NLP

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
```

In [4]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\train_tweet.csv")
df.head()
```

Out[4]:

	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 [5]:
```
df.isnull().sum()
```

Out[5]:
```
id       0
label    0
tweet    0
dtype: int64
```

In [6]:
```
df.drop('id',axis=1,inplace=True)
```

In [7]:
```
def cleantweet(text):
  text=re.sub(r'@[A-Za-z0-9]+','',text)
  text=re.sub(r'#','',text)
  text=re.sub(r'RT[\s]+','',text)
  text=re.sub(r'https?:\/\/\S+','',text)

  return text
```

In [8]:
```
df['tweet']=df['tweet'].apply(cleantweet)
```

In [9]:
```
df
```

Out[9]:

	label	tweet
0	0	when a father is dysfunctional and is so sel...
1	0	thanks for lyft credit i can't use cause the...
2	0	bihday your majesty
3	0	model i love u take with u all the time in u...
4	0	factsguide: society now motivation
...	...	...
31957	0	ate isz that youuu?ð ð ð ð ð ð ð ...

	label	tweet
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	sikh temple vandalised in in calgary, wso con...
31961	0	thank you for you follow

31962 rows × 2 columns

In [10]:
```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[10]:
```
True
```

In [11]:
```
STOPWORDS = set(stopwords.words('english'))
def clean_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
df['tweet'] = df['tweet'].apply(clean_stopwords)
df['tweet'].head()
```

Out[11]:
```
0    father dysfunctional selfish drags kids dysfun...
1    thanks lyft credit can't use cause offer wheel...
2                                     bihday majesty
3    model love u take u time urð ±!!! ð ð ð ...
4                        factsguide: society motivation
Name: tweet, dtype: object
```

In [12]:
```
import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
df['tweet']= df['tweet'].apply(lambda x: stemming_on_text(x))
df.head()
```

Out[12]:

	label	tweet
0	0	father dysfunctional selfish drags kids dysfun...
1	0	thanks lyft credit can't use cause offer wheel...
2	0	bihday majesty
3	0	model love u take u time urð ±!!! ð ð ð ...
4	0	factsguide: society motivation

In [13]:
```
lm = nltk.WordNetLemmatizer()
nltk.download('wordnet')
nltk.download('omw-1.4')
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
```

```
    return data
df['tweet'] = df['tweet'].apply(lambda x: lemmatizer_on_text(x))
df.head()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
```

Out[13]:

	label	tweet
0	0	father dysfunctional selfish drags kids dysfun...
1	0	thanks lyft credit can't use cause offer wheel...
2	0	bihday majesty
3	0	model love u take u time urð ±!!! ð ð ð ...
4	0	factsguide: society motivation

In [14]:

```
from nltk.tokenize import  word_tokenize
nltk.download('punkt')
def tweet_tokenizer(tweet):
  return word_tokenize(tweet)

df["tweet"]=df["tweet"].apply(tweet_tokenizer)
df
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[14]:

	label	tweet
0	0	[father, dysfunctional, selfish, drags, kids, ...
1	0	[thanks, lyft, credit, ca, n't, use, cause, of...
2	0	[bihday, majesty]
3	0	[model, love, u, take, u, time, urð ±, !, !, ...
4	0	[factsguide, :, society, motivation]
...	...	...
31957	0	[ate, isz, youuu, ?, ð ð ð ð ð ð ð...
31958	0	[see, nina, turner, airwaves, trying, wrap, ma...
31959	0	[listening, sad, songs, monday, morning, otw, ...
31960	1	[sikh, temple, vandalised, calgary, ,, wso, co...
31961	0	[thank, follow]

31962 rows × 2 columns

In [15]:

```
X=df.tweet
y=df.label
```

In [17]:

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.5, random_state =
```

In [18]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train.apply(lambda x: ' '.join(x)))
```

Out[18]:
```
TfidfVectorizer(max_features=500000, ngram_range=(1, 2))
```

In [19]:
```
X_train = vectoriser.transform(X_train.apply(lambda x: ' '.join(x)))
X_test  = vectoriser.transform(X_test.apply(lambda x: ' '.join(x)))
```

In [20]:
```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression()
lr.fit(X_train,y_train)
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
y_pred1 = lr.predict(X_test)
```

In [22]:
```
from sklearn.metrics import confusion_matrix,classification_report
cf_matrix = confusion_matrix(y_test, y_pred1)
```

In [23]:
```
print(classification_report(y_test, y_pred1))
```

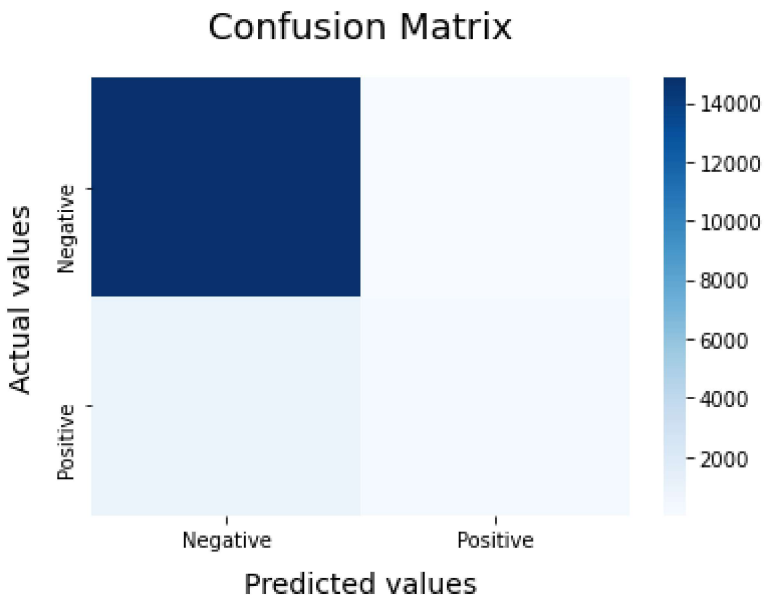
```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     14852
           1       0.91      0.17      0.29      1129

    accuracy                           0.94     15981
   macro avg       0.92      0.58      0.63     15981
weighted avg       0.94      0.94      0.92     15981
```

In [24]:
```
categories = ['Negative','Positive']
sns.heatmap(cf_matrix, cmap = 'Blues',fmt = '',xticklabels = categories, yticklabels =
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[24]:
```
Text(0.5, 1.0, 'Confusion Matrix')
```


Confusion Matrix
Actual values
Negative
Positive
Negative
Positive
Predicted values
14000
12000
10000
8000
6000
4000
2000

In [ ]: