Report

Data Scraping :-

Code Snippet :-

```
from selenium import webdriver
from bs4 import BeautifulSoup
from googlesearch import search
from time import sleep
from pyvirtualdisplay import Display
from requests import get
import pandas as pd
import json
import re

# display = Display(visible=0, size=(800,600))
# display.start()

chromeOptions = webdriver.ChromeOptions()
prefs = {"profile.managed_default_content_settings.images" : 2}
chromeOptions.add_experimental_option("prefs", prefs)

url = 'https://www.youtube.com/'
youtube_list = []

def load_complete_page(driver, clicks):
    for i in range(clicks):
        driver.execute_script("window.scrollBy(0,2500)")
        sleep(2)

try:
    driver = webdriver.Chrome(executable_path='/usr/share/safaridriver',
chrome_options=chromeOptions)
    driver.get(url)
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')
    search = driver.find_element_by_css_selector('input#search')
    search.send_keys("travel blog")
    driver.find_element_by_css_selector('#search-icon-legacy').click()
    sleep(5)
```

```
    load_complete_page(driver, 300)

    container = driver.find_elements_by_tag_name('ytd-video-renderer')
    for i, each in enumerate(container):
        video_id = each.find_element_by_tag_name('a').get_attribute('href')
        title = each.find_element_by_css_selector('#video-title').text
        description = each.find_element_by_css_selector('#description-text').text
        category = 'travel'

        content = {
            'video_id': video_id,
            'title': title,
            'description': description,
            'category': category
        }

        youtube_list.append(content)
        print (i, content)

except Exception as e:
    print ('Some error occured: ', e)
finally:
    driver.close()
    # display.stop()

    df = pd.DataFrame(youtube_list)
    df = df[['video_id', 'title', 'description', 'category']]
    df.to_csv('youtube.csv', index=False)
```

NOTE :- Above Code Snippet is just for travel vlogs from **youtube** . We have done the similar thing for different categories and different sites like **Dailymotion** .

Preprocessing of dataset : -

In preprocessing ,

- Removed all unwanted symbols , alphanumerics , numericals etc from the Title and Description columns .
- Only lowercase raw text for the classification .
- For the classification , used both Title and Description columns .

NOTE :- Code snippets are in the code page itself .

Techniques or models used for the classification :-

- **Logistic Regression (Linear classifier) :-**
  Reasons to use :-
    - Works extremely well with high dimensional dataset .
    - Model Interpretability is very good .
    - Simple Model with simple assumptions .
    - Computationally cheap and it works extremely well with text classification .

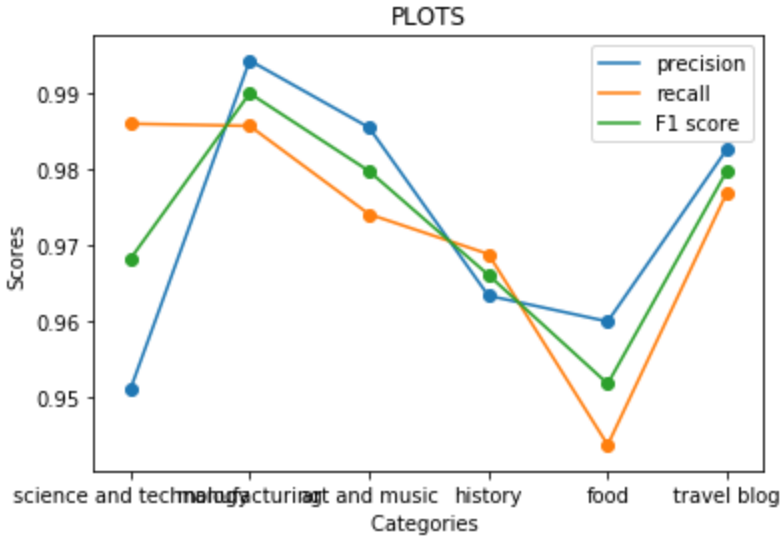
- **Shallow NN (2 hidden layer neural net) :-**
  Reasons to use :-
    - Works extremely well with high dimensional dataset with good accuracy .
    - Worked extremely well with this text dataset classification .
    - Handles trade-off between bias and variance extremely well in this problem.

- **LSTM :-**
  Reasons to use :-
    - Retains sequence relationship between the words of a sentence which adds advantage to the classification.
    - Handles trade-off between bias and variance extremely well in this problem .
    - Worked extremely well with this text classification.

NOTE :- Code snippets for all the models are in the code page itself .

