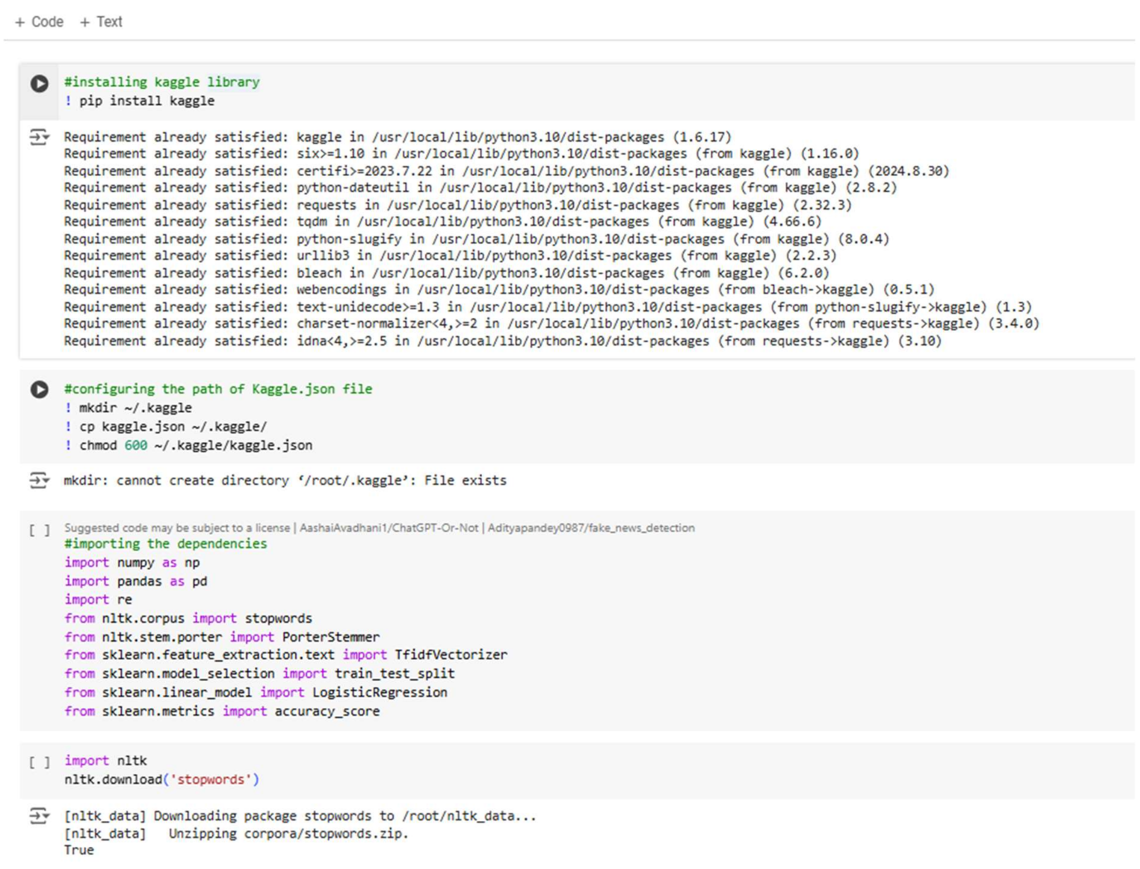


```
#installing kaggle library
! pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.8.30)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.6)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.2.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.10)
```

```
#configuring the path of Kaggle.json file
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

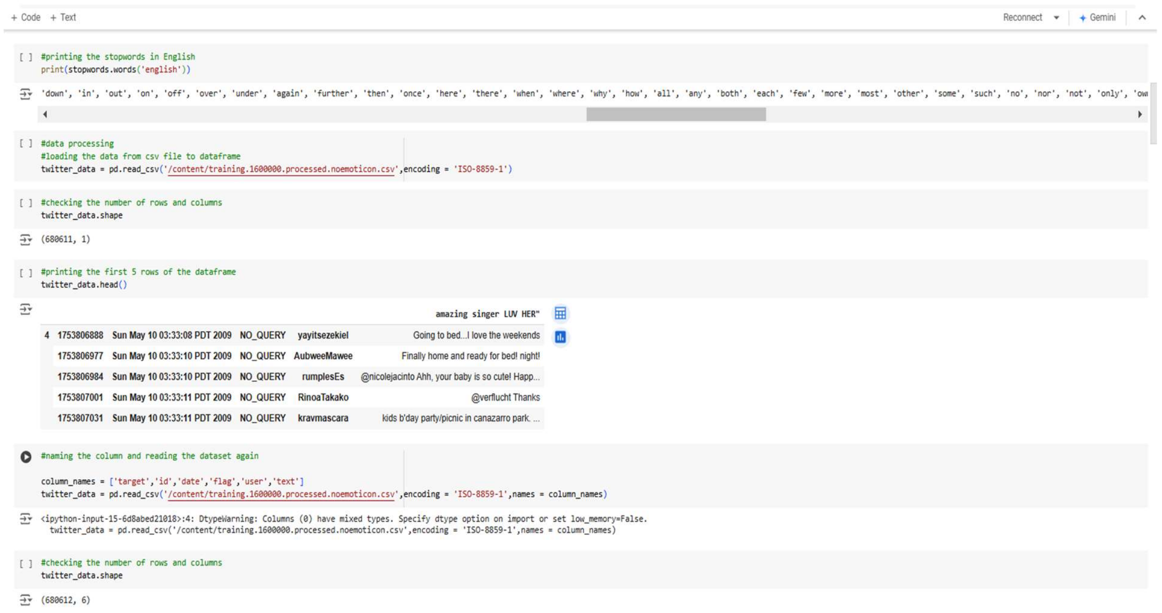
```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

