

✓ Sentiment Analysis

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
import nltk
```

✓ Install necessary libraries in Google Colab

```
!pip install nltk seaborn
```



```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
```

✓ Download NLTK data

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('punkt_tab')
```

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

✓ Load Dataset

```
data = {
    'text': [
        "I love this product! It's amazing.",
        "Worst experience ever. Do not recommend.",
        "Okayish, not too great but not bad either.",
        "Absolutely fantastic! Worth every penny.",
        "Terrible quality. Waste of money.",
        "Loved it! Highly recommend to everyone.",
        "Not happy with the purchase. Poor service.",
        "Satisfied with the product. Decent quality.",
        "Exceeded my expectations! Five stars.",
        "Horrible experience. Never buying again."
    ],
    'sentiment': [1, 0, 1, 1, 0, 1, 0, 1, 1, 0]
}
data
```

```
➦ {'text': ["I love this product! It's amazing.",
            'Worst experience ever. Do not recommend.',
            'Okayish, not too great but not bad either.',
            'Absolutely fantastic! Worth every penny.',
            'Terrible quality. Waste of money.',
            'Loved it! Highly recommend to everyone.',
            'Not happy with the purchase. Poor service.',
            'Satisfied with the product. Decent quality.',
            'Exceeded my expectations! Five stars.',
            'Horrible experience. Never buying again.'],
    'sentiment': [1, 0, 1, 1, 0, 1, 0, 1, 1, 0]}
```

Convert to DataFrame

```
df = pd.DataFrame(data)
print("Sample Dataset:")
print(df.head())
```

↗ Sample Dataset:


	text	sentiment
0	I love this product! It's amazing.	1
1	Worst experience ever. Do not recommend.	0
2	Okayish, not too great but not bad either.	1
3	Absolutely fantastic! Worth every penny.	1
4	Terrible quality. Waste of money.	0

Data Preprocessing

```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'^a-zA-Z\s', '', text)
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    return ' '.join(tokens)
```

```
df['cleaned_text'] = df['text'].apply(preprocess_text)
df.head()
```

↗

	text	sentiment	cleaned_text	
0	I love this product! It's amazing.	1	love product amazing	
1	Worst experience ever. Do not recommend.	0	worst experience ever recommend	
2	Okayish, not too great but not bad either.	1	okayish great bad either	
3	Absolutely fantastic! Worth every penny.	1	absolutely fantastic worth every penny	
4	Terrible quality. Waste of money.	0	terrible quality waste money	

Next steps:

[Generate code with df](#)

 [View recommended plots](#)

[New interactive sheet](#)

Vectorization

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['cleaned_text'])
y = df['sentiment']
```

✓ Split Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ Model Implementation

```
model = LogisticRegression()
model.fit(X_train, y_train)
```



▼ LogisticRegression ⓘ ?

```
LogisticRegression()
```

✓ Model Evaluation

```
y_pred = model.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
print("Confusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

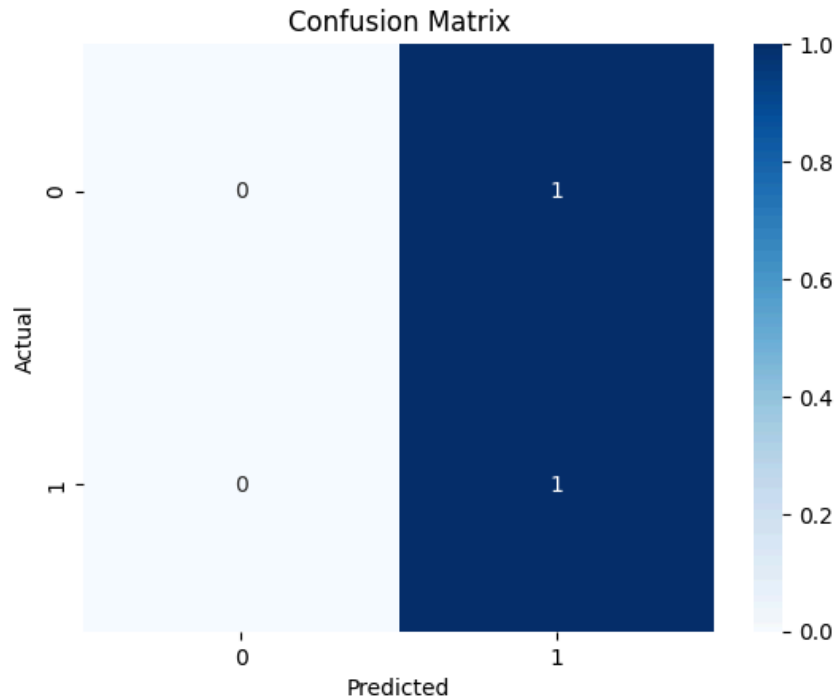


```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

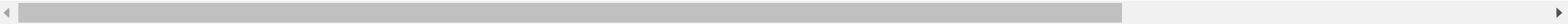
Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.50	1.00	0.67	1
accuracy			0.50	2
macro avg	0.25	0.50	0.33	2
weighted avg	0.25	0.50	0.33	2

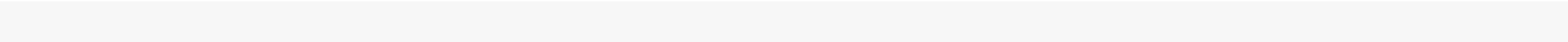
Confusion Matrix:



Accuracy: 50.00%



Insights



```
print("\nKey Insights:")
print(f"Model achieved an accuracy of {accuracy * 100:.2f}%.")
print("The model can be further improved with more data and hyperparameter tuning.")
```



Key Insights:

Model achieved an accuracy of 50.00%.

The model can be further improved with more data and hyperparameter tuning.