NLP-HOTEL REVIEWS

Under the guidelines

Advith
hareesh

Project memebers

Muddangula.Shirisha
Yenneti.Lekha Sree
Atike.Bhadralaxmi
Barla.Akshitha
Karavadi.Jaya Lakshmi
Dappu.Vaishnavi
Gorantla.Hema Latha

Content

Business Objective

The goal of this project is to build unsupervised Natural Language Processing (NLP) machine learning models that decide whether a text review is positive review or negative review. This project, will help hotels to determine the category of text review and cluster them automaticity to improve their services.

EDA

- Exploratory data analysis (EDA) is an approach to analyzing and understanding a dataset. It generally includes visualizing the data, summarizing the main characteristics, and identifying patterns and relationships within the data.
- EDA is typically the first step in the data analysis process, and it is an important way to gain insights and identify potential issues before building models or making predictions.

```
hotel.info()
```

executed in 58ms, finished 18:38:03 2023-02-06

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20491 entries, 0 to 20490
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  20491 non-null  object
 1   Rating  20491 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 320.3+ KB
```

```
hotel.shape  # No.of rows and columns present in our dataset
```

executed in 11ms, finished 18:15:57 2023-01-24

```
(20491, 2)
```

```
hotel.size  # Its shows size of our dataset
```

executed in 10ms, finished 18:15:57 2023-01-24

```
40982
```

- To find the null values in the data set ,null value present drop the null value or replace th null values
- Check Duplicated values in the data set
- In case present the duplicated values drop the duplicated values

```
[6]: hotel.describe(include='O')                                    #Sta
```
executed in 43ms, finished 18:15:57 2023-01-24

[6]:

	Review
count	20491
unique	20491
top	nice hotel expensive parking got good deal sta...
freq	1

```
[6]: hotel.describe().T
```
executed in 41ms, finished 18:15:57 2023-01-24

[6]:

	count	mean	std	min	25%	50%	75%	max
Rating	20491.0	3.952223	1.23303	1.0	3.0	4.0	5.0	5.0

```
sorted(hotel['Rating'].unique()) #Unique
```
executed in 12ms, finished 18:15:57 2023-01-24

```
[1, 2, 3, 4, 5]
```

```
hotel['Rating'].value_counts()  #Count
```
executed in 26ms, finished 18:15:57 2023-01-24

```
5    9054
4    6039
3    2184
2    1793
1    1421
Name: Rating, dtype: int64
```

There is no null values and dulicate values

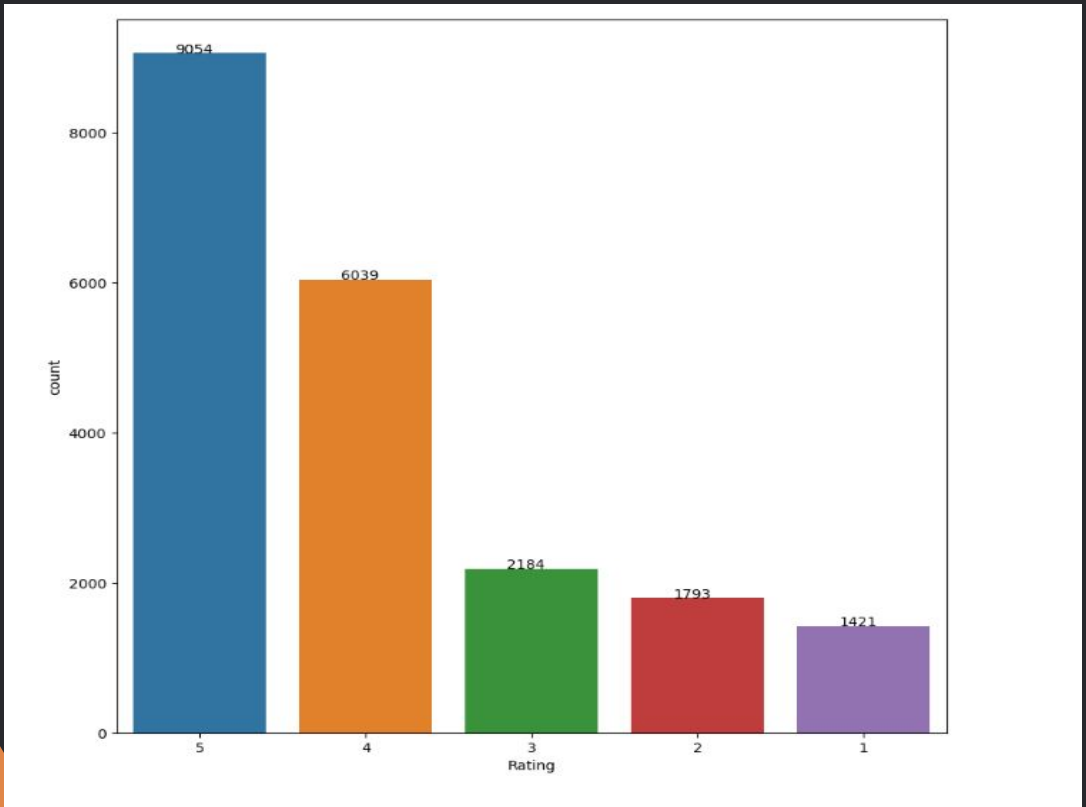
```
hotel.nunique() #
```
executed in 122ms, finished 18:15:57 2023-01-24

```
Review    20491
Rating        5
dtype: int64
```

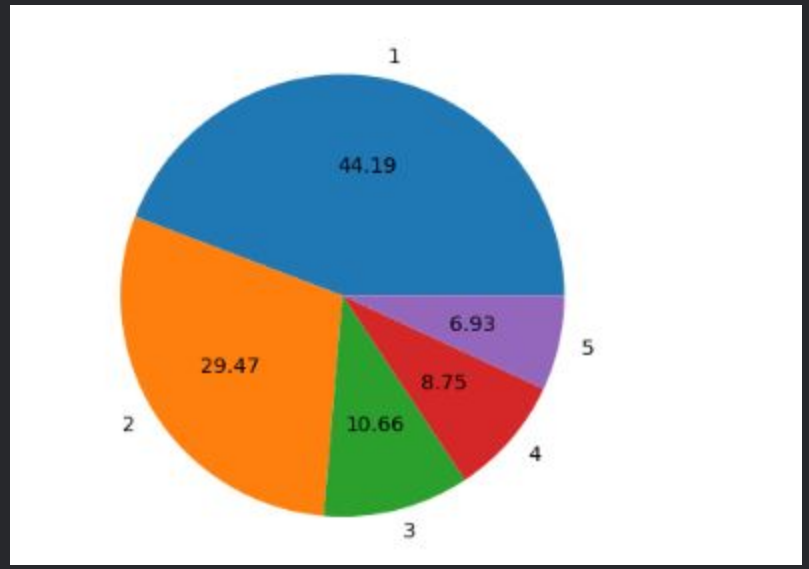
```
hotel.isnull().sum() #Checking the null values
```
executed in 27ms, finished 18:15:57 2023-01-24

```
Review    0
Rating    0
dtype: int64
```

Data Visualization



countplot() method is used to Show the counts of observations in each categorical bin using bars.

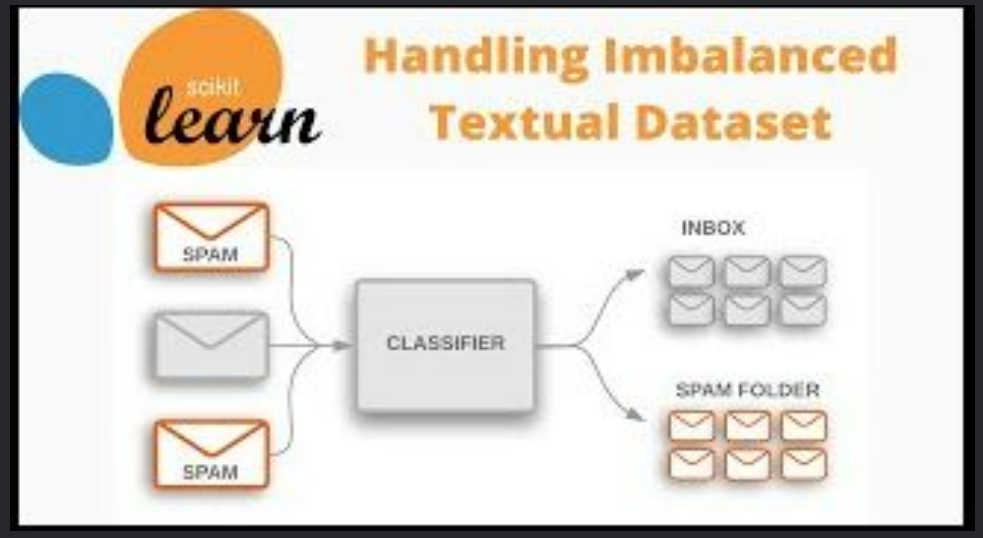


A pie chart helps organize and show data as a percentage of a whole.

Data Pre-processing

The preprocessing steps for text data are different from the traditional data preprocessing for structured data. Some common NLP data preprocessing techniques include

- ❑Tokenization: Breaking down the text into individual words or phrases (tokens)
- ❑Stop-word removal: Removing common words like "a", "an", "the", etc. that do not carry meaning
- ❑Stemming or Lemmatization: Reducing words to their base form to reduce the dimensionality of the data
- ❑Removing punctuation and special characters
- ❑Lowercasing or uppercasing all text
- ❑Removing numbers
- ❑Removing whitespaces
- ❑Removing HTML tags
- ❑Replacing synonyms or words with similar meaning



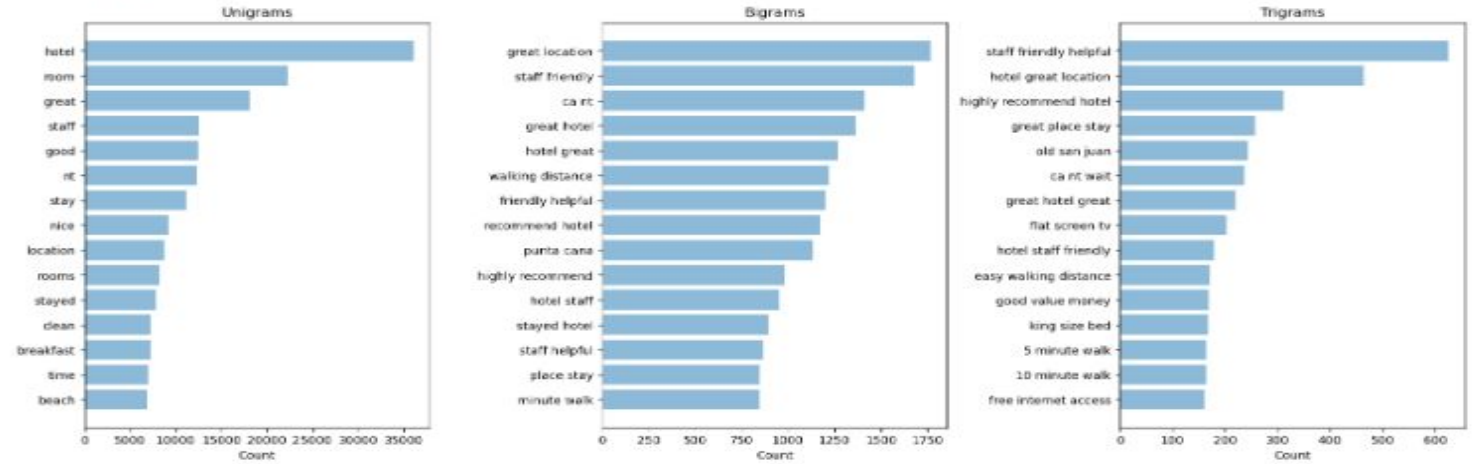
N-gram Analysis - Bigram and Trigram

Sentiment analysis of Bigram/Trigram :

N-grams analyses are often used to see which words often show up together. I often like to investigate combinations of two words or three words, i.e., Bigrams/Trigrams. An n-gram is a contiguous sequence of n items from a given sample of text or speech.

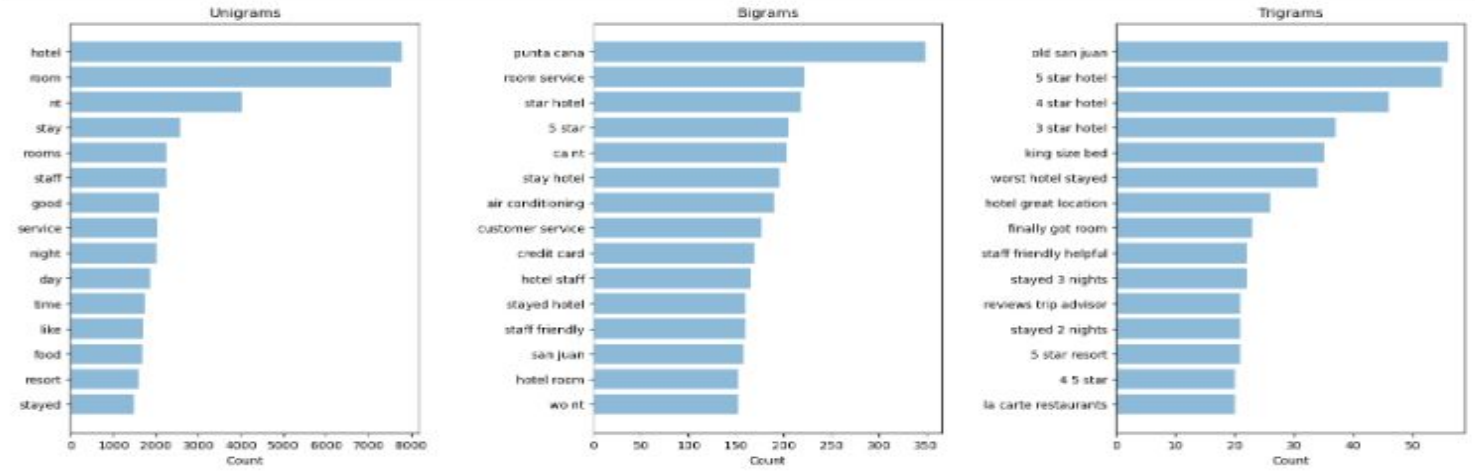
```
In [25]: textTrends(posReviews['Review'])
```

executed in 7m 46s, finished 18:45:54 2023-02-06




