Data Collection

Importation of libraries

In [1]:
```
import csv
from bs4 import BeautifulSoup
import requests

import pandas as pd

import time
time.sleep(2)

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
urls = []

url1 = 'https://www.imdb.com/title/tt0117951/reviews/?ref_=tt_ql_urv'
url2 = 'https://www.imdb.com/title/tt2235695/reviews/?ref_=tt_ql_urv'
url3 = 'https://www.imdb.com/title/tt8108200/reviews/?ref_=tt_ql_urv'
url4 = 'https://www.imdb.com/title/tt0347304/reviews/?ref_=tt_ql_urv'
url5 = 'https://www.imdb.com/title/tt0033467/reviews/?ref_=tt_ql_urv'
url6 = 'https://www.imdb.com/title/tt2574698/reviews/?ref_=tt_ql_urv'
url7 = 'https://www.imdb.com/title/tt1517561/reviews/?ref_=tt_ql_urv'
```

In [3]:
```
urls.append(url1)
urls.append(url2)
urls.append(url3)
urls.append(url4)
urls.append(url5)
urls.append(url6)
urls.append(url7)
```

In [4]:
```
content = []

for url in urls:
    page = requests.get(url, timeout=2.50)
    page_content = page.content
    soup = BeautifulSoup(page_content, 'html.parser')
    content.append(soup.find_all('div', class_= 'review-container'))
```

In [5]:
```
#print(content)
```

In [6]:
```
movie = pd.DataFrame(columns=['Review','Rating'])
```

In [7]:
```
review = []
rating = []
count = 0
for cc in content:
    for c in cc:
        count+= 1

        print('\nMovie review ', count)
        #Get review.
        str = c.find_all('a', attrs={'class':'title'})
        rReview =''
        for s in str:
            #print('Review is: ',s.get_text())
            rReview = s.get_text()

        #Get rating.
        ratings = c.find_all('span', attrs={'class':''})
        rVal = []
        for r in ratings:
            str1 = r.get_text().strip()
            rVal.append(str1)

        val = rVal[0]

        if(len(val) > 2):
            continue
        else:
            review.append(rReview)
            rating.append(val)

            print('Review: ', rReview)
            print('Rating: ',val)

movie['Review'] = review
movie['Rating'] = rating
```

```
Rating:  5

Movie review  29
Review:   Pretty good, perhaps missing a little something.

Rating:  6

Movie review  30
Review:   No charm or suspense

Rating:  2

Movie review  31
Review:   I really really wanted to love this movie

Rating:  5

Movie review  32
Review:   I was happily surprised!
```

In [8]:
```
movie.head()
```

Out[8]:

	Review	Rating
0	Now I See Why\n	9
1	Choose life\n	8
2	One Of THE Defining Movies Of The 90s And A M...	10
3	The Wild & Crazy Indie Smash Hit that Started...	10
4	In your face cinema\n	7

In [9]:
```
movie.shape
```

Out[9]:
```
(154, 2)
```

In [10]:
```
movie.to_csv('PreciousAdaugoReginald-2325671-Reviews.csv', index=False)
```

Test Processing and Analysis

In [11]:
```
import string
import re

#import nltk
#nltk.download()
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split
```

In [12]:
```
textFeatures = movie['Review'].copy()
textFeatures.shape
```

Out[12]: (154,)

In [13]:
```
#Preparing text for Wordcloud
text = []
for t in textFeatures:
  text.append(t)

all_text = ', '.join(t for t in text)
#print(all_text)
print(len(all_text))
```

```
6406
```

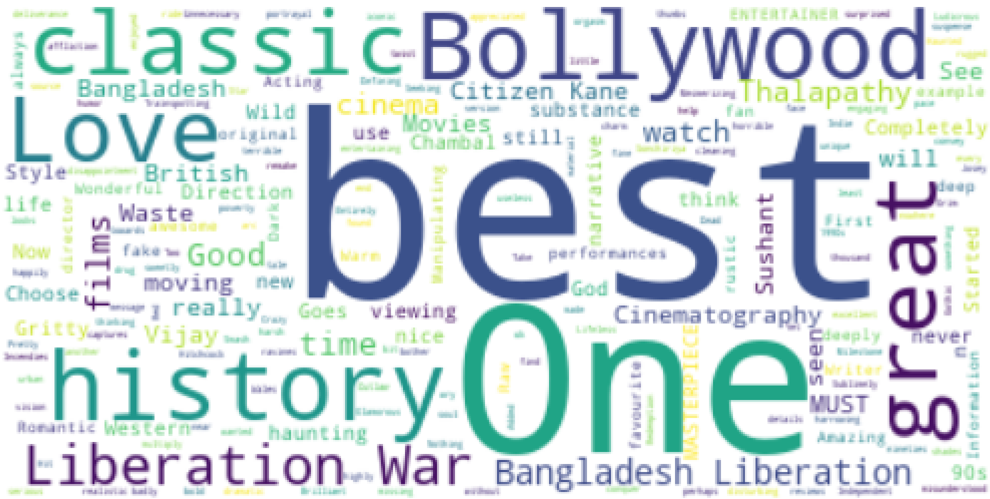
In [14]:
```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [15]:
```
# Create stopword list
stopwords = set(STOPWORDS)
stopwords.update(["br", "im", "thats", "film", "movie"]) #"im","lol","Xa","film"])

