In [1]:
```python
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
from matplotlib import style
style.use('ggplot')

plt.style.use('fivethirtyeight')
```

In [2]:
```python
df = pd.read_csv(r"D:\900338-SonaSelvaraj(Week 4)\Twitter Sentiments.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [4]:
```python
df.tail()
```

Out[4]:

	id	label	tweet
31957	31958	0	ate @user isz that youuu?ð ð ð ð ð...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      31962 non-null  int64
 1   label   31962 non-null  int64
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

PREPROCESSING

In [6]:
```python
#remove pattern
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
```

In [7]:
```python
#remove twitter handles(@user)
df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")
```

In [8]:
```python
df.head()
```

Out[8]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide: society now #motivation

In [9]:
```python
#remove special characters, numbers and punctuation
df['clean_tweet'] = df['clean_tweet'].str.replace("[^a-zA-Z#]"," ")
df.head()
```

Out[9]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide society now #motivation

In [10]:
```python
# remove short words
df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
df.head()
```

Out[10]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model love take with time
4	5	0	factsguide: society now #motivation	factsguide society #motivation

In [11]:
```python
# individual words considered as tokens
tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

Out[11]:
```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                              [bihday, your, majesty]
3                     [#model, love, take, with, time]
4                   [factsguide, society, #motivation]
Name: clean_tweet, dtype: object
```

In [12]:
```python
# stem the words
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()
```

Out[12]:
```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                              [bihday, your, majesti]
3                     [#model, love, take, with, time]
4                         [factsguid, societi, #motiv]
Name: clean_tweet, dtype: object
```

In [13]:
```python
# combine words into single sentence
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

df['clean_tweet'] = tokenized_tweet
df.head()
```

Out[13]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they offer wheelchair ...
2	3	0	bihday your majesty	bihday your majesti
3	4	0	#model i love u take with u all the time in ...	#model love take with time
4	5	0	factsguide: society now #motivation	factsguid societi #motiv

