```
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential

df = pd.read_csv("/content/drive/MyDrive/movie/amazon_reviews.csv")

 df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 4915,\n  \"fields\":
[\n    {\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1418,\n        \"min\": 0,\n
\"max\": 4914,\n        \"num_unique_values\": 4915,\n
\"samples\": [\n          2346,\n          4344,\n          691\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"reviewerName\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 4594,\n        \"samples\": [\n
\"nta699\",\n          \"Maximus\",\n          \"G. Jackson\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"overall\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.9968451383820338,\n        \"min\": 1.0,\n        \"max\": 5.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          5.0,\n
2.0,\n          3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"reviewText\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4912,\n
\"samples\": [\n          \"Bought this for extended memory in a
Galaxy S III.  Great price. Easy to install and the phone recognized
it instantly. Does what its supposed to do.\",\n          \"I bought
this after I found out that you can add a memory card to the Samsung
galaxy s4! I never even knew that you could, and so once I found out I
bought this and installed it. It's worked perfectly since and holds
all of my songs and videos.  Great buy!\",\n          \"Using it on a
Canon 6D. No camplains at all. Really fast access. The adapter itself
is not at same level, even though. Bad connection, had to use a
previous adapter that I already had.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"reviewTime\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 690,\n        \"samples\": [\n          \"2013-
03-12\",\n          \"2013-09-10\",\n          \"2012-12-09\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"day_diff\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
209,\n        \"min\": 1,\n        \"max\": 1064,\n
```

\"num_unique_values\": 690,\n   \"samples\": [\n   636,\n
454,\n   729\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"helpful_yes\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 41,\n   \"min\": 0,\n
\"max\": 1952,\n   \"num_unique_values\": 23,\n
