Twitter Sentimental Analysis

GROUP NUMBER:- 01

Group Members

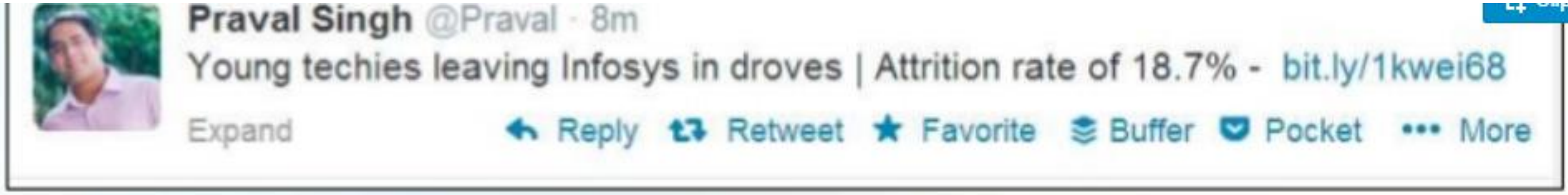
Name	Roll Number
Prathamesh Chikane	06
Subodh Halpatrao	18
Sutej Kulkarni	35

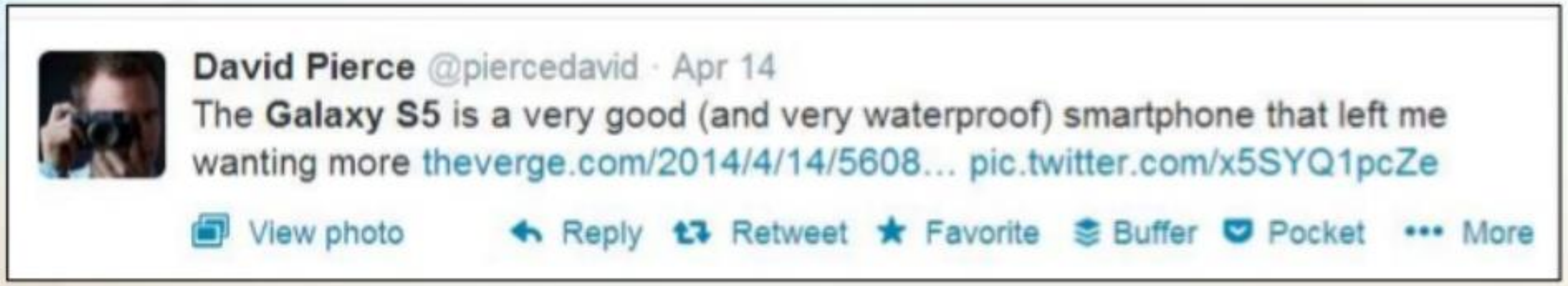
What is Sentiment Analysis?

It is classification of the polarity of a given text in the document, sentence or a phrase

The goal is to determine whether the expressed opinion in the text is positive, neutral or negative.

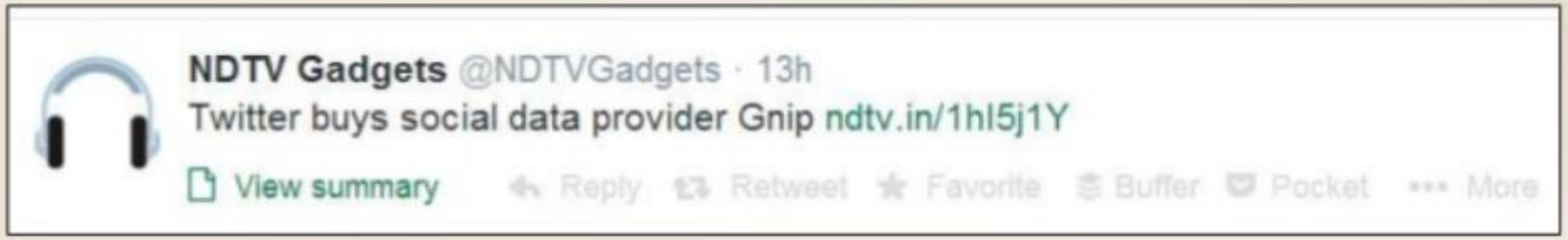
Negative


Praval Singh @Praval · 8m
Young techies leaving Infosys in droves | Attrition rate of 18.7% - bit.ly/1kwei68


David Pierce @piercedavid · Apr 14
The **Galaxy S5** is a very good (and very waterproof) smartphone that left me wanting more theverge.com/2014/4/14/5608… pic.twitter.com/x5SYQ1pcZe

Positive

Neutral


NDTV Gadgets @NDTVGadgets · 13h
Twitter buys social data provider Gnip ndtv.in/1hI5j1Y

Why is Sentiment Analysis Important ?

- Microblogging has become popular communication tool
- Opinion of the mass is important
- Political party may want to know whether people support their program or not.
- Before investing into a company, one can leverage the sentiment of the people for the company to find out where it stands.
- A company might want to find out the reviews of its products

Using Twitter for Sentiment Analysis

- Popular microblogging site
- 240+ million active users
- Twitter audience varies from common people to celebrities
- Users often discuss current affairs and share personal views on various subjects
- Tweets are small in length and hence unambiguous

Problem Statement

The problem statement at hand consists of two subtasks:

- Phrase Level Sentiment Analysis in Twitter:

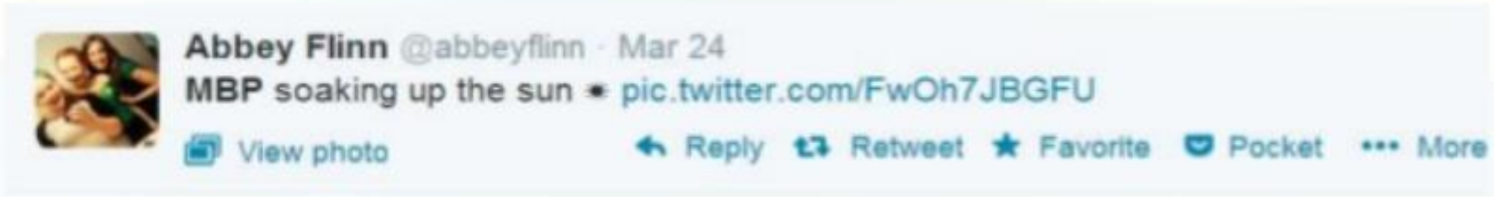
Given a message containing a marked instance of a word or a phrase, determine whether that instance is positive, negative or neutral in that context.

- Sentence Level Sentiment Analysis in Twitter:

Given a message, decide whether the message is of positive, negative or neutral sentiment. For messages conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen

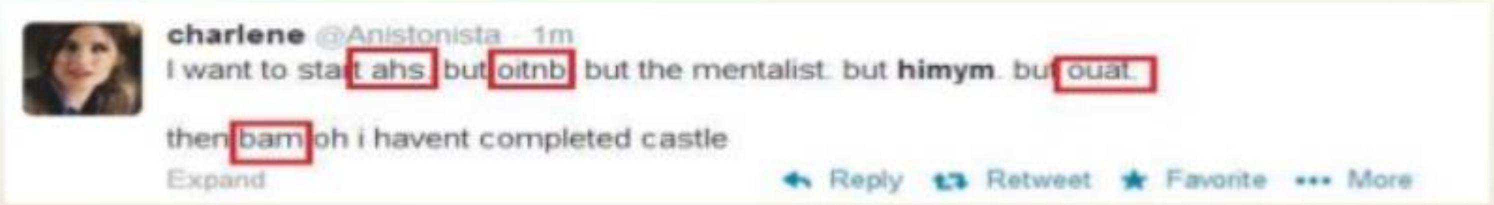
Challenges

- Tweets are highly unstructured and also non-grammatical



Abbey Flinn @abbeyflinn · Mar 24
MBP soaking up the sun ☀ pic.twitter.com/FwOh7JBGFU

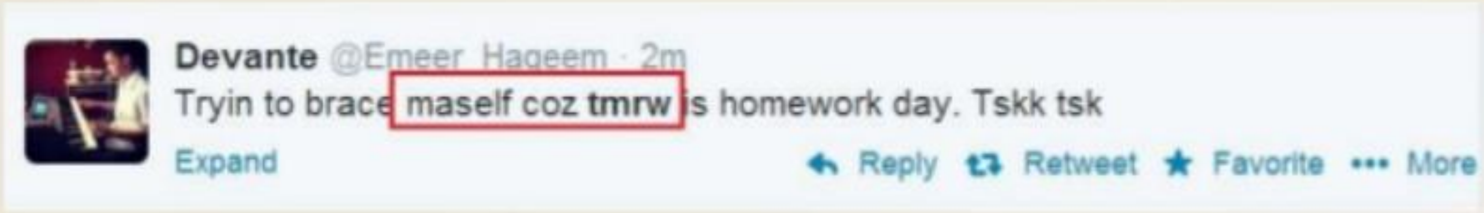
- Out of Vocabulary Words



charlene @Anistonista · 1m
I want to start ahs. but oitnb. but the mentalist. but **himym**. but ouat.

then bam oh i havent completed castle

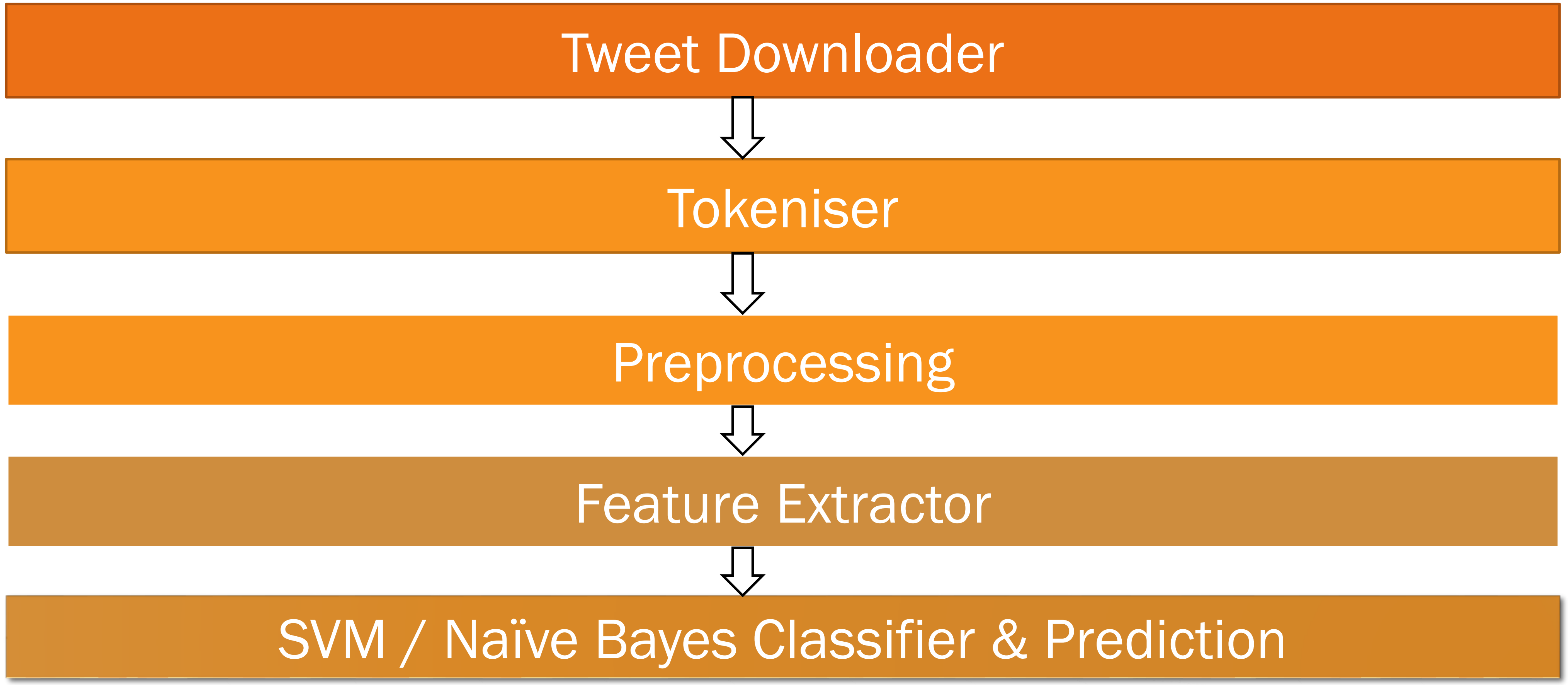
- Lexical Variation



Devante @Emeer_Hageem · 2m
Tryin to brace maself coz **tmrw** is homework day. Tskk tsk

- Extensive usage of acronyms like *asap, lol, afaik*

Approach


Tweet Downloader
Tokeniser
Preprocessing
Feature Extractor
SVM / Naïve Bayes Classifier & Prediction

Approach

- ❑Tweet Downloader

Download the tweets using Twitter API

- ❑Tokenisation

Twitter specific POS tagger developed by Social Media Search

- ❑Preprocessing

Removing Non-English Tweets, Replacing Emoticons by their polarity, Remove URL, Target Mentions, Hashtags

Remove Nouns and Prepositions, Replace sequence of repeated characters.

Approach

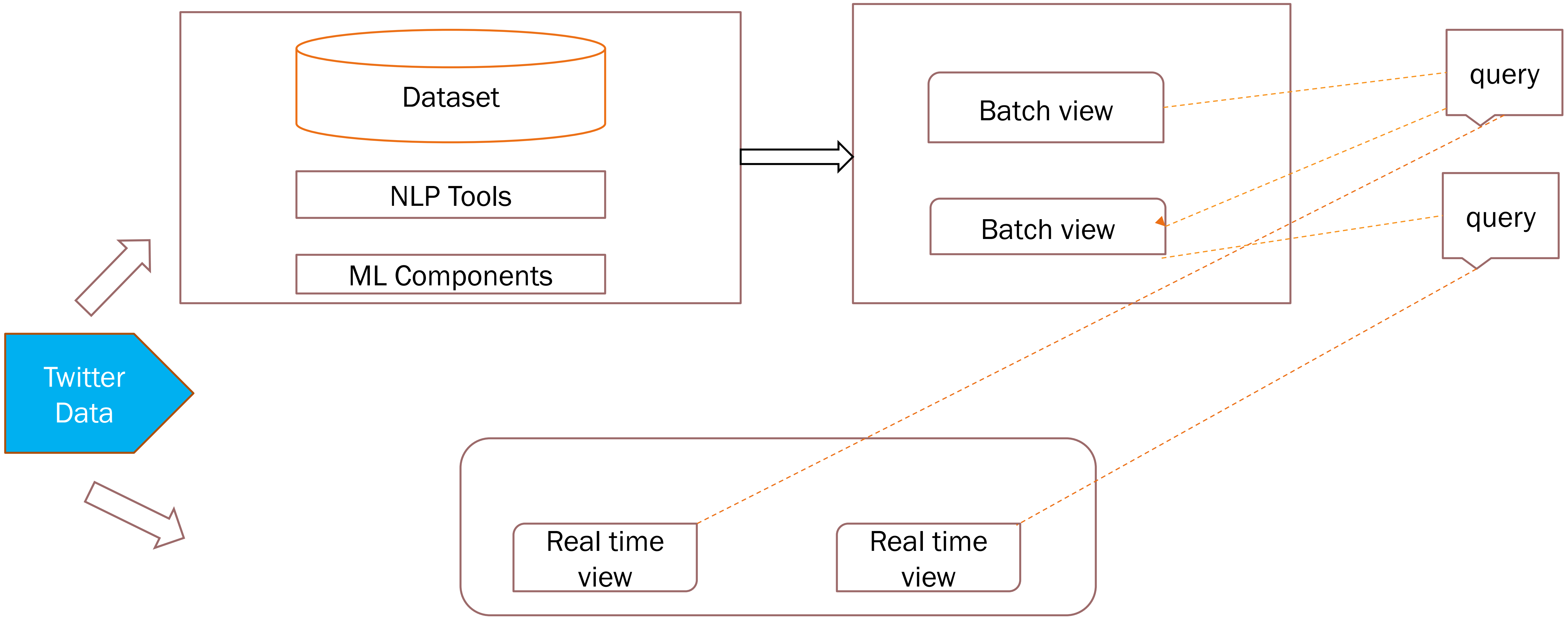
❑Feature Extractor
> Polarity Score of the tweet
> Percentage of capitalised words
> Number of positive/negative capitalised words
> Number of positive/negative hashtags
> Number of negations

❑Classifier and Prediction

The Feature extracted are next passed on to SVM / Naïve Bayes classifier

The Model built is used to predict the sentiment of the tweets.

Design Plan


Dataset
NLP Tools
ML Components
Batch view
Batch view
query
query
Twitter Data
Real time view
Real time view

Implementation Plan

Here’s what our Implementation plan will look like:

- ❑Gather relevant tweets from Twitter
- ❑Preprocessing (stopword removal)
- ❑Applying the right sentiment analysis algorithm
- ❑Analyze the results
- ❑Discuss further improvement and next steps

Extracting Tweets

We’ll start by grabbing the tweets we want from Twitter, In one of the later stages, we will be extracting numeric features from our Twitter text data. This feature space is created using all the unique words present in the entire data. So, if we preprocess our data well, then we would be able to get a better quality feature space.

Pre-processing Tweets

The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that you are working with noisy and inconsistent data. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don’t carry much weightage in context to the text.

Stopwords are words that aren’t integral to the meaning of a text, and are usually removed as part of a Natural Language Processing workflow. The tweets are now reassembled as sentences, but without stopwords removing these extra elements should give the sentiment analysis algorithm a better shot.

Applying the right sentiment analysis algorithm

Choosing which sentiment algorithm to use depends on a number of factors: you need to take into account the required level of detail, speed, cost, and accuracy among other things.

Following are the 2 algorithms shortlisted by us:

1. Naïve Bayes
2. Support Vector Machine

Naïve Bayes Algorithm

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods, Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation bel

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

How Naive Bayes algorithm works?

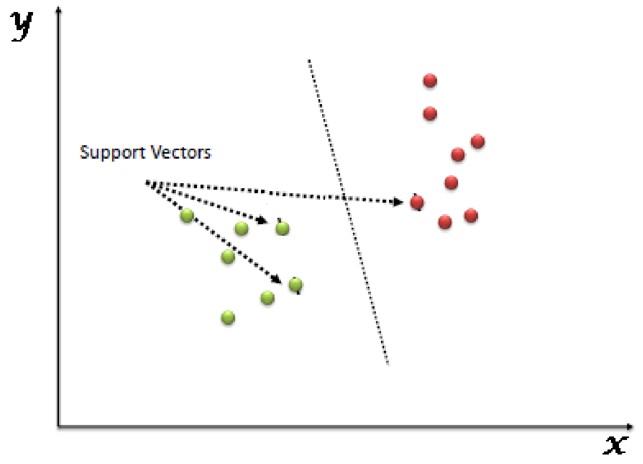
Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Support Vector Machine (SVM)

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well



How SVM Works?

Unlike other classifiers, the support vector machine is explicitly told to find the best separating line. How? The support vector machine searches for the closest points, which it calls the "support vectors" (the name "support vector machine" is due to the fact that points are like vectors and that the best line "depends on" or is "supported by" the closest points).

Once it has found the closest points, the SVM draws a line connecting them. It draws this connecting line by doing vector subtraction (point A - point B). The support vector machine then declares the best separating line to be the line that bisects -- and is perpendicular to -- the connecting line.

The support vector machine is better because when you get a new sample (new points), you will have already made a line that keeps B and A as far away from each other as possible, and so it is less likely that one will spillover across the line into the other's territory.

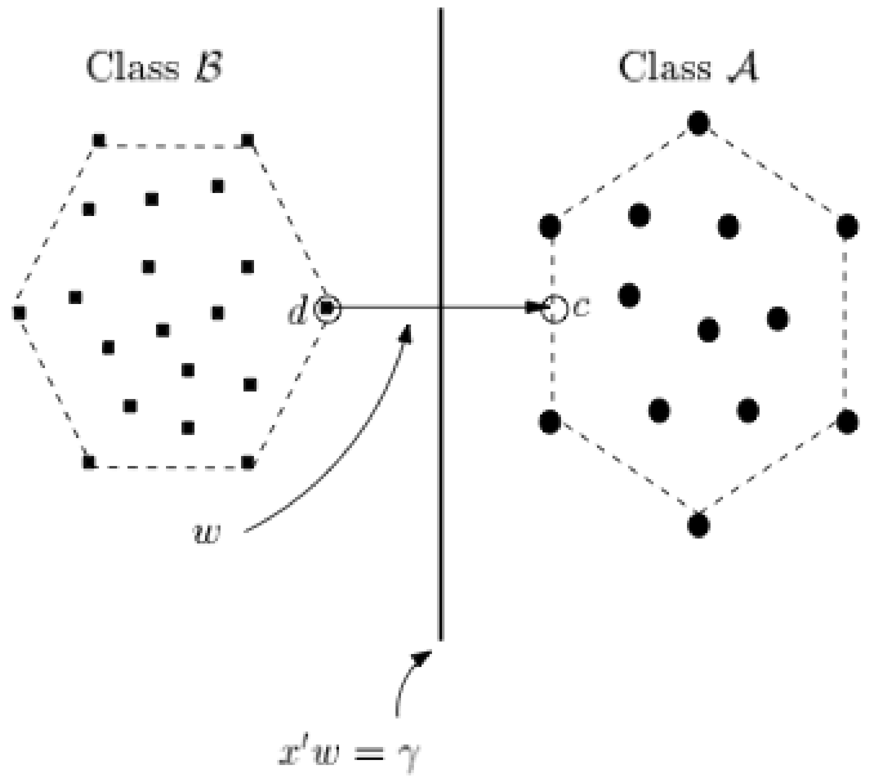
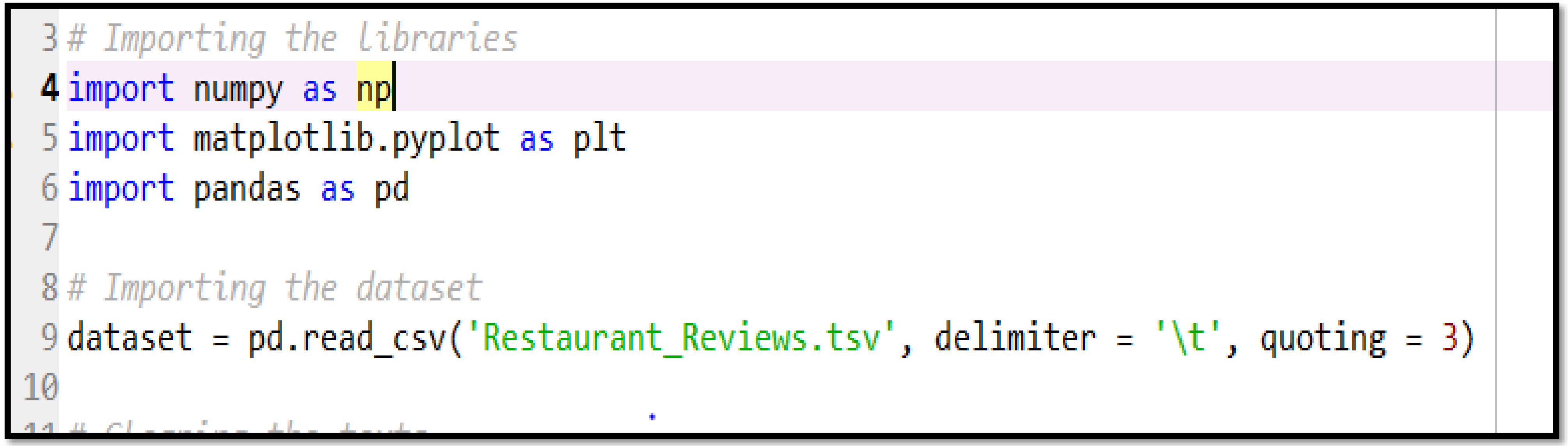


Figure 2. The two closest points of the convex hulls determine the separating plane.

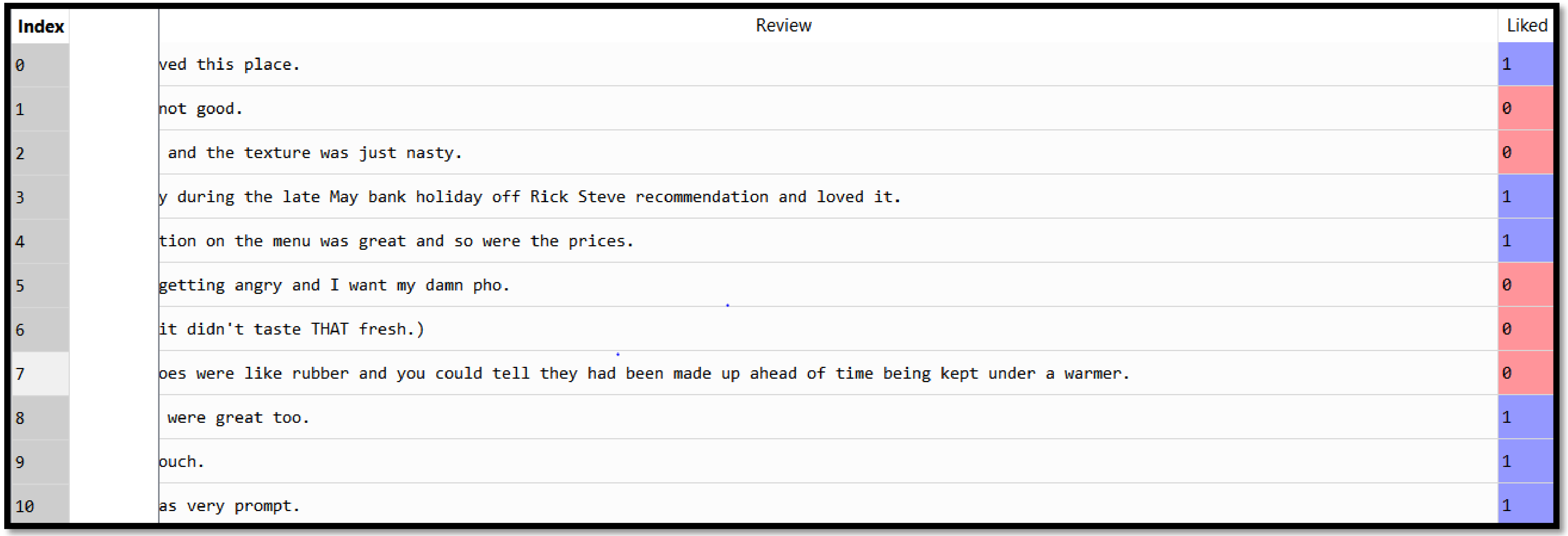
Naïve Bayes implementation



```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Restaurant_Reviews.tsv', delimiter = '\t', quoting = 3)

```

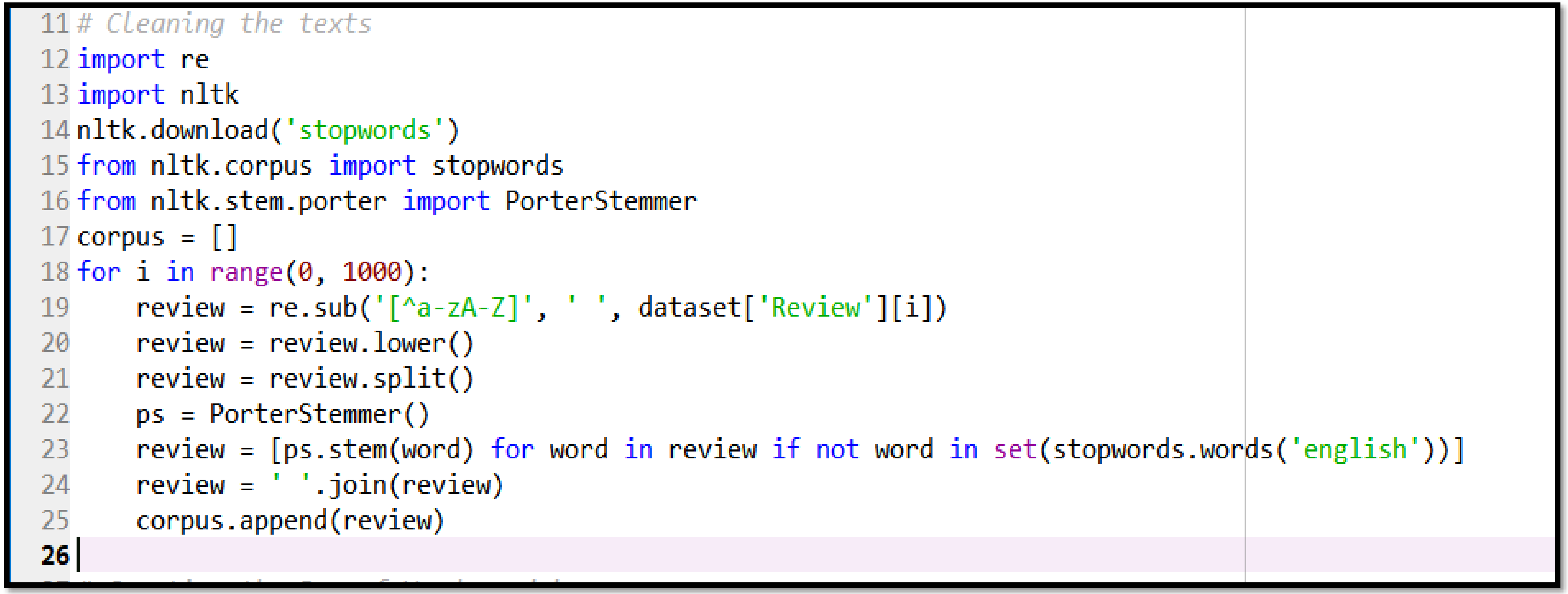
TSV tweets dataset

- CSV after coverted into TSV.



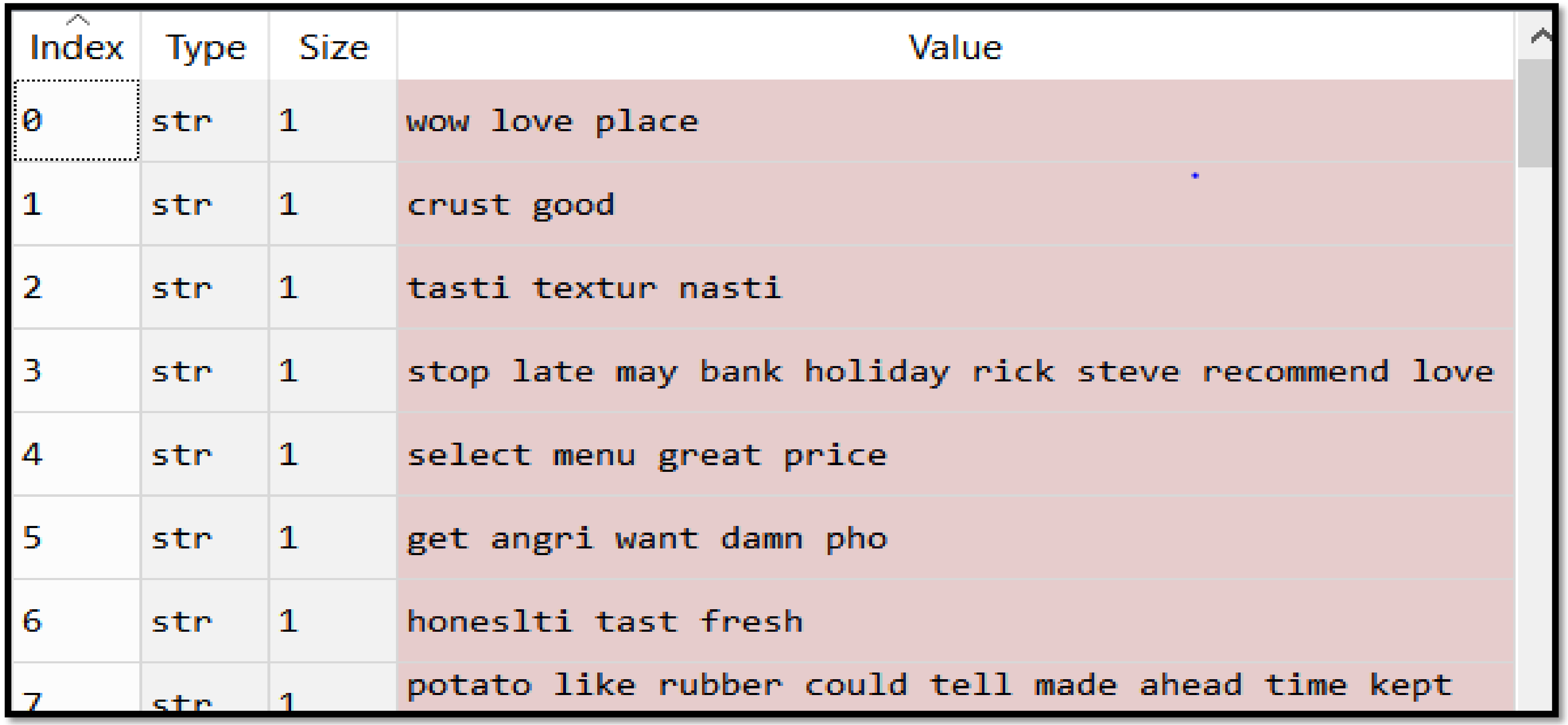
Index	Review	Liked
0	ved this place.	1
1	not good.	0
2	and the texture was just nasty.	0
3	y during the late May bank holiday off Rick Steve recommendation and loved it.	1
4	tion on the menu was great and so were the prices.	1
5	getting angry and I want my damn pho.	0
6	it didn't taste THAT fresh.)	0
7	oes were like rubber and you could tell they had been made up ahead of time being kept under a warmer.	0
8	were great too.	1
9	ouch.	1
10	as very prompt.	1

Text Cleaning and creating a corpus



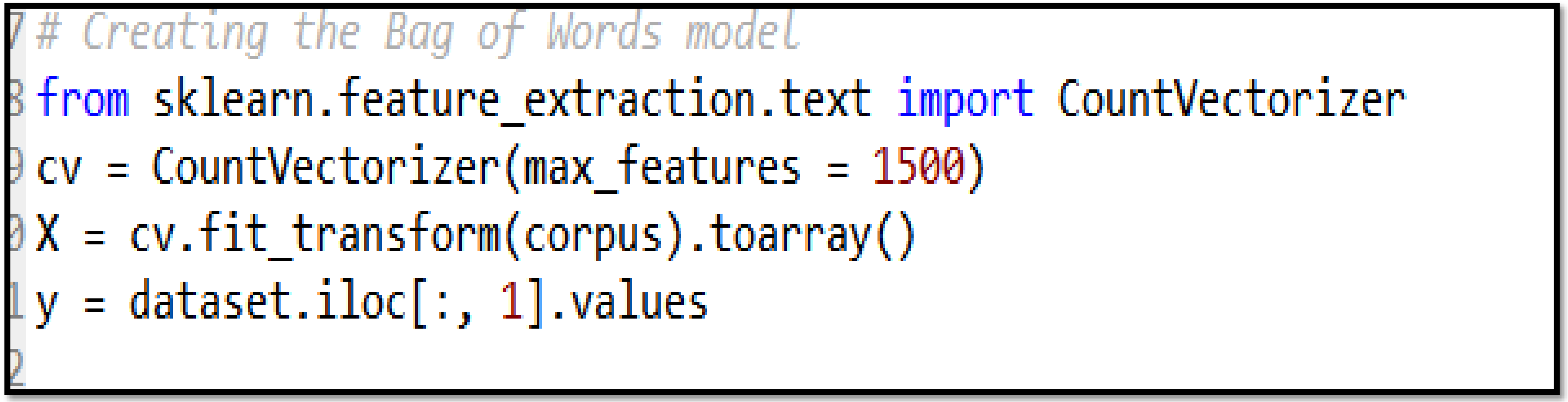
```python
# Cleaning the texts
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

```

Corpus of tweets



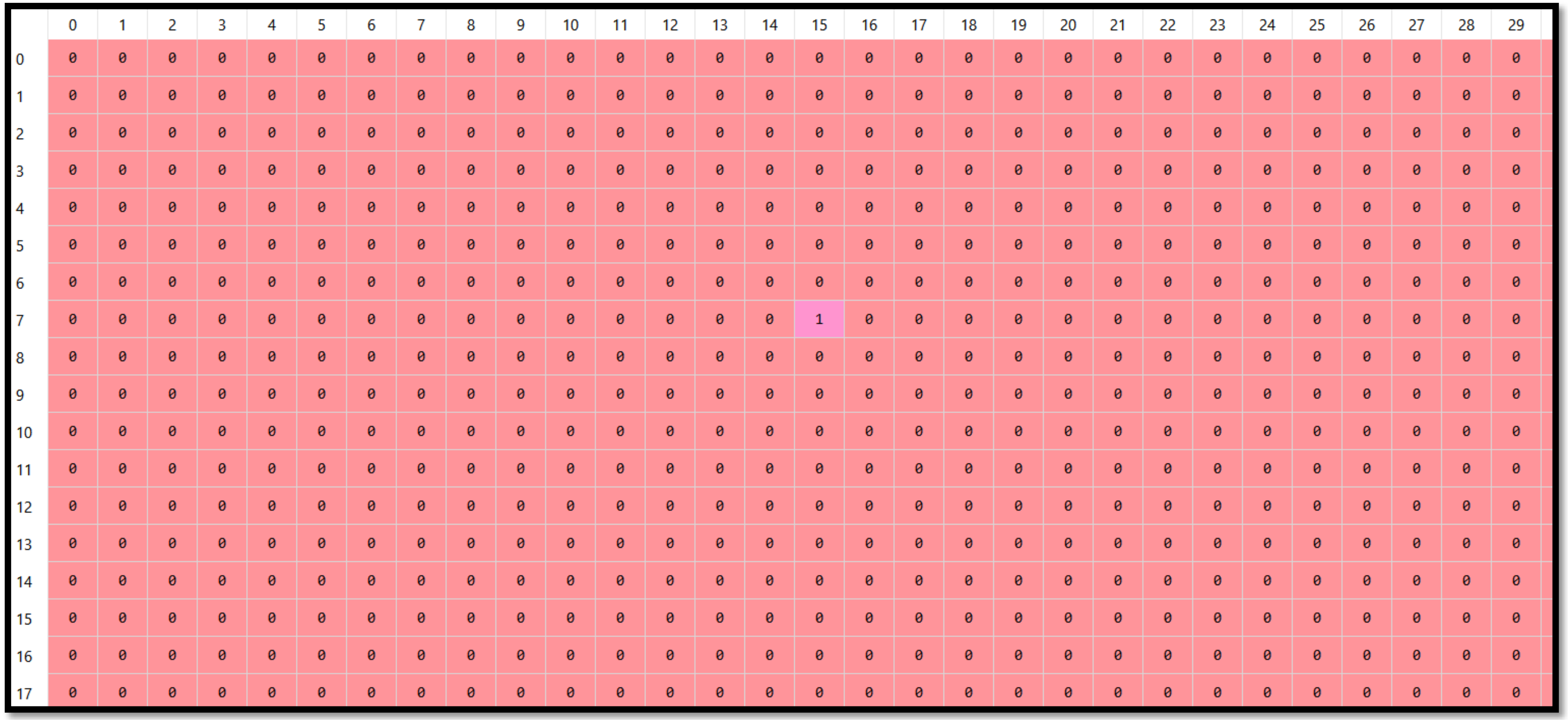
Index	Type	Size	Value
0	str	1	wow love place
1	str	1	crust good
2	str	1	tasti textur nasti
3	str	1	stop late may bank holiday rick steve recommend love
4	str	1	select menu great price
5	str	1	get angri want damn pho
6	str	1	honeslti tast fresh
7	str	1	potato like rubber could tell made ahead time kept

Creating Bag of Words



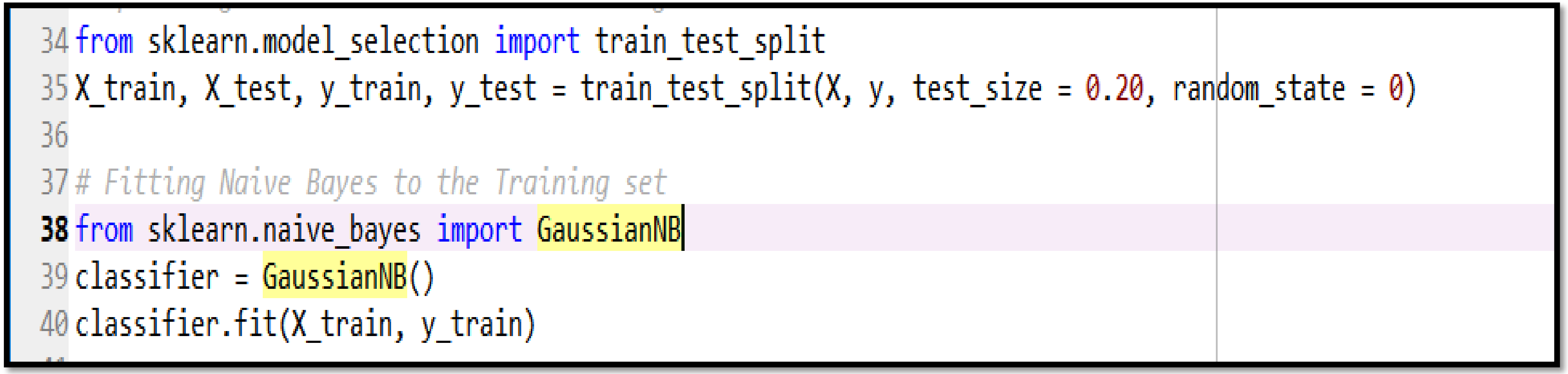
```
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values
```

Bag of words



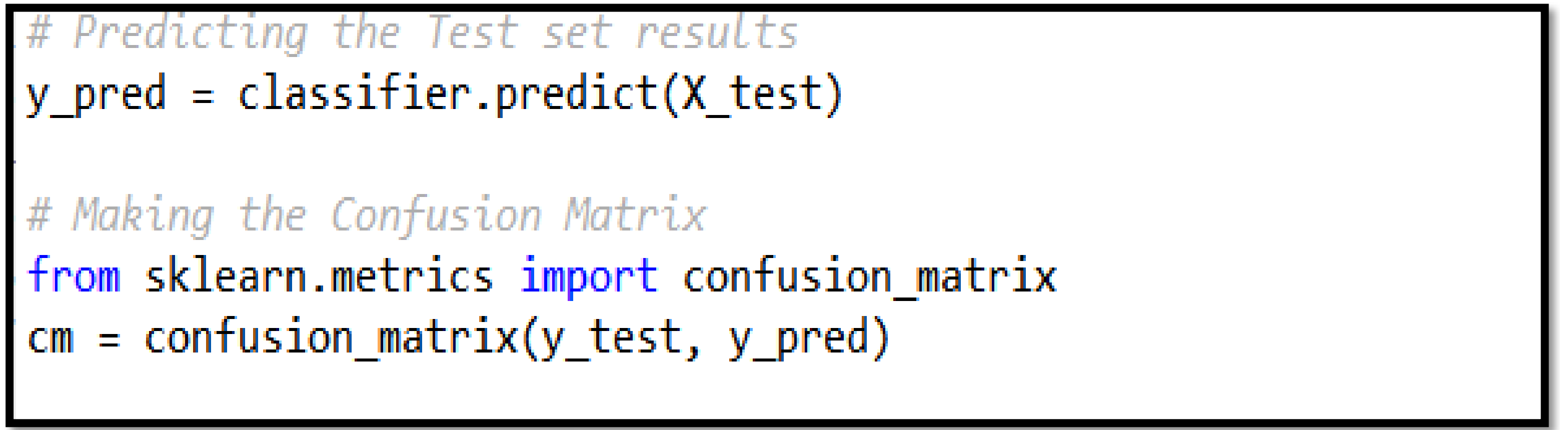
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Splitting and training

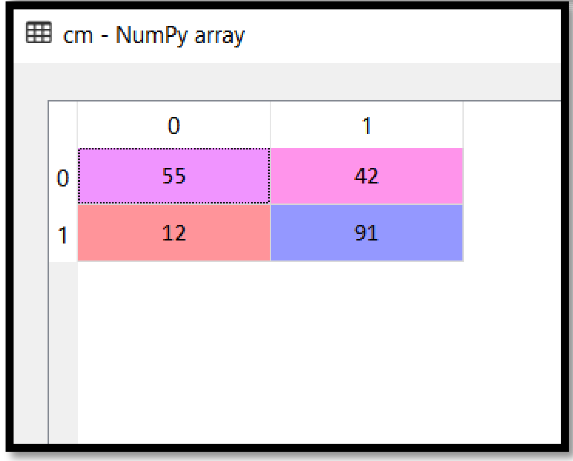


```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Prediction and confusion matrix



```
# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```



cm - NumPy array

	0	1
0	55	42
1	12	91

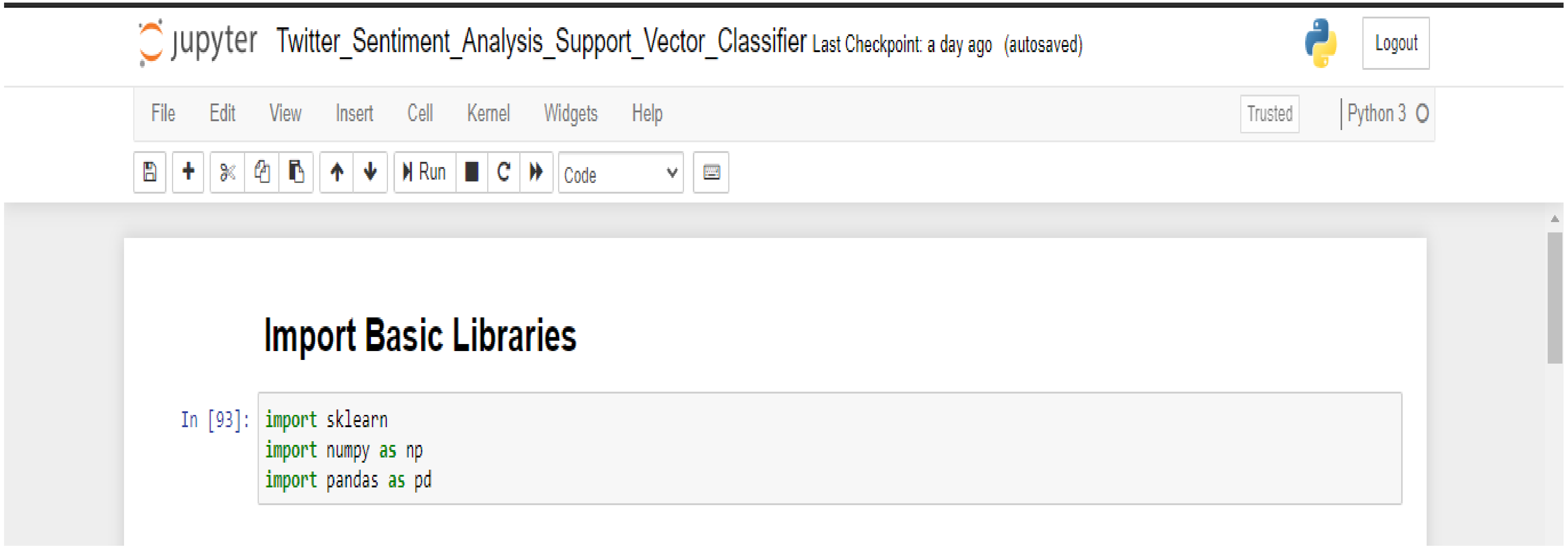
Accuracy

Total tweets considered for testing is 200.

Correct predictions are 146.hence accuracy is (146/200) x100=73%

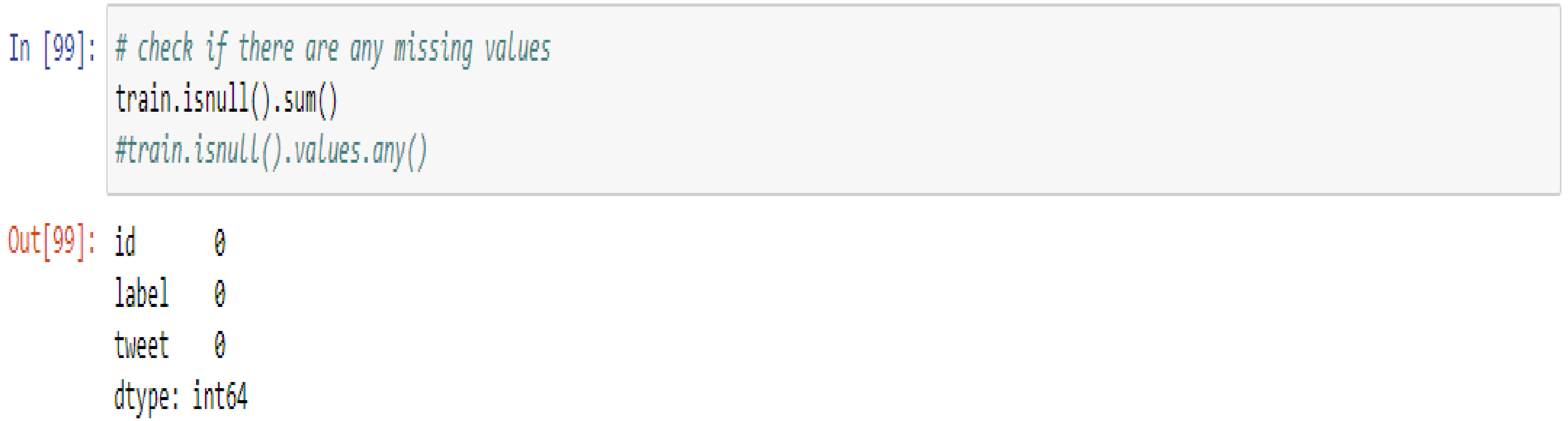
Accuracy is 73%.

SVM Implementation

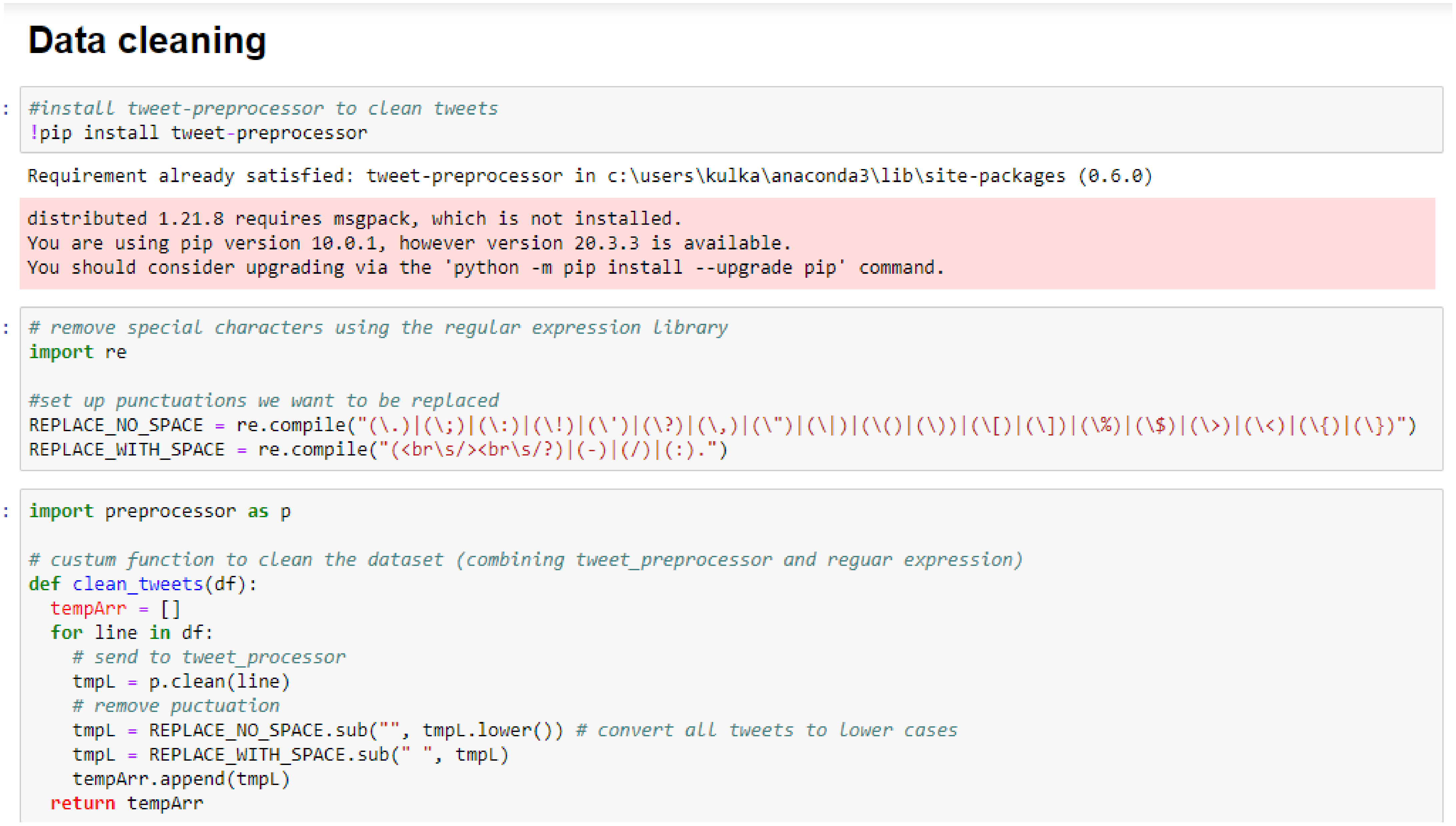

jupyter Twitter_Sentiment_Analysis_Support_Vector_Classifier Last Checkpoint: a day ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Import Basic Libraries

```
In [93]: import sklearn
         import numpy as np
         import pandas as pd
```



```
In [99]: # check if there are any missing values
         train.isnull().sum()
         #train.isnull().values.any()

Out[99]: id       0
         label    0
         tweet    0
         dtype: int64
```



Data cleaning

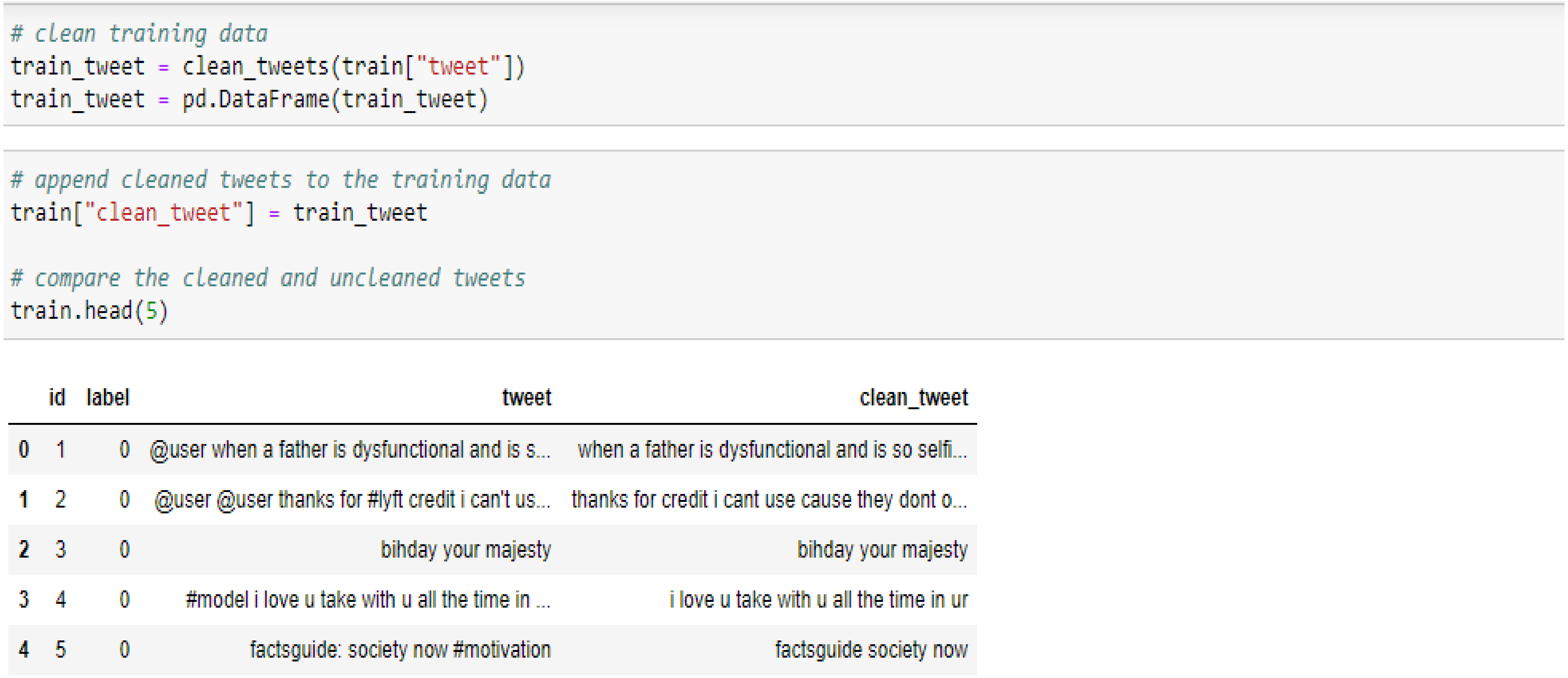
```
#install tweet-preprocessor to clean tweets
!pip install tweet-preprocessor
```

```
Requirement already satisfied: tweet-preprocessor in c:\users\kulka\anaconda3\lib\site-packages (0.6.0)
```

```
distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 20.3.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
# remove special characters using the regular expression library
import re

#set up punctuations we want to be replaced
REPLACE_NO_SPACE = re.compile("(\.)|(\;)|(\:)|(\!)|(\')|(\?)|(\,)|(\")|(\|)|(\()|(\))|(\[)|(\])|(\%)|(\$)|(\>)|(\<)|(\{)|(\})")
REPLACE_WITH_SPACE = re.compile("(<br\s/><br\s/?)|(-)|(/)|(:).")
```

```
import preprocessor as p

# custum function to clean the dataset (combining tweet_preprocessor and reguar expression)
def clean_tweets(df):
  tempArr = []
  for line in df:
    # send to tweet_processor
    tmpL = p.clean(line)
    # remove puctuation
    tmpL = REPLACE_NO_SPACE.sub("", tmpL.lower()) # convert all tweets to lower cases
    tmpL = REPLACE_WITH_SPACE.sub(" ", tmpL)
    tempArr.append(tmpL)
  return tempArr
```

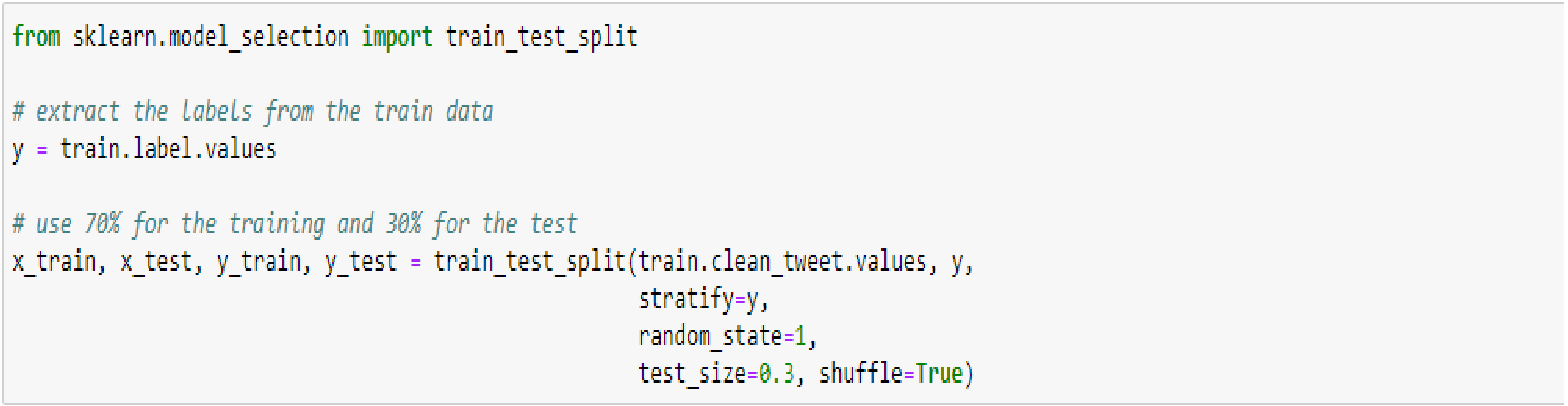


```python
# clean training data
train_tweet = clean_tweets(train["tweet"])
train_tweet = pd.DataFrame(train_tweet)
```

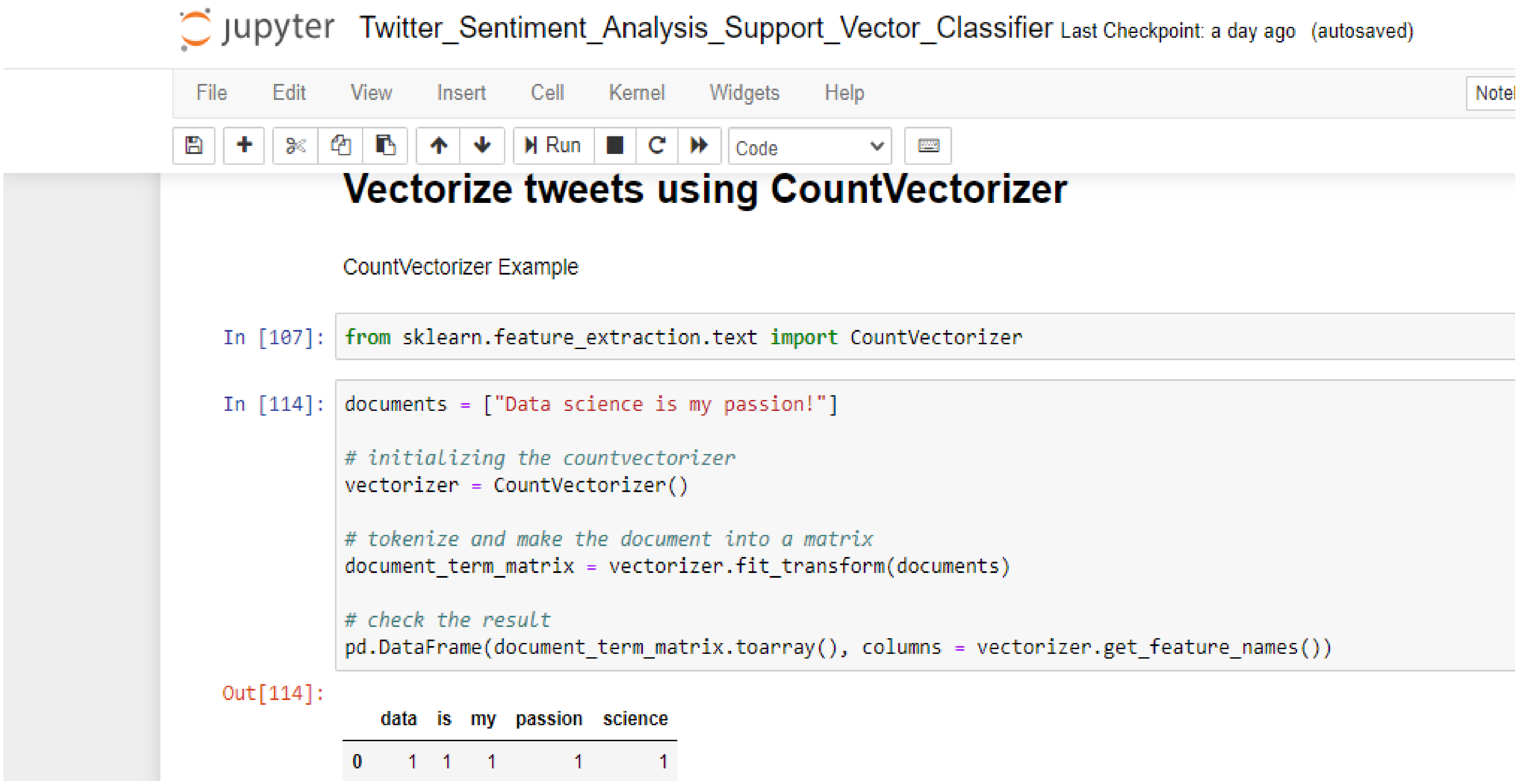
```python
# append cleaned tweets to the training data
train["clean_tweet"] = train_tweet

# compare the cleaned and uncleaned tweets
train.head(5)
```

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so selfi...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for credit i cant use cause they dont o...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	i love u take with u all the time in ur
4	5	0	factsguide: society now #motivation	factsguide society now

Test and Train split



```python
from sklearn.model_selection import train_test_split

# extract the labels from the train data
y = train.label.values

# use 70% for the training and 30% for the test
x_train, x_test, y_train, y_test = train_test_split(train.clean_tweet.values, y,
                                                    stratify=y,
                                                    random_state=1,
                                                    test_size=0.3, shuffle=True)
```


jupyter Twitter_Sentiment_Analysis_Support_Vector_Classifier Last Checkpoint: a day ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Notel

Vectorize tweets using CountVectorizer

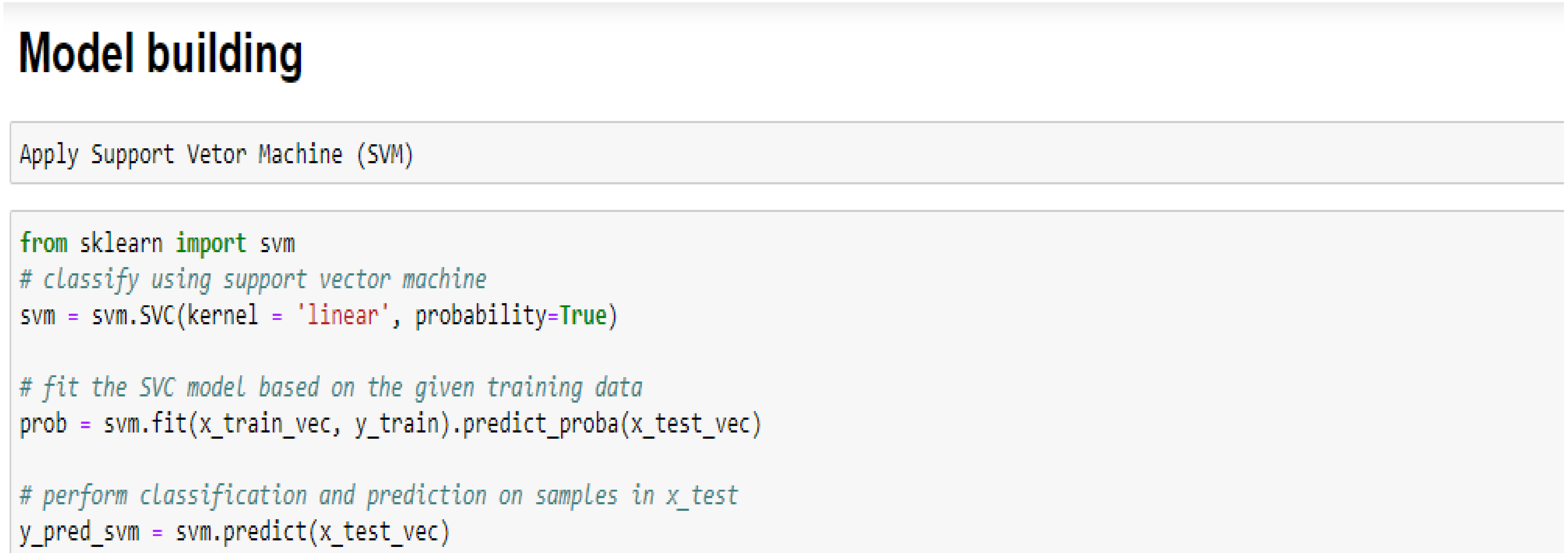
CountVectorizer Example

```
In [107]: from sklearn.feature_extraction.text import CountVectorizer
```

```
In [114]: documents = ["Data science is my passion!"]

          # initializing the countvectorizer
          vectorizer = CountVectorizer()

          # tokenize and make the document into a matrix
          document_term_matrix = vectorizer.fit_transform(documents)

          # check the result
          pd.DataFrame(document_term_matrix.toarray(), columns = vectorizer.get_feature_names())
```

Out[114]:

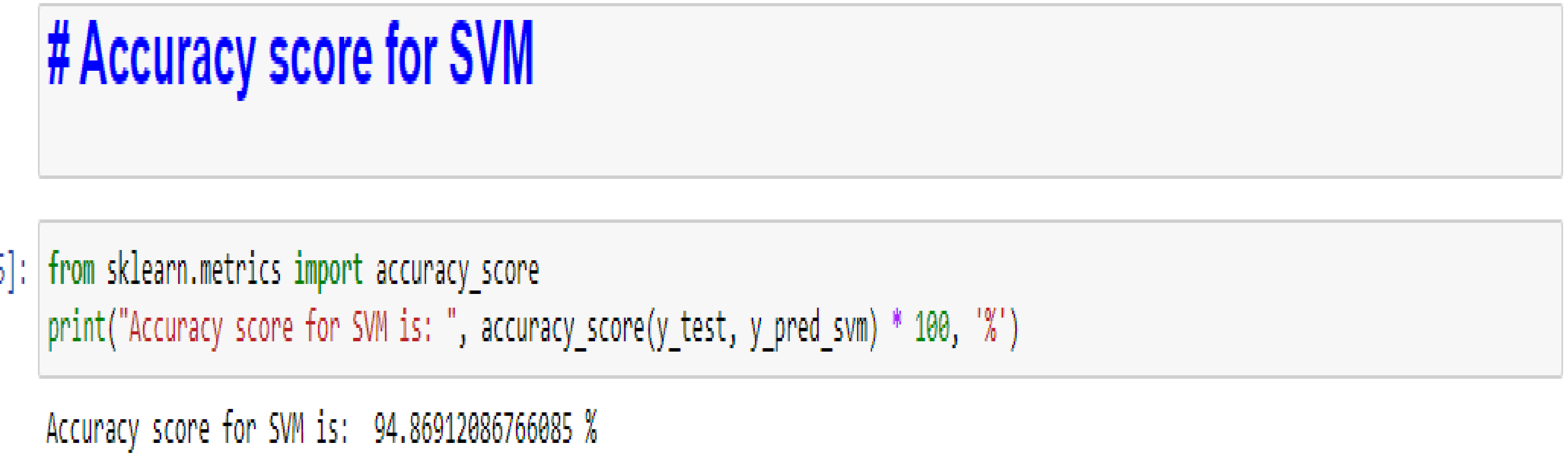
	data	is	my	passion	science
0	1	1	1	1	1



Model building

```
Apply Support Vetor Machine (SVM)
```

```python
from sklearn import svm
# classify using support vector machine
svm = svm.SVC(kernel = 'linear', probability=True)

# fit the SVC model based on the given training data
prob = svm.fit(x_train_vec, y_train).predict_proba(x_test_vec)

# perform classification and prediction on samples in x_test
y_pred_svm = svm.predict(x_test_vec)
```



```
# Accuracy score for SVM
```

```
]: from sklearn.metrics import accuracy_score
   print("Accuracy score for SVM is: ", accuracy_score(y_test, y_pred_svm) * 100, '%')
```

```
Accuracy score for SVM is:  94.86912086766085 %
```

Further Improvements and Next Steps

There are a few ways we can improve this pipeline to make it look more like something production ready.

- Gather (a lot) more tweets – the more tweets we have, the more confident we can be in our sentiment estimation.
- Explore other algorithms – depending on the business goal, other algorithms might be better suited to this type of analysis. For example, the TextBlob Python package returns a measure of subjectivity for a given string of text.
- Making a web model of the project with real time input of text and classification of its sentiment.

CONCLUSION

From our research we have concluded that Naive Bayes and Support Vector Machine are two of the best approach for our project, After analysing the final result of both of them the best of either two algorithm is SVM and the same will be selected for our further process.