[ ] Suggested code may be subject to a license | AashaiAvadhani1/ChatGPT-Or-Not | Adityapandey0987/fake_news_detection

```
#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
```

```
import nltk
nltk.download('stopwords')
```

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



```
#printing the stopwords in English
print(stopwords.words('english'))
```

```
'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'ow
```

```
#data processing
#loading the data from csv file to dataframe
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv',encoding = 'ISO-8859-1')
```

```
#checking the number of rows and columns
twitter_data.shape
```

```
(680611, 1)
```

```
#printing the first 5 rows of the dataframe
twitter_data.head()
```

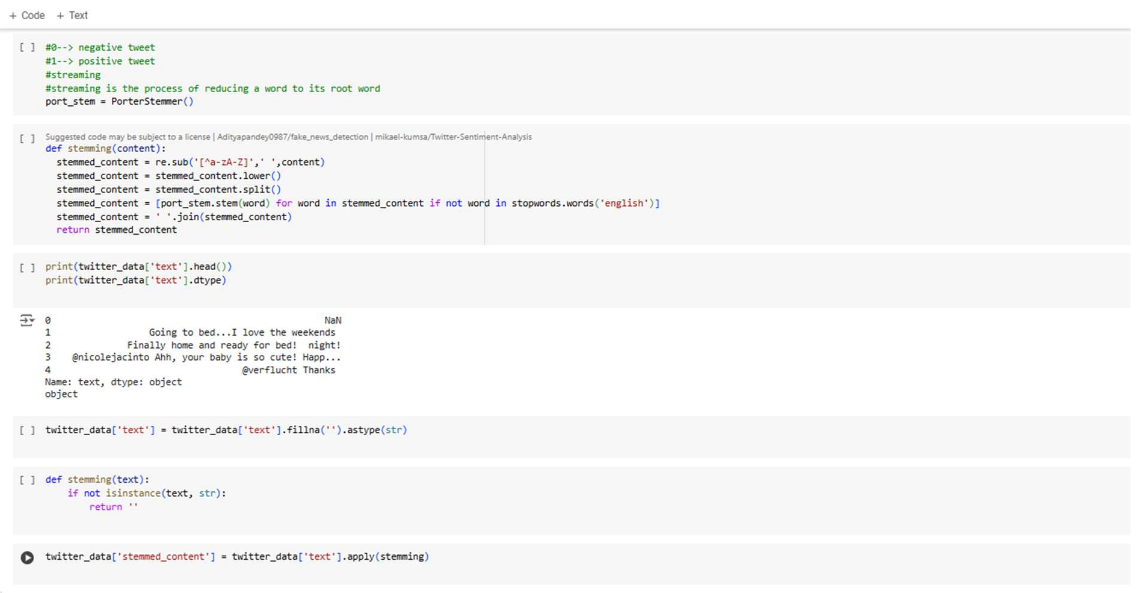
					amazing singer LUV HER"
4	1753806888	Sun May 10 03:33:08 PDT 2009	NO_QUERY	yayitsezekiel	Going to bed...I love the weekends
	1753806977	Sun May 10 03:33:10 PDT 2009	NO_QUERY	AubweeMawee	Finally home and ready for bed! night!
	1753806984	Sun May 10 03:33:10 PDT 2009	NO_QUERY	rumplesEs	@nicolejacinto Ahh, your baby is so cute! Happ...
	1753807001	Sun May 10 03:33:11 PDT 2009	NO_QUERY	RinoaTakako	@verflucht Thanks
	1753807031	Sun May 10 03:33:11 PDT 2009	NO_QUERY	kravmascara	kids b'day party/picnic in canazarro park. ...

```
#naming the column and reading the dataset again

column_names = ['target','id','date','flag','user','text']
twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv',encoding = 'ISO-8859-1',names = column_names)
```

```
<ipython-input-15-6d8abed21018>:4: DtypeWarning: Columns (0) have mixed types. Specify dtype option on import or set low_memory=False.
  twitter_data = pd.read_csv('/content/training.1600000.processed.noemoticon.csv',encoding = 'ISO-8859-1',names = column_names)
```

```
#checking the number of rows and columns
twitter_data.shape
```

```
(680612, 6)
```



