learn-co-curriculum / dsc-document-classification-with-naive-bayes-lab Public

View license

0 stars 134 forks

Star Watch

Code Issues Pull requests Actions Projects Security Insights

solution

This branch is 5 commits ahead, 5 commits behind master.

sumedh10 update readme ... on Nov 14, 2019 6

View code

README.md

Document Classification with Naive Bayes - Lab

Introduction

In this lesson, you'll practice implementing the Naive Bayes algorithm on your own.

Objectives

In this lab you will:

- Implement document classification using Naive Bayes

Import the dataset

To start, import the dataset stored in the text file `'SMSSpamCollection'`.

```
# Import the data
import pandas as pd
df = pd.read_csv('SMSSpamCollection', sep='\t', names=['label', 'text'])
df.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Account for class imbalance

To help your algorithm perform more accurately, subset the dataset so that the two classes are of equal size. To do this, keep all of the instances of the minority class (spam) and subset examples of the majority class (ham) to an equal number of examples.

```
# Your code here
minority = df[df['label'] == 'spam']
undersampled_majority = df[df['label'] == 'ham'].sample(n=len(minority))
df2 = pd.concat([minority, undersampled_majority])
df2.label.value_counts()
```

```
ham     747
spam    747
```

```
Name: label, dtype: int64
```

```
p_classes = dict(df2['label'].value_counts(normalize=True))
p_classes
```

```
{'ham': 0.5, 'spam': 0.5}
```

```
df2.iloc[0]
```

```
label                                                 spam
text     Free entry in 2 a wkly comp to win FA Cup fina...
Name: 2, dtype: object
```

Train-test split

Now implement a train-test split on the dataset:

```
from sklearn.model_selection import train_test_split
X = df2['text']
y = df2['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=17)
train_df = pd.concat([X_train, y_train], axis=1)
test_df = pd.concat([X_test, y_test], axis=1)
```

Create the word frequency dictionary for each class

Create a word frequency dictionary for each class:

```
# Will be a nested dictionary of class_i : {word1:freq, word2:freq..., wordn:freq},.
class_word_freq = {}
classes = train_df['label'].unique()
for class_ in classes:
    temp_df = train_df[train_df['label'] == class_]
    bag = {}
    for row in temp_df.index:
        doc = temp_df['text'][row]
        for word in doc.split():
            bag[word] = bag.get(word, 0) + 1
    class_word_freq[class_] = bag
```

Count the total corpus words

Calculate V, the total number of words in the corpus:

```
vocabulary = set()
for text in train_df['text']:
    for word in text.split():
        vocabulary.add(word)
V = len(vocabulary)
V
```

```
5977
```

Create a bag of words function

Before implementing the entire Naive Bayes algorithm, create a helper function `bag_it()` to create a bag of words representation from a document's text.

```
def bag_it(doc):
    bag = {}
    for word in doc.split():
        bag[word] = bag.get(word, 0) + 1
    return bag
```

Implementing Naive Bayes

Now, implement a master function to build a naive Bayes classifier. Be sure to use the logarithmic probabilities to avoid underflow.

```
def classify_doc(doc, class_word_freq, p_classes, V, return_posteriors=False):
    bag = bag_it(doc)
    classes = []
    posteriors = []
    for class_ in class_word_freq.keys():
        p = np.log(p_classes[class_])
        for word in bag.keys():
            num = bag[word]+1
            denom = class_word_freq[class_].get(word, 0) + V
            p += np.log(num/denom)
        classes.append(class_)
        posteriors.append(p)
    if return_posteriors:
        print(posteriors)
    return classes[np.argmax(posteriors)]
```

Test your classifier

Finally, test your classifier and measure its accuracy. Don't be perturbed if your results are sub-par; industry use cases would require substantial additional preprocessing before implementing the algorithm in practice.

```
import numpy as np

y_hat_train = X_train.map(lambda x: classify_doc(x, class_word_freq, p_classes, V))
residuals = y_train == y_hat_train
residuals.value_counts(normalize=True)
```

```
False    0.754464
True     0.245536
dtype: float64
```

Level up (Optional)

Rework your code into an appropriate class structure so that you could easily implement the algorithm on any given dataset.

Summary

Well done! In this lab, you practiced implementing Naive Bayes for document classification!

Releases

No releases published

Packages

No packages published

Contributors 4

sumedh10 Sumedh Panchadhar

LoreDirick Lore Dirick

Matthew-Mitchell Matthew Mitchell

mathymitchell

Languages

● **Jupyter Notebook** 100.0%