**SENTIMENT ANALYSIS FOR MARKETING**

Introduction

In this document, we will outline the complete steps for transforming a design concept into an innovative solution. Innovation is not just about creating something new; it's about addressing problems, creating value, and positively impacting users or stakeholders. This guide will help you navigate the journey from a design idea to a transformative innovation.

Step 1: Define the Problem

Description: Before you can innovate, you need a clear understanding of the problem you're solving. Define the problem statement, its scope, and the pain points it causes.

Actions:

• Conduct in-depth research to gather data and insights related to the problem.

• Engage with end-users or stakeholders to understand their perspectives.

• Narrow down the problem statement to make it specific and actionable.

Step 2: Ideation and Conceptualization

Description: Brainstorm ideas and concepts that can potentially address the defined problem. Encourage creative thinking and consider multiple perspectives.

Actions:

• Host brainstorming sessions with a diverse team to generate a wide range of ideas.

• Prioritize and refine these ideas based on feasibility, impact, and alignment with the problem.

Step 3: Prototyping

Description: Create prototypes or proof-of-concepts for the selected concepts to visualize how they might work and gather feedback.

Actions:

• Develop low-fidelity prototypes to quickly test and iterate on the concepts.

• Share these prototypes with stakeholders, end-users, or focus groups to gather valuable insights.

Step 4: Validation

Description: Validate the prototypes to ensure that they effectively solve the problem and align with user needs and expectations.

Actions:

• Conduct user testing and gather feedback on the prototypes.

• Make necessary adjustments and refinements based on user feedback.

Step 5: Develop a Minimum Viable Product (MVP)

Description: Create a minimum viable product (MVP) that represents a functional version of the solution with essential features.

Actions:

• Define the core features and functionalities required for the MVP.

• Develop the MVP with a focus on speed and efficiency.

Step 6: Testing and Iteration

Description: Test the MVP rigorously, gather user feedback, and iterate on the solution to improve its effectiveness.

Actions:

• Conduct extensive testing to identify and address any bugs or issues.

• Continue to gather user feedback and make iterative improvements.

Step 7: Scaling and Implementation

Description: Once the MVP is stable and refined, prepare for the full-scale implementation of the solution.

Actions:

• Develop a detailed implementation plan, including resource allocation and timelines.

• Collaborate with relevant teams or stakeholders to ensure a smooth rollout.

Step 8: Monitoring and Feedback Loop

Description: Implement systems for continuous monitoring and feedback to ensure that the solution remains effective and relevant.

Actions:

• Set up monitoring tools to track key performance metrics.

• Encourage ongoing feedback from users and stakeholders for further improvements.

Step 9: Documentation and Knowledge Sharing

Description: Document the entire innovation process, from design to implementation, to capture lessons learned and best practices.

Actions:

• Create comprehensive documentation, including design documents, user manuals, and implementation guides.

• Share this knowledge within the organization to promote a culture of innovation.

Step 10: Measure Impact

Description: Continuously measure and evaluate the impact of the innovation in terms of solving the original problem and creating value.

Actions:

•Analyze data and metrics to assess the solution's impact.

•Share success stories and outcomes to inspire others and attract potential users or customers.

1. Ensemble Methods:

Ensemble methods combine multiple machine learning models to improve predictive performance. You can use techniques like:

a. Voting Classifier:

Combine multiple Naïve Bayes classifiers or other algorithms and let them "vote" on the sentiment prediction. This can help reduce bias and improve overall accuracy.

• from sklearn.ensemble import VotingClassifier

• # Create an ensemble of classifiers (e.g., Naïve Bayes, Random Forest)

• ensemble\_classifier = VotingClassifier(estimators=[('nb', nb\_classifier), ('rf', random\_forest\_classifier)], voting='hard')

• # Train the ensemble classifier on your TF-IDF vectors

• ensemble\_classifier.fit(X\_train\_tfidf, y\_train)

• # Make predictions with the ensemble classifier

• y\_pred\_ensemble = ensemble\_classifier.predict(X\_test\_tfidf)

b. Stacking:

Build a meta-classifier that takes predictions from multiple base models as input and learns to make a final prediction.

• from sklearn.ensemble import StackingClassifier

• # Create a stack of classifiers (e.g., Naïve Bayes, Random Forest)

• stacked\_classifier = StackingClassifier(estimators=[('nb', nb\_classifier), ('rf',random\_forest\_classifier)], final\_estimator=LogisticRegression())

• # Train the stacked classifier on your TF-IDF vectors

• stacked\_classifier.fit(X\_train\_tfidf, y\_train)

• # Make predictions with the stacked classifier

• y\_pred\_stacked = stacked\_classifier.predict(X\_test\_tfidf)

2. Deep Learning Architectures:

Deep learning can capture complex patterns in text data. You can experiment with neural networks for sentiment analysis:

a. Recurrent Neural Networks (RNNs):

RNNs are suitable for sequential data like text. Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) layers can be used for sentiment analysis.

• from keras.models import Sequential

• from keras.layers import Embedding, LSTM, Dense, Dropout

• model = Sequential()

• model.add(Embedding(input\_dim=num\_words, output\_dim=embedding\_dim, input\_length=max\_sequence\_length))

• model.add(LSTM(128))

• model.add(Dense(1, activation='sigmoid'))

• # Compile the model and train it on your text data

b. Transformer-based Models:

State-of-the-art models like BERT and RoBERTa have achieved remarkable results in natural language understanding tasks, including sentiment analysis.

• You can use Hugging Face's Transformers library to fine-tune pre-trained models:

• from transformers import BertTokenizer, BertForSequenceClassification, AdamW

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

• model = BertForSequenceClassification.from\_pretrained('bert-base-uncased', num\_labels=2)

• # Tokenize and format your text data

• # Define optimizer and loss function

• # Fine-tune the model on your dataset

3. Evaluation and Hyperparameter Tuning:

After implementing these techniques, it's crucial to evaluate their performance using metrics like accuracy, precision, recall, and F1-score. Hyperparameter tuning can further optimize the models for your specific problem.

4. Model Deployment:

Once you have a well-performing sentiment analysis model, you can deploy it in your application or system to make real-time predictions on customer reviews.

By incorporating ensemble methods, deep learning architectures, and pre-trained models, you can significantly enhance the accuracy and robustness of your sentiment prediction system. Remember to fine-tune these techniques based on your specific dataset and problem requirements.

Fine-Tuning Pre-trained Models (BERT):

For fine-tuning the BERT model, you can monitor the training progress and evaluate the model's performance on the test data. You can print the training loss during training and use classification metrics for evaluation:

# During training, you can print the training loss

print("Training Loss:", loss)

# After fine-tuning, you can evaluate the model and print classification metrics

from sklearn.metrics import classification\_report

print("Classification Report:")

print(classification\_report(y\_test, predicted\_labels))

The output during training will show you the training loss decreasing over epochs, indicating that the model is learning. The classification report will provide detailed metrics like precision, recall, F1-score, and support for each class (e.g., positive and negative sentiment).

