Homework 3 (20 Pts)

STAT 451: Machine Learning (Fall 2020)
Instructor: Sebastian Raschka (sraschka@wisc.edu (mailto:sraschka@wisc.edu))

Course website: http://pages.stat.wisc.edu/~sraschka/teaching/stat451-fs2020/ (http://pages.stat.wisc.edu/~sraschka/teaching/stat451-fs2020/)

Due: Dec 04, (before 11:59 pm).

How to submit

As mentioned in the lecture, you need to send the `.ipynb` file with your answers plus an `.html` file, which will serve as a backup for us in case the `.ipynb` file cannot be opened on my or the TA's computer. In addition, you may also export the notebook as PDF and upload it as well.

The homework solution should be uploaded on Canvas. You can submit it as often as you like before the deadline.

Important

- The cells that require your code answer are marked