DATA MINING TECHNOLOGY FOR BUSINESS AND SOCIETY HW3

Cecilia Martinez Oliva, Giorgio Giannone

Contents

Introduction

In this work we have to solve a classification problem using different models and comparing the results.

1. First Part

In this part we have to classify comments as spam or ham.

1.1 Train/Test

	Training Set	**Test Set**
Spam(0)	122	53
Ham(1)	120	52

1.2 The list of all parameters to tune with the corresponding set of values.

TfidfVectorizer:

```
'ngram_range' : [(1,1),(1,2)]
'tokenizer'   : [None, stemming_tokenizer]
'stop_words'  : [None, 'english']
```

KNeighborsClassifier:

```
'n_neighbors' : [1,3,5,7,9,11]
'leaf_size' : [2,3]
```

```
(we use a short list for leaf_size because the optimal value is always small and
less dominant respect the other parameters.)
```

1.3 A simply and short description on how to perform the training-validation process using more than one CPU core.

To optimize the 10-fold-cross-validation step we can use the flag -1 for the parameter n_jobs in the function GridSearchCV: in this way we use all the available cores in the CPU for our jobs and we reduce the total time for the model tuning.

1.4 The best configuration of parameters found by the GridSearchCV object at the end of the 10-Fold-Cross-Validation process.

TfidfVectorizer:

```
'ngram_range': (1, 2)
'tokenizer': None
'stop_words': None
```

KNeighborsClassifier:

```
'n_neighbors': 11
'leaf_size': 2
```

Usually the model works better with stemming and removing the stopwords. In this particular case, the outcomes are better when we do not preprocess the words: we think that this happens because the comments are extremely short (normally < 10 words) and, instead to remove noise, we lose information: it seems that stopwords and inflections help the classification task, probably capturing some semantic inside the short texts.

1.5 The output of the metrics.classification_report tool.

```
---------------------------------------------------------
             precision    recall  f1-score   support

          0       0.92      0.85      0.88        53
          1       0.86      0.92      0.89        52

avg / total       0.89      0.89      0.89       105

---------------------------------------------------------
```

1.6 The Confusion-Matrix with labels on columns and rows.

	Predicted – Spam	**Predicted – Ham**
Tested - Spam	45	8
Tested – Ham	4	48

1.7 The Normalized-Accuracy value.

```
0.8857
```

1.8 The Matthews-Correlation-Coefficient value.

```
0.7738
```

2. Second-Part

In this part we have to classify comments as positive or negative.

We decided to use a Support Vector Machine with exponential kernel and a Stochastic Gradient Descent algorithm with linear SVM as classifier (they have similar names but they are two distinct classifier with different hyperparameters and different fitting procedure) to compare the results with the KNN classifier.

2.1 Train/Test

	Training Set	**Test Set**
Negative(0)	308	308
Positive(1)	249	250

2.2 The list of all parameters to tune with the corresponding set of values.

TfidfVectorizer:

```
'ngram_range' : [(1,1),(1,2)]
'tokenizer'   : [None, stemming_tokenizer]
'stop_words'  : [None, 'english']
```

Neigh---

KNeighborsClassifier:

```
'n_neighbors' : [1,3,5,7,9,11]
'leaf_size' : [2,3]
```

SVM_RBF---

SVC:

```
'C' : [0.1, 0.5, 1, 5, 10, 50, 100]
'gamma' : [0.1, 0.5, 1, 3, 6, 10]
```

SGD---

SGDClassifier:

```
'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10]
'penalty': ['l1', 'l2']
```

2.3 The best configuration of parameters found by the GridSearchCV object at the end of the 10-Fold-Cross-Validation process.

Neigh---

TfidfVectorizer:
```
'ngram_range': (1, 2)
'stop_words': 'english'
'tokenizer': stemming_tokenizer
```

KNeighborsClassifier :
```
'leaf_size': 2
'n_neighbors': 3
```

SVM_RBF--

TfidfVectorizer:
```
'ngram_range': (1, 1)
'stop_words': 'english'
'tokenizer': stemming_tokenizer
```

SVC:
```
'C': 50
'gamma': 0.1
```

SGD--

TfidfVectorizer:
```
'ngram_range': (1, 1)
'stop_words': 'english'
'tokenizer': stemming_tokenizer
```

SGDClassifier:
```
'alpha': 0.001
'penalty': 'l1'
```

2.4 The output of the metrics.classification_report tool.

Neigh---
```
---------------------------------------------------------
             precision    recall  f1-score   support

          0       0.94      0.83      0.88       250
          1       0.88      0.96      0.91       308

avg / total       0.90      0.90      0.90       558

---------------------------------------------------------
```

SVM_RBF--

```
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.96      0.96      0.96       250
          1       0.96      0.97      0.97       308

avg / total       0.96      0.96      0.96       558

-----------------------------------------------------
```

SGD--

```
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.97      0.98      0.97       250
          1       0.98      0.97      0.98       308

avg / total       0.98      0.98      0.98       558

-----------------------------------------------------
```

2.5 The Confusion-Matrix with labels on columns and rows.

Neigh--

	Predicted – Negative	**Predicted – Positive**
Tested - Negative	208	42
Tested – Positive	13	295

SVM_RBF--

	Predicted – Negative	**Predicted – Positive**
Tested - Negative	239	11
Tested – Positive	9	299

SGD---

	Predicted – Negative	**Predicted – Positive**
Tested - Negative	245	5
Tested – Positive	8	300

2.6 The Normalized-Accuracy value.

Neigh--

0.9014

SVM_RBF--

0.9641

SGD---

0.9767

2.7 The Matthews-Correlation-Coefficient value.

Neigh--

```
0.8030
```

SVM_RBF--

```
0.9275
```

SGD---

```
0.9530
```