SENTIMENT ANALYSIS FOR MARKETING

TEAM MEMBER

211121104054 : THULASI NAVANEETHAN N

Phase-4 submission document

Project Title: Sentiment Analysis For Marketing

Phase 4: Development Part 2

Topic:

Continue building the house price prediction model by feature engineering, model training, and evaluation

Introduction To Sentiment Analysis

Sentiment analysis refers to analyzing an opinion or feelings about something using data like text or images, regarding almost anything. Sentiment analysis helps companies in their decision-making process. For instance, if public sentiment towards a product is not so good, a company may try to modify the product or stop the production altogether in order to avoid any losses.

There are many sources of public sentiment e.g. public interviews, opinion polls, surveys, etc. However, with more and more people joining social media platforms, websites like Facebook and Twitter can be parsed for public sentiment.

Problem Definition

Given tweets about six US airlines, the task is to predict whether a tweet contains positive, negative, or neutral sentiment about the airline. This is a typical supervised learning task where given a text string, we have to categorize the text string into predefined categories.

Solution

To solve this problem, we will follow the typical machine learning pipeline. We will first import the required libraries and the dataset. We will then do exploratory data analysis to see if we can find any trends in the dataset. Next, we will perform text preprocessing to convert textual data to numeric data that can be used by a machine learning algorithm. Finally, we will use machine learning algorithms to train and test our sentiment analysis models.

Overview of the process:

The following is an overview of the process of building a house price prediction model by feature selection, model training, and evaluation:

1. **Prepare the data**: This includes cleaning the data, removing outliers, and handling missing values.

2. **Perform feature selection**: This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3**. Train the model**: There are many different machine learning algorithms that can be used for house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.
4. **Evaluate the model**: This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.
5. **Deploy the model**: Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the house prices of new houses.

PROCEDURE :

Feature selection:

1. Identify the target variable. This is the variable that you want to predict,.
2. Explore the data. This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.
3. Remove redundant features. If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.
4. Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

PROGRAM :

```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
%matplotlib inline

airline_tweets = pd.read_csv(r'/content/Tweets.csv')
airline_tweets.head()
```

```
             tweet_id airline_sentiment  airline_sentiment_confidence  \
0  570306133677760513           neutral                        1.0000
1  570301130888122368          positive                        0.3486
2  570301083672813571           neutral                        0.6837
3  570301031407624196          negative                        1.0000
4  570300817074462722          negative                        1.0000

  negativereason  negativereason_confidence         airline  \
0            NaN                        NaN  Virgin America
1            NaN                     0.0000  Virgin America
2            NaN                        NaN  Virgin America
3     Bad Flight                     0.7033  Virgin America
4     Can't Tell                     1.0000  Virgin America

  airline_sentiment_gold        name negativereason_gold  retweet_count  \
0                    NaN     cairdin                 NaN              0
1                    NaN    jnardino                 NaN              0
2                    NaN  yvonnalynn                 NaN              0
3                    NaN    jnardino                 NaN              0
4                    NaN    jnardino                 NaN              0

                                                 text tweet_coord  \
0               @VirginAmerica What @dhepburn said.         NaN
1  @VirginAmerica plus you've added commercials t...         NaN
2  @VirginAmerica I didn't today... Must mean I n...         NaN
3  @VirginAmerica it's really aggressive to blast...         NaN
4  @VirginAmerica and it's a really big bad thing...         NaN

               tweet_created tweet_location               user_timezone
0  2015-02-24 11:35:52 -0800            NaN  Eastern Time (US & Canada)
1  2015-02-24 11:15:59 -0800            NaN  Pacific Time (US & Canada)
2  2015-02-24 11:15:48 -0800      Lets Play  Central Time (US & Canada)
3  2015-02-24 11:15:36 -0800            NaN  Pacific Time (US & Canada)
4  2015-02-24 11:14:45 -0800            NaN  Pacific Time (US & Canada)
```

Let's explore the dataset a bit to see if we can find any trends. But before that, we will change the default plot size to have a better view of the plots.

```
plot_size = plt.rcParams["figure.figsize"]
print(plot_size[0])
print(plot_size[1])

plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size
```

