:.2f})')
plt.plot(fpr3, tpr3, label=f'{models[2]} (AUC = {auc3:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random
Guess')
```

RESULT:



Objective - 2

BI-LSTM with word2vec/ Fasttext word embedding

1. Using word2vec embedding

Word2Vec:

Word2Vec is a popular algorithm for generating word embeddings, which are dense vector representations of words in a continuous vector space. It captures semantic and syntactic relationships between words based on their co-occurrence patterns in a given corpus. There are two types of Word2Vec models: Continuous Bag-of-Words (CBOW) and Skip-gram. CBOW predicts the current word based on its context words, while Skip-gram predicts the context words given the current word. The resulting word embeddings from Word2Vec provide a numerical representation of words that can be used as inputs to various natural language processing tasks.

Bidirectional LSTM:

A BI-LSTM extends the LSTM architecture by incorporating two separate LSTM layers: one that processes the input sequence in a forward direction and another that processes it in a backward direction. The forward LSTM reads the input sequence from the beginning to the end, while the backward LSTM reads it in reverse. This bidirectional nature allows the model to capture both past and future contextual information for each word in the sequence. The outputs of the forward and backward LSTMs are concatenated to form the final output sequence, which provides a comprehensive representation of the input sequence with contextual information from both directions.

BI-LSTM with Word2Vec:

In the context of natural language processing tasks like text classification or named entity recognition, a BI-LSTM with Word2Vec combines the power of bidirectional modelling with pre-trained Word2Vec embeddings. The Word2Vec embeddings are used as the initial input representations for the BI-LSTM model. Each word in the input sequence is replaced by its corresponding Word2Vec embedding vector. These word embeddings are then fed as input to the BI-LSTM layers.

RESULT:

```
Accuracy
[0.8101337552070618, 0.8236578702926636, 0.8275212049484253, 0.8292667865753174, 0.8306871056556702]
[0.8301315903663635, 0.8363091945648193, 0.8367763161659241, 0.8376578688621521, 0.8394736647605896]
Loss
[0.4117514491081238, 0.3878689706325531, 0.38070905208587646, 0.3773415684700012, 0.37487462162971497]
[0.3777677118778229, 0.36764153838157654, 0.3645578622817993, 0.36553025245666504, 0.3612690567970276]
```

	precision	recall	f1-score	support
0	0.83	0.85	0.84	39989
1	0.85	0.83	0.84	40011
accuracy			0.84	80000
macro avg	0.84	0.84	0.84	80000
weighted avg	0.84	0.84	0.84	80000

2. Using Fasttext word embedding

FastText:

FastText is an extension of Word2Vec that introduces sub word information into word embeddings. Unlike Word2Vec, which represents each word as a single vector, FastText breaks words into smaller sub word units called character n-grams. It represents a word as the sum of the embeddings of its character n-grams, along with an additional representation for the entire word. This approach allows FastText to capture morphological information and handle out-of-vocabulary words more effectively compared to traditional word-level embeddings.

BI-LSTM with FastText:

When combining a BI-LSTM with FastText word embeddings, the FastText embeddings are utilized as the initial input representations for the BI-LSTM model. Each word in the input sequence is replaced by its corresponding FastText embedding vector. These word embeddings, which include sub word information, are then fed as input to the BI-LSTM layers.

During training, the parameters of the BI-LSTM layers are updated through backpropagation, allowing the model to learn task-specific features and representations from the input data. The bidirectional LSTM layers enable the model to capture dependencies between words in both forward and backward directions, facilitating a more comprehensive understanding of the overall context.

RESULT:

```
Accuracy
[0.8029385805130005, 0.8173033595085144, 0.8211461901664734, 0.8229078650474548, 0.8246527910232544]
[0.8264868259429932, 0.8308420777320862, 0.8329802751541138, 0.8345789313316345, 0.8352500200271606]
Loss
[0.4238697588443756, 0.3987767696380615, 0.3919205367565155, 0.3884671628475189, 0.38576337695121765]
[0.3845439851284027, 0.3753381669521332, 0.3724451363086700 4, 0.3709116578102112, 0.368075430393219]
```

	precision	recall	f1-score	support
0	0.83	0.84	0.84	39989
1	0.84	0.83	0.83	40011
accuracy			0.84	80000
macro avg	0.84	0.84	0.84	80000
weighted avg	0.84	0.84	0.84	80000

Algorithm	Training Accuracy	Validation Accuracy	F1 Score	Accuracy during training	Accuracy during validation	Loss during training	Loss during validation
Word2Vec Model	0.830	0.84	0.84	[0.81, 0.82, 0.83, 0.83, 0.83]	[0.83, 0.84, 0.84, 0.84, 0.84]	[0.41, 0.38, 0.38, 0.37, 0.37]	[0.38, 0.37, 0.37, 0.37, 0.37]
FastText Model	0.831	0.84	0.84	[0.81, 0.82, 0.83, 0.83, 0.83]	[0.83, 0.84, 0.84, 0.84, 0.84]	[0.41, 0.38, 0.38, 0.37, 0.37]	[0.38, 0.37, 0.37, 0.37, 0.37]

Objective - 3

Transformer based model with BERT-based word embedding.

A transformer-based model with BERT-based word embedding combines two powerful techniques in natural language processing (NLP): the transformer architecture and BERT (Bidirectional Encoder Representations from Transformers) word embeddings.

The transformer architecture is a neural network model that revolutionized NLP tasks by introducing a self-attention mechanism. It allows the model to focus on different parts of the input sequence during processing, capturing both local and global dependencies effectively. The transformer consists of an encoder-decoder structure, but in the case of BERT, only the encoder part is used for pre-training.

BERT, on the other hand, is a pre-training technique that learns word embeddings by training a transformer-based model on a large amount of unlabelled text data. It is "bidirectional" because it can consider both the left and right context of a word, which is a departure from traditional methods that only consider left-to-right or right-to-left context.

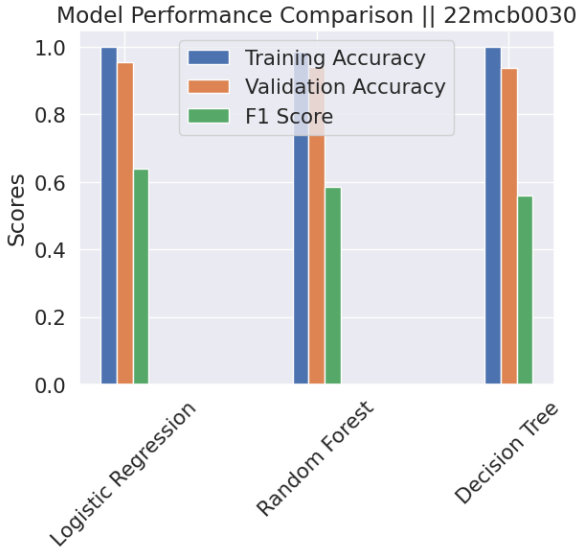
RESULT:

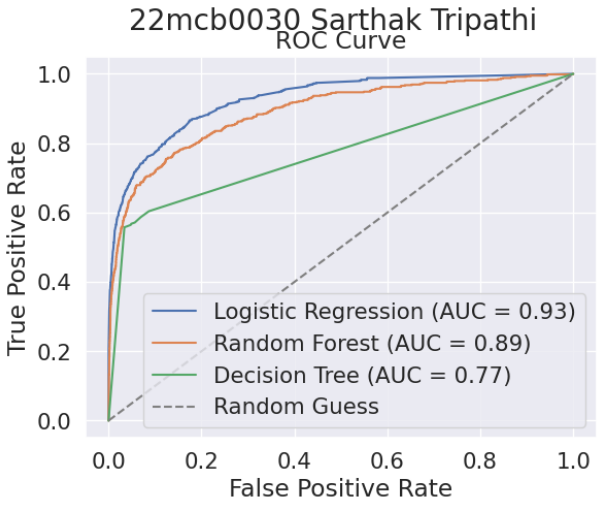
```
Validation loss: 0.5470880227429527
F1 Score (weighted): 0.796168106548527