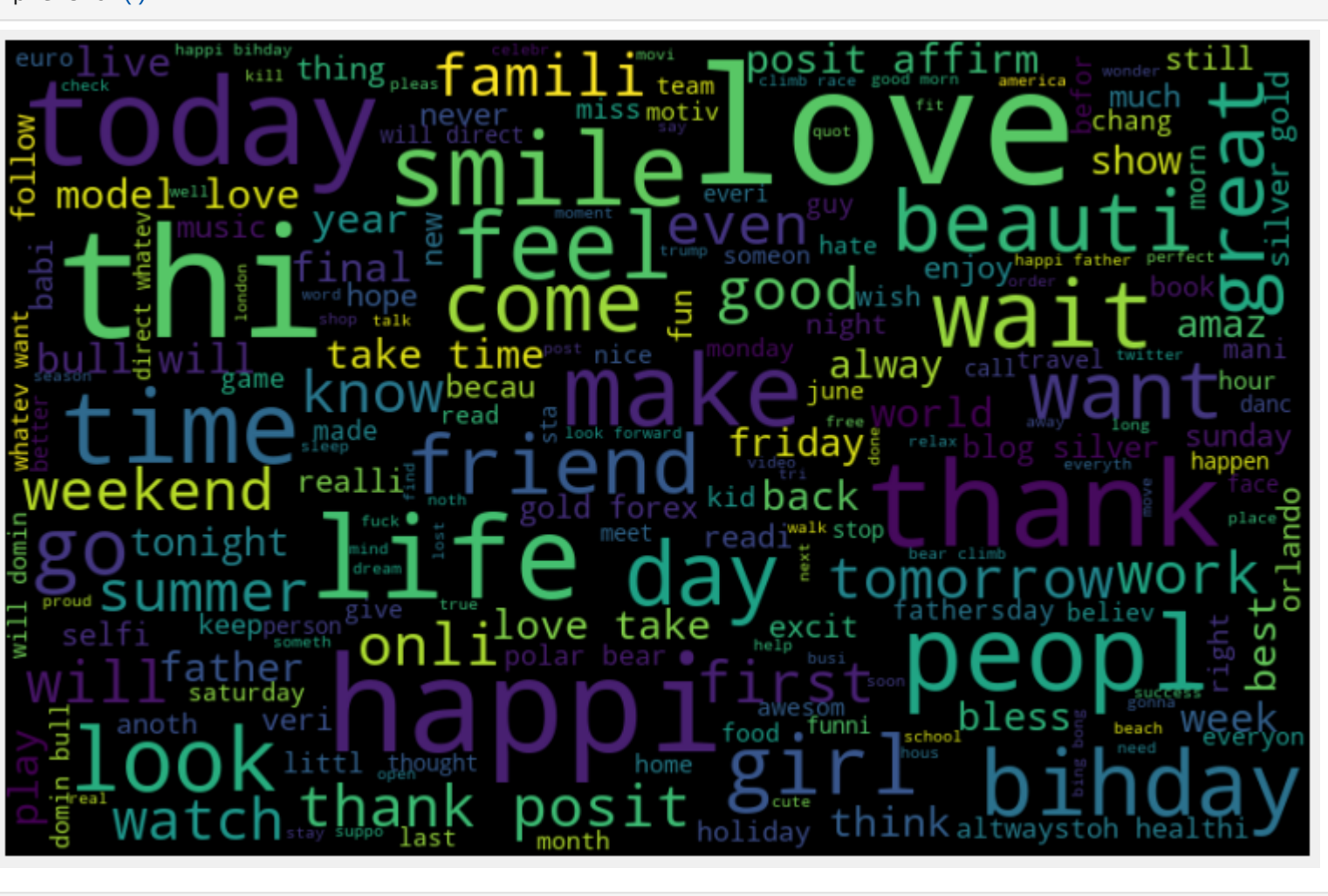
Exploratory Data Analysis

In [14]:
```python
pip install wordcloud
```

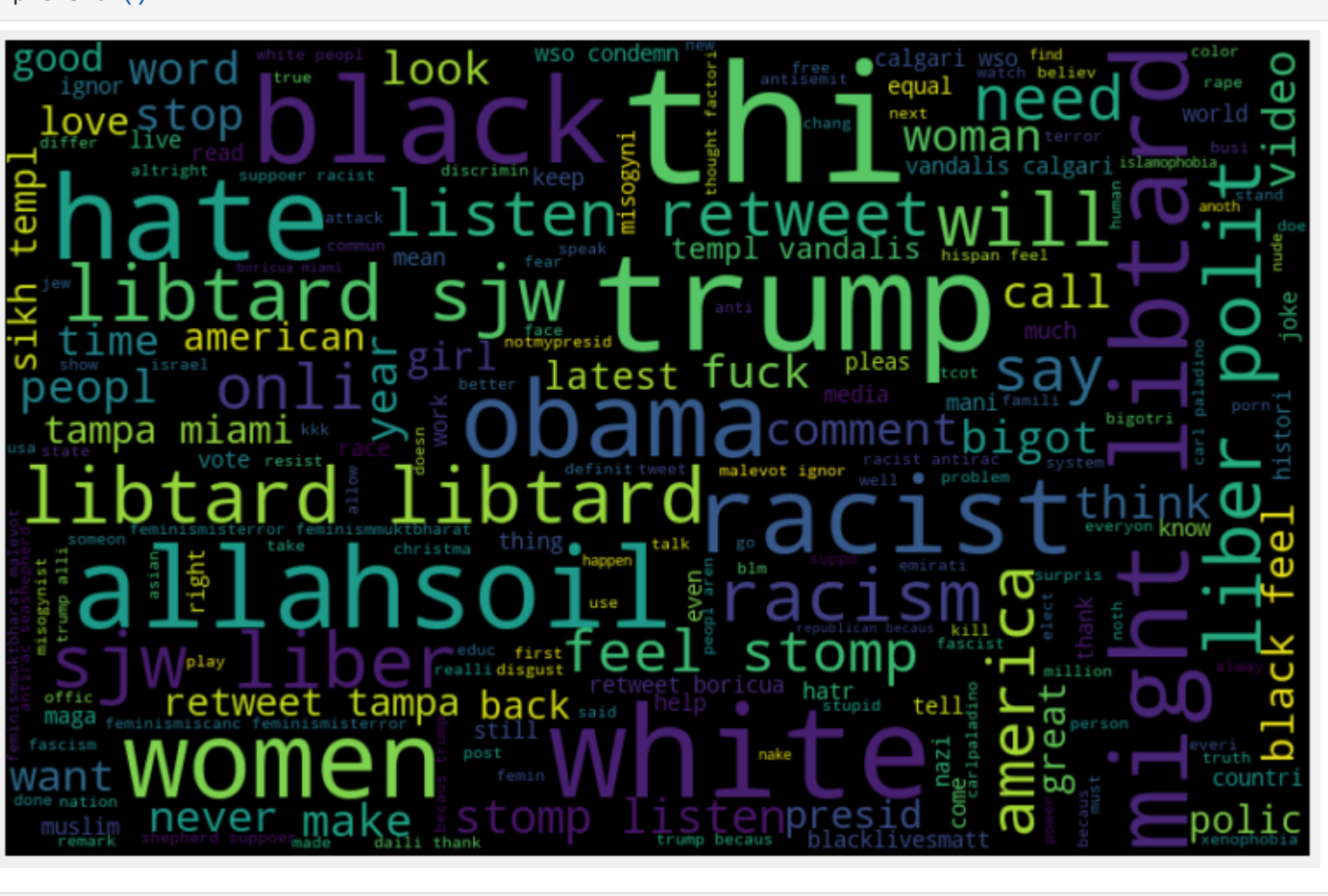
```
Requirement already satisfied: wordcloud in c:\users\sona\anaconda3\lib\site-packages (1.6.0)Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pillow in c:\users\sona\anaconda3\lib\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\sona\anaconda3\lib\site-packages (from wordcloud) (1.19.2)
Requirement already satisfied: matplotlib in c:\users\sona\anaconda3\lib\site-packages (from wordcloud) (3.3.2)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\sona\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2022.6.15)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\sona\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.3.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\sona\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\sona\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\sona\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: six in c:\users\sona\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->wordcloud) (1.15.0)
```

In [15]:
```python
# visualize the frequent words
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 [16]:
```python
# frequent words visualization for +ve
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).generate(all_words)

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



In [17]:
```python
# frequent words visualization for -ve
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 [18]:
```python
# extract the hashtag
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 [19]:
```python
# extract hashtags from non-racist/sexist tweets
ht_positive = hashtag_extract(df['clean_tweet'][df['label']==0])

# extract hashtags from racist/sexist tweets
ht_negative = hashtag_extract(df['clean_tweet'][df['label']==1])
```

In [20]:
```python
ht_positive[:5]
```

Out[20]:
```
[['run'], ['lyft', 'disapoint', 'getthank'], [], ['model'], ['motiv']]
```

In [21]:
```python
ht_negative[:5]
```

Out[21]:
```
[['cnn', 'michigan', 'tcot'],
 ['australia',
  'opkillingbay',
  'seashepherd',
  'helpcovedolphin',
  'thecov',
  'helpcovedolphin'],
 [],
 [],
 ['neverump', 'xenophobia']]
```

In [22]:
```python
# unnest list
ht_positive = sum(ht_positive, [])
ht_negative = sum(ht_negative, [])
```

In [23]:
```python
ht_negative[:5]
```

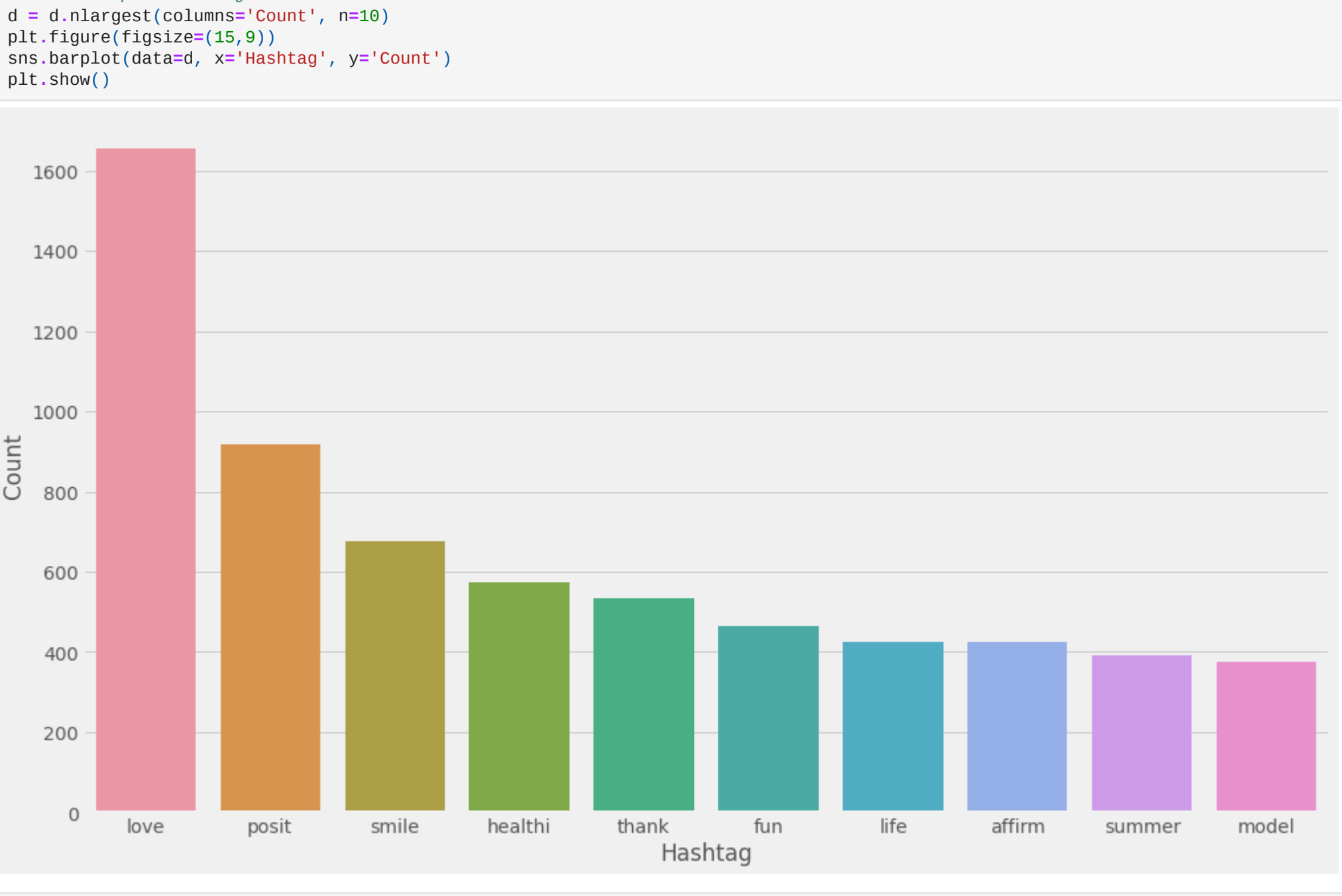
Out[23]:
```
['cnn', 'michigan', 'tcot', 'australia', 'opkillingbay']
```

In [24]:
```python
freq = nltk.FreqDist(ht_positive)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

Out[24]:

	Hashtag	Count
0	run	72
1	lyft	2
2	disapoint	1
3	getthank	2
4	model	375

In [25]:
```python
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```