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(all_text)

# Display the image
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
#save the generated image to a file
#wordcloud.to_file("wordcloud_cb_all.png")
```



Sentiment Identification with the use of VADER

In [16]:
```
import nltk
nltk.download('vader_lexicon')

from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\princ\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [17]:
```python
sid = SentimentIntensityAnalyzer()
c = 0
for t in text:
    c+=1
    print(c, t)
    ss = sid.polarity_scores(t)
    print(ss)

    if(ss['compound'] >= 0.05):
        print('positive')

    elif(ss['compound'] <= -0.05):
        print('negative')

    else:
        print('neutral')
    print('\n')
```

```
positive


15  With God's help I'll conquer this terrible affliction.

{'neg': 0.294, 'neu': 0.487, 'pos': 0.219, 'compound': -0.2323}
negative


16  One of the best and most iconic British films ever made. To 1996 what Star Wars is to 1977.

{'neg': 0.145, 'neu': 0.685, 'pos': 0.169, 'compound': 0.1531}
positive


17  Choose the best British film of the nineties

{'neg': 0.0, 'neu': 0.625, 'pos': 0.375, 'compound': 0.6369}
positive
```

Sentimental classification using Machine Learning

Preparation of 'Truth Set'

In [18]:
```
label = []

for r in movie['Rating']:
    r = int(r)
    if (r>5):
        label.append('1') #Positive
    elif(r<5):
        label.append('-1') #Negative
    elif(r==5):
        label.append('0') #Netural

movie['class-label'] = label
```

In [19]:
```
movie['class-label'].value_counts()
```

Out[19]:
```
1     110
-1     37
0       7
Name: class-label, dtype: int64
```

In [20]:
```
movie = movie[movie['class-label']!='0']
```

In [21]:
```
movie['class-label'].value_counts()
```

Out[21]:
```
1     110
-1     37
Name: class-label, dtype: int64
```

In [22]:
```
textFeatures = movie['Review'].copy()
textFeatures.shape
```

Out[22]:
```
(147,)
```

In [23]:
```
import nltk
nltk.download('punkt')

# Stemming using TextBlob library for stemming
from textblob import TextBlob
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\princ\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [24]:
```
def textblob_tokenizer(input_str):
    blob = TextBlob(input_str.lower())
    tokens = blob.words
    words = [token.stem() for token in tokens]
    return words
```

In [25]:
```
#Toy example:
print(textblob_tokenizer('Q: studed studing!!! I miss uuuu! It&#039;s'))
```

```
['q', 'stude', 'stude', 'i', 'miss', 'uuuu', 'it', '039', 's']
```

In [26]:
```
#countvectorizer convers each review into a vector based on the word count.
countvectorizer = CountVectorizer(analyzer= 'word', stop_words= 'english',
                                  tokenizer=textblob_tokenizer)

#convers text into a vector based on tf-idf weighting scheme.
tfidfvectorizer = TfidfVectorizer(analyzer= 'word', stop_words= 'english',
                                  tokenizer=textblob_tokenizer)
```

In [27]:
```
textFeatures
```

Out[27]:
```
0                                        Now I See Why\n
1                                          Choose life\n
2        One Of THE Defining Movies Of The 90s And A M...
3        The Wild & Crazy Indie Smash Hit that Started...
4                                  In your face cinema\n
                              ...
149                        My all time favourite movie\n
150                                         Favourite❤\n
151              Best love movie and a good film forever\n
152                                     The best movie\n
153                             Epic movie for 90s kids\n
Name: Review, Length: 147, dtype: object
```

In [28]:
```
count_matrix = countvectorizer.fit_transform(textFeatures)
tfidf_matrix = tfidfvectorizer.fit_transform(textFeatures)
```

In [29]:
```
#print(tfidf_matrix) #print elements of the matrix.
```

In [30]:
```
print(tfidf_matrix.shape)
print(count_matrix.shape)
```

