```
#!/usr/bin/env python
# coding: utf-8

# ## Importing Libraries

# In[3]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
get_ipython().run_line_magic('pip', 'install nltk')
import nltk
import warnings
get_ipython().run_line_magic('matplotlib', 'inline')


# ## Importing Dataset

# In[4]:


df = pd.read_csv(r"C:\Users\gauta\OneDrive\Documents\brainwave intern\Twitter Sentiments.csv")
df.head()


# ##  Preprocessing

# In[6]:


df.info()


# ## Data cleaning

# In[8]:


def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt


df.head()


# In[9]:


df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")


df.head()


# In[10]:


df['clean_tweet'] = df['clean_tweet'].str.replace("[^a-zA-Z#]", " ")
```

```
df.head()


# In[11]:


df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x.split() if
len(w)>3]))


df.head()


# In[12]:


tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())


tokenized_tweet.head()


# In[13]:


from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda sentence: [stemmer.stem(word) for word in
sentence])

tokenized_tweet.head()


# In[14]:


for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

df['clean_tweet'] = tokenized_tweet

df.head()


# ## Exploratory Data Analysis (EDA)

# In[15]:


get_ipython().system('pip install wordcloud')


# In[16]:


all_words = " ".join([sentence for sentence in df['clean_tweet']])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42,
max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

```
plt.show()


# In[32]:


all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==0]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100,
colormap='viridis').generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')

plt.show()


# In[18]:


all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42,
max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')

plt.show()


# In[19]:


def hashtag_extract(tweets):
    hashtags = []

    # loop words in the tweet
    for tweet in tweets:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)
    return hashtags


# In[20]:


ht_positive = hashtag_extract(df['clean_tweet'][df['label']==0])


ht_negative = hashtag_extract(df['clean_tweet'][df['label']==1])


ht_positive[:5]


# In[ ]:


ht_positive = sum(ht_positive, [])
ht_negative = sum(ht_negative, [])
```

```
# In[24]:


ht_positive[:5]


# In[25]:


freq = nltk.FreqDist(ht_positive)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})


d.head()


# In[31]:


d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count', palette="husl")

plt.show()


# In[33]:


freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})


d.head()


# In[39]:


d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count', palette="husl")
plt.show()


# ## Input split

# In[41]:


pip install scikit-learn


# In[42]:


from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000,
stop_words='english')
bow = bow_vectorizer.fit_transform(df['clean_tweet'])


# In[43]:
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bow, df['label'], random_state=42,
test_size=0.25)


# ## Training the model
#

# In[44]:


from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score


# In[45]:


model = LogisticRegression()
model.fit(x_train, y_train)


# In[46]:


pred = model.predict(x_test)
f1_score(y_test, pred)


# In[47]:


accuracy_score(y_test,pred)


# In[48]:


pred_prob = model.predict_proba(x_test)
pred = pred_prob[:, 1] >= 0.3
pred = pred.astype(np.int64)

f1_score(y_test, pred)


# In[49]:


accuracy_score(y_test,pred)


# In[50]:


pred_prob[0][1] >= 0.3


# In[ ]:
```