INTERN:-Bhagyashri Sharad Pisal

Intern ID:- CT4MOTH

Domain:- Machine Learning

Duration:-January 25, 2025, to May 25, 2025

Company:- CODETECH IT SOLUTIONS

Mentor:- Neela Santhosh Kumar

TASK TWO: SENTIMENT ANALYSIS WITH NLP

PERFORM SENTIMENT ANALYSIS ON A DATASET OF CUSTOMER REVIEWS USING TF-IDF VECTORIZATION AND LOGISTIC REGRESSION.

DELIVERABLE: A JUPYTER NOTEBOOK SHOWCASING PREPROCESSING, MODELING,AND SENTIMENT EVALUATION

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Step 1: Load the dataset
# Creating a simple dataset with customer reviews and sentiment labels
# 1 = Positive sentiment, 0 = Negative sentiment
data = pd.DataFrame({
    "review": [
        "This product is amazing!",
        "Terrible service, would not recommend.",
        "I love it, works perfectly.",
        "Not worth the money. Completely dissatisfied.",
        "Very good quality and fast delivery.",
        "Awful experience. The item arrived broken.",
        "Exceptional! Exceeded my expectations.",
        "Mediocre at best. Expected better.",
        "Fantastic! Will buy again.",
        "Poor customer service. Never buying here again."
    ],
    "sentiment": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # Sentiment labels
})
```

```
# Display dataset preview
print("Dataset preview:")
print(data.head())
```

```
Dataset preview:
                                       review  sentiment
0                     This product is amazing!          1
1         Terrible service, would not recommend.          0
2                   I love it, works perfectly.          1
3  Not worth the money. Completely dissatisfied.          0
4           Very good quality and fast delivery.          1
```

```
# Step 2: Split the dataset into features (X) and labels (y)
X = data['review']
y = data['sentiment']
```

```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Step 4: Vectorize the text data using TF-IDF
# Transform the text data into numerical format for modeling
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

```
# Step 5: Initialize the Logistic Regression model
model = LogisticRegression(random_state=42)
```

```
# Step 6: Train the model on the training data
model.fit(X_train_tfidf, y_train)
```

```
LogisticRegression(random_state=42)
```

```
# Step 7: Make predictions on the test set
y_pred = model.predict(X_test_tfidf)
```

Generate | 10 random numbers using numpy | Close

```
# Step 8: Evaluate the model
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

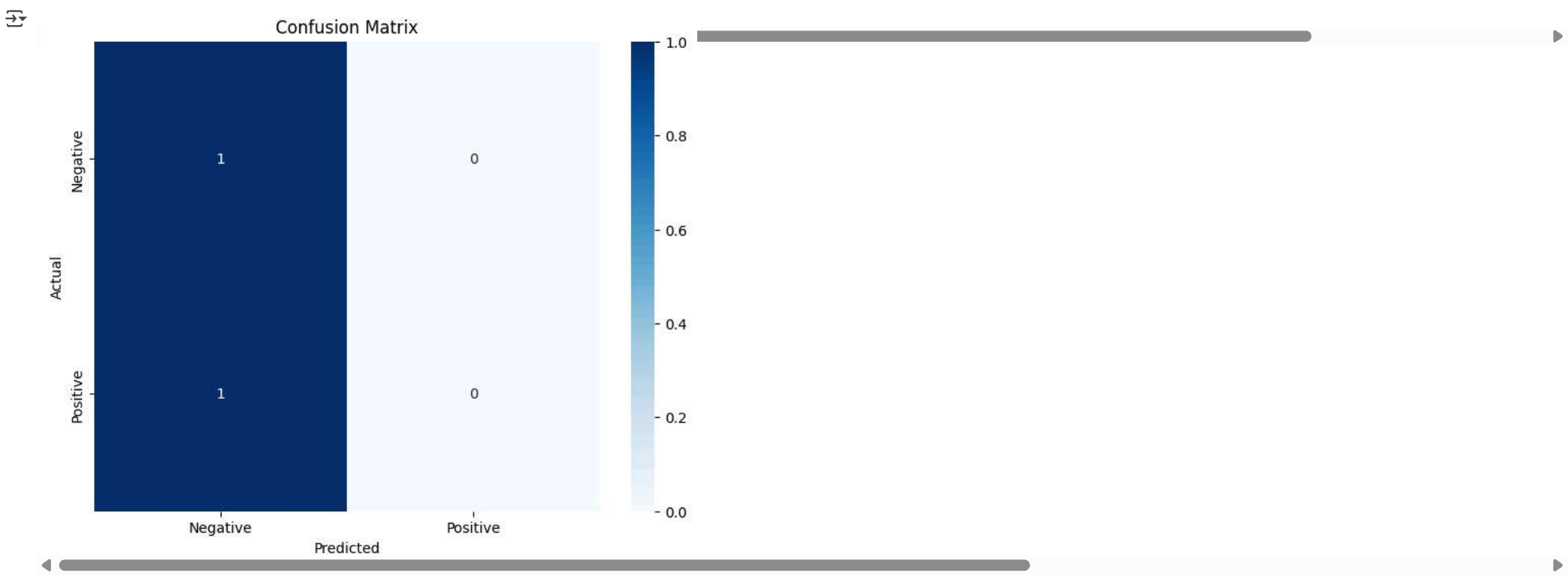
```
Model Accuracy: 0.50
```

