Train-Test Split

Introduction

In this exercise we will work with the concept of splitting datasets into training and test sets ad dig deeper into the implementations of said concepts, to get a better understading of what is happening in the background of machine learning.

Setup the Housing data form §2 [HOML]

We use the housing data from the book, this cell will set everything up for you...

In [10]:

```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline

import matplotlib
import matplotlib.pyplot as plt
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

# Where to save the figures
PROJECT_ROOT_DIR = ".."
CHAPTER_ID = "end_to_end_project"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    #path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("IGNORING: Saving figure", fig_id)
    #if tight_layout:
    #    plt.tight_layout()
    #plt.savefig(path, format=fig_extension, dpi=resolution)

# Ignore useless warnings (see SciPy issue #5998)
import warnings
warnings.filterwarnings(action="ignore", message="^internal gelsd")

import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("../datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "../datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
        os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()

fetch_housing_data()

import pandas as pd
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

```
housing = load_housing_data()

#housing.head()
print("housing.shape=",housing.shape,"\n")
housing.info()

%matplotlib inline
import matplotlib.pyplot as plt
#housing.hist(bins=50, figsize=(20,15))
#save_fig("attribute_histogram_plots")
#plt.show()

# NOTE: ITMAL, convert Pandas dataframe to numpy array, i.e. matrix
#       and use H later instead of housing
H = housing.values
print('H.shape=',H.shape,", type(H)=",type(H))

print('OK')
```

```
housing.shape= (20640, 10)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
H.shape= (20640, 10) , type(H)= <class 'numpy.ndarray'>
OK
```

Create our own train-test split function

Qa Create Your Own Split Function

create your own split function, that can do the data shuffling (as it is now) or do a simpler split without shuffling.

Notice that it would be better to name the function `my_split_train_test` to avoid clashing problems later with the Scikit-learn function of the same name. The `test_ratio` parameter has also been renamed to `test_size`.

Also note that the split function in [HOML] operates on Pandas data frames, and this will give us a mixup problem later, when we pass the function numpy arrays (matrices).

Test that your new split function returns the same number of train and test data no matter if shuffleling is on or off, using the test stub below.

Qa Implementation

In [3]:

```
# TODO: Qa...define your my_split_train_test here

def my_split_train_test(data, test_size, shuffle=False):

    if shuffle == True:
        indices = np.random.permutation(len(data))

    else:
        indices = np.arange(len(data))


    test_set_size = int(len(data) * test_size)
    test_indices = indices[len(data)-test_set_size:len(data)]
    train_indices = indices[0:len(data)-test_set_size]
    return data.iloc[train_indices], data.iloc[test_indices]


# TEST VECTORS: use the housing panda dataframe or the H numpy object, your choice
dat=housing
#dat=H

def TestSize(train_set, test_set):
    # works only for 0.2 split
    expected_n_train=16512
    expected_n_test=4128
    assert len(train_set)==expected_n_train, 'Oh, mismatch in expected train n'
    assert len(test_set) ==expected_n_test,  'Oh, mismatch in expected test n'
    print(len(train_set), "train +", len(test_set), "test","..OK")

train_set, test_set = my_split_train_test(dat, 0.2, shuffle=False)
TestSize(train_set, test_set)

train_set, test_set = my_split_train_test(dat, 0.2, shuffle=True)
TestSize(train_set, test_set)

from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(dat, test_size=0.2, shuffle=True, random_state=4
2)
TestSize(train_set, test_set)

train_set, test_set = train_test_split(dat, test_size=0.2, shuffle=False)
TestSize(train_set, test_set)
```

```
16512 train + 4128 test ..OK
16512 train + 4128 test ..OK
16512 train + 4128 test ..OK
16512 train + 4128 test ..OK
```

Qa Result

In the above printouts it's clear that our implementation of the Split_train_test function splits the set in two parts of the same size as the inbuilt version

Qb Why Shuffling

Explain why disabling shuffling is a bad idea?

Disabling shuffling is a bad idea since you may not have acquired your data randomly. Maybe you've gone from sector to sector which means that taking the first 20% of the dataset might only contain data from one sector, instead of a mix of all the sectors.

Qc Test and Compare

Compare your split function with the one from Scikit-learn, first using the simple X-y data set generated below and then using the housing data via the `H` numpy array variable.

Splitting the dataset via your split function and the built-in split does not yield a logical true for the comparison

```
(y_train == y_train_my).all().all()
```

Why is it so? Find the exact values in `H[i,j]` that are not equal and explain the problem.

Qc Implementation

In [9]:

```
# Simple data for Qc

import numpy as np
X1, y = np.arange(10).reshape((5, 2)), np.array([list(range(5))])

print("X=",X1)
print("y=",y)

# TODO: Qc...
dataset = pd.DataFrame({'Column1':X1[:,0],'Column2':X1[:,1]})


H = dat

def test_tts(X):
    # TEST VECTORS: notice that H is not splitted into X-y parts
    train, test = train_test_split(X, test_size=0.2, shuffle=False)
    print("build-in split: len(train)=",len(train),", len(test)=", len(test))

    train_my, test_my = my_split_train_test(X, test_size=0.2, shuffle=False)
    print("my split:       len(train)=",len(train_my),", len(test)=", len(test_my))

    # Test for equality here...
    assert train.shape==train_my.shape
    equal_train=(train.values == train_my.values).all().all()
    equal_test =(test.values == test_my.values).all().all()

    # TODO: why not equal?
    print("equal_train=", equal_train, ", equal_test=",equal_test)


test_tts(H)
test_tts(dataset)
```

```
X= [[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
y= [[0 1 2 3 4]]
build-in split: len(train)= 16512 , len(test)= 4128
my split:       len(train)= 16512 , len(test)= 4128
equal_train= False , equal_test= False
build-in split: len(train)= 4 , len(test)= 1
my split:       len(train)= 4 , len(test)= 1
equal_train= True , equal_test= True
```

Qc Results

In the above printouts we can see the two sets are equal in value (not just size). However changing the splitting ratio might cause some differences with smaller datasets, since one algorithm might round up while the other will round down.