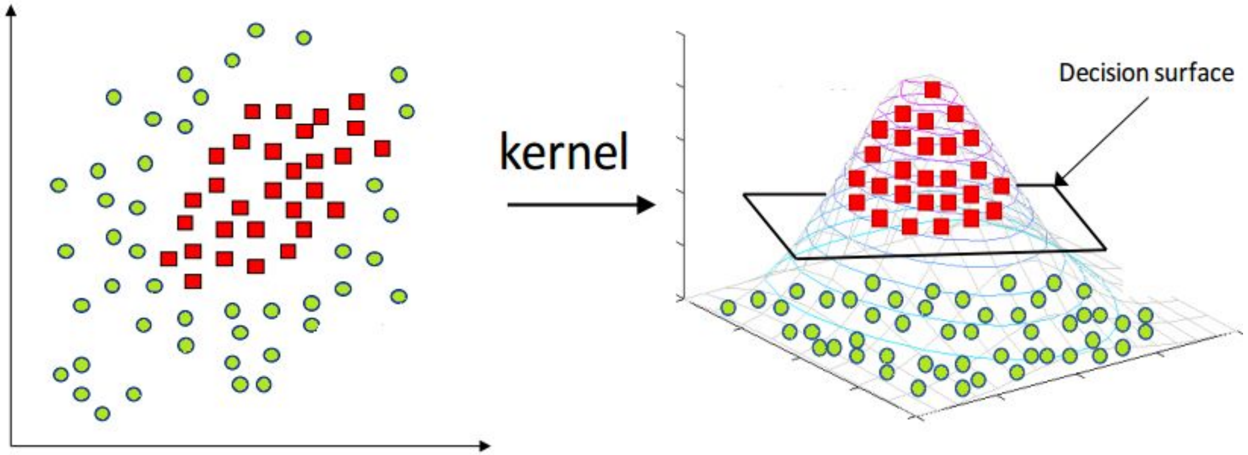
Sentiment Analysis using SVM

SVM(Support Vector Machine)?

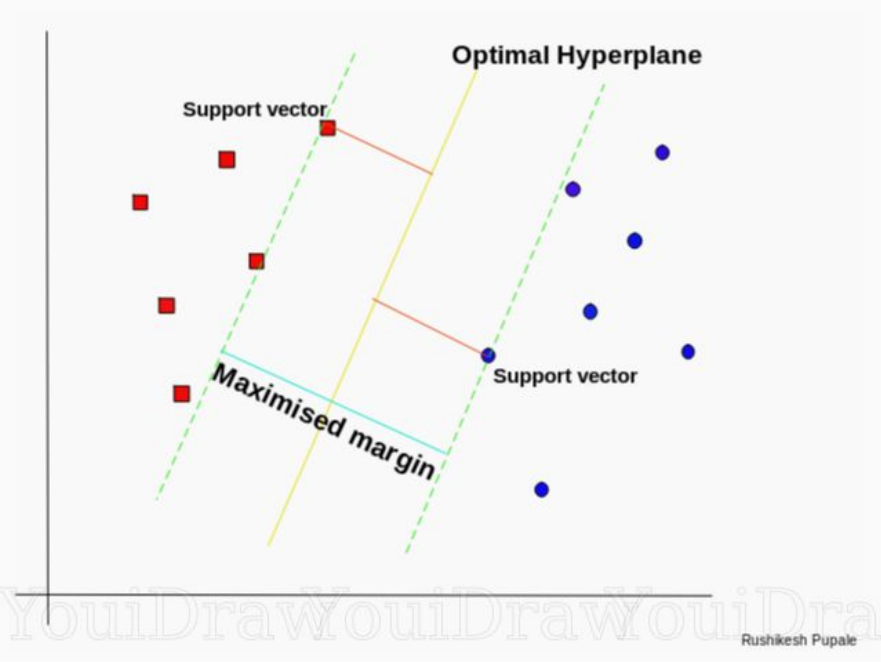
- A supervised machine learning algorithm which can be used for both classification or regression problems
- Widely used in classification objectives
- Highly preferred by many as it produces significant accuracy with less computation power.

How SVM works?

