Homework 3

Data Mining Technology for Business and Society

Gabriel Nespoli, Giusy Rose Tiro, Sergio Ballesteros

Part I

1. Data:

	Training set	Test set
Ham	120	52
Spam	122	53

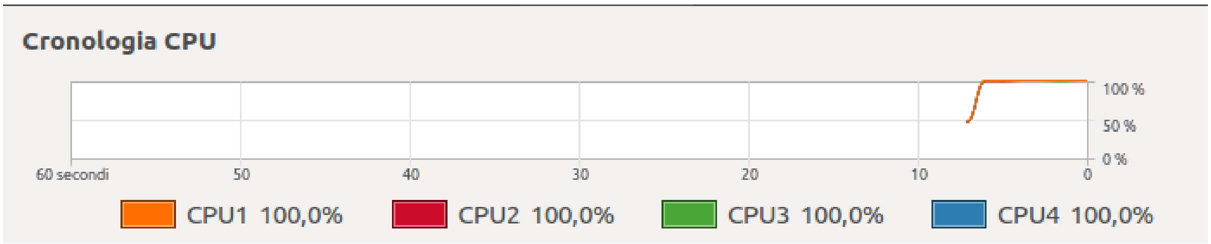
2. Parameter tested

It was tested the following parameters and the respective values for the KNN classifier:

- Number of neighbors to use by default for *k_neighbors* queries: **5**, **10** and **15**
- Weight function used in prediction: **uniform** and **distance**
- Algorithm used to compute the nearest neighbors: **auto**, **ball_tree** and **kd_tree**
- Leaf size: **30** and **40**
- Power parameter for the Minkowski metric: **1** and **2**
- Distance metric to use for the tree: **minkowski** and **euclidean**
- Tokenizer: **None** and **stemming_tokenizer**
- The lower and upper boundary of the range of n-values for different n-grams to be extracted: **(1, 1)** and **(1, 2)**

3. Description on how to performance of the training-validation process using more than one CPU core:

It was also set the number of jobs running in parallel to the number of CPUs (the machine used has 4 CPUs), setting the parameter *n_jobs*=-1 in the ***GrideSearchCV***.



4. Best parameters:

After a long process, the best parameters were:

n_neighbors	10
weights	distance
algorithm	auto
leaf_size	30
p	2
metric	minkowski
ngram_range	(1, 2)
tokenizer	None

5. Output of the metrics.classification_report tool

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

        Ham       0.83      0.96      0.89        52
       Spam       0.96      0.81      0.88        53

avg / total       0.90      0.89      0.89       105

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

6. Confusion-Matrix

		Predicted	
		Ham	**Spam**
Actual	**Ham**	50	2
	Spam	10	43

7. Normalized-Accuracy value: `0.89`

8. Matthews-Correlation-Coefficient value: `0.78`

Part II

1. Data:

	Training set	Test set
Positive reviews	308	308
Negative reviews	249	250

KNN Classifier

2. Parameter tested:

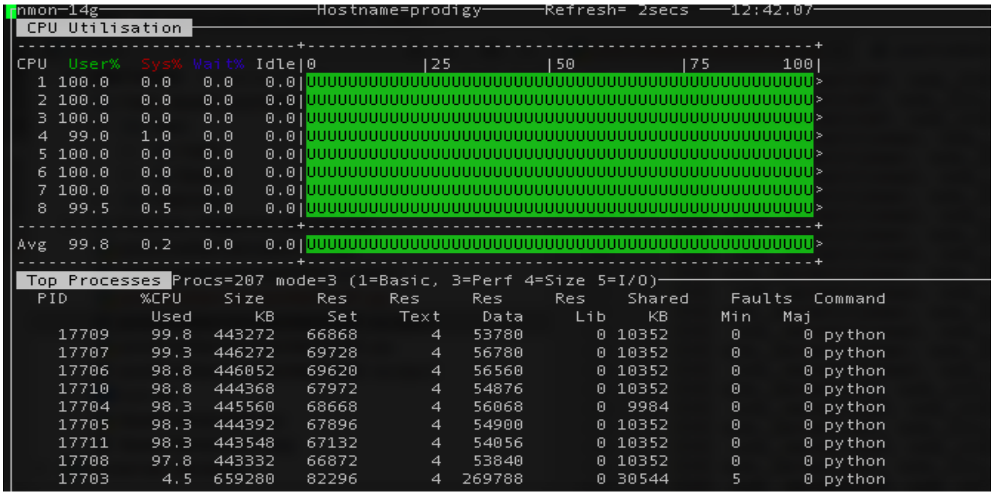
- Number of neighbors to use by default for *k_neighbors* queries: **5**, **10** and **15**
- Weight function used in prediction: **uniform** and **distance**
- Algorithm used to compute the nearest neighbors: **auto**, **ball_tree** and **kd_tree**
- Leaf size: **30** and **40**
- Power parameter for the Minkowski metric: **1** and **2**
- Distance metric to use for the tree: **minkowski** and **Euclidean**
- Tokenizer: **None** and **stemming_tokenizer**
- The lower and upper boundary of the range of n-values for different n-grams to be extracted: **(1, 1)** and **(1, 2)**

3. Description on how to performance of the training-validation process using more than one CPU core:

In the case of the SVM method only the CV can be parallelized. Instead, when dealing with decision trees, like random forest, we can parallelize the CV, but also for each CV the program can run different decision trees inside one CV; the same can be said for the KNN method, where we can run parallel jobs for neighbours search.