+ Code + Text

```
#0--> negative tweet
#1--> positive tweet
#streaming
#streaming is the process of reducing a word to its root word
port_stem = PorterStemmer()
```

Suggested code may be subject to a license | Adityapandey0987/fake_news_detection | mikael-kumsa/Twitter-Sentiment-Analysis

```
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
```

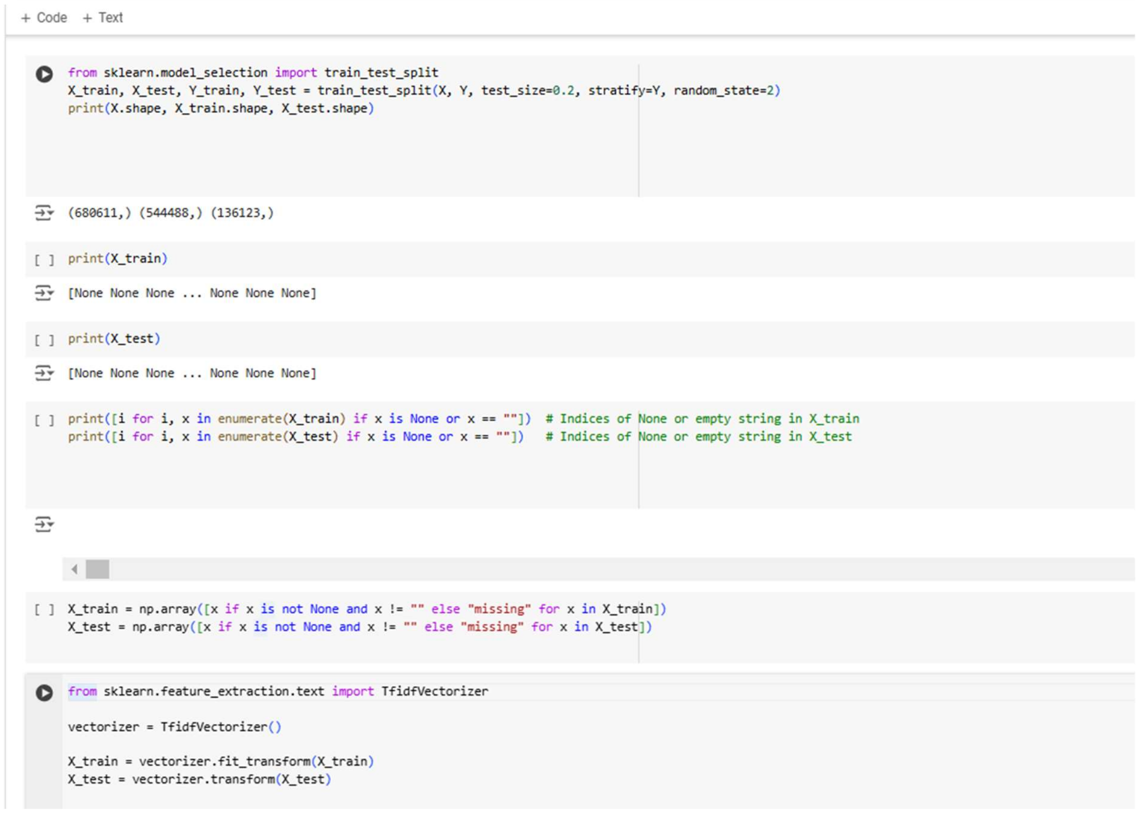
```
print(twitter_data['text'].head())
print(twitter_data['text'].dtype)
```

```
0                                              NaN
1                     Going to bed...I love the weekends
2                 Finally home and ready for bed!  night!
3    @nicolejacinto Ahh, your baby is so cute! Happ...
4                                   @verflucht Thanks
Name: text, dtype: object
object
```

```
twitter_data['text'] = twitter_data['text'].fillna('').astype(str)
```

```
def stemming(text):
    if not isinstance(text, str):
        return ''
```

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



+ Code + Text

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

```
(680611,) (544488,) (136123,)
```

```
print(X_train)
```

```
[None None None ... None None None]
```