```
In [26]: textTrends(negReviews['Review'])
```

executed in 58.7s, finished 18:46:53 2023-02-06




```
In [27]: # DATA PREPROCESSING
```

Removing STOP-WORDS , EXTRA WHITE-SPACE & PUNCTUATIONS

```
#Removing stop words function
def remove_stopwords(text):
    removed = []
    stop_words = list(stopwords.words("english"))
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        if tokens[i] not in stop_words:
            removed.append(tokens[i])
    return " ".join(removed)
```

executed in 13ms, finished 18:46:57 2023-02-06

```
#Remove extra white spaces function
def remove_extra_white_spaces(text):
    single_char_pattern = r'\s+[a-zA-Z]\s+'
    without_sc = re.sub(pattern=single_char_pattern, repl=" ", string=text)
    return without_sc
```

executed in 13ms, finished 18:46:57 2023-02-06

```
#Removing punctuations function
def remove_punctuation(text):
    return text.translate(str.maketrans('', '', string.punctuation))
```

executed in 13ms, finished 18:46:57 2023-02-06

```
#Removing numbers and other numeric values function
def remove_numbers(text):
    number_pattern = r'\d+'
    without_number = re.sub(pattern=number_pattern, repl=" ", string=text)
    return without_number
```

executed in 14ms, finished 18:46:56 2023-02-06

- Stop word removal is one of the most commonly used preprocessing steps across different NLP applications.
- The basic approach is to use the lstrip() function from the inbuilt python string library. The function lstrip() removes any unnecessary spaces .
- One of the easiest ways to remove punctuation from a string in Python is to use the str. translate() method.
- Python provides a regex module that has a built-in function sub() to remove numbers from the string. This method replaces all the occurrences of the given pattern in the string with a replacement string

Lemmatization

Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.

```
#Lemmatizing function
def lemmatizing(text):
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(text)
    for i in range(len(tokens)):
        lemma_word = lemmatizer.lemmatize(tokens[i])
        tokens[i] = lemma_word
    return " ".join(tokens)
```

executed in 14ms, finished 18:46:58 2023-02-06

Word Cloud

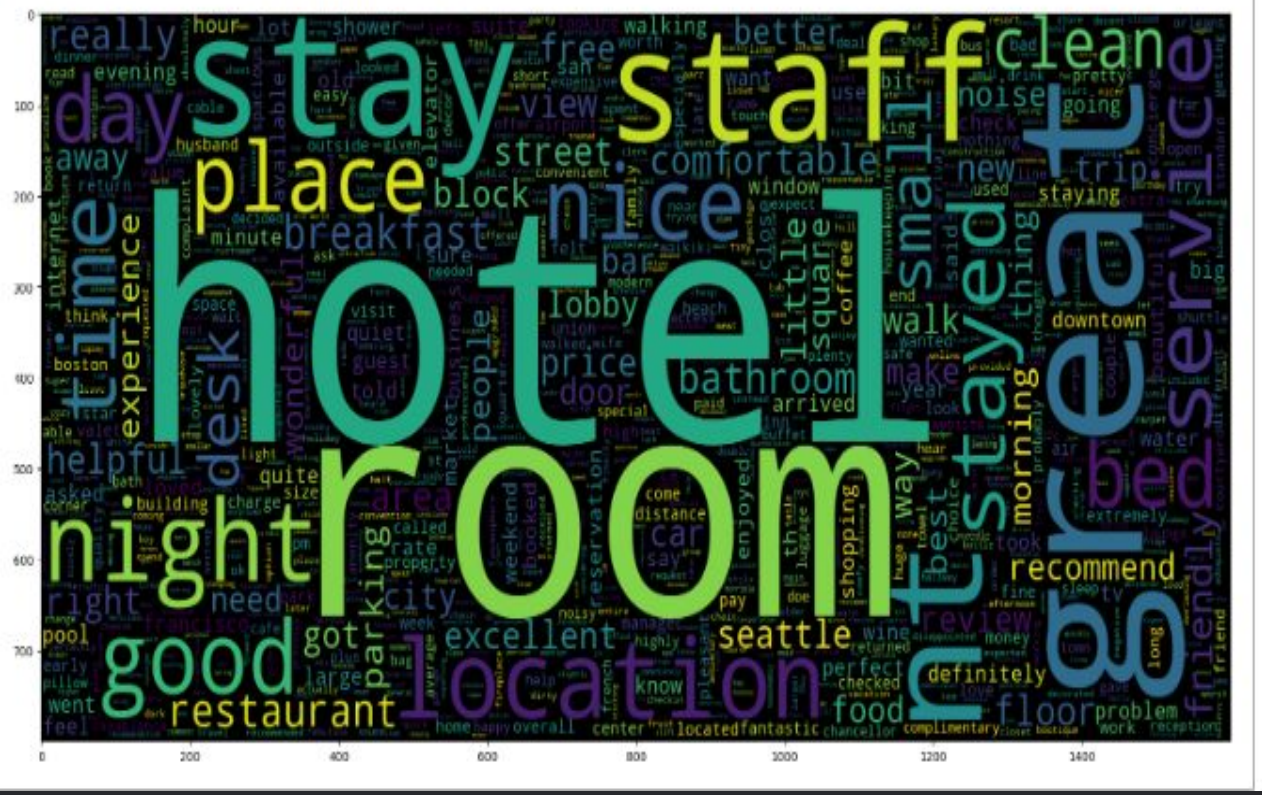
- A word cloud, also known as a tag cloud, is a data visualization technique used to display the most frequently occurring words in a text corpus.
- The words are typically displayed in different sizes and font weights, with the most frequent words appearing larger and bolder than the less frequent words.
- Word clouds are often used in text mining, content analysis, and sentiment analysis applications.

Positive & Negative reviews (Word Cloud)