```
(147, 344)
(147, 344)
```

Building of ML models

In [31]:
```
features_train, features_test, labels_train, labels_test = train_test_split(
    tfidf_matrix, movie['class-label'], test_size=0.3,random_state=8)
print(features_train.shape, features_test.shape, labels_train.shape, labels_test.shape)
```

```
(102, 344) (45, 344) (102,) (45,)
```

In [32]:
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
```

In [33]:
```
#SVM classifier

from sklearn.svm import SVC
print("\nEvaluation for SVM \n")
svc = SVC(kernel='sigmoid', gamma=1.0)
svc.fit(features_train, labels_train)
prediction = svc.predict(features_test)
acc = accuracy_score(labels_test,prediction)
print('Accuracy:', acc)

from sklearn.metrics import precision_score
prec = precision_score(labels_test,prediction, average='weighted')
print('Precision:', prec)

from sklearn.metrics import recall_score
recall = recall_score(labels_test,prediction, average='weighted')
print('Recall:', recall)

from sklearn.metrics import f1_score
f1 = f1_score(labels_test,prediction, average='weighted')
print('F-1 measure: ', f1)
print('\nConfusion Matrix:\n')
print(confusion_matrix(labels_test, prediction))
print(classification_report(labels_test, prediction))
#print(prediction)
```

```
Evaluation for SVM

Accuracy: 0.8888888888888888
Precision: 0.9021164021164021
Recall: 0.8888888888888888
F-1 measure:  0.8671525380386138

Confusion Matrix:

[[ 3  5]
 [ 0 37]]
              precision    recall  f1-score   support

          -1       1.00      0.38      0.55         8
           1       0.88      1.00      0.94        37

    accuracy                           0.89        45
   macro avg       0.94      0.69      0.74        45
weighted avg       0.90      0.89      0.87        45
```

In [34]:
```
#Decision Tree

print("\nEvaluation for Decision Tree \n")
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(features_train, labels_train)
prediction = dtree.predict(features_test)
acc = accuracy_score(labels_test,prediction)
print('Accuracy: ', acc)
prec = precision_score(labels_test,prediction, average='weighted')
print('Precision: ', prec)
recall = recall_score(labels_test,prediction, average='weighted')
print('Recall: ', recall)
f1 = f1_score(labels_test,prediction, average='weighted')
print('F-1 measure: ',f1)
print('\nConfusion Matrix:\n')
print(confusion_matrix(labels_test, prediction))
print(classification_report(labels_test, prediction))
```

```
Evaluation for Decision Tree

Accuracy:  0.8888888888888888
Precision:  0.8822222222222222
Recall:  0.8888888888888888
F-1 measure:  0.8782328782328783

Confusion Matrix:

[[ 4  4]
 [ 1 36]]
              precision    recall  f1-score   support

          -1       0.80      0.50      0.62         8
           1       0.90      0.97      0.94        37

    accuracy                           0.89        45
   macro avg       0.85      0.74      0.78        45
weighted avg       0.88      0.89      0.88        45
```

The 'Try it Yourself' exercise answer

wordcloud for positive reviews

In [35]:

```
#Preparing text for Wordcloud
text = []
for t in textFeatures:
  text.append(t)

positive_text = ', '.join(t for t in text)
#print(positive_text)
print(len(positive_text))
```

```
6207
```

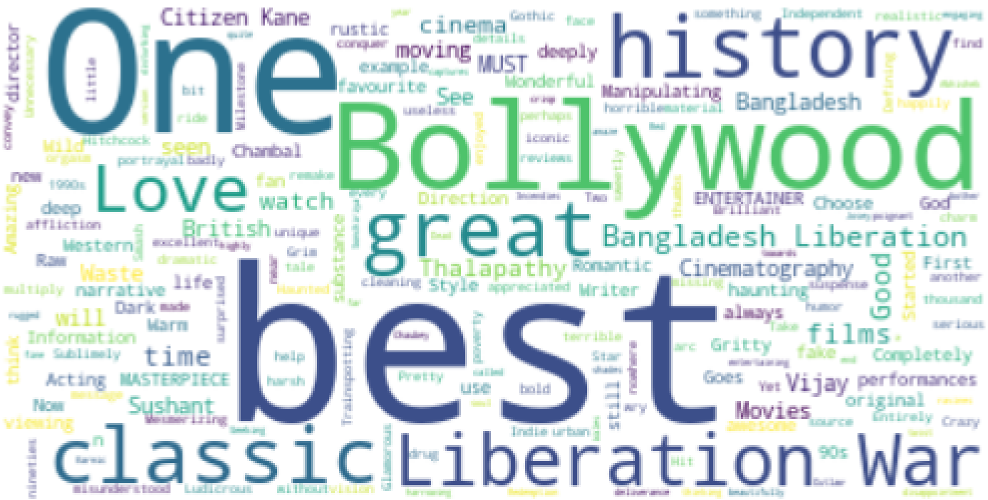
In [36]:
```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [37]:
```
# Create stopword list
stopwords = set(STOPWORDS)
stopwords.update(["br", "im", "thats", "film", "movie"]) #"im","lol","Xa","film"])

