**SENTIMENT ANALYSIS FOR MARKETING**

**PHASE 5**

|  |  |
| --- | --- |
| NAME | SRIRAM V |
| TEAM ID | proj-212173-Team\_2 |

**INTODUCTION:**

In today's dynamic marketing landscape, understanding and leveraging the power of customer sentiment is paramount. With the explosion of digital data, the art of making data-driven marketing decisions has become a game-changer. As explores the evolving realms of marketing, we embrace the fusion of Artificial Intelligence (AI) and Machine Learning (ML) techniques to decipher the hidden gems within customer feedback. Welcome to the realm of Sentiment Analysis for Marketing – where AI and ML unlock invaluable insights that revolutionize marketing strategies.

**Overview of the process:**

Sentiment analysis in marketing is a process that involves the use of natural language processing (NLP) techniques to assess and understand the sentiment or emotions expressed in customer feedback, comments, reviews, and other textual data. Here's an overview of the sentiment analysis process for marketing.

**IMPORT LIBRARIES :**

import pandas as pd

import numpy as np

import torch

import tokenize

import seaborn as sns

import matplotlib.pyplot as plt

import nltk

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import CountVectorizer

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import re

import nltk

nltk.download('stopwords')

nltk.download('wordnet')

import tensorflow as tf

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from tensorflow.keras.preprocessing.sequence import pad\_sequences

import torch

from transformers import TFBertForSequenceClassification,BertTokenizer,AdamW,get\_linear\_schedule\_with\_warmup,AutoModel,AutoTokenizer,BertModel

# Load the dataset

data = pd.read\_csv('Tweets.csv')

# Display the first 5 rows of the dataframe

data.head()

OUT:

|  | **tweet\_id** | **airline\_sentiment** | **airline\_sentiment\_confidence** | **negativereason** | **negativereason\_confidence** | **airline** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 5.703060e+17 | neutral | 1.0000 | NaN | NaN | Virgin America |
| **1** | 5.703010e+17 | positive | 0.3486 | NaN | 0.0000 | Virgin America |
| **2** | 5.703010e+17 | neutral | 0.6837 | NaN | NaN | Virgin America |
| **3** | 5.703010e+17 | negative | 1.0000 | Bad Flight | 0.7033 | Virgin America |
| **4** | 5.703010e+17 | negative | 1.0000 | Can't Tell | 1.0000 | Virgin America |

| **airline\_sentiment\_gold** | **name** | **negativereason\_gold** | **retweet\_count** | **text** | **tweet\_coord** | **tweet\_created** | **tweet\_location** | **user\_timezone** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NaN | cairdin | NaN | 0 | @VirginAmerica What @dhepburn said. | NaN | 24-02-2015 11:35 | NaN | Eastern Time (US & Canada) |
| NaN | jnardino | NaN | 0 | @VirginAmerica plus you've added commercials t... | NaN | 24-02-2015 11:15 | NaN | Pacific Time (US & Canada) |
| NaN | yvonnalynn | NaN | 0 | @VirginAmerica I didn't today... Must mean I n... | NaN | 24-02-2015 11:15 | Lets Play | Central Time (US & Canada) |
| NaN | jnardino | NaN | 0 | @VirginAmerica it's really aggressive to blast... | NaN | 24-02-2015 11:15 | NaN | Pacific Time (US & Canada) |
| NaN | jnardino | NaN | 0 | @VirginAmerica and it's a really big bad thing... | NaN | 24-02-2015 11:14 | NaN | Pacific Time (US & Canada) |

#load the dataset

data.columns

data.info()

print(data.shape)

print(data['airline\_sentiment'].value\_counts())

OUT:

<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 float64

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(3), int64(1), object(11)

memory usage: 1.7+ MB

(14640, 15)

negative 9178

neutral 3099

positive 2363

Name: airline\_sentiment, dtype: int64

data.head()

df = data[['airline\_sentiment','text']]

df

|  |  |  |
| --- | --- | --- |
|  | **airline\_sentiment** | **text** |
| **0** | neutral | @VirginAmerica What @dhepburn said. |
| **1** | positive | @VirginAmerica plus you've added commercials t... |
| **2** | neutral | @VirginAmerica I didn't today... Must mean I n... |
| **3** | negative | @VirginAmerica it's really aggressive to blast... |
| **4** | negative | @VirginAmerica and it's a really big bad thing... |
| **...** | ... | ... |
| **14635** | positive | @AmericanAir thank you we got on a different f... |
| **14636** | negative | @AmericanAir leaving over 20 minutes Late Flig... |
| **14637** | neutral | @AmericanAir Please bring American Airlines to... |
| **14638** | negative | @AmericanAir you have my money, you change my ... |
| **14639** | neutral | @AmericanAir we have 8 ppl so we need 2 know h... |

14640 rows × 2 columns

#preprocess the data

def no\_emo(text):

emoji\_pattern = re.compile("[" u"\U0001F600-\U0001F64F"

# emoticons

u"\U0001F300-\U0001F5FF"

# symbols & pictographs

u"\U0001F680-\U0001F6FF"

# transport & map symbols

u"\U0001F1E0-\U0001F1FF"

# flags (iOS)

"]+", flags=re.UNICODE)

return (emoji\_pattern.sub(r'', text))

def preprocess\_text(df):

df['text'] =df['text'].apply(lambda x : x.lower().strip())

#case norm

df['text'] =df['text'].apply(lambda x: re.sub("\S\*@\S\*\s?", '', x))

#email remove

df['text'] =df['text'].apply(lambda x: re.sub(r'http\S+', '', x))

# http remove

df['text'].apply(no\_emo)

# remove emojis

df['text'] =df['text'].apply(lambda x: re.sub('[^a-zA-Z\n\.]', ' ', x))

#Remove special characters, non-text characters

df['text'] =df['text'].apply(lambda x:re.sub(r'([^\w\s]|\_)+', ' ', x))

#Remove repeated punctuations

df['text'] =df['text'].apply(lambda x:re.sub(r'\s+', ' ',x))

#Remove white spaces

df['text'] = df['text'].apply(lambda x: re.sub(r'\bamp\b', '', x))

df['text'] =df['text'].apply(lambda x:x.strip())

return df

data['labels'] = data["airline\_sentiment"].apply(lambda x: 0 if x == "negative" else 1 if x == "neutral" else 2)

data = preprocess\_text(data)

def convert\_to\_numeric(text):

tokenizer = tf.keras.preprocessing.text.Tokenizer()

tokenizer.fit\_on\_texts(data['text'])

word\_index = tokenizer.word\_index

sequences = tokenizer.texts\_to\_sequences(data['text'])

padded\_sequences = pad\_sequences(sequences, maxlen=100)

return padded\_sequences

nltk.download('punkt')

nltk.download('stopwords')

OUT:

[nltk\_data] Downloading package punkt to /root/nltk\_data...

[nltk\_data] Unzipping tokenizers/punkt.zip.

[nltk\_data] Downloading package stopwords to /root/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

True

df['text'] = df['text'].apply(preprocess\_text)

df

OUT:

<ipython-input-17-919540b8885e>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy>

df['text'] = df['text'].apply(preprocess\_text)

|  | **airline\_sentiment** | **text** |
| --- | --- | --- |
| **0** | neutral | virginamericadhepburnsaid |
| **1** | positive | virginamericaplusaddedcommercialsexperiencetacky |
| **2** | neutral | virginamericatodaymustmeanneedtakeanothertrip |
| **3** | negative | virginamericareallyaggressiveblastobnoxiousent... |
| **4** | negative | virginamericareallybigbadthing |
| **...** | ... | ... |
| **14635** | positive | americanairthankgotdifferentflightchicago |
| **14636** | negative | americanairleaving20minuteslateflightwarningsc... |
| **14637** | neutral | americanairpleasebringamericanairlinesblackber... |
| **14638** | negative | americanairmoneychangeflightanswerphonessugges... |
| **14639** | neutral | americanair8pplneed2knowmanyseatsnextflightplz... |

14640 rows × 2 columns

#Tokenization & Vectorization

import torch

from transformers import TFBertForSequenceClassification,BertTokenizer,AdamW,get\_linear\_schedule\_with\_warmup,AutoModel,AutoTokenizer,BertModel

from sklearn.feature\_extraction.text import TfidfVectorizer

# Create TF-IDF vectorizer

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)

# Fit and transform your cleaned text data into numerical features

X = tfidf\_vectorizer.fit\_transform(df['text'])

print(X)

#splitting the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, df['airline\_sentiment'], test\_size=0.2, random\_state=42)

# Create and train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

OUT:

(0, 4600) 1.0

(1, 4831) 1.0

(2, 4925) 1.0

(3, 4847) 1.0

(4, 4848) 1.0

(5, 4867) 1.0

(6, 4981) 1.0

(7, 4851) 1.0

(8, 4968) 1.0

(10, 4728) 1.0

(11, 4769) 1.0

(12, 4676) 1.0

(13, 4954) 1.0

(14, 4903) 1.0

(16, 4616) 1.0

(17, 4625) 1.0

(19, 4729) 1.0

(20, 4622) 1.0

(21, 4759) 1.0

(22, 4762) 1.0

(23, 4775) 1.0

(24, 4688) 1.0

(25, 4889) 1.0

(26, 4692) 1.0

(27, 4788) 1.0

: :

(13151, 133) 1.0

(13169, 4999) 1.0

(13210, 131) 1.0

(13213, 38) 1.0

(13278, 36) 1.0

(13322, 23) 1.0

(13339, 131) 1.0

(13346, 1) 1.0

(13442, 131) 1.0

(13522, 54) 1.0

(13552, 131) 1.0

(13565, 23) 1.0

(13680, 4984) 1.0

(13766, 133) 1.0

(13864, 122) 1.0

(13884, 60) 1.0

(13995, 5) 1.0

(14020, 36) 1.0

(14386, 148) 1.0

(14392, 111) 1.0

(14512, 92) 1.0

(14543, 109) 1.0

(14544, 92) 1.0

(14556, 131) 1.0

(14630, 133) 1.0

Accuracy: 0.655396174863388

#Evaluate the model on the test set

accuracy = model.score(test\_vec, test\_labels)

print(f'Test accuracy: {accuracy:.4f}')

OUT:

Test accuracy: 0.6554

train\_data, val\_data = train\_test\_split(df, test\_size=0.2, random\_state=42)

tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')

train\_encoding=tokenizer(list(train\_data['text']),truncation=True,padding=True)

valid\_encoding=tokenizer(list(val\_data['text']),truncation=True,padding=True)

sentiment\_dict = {'positive': 0, 'negative': 1, 'neutral': 2}

train\_labels = train\_data['airline\_sentiment'].map(sentiment\_dict).values.astype('int64')

valid\_labels = val\_data['airline\_sentiment'].map(sentiment\_dict).values.astype('int64')

print(len(train\_labels))

print(len(valid\_labels))

print(len(train\_encoding))

print(len(valid\_encoding))

OUT:  
11712

2928

3

3

# Calculate the distribution of sentiment

sentiment\_distribution = data['airline\_sentiment'].value\_counts()

# Most common reasons for negative sentiments

common\_negative\_reasons = data[data['airline\_sentiment'] == 'negative']['negativereason'].value\_counts()

# Analyze the impact of airline sentiment confidence

data['airline\_sentiment\_confidence'].groupby(data['airline\_sentiment']).mean()

# Explore the relationship between sentiment and airline

sentiment\_by\_airline = data.groupby(['airline', 'airline\_sentiment']).size().unstack()

sentiment\_by\_airline

OUT:

| **airline\_sentiment** | **negative** | **neutral** | **positive** |
| --- | --- | --- | --- |
| **airline** |  |  |  |
| **American** | 1960 | 463 | 336 |
| **Delta** | 955 | 723 | 544 |
| **Southwest** | 1186 | 664 | 570 |
| **US Airways** | 2263 | 381 | 269 |
| **United** | 2633 | 697 | 492 |
| **Virgin America** | 181 | 171 | 152 |

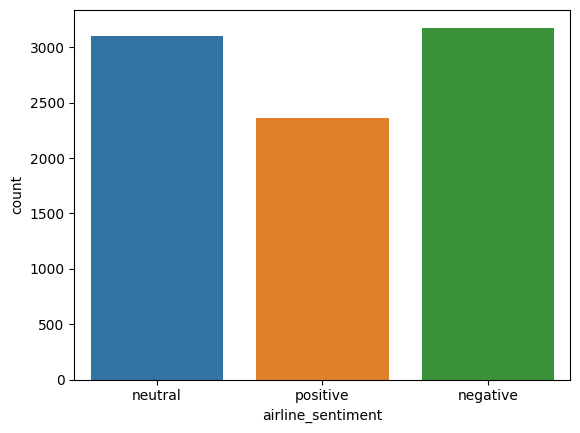
#Count plotting

import seaborn as sns

df\_new=data.drop(data[data.airline\_sentiment =='negative'].iloc[:6000].index)

sns.countplot(data=df\_new, x='airline\_sentiment')

OUT:  
<Axes: xlabel='airline\_sentiment', ylabel='count'>



from wordcloud import WordCloud

text = " ".join(tweet for tweet in data['text'])

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text)

plt.figure(figsize=(10, 5))

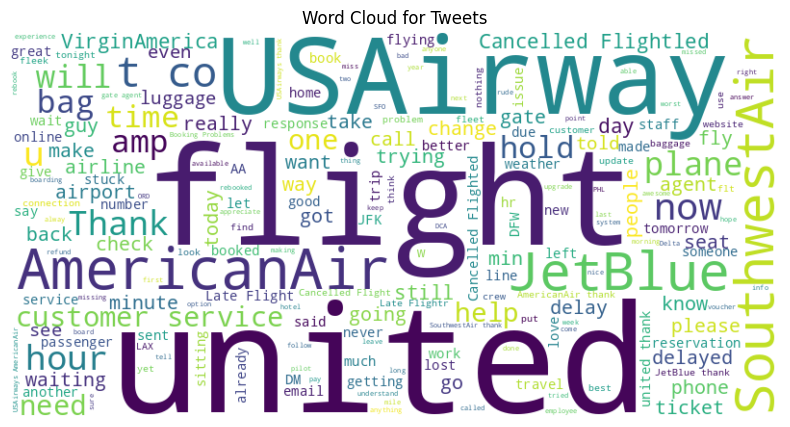
plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.title("Word Cloud for Tweets")

plt.show()

OUT:



data['tweet\_length'] = data['text'].apply(len)

plt.hist(data['tweet\_length'], bins=20)

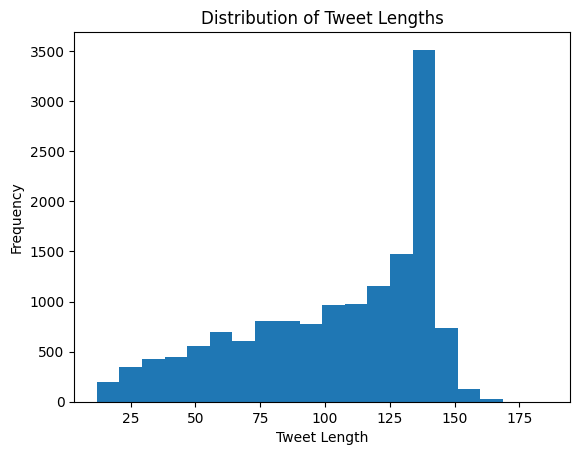
plt.title("Distribution of Tweet Lengths")

plt.xlabel("Tweet Length")

plt.ylabel("Frequency")

plt.show()

OUT:



from sklearn.feature\_extraction.text import CountVectorizer # top 20 most common words function

def common\_words(rev):

texts = data[data['airline\_sentiment'] == rev]['text'].values

vec = CountVectorizer(stop\_words='english').fit(texts)

bag\_of\_words = vec.transform(texts)

sum\_words = bag\_of\_words.sum(axis=0)

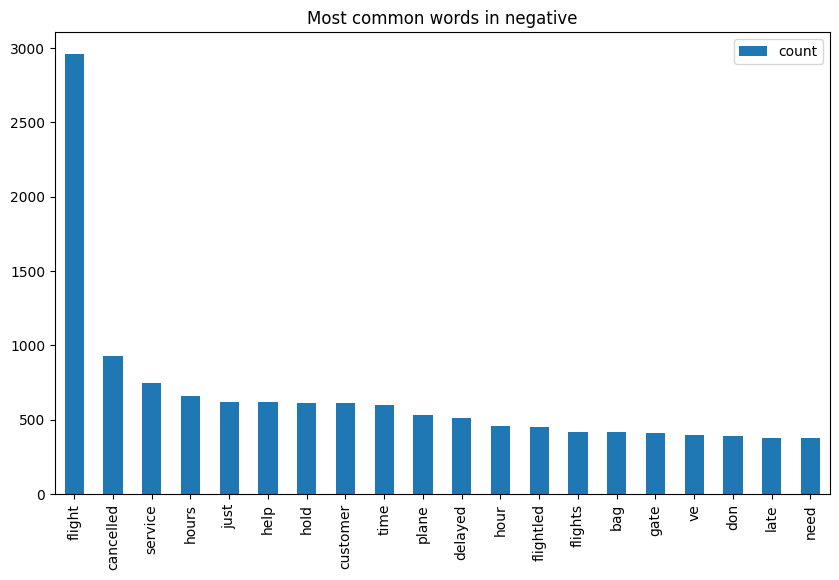
words\_freq = [(word, sum\_words[0, idx]) for word, idx in vec.vocabulary\_.items()]

return sorted(words\_freq, key = lambda x: x[1], reverse=True)[:20]

top\_neg = dict(common\_words('negative'))

pd.DataFrame.from\_dict(top\_neg, orient='index', columns=['count']).plot(kind='bar', figsize=(10, 6),title = 'Most common words in negative');

OUT:

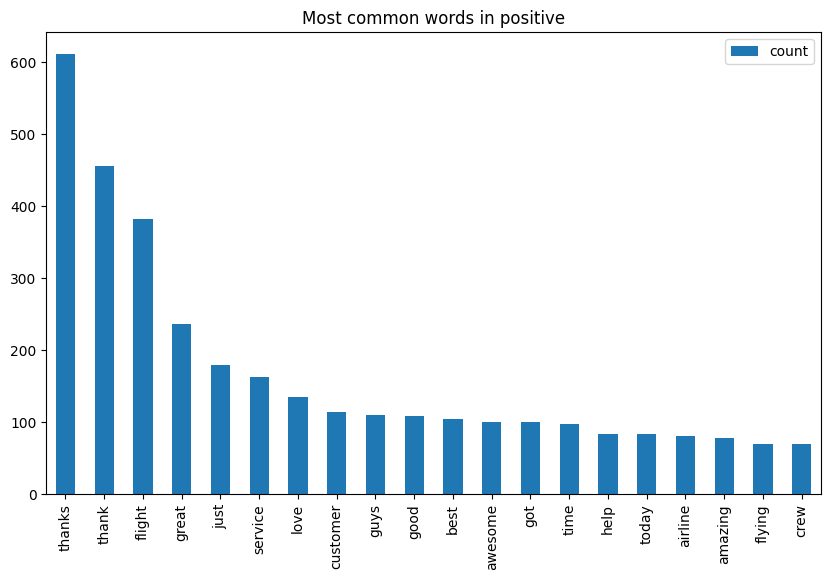


#positive words

top\_neg = dict(common\_words('positive'))

pd.DataFrame.from\_dict(top\_neg, orient='index', columns=['count']).plot(kind='bar', figsize=(10, 6),title = 'Most common words in positive');

OUT:



!pip install datasets

x = data['text']

y = data['labels']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=101)

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

tfv = TfidfVectorizer(min\_df=3, max\_features=None, strip\_accents='unicode', analyzer='word',token\_pattern=r'\w{1,}', ngram\_range=(1, 2), use\_idf=1,smooth\_idf=1,sublinear\_tf=1, stop\_words = 'english')

tfv.fit(X\_train)

TfidfVectorizer(min\_df=3, ngram\_range=(1, 2), smooth\_idf=1, stop\_words='english', strip\_accents='unicode', sublinear\_tf=1, token\_pattern='\\w{1,}', use\_idf=1)

X\_train\_tfv = tfv.transform(X\_train)

X\_test\_tfv = tfv.transform(X\_test)

X\_train\_tfv

from sklearn.svm import LinearSVC

svc = LinearSVC()

svc.fit(X\_train\_tfv,y\_train)

LinearSVC()

from sklearn.pipeline import Pipeline

pipe = Pipeline([('tfidf',TfidfVectorizer()), ('svc',LinearSVC())])

pipe.fit(data['text'],data['labels'])

Pipeline(steps=[('tfidf', TfidfVectorizer()), ('svc', LinearSVC())])

new\_positive\_tweet = ['good flight']

pipe.predict(new\_positive\_tweet)

new\_negative\_tweet = ['bad flight']

pipe.predict(new\_negative\_tweet)

new\_neutral\_tweet = ['ok flight']

pipe.predict(new\_neutral\_tweet)

##pandasDF --> Hugging Face dataset

from datasets import Dataset

dataset = {"text": data["text"].tolist(), "labels":data["labels"].tolist()}

dataset = Dataset.from\_dict(dataset)

dataset = dataset.train\_test\_split(train\_size=0.8, seed=101)

dataset

OUT:

DatasetDict({

train: Dataset({

features: ['text', 'labels'],

num\_rows: 11712

})

test: Dataset({

features: ['text', 'labels'],

num\_rows: 2928

})

})

import tensorflow as tf

from transformers import TFAutoModelForSequenceClassification, AutoTokenizer, AutoConfig,DataCollatorWithPadding

from scipy.special import softmax

checkpoint = 'cardiffnlp/twitter-roberta-base-sentiment-latest'

batch\_size = 16

num\_epochs = 5

config = AutoConfig.from\_pretrained(checkpoint)

tokenizer = AutoTokenizer.from\_pretrained(checkpoint)

model = TFAutoModelForSequenceClassification.from\_pretrained(checkpoint, num\_labels=3)

def tokenize\_function(example):

return tokenizer(example['text'], truncation=True, max\_length = 35)

tokenized\_datasets = dataset.map(tokenize\_function, batched=True,)

data\_collator = DataCollatorWithPadding(tokenizer=tokenizer, return\_tensors="tf")

from transformers import pipeline

classifier = pipeline("sentiment-analysis",tokenizer=tokenizer,model=model)

predicted\_labels = []

for text in X\_test:

result = classifier(text)

predicted\_label = result[0]['label']

predicted\_labels.append(predicted\_label)

df = pd.DataFrame(X\_test)

df['predictions'] = predicted\_labels

df['labels'] = df["predictions"].apply(lambda x: 0 if x == "negative" else 1

if x == "neutral" else 2)

df.head()

OUT:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **text** | **predictions** | **Labels** |
| **4814** | thanks very excited to see it d | positive | 2 |
| **150** | does that mean you don t have a policy for des... | neutral | 1 |
| **5322** | any official word whether flight from bwi to m... | neutral | 1 |
| **4885** | i miss mine terribly a for my th anniversary w... | neutral | 1 |
| **7504** | at what time all these passengers were sitting... | neutral | 1 |

from sklearn.metrics import confusion\_matrix

import seaborn as sns

print(classification\_report(y\_test,df['labels']))

OUT:  
 precision recall f1-score support

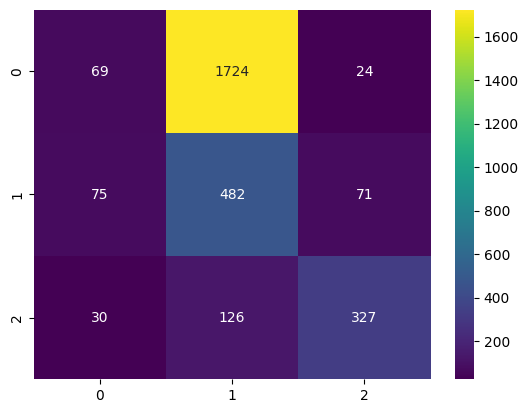
0 0.40 0.04 0.07 1817

1 0.21 0.77 0.33 628

2 0.77 0.68 0.72 483

sns.heatmap(confusion\_matrix(y\_test,df['labels']),cmap='viridis',annot=True,fmt='d')

OUT:  
<Axes: >



new\_tweet = ['amazing product ']

classifier(new\_positive\_tweet)

OUT:

[{'label': 'positive', 'score': 0.402895987033844}]

new\_tweet = ['worst experience in flight']

classifier(new\_negative\_tweet)

OUT:

[{'label': 'negative', 'score': 0.3756020665168762}]

new\_tweet = ['ok flight']

classifier(new\_neutral\_tweet)

OUT:

[{'label': 'negative', 'score': 0.3681403398513794}]

**CONCLUSION:**

Sentiment analysis of Twitter US airline data using the

BERT model is a powerful and effective tool for understanding

customer opinions and emotions in the airline industry. This

approach allows airlines to gain valuable insights into passenger

sentiment, which can be pivotal for various aspects of their

operations and customer service:

1**. Improved Customer Service**: By monitoring sentiment,

airlines can proactively address customer concerns and issues,

leading to better customer experiences.

2. **Crisis Management**: Sentiment analysis using BERT can

help airlines identify and respond to potential PR crises quickly.

3**. Marketing and Campaigns**: Airlines can fine-tune their

marketing strategies based on the sentiments expressed by

customers on social media, enabling more targeted and resonant

campaigns.

4. **Product and Service Enhancement**: Understanding

customer sentiment provides valuable feedback for improving

in-flight services, amenities, and operational aspects.

5. **Real-time Feedback Loop**: The use of BERT in sentiment

analysis ensures that airlines have access to real-time feedback,

enabling them to adapt swiftly to customer preferences and

concerns.

In essence, sentiment analysis using BERT is a vital tool for

airlines to gauge and react to customer sentiment, thereby

enhancing customer satisfaction, refining marketing strategies,

and ultimately improving their overall services. It demonstrates

the power of NLP and machine learning in gaining insights from

vast social media data.

Sentiment analysis for marketing is a valuable tool for understanding customer perceptions of competitor products. By following the outlined design thinking process, we can gather, preprocess, analyze, and visualize customer feedback data to derive meaningful insights that drive informed business decisions and marketing strategies.