But in the code we only use the parallelization in the CV, and not inside the classifiers. This is due the CV already uses all the cores, and adding parallelization to the classifiers did not add any advantage, but in fact made the program even slower.

Here in the picture we show again that the program is using the 8 threads on another machine.



```
nmon─14g──────────────────Hostname=prodigy──────Refresh= 2secs ───12:42.07──
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  2 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  3 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  4  99.0   1.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  5 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  6 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  7 100.0   0.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
  8  99.5   0.5   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
---------------------------+-------------------------------------------------+
Avg  99.8   0.2   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU>
---------------------------+-------------------------------------------------+
 Top Processes Procs=207 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)───────────────
  PID       %CPU   Size    Res    Res    Res     Res   Shared   Faults  Command
             Used     KB    Set   Text   Data     Lib     KB    Min   Maj
    17709    99.8 443272  66868      4  53780       0 10352     0     0 python
    17707    99.3 446272  69728      4  56780       0 10352     0     0 python
    17706    98.8 446052  69620      4  56560       0 10352     0     0 python
    17710    98.8 444368  67972      4  54876       0 10352     0     0 python
    17704    98.3 445560  68668      4  56068       0  9984     0     0 python
    17705    98.3 444392  67896      4  54900       0 10352     0     0 python
    17711    98.3 443548  67132      4  54056       0 10352     0     0 python
    17708    97.8 443332  66872      4  53840       0 10352     0     0 python
    17703     4.5 659280  82296      4 269788       0 30544     5     0 python
```

4. Best parameters:

n_neighbors	5
weights	distance
algorithm	auto
leaf_size	30
p	1
metric	minkowski
ngram_range	(1, 1)
tokenizer	function stemming_tokenizer

5. Output of the metrics.classification_report tool

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

 Positive       0.80      0.96      0.87       308
 negative       0.94      0.70      0.80       250

avg / total       0.86      0.84      0.84       558

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

6. Confusion-Matrix

		Predicted	
		Ham	**Spam**
Actual	**Ham**	296	12
	Spam	75	175

7. Normalized-Accuracy value: `0.84`

8. Matthews-Correlation-Coefficient value: `0.70`

SVM classifier

1. Parameter tested:

- Kernel type to be used in the algorithm: linear, rbf, poly and sigmoid
- Degree of the polynomial kernel function (‘poly’): 2 and 3
- Independent term in kernel function (coef0): 0 and 1
- Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’: 1e-2, 1e-3 and auto,
- Penalty parameter C of the error term: 1,5 and 10
- Tokenizer: **None** and **stemming_tokenizer**
- The lower and upper boundary of the range of n-values for different n-grams to be extracted: **(1, 1)** and **(1, 2)**

2. Best parameters:

C	1
coef0	0.0
degree	2
gamma	0.01
kernel	linear
ngram_range	(1, 2)
tokenizer	function stemming_tokenizer

3. Output of the metrics.classification_report tool:

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

 Positive       0.95       0.99       0.97       308
 negative       0.98       0.94       0.96       250

avg / total      0.97       0.97       0.97       558

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

4. Confusion-Matrix

		Predicted	
		Ham	**Spam**
Actual	**Ham**	304	4
	Spam	15	235

5. Normalized-Accuracy value: `0.97`

6. Matthews-Correlation-Coefficient value: `0.93`

Random Forest Classifier Method

1. Parameter tested:

- Number of trees in the forest:100 and 800,
- Criterion function to measure the quality of a split:gini and entropy,
- The number of features to consider when looking for the best split: sqrt and log2
- The maximum depth of the tree :10,50 and None, where None means that nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- The minimum number of samples required to split an internal node: 2 and 10
- Tokenizer: **None** and **stemming_tokenizer**
- The lower and upper boundary of the range of n-values for different n-grams to be extracted: **(1, 1)** and **(1, 2)**

2. Best parameters：

- ***n_estimators*** 800
- ***criterion*** entropy
- ***max_features*** sqrt
- ***max_depth*** 50
- ***min_samples_split*** 10
- ***ngram_range*** (1, 1)
- ***tokenizer*** function stemming_tokenizer

3. Output of the metrics.classification_report tool

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

 Positive       0.93      0.97      0.95       308
 negative       0.96      0.92      0.94       250

avg / total       0.95      0.95      0.95       558

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

4.Confusion-Matrix

		Predicted	
		Ham	**Spam**
Actual	**Ham**	299	9
	Spam	21	229

5. Normalized-Accuracy value: 0.95

6. Matthews-Correlation-Coefficient value: 0.89