Practical - 10

Name : Shivam Tawari
Roll no : A-58
Subject : NLP

Aim : Write a python program for text classification with NLTK.

Theory:

Introduction:

The goal of this practical is to implement text classification. Maybe we're trying to classify text as about politics or the military. Maybe we're trying to classify it by the gender of the author who wrote it. A fairly popular text classification task is to identify a body of text as either spam or not spam, for things like email filter. In our case, we're going to try to create a sentiment analysis algorithm.

To do this, we're going to start by trying to use the movie reviews database that is part of the NLTK corpus, from there we'll try to use words as "features" which are a part of either a positive or negative movie review. The NLTK corpus dataset has the reviews, and they are labelled already as positive or negative. This means we can train and test with this data.

* TF - IDF :

Term frequency - Inverse Document frequency (TF-IDF) is a numerical statistic that is intented to reflect how important a word is to a document in a collection of corpus. It i's often used as a weighting factor in searches of IR, text mining, and user modelling.

* Logistic Regression :

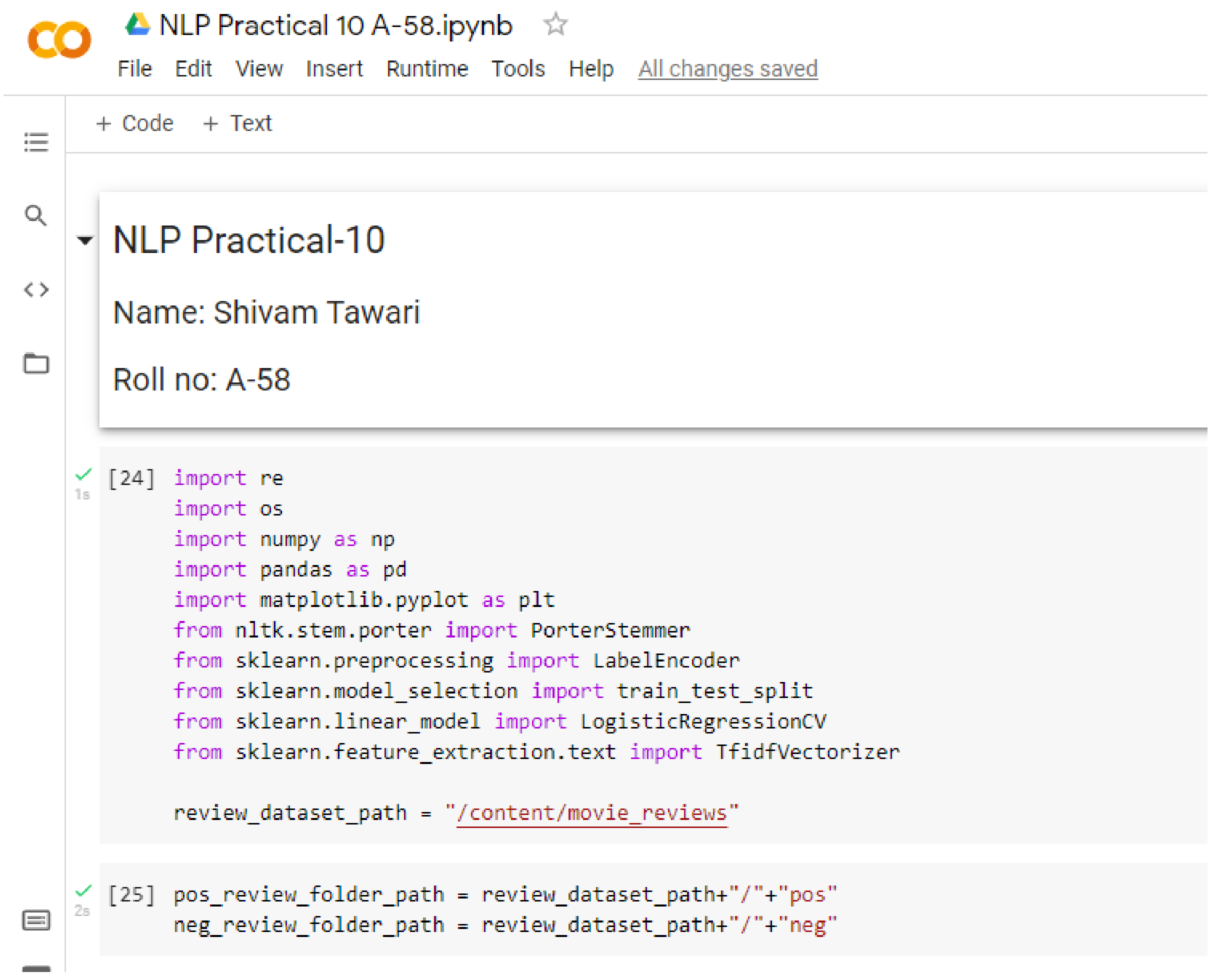
Logistic regression is a supervised learning algorithm. In classification problem, the target variable can take only discrete values for given set of features. The logistic model is used to model the probability of a certain class or event existing such as pass / fail, win / lose, positive / negative.

