Using several classifiers and tuning parameters - Parameters grid

From official `scikit-learn` documentation (http://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html)

Adapted by Claudio Sartori

Example of usage of the ***model selection*** features of `scikit-learn` and comparison of several classification methods.

1. import a sample dataset
2. split the dataset into two parts: train and test
   - the *train* part will be used for training and validation (i.e. for *development*)
   - the *test* part will be used for test (i.e. for *evaluation*)
   - the fraction of test data will be *ts* (a value of your choice between 0.2 and 0.5)
3. `GridSearchCV` (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) performs multiple cross-validation experiments to train and test a model with different combinations of parameter values

   - for each parameter we set a list of values to test, the `fit` method will generate every possible combination, fit a model with it and evaluate its performance
   - we choose a *score function* which will be used for the optimization
     - e.g. `accuracy_score`, `precision_score`, `cohen_kappa_score`, `f1_score`, see this (https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter) and this (http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics) for reference
   - the output is a dictionary that contains
     - the set of parameters which maximize the score
     - the test scores
4. prepare the parameters for the grid
   - it is a list of dictionaries
5. set the parameters by cross validation and the *score functions* to choose from
6. Loop on scores and, for each score, loop on the model labels (see details below)

In [1]:
```
"""
http://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html
@author: scikit-learn.org and Claudio Sartori
"""
import warnings
warnings.filterwarnings('ignore') # uncomment this line to suppress warnings

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
print(__doc__) # print information included in the triple quotes at the beginning

# Loading a standard dataset
#dataset = datasets.load_digits()
# dataset = datasets.fetch_olivetti_faces() # 40 classes!
# dataset = datasets.fetch_covtype()        # 581012 examples    54 features
# dataset = datasets.load_iris()    # 150 examples -- 4 features -- 3 classes
dataset = datasets.load_wine()      # 178 examples -- 13 features -- 3 classes
# dataset = datasets.load_breast_cancer() # binary
```

```
http://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html
@author: scikit-learn.org and Claudio Sartori
```

Prepare the environment

The `dataset` module contains, among others, a few sample datasets.

See this page (http://scikit-learn.org/stable/datasets/index.html) for reference

In the following:

- Load a dataset using the `dataset` module (output refers to the wine dataset)
- Prepare the data and the target in X and y.

In [2]:
```

```

```
    178 examples -- 13 features -- 3 classes
```

Train-test split:

- Set the test set size `ts` .
- Set the random state to 44.
- Split the dataset into the train and test parts

In [3]:
```
Training on 124 examples
```

Try GridSearchCV with a DecisionTreeClassifier

Use `GridSearchCV` to get the best `max_depth` value for a `DecisionTreeClassifier` evaluating accuracy:

- Define the parameters to be tested and the range of values for each one
- Get a `GridSearchCV` instance for a `DecisionTreeClassifier`
- Fit the instance to the training data

It's ok to get results that are different than the output

In [4]:
```
Best parameters: {'max_depth': 15}
```

The function below groups all the outputs

Write a `print_results` function that takes a fitted model and uses its attributes to inspect the results of the search with the parameter grid.

The attributes are:
`model.best_params_`
`model.cv_results_['mean_test_score']`
`model.cv_results_['std_test_score']`
`model.cv_results_['params']`

The report is generated by the `classification_report` function imported from `sklearn.metrics` , which takes as argument the true test labels and the predicted test labels.

The +/- in the results is obtained doubling the `std_test_score` . Mean and standard test scores are computed considering the various results on the cross-validation chunks.

The function will be used to print the results for each set of parameters in the last part of the exercise.

Use `print_results` to show the result of the tuning above.

In [5]:

```
Best parameters set found on train set:

{'max_depth': 15}

Grid scores on train set:

0.580 (+/-0.203) for {'max_depth': 1}
0.822 (+/-0.110) for {'max_depth': 2}
0.847 (+/-0.058) for {'max_depth': 3}
0.847 (+/-0.127) for {'max_depth': 4}
0.855 (+/-0.094) for {'max_depth': 5}
0.855 (+/-0.107) for {'max_depth': 6}
0.863 (+/-0.080) for {'max_depth': 7}
0.847 (+/-0.076) for {'max_depth': 8}
0.823 (+/-0.106) for {'max_depth': 9}
0.855 (+/-0.094) for {'max_depth': 10}
0.847 (+/-0.076) for {'max_depth': 11}
0.847 (+/-0.116) for {'max_depth': 12}
0.831 (+/-0.057) for {'max_depth': 13}
0.807 (+/-0.126) for {'max_depth': 14}
0.871 (+/-0.092) for {'max_depth': 15}
0.847 (+/-0.091) for {'max_depth': 16}
0.863 (+/-0.094) for {'max_depth': 17}
0.855 (+/-0.129) for {'max_depth': 18}
0.831 (+/-0.104) for {'max_depth': 19}

Detailed classification report for the best parameter set:

The model is trained on the full train set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           0       0.90      0.95      0.93        20
           1       0.95      0.86      0.90        22
           2       0.92      1.00      0.96        12

    accuracy                           0.93        54
   macro avg       0.93      0.94      0.93        54
weighted avg       0.93      0.93      0.93        54
```

