Deep Learning Project - Movie Reviews Sentiment Analysis

Names :

1. Anoop Reddy Kallem - VU21CSEN0500062
2. V Geetha Priya - VU21CSEN0500065
3. G Siva Sai Rushyendra - VU21CSEN0500066

Framework used :

1. spaCy
2. sklearn
3. Pandas

Machine Learning model used : LinearSVC - Linear Support Vector Machine Classifier

Metrics : Precision, Recall, F1-score, Support

Import spaCy

Import displacy for displaying word dependecies

In [2]:
```
import spacy
from spacy import displacy
```

Load English Language from spaCy

In [3]:
```
nlp=spacy.load('en_core_web_sm')
```

Using `nlp()` over a pre-defined text

In [7]:
```
text="This movie is very bad. This is worst than the one I watch a week ago."
doc=nlp(text)
doc
```

Out[7]:
```
This movie is very bad. This is worst than the one I watch a week ago.
```

Tokenization of the pre-defined text

In [8]:
```
for token in doc:
    print(token)
```

```
This
movie
is
very
bad
.
This
is
worst
than
the
one
I
watch
a
week
ago
.
```

Imputing Sentencizer for separating sentences.

In [9]:
```
sentencizer = nlp.create_pipe('sentencizer')
nlp.add_pipe('sentencizer', before='parser')
```

Out[9]:
```
<spacy.pipeline.sentencizer.Sentencizer at 0x2c0f8856d00>
```

In [10]:
```
for sentencizer in doc.sents:
    print(sentencizer)
```

```
This movie is very bad.
This is worst than the one I watch a week ago.
```

Import `STOP_WORDS` from spaCy English.

STOP_WORDS are the words which repeat more times in a mostly for the purpose of joining sentences, maintaining the syntaxes and semantics of the given context.

In [11]:
```
from spacy.lang.en.stop_words import STOP_WORDS
```

In [12]:
```
stopwords=list(STOP_WORDS)
print(stopwords)
```

```
['see', 'whole', 'except', 'a', 'quite', 'them', 'himself', 'hereafter', 'beyond',
'always', 'call', 'was', 'too', 'yourself', 'nevertheless', 'from', 'as', 'using',
'been', 'these', 'show', 'i', "'m", 'perhaps', 'your', 'all', 'whereas', 'have',
'upon', 'along', 'other', 'nor', 'somehow', 'about', 'forty', 'yourselves', 'where
after', 'with', 'since', 'mostly', 'not', '‘ll', 'wherein', 'third', 'name', 'the
n', 'still', 'everyone', 'between', 'something', 'because', 'behind', 'many', 'aga
in', 'seem', 'towards', 'twelve', '‘d', 'whereupon', 'until', 'under', 'had', 'u
p', 'here', '’re', 'indeed', 'what', 'whenever', 'amongst', 'than', 'however', 'th
roughout', 'he', 'we', 'first', 'yours', 'if', 'must', 'regarding', "'re", 'becomi
ng', 'why', 'of', "'ll", 'herself', 'amount', 'others', 'whither', 'enough', 'befo
re', 'thereupon', 'him', 'whereby', 'front', 'but', 'is', 'never', '’m', 'while',
'above', 'could', 'hers', 'am', 'keep', 'side', 'four', 'ever', 'just', 'several',
'no', 'afterwards', 'being', 'may', 'please', 'therein', 'made', 'themselves', 'fi
fteen', 'fifty', 'via', 'how', 'whose', 'by', 'toward', "n't", 'ca', 'give', 'us',
'own', 'full', 'seemed', 'anyone', 'his', '’ll', "'ve", 'well', 'also', 'everywher
e', 'against', 'almost', 'has', 'around', 'hereby', '’s', 'it', 'any', 'among', 'c
an', 'seems', 'they', 'through', 'anyhow', 'at', 'beforehand', 'become', 'two', 'l
ess', 'on', 'either', 'beside', 'put', 'none', 'some', 'down', 'make', 'our', 'of
f', 'six', 'most', 'part', 'eleven', 'empty', 'get', 'one', 'often', 'used', 'some
one', 'whence', 'nowhere', 'alone', 'there', '‘ve', 'few', 'say', 'next', "'d", 'a
fter', 'those', 'ten', 'only', 'more', 'thereafter', 'she', '’ve', 'nine', 'much',
'and', 'done', 'this', 'unless', '‘s', 'across', 'bottom', 'hence', 'another', 'th
ence', '‘m', 'ourselves', 'out', 'take', 'due', 'very', 'without', 're', 'which',
'now', 'should', 'meanwhile', 'moreover', 'mine', 'its', 'who', 'will', 'yet', 'fo
r', 'various', 'seeming', 'the', 'everything', 'did', 'back', 'somewhere', 'anywa
y', 'already', 'into', 'same', 'you', 'five', 'onto', 'twenty', 'whom', 'doing',
'former', 'elsewhere', 'three', 'eight', 'thru', 'over', 'anything', 'below', 'tog
ether', 'ours', 'where', 'sometimes', 'rather', 'such', 'thus', 'thereby', 'althou
gh', 'move', 'sixty', 'anywhere', 'latterly', 'me', 'otherwise', 'latter', 'n‘t',
'namely', 'each', 'my', 'becomes', 'n’t', 'else', 'or', 'within', 'myself', 'per',
'might', 'top', 'so', 'really', 'nothing', 'cannot', 'every', 'noone', 'during',
'would', 'that', 'when', 'her', 'itself', 'an', 'though', 'became', 'further', 'he
rein', 'serious', 'even', 'were', 'whatever', 'in', 'once', 'hundred', 'does', 'so
metime', 'last', 'whoever', 'whether', 'least', 'both', 'their', 'are', 'formerl
y', '’d', '‘re', 'wherever', 'do', 'go', 'therefore', 'be', 'nobody', 'to', 'besid
es', 'hereupon', 'neither', "'s"]
```

