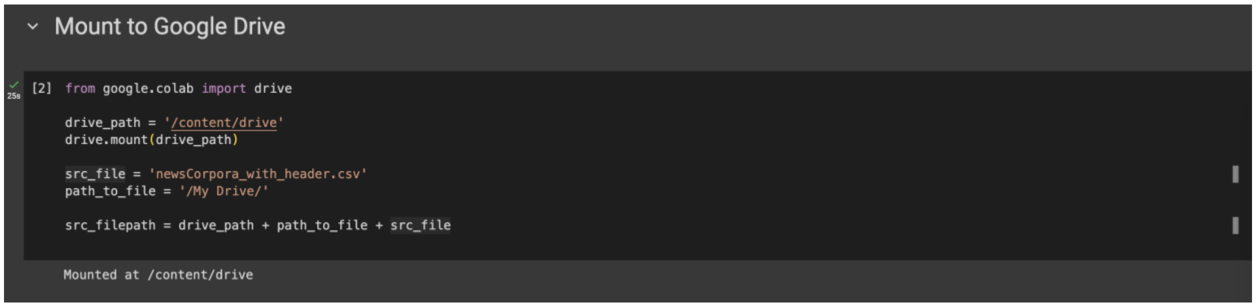
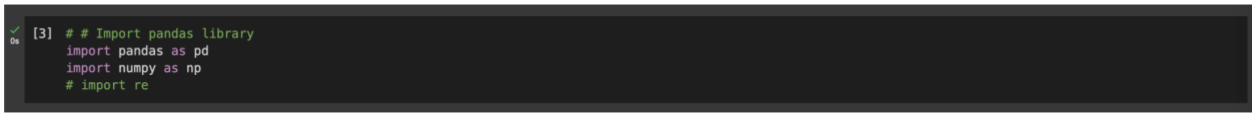
Royal University of Phnom Penh
Faculty of Engineering

Class	: A
Course	: IDM
Generation	: 7th
Year	: 4
Group	: 13 (Pho Phopversna, Nou Soveasna, Nath Sovanroth, Nhil Ratha)
Lecturer	: Chap Chanpiseth

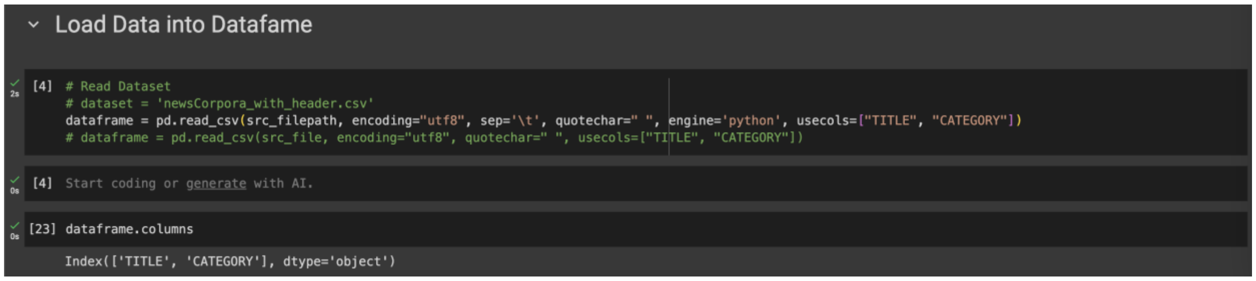


Mount to Google Drive

```
[2] from google.colab import drive

    drive_path = '/content/drive'
    drive.mount(drive_path)

    src_file = 'newsCorpora_with_header.csv'
    path_to_file = '/My Drive/'

    src_filepath = drive_path + path_to_file + src_file

    Mounted at /content/drive
```



```
[3] # # Import pandas library
    import pandas as pd
    import numpy as np
    # import re
```

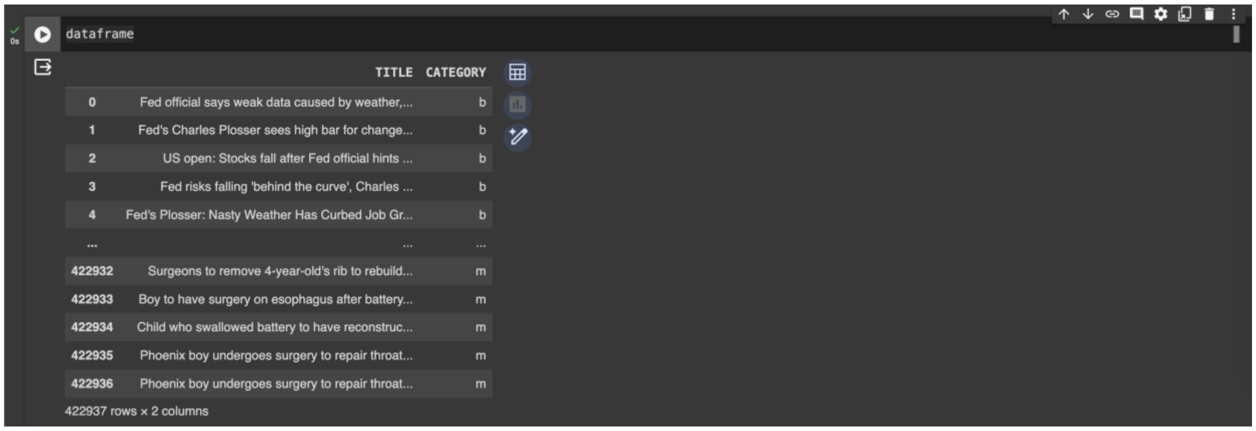


Load Data into Datafame

```
[4] # Read Dataset
    # dataset = 'newsCorpora_with_header.csv'
    dataframe = pd.read_csv(src_filepath, encoding="utf8", sep='\t', quotechar=" ", engine='python', usecols=["TITLE", "CATEGORY"])
    # dataframe = pd.read_csv(src_file, encoding="utf8", quotechar=" ", usecols=["TITLE", "CATEGORY"])

[4] Start coding or generate with AI.

[23] dataframe.columns

    Index(['TITLE', 'CATEGORY'], dtype='object')
```

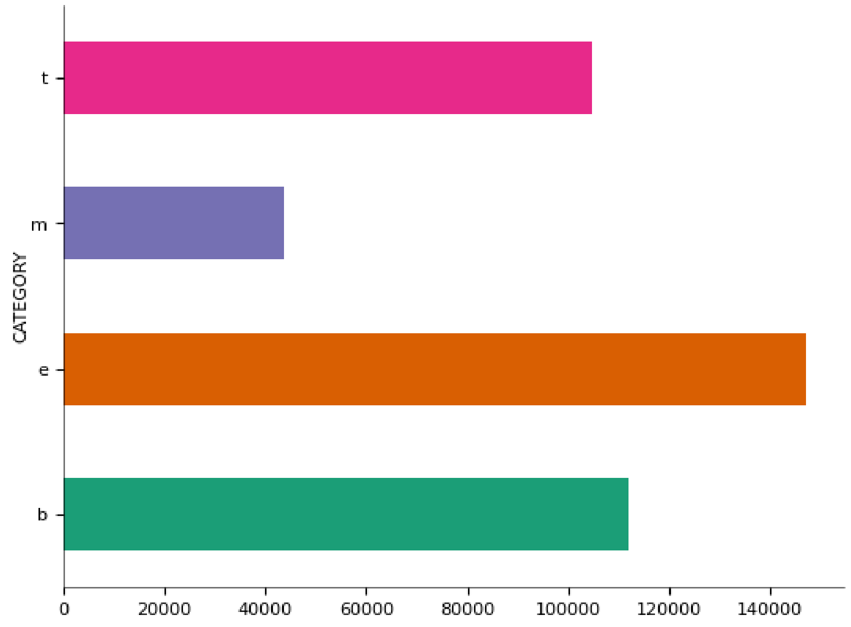
Load Data into Datafame

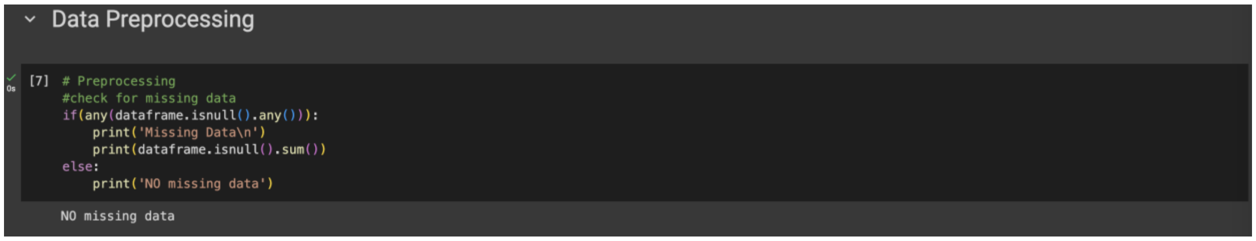
```
[4] # Read Dataset
    # dataset = 'newsCorpora_with_header.csv'
    dataframe = pd.read_csv(src_filepath, encoding="utf8", sep='\t', quotechar=" ", engine='python', usecols=["TITLE", "CATEGORY"])
    # dataframe = pd.read_csv(src_file, encoding="utf8", quotechar=" ", usecols=["TITLE", "CATEGORY"])

[4] Start coding or generate with AI.

[23] dataframe.columns

    Index(['TITLE', 'CATEGORY'], dtype='object')
```



```
dataframe
```

```
                                                TITLE  CATEGORY
0          Fed official says weak data caused by weather,...   b
1          Fed's Charles Plosser sees high bar for change...   b
2               US open: Stocks fall after Fed official hints ...   b
3              Fed risks falling 'behind the curve', Charles ...   b
4      Fed's Plosser: Nasty Weather Has Curbed Job Gr...   b
...                                                        ...   ...
422932     Surgeons to remove 4-year-old's rib to rebuild...   m
422933   Boy to have surgery on esophagus after battery...   m
422934  Child who swallowed battery to have reconstruc...   m
422935     Phoenix boy undergoes surgery to repair throat...   m
422936     Phoenix boy undergoes surgery to repair throat...   m
422937 rows × 2 columns
```

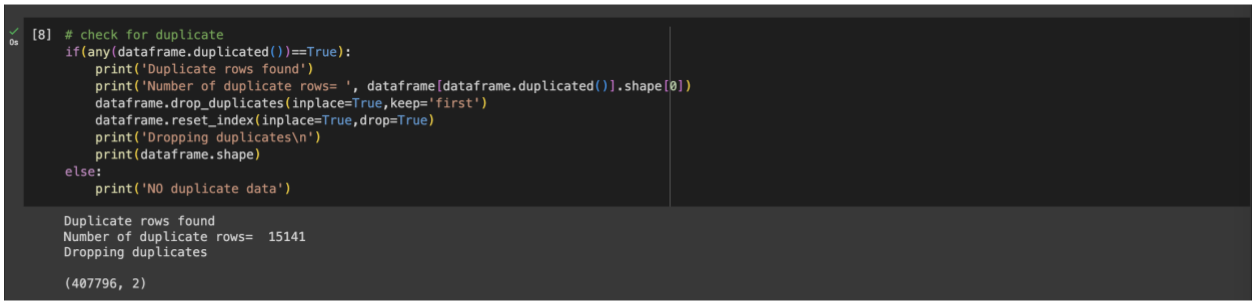




Data Preprocessing

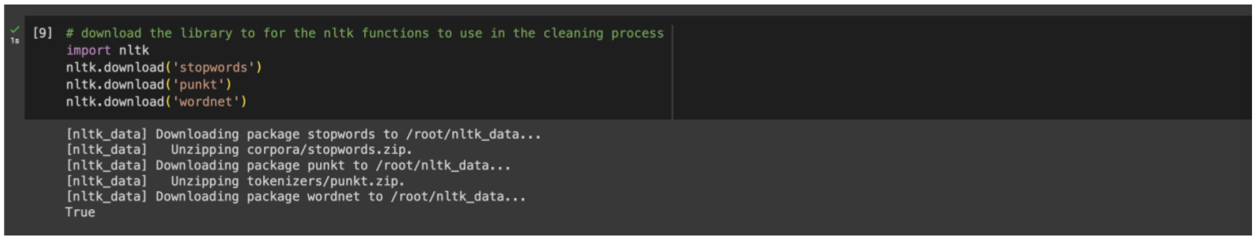
```
[7] # Preprocessing
    #check for missing data
    if(any(dataframe.isnull().any())):
        print('Missing Data\n')
        print(dataframe.isnull().sum())
    else:
        print('NO missing data')
```

```
NO missing data
```

