Data Preparation/Feature Engineering

1. Overview

The data preparation and feature engineering phase are crucial in a machine learning project as it lays the foundation for model training. This phase involves collecting, cleaning, and transforming raw data into a format suitable for machine learning algorithms. Feature engineering enhances the model's ability to extract patterns and relationships from the data, ultimately improving predictive performance.

2. Data Collection

The dataset used in the project originates from the TripAdvisor Hotel Reviews dataset obtained from Kaggle. It consists of 20,491 customer reviews in English, organized into two primary columns: "Review" and "Rating." During data collection, no additional steps were taken as the Kaggle dataset provided a structured format suitable for analysis.

3. Data Cleaning

Data preprocessing is the process of cleaning primary data and converting it into a format suitable for analysis. It is an important step in the data mining process because it can help improve the accuracy and efficiency of mining algorithms. User comments are often written casually without following grammar or spelling rules. Before using the data to evaluate the proposed method, we need to clean it up to remove errors. This process is called preprocessing and includes the following steps:

1. (Lowercasing) converts all characters to lowercase. This keeps the text consistent and eliminates the differences that capital letters can cause. For example, "Hello" and "hello" are considered the same word after case folding.
2. (Word splitting) separates a sentence into individual words or tokens. This makes it easy to analyze the text and identify important words. For example, the sentence "I love eating pizza" would be tokenized into "I", "love", "eating", "pizza".
3. (removing the pause words) is the removal of common words that do not add meaning to the text. This can help improve the accuracy of the analysis. For example, words like "and", "or", and "is" are common pause words.
4. (Slang word conversion) replacing slang words with their standard equivalents. This ensures that the text can be understood by everyone. For example, the slang word "LOL" can be converted to "laugh out loud".
5. (Rooting) reduces words to their root form. This can help identify words with similar meanings. For example, the words "walked", "walking", and "walker" can all be reduced to the root word "walk".

In Table 1 below, there is a sample data from TripAdvisor hotel reviews that has not undergone preprocessing yet.

Table 1: total data 20491 rows × 2 columns

	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member ...	2
2	nice rooms not 4* experience hotel monaco seat..	3
3	unique, great stay, wonderful time hotel monac...	5
...		
20488	ok just looks nice modern outside, desk staff ...	2
20489	hotel theft ruined vacation hotel opened sept ...	1
20490	people talking, can't believe excellent ratin...	2

The data is later processed through a data cleaning procedure. Among these steps are converting texts to lowercase, removing numbers, eliminating double spaces, tokenization, removing stop words, and performing lemmatization. This way, the data becomes ready for processing with machine learning classification algorithms.

Table 2: Tripadvisor Data After Data Preprocessing

	clean_word	Rating
0	nice hotel expens park got good deal stay hote...	4
1	ok noth special charg diamond member hilton de.	2
2	nice room experi hotel monaco seattl good hote...	3
3	uniqu great stay wonder time hotel monaco loca..	5
4	great stay great stay went seahawk game awesom	5
....	ok just looks nice modern outside, desk staff ...	2

After the words are transformed into a numerical format understandable by machine learning algorithms, the next step involves binary classification using machine learning algorithms. Preprocessing: If the rating score is less than 3, the value is set to 0; if the rating score is 3 or higher, the value is set to 1.

Table 3: New Labeling Results of the Binary Model.

	Review	Rating	sentiment
0	nice hotel expens park got good deal stay hote...	4	1
1	ok noth special charg diamond member hilton...	2	0
2	nice room experi hotel monaco seattl good hote..	3	1
3	uniqu great stay wonder time hotel monaco loca.	5	1
.....	......	...	...
.....	great stay great stay went seahawk game ...	5	1

4. Exploratory Data Analysis (EDA)

TripAdvisor hotel reviews dataset generally indicates a dominance of 5 and 4 ratings, reflecting predominantly positive sentiments. When examining the data, it can be observed that the combined ratings of 5 and 4 constitute more than 70% of the total evaluations.