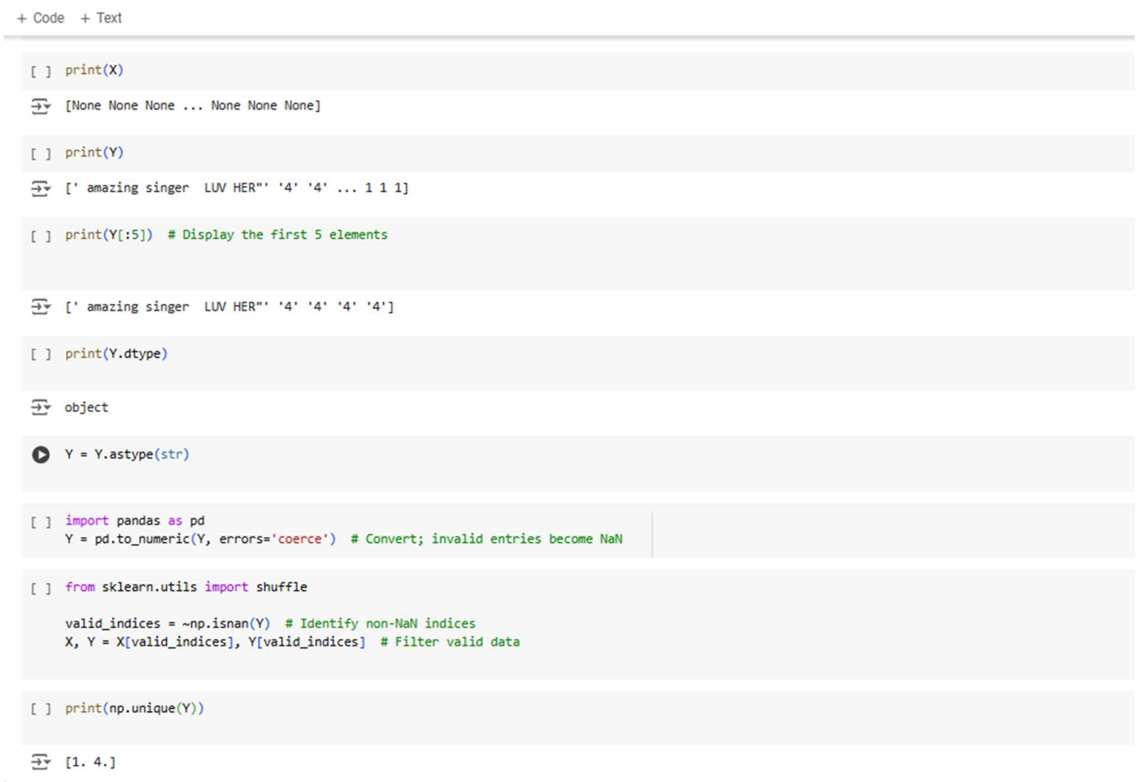
```
print(X_test)
```

```
[None None None ... None None None]
```

```
print([i for i, x in enumerate(X_train) if x is None or x == ""])  # Indices of None or empty string in X_train
print([i for i, x in enumerate(X_test) if x is None or x == ""])   # Indices of None or empty string in X_test
```

```
X_train = np.array([x if x is not None and x != "" else "missing" for x in X_train])
X_test = np.array([x if x is not None and x != "" else "missing" for x in X_test])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

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



```
print(X)
```

```
[None None None ... None None None]
```

```
print(Y)
```

```
[' amazing singer  LUV HER"' '4' '4' ... 1 1 1]
```

```
print(Y[:5])  # Display the first 5 elements
```

```
[' amazing singer  LUV HER"' '4' '4' '4' '4']
```

```
print(Y.dtype)
```

```
object
```

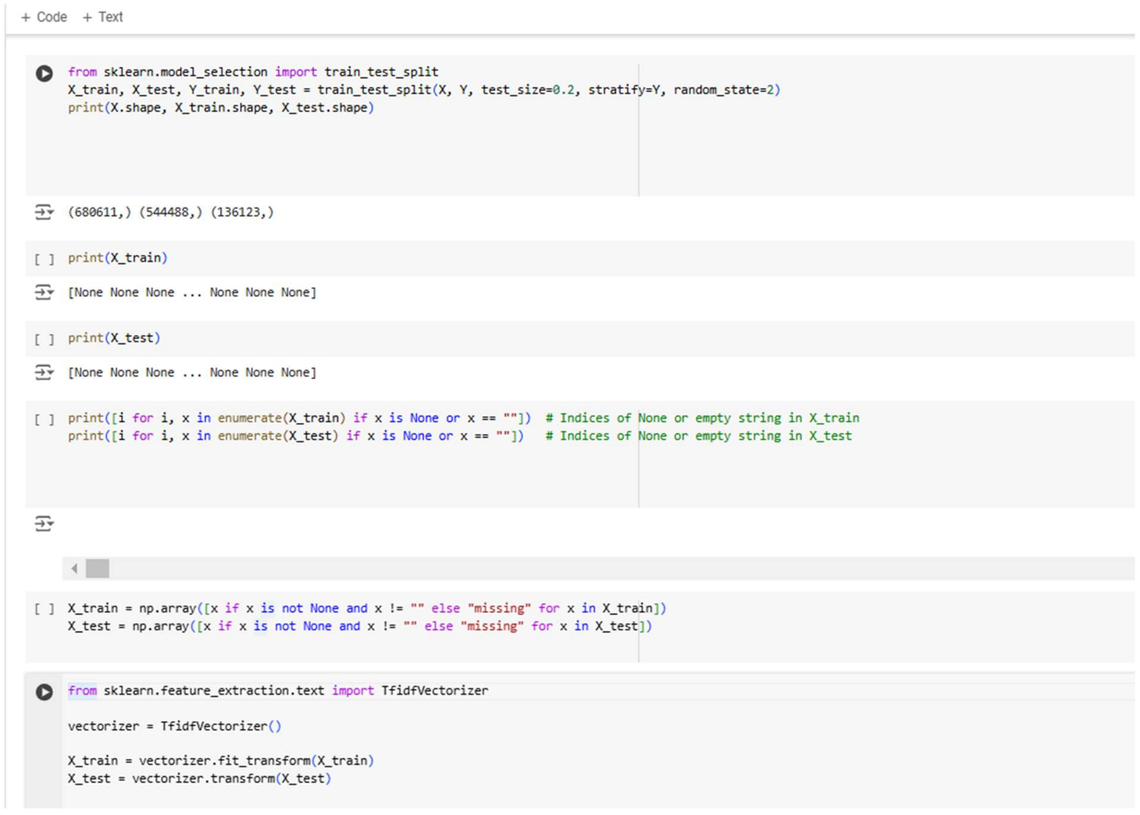
```
Y = Y.astype(str)
```