- **IDEA:** The algorithm creates a line or a hyperplane which separates the data into classes
- SVM draws that hyperplane by transforming our data with the help of mathematical functions called “Kernels”



- Finding support vectors
- Compute the distance(margin) between the line and the support vectors
- The hyperplane for which the margin is maximum is the optimal hyperplane



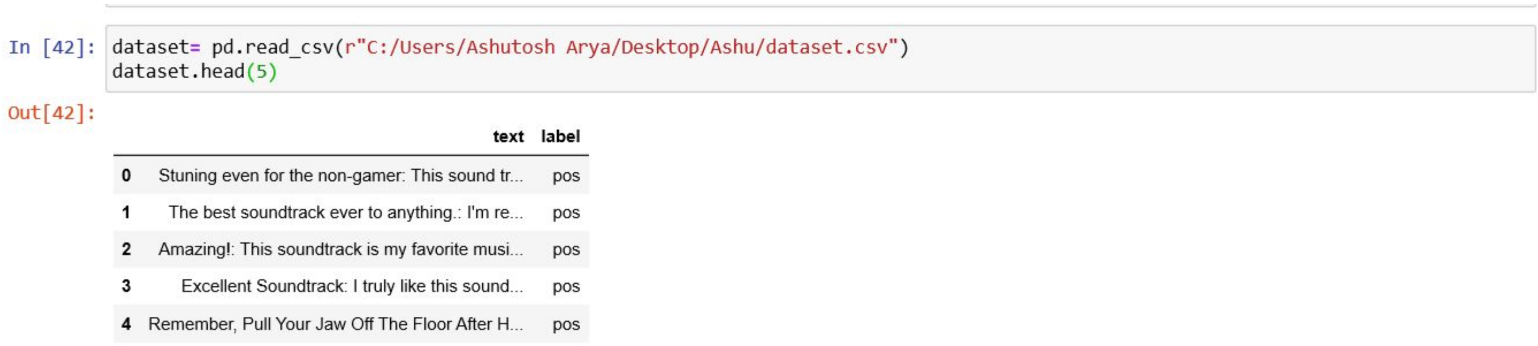
Steps involved

- Adding Required Libraries
- Getting the dataset
- Data pre-processing
- Train and Test Data sets
- Encoding
- Word Vectorization
- Predicting the outcome

Dataset

- Amazon Review Data set which has 10,000 rows of Text data
- Classified into “pos” and “neg”
- Has two columns “Text” and “Label”

Source: github.com/Gunjitbedi/Text-Classification/blob/master/corpus.csv



```
In [42]: dataset= pd.read_csv(r"C:/Users/Ashutosh Arya/Desktop/Ashu/dataset.csv")
         dataset.head(5)
```

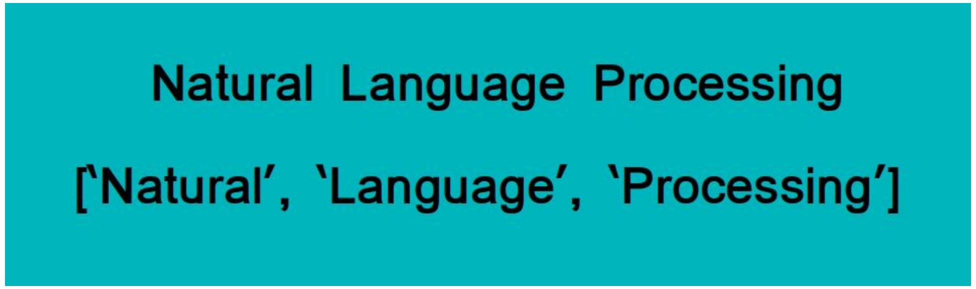
Out[42]:

	text	label
0	Stuning even for the non-gamer: This sound tr...	pos
1	The best soundtrack ever to anything.: I'm re...	pos
2	Amazing!: This soundtrack is my favorite musi...	pos
3	Excellent Soundtrack: I truly like this sound...	pos
4	Remember, Pull Your Jaw Off The Floor After H...	pos

Important concepts

- **Tokenization**

Process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. *word_tokenize* and *sent_tokenize* functions in NLTK library.

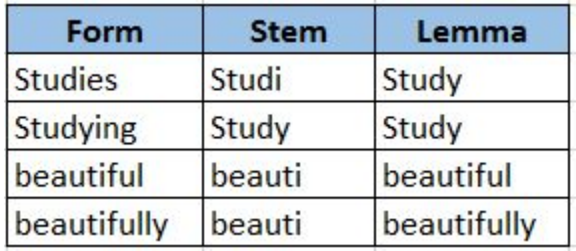


Natural Language Processing

[‘Natural’, ‘Language’, ‘Processing’]

- **Word Stemming/Lemmatization**

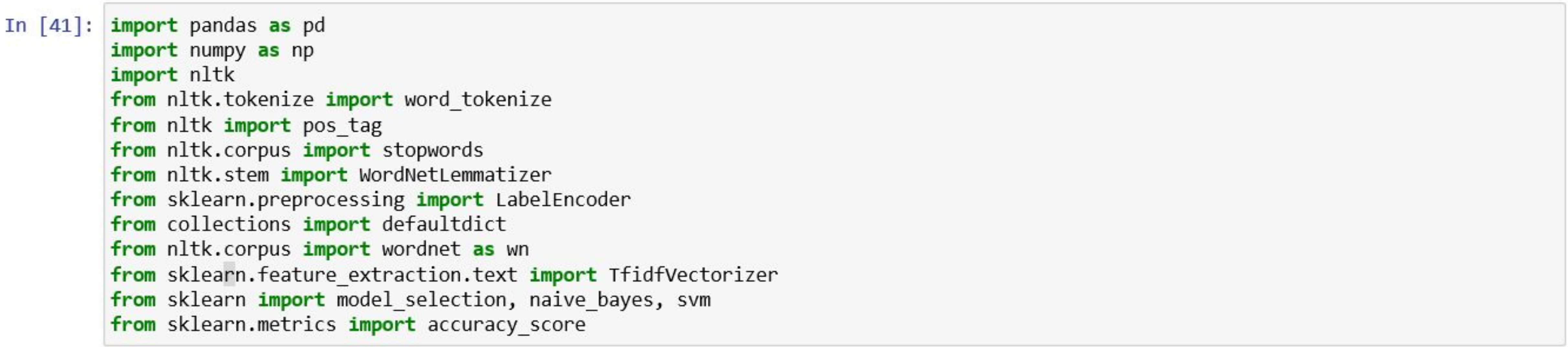
Reducing the inflectional forms of each word into a common base or root. Lemmatization is closely related to stemming.*WordNetLemmatizer* in the NLTK library.



Form	Stem	Lemma
Studies	Studi	Study
Studying	Study	Study
beautiful	beauti	beautiful
beautifully	beauti	beautifully

