Certificate Course in Machine Learning using Python [6 Weeks]

Dashboard My courses Certificate Course in Machine Learning using Python [6 Weeks] Day 21

Text Classification Problem: Model Evaluation

Text Classification Problem: Model Evaluation

Attempt: 1

Text Classification Problem: Model Evaluation

Preparing confusion matrix of y_test and y_predict

```
from sklearn import metrics

metrics.confusion_matrix(y_test,y_predict)
```

Output:

```
array([[1188,    6],
       [  28,  171]]
```

	Predicted ham	**Predicted spam**
Actual ham	**1188**	**6**
Actual spam	**28**	**171**

By looking the confusion matrix, there is only 6 observation is predicted as spam but actually they are ham. But, there is 28 observations are predicted as ham but they are actually spam.

Print message text for the false positives (FP)

```
# ham incorrectly classified as spam

X_test[(y_predict=="spam") & (y_test=="ham")]
```

Print message text for the False Negatives (FN)

```
# spam incorrectly classified as ham

X_test[(y_predict=="ham") & (y_test=="spam")]
```

Model Improvement

Stop Words examples- the, is, have, has, would, in, an etc. These are the words which may not be playing any role with respect to classification as ham or spam. Therefore stop words should be removed from the features

Removing stop words from the dataset

```
from sklearn.feature_extraction.text import CountVectorizer

vect1=CountVectorizer(stop_words='english')

X_train_1=vect1.fit_transform(X_train)

X_train_1
```

Note: Now matrix size is reduced to 4179x7182

Specifying the ngram_range in CountVectorizor():

To use combination of two words in sparse matrix.

ngram_range: The lower and upper boundary of the range of n-values for different word n-grams or char n-grams to be extracted.

Example: The sentence ‘Good Job Done’ contains the 2-grams ‘Good Job’ and ‘Job Done’.

```
vect2=CountVectorizer(ngram_range=(1,2))

data_matrix=vect2.fit_transform(simple_text)

df=pd.DataFrame(data_matrix.toarray(),columns=vect2.get_feature_names())

print(df)
```

- **Ignore Terms/words that appear in more than 50% of documents/messages/texts**

```
vect3=CountVectorizer(max_df=0.50)

X_train_3=vect3.fit_transform(X_train)

X_train_3
```

- **Only keep words that appear in at least 2 documents/messages/text**

```
vect4=CountVectorizer(min_df=2)

X_train_4=vect4.fit_transform(X_train)

X_train_4
```

Applying these modifications at once

Training Data

```
vect_combined=CountVectorizer(stop_words='english', ngram_range=(1,2), min_df=2, max_df=0.5)

X_train_c=vect_combined.fit_transform(X_train)

X_train_c
```

Testing Data

```
X_test_c=vect_combined.transform(X_test)

X_test_c
```

Now Applying the MultinomialNB

```
from sklearn.naive_bayes import MultinomialNB

nb=MultinomialNB()

nb.fit(X_train_c,y_train)

y_predict=nb.predict(X_test_c)
```

Printing accuracy

```
from sklearn import metrics

metrics.accuracy_score(y_test,y_predict)
```

Printing Confusion Matrix

```
metrics.confusion_matrix(y_test,y_predict)
```

Next

PREVIOUS ACTIVITY

◀ Text Data: sms.txt

Jump to...

NEXT ACTIVITY

Text Classification: Fine Tuning: Python File ▶

🌐 http://nielit.gov.in/gorakhpur/

✉ abhinav@nielit.gov.in or ajay.verma@nielit.gov.in