# PREDICTING HOUSE PRICE PREDICTION USING MACHINE LEARNING

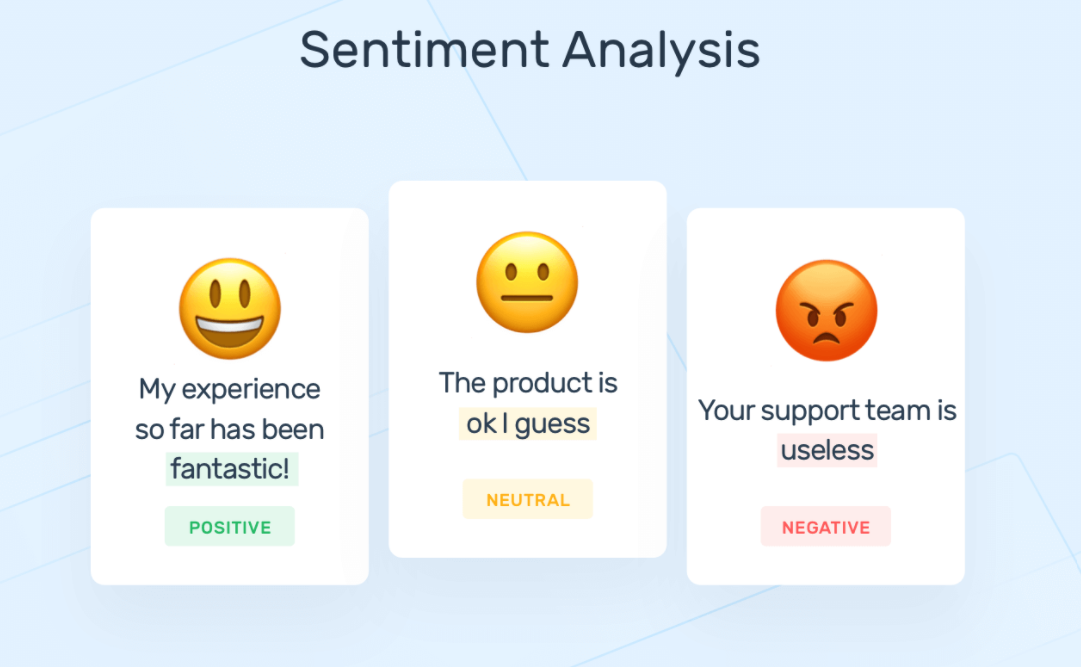
**T E A M M E M B E R**

**211121104054 : THULASI NAVANEETHAN . N**

**Phase 5 submission document**

**Project Title : House Price Predictor**

**Phase 5 : Project Documentation & Submission**

**Topic : In this section we will document the complete project and prepare it for submission**

# 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

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) technique that involves the use of machine learning and computational linguistics to determine and categorize the sentiment or emotional tone expressed in a piece of text. This text could be in the form of social media posts, customer reviews, news articles, or any other text data. Sentiment analysis is a valuable tool for understanding and extracting insights from large volumes of textual data, as it helps automate the process of assessing public opinion and sentiment about various topics, products, or services.

A sentiment analysis model typically involves the following key steps:

1. Text Preprocessing:

The first step in sentiment analysis is to clean and preprocess the text data. This includes tasks like removing punctuation, converting text to lowercase, and tokenization (splitting text into individual words or tokens).

2. Feature Extraction:

After preprocessing, the model extracts relevant features from the text data. These features can be as simple as word counts or more complex, such as word embeddings, which capture the semantic meaning of words.

3. Sentiment Classification:

The core of sentiment analysis is classifying the sentiment expressed in the text. This classification can be binary (positive/negative) or more fine-grained, with multiple sentiment classes (e.g., positive, negative, neutral, or different emotional states like happy, sad, angry).

4. Model Training:

Sentiment analysis models are typically trained on labeled datasets, where the sentiment of each text is known. Machine learning algorithms, such as support vector machines, decision trees, or deep learning techniques like recurrent neural networks (RNNs) or transformer models, are often used to build sentiment classifiers. These models learn to recognize patterns in the data and make predictions about sentiment.

5. Evaluation and Testing:

Once the model is trained, it is evaluated using a separate dataset to assess its performance. Common evaluation metrics include accuracy, precision, recall, and F1 score.