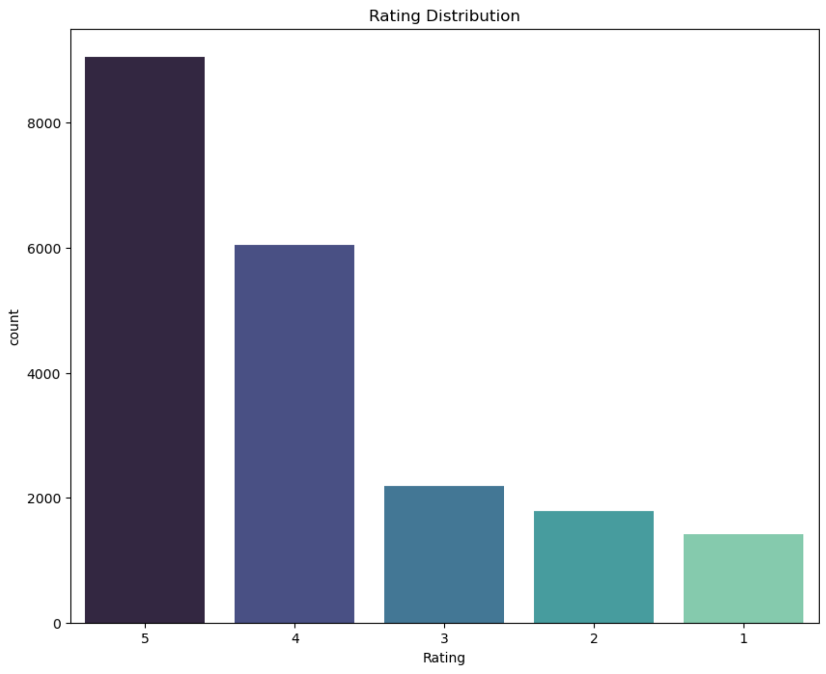

Figure 1: Star rating reviews.

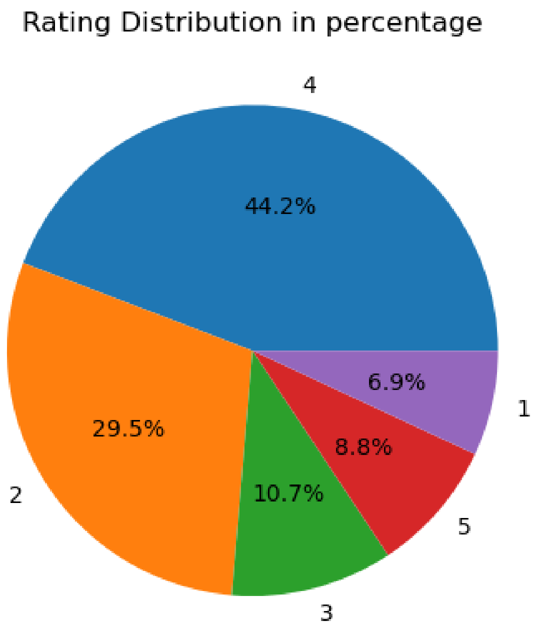

Figure 2: Star rating reviews.

A positive and negative word cloud is a visualization derived from data that is grouped based on the frequency of the most commonly used words after data preprocessing.