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(positive_text)

# Display the image
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
#save the generated image to a file
#wordcloud.to_file("wordcloud_cb_positive.png")
```



wordcloud for negative reviews

In [38]:
```
#Preparing text for Wordcloud
text = []
for t in textFeatures:
  text.append(t)

negative_text = ', '.join(t for t in text)
#print(negative_text)
print(len(negative_text))
```

```
6207
```

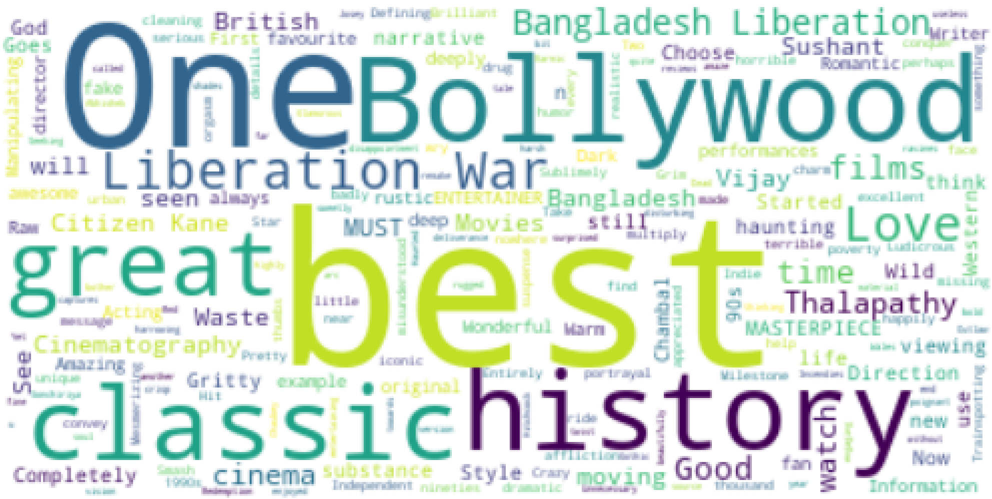
In [39]:
```
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [40]:
```
# Create stopword list
stopwords = set(STOPWORDS)
stopwords.update(["br", "im", "thats", "film", "movie"]) #"im","lol","Xa","film"])

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(negative_text)

# Display the image
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
#save the generated image to a file
#wordcloud.to_file("wordcloud_cb_negative.png")
```



Report

For this precise data set sentimental analysis was performed using a tool called Vader.

This tool is a rule based sentiment that is attuned to understand sentiments on social media for example it tells if a comment is positive negative or neutral'

To perform a sentimental analysis there are several steps that should be taken into consideration first of all data has to be collected to create a dataframe in this case reviews on ratings will be collected to create a dataframe called movie, the next step would be the performance of analysis and test processing. One of the things that this step involves easy creation of word clouds. These word clouds show an image of the most prevalent words that show sentiment for all the reviews.

Then Vader given the data frame movie will perform a sentiment identification which involves assigning positive negative or neutral to reviews. There are limitations. This means that the sentiment algorithm might not correctly classify a comment in the right way in the sense that the algorithm can say that the comment is positive whereas in reality it is negative. And sarcastic comments will not be easily identified correctly by the sentiment algorithm.

An example is the sentence: With God's help I'll conquer this terrible affliction (review no 15). the algorithm classifies it as negative but in reality it is a positive comment.

Then when it comes to evaluation for decision tree and evaluation for SVM. It can be seen that SVM is more precise in classifying negative comments as negative due to it's value of 1 where as the classification tree is better at classifying positive comments as positive due to its precision value of O.9 compared to 0.88 in SVM. But overall the best evaluation matrix to use is SVM.