Loop on scores and, for each score, loop on the model labels

- iterate varying the score function
  1. iterate varying the classification model among Decision Tree, Naive Bayes, Linear Perceptron, Support Vector
     - activate the *grid search*
       A. the resulting model will be the best one according to the current score function
     - print the best parameter set and the results for each set of parameters using the above defined function
     - print the classification report
     - store the `.best score_` in a dictionary for a final report
  2. print the final report for the current *score funtion*

In [6]:

```
# The code below is intended to ease the remainder of the exercise

model_lbls = [
              'dt',
              'nb',
#               'lp',
#               'svc',
#               'knn',
             ]

# Set the parameters to be explored by the grid for each classifier
tuned_param_dt = [{'max_depth': list(range(1,20))}]
tuned_param_nb = [{'var_smoothing': [10, 1, 1e-1, 1e-2, 1e-3, 1e-4,
1e-5, 1e-6, 1e-07, 1e-8, 1e-9, 1e-10]}]
tuned_param_lp = [{'early_stopping': [True]}]
tuned_param_svc = [{'kernel': ['rbf'],
                    'gamma': [1e-3, 1e-4],
                    'C': [1, 10, 100, 1000],
                    },
                    {'kernel': ['linear'],
                     'C': [1, 10, 100, 1000],
                    },
                   ]
tuned_param_knn =[{'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]

# set the models to be fitted specifying name, estimator and parame
ter structure
models = {
    'dt': {'name': 'Decision Tree       ',
           'estimator': DecisionTreeClassifier(),
           'param': tuned_param_dt,
          },
    'nb': {'name': 'Gaussian Naive Bayes',
           'estimator': GaussianNB(),
           'param': tuned_param_nb
          },
    'lp': {'name': 'Linear Perceptron   ',
           'estimator': Perceptron(),
           'param': tuned_param_lp,
          },
    'svc':{'name': 'Support Vector      ',
           'estimator': SVC(),
           'param': tuned_param_svc
          },
    'knn':{'name': 'K Nearest Neighbor ',
           'estimator': KNeighborsClassifier(),
           'param': tuned_param_knn

    }
}

# scores to be explored
scores = [
          'precision',
#           'recall',
         ]
```

In [7]:

```
==========================================
# Tuning hyper-parameters for precision

------------------------------------------
Trying model Decision Tree
Best parameters set found on train set:

{'max_depth': 12}

Grid scores on train set:

0.419 (+/-0.141) for {'max_depth': 1}
0.837 (+/-0.097) for {'max_depth': 2}
0.868 (+/-0.040) for {'max_depth': 3}
0.867 (+/-0.072) for {'max_depth': 4}
0.871 (+/-0.069) for {'max_depth': 5}
0.863 (+/-0.101) for {'max_depth': 6}
0.855 (+/-0.051) for {'max_depth': 7}
0.853 (+/-0.080) for {'max_depth': 8}
0.876 (+/-0.124) for {'max_depth': 9}
0.886 (+/-0.069) for {'max_depth': 10}
0.887 (+/-0.067) for {'max_depth': 11}
0.888 (+/-0.064) for {'max_depth': 12}
0.851 (+/-0.109) for {'max_depth': 13}
0.845 (+/-0.118) for {'max_depth': 14}
0.875 (+/-0.080) for {'max_depth': 15}
0.881 (+/-0.056) for {'max_depth': 16}
0.863 (+/-0.083) for {'max_depth': 17}
0.861 (+/-0.117) for {'max_depth': 18}
0.867 (+/-0.123) for {'max_depth': 19}

Detailed classification report for the best parameter set:

The model is trained on the full train set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           0       0.90      0.95      0.93        20
           1       0.95      0.91      0.93        22
           2       1.00      1.00      1.00        12

    accuracy                           0.94        54
   macro avg       0.95      0.95      0.95        54
weighted avg       0.95      0.94      0.94        54


------------------------------------------
Trying model Gaussian Naive Bayes
Best parameters set found on train set:

{'var_smoothing': 1e-06}

Grid scores on train set:
```

```
0.481 (+/-0.010) for {'var_smoothing': 10}
0.663 (+/-0.296) for {'var_smoothing': 1}
0.754 (+/-0.163) for {'var_smoothing': 0.1}
0.743 (+/-0.190) for {'var_smoothing': 0.01}
0.755 (+/-0.205) for {'var_smoothing': 0.001}
0.878 (+/-0.196) for {'var_smoothing': 0.0001}
0.949 (+/-0.103) for {'var_smoothing': 1e-05}
0.972 (+/-0.080) for {'var_smoothing': 1e-06}
0.958 (+/-0.083) for {'var_smoothing': 1e-07}
0.958 (+/-0.083) for {'var_smoothing': 1e-08}
0.967 (+/-0.062) for {'var_smoothing': 1e-09}
0.967 (+/-0.062) for {'var_smoothing': 1e-10}

Detailed classification report for the best parameter set:

The model is trained on the full train set.
The scores are computed on the full test set.

              precision    recall  f1-score   support

           0       0.95      0.95      0.95        20
           1       0.95      0.91      0.93        22
           2       0.92      1.00      0.96        12

    accuracy                           0.94        54
   macro avg       0.94      0.95      0.95        54
weighted avg       0.94      0.94      0.94        54


Summary of results for precision
Estimator
Decision Tree            - score: 88.77%
Gaussian Naive Bayes     - score: 97.21%
```