Drop STOP_WORDS

In [13]:
```
for token in doc:
    if token.is_stop==False:
        print(token)
```

```
movie
bad
.
worst
watch
week
ago
.
```

Lemmatization - finding the base word of an existing word in the dataset

In [14]:
```
for lem in doc:
    print(lem.text,lem.lemma_)
```

```
This this
movie movie
is be
very very
bad bad
. .
This this
is be
worst bad
than than
the the
one one
I I
watch watch
a a
week week
ago ago
. .
```

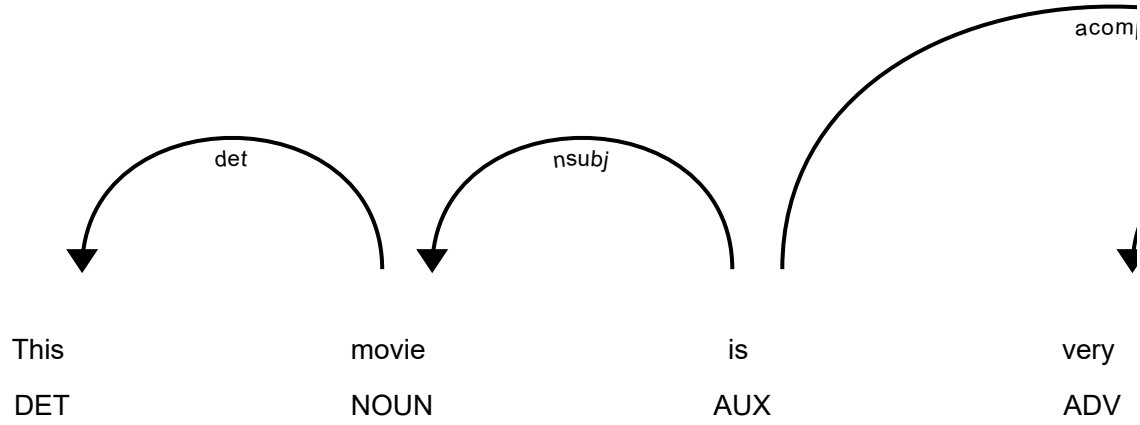
Tagging each word in the text with a Parts-Of-Speech tag