\"samples\": [\n   1428,\n   6,\n   0\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"helpful_no\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 4,\n   \"min\": 0,\n
\"max\": 183,\n   \"num_unique_values\": 17,\n
\"samples\": [\n   0,\n   1,\n   10\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"total_vote\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
44,\n   \"min\": 0,\n   \"max\": 2020,\n
\"num_unique_values\": 26,\n   \"samples\": [\n   495,\n
1505,\n   0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"score_pos_neg_diff\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 39,\n   \"min\": -130,\n
\"max\": 1884,\n   \"num_unique_values\": 27,\n
\"samples\": [\n   -2,\n   52,\n   -3\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"score_average_rating\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.25606237802879933,\n   \"min\":
0.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 28,\n
\"samples\": [\n   0.3333333333333333,\n
0.3076923076923077,\n   0.8571428571428571\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"wilson_lower_bound\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.0771874411204749,\n   \"min\": 0.0,\n   \"max\":
0.9575439475520824,\n   \"num_unique_values\": 40,\n
\"samples\": [\n   0.6456695649333126,\n
0.4364971778135299,\n   0.3755346297625253\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4915 entries, 0 to 4914
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

```
 0   Unnamed: 0            4915 non-null   int64
 1   reviewerName          4914 non-null   object
 2   overall               4915 non-null   float64
 3   reviewText            4914 non-null   object
 4   reviewTime            4915 non-null   object
 5   day_diff              4915 non-null   int64
 6   helpful_yes           4915 non-null   int64
 7   helpful_no            4915 non-null   int64
 8   total_vote            4915 non-null   int64
 9   score_pos_neg_diff    4915 non-null   int64
 10  score_average_rating  4915 non-null   float64
 11  wilson_lower_bound    4915 non-null   float64
dtypes: float64(3), int64(6), object(3)
memory usage: 460.9+ KB
```

```
null_values = df.isnull().sum()
print("Null values in the entire Data:")
print(null_values)
```

```
Null values in the entire Data:
Unnamed: 0              0
reviewerName            1
overall                 0
reviewText              1
reviewTime              0
day_diff                0
helpful_yes             0
helpful_no              0
total_vote              0
score_pos_neg_diff      0
score_average_rating    0
wilson_lower_bound      0
dtype: int64
```

```
df.dropna(inplace=True)
```

```
null_values = df.isnull().sum()
null_values
```

```
Unnamed: 0              0
reviewerName            0
overall                 0
reviewText              0
reviewTime              0
day_diff                0
helpful_yes             0
helpful_no              0
total_vote              0
score_pos_neg_diff      0
score_average_rating    0
```

```
wilson_lower_bound      0
dtype: int64
```

```
df.drop_duplicates(inplace=True)
```

```
import pandas as pd
import string

# Sample dataframe
data = {'Review': ["This is a great movie!", "I didn't like it.",
"Amazing performance by the lead actor."]}
df = pd.DataFrame(data)

# Convert reviews to lowercase
df['Review'] = df['Review'].apply(lambda x: x.lower())

# Remove punctuation from reviews
df['Review'] = df['Review'].apply(lambda x:
x.translate(str.maketrans('', '', string.punctuation)))

print(df)
```

```
                                  Review
0                 this is a great movie
1                      i didnt like it
2  amazing performance by the lead actor
```

```
 df['Review']
```

```
0                    this is a great movie
1                         i didnt like it
2    amazing performance by the lead actor
Name: Review, dtype: object
```

```
from sklearn.feature_extraction.text import CountVectorizer
# Assuming 'df' is your Data containing text data
text_data = df['Review']
vectorizer = CountVectorizer()
feature_matrix = vectorizer.fit_transform(text_data)
feature_names = vectorizer.get_feature_names_out()
```

```
feature_names
```

```
array(['actor', 'amazing', 'by', 'didnt', 'great', 'is', 'it', 'lead',
       'like', 'movie', 'performance', 'the', 'this'], dtype=object)
```

```
import sklearn.feature_extraction.text as text
count_vectorizer = text.CountVectorizer()
```

```
count_vectorizer.fit(df.Review)
```

```
CountVectorizer()
```

```
data_features = count_vectorizer.transform(df.Review)

density = (data_features.getnnz() * 100) / (data_features.shape[0]
*data_features.shape[1])
print("Density of the matrix: ", density)
```

```
Density of the matrix:  33.333333333333336
```

```
feature_counts = df['Review'].value_counts()