Epoch {epoch}
Training loss: 0.41684663620969603
100% 7/7 [00:03<00:00, 1.97it/s]
Validation loss: 0.5726030554090228
F1 Score (weighted): 0.8245902369003772

Epoch {epoch}
Training loss: 0.30569675548504743
100% 7/7 [00:03<00:00, 1.96it/s]
Validation loss: 0.5563450115067619
F1 Score (weighted): 0.831301512604119

Epoch {epoch}
Training loss: 0.2213290510255666
100% 7/7 [00:03<00:00, 1.96it/s]
Validation loss: 0.5847831943205425
F1 Score (weighted): 0.8378261225031274
```

```
[109] accuracy_per_class(predictions, true_vals )

    Class: happy
    Accuracy:166/171

    Class: not-relevant
    Accuracy:19/32

    Class: angry
    Accuracy:7/9

    Class: disgust
    Accuracy:0/1

    Class: sad
    Accuracy:0/5

    Class: surprise
    Accuracy:2/5
```

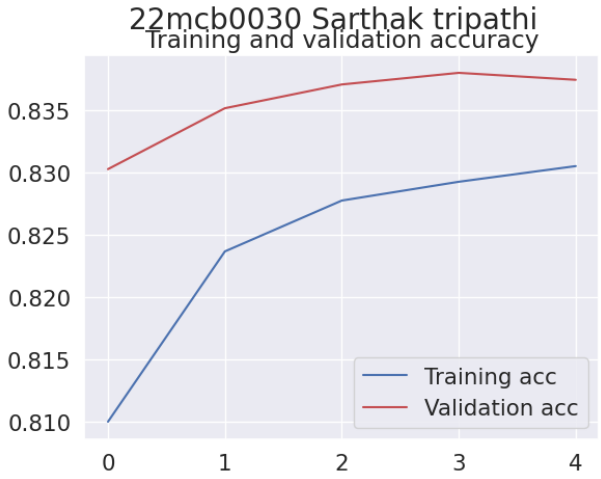