```
import pandas as pd
Y = pd.to_numeric(Y, errors='coerce')  # Convert; invalid entries become NaN
```

```
from sklearn.utils import shuffle

valid_indices = ~np.isnan(Y)  # Identify non-NaN indices
X, Y = X[valid_indices], Y[valid_indices]  # Filter valid data
```

```
print(np.unique(Y))
```

```
[1. 4.]
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

```
(680611,) (544488,) (136123,)
```

```
print(X_train)
```

```
[None None None ... None None None]
```

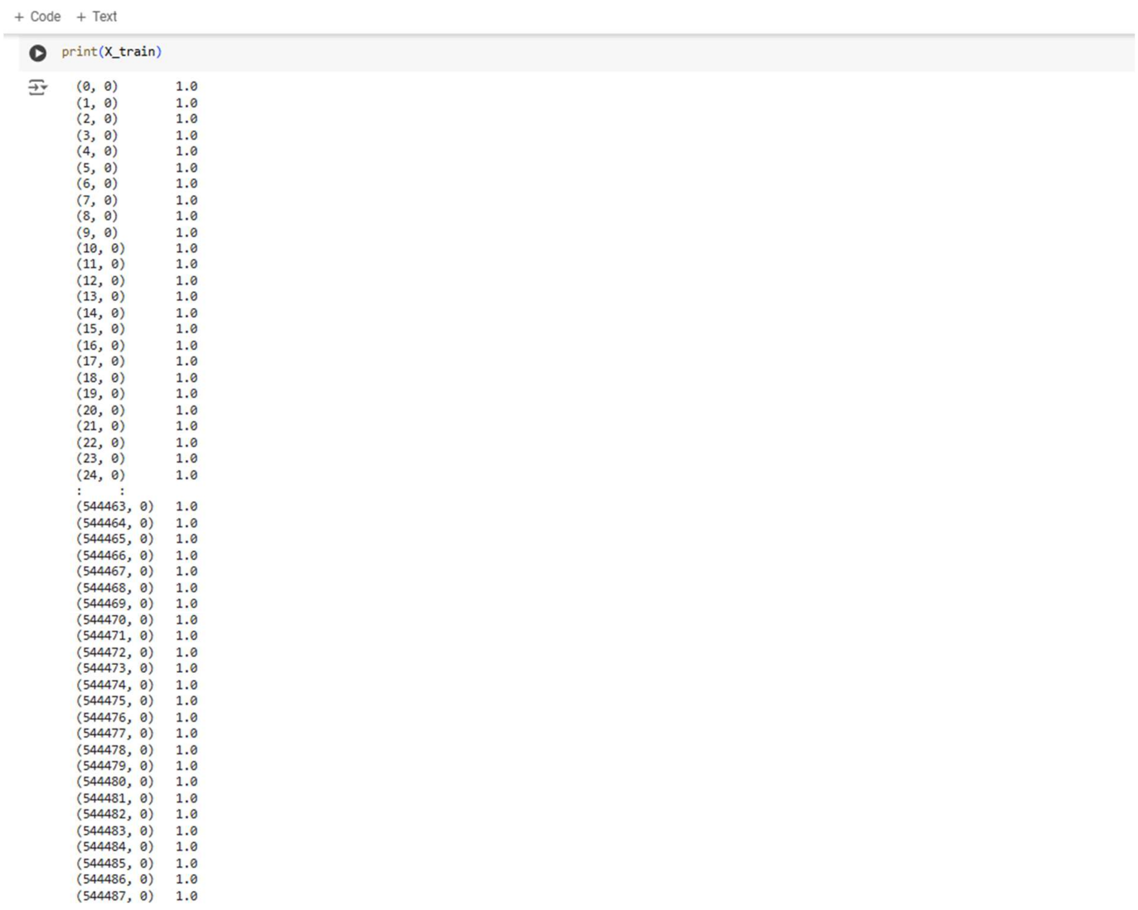
```
print(X_test)
```

```
[None None None ... None None None]
```

```
print([i for i, x in enumerate(X_train) if x is None or x == ""])  # Indices of None or empty string in X_train
print([i for i, x in enumerate(X_test) if x is None or x == ""])   # Indices of None or empty string in X_test
```