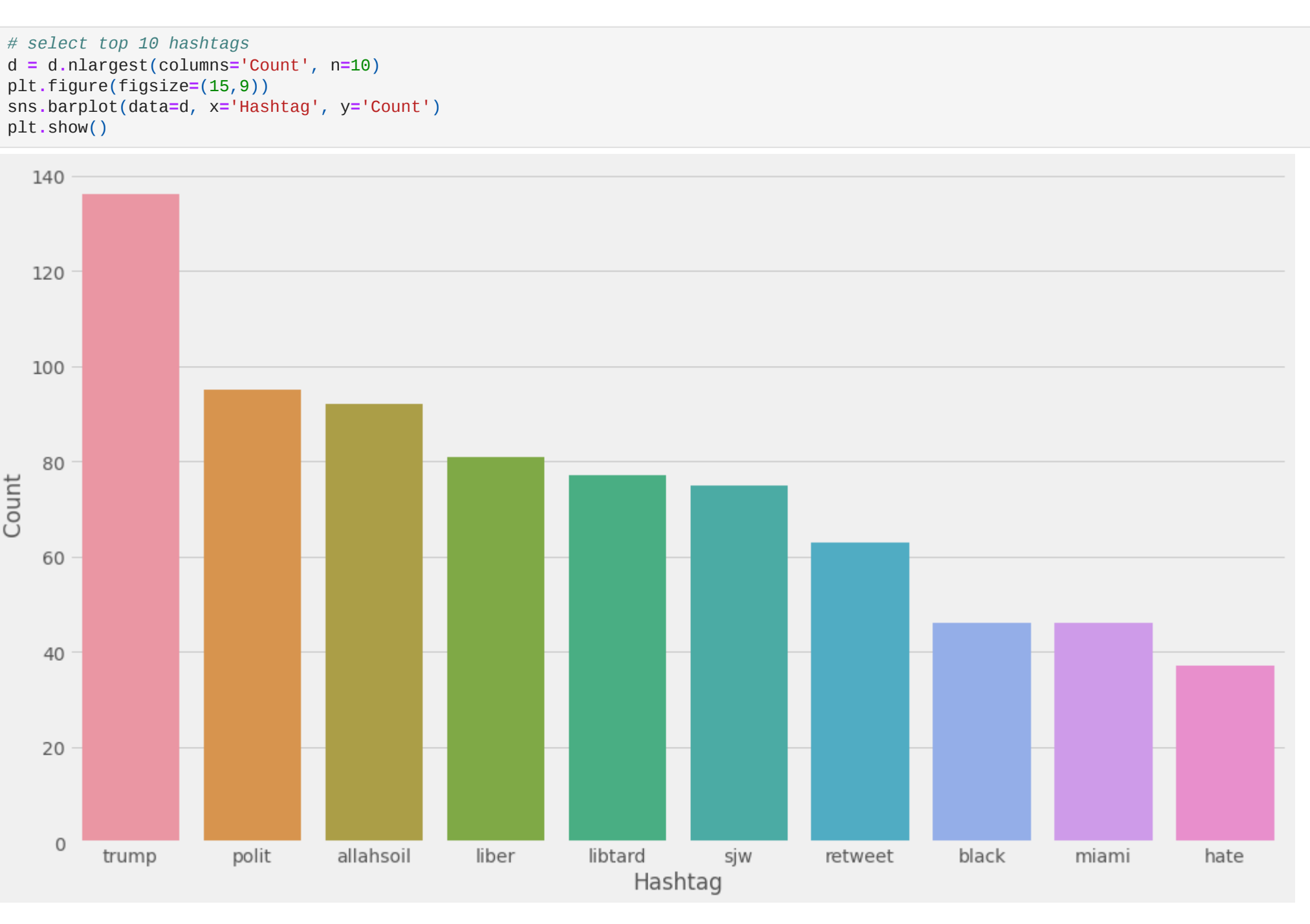


In [26]:
```python
freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

Out[26]:

	Hashtag	Count
0	cnn	10
1	michigan	2
2	tcot	14
3	australia	6
4	opkillingbay	5

In [27]:
```python
# select top 10 hashtags
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Hashtag', y='Count')
plt.show()
```



Input Split

In [28]:
```python
# feature extraction
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 [29]:
```python
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)
```

Model Training

SVM CLASSIFICATION

In [30]:
```python
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(bow, df['label'], random_state=42, test_size=0.25)
```

In [31]:
```python
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [36]:
```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
from sklearn.metrics import confusion_matrix,classification_report,ConfusionMatrixDisplay
from sklearn.metrics import f1_score

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9473157301964711
```

In [37]:
```python
# training
model = svm.SVC(kernel='linear', C = 1.0)
model.fit(x_train, y_train)
```

Out[37]:
```
SVC(kernel='linear')
```

In [38]:
```python
# testing
pred = model.predict(x_test)
f1_score(y_test, pred)
```

Out[38]:
```
0.4994054696789536
```

In [39]:
```python
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

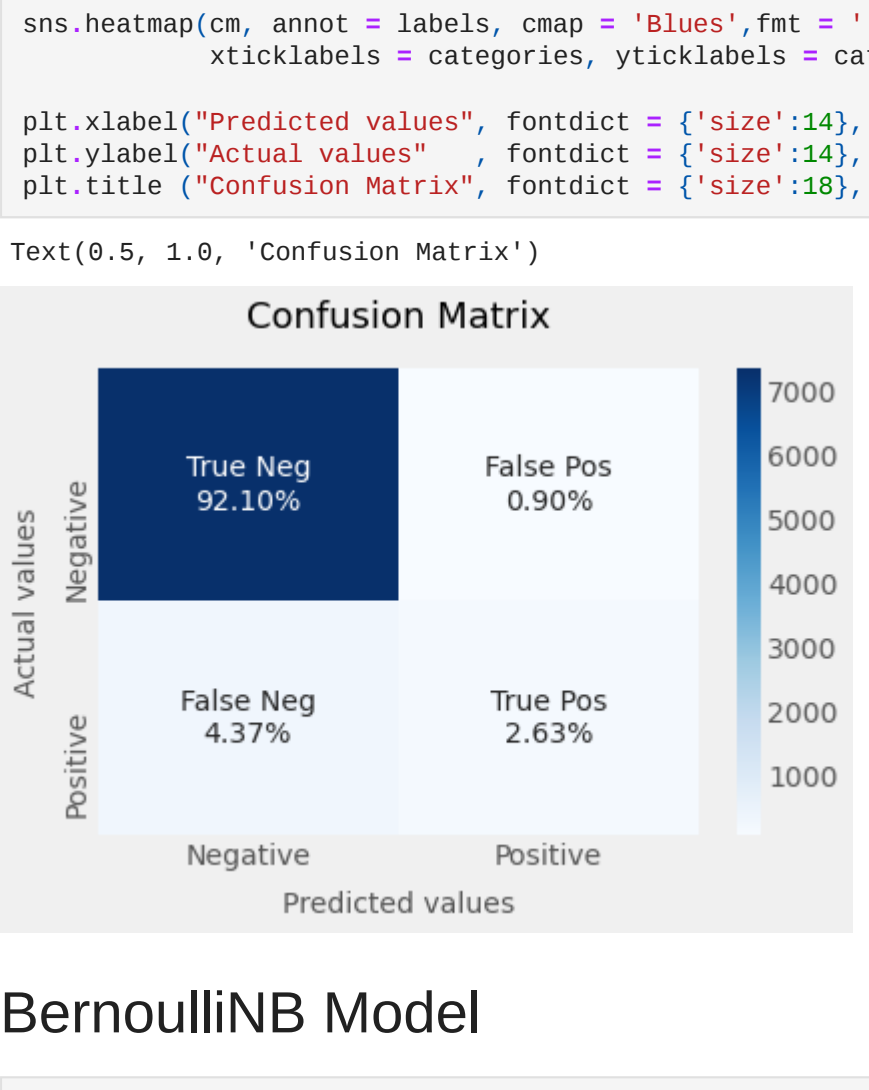
```
[[7360   72]
 [ 349  210]]


              precision    recall  f1-score   support

           0       0.95      0.99      0.97      7432
           1       0.74      0.38      0.50       559

    accuracy                           0.95      7991
   macro avg       0.85      0.68      0.74      7991
weighted avg       0.94      0.95      0.94      7991
```

In [40]:
```python
pip install reportlab
```

```
Requirement already satisfied: reportlab in c:\users\sona\anaconda3\lib\site-packages (3.6.11)
Requirement already satisfied: pillow>=9.0.0 in c:\users\sona\anaconda3\lib\site-packages (from reportlab) (9.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [44]:
```python
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm)

categories  = ['Negative','Positive']
group_names = ['True Neg','False Pos', 'False Neg','True Pos']
group_percentages = ['{0:.2%}'.format(value) for value in cm.flatten() / np.sum(cm)]

labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_percentages)]
labels = np.asarray(labels).reshape(2,2)

sns.heatmap(cm, annot = labels, cmap = 'Blues',fmt = '',
            xticklabels = categories, yticklabels = categories)

plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
plt.ylabel("Actual values"   , fontdict = {'size':14}, labelpad = 10)
plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Out[44]:
```
Text(0.5, 1.0, 'Confusion Matrix')
```

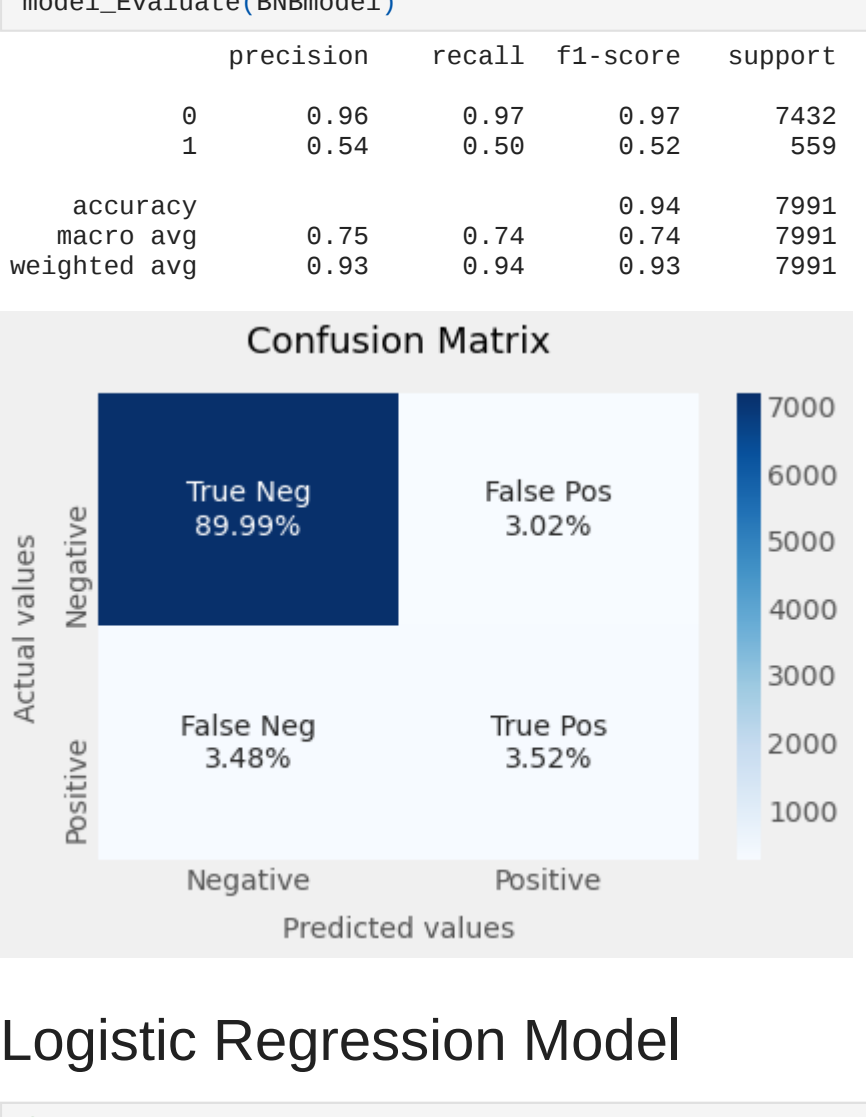


BernoulliNB Model

In [45]:
```python
def model_Evaluate(model):

    # Predict values for Test dataset
    y_pred = model.predict(X_test)


    # Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))

    # Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)

