import pandas as pd

import re

df = pd.read\_csv("training\_data.csv", encoding='latin-1')

df2 = df.sample(n = 1500, random\_state = 30)

df3 = df2[['text','Target']]

p1 = re.compile(r"@\w+\s")

p2 = re.compile(r"\W+")

tweet = df3['text']

tweet = tweet.replace({p1: ""}, regex=True)

tweet = tweet.replace({p2: " "}, regex=True)

df3['text'] = tweet

print(df3)

df3.loc[df3['Target'] > 3, 'Target'] = 1

df3.loc[df3['Target'] < 1, 'Target'] = -1

df3.loc[df3['Target'] == 2, 'Target'] = 0

print(df3)

#data has been cleaned and formatted

#making train and test data

#x = independant which in this case is the column 'text'

#y = depedant which in this case is the column 'Target'

from sklearn.model\_selection import train\_test\_split

X = df['text']

y = df['Target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20)

from sklearn.svm import SVC

from numpy import array

from numpy import argmax

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import OneHotEncoder

# values = array(X\_train)

# print(values)

# # integer encode

# label\_encoder = LabelEncoder()

# integer\_encoded = label\_encoder.fit\_transform(values)

# print(integer\_encoded)

# # binary encode

# onehot\_encoder = OneHotEncoder(sparse=False)

# integer\_encoded = integer\_encoded.reshape(len(integer\_encoded), 1)

# onehot\_encoded = onehot\_encoder.fit\_transform(integer\_encoded)

# print(onehot\_encoded)

#training algo

# svclassifier = SVC(kernel='linear')

# svclassifier.fit(X\_train, y\_train)

#testing algo

y\_pred = svclassifier.predict(X\_test)

print(y\_pred)

**CODE FROM HERE:** [**https://www.kaggle.com/lbronchal/sentiment-analysis-with-svm**](https://www.kaggle.com/lbronchal/sentiment-analysis-with-svm)

#making train and test data

#x = independant which in this case is the column 'text'

#y = depedant which in this case is the column 'Target'

from sklearn.model\_selection import train\_test\_split

X = df3['text']

y = df3['Target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X.values, y.values, test\_size = 0.20, random\_state=30)

%matplotlib inline

%config InlineBackend.figure\_format = 'retina'

import numpy as np

import pandas as pd

from bs4 import BeautifulSoup

import matplotlib.pyplot as plt

import seaborn as sns

import nltk

from nltk.corpus import stopwords

from nltk.stem import SnowballStemmer

from nltk.tokenize import TweetTokenizer

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split, StratifiedKFold, cross\_val\_score

from sklearn.pipeline import make\_pipeline, Pipeline

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import make\_scorer, accuracy\_score, f1\_score

from sklearn.metrics import roc\_curve, auc

from sklearn.metrics import confusion\_matrix, roc\_auc\_score, recall\_score, precision\_score

def tokenize(text):

tknzr = TweetTokenizer()

return tknzr.tokenize(text)

def stem(doc):

return (stemmer.stem(w) for w in analyzer(doc))

en\_stopwords = set(line.strip() for line in open("stopwords.txt"))

vectorizer = CountVectorizer(

analyzer = 'word',

tokenizer = tokenize,

lowercase = True,

ngram\_range=(1, 1),

stop\_words = en\_stopwords)

kfolds = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=30)

np.random.seed(1)

pipeline\_svm = make\_pipeline(vectorizer,

SVC(probability=True, kernel="linear", class\_weight="balanced"))

grid\_svm = GridSearchCV(pipeline\_svm,

param\_grid = {'svc\_\_C': [0.01, 0.1, 1, 10]},

cv = kfolds,

scoring="roc\_auc",

verbose=1,

n\_jobs=-1)

grid\_svm.fit(X\_train, y\_train)

grid\_svm.score(X\_test, y\_test)

grid\_svm.best\_params\_