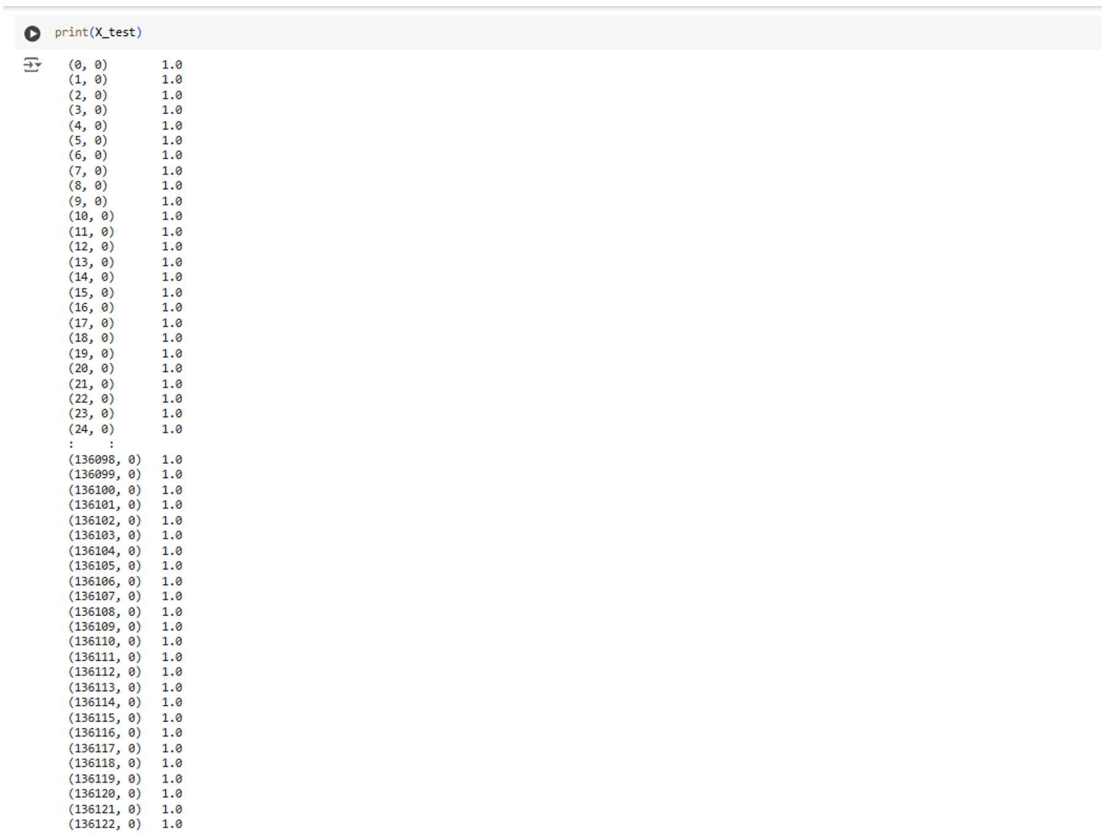
```
X_train = np.array([x if x is not None and x != "" else "missing" for x in X_train])
X_test = np.array([x if x is not None and x != "" else "missing" for x in X_test])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

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



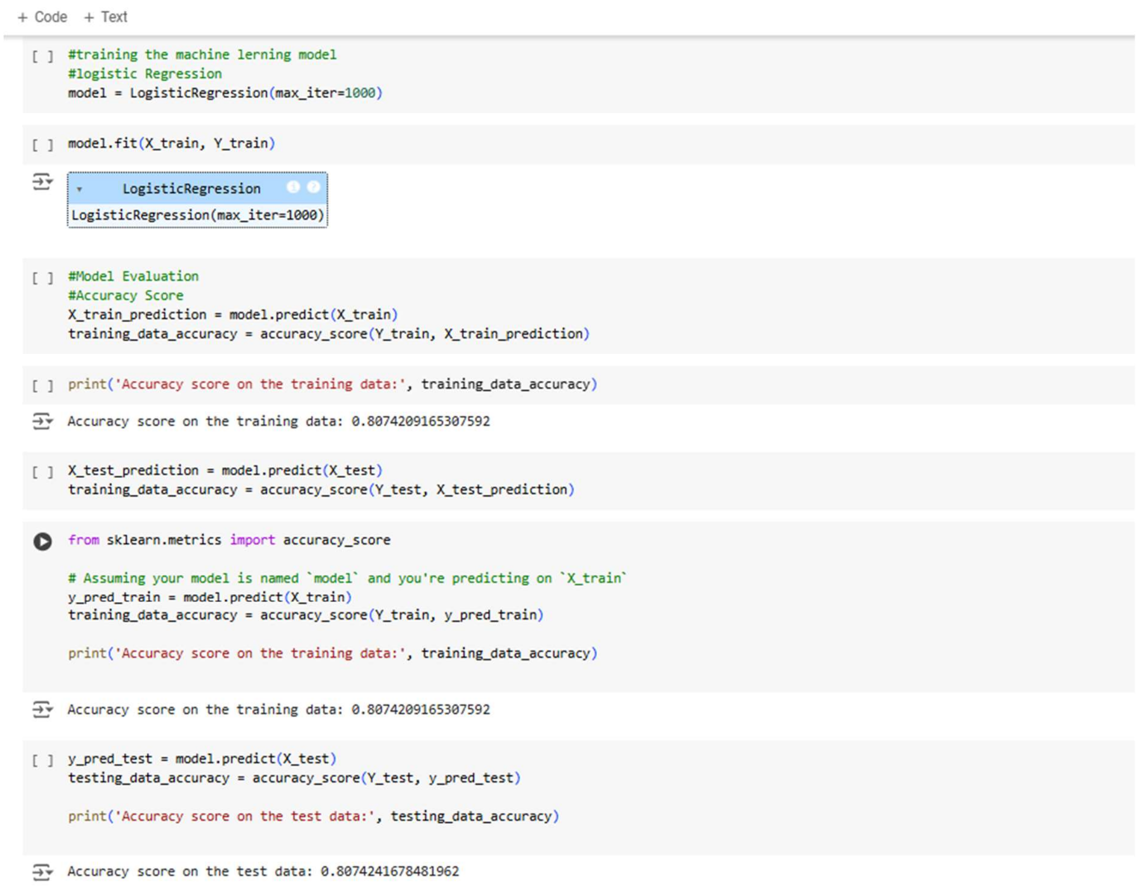
```
print(X_train)
```

```
  (0, 0)	1.0
  (1, 0)	1.0
  (2, 0)	1.0
  (3, 0)	1.0
  (4, 0)	1.0
  (5, 0)	1.0
  (6, 0)	1.0
  (7, 0)	1.0
  (8, 0)	1.0
  (9, 0)	1.0
  (10, 0)	1.0
  (11, 0)	1.0
  (12, 0)	1.0
  (13, 0)	1.0
  (14, 0)	1.0
  (15, 0)	1.0
  (16, 0)	1.0
  (17, 0)	1.0
  (18, 0)	1.0
  (19, 0)	1.0
  (20, 0)	1.0
  (21, 0)	1.0
  (22, 0)	1.0
  (23, 0)	1.0
  (24, 0)	1.0
  :	:
  (544463, 0)	1.0
  (544464, 0)	1.0
  (544465, 0)	1.0
  (544466, 0)	1.0
  (544467, 0)	1.0
  (544468, 0)	1.0
  (544469, 0)	1.0
  (544470, 0)	1.0
  (544471, 0)	1.0
  (544472, 0)	1.0
  (544473, 0)	1.0
  (544474, 0)	1.0
  (544475, 0)	1.0
  (544476, 0)	1.0
  (544477, 0)	1.0
  (544478, 0)	1.0
  (544479, 0)	1.0
  (544480, 0)	1.0
  (544481, 0)	1.0
  (544482, 0)	1.0
  (544483, 0)	1.0
  (544484, 0)	1.0
  (544485, 0)	1.0
  (544486, 0)	1.0
  (544487, 0)	1.0
```



```
print(X_test)
```

```
  (0, 0)	1.0
  (1, 0)	1.0
  (2, 0)	1.0
  (3, 0)	1.0
  (4, 0)	1.0
  (5, 0)	1.0
  (6, 0)	1.0
  (7, 0)	1.0
  (8, 0)	1.0
  (9, 0)	1.0
  (10, 0)	1.0
  (11, 0)	1.0
  (12, 0)	1.0
  (13, 0)	1.0
  (14, 0)	1.0
  (15, 0)	1.0
  (16, 0)	1.0
  (17, 0)	1.0
  (18, 0)	1.0
  (19, 0)	1.0
  (20, 0)	1.0
  (21, 0)	1.0
  (22, 0)	1.0
  (23, 0)	1.0
  (24, 0)	1.0
  :	:
  (136098, 0)	1.0
  (136099, 0)	1.0
  (136100, 0)	1.0
  (136101, 0)	1.0
  (136102, 0)	1.0
  (136103, 0)	1.0
  (136104, 0)	1.0
  (136105, 0)	1.0
  (136106, 0)	1.0
  (136107, 0)	1.0
  (136108, 0)	1.0
  (136109, 0)	1.0
  (136110, 0)	1.0
  (136111, 0)	1.0
  (136112, 0)	1.0
  (136113, 0)	1.0
  (136114, 0)	1.0
  (136115, 0)	1.0
  (136116, 0)	1.0
  (136117, 0)	1.0
  (136118, 0)	1.0
  (136119, 0)	1.0
  (136120, 0)	1.0
  (136121, 0)	1.0
  (136122, 0)	1.0
```