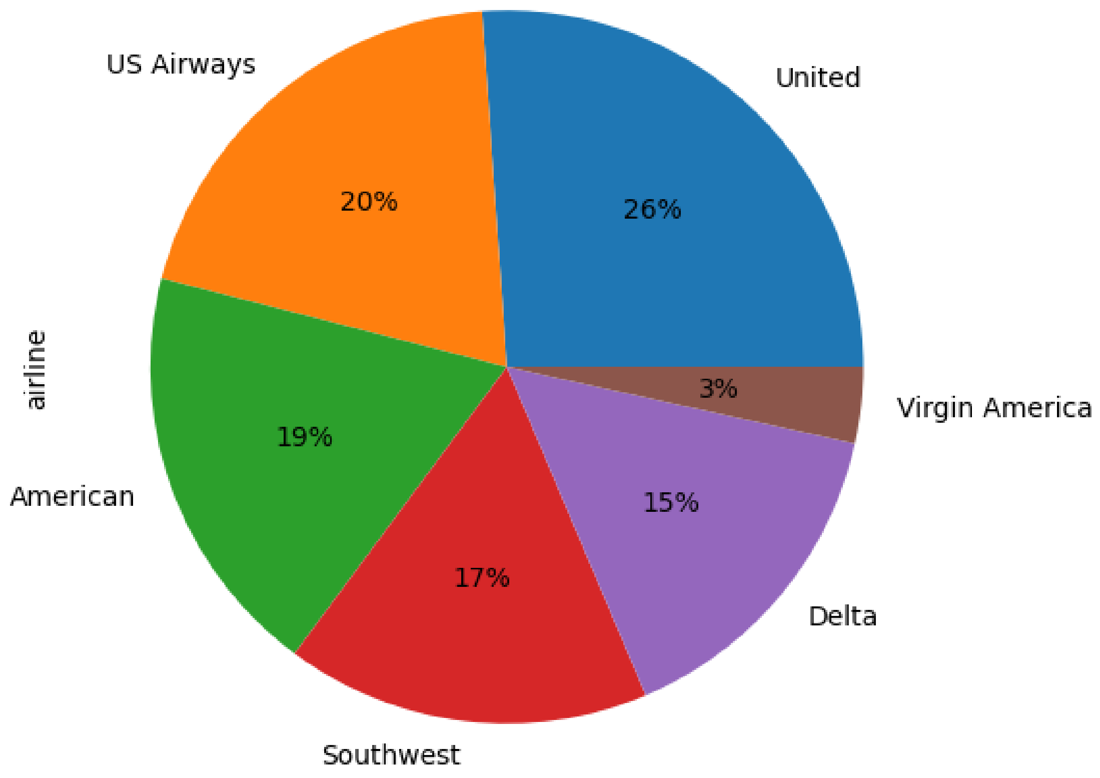
```
6.4
4.8
```

Exploration Of Data

Let's first see the number of tweets for each airline. We will plot a pie chart for that:

```
airline_tweets.airline.value_counts().plot(kind='pie', autopct='%1.0f%%')
```

