**Phase 5: DOCUMENTATION AND SUBMISSION**

**PROBLEM DEFINITION:**

The objective of this project is to perform sentimet analysis on customer feedback related to competitor products in order to extract valuable insights that can inform business decisions. By understanding customer sentiments towards competing products, our aim is to identify both strengths and weaknesses, enabling us to improve our own offerings and gain a competitive advantage in the market.

**DESIGN THINKING:**

**Data collection:** Gather a labeled dataset of Tweets. Where each tweet is labeled as either positive or negative or neural.

**Data Cleaning and Preprocessing:**To normalize the text, remove special characters, stopwords, and perform stemming or lemmatization on the text data.

**Feature engineering:** To represent communications quantitatively, significant characteristics from the text data, such as word frequency, n-grams, and text length, are extracted.

**Model selection:** Pick the best deep learning or machine learning models for classification, such as Naive Bayes, SVMs, or neural networks like CNN or LSTM.To assess the performance of the model, divide the dataset into training and testing sets.

**Model training:** Use the training set of data to train the selected model, then adjust the hyperparameters for the optimum performance. Employ techniques like cross-validation to prevent overfitting.

**Evaluation Metrics:** To quantify false positives and false negatives, measure the model's performance using evaluation metrics like precision, recall, F1-score, and accuracy.

**Threshold Optimization:** Depending on the requirements of the application, change the classification threshold to strike a balance between false positives and false negatives.

Implement tools that will allow users to report false positives and false negatives, which can be utilized to improve and tweak the model

Dataset used link : https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

**DESCRIPTION OF DATASET:**

The "Twitter US Airline Sentiment" dataset typically contains tweets from customers who have expressed their sentiments or opinions about various U.S. airlines on Twitter. The dataset includes the following key information:

**1. Data Source:** The dataset consists of tweets posted on Twitter that mention different U.S. airlines. These tweets were collected from the social media platform.

**2. Sentiment Labels:** Each tweet in the dataset is labeled with one of three sentiment categories: positive, negative, or neutral. These labels indicate the sentiment or emotional tone expressed in the tweet regarding the airline mentioned.

**3. Attributes:** The primary attributes in the dataset typically include the tweet text, the airline mentioned, and the sentiment label.

**4. Size:** The dataset usually contains thousands of tweets, providing a substantial amount of text data for sentiment analysis tasks.

**DATA PREPROCESSING STEPS:**

**1. Data Loading:** Download the dataset from Kaggle or the source you have and load it into your preferred data analysis environment, such as Python with libraries like Pandas.

**2. Data Inspection:** Examine the dataset to understand its structure and contents. Check the column names, data types, and the number of rows and columns. This will help you get a sense of what you're working with.

**3. Data Cleaning:**

- **Handle Missing Values:** Check for missing data and decide on an appropriate strategy (e.g., filling in missing values or removing rows/columns).

- **Remove Duplicates:** Eliminate duplicate records if present.

- **Correct Data Types:** Ensure that the data types of columns are appropriate.

**4. Text Cleaning:**

- **Lowercasing:** Convert text to lowercase to ensure consistency.

- **Removing Special Characters:** Remove non-alphanumeric characters, punctuation, and symbols that may not be relevant for sentiment analysis.

- **Tokenization:** Split text into individual words or tokens.

- **Stopword Removal:** Remove common stopwords (e.g., "and," "the," "is") that do not carry much sentiment information.

- **Lemmatization or Stemming:** Reduce words to their base or root form to standardize text (e.g., "running" to "run").

- **Handling URLs and User Mentions:** Replace or remove URLs and Twitter usernames, as they may not provide useful sentiment information.

**5. Exploratory Data Analysis (EDA):** Perform basic statistical analysis and visualization to understand the distribution of sentiment labels and other important characteristics of the data.

**6. Text Vectorization:** Convert the text data into numerical form using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) .

**7. Splitting the Data:** Divide the dataset into training, validation, and test sets to train and evaluate your model.

**8. Model-specific Preprocessing:** Some models might require specific preprocessing steps, such as padding sequences for recurrent neural networks (RNNs) or one-hot encoding for traditional machine learning models.

**9. Final Data Preparation:** Ensure that your data is in a format suitable for your machine learning or deep learning models.

**FEATURE EXTRACTION TECHNIQUES:**

**1. Bag of Words (BoW):**

- **Tokenization:** Splitting the text into individual words or tokens.

- **Vocabulary:** Create a vocabulary of unique words in the dataset.

- **Count Vectorization:** Represent each document as a vector of word counts in the vocabulary.

**2. Term Frequency-Inverse Document Frequency (TF-IDF):**

**- TF-IDF Vectorization:** Instead of using raw word counts, use TF-IDF scores, which reflect the importance of words in the document relative to the entire dataset.

**3. Word Embeddings:**

- **Word2Vec, GloVe, or FastText:** Pre-trained word embeddings that capture semantic relationships between words. You can use these embeddings to represent words in the dataset.

- **Average Word Embeddings:** For a document, you can average the word embeddings of its constituent words to create a document-level feature vector.

**4. N-grams:** Include not just individual words but also sequences of words (bigrams, trigrams) to capture some context and phrase-level information.

**5. Word Frequency Features:** Include features based on the frequency of specific words or phrases in the text.

**6. Sentiment Lexicons:** Use sentiment lexicons or dictionaries to calculate sentiment scores for each document, which can be used as features.

**7. Text Length and Structure:** Features like the length of the text, presence of hashtags, mentions, or URLs can be used.

**MACHINE LEARNING ALGORITHMS:**

**Naive Bayes:** Naive Bayes algorithms, such as Multinomial Naive Bayes, are commonly used for text classification tasks, including spam detection. They work well with text data and are relatively simple to implement.

**Support Vector Machines (SVM):** SVMs can be effective for text classification. They aim to find a hyperplane that best separates spam and ham messages.

**Deep Learning Models (e.g., Neural Networks):** You can also explore deep learning techniques using neural networks, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for text classification.

**MODEL TRAINING PROCESS:**

**Data Splitting:** Divide your dataset into training and testing sets to train and evaluate your model's performance.

**Feature Extraction:** Apply the feature extraction techniques mentioned earlier to represent text data as numerical features.

**Model Selection:** Choose one or more of the machine learning algorithms mentioned above.

**Model Training:** Train the selected model(s) on the training data. This involves feeding the algorithm with the labeled examples (spam/ham messages) to learn the patterns in the data.

**Model Evaluation:** After training, evaluate the model's performance on the testing dataset using appropriate evaluation metrics.

**EVALUATION METRICS:**

**Accuracy:** Measures the overall correctness of the model's predictions. However, accuracy may not be the best metric when classes are imbalanced.

**Precision:** Calculates the ratio of true positive predictions to all positive predictions. It's a measure of how many of the predicted spam messages are actually spam.

**Recall (Sensitivity):** Measures the ratio of true positive predictions to all actual positive instances. It's a measure of how many spam messages were correctly identified.

**F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics.

**Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC):** These are used for binary classification problems and provide a graphical representation of the model's performance and an associated metric for its quality.

**Confusion Matrix:** A table showing true positive, true negative, false positive, and false negative predictions, which can be used to compute various metrics.

**INNOVATIVE TECHNIQUES:**

**1. BERT (Bidirectional Encoder Representations from Transformers):** BERT is a pre-trained deep learning model that has been highly successful in various NLP tasks, including sentiment analysis. Fine-tuning BERT on your dataset can help capture complex contextual relationships within the text and improve sentiment analysis accuracy.

**2. Transformers for Multi-Modal Data:** If your dataset includes not only text but also other data like images or videos, you can use multi-modal models. Models like CLIP or DALL-E, which combine vision and language, can be used to analyze sentiment based on both textual and visual content.

**3. Transfer Learning:** You can leverage pre-trained sentiment analysis models and fine-tune them on your specific dataset. This can save a lot of training time and resources while achieving good results.

**4. Emotion Analysis:** Instead of just binary sentiment analysis (positive/negative), you can perform emotion analysis to detect more granular emotional states, such as happiness, anger, sadness, etc. This can provide a deeper understanding of user sentiment.

**5. Hybrid Models:** Combine traditional machine learning techniques with deep learning approaches to create hybrid models. For example, you can use traditional feature engineering techniques in combination with neural networks.

**6. Contextual Embeddings:** Utilize contextual word embeddings like ELMo or GPT (Generative Pre-trained Transformer) to capture word meaning in context, which can improve sentiment analysis performance.

**7. Time Series Analysis:** If your dataset has a temporal dimension (e.g., tweets collected over time), you can perform time series analysis to understand how sentiment changes over time and identify patterns.

**8. Ensemble Methods:** Combine the predictions of multiple models using ensemble techniques like stacking or bagging to improve overall performance.

**CODING:**

**LOAD THE DATASET:**

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

df.head()

**PREPROCESS THE DATASET:**

import pandas as pd

import nltk

from nltk.corpus import stopwords

import re

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

df = df.drop\_duplicates()

df['text'] = df['text'].str.lower()

df['text'] = df['text'].apply(lambda x: re.sub(r'https?://[^\s]+|www\.[^\s]+|\@[^\s]+', '', x))

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

import nltk

nltk.download('punkt')

df['text'] = df['text'].apply(lambda x: nltk.word\_tokenize(x))

import nltk

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

df['text'] = df['text'].apply(lambda x: [word for word in x if word not in stop\_words])

import nltk

nltk.download('punkt')

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

df.to\_csv('preprocessed\_dataset.csv', index=False)

df.head()

**MODEL TRAINING:**

import pandas as pd

import numpy as np

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, confusion\_matrix

from joblib import dump

df = pd.read\_csv('preprocessed\_dataset.csv')

df['text'].fillna('',inplace=True)

X = df['text']

y = df['airline\_sentiment']

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

tfidf\_vectorizer = TfidfVectorizer()

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

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

classifier = MultinomialNB()

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

y\_pred = classifier.predict(X\_test\_tfidf)

**EVALUATION THE DATASET:**

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

classification\_rep = classification\_report(y\_test, y\_pred, target\_names=df['airline\_sentiment'].unique())

confusion = confusion\_matrix(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:')

print(classification\_rep)

print('Confusion Matrix:')

print(confusion)

**TEST THE DATASET:**

dump(classifier, 'sentiment\_model.joblib')

sample\_texts = ["I love this airline!", "The service is terrible.", "Neutral comment."]

sample\_tfidf = tfidf\_vectorizer.transform(sample\_texts)

sample\_predictions = classifier.predict(sample\_tfidf)

for text, prediction in zip(sample\_texts, sample\_predictions):

print(f"Text: {text}")

print(f"Predicted Sentiment: {prediction}")

print("")

**VISUALIZATION THE DATASET:**

import matplotlib.pyplot as plt

import seaborn as sns

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

plt.figure(figsize=(8, 6))

sns.countplot(data=df, x='airline\_sentiment', order=sentiment\_counts.index)

plt.title('Sentiment Distribution')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

airline\_counts = df['airline'].value\_counts()

plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='airline', order=airline\_counts.index)

plt.title('Airline Distribution')

plt.xlabel('Airline')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**CONFUSION MATRIX VISUALIZATION:**

confusion = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues',

xticklabels=['Negative', 'Neutral', 'Positive'],

yticklabels=['Negative', 'Neutral', 'Positive'])

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()