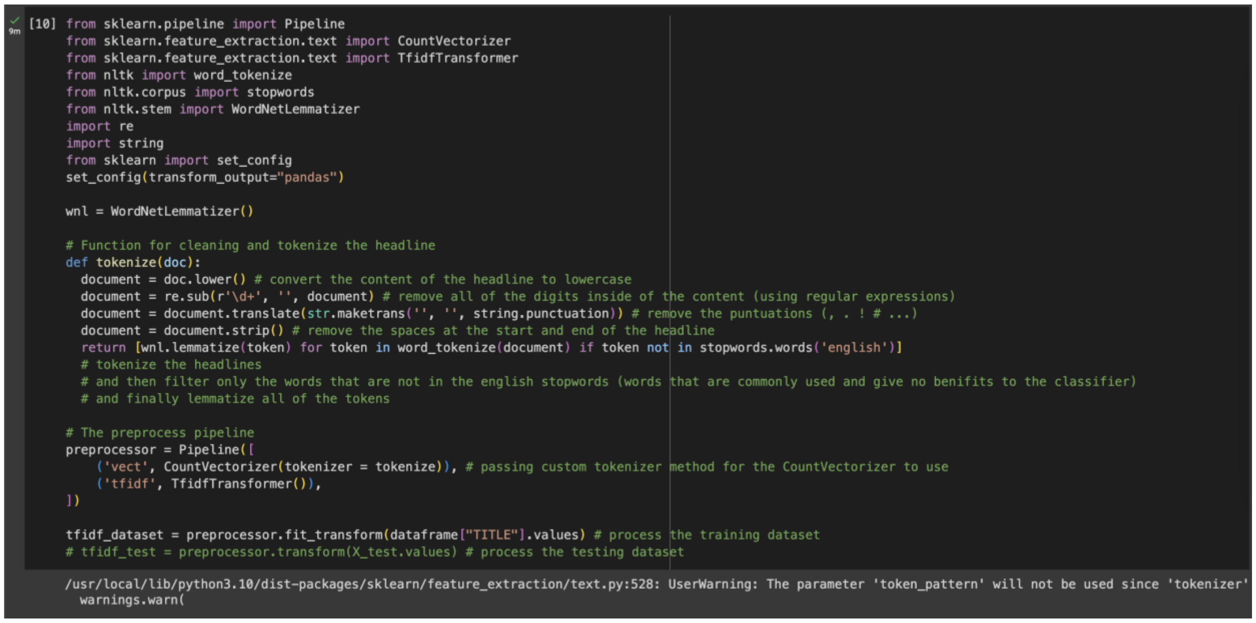


```
[8] # check for duplicate
    if(any(dataframe.duplicated())==True):
        print('Duplicate rows found')
        print('Number of duplicate rows= ', dataframe[dataframe.duplicated()].shape[0])
        dataframe.drop_duplicates(inplace=True,keep='first')
        dataframe.reset_index(inplace=True,drop=True)
        print('Dropping duplicates\n')
        print(dataframe.shape)
    else:
        print('NO duplicate data')
```

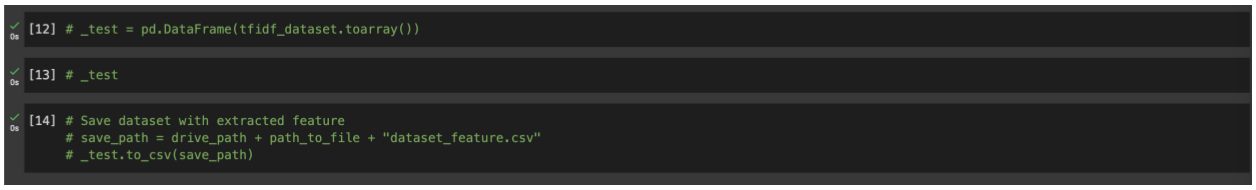
```
Duplicate rows found
Number of duplicate rows=  15141
Dropping duplicates

(407796, 2)
```