COMAPARITIVE RESULT ANALYSIS

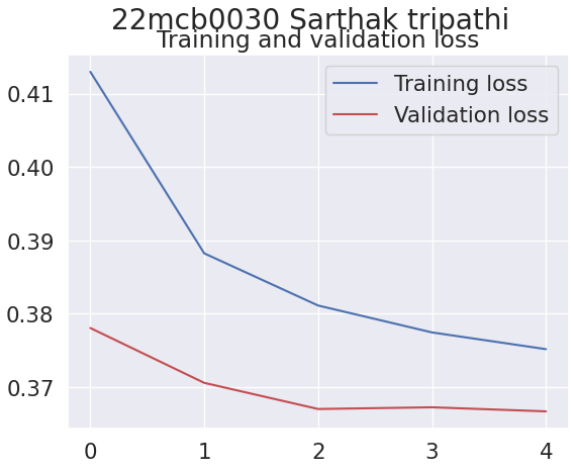
Comparative analysis of logistic regression, random forest and decision tree classification:



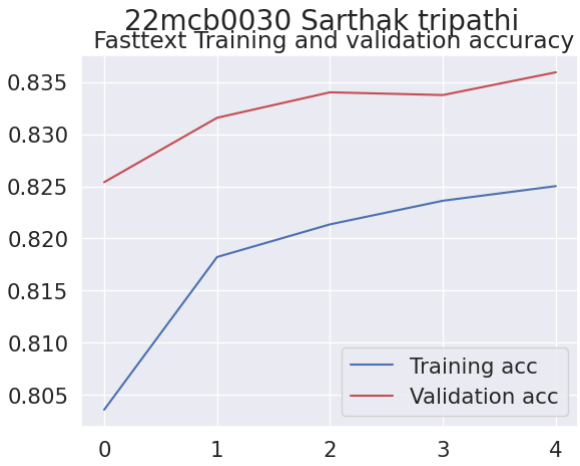


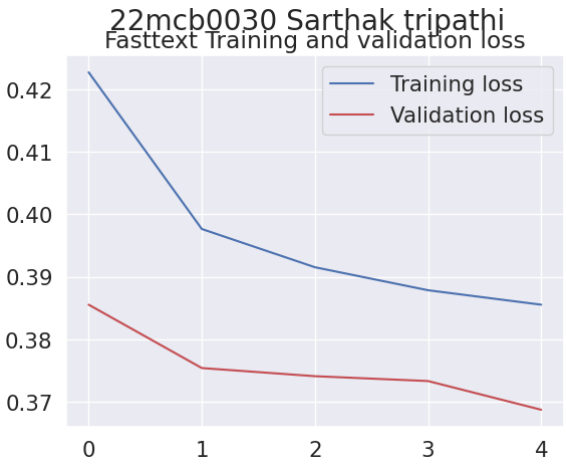
Word2vec Analysis



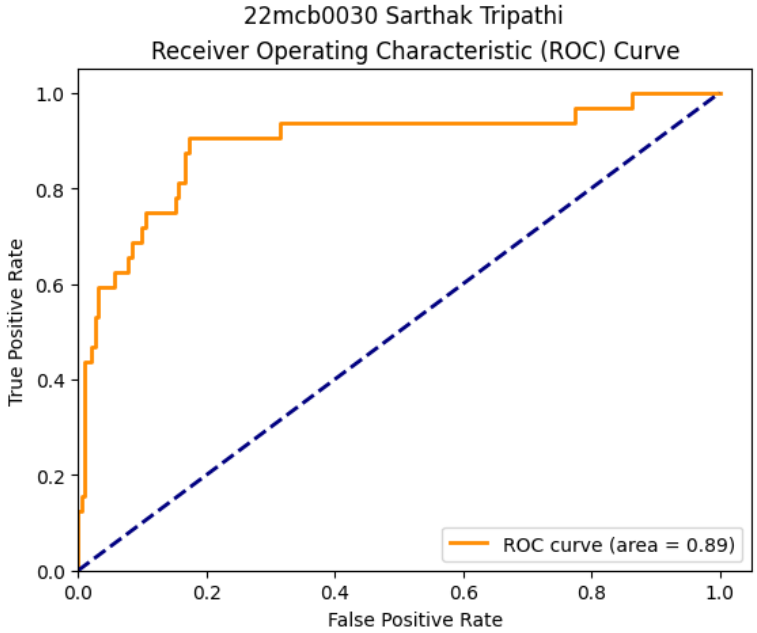


Fasttext Analysis

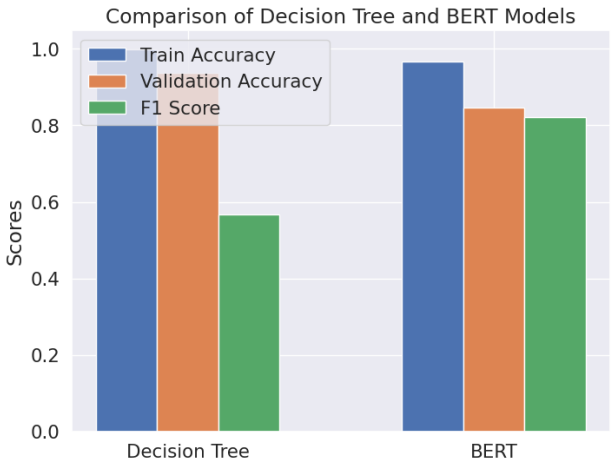


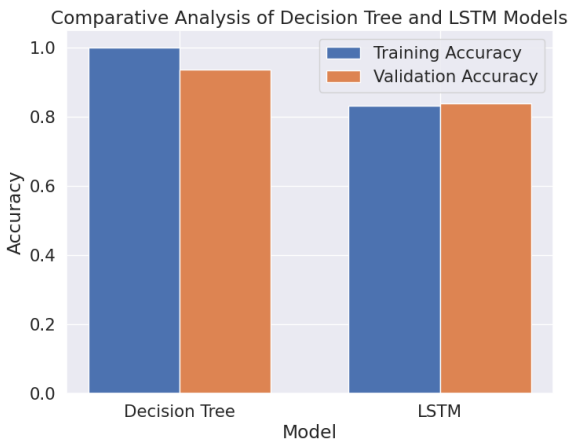


BERT Analysis

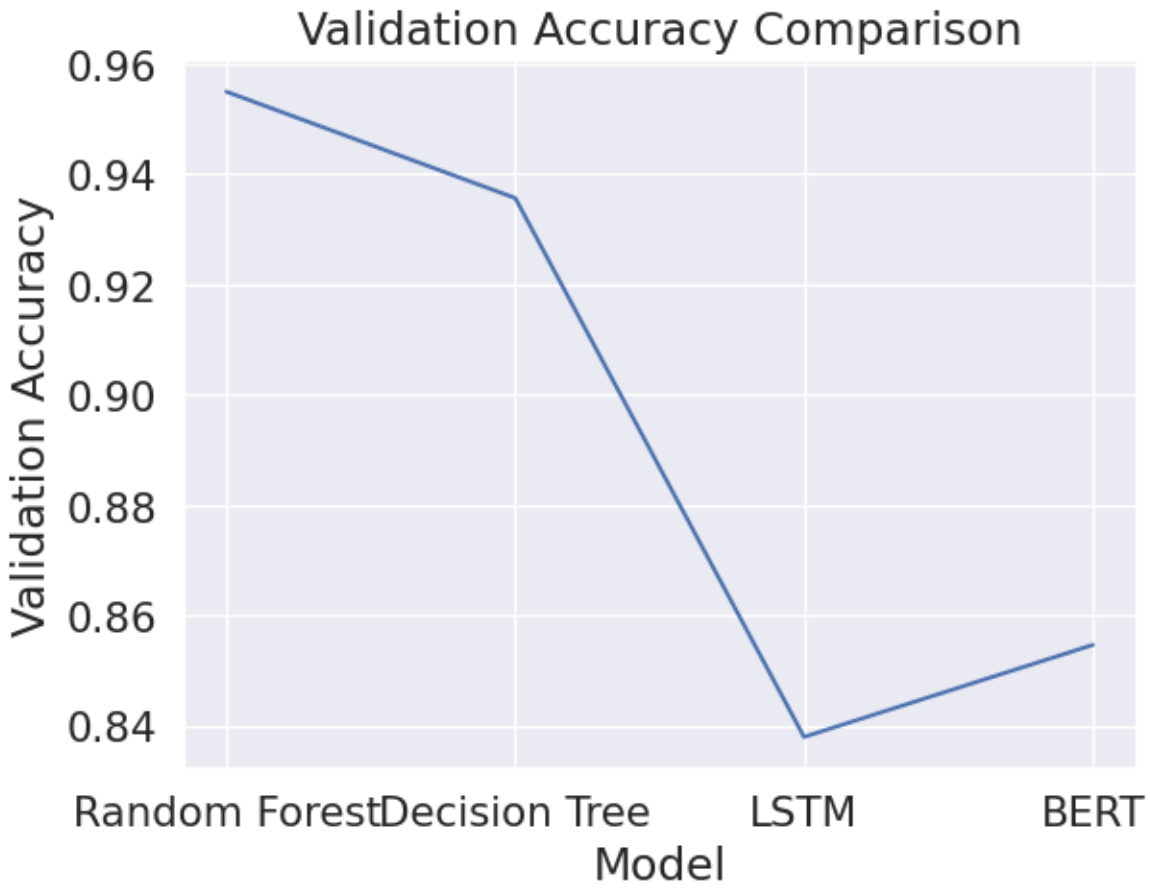


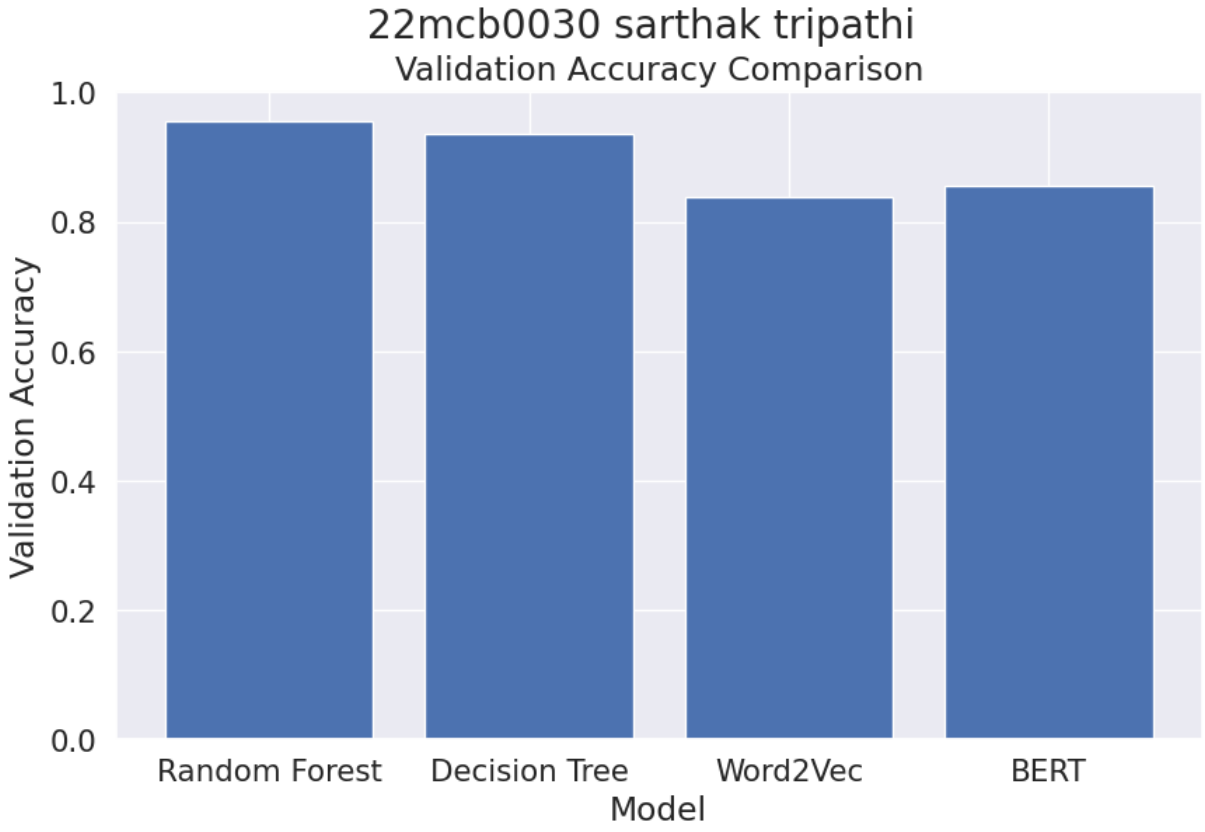
Now let's compare the result of different models:





Now for the final conclusion let's see the results of final comparison of these models:





Conclusion

The results obtained from the different models, it can be concluded that the random forest regressor has the highest training accuracy