```
# Print classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
ation Report:

              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
   avg       0.25      0.50      0.33         2

l/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
```

```
rf(average, modifier, f"{metric.capitalize()} is", len(result))
l/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
rf(average, modifier, f"{metric.capitalize()} is", len(result))
l/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_divisio
rf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Step 9: Plot the Confusion Matrix
# Visualize the confusion matrix to better understand predictions
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

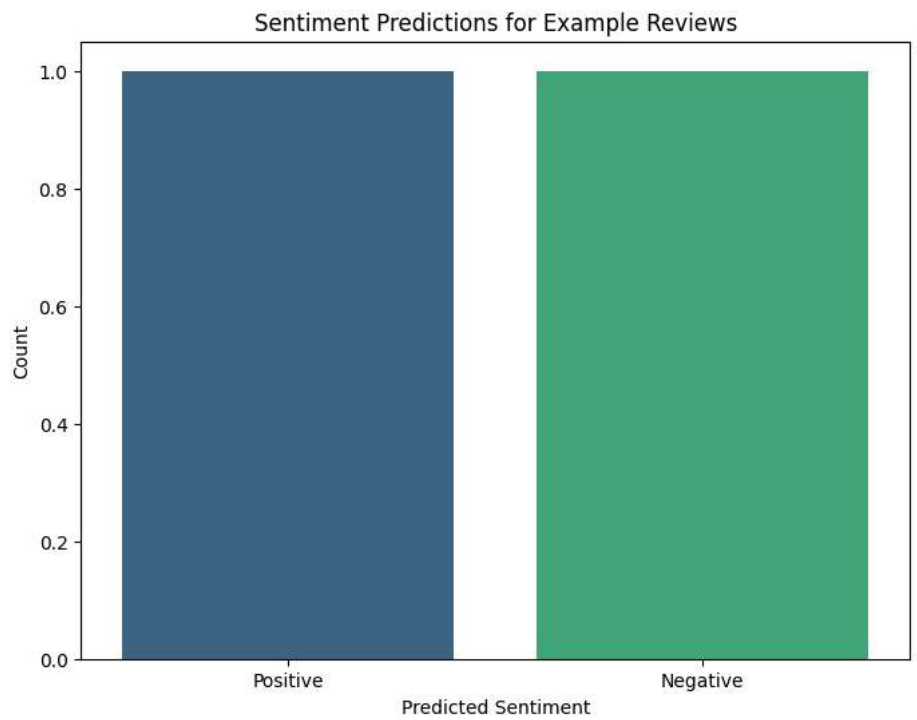



```
# Step 10: Example predictions on new reviews
# Input some example reviews to test the model
example_reviews = [
    "The product quality is excellent!",
    "Horrible experience, will not return.",
    "Decent product but expected better.",
    "Absolutely love it! Fast delivery too."
]
# Transform the example reviews using the TF-IDF vectorizer
example_tfidf = vectorizer.transform(example_reviews)

# Predict the sentiment for each example review
example_preds = model.predict(example_tfidf)

# Display predictions
for review, sentiment in zip(example_reviews, example_preds):
    sentiment_label = "Positive" if sentiment == 1 else "Negative"
    print(f"Review: '{review}' | Sentiment: {sentiment_label}")
```

```
Review: 'The product quality is excellent!' | Sentiment: Positive
Review: 'Horrible experience, will not return.' | Sentiment: Negative
Review: 'Decent product but expected better.' | Sentiment: Negative
Review: 'Absolutely love it! Fast delivery too.' | Sentiment: Positive
```

```
# Step 11: Visualize Example Predictions
# Create a bar plot to visualize the sentiment predictions
sentiments = ["Positive" if s == 1 else "Negative" for s in example_preds]
plt.figure(figsize=(8, 6))
sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
plt.title("Sentiment Predictions for Example Reviews")
plt.xlabel("Predicted Sentiment")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-14-10ca4184b91f>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
```



```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

Generate | a slider using jupyter widgets | Close

```
# Step 1: Load the dataset
# Creating a simple dataset with customer reviews and sentiment labels
# 1 = Positive sentiment, 0 = Negative sentiment
data = pd.DataFrame({
    "review": [
        "This product is amazing!",
        "Terrible service, would not recommend.",
        "I love it, works perfectly.",
        "Not worth the money. Completely dissatisfied.",
        "Very good quality and fast delivery.",
        "Awful experience. The item arrived broken.",
        "Exceptional! Exceeded my expectations.",
        "Mediocre at best. Expected better.",
        "Fantastic! Will buy again.",
        "Poor customer service. Never buying here again."
    ],
    "sentiment": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # Sentiment labels
})


# Display dataset preview
print("Dataset preview:")
print(data.head(0))
```

```
Dataset preview:
Empty DataFrame
Columns: [review, sentiment]
Index: []
```

Generate | 10 random numbers using numpy | Close

```
print(data.head(5))
print(data.tail(5))
```

```
                                          review  sentiment
0                       This product is amazing!          1
1         Terrible service, would not recommend.          0
2                    I love it, works perfectly.          1
3  Not worth the money. Completely dissatisfied.          0
4           Very good quality and fast delivery.          1
                                            review  sentiment
5       Awful experience. The item arrived broken.          0
6           Exceptional! Exceeded my expectations.          1
7               Mediocre at best. Expected better.          0
8                       Fantastic! Will buy again.          1
9  Poor customer service. Never buying here again.          0
```

```
# Step 2: Split the dataset into features (X) and labels (y)
X = data['review']
y = data['sentiment']


# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Vectorize the text data using TF-IDF
# Transform the text data into numerical format for modeling
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Generate | create a dataframe with 2 columns and 10 rows | Close

```
# Step 5: Initialize the Logistic Regression model
model = LogisticRegression(random_state=42)


# Step 6: Train the model on the training data
model.fit(X_train_tfidf, y_train)
```

```
▾ LogisticRegression
LogisticRegression(random_state=42)
```

