Naive Bayes Classifier-Independent Work-2

By Rustamova Malakkhanim | 604.19E

Instructions

In statistics, naive Bayes classifiers are a family of simple "probabiltic classifiers" based on applying Bayes theorem with strong(naive) independence assumptions between the features. They are among the simplest Bayesian network models,

but coupled with kernel density estimation, they can achieve high accuracy levels. Naive Bayes classifiers are high scalable, requiring a number of parameters linear in the number of variables in a learning problem.

Today, I am going to solve a text classification problem with the help of scikit-learn and import some libraries for train-test splitting, MultinomialNB, data pre-processing.

```
from sklearn.naive_bayes import MultinomialNB
import pandas as pd
from sklearn.model_selection import train_test_split
import joblib
from sklearn.feature_extraction.text import CountVectorizer
```

Then, I import train data with the help of a pandas and do some modifications on it.

```
data=pd.read_excel(r"C:\Users\user\Desktop\exercise.xlsx")
```

```
data
```

	Text	Authority
0	For the Azerbaijani partners we have no restri...	Author 1
1	You can travel, to Europe for mobility.	Author 1
2	Try to gather as many colleagues as possible a...	Author 1
3	It looks promising in some parts of the world ...	Author 1
4	We are all looking forward to your visit and N...	Author 2
5	I know that some of you are arriving, some in ...	Author 2
6	One extremely important point: Each year we ha...	Author 2

```
data['Text'] = data['Text'].str.strip().str.lower()
```

```
data
```

	Text	Authority
0	for the azerbaijani partners we have no restri...	Author 1
1	you can travel, to europe for mobility.	Author 1
2	try to gather as many colleagues as possible a...	Author 1
3	it looks promising in some parts of the world ...	Author 1
4	we are all looking forward to your visit and n...	Author 2

I have changed text column to lower characters because I need good accuracy.

I take Y as target value(class) and X as features.

I split data into 2 parts, train and test data with the help of 'train_test_split'. Predicted y is the predicted values of our target value(class).

```
x = data['Text']
y = data['Authority']
```

```
x, x_test, y, y_test = train_test_split(x,y, stratify=y, test_size=0.25,
                                        random_state=42)
```

3

Then I use CountVectorizer which provides a powerful way to extract and represent features from our text data. It allows us to perform custom preprocessing, custom tokenization, eliminate stop words and limit vocabulary size. Also I have used fit function for model fitting.

```
vec = CountVectorizer(stop_words='english')
x = vec.fit_transform(x).toarray()
x_test = vec.transform(x_test).toarray()
```

```
clf.fit(x, y)
```

```
MultinomialNB()
```

And finally its score is 0.5, I think it is because of our data size when we increase the number of data it will be much more than 0.5.

```
clf.score(x_test, y_test)
```

```
0.5
```

Then I use predict function for predicting the author of text.

```
: clf.predict(vec.transform(['just a week before most of you will be setting off f
```

```
: array(['Author 2'], dtype='<U8')
```

It's probability will be 0.5