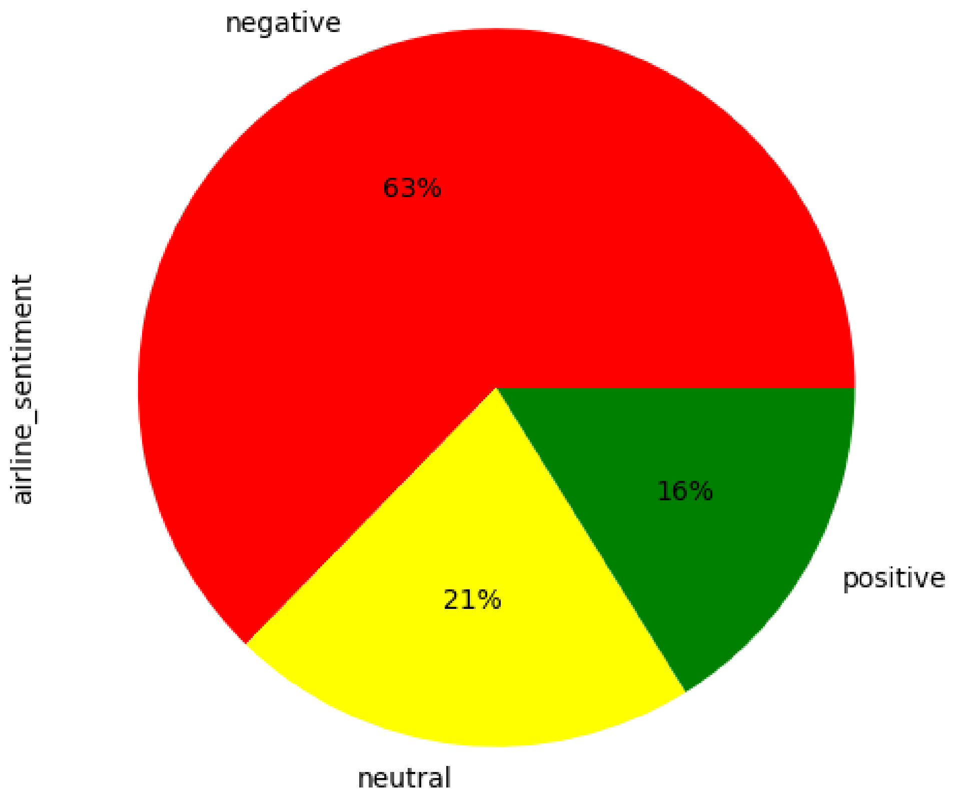
```
<Axes: ylabel='airline'>
```



Let's now see the distribution of sentiments across all the tweets.

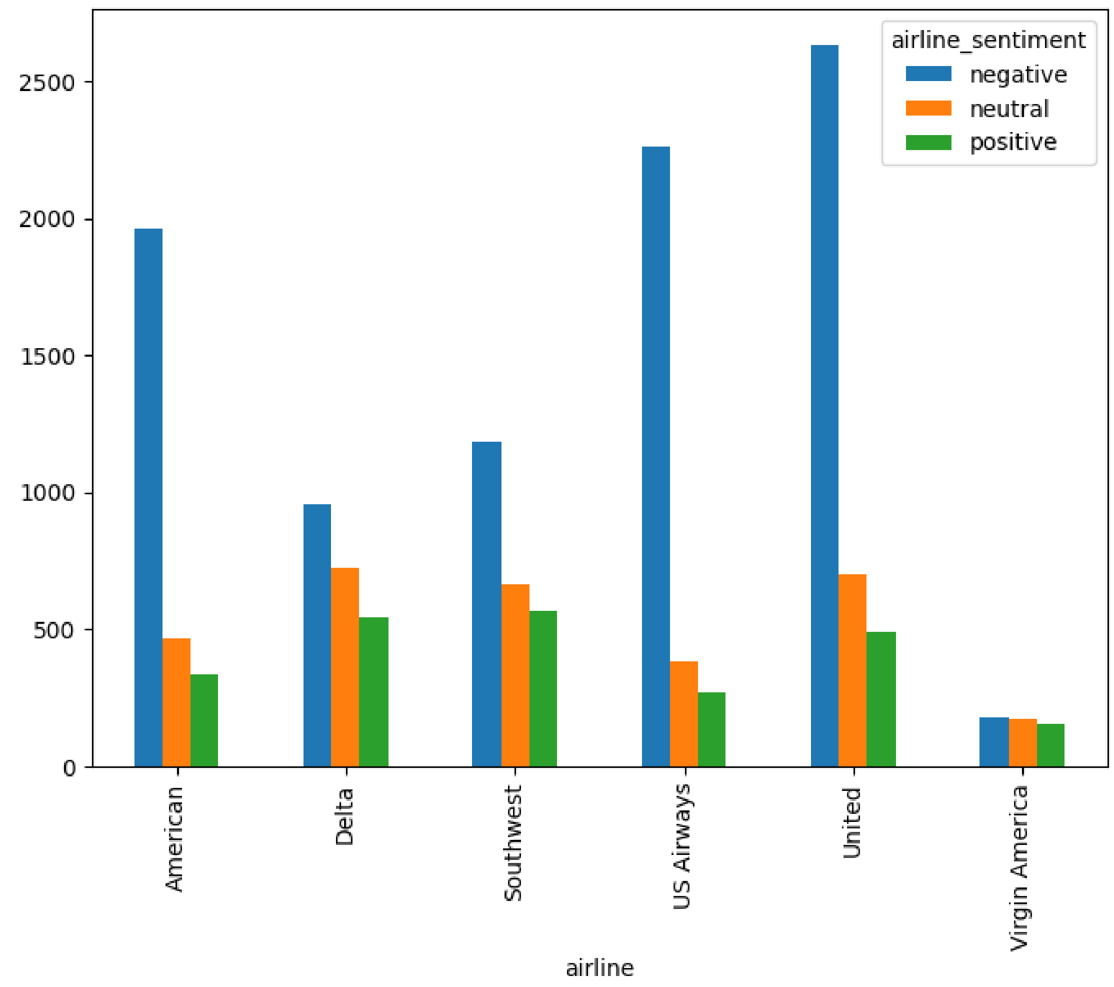
```
airline_tweets.airline_sentiment.value_counts().plot(kind='pie', autopct='%1.
0f%%', colors=["red", "yellow", "green"])
```

```
<Axes: ylabel='airline_sentiment'>
```



```
airline_sentiment = airline_tweets.groupby(['airline', 'airline_sentiment']).
airline_sentiment.count().unstack()
airline_sentiment.plot(kind='bar')
```

```
<Axes: xlabel='airline'>
```