Conclusion:
Hence, we have implemented a python program for text classification.

Practical 10

Name: Shivam Tawari

Roll no: A – 58


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

NLP Practical-10

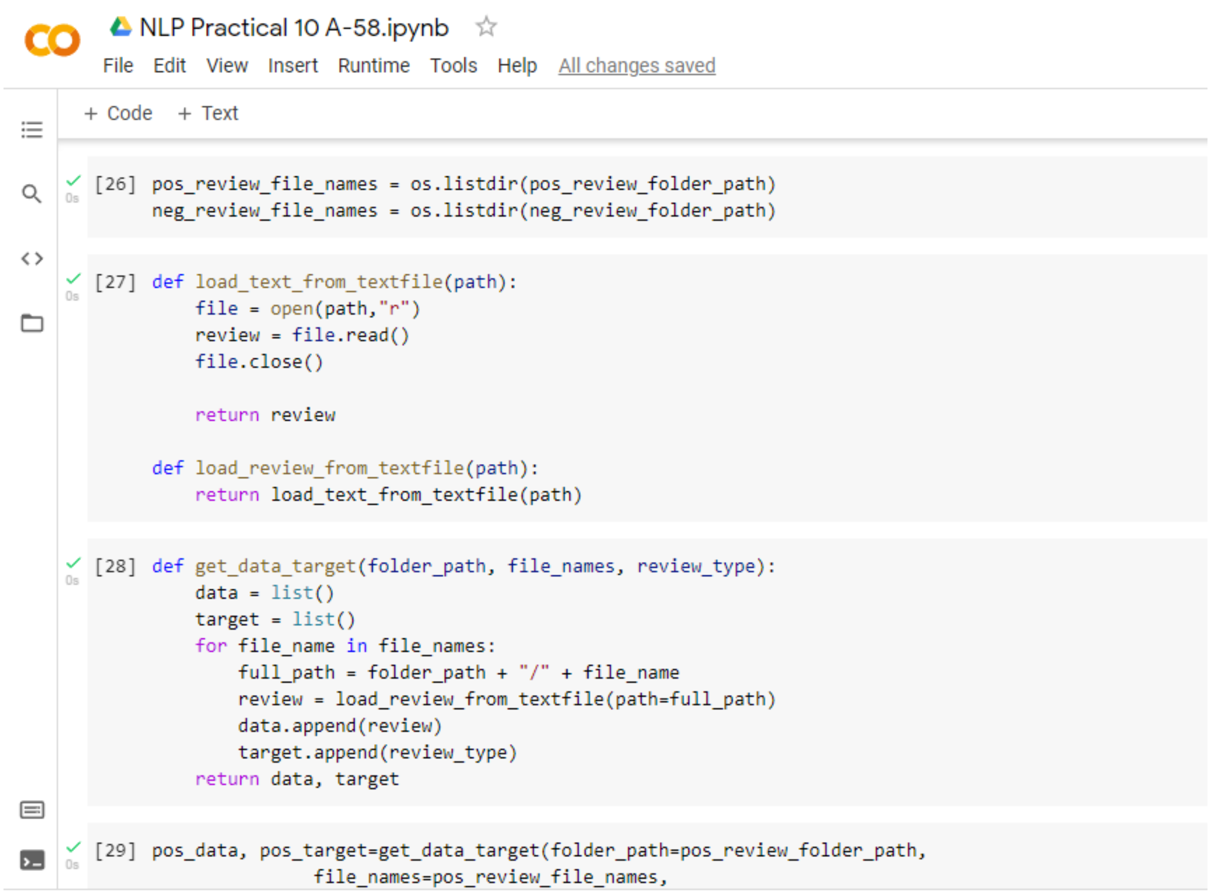
Name: Shivam Tawari

Roll no: A-58

```
[24] import re
     import os
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from nltk.stem.porter import PorterStemmer
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegressionCV
     from sklearn.feature_extraction.text import TfidfVectorizer

     review_dataset_path = "/content/movie_reviews"
```

```
[25] pos_review_folder_path = review_dataset_path+"/"+"pos"
     neg_review_folder_path = review_dataset_path+"/"+"neg"
```


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[26] pos_review_file_names = os.listdir(pos_review_folder_path)
     neg_review_file_names = os.listdir(neg_review_folder_path)
```

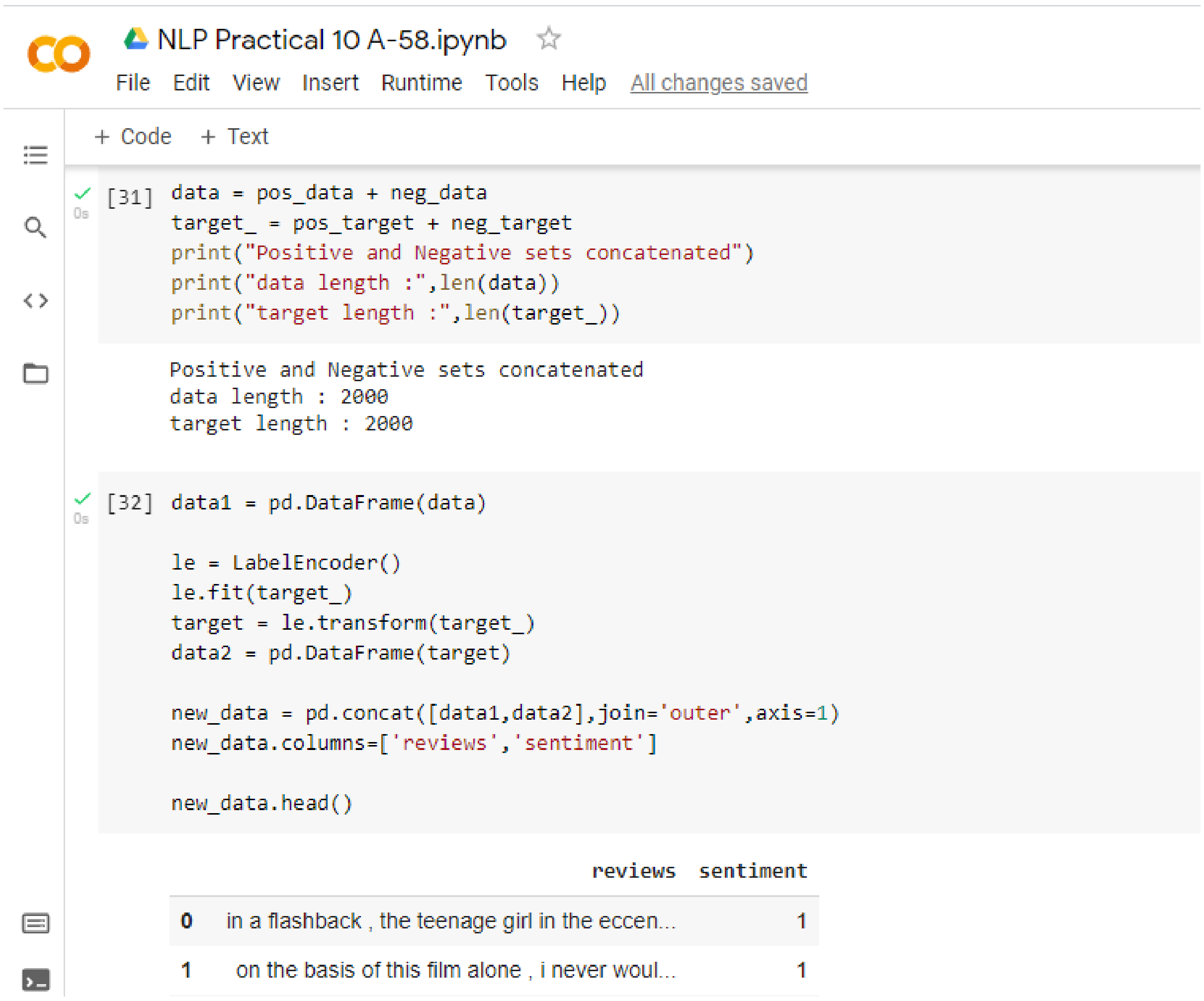
```
[27] def load_text_from_textfile(path):
         file = open(path,"r")
         review = file.read()
         file.close()

         return review

     def load_review_from_textfile(path):
         return load_text_from_textfile(path)
```

```
[28] def get_data_target(folder_path, file_names, review_type):
         data = list()
         target = list()
         for file_name in file_names:
             full_path = folder_path + "/" + file_name
             review = load_review_from_textfile(path=full_path)
             data.append(review)
             target.append(review_type)
         return data, target
```

```
[29] pos_data, pos_target=get_data_target(folder_path=pos_review_folder_path,
                   file_names=pos_review_file_names,
```


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
[31] data = pos_data + neg_data
     target_ = pos_target + neg_target
     print("Positive and Negative sets concatenated")
     print("data length :",len(data))
     print("target length :",len(target_))
```

```
Positive and Negative sets concatenated
data length : 2000
target length : 2000
```

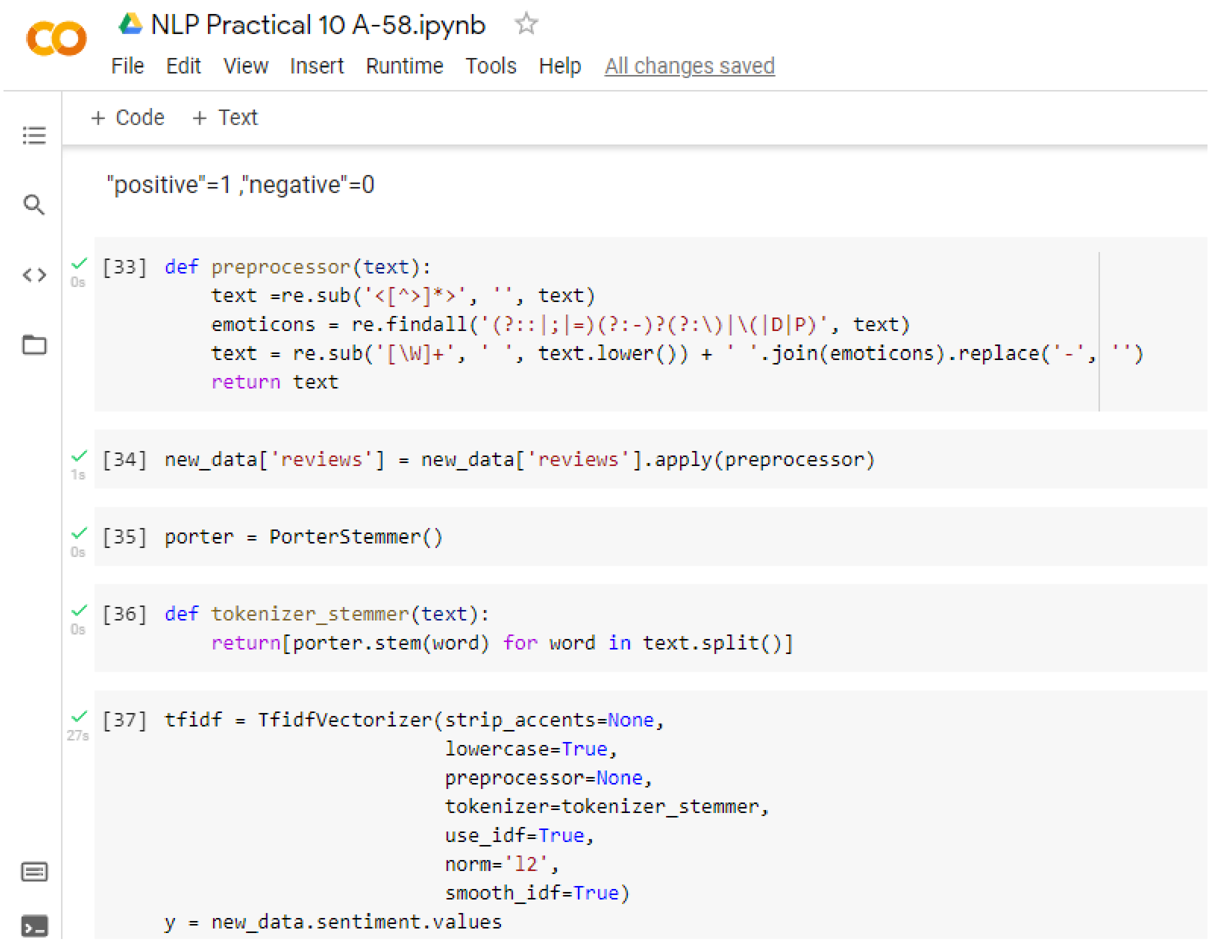
```
[32] data1 = pd.DataFrame(data)

     le = LabelEncoder()
     le.fit(target_)
     target = le.transform(target_)
     data2 = pd.DataFrame(target)

     new_data = pd.concat([data1,data2],join='outer',axis=1)
     new_data.columns=['reviews','sentiment']

     new_data.head()
```