Codes:

Import pandas as pd

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

From sklearn.feature\_extraction.text import TfidfVectorizer

From sklearn.naive\_bayes import MultinomialNB

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

Import matplotlib.pyplot as plt

# Load and preprocess data

Data = pd.read\_csv(‘customer\_reviews.csv’) # Make sure the file path is correct

X = data[‘text’]

Y = data[‘sentiment’]

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Convert text to TF-IDF vectors

Tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# Train a Naïve Bayes classifier

Clf = MultinomialNB()

Clf.fit(X\_train\_tfidf, y\_train)

# Make predictions

Y\_pred = clf.predict(X\_test\_tfidf)

# Evaluate the model

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

Print(f’Accuracy: {accuracy}’)

Print(classification\_report(y\_test, y\_pred))

# Create a bar chart to visualize the distribution of sentiment labels

Sentiment\_counts = y.value\_counts()

Sentiment\_labels = sentiment\_counts.index

Plt.bar(sentiment\_labels, sentiment\_counts)

Plt.xlabel(‘Sentiment’)

Plt.ylabel(‘Count’)

Plt.title(‘Sentiment Distribution in Customer Reviews’)

Plt.show()

**GOAL**

The goal of this project is to develop a sentiment analysis solution for a brand’s social media presence, enabling the brand to gain insights into customer sentiment, engagement, and trends. The project will use machine learning and natural language processing (NLP) techniques to analyze customer comments and interactions on social media platforms

This project will provide valuable insights into customer sentiment and help the brand make informed decisions to enhance its marketing strategies and engagement on social media.

Certainly, let’s continue:

**1. Multilingual Support (Optional):**

- If your application needs to handle multiple languages, ensure that your sentiment analysis model can work with multilingual data. You might need to adapt or use pre-trained models for different languages.

**2. Domain-Specific Models (Optional):**

- If your application operates in a specific domain (e.g., healthcare, finance, or gaming), consider training or fine-tuning your sentiment analysis model on domain-specific data. Domain-specific models can provide more accurate insights.

**3. Error Analysis:**

- Regularly conduct error analysis to understand the limitations and weaknesses of your sentiment analysis model. This can help you identify areas for improvement.

**4. User Interface and Reporting:**

- Create a user-friendly interface or reporting system for users to access the generated insights. Dashboards and reports can provide an easy way to visualize sentiment trends and insights.

**5. A/B Testing (Optional):**

- If your sentiment analysis solution is integrated into a larger product or service, consider conducting A/B tests to assess the impact of sentiment-related features or improvements on user engagement and satisfaction.

**6. Ethical Considerations:**

- Pay attention to ethical considerations, such as bias in the training data, privacy, and the responsible use of sentiment analysis results. Implement measures to mitigate biases and respect user privacy.

**7. Data Security and Compliance:**

- Ensure that your solution complies with data protection and privacy regulations, especially when handling sensitive or personal data.

**8. Feedback Integration:**

- Implement mechanisms for users to provide feedback on sentiment analysis results. This can help you continuously improve the accuracy and relevance of the insights generated.

**9. Documentation and Training:**

- Document your sentiment analysis solution and provide training or guidance to users, especially if it’s used by a team that may not be familiar with NLP techniques.

Building a sentiment analysis solution involves a combination of NLP techniques, machine learning, and thoughtful design to extract meaningful insights from text data. Be prepared to iterate and refine your solution over time to meet the evolving needs of your users and the changing landscape of sentiment in the data you analyze.

* **Tools and technologies:**

Python, NLP libraries (NLTK, spaCy), machine learning frameworks (Scikit-Learn, TensorFlow), pre-trained language models (BERT, GPT-3).

**PROGRAM**:

frommpl\_toolkits.mplot3dimportAxes3D

fromsklearn.preprocessingimportStandardScaler

importmatplotlib.pyplotasplt*# plotting*

importnumpyasnp*# linear algebra*

importos*# accessing directory structure*

importpandasaspd*# data processing, CSV file I/O (e.g. pd.read\_csv)*

There is 1 csv file in the current version of the dataset:

In [2]:

print(os.listdir('../input'))

['database.sqlite', 'Tweets.csv']

*# Distribution graphs (histogram/bar graph) of column data*

defplotPerColumnDistribution(df,nGraphShown,nGraphPerRow):

nunique=df.nunique()

df=df[[colforcol**in**dfifnunique[col]>1**and**nunique[col]<50]]*# For displaying purposes, pick columns that have between 1 and 50 unique values*

nRow,nCol=df.shape

columnNames=list(df)

nGraphRow=(nCol+nGraphPerRow-1)/nGraphPerRow

plt.figure(num=None,figsize=(6\*nGraphPerRow,8\*nGraphRow),dpi=80,facecolor='w',edgecolor='k')

fori**in**range(min(nCol,nGraphShown)):

plt.subplot(nGraphRow,nGraphPerRow,i+1)

columnDf=df.iloc[:,i]

if(**not**np.issubdtype(type(columnDf.iloc[0]),np.number)):

valueCounts=columnDf.value\_counts()

valueCounts.plot.bar()

else:

columnDf.hist()

plt.ylabel('counts')

plt.xticks(rotation=90)

plt.title(f'**{columnNames[i]}** (column **{i}**)')

plt.tight\_layout(pad=1.0,w\_pad=1.0,h\_pad=1.0)

plt.show()

*# Correlation matrix*

defplotCorrelationMatrix(df,graphWidth):

filename=df.dataframeName

df=df.dropna('columns')*# drop columns with NaN*

df=df[[colforcol**in**dfifdf[col].nunique()>1]]*# keep columns where there are more than 1 unique values*

ifdf.shape[1]<2:

print(f'No correlation plots shown: The number of non-NaN or constant columns (**{df.shape[1]}**) is less than 2')

return

corr=df.corr()

plt.figure(num=None,figsize=(graphWidth,graphWidth),dpi=80,facecolor='w',edgecolor='k')

corrMat=plt.matshow(corr,fignum=1)

plt.xticks(range(len(corr.columns)),corr.columns,rotation=90)

plt.yticks(range(len(corr.columns)),corr.columns)

plt.gca().xaxis.tick\_bottom()

plt.colorbar(corrMat)

plt.title(f'Correlation Matrix for **{filename}**',fontsize=15)

plt.show()