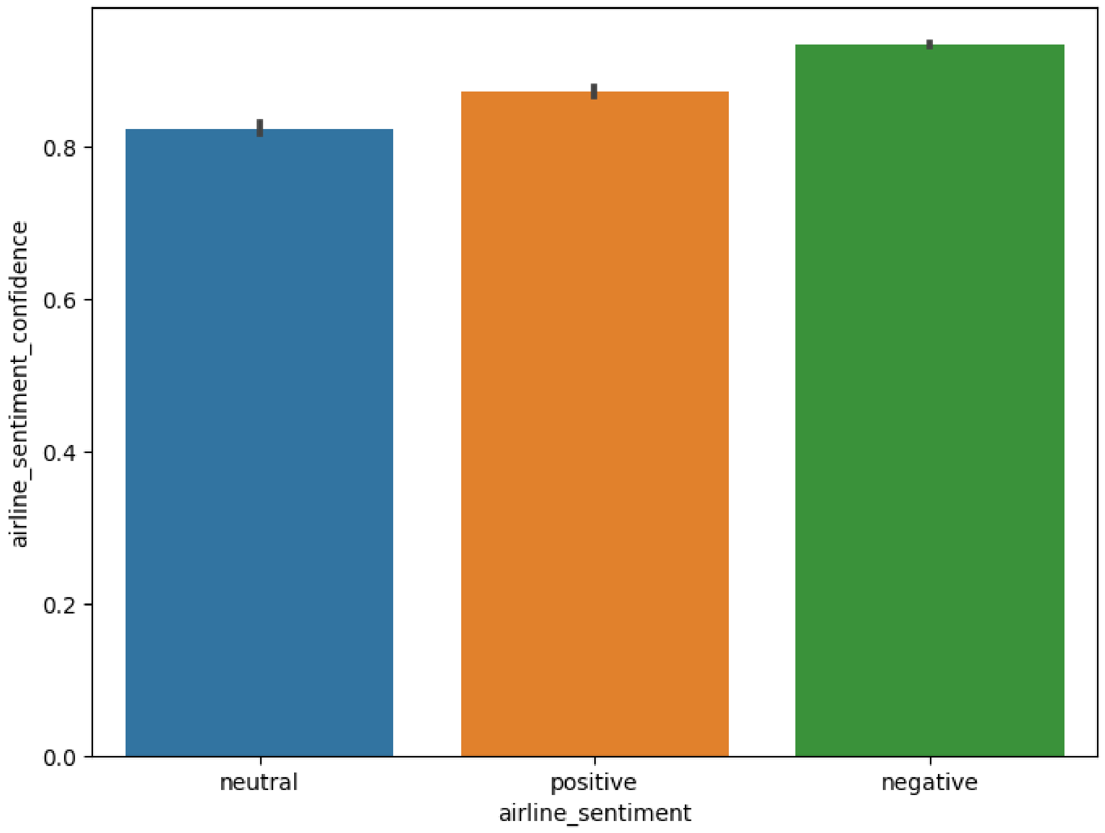


It is evident from the output that for almost all the airlines, the majority of the tweets are negative, followed by neutral and positive tweets. Virgin America is probably the only airline where the ratio of the three sentiments is somewhat similar.