feature_counts
```

```
Review
this is a great movie                    1
i didnt like it                          1
amazing performance by the lead actor    1
Name: count, dtype: int64
```

```
features = vectorizer.get_feature_names_out() # Replace with the
variable that␣holds feature names
features_counts = np.sum(data_features.toarray(), axis=0)
features_counts_df = pd.DataFrame({'features': features,
'counts':features_counts})

count_of_single_occurrences
=len(features_counts_df[features_counts_df['counts'] == 1])
count_of_single_occurrences
```

```
13
```

```
count_vectorizer = CountVectorizer(max_features=10000)
feature_vector = count_vectorizer.fit_transform(df['Review'])
features = count_vectorizer.get_feature_names_out()
data_features = feature_vector.toarray()
features_counts = np.sum(data_features, axis=0)
feature_counts = pd.DataFrame({'features': features, 'counts':
features_counts})

top_features_counts = feature_counts.sort_values('counts',
ascending=False).head(15)

top_features_counts
```


```
{"summary":"{\n  \"name\": \"top_features_counts\",\n  \"rows\": 13,\n
\"fields\": [\n    {\n      \"column\": \"features\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 13,\n        \"samples\": [\n
\"the\",\n          \"movie\",\n          \"actor\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"counts\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 1,\n        \"max\": 1,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          1\n        ],\n        \"semantic_type\":
```

```
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"top_features_counts"}
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
english_stop_words = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
df['Review'][0:10]
```

```
0                    this is a great movie
1                         i didnt like it
2    amazing performance by the lead actor
Name: Review, dtype: object
```

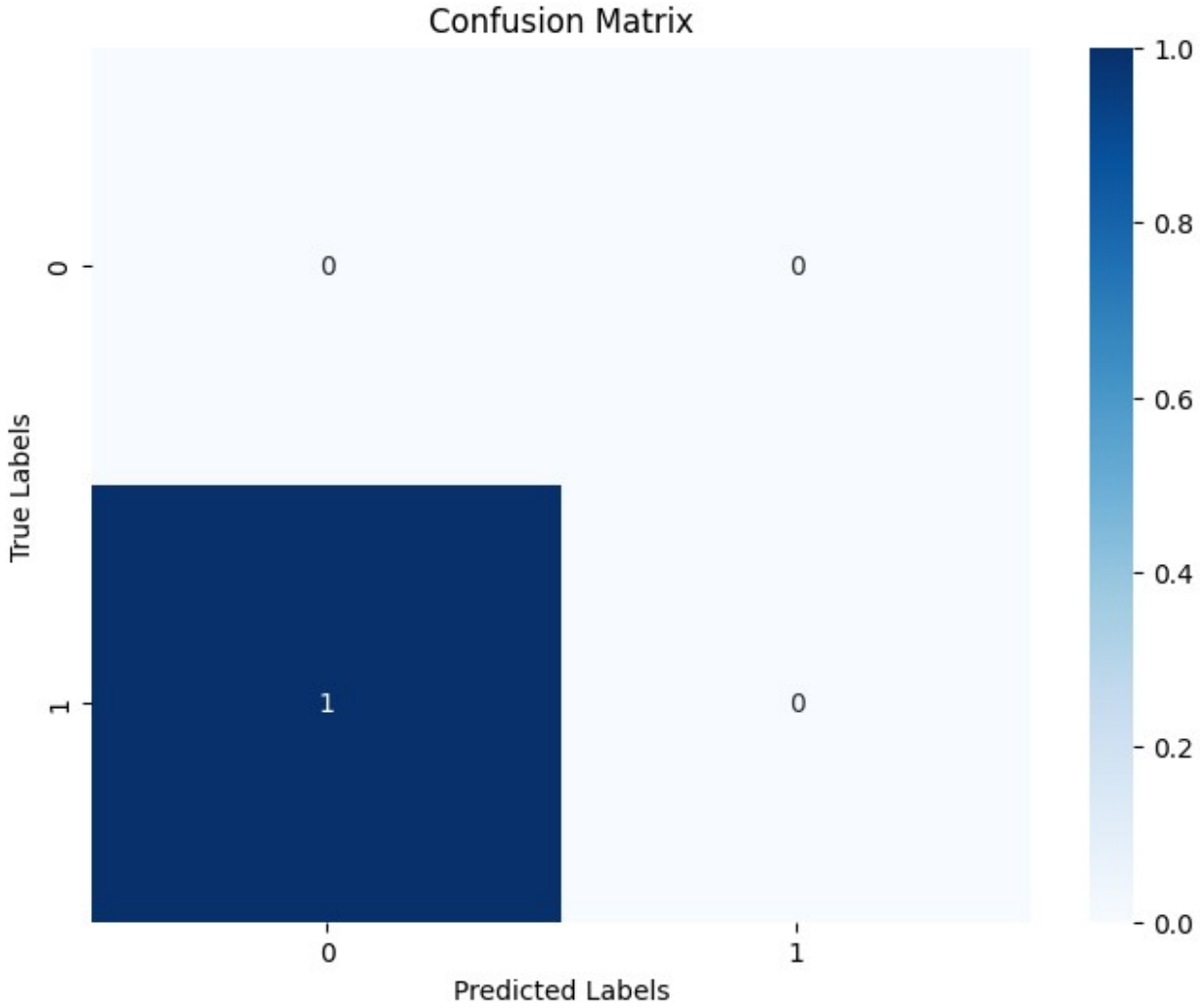
```
import pandas as pd
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Sample data for demonstration
data = {'Review': ["This is a great movie!", "I didn't like it.",
"Amazing performance by the lead actor."],
        'Sentiment': [1, 0, 1]}  # Assuming 1 is positive and 0 is
negative
df = pd.DataFrame(data)

# Convert reviews to lowercase
df['Review'] = df['Review'].apply(lambda x: x.lower())