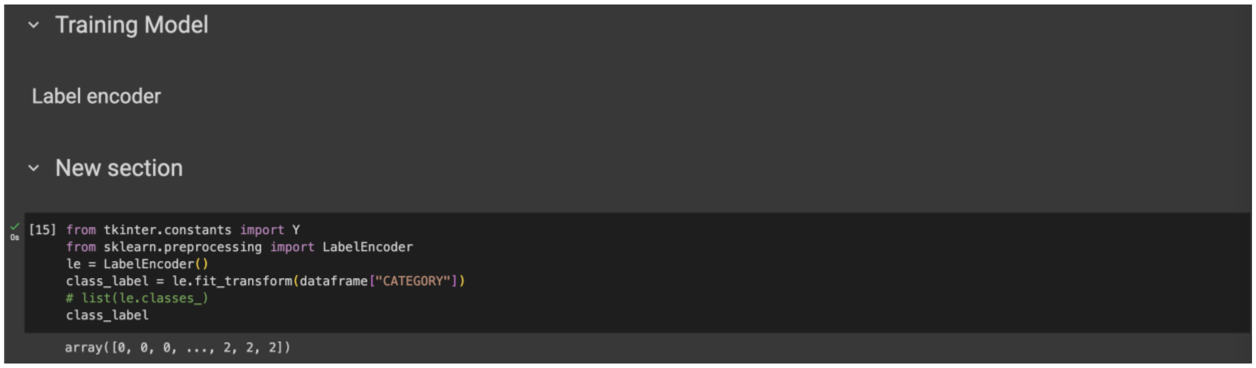
```
[9] # download the library to for the nltk functions to use in the cleaning process
    import nltk
    nltk.download('stopwords')
    nltk.download('punkt')
    nltk.download('wordnet')

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



```
[10] from sklearn.pipeline import Pipeline
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.feature_extraction.text import TfidfTransformer
     from nltk import word_tokenize
     from nltk.corpus import stopwords
     from nltk.stem import WordNetLemmatizer
     import re
     import string
     from sklearn import set_config
     set_config(transform_output="pandas")

     wnl = WordNetLemmatizer()

     # Function for cleaning and tokenize the headline
     def tokenize(doc):
       document = doc.lower() # convert the content of the headline to lowercase
       document = re.sub(r'\d+', '', document) # remove all of the digits inside of the content (using regular expressions)
       document = document.translate(str.maketrans('', '', string.punctuation)) # remove the puntuations (, . ! # ...)
       document = document.strip() # remove the spaces at the start and end of the headline
       return [wnl.lemmatize(token) for token in word_tokenize(document) if token not in stopwords.words('english')]
       # tokenize the headlines
       # and then filter only the words that are not in the english stopwords (words that are commonly used and give no benifits to the classifier)
       # and finally lemmatize all of the tokens

     # The preprocess pipeline
     preprocessor = Pipeline([
         ('vect', CountVectorizer(tokenizer = tokenize)), # passing custom tokenizer method for the CountVectorizer to use
         ('tfidf', TfidfTransformer()),
     ])

     tfidf_dataset = preprocessor.fit_transform(dataframe["TITLE"].values) # process the training dataset
     # tfidf_test = preprocessor.transform(X_test.values) # process the testing dataset

     /usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer'
       warnings.warn(
```



```
[12] # _test = pd.DataFrame(tfidf_dataset.toarray())

[13] # _test

[14] # Save dataset with extracted feature
     # save_path = drive_path + path_to_file + "dataset_feature.csv"
     # _test.to_csv(save_path)
```

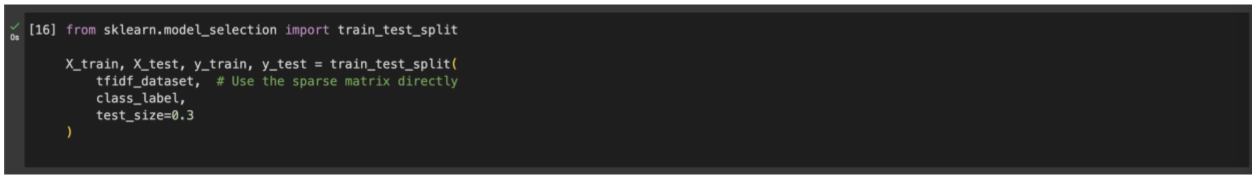


Training Model

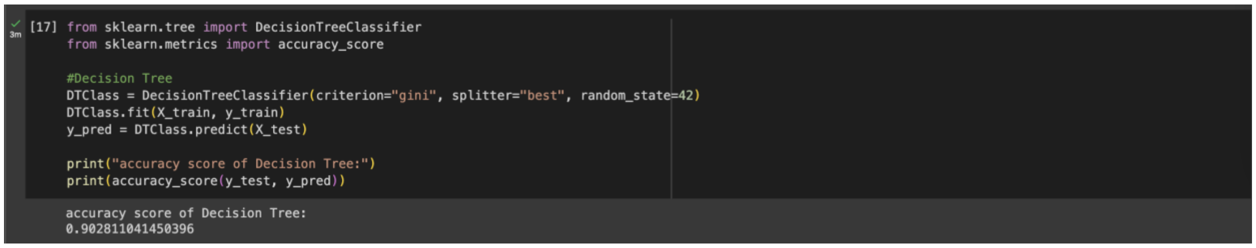
Label encoder

New section

```
[15] from tkinter.constants import Y
     from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     class_label = le.fit_transform(dataframe["CATEGORY"])
     # list(le.classes_)
     class_label

     array([0, 0, 0, ..., 2, 2, 2])
```

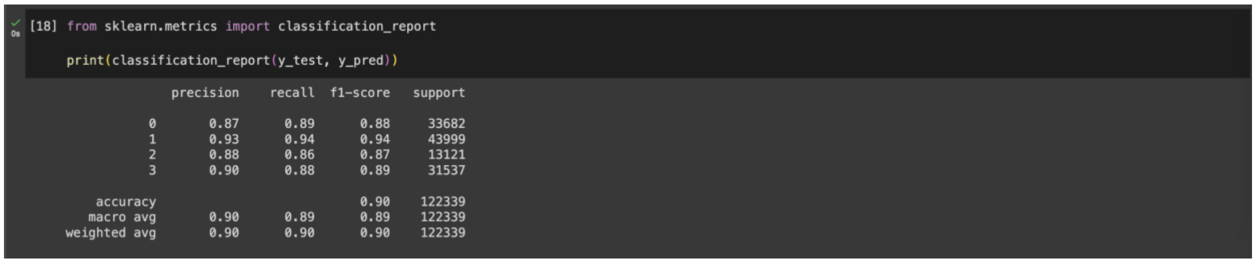


```
[16] from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(
         tfidf_dataset,  # Use the sparse matrix directly
         class_label,
         test_size=0.3
     )
```



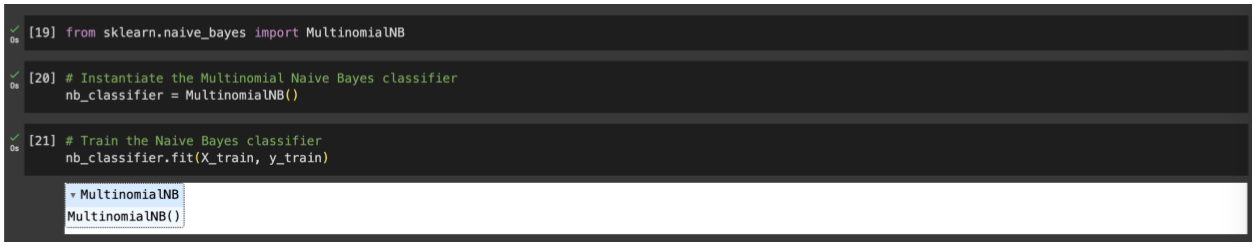
```
[17] from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score

     #Decision Tree
     DTClass = DecisionTreeClassifier(criterion="gini", splitter="best", random_state=42)
     DTClass.fit(X_train, y_train)
     y_pred = DTClass.predict(X_test)

     print("accuracy score of Decision Tree:")
     print(accuracy_score(y_test, y_pred))
```

```
accuracy score of Decision Tree:
0.902811041450396
```



```
[18] from sklearn.metrics import classification_report

     print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88     33682
           1       0.93      0.94      0.94     43999
           2       0.88      0.86      0.87     13121
           3       0.90      0.88      0.89     31537