Finally, let's use the Seaborn library to view the average confidence level for the tweets belonging to three sentiment categories.

```
import seaborn as sns

sns.barplot(x='airline_sentiment', y='airline_sentiment_confidence' , data=ai
rline_tweets)
```

```
<Axes: xlabel='airline_sentiment', ylabel='airline_sentiment_confidence'>
```



From the output, you can see that the confidence level for negative tweets is higher compared to positive and neutral tweets.

Data Cleaning

Tweets contain many slang words and punctuation marks. We need to clean our tweets before they can be used for training the machine learning model. However, before cleaning the tweets, let's divide our dataset into feature and label sets.

Our feature set will consist of tweets only. If we look at our dataset, the 11th column contains the tweet text. Note that the index of the column will be 10 since pandas columns follow zero-based indexing scheme where the first column is called 0th column. Our label set will consist of the sentiment of the tweet that we have to predict. The sentiment of the tweet is in the second column (index 1). To create a feature and a label set, we can use the iloc method off the pandas data frame.

```
features = airline_tweets.iloc[:, 10].values
labels = airline_tweets.iloc[:, 1].values

print("Missing Values by Column")
print("-"*30)
print(airline_tweets.isna().sum())
print("-"*30)
print("TOTAL MISSING VALUES:",airline_tweets.isna().sum().sum())
```

```
Missing Values by Column
------------------------------
tweet_id                            0
airline_sentiment                   0
airline_sentiment_confidence        0
negativereason                   5462
negativereason_confidence        4118
airline                             0
airline_sentiment_gold          14600
name                                0
negativereason_gold             14608
retweet_count                       0
text                                0
tweet_coord                     13621
tweet_created                       0
tweet_location                   4733
user_timezone                    4820
dtype: int64
------------------------------
TOTAL MISSING VALUES: 61962
```

Once we divide the data into features and training set, we can preprocess data in order to clean it.

```
processed_features = []

for sentence in range(0, len(features)):
    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', str(features[sentence]))

    # remove all single characters
    processed_feature= re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters from the start
    processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)

    # Substituting multiple spaces with single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)

    # Removing prefixed 'b'
    processed_feature = re.sub(r'^b\s+', '', processed_feature)

    # Converting to Lowercase
    processed_feature = processed_feature.lower()

    processed_features.append(processed_feature)
```

TF-IDF

In the bag of words approach, each word has the same weight. The idea behind the TF-IDF approach is that the words that occur less in all the documents and more in individual document contribute more towards classification.

TF-IDF is a combination of two terms. Term frequency and Inverse Document frequency. They can be calculated as:

TF = (Frequency of a word in the document) / (Total words in the document)

IDF = Log((Total number of docs) / (Number of docs containing the word))

TF-IDF using the Scikit-Learn Library

Luckily for us, Python's Scikit-Learn library contains the TfidfVectorizer class that can be used to convert text features into TF-IDF feature vectors. The following script performs this:

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer (max_features=2500, min_df=7, max_df=0.8, stop_w
ords=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In the code above, we define that the max_features should be 2500, which means that it only uses the 2500 most frequently occurring words to create a bag of words feature vector. Words that occur less frequently are not very useful for classification.

Similarly, max_df specifies that only use those words that occur in a maximum of 80% of the documents. Words that occur in all documents are too common and are not very useful for classification. Similarly, min-df is set to 7 which shows that include words that occur in at least 7 documents.