# Remove punctuation from reviews
df['Review'] = df['Review'].apply(lambda x:
x.translate(str.maketrans('', '', string.punctuation)))

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df['Review'],
df['Sentiment'], test_size=0.2, random_state=42)

# Vectorize the text data
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Train an SVM model
model = SVC()
model.fit(X_train_vectorized, y_train)
```

```python
# Predict on the test set
y_pred = model.predict(X_test_vectorized)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print results
print(f"Accuracy: {accuracy}")
print(f"Classification Report:\n{report}")
```

```
Accuracy: 0.0
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       0.0
           1       0.00      0.00      0.00       1.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
```

```
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

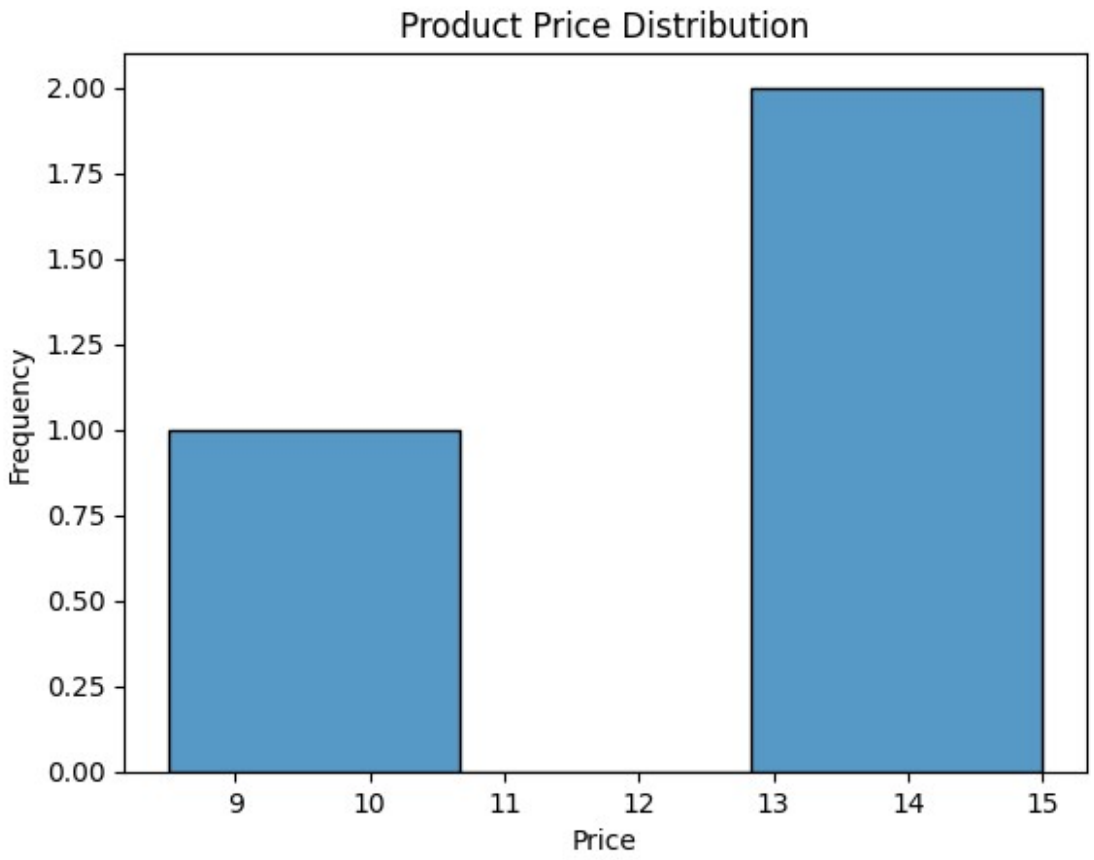


```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test =
train_test_split(df['Review'],df['Sentiment'], test_size=0.2,
random_state=42)
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
model = RandomForestClassifier()
model.fit(X_train_vectorized, y_train)
y_pred = model.predict(X_test_vectorized)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Accuracy: ", accuracy)
print("Classification Report:\n", report)
```

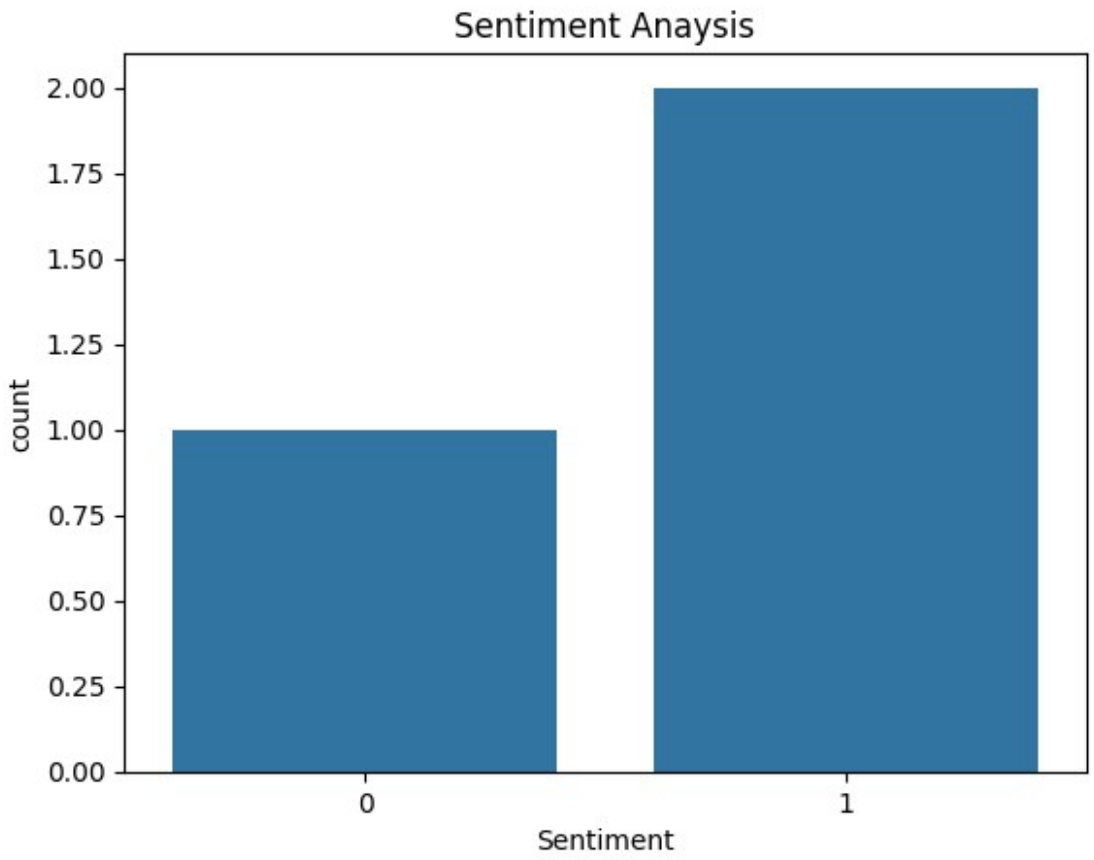
```
Accuracy:  0.0
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       0.0
           1       0.00      0.00      0.00       1.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Sample data for demonstration
data = {
    'Review': ["This is a great movie!", "I didn't like it.", "Amazing
performance by the lead actor."],
    'Sentiment': [1, 0, 1],  # Assuming 1 is positive and 0 is
negative
    'product_price': [12.99, 8.50, 15.00]  # Sample product prices
}
df = pd.DataFrame(data)

# Visualize the distribution of product prices
sns.histplot(df['product_price'])
plt.title('Product Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```




```
sns.countplot(data=df, x='Sentiment')
plt.title('Sentiment Anaysis')
plt.show()
```




```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 5))
plt.hist(features_counts_df['counts'], bins=50, range=(0, 5000))
plt.xlabel('Frequency of Words')
plt.ylabel('Density')
plt.show()
```