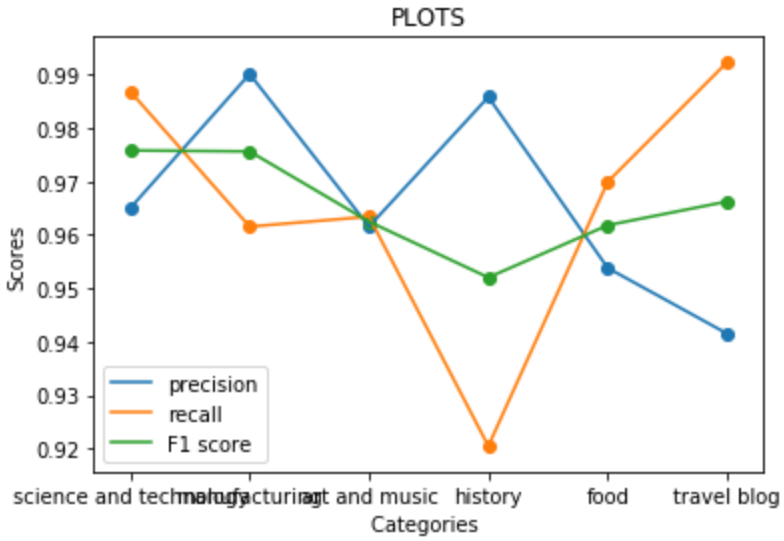
Precision, Recall, and F1 Score :-

- **Logistic Regression (Linear classifier) :-**



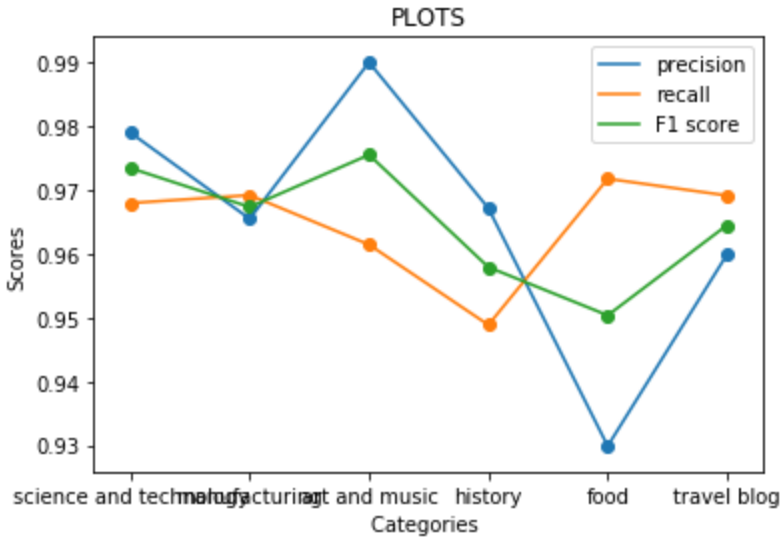
Recall :- `array([0.98587571, 0.98559078, 0.97398844, 0.96875 ,0.94366197,0.97687861])`
Precision :- `array([0.95095368, 0.99418605, 0.98538012, 0.96327684, 0.95988539, 0.98255814])`
F1 :- `array([0.96809986, 0.98986975, 0.97965116, 0.96600567, 0.95170455, 0.97971014])`

- **Shallow NN (2 hidden layer neural net) :-**



Recall :- `array([0.98681733, 0.96153846, 0.96339114, 0.92045455, 0.96992481, 0.99229287])`
Precision :- `array([0.96500921, 0.99009901, 0.96153846, 0.98580122, 0.95378928, 0.94149909])`
F1 :- `array([0.97579143, 0.97560976, 0.96246391, 0.95200784, 0.96178938, 0.96622889])`

- **LSTM :-**



Recall :- `array([0.96798493, 0.96923077, 0.96146435, 0.94886364, 0.97180451, 0.96917148])`

Precision :- `array([0.97904762, 0.96551724, 0.99007937, 0.96718147, 0.92985612, 0.95992366])`

F1 :- `array([0.97348485, 0.96737044, 0.97556207, 0.95793499, 0.95036765, 0.96452541])`

NOTE :- Code snippets are in the code page itself .

Results :-

- **Logistic Regression (Linear classifier) :-**

  In this model , “One Over Rest “ technique is used for multiclass classification where each class is being separated by other classes by plane / hyperplane . Classification is done over these planes .

  The accuracy of the model is pretty high because : -
  - Tuned the hyperparameter(lambda) of the model with cross validate dataset before testing on the test dataset .
  - Used L2 regularization to make the trade-off between overfitting and underfitting .
  - Used “One Over Rest “ technique for multiclass classification .

  How results would have been different ?
  - Don’t tune the hyperparameter (lambda) before testing on the test dataset .
  - Wouldn’t have used any regularizer which easily overfit our model.

- **Shallow NN (2 hidden layer neural net) :-**

  In this model , Neurons are being trained . Weights are being optimized by backpropagation . Classification is done with these weights and neurons .

  The accuracy of the model is pretty high because : -
  - Used activation function as ReLu instead of Sigmoid .
  - Initialized the weights matrix as RandomNormal(mean=0.0, stddev=0.1, seed=None).
  - Used Softmax function as activation in the final dense layer .
  - Used optimizer = 'adam' .

  How results would have been different ?
  - Used Sigmoid instead ReLu .

- Used different optimizer like adagrad .
- Initialization should be done with different method .
- If used deep NNs without dropout and batch normalization would have done overfitting of our model .

- **LSTM :-**

  In this model , padded sentences is being given the input .Weights are being optimized by backpropagation . Classification is done with these weights and LSTMs .

  The accuracy of the model is pretty high because : -
  - Padded every sentence .
  - Used optimizer = 'adam' .
  - Used dropout and recurrent_dropout to avoid overfitting of our model .
  - Used SpatialDropout1D.
  - Used Softmax as the activation function .

  How results would have been different ?
  - Used different optimizer like adagrad .
  - Won't have used dropout and recurrent_dropout .
  - Won't padded the sentences .

NOTE :- Code snippets for all the models and its results are in the code page itself .

----------END----------