PROJECT DOCUMENTATION

PROBLEM STATEMENT

The problem at hand involves performing sentiment analysis on customer feedback to gain valuable insights into competitor products. By understanding customer sentiments, companies can identify strengths and weaknesses in competing products, thereby improving their own offerings. This project requires leveraging various Natural Language Processing (NLP) methods to extract meaningful insights from customer feedback.

DESIGN THINKING APPROACH

To effectively solve the problem of performing sentiment analysis on customer feedback, we will follow a structured design thinking approach. This approach involves several key steps, as outlined below:

DATA COLLECTION

Identify a dataset containing customer reviews and sentiments about competitor products . Identify and gather relevant datasets from sources such as online review platforms, social media, or customer feedback forms. Ensure the dataset is sufficiently large to provide meaningful insights and a representative sample of customer sentiment. Verify the data

quality by checking for missing values, duplicates, and inconsistencies.

DATA PREPROCESSING

Clean and preprocess the textual data for analysis. Remove any special characters, punctuation, and irrelevant information that may not contribute to sentiment analysis. Split the text into individual words or tokens for further analysis. Eliminate common stop words (e.g., “and,” “the,” “is”) to reduce noise in the data. Normalize text by converting it to lowercase to ensure consistent.

SENTIMENT ANALYSIS TECHNIQUES

Employ different NLP techniques like Bag of Words, Word Embeddings, or Transformer models for sentiment analysis. Create a BoW representation of the text data, which counts the frequency of words in each document. Utilize pre-trained word embeddings to capture semantic meaning and relationships between words. Leverage advanced transformer-based models for deep contextualized sentiment analysis.

FEATURE EXTRACTION

Extract features and sentiments from the text data. Assign sentiment scores (positive, negative, neutral) to each customer review using the chosen sentiment analysis technique. Extract relevant features from the text, such as product mentions, key phrases, or specific attributes that customers mention in their feedback.

VISUALIZATION

Create visualizations to depict the sentiment distribution and analyze trends. Visualize the distribution of sentiment scores using bar charts, histograms, or pie charts to understand the overall sentiment of customer feedback. Track sentiment trends over time to identify patterns or changes in customer perceptions. Generate word clouds to highlight frequently mentioned words or phrases in customer reviews.

INSIGHTS GENERATION

Extract meaningful insights from the sentiment analysis results to guide business decisions. Compare the sentiment scores of competitor products to identify strengths and weaknesses.Identify recurring themes or issues in customer feedback that require attention. Provide actionable recommendations for product improvement or marketing strategies based on the insights gained.

INNOVATION TECHNIQUES

To convert the outlined design thinking approach into innovation and solve the problem effectively

ADVANCED SENTIMENT ANALYSIS MODELS - Explore the latest developments in NLP, including state-of-the-art transformer models like GPT-4 or similar successors to enhance the accuracy of sentiment analysis.Instead of relying solely on existing datasets, consider creating a custom dataset specific to competitor products in the airline industry. This can involve

scraping social media, customer service interactions, and surveys to gather a more targeted dataset.

REAL -TIME ANALYSIS -Develop a real-time sentiment analysis system that can process and analyze customer feedback as it comes in. This allows companies to respond to issues or positive feedback prompt.

MULTIMODAL ANALYSIS -Incorporate multimodal analysis by analyzing not only text but also images and audio in customer feedback. This can provide a more comprehensive understanding of sentiment. Go beyond simple positive, negative, or neutral sentiment classification. Implement emotion detection to understand the specific emotions expressed in customer feedback, such as anger, joy, or frustration. Customize sentiment analysis for each airline based on its specific products and services. This personalization can lead to more actionable insights for each company. Create interactive dashboards that allow stakeholders to explore sentiment data visually. Use tools like Tableau or Power BI to make data-driven decisions easier.

AUTOMATED RESPONSE SYSTEM -Develop an automated response system that suggests appropriate responses to customer feedback, especially for negative sentiments. This can improve customer service and satisfaction.

PREDICTIVE ANALYSIS - Implement predictive analytics to forecast future sentiment trends based on historical data. This can help airlines proactively address potential issues. Establish a

feedback loop where insights gained from sentiment analysis are directly integrated into the product development and marketing strategies. Continuously iterate and improve based on customer feedback. Ensure ethical handling of customer data and feedback. Anonymize data when necessary and adhere to privacy regulations like GDPR or CCPA. Create user-friendly interfaces for non-technical stakeholders to access and understand sentiment analysis results easily.

MACHINE LEARNING EXPLAINABILITY -Incorporate machine learning explainability techniques to make the sentiment analysis model's decisions more transparent and understandable. Foster collaboration between departments (e.g., marketing, customer service, product development) based on the insights generated from sentiment analysis. Continuously monitor the performance of the sentiment analysis system and adapt it to evolving customer feedback patterns and changes in the competitive landscape.

PHASES OF DEVELOPMENT

To begin building the project for sentiment analysis on customer feedback, let's elaborate on the process of loading and preprocessing the dataset

DATA COLLECTION -Identify and gather relevant datasets from sources such as online review platforms, social media, or customer feedback forms. Ensure the data is in a format that can be easily imported into your chosen programming

environment. Make sure the dataset is sufficiently large to provide meaningful insights and is representative of customer sentiments. Larger datasets often yield more accurate results. Verify the data quality by checking for missing values, duplicates, and inconsistencies. This may involve data cleaning to handle issues like incorrect formatting or irrelevant information.

DATA PREPROCESSING -Remove any special characters, punctuation, and irrelevant information that may not contribute to sentiment analysis. Common cleaning steps include removing HTML tags, emojis, and URLs. Split the text into individual words or tokens for further analysis. Tokenization can be performed using libraries such as NLTK or spaCy. Eliminate common stopwords (e.g., “and,” “the,” “is”) to reduce noise in the data. You can use predefined lists of stopwords or create your own. Normalize text by converting it to lowercase to ensure consistency in text data. This step ensures that “good” and “Good” are treated as the same word.

Let’s dive into the next steps of the project after the design thinking phase. This is where we will continue building the sentiment analysis solution by performing various activities, including feature engineering, model training, and evaluation.

FEATURE ENGINEERING

Enhance the data and prepare it for model training by creating new features and optimizing the existing ones. Consider using n-grams (e.g., bigrams, trigrams) to capture more context and

phrases in the text data. Identify and select the most relevant features that contribute to sentiment analysis. Explore the sentiment conveyed by emoticons and emojis in customer feedback. Convert them into meaningful features.

MODEL TRAINING

Train machine learning or deep learning models to predict sentiment. Choose suitable algorithms (e.g., Logistic Regression, Random Forest, LSTM, Transformer) for sentiment classification. You may want to experiment with multiple models to find the best performer. Divide the dataset into training, validation, and testing sets. This ensures that you can evaluate the model's performance effectively. Optimize model hyperparameters to achieve the best performance. Use techniques like grid search or random search to fine-tune the model.

MODEL EVALUATION

Assess the performance of the sentiment analysis models and understand how well they predict sentiment. Use appropriate evaluation metrics (e.g., accuracy, precision, recall, F1-score) to measure the model's performance. Choose metrics that are relevant to your specific business objectives. Employ cross-validation techniques (e.g., k-fold cross-validation) to ensure the model's generalization and robustness. Analyze the confusion matrix to understand where the model is making errors (e.g., false positives, false negatives).

INSIGHTS GENERATION

Dive deeper into insights by considering the model's predictions and performance. Analyze which words, phrases, or features contribute the most to sentiment predictions. This helps you understand what drives customer sentiments. Examine misclassified reviews to identify common reasons for misclassification. This can lead to further improvements in the model. Refine and update your recommendations based on the model's predictions and insights.

DEPLOYMENT

Deploy the sentiment analysis model for real-time or batch analysis. Develop an API or web application that allows users to submit customer feedback and receive sentiment analysis results in real-time. Ensure that the deployed solution can handle a large volume of data efficiently. Implement monitoring and feedback loops to continuously improve the model and its performance in a production environment.

FEEDBACK LOOP

Continuously improve the sentiment analysis system based on user feedback and changing customer sentiments. Collect feedback from users and business stakeholders regarding the accuracy and usefulness of the sentiment analysis. Make periodic updates to the model based on feedback and changing customer sentiments to maintain its relevance and accuracy.

Let’s continue building the sentiment analysis solution by elaborating on the “Employing NLP techniques” and “Generating Insights” sections.

EMPLOYING NLP TECHNIQUES

Implement Natural Language Processing (NLP) techniques for sentiment analysis on customer feedback. The Bag of Words (BoW) technique creates a representation of text data by counting the frequency of words in each document. It’s a basic but effective approach. Create a vocabulary of unique words in the dataset. Count how many times each word in the vocabulary appears in each customer review. Use these word counts as features for sentiment analysis.

WORD EMBEDDINGS

Word embeddings are pre-trained models that capture the semantic meaning and relationships between words. Utilize pre-trained Word2Vec or GloVe models to convert words in customer reviews into vectors. Calculate the vector representation for each review by averaging the vectors of individual words. Use these vectors as features for sentiment analysis.

TRANSFORMER MODELS

Transformer models, like BERT and GPT, are advanced deep learning models that understand context and relationships in text. Fine-tune a pre-trained transformer model for sentiment

analysis on your dataset. Use the model to predict the sentiment (positive, negative, neutral) of each customer review.

GENERATING INSIGHTS

Extract meaningful insights from the sentiment analysis results.

COMPETITOR ANALYSIS

Compare the sentiment scores of competitor products to identify strengths and weaknesses. Calculate the average sentiment scores for each competitor product. Identify which products receive the most positive and negative feedback. Use these insights to understand market sentiment towards different products.

CUSTOMER FEEDBACK TRENDS

Identify recurring themes or issues in customer feedback that require attention. Use topic modeling techniques (e.g., Latent Dirichlet Allocation or LDA) to discover common topics in customer reviews. Analyze which topics are frequently mentioned and their associated sentiment. Identify areas where customers are most dissatisfied or satisfied.

RECOMMENDATIONS

Provide actionable recommendations for product improvement or marketing strategies based on the insights gained. Use the insights from competitor analysis and customer feedback trends to formulate specific recommendations. Provide actionable suggestions for improving product features, addressing

recurring issues, or tailoring marketing campaigns to align with customer sentiment.

THE DATASET USED

The dataset used in this work is taken from https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment?select=Tweets.csv.

This dataset consist of the reviews submitted by the individuals who travelled through various Airlines.

Here in the database the reviews are observed and based on the observation is categorized in to 3. Categories: Positive, Negative, Neutral

PURPOSE: The purpose is to create a model which on providing the tweets (reviews) to the training should provide the outcome that whether a particular tweet done by a individual is a positive response or a negative one or neutral.

All reviews that depict positive response contains good experience of the traveller with airlines.

All reviews that depict negative response contains difficulty faced by traveller

Neutral responses will be that which are not specific to be considered in positive or negative.

OUTCOME: This will help the Airline find at their deficiencies so that they could work on it.

Import the library and dataset that will be used.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/twitter-airline-sentiment/Tweets.csv
```

```
/kaggle/input/twitter-airline-sentiment/database.sqlite
```

In [4]:

```
df = pd.read_csv('/kaggle/input/twitter-airline-sentiment/Tweets.csv')
```

In [5]:

```
df.head()
```

Out[5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   tweet_id                      14640 non-null  int64  
 1   airline_sentiment             14640 non-null  object 
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object 
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object 
 6   airline_sentiment_gold        40 non-null     object 
 7   name                          14640 non-null  object 
 8   negativereason_gold           32 non-null     object 
 9   retweet_count                 14640 non-null  int64  
 10  text                          14640 non-null  object 
 11  tweet_coord                   1019 non-null   object 
 12  tweet_created                 14640 non-null  object 
 13  tweet_location                9907 non-null   object 
 14  user_timezone                 9820 non-null   object 
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

Exploratory Data Analysis

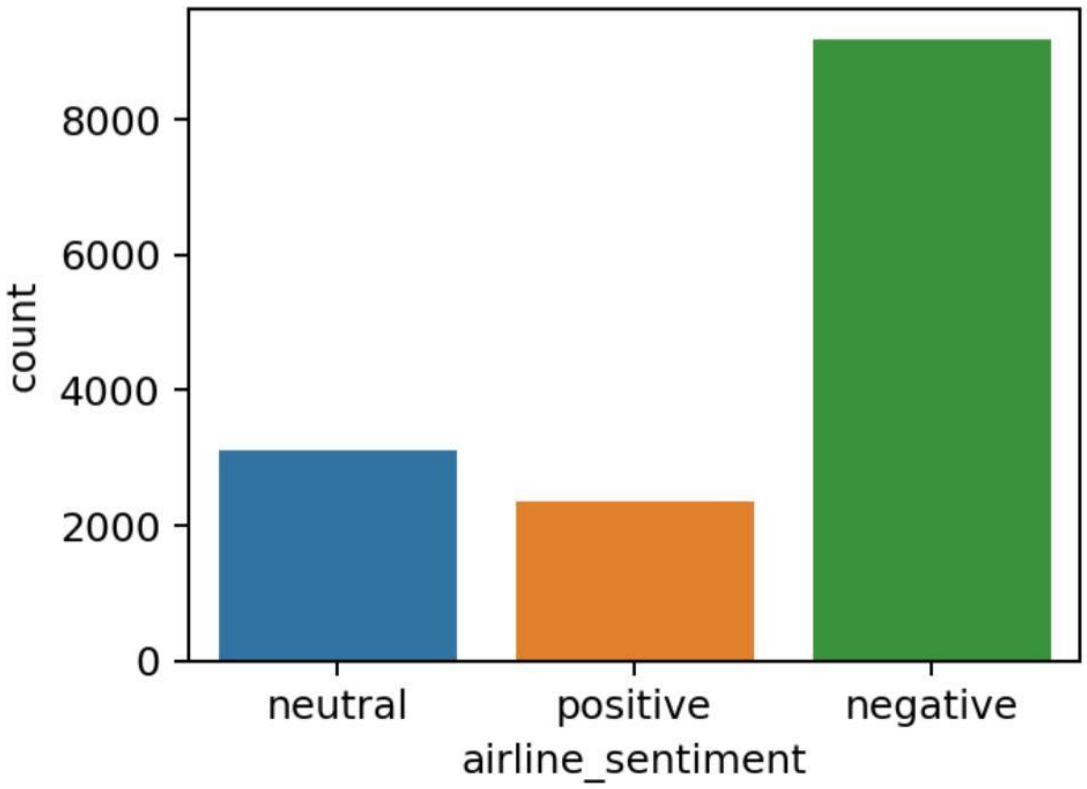
Create a countplot to see the distribution of each sentiment type

In [7]:

```
plt.figure(figsize=(4,3),dpi=180)
sns.countplot(x=df['airline_sentiment'])
```

Out[7]:

```
<AxesSubplot:xlabel='airline_sentiment', ylabel='count'>
```




```
df['airline_sentiment'].value_counts()
```

Out[8]:

```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

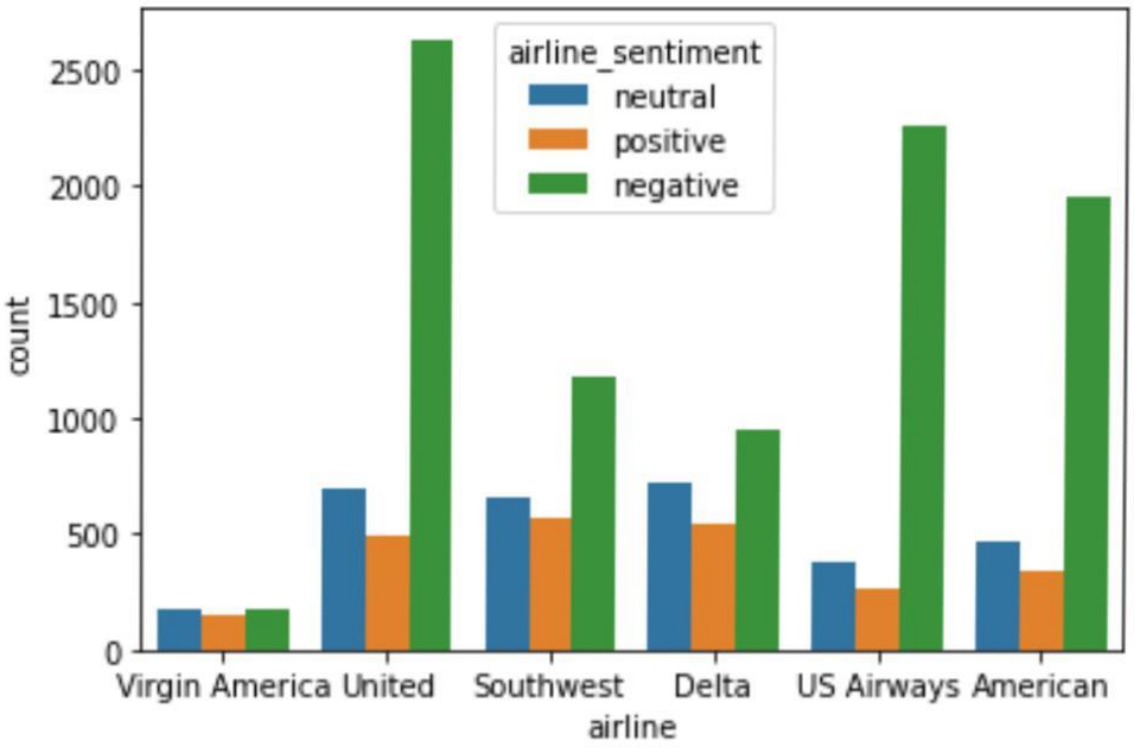
From the graph above, we can see that the labels are imbalance with more than half of the tweets are negative.

Sentiment for each airline

In [9]:

```
sns.countplot(x=df['airline'],hue=df['airline_sentiment'])
```

```
Out[9]: <AxesSubplot:xlabel='airline', ylabel='count'>
```



United has the most negative sentiments, while Virgin America has the lowest. Let's look at percentage of the negative reviews for each airline.

