2019120006_DA_6

November 22, 2022

1 Experiment 6: Classification

1.0.1 Name: Jayesh Bane

1.0.2 UID: 2019120006

1.0.3 Class: BE EXTC

In this experiment we are using a spam email dataset for classfication purposes. We will be implementing 3 models namely; Naive Bayes, CART and Random Forest.

The dataset contains two columns: - Text - Label

[ ]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score ,f1_score , precision_score ,␣
↪recall_score
```

Importing the necessary libraries for the experiment.

[ ]:
```
df = pd.read_csv('spam.csv',encoding='ISO-8859-1')
df = df.iloc[:,:2]
df.rename(columns={'v1':'label', 'v2' : 'email'},inplace=True)
```

Reading the data from CSV file.

[ ]:
```
df['label'] = df.label.map({'ham':0 , 'spam':1})
df.head()
```

[ ]:

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

Outputting the read dataframe.

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(df['sms'], df['label'],␣
↪test_size=0.33, random_state=42)
```

Splitting the data into testing and training sets.

[ ]:
```
count_vector = CountVectorizer()
train_data = count_vector.fit_transform(X_train)
testing_data = count_vector.transform(X_test)
```

Using count vectorizer to count the frequencies of each word in the training and test data.

[ ]:
```
count_vector = CountVectorizer()
col_name = count_vector.fit(df['sms']).get_feature_names_out()
data = count_vector.transform(list(df['sms'])).toarray()
BOW = pd.DataFrame(data, columns= col_name)
BOW.head()
```

[ ]:

	00	000	000pes	008704050406	0089	0121	01223585236	01223585334	\
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	

	0125698789	02	…	ó_	û_	û_thanks	ûªm	ûªt	ûªve	ûï	ûïharry	ûò	\
0	0	0	…	0	0	0	0	0	0	0	0	0	
1	0	0	…	0	0	0	0	0	0	0	0	0	
2	0	0	…	0	0	0	0	0	0	0	0	0	
3	0	0	…	0	0	0	0	0	0	0	0	0	
4	0	0	…	0	0	0	0	0	0	0	0	0	

	ûówell
0	0
1	0
2	0
3	0
4	0

[5 rows x 8672 columns]

[ ]:
```
naive_bayes = MultinomialNB()
naive_bayes.fit(train_data, y_train)
```

[ ]:
```
MultinomialNB()
```

Creating a Naive Bayes Classifier.

[ ]:
```
predection = naive_bayes.predict(testing_data)
```

Training the Naive Bayes Classifier with the testing data.

```
[ ]: print('Accuracy score: {}'.format(accuracy_score(y_test, predection)))
     print('precision_score: {}'.format(precision_score(y_test, predection)))
     print('recall_score: {}'.format(recall_score(y_test, predection)))
     print('f1_score: {}'.format(f1_score(y_test, predection)))
```

```
Accuracy score: 0.9825992387166939
precision_score: 0.9741379310344828
recall_score: 0.8968253968253969
f1_score: 0.9338842975206612
```

Outputting the necessary metrics of the model.

1.0.4 CART Classifier

```
[ ]: from sklearn.tree import DecisionTreeClassifier
```

```
[ ]: cart = DecisionTreeClassifier()
     cart.fit(train_data, y_train)
```

```
[ ]: DecisionTreeClassifier()
```

```
[ ]: cart_prediction = cart.predict(testing_data)
```

```
[ ]: print('Accuracy score: {}'.format(accuracy_score(y_test, cart_prediction)))
     print('precision_score: {}'.format(precision_score(y_test, cart_prediction)))
     print('recall_score: {}'.format(recall_score(y_test, cart_prediction)))
     print('f1_score: {}'.format(f1_score(y_test, cart_prediction)))
```

```
Accuracy score: 0.957041870581838
precision_score: 0.8392156862745098
recall_score: 0.8492063492063492
f1_score: 0.8441814595660749
```

1.0.5 Random Forest

```
[ ]: from sklearn.ensemble import RandomForestClassifier
```

```
[ ]: rf = RandomForestClassifier()
```

```
[ ]: rf.fit(train_data, y_train)
```

```
[ ]: RandomForestClassifier()
```

```
[ ]: rf_prediction = rf.predict(testing_data)
```

```
[ ]: print('Accuracy score: {}'.format(accuracy_score(y_test, rf_prediction)))
     print('precision_score: {}'.format(precision_score(y_test, rf_prediction)))
     print('recall_score: {}'.format(recall_score(y_test, rf_prediction)))
     print('f1_score: {}'.format(f1_score(y_test, rf_prediction)))
```

```
Accuracy score: 0.9738988580750407
precision_score: 1.0
recall_score: 0.8095238095238095
f1_score: 0.8947368421052632
```

1.0.6 Conclusion

- Implemented 3 models for Email Spam Classification namely; Naive Bayes, Decision Tree (CART) and Random Forest.
- We can clearly see that the Naive Bayes classifier performs the best out of all three, having the highest scores in all 4 metrics present.
- We have used a Multinomial Naive Bayes classifier as it works best for discrete features, which in our case is the word count.