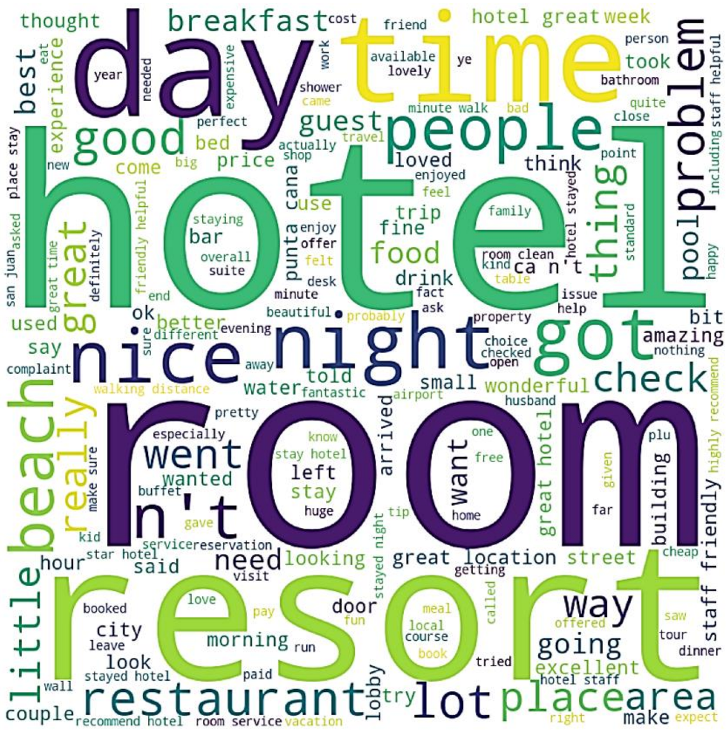

Figure 3: Word cloud Word Occurrence Frequency

20 most common words

[('hotel', 52900), ('room', 46357), ('great', 21095), ('good', 17060), ('staff', 16285), ('stay', 15339), ('night', 14073), ('day', 12952), ('time', 12601), ('nice', 12409), ('location', 11234), ('service', 10533), ('stayed', 10469), ('restaurant', 10090), ('beach', 10043), ('breakfast', 9654), ('place', 9381), ('clean', 9372), ('food', 9255), ('resort', 8918)]

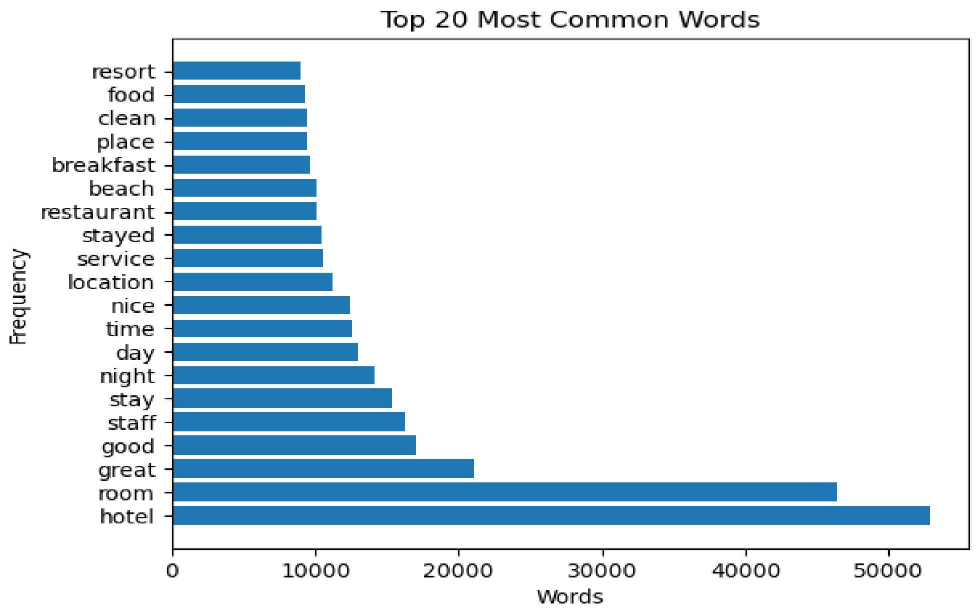

Figure 4: Top 20 most common words

Hotel reviews are classified as positive if the hotel has a rating of 3 stars or higher, and negative if it has a rating of 2 stars or lower.