    categories  = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / np.sum(cf_matrix)]

    labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)

    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
                xticklabels = categories, yticklabels = categories)

    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
    plt.ylabel("Actual values"   , fontdict = {'size':14}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

In [46]:
```python
from sklearn.naive_bayes import BernoulliNB

BNBmodel = BernoulliNB(alpha = 2)
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      7432
           1       0.54      0.50      0.52       559

    accuracy                           0.94      7991
   macro avg       0.75      0.74      0.74      7991
weighted avg       0.93      0.94      0.93      7991
```

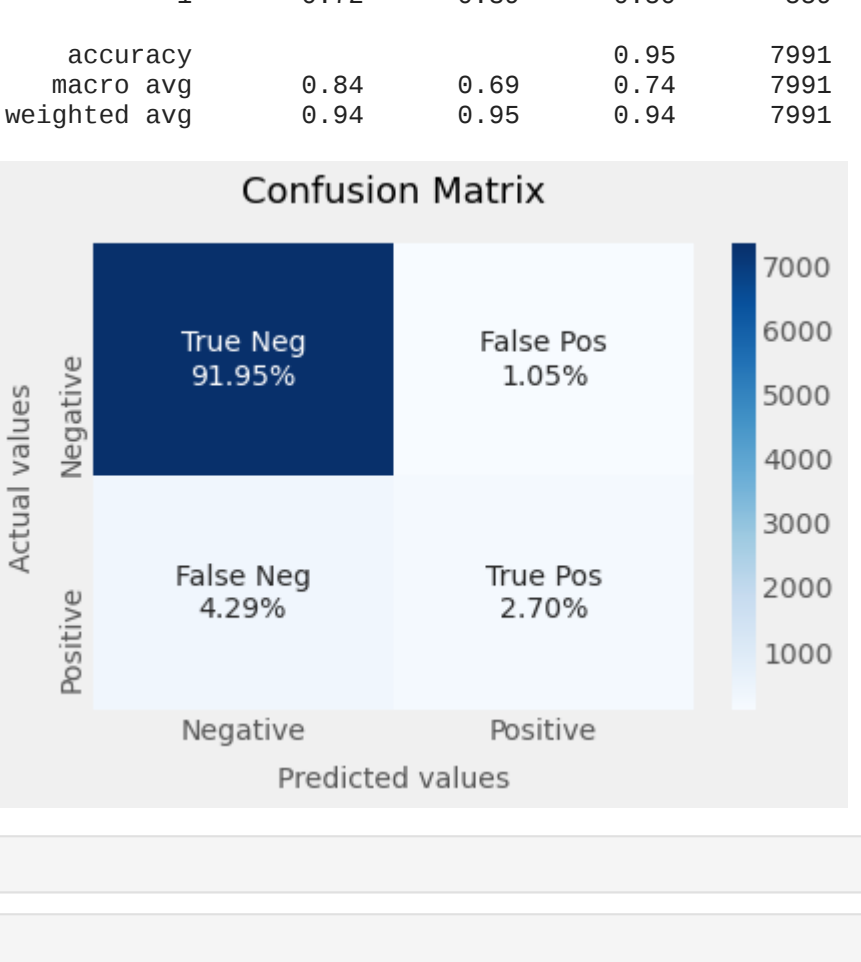


Logistic Regression Model

In [49]:
```python
from sklearn.linear_model import LogisticRegression
```

In [50]:
```python
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      7432
           1       0.72      0.39      0.50       559

    accuracy                           0.95      7991
   macro avg       0.84      0.69      0.74      7991
weighted avg       0.94      0.95      0.94      7991
```



In [ ]:

In [ ]: