Homework 3 With Solutions (40 Pts)

STAT 451: Machine Learning (Fall 2021)
Instructor: Sebastian Raschka (sraschka@wisc.edu)

In [1]:
```
%load_ext watermark
%watermark  -d -u -a 'Sebastian Raschka' -v -p numpy,scipy,matplotlib,sklearn
```

```
Author: Sebastian Raschka

Last updated: 2021-11-30

Python implementation: CPython
Python version       : 3.7.6
IPython version      : 7.12.0

numpy     : 1.21.2
scipy     : 1.7.1
matplotlib: 3.4.3
sklearn   : 1.0.1
```

1. Hyperparameter Tuning and Model Selection

In this exercise, you will be working with a diabetes dataset which is available from OpenML (https://www.openml.org/d/37 (https://www.openml.org/d/37)).

The dataset contains information about 768 patients along with the Diabetes diagnosis. The Diabetes diagnosis is a binary label, where "tested_positive" means that a patient has diabetes and "tested_negative" means that a patient does not have diabetes.

I additional to the class label, there are 8 numeric features in the dataset, which are listed below:

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

1.1 Using Grid Search for Hyperparameter Tuning

1.1.1) Load the Dataset [5 Pts]

Use pandas to load the dataset from the `dataset_37_diabetes` file from OpenML. (Hint: I provided the correct link for that, but you need to change something in the `read_csv` default code to load it correctly.)

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
#------------------#
##### STUDENTS #####
#------------------#

import pandas as pd


df = pd.read_csv('https://www.openml.org/data/get_csv/37/dataset_37_diabetes.arff') # YOUR CODE
df.head()
```

Out[3]:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive

1.1.2) Preprocess the class label [5 Pts]

Convert the class label using pandas `apply` or `map` method. The mapping should be as follows:

- `'tested_positive'` should be converted to `1`
- `'tested_negative'` should be converted to `0`

In [4]:
```
#------------------#
##### STUDENTS #####
#------------------#

df['class'] = df['class'].map({"tested_positive":1, "tested_negative":0})
df.head()
```

Out[4]:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

1.1.3) Split dataset into training and test sets [5 Pts]

- Split the dataset into 70% training and 30% test data
- Perform a stratified split
- use `0` as the random seed for shuffling

In [5]:
```
#------------------#
##### STUDENTS #####
#------------------#

from sklearn.model_selection import train_test_split


y = df['class'].values
X = df.iloc[:, :-1].values

X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)# YOUR CODE
```

1.1.4) Gridsearch and model selection [5 Pts]

Now, your task is to use `GridSearchCV` from scikit-learn to find the best parameters for `max_depth` and `criterion` for a decision tree. For max_depth, the values `[1, 2, 3, 4, 5, 10, 15, 20, None]` should be tried, and for criterion both Gini and Entropy should be considered.

In [6]:
```
#------------------#
##### STUDENTS #####
#------------------#


from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV


tree = DecisionTreeClassifier(random_state=123)

param_grid = {
    'max_depth': [1, 2, 3, 4, 5, 10, 15, 20, None],
    'criterion': ['gini','entropy']
}

gs = GridSearchCV(estimator=tree,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=10)

gs.fit(X_train, y_train)

print('Best Accuracy: %.2f%%' % (gs.best_score_*100))
```

```
Best Accuracy: 75.04%
```

Next, print the best hyperparameters obtained from the `GridSearchCV` run. Also, compute the accuracy the model, which uses the best hyperparameter settings and was trained on the whole training set, on the test set (`X_test` , `y_test`).

In [7]:
```
#------------------#
##### STUDENTS #####
#------------------#


print('Best Params: %s' % gs.best_params_)

## model is already fit to the whole training set because  we used `refit=True` in GridSearchCV
print('Test Accuracy: %.2f%%' % gs.best_estimator_.score(X_test, y_test))
```

```
Best Params: {'criterion': 'entropy', 'max_depth': 4}
Test Accuracy: 0.74%
```

In this exercise, you are asked to compute the accuracy of the model from the previous exercise (1.1), using the train set (`X_train` , `y_train`), via different bootstrap methods.

1.2.1 Compare the Out-of-Bag, .632, and .632+ bootstrap approaches [5 pts]

For computing the bootstrap estimates and confidence intervals, you are going to use the `bootstrap_point632_score` function implemented in MLxtend: http://rasbt.github.io/mlxtend/user_guide/evaluate/bootstrap_point632_score/ (http://rasbt.github.io/mlxtend/user_guide/evaluate/bootstrap_point632_score/)

The accruacy should be the mean accuracy over the 200 bootstrap values that the `bootstrap_point632_score` method returns.

- For this, use the best model you obtained from the previous exercise 1.1.4)
- use 200 bootstrap rounds
- set the random seed to 1

Compute Out-of-bag Bootstrap:

In [8]:
```
#------------------#
##### STUDENTS #####
#------------------#

from mlxtend.evaluate import bootstrap_point632_score
import numpy as np


# Compute Out-of-bag Bootstrap
scores = bootstrap_point632_score(gs,
                                  X_train, y_train,
                                  n_splits = 200,
                                  method = 'oob',
                                  random_seed = 1)


# Compute accuracy (average over the bootstrap rounds)
acc = np.mean(scores)
print('Accuracy: %.2f%%' % (100*acc))

# Compute the 95% confidence interval around the accuracy estimate
lower = np.percentile(scores, 2.5)
upper = np.percentile(scores, 97.5)
print('95%% Confidence interval: [%.2f, %.2f]' % (lower, upper))
```

```
Accuracy: 67.44%
95% Confidence interval: [0.61, 0.74]
```

.632 Bootstrap:

In [10]:
```
#------------------#
##### STUDENTS #####
#------------------#


# Compute .632 Bootstrap
scores = bootstrap_point632_score(gs,
                                  X_train, y_train,
                                  n_splits = 200,
                                  random_seed = 1)


# Compute accuracy (average over the bootstrap rounds)
acc = np.mean(scores)
print('Accuracy: %.2f%%' % (100*acc))

# Compute the 95% confidence interval around the accuracy estimate
lower = np.percentile(scores, 2.5)
upper = np.percentile(scores, 97.5)
print('95%% Confidence interval: [%.2f, %.2f]' % (lower, upper))
```

```
Accuracy: 74.97%
95% Confidence interval: [0.69, 0.80]
```

Compute .632+ Bootstrap:

In [11]:
```
#------------------#
##### STUDENTS #####
#------------------#


# Compute .632+ Bootstrap
scores = bootstrap_point632_score(gs,
                                  X_train, y_train,
                                  n_splits = 200,
                                  method = '.632+',
                                  random_seed = 1)


# Compute accuracy (average over the bootstrap rounds)
acc = np.mean(scores)
print('Accuracy: %.2f%%' % (100*acc))

# Compute the 95% confidence interval around the accuracy estimate
lower = np.percentile(scores, 2.5)
upper = np.percentile(scores, 97.5)
print('95%% Confidence interval: [%.2f, %.2f]' % (lower, upper))
```

```
Accuracy: 71.07%
95% Confidence interval: [0.63, 0.78]
```

1.2.2 Analyzing the different bootstrap results

- 1) **[5 Pts]** Based on what you have learned it class, which of the three bootstrap methods (out-of-bag, 0.632, or 0.632+) do you expect to yield a generalization accuracy estimate from the training set that is closest to the true generalization performance of the model? Explain your reasoning in 1-3 sentences. (Tip: Think about optimistic and pessimistic bias).

Out of bag would be the best because when we adjust for optimistic bias, the accuracy is around 77%. This is much closer to the training accuracy value of 75% from before. The other values after adjusting for optimistic bias aren't as close to 75%.

- 2) **[5 Pts]** Based on your observations from the experiment in 1.2.1), which bootstrap approach (out-of-bag, 0.632, or 0.632+) yields an accuracy estimate from the training dataset that is closest to the test set accuracy from exercise 1.1.4? Is this reasonable? Explain your answer in 1-3 sentences. Also, to answer this question, assume that the test set accuracy from 1.1.4) is a perfect estimate of the true generalization accuracy of the model.

It is 0.632 because the value of 74.97 is much closer to 75.04 than the 0.632+ value of 71.07. It is reasonable because we don't need to adjust for bias since we are given that it is a perfect estimate.

- 3) **[5 Pts]** Based on your observations from the experiment in 1.2.1), are the overall results consistent with what you expected in your answer above (question 1))? Explain your reasoning in 3-5 sentences. Also, to answer this question, assume that the test set accuracy from 1.1.4) is a perfect estimate of the true generalization accuracy of the model.

Tip: Discuss which methods are optimistically and pessimistically biased and whether this was expected.

Yes, the accuracy is within the confidence intervals adjusting for bias. OOP can be optimistically bias, and adjusting for this brings us to close to the original model.