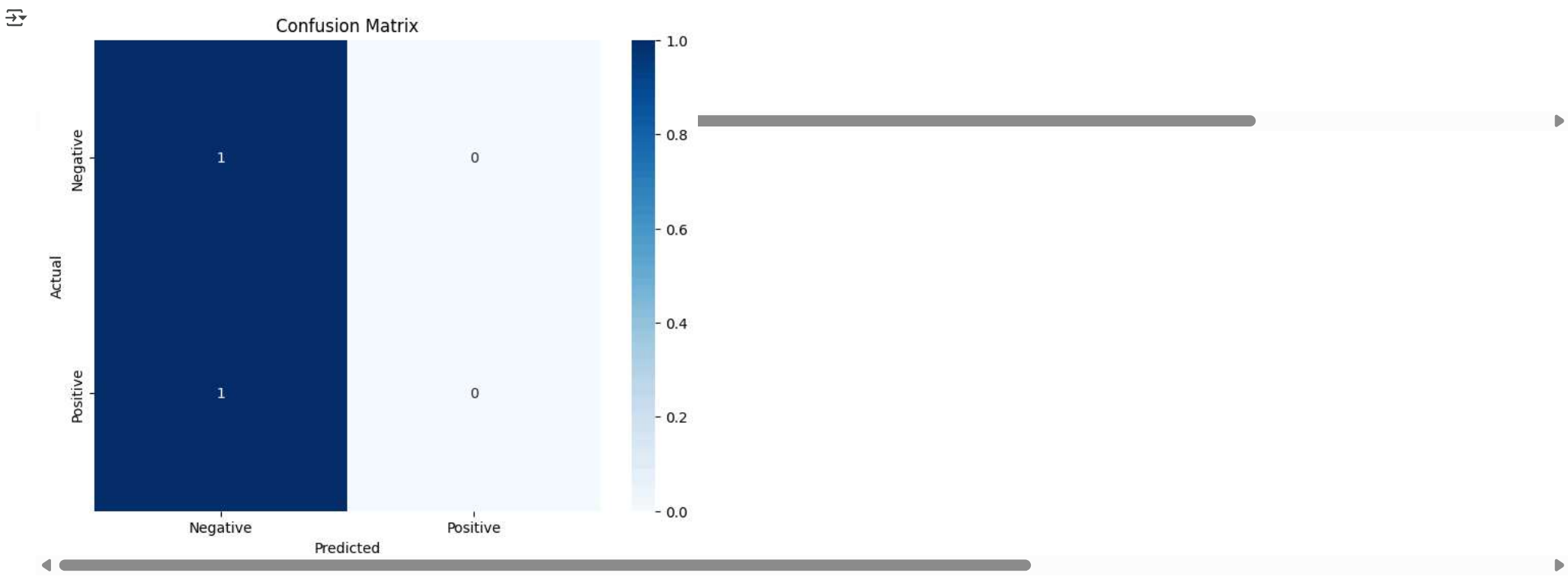
```
# Step 7: Make predictions on the test set
y_pred = model.predict(X_test_tfidf)


# Step 8: Evaluate the model
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
Model Accuracy: 0.50
```

```
# Print classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zer
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Step 9: Plot the Confusion Matrix
# Visualize the confusion matrix to better understand predictions
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```




```
# Step 10: Example predictions on new reviews
# Input some example reviews to test the model
example_reviews = [
    "The product quality is excellent!",
    "Horrible experience, will not return.",
    "Decent product but expected better.",
    "Absolutely love it! Fast delivery too."
]

# Transform the example reviews using the TF-IDF vectorizer
example_tfidf = vectorizer.transform(example_reviews)

# Predict the sentiment for each example review
example_preds = model.predict(example_tfidf)

# Display predictions
for review, sentiment in zip(example_reviews, example_preds):
    sentiment_label = "Positive" if sentiment == 1 else "Negative"
    print(f"Review: '{review}' | Sentiment: {sentiment_label}")
```

```
Review: 'The product quality is excellent!' | Sentiment: Positive
Review: 'Horrible experience, will not return.' | Sentiment: Negative
Review: 'Decent product but expected better.' | Sentiment: Negative
Review: 'Absolutely love it! Fast delivery too.' | Sentiment: Positive
```

```
# Step 11: Visualize Example Predictions
# Create a bar plot to visualize the sentiment predictions
sentiments = ["Positive" if s == 1 else "Negative" for s in example_preds]
plt.figure(figsize=(8, 4))
sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
plt.title("Sentiment Predictions for Example Reviews")
plt.xlabel("Predicted Sentiment")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-28-4dbf9e5b7dda>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=sentiments, y=[1]*len(sentiments), palette="viridis", dodge=False)
```