42xtiwsez

February 21, 2025

[1]:
```
# Installing kaggle library
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages
(1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-
packages (from kaggle) (1.17.0)
Requirement already satisfied: certifi>=2023.7.22 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2025.1.31)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from kaggle) (4.67.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-
packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.11/dist-
packages (from kaggle) (2.3.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages
(from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-
packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.11/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->kaggle) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->kaggle) (3.10)
```

[3]:
```
# Upload your kaggle json file
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

```
[3]: {'kaggle.json':
     b'{"username":"precioustirivanhu","key":"95473051a0441fd19820146323c863fc"}'}
```

```
[5]: # Configuring thr path of kaggle json file
     !mkdir -p ~/.kaggle
     !cp kaggle.json ~/.kaggle/
     !chmod 600 ~/.kaggle/kaggle.json
```

```
[6]: # Import Twitter Sentiment Analysis dataste
     !kaggle datasets download -d kazanova/sentiment140
```

```
Dataset URL: https://www.kaggle.com/datasets/kazanova/sentiment140
License(s): other
Downloading sentiment140.zip to /content
 90% 73.0M/80.9M [00:00<00:00, 219MB/s]
100% 80.9M/80.9M [00:00<00:00, 191MB/s]
```

```
[7]: # API for fetch the dataset from kaggle
     !unzip /content/sentiment140.zip
```

```
Archive:  /content/sentiment140.zip
  inflating: training.1600000.processed.noemoticon.csv
```

```
[10]: # Checking if the dataset is ready to be used

      dataset = '/content/training.1600000.processed.noemoticon.csv'


      print(f'The dataset {dataset} is ready to be used.')
```

```
The dataset /content/training.1600000.processed.noemoticon.csv is ready to be
used.
```

```
[15]: !pip install --upgrade scikit-learn

      # Importing the Dependencies

      import numpy as np
      import pandas as pd
      import re
      from nltk.corpus import stopwords
      from nltk.stem.porter import PorterStemmer
      from sklearn.feature_extraction.text import TfidfVectorizer
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
```

```
packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
```

[16]:
```python
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[16]:
```
True
```

[17]:
```python
# Printing the stopwords in English
print(stopwords.words('english'))
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an',
'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been',
'before', 'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn',
"couldn't", 'd', 'did', 'didn', "didn't", 'do', 'does', 'doesn', "doesn't",
'doing', 'don', "don't", 'down', 'during', 'each', 'few', 'for', 'from',
'further', 'had', 'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven',
"haven't", 'having', 'he', "he'd", "he'll", 'her', 'here', 'hers', 'herself',
"he's", 'him', 'himself', 'his', 'how', 'i', "i'd", 'if', "i'll", "i'm", 'in',
'into', 'is', 'isn', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself',
"i've", 'just', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most',
'mustn', "mustn't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'not',
'now', 'o', 'of', 'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours',
'ourselves', 'out', 'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she',
"she'd", "she'll", "she's", 'should', 'shouldn', "shouldn't", "should've", 'so',
'some', 'such', 't', 'than', 'that', "that'll", 'the', 'their', 'theirs',
'them', 'themselves', 'then', 'there', 'these', 'they', "they'd", "they'll",
"they're", "they've", 'this', 'those', 'through', 'to', 'too', 'under', 'until',
'up', 've', 'very', 'was', 'wasn', "wasn't", 'we', "we'd", "we'll", "we're",
'were', 'weren', "weren't", "we've", 'what', 'when', 'where', 'which', 'while',
'who', 'whom', 'why', 'will', 'with', 'won', "won't", 'wouldn', "wouldn't", 'y',
'you', "you'd", "you'll", 'your', "you're", 'yours', 'yourself', 'yourselves',
"you've"]
```

[18]:
```python
# Data processing
# loading the data from csv file to pandas dataframe
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.
 ↪csv', encoding='latin-1')
```

```
[19]: # Checking the number of rows & columns in the twitter_dataset
      twitter_data.shape

[19]: (1599999, 6)

[20]: # Print the first five rows of the dataframe
      twitter_data.head()

[20]:    0  …  @switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You
      shoulda got David Carr of Third Day to do it. ;D
      0  0  …  is upset that he can't update his Facebook by …
      1  0  …  @Kenichan I dived many times for the ball. Man…
      2  0  …    my whole body feels itchy and like its on fire
      3  0  …  @nationwideclass no, it's not behaving at all…
      4  0  …                        @Kwesidei not the whole crew

      [5 rows x 6 columns]

[22]: # Naming the columns and reading the dataset again
      column_names = ['target', 'ids', 'date', 'flag', 'user', 'text']
      twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.
       ↪csv',names=column_names, encoding='latin-1', header=None)
      twitter_data.head()

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

      [5 rows x 6 columns]

[23]: # Check if there are any missing values
      twitter_data.isnull().sum()

[23]: target    0
      ids       0
      date      0
      flag      0
      user      0
```

```
text      0
dtype: int64
```

[24]:
```
# Whats the distribution of this target variable
twitter_data['target'].value_counts()
```

[24]:
```
target
0    800000
4    800000
Name: count, dtype: int64
```

[26]:
```
# Convert the 4 to 1
twitter_data.replace({"target": {4: 1}}, inplace=True)
```

[27]:
```
# heck the distribution of the target variable
twitter_data['target'].value_counts()
```

[27]:
```
target
0    800000
1    800000
Name: count, dtype: int64
```

[28]:
```
# o = Negative tweet
# 1 = Positive tweet
```

[29]:
```
# Stemming - process of reducing key words to its root word. The dataset is␣
↪humngous. Hence we are using this method for easy process.
port_stem = PorterStemmer()
```

[30]:
```
def stemming(content):
  stemmed_content = re.sub('[^a-zA-Z]', ' ', content)
  stemmed_content = stemmed_content.lower()
  stemmed_content = stemmed_content.split()
  stemmed_content = [port_stem.stem(word) for word in stemmed_content if not␣
↪word in stopwords.words('english')]
  stemmed_content = ' '.join(stemmed_content)
  return stemmed_content
```

[31]:
```
twitter_data['stemmed_content'] = twitter_data['text'].apply(stemming)
```

[32]:
```
# Print the first five rows of this dataset
twitter_data.head()
```

[32]:
```
   target  …                                   stemmed_content
0       0  …  switchfoot http twitpic com zl awww bummer sho…
1       0  …  upset updat facebook text might cri result sch…
2       0  …  kenichan dive mani time ball manag save rest g…
```

```
3       0  …                        whole bodi feel itchi like fire
4       0  …                          nationwideclass behav mad see

[5 rows x 7 columns]
```

[33]:
```python
# Print twitter data and the stemmed content
print(twitter_data['stemmed_content'])
```

```
0           switchfoot http twitpic com zl awww bummer sho…
1           upset updat facebook text might cri result sch…
2           kenichan dive mani time ball manag save rest g…
3                                whole bodi feel itchi like fire
4                                  nationwideclass behav mad see
                                                  …
1599995                              woke school best feel ever
1599996    thewdb com cool hear old walt interview http b…
1599997                             readi mojo makeov ask detail
1599998    happi th birthday boo alll time tupac amaru sh…
1599999    happi charitytuesday thenspcc sparkschar speak…
Name: stemmed_content, Length: 1600000, dtype: object
```

[34]:
```python
# Print twitter data and the target
print(twitter_data['target'])
```

```
0          0
1          0
2          0
3          0
4          0
          ..
1599995    1
1599996    1
1599997    1
1599998    1
1599999    1
Name: target, Length: 1600000, dtype: int64
```

[35]:
```python
# Separating the data and label
X = twitter_data['stemmed_content'].values
Y = twitter_data['target'].values
```

[36]:
```python
print(X)
```

```
['switchfoot http twitpic com zl awww bummer shoulda got david carr third day'
 'upset updat facebook text might cri result school today also blah'
 'kenichan dive mani time ball manag save rest go bound' …
 'readi mojo makeov ask detail'
 'happi th birthday boo alll time tupac amaru shakur'
 'happi charitytuesday thenspcc sparkschar speakinguph h']
```

[37]:
```python
print(Y)
```

```
[0 0 0 … 1 1 1]
```

[38]:
```python
# Spliting the data into training & testing data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.2,␣
 ↪stratify =Y, random_state=2)
```

[39]:
```python
print(X.shape,X_train.shape, X_test.shape)
```

```
(1600000,) (1280000,) (320000,)
```

[40]:
```python
print(X_train)
```

```
['watch saw iv drink lil wine' 'hatermagazin'
 'even though favourit drink think vodka coke wipe mind time think im gonna find
new drink'
 … 'eager monday afternoon'
 'hope everyon mother great day wait hear guy store tomorrow'
 'love wake folger bad voic deeper']
```

[41]:
```python
# Convert textual values to numerical values. Use feature extraction, a method␣
 ↪called vectorizer is used to achieve this.
vectorizer = TfidfVectorizer()
vectorizer.fit(X_train)

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

[42]:
```python
print(X_train) # Vectorizer has successfully converted the text to numerical␣
 ↪values
```

```
  (0, 436713)   0.27259876264838384
  (0, 354543)   0.3588091611460021
  (0, 185193)   0.5277679060576009
  (0, 109306)   0.3753708587402299
  (0, 235045)   0.41996827700291095
  (0, 443066)   0.4484755317023172
  (1, 160636)   1.0
  (2, 109306)   0.4591176413728317
  (2, 124484)   0.1892155960801415
  (2, 407301)   0.18709338684973031
  (2, 129411)   0.29074192727957143
  (2, 406399)   0.32105459490875526
  (2, 433560)   0.3296595898028565
```

```
(2, 77929)	0.31284080750346344
(2, 443430)	0.3348599670252845
(2, 266729)	0.24123230668976975
(2, 409143)	0.15169282335109835
(2, 178061)	0.1619010109445149
(2, 150715)	0.18803850583207948
(2, 132311)	0.2028971570399794
(2, 288470)	0.16786949597862733
(3, 406399)	0.29029991238662284
(3, 158711)	0.4456939372299574
(3, 151770)	0.278559647704793
(3, 56476)	0.5200465453608686
:	:
(1279996, 318303)	0.21254698865277744
(1279996, 434014)	0.27189450523324465
(1279996, 390130)	0.2206474219107611
(1279996, 373144)	0.35212500999832036
(1279996, 238077)	0.5249170684084672
(1279996, 238078)	0.5606696159563151
(1279997, 5685)	0.48650358607431304
(1279997, 273084)	0.4353549002982409
(1279997, 112591)	0.7574829183045267
(1279998, 412553)	0.2816582375021589
(1279998, 93795)	0.21717768937055476
(1279998, 169461)	0.2659980990397061
(1279998, 124765)	0.32241752985927996
(1279998, 435463)	0.2851807874350361
(1279998, 153281)	0.28378968751027456
(1279998, 156297)	0.3137096161546449
(1279998, 162047)	0.34691726958159064
(1279998, 275288)	0.38703346602729577
(1279998, 385313)	0.4103285865588191
(1279999, 242268)	0.19572649660865402
(1279999, 31410)	0.248792678366695
(1279999, 435572)	0.31691096877786484
(1279999, 433612)	0.3607341026233411
(1279999, 135384)	0.6130934129868719
(1279999, 96224)	0.5416162421321443
```

[43]:
```
print(X_test) # Vectorizer has successfully converted text to numerical values
```

```
(0, 15110)	0.1719352837797837
(0, 31168)	0.1624772418052177
(0, 67828)	0.26800375270827315
(0, 106069)	0.36555450010904555
(0, 132364)	0.255254889555786
(0, 138164)	0.23688292264071406
(0, 171378)	0.2805816206356074
```

```
  (0, 271016)	0.45356623916588285
  (0, 279082)	0.17825180109103442
  (0, 388348)	0.2198507607206174
  (0, 398906)	0.34910438732642673
  (0, 409143)	0.3143047059807971
  (0, 420984)	0.17915624523539805
  (1, 6463)	0.30733520460524466
  (1, 15110)	0.211037449588008
  (1, 145393)	0.575262969264869
  (1, 217562)	0.40288153995289894
  (1, 256777)	0.28751585696559306
  (1, 348135)	0.4739279595416274
  (1, 366203)	0.24595562404108307
  (2, 22532)	0.3532582957477176
  (2, 34401)	0.37916255084357414
  (2, 89448)	0.36340369428387626
  (2, 183312)	0.5892069252021465
  (2, 256834)	0.2564939661498776
  :	:
  (319994, 443794)	0.2782185641032538
  (319995, 107868)	0.33399349737546963
  (319995, 109379)	0.3020896484890833
  (319995, 155493)	0.2770682832971669
  (319995, 213324)	0.2683969144317079
  (319995, 232891)	0.2574127854589077
  (319995, 296662)	0.3992485679384015
  (319995, 315813)	0.2848229914563413
  (319995, 324496)	0.36131679336475747
  (319995, 416257)	0.23816465111736282
  (319995, 420984)	0.22631428606830148
  (319995, 444934)	0.32110928175992615
  (319996, 397506)	0.9101400928717545
  (319996, 438709)	0.4143006291901984
  (319997, 98792)	0.4463892055808332
  (319997, 169411)	0.403381646999604
  (319997, 261286)	0.37323893626855326
  (319997, 288421)	0.48498483387153407
  (319997, 349904)	0.32484594100566083
  (319997, 416695)	0.29458327588067873
  (319997, 444770)	0.2668297951055569
  (319998, 130192)	0.6941927210956169
  (319998, 438748)	0.719789181620468
  (319999, 389755)	0.9577980203954275
  (319999, 400636)	0.2874420848216212
```

[44]:
```
# Training the model using the Logistic Regression
model = LogisticRegression(max_iter=1000)
```

```
[45]: model.fit(X_train, Y_train) # This were the model will learn from the data
```

```
[45]: LogisticRegression(max_iter=1000)
```

```
[47]: # Model Evaluation
      # The only metrics used in this project is accuracy_score
      X_train_prediction = model.predict(X_train)
      training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
[48]: print('Accuracy score  on training data:', training_data_accuracy)
```

```
Accuracy score  on training data: 0.79871953125
```

```
[51]: # Check the accuracy score on test data
      X_test_prediction = model.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
[54]: print('Accuracy score on training data:', test_data_accuracy)
```

```
Accuracy score on training data: 0.77668125
```

```
[ ]: # The model performed well.
     # Accuracy score for testing data was 79.9%
     # Accuracy score for training data was 79.8%
```

```
[55]: import pickle
```

```
[56]: filename = "training_model.sav"
      pickle.dump(model, open(filename, 'wb'))
```

```
[60]: # How to use the saved model for future predictions
      loaded_model = pickle.load(open('/content/training_model.sav', 'rb'))
```

```
[66]: X_new = X_test[200]
      print(Y_test[200])

      prediction = model.predict(X_new)
      print(prediction)

      if prediction[0] == 0:
        print('The tweet is negative')
      else:
        print('The tweet is positive')
```

```
1
[1]
The tweet is positive
```

```
[ ]:
```