6. Application:

After the model is trained and evaluated, it can be applied to analyze the sentiment of new, unseen text data. This can be used for various purposes, such as monitoring social media sentiment about a brand, tracking public opinion about a political candidate, or automatically categorizing customer reviews.

Sentiment analysis has a wide range of applications in various industries, including marketing, customer service, finance, and social listening. It can help organizations gain valuable insights into how people feel about their products or services, identify potential issues or opportunities, and make data-driven decisions to improve their offerings or better engage with their target audience.

The field of sentiment analysis continues to evolve, with more sophisticated models and techniques emerging, especially in the era of deep learning and transformer-based models like BERT and GPT. These models have improved the accuracy and fine-grained analysis of sentiment in text, making sentiment analysis an essential tool for businesses and researchers alike.

**Tools And Softwares Commonly Used In The Process :**

There are several tools and software commonly used in the process of sentiment analysis, ranging from programming libraries for developers to user-friendly platforms for non-technical users. Here's a list of some of the popular tools and software used in sentiment analysis:

1. Natural Language Processing Libraries:

- NLTK (Natural Language Toolkit): A Python library for natural language processing, offering tools for tokenization, stemming, and sentiment analysis.

2. TextBlob: A simplified NLP library for Python that provides an easy-to-use API for tasks like sentiment analysis, part-of-speech tagging, and more.

3. spaCy: An open-source NLP library for Python that offers efficient text processing, including sentiment analysis, part-of-speech tagging, and named entity recognition.

4. scikit-learn: A machine learning library for Python that includes tools for building and evaluating sentiment analysis models.

5. Transformers (Hugging Face): The Transformers library provides pre-trained transformer models, such as BERT, GPT-3, and RoBERTa, which can be fine-tuned for sentiment analysis.

6. VADER (Valence Aware Dictionary and sEntiment Reasoner): A lexicon and rule-based sentiment analysis tool designed for social media text.

7. IBM Watson Natural Language Understanding: A cloud-based NLP service that offers sentiment analysis among other features, making it suitable for both developers and non-technical users.

8. Google Cloud Natural Language API: A cloud-based service that provides sentiment analysis, entity recognition, and other NLP capabilities.

9. Amazon Comprehend: Amazon's NLP service that includes sentiment analysis as one of its features.

10. RapidMiner: A data science platform that offers sentiment analysis as part of its analytics and data preprocessing capabilities.

11. Lexalytics: A text analytics platform that provides sentiment analysis, entity recognition, and various other NLP features.

12. Sentiment Analysis Tools for Social Media:

- Brandwatch: A social listening and analytics tool that includes sentiment analysis for tracking brand mentions on social media platforms.

- Socialbakers: A social media marketing and analytics platform with sentiment analysis capabilities.

- Talkwalker: A social media analytics tool that offers sentiment analysis, trend tracking, and influencer identification.

13. Tableau: A data visualization tool that can be used to create sentiment analysis dashboards and visualizations.

14. Excel with Text Analysis Add-Ins: Excel users can leverage add-ins like the "Microsoft Text Analytics" add-in to perform basic sentiment analysis.

15. RapidAPI: A platform that offers various sentiment analysis APIs, making it easy to integrate sentiment analysis into applications.

These tools and software options cater to a variety of needs, from text analysis for developers to user-friendly platforms for business and marketing professionals. The choice of tool or software depends on factors such as the specific use case, technical expertise, and budget constraints.

# Introduction to Design Thinking

# Design thinking is a human-centered problem-solving approach that focuses on understanding user needs, ideating creative solutions, and iterating through prototyping and testing. It encourages a collaborative and iterative process to develop innovative solutions to complex problems. When presenting the principles and process of design thinking in a document, you can structure it as follows:

# Design thinking is a creative and user-centric problem-solving framework that empowers teams to tackle complex challenges. It emphasizes empathy, collaboration, and iteration to generate innovative solutions. This document provides an overview of design thinking principles and its five key stages.

# Key Principles of Design Thinking

# 1. \*\*Empathy\*\*: Understand and empathize with the needs and perspectives of the end-users.

# 2. \*\*Define\*\*: Clearly define the problem or challenge you're addressing.

# 3. \*\*Ideate\*\*: Generate a wide range of creative solutions without judgment.

# 4. \*\*Prototype\*\*: Develop tangible representations of your ideas.

# 5. \*\*Test\*\*: Gather feedback from users through testing and refine your solutions.

# Five Stages of Design Thinking

# 1. Empathize

# In this stage, the focus is on understanding the end-users' needs, goals, and pain points. Techniques include interviews, observations, and surveys to gather insights.

# 2. Define

# Define the problem you're addressing based on the insights gained in the empathy stage. Create a problem statement that guides your design efforts.

# 3. Ideate

# This stage encourages brainstorming and idea generation without constraints. Consider various approaches and solutions, no matter how unconventional they may seem.

# 4. Prototype

# Develop tangible representations of your ideas, which can be sketches, wireframes, or even fully functional models. Prototypes are created to test and iterate upon.

# 5. Test

# Gather user feedback by testing your prototypes. Use this feedback to refine and improve your solutions.

# 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.

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

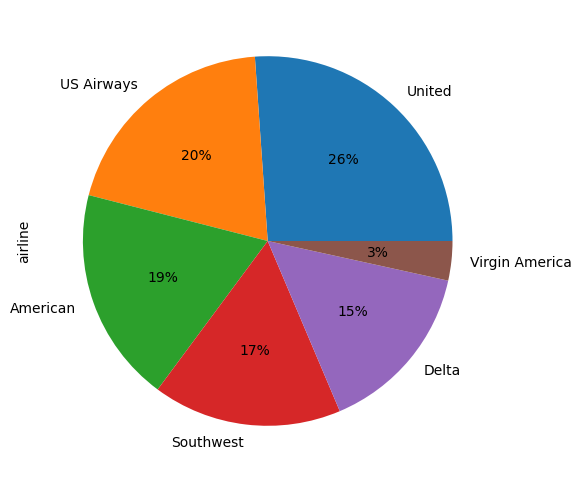
6.4  
4.8

# 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%%')

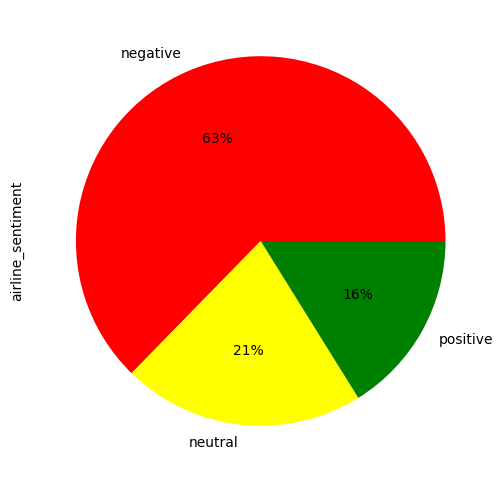
<Axes: ylabel='airline'>



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

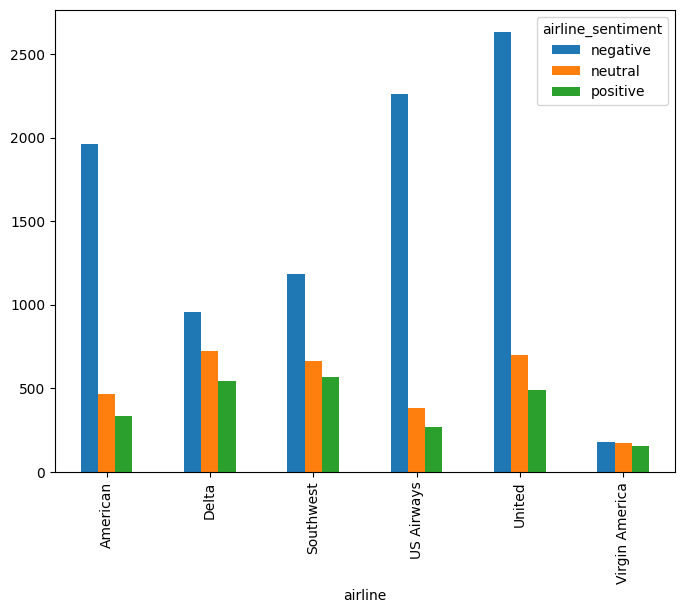
airline\_tweets.airline\_sentiment.value\_counts().plot(kind='pie', autopct='%1.0f%%', colors=["red", "yellow", "green"])