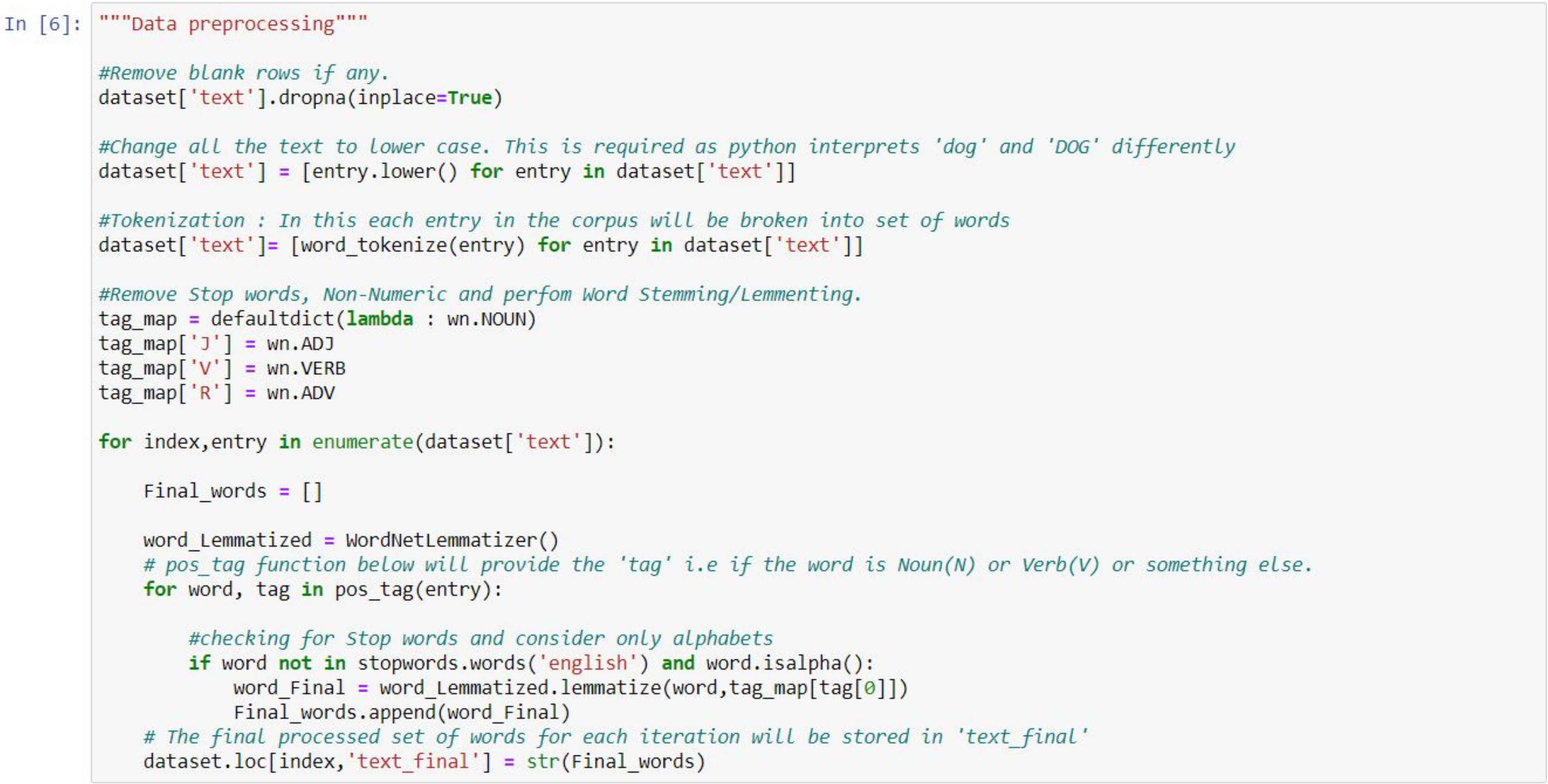
CODE

Libraries used



```
In [41]: import pandas as pd
         import numpy as np
         import nltk
         from nltk.tokenize import word_tokenize
         from nltk import pos_tag
         from nltk.corpus import stopwords
         from nltk.stem import WordNetLemmatizer
         from sklearn.preprocessing import LabelEncoder
         from collections import defaultdict
         from nltk.corpus import wordnet as wn
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn import model_selection, naive_bayes, svm
         from sklearn.metrics import accuracy_score
```

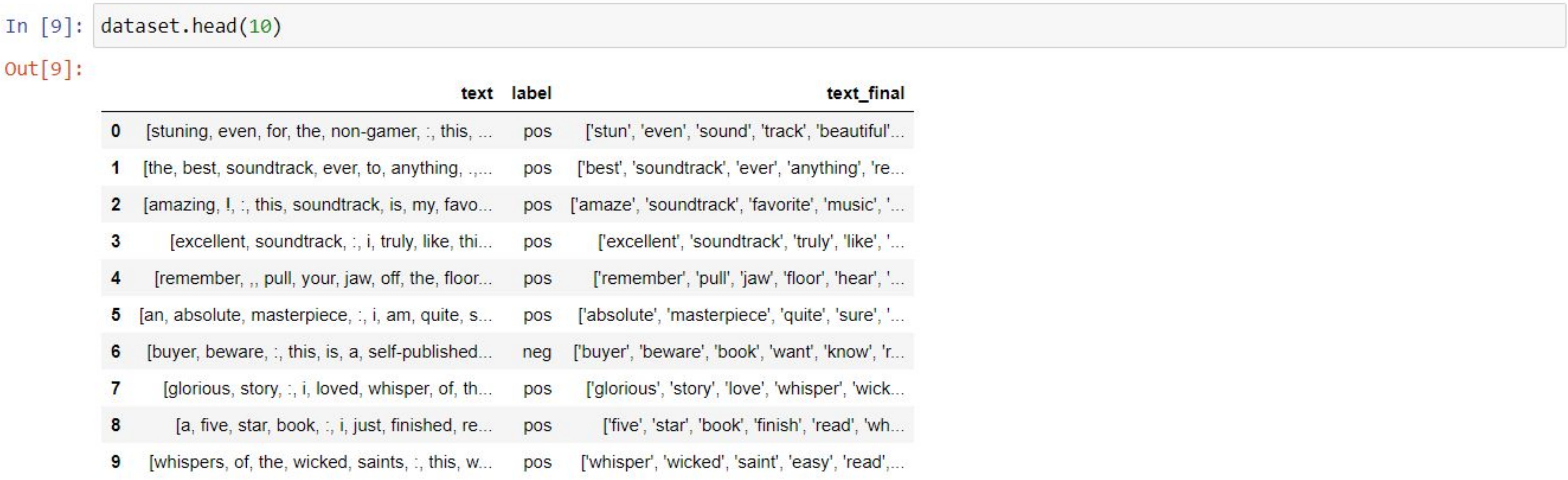
Data preprocessing



In [6]:

```
"""Data preprocessing"""

#Remove blank rows if any.
dataset['text'].dropna(inplace=True)

#Change all the text to lower case. This is required as python interprets 'dog' and 'DOG' differently
dataset['text'] = [entry.lower() for entry in dataset['text']]

#Tokenization : In this each entry in the corpus will be broken into set of words
dataset['text']= [word_tokenize(entry) for entry in dataset['text']]

#Remove Stop words, Non-Numeric and perfom Word Stemming/Lemmenting.
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV

for index,entry in enumerate(dataset['text']):

    Final_words = []

    word_Lemmatized = WordNetLemmatizer()
    # pos_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V) or something else.
    for word, tag in pos_tag(entry):

        #checking for Stop words and consider only alphabets
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)
    # The final processed set of words for each iteration will be stored in 'text_final'
    dataset.loc[index,'text_final'] = str(Final_words)
```

Processed data

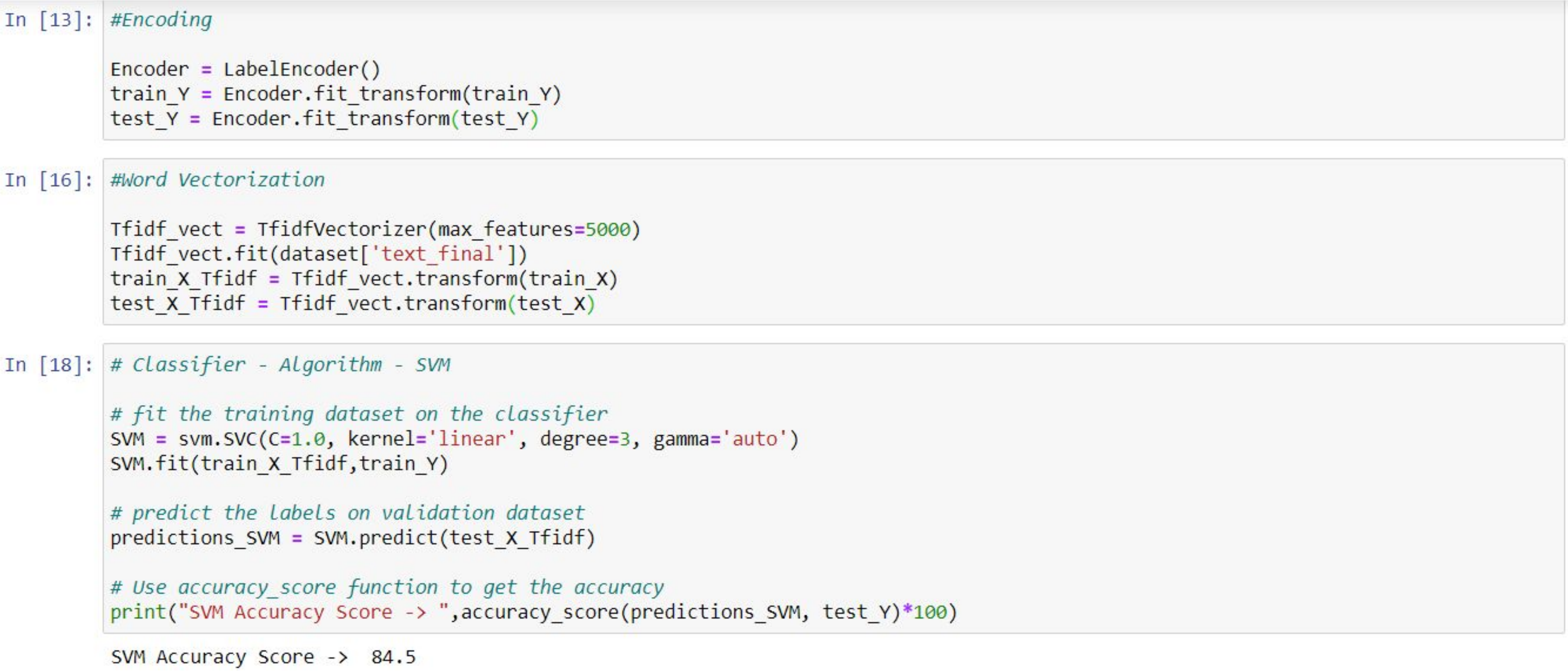


```
In [9]: dataset.head(10)
```

Out[9]:

	text	label	text_final
0	[stuning, even, for, the, non-gamer, :, this, ...	pos	['stun', 'even', 'sound', 'track', 'beautiful'...
1	[the, best, soundtrack, ever, to, anything, .,...	pos	['best', 'soundtrack', 'ever', 'anything', 're...
2	[amazing, !, :, this, soundtrack, is, my, favo...	pos	['amaze', 'soundtrack', 'favorite', 'music', '...
3	[excellent, soundtrack, :, i, truly, like, thi...	pos	['excellent', 'soundtrack', 'truly', 'like', '...
4	[remember, ,, pull, your, jaw, off, the, floor...	pos	['remember', 'pull', 'jaw', 'floor', 'hear', '...
5	[an, absolute, masterpiece, :, i, am, quite, s...	pos	['absolute', 'masterpiece', 'quite', 'sure', '...
6	[buyer, beware, :, this, is, a, self-published...	neg	['buyer', 'beware', 'book', 'want', 'know', 'r...
7	[glorious, story, :, i, loved, whisper, of, th...	pos	['glorious', 'story', 'love', 'whisper', 'wick...
8	[a, five, star, book, :, i, just, finished, re...	pos	['five', 'star', 'book', 'finish', 'read', 'wh...
9	[whispers, of, the, wicked, saints, :, this, w...	pos	['whisper', 'wicked', 'saint', 'easy', 'read',...

Encoding, word vectorization and model



```
In [13]: #Encoding

         Encoder = LabelEncoder()
         train_Y = Encoder.fit_transform(train_Y)
         test_Y = Encoder.fit_transform(test_Y)
```

```
In [16]: #Word Vectorization

         Tfidf_vect = TfidfVectorizer(max_features=5000)
         Tfidf_vect.fit(dataset['text_final'])
         train_X_Tfidf = Tfidf_vect.transform(train_X)
         test_X_Tfidf = Tfidf_vect.transform(test_X)
```

```
In [18]: # Classifier - Algorithm - SVM

         # fit the training dataset on the classifier
         SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
         SVM.fit(train_X_Tfidf,train_Y)

         # predict the labels on validation dataset
         predictions_SVM = SVM.predict(test_X_Tfidf)

         # Use accuracy_score function to get the accuracy
         print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, test_Y)*100)

         SVM Accuracy Score ->  84.5
```

Result

- SVM Accuracy Score -> 84.5