In [10]:

```
neg_review = df[df['airline_sentiment']=='negative']
```

In [11]:

```
neg_review.shape
```

Out[11]:

```
(9178, 15)
```

In [12]:

```
neg_review['airline_sentiment'].unique()
```

Out[12]:

```
array(['negative'], dtype=object)
```

In [13]:

```
total_neg = pd.DataFrame(neg_review.groupby('airline')['airline_sentiment'].
count())
```

In [14]:

```
total_neg = total_neg.reset_index()
```

In [15]:

```
total_neg
```

Out[15]:

	airline	airline_sentiment
0	American	1960
1	Delta	955
2	Southwest	1186
3	US Airways	2263
4	United	2633
5	Virgin America	181

```
all_review = pd.DataFrame(df.groupby('airline')['airline_sentiment'].count()
).reset_index()
```

In [17]:

```
all_review
```

Out[17]:

	airline	airline_sentiment
0	American	2759
1	Delta	2222
2	Southwest	2420
3	US Airways	2913
4	United	3822
5	Virgin America	504

```
all_review.columns = ['airline','total_reviews']In [19]:all review
```

Out[19]:

	airline	total_reviews
0	American	2759
1	Delta	2222
2	Southwest	2420
3	US Airways	2913
4	United	3822
5	Virgin America	504

```
all_review['neg_reviews'] = total_neg ['airline_sentiment']
  all_review['neg_percent_reviews'] = all_review['neg_reviews'] / all_rev
  iew['total_reviews']
```

In [21]: all_review

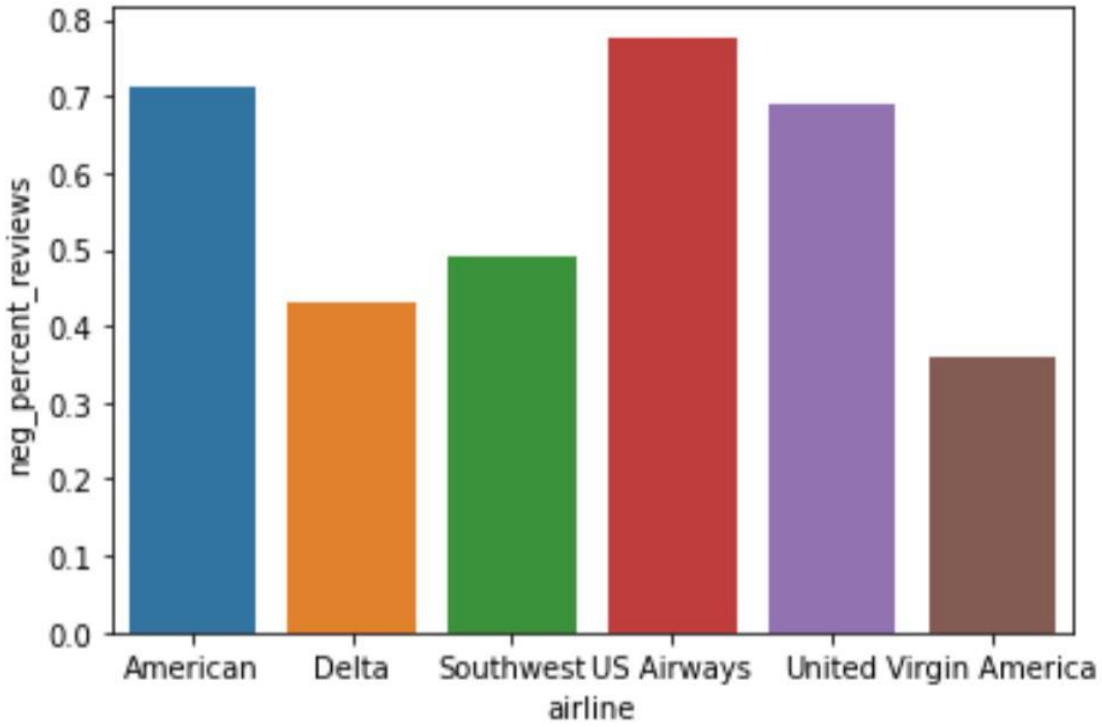
Out[21]:

	airline	total_reviews	neg_reviews	neg_percent_rev
0	American	2759	1960	0.710402
1	Delta	2222	955	0.429793
2	Southwest	2420	1186	0.490083
3	US Airways	2913	2263	0.776862
4	United	3822	2633	0.688906
5	Virgin America	504	181	0.359127

```
sns.barplot(data=all_review,x="airline",y="neg_percent_reviews")
```

Out[22]:

```
<AxesSubplot:xlabel='airline', ylabel='neg_percent_reviews'
```



From the graph:

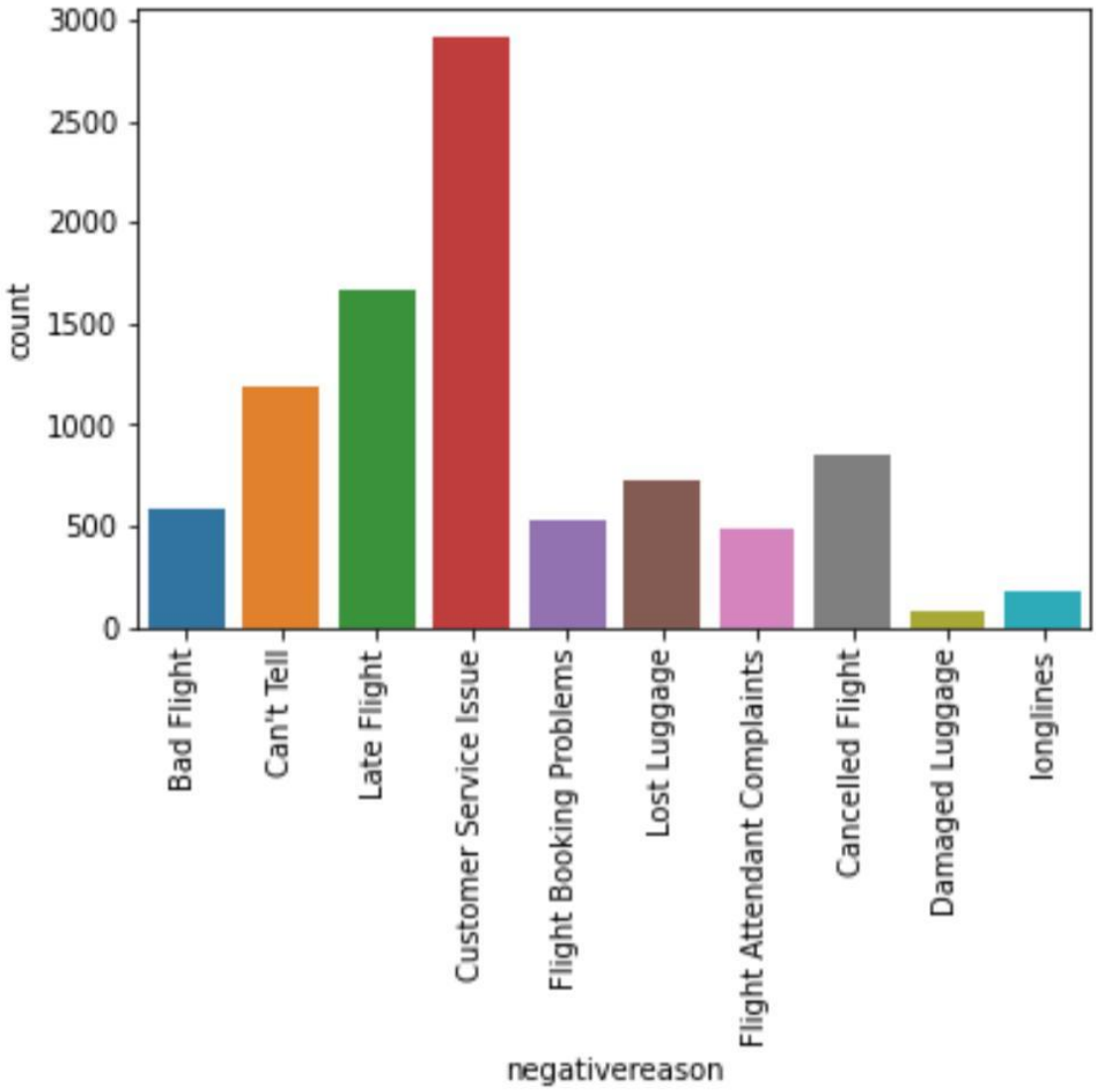
- Virgin America has the most balanced sentiment.
- American, US Airways, and United each has majority negative sentiment.

Negative Reasons

In [23]:

```
sns.countplot(x=df['negativereason'])
```

```
plt.xticks(rotation=90);
```



Customer service issue is the main reason of negative sentiment.

Data Cleaning

From the dataset, we only need two columns (text and airline_sentiment)

In [24]:

```
df = df[['text','airline_sentiment']]
```

In [25]:

```
df.head()
```

Out[25]:

	text	airline_sentiment
0	@VirginAmerica What @dhepburn said.	neutral
1	@VirginAmerica plus you've added commercials t...	positive
2	@VirginAmerica I didn't today... Must mean I n...	neutral
3	@VirginAmerica it's really aggressive to blast...	negative
4	@VirginAmerica and it's a really big bad thing...	negative

```
df.shape
```

Out[26]:

```
(14640, 2)
```

Check the missing values

In [27]:

```
df.isna().sum()
```

Out[27]:

```
text                0
airline_sentiment   0
dtype: int64
```

Check the duplicates

In [28]:

```
df.duplicated().sum()
```

Out[28]:

```
188
```

Remove the duplicates

In [29]:

```
df = df.drop_duplicates()
```

In [30]:

```
df.duplicated().sum()
```

Out[30]:

```
0
```

In [31]:

```
df.shape
```

Out[31]:

```
(14452, 2)
```

Clean the text data

In [32]:

```
import re
```

In [33]:

```
def clean_text(text):
```

```
    text = str(text).lower() # convert text to lowercase
    text = re.sub('\[.*?\]', '', text)
    text = re.sub(r'(http|https|ftp|ssh)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w
.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?', '', text) # Remove URl and tags
    text = re.sub('<.*?>+', '', text)
    text = re.sub(r'[^a-z0-9\s]', '', text) # Remove punctuation
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

In [34]:

```
df['text'] = df['text'].apply(clean_text)
```

In [35]:

```
df['text'].head()
```

Out[35]:

```
0                    virginamerica what dhepburn said
1    virginamerica plus youve added commercials to ...
2    virginamerica i didnt today must mean i need t...
3    virginamerica its really aggressive to blast o...
4    virginamerica and its a really big bad thing a...
Name: text, dtype: object
```

Train Test Split

The approach here will use Cross Validation on 90% of the dataset, and then judge the results on a final test set of 10% to evaluate the model

Split the dataset into features (X) and label/target (y)

In [36]:

```
X = df['text']
y = df['airline_sentiment']
```

Split the dataset into training set and testing set with ratio 90% : 10%.

In [37]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, r
andom_state=42)
```

In [38]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[38]:

```
((13006,), (1446,), (13006,), (1446,))
```

Feature Engineering

Use TfidfVectorizer to convert a collection of raw documents to a matrix of TF-IDF features.

In [39]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [40]:

```
vectorizer = TfidfVectorizer(stop_words='english')
```

In [41]:

```
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [42]:

```
X_train
```

Out[42]:

```
<13006x12004 sparse matrix of type '<class 'numpy.float64'>'
	with 112877 stored elements in Compressed Sparse Row format>
```

In [43]:

```
X_test
```

Out[43]:

```
<1446x12004 sparse matrix of type '<class 'numpy.float64'>'
	with 11710 stored elements in Compressed Sparse Row format>
```

Grid Search & Cross Validation

Create a grid search function with cross validation.

In [44]:

```
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV
```

In [45]:

```
def grid_search(model, parameters):
    # Use f1_weighted as scoring since we already know that the dataset has
imbalance labels
    grid = HalvingGridSearchCV(estimator=model, param_grid=parameters,fact
or=2, cv=5,
                               scoring='f1_weighted',random_state=42,error
_score=0)
    grid.fit(X_train, y_train)
    print('Best Score : ',grid.best_score_)
    print('Best parameters : ',grid.best_params_)
```

Logistic Regression Model

Create a base model of logistic regression, then perform a grid search to find the best parameters for final model.

In [46]:

```
from sklearn.linear_model import LogisticRegression
```

In [47]:

```
logreg = LogisticRegression(class_weight='balanced')
```

In [48]:

```
logreg_param = [{'penalty':['l1','l2','none'],
                 'solver': ['liblinear', 'saga'],
                 'C':[0.001,0.01,0.1,1,10],
                 'multi_class':['auto','ovr','multinomial']},
                {'penalty':['l2','none'],
                 'solver': ['sag','newton-cg','lbfgs'],
```

```
                'C':[0.001,0.01,0.1,1,10],
                'multi_class':['auto','ovr','multinomial']}]
```

In [49]:

```
# grid_search(logreg,logreg_param)
```

Use the grid search results - best parameters for final logistic regression model.

In [50]:

```
logreg_model = LogisticRegression(class_weight='balanced',C=0.1,multi_clas
s='multinomial',penalty='l2',solver='sag')
```

KNN Model

Create a base model of KNN, then perform a grid search to find the best parameters for final model.

In [51]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [52]:

```
knn = KNeighborsClassifier()
```

In [53]:

```
knn_param = {'n_neighbors':list(range(5,30)),
             'weights':['uniform', 'distance'],
             'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute']}
```

In [54]:

```
# grid_search(knn, knn_param)
```

Use the grid search results - best parameters for final KNN model.

In [55]:

```
KNN_model = KNeighborsClassifier(algorithm='ball_tree',n_neighbors=12,weig
hts='distance')
```

SVC Model

Create a base model of Support Vector Classifier, then perform a grid search to find the best parameters for final model.

In [56]:

```
from sklearn.svm import SVC
```

In [57]:

```
svc = SVC(class_weight='balanced')
```

In [58]:

```
svc_param = {'C':[0.001, 0.01, 0.1, 1, 10],
             'kernel':['linear','poly','rbf','sigmoid'],
             'gamma':['scale','auto',0.001, 0.01, 0.1, 1, 10]}
```

In [59]:

```
# grid_search(svc, svc_param)
```

Use the grid search results - best parameters for final SVC model.

In [60]:

```
SVC_model = SVC(class_weight='balanced',C=1,gamma=1,kernel='linear')
```

Naive Bayes Model

The Naive Bayes model here will use default parameters.

In [61]:

```
from sklearn.naive_bayes import BernoulliNB, GaussianNB, MultinomialNB, Co
mplementNB
```

In [62]:

```
BNB_model = BernoulliNB()
GNB_model = GaussianNB()
MNB_model = MultinomialNB()
CNB_model = ComplementNB()
NB_param = {}
```

In [63]:

```
grid_search(BNB_model,NB_param)
```

```
Best Score :  0.6826802304531071
Best parameters :  {}
```

In [64]:

```
grid_search(MNB_model,NB_param)
```

```
Best Score :  0.5924478934755192
Best parameters :  {}
```

In [65]:

```
grid_search(CNB_model,NB_param)
```

```
Best Score :  0.7369253091498471
Best parameters :  {}
```

In [66]:

```
grid2 = HalvingGridSearchCV(estimator=GNB_model, param_grid=NB_param,facto
r=2, cv=5,
                            scoring='f1_weighted',random_state=42,error
_score=0)
grid2.fit(X_train.toarray(), y_train)
print('Best Score : ',grid2.best_score_)
print('Best parameters : ')
print(grid2.best_params_)
```

```
Best Score :  0.5148436062659207
Best parameters :
{}
```

Based on the results, MNB_model and GNB_model did not perform really well (with f1_score below 0.6). So, I will not use them for final evaluation. I will use BNB_model and CNB_model instead.

In [67]:

```
BNB_model = BernoulliNB()
CNB_model = ComplementNB()
```

Decision Tree Model

Create a base model of decision tree, then perform a grid search to find the best parameters for final model.

In [68]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [69]:

```
DecTree = DecisionTreeClassifier(class_weight='balanced', random_state=42)
```

In [70]:

```
DT_param = {'criterion':['gini','entropy','log_loss'],
            'max_features':['sqrt','log2', None],
            'max_depth':[None,5,6,7,8,9,10,11,12,13,14,15]}
```

In [71]:

```
# grid_search(DecTree,DT_param)
```

Use the grid search results - best parameters for final Decision Tree model.

In [72]:

```
DTC_model = DecisionTreeClassifier(criterion='gini',max_features=None,max_
depth=None,class_weight='balanced', random_state=42)
```

Random Forest Model

Create a base model of random forest, then perform a grid search to find the best parameters for final model.

In [73]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [74]:

```
RFC = RandomForestClassifier(random_state=42, class_weight='balanced')
```

In [75]:

```
RFC_param = {'criterion':['gini','entropy','log_loss'],
            'max_features':['sqrt','log2', None],
             'n_estimators':[50,100,150,200,250,300,400],
             'max_depth':[None,5,6,7,8,9,10,11,12,13,14,15]}
```

In [76]:

```
# grid_search(RFC, RFC_param)
```

Use the grid search results - best parameters for final Random Forest model.

In [77]:

```
RFC_model = RandomForestClassifier(criterion='entropy',max_features=None,n
_estimators=400,max_depth=15,random_state=42, class_weight='balanced')
```

Bagging Classifier Model

Bagging Classifier Model will use final decision tree model (DTC_model) as base estimator.

In [78]:

```
from sklearn.ensemble import BaggingClassifier
```

In [79]:

```
Bagging = BaggingClassifier(base_estimator=DTC_model,random_state=42)
```

In [80]:

```
bag_param = {'n_estimators':[50,100,150,200,250,300,400],
             'bootstrap':[True,False]}
```

In [81]:

```
# grid_search(Bagging,bag_param)
```

Use the grid search results - best parameters for final Bagging Classifier model.

In [82]:

```
Bagging_model = BaggingClassifier(base_estimator=DTC_model,random_state=42
,bootstrap=True,n_estimators=300)
```

AdaBoost Model

AdaBoost Model will use final decision tree model (DTC_model) as base estimator.

In [83]:

```
from sklearn.ensemble import AdaBoostClassifier
```

In [84]:

```
Adaboost = AdaBoostClassifier(base_estimator=DTC_model,random_state=42)
```

In [85]:

```
Ada_param = {'n_estimators':[50,100,150,200,250,300,400],
             'learning_rate':[0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1]}
```

In [86]:

```
# grid_search(Adaboost,Ada_param)
```

Use the grid search results - best parameters for final AdaBoost model.

In [87]:

```
Adaboost_model = AdaBoostClassifier(learning_rate=0.1,n_estimators=100,bas
e_estimator=DTC_model,random_state=42)
```

Gradient Boosting Model

Create a base model of gradient boosting, then perform a grid search to find the best parameters for final model.

In [88]:

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [89]:

```
GB = GradientBoostingClassifier(random_state=42)
```

In [90]:

```
GB_param = {'n_estimators':[50,100,150,200,250,300,400],
            'learning_rate':[0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 1],
           'max_features':['sqrt','log2',None],
           'max_depth':[None,5,6,7,8,9,10,11,12,13,14,15]}
```

In [92]:

```
# grid_search(GB,GB_param)
```

```
Best Score :  0.7171231221060188
Best parameters :  {'learning_rate': 0.2, 'max_depth': 7, 'max_features'
: None, 'n_estimators': 200}
```

Use the grid search results - best parameters for final Gradient Boosting model.

In [93]:

```
GB_model = GradientBoostingClassifier(learning_rate=0.2,max_depth=7,max_fe
atures=None,n_estimators=200,random_state=42)
```

Final Model Evaluation

List all the final models we obtain from grid search above.

In [94]:

```
models = [logreg_model, KNN_model, SVC_model, BNB_model, CNB_model, DTC_mo
del,
          RFC_model, Bagging_model, Adaboost_model, GB_model]
```

In [95]:

```
from sklearn.metrics import accuracy_score,f1_score, ConfusionMatrixDispla
y,classification_report
```

In [96]:

```
accuracy_scores = []
f1_scores = []
```

In [97]:

```
for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test,y_pred)
    accuracy_scores.append(acc)

    f1 = f1_score(y_test,y_pred,average='weighted')
    f1_scores.append(f1)

    print(model)
    print()
    print(classification_report(y_test,y_pred,labels=model.classes_))
    print()
    ConfusionMatrixDisplay.from_predictions(y_test,y_pred,labels=model.cla
sses_)
    print()
```

```
LogisticRegression(C=0.1, class_weight='balanced', multi_class='multinom
ial',
                   solver='sag')

              precision    recall  f1-score   support

    negative       0.89      0.79      0.84       889
     neutral       0.52      0.71      0.60       313
    positive       0.71      0.67      0.69       244

    accuracy                           0.75      1446
   macro avg       0.71      0.72      0.71      1446
weighted avg       0.78      0.75      0.76      1446



KNeighborsClassifier(algorithm='ball_tree', n_neighbors=12, weights='dis
tance')
```

```
              precision    recall  f1-score   support

    negative       0.80      0.88      0.84       889
     neutral       0.58      0.45      0.51       313
    positive       0.67      0.60      0.63       244

    accuracy                           0.74      1446
   macro avg       0.68      0.64      0.66      1446
weighted avg       0.73      0.74      0.73      1446



SVC(C=1, class_weight='balanced', gamma=1, kernel='linear')

              precision    recall  f1-score   support

    negative       0.90      0.80      0.85       889
     neutral       0.55      0.71      0.62       313
    positive       0.71      0.71      0.71       244

    accuracy                           0.77      1446
   macro avg       0.72      0.74      0.73      1446
weighted avg       0.79      0.77      0.77      1446



BernoulliNB()

              precision    recall  f1-score   support

    negative       0.73      0.96      0.83       889
     neutral       0.65      0.37      0.47       313
    positive       0.80      0.32      0.45       244

    accuracy                           0.73      1446
   macro avg       0.73      0.55      0.58      1446
weighted avg       0.73      0.73      0.69      1446



ComplementNB()

              precision    recall  f1-score   support

    negative       0.80      0.95      0.87       889
     neutral       0.69      0.39      0.50       313
```

```
    positive       0.71      0.63      0.67       244

    accuracy                           0.77      1446
   macro avg       0.73      0.65      0.68      1446
weighted avg       0.76      0.77      0.75      1446



DecisionTreeClassifier(class_weight='balanced', random_state=42)

              precision    recall  f1-score   support

    negative       0.80      0.72      0.76       889
     neutral       0.38      0.52      0.44       313
    positive       0.62      0.58      0.60       244

    accuracy                           0.65      1446
   macro avg       0.60      0.60      0.60      1446
weighted avg       0.68      0.65      0.66      1446



RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=15, max_features=None, n_estimators=400
,
                       random_state=42)

              precision    recall  f1-score   support

    negative       0.83      0.74      0.78       889
     neutral       0.43      0.60      0.50       313
    positive       0.70      0.59      0.64       244

    accuracy                           0.69      1446
   macro avg       0.65      0.64      0.64      1446
weighted avg       0.72      0.69      0.70      1446



BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight='ba
lanced',
                                                        random_state=42)
,
                  n_estimators=300, random_state=42)

              precision    recall  f1-score   support
```

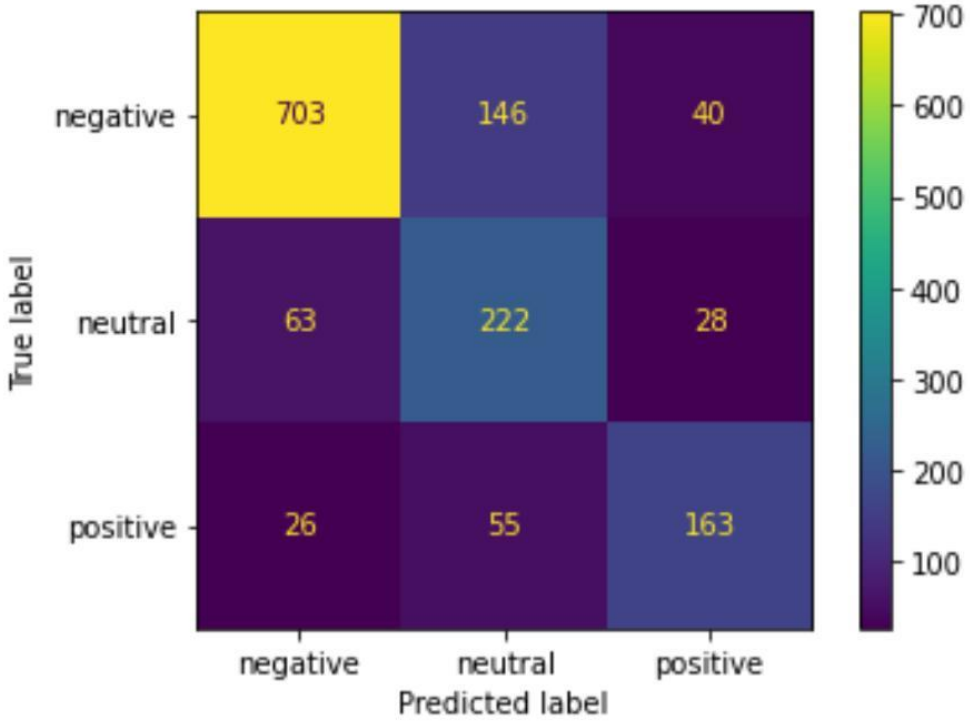
```
              precision    recall  f1-score   support

    negative       0.82      0.85      0.84       889
     neutral       0.52      0.52      0.52       313
    positive       0.69      0.60      0.64       244

    accuracy                           0.74      1446
   macro avg       0.68      0.66      0.67      1446
weighted avg       0.73      0.74      0.74      1446
```

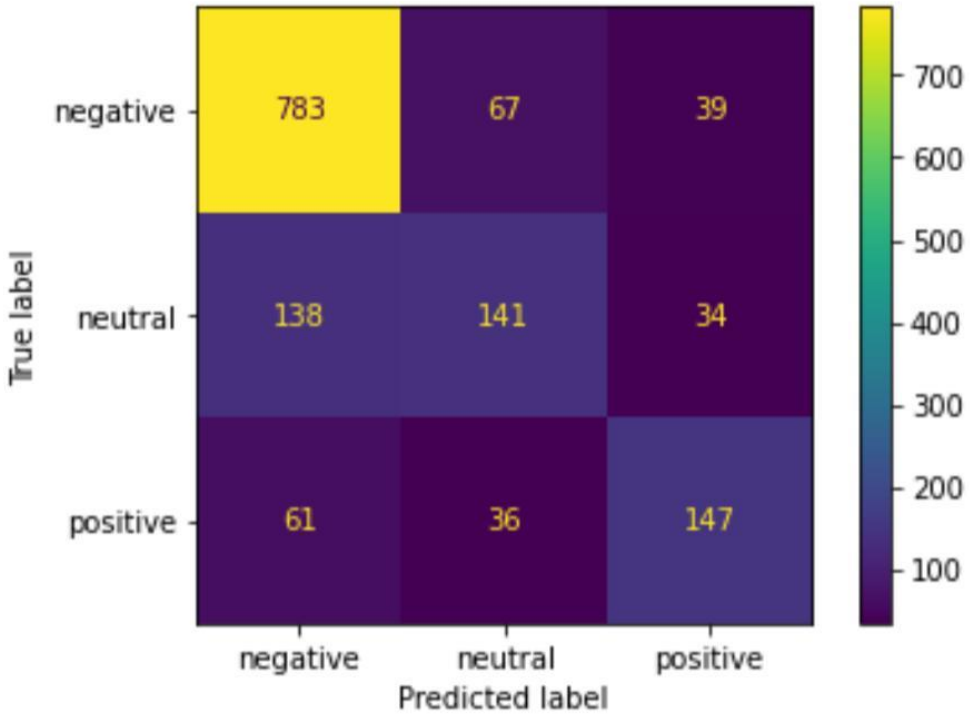
```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(class_weight='b
alanced',
                                                         random_state=42
),
                   learning_rate=0.1, n_estimators=100, random_state=42)
```

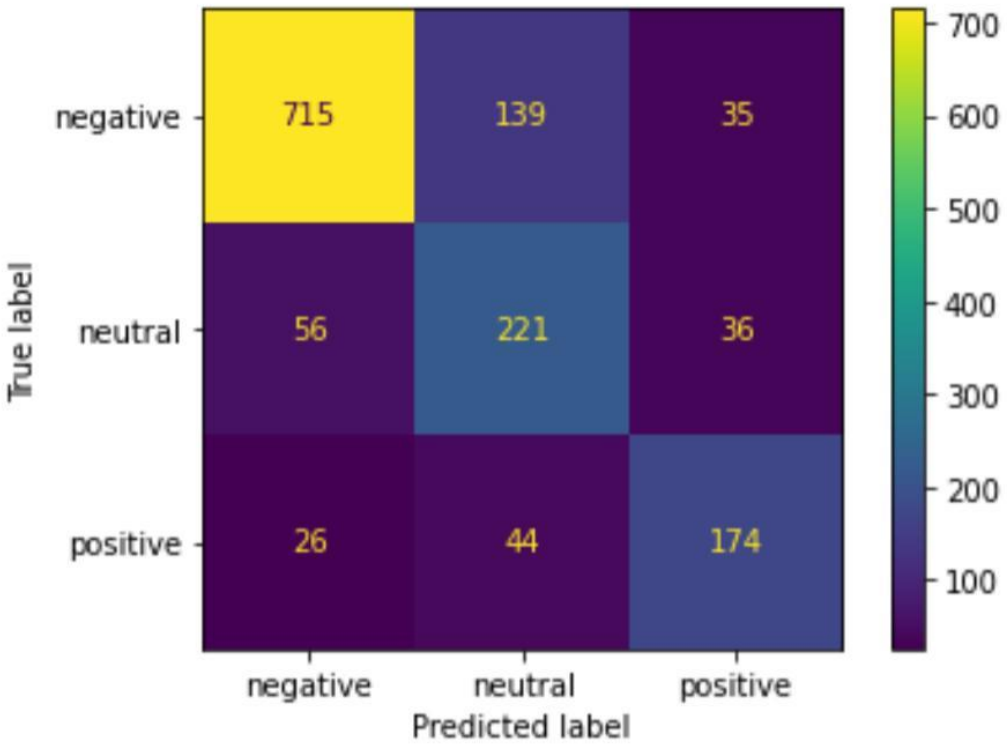
```
              precision    recall  f1-score   support

    negative       0.73      0.72      0.72       889
     neutral       0.37      0.41      0.39       313
    positive       0.43      0.40      0.41       244

    accuracy                           0.60      1446
   macro avg       0.51      0.51      0.51      1446
weighted avg       0.60      0.60      0.60      1446
```

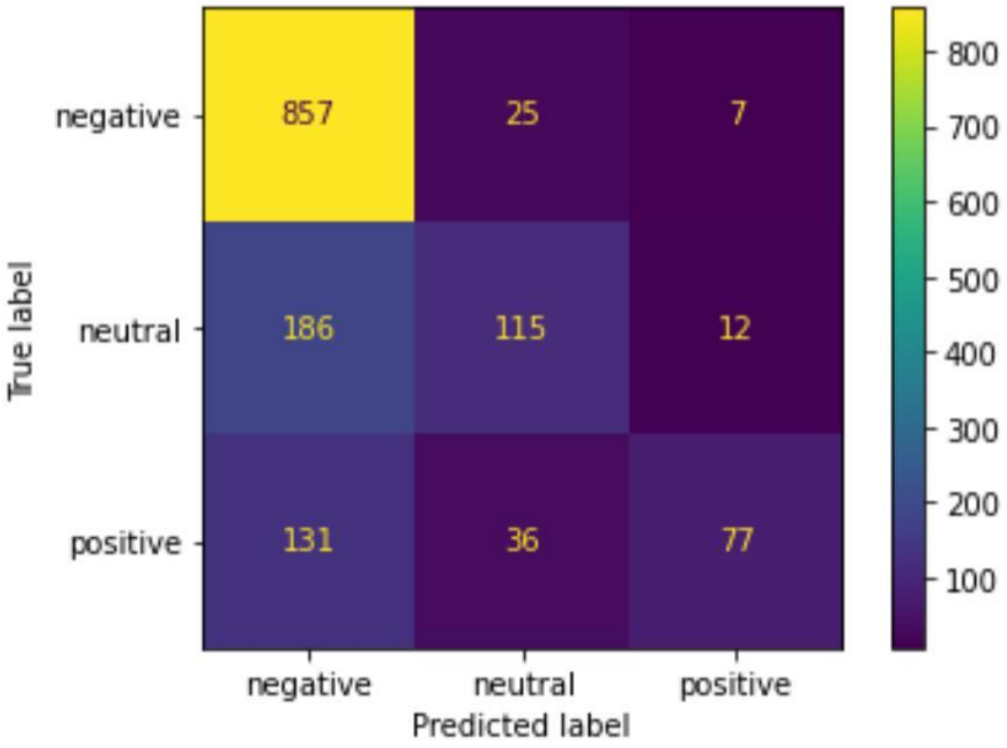
```
GradientBoostingClassifier(learning_rate=0.2, max_depth=7, n_estimators=
200,
                           random_state=42)
```

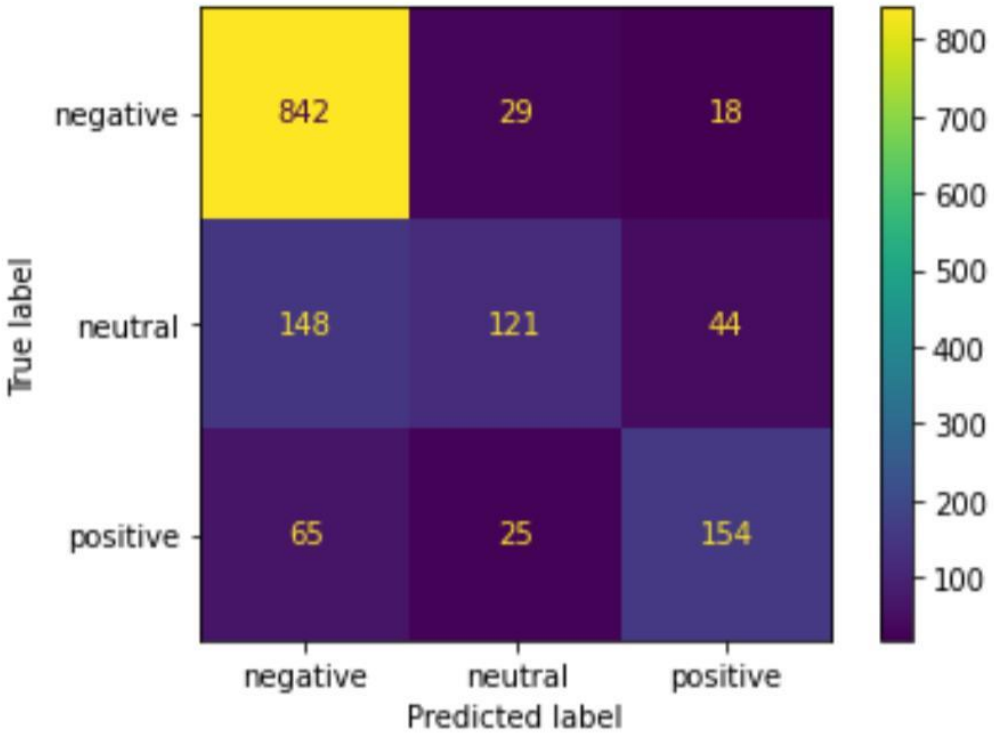
```
              precision    recall  f1-score   support

    negative       0.80      0.92      0.86       889
     neutral       0.63      0.46      0.53       313
    positive       0.76      0.59      0.67       244

    accuracy                           0.77      1446
   macro avg       0.73      0.66      0.69      1446
weighted avg       0.76      0.77      0.76      1446
```

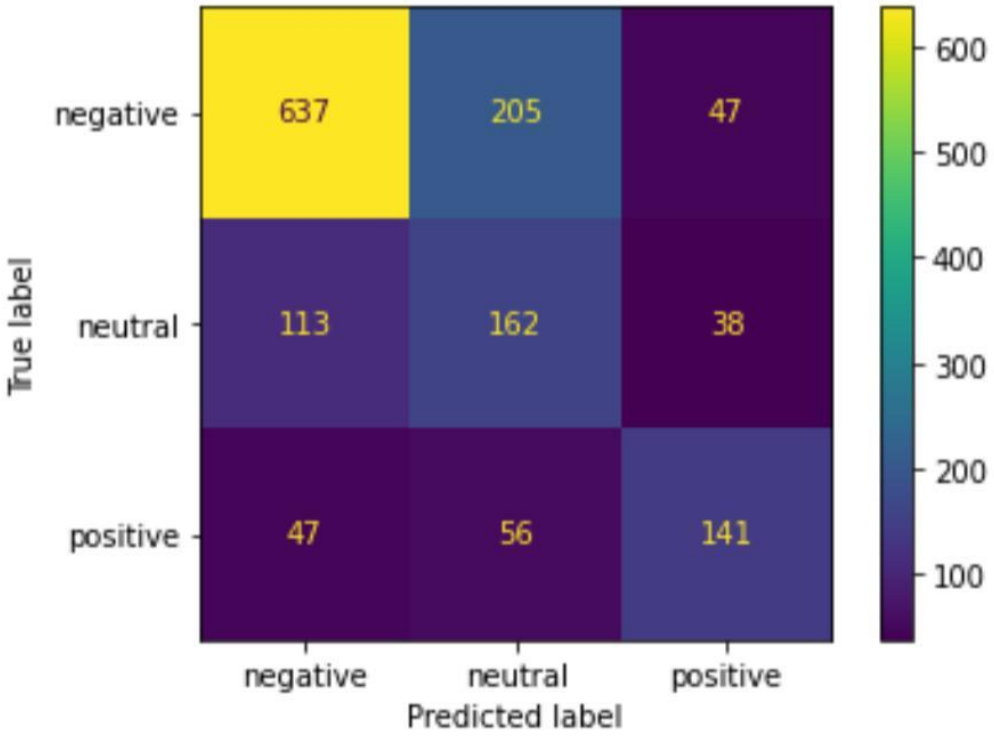

negative
neutral
positive
True label
703
146
40
63
222
28
26
55
163
negative
neutral
positive
Predicted label
700
600
500
400
300
200
100

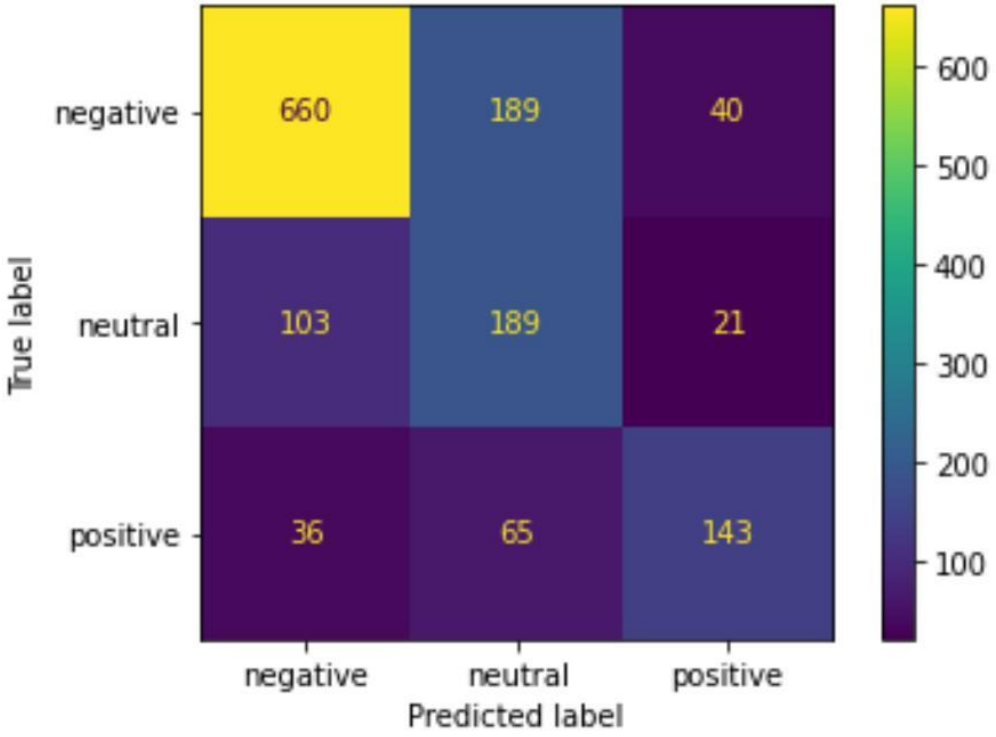

negative
neutral
positive
True label
783
67
39
138
141
34
61
36
147
negative
neutral
positive
Predicted label
700
600
500
400
300
200
100

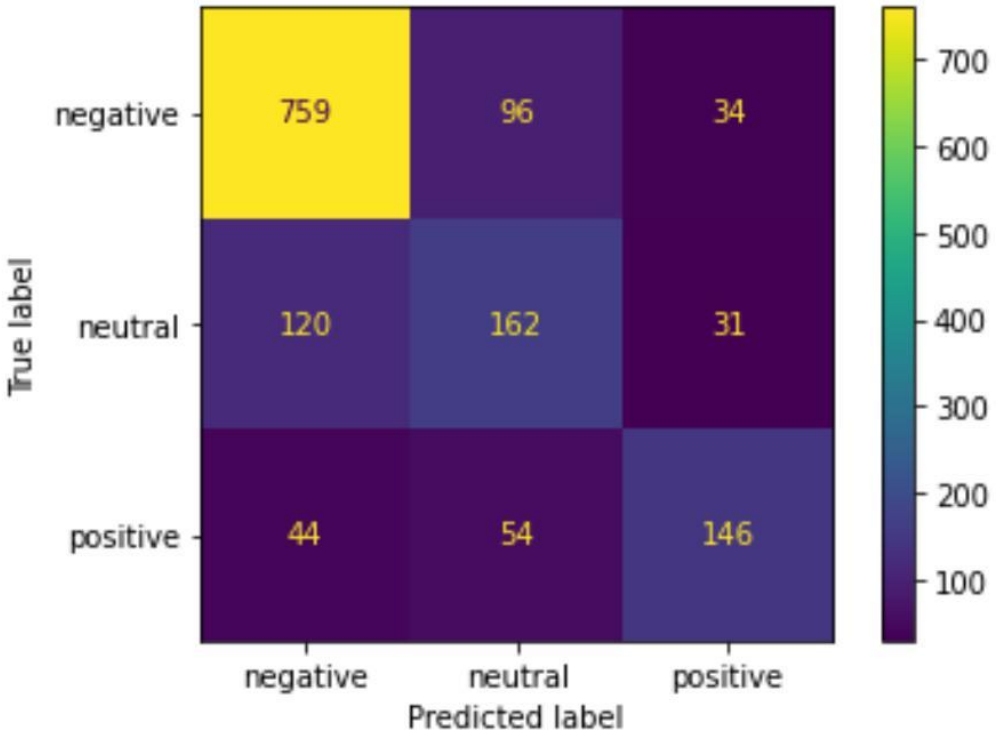

negative
neutral
positive
True label
715
139
35
56
221
36
26
44
174
negative
neutral
positive
Predicted label
700
600
500
400
300
200
100

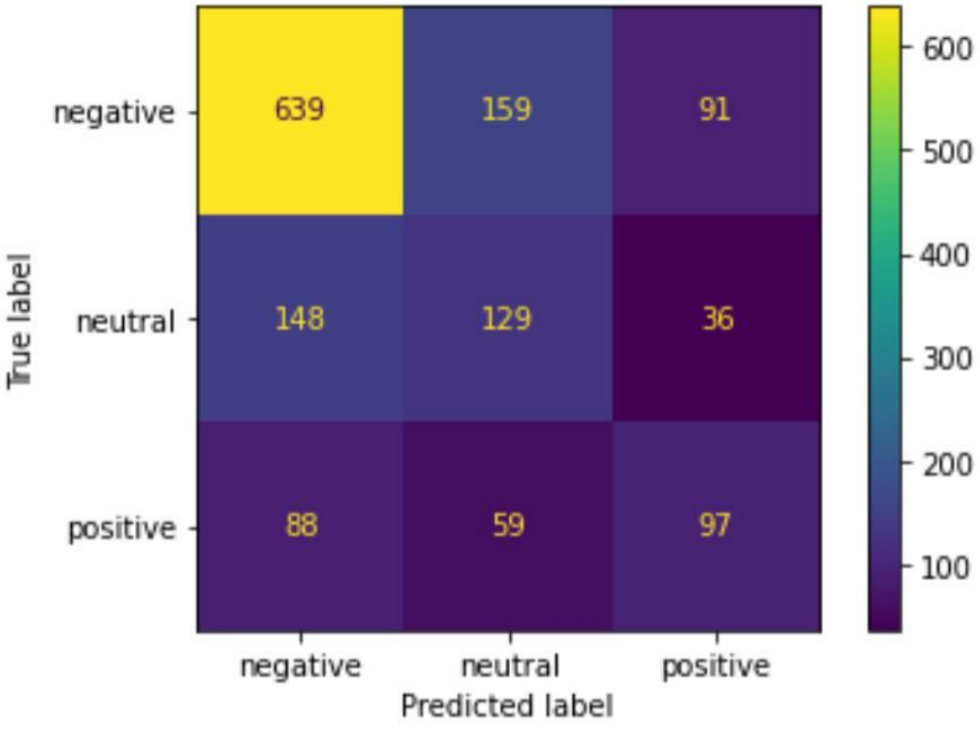

negative
neutral
positive
True label
857
25
7
186
115
12
131
36
77
negative
neutral
positive
Predicted label
800
700
600
500
400
300
200
100

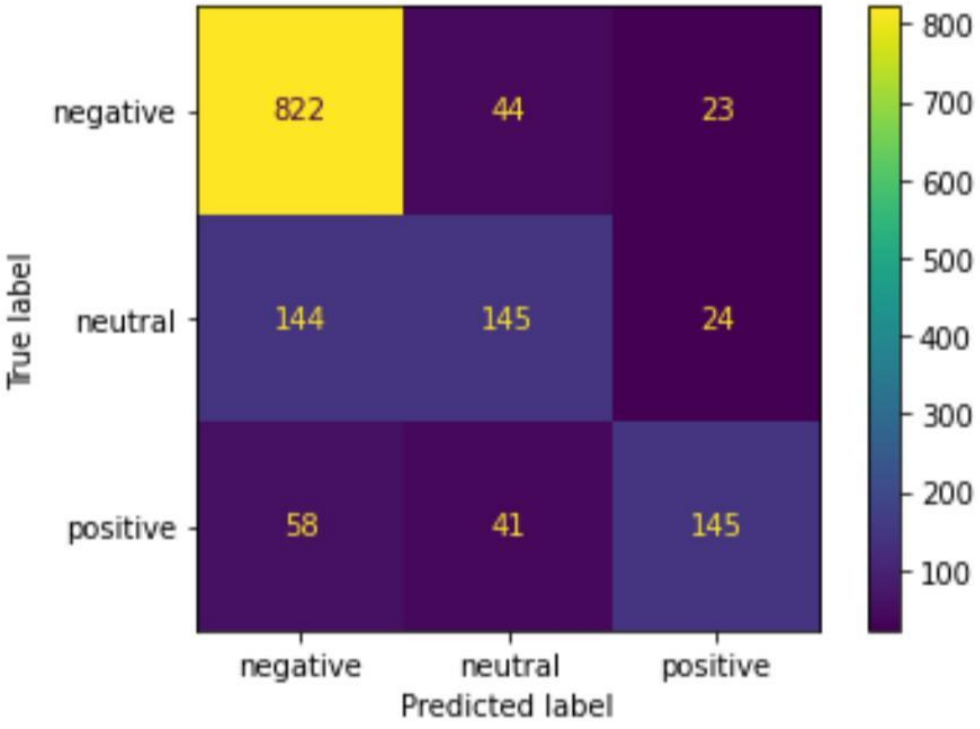

negative
neutral
positive
True label
842
29
18
148
121
44
65
25
154
negative
neutral
positive
Predicted label
800
700
600
500
400
300
200
100


negative
neutral
positive
True label
637
205
47
113
162
38
47
56
141
negative
neutral
positive
Predicted label
600
500
400
300
200
100


negative
neutral
positive
True label
660
189
40
103
189
21
36
65
143
negative
neutral
positive
Predicted label
600
500
400
300
200
100


negative
neutral
positive
True label
759
96
34
120
162
31
44
54
146
negative
neutral
positive
Predicted label
700
600
500
400
300
200
100


negative
neutral
positive
True label
639
159
91
148
129
36
88
59
97
negative
neutral
positive
Predicted label
600
500
400
300
200
100


negative
neutral
positive
True label
822
44
23
144
145
24
58
41
145
negative
neutral
positive
Predicted label
800
700
600
500
400
300
200
100

```
accuracy_scores
```

Out[98]:

```
[0.7524204702627939,
 0.7406639004149378,
 0.7676348547717843,
 0.7254495159059474,
 0.7724757952973721,
 0.6500691562932227,
 0.686030428769018,
 0.7378976486860305,
 0.5982019363762102,
 0.7690179806362379]
```

In [99]:

```
f1_scores
```

Out[99]:

```
[0.7606137202539621,
 0.7310860343450013,
 0.7749728757879148,
 0.6890341245813203,
 0.7528989966829184,
 0.6610976741599717,
 0.696720214801226,
 0.7355508223614842,
 0.5999801335900921,
 0.7561888785552755]
```

In [100]:

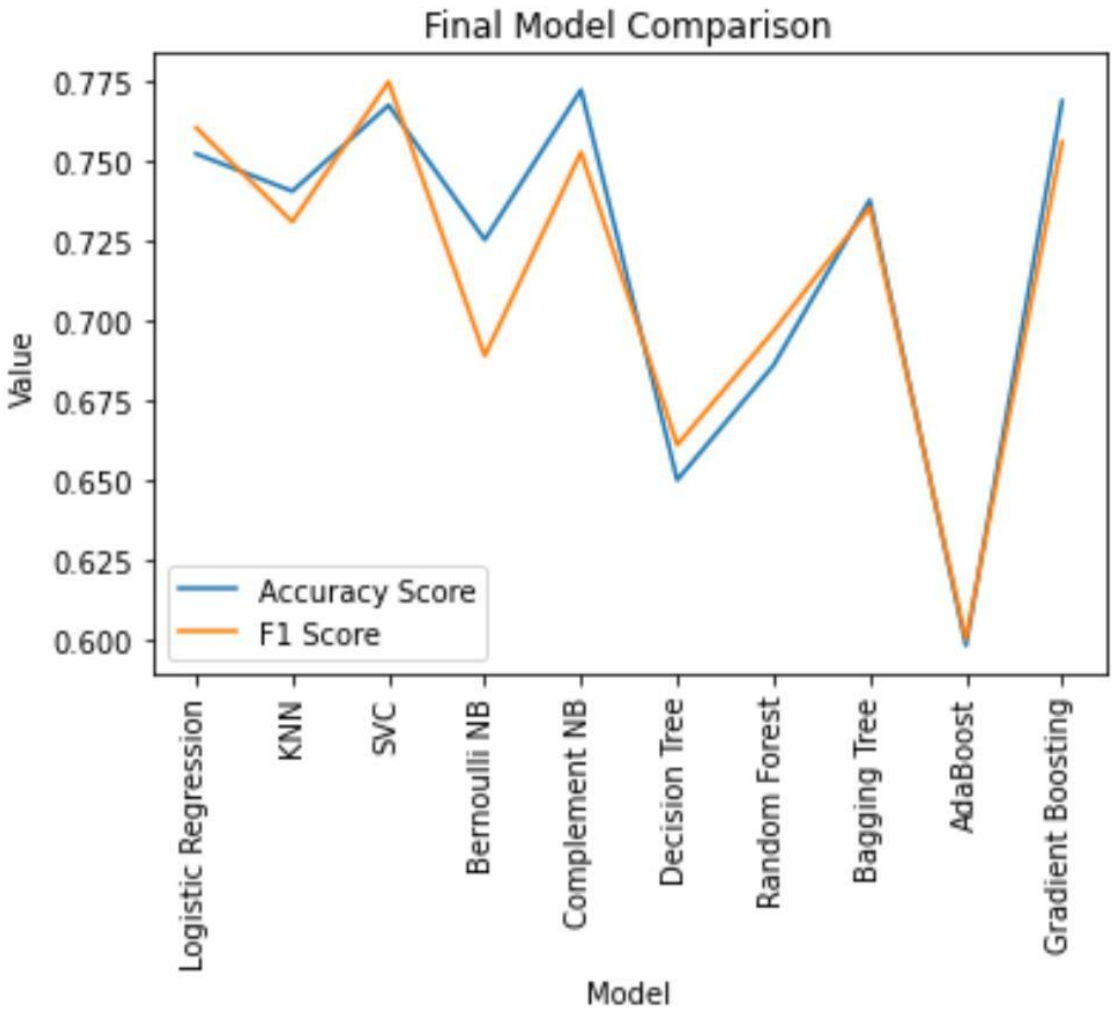
```
clf_model = ['Logistic Regression', 'KNN', 'SVC', 'Bernoulli NB', 'Complem
ent NB', 'Decision Tree',
          'Random Forest', 'Bagging Tree', 'AdaBoost', 'Gradient Boosting'
 ]
```

Create a plot to visualize the model performance

In [101]:

```
plt.plot(clf_model,accuracy_scores,label='Accuracy Score')
plt.plot(clf_model,f1_scores,label='F1 Score')
```

```
plt.xlabel('Model')
plt.ylabel('Value')
plt.title('Final Model Comparison')
plt.legend()
plt.xticks(rotation=90);
```



Because the class labels in dataset are unbalanced, so we use the f1 score to determine the best model.

CONCLUSION

After performing cross validation and hyper parameter tuning via grid search, also evaluating the final 10 models to unseen dataset, here are some conclusion.

- The best model is SVC (C=1, gamma=1, kernel='linear') with 76.8% accuracy and 77.5% f1 score.
- Decision Tree improves significantly after used on Bagging Classifier.
- Neutral is the hardest class label to predict accurately.

This design thinking approach outlines a structured methodology f or tackling the problem of preforming sentiment analysis on customer feedback a gain insights into competitor products. By allowing these steps, we aim to extract valuable information from textual data, visualize trends, and generate actio nable insights that can drive informed business decisions and enha nce the company's competitive edge in the market. By infusing innovation into design thinking approach, can createa powerful sol ution for performing sentiment analysis on customer in the airline industry, Ultimately leading to improved products and services, enhanced customer satisfaction, and a Competitive edge in the market. By following these steps, we will have a clean and loaded dataset readyfor sentiment analysis. Fro m there, we can proceed with applying sentiment analysis techniq ue, feature extraction, visualization, and insights

generation as outlined in our design thinking approach. Then following these additional steps, we'll not only build a sentiment analysis system but also ensure that it evolves over time to adapt to changing customer sentiments and provide valuable insights to guide business decisions. This iterative approach is crucial for maintaining the system's effectiveness and relevance. By employing NLP techniques and generating insights from the sentiment analysis, we'll be able to not only understand customer sentiment but also derive actionable recommendations that can drive business decisions, improve products, and enhance the company's competitive edge in the market. This process can be iterative, with continuous updates and improvements based on changing customer sentiments and feedback.