<Axes: ylabel='airline\_sentiment'>



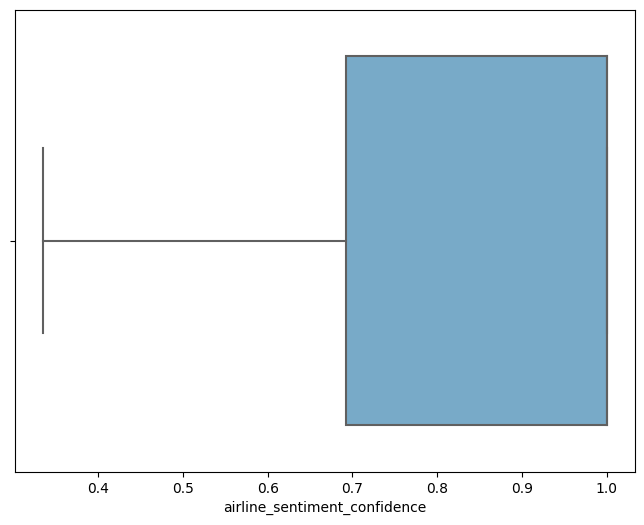
airline\_sentiment = airline\_tweets.groupby(['airline', 'airline\_sentiment']).airline\_sentiment.count().unstack()  
airline\_sentiment.plot(kind='bar')

<Axes: xlabel='airline'>



sns.boxplot(airline\_tweets, x='airline\_sentiment\_confidence', palette='Blues')

<Axes: xlabel='airline\_sentiment\_confidence'>

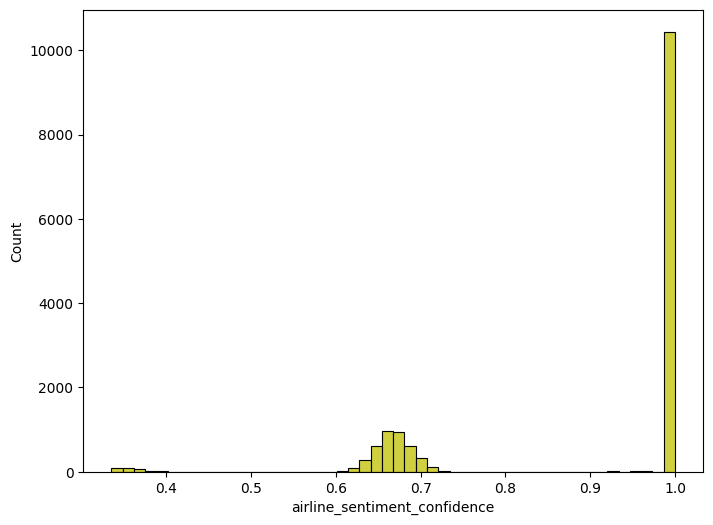


It is evident from the output that for almost all the airlines, the majority of the tweets are negative, followed by neutral and positive tweets. Virgin America is probably the only airline where the ratio of the three sentiments is somewhat similar.

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

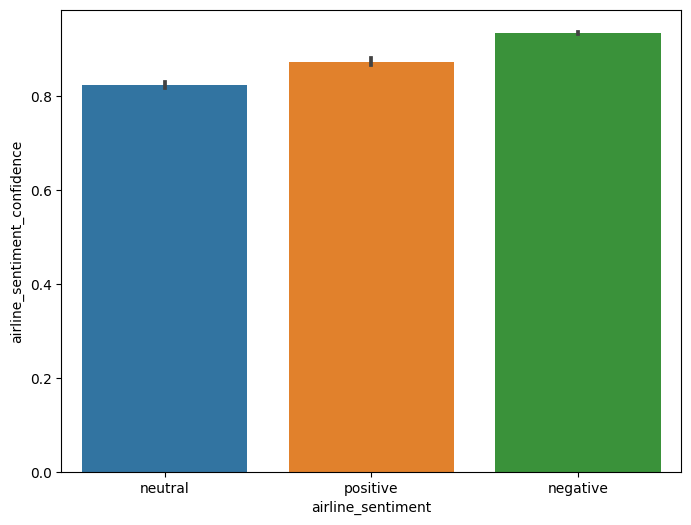
sns.histplot(airline\_tweets, x='airline\_sentiment\_confidence', bins=50, color='y')

<Axes: xlabel='airline\_sentiment\_confidence', ylabel='Count'>



import seaborn as sns  
  
sns.barplot(x='airline\_sentiment', y='airline\_sentiment\_confidence' , data=airline\_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\_words=stopwords.words('english'))  
processed\_features = vectorizer.fit\_transform(processed\_features).toarray()

[nltk\_data] Downloading package stopwords to /root/nltk\_data...  
[nltk\_data] Unzipping corpora/stopwords.zip.