Dividing Data into Training and Test Sets

In the previous section, we converted the data into the numeric form. As the last step before we train our algorithms, we need to divide our data into training and testing sets. The training set will be used to train the algorithm while the test set will be used to evaluate the performance of the machine learning model.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(processed_features, label
s, test_size=0.2, random_state=0)
```

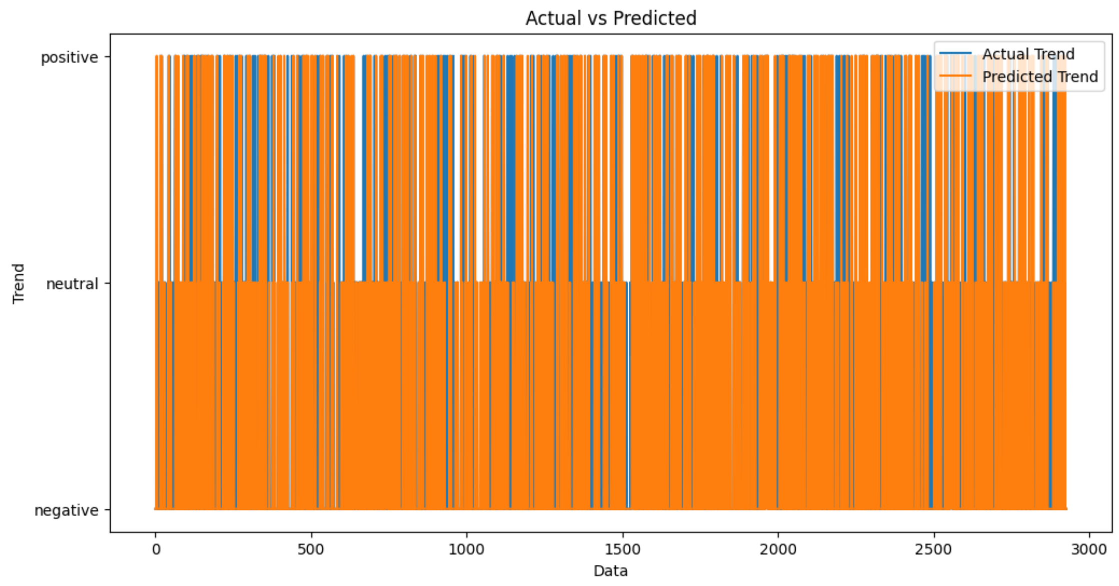
```
from sklearn.ensemble import RandomForestClassifier

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=200, random_state=0)
```

```
predictions = text_classifier.predict(X_test)
```

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(y_test)), y_test, label='Actual Trend')
plt.plot(np.arange(len(y_test)), predictions, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



```
from sklearn.metrics import confusion_matrix, accuracy_score

print(confusion_matrix(y_test,predictions))
print('accuracy score',accuracy_score(y_test, predictions))
```

```
[[1723  108   39]
 [ 326  248   40]
 [ 132   58  254]]
accuracy score 0.7599043715846995
```

CONCLUSION :

In conclusion, the sentiment analysis model we have developed is a powerful tool fo
r understanding and categorizing sentiments within textual data. This model leverages Nat
ural Language Processing techniques, particularly TF-IDF feature extraction and Logistic Re
gression, to process and analyze text data. Here are some key takeaways from this sentime
nt analysis model:

1. **Text Data Transformation:**The model effectively transforms raw text data into numeric
al feature vectors using TF-IDF vectorization. This allows for the quantification of textual in
formation, making it suitable for machine learning.

2. **Stop Word Removal and Lowercasing**: The removal of common English stop words an
d converting text to lowercase ensures that the analysis focuses on the most meaningful co
ntent while maintaining consistency in text representation.

3. **Training and Testing** :The model differentiates between training and testing datasets, e
nsuring that it generalizes well to unseen data. It fits the TF-IDF vectorizer on the training d
ata and then uses the same vectorizer to transform the test data.

4. **Interpretation of Results**: The model provides sentiment labels (positive, negative, or n
eutral) for analyzed text, enabling decision-makers to gauge public opinion, identify areas f
or improvement, and make informed decisions.

8. **Customization and Enhancement**: Depending on the specific requirements and domain
, the model can be customized further by fine-tuning the vectorizer, using more advanced al
gorithms, or incorporating additional features like sentiment lexicons.

Overall, this sentiment analysis model serves as a valuable tool for organizations see
king to extract actionable insights from the vast amount of text data available in today's dig
ital landscape. It empowers businesses to better understand customer sentiment, track mar
ket trends, and make data-driven decisions that can lead to improved products and service
s.