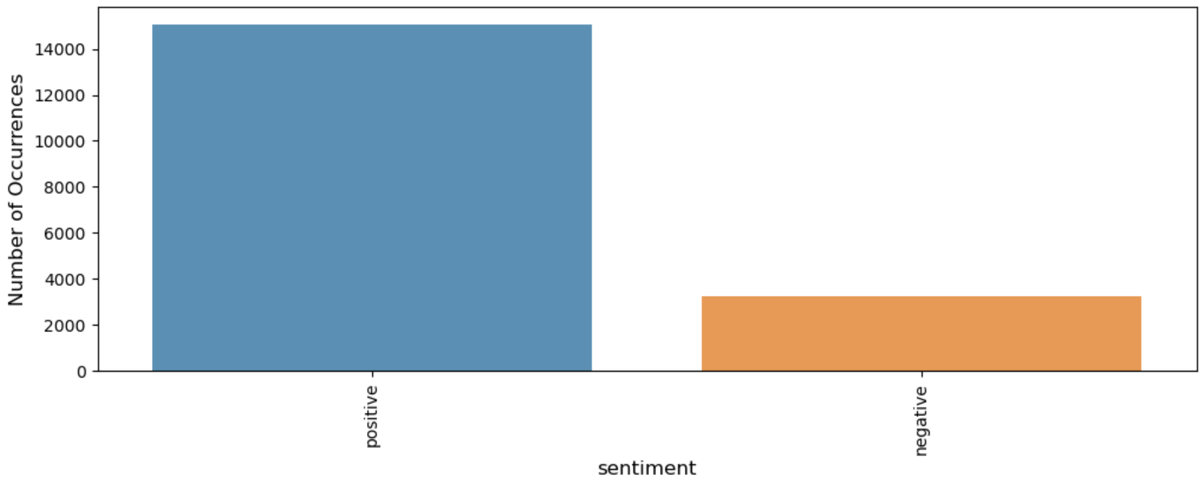

Figure 5: Positive and negative data set

5. Feature Engineering

- Created a new feature, 'clean_text,' through a multi-step process, including text cleaning, case conversion, stop-word removal, stemming, and lemmatization.
- Introduced the 'Sentiment' feature, categorizing reviews into positive (5, 4) and negative (1, 2, 3) sentiments based on ratings.

6. Data Transformation

- Implemented text preprocessing steps, including converting text to lowercase, removing symbols, hyperlinks, and stopwords, and applying stemming and lemmatization.
- Utilized TF-IDF Vectorizer for feature representation, transforming the textual data into numerical vectors.
- Split the data into training and test sets, ensuring that the machine learning algorithms were trained and evaluated on separate datasets.

Model Exploration

1. Model Selection

- **Rationale:**
    - Decision Tree Classifier was chosen for sentiment analysis on hotel reviews. Decision trees are suitable for classification tasks, and their interpretability makes them useful for understanding feature importance.
    - Logistic Regression was selected for its simplicity and effectiveness in binary classification problems, making it a good baseline model.
    - Naive Bayes Classifier was included due to its efficiency, simplicity, and effectiveness in text classification tasks.
- **Strengths and Weaknesses:**
    - **Decision Tree:**
        - *Strengths:* Handles non-linear relationships, easy to interpret, requires minimal data preprocessing.
        - *Weaknesses:* Prone to overfitting, sensitivity to small variations in the data.
    - **Logistic Regression:**
        - *Strengths:* Simple and interpretable, works well for linearly separable data, less prone to overfitting.
        - *Weaknesses:* Assumes a linear relationship between features and the log-odds of the response.
    - **Naive Bayes:**
        - *Strengths:* Efficient, works well with high-dimensional data, handles missing data and imbalanced datasets.
        - *Weaknesses:* Assumes independence between features, which may not hold in all cases.

2. Model Training

The data is split into training and test sets to allow machine learning algorithms to learn from the training data 80% of the dataset is used for training, while the remaining 20% is reserved for testing.

- **Decision Tree:**
    - Trained using the default hyperparameters.
- **Logistic Regression:**
    - Trained using the default hyperparameters.
- **Naive Bayes:**
    - No hyperparameter tuning was performed.

3. Model Evaluation

- Accuracy, precision, recall, F1-score were used to assess model performance.
- Confusion matrices and visualizations were employed to understand true positive, true negative, false positive, and false negative predictions.

Evaluation Metrics:

After training the model, predictions are made on the training data using Decision Tree, Logistic Regression, and Naive Bayes algorithms. An ensemble learning model combining these three classifiers is also proposed.

1. Decision Tree Algorithm.

The accuracy score obtained during the training phase was determined to be 1.0, indicating a perfect fit to the training data. However, in the testing phase, the accuracy score was observed to be 0.775, indicating a slightly lower success rate on the test data for the model.

Table 4: Precision, recall and F1 score calculation results.

	Precision	*Recall*	*F1-score*	*Support*
positive	0.62	0.58	0.60	406
negative	0.85	0.85	0.84	994
Accuracy			0.78	1400
Macro ang	0.73	0.72	0.72	1400
Weighted avg	0.77	0.78	0.77	1400

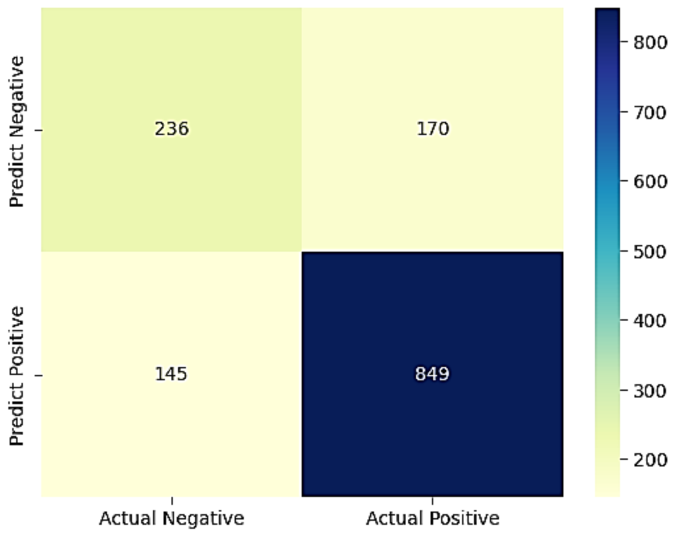

Figure 6: Accuracy of the decision tree algorithm model

2. Logistic Regression Algorithm.

The accuracy score obtained during the training process was determined to be 0.9268, indicating good performance in fitting the training data. In the testing process, the accuracy score obtained was 0.8871, demonstrating successful performance on the test data for the model as well.

Table 5: Precision, recall and F1 score calculation results.

	Precision	*Recall*	*F1-score*	*Support*
positive	0.91	0.68	0.78	406
negative	0.88	0.97	0.92	994
Accuracy			0.89	1400
Macro ang	0.90	0.83	0.85	1400
Weighted avg	0.89	0.89	0.88	1400

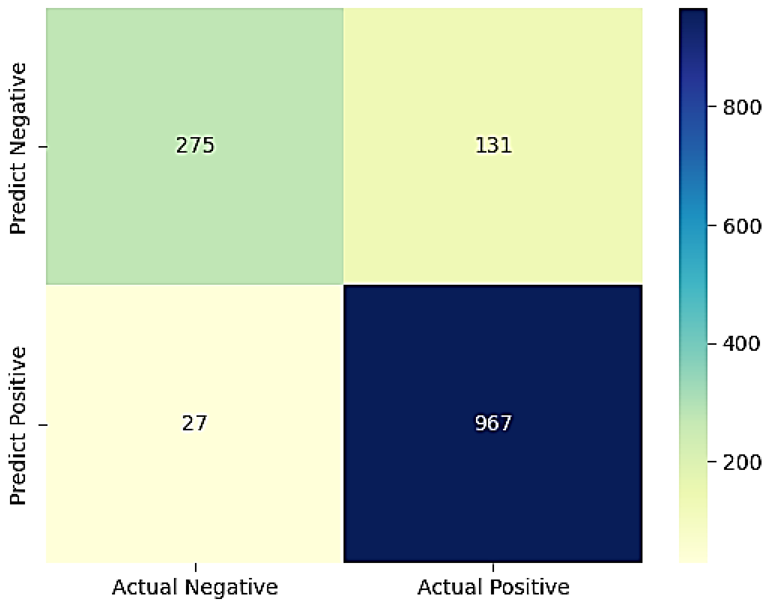

Figure 7: Accuracy of logistic regression algorithm model

3. Naive Bayes Algorithm.

The training accuracy score was calculated as 86.45%, indicating a good success in fitting the training data. However, the test accuracy score was determined to be 62.79%, showing that the model exhibited lower performance on the test data.

