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

import pandas as pd
import pandas.util.testing as tm
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
get_ipython().system('pip install jupyterthemes')
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y axes clearly
# If you don't run this code line, you will notice that the xlabel and ylabel on any plot is black on black
and it will be hard to see them.


# Load the data
tweets_df = pd.read_csv('twitter.csv')



tweets_df



tweets_df.info()




tweets_df.describe()



tweets_df['tweet']



tweets_df = tweets_df.drop(['id'], axis=1)




sns.heatmap(tweets_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```

```
tweets_df.hist(bins = 30, figsize = (13,5), color = 'r')
```

```
sns.countplot(tweets_df['label'], label = "Count")
```

```
# Let's get the length of the messages
tweets_df['length'] = tweets_df['tweet'].apply(len)
```

```
tweets_df
```

```
tweets_df['length'].plot(bins=100, kind='hist')
```

```
tweets_df.describe()
```

```
# Let's see the shortest message
tweets_df[tweets_df['length'] == 11]['tweet'].iloc[0]
```

```
tweets_df[tweets_df['length'] == 84]['tweet'].iloc[0]
```

```
positive = tweets_df[tweets_df['label']==0]
```

```
positive
```

```
negative = tweets_df[tweets_df['label']==1]
```

```
negative
```

```
sentences = tweets_df['tweet'].tolist()
len(sentences)
```

```
sentences_as_one_string =" ".join(sentences)
```

```
sentences_as_one_string
```

```
get_ipython().system('pip install WordCloud')
from wordcloud import WordCloud

plt.figure(figsize=(20,20))
plt.imshow(WordCloud().generate(sentences_as_one_string))
```

```
negative_list = negative['tweet'].tolist()
negative_list
negative_sentences_as_one_string = " ".join(negative_list)
plt.figure(figsize=(20,20))
plt.imshow(WordCloud().generate(negative_sentences_as_one_string))
```

```
import string
string.punctuation
```

```
Test = 'Good morning beautiful people :)... I am having fun learning Machine learning and AI!!'
```

```
Test_punc_removed = [char for char in Test if char not in string.punctuation]
Test_punc_removed
```

```
# Join the characters again to form the string.
Test_punc_removed_join = ''.join(Test_punc_removed)
Test_punc_removed_join
```

```
import nltk # Natural Language tool kit

nltk.download('stopwords')
```

```
# You have to download stopwords Package to execute this command
from nltk.corpus import stopwords
stopwords.words('english')
```

```
Test_punc_removed_join
```

```
Test_punc_removed_join_clean = [word for word in Test_punc_removed_join.split() if word.lower()
not in stopwords.words('english')]
```

```
Test_punc_removed_join_clean # Only important (no so common) words are left
```

```
from sklearn.feature_extraction.text import CountVectorizer
sample_data = ['This is the first paper.','This document is the second paper.','And this is the third
one.','Is this the first paper?']

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sample_data)
```

```
print(vectorizer.get_feature_names())
```

```
print(X.toarray())
```

```
def message_cleaning(message):
    Test_punc_removed = [char for char in message if char not in string.punctuation]
    Test_punc_removed_join = ''.join(Test_punc_removed)
    Test_punc_removed_join_clean = [word for word in Test_punc_removed_join.split() if word.lower()
not in stopwords.words('english')]
    return Test_punc_removed_join_clean
```

```
tweets_df_clean = tweets_df['tweet'].apply(message_cleaning)
```

```
print(tweets_df_clean[5]) # show the cleaned up version
```

```
print(tweets_df['tweet'][5]) # show the original version
```

```
from sklearn.feature_extraction.text import CountVectorizer
# Define the cleaning pipeline we defined earlier
tweets_countvectorizer = CountVectorizer(analyzer = message_cleaning, dtype =
'uint8').fit_transform(tweets_df['tweet']).toarray()
```

```
print(vectorizer.get_feature_names())
```

```
tweets_countvectorizer.shape
```

```
X = tweets_countvectorizer
```

```
y = tweets_df['label']
```

```
X.shape
```

```
y.shape
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(tweets_countvectorizer, tweets_df['label'],
test_size=0.2)
```

```
X_train.dtype
```

```
from sklearn.naive_bayes import MultinomialNB

NB_classifier = MultinomialNB()
NB_classifier.fit(X_train, y_train)
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
y_predict_test = NB_classifier.predict(X_test)
cm = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm, annot=True)
```

```
print(classification_report(y_test, y_predict_test))
```