```
# Plot a cloud of words for hotel positive reviews

hotel_positive = hotel['Review'][:1642]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(hotel_positive))
plt.imshow(wc)
```

executed in 9.49s, finished 18:47:24 2023-02-06

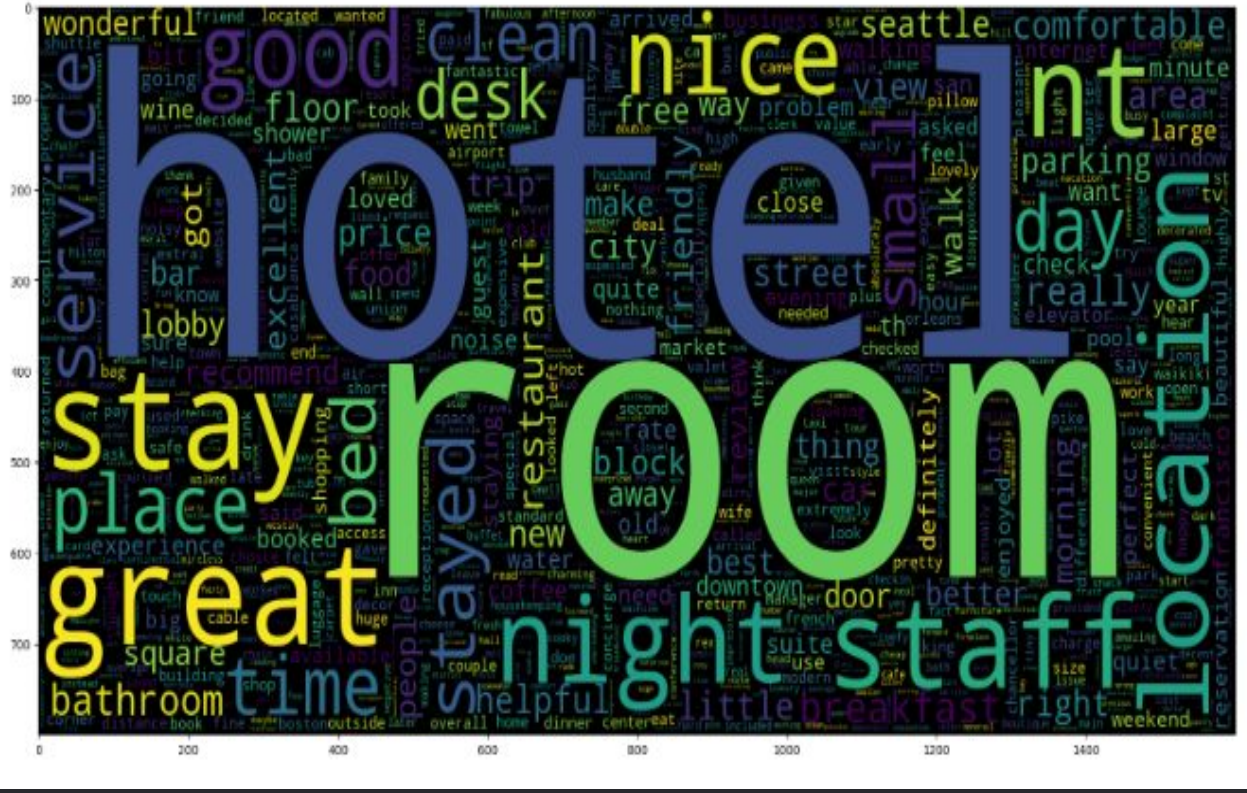
```
<matplotlib.image.AxesImage at 0x21bf6620a30>
```




```
# Plot a cloud of words for hotel negetavie reviews

hotel_negative = hotel['Review'][:1642]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(hotel_negative))
plt.imshow(wc)
```

executed in 9.35s, finished 18:47:33 2023-02-06

```
<matplotlib.image.AxesImage at 0x21bfb689310>
```



Feature extraction

In Natural Language Processing (NLP), feature extraction refers to the process of extracting relevant information from text data in order to represent it in a form that can be used by machine learning models. There are several common feature extraction techniques used in NLP, such as

Some common feature extraction techniques for text data are:

- Bag of Words
- TF-IDF
- Word2Vec
- GloVe
- BERT

```
#Tf-idf vectorizer
tf=TfidfVectorizer(ngram_range=(1,2))
#transformed train emails
tf_train=tf.fit_transform(X_train)
#transformed test emails
tf_test=tf.transform(X_test)
print('Tfidf_train:',tf_train.shape)
print('Tfidf_test:',tf_test.shape)
```

executed in 7.27s, finished 18:47:40 2023-02-06

```
Tfidf_train: (18441, 907819)
Tfidf_test: (2050, 907819)
```

```
# countervoctorizer(bog of words)
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
Bow=CountVectorizer(min_df=0,max_df=1,binary=False,ngram_range=(1,2))
#transformed train emails
Bow_train=Bow.fit_transform(X_train)
#transformed test emails
Bow_test=Bow.transform(X_test)
print('BOW_cv_train:',Bow_train.shape)
print('BOW_cv_test:',Bow_test.shape)
```

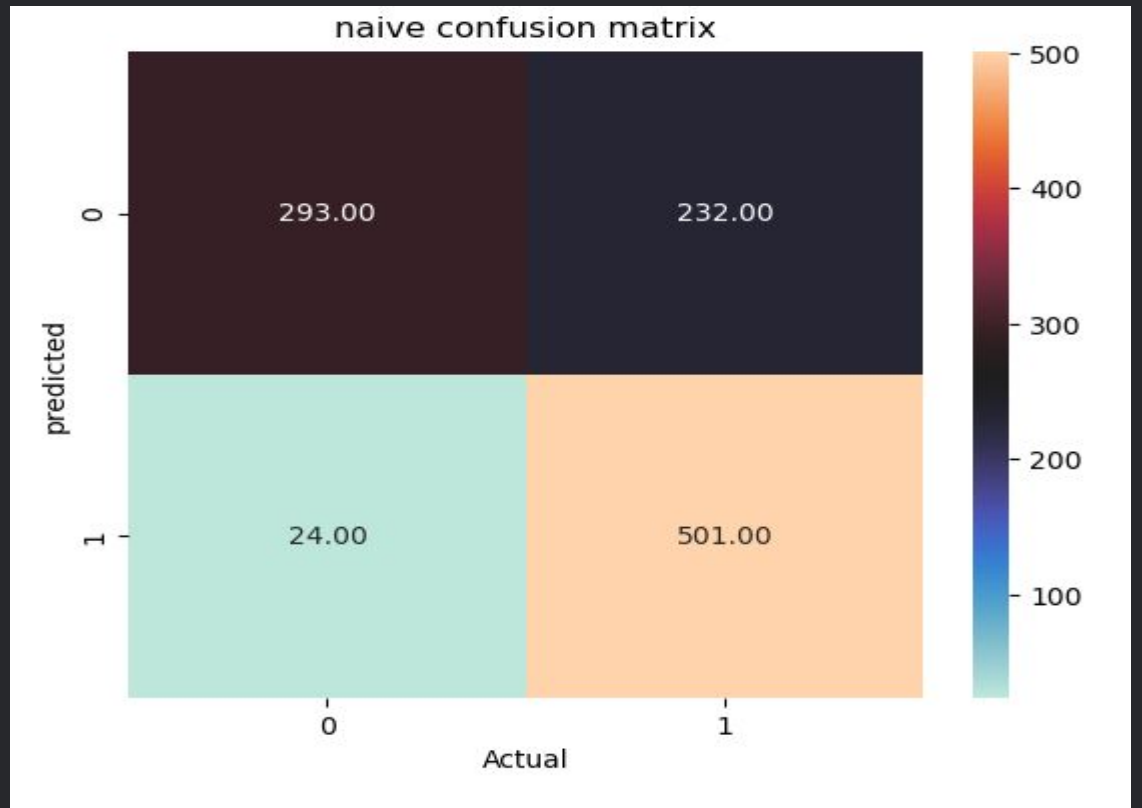
executed in 6.58s, finished 18:47:47 2023-02-06

```
BOW_cv_train: (18441, 698116)
BOW_cv_test: (2050, 698116)
```

Naive Bayes model

- Naive Bayes is a probabilistic algorithm that is based on Bayes' theorem. It is a simple yet powerful technique for classification tasks.
- Naive Bayes assumes that all the features of the data are independent of each other, which is called class conditional independence. Naive Bayes algorithm is commonly used in text classification and spam filtering, it's also useful in problems where the number of features is greater than the number of observations.

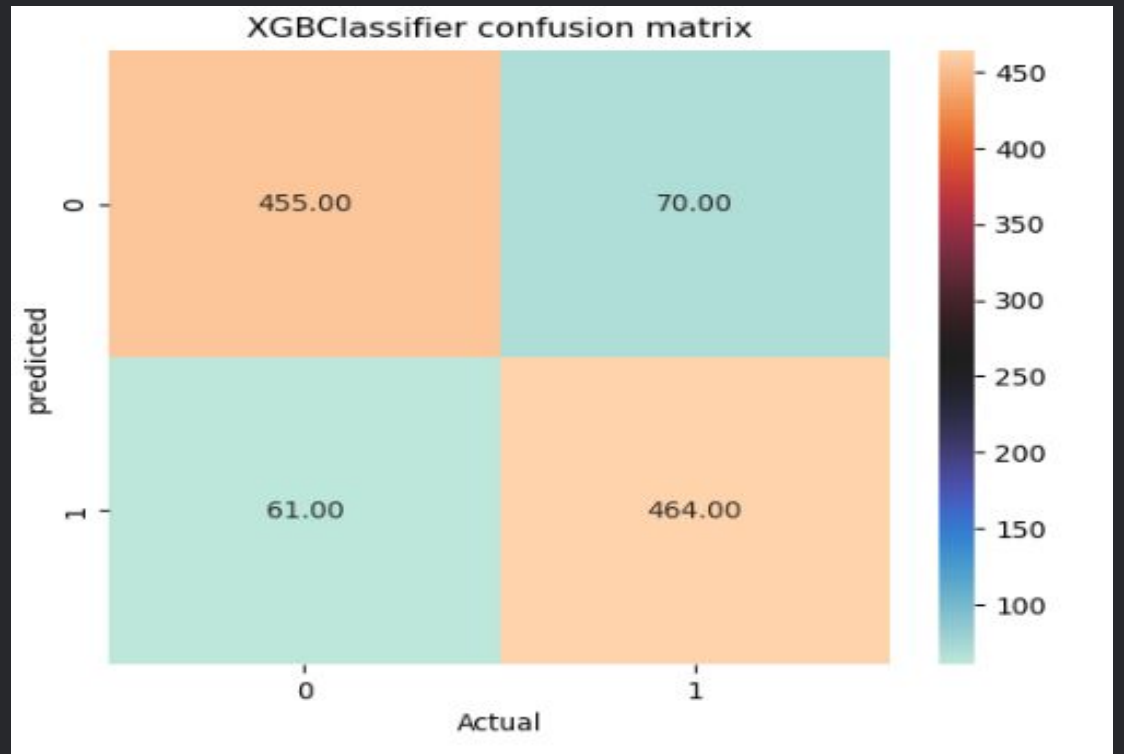
	precision	recall	f1-score	support
0	0.92	0.56	0.70	525
1	0.68	0.95	0.80	525
accuracy			0.76	1050
macro avg	0.80	0.76	0.75	1050
weighted avg	0.80	0.76	0.75	1050



XG boostclassifier model

- XGBoost (Extreme Gradient Boosting) is an optimized version of Gradient Boosting algorithm. It is an open-source library that is widely used for supervised learning, mainly for classification and regression problems. It was specifically designed to improve the computational efficiency and model performance of Gradient Boosting.
- It was specifically designed to improve the computational efficiency and model performance of Gradient Boosting.

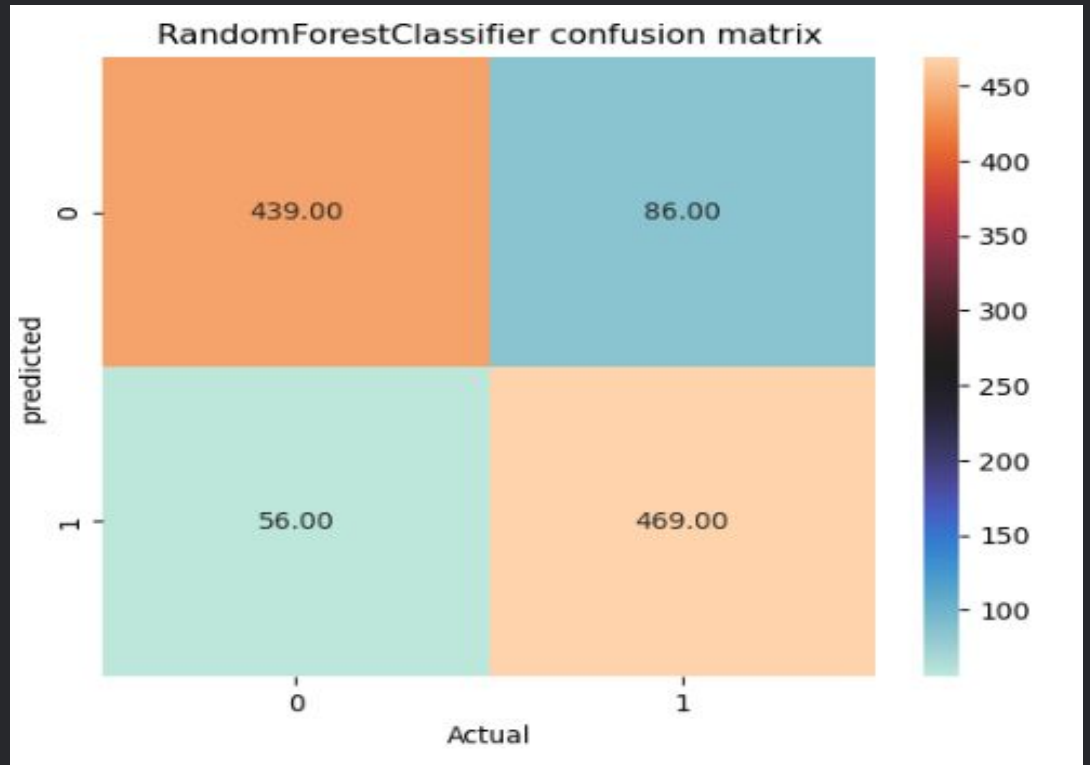
	precision	recall	f1-score	support
0	0.88	0.87	0.87	525
1	0.87	0.88	0.88	525
accuracy			0.88	1050
macro avg	0.88	0.88	0.88	1050
weighted avg	0.88	0.88	0.88	1050



Randomforest classifier model

- Random Forest is an ensemble learning method that uses multiple decision trees to make predictions.
- It is a type of supervised learning algorithm, typically used for classification and regression tasks.
- The basic idea behind Random Forest is to combine the predictions of multiple decision trees to reduce overfitting and improve the overall performance of the model.

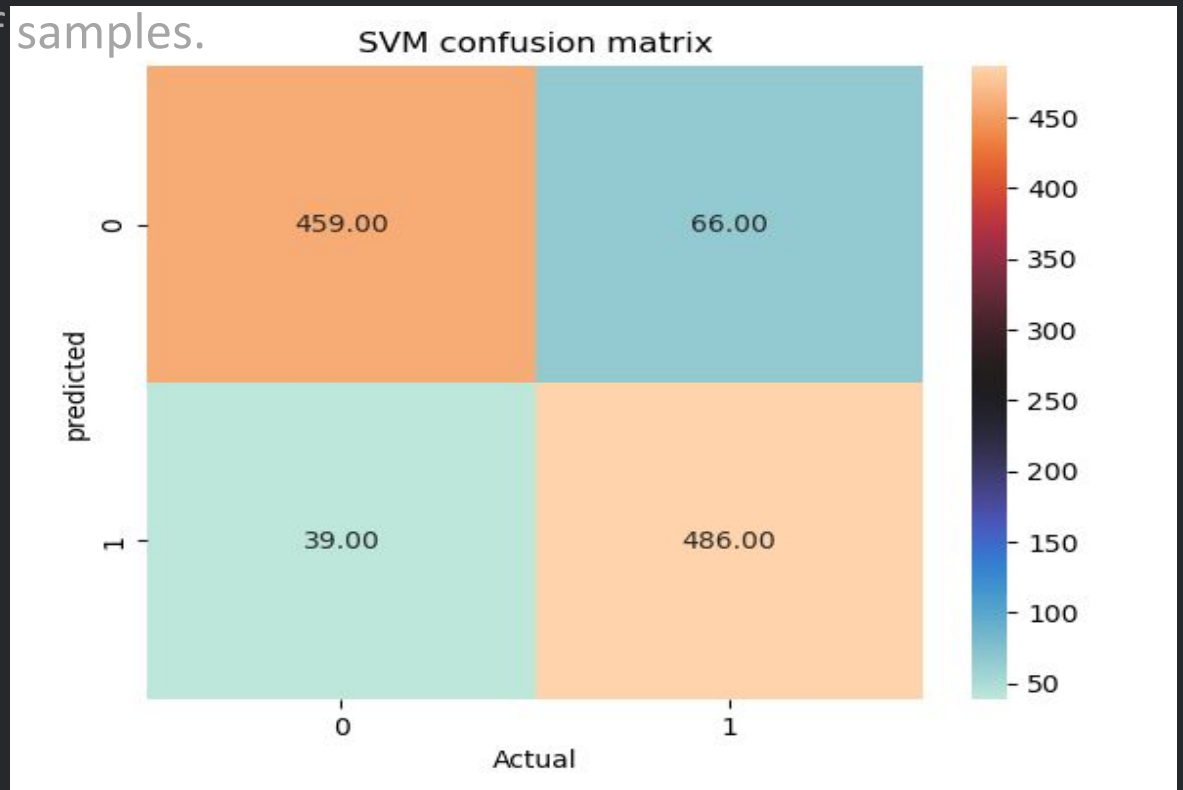
	precision	recall	f1-score	support
0	0.89	0.84	0.86	525
1	0.85	0.89	0.87	525
accuracy			0.86	1050
macro avg	0.87	0.86	0.86	1050
weighted avg	0.87	0.86	0.86	1050



Support Vector Machine(SVM)

- Support Vector Machine (SVM) is a supervised learning algorithm that can be used for classification or regression tasks. The idea behind SVM is to find the hyperplane in a high-dimensional space that maximally separates the different classes. The data points closest to the hyperplane are called support vectors and have the greatest impact on the position of the hyperplane.
- SVM uses a technique called kernel trick to transform the data into a higher dimensional space where a linear hyperplane can be used for classification. It is particularly useful for problems where the number of dimensions is greater than the number of samples.

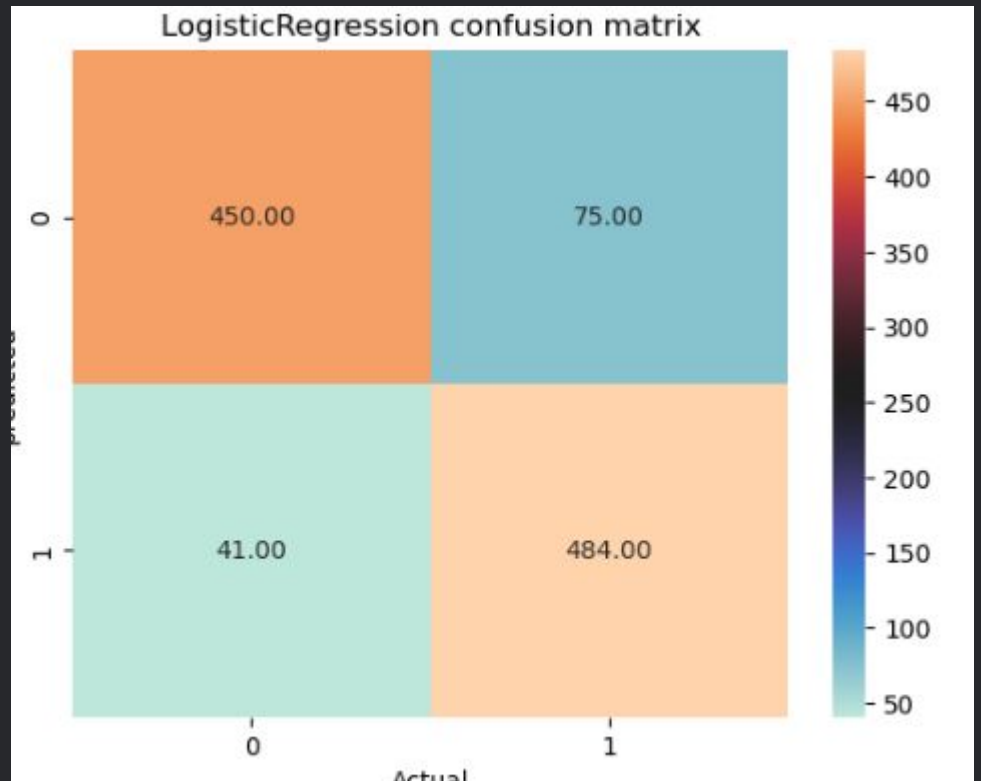
	precision	recall	f1-score	support
0	0.92	0.87	0.90	525
1	0.88	0.93	0.90	525
accuracy			0.90	1050
macro avg	0.90	0.90	0.90	1050
weighted avg	0.90	0.90	0.90	1050



Logistic Regression

- Logistic Regression is a statistical method used for binary classification problems, where the goal is to predict a binary outcome based on input features.
- It uses a logistic function to model the probability of the target variable taking on a particular value and is trained using labeled data to find the optimal coefficients that maximize the likelihood of the observed data. It's simple, interpretable and has been widely used in many industries.

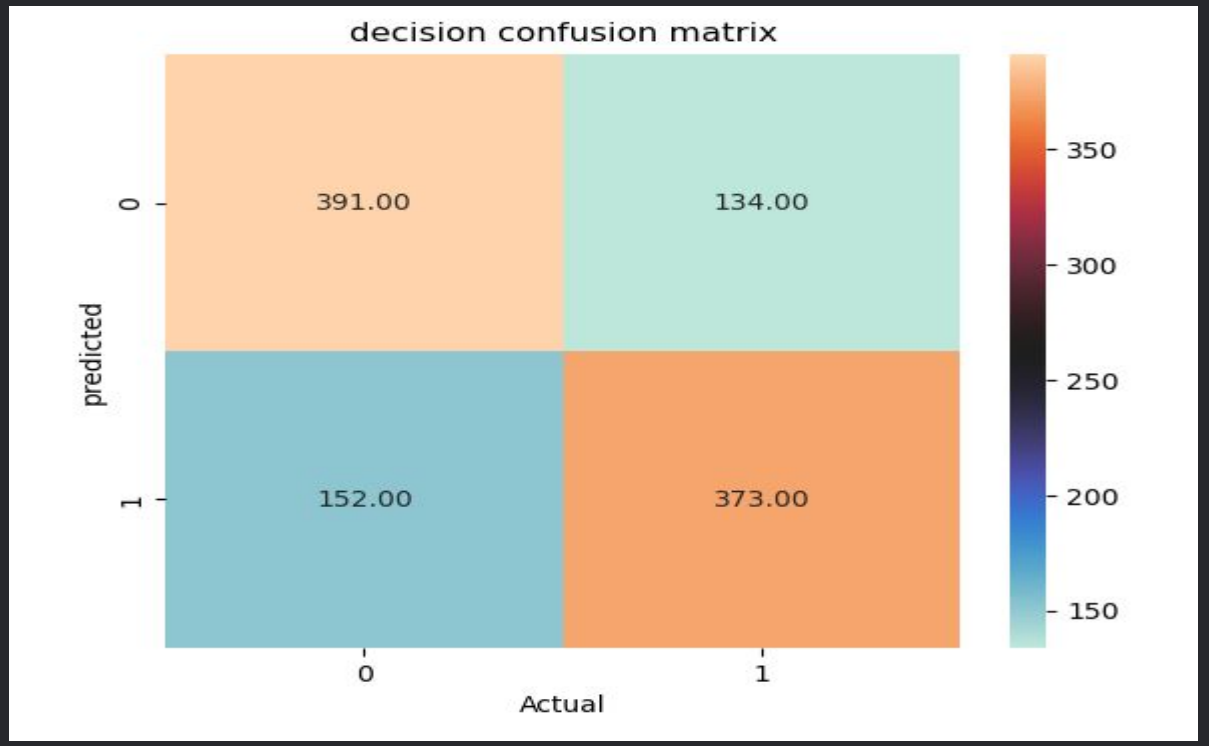
	precision	recall	f1-score	support
0	0.92	0.86	0.89	525
1	0.87	0.92	0.89	525
accuracy			0.89	1050
macro avg	0.89	0.89	0.89	1050
weighted avg	0.89	0.89	0.89	1050



DecisionTreeClassifier Model

- The main advantage of the decision tree classifier is its ability to using different feature subsets and decision rules at different stages of classification.
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

	precision	recall	f1-score	support
0	0.72	0.74	0.73	525
1	0.74	0.71	0.72	525
accuracy			0.73	1050
macro avg	0.73	0.73	0.73	1050
weighted avg	0.73	0.73	0.73	1050



Conclusion:

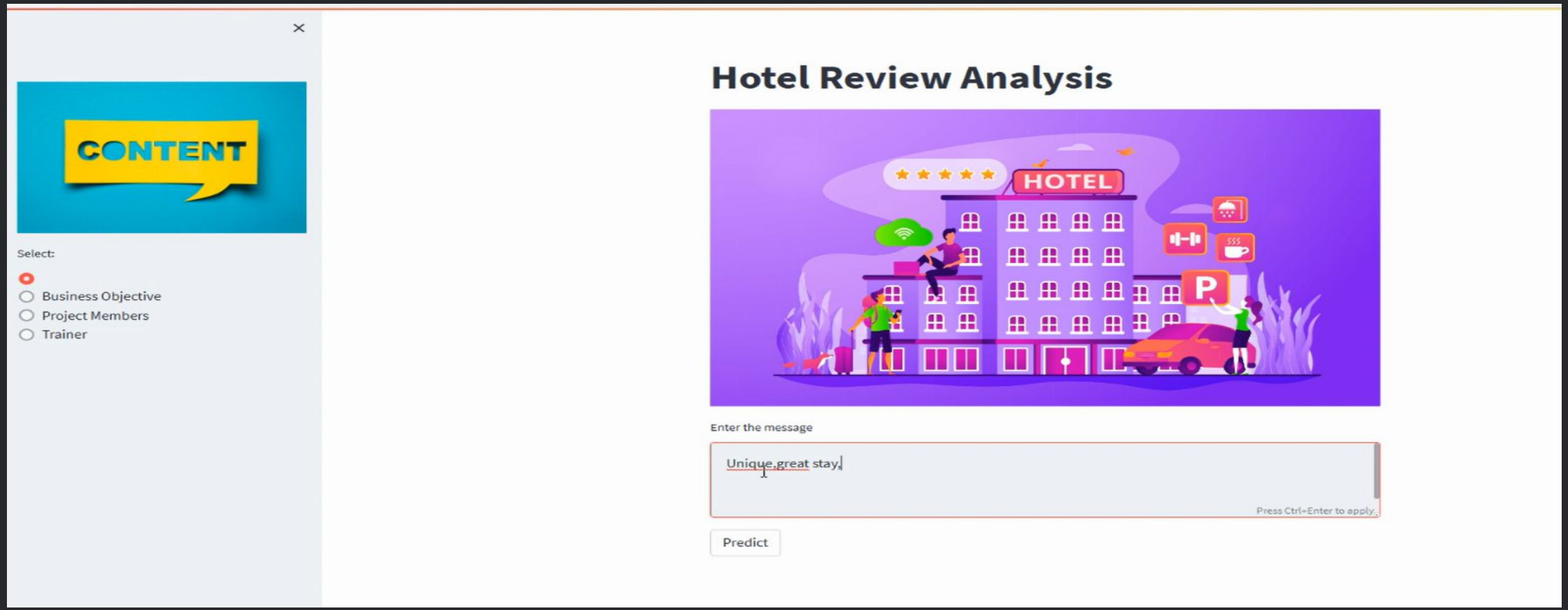
- When comparing the accuracy of different models, the model with the highest accuracy is usually a good choice for deployment. However, it's important to consider other evaluation metrics, such as F1-score, precision, and confusion matrix, as they provide a more complete picture of the model's performance.
- If two models have similar accuracy, the one with the higher F1-score is often a better choice, as it takes into account both precision and recall and provides a better overall picture of the model's performance. Additionally, you should examine the confusion matrix to ensure that the model is making appropriate predictions for all classes in the dataset. Ultimately, the best model for deployment will depend on the specific use case and the desired trade-off between accuracy, precision, recall, and other factors.

DEPLOYMENT

```python
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

# data preprossing


def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()

    # Tokenize
    tokens = nltk.word_tokenize(text)

    # Remove non-alphanumeric characters
    tokens = [token for token in tokens if token.isalnum()]

    # Remove stop words and punctuation
    stop_words = set(stopwords.words('english'))
    tokens = [token for token in tokens if token not in stop_words and token not in string.punctuation]

    # Stem the tokens
    stemmed_tokens = [PorterStemmer().stem(token) for token in tokens]

    # Join the processed tokens into a single string
    processed_text = " ".join(stemmed_tokens)

    return processed_text


# load the pickle files
tfidf_vectorizer = pickle.load(open('vectorizer.pkl','rb'))
model = pickle.load(open('model_svm.pkl','rb'))

st.title("Hotel Review Analysis")
st.image('https://uploads-ssl.webflow.com/60fd4503684b46390cc0d337/6346af0655b9625759e278ec_01.png')
st.sidebar.image('https://media.istockphoto.com/id/507474468/photo/banner-with-the-phrase-cut.jpg?s=612x612&w=0&k=20&c=JPlwsVUZ0y4pdOUzGsEAUIF4m40O2G0ltbrHzmI4yck=')

# Create a radio button to select a value
selected_value = st.sidebar.radio("Select:", [' ','Business Objective', 'Project Members', 'Trainer'])

if selected_value == 'Business Objective':
    st.title('Business Objective')
    st.write('''The major objective is what are the attributes that travelers are
considering while selecting a hotel. With this manager can understand which
elements of their hotel influence more in forming a positive review or improves
```

```python
    if selected_value == 'Project Members':
        st.title('Project Members')
        st.write('''1. Yenneti lekhasree
    2. Atike Bhadralaxmi

    3. Muddangula.Shirisha

    4. Dappu vaishnavi

    5. Akshitha mudhiraj

    6. karavadi Jayalakshmi

    7. Hema gorentla

    ''')

    if selected_value == 'Trainer':
        st.title('Trainer')
        st.write('Advaith')



    input_text = st.text_area("Enter the message")

    if st.button('Predict'):

        # 1. Preprocess
        transformed_text = preprocess_text(input_text)

        # 2. Vectorize
        vector_input = tfidf_vectorizer.transform([transformed_text])

        # 3. Predict
        result = model.predict(vector_input)[0]

        # 4. Display
        if result == 0:
            st.header("Negative Review")
        else:
            st.header("Positive Review")
```

Dashboard:


CONTENT
Select:
Business Objective
Project Members
Trainer
Hotel Review Analysis
HOTEL
Enter the message
Unique,great stay,
Press Ctrl+Enter to apply
Predict

Problems facing

1. In many email datasets, the number of Abusive emails is significantly lower than the number of non-Abusive emails. This can lead to biased models that have a high accuracy for the majority class but poor performance for the minority class.
2. Emails can contain a wide variety of content, from text to images, and can also be written in multiple languages. This can make it challenging to design a single model that can effectively classify all types of emails.
3. Aveling large email datasets can be time-consuming and expensive, making it difficult to obtain large annotated datasets for training and evaluation.
4. Due to the high dimensionality of email content, models can easily overfit to the training data, leading to poor generalization performance

Reference

1. “Text Classification for Sentiment Analysis: A Survey” by Mohammad et al. (2019) - a comprehensive survey of various text classification methods and their applications, including email classification.
2. "Spam Filter based on Natural Language Processing and Machine Learning" by Al-Shammari et al. (2019) - a study that presents a spam filtering system based on NLP techniques and machine learning algorithms.
3. "Classifying Email into Spam and Ham using Machine Learning" by Zhang et al. (2017) - a study that compares the performance of several machine learning algorithms for email classification.
4. "An Effective Email Classification Framework based on Machine Learning Techniques" by Kumar and Namasivayam (2015) - a study that proposes a machine learning-based framework for email classification.

Thank You