	reviews	sentiment
0	in a flashback , the teenage girl in the eccen...	1
1	on the basis of this film alone , i never woul...	1


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

"positive"=1 ,"negative"=0

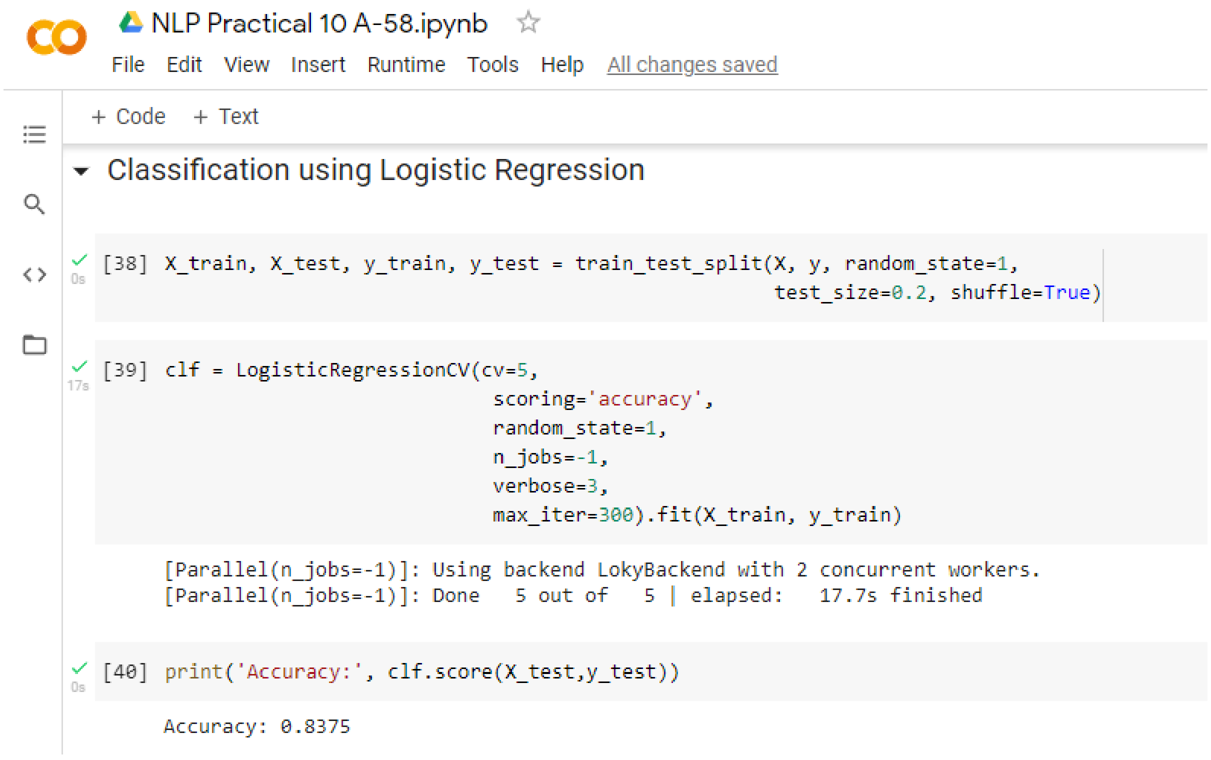
```
[33] def preprocessor(text):
         text =re.sub('<[^>]*>', '', text)
         emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', text)
         text = re.sub('[\W]+', ' ', text.lower()) + ' '.join(emoticons).replace('-', '')
         return text
```

```
[34] new_data['reviews'] = new_data['reviews'].apply(preprocessor)
```

```
[35] porter = PorterStemmer()
```

```
[36] def tokenizer_stemmer(text):
         return[porter.stem(word) for word in text.split()]
```

```
[37] tfidf = TfidfVectorizer(strip_accents=None,
                             lowercase=True,
                             preprocessor=None,
                             tokenizer=tokenizer_stemmer,
                             use_idf=True,
                             norm='l2',
                             smooth_idf=True)
     y = new_data.sentiment.values
```


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

Classification using Logistic Regression

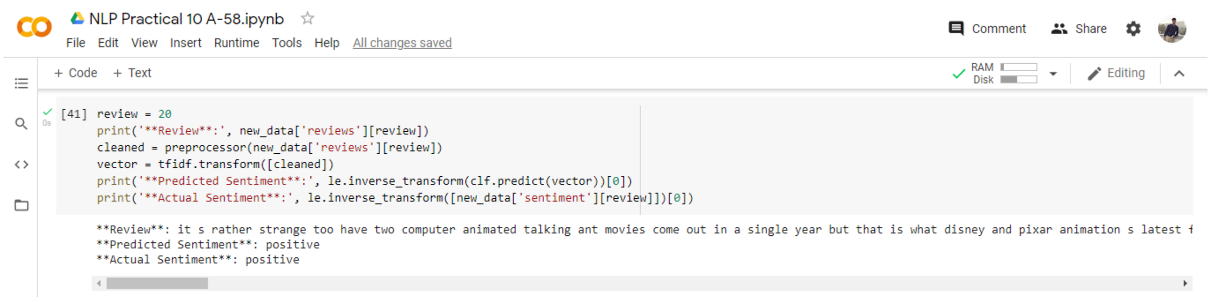
```
[38] X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1,
                                                         test_size=0.2, shuffle=True)
```

```
[39] clf = LogisticRegressionCV(cv=5,
                                scoring='accuracy',
                                random_state=1,
                                n_jobs=-1,
                                verbose=3,
                                max_iter=300).fit(X_train, y_train)

     [Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
     [Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   17.7s finished
```

```
[40] print('Accuracy:', clf.score(X_test,y_test))

     Accuracy: 0.8375
```


NLP Practical 10 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
[41] review = 20
     print('**Review**:', new_data['reviews'][review])
     cleaned = preprocessor(new_data['reviews'][review])
     vector = tfidf.transform([cleaned])
     print('**Predicted Sentiment**:', le.inverse_transform(clf.predict(vector))[0])
     print('**Actual Sentiment**:', le.inverse_transform([new_data['sentiment'][review]])[0])
```

```
**Review**: it s rather strange too have two computer animated talking ant movies come out in a single year but that is what disney and pixar animation s latest f
**Predicted Sentiment**: positive
**Actual Sentiment**: positive
```