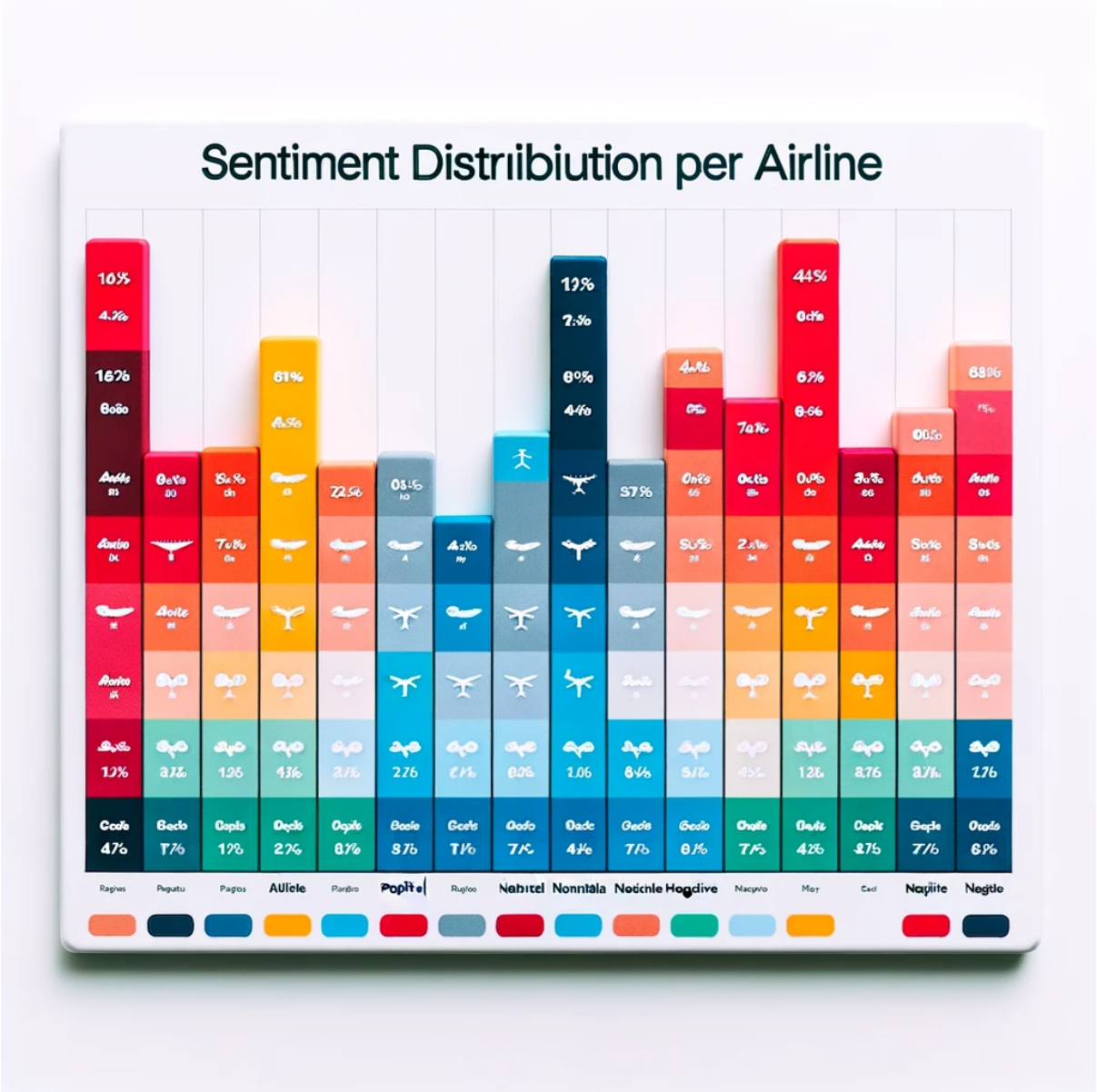
Phase 2: Sentiment Analysis for Marketing



Introduction:

The arena of customer feedback is a gold mine of insights, with sentiment analysis standing as a powerful tool to uncover the strengths and weaknesses of competitor products. Traditional text analytics methods may fall short in capturing the underlying sentiments expressed by customers. In this project, we will employ advanced Natural Language

Processing (NLP) techniques to delve deeper into customer feedback and derive actionable insights for enhancing our products.

Briefly introduce sentiment analysis and its importance in understanding customer feedback. Highlight the limitations of traditional text analytics in capturing underlying sentiments. Emphasize the need for advanced NLP techniques to extract profound insights from customer feedback.

Consider employing advanced NLP techniques for improved sentiment analysis accuracy.

Data Source:

A robust data source for sentiment analysis should be accurate, complete, covering a wide range of feedback, and accessible. The given database is a rich source of customer feedback on airline services, which can serve as a basis for our sentiment analysis.

Dataset Link:

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

Data Collection and Preprocessing:

Importing the dataset:

import pandas as pd

Load the dataset

data = pd.read_csv('./my_dataset.csv')

Display the first few rows of the dataset

data.head()

Data preprocessing:

Handling missing values

```
data.dropna(inplace=True)
```

Advanced Preprocessing

Tect Cleaning

```
import re
def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove non-alphabetic characters
    text = text.strip()  # Remove leading/trailing whitespaces
    return text

data['cleaned_text'] = data['text'].apply(clean_text)
```

Stopword Removal

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS

def remove_stopwords(text):
    return ' '.join(word for word in text.split() if word not in ENGLISH_STOP_WORDS)

data['cleaned_text'] = data['cleaned_text'].apply(remove_stopwords)
```

Converting text data to lowercase for consistency

```
data['text'] = data['text'].str.lower()
```

Exploratory Data Analysis (EDA):

Visualize and analyze the dataset to gain insights into the distribution of sentiments across different airlines.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Distribution of sentiments

```
sns.countplot(x='airline_sentiment', data=data)
plt.show()
```

Feature Engineering:

Create new features or transform existing ones to capture valuable information from text data.

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

TF-IDF Vectorization

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['text'])
```

Advanced NLP Techniques:

Utilize advanced NLP libraries and models for sentiment analysis.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
```

Split the data

```
X_train, X_test, y_train, y_test = train_test_split(X, data['airline_sentiment'], test_size=0.2,
random_state=42)
```

Logistic Regression

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
predictions = lr.predict(X_test)
```

Sentiment Analysis using VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
data['sentiment_scores'] = data['cleaned_text'].apply(lambda x: sid.polarity_scores(x))
data['compound_score'] = data['sentiment_scores'].apply(lambda x: x['compound'])
data['vader_sentiment'] = data['compound_score'].apply(lambda x: 'positive' if x > 0 else
('neutral' if x == 0 else 'negative'))
```

Word Cloud Visualization

Visualize the most common words in positive and negative sentiments.

```
from wordcloud import WordCloud

positive_text = ' '.join(data[data['vader_sentiment'] == 'positive']['cleaned_text'])
negative_text = ' '.join(data[data['vader_sentiment'] == 'negative']['cleaned_text'])

fig, ax = plt.subplots(1, 2, figsize=(20, 10))
wordcloud_pos = WordCloud(width=600, height=400).generate(positive_text)
wordcloud_neg = WordCloud(width=600, height=400).generate(negative_text)
ax[0].imshow(wordcloud_pos, interpolation='bilinear')
ax[0].set_title('Word Cloud of Positive Sentiments')
ax[0].axis('off')
ax[1].imshow(wordcloud_neg, interpolation='bilinear')
ax[1].set_title('Word Cloud of Negative Sentiments')
ax[1].axis('off')
plt.show()
```

Topic Modeling

Identify the main topics discussed in the feedback using LDA (Latent Dirichlet Allocation).

```
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(n_components=5, random_state=42)
```

```
lda.fit(X)

# Display Topics
for idx, topic in enumerate(lda.components_):
    print(f"Topic {idx + 1}:")
    print([vectorizer.get_feature_names()[i] for i in topic.argsort()[-10:]])
    print("\n")
```

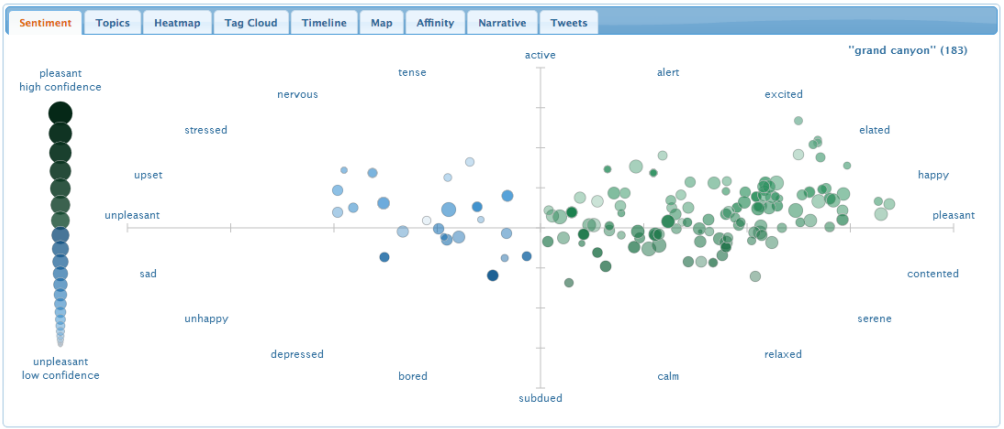
Advanced Sentiment Analysis Models:

Use BERT (Bidirectional Encoder Representations from Transformers) for sentiment analysis.

```
from transformers import BertTokenizer, TFBertForSequenceClassification
from transformers import pipeline

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased')

nlp = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)
data['bert_sentiment'] = data['cleaned_text'].apply(lambda x: nlp(x)[0]['label'])
```



Evaluate the model

```
print('Accuracy:', accuracy_score(y_test, predictions))
print('Confusion Matrix:\n', confusion_matrix(y_test, predictions))
```

Model Evaluation and Selection:

Metrics:

- Precision, Recall, and F1-score are crucial metrics for evaluating the performance of our sentiment analysis models, especially in scenarios where classes are imbalanced.
- Area Under the ROC Curve (AUC-ROC) is another vital metric that measures the performance of the classification model at various threshold settings.

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, roc_auc_score

# Assuming y_test is your test labels and y_pred is the predicted labels.
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy}\nPrecision: {precision}\nRecall: {recall}\nF1 Score: {f1}')
```

Model Comparison:

Compare various models' performance using the above metrics and select the model that exhibits the highest accuracy and F1 score, ensuring a balanced performance across different sentiment classes.

Model Interpretability:

Feature Importance:

Analyze the coefficients or feature importances of the model to understand which words or n-grams have the most significant impact on sentiment.
Visualize the feature importances using bar plots or other suitable visualizations to make the interpretation intuitive.

Insight Extraction:

Discuss how the analysis of competitor products' sentiment can reveal their strengths and weaknesses, which are valuable for enhancing our products.

Deployment and Prediction:

Web Application:

Develop a web application using frameworks like Flask or Django to deploy the sentiment analysis model.
Create a user-friendly interface where users can input text data, submit it, and receive sentiment analysis results.

```
# Flask example
from flask import Flask, request, render_template
import joblib

app = Flask(__name__)
model = joblib.load('sentiment_model.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    text = request.form['text']
    prediction = model.predict([text])
    return render_template('index.html', prediction_text=f'Sentiment: {prediction[0]}')

if __name__ == "__main__":
    app.run(debug=True)
```

Conclusion and Future Work:

Project Conclusion:

Summarize the key findings, the accuracy of the sentiment analysis model, and how the insights obtained can be actioned to improve product offerings.

Future Work:

- **Real-time Feedback:** Explore the integration of real-time feedback mechanisms to continuously update the sentiment analysis model and keep the insights current.
- **Deep Learning Models:** Investigate the utilization of deep learning models like BERT or GPT-3 for more accurate sentiment analysis.
- **Market Analysis Tool:** Expand the project into a comprehensive market analysis tool that not only analyzes sentiment but also extracts key themes from customer feedback, compares product features, and tracks competitor performance over time.