Sentimental Analysis for Marketing

Phase 5

Data Set 1:

Data Set Link:

[https://www.kaggle.com/datasets/crowdflower/twitter -airline-sentiment](https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment)

The training of dataset consists of the following steps:

- **Unpacking of data:** The huge dataset of reviews obtained from amazon.com comes in a .json file format. A small python code has been implemented in order to read the dataset from those files and dump them in to a pickle file for easier and fastaccess and object serialization.

```
with open(data_file, 'r') as file_handler:
    for review in file_handler.readlines():
        df[i] = ast.literal_eval(review)
        i += 1

reviews_df = pd.DataFrame.from_dict(df, orient = 'index')
reviews_df.to_pickle('reviews_digital_music.pickle')
```

  Hence initial fetching of data is done in this section using Python File Handlers.

- **Preparing Data for Sentiment Analysis:**
  - **i)** The pickle file is hence loaded in this step and the data besides the one used for sentiment analysis is removed. As shown in our sample dataset in Page 11, there are a lot of columns in the data out of which only rating and text review is what we require. So, the column, “reviewSummary” is dropped from the data file.
  - **ii)** After that, the review ratings which are 3 out of 5 are removed as they signify neutral review, and all we are concerned of is positive and negative reviews.
  - **iii)** The entire task of preprocessing the review data is handled by this

```python
reviews_df.drop(columns = ['reviewSummary'], inplace = True)
reviews_df['reviewRating'] = reviews_df.reviewRating.astype('int')
```

```python
reviews_df = reviews_df[reviews_df.reviewRating != 3] # Ignoring 3-star reviews -> neutral
reviews_df = reviews_df.assign(sentiment = np.where(reviews_df['reviewRating'] >= 4, 1, 0)) # 1 -> Positive, 0 -> Negati
```

utility class- “NltkPreprocessor”.

```python
class NltkPreprocessor:

    def __init__(self, stopwords = None, punct = None, lower = True, strip = True):
        self.lower = lower
        self.strip = strip
        self.stopwords = stopwords or set(sw.words('english'))
        self.punct = punct or set(string.punctuation)
        self.lemmatizer = WordNetLemmatizer()

    def tokenize(self, document):
        tokenized_doc = []

        for sent in sent_tokenize(document):
            for token, tag in pos_tag(wordpunct_tokenize(sent)):
                token = token.lower() if self.lower else token
                token = token.strip() if self.strip else token
                token = token.strip('_0123456789') if self.strip else token
                # token = re.sub(r'\d+', '', token)

                if token in self.stopwords:
                    continue

                if all(char in self.punct for char in token):
                    continue

                lemma = self.lemmatize(token, tag)
                tokenized_doc.append(lemma)

        return tokenized_doc

    def lemmatize(self, token, tag):
        tag = {
            'N': wn.NOUN,
            'V': wn.VERB,
            'R': wn.ADV,
            'J': wn.ADJ
        }.get(tag[0], wn.NOUN)

        return self.lemmatizer.lemmatize(token, tag)
```

iv) The time required to prepare the following data is hence displayed.

```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.163 s
```

The time taken to preprocess the data is calculated and displayed

- **Preprocessing Data:**This is a vital part of training the dataset. Here Words present in the file are accessed both as a solo word and also as pair of words. Because, for example the word “bad” means negative but when someone writes “not bad” it refers to as positive. In such cases considering single word for training data will work otherwise. So words in pairs are checked to find the occurrence to modifiers before any adjective which if present which might provide a different meaning to the outlook.

```
X = reviews_df_preprocessed.iloc[:, -1].values
y = reviews_df_preprocessed.iloc[:, -2].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

- **Training Data/ Evaluation:**The main chunk of code that does the whole evaluation of sentimental analysis based on the preprocessed data is a part of this. The following are the steps followed:

```
pipeline = Pipeline([
            ('vect', TfidfVectorizer(ngram_range = (1,2), stop_words = 'english', sublinear_tf = True)),
            ('chi', SelectKBest(score_func = chi2, k = 50000)),
            ('clf', LinearSVC(C = 1.0, penalty = 'l1', max_iter = 3000, dual = False, class_weight='balanced'))
        ])

model = pipeline.fit(X_train, y_train)
```

i) The Accuracy, Precision, Recall, and Evaluation time is calculated and displayed.

ii) Navie Bayes, Logistic Regression, Linear SVM and Random forest classifiers are applied on the dataset for evaluation of sentiments.

iii) Prediction of test data is done and Confusion Matrix of prediction isdisplayed. **iv)** Total positive and negative reviews are counted.

v) A review like sentence is taken as input on the console and if positive the console gives 1 as output and 0 for negative input.

Results and Sample Output

The ultimate outcome of this Training of Public reviews dataset is that, the machine is capable of judging whether an entered sentence bears positive response or negative response.

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while **Recall** (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Both precision and recall are therefore based on an understanding and measure of relevance.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

F_1 score (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F_1 score is the harmonic average of the precision and recall, where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

In statistics, a **receiver operating characteristic curve**, i.e. **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The Total Operating Characteristic (TOC) expands on the idea of ROC by showing the total information in the two-by-two contingency table for each threshold. ROC gives only two bits of relative information for each threshold, thus the TOC gives strictly more information than the ROC.

When using normalized units, the area under the curve (often referred to as simply the AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). This can be seen as follows: the area under the curve is given by (the integral boundaries are reversed as large T has a lower value on the x-axis).

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

The machine evaluates the accuracy of training the data along with precision Recall and F_1

The Confusion matrix of evaluation is calculated.

It is thus capable of judging an externally written review as positive or negative.

A positive review will be marked as [1], and a negative review will be hence marked as [0].

Results obtained using Hold-out Strategy(Train-Test split) [values rounded upto 2 decimal places]**.**

Name of classifier	F_1	Accuracy	Precision	Recall	ROC AUC
Multinomial NB	**85.25%**	**85.31%**	**85.56%**	**84.95%**	**85.31%**
Logistic Regression	**88.12%**	**88.05%**	**87.54%**	**88.72%**	**88.05%**
Linear SVC	**88.12%**	**88.11%**	**87.59%**	**88.80%**	**88.11%**
Random Forest	**82.43%**	**81.82%**	**79.74%**	**85.30%**	**81.83%**

The Confusion Matrix Format is as follows:

True Negative	False Positive
False Negative	True Positive

The Confusion Matrix of Each Classifier are as follows:

68556	11470
12032	67942

Classifier 1: Multinomial NB

69928	10098
9023	70951

Classifier 2: Logistic Regression

69963	10063
8955	17019

Classifier 3: Liner SVC

62695	17331
11749	68225

Classifier 4: Random Forest

The following are the images of such sample output after successful dataset training using the classifiers:

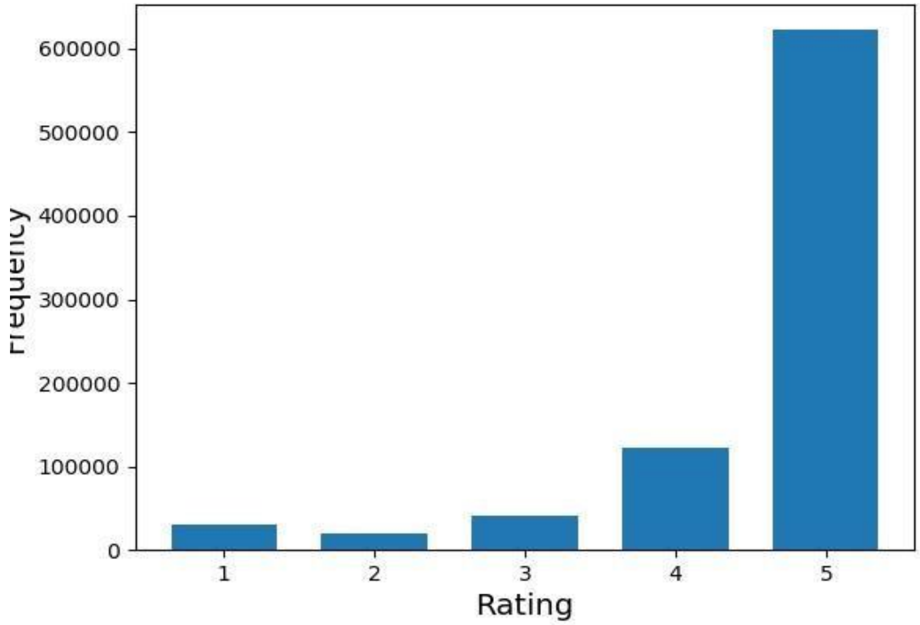
```
Pranits-MacBook-Air:sentiment-analysis pranit$ python3 sentiment_analyzer.py

Holdout Strategy...

Splitting data using Train-Test split...
Splitting data completed!
Splitting time:  0.201 s

Training data... Classifier MNB
Training data completed!
Training time:  183.1 s

Training data... Classifier LR
Training data completed!
Training time:  217.264 s

Training data... Classifier SVM
Training data completed!
Training time:  204.015 s

Training data... Classifier RF
Training data completed!
Training time:  719.168 s

Predicting Test data... Classifier MNB
Prediction completed!
Prediction time:  28.198 s

Predicting Test data... Classifier LR
Prediction completed!
Prediction time:  27.013 s

Predicting Test data... Classifier SVM
Prediction completed!
Prediction time:  27.175 s

Predicting Test data... Classifier RF
Prediction completed!
Prediction time:  39.286 s

Evaluating results... Classifier MNB
Results evaluated!
Evaluation time:  0.34 s

Evaluating results... Classifier LR
Results evaluated!
Evaluation time:  0.325 s

Evaluating results... Classifier SVM
Results evaluated!
Evaluation time:  0.318 s

Evaluating results... Classifier RF
Results evaluated!
```

```
Evaluating results... Classifier MNB
Results evaluated!
Evaluation time:  0.34 s

Evaluating results... Classifier LR
Results evaluated!
Evaluation time:  0.325 s

Evaluating results... Classifier SVM
Results evaluated!
Evaluation time:  0.318 s

Evaluating results... Classifier RF
Results evaluated!
Evaluation time:  0.315 s

Evaluation metrics of classifier MNB
Accuracy: 0.8531125
Precision: 0.8555633909232861
Recall: 0.8495511041088354
f1: 0.852546647760782
ROC AUC: 0.8531113429223856
Confusion Matrix: [[68556 11470]
 [12032 67942]]
Evaluation metrics of classifier LR
Accuracy: 0.88049375
Precision: 0.8754087033769695
Recall: 0.8871758321454473
f1: 0.8812529887034772
ROC AUC: 0.8804959209711317
Confusion Matrix: [[69928 10098]
 [ 9023 70951]]
Evaluation metrics of classifier SVM
Accuracy: 0.8811375
Precision: 0.8758910732345033
Recall: 0.8880261084852578
f1: 0.8819168487979336
ROC AUC: 0.8811397380703848
Confusion Matrix: [[69963 10063]
 [ 8955 71019]]
Evaluation metrics of classifier RF
Accuracy: 0.81825
Precision: 0.7974309224367666
Recall: 0.8530897541701052
f1: 0.8243218751887875
ROC AUC: 0.8182613192413517
Confusion Matrix: [[62695 17331]
 [11749 68225]]

Total number of observations: 160000
Positives in observation: 79974
Negatives in observation: 80026
Majority class is: 50.01624999999999%
Pranits-MacBook-Air:sentiment-analysis pranit$
```

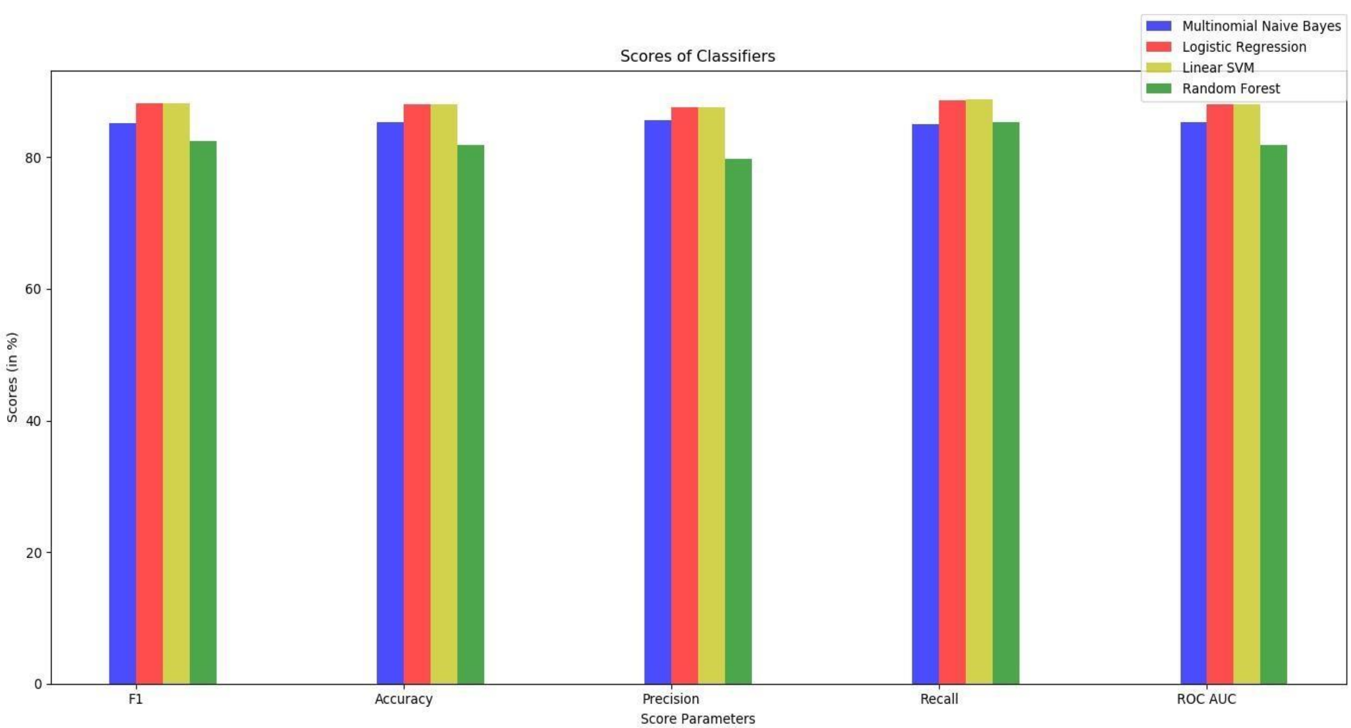
```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$ python3 sentiment_analyzer.py
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.131 s

Training data...
Training data completed!
Training time:  244.431 s

Predicting Test data...
Prediction completed!
Prediction time:  11.46 s

Evaluating results...
Accuracy: 0.94855693908754
Precision: 0.983433383243815
Recall: 0.9613014112497147
f1: 0.9722414612616284
Results evaluated!
Evaluation time:  0.084 s

Confusion matrix: [[  7575   2412]
 [  5764 143182]]

Total number of observations: 158933
Positives in observation: 148946
Negatives in observation: 9987
Majority class is: 93.7162200424078%
Worst product ever
[0]
```

```
administrator@administrator-OptiPlex-3040:~/Desktop/sentiment_analysis$ python3 sentiment_analyzer.py
Preprocessing data...
Preprocessing data completed!
Preprocessing time:  0.163 s

Training data...
Training data completed!
Training time:  239.406 s

Predicting Test data...
Prediction completed!
Prediction time:  11.402 s

Evaluating results...
Accuracy: 0.9486261506420944
Precision: 0.983467838868093
Recall: 0.9613416943053187
f1: 0.9722789017488227
Results evaluated!
Evaluation time:  0.086 s

Confusion matrix: [[  7580   2407]
 [  5758 143188]]

Total number of observations: 158933
Positives in observation: 148946
Negatives in observation: 9987
Majority class is: 93.7162200424078%
not a good product
[1]
```

The Bar Graph showing the Frequency of Ratings in the dataset



This Bar graph shows the score of each classifier after successful training. The parameters be: F_1 Score, Accuracy, Precision, Recall and Roc-Auc.



Program

Code:

Loading the dataset:

```
import json import pickle import

numpy as np from matplotlib import

pyplot as plt from textblob import

TextBlob


# fileHandler = open('datasets/reviews_digital_music.json', 'r')

# reviewDatas = fileHandler.read().split('\n')

# reviewText = []

# reviewRating = []


# for review in reviewDatas:
#         if review == "":
#                 continue
#         r = json.loads(review)
#         reviewText.append(r['reviewText'])
#         reviewRating.append(r['overall'])

# fileHandler.close()

# saveReviewText = open('review_text.pkl', 'wb')

# saveReviewRating = open('review_rating.pkl','wb')  #

pickle.dump(reviewText, saveReviewText) #

pickle.dump(reviewRating, saveReviewRating)

reviewTextFile = open('review_text.pkl', 'rb')

reviewRatingFile = open('review_rating.pkl', 'rb')
```

```
reviewText = pickle.load(reviewTextFile)
reviewRating = pickle.load(reviewRatingFile)
# print(len(reviewText))
# print(reviewText[0])
# print(reviewRating[0]) # ratings = np.array(reviewRating)   plt.hist(ratings,
bins=np.arange(ratings.min(), ratings.max()+2)-0.5, rwidth=0.7)
plt.xlabel('Rating', fontsize=14)  plt.ylabel('Frequency', fontsize=14)
plt.title('Histogram of Ratings', fontsize=18) plt.show() lang = {} i = 0 for review
in reviewText:
        tb = TextBlob(review)
l = tb.detect_language()   if l
!= 'en':
                lang.setdefault(l, [])
lang[l].append(i)
 print(i, l)          i += 1 print(lang)
```

Scrapping data:

```
from selenium import webdriver from
selenium.webdriver.chrome.options import Options from
bs4 import BeautifulSoup import openpyxl class Review():
def __init__(self):
                self.rating=""
                self.info=""
                self.review=""
def scrape():
  options = Options()    options.add_argument("--headless") # Runs Chrome in headless
mode.        options.add_argument('--no-sandbox')  #  #  Bypass  OS  security  model
options.add_argument('start-maximized')            options.add_argument('disableinfobars')
```

```
options.add_argument("--disable-extensions")
driver=webdriver.Chrome(executable_path=r'C:\chromedriver\chromedriver.exe')
        url='https://www.amazon.com/Moto-PLUS-5th-Generation-Exclusive/product-
reviews/B0785NN142/ref=cm_cr_arp_d_paging_btm_2?ie=UTF8&reviewerType=all_reviews&pageNumb
er=5'
        driver.get(url)

  soup=BeautifulSoup(driver.page_source,'lxml')
ul=soup.find_all('div',class_='a-section             review')
review_list=[]   for d in ul:
                a=d.find('div',class_='a-row')     sib=a.findNextSibling()
                b=d.find('div',class_='a-row a-spacing-medium review-data')
                '''print sib.text'''
                new_r=Review()
 new_r.rating=a.text                   new_r.info=sib.text
new_r.review=b.text

                review_list.append(new_r)
driver.quit()       return review_list def main():
        m = scrape()
        i=1  for  r  in
        m:

                book = openpyxl.load_workbook('Sample.xlsx')                          sheet =
book.get_sheet_by_name('Sample Sheet')                       sheet.cell(row=i, column=1).value = r.rating
sheet.cell(row=i, column=1).alignment = openpyxl.styles.Alignment(horizontal='center', vertical='center',
wrap_text=True)
   sheet.cell(row=i, column=3).value = r.info      sheet.cell(row=i,
column=3).alignment =
```

```
openpyxl.styles.Alignment(horizontal='center', vertical='center', wrap_text=True)
sheet.cell(row=i, column=5).value = r.review.encode('utf-8')                    sheet.cell(row=i,
column=5).alignment = openpyxl.styles.Alignment(horizontal='center', vertical='center', wrap_text=True)
                    book.save('Sample.xlsx')
                    i=i+1                 if
__name__ == '__main__':
    main()
```

Preprocessing Data:

```
import string from nltk.corpus import stopwords as sw from nltk.corpus import wordnet
as wn from nltk import wordpunct_tokenize from nltk import sent_tokenize from nltk import
WordNetLemmatizer from nltk import pos_tag class NltkPreprocessor:   def __init__(self,
stopwords = None, punct = None, lower = True, strip = True):
 self.lower = lower                      self.strip = strip
                    self.stopwords = stopwords or set(sw.words('english'))
                    self.punct = punct or set(string.punctuation)
                    self.lemmatizer = WordNetLemmatizer()

          def tokenize(self, document):
tokenized_doc = []

                    for sent in sent_tokenize(document):                           for token, tag in
pos_tag(wordpunct_tokenize(sent)):                                  token = token.lower() if
self.lower else token                                  token = token.strip() if self.strip else token
                    token = token.strip('_0123456789') if self.strip else token
                              # token = re.sub(r'\d+', '', token)
                         if token in self.stopwords:
                                   continue
if all(char in self.punct for char in token):
                                   continue
```

```
lemma = self.lemmatize(token, tag)
tokenized_doc.append(lemma)

return tokenized_doc
def lemmatize(self, token, tag):
tag = {
'N': wn.NOUN,
'V': wn.VERB,
'R': wn.ADV,
'J': wn.ADJ
}.get(tag[0], wn.NOUN)
return self.lemmatizer.lemmatize(token, tag)
```

Sentiment Analysis:

```
import ast import numpy as np import pandas as pd
import re from nltk.corpus import stopwords from nltk.stem
import SnowballStemmer from sklearn.model_selection
import train_test_split
from sklearn.feature_selection import SelectKBest, chi2, SelectPercentile, f_classif from
sklearn.feature_extraction.text import TfidfVectorizer from sklearn.pipeline import Pipeline from
sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
confusion_matrix from sklearn.svm import LinearSVC # from textblob import TextBlob from time import
time

def getInitialData(data_file):
print('Fetching initial data...')
t = time()

i = 0   df = {}    with open(data_file, 'r') as
file_handler:                for review in
file_handler.readlines():
```

```
df[i] = ast.literal_eval(review)
        i += 1  reviews_df =
pd.DataFrame.from_dict(df, orient =
'index')

 reviews_df.to_pickle('reviews_digital_music.pickle')  print('Fetching data completed!')  print('Fetching time:
', round(time()-t, 3), 's\n')



# def filterLanguage(text):
#       text_blob = TextBlob(text)
#       return text_blob.detect_language()


def     prepareData(reviews_df):
print('Preparing   data...')      t   =
time()


    reviews_df.rename(columns = {"overall" : "reviewRating"}, inplace=True)
 reviews_df.drop(columns = ['reviewerID', 'asin', 'reviewerName', 'helpful', 'summary', 'unixReviewTime',
'reviewTime'], inplace = True)



    reviews_df = reviews_df[reviews_df.reviewRating != 3.0] # Ignoring 3-star reviews -> neutral
 reviews_df = reviews_df.assign(sentiment = np.where(reviews_df['reviewRating'] >= 4.0, 1, 0)) # 1 ->
Positive, 0 -> Negative


    stemmer = SnowballStemmer('english')
stop_words = stopwords.words('english')


    # print(len(reviews_df.reviewText))
    # filterLanguage = lambda text: TextBlob(text).detect_language()
```

```
# reviews_df = reviews_df[reviews_df['reviewText'].apply(filterLanguage) == 'en']
# print(len(reviews_df.reviewText))

reviews_df = reviews_df.assign(cleaned = reviews_df['reviewText'].apply(lambda text: ' '.join([stemmer.stem(w) for w in re.sub('[^a-z]+|(quot)+', ' ', text.lower()).split() if w not in stop_words])))
reviews_df.to_pickle('reviews_digital_music_preprocessed.pickle')

print('Preparing data completed!')
print('Preparing time: ', round(time()-t, 3), 's\n')

def preprocessData(reviews_df_preprocessed):
print('Preprocessing data...')  t =
time()

X = reviews_df_preprocessed.iloc[:, -1].values
y = reviews_df_preprocessed.iloc[:, -2].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

print('Preprocessing data completed!')
print('Preprocessing time: ', round(time()-t, 3), 's\n')

return X_train, X_test, y_train, y_test

def evaluate(y_test, prediction):
print('Evaluating results...')
t = time()

print('Accuracy: {}'.format(accuracy_score(y_test, prediction)))
print('Precision: {}'.format(precision_score(y_test, prediction)))
```

```
print('Recall: {}'.format(recall_score(y_test, prediction)))            print('f1:
{}'.format(f1_score(y_test, prediction)))

            print('Results evaluated!')
 print('Evaluation time: ', round(time()-t, 3), 's\n')

# getInitialData('datasets/reviews_digital_music.json')
# reviews_df = pd.read_pickle('reviews_digital_music.pickle')

# prepareData(reviews_df) reviews_df_preprocessed =
pd.read_pickle('reviews_digital_music_preprocessed.pickle')
# print(reviews_df_preprocessed.isnull().values.sum()) # Check for any null values

X_train, X_test, y_train, y_test = preprocessData(reviews_df_preprocessed)

print('Training data...') t
= time()

pipeline = Pipeline([
                                        ('vect', TfidfVectorizer(ngram_range = (1,2), stop_words = 'english',
sublinear_tf = True)),
                                        ('chi', SelectKBest(score_func = chi2, k = 50000)),
                                        ('clf', LinearSVC(C = 1.0, penalty = 'l1', max_iter = 3000, dual = False,
class_weight = 'balanced'))
                              ])

model = pipeline.fit(X_train, y_train)

print('Training data completed!') print('Training
time: ', round(time()-t, 3), 's\n')
```

print('Predicting Test data...') t

= time()

prediction = model.predict(X_test)

print('Prediction completed!')

print('Prediction time: ', round(time()-t, 3), 's\n')

evaluate(y_test, prediction)

print('Confusion matrix: {}'.format(confusion_matrix(y_test, prediction))) print()

l = (y_test == 0).sum() + (y_test ==

1).sum() s = y_test.sum() print('Total

number of observations: ' + str(l))

print('Positives in observation: ' + str(s)) print('Negatives

in observation: ' + str(l - s)) print('Majority class is: ' +

str(s / l * 100) + '%')

Graph Plotting Code: import numpy as np import matplotlib.pyplot as plt from matplotlib.ticker import MaxNLocator from collections import namedtuple n_groups = 5 score_MNB = (85.25, 85.31, 85.56, 84.95, 85.31) score_LR = (88.12, 88.05, 87.54, 88.72, 88.05) score_LSVC=(88.12, 88.11, 87.59, 88.80, 88.11) score_RF=(82.43, 81.82, 79.74, 85.30, 81.83)

#n1=(score_MNB[0], score_LR[0], score_LSVC[0], score_RF[0])

#n2=(score_MNB[1], score_LR[1], score_LSVC[1], score_RF[1])

#n3=(score_MNB[2], score_LR[2], score_LSVC[2], score_RF[2])

#n4=(score_MNB[3], score_LR[3], score_LSVC[3], score_RF[3])

#n5=(score_MNB[4], score_LR[4], score_LSVC[4], score_RF[4])

```
fig, ax = plt.subplots() index = np.arange(n_groups) bar_width =
0.1 opacity = 0.7 error_config = {'ecolor': '0.3'} rects1 =
ax.bar(index,score_MNB, bar_width, alpha=opacity,
color='b', error_kw=error_config,
label='Multinomial Naive Bayes') z=index
+ bar_width rects2 = ax.bar(z, score_LR,
bar_width, alpha=opacity,
color='r', error_kw=error_config,
label='Logistic Regression') z=z+ bar_width
rects3 = ax.bar(z, score_LSVC, bar_width,
alpha=opacity, color='y',
error_kw=error_config, label='Linear
SVM') z=z+ bar_width
rects4 = ax.bar(z, score_RF, bar_width,
alpha=opacity, color='g',
error_kw=error_config, label='Random
Forest') ax.set_xlabel('Score Parameters')
ax.set_ylabel('Scores (in %)') ax.set_title('Scores
of Classifiers') ax.set_xticks(index + bar_width /
2)
ax.set_xticklabels(('F1', 'Accuracy', 'Precision', 'Recall', 'ROC AUC'))
ax.legend(bbox_to_anchor=(1, 1.02), loc=5, borderaxespad=0) fig.tight_layout()
plt.show()
```

	tweet_id	airline_sentimen	airline_sentimen	negativereason	negativereason_	airline	airline_sentimen	name	negativereason_	retweet_count	text	tweet_coord	tweet_created	tweet_location	user_timezone
2	5703061336777	neutral	1			Virgin America		cairdin		0	@VirginAmerica What @dhepbur		2015-02-24 11:35:52 -0800		Eastern Time (US & Canada)
3	5703011308881	positive	0.3486		0	Virgin America		jnardino		0	@VirginAmerica plus you've adde		2015-02-24 11:15:59 -0800		Pacific Time (US & Canada)
4	5703010836728	neutral	0.6837			Virgin America		yvonnalynn		0	@VirginAmerica I didn't today... M		2015-02-24 11:1	Lets Play	Central Time (US & Canada)
5	5703010314076	negative	1	Bad Flight	0.7033	Virgin America		jnardino		0	@VirginAmerica it's really aggres		2015-02-24 11:15:36 -0800		Pacific Time (US & Canada)
6	5703008170744	negative	1	Can't Tell	1	Virgin America		jnardino		0	@VirginAmerica and it's a really b		2015-02-24 11:14:45 -0800		Pacific Time (US & Canada)
7	5703007670741	negative	1	Can't Tell	0.6842	Virgin America		jnardino		0	@VirginAmerica seriously would p it's really the only bad thing about		2015-02-24 11:14:33 -0800		Pacific Time (US & Canada)
8	5703006169013	positive	0.6745		0	Virgin America		cjmcginnis		0	@VirginAmerica yes, nearly every		2015-02-24 11:1	San Francisco C	Pacific Time (US & Canada)
9	5703002485533	neutral	0.634			Virgin America		pilot		0	@VirginAmerica Really missed a		2015-02-24 11:1	Los Angeles	Pacific Time (US & Canada)
10	5702999532869	positive	0.6559			Virgin America		dhepburn		0	@virginamerica Well, I didn't…but		2015-02-24 11:1	San Diego	Pacific Time (US & Canada)
11	5702954596312	positive	1			Virgin America		YupitsTate		0	@VirginAmerica it was amazing, a		2015-02-24 10:5	Los Angeles	Eastern Time (US & Canada)
12	5702941891430	neutral	0.6769		0	Virgin America		idk_but_youtube		0	@VirginAmerica did you know tha		2015-02-24 10:4	1/1 loner squad	Eastern Time (US & Canada)
13	5702897244532	positive	1			Virgin America		HyperCamiLax		0	@VirginAmerica I <3 pretty grap		2015-02-24 10:3	NYC	America/New_York
14	5702895840614	positive	1			Virgin America		HyperCamiLax		0	@VirginAmerica This is such a gr		2015-02-24 10:3	NYC	America/New_York
15	5702874084381	positive	0.6451			Virgin America		mollanderson		0	@VirginAmerica @virginmedia I'm		2015-02-24 10:21:28 -0800		Eastern Time (US & Canada)
16	5702859048095	positive	1			Virgin America		sjespers		0	@VirginAmerica Thanks!		2015-02-24 10:1	San Francisco, C	Pacific Time (US & Canada)
17	5702824691210	negative	0.6842	Late Flight	0.3684	Virgin America		smartwatermelon		0	@VirginAmerica SFO-PDX sched		2015-02-24 10:0	palo alto, ca	Pacific Time (US & Canada)
18	5702777243857	positive	1			Virgin America		ItzBrianHunty		0	@VirginAmerica So excited for my		2015-02-24 09:4	west covina	Pacific Time (US & Canada)
19	5702769173011	negative	1	Bad Flight	1	Virgin America		heatherovieda		0	@VirginAmerica I flew from NYC		2015-02-24 09:3	this place called	Eastern Time (US & Canada)
20	5702706846199	positive	1			Virgin America		thebrandiray		0	I ❤️ flying @VirginAmerica. 😊👍		2015-02-24 09:1	Somewhere cele	Atlantic Time (Canada)
21	5702679566487	positive	1			Virgin America		JNLpierce		0	@VirginAmerica you know what w		2015-02-24 09:0	Boston | Walthar	Quito
22	5702658835133	negative	0.6705	Can't Tell	0.3614	Virgin America		MISSGJ		0	@VirginAmerica why are your firs		2015-02-24 08:55:56 -0800		
23	5702641451168	positive	1			Virgin America		DT_Les		0	@VirginAmerica	[40.74804263, -7	2015-02-24 08:49:01 -0800		
24	5702594202878	positive	1			Virgin America		ElvinaBeck		0	@VirginAmerica I love the hipster		2015-02-24 08:3	Los Angeles	Pacific Time (US & Canada)
25	5702588222975	neutral	1			Virgin America		rjlynch21086		0	@VirginAmerica will you be makir		2015-02-24 08:2	Boston, MA	Eastern Time (US & Canada)
26	5702565535020	negative	1	Customer Servic	0.3557	Virgin America		ayeevickiee		0	@VirginAmerica you guys messe		2015-02-24 08:1	714	Mountain Time (US & Canada)
27	5702491024049	negative	1	Customer Servic	1	Virgin America		Leora13		0	@VirginAmerica status match pro		2015-02-24 07:49:15 -0800		
28	5702396328073	negative	1	Can't Tell	0.6614	Virgin America		meredithjlynn		0	@VirginAmerica What happened		2015-02-24 07:11:37 -0800		
29	5702178315576	neutral	0.6854			Virgin America		AdamSinger		0	@VirginAmerica do you miss me?		2015-02-24 05:4	San Francisco, C	Central Time (US & Canada)
30	5702078864937	negative	1	Bad Flight	1	Virgin America		blackjackpro911		0	@VirginAmerica	[42.361016, -71.	2015-02-24 05:0	San Mateo, CA & Las Vegas, NV	
31	5701245961809	neutral	0.615		0	Virgin America		TenantsUpstairs		0	@VirginAmerica	[33.94540417, -	2015-02-23 23:3	Brooklyn	Atlantic Time (Canada)
32	5701140218542	negative	1	Flight Booking P	1	Virgin America		jordanpichler		0	@VirginAmerica hi! I just bked a c		2015-02-23 22:52:29 -0800		Vienna
33	5700947013714	neutral	1			Virgin America		JCervantezzz		0	@VirginAmerica Are the hours of		2015-02-23 21:3	California, San F	Pacific Time (US & Canada)
34	5700884041566	negative	1	Customer Servic	1	Virgin America		Cuschoolie1		0	@VirginAmerica	[33.94209449, -	2015-02-23 21:1	Washington DC	Quito
35	5700845827808	negative	1	Customer Servic	1	Virgin America		amanduhmccarty		0	@VirginAmerica awaiting my retu		2015-02-23 20:55:30 -0800		Pacific Time (US & Canada)
36	5700767929936	positive	1			Virgin America		NorthTxHomeTeam		0	@VirginAmerica	[33.2145038, -96	2015-02-23 20:2	Texas	Central Time (US & Canada)
37	5700519912773	neutral	0.6207			Virgin America		miaerolinea		0	Nice RT @VirginAmerica: Vibe wi		2015-02-23 18:4	Worldwide	Caracas
38	5700513815343	positive	1			Virgin America		Nicsplace		0	@VirginAmerica Moodlighting is t		2015-02-23 18:4	Central Texas	
39	5700453935656	positive	1			Virgin America		Nicsplace		0	@VirginAmerica @freddieawards		2015-02-23 18:1	Central Texas	
40	5700389414971	neutral	0.6791		0	Virgin America		elisha_malulani		0	@VirginAmerica when can I book		2015-02-23 17:5	i'm creating a m	Pacific Time (US & Canada)
41	5700358768450	negative	1	Customer Servic	1	Virgin America		DannyDouglass		0	@VirginAmerica Your chat suppo		2015-02-23 17:4	San Francisco, C	Pacific Time (US & Canada)
42	5700335933946	positive	0.6639			Virgin America		jamesferrandini		0	@VirginAmerica View of downtow		2015-02-23 17:32:54 -0800		
43	5700254823448	negative	0.6688	Flight Booking P	0.6688	Virgin America		will_lenzenjr		0	@VirginAmerica Hey, first time fly		2015-02-23 17:0	Iowa City	Central Time (US & Canada)
44	5700163042849	neutral	1			Virgin America		GottAmanda		0	@VirginAmerica	[34.0219817, -11	2015-02-23 16:2	Los Angeles	
45	5700154087884	neutral	0.6578		0	Virgin America		KGervaise		0	@VirginAmerica I have an unused		2015-02-23 16:2	Georgia	Pacific Time (US & Canada)
46	5700135236500	neutral	1			Virgin America		papamurat		0	@VirginAmerica are flights leavin		2015-02-23 16:13:09 -0800		
47	5700122575490	positive	1			Virgin America		arieldaie		0	@VirginAmerica I'm #elevategold		2015-02-23 16:0	Los Angeles	
48	5700113414838	neutral	0.6799			Virgin America		vacations7		0	@VirginAmerica DREAM http://t.		2015-02-23 16:0	Turks and caicos	
49	5700105717072	positive	1			Virgin America		ChelseaPoe666		0	@VirginAmerica wow this just ble		2015-02-23 16:0	Oakland via Mid	Atlantic Time (Canada)
50	5700105394993	neutral	1			Virgin America		BobGlavinVO		0	@VirginAmerica @ladygaga @ca		2015-02-23 16:0	New York, NY	Eastern Time (US & Canada)
51	5700097134478	neutral	0.6436			Virgin America		lisaaiko		0	@VirginAmerica @ladygaga @ca		2015-02-23 15:58:00 -0800		
52	5700090354553	neutral	0.6764		0	Virgin America		grantbrowne		0	@VirginAmerica Is flight 769 on it		2015-02-23 15:5	Worldwide	Central Time (US & Canada)
53	5700068860129	positive	0.657			Virgin America		joyabsalon		0	@VirginAmerica @ladygaga @ca		2015-02-23 15:4	Northern Virginia	Eastern Time (US & Canada)
54	5700043917318	neutral	1			Virgin America		2v		0	@VirginAmerica wish you flew ou		2015-02-23 15:3	Los Angeles / At	Eastern Time (US & Canada)
55	5700011949004	neutral	0.7118		0	Virgin America		KSmithFoundHere		0	@VirginAmerica @ladygaga @ca		2015-02-23 15:24:09 -0800		Atlantic Time (Canada)
56	5700000716448	neutral	1			Virgin America		papamurat		0	@VirginAmerica Will flights be lea		2015-02-23 15:19:41 -0800		
57	5699964122865	negative	0.6939	Flight Booking P	0.6939	Virgin America		murphicus		0	@VirginAmerica hi! i'm so excited		2015-02-23 15:0	new york, new y	Eastern Time (US & Canada)
58	5699962454621	positive	1			Virgin America		VinnieFerra		0	@VirginAmerica you know it. Nee		2015-02-23 15:0	brooklyn, Ny	Pacific Time (US & Canada)
59	5699902226094	positive	0.635			Virgin America		KevinDemsi		0	@VirginAmerica @ladygaga @ca		2015-02-23 14:4	Bali, Republic of	Kuala Lumpur
60	5699901632098	neutral	0.7007			Virgin America		giffgaffman		0	@VirginAmerica @ladygaga @ca		2015-02-23 14:4	UK, USA.	
61	5699895044313	neutral	1			Virgin America		HanlonBrothers		0	@VirginAmerica New marketing s		2015-02-23 14:3	Gold Coast, Aus	Brisbane
62	5699893216980	neutral	1			Virgin America		emilybg78		0	@VirginAmerica @ladygaga @ca		2015-02-23 14:3	Stockton, CA	Arizona
63	5699890345015	negative	1	Customer Servic	1	Virgin America		rachie1126		0	@VirginAmerica I called a 3-4 we		2015-02-23 14:3	New York, NY	Eastern Time (US & Canada)
64	5699876224848	neutral	0.6858			Virgin America		adawson66		0	@VirginAmerica	[33.57963333, -	2015-02-23 14:30:13 -0800		
65	5699867825670	neutral	1			Virgin America		SocialPLC		0	@VirginAmerica @LadyGaga @C		2015-02-23 14:2	Twin Cities, Minr	Eastern Time (US & Canada)
66	5699863480415	positive	1			Virgin America		jeffreymace01		0	@VirginAmerica @ladygaga @ca		2015-02-23 14:25:09 -0800		
67	5699823076347	neutral	0.6814		0	Virgin America		1stcrown		0	@VirginAmerica Flight 0736 DAL		2015-02-23 14:0	USA	Central Time (US & Canada)
68	5699766201585	negative	1	Customer Servic	1	Virgin America		onerockgypsy		0	@VirginAmerica heyyyy guyyyys.		2015-02-23 13:4	next city	Pacific Time (US & Canada)
69	5699738213961	negative	1	Late Flight	0.6789	Virgin America		noelduan		0	@VirginAmerica Hi, Virgin! I'm on		2015-02-23 13:3	SF ↔ NY	Eastern Time (US & Canada)
70	5699725084992	positive	0.6922			Virgin America		Travelzoo		0	@VirginAmerica Congrats on win		2015-02-23 13:3	New York, NY	Pacific Time (US & Canada)
71	5699670199587	negative	1	Lost Luggage	1	Virgin America		gianagon		0	@VirginAmerica	[40.6413712, -73	2015-02-23 13:0	New York + Pan	Eastern Time (US & Canada)
72	5699618662246	neutral	1			Virgin America		bxchen		0	@virginamerica Need to change r		2015-02-23 12:4	San Francisco, C	Eastern Time (US & Canada)
73	5699498911636	neutral	0.6492		0	Virgin America		seimatrun		0	@VirginAmerica I emailed your cu		2015-02-23 12:0	Los Angeles	
74	5699489668733	neutral	1			Virgin America		jamied7		0	@VirginAmerica hi I just booked a		2015-02-23 11:5	London, England	London
75	5699463621266	negative	1	Flight Attendant	0.3516	Virgin America		seimatrun		0	@VirginAmerica your airline is aw		2015-02-23 11:4	Los Angeles	
76	5699429036838	positive	1			Virgin America		mrmichaellay		0	@VirginAmerica	[36.08457854, -	2015-02-23 11:3	Floridian from Cl	Eastern Time (US & Canada)
77	5699419574907	positive	1			Virgin America		TaylorLumsden		0	@VirginAmerica awesome. I flew		2015-02-23 11:2	Dallas, Texas	Mountain Time (US & Canada)
78	5699408349944	neutral	1			Virgin America		campusmoviefest		0	@VirginAmerica Or watch some c		2015-02-23 11:2	USA	Eastern Time (US & Canada)
79	5699403237465	neutral	1			Virgin America		TaylorLumsden		0	@VirginAmerica first time flying y		2015-02-23 11:2	Dallas, Texas	Mountain Time (US & Canada)
80	5699352320333	negative	1	Customer Servic	1	Virgin America		meme_meng		0	@VirginAmerica what is going on		2015-02-23 11:02:02 -0800		
81	5699343958654	neutral	1			Virgin America		kyle_romanoff		0	@VirginAmerica what happened t		2015-02-23 10:58:43 -0800		
82	5699338169633	negative	1	Customer Servic	1	Virgin America		GunsNDip		0	@VirginAmerica why can't you su		2015-02-23 10:56:25 -0800		Pacific Time (US & Canada)
83	5699337779311	positive	1			Virgin America		artisticwritr87		0	@VirginAmerica I've applied more		2015-02-23 10:5	Seattle, WA	Pacific Time (US & Canada)
84	5699334055063	negative	0.6792	Late Flight	0.3477	Virgin America		arieldaie		0	@VirginAmerica you're the best!!		2015-02-23 10:5	Los Angeles	
85	5699333605643	negative	1	Can't Tell	1	Virgin America		GunsNDip		0	@VirginAmerica I have no interes		2015-02-23 10:54:36 -0800		Pacific Time (US & Canada)
86	5699292431460	negative	1	Can't Tell	1	Virgin America		GunsNDip		0	@VirginAmerica it was a disappoi		2015-02-23 10:38:14 -0800		Pacific Time (US & Canada)
87	5699269988243	negative	1	Flight Booking P	1	Virgin America		jsatk		0	@VirginAmerica	[0.0, 0.0]	2015-02-23 10:2	Lower Pacific He	Pacific Time (US & Canada)
88	5699233949904	neutral	0.6705		0	Virgin America		serenaklal		0	@VirginAmerica Can't bring up m		2015-02-23 10:1	Chicago	Eastern Time (US & Canada)
89	5699220085882	neutral	1			Virgin America		openambit1		0	@VirginAmerica Random Q: what		2015-02-23 10:09:30 -0800		
90	5699208249053	neutral	0.6545		0	Virgin America		cabowine		0	@VirginAmerica I <3 Flying VA		2015-02-23 10:0	Los Cabos,Mexi	Arizona
91	5699190412441	negative	1	Can't Tell	0.6513	Virgin America		MaryAnnTaylorT		0	@VirginAmerica Why is the site d		2015-02-23 09:5	New York, NY	Arizona
92	5699159411920	neutral	1			Virgin America		RamotControl		0	@VirginAmerica "You down with F		2015-02-23 09:45:23 -0800		Pacific Time (US & Canada)
93	5699133394274	neutral	0.6639		0	Virgin America		losermelon		0	@VirginAmerica hi, i did not get p		2015-02-23 09:35:03 -0800		
94	5699118169370	negative	1	Cancelled Flight	1	Virgin America		AlisonK33774854		0	@VirginAmerica I like the TV and		2015-02-23 09:29:00 -0800		
95	5699116741587	negative	1	Late Flight	1	Virgin America		GunsNDip		0	@VirginAmerica just landed in LA		2015-02-23 09:28:26 -0800		Pacific Time (US & Canada)
96	5699112189425	neutral	0.6765		0	Virgin America		yazdanagh		0	@VirginAmerica why is flight 345		2015-02-23 09:26:37 -0800		
97	5699109818680	negative	1	Customer Servic	0.6863	Virgin America		MerchEngines		0	@VirginAmerica Is it me, or is you		2015-02-23 09:2	Los Angeles, CA	Arizona
98	5699092245216	negative	1	Customer Servic	0.6771	Virgin America		ColorCartel		0	@VirginAmerica I can't check in o		2015-02-23 09:1	Austin, TX	Mountain Time (US & Canada)
99	5699073364850	negative	1	Can't Tell	0.659	Virgin America		MustBeSpoken		0	@VirginAmerica - Let 2 scanned i		2015-02-23 09:11:12 -0800		
100	5698968056110	negative	1	Flight Booking P	0.6714	Virgin America		mattbunk		0	@virginamerica What is your pho		2015-02-23 08:2	Sterling Heights,	Eastern Time (US & Canada)
101	5698944496203	negative	1	Customer Servic	1	Virgin America		louisjenny		0	@VirginAmerica is anyone doing		2015-02-23 08:1	Washington DC	Quito
102	5698944070019	neutral	1			Virgin America		STravelsW		0	@VirginAmerica trying to add my		2015-02-23 08:1	Manhattan Beach, CA	
103	5698921996906	negative	1	Late Flight	0.6882	Virgin America		GunsNDip		0	@VirginAmerica why must a trave		2015-02-23 08:11:03 -0800		Pacific Time (US & Canada)
104	5698914692107	neutral	1			Virgin America		joeyrenagade		0	@VirginAmerica check out new m		2015-02-23 08:0	Greater Los Angeles	
105	5698914361008	negative	0.6925	Late Flight	0.3521	Virgin America		mrmichaellay		0	@virginamerica	[0.0, 0.0]	2015-02-23 08:0	Floridian from Cl	Eastern Time (US & Canada)
106	5698873107134	negative	1	Late Flight	0.3486	Virgin America		GunsNDip		0	@VirginAmerica your no Late Flig		2015-02-23 07:51:37 -0800		Pacific Time (US & Canada)
107	5698870494460	positive	1			Virgin America		TheDuchessSF		0	@VirginAmerica - amazing custor		2015-02-23 07:5	Online	Pacific Time (US & Canada)
108	5698845517128	negative	1	Customer Servic	1	Virgin America		BeLeather		0	@VirginAmerica	[0.0, 0.0]	2015-02-23 07:40:39 -0800		Pacific Time (US & Canada)
109	5698844078524	negative	1	Flight Booking P	0.6366	Virgin America		BeLeather		0	@VirginAmerica	[0.0, 0.0]	2015-02-23 07:40:05 -0800		Pacific Time (US & Canada)
110	5698815485157	neutral	0.6593		0	Virgin America		drcaseydrake		0	@VirginAmerica	[37.79374402, -	2015-02-23 07:2	Dallas, TX	
111	5698736697003	positive	0.6823		0	Virgin America		flyfromWAS		0	@VirginAmerica has getaway dea		2015-02-23 06:5	Washington, DC	Eastern Time (US & Canada)
112	5698736681317	neutral	1			Virgin America		flyfromSEA		0	@VirginAmerica has getaway dea		2015-02-23 06:5	Seattle	Central Time (US & Canada)
113	5698736656402	positive	0.6806			Virgin America		flyfromNYC		0	@VirginAmerica has getaway dea		2015-02-23 06:5	New York City, N	Central Time (US & Canada)
114	5698736646127	neutral	0.6529			Virgin America		flyfromLAX		0	@VirginAmerica has getaway dea		2015-02-23 06:5	Los Angeles, CA	Central Time (US & Canada)
115	5698720586136	positive	0.678			Virgin America		Silvanabfer		0	@VirginAmerica Have a great we		2015-02-23 06:51:01 -0800		
116	5698612097819	positive	0.3482		0	Virgin America		AdamJdubs		0	@VirginAmerica come back to #P		2015-02-23 06:0	Earth	Eastern Time (US & Canada)
117	5698479201926	negative	1	Late Flight	1	Virgin America		nicholas_v		0	@VirginAmerica	[26.074379, -80.	2015-02-23 05:15:06 -0800		Eastern Time (US & Canada)
118	5698143397853	positive	1			Virgin America		tstashajones		0	@VirginAmerica is the best airline		2015-02-23 03:0	Halifax, Nova Sc	Eastern Time (US & Canada)
119	5697776073711	positive	1			Virgin America		SkateMamas		0	@VirginAmerica and again! Anoth		2015-02-23 00:3	Los Angeles, CA	
120	5697740782334	positive	1			Virgin America		dngoo		0	@VirginAmerica your beautiful fro		2015-02-23 00:2	Near a park, wat	Pacific Time (US & Canada)
121	5697703636235	positive	1			Virgin America		SamBrittenham		0	@VirginAmerica Love the team ru		2015-02-23 00:0	USA	Pacific Time (US & Canada)
122	5697483167763	negative	0.6832	Customer Servic	0.3773	Virgin America		usagibrian		0	@VirginAmerica Use another bro		2015-02-22 22:3	San Francisco C	Pacific Time (US & Canada)
123	5697412217839	negative	1	Flight Booking P	0.6767	Virgin America		usagibrian		0	@VirginAmerica And now the fligh		2015-02-22 22:1	San Francisco C	Pacific Time (US & Canada)
124	5697376036179	negative	1	Customer Servic	0.6527	Virgin America		KindofLuke		0	@VirginAmerica I like the custom		2015-02-22 21:56:44 -0800		
125	5697141277922	positive	1			Virgin America		ptbrodie		0	@VirginAmerica thanks to your o		2015-02-22 20:2	San Francisco	
126	5696751443538	positive	1			Virgin America		cheryleng		0	@VirginAmerica	[33.9469039, -11	2015-02-22 17:48:33 -0800		Pacific Time (US & Canada)
127	5696743581359	neutral	1			Virgin America		HishamSharaby		0	@VirginAmerica Do you provide c		2015-02-22 17:4	New York	Eastern Time (US & Canada)
128	5696664772650	negative	0.6703	Customer Servic	0.6703	Virgin America		ChrisFordisHere		0	@VirginAmerica	[51.04345575, -	2015-02-22 17:1	NYC	Tehran
129	5696524979477	positive	1			Virgin America		JKF1897		0	@VirginAmerica completely awes		2015-02-22 16:18:33 -0800		
130	5696491164872	neutral	1			Virgin America		F6x		0	@VirginAmerica	[40.64662464, -	2015-02-22 16:0	San Francisco, C	Pacific Time (US & Canada)
131	5696432624592	neutral	0.6535		0	Virgin America		lawyang588		0	@VirginAmerica is flight 882 Canc		2015-02-22 15:41:51 -0800		
132	5696428455162	negative	1	Customer Servic	0.6448	Virgin America		melokudo		0	@VirginAmerica you are failing yo		2015-02-22 15:40:12 -0800		Eastern Time (US & Canada)
133	5696343183492	negative	1	Can't Tell	1	Virgin America		ChrysiChrysic		0	@VirginAmerica @FiDiFamilies u		2015-02-22 15:06:19 -0800		
134	5696336309783	neutral	1			Virgin America		nikkisixxfan93		0	@VirginAmerica has flight numbe		2015-02-22 15:0	Sacramento,Cali	Pacific Time (US & Canada)
135	5696332795460	negative	1	Cancelled Flight	0.6875	Virgin America		ChrysiChrysic		0	@VirginAmerica @ChrysiChrysic		2015-02-22 15:02:11 -0800		
136	5696300922734	negative	1	Late Flight	1	Virgin America		MOCBlogger		0	@VirginAmerica Another delayed		2015-02-22 14:4	San Diego	Alaska
137	5696274804257	negative	1	Customer Servic	1	Virgin America		tfaz		0	@VirginAmerica I need to registe		2015-02-22 14:3	Oakland, Califor	Pacific Time (US & Canada)
138	5696257392319	positive	1			Virgin America		lisaptv		1	@virginamerica you ROCK for ma		2015-02-22 14:32:14 -0800		Mountain Time (US & Canada)
139	5696256097997	neutral	1			Virgin America		dropapp		0	@VirginAmerica, @reallytallchris		2015-02-22 14:31:43 -0800		
140	5696201023893	positive	0.7011			Virgin America		HollywoodHotMom		0	@VirginAmerica always!!! Xoxo		2015-02-22 14:0	All Over!	Pacific Time (US & Canada)
141	5696195693728	negative	1	Flight Booking P	1	Virgin America		GoShar2012		0	@VirginAmerica why can't we boo		2015-02-22 14:0	Providence, RI	Eastern Time (US & Canada)

Data Set 2:

Amazon product data

Description

This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

Files

"Small" subsets for experimentation.

Books	5-core (8,898,041 reviews)	ratings only (22,507,155 ratings)
Electronics	5-core (1,689,188 reviews)	ratings only (7,824,482 ratings)
Movies and TV	5-core (1,697,533 reviews)	ratings only (4,607,047 ratings)
CDs and Vinyl	5-core (1,097,592 reviews)	ratings only (3,749,004 ratings)
Clothing, Shoes and Jewelry	5-core (278,677 reviews)	ratings only (5,748,920 ratings)
Home and Kitchen	5-core (551,682 reviews)	ratings only (4,253,926 ratings)
Kindle Store	5-core (982,619 reviews)	ratings only (3,205,467 ratings)
Sports and Outdoors	5-core (296,337 reviews)	ratings only (3,268,695 ratings)
Cell Phones and Accessories	5-core (194,439 reviews)	ratings only (3,447,249 ratings)
Health and Personal Care	5-core (346,355 reviews)	ratings only (2,982,326 ratings)

Toys and Games	5-core (167,597 reviews)	ratings only (2,252,771 ratings)
Video Games	5-core (231,780 reviews)	ratings only (1,324,753 ratings)
Tools and Home Improvement	5-core (134,476 reviews)	ratings only (1,926,047 ratings)
Beauty	5-core (198,502 reviews)	ratings only (2,023,070 ratings)
Apps for Android	5-core (752,937 reviews)	ratings only (2,638,172 ratings)
Office Products	5-core (53,258 reviews)	ratings only (1,243,186 ratings)
Pet Supplies	5-core (157,836 reviews)	ratings only (1,235,316 ratings)
Automotive	5-core (20,473 reviews)	ratings only (1,373,768 ratings)
Grocery and Gourmet Food	5-core (151,254 reviews)	ratings only (1,297,156 ratings)
Patio, Lawn and Garden	5-core (13,272 reviews)	ratings only (993,490 ratings)
Baby	5-core (160,792 reviews)	ratings only (915,446 ratings)
Digital Music	5-core (64,706 reviews)	ratings only (836,006 ratings)
Musical Instruments	5-core (10,261 reviews)	ratings only (500,176 ratings)
Amazon Instant Video	5-core (37,126 reviews)	ratings only (583,933 ratings)

Complete review data

Please see the **per-category** files below, and only download these (large!) files if you really need them:

raw review data (20gb) - all 142.8 million reviews

The above file contains some duplicate reviews, mainly due to near-identical products whose reviews Amazon merges, e.g. VHS and DVD versions of the same movie. These duplicates have been removed in the files below:

user review data (18gb) - duplicate items removed (83.68 million reviews), sorted by user **product review data** (18gb) - duplicate items removed, sorted by product **ratings only** (3.2gb) - same as above, in csv form without reviews or metadata

5-core (9.9gb) - subset of the data in which all users and items have at least 5 reviews (41.13 million reviews)

Finally, the following file removes duplicates more aggressively, removing duplicates even if they are written by different users. This accounts for users with multiple accounts or plagiarized reviews. Such duplicates account for less than 1 percent of reviews, though this dataset is probably preferable for sentiment analysis type tasks:

aggressively deduplicated data (18gb) - no duplicates whatsoever (82.83 million reviews)

Format is one-review-per-line in (loose) json. See examples below for further help reading the data.

Sample review:


```
{ "reviewerID": "A2SUAM1J3GNN3B", "asin": "0000013714",
"reviewerName": "J. McDonald", "helpful": [2, 3], "reviewText": "I
bought this for my husband who plays the piano. He is having a
wonderful time playing these old hymns. The music is at times hard
to read because we think the book was published for singing from
more than playing from. Great purchase though!", "overall": 5.0,
"summary": "Heavenly Highway Hymns", "unixReviewTime": 1252800000,
"reviewTime": "09 13, 2009" }
```


where

- `reviewerID` - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- `asin` - ID of the product, e.g. 0000013714
- `reviewerName` - name of the reviewer
- `helpful` - helpfulness rating of the review, e.g. 2/3
- `reviewText` - text of the review
- `overall` - rating of the product
- `summary` - summary of the review
- `unixReviewTime` - time of the review (unix time)

- `reviewTime` - time of the review (raw)

Metadata

Metadata includes descriptions, price, sales-rank, brand info, and co-purchasing links:

metadata (3.1gb) - metadata for 9.4 million products

Sample metadata:


```
{ "asin": "0000031852", "title": "Girls Ballet Tutu Zebra Hot
Pink", "price": 3.17, "imUrl":
"http://ecx.imagesamazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
"related": { "also_bought": ["B00JHONN1S", "B002BZX8Z6",
"B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00D0WDS9A", "B00D0GCI8S",
"0000031895", "B003AVKOP2", "B003AVEU6G", "B003IEDM9Q",
"B002R0FA24", "B00D23MC6W", "B00D2K0PA0", "B00538F5OK",
"B00CEV86I6", "B002R0FABA", "B00D10CLVW", "B003AVNY6I",
"B002GZGI4E", "B001T9NUFS", "B002R0F7FE", "B00E1YRI4C",
"B008UBQZKU", "B00D103F8U", "B007R2RM8W"], "also_viewed":
["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y", "B00D23MC6W",
"B00AFDOPDA", "B00E1YRI4C", "B002GZGI4E", "B003AVKOP2",
"B00D9C1WBM", "B00CEV8366", "B00CEUX0D8", "B0079ME3KU",
"B00CEUWY8K", "B004FOEEHC", "0000031895", "B00BC4GY9Y",
"B003XRKA7A", "B00K18LKX2", "B00EM7KAG6", "B00AMQ17JA",
"B00D9C32NI", "B002C3Y6WG", "B00JLL4L5Y", "B003AVNY6I",
"B008UBQZKU", "B00D0WDS9A", "B00613WDTQ", "B00538F5OK",
"B005C4Y4F6", "B004LHZ1NY", "B00CPHX76U", "B00CEUWUZC",
"B00IJVASUE", "B00GOR07RE", "B00J2GTM0W", "B00JHNSNSM",
"B003IEDM9Q", "B00CYBU84G", "B008VV8NSQ", "B00CYBULSO",
"B00I2UHSZA", "B005F50FXC", "B007LCQI3S", "B00DP68AVW",
"B009RXWNSI", "B003AVEU6G", "B00HSOJB9M", "B00EHAGZNA",
"B0046W9T8C", "B00E79VW6Q", "B00D10CLVW", "B00B0AVO54",
"B00E95LC8Q", "B00GOR92SO", "B007ZN5Y56", "B00AL2569W",
"B00B608000", "B008F0SMUC", "B00BFXLZ8M"], "bought_together":
["B002BZX8Z6"] }, "salesRank": {"Toys & Games": 211836}, "brand":
"Coxlures", "categories": [["Sports & Outdoors", "Other Sports",
"Dance"]] }
```


where

- `asin` - ID of the product, e.g. 0000031852
- `title` - name of the product

- `price` - price in US dollars (at time of crawl)
- `imUrl` - url of the product image
- `related` - related products (also bought, also viewed, bought together, buy after viewing)
- `salesRank` - sales rank information
- `brand` - brand name
- `categories` - list of categories the product belongs to

Visual Features

We extracted visual features from each product image using a deep CNN (see citation below). Image features are stored in a binary format, which consists of 10 characters (the product ID), followed by 4096 floats (repeated for every product). See files below for further help reading the data.

visual features (141gb) - visual features for all products

The images themselves can be extracted from the `imUrl` field in the metadata files.

Per-category files

Below are files for individual product categories, which have already had duplicate item reviews removed.

Books	reviews (22,507,155 reviews)	metadata (2,370,585 products)	image features
Electronics	reviews (7,824,482 reviews)	metadata (498,196 products)	image features
Movies and TV	reviews (4,607,047 reviews)	metadata (208,321 products)	image features
CDs and Vinyl	reviews (3,749,004 reviews)	metadata (492,799 products)	image features
Clothing, Shoes and Jewelry	reviews (5,748,920 reviews)	metadata (1,503,384 products)	image features

Home and Kitchen	reviews (4,253,926 reviews)	metadata (436,988 products)	image features
Kindle Store	reviews (3,205,467 reviews)	metadata (434,702 products)	image features
Sports and Outdoors	reviews (3,268,695 reviews)	metadata (532,197 products)	image features
Cell Phones and Accessories	reviews (3,447,249 reviews)	metadata (346,793 products)	image features
Health and Personal Care	reviews (2,982,326 reviews)	metadata (263,032 products)	image features
Toys and Games	reviews (2,252,771 reviews)	metadata (336,072 products)	image features
Video Games	reviews (1,324,753 reviews)	metadata (50,953 products)	image features
Tools and Home Improvement	reviews (1,926,047 reviews)	metadata (269,120 products)	image features
Beauty	reviews (2,023,070 reviews)	metadata (259,204 products)	image features
Apps for Android	reviews (2,638,173 reviews)	metadata (61,551 products)	image features
Office Products	reviews (1,243,186 reviews)	metadata (134,838 products)	image features
Pet Supplies	reviews (1,235,316 reviews)	metadata (110,707 products)	image features

Automotive	reviews (1,373,768 reviews)	metadata (331,090 products)	image features
Grocery and Gourmet Food	reviews (1,297,156 reviews)	metadata (171,760 products)	image features
Patio, Lawn and Garden	reviews (993,490 reviews)	metadata (109,094 products)	image features
Baby	reviews (915,446 reviews)	metadata (71,317 products)	image features
Digital Music	reviews (836,006 reviews)	metadata (279,899 products)	image features
Musical Instruments	reviews (500,176 reviews)	metadata (84,901 products)	image features
Amazon Instant Video	reviews (583,933 reviews)	metadata (30,648 products)	image features

Citation

Please cite one or both of the following if you use the data in any way:

Ups and downs: Modeling the visual evolution of fashion trends with one

Code

Reading the data

Data can be treated as python dictionary objects. A simple script to read any of the above the data is as follows:

```
def parse(path): g = gzip.open(path, 'r') for l in g: yield eval(l)
```

Convert to 'strict' json

The above data can be read with python 'eval', but is not strict json. If you'd like to use some language other than python, you can convert the data to strict json as follows:

```
import json import gzip def parse(path): g = gzip.open(path, 'r')
for l in g: yield json.dumps(eval(l)) f = open("output.strict",
'w') for l in parse("reviews_Video_Games.json.gz"): f.write(l +
'\n')
```

Pandas data frame

This code reads the data into a pandas data frame:

```
import pandas as pd import gzip def parse(path): g =
gzip.open(path, 'rb') for l in g: yield eval(l) def getDF(path): i
= 0 df = {} for d in parse(path): df[i] = d i += 1 return
pd.DataFrame.from_dict(df, orient='index') df =
getDF('reviews_Video_Games.json.gz')
```

Read image features

```
import array def readImageFeatures(path): f = open(path, 'rb')
while True: asin = f.read(10) if asin == '': break a =
array.array('f') a.fromfile(f, 4096) yield asin, a.tolist()
```

Example: compute average rating

```
ratings = [] for review in parse("reviews_Video_Games.json.gz"):
ratings.append(review['overall']) print sum(ratings) / len(ratings)
```

Example: latent-factor model in [mymedialite]

Predicts ratings from a rating-only CSV file

```
./rating_prediction --recommender=BiasedMatrixFactorization -
training-file=ratings_Video_Games.csv --test-ratio=0.1
```

Sentiment Analysis & Machine Learning

Sentiment analysis is a machine learning tool that analyzes texts for polarity, from positive to negative. By training machine learning tools with examples of emotions in text, machines automatically learn how to detect sentiment without human input.

To put it simply, machine learning allows computers to learn new tasks without being expressly programmed to perform them. Sentiment analysis models can be trained to read beyond mere definitions, to understand things like, context, sarcasm, and misapplied words. For example:

“Super user-friendly interface. Yeah right. An engineering degree would be helpful.”

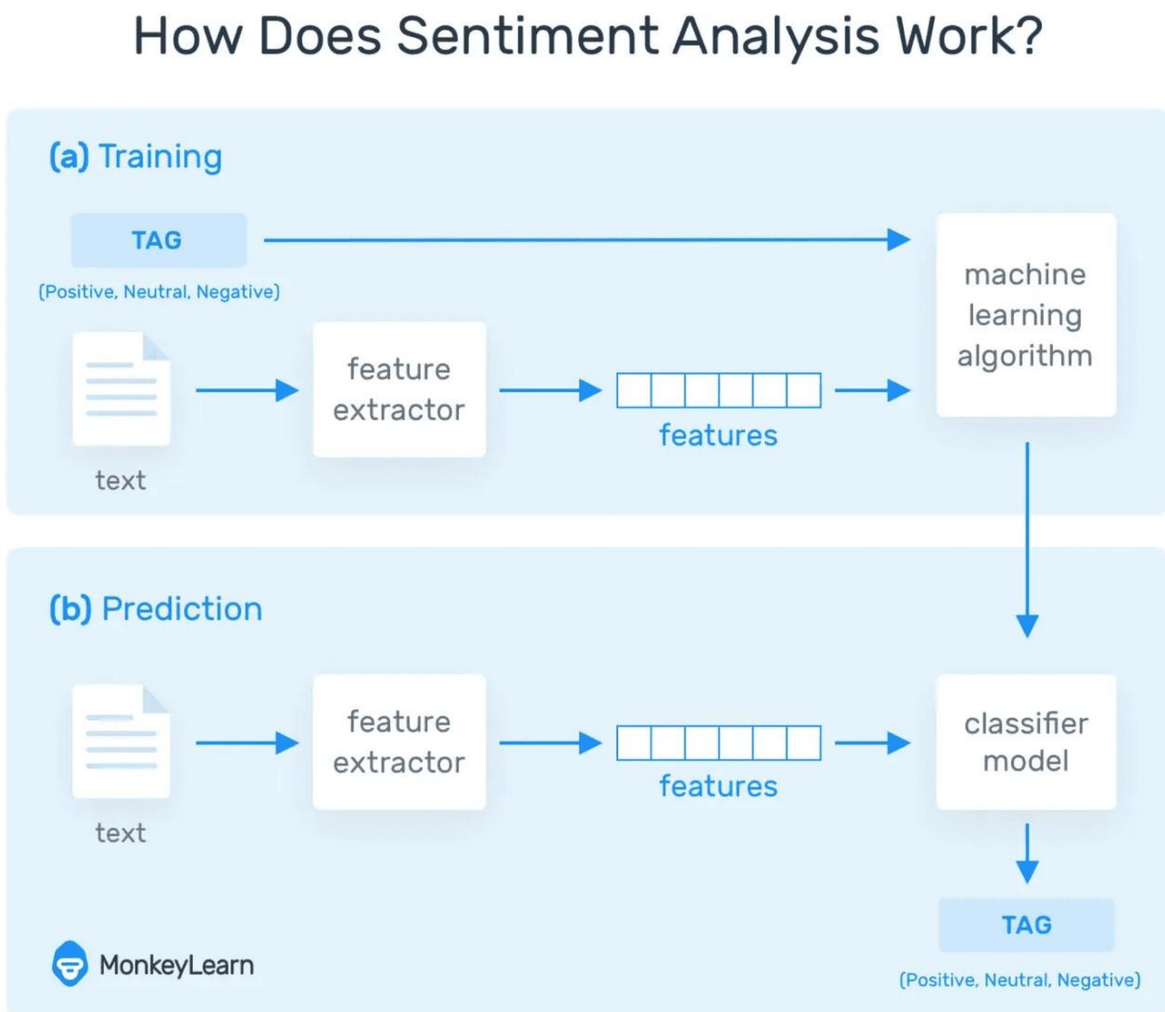
Out of context, the words ‘super user-friendly’ and ‘helpful’ could be read as positive, but this is clearly a negative comment. Using sentiment analysis, computers can automatically process text data and understand it just as a human would, saving hundreds of employee hours.

Imagine using machine learning to process customer service tickets, categorize them in order of urgency, and automatically route them to the correct department or employee. Or, to analyze thousands of product reviews and social media posts to gauge brand sentiment.

Read on to learn more about how machine learning works and how it can help your business

How Does Sentiment Analysis with Machine Learning Work?

There are a number of techniques and complex algorithms used to command and train machines to perform sentiment analysis. There are pros and cons to each. But, used together, they can provide exceptional results. Below are some of the most used algorithms.



Naive Bayes

Naive Bayes is a fairly simple group of probabilistic algorithms that, for sentiment analysis classification, assigns a probability that a given word or phrase should be considered positive or negative.

Essentially, this is how Bayes' theorem works. *The probability of A, if B is true, is equal to the probability of B, if A is true, times the probability of A being true, divided by the probability of B being true*:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

But that's a lot of math! Basically, Naive Bayes calculates words against each other. So, with machine learning models trained for word polarity, we can calculate the likelihood that a word, phrase, or text is positive or negative.

When techniques like lemmatization, stopword removal, and TF-IDF are implemented, Naive Bayes becomes more and more predictively accurate.

Linear Regression

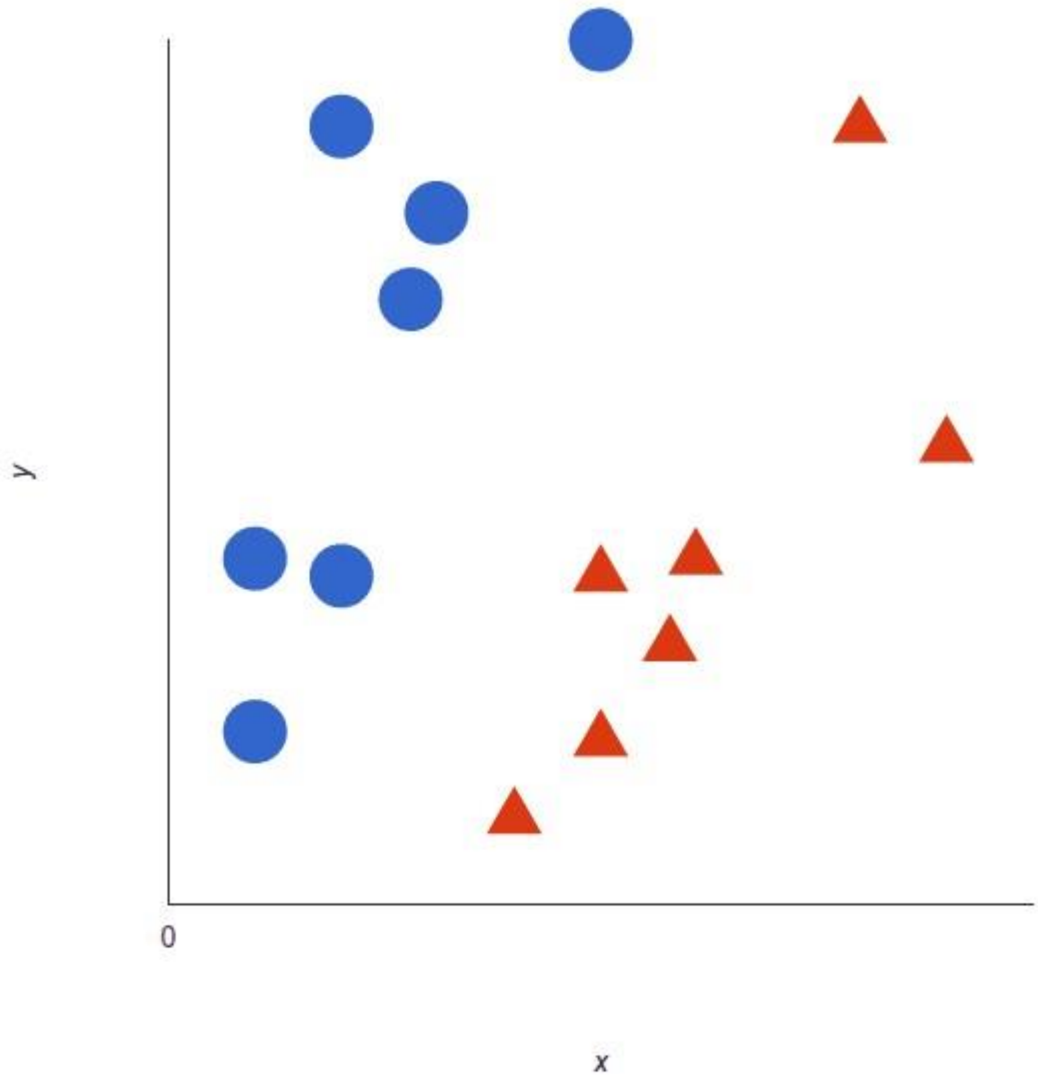
Linear regression is a statistical algorithm used to predict a *Y* value, given *X* features. Using machine learning, the data sets are examined to show a relationship. The relationships are then placed along the *X/Y* axis, with a straight line running through them to predict further relationships.

Linear regression calculates how the *X* input (words and phrases) relates to the *Y* output (polarity). This will determine where words and phrases fall on a scale of polarity from "really positive" to "really negative" and everywhere in between.

Support Vector Machines (SVM)

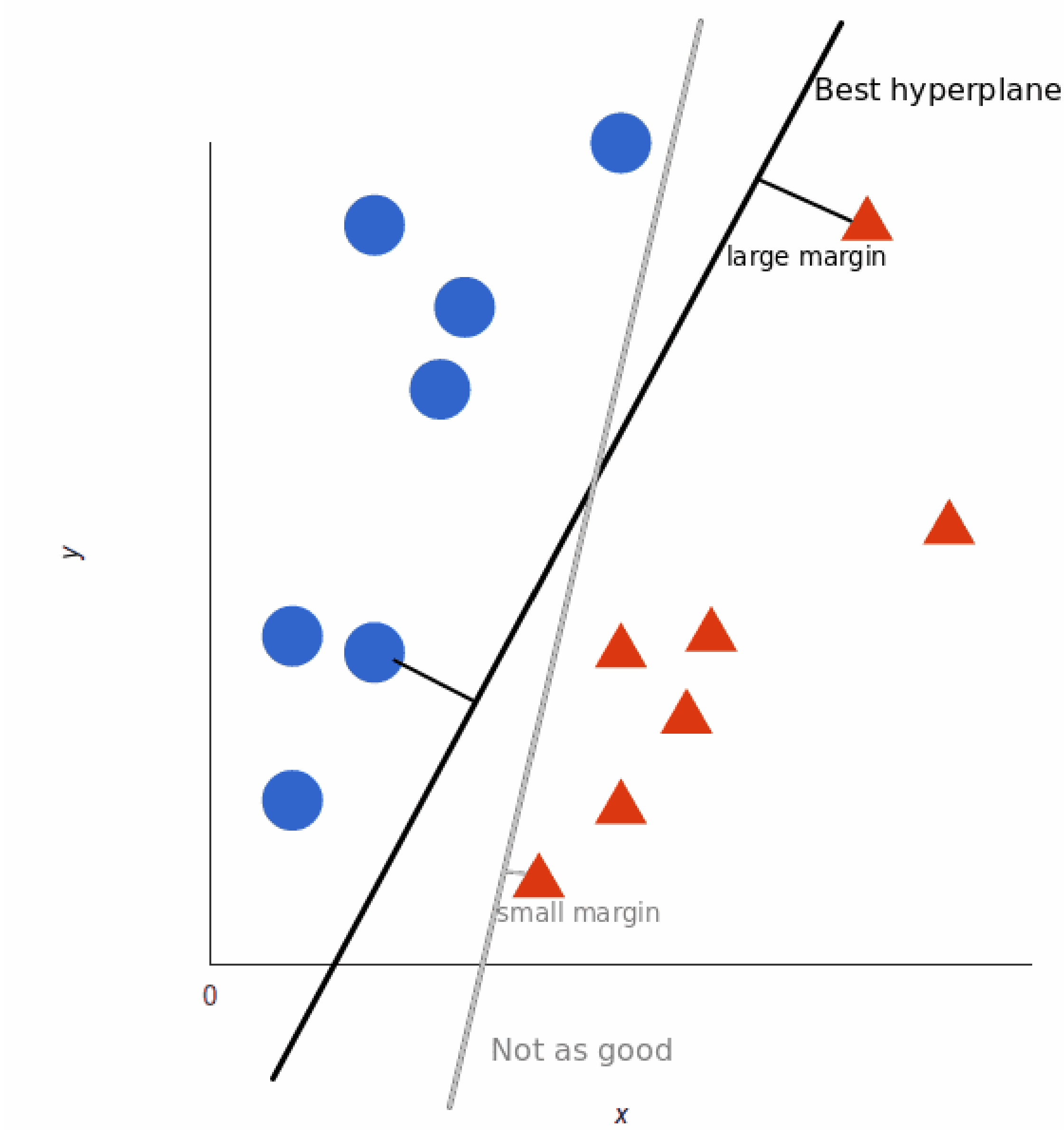
A support vector machine is another supervised machine learning model, similar to linear regression but more advanced. SVM uses algorithms to train and classify text within our sentiment polarity model, taking it a step beyond *X/Y* prediction.

For a simple visual explanation, we'll use two tags: *red* and *blue*, with two data features: *X* and *Y*. We'll train our classifier to output an *X/Y* coordinate as either *red* or *blue*.

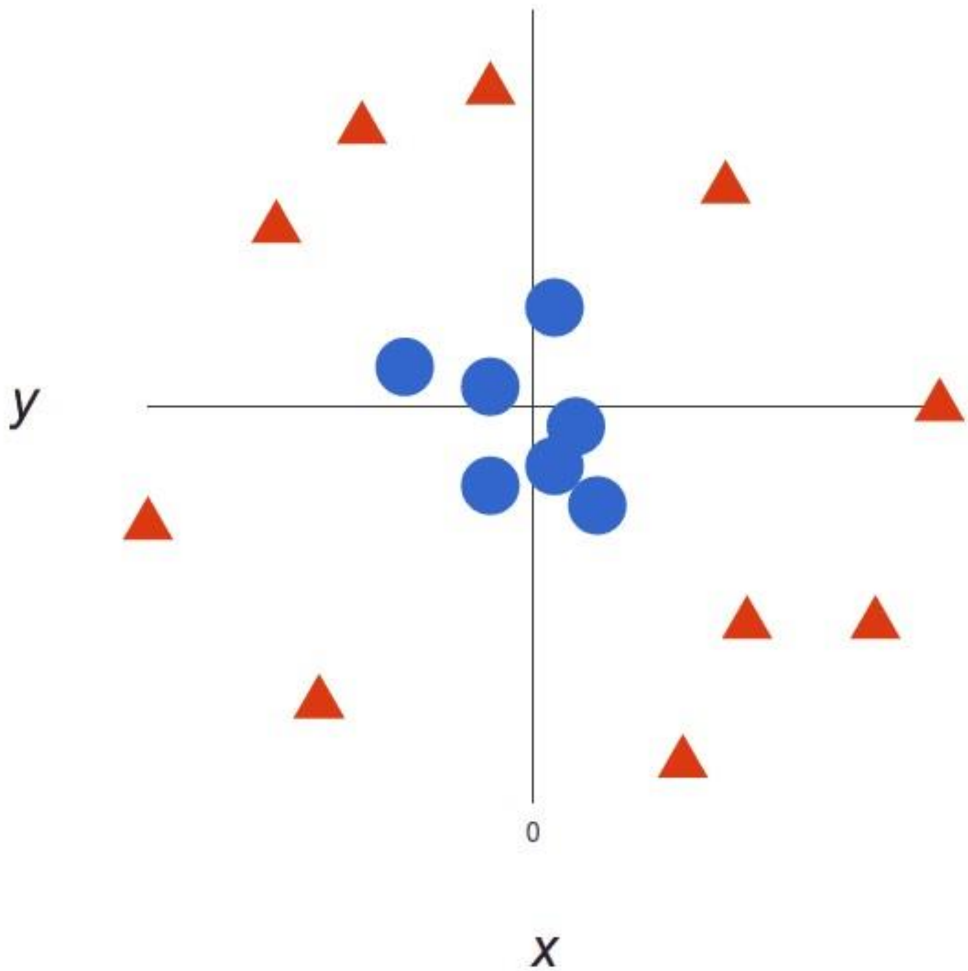


The SVM then assigns a hyperplane that best separates the tags. In two dimensions this is simply a line (like in linear regression). Anything on one side of the line is *red* and anything on the other side is *blue*. For sentiment analysis this would be *positive* and *negative*.

In order to maximize machine learning, the best hyperplane is the one with the largest distance between each tag:

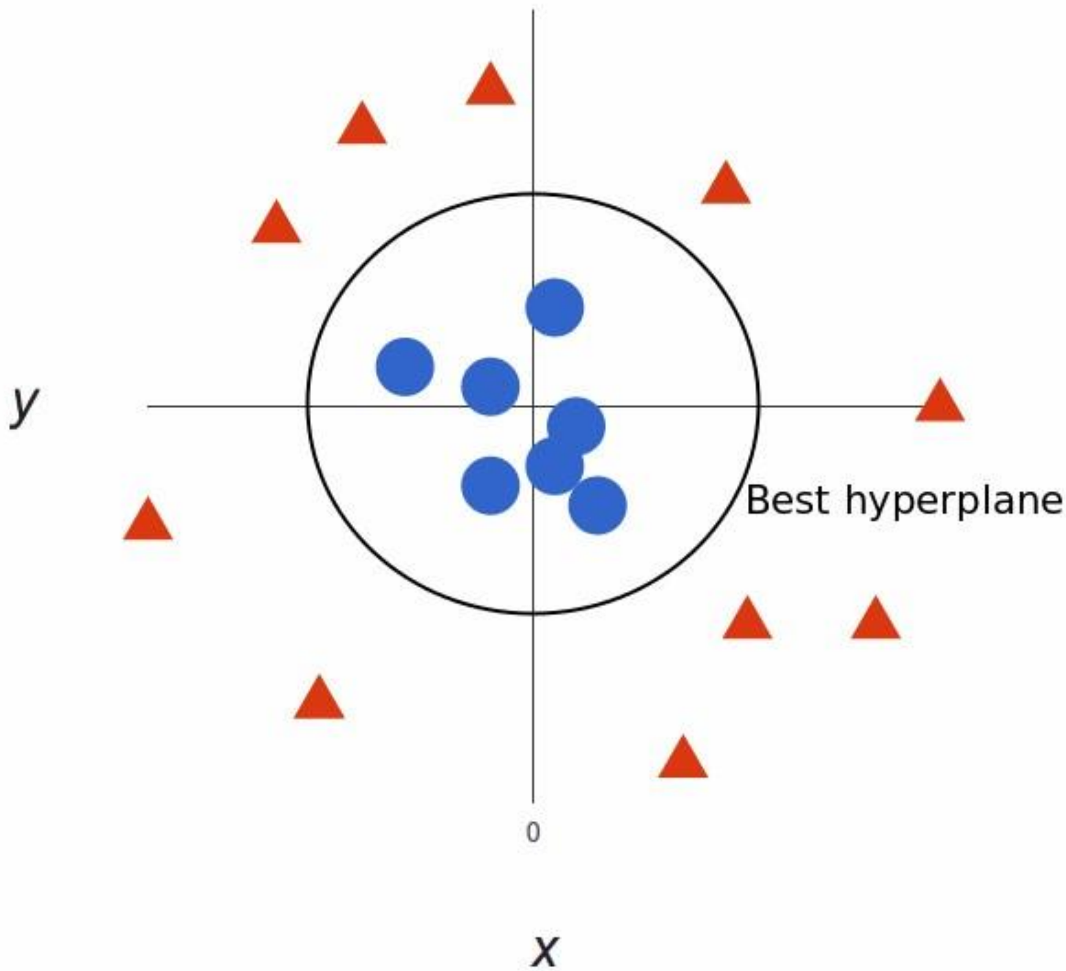


However, as data sets become more complex, it may not be possible to draw a single line to classify the data into two camps:



Using SVM, the more complex the data, the more accurate the predictor will become. Imagine the above in three dimensions, with a *Z* axis added, so it becomes a circle.

Mapped back to two dimensions with the best hyperplane, it looks like this:



Very simply put, SVM allows for more accurate machine learning because it's multidimensional.

Deep Learning

Deep learning is a subfield of machine learning that aims to calculate data as the human brain does using "artificial neural networks."

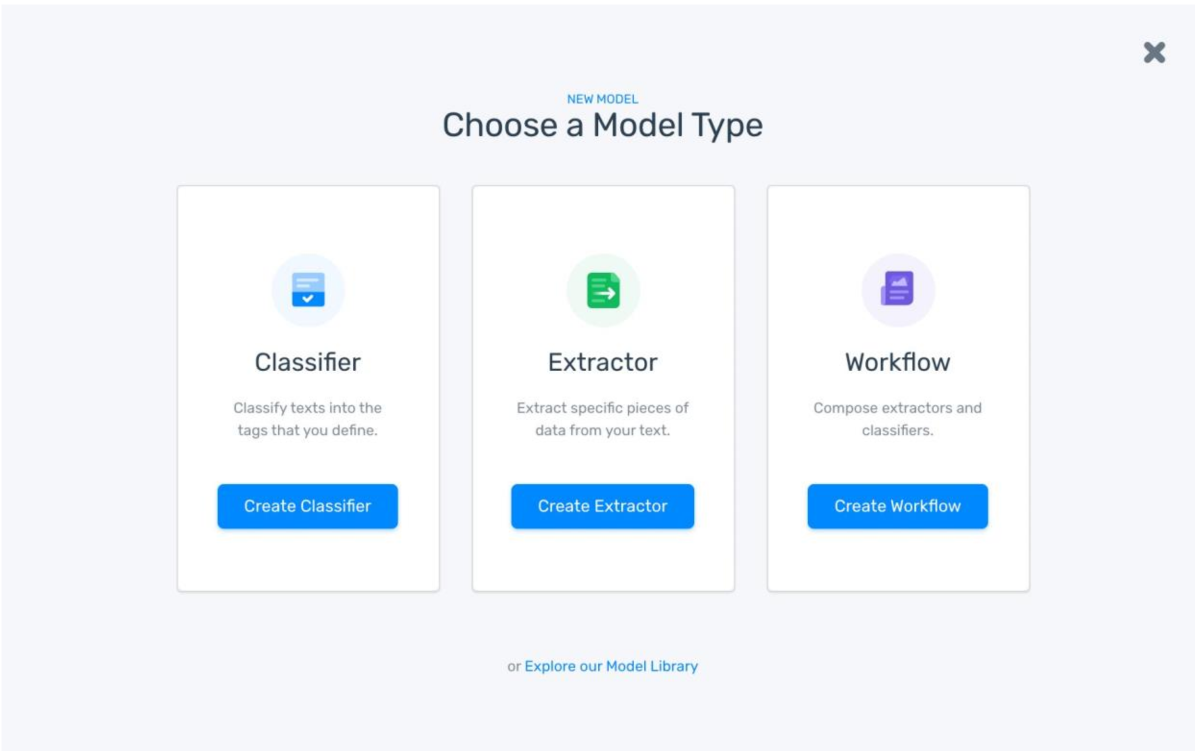
Deep learning is *hierarchical* machine learning. In other words, it's multi-level, and allows a machine to automatically 'chain' a number of human-created processes together. By allowing multiple algorithms to be used progressively, while moving from step to step, deep learning is able to solve complex problems in the same way humans do.

Sentiment Analysis with Machine Learning Tutorial

As you can see from the above, the calculations and algorithms involved in sentiment analysis are quite complex. But with user-friendly tools, sentiment analysis with machine learning is accessible to everyone, whether you have a computer science background or not.

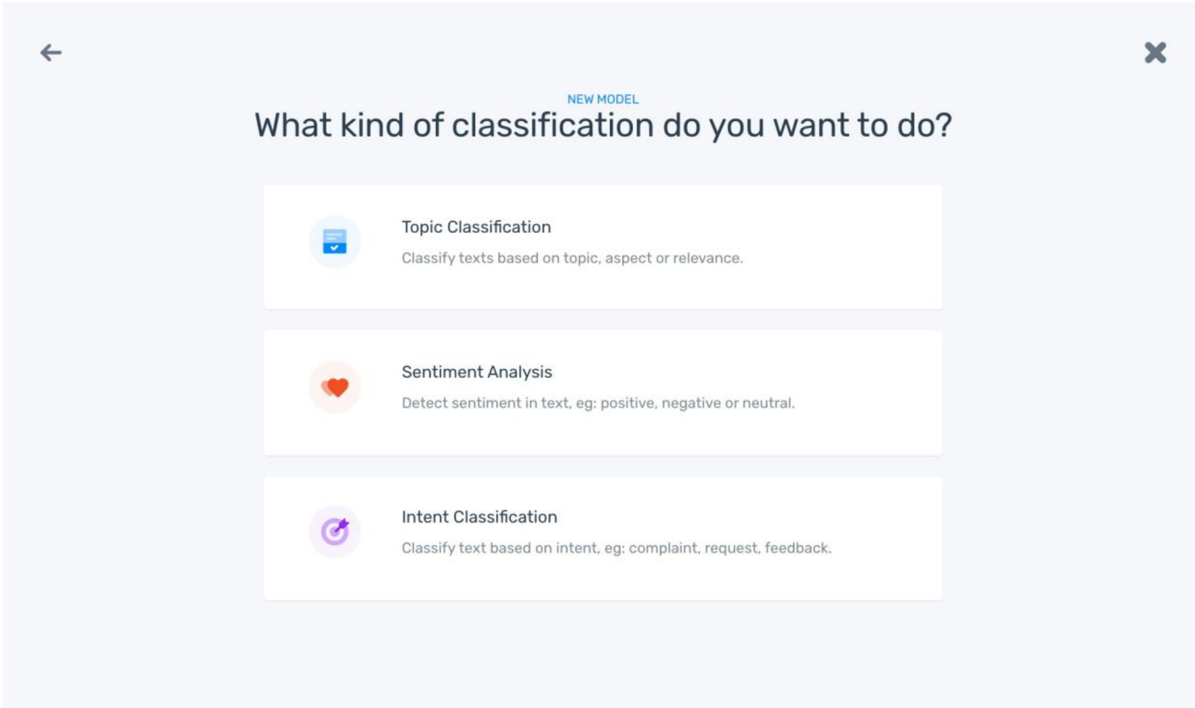
1. Choose your model

Once you've signed up, go to the dashboard and choose '*Create a model*', then click '*Classifier,*':



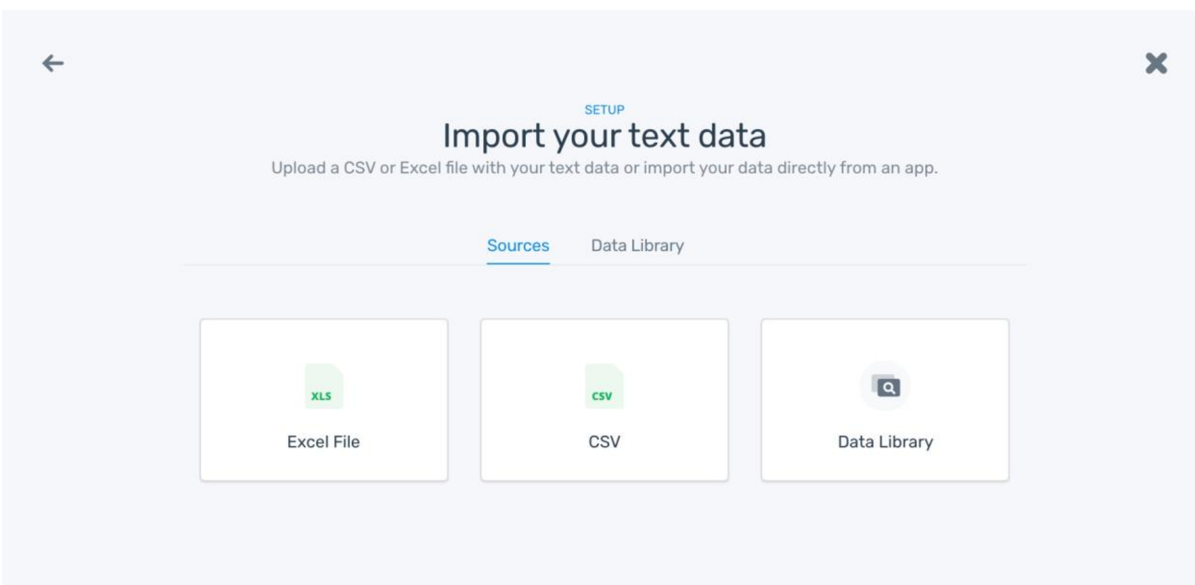
2. Choose your classifier

We want to show how machine learning works oncustomer opinions, so click on '*Sentiment Analysis*':



3. Import your data

You can import data from an app or upload a CSV or Excel file. This will be used to train your sentiment analysis model.

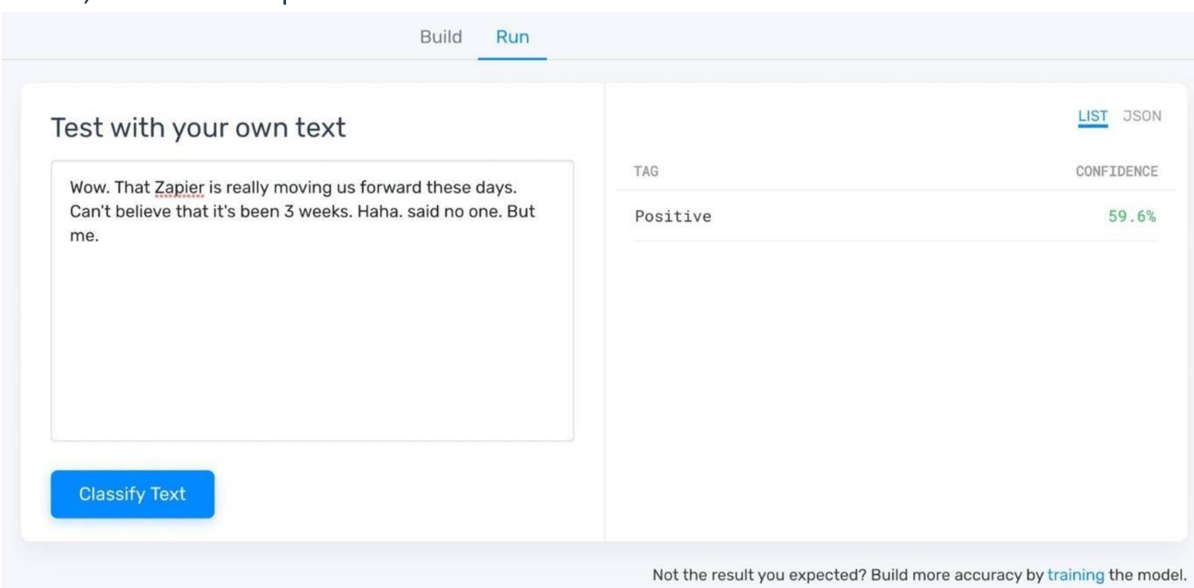


4. Tag tweets to train your sentiment analysis classifier

Here's where we see machine learning at work. Tag each tweet as *Positive*, *Negative*, or *Neutral* to train your model based on the opinion within the text. Once you tag a few, the model will begin making its own predictions. Correct them, if the model has tagged them wrong:

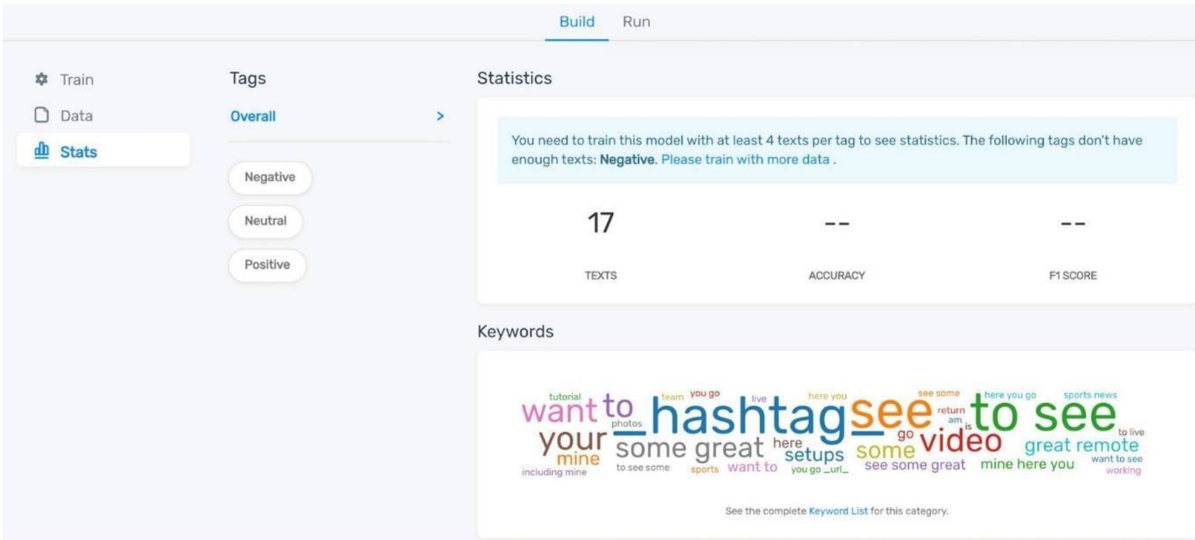
5. Test your classifier

Once the model has been trained with some examples, you can paste your own text to see how they're classified. If it's not tagging correctly, you can keep training. The more you train the model, the better it's predictions will become:



MonkeyLearn shows a number of [sentiment analysis statistics] to help understand how well machine learning is working: *Precision* and *Recall* are tag level statistics, and *Accuracy* and *F1 Score* are statistics on the overall model. The keyword cloud helps visualize the most used words.

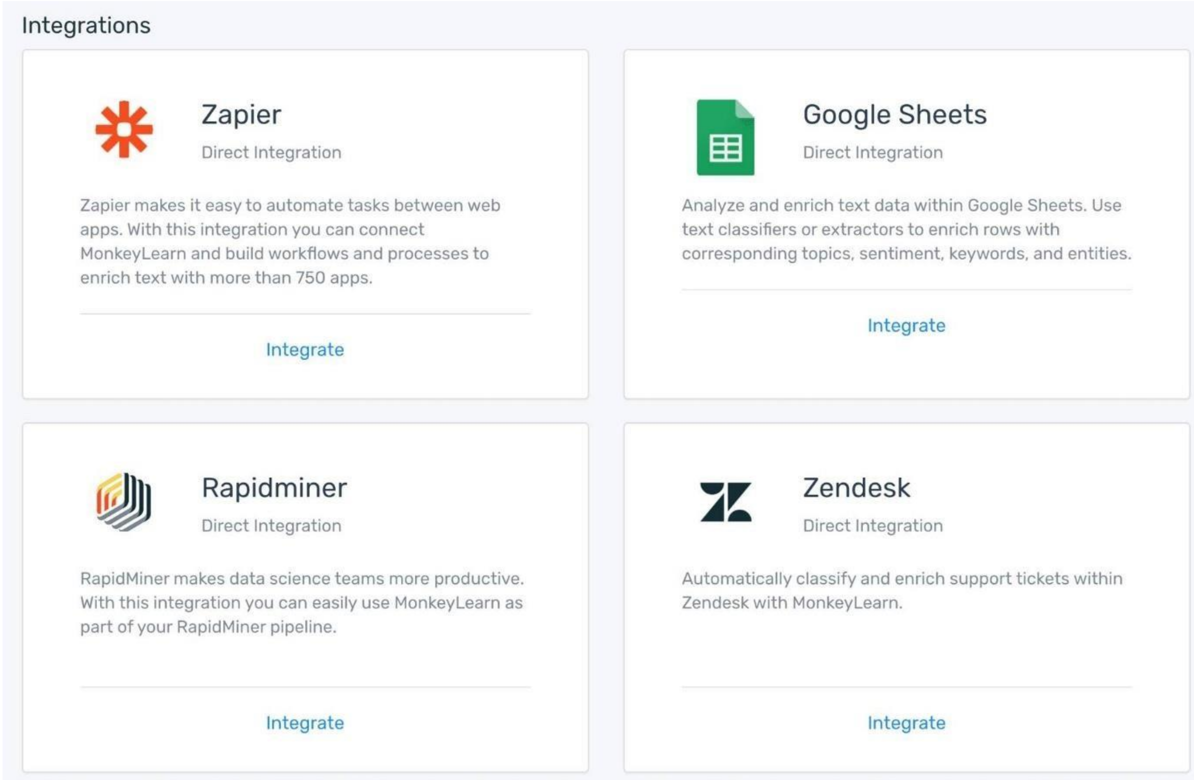
In the example below more tags are needed for *Negative.*



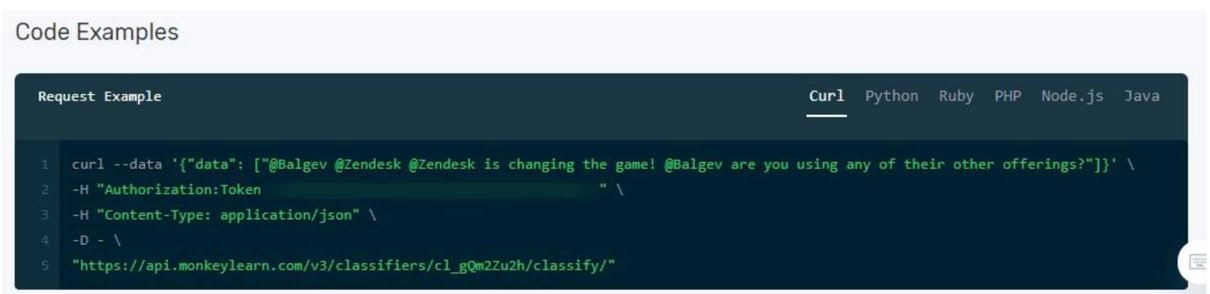
6. Put your machine learning to work

Once your model is trained, you can upload huge amounts of data. MonkeyLearn offers three ways to upload your data:

- Batch Analysis: upload a CSV or Excel file with new text. MonkeyLearn will process the data and provide your sentiment results.
- Integrations: offers simple integrations with apps you probably already use:



API: easy programming for quick plug-in analysis:



Put Machine Learning to Work for You

Sentiment analysis using machine learning can help any business analyze public opinion, improve customer support, and automate tasks with fast turnarounds. Not only saving you time, but also money. Sentiment analysis results will also give you real actionable insights, helping you make the right decisions.

While machine learning can be complex, SaaS tools make it simple for everyone to use.

Tools are also completely scalable, and can be effortlessly configured to your specific needs.

Example:

A Twitter sentiment analysis determines negative, positive, or neutral emotions within the text of a tweet using NLP and ML models. Sentiment analysis or opinion mining refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of people on social media for a variety of topics.

What is Twitter Sentiment Analysis?

Twitter sentiment analysis analyzes the sentiment or emotion of tweets. It uses natural language processing and machine learning algorithms to classify tweets automatically as positive, negative, or neutral based on their content. It can be done for individual tweets or a larger dataset related to a particular topic or event.

Why is Twitter Sentiment Analysis Important?

1. **Understanding Customer Feedback:** By analyzing the sentiment of customer feedback, companies can identify areas where they need to improve their products or services.
2. **Reputation Management**: Sentiment analysis can help companies monitor their brand reputation online and quickly respond to negative comments or reviews.

3. **Political Analysis**: Sentiment analysis can help political campaigns understand public opinion and tailor their messaging accordingly.
4. **Crisis Management:** In the event of a crisis, sentiment analysis can help organizations monitor social media and news outlets for negative sentiment and respond appropriately.
5. **Marketing Research:** Sentiment analysis can help marketers understand consumer behavior and preferences, and develop targeted advertising campaigns.

How to Do Twitter Sentiment Analysis?

In this article, we aim to analyze Twitter sentiment analysis using machine learning algorithms, the sentiment of tweets provided from the **Sentiment140 dataset** by developing a machine learning pipeline involving the use of three classifiers (**Logistic Regression, Bernoulli Naive Bayes, and SVM**)along with using **Term Frequency-Inverse Document Frequency** (**TF-IDF**). The performance of these classifiers is then evaluated using **accuracy** and **F1 Scores**.

For data preprocessing, we will be using Natural Language Processing's (NLP) NLTK library.

Twitter Sentiment Analysis: Problem Statement

In this project, we try to implement an NLP **Twitter sentiment analysis model** that helps to overcome the challenges of sentiment classification of tweets. We will be classifying the tweets into positive or negative sentiments. The necessary details regarding the dataset involving the Twitter sentiment analysis project are:

The dataset provided is the **Sentiment140 Dataset** which consists of **1,600,000 tweets** that have been extracted using the Twitter API. The various columns present in this Twitter data are:

- **target:** the polarity of the tweet (positive or negative)
- **ids:** Unique id of the tweet
- **date:** the date of the tweet
- **flag:** It refers to the query. If no such query exists, then it is NO QUERY.
- **user:** It refers to the name of the user that tweeted
- **text:** It refers to the text of the tweet

Twitter Sentiment Analysis: Project Pipeline

The various steps involved in the **Machine Learning Pipeline** are:

- Import Necessary Dependencies
- Read and Load the Dataset
- Exploratory Data Analysis
- Data Visualization of Target Variables
- Data Preprocessing
- Splitting our data into Train and Test sets.
- Transforming Dataset using TF-IDF Vectorizer
- Function for Model Evaluation
- Model Building
- Model Evaluation

Let’s get started,

Step-1: Import the Necessary Dependencies

```
# utilities import re import
numpy as np import pandas as pd
# plotting import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# nltk
from nltk.stem import WordNetLemmatizer
# sklearn
from sklearn.svm import LinearSVC from sklearn.naive_bayes
import BernoulliNB from sklearn.linear_model import
LogisticRegression from sklearn.model_selection import
train_test_split from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report
```

Step-2: Read and Load the Dataset

```
# Importing the dataset
DATASET_COLUMNS=['target','ids','date','flag','user','text'] DATASET_ENCODING
= "ISO-8859-1"
df = pd.read_csv('Project_Data.csv', encoding=DATASET_ENCODING,
names=DATASET_COLUMNS) df.sample(5)
```

Output:

	target	ids	date	flag	user	text
305165	0	1999924339	Mon Jun 01 21:04:24 PDT 2009	NO_QUERY	twentyred25	man my b-day is coming up but i dont know what...
673186	0	2247413241	Fri Jun 19 19:03:37 PDT 2009	NO_QUERY	belenneleb	@officialutl_ i need they to come back here. ...
573387	0	2209824277	Wed Jun 17 10:50:29 PDT 2009	NO_QUERY	TeenieWahine	@krystyn13 Sorry to hear that
246882	0	1982346860	Sun May 31 11:01:36 PDT 2009	NO_QUERY	BrianWCollins	@TheJoeLynch I've only seen 3 (Leon, 5th Eleme...
669112	0	2246153162	Fri Jun 19 17:10:15 PDT 2009	NO_QUERY	heidioftheopera	kinda feels bad for missing out on the Solstic...

Step-3: Exploratory Data Analysis

3.1: Five top records of data

`df.head()` **Output:**

	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

3.2: Columns/features in data

`df.columns` **Output:**

```
Index(['target', 'ids', 'date', 'flag', 'user', 'text'], dtype='object')
```

3.3: Length of the dataset

```
print('length of data is', len(df))
```

Output: `length of data is 1048576`

3.4: Shape of data `df. shape`

Output:

```
(1048576, 6)
```

3.5: Data information

`df.info()` **Output:**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048576 entries, 0 to 1048575
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   target  1048576 non-null  int64
 1   ids     1048576 non-null  int64
 2   date    1048576 non-null  object
 3   flag    1048576 non-null  object
 4   user    1048576 non-null  object
 5   text    1048576 non-null  object
dtypes: int64(2), object(4)
memory usage: 48.0+ MB
```

3.6: Datatypes of all columns

`df.dtypes` **Output:**

```
target     int64 ids
int64 date
object flag
```

```
object user
object text
object dtype: object
```

3.7: Checking for null values

```
np.sum(df.isnull().any(axis=1))
```
Output:

```
0
```

3.8: Rows and columns in the dataset

```
print('Count of columns in the data is:  ', len(df.columns)) print('Count of rows in the data is:  ', len(df))
```
Output:

```
Count of columns in the data is:   6
Count of rows in the data is:   1048576
```

3.9: Check unique target values

```
df['target'].unique()
```
Output:

```
array([0, 4], dtype=int64)
```

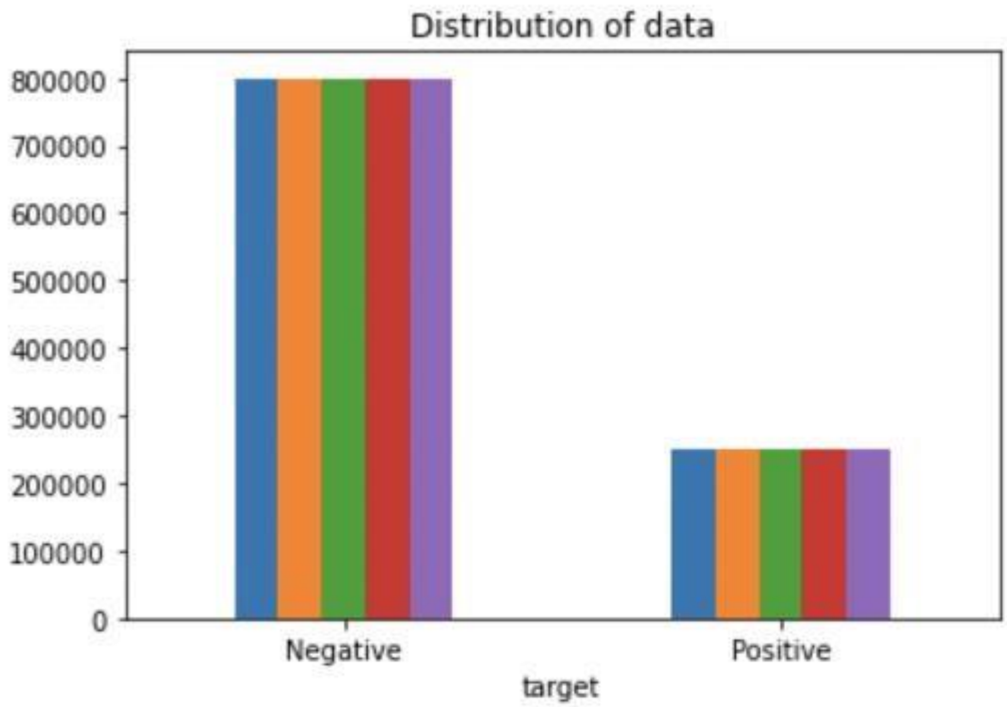
3.10: Check the number of target values

```
df['target'].nunique()
```
Output:

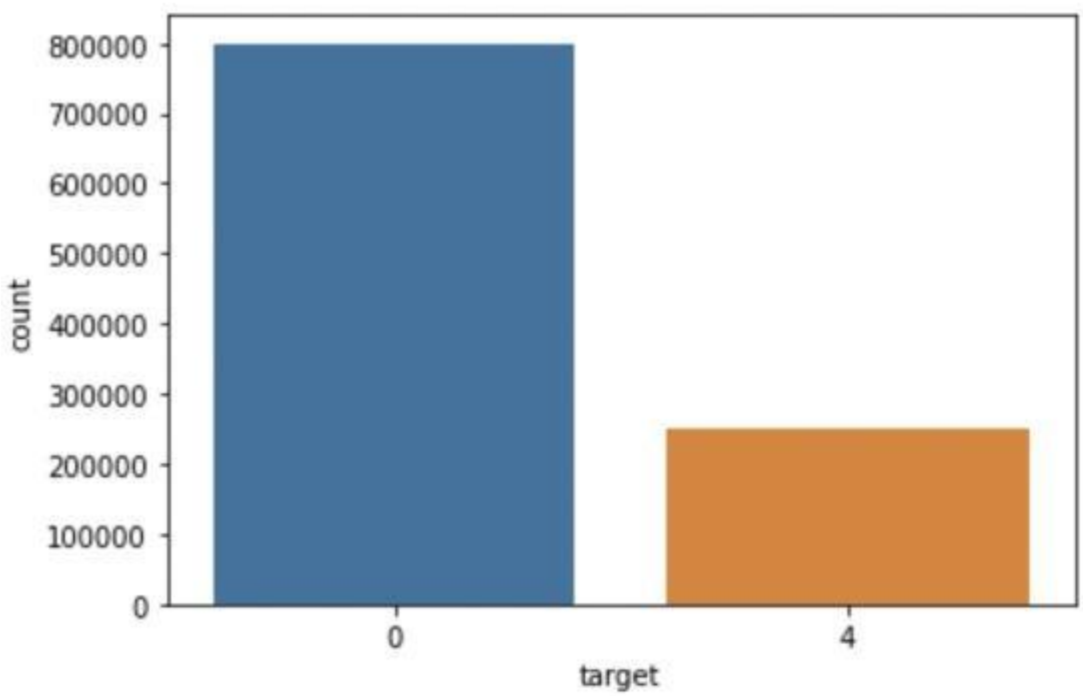
```
2
```

Step-4: Data Visualization of Target Variables

```
# Plotting the distribution for dataset.
ax = df.groupby('target').count().plot(kind='bar', title='Distribution of
data',legend=False)
ax.set_xticklabels(['Negative','Positive'], rotation=0) #
Storing data in lists.
text, sentiment = list(df['text']), list(df['target'])
```
Output:




```
import seaborn as sns
sns.countplot(x='target', data=df)
```
Output:



Step-5: Data Preprocessing

In the above-given problem statement, before training the model, we performed various pre-processing steps on the dataset that mainly dealt with removing stopwords, removing special characters like emojis, hashtags, etc. The text document is then converted into lowercase for better generalization.

Subsequently, the punctuations were cleaned and removed, thereby reducing the unnecessary noise from the dataset. After that, we also removed the repeating characters from the words along with removing the URLs as they do not have any significant importance.

At last, we then performed **Stemming(reducing the words to their derived stems)** and **Lemmatization(reducing the derived words to their root form, known as lemma)** for better results.

5.1: Selecting the text and Target column for our further analysis

```
data=df[['text','target']]
```

5.2: Replacing the values to ease understanding. (Assigning 1 to Positive sentiment 4) `data['target'] = data['target'].replace(4,1)` ***5.3: Printing unique values of target variables*** `data['target'].unique()` **Output:** `array([0, 1], dtype=int64)`

5.4: Separating positive and negative tweets

```
data_pos = data[data['target'] == 1] data_neg
= data[data['target'] == 0]
```

5.5: Taking one-fourth of the data so we can run it on our machine easily

```
data_pos = data_pos.iloc[:int(20000)] data_neg = data_neg.iloc[:int(20000)]
```

5.6: Combining positive and negative tweets

`dataset = pd.concat([data_pos, data_neg])` ***5.7: Making statement text in lowercase***

```
dataset['text']=dataset['text'].str.lower()
dataset['text'].tail()
```
Output:

```
19995    not much time off this weekend, work trip to m...
19996                           one more day of holidays
19997    feeling so down right now .. i hate you damn h...
19998    geez,i hv to read the whole book of personalit...
19999    i threw my sign at donnie and he bent over to ...
Name: text, dtype: object
```

5.8: Defining set containing all stopwords in English.

```
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 'am', 'an',
'and','any','are', 'as', 'at', 'be', 'because', 'been', 'before',
             'being', 'below', 'between','both', 'by', 'can', 'd', 'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once',
'only', 'or', 'other', 'our', 'ours','ourselves', 'out', 'own',
're','s', 'same', 'she', "shes", 'should', "shouldve",'so', 'some', 'such',
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs', 'them',
             'themselves', 'then', 'there', 'these', 'they', 'this', 'those',
             'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while', 'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll", "youre",
"youve", 'your', 'yours', 'yourself', 'yourselves']
```

5.9: Cleaning and removing the above stop words list from the tweet text

```
STOPWORDS = set(stopwordlist) def cleaning_stopwords(text):     return "
".join([word for word in str(text).split() if word not in STOPWORDS])
dataset['text'] = dataset['text'].apply(lambda text: cleaning_stopwords(text))
dataset['text'].head()
```
Output:

```
800000                   love @health4uandpets u guys r best!!
800001    im meeting one besties tonight! cant wait!! - ...
800002    @darealsunisakim thanks twitter add, sunisa! g...
800003    sick really cheap hurts much eat real food plu...
800004                      @lovesbrooklyn2 effect everyone
Name: text, dtype: object
```

5.10: Cleaning and removing punctuations

```
import string
```

```
english_punctuations = string.punctuation
punctuations_list = english_punctuations def
cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
return text.translate(translator)
dataset['text']= dataset['text'].apply(lambda x: cleaning_punctuations(x))
dataset['text'].tail()
```

Output:

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                   one day holidays
19997                      feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: text, dtype: object
```

5.11: Cleaning and removing repeating characters

```
def cleaning_repeating_char(text):
    return re.sub(r'(.)1+', r'1', text)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_repeating_char(x))
dataset['text'].tail()
```

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                   one day holidays
19997                         feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.12: Cleaning and removing URLs

```
def cleaning_URLs(data):     return
re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data) dataset['text'] =
dataset['text'].apply(lambda x: cleaning_URLs(x))
dataset['text'].tail()
```

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                   one day holidays
19997                         feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.13: Cleaning and removing numeric numbers

```
def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)
dataset['text'] = dataset['text'].apply(lambda x: cleaning_numbers(x))
dataset['text'].tail()
```

Output:

```
19995    not much time of wekend work trip malmï¿½ fris...
19996                                      one day holidays
19997                         feling right hate damn humprey
19998    gezi hv read whole bok personality types embar...
19999    threw sign donie bent over get but thinge made...
Name: text, dtype: object
```

5.14: Getting tokenization of tweet text

```
from nltk.tokenize import RegexpTokenizer tokenizer
= RegexpTokenizer(r'w+')
dataset['text'] = dataset['text'].apply(tokenizer.tokenize) dataset['text'].head()
```

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001     [im, meting, one, besties, tonight, cant, wait...
800002     [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003     [sick, realy, cheap, hurts, much, eat, real, f...
800004                     [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

5.15: Applying stemming

```
import nltk st = nltk.PorterStemmer() def
stemming_on_text(data):     text =
[st.stem(word) for word in data]     return
data
dataset['text']= dataset['text'].apply(lambda x: stemming_on_text(x))
dataset['text'].head()
```

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001     [im, meting, one, besties, tonight, cant, wait...
800002     [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003     [sick, realy, cheap, hurts, much, eat, real, f...
800004                     [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

5.16: Applying lemmatizer

```
lm = nltk.WordNetLemmatizer() def
lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
return data
dataset['text'] = dataset['text'].apply(lambda x: lemmatizer_on_text(x))
dataset['text'].head()
```

Output:

```
800000              [love, healthuandpets, u, guys, r, best]
800001    [im, meting, one, besties, tonight, cant, wait...
800002    [darealsunisakim, thanks, twiter, ad, sunisa, ...
800003    [sick, realy, cheap, hurts, much, eat, real, f...
800004                     [lovesbroklyn, efect, everyone]
Name: text, dtype: object
```

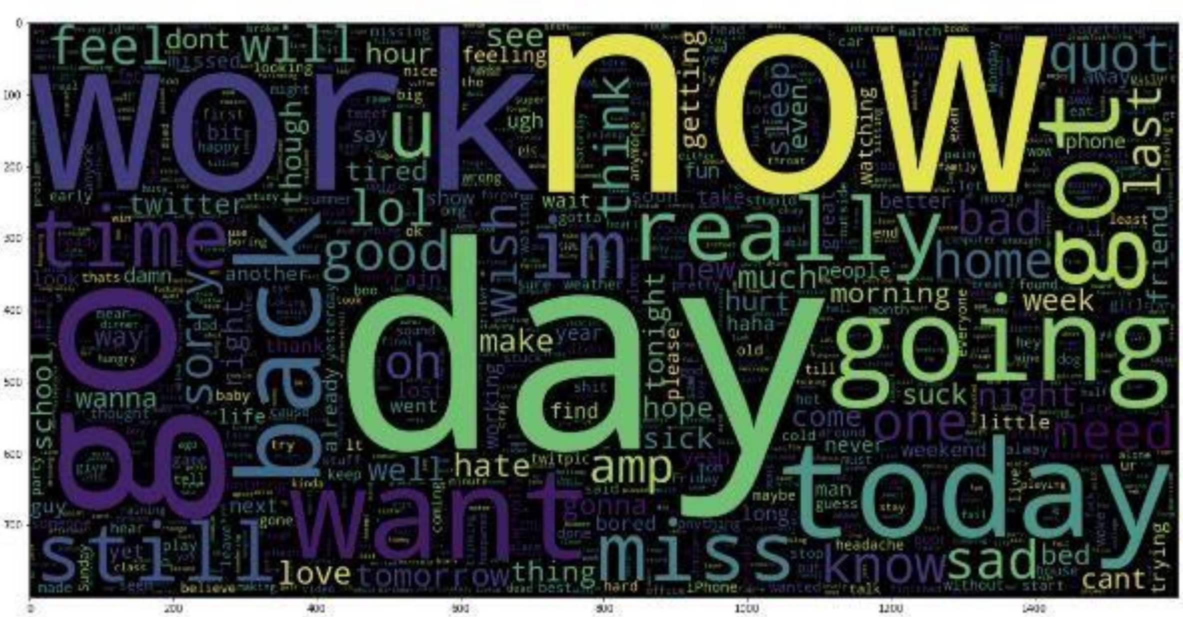
5.17: Separating input feature and label

```
X=data.text y=data.target
```

5.18: Plot a cloud of words for negative tweets

```
data_neg = data['text'][:800000] plt.figure(figsize
= (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
collocations=False).generate(" ".join(data_neg)) plt.imshow(wc)
```

Output:



5.19: Plot a cloud of words for positive tweets

```
data_pos = data['text'][800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20)) plt.imshow(wc)
```

Output:

Step-6: Splitting Our Data Into Train and Test Subsets

```
# Separating the 95% data for training data and 5% for testing data X_train,
X_test, y_train, y_test = train_test_split(X,y,test_size = 0.05, random_state
=26105111)
```

Step-7: Transforming the Dataset Using TF-IDF Vectorizer

7.1: Fit the TF-IDF Vectorizer

```
vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train) print('No. of feature_words: ',
len(vectoriser.get_feature_names()))
```

Output:

```
No. of feature_words:  500000
```

7.2: Transform the data using TF-IDF Vectorizer

```
X_train = vectoriser.transform(X_train) X_test
= vectoriser.transform(X_test)
```

Step-8: Function for Model Evaluation

After training the model, we then apply the evaluation measures to check how the model is performing. Accordingly, we use the following evaluation parameters to check the performance of the models respectively:

- Accuracy Score
- Confusion Matrix with Plot
- ROC-AUC Curve

```
def model_Evaluate(model): #
Predict values for Test dataset
y_pred = model.predict(X_test)
# Print the evaluation metrics for the dataset.
print(classification_report(y_test, y_pred)) #
Compute and plot the Confusion matrix cf_matrix
= confusion_matrix(y_test, y_pred) categories =
['Negative','Positive']
group_names = ['True Neg','False Pos', 'False Neg','True Pos']
group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() /
np.sum(cf_matrix)]
labels = [f'{v1}n{v2}' for v1, v2 in zip(group_names,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '', xticklabels
= categories, yticklabels = categories)
plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)
plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Step-9: Model Building

In the problem statement, we have used three different models respectively :

- Bernoulli Naive Bayes Classifier
- SVM (Support Vector Machine)
- Logistic Regression

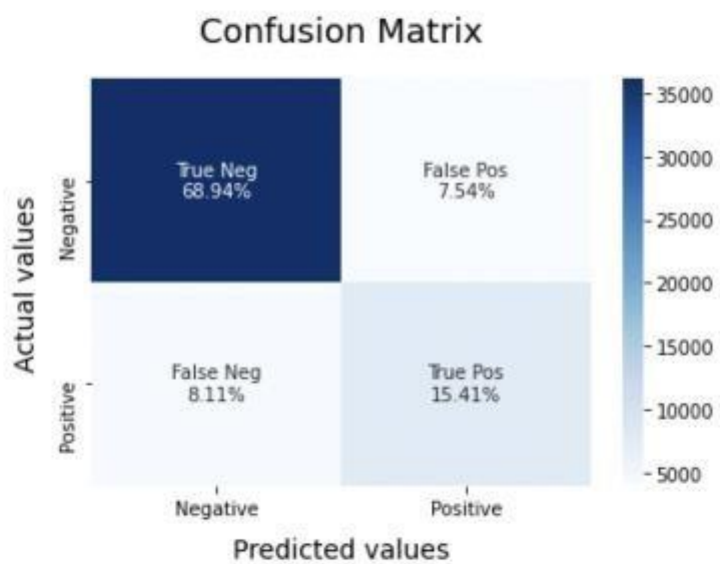
The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models, and then try to find out the one which gives the best performance among them.

8.1: Model-1

```
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel) y_pred1 =
BNBmodel.predict(X_test)
```

Output:

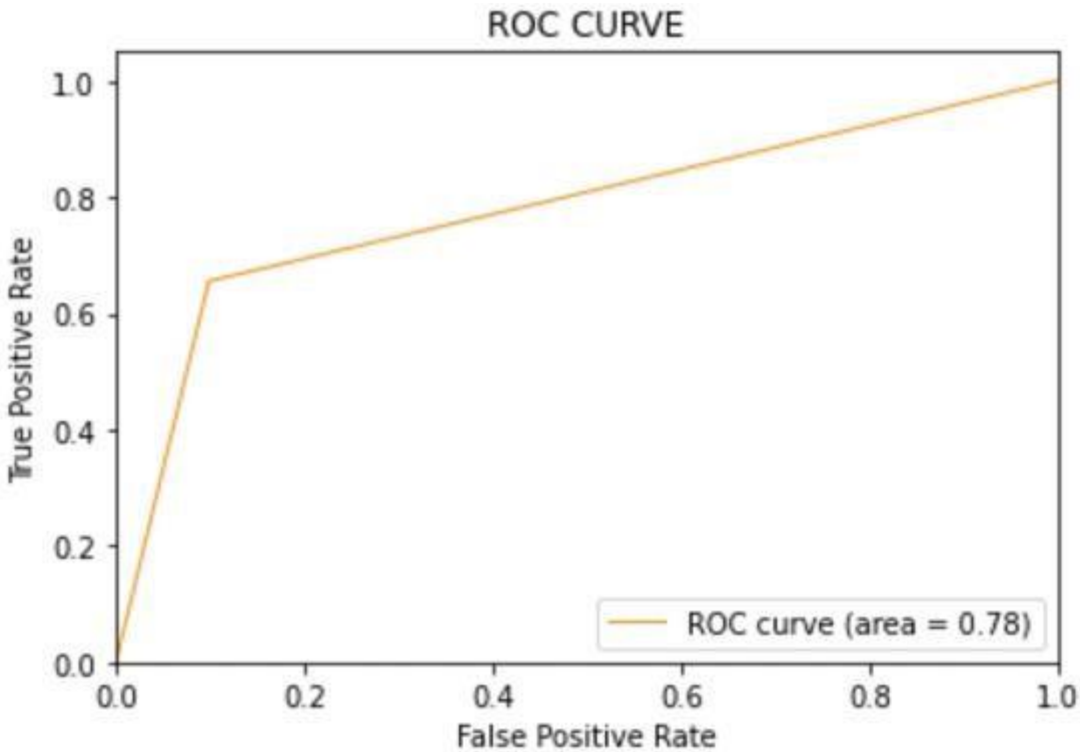
```
              precision    recall  f1-score   support

           0       0.89      0.90      0.90     40097
           1       0.67      0.66      0.66     12332

    accuracy                           0.84     52429
   macro avg       0.78      0.78      0.78     52429
weighted avg       0.84      0.84      0.84     52429
```



8.2: Plot the ROC-AUC Curve for model-1

```
from sklearn.metrics import roc_curve, auc fpr,
tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr) plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' %
roc_auc) plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive
Rate') plt.ylabel('True Positive Rate') plt.title('ROC CURVE')
plt.legend(loc="lower right") plt.show()
```

Output:

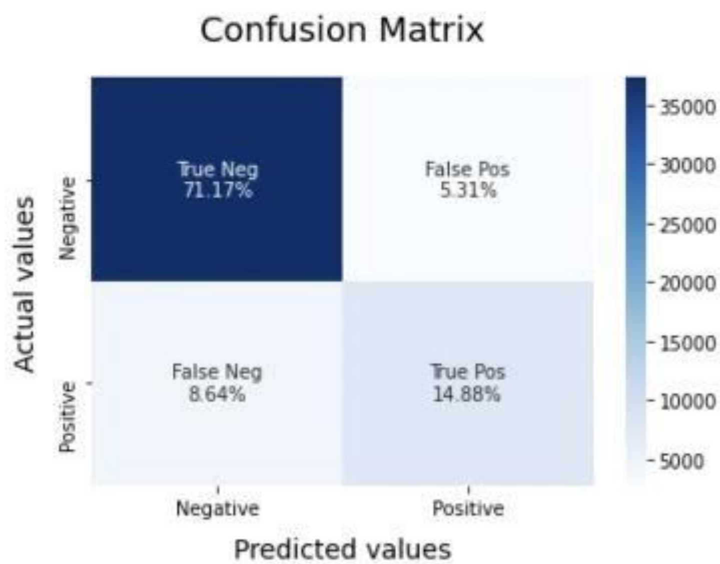


8.3: Model-2:

```
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel) y_pred2 =
SVCmodel.predict(X_test)
```

Output:

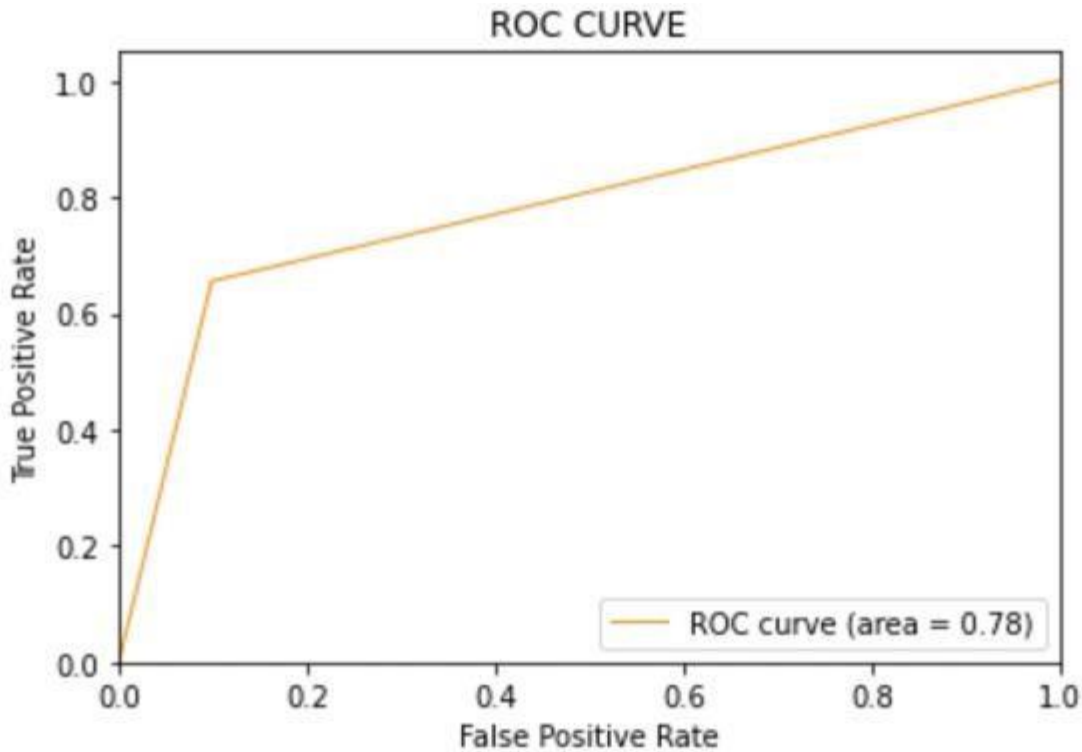
	precision	recall	f1-score	support
0	0.89	0.93	0.91	40097
1	0.74	0.63	0.68	12332
accuracy			0.86	52429
macro avg	0.81	0.78	0.80	52429
weighted avg	0.86	0.86	0.86	52429



8.4: Plot the ROC-AUC Curve for model-2

```
from sklearn.metrics import roc_curve, auc fpr,
tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr) plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' %
roc_auc) plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive
Rate') plt.ylabel('True Positive Rate') plt.title('ROC CURVE')
plt.legend(loc="lower right") plt.show()
```

Output:

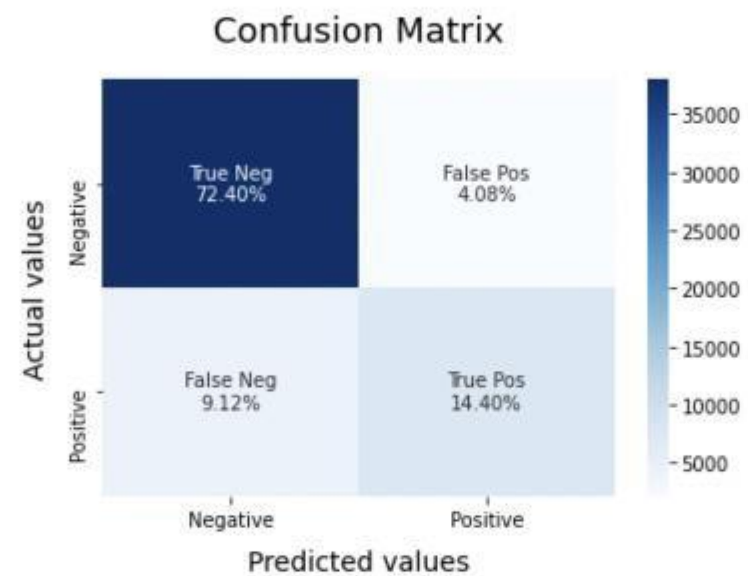


8.5: Model-3

```
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel) y_pred3 =
LRmodel.predict(X_test)
```

Output:

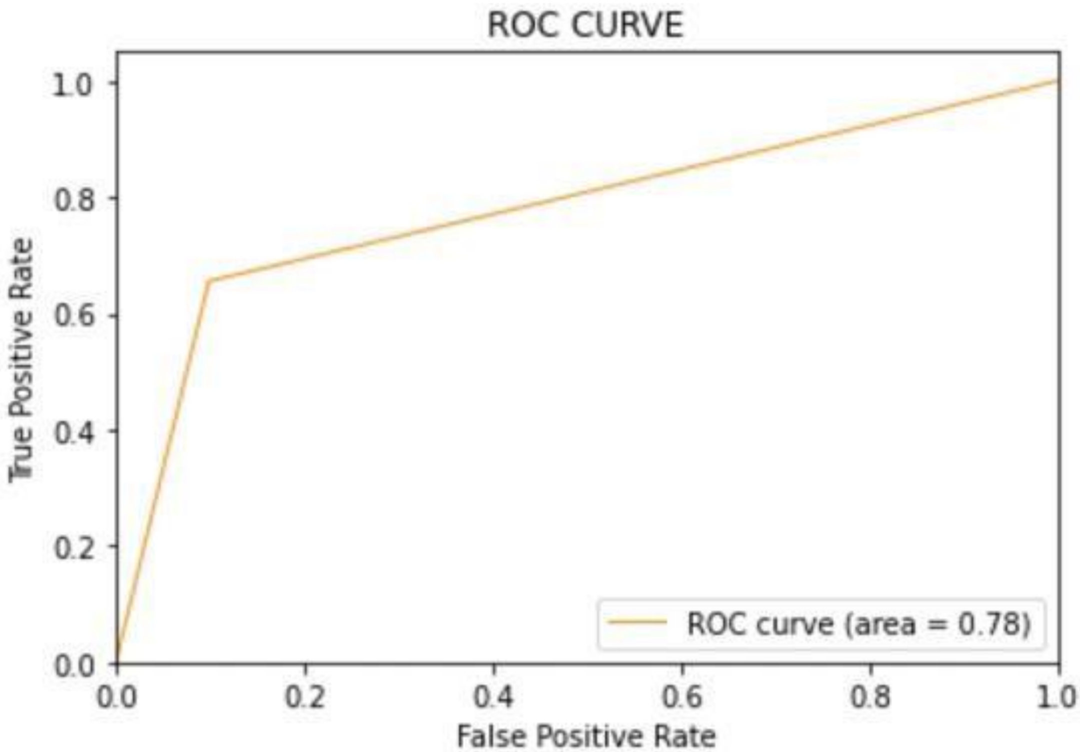
	precision	recall	f1-score	support
0	0.89	0.95	0.92	40097
1	0.78	0.61	0.69	12332
accuracy			0.87	52429
macro avg	0.83	0.78	0.80	52429
weighted avg	0.86	0.87	0.86	52429



8.6: Plot the ROC-AUC Curve for model-3

```
from sklearn.metrics import roc_curve, auc fpr,
tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr) plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' %
roc_auc) plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05]) plt.xlabel('False Positive
Rate') plt.ylabel('True Positive Rate') plt.title('ROC CURVE')
plt.legend(loc="lower right") plt.show()
```

Output:



Step-10: Model Evaluation

Upon evaluating all the models, we can conclude the following details i.e.

Accuracy: As far as the accuracy of the model is concerned, Logistic Regression performs better than SVM, which in turn performs better than Bernoulli Naive Bayes.

F1-score: The F1 Scores for class 0 and class 1 are :
(a) For class 0: Bernoulli Naive Bayes(accuracy = 0.90) < SVM (accuracy =0.91) < Logistic Regression (accuracy = 0.92)
(b) For class 1: Bernoulli Naive Bayes (accuracy = 0.66) < SVM (accuracy = 0.68) < Logistic Regression (accuracy = 0.69)

AUC Score: All three models have the same ROC-AUC score.

We, therefore, conclude that the Logistic Regression is the best model for the abovegiven dataset.

In our problem statement, **Logistic Regression** follows the principle of **Occam's Razor,** which defines that for a particular problem statement, if the data has no assumption, then the simplest model works the best. Since our dataset does not have any assumptions and Logistic Regression is a simple model. Therefore, the concept holds true for the above-mentioned dataset.

Conclusion

We hope through this article, you got a basic of how Sentimental Analysis is used to understand public emotions behind people's tweets. As you've read in this article, Twitter Sentimental Analysis helps us preprocess the data (tweets) using different methods and feed it into ML models to give the best accuracy.

Key Takeaways

- Twitter Sentimental Analysis is used to identify as well as classify the sentiments that are expressed in the text source.
- Logistic Regression, SVM, and Naive Bayes are some of the ML algorithms that can be used for Twitter Sentimental Analysis.