Code

Project 3: Movie Review Sentiment Analysis

Spring 2021

You are provided with a dataset consisting of 50,000 IMDB movie reviews, where each review is labelled as positive or negative. The goal is to build a binary classification model to predict the sentiment of a movie review.

Movie Review Data

Dataset

Download the dataset, `alldata.tsv` , from the Resouces page on Piazza. The dataset has 50,000 rows (i.e., reviews) and 4 columns:

- Col 1: “id”, the identification number;
- Col 2: “sentiment”, 0 = negative and 1 = positive;
- Col 3: “score”, the 10-point score assigned by the reviewer. Scores 1-4 correspond to negative sentiment; Scores 7-10 corrspond to positive sentiment. This dataset contains no reviews with score 5 or 6.
- Col 4: “review”.

Test IDs

Download `splits_S21.csv` from the Resouces page. This file contains 25,000 rows and 5 columns, each column containing the 25,000 row-numbers of a test data.

You can use the following code to generate the 5 sets of training/test splits with each set (3 files: `train.tsv` , `test.tsv` , `test_y.tsv`) stored in a subfolder. Note that the training data do not contain the “score” column; this is to avoid students mistakenly using “score” as an input feature.

Hide

```
data <- read.table("alldata.tsv", stringsAsFactors = FALSE,
                   header = TRUE)
testIDs <- read.csv("splits_S21.csv", header = TRUE)
for(j in 1:5){
  dir.create(paste("split_", j, sep=""))
  train <- data[-testIDs[,j], c("id", "sentiment", "review") ]
  test <- data[testIDs[,j], c("id", "review")]
  test.y <- data[testIDs[,j], c("id", "sentiment", "score")]

  tmp_file_name <- paste("split_", j, "/", "train.tsv", sep="")
  write.table(train, file=tmp_file_name,
              quote=TRUE,
              row.names = FALSE,
              sep='\t')
  tmp_file_name <- paste("split_", j, "/", "test.tsv", sep="")
  write.table(test, file=tmp_file_name,
              quote=TRUE,
              row.names = FALSE,
              sep='\t')
  tmp_file_name <- paste("split_", j, "/", "test_y.tsv", sep="")
  write.table(test.y, file=tmp_file_name,
            quote=TRUE,
            row.names = FALSE,
            sep='\t')
}
```

Source

A subset of this data set was used in a Kaggle competition: Bag of Words Meets Bags of Popcorn (https://www.kaggle.com/c/word2vec-nlp-tutorial). You can find useful discussion and sample code there.

Performance Target

The goal is to build a binary classification model to predict the sentiment of a review with a **vocabulary size** less than or equal to **1000**. You should use the same vocabulary for all five training/test datasets.

Evaluation metric is **AUC** on the test data. You should try to produce AUC equal to or bigger than **0.96** over all five test data.

What to Submit

Submit the following **four** items on Coursera:

- A txt file named **myvocab.txt** that containing your word list: one word in a row and the number of rows should be less than 2000.
- An R/Python Markdown file in HTML explaining how you construct your vocabulary.
- A report (4-page maximum, PDF or HTML) that provides the technical detail of your model, your implementation and any interesting findings. In addition, report the performance for the five test datasets, running time of your code and the computer system you use (e.g., Macbook Pro, 2.53 GHz, 4GB memory, or AWS t2.large).

You **do not** have to explain how you construct the vocabulary in the report since it should already been included in the Markdown file.

- R/Python code in a single file named **mymain.R** (or **mymain.py**) that takes your myvocab.txt, a training data and a test data as input and outputs one submission file (the format of the submission file is given below).

  Your mymain.R should look like the following.

Hide

```
####################################
# Load your vocabulary and training data
####################################
myvocab <- scan(file = "myvocab.txt", what = character())
train <- read.table("train.tsv", stringsAsFactors = FALSE,
                    header = TRUE)

####################################
#
# Train a binary classification model
#
####################################


test <- read.table("test.tsv", stringsAsFactors = FALSE,
                     header = TRUE)

####################################
# Compute prediction
# Store your prediction for test data in a data frame
# "output": col 1 is test$id
#           col 2 is the predited probabilities
####################################

write.table(output, file = "mysubmission.txt",
            row.names = FALSE, sep='\t')
```

Code Evaluation

We shall run the command "source(mymain.R)" in a directory, in which there are only three files: `myvocab.txt`, `train.tsv` and `test.tsv`.

After running your code, we should see **one** txt file in the same directory named `mysubmission.txt`. Then we shall move `test_y.tsv` to this directory, and load `test_y.tsv` and `mysubmission.txt` to compute AUC.

Submission File Format: `mysubmission.txt` should look like the following:

```
"id"    "prob"
47604   0.940001011154441
36450   0.584891891011812
30088   0.499236341444505
18416   0.00687786009135037
```

Our evaluation R code looks like the following:

Hide

```
library(pROC)
source(mymain.R)
# move test_y.tsv to this directory
test.y <- read.table("test_y.tsv", header = TRUE)
pred <- read.table("mysubmission.txt", header = TRUE)
pred <- merge(pred, test.y, by="id")
roc_obj <- roc(pred$sentiment, pred$prob)
pROC::auc(roc_obj)
```

The evaluation for **Python code** is similar: after typing "python mymain.py" in the directory containing three files `myvocab.txt`, `train.tsv` and `test.tsv`, we should see **one** txt file in the same directory named `mysubmission.txt`. Then we can just use the R code above to compute the correspoding AUC, which should be the same as the output from `sklearn.metrics.roc_auc_score`.