DEEPAK SINGH — Published On December 27, 2021 and Last Modified On August 22nd, 2022

Advanced Data Visualization NLP Python Text

This article was published as a part of the Data Science Blogathon

Introduction

A news article discusses current or recent news of either general interest (i.e. daily news (i.e. political or trade news magazines, club newsletters, or technology news websites) accounts of eyewitnesses to the happening event. We must have seen the news divided int news website. Some of the popular categories that you'll see on almost any news webs sports, etc. If you want to know how to classify news categories using machine learning, thi

Every news website classifies the news article before publishing it so that every time vi easily click on the type of news that interests them. For example, I like to read the latest time I visit a news website, I click on the technology section. But you may or may not like t may be interested in politics, business, entertainment, or maybe sports. Currently, the n hand by the content managers of news websites. But to save time, they can also implement their websites that read the news headline or the content of the news and classifies the cat

Text Classification

Text classification datasets are used to categorize natural language texts according to classifying news articles by topic, or classifying book reviews based on a positive classification is also helpful for language detection, organizing customer feedback, and frau

While this process is time-consuming when done manually, it can be automated with machi

Category classification, for news, is a multi-label text classification problem. The goal is to a to a news article. A standard technique in multi-label text classification is to use a set of bin

Data preprocessing is the process of transforming raw data into an understandable forma
in data mining as we cannot work with raw data. The quality of the data should be chec
learning or data mining algorithms.

Import Libraries

let's import the necessary Python libraries and the dataset that we need for this task.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
nltk.download('punkt')
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import make_scorer, roc_curve, roc_auc_score
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
```

Import Dataset

Text Classification of News Articles

Check the shape (row and column) of the dataset.

```
dataset.shape
```

```
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ArticleId  1490 non-null   int64
 1   Text       1490 non-null   object
 2   Category   1490 non-null   object
dtypes: int64(1), object(2)
memory usage: 35.0+ KB
(1490, 3)
```

Check Information of Columns of Dataset

Count Values of Categories

There are five news categories i.e. Sports, Business, Politics, Entertainment, Tech.

```
dataset['Category
```

```
sport            346
business         336
politics         274
entertainment    273
tech             261
Name: Category, dtype: int64
```

Convert Categories Name into Numerical Index

Convert the given news categories into categorical values.

```
# Associate Category names with numerical index and save it in new column CategoryId
target_category = dataset['Category'].unique()
print(target_category)
```

```
['business' 'tech' 'politics' 'sport' 'entertainment']
```

```
dataset['CategoryId'] = dataset['Category'].factorize()[0]
dataset.head()
```

	ArticleId	Text	Category	C
0	1833	worldcom ex-boss launches defence lawyers defe...	business	
1	154	german business confidence slides german busin...	business	
2	1101	bbc poll indicates economic gloom citizens in ...	business	
3	1976	lifestyle governs mobile choice faster bett...	tech	
4	917	enron bosses in $168m payout eighteen former e...	business	

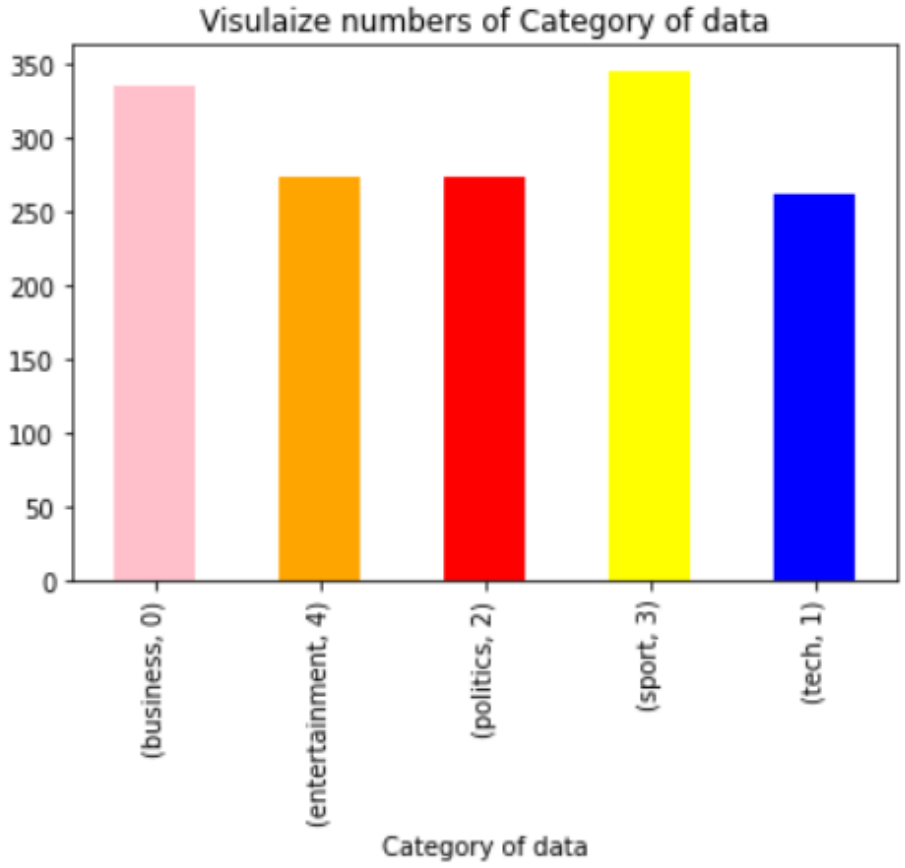
3	tech	1
5	politics	2
6	sport	3
7	entertainment	4

In data mining, Exploratory Data Analysis (EDA) is an approach to analyzing datasets to summarize the with visual methods. EDA is used for seeing what the data can tell us before the modeling task. It is not numbers or a whole spreadsheet and determine important characteristics of the data. It may be tediou to derive insights by looking at plain numbers. Exploratory data analysis techniques have been devised

Visualizing Data

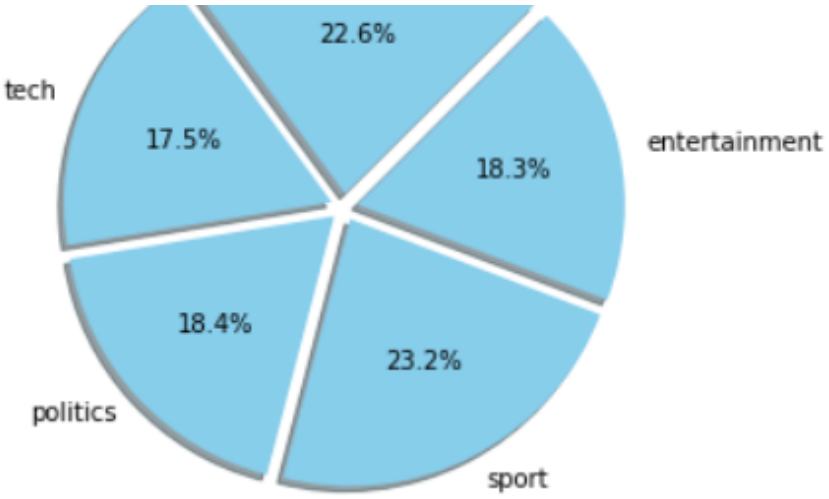
The below graph shows the news article count for category from our

```
dataset.groupby('Category').CategoryId.value_counts().plot(kind = "bar", color = ["pink",
"blue"])
plt.xlabel("Category of data")
plt.title("Visulaize numbers of Category of data")
plt.show()
```




```
fig = plt.figure(figsize = (5,5))
colors = ["skyblue"]
```

Text Classification of News Articles



Visualizing Category Related Words

Here we use the word cloud module to show the category-related words.

Word Cloud is a data visualization technique used for representing text data in which the size of each w
importance. Significant textual data points can be highlighted using a word cloud. Word clouds are wid
social network websites.

```
from wordcloud import WordCloud

stop = set(stopwords.words('english'))

business = dataset[dataset['CategoryId'] == 0]

business = business['Text']

tech = dataset[dataset['CategoryId'] == 1]

tech = tech['Text']

politics = dataset[dataset['CategoryId'] == 2]

politics = politics['Text']

sport = dataset[dataset['CategoryId'] == 3]

sport = sport['Text']

entertainment = dataset[dataset['CategoryId'] == 4]

entertainment = entertainment['Text']

def wordcloud_draw(dataset, color = 'white'):
```

```
wordcloud_draw(business, 'white')

print("tech related words:")

wordcloud_draw(tech, 'white')

print("politics related words:")

wordcloud_draw(politics, 'white')

print("sport related words:")

wordcloud_draw(sport, 'white')

print("entertainment related words:")

wordcloud_draw(entertainment, 'white')
```

business related words:

tech related words:

sport related words:

Show Text Column of Dataset

```
text = dataset["Text"]
text.head(10)
```

```
0    worldcom ex-boss launches defence lawyers defe...
1    german business confidence slides german busin...
2    bbc poll indicates economic gloom citizens in ...
3    lifestyle  governs mobile choice  faster  bett...
4    enron bosses in $168m payout eighteen former e...
5    howard  truanted to play snooker  conservative...
6    wales silent on grand slam talk rhys williams ...
7    french honour for director parker british film...
8    car giant hit by mercedes slump a slump in pro...
9    fockers fuel festive film chart comedy meet th...
Name: Text, dtype: object
```

Show Category Column of Dataset

```
category = dataset['Categor
category.head(10)
```

```
0          business
1          business
2          business
3              tech
4          business
5          politics
6             sport
```

Remove All Tags

```
dataset['Text'] = dataset['Text'].apply(special_char)
```

We convert all articles or text to lower case.

It is one of the simplest and most effective forms of text preprocessing. It is applicable to most text mi
help in cases where your dataset is not very large and significantly helps with the consistency of expec

```
def convert_lower(text):
   return text.lower()
dataset['Text'] = dataset['Text'].apply(convert_lower)
dataset['Text'][1]
```

Remove all Stopwo

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been progra
indexing entries for searching and when retrieving them as the result of a search query.
We would not want these words to take up space in our database, or take up the valuable processing ti
them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit)
stored in 16 different languages.

```
def remove_stopwords(text):
  stop_words = set(stopwords.words('english'))
  words = word_tokenize(text)
  return [x for x in words if x not in stop_words]
dataset['Text'] = dataset['Text'].apply(remove_stopwords)
dataset['Text'][1]
```

Lemmatizing the

Lemmatization is the process of grouping together the different inflected forms of a word so they can l
Lemmatization is similar to stemming but it brings context to the words. So it links words with similar m
lemmatization is preferred over Stemming because lemmatization does morphological analysis of the w

```
def lemmatize_word(text):
  wordnet = WordNetLemmatizer()
  return " ".join([wordnet.lemmatize(word) for word in text])
dataset['Text'] = dataset['Text'].apply(lemmatize_word)
dataset['Text'][1]
```

`dataset`

After Cleaning Text our Dataset

	ArticleId	Text	Cate
0	1833	worldcom ex bos launch defence lawyer defendin...	bus
1	154	german business confidence slide german busine...	bus
2	1101	bbc poll indicates economic gloom citizen majo...	bus
3	1976	lifestyle governs mobile choice faster better ...	
4	917	enron boss 168m payout eighteen former enron d...	bus

sentence is a frequent word, we set it as 1, else we set it as 0.

Whenever we apply any algorithm in NLP, it works on numbers. We cannot directly feed our text into t
Words model is used to preprocess the text by converting it into a *bag of words*, which keeps a count o
most frequently used words.

```
from sklearn.feature_extraction.text import CountVectorizer
x = np.array(dataset.iloc[:,0].values)
y = np.array(dataset.CategoryId.values)
cv = CountVectorizer(max_features = 5000)
x = cv.fit_transform(dataset.Text).toarray()
print("X.shape = ",x.shape)
print("y.shape = ",y.shape)
```

```
Test Accuracy Score of Basic Logistic Regression: % 97.09
Precision : 0.970917225950783
Recall    : 0.970917225950783
F1-score   : 0.9709172259507831
```

```
X.shape =  (1490, 5000)
y.shape =  (1490,)
```

Train Test and Split the Dataset

We need to split a dataset into train and test sets to evaluate how well our machine learning model per
the model, the statistics of the train set are known. The second set is called the test data set, this set is

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state =
print(len(x_train))
print(len(x_test))
```

```
1043
447
```

Create Empty List

```
#create list of model and accuracy
perform_list = [ ]
```

Create, Fit and Predict all ML Model

```
def run_model(model_name, est_c, est_pnlty):

mdl=''

if model_name == 'Logistic Regression':

mdl = LogisticRegression()
```

```
mdl = GaussianNB()

oneVsRest = OneVsRestClassifier(mdl)

oneVsRest.fit(x_train, y_train)

y_pred = oneVsRest.predict(x_test)

# Performance metrics

accuracy = round(accuracy_score(y_test, y_pred) * 100, 2)

# Get precision, recall, f1 scores

precision, recall, f1score, support = score(y_test, y_pred, average='micro')

print(f'Test Accuracy Score of Basic {model_name}: % {accuracy}')

print(f'Precision : {precision}')

print(f'Recall : {recall}')

print(f'F1-score : {f1score}')

# Add performance parameters to list

perform_list.append(dict([

('Model', model_name),

('Test Accuracy', round(accuracy, 2)),

('Precision', round(precision, 2)),

('Recall', round(recall, 2)),

('F1', round(f1score, 2))

]))
```

Logistic Regression

```
run_model('Logistic Regression', est_c=None, est_pnlty=
```

```
Test Accuracy Score of Basic Logistic Regression: % 97.09
Precision : 0.970917225950783
Recall    : 0.970917225950783
F1-score   : 0.9709172259507831
```

Ran

```
run_model('Random Forest', est_c=None, est_pnlty=None)
```

```
F1-score    : 0.9664429530201343
```

```
run_model('Decision Tree Classifier', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Decision Tree Classifier: % 83.22
Precision : 0.8322147651006712
Recall    : 0.8322147651006712
F1-score    : 0.8322147651006712
```

K

```
run_model('K Nearest Neighbour', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic K Nearest Neighbour: % 73.6
Precision : 0.7360178970917226
Recall    : 0.7360178970917226
F1-score    : 0.7360178970917226
```

Gauss

```
run_model('Gaussian Naive Bayes', est_c=None, est_pnlty=None)
```

```
Test Accuracy Score of Basic Gaussian Naive Bayes: % 76.06
Precision : 0.7606263982102909
Recall    : 0.7606263982102909
F1-score    : 0.7606263982102909
```

Create Dataframe of Model, Accuracy, Precision, R

```
model_performance = pd.DataFrame(data=perform_list)
model_performance = model_performance[['Model', 'Test Accuracy', 'Precision', 'Recall', 'F
model_performance
```

	Model	Test Accuracy	Precision	Recall	F1
0	Logistic Regression	97.09	0.97	0.97	0.97
1	Random Forest	97.99	0.98	0.98	0.98
2	Multinomial Naive Bayes	97.09	0.97	0.97	0.97
3	Support Vector Classifer	96.64	0.97	0.97	0.97
4	Decision Tree Classifier	83.22	0.83	0.83	0.83
5	K Nearest Neighbour	73.60	0.74	0.74	0.74
6	Gaussian Naive Bayes	76.06	0.76	0.76	0.76

```
y_pred1 = cv.transform(['Hour ago, I contemplated retirement for a lot of reasons. I felt
sensitive enough to my injuries. I felt like a lot of people were backed, why not me? I ha
won a lot of games for the team, and I am not feeling backed, said Ashwin'])
yy = classifier.predict(y_pred1)
result = ""
if yy == [0]:
  result = "Business News"
elif yy == [1]:
  result = "Tech News"
elif yy == [2]:
  result = "Politics News"
elif yy == [3]:
  result = "Sports News"
elif yy == [1]:
  result = "Entertainment News"
print(result)
```

```
Sports News
```

Conclusion

Finally after doing Data cleaning and Data Preprocessing (cleaning data, train_test_split model, creatin machine learning model) we got the accuracy scores and we can say that Random Forest Classification all machine learning models.

And at last, we also predict the category of different news articles.

GitHub repository for web scraping and data preprocessing is here.

If you have any queries, please let me know here.

Read more articles on Text Classification, here.

Thank you for your time and for reading my article. Please feel free to contact me if you have any quest comments.

The media shown in this article is not owned by Analytics Vidhya and are used at the Author's discre

blogathon data analysis data visualization dataset kaggle text classification

About the Author

Next Post

All NLP tasks using Transformers Pipeline

2 thoughts on "Text Classification of News Articles"

Classification of news article texts – News Couple says:
December 27, 2021 at 9:32 pm

[...] Source link [...]

Reply

Mahimagts11 says:
February 20, 2023 at 4:33 pm

A text dataset is a collection of data that consists of text documents, such as articles, boo
reviews, or any other form of written or typed text. Text datasets are used in various appl
processing (NLP), machine learning, and data mining. These datasets can be used to train
such as language models, sentiment analysis models, and text classification models. Text
places, including web scraping, public domain books, online forums, and social media plat

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

Beginner's Guide to Build Your Own Large Language Models from..

Aravindpai Pai - JUL 05, 2023