    accuracy                           0.90    122339
   macro avg       0.90      0.89      0.89    122339
weighted avg       0.90      0.90      0.90    122339
```

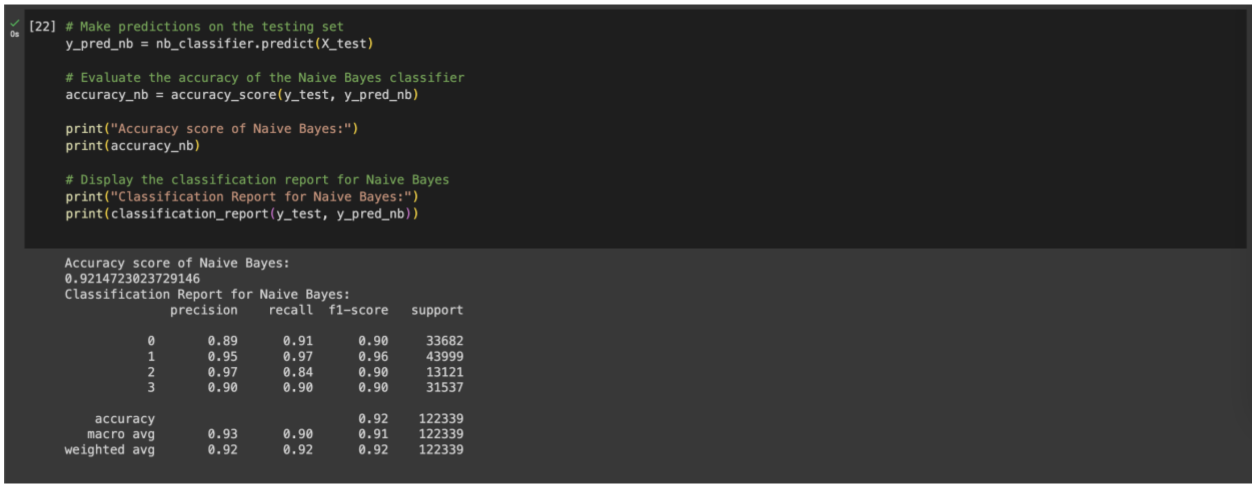


```
[19] from sklearn.naive_bayes import MultinomialNB
```

```
[20] # Instantiate the Multinomial Naive Bayes classifier
     nb_classifier = MultinomialNB()
```

```
[21] # Train the Naive Bayes classifier
     nb_classifier.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

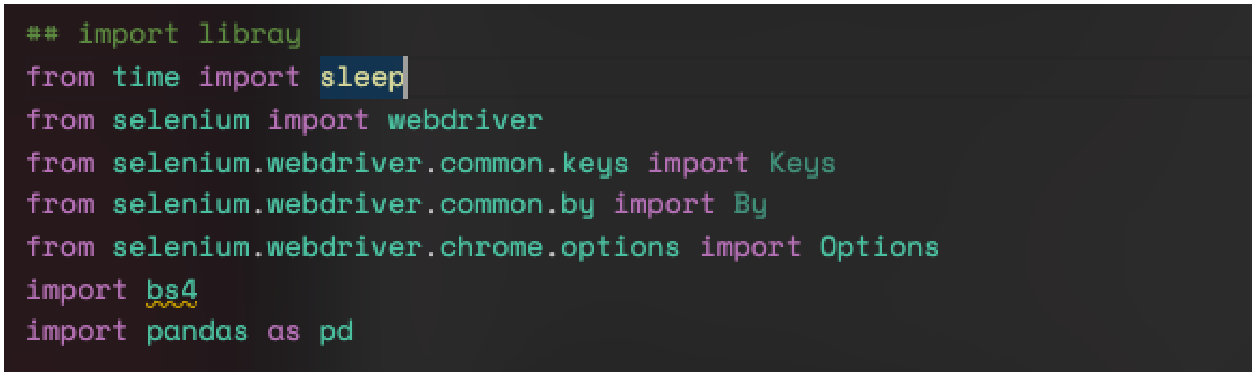


```
[22] # Make predictions on the testing set
     y_pred_nb = nb_classifier.predict(X_test)

     # Evaluate the accuracy of the Naive Bayes classifier
     accuracy_nb = accuracy_score(y_test, y_pred_nb)

     print("Accuracy score of Naive Bayes:")
     print(accuracy_nb)

     # Display the classification report for Naive Bayes
     print("Classification Report for Naive Bayes:")
     print(classification_report(y_test, y_pred_nb))
```

```
Accuracy score of Naive Bayes:
0.9214723023729146
Classification Report for Naive Bayes:
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     33682
           1       0.95      0.97      0.96     43999
           2       0.97      0.84      0.90     13121
           3       0.90      0.90      0.90     31537

    accuracy                           0.92    122339
   macro avg       0.93      0.90      0.91    122339
weighted avg       0.92      0.92      0.92    122339
```