of 0.999 and validation accuracy of 0.955, indicating that the model is performing well on both the training and validation datasets. The F1 score of 0.637 suggests that the model is moderately effective in predicting the target variable.

Similarly, the decision tree classifier also has a high training accuracy of 0.999 and a validation accuracy of 0.936, indicating good performance on both datasets. However, the F1 score of 0.563 indicates that the model is less effective than the random forest regressor in predicting the target variable.

Moving on to the NLP models, both word2vec and fasttext models have similar precision, recall, and f1-scores for both classes, demonstrating that they are equally effective in classifying text data with an overall accuracy of 84%.

In the third Objective, the use of BERT model resulted in a validation loss of 0.556 and a weighted F1 score of 0.831, indicating that the model performs reasonably well in predicting the target variable, but not as accurately as the random forest model.

Overall, the choice of the best model would depend on the specific application and the importance of accuracy, speed, and interpretability. The random forest model may be preferred if a high level of accuracy is crucial, while the NLP models may be more suitable for text classification tasks.

Following is the table summerizing the results:

Algorithm	Training Accuracy	Validation Accuracy	F1 Score
Random Forest Regressor	0.999	0.955	0.637
Decision Tree Classifier	0.999	0.936	0.563
Word2Vec Model	0.830	0.84	0.84
FastText Model	0.831	0.84	0.84
BERT Model	0.970	0.911	0.831