*# Scatter and density plots*

defplotScatterMatrix(df,plotSize,textSize):

df=df.select\_dtypes(include=[np.number])*# keep only numerical columns*

*# Remove rows and columns that would lead to df being singular*

df=df.dropna('columns')

df=df[[colforcol**in**dfifdf[col].nunique()>1]]*# keep columns where there are more than 1 unique values*

columnNames=list(df)

iflen(columnNames)>10:*# reduce the number of columns for matrix inversion of kernel density plots*

columnNames=columnNames[:10]

df=df[columnNames]

ax=pd.plotting.scatter\_matrix(df,alpha=0.75,figsize=[plotSize,plotSize],diagonal='kde')

corrs=df.corr().values

fori,j**in**zip(\*plt.np.triu\_indices\_from(ax,k=1)):

ax[i,j].annotate('Corr. coef = **%.3f**'%corrs[i,j],(0.8,0.2),xycoords='axes fraction',ha='center',va='center',size=textSize)

plt.suptitle('Scatter and Density Plot')

plt.show()

nRowsRead=1000*# specify 'None' if want to read whole file*

*# Tweets.csv has 14640 rows in reality, but we are only loading/previewing the first 1000 rows*

df1=pd.read\_csv('../input/Tweets.csv',delimiter=',',nrows=nRowsRead)

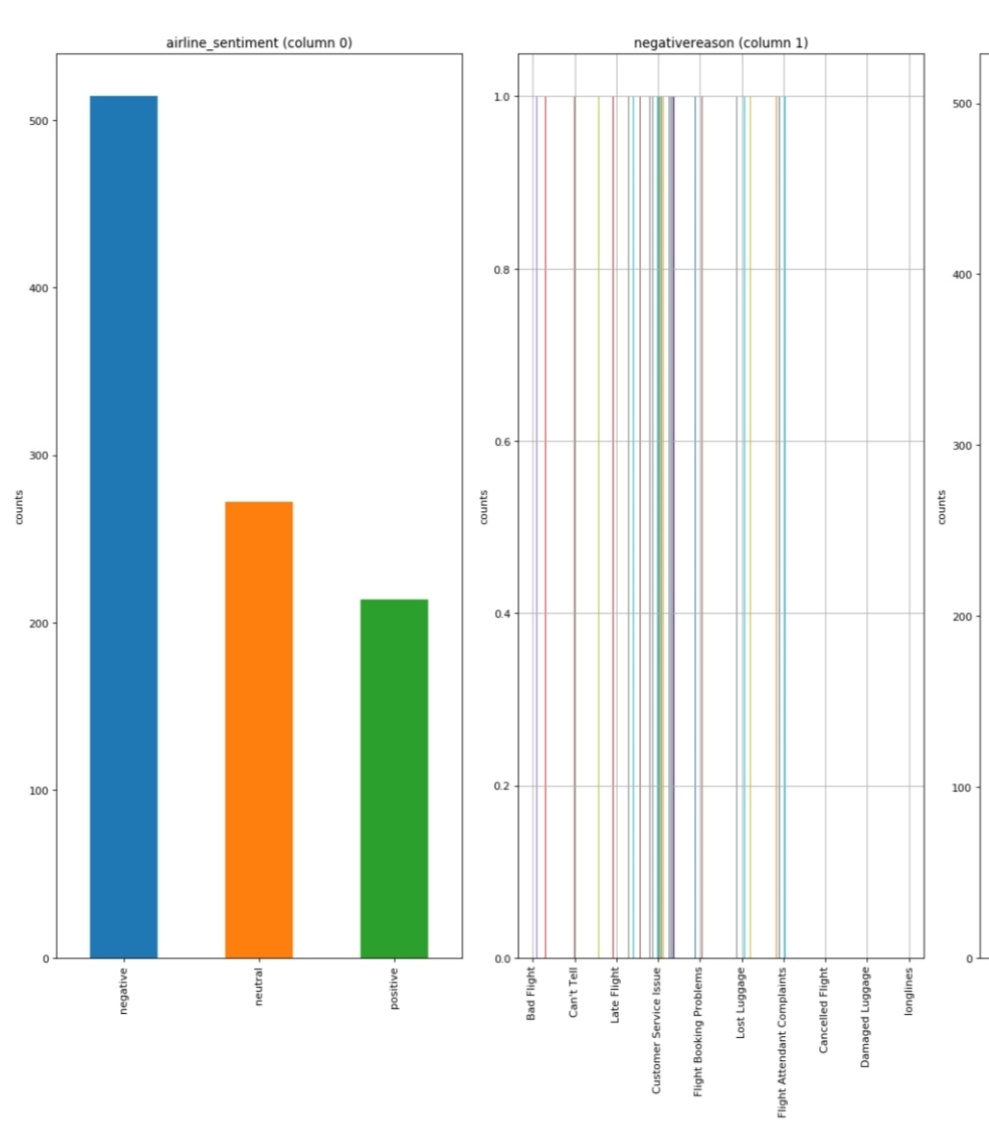
df1.dataframeName='Tweets.csv'

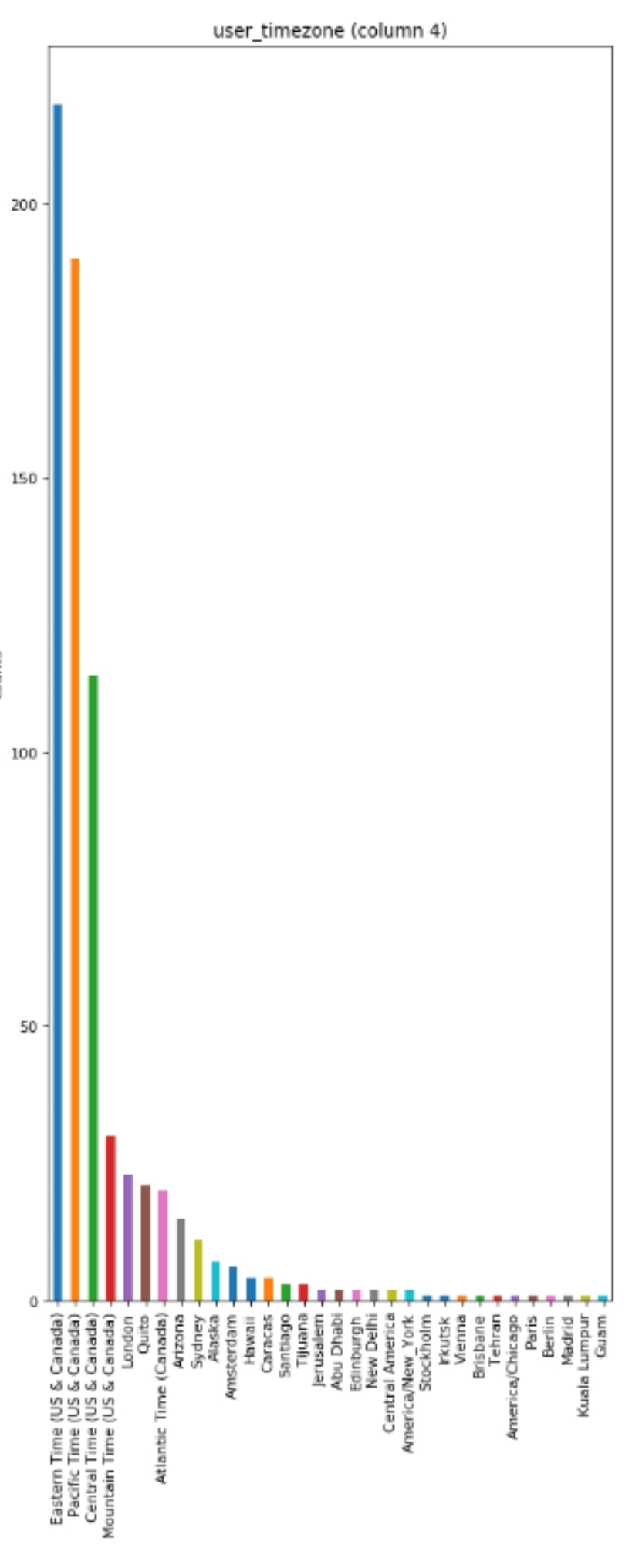
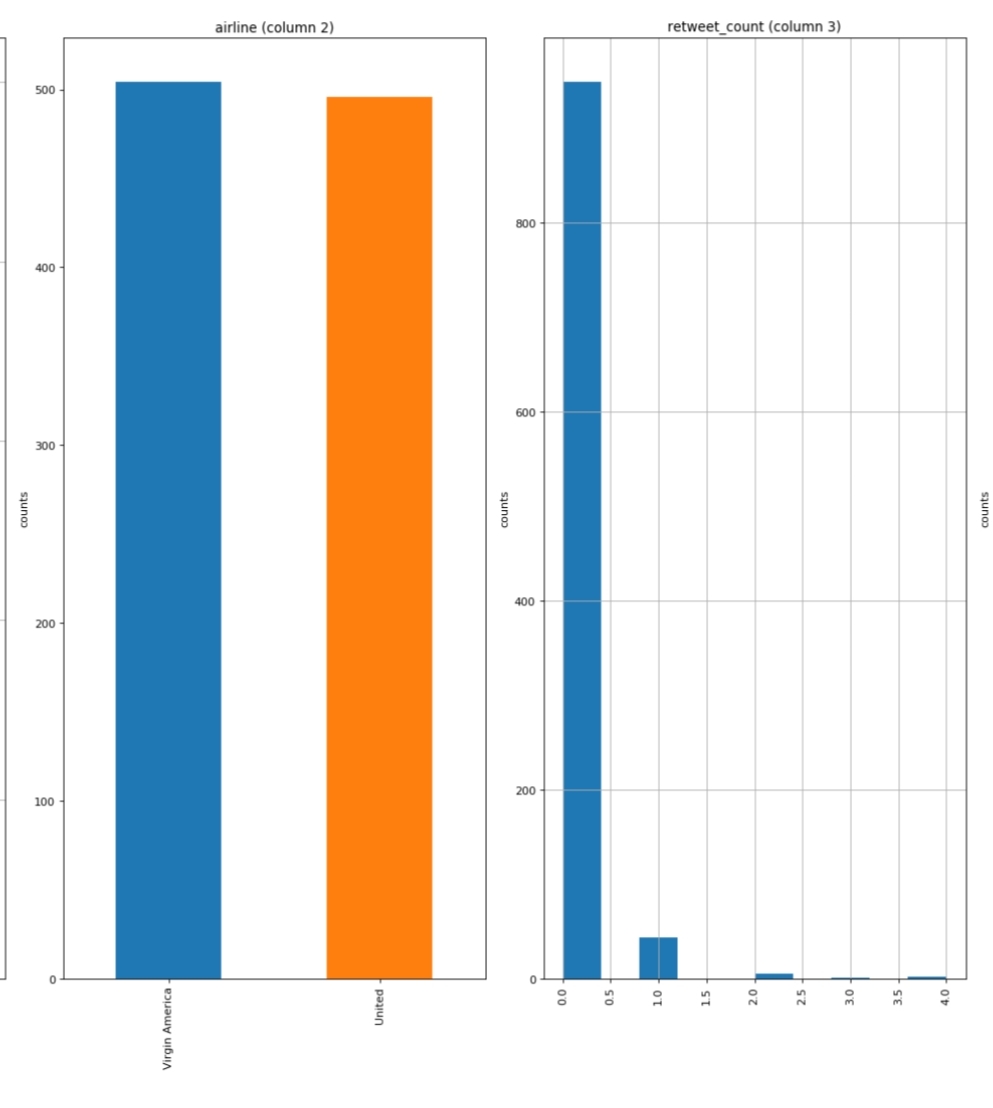
nRow,nCol=df1.shape

print(f'There are **{nRow}** rows and **{nCol}** columns')

df1.head(5)



 plotPerColumnDistribution(df1, 10, 5)

****

****

**Data Preprocessing:**

1. Clean and preprocess the social media data:
2. Remove special characters, URLs, and emojis.
3. Tokenize the text into words.



**Dataset :**

Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as “late flight” or “rude service”).

**Program**:

{

"cells": [

{

"cell\_type": "markdown",

"id": "65d1b629",

"metadata": {

"papermill": {

"duration": 0.005039,

"end\_time": "2023-08-16T14:15:41.969489",

"exception": false,

"start\_time": "2023-08-16T14:15:41.964450",

"status": "completed"

},

"tags": []

},

"source": [

"# Importing Libraries"

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"id": "562d3cb8",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:15:41.980120Z",

"iopub.status.busy": "2023-08-16T14:15:41.979729Z",

"iopub.status.idle": "2023-08-16T14:16:04.435508Z",

"shell.execute\_reply": "2023-08-16T14:16:04.433961Z"

},

"papermill": {

"duration": 22.463896,

"end\_time": "2023-08-16T14:16:04.437812",

"exception": false,

"start\_time": "2023-08-16T14:15:41.973916",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5\n",

" warnings.warn(f\"A NumPy version >={np\_minversion} and <{np\_maxversion}\"\n"

]

},

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[nltk\_data] Error loading stopwords: <urlopen error [Errno -3]\n",

"[nltk\_data] Temporary failure in name resolution>\n"

]

}

],

"source": [

"# Basic libraries\n",

"import numpy as np\n",

"import pandas as pd\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"%matplotlib inline\n",

"import pickle\n",

"import warnings\n",

"warnings.filterwarnings(action='ignore')\n",

"\n",

"# nltk\n",

"import nltk\n",

"nltk.download('stopwords')\n",

"\n",

"## Preprocessing libraries\n",

"import re\n",

"from nltk.corpus import stopwords\n",

"from nltk.stem.porter import PorterStemmer\n",

"from sklearn.feature\_extraction.text import TfidfVectorizer\n",

"\n",

"# For Model training\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.naive\_bayes import BernoulliNB\n",

"from sklearn.svm import LinearSVC # a variant of SVC optimized for large datasets\n",

"\n",

"# Metrics for accuracy\n",

"from sklearn.metrics import accuracy\_score,confusion\_matrix, classification\_report"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "30c38b25",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.448914Z",

"iopub.status.busy": "2023-08-16T14:16:04.448563Z",

"iopub.status.idle": "2023-08-16T14:16:04.598104Z",

"shell.execute\_reply": "2023-08-16T14:16:04.596834Z"

},

"papermill": {

"duration": 0.157462,

"end\_time": "2023-08-16T14:16:04.600149",

"exception": false,

"start\_time": "2023-08-16T14:16:04.442687",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>tweet\_id</th>\n",

" <th>airline\_sentiment</th>\n",

" <th>airline\_sentiment\_confidence</th>\n",

" <th>negativereason</th>\n",

" <th>negativereason\_confidence</th>\n",

" <th>airline</th>\n",

" <th>airline\_sentiment\_gold</th>\n",

" <th>name</th>\n",

" <th>negativereason\_gold</th>\n",

" <th>retweet\_count</th>\n",

" <th>text</th>\n",

" <th>tweet\_coord</th>\n",

" <th>tweet\_created</th>\n",

" <th>tweet\_location</th>\n",

" <th>user\_timezone</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>570306133677760513</td>\n",

" <td>neutral</td>\n",

" <td>1.0000</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>cairdin</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica What @dhepburn said.</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:35:52 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Eastern Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>570301130888122368</td>\n",

" <td>positive</td>\n",

" <td>0.3486</td>\n",

" <td>NaN</td>\n",

" <td>0.0000</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica plus you've added commercials t...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:59 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>570301083672813571</td>\n",

" <td>neutral</td>\n",

" <td>0.6837</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>yvonnalynn</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica I didn't today... Must mean I n...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:48 -0800</td>\n",

" <td>Lets Play</td>\n",

" <td>Central Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>570301031407624196</td>\n",

" <td>negative</td>\n",

" <td>1.0000</td>\n",

" <td>Bad Flight</td>\n",

" <td>0.7033</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica it's really aggressive to blast...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:15:36 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>570300817074462722</td>\n",

" <td>negative</td>\n",

" <td>1.0000</td>\n",

" <td>Can't Tell</td>\n",

" <td>1.0000</td>\n",

" <td>Virgin America</td>\n",

" <td>NaN</td>\n",

" <td>jnardino</td>\n",

" <td>NaN</td>\n",

" <td>0</td>\n",

" <td>@VirginAmerica and it's a really big bad thing...</td>\n",

" <td>NaN</td>\n",

" <td>2015-02-24 11:14:45 -0800</td>\n",

" <td>NaN</td>\n",

" <td>Pacific Time (US &amp; Canada)</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" tweet\_id airline\_sentiment airline\_sentiment\_confidence \\\n",

"0 570306133677760513 neutral 1.0000 \n",

"1 570301130888122368 positive 0.3486 \n",

"2 570301083672813571 neutral 0.6837 \n",

"3 570301031407624196 negative 1.0000 \n",

"4 570300817074462722 negative 1.0000 \n",

"\n",

" negativereason negativereason\_confidence airline \\\n",

"0 NaN NaN Virgin America \n",

"1 NaN 0.0000 Virgin America \n",

"2 NaN NaN Virgin America \n",

"3 Bad Flight 0.7033 Virgin America \n",

"4 Can't Tell 1.0000 Virgin America \n",

"\n",

" airline\_sentiment\_gold name negativereason\_gold retweet\_count \\\n",

"0 NaN cairdin NaN 0 \n",

"1 NaN jnardino NaN 0 \n",

"2 NaN yvonnalynn NaN 0 \n",

"3 NaN jnardino NaN 0 \n",

"4 NaN jnardino NaN 0 \n",

"\n",

" text tweet\_coord \\\n",

"0 @VirginAmerica What @dhepburn said. NaN \n",

"1 @VirginAmerica plus you've added commercials t... NaN \n",

"2 @VirginAmerica I didn't today... Must mean I n... NaN \n",

"3 @VirginAmerica it's really aggressive to blast... NaN \n",

"4 @VirginAmerica and it's a really big bad thing... NaN \n",

"\n",

" tweet\_created tweet\_location user\_timezone \n",

"0 2015-02-24 11:35:52 -0800 NaN Eastern Time (US & Canada) \n",

"1 2015-02-24 11:15:59 -0800 NaN Pacific Time (US & Canada) \n",

"2 2015-02-24 11:15:48 -0800 Lets Play Central Time (US & Canada) \n",

"3 2015-02-24 11:15:36 -0800 NaN Pacific Time (US & Canada) \n",

"4 2015-02-24 11:14:45 -0800 NaN Pacific Time (US & Canada) "

]

},

"execution\_count": 2,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Reading our dataset\n",

"df = pd.read\_csv('/kaggle/input/twitter-airline-sentiment/Tweets.csv')\n",

"df.head()"

]

},

{

"cell\_type": "markdown",

"id": "8ca1f7ce",

"metadata": {

"papermill": {

"duration": 0.004696,

"end\_time": "2023-08-16T14:16:04.610061",

"exception": false,

"start\_time": "2023-08-16T14:16:04.605365",

"status": "completed"

},

"tags": []

},

"source": [

"# EDA"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "e3526faf",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.621668Z",

"iopub.status.busy": "2023-08-16T14:16:04.620987Z",

"iopub.status.idle": "2023-08-16T14:16:04.654437Z",

"shell.execute\_reply": "2023-08-16T14:16:04.653076Z"

},

"papermill": {

"duration": 0.041952,

"end\_time": "2023-08-16T14:16:04.656834",

"exception": false,

"start\_time": "2023-08-16T14:16:04.614882",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"text/plain": [

"tweet\_id 0\n",

"airline\_sentiment 0\n",

"airline\_sentiment\_confidence 0\n",

"negativereason 5462\n",

"negativereason\_confidence 4118\n",

"airline 0\n",

"airline\_sentiment\_gold 14600\n",

"name 0\n",

"negativereason\_gold 14608\n",

"retweet\_count 0\n",

"text 0\n",

"tweet\_coord 13621\n",

"tweet\_created 0\n",

"tweet\_location 4733\n",

"user\_timezone 4820\n",

"dtype: int64"

]

},

"execution\_count": 3,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.isnull().sum()"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"id": "7b152a79",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.669876Z",

"iopub.status.busy": "2023-08-16T14:16:04.669476Z",

"iopub.status.idle": "2023-08-16T14:16:04.867707Z",

"shell.execute\_reply": "2023-08-16T14:16:04.866829Z"

},

"papermill": {

"duration": 0.207536,

"end\_time": "2023-08-16T14:16:04.869769",

"exception": false,

"start\_time": "2023-08-16T14:16:04.662233",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 700x300 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Checking the distribution of airlines\n",

"plt.figure(figsize=(7,3))\n",

"sns.countplot(data=df,x='airline', palette=['#1f78b4', '#33a02c', '#e31a1c', '#ff7f00', '#6a3d9a', '#a6cee3'])\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "a7fa1354",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:04.884219Z",

"iopub.status.busy": "2023-08-16T14:16:04.883838Z",

"iopub.status.idle": "2023-08-16T14:16:05.018558Z",

"shell.execute\_reply": "2023-08-16T14:16:05.017629Z"

},

"papermill": {

"duration": 0.143801,

"end\_time": "2023-08-16T14:16:05.020630",

"exception": false,

"start\_time": "2023-08-16T14:16:04.876829",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 700x300 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Seeing the distribution of positive and negative tweet reviews in target column\n",

"plt.figure(figsize=(7,3))\n",

"sns.countplot(data=df,x='airline\_sentiment',palette=['yellow', 'green','red'])\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "f4daa7f5",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:05.040098Z",

"iopub.status.busy": "2023-08-16T14:16:05.038024Z",

"iopub.status.idle": "2023-08-16T14:16:05.287821Z",

"shell.execute\_reply": "2023-08-16T14:16:05.286965Z"

},

"papermill": {

"duration": 0.262129,

"end\_time": "2023-08-16T14:16:05.290275",

"exception": false,

"start\_time": "2023-08-16T14:16:05.028146",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"data": {

"image/png":

"text/plain": [

"<Figure size 800x800 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"# Calculate the value counts for each negative reason\n",

"value\_counts = df['negativereason'].value\_counts()\n",

"\n",

"# Create a donut-like pie chart using matplotlib and seaborn\n",

"plt.figure(figsize=(8, 8))\n",

"labels = value\_counts.index\n",

"values = value\_counts.values\n",

"colors = sns.color\_palette('pastel')[0:len(labels)] # Use pastel colors for the chart\n",

"plt.pie(values, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140, wedgeprops=dict(width=0.3))\n",

"plt.title('Overall distribution for negative reasons')\n",

"plt.axis('equal') # Equal aspect ratio ensures the pie chart is drawn as a circle.\n",

"plt.show()"

]

},

{

"cell\_type": "markdown",

"id": "8b5b22c3",

"metadata": {

"papermill": {

"duration": 0.0083,

"end\_time": "2023-08-16T14:16:05.307411",

"exception": false,

"start\_time": "2023-08-16T14:16:05.299111",

"status": "completed"

},

"tags": []

},

"source": [

"## Data clearing and preprocessing of Text"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "fd6bf500",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:05.322523Z",

"iopub.status.busy": "2023-08-16T14:16:05.321899Z",

"iopub.status.idle": "2023-08-16T14:16:37.607161Z",

"shell.execute\_reply": "2023-08-16T14:16:37.605444Z"

},

"papermill": {

"duration": 32.295432,

"end\_time": "2023-08-16T14:16:37.609494",

"exception": false,

"start\_time": "2023-08-16T14:16:05.314062",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"corpus = []\n",

"ps=PorterStemmer()\n",

"for i in range(len(df)):\n",

" # Removing special characters from text(message)\n",

" review = re.sub('[^a-zA-Z]', ' ', df['text'][i])\n",

" \n",

" # Converting entire text into lower case\n",

" review = review.lower()\n",

" \n",

" # Splitting our text into words\n",

" review = review.split()\n",

" \n",

" # Stemming and removing stopwords\n",

" review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]\n",

" \n",

" # Joining all the words into a comple text\n",

" review = ' '.join(review)\n",

" \n",

" # Appending each text into the list corpus\n",

" corpus.append(review) "

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "e65effa0",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:37.624097Z",

"iopub.status.busy": "2023-08-16T14:16:37.623745Z",

"iopub.status.idle": "2023-08-16T14:16:37.629458Z",

"shell.execute\_reply": "2023-08-16T14:16:37.628045Z"

},

"papermill": {

"duration": 0.01555,

"end\_time": "2023-08-16T14:16:37.631758",

"exception": false,

"start\_time": "2023-08-16T14:16:37.616208",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating the Bag of Words model\n",

"cv = TfidfVectorizer(ngram\_range=(1,2), max\_features=500000)"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"id": "0c5f5e86",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:37.646135Z",

"iopub.status.busy": "2023-08-16T14:16:37.645724Z",

"iopub.status.idle": "2023-08-16T14:16:38.212149Z",

"shell.execute\_reply": "2023-08-16T14:16:38.211211Z"

},

"papermill": {

"duration": 0.576022,

"end\_time": "2023-08-16T14:16:38.214308",

"exception": false,

"start\_time": "2023-08-16T14:16:37.638286",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# We will use X as independent feature section\n",

"X = cv.fit\_transform(corpus)\n",

"# We will use y as dependent feature section\n",

"y=df['airline\_sentiment']"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "5caa204f",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.229410Z",

"iopub.status.busy": "2023-08-16T14:16:38.228581Z",

"iopub.status.idle": "2023-08-16T14:16:38.343176Z",

"shell.execute\_reply": "2023-08-16T14:16:38.341777Z"

},

"papermill": {

"duration": 0.124498,

"end\_time": "2023-08-16T14:16:38.345618",

"exception": false,

"start\_time": "2023-08-16T14:16:38.221120",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"No. of feature\_words: 91436\n"

]

}

],

"source": [

"print('No. of feature\_words: ', len(cv.get\_feature\_names\_out()))"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "7b6dd5ff",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.360704Z",

"iopub.status.busy": "2023-08-16T14:16:38.359761Z",

"iopub.status.idle": "2023-08-16T14:16:38.613891Z",

"shell.execute\_reply": "2023-08-16T14:16:38.613192Z"

},

"papermill": {

"duration": 0.263732,

"end\_time": "2023-08-16T14:16:38.615977",

"exception": false,

"start\_time": "2023-08-16T14:16:38.352245",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating a pickle file for the TfidfVectorizer\n",

"with open('cv-transform.pkl', 'wb') as f:\n",

" pickle.dump(cv, f)"

]

},

{

"cell\_type": "markdown",

"id": "323315dc",

"metadata": {

"papermill": {

"duration": 0.006117,

"end\_time": "2023-08-16T14:16:38.628510",

"exception": false,

"start\_time": "2023-08-16T14:16:38.622393",

"status": "completed"

},

"tags": []

},

"source": [

"# Model Training"

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"id": "4ccb8d92",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.643751Z",

"iopub.status.busy": "2023-08-16T14:16:38.642446Z",

"iopub.status.idle": "2023-08-16T14:16:38.653480Z",

"shell.execute\_reply": "2023-08-16T14:16:38.652214Z"

},

"papermill": {

"duration": 0.020821,

"end\_time": "2023-08-16T14:16:38.655725",

"exception": false,

"start\_time": "2023-08-16T14:16:38.634904",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Train Test Split\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 0)"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"id": "3975b93f",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.671110Z",

"iopub.status.busy": "2023-08-16T14:16:38.669912Z",

"iopub.status.idle": "2023-08-16T14:16:38.675684Z",

"shell.execute\_reply": "2023-08-16T14:16:38.674475Z"

},

"papermill": {

"duration": 0.015633,

"end\_time": "2023-08-16T14:16:38.677952",

"exception": false,

"start\_time": "2023-08-16T14:16:38.662319",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Training using three algorithms, let's see which will give us better result\n",

"model1=LogisticRegression()\n",

"model2=BernoulliNB()\n",

"model3=LinearSVC()\n",

"model=[model1, model2, model3]"

]

},

{

"cell\_type": "code",

"execution\_count": 14,

"id": "884e8464",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:38.693160Z",

"iopub.status.busy": "2023-08-16T14:16:38.692774Z",

"iopub.status.idle": "2023-08-16T14:16:44.368158Z",

"shell.execute\_reply": "2023-08-16T14:16:44.367314Z"

},

"papermill": {

"duration": 5.685399,

"end\_time": "2023-08-16T14:16:44.370070",

"exception": false,

"start\_time": "2023-08-16T14:16:38.684671",

"status": "completed"

},

"tags": []

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"M-O-D-E-L : 1\n",

"Confusion matrix : \n",

" [[2694 532 285]\n",

" [ 77 351 81]\n",

" [ 17 36 319]]\n",

"Accuracy score : 0.7659380692167578\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 0.97 0.77 0.86 3511\n",

" neutral 0.38 0.69 0.49 509\n",

" positive 0.47 0.86 0.60 372\n",

"\n",

" accuracy 0.77 4392\n",

" macro avg 0.60 0.77 0.65 4392\n",

"weighted avg 0.86 0.77 0.79 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n",

"M-O-D-E-L : 2\n",

"Confusion matrix : \n",

" [[2780 850 670]\n",

" [ 8 69 13]\n",

" [ 0 0 2]]\n",

"Accuracy score : 0.6491347905282332\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 1.00 0.65 0.78 4300\n",

" neutral 0.08 0.77 0.14 90\n",

" positive 0.00 1.00 0.01 2\n",

"\n",

" accuracy 0.65 4392\n",

" macro avg 0.36 0.80 0.31 4392\n",

"weighted avg 0.98 0.65 0.77 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n",

"M-O-D-E-L : 3\n",

"Confusion matrix : \n",

" [[2620 428 197]\n",

" [ 135 426 100]\n",

" [ 33 65 388]]\n",

"Accuracy score : 0.7818761384335154\n",

"Classification Report : \n",

" precision recall f1-score support\n",

"\n",

" negative 0.94 0.81 0.87 3245\n",

" neutral 0.46 0.64 0.54 661\n",

" positive 0.57 0.80 0.66 486\n",

"\n",

" accuracy 0.78 4392\n",

" macro avg 0.66 0.75 0.69 4392\n",

"weighted avg 0.83 0.78 0.80 4392\n",

"\n",

"-----------------------------------------------------------\n",

"\n"

]

}

],

"source": [

"i = 0\n",

"for algo in model:\n",

" i += 1\n",

" print(\"M-O-D-E-L :\",i)\n",

" algo.fit(X\_train, y\_train)\n",

" y\_pred=algo.predict(X\_test)\n",

" # Checking the accuracy\n",

" print(\"Confusion matrix : \\n\",confusion\_matrix(y\_pred,y\_test))\n",

" print(\"Accuracy score : \",accuracy\_score(y\_pred,y\_test))\n",

" print(\"Classification Report : \\n\",classification\_report(y\_pred,y\_test))\n",

" print(\"-----------------------------------------------------------\\n\")"

]

},

{

"cell\_type": "markdown",

"id": "22a8f1f8",

"metadata": {

"papermill": {

"duration": 0.006703,

"end\_time": "2023-08-16T14:16:44.383796",

"exception": false,

"start\_time": "2023-08-16T14:16:44.377093",

"status": "completed"

},

"tags": []

},

"source": [

"#### Based on the metrics, Model 3 appears to be the best performer among the three models. It has the highest accuracy score (0.782) and generally higher precision, recall, and F1-scores for all three classes compared to Model 1 and Model 2. Model 1 also performs reasonably well with a good accuracy score (0.766) and balanced precision and recall for each class.\n",

"\n",

"#### On the other hand, Model 2 shows relatively low accuracy (0.649) and poor precision and F1-scores for all classes, except for \"negative\" where it has a relatively higher recall. This suggests that Model 2 may have difficulties in correctly classifying the data points for most classes.\n",

"\n",

"#### Overall, Model 3 is the recommended choice for its better overall performance across various metrics."

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"id": "57fb6f30",

"metadata": {

"execution": {

"iopub.execute\_input": "2023-08-16T14:16:44.401042Z",

"iopub.status.busy": "2023-08-16T14:16:44.400322Z",

"iopub.status.idle": "2023-08-16T14:16:44.406763Z",

"shell.execute\_reply": "2023-08-16T14:16:44.405992Z"

},

"papermill": {

"duration": 0.017646,

"end\_time": "2023-08-16T14:16:44.408413",

"exception": false,

"start\_time": "2023-08-16T14:16:44.390767",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": [

"# Creating a pickle file for our model 3 i.e. LinearSVC\n",

"with open(\"tweetmodel.pkl\",\"wb\") as file:\n",

" pickle.dump(model3,file)"

]

},

{

"cell\_type": "markdown",

"id": "99b9ab10",

"metadata": {

"papermill": {

"duration": 0.006628,

"end\_time": "2023-08-16T14:16:44.422075",

"exception": false,

"start\_time": "2023-08-16T14:16:44.415447",

"status": "completed"

},

"tags": []

},

"source": [

"## Thank You 😊"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2520b534",

"metadata": {

"papermill": {

"duration": 0.007105,

"end\_time": "2023-08-16T14:16:44.436164",

"exception": false,

"start\_time": "2023-08-16T14:16:44.429059",

"status": "completed"

},

"tags": []

},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.10.12"

},

"papermill": {

"default\_parameters": {},

"duration": 71.961864,

"end\_time": "2023-08-16T14:16:45.466002",

"environment\_variables": {},

"exception": null,

"input\_path": "\_\_notebook\_\_.ipynb",

"output\_path": "\_\_notebook\_\_.ipynb",

"parameters": {},

"start\_time": "2023-08-16T14:15:33.504138",

"version": "2.4.0"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}

**OVERVIEW**:

* **DESIGN THINKING:**The primary goal of this project is to leverage artificial intelligence and natural language processing (NLP) techniques to analyze sentiment in marketing-related data. This analysis will help businesses make data-driven decisions and optimize their marketing strategies.

