Report on Scraping YouTube Video

Scraping YouTube (or similar websites like Dailymotion) to collect data for classifying videos (based on description) in following 6 categories –

- Travel Blogs
- Science and Technology
- Food
- Manufacturing
- History
- Art and Music

Step 1: -

- First, I Imported Necessary Libraries

  {

  from selenium import webdriver

  ```python
  import pandas as pd
  from selenium.webdriver.common.by import By
  from selenium.webdriver.support.ui import WebDriverWait
  from selenium.webdriver.support import expected_conditions as EC
  ```

  }

- Then I called Chrome Web driver i.e, (driver = webdriver.Chrome())
- Then I Fetching all the links of a particular category on a page
- Then created dataframe for each categories.

  ```python
  {(columns = ['link', 'title', 'description', 'catagory'])}
  ```

- Scraping the video id, title, description and storing them into the dataframe
- Then, concatenated dataframe together.

  ```python
  frames = [df_travel, df_science, df_food, df_manufacturing,
  df_history, df_artndance]
  ```

- Lastly stored the data in csv file.

  Example:-

4	N6wv6r8KLk	It's SNOWING in INDIA | SOLANG VALLEY, MANALI	It's snowing in India??? Seriously though… We …	Travel
5	RHON79UmBXY	VIETNAM : INDIAN Solo Travel Journey just 6000…	Business query:- mithileshblacky0@gmail.com\n\…	Travel
6	kTuDFngiLE	Remote Cabin life in Indian Himalayas	Backpacking India World Trip Vlog 461.\nDon't …	Travel
7	dDiEcXXulmY	THE TRUTH ABOUT INDIA: Expectation vs Reality …	Time to bash some misconceptions and myths abo…	Travel
8	2YbsFSOV	I BOUGHT A ROYAL ENFIELD in INDIA!! (FINALLY!!)	Travel Vlogger, Conner Sullivan, looks to buy …	Travel

<u>Step 2: -</u>

    i)   I have chosen **SVM**s cause it giving more accuracy then other two.

    ii)   Chosen **Random Forest** for **Bagging and Boosting** i.e., Ensemble learning is a machine learning paradigm where multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results. The main hypothesis is that when weak models are correctly combined, we can obtain more accurate and/or robust models.

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree.

<u>Step 3: -</u>

    i)   The data I scrapped is in excel sheet format named "**yt_dataset.csv**"

    ii)   First, I Gathered data
Data pre-processing
Researching the model that will be best for the type of data
Training and testing the model
Evaluation

Library and Framework used: -
(**Selenium, Chrome Web, NLTK**)

My first step is to gather data from YouTube like video link, description, title and lastly category. And then I used to preprocess the data cause the data isn't structured, as the data is **unstructured**, I have structured the data using many methods by splitting with the help of **Selenium**, the best framework to scrap web contents like images, link, paragraph and many contents.

Then, after the **preprocessing step**, next step is to remove missing data and checking the outliers. Also I have **removed stop words** and unnecessary words from description.
Now performing label encoder as we have many categories and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

Lastly, I have **splitted the data into 80:20 where 80% is Training data and 20%** is Test Data. Now I have used multiple Machine Learning model to predict which model perform best on this dataset and after evaluation the result as follow.

- **Support Vector Machine** : - 95.64% Accuracy
- **Random Forest**   :- 96.05% Accuracy
- **Naïve Bayes**   :- 81.07% Accuracy

So, Support Vector Machine is the best Model for this dataset.

iii)   The reason behind I have chosen **Support Vector Machine** i.e., Machine Learning Model because it was giving more accuracy than other two (Linear Classifier and Naïve Bayes Classifier)

The biggest difference between the models you're building from a **"features" point of view is that Naive Bayes treats them as independent, whereas SVM looks at the interactions between them to a certain degree**, as long as you're using a non-linear kernel (Gaussian, rbf, poly etc.)

iv)   Precision, Recall and F1 Score as follow: -
F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

Naive Bayes

```
In [464]: classifier.score(X_test, y_test)

Out[464]: 0.9605970149253731

In [485]: print(classification_report(y_test, y_pred))

                         precision    recall  f1-score   support

          Art & Dance        0.95      0.97      0.96       313
                 Food        0.96      0.99      0.98       272
              History        0.96      0.98      0.97       287
        Manufacturing        0.95      0.94      0.94       241
              Science        0.97      0.94      0.96       289
               Travel        0.98      0.93      0.96       273

            micro avg        0.96      0.96      0.96      1675
            macro avg        0.96      0.96      0.96      1675
         weighted avg        0.96      0.96      0.96      1675
```

Random Forest

```
In [474]: classifier2.score(X_test, y_test)

Out[474]: 0.8107462686567164

In [475]: # Making the Confusion Matrix
          cm2 = confusion_matrix(y_test, y_pred2)

In [476]: cm2

Out[476]: array([[289,   4,  10,   2,   0,   8],
                 [  0, 253,   1,   0,   0,  18],
                 [ 40,   4, 194,  16,  10,  23],
                 [  1,   7,   4, 219,   4,   6],
                 [  4,   6,  59,  15, 199,   6],
                 [  3,  55,   4,   0,   7, 204]])
```

Support Vector Machine

```
In [469]: classifier1.score(X_test, y_test)
Out[469]: 0.9564179104477611

In [470]: # Making the Confusion Matrix
          cm1 = confusion_matrix(y_test, y_pred1)

In [471]: cm1
Out[471]: array([[301,   0,   4,   6,   0,   2],
                 [  0, 266,   0,   1,   0,   5],
                 [  2,   1, 278,   3,   2,   1],
                 [  0,   1,   4, 229,   4,   3],
                 [  1,   0,   9,   9, 270,   0],
                 [  2,   4,   1,   6,   2, 258]])
```

v)   In ML, there is no specific model or an algorithm which can give 100% result to every single dataset. We need to understand the data before we apply any algorithm and build our model depending on the desired result. This dataset gives us 100% accuracy, which is nearly impossible. From the above models, **Random Forest and SVMs** gives optimal accuracy compared to other algorithms because it works best with continuous data and it also applies a nonlinear relationship to the features. By using this algorithm, you reduce the chances of overfitting and the variance in the data which thus leads to better accuracy.

--Thank You--