Table 6: Precision, recall and F1 score calculation results.

	Precision	*Recall*	*F1-score*	*Support*
positive	0.38	0.46	0.42	406
negative	0.76	0.70	0.73	994
Accuracy			0.89	1400
Macro ang	0.57	0.58	0.57	1400
Weighted avg	0.65	0.63	0.64	1400

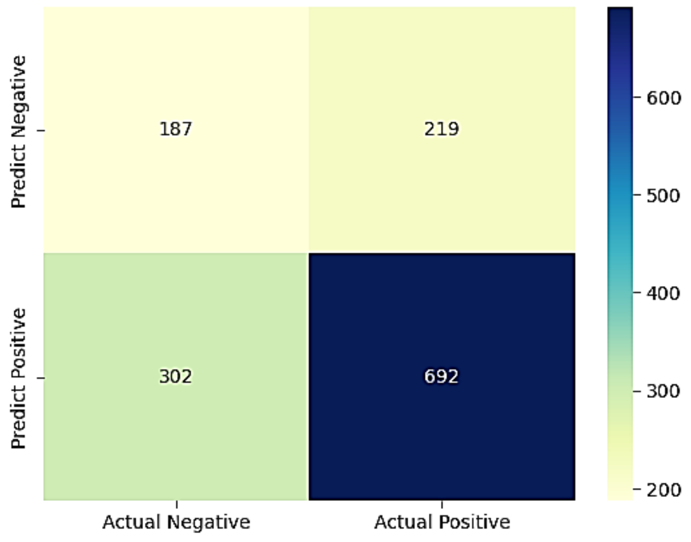

Figure 8: Accuracy of Naive Bayes algorithm model

4. VotingClassifier Model.

Using the VotingClassifier method, a new model has been created. This model combines Decision Tree, Logistic Regression, and Naive Bayes classifiers. The model has been trained on the training set and made predictions for the test set. Accuracy scores have been calculated for both training and testing. This approach aims to improve prediction performance by combining multiple classifiers and evaluates the model's success based on accuracy.

In the new model, the training accuracy is calculated as 99.89%, indicating an excellent fit to the training data. The test accuracy score is determined as 84.93%, demonstrating that the model performs well on the test data.