```
#training the machine lerning model
#logistic Regression
model = LogisticRegression(max_iter=1000)
```

```
model.fit(X_train, Y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
#Model Evaluation
#Accuracy Score
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
print('Accuracy score on the training data:', training_data_accuracy)
```

```
Accuracy score on the training data: 0.8074209165307592
```

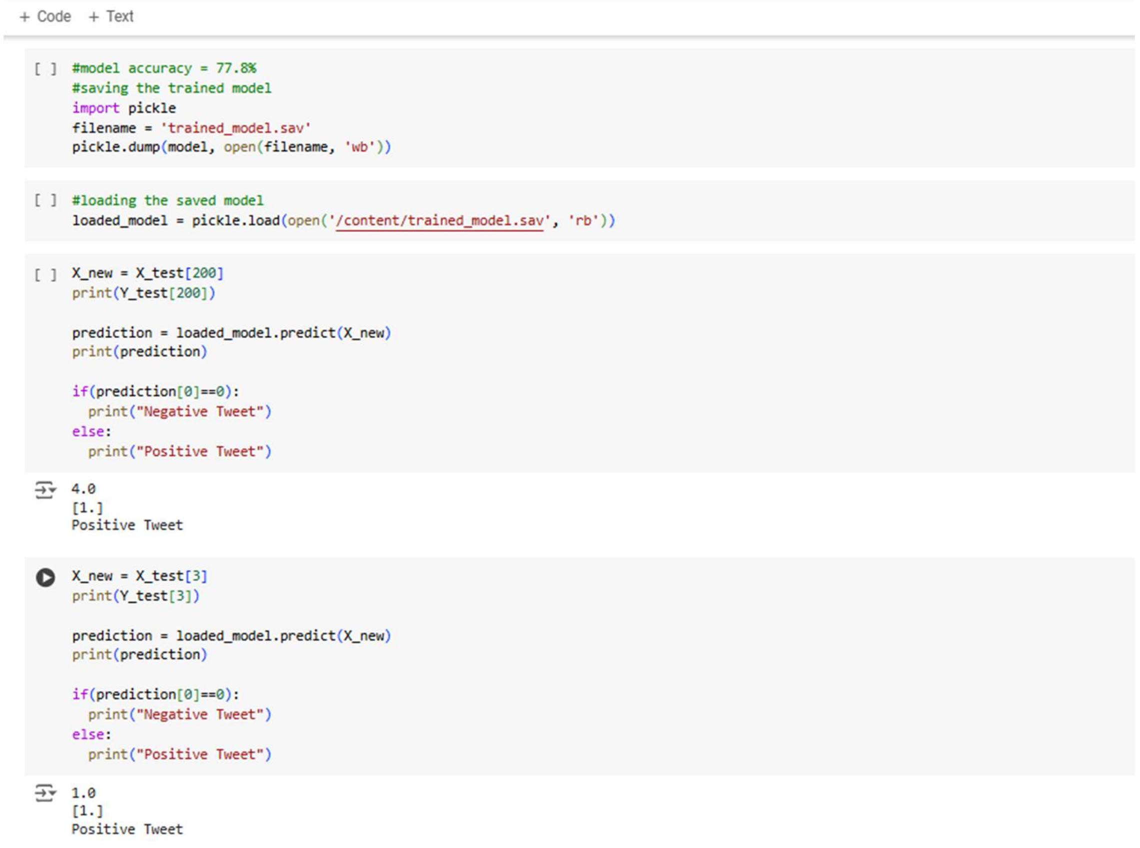
```
X_test_prediction = model.predict(X_test)
training_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
from sklearn.metrics import accuracy_score

# Assuming your model is named `model` and you're predicting on `X_train`
y_pred_train = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, y_pred_train)

print('Accuracy score on the training data:', training_data_accuracy)
```

```
Accuracy score on the training data: 0.8074209165307592
```

```
y_pred_test = model.predict(X_test)
testing_data_accuracy = accuracy_score(Y_test, y_pred_test)

print('Accuracy score on the test data:', testing_data_accuracy)
```

```
Accuracy score on the test data: 0.8074241678481962
```



```
#model accuracy = 77.8%
#saving the trained model
import pickle
filename = 'trained_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
#loading the saved model
loaded_model = pickle.load(open('/content/trained_model.sav', 'rb'))
```

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

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

if(prediction[0]==0):
  print("Negative Tweet")
else:
  print("Positive Tweet")
```

```
4.0
[1.]
Positive Tweet
```

```
X_new = X_test[3]
print(Y_test[3])

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

if(prediction[0]==0):
  print("Negative Tweet")
else:
  print("Positive Tweet")
```

```
1.0
[1.]
Positive Tweet
```