In [15]:
```
pos_list=[]
for token in doc:
    print(token.text,token.pos_,spacy.explain(token.pos_))
```

```
This DET determiner
movie NOUN noun
is AUX auxiliary
very ADV adverb
bad ADJ adjective
. PUNCT punctuation
This PRON pronoun
is AUX auxiliary
worst ADJ adjective
than ADP adposition
the DET determiner
one NOUN noun
I PRON pronoun
watch VERB verb
a DET determiner
week NOUN noun
ago ADV adverb
. PUNCT punctuation
```

This code snippet : `displacy.render(doc)` performs the following functionality - This command renders the processed `doc` using spaCy's built-in visualization tool called `displacy`. The `displacy.render()` function takes the processed doc as input and generates a visualization of the analyzed text. The visualization typically includes the original text with annotations such as part-of-speech tags, named entities, and syntactic dependencies, displayed in an interactive and visually appealing format.

In [16]:
```
doc=nlp(text)
displacy.render(doc)
```



In [17]:
```
doc=nlp(text)
displacy.render(doc,style='ent')
```

This movie is very bad. This is worst than the one I watch a week ago **DATE** .

Importing `Vectorizer` , `Pipeline` , `Train Test Split` and `Metrics` .

In [18]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
import pandas as pd
```

Reading IMDB Dataset with 50,000 records and mapping `positive` and `negative` values to `1` and `0` respectively.

In [19]:
```
df=pd.read_csv('IMDB Dataset.csv')
df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
df
```

Out[19]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1
...	...	...
49995	I thought this movie did a down right good job...	1
49996	Bad plot, bad dialogue, bad acting, idiotic di...	0
49997	I am a Catholic taught in parochial elementary...	0
49998	I'm going to have to disagree with the previou...	0
49999	No one expects the Star Trek movies to be high...	0

50000 rows × 2 columns

In [20]:
```
column_names=['Reviews','Sentiments']
df.columns=column_names
df
```

Out[20]:

	Reviews	Sentiments
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1
...	...	...
49995	I thought this movie did a down right good job...	1
49996	Bad plot, bad dialogue, bad acting, idiotic di...	0
49997	I am a Catholic taught in parochial elementary...	0
49998	I'm going to have to disagree with the previou...	0
49999	No one expects the Star Trek movies to be high...	0

50000 rows × 2 columns

In [22]:
```
df.shape
df['Sentiments'].value_counts()
```

Out[22]:
```
Sentiments
1    25000
0    25000
Name: count, dtype: int64
```

In [23]:
```
import string
```

Cleaning text data by removing punctuations, stop words, etc

In [24]:
```
puct=string.punctuation
puct
def text_data_cleaning(sentence):
    doc=nlp(sentence)
    tokens=[]
    for token in doc:
        if token.lemma_ != "-PRON-":
            temp=token.lemma_.lower().strip()
        else :
            temp=token.lower_
        tokens.append(temp)
    cleaned_tokens=[]
    for token in tokens:
        if token not in stopwords and token not in puct:
            cleaned_tokens.append(token)
    return cleaned_tokens
```

In [25]:
```
text_data_cleaning("hello how are you. i am fine.")
```

Out[25]:
```
['hello', 'fine']
```

Importing LinearSCV

In [22]:
```
from sklearn.svm import LinearSVC
```

In [23]:
```
tfidf=TfidfVectorizer(tokenizer=text_data_cleaning)
classifier=LinearSVC()
```

Declaring Feature column and Target column

In [24]:
```
X=df['Reviews']
y=df['Sentiments']
```

Splitting data into training and testing

`Training - 70%` `Testing - 30%` `random_state=32`

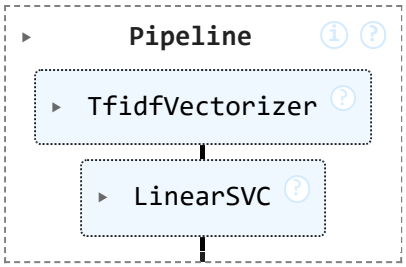
In [25]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=32)
```

`Fitting the model to the data`

In [26]:
```
clf=Pipeline([('tfidf',tfidf),('clf',classifier)])
clf.fit(X_train,y_train)
```

```
C:\Users\kalle\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2k
fra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\feature_extractio
n\text.py:525: UserWarning: The parameter 'token_pattern' will not be used since
'tokenizer' is not None'
  warnings.warn(
C:\Users\kalle\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2k
fra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\svm\_classes.py:3
1: FutureWarning: The default value of `dual` will change from `True` to ``'auto'``
in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

Out[26]:



Predicting the review sentiment

In [27]:
```
y_pred=clf.predict(X_test)
```

In [28]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      7385
           1       0.89      0.90      0.89      7615

    accuracy                           0.89     15000
   macro avg       0.89      0.89      0.89     15000
weighted avg       0.89      0.89      0.89     15000
```

In [29]:
```
confusion_matrix(y_test,y_pred)
```

Out[29]:
```
array([[6521,  864],
       [ 760, 6855]], dtype=int64)
```

Take custom input review from end user to classify it as Positive or Negative

In [42]:
```
review_input = []
print("Enter the number of reviews to input:")
n = int(input())

for i in range(n):
    print("Please enter your review:")
    print("\n")
    x = input()
    print(x)
    review_input.append(x)

# print(review_input)

predictions = clf.predict(review_input)

# Print predictions along with reviews
for review, prediction in zip(review_input, predictions):
    if prediction == 0:
        print("\n")
        print(review)
        print("===> Negative")
    else:
        print("\n")
        print(review)
        print("===> Positive")
```

```
Enter the number of reviews to input:
Please enter your review:


This movie does a great job of explaining the problems that we faced and the fears
that we had before we put man into space. As a history of space flight, it is stil
l used today in classrooms that can get one of the rare prints of it. Disney has s
hown it on "Vault Disney" and I wish they would do so again.
Please enter your review:


I'll not comment a lot, what's to??? Stereotype characters, absolute ignorance abo
ut Colombia's reality, awful mise en scene, poor color choice, NOT funny (it suppo
sed to be a comedy and they expect that you will laugh because some distend music
it's beside the nonsense scenes), Very poor actors direction (if you see somewhere
those people, I mean the interpreters, you'll know they are at least good, but see
ing this so call film, it is impossible to guess it), you get tired of the musi
c... this "comedy" has no rhythm, the only good rhythm in it, it's the rap sing in
the final credits....pathetic, doesn't it? etc...etc... It has been a long time I
haven't seen a movie so bad!!
Please enter your review:


If you really really REALLY enjoy movies featuring ants building dirt-mirrors, eat
ing non-ants, and conquering the world with a voice-over narrative, then this is t
he movie for you.


This movie does a great job of explaining the problems that we faced and the fears
that we had before we put man into space. As a history of space flight, it is stil
l used today in classrooms that can get one of the rare prints of it. Disney has s
hown it on "Vault Disney" and I wish they would do so again.
===> Positive


I'll not comment a lot, what's to??? Stereotype characters, absolute ignorance abo
ut Colombia's reality, awful mise en scene, poor color choice, NOT funny (it suppo
sed to be a comedy and they expect that you will laugh because some distend music
it's beside the nonsense scenes), Very poor actors direction (if you see somewhere
those people, I mean the interpreters, you'll know they are at least good, but see
ing this so call film, it is impossible to guess it), you get tired of the musi
c... this "comedy" has no rhythm, the only good rhythm in it, it's the rap sing in
the final credits....pathetic, doesn't it? etc...etc... It has been a long time I
haven't seen a movie so bad!!
===> Negative


If you really really REALLY enjoy movies featuring ants building dirt-mirrors, eat
ing non-ants, and conquering the world with a voice-over narrative, then this is t
he movie for you.
===> Negative
```