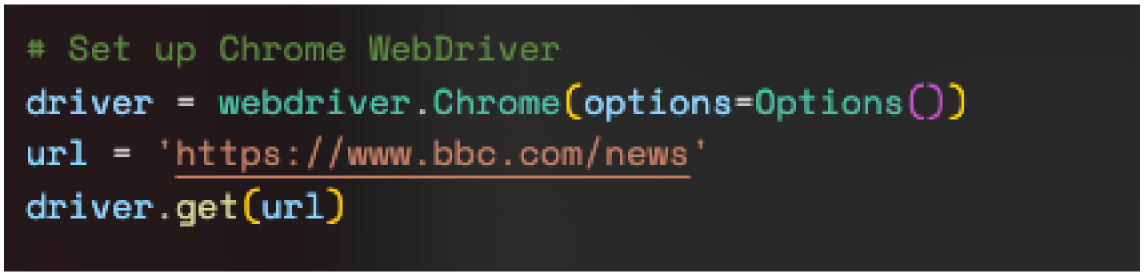
Web Scraping Data

1. Import Library



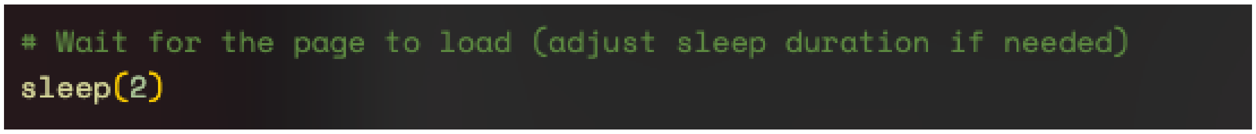
```
## import librQy
from time import sleep
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
import bs4
import pandas as pd
```

2. Set up chrome WebDrive



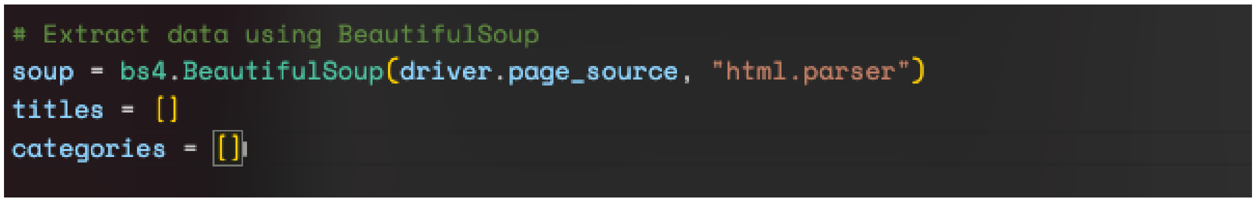
```
# Set up Chrome WebDriver
driver = webdriver.Chrome(options=Options())
url = 'https://www.bbc.com/news'
driver.get(url)
```

3. Sleep



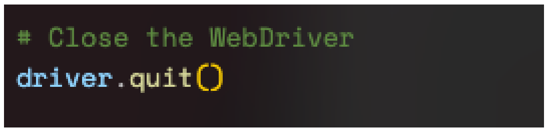
```
# Wait for the page to load (adjust sleep duration if needed)
sleep(2)
```

4. Extract data using BeautifulSoup



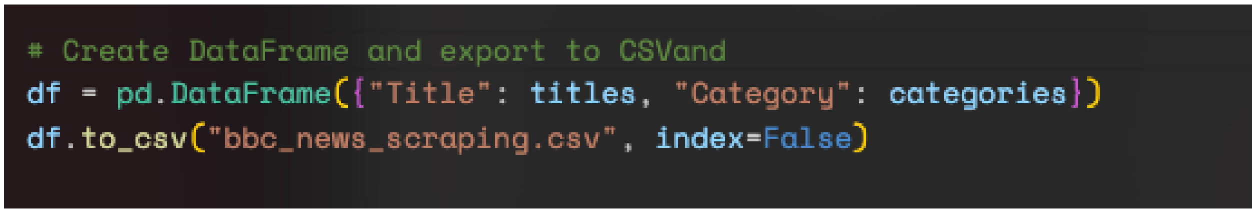
```
# Extract data using BeautifulSoup
soup = bs4.BeautifulSoup(driver.page_source, "html.parser")
titles = []
categories = []
```

5. Close or end WebDriver



```
# Close the WebDriver
driver.quit()
```

6. Create Dataframe and export csv



```python
# Create DataFrame and export to CSVand
df = pd.DataFrame({"Title": titles, "Category": categories})
df.to_csv("bbc_news_scraping.csv", index=False)
```