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, labels, test\_size=0.2, random\_state=0)

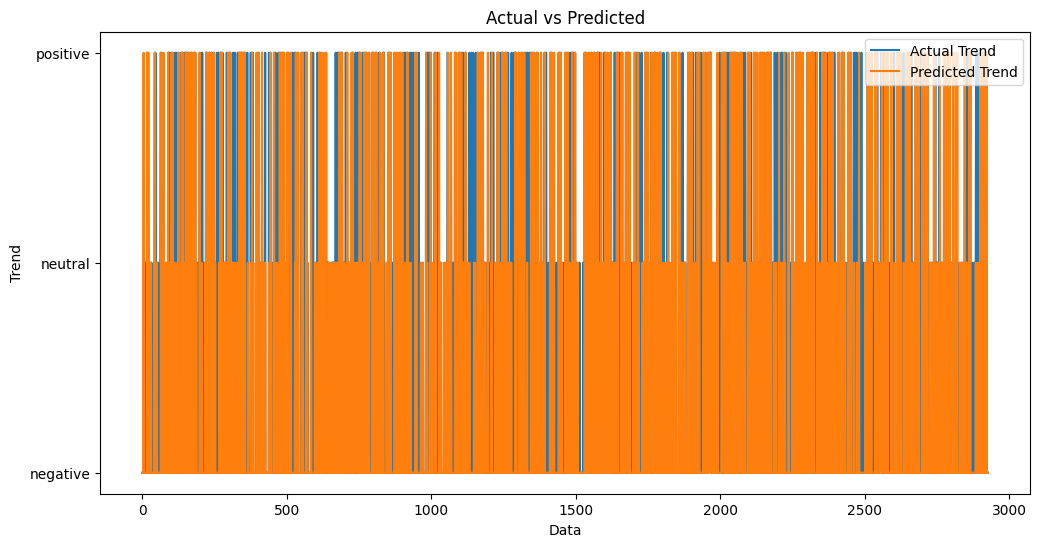
from sklearn.ensemble import RandomForestClassifier  
  
text\_classifier = RandomForestClassifier(n\_estimators=200, random\_state=0)  
text\_classifier.fit(X\_train, y\_train)

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 the realm of data-driven endeavors, from data analysis to machine learning, the significance of a well-prepared dataset cannot be overstated. The process of building, loading, and preprocessing the dataset serves as the cornerstone of success.

.

Once the data is loaded, \*\*data preprocessing\*\* takes center stage. This is where the raw data undergoes a transformation from its rough, unrefined state to a polished, analysis-ready format. Missing values are addressed, anomalies are rectified, and the data is standardized. Data cleaning and transformation are key to ensuring that the dataset is accurate, consistent, and devoid of errors.

Data preprocessing isn't just about tidying up the data; it's also about molding it into a format that can be consumed by your chosen analysis or machine learning algorithms. \*\*Feature engineering\*\*, one of the critical steps, can unearth insights and relationships within the data that may not be immediately apparent. It can open doors to new possibilities and avenues of exploration.

After preprocessing, the dataset is ready for \*\*data exploration\*\*, a phase that allows you to delve into the data's depths. Descriptive statistics and visualizations reveal the stories hidden within the dataset. Feature relationships become clearer, and you may uncover patterns that can drive further analysis and decision-making.

Finally, after all these meticulous steps, the dataset is saved, safeguarding your hard work and preparation for the challenges ahead. It's not just about preserving your efforts; it's also about maintaining data integrity for future use.

In conclusion, the process of building, loading, and preprocessing a dataset isn't merely a series of technical tasks; it's an art form and a science. It requires attention to detail, a commitment to data quality, and an unwavering dedication to ensuring that your data is primed for the tasks that follow. A well-prepared dataset is the catalyst for informed decisions, the key to unlocking valuable insights, and the driving force behind innovative solutions.