```
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/ML_TEAM5/1SV21CS010/TEACHABLE
MACHINE/data.csv")

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5842,\n  \"fields\":
[\n    {\n      \"column\": \"Sentence\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 5322,\n
\"samples\": [\n          \"It is now the leading private road
ambulance service company in Finland .\",\n          \"Finnish silicon
wafers manufacturer Okmetic Oyj said it swung to a net profit of 4.9
mln euro $ 6.3 mln in the first nine months of 2006 from a net loss of
1.8 mln euro $ 2.3 mln a year earlier .\",\n          \"$GILD  is
expanding its research facilities...keeping up with the pace of
innovation  https://t.co/uOE7FJ4LOP\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sentiment\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"positive\",\n          \"negative\",\n          \"neutral\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5842 entries, 0 to 5841
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Sentence   5842 non-null   object
 1   Sentiment  5842 non-null   object
dtypes: object(2)
memory usage: 91.4+ KB
```

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

print("Null values in the entire Data:")
print(null_values)
```

```
Null values in the entire Data:
Sentence     0
Sentiment    0
dtype: int64
```

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

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

```
Sentence     0
Sentiment    0
dtype: int64
```

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

```
import string
df['Sentiment']  =  df['Sentiment'].apply(lambda  x:  x.lower())
df['Sentiment']  =  df['Sentiment'].apply(lambda  x:
x.translate(str.maketrans('',  '',
string.punctuation)))
```

```
print(df.columns)
```

```
Index(['Sentence', 'Sentiment'], dtype='object')
```

```
df['Sentiment']
```

```
0       positive
1       negative
2       positive
3        neutral
4        neutral
          ...
5837    negative
5838     neutral
5839     neutral
5840     neutral
5841    positive
Name: Sentiment, Length: 5836, dtype: object
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
text_data = df['Sentiment']
vectorizer = CountVectorizer()
feature_matrix = vectorizer.fit_transform(text_data)
feature_names = vectorizer.get_feature_names_out()
feature_names
```

```
array(['negative', 'neutral', 'positive'], dtype=object)
```

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

```
count_vectorizer.fit(df.Sentiment)
```

```
CountVectorizer()
```

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

```
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['Sentiment'].value_counts()
feature_counts
```

```
Sentiment
neutral     3124
positive    1852
negative     860
Name: count, dtype: int64
```

```
features = vectorizer.get_feature_names_out() # Replace with the
variable thatholds 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]) # Removed
the extra space before the equal sign
count_of_single_occurrences
```

```
0
```

```
count_vectorizer = CountVectorizer(max_features=10000)
feature_vector = count_vectorizer.fit_transform(df['Sentiment'])
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\": 3,\n
\"fields\": [\n   {\n   \"column\": \"features\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n
\"neutral\",\n   \"positive\",\n   \"negative\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"counts\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 1134,\n
\"min\": 860,\n   \"max\": 3124,\n   \"num_unique_values\":
3,\n   \"samples\": [\n   3124,\n   1852,\n
860\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['Sentiment'][0:10]
```

```
0    positive
1    negative
2    positive
3     neutral
4     neutral
5    positive
6    negative
7    negative
8    positive
9     neutral
Name: Sentiment, dtype: object
```

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
X_train, X_test, y_train, y_test =
train_test_split(df['Sentiment'],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 = SVC()
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)
```

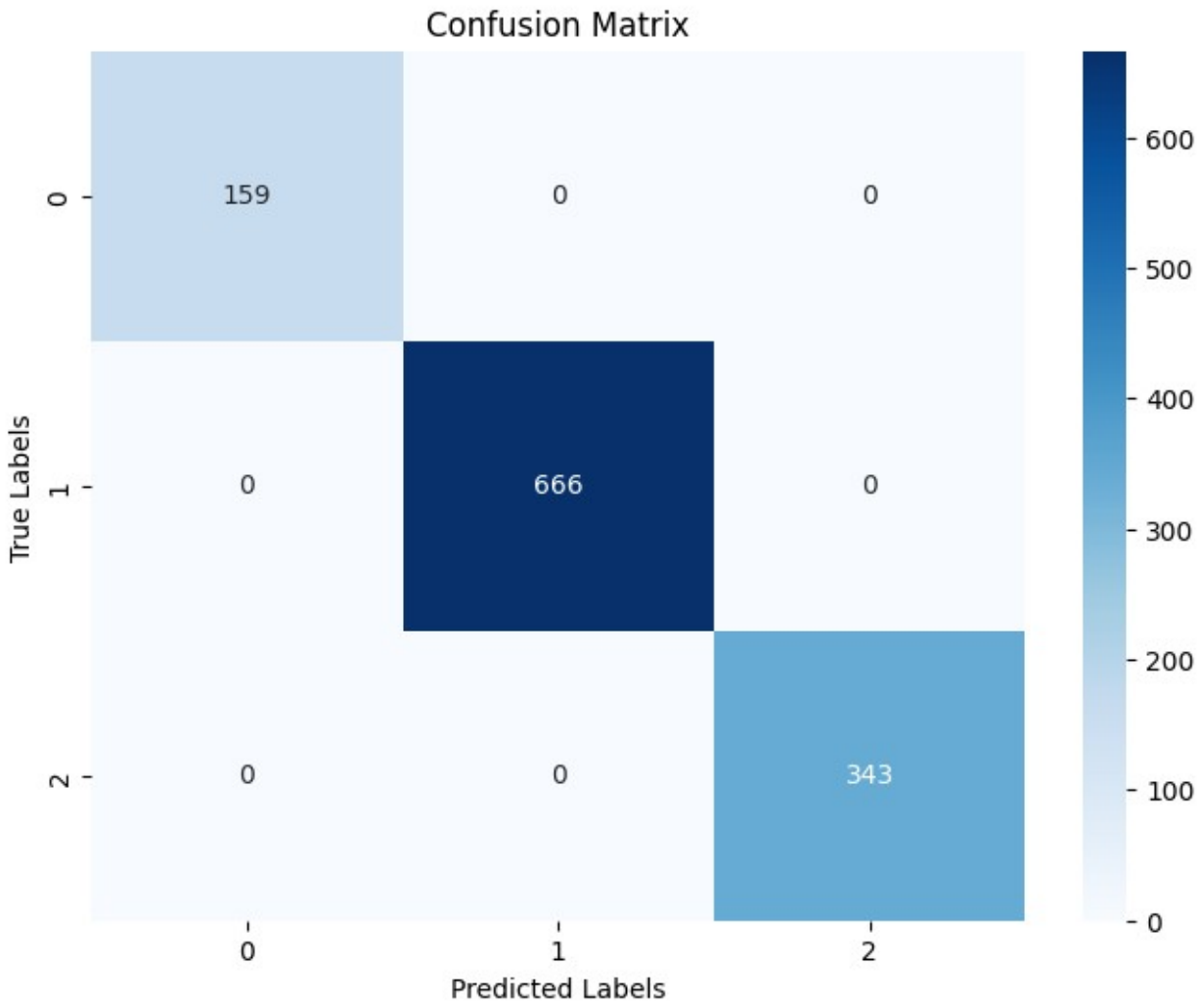
```
Accuracy:  1.0
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00       159
     neutral       1.00      1.00      1.00       666
    positive       1.00      1.00      1.00       343

    accuracy                           1.00      1168
   macro avg       1.00      1.00      1.00      1168
weighted avg       1.00      1.00      1.00      1168
```

```
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()
```



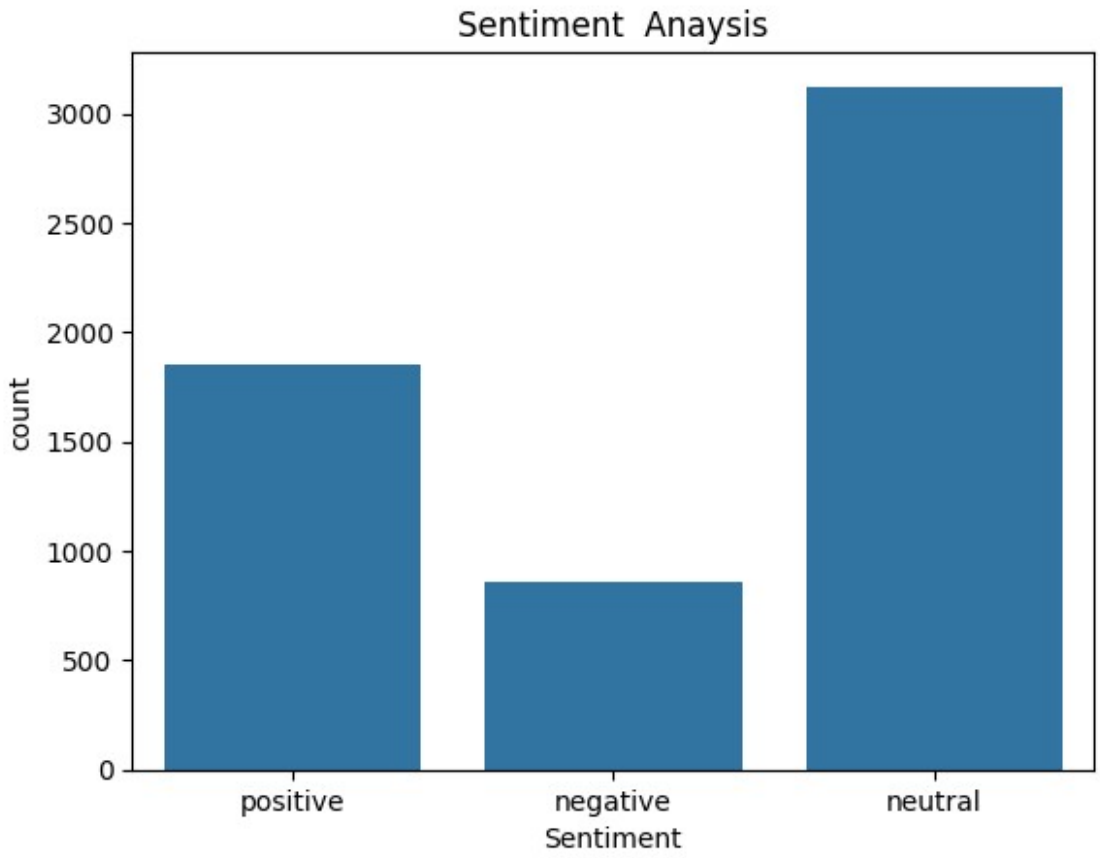

```
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(df['Sentiment'],
df['Sentiment'], test_size=0.2, random_state=42)
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
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:  1.0
Classification Report:
              precision    recall  f1-score   support
```

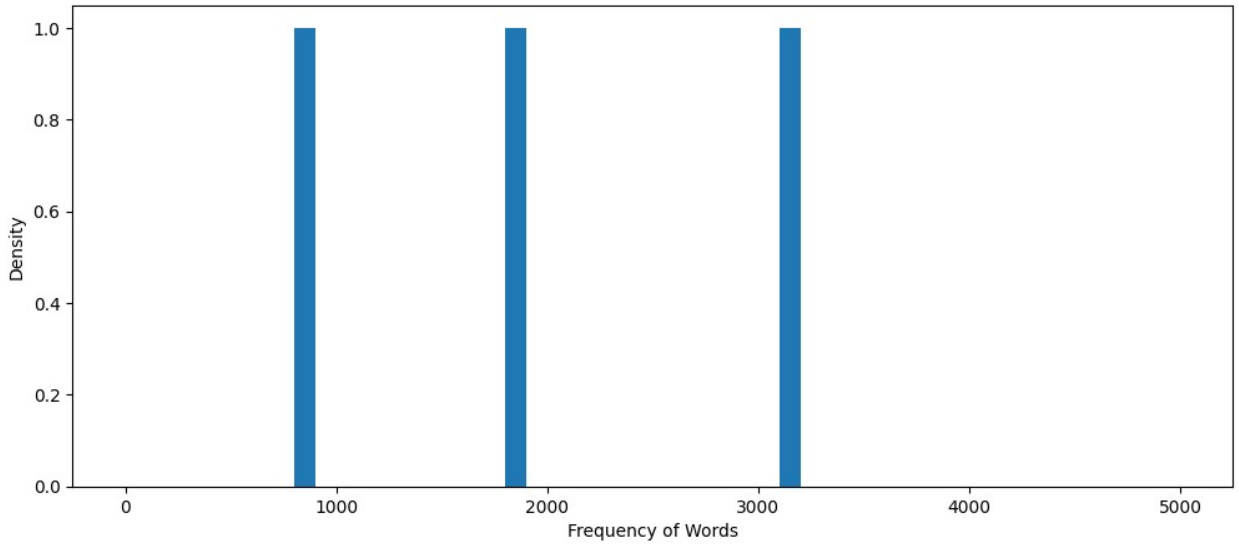
```
    negative       1.00      1.00      1.00       159
     neutral       1.00      1.00      1.00       666
    positive       1.00      1.00      1.00       343

    accuracy                           1.00      1168
   macro avg       1.00      1.00      1.00      1168
weighted avg       1.00      1.00      1.00      1168
```

```
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()
```


1.0
0.8
0.6
0.4
0.2
0.0
Density
0
1000
2000
3000
4000
5000
Frequency of Words