Learning from Data – Week 2
Assignment 2: Decision Tree, K-NN

General remarks

This assignment is meant to get your further acquainted with scikit-learn and in particular the principles of the algorithms we have discussed. You will also explore a little further the contribution of features and varying parameters in running classification experiments. In the discussion of the results bear in mind the theoretical points we have touched on in class.

Specifically, for the practical parts, you are asked to train and test a few models using different algorithms on the same kind of data that we used for Assignment 1, but **only considering the six-way classification**. Exploiting coding you have done for Assignment 1 regarding measures, you will have to discuss how the different models perform, and why.

What you have to do:

- upload on Nestor the code with your best model (see Exercise 2.3).
  Important: You have to assume that we will run it using arguments like this:
  `LFDassignment2.py <trainset> <testset>`
  where trainset and testset have the same format. You are given the trainset, which is the same dataset as last week, but we're holding out the test set.

- fill in a Google form where you will answer a few questions: `https://forms.gle/qxA4msjr5CEibjW47`. When filling this in, please refer to this pdf for more details on the questions, instructions and clarifications.

Deadline: 21 September, 11:59pm.

This assignment is to be completed **individually**.

Data

For this assignment, we will be using the collection of reviews we used for Assignment 1. The labels we are interested in are the six denoting the topics: `books, camera, dvd, health, music, software`. You will find the data on Nestor.

Exercise 2.1 – Decision Tree

2.1.1 Theory

colour	size	shape	edible
yellow	small	round	yes
yellow	small	round	no
green	small	irregular	yes
green	large	irregular	no
yellow	large	round	yes
yellow	small	round	yes
yellow	small	round	yes
yellow	small	round	yes
green	small	round	no
yellow	large	round	no
yellow	large	round	yes
yellow	large	round	no
yellow	large	round	no
yellow	large	round	no
yellow	small	irregular	yes
yellow	large	irregular	yes

Consider the dataset on the left. Given that the last column is the class you want to predict, what is the tree that best represents this data? You have to figure it out without running it through scikit-learn! It's actually a good idea to draw it by hand, thinking what the first branch will be, and why. If you want to show calculations you can add them in the picture.

2.1.2 Practice

Scikit-learn provides an implementation of a decision tree:

```
1  from sklearn.tree import DecisionTreeClassifier
```

scikit-learn.org/stable/modules/tree.html

Pruning isn't supported in scikit-learn, but you can set several parameters of the algorithm, such as tree size, minimal number of samples for a split, and so on (also check the slides for info on this). Please, run the relevant experiments and answer the questions about pruning and about changing parameters in the Google form.

Exercise 2.2 – K-Nearest Neighbor

Scikit-learn also provides an implementation of the K-nearest neighbor algorithm:

```
1  from sklearn.neighbors import KNeighborsClassifier
```

scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

Run a K-NN classifier on the data performing a six-way classification. Experiment with different values of K seeing what happens on your development set. Plot accuracy and f-scores on a graph, you can upload this on the Google form under question KNN-1. The following questions in the form also have to do with varying K and varying performance.

Exercise 2.3 – Comparisons

2.3.1 Running times

Check the time it takes to train and test the model, and compare it with what you observed for the Decision Tree (see above) and Naive Bayes. On the form, there is a question where you have to explain why the algorithms have different run times at train and test time.

A way to check time (but you can choose whatever you prefer):

```
1  import time
2
3  t0 = time.time()
4  classifier.fit(X, y)
5  train_time = time.time() - t0
6  print("training time: ", train_time)
```

You should do the same at test time, and for both models, to compare.

2.3.2 Best Model

We ask you to submit a script that will run your "best model". Your best model will be run on held out data, i.e. data you haven't trained nor developed on. Now, how can you make a better model? Say that you're working with a decision tree, you can modify parameter values such as those mentioned above. If you're working with K-NN, you can choose an optimal K. How do you *tune* such parameters? How do you choose the *optimal* K? By cross-validating your training set or testing on your development set! Also, you can still decide you'd rather use Naive Bayes, maybe with a selection of features (see below)?

Another way to improve your model is to get a little bit more sophisticated on the features you're using. The Assignment 1 script in its original form takes all words as features. Do you want to refine this? You can for example use a stop list to rule out certain uninformative words (already using tfidf takes partially care of this). You might also want to perform some normalisation. For example, how about turning all numbers into just some uniform placeholder? Or maybe lemmatisation? Or stemming? Remember that NLTK provides nice functionalities for basic language (pre)processing.

Still regarding features, we know that *feature selection* can help. There are feature selection modules in scikit-learn, although it might not be so straightforward (yet) to use them. You can obviously try! You can check the whole suite of feature_selection methods available in sciki-learn at `scikit-learn.org/stable/modules/feature_selection.html`.

It's entirely up to you to choose which model you want us to run on test data. Please, upload on Nestor the script that runs your best model, and provide a brief summary of your choices (algorithm, parameters, features) in the Google form, where asked. Please, remember that we will run your script calling both training and test sets as arguments, as specified above. You can check that everything works by splitting the data and saving it in different files (a train file and a test file). Important: we also expect your script to output the standard evaluation measures you have already used for Assignment 1.