MET CS 521 Information Structures with Python

Final Project Write-up Paper

Instructor : Alan Burstein

Student name : Menghao Xu

Date : 12th Aug, 2020

Abstract

This write-up paper mainly described how to use Python to get data files from web, load data into data frame and make analysis. My project is about fake news detector. With the help of machine learning, we can predict future results according to previous data recording.

Files included

There are total 5 files included in my final project, The sequence described in this article is the sequence of operations for the entire program. For the 4th file, it is just this write-up paper which showed some analysis of results and conclusions based on this final project experience. And 5th is a sample file '123.txt' for testing extension 2.

1st: README.md

This file briefly showed how can reader understand and follow simply steps to use my project to generate final results.

2nd: download_script.ipynb (Script to download code)

It is the launching part of the entire program. With the request package help, we can download our target files from specific website. And then, we can use open and write functions to alter any information in the file.

By loading the written codes included, reader would automatically download our main data files the 'downloaded_fake_news.csv', which contains thousands of article titles, texts and their own fake or real properties.

3rd: news_exploration.ipynb (Script to generate analysis)

The script is the core analysis part of the project. Firstly, i set almost all the importing packages process to the head part, like pandas helps me transform information in csv file into a good visual data frame and sklearn package, which contains different kinds of tools, helps me transform text information into vectors. Then, by filtering some key values, we can easily generate some visual components, like all the titles content in the file, all the lines with Real or Fake labels and the first to forth lines in the data frame.

Extensions

(1) The next half of the file's content are two extensions(Trying different models and A simple application to upload and test articles) based on current codes status.

For the first extension, there are SVM, LinearSVC, LogisticRegression and MultinomialNB, four models that i tried to analysis the fake detector's ability. I set SVM model in the first position as a reference target. Before doing the prediction, we need to use TfidfVectorizer() function to transform current text into vectors, which could be more easily to be read and show. After title contents' transformation, i found that there are 6335 columns and 10071 vectors generated in current feature. To simplify the huge number of results, by using KFold method that i imported from sklearn before, i sorted all the results needed be showed into 5 groups, every group contains 1267 values. There previous work are suitable for all the next models.

Now, it is time for splitting data into train and test data sets. First step is setting different model function into its own model value, like SVC(), LogisticRegression() and MultinomialNB(alpha=1). That's the main difference of set up progress between these models. Then, using model.fit() to split data into train data and attach its label with them. Final step is make model.predict to generate predictions made by detectors in different models. There are two functions classification_report and recall_score used to show metrics for analysis.

There are some summaries that we can found and set as references, text summary of the precision, recall, F1 score for each class. And reported averages include macro average (averaging the unweighted mean per label) and weighted average (averaging the support-weighted mean per label).

Taking the SVM model as a example, the summary shows precision of FAKE is around 78%-81% in the five classes, means the fake news test accuracy rate is about 80%, on the other hand, REAL' rate is in range 84%-87% which is little higher than FAKE. For the recall parameter, it shows the correct sample ratio in the total test set. FAKE recall rate is about 86%, REAL's is relative lower, 78%. Another important key value is weighted avg, which shows the same data in all the models of this task. The weighted ave rate is nearly 83% in SVM model. In general, 83% accuracy rate detector is a reliable and recommended tool. So the SVM for predicting fake news is a successful case.

For the next models, the operation steps are much similar to the SVM, so i drew a table to show all the summary results as below,

	LogisticRegression		LinearSVC		MultinomialNB	
	Precision	Recall	Precision	Recall	Precision	Recall
FAKE	77%-81%	84%-87%	79%-82%	79%-85%	83%-87%	73%-78%
REAL	82%-86%	76%-78%	78%-83%	78%-81%	75%-80%	86%-88%
macro avg	81%-83%		79%-82%		81-84%	80%-83%
weighted avg	81%-83%		79%-82%		81-84%	80%-83%

From the table we can find that LogisticRegression model has more accuracy in FAKE recall area and REAL precision area. Interestingly, MultinomialNB shows the opposite results(good in FAKE precision and REAL recall). For average data both of them are stable and reliable with about 81%-83% accuracy ratio. For the LinearSVC model, all the data summary even lower than 80%, so this model is not that satisfactory.

When compared to SVM model, as the data ratio described, all the data of SVM would be little higher 2% than either LogisticRegression or MultinomiaNB. So this paper suggested, when testing a fake news detector, we can use SVM model as a primary option with about 83% weighted ave rate.

(2) The second extension is a simply application. Users could input or upload his/her own text information and generate some analysis of text structure and words importance automatically. For better showing the process and analytic results, i set three text lines as default, which could be changed any time.

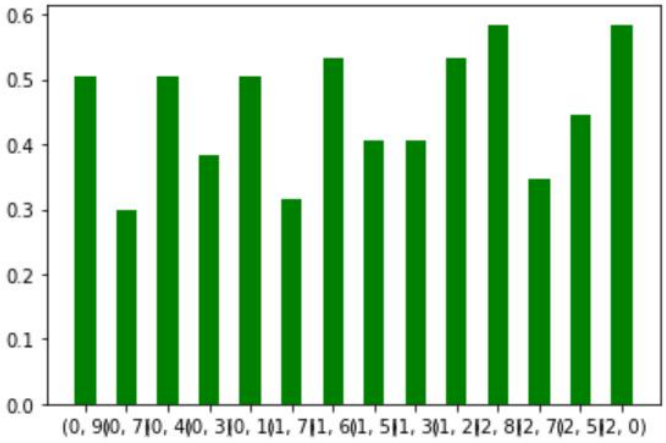
In the first step, i used CountVectorizer function from sklearn package to transform all the text info into vectors. We can find that all the key values changed into multiple tuples with its own occurred frequency. Then by using vectorizer.get_feature_names, there would show a sorted list with every unique word in the text that input before. Using toarray function, we will generate a matric. Users can get which accurate position in the matric showed one vector once. There also followed matric_name.shape function for easily getting the architecture of numerous projects.

In addition to previous details, there is one core value of the whole extension, TF-IDF value. It is a technique commonly used in information processing and data mining. This technology uses a statistical method to calculate the importance of a word in the entire corpus based on the number of occurrences of the word in the text and the frequency of the document in the entire corpus. We can use this advantage to filter out some common but irrelevant words, while retaining important words that affect the entire text.

By printing the tfidf result, we can catch all the word position in the whole matric and its term frequency.

```
TF-IDF Value
  (0, 9)        0.5046113401371842
  (0, 7)        0.2980315863446099
  (0, 4)        0.5046113401371842
  (0, 3)        0.3837699307603192
  (0, 1)        0.5046113401371842
  (1, 7)        0.3154441510317797
  (1, 6)        0.5340933749435833
  (1, 5)        0.4061917781433946
  (1, 3)        0.4061917781433946
  (1, 2)        0.5340933749435833
  (2, 8)        0.5844829010200651
  (2, 7)        0.34520501686496574
  (2, 5)        0.444514311537431
  (2, 0)        0.5844829010200651
```

The final step is mainly contributed to visual component part. With the help of matplotlib package, i can set X-axis as words' position labels and Y-axis as its term frequency and generate a bar chart to show the finding more obviously. In the bar setting part, i can also change default parameters in the function to set any color i like and the width of each bar.



Conclusion

Most data predictions are directly related to numbers, however the special feature of this project is that it can process and analysis text information. Through the study of this project, I learned how to obtain and load text data and how to perform data predictive analysis on text. The core part of the project is to master how to convert text information into numbers or vector information. Thanks to the help of sklearn package,so that i can easily achieve this goal. This article only shows some parts of the basic analysis and display skills, and there will be more and more interesting functions to be developed and learned in the further learning.