4. Code Implementation

```
# Import necessary libraries
import numpy as np
import pandas as pd
import re
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
from gensim.corpora.dictionary import Dictionary
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier
from wordcloud import WordCloud
import string
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from nltk.stem import WordNetLemmatizer

# Download necessary NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')

# Load the dataset
df = pd.read_csv("tripadvisor_hotel_reviews.csv")
df.head(5)

# Display information about the dataset
df.info()

# Visualize the distribution of ratings
df.Rating.value_counts().plot.pie(y="Review", autopct="%.2f%%", figsize=(6,
6))

# Data Cleaning
def clean_text(text):
    """
    Function to clean the text data by converting to lowercase and removing
symbols.
    """
    text = text.str.lower()
    text = text.apply(lambda T: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", T))
    return text

df['Review'] = clean_text(df['Review'])
df['Review'] = df['Review'].str.replace('#', '')

# Data Preprocessing
punc = string.punctuation
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
```

```
lemmatizer = WordNetLemmatizer()

def data_preprocessing(txt):
    """
    Function for data preprocessing including lowercase conversion, 
punctuation removal,
    stopword removal, stemming, and lemmatization.
    """
    txt = txt.lower()
    txt = "".join([x for x in txt if x not in punc])
    txt = " ".join([word for word in str(txt).split() if word not in 
stop_words])
    txt = " ".join([stemmer.stem(word) for word in txt.split()])
    txt = " ".join([lemmatizer.lemmatize(word) for word in txt.split()])
    return txt

df['clean_text'] = df['Review'].apply(data_preprocessing)
df.head()

# Sentiment Analysis
pos = [5, 4]
neg = [1, 2, 3]

def sentiment(rating):
    """
    Function to assign sentiment based on ratings.
    """
    if rating in pos:
        return 1
    elif rating in neg:
        return 0

df['Sentiment'] = df['Rating'].apply(sentiment)
df.head()

# Visualize Sentiment Distribution
fig = go.Figure([go.Bar(x=df.Sentiment.value_counts().index, 
y=df.Sentiment.value_counts().tolist())])
fig.update_layout(
    title="Values in each Sentiment",
    xaxis_title="Sentiment",
    yaxis_title="Values")
fig.show()

# Feature Engineering - TF-IDF Vectorization
docs = list(df['Review'])[:7000]
tfidf_vectorizer = TfidfVectorizer(use_idf=True, max_features=20000)
tfidf_vectorizer_vectors = tfidf_vectorizer.fit_transform(docs)

X = tfidf_vectorizer_vectors.toarray()
Y = df['Sentiment'][:7000]

# Model Training and Evaluation - Decision Tree
SEED = 123
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, 
random_state=SEED, stratify=Y)

dt = DecisionTreeClassifier(random_state=SEED)
dt.fit(X_train, y_train)
y_pred_test = dt.predict(X_test)
```

```
print("Training Accuracy score: " + str(round(accuracy_score(y_train,
dt.predict(X_train)), 4)))
print("Testing Accuracy score: " + str(round(accuracy_score(y_test,
dt.predict(X_test)), 4)))


# Classification Report and Confusion Matrix Visualization
print(classification_report(y_test, y_pred_test, target_names=['positive',
'negative']))
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual
Positive'],
                         index=['Predict Negative', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()

# Model Training and Evaluation - Logistic Regression
lr = LogisticRegression(random_state=SEED).fit(X_train, y_train)
y_pred_test = lr.predict(X_test)
print("Training Accuracy score: " + str(round(accuracy_score(y_train,
lr.predict(X_train)), 4)))
print("Testing Accuracy score: " + str(round(accuracy_score(y_test,
lr.predict(X_test)), 4)))

# Classification Report and Confusion Matrix Visualization
print(classification_report(y_test, y_pred_test, target_names=['positive',
'negative']))
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual
Positive'],
                         index=['Predict Negative', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()

# Model Training and Evaluation - Naive Bayes
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_test = gnb.predict(X_test)
print("Training Accuracy score: " + str(round(accuracy_score(y_train,
gnb.predict(X_train)), 4)))
print("Testing Accuracy score: " + str(round(accuracy_score(y_test,
gnb.predict(X_test)), 4)))

# Classification Report and Confusion Matrix Visualization
print(classification_report(y_test, y_pred_test, target_names=['positive',
'negative']))
cm = confusion_matrix(y_test, y_pred_test)
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Negative', 'Actual
Positive'],
                         index=['Predict Negative', 'Predict Positive'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
plt.show()

# Model Training and Evaluation - Voting Classifier
classifiers = [('Decision Tree', dt),
               ('Logistic Regression', lr),
               ('Naive Bayes', gnb)
              ]
vc = VotingClassifier(estimators=classifiers)
vc.fit(X_train, y_train)
```

```
print("Training Accuracy score: " + str(round(accuracy_score(y_train,
vc.predict(X_train)), 4)))
print("Testing Accuracy score: " + str(round(accuracy_score(y_test,
vc.predict(X_test)), 4)))

# Word Cloud Visualization
all_words = ' '.join(df['Review']).lower().split()
word_counts = Counter(all_words)
top_words = word_counts.most_common(20)

wordcloud = WordCloud(width=800, height=400,
background_color='white').generate_from_frequencies
```