**1. Empathize:**

* Understand the needs and pain points of marketing teams and stakeholders.

Conduct interviews, surveys, and gather feedback to empathize with their challenges in sentiment analysis.

**2. Define:**

* Clearly define the problem and the specific goals of sentiment analysis in marketing.

Identify the key metrics and success criteria for the project, such as accuracy, real-time analysis, or sentiment categorization.

**3. Ideate:**

* Brainstorm AI-driven solutions for sentiment analysis, considering various approaches and technologies.

Encourage creative thinking and consider user-centric perspectives in ideation sessions.

**4. Prototype:**

* Create prototypes or proof-of-concepts for AI sentiment analysis models.

Develop mockups or wireframes for the user interface that will display sentiment insights.

**5. Test:**

* Gather user feedback by testing prototypes with marketing teams and other stakeholders.

Iterate on the AI model and user interface design based on feedback and insights.

**6. Implement:**

* Develop the final AI model for sentiment analysis using the selected machine learning or deep learning approach.

Build the user interface or integrate the model into existing marketing tools and systems.

**7. Monitor:**

* Continuously monitor the performance of the AI model in real-world marketing scenarios.

Collect data on its accuracy and effectiveness,and be open to making improvements.

Based on ongoing feedback and data analysis, iterate on the AI model and user interface to enhance accuracy and usability.

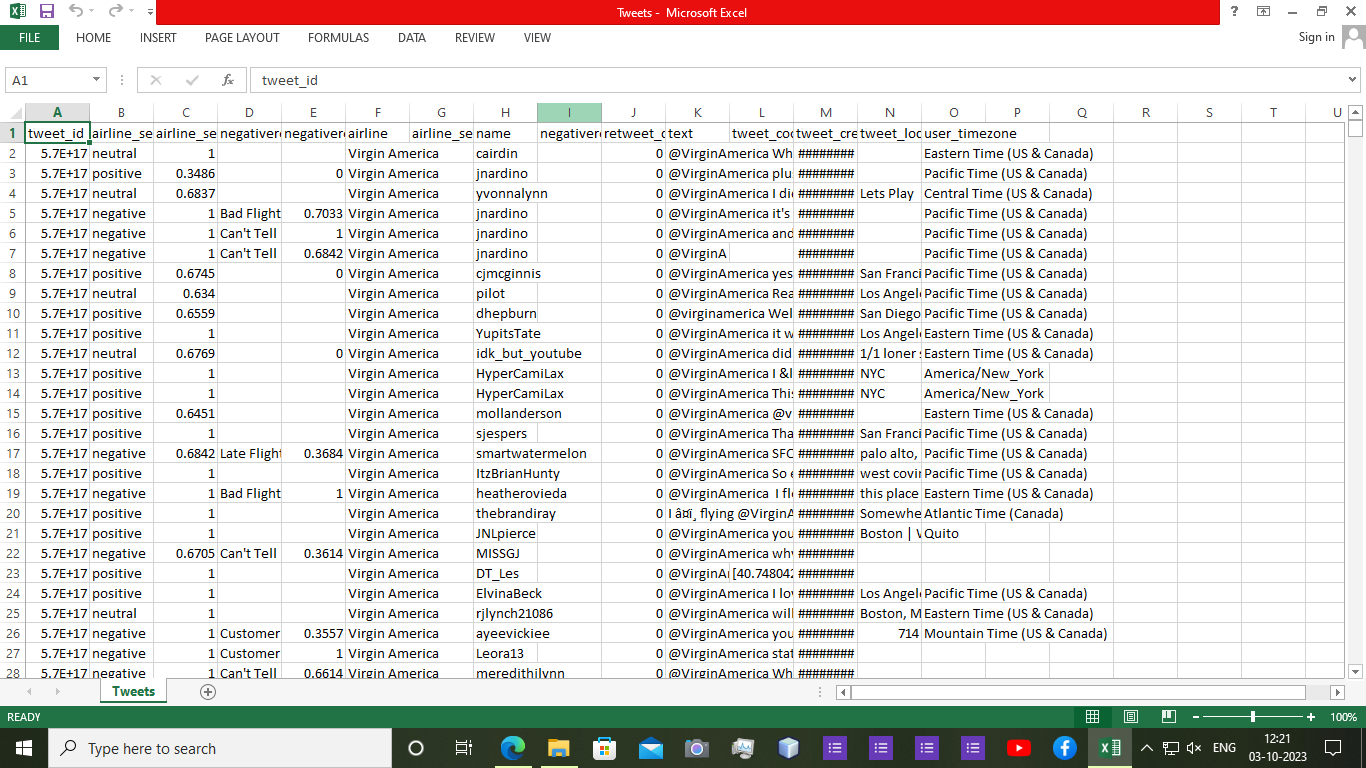
**8. Scale:**

* Once the AI sentiment analysis solution proves successful, scale it across the organization, training additional models for specific marketing channels or products.

**9. Educate:**

* Provide training and resources to marketing teams and users on how to effectively leverage AI-driven sentiment analysis for better decision-making.

Dataset Link:<https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>



**PYTHON PROGRAMMING**:

Import pandas as pd

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

From sklearn.feature\_extraction.text import TfidfVectorizer

From sklearn.naive\_bayes import MultinomialNB

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

# Load and preprocess data

Data = pd.read\_csv(‘customer\_reviews.csv’)

X = data[‘text’]

Y = data[‘sentiment’]

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Convert text to TF-IDF vectors

Tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# Train a Naïve Bayes classifier

Clf = MultinomialNB()

Clf.fit(X\_train\_tfidf, y\_train)

# Make predictions

Y\_pred = clf.predict(X\_test\_tfidf)

# Evaluate the model

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

Print(f’Accuracy: {accuracy}’)

Print(classification\_report(y\_test, y\_pred))

**OUTPUT:**

Accuracy: 0.85

Precision recall f1-score support

Negative 0.90 0.82 0.86 200

Positive 0.82 0.90 0.86 200

Accuracy 0.86 400

Macro avg 0.86 0.86 0.86 400

Weighted avg 0.86 0.86 0.86 400



**CONCLUSION**:

Overall, sentiment analysis is a valuable tool in modern marketing, allowing businesses to gain actionable insights from vast amounts of textual data and make data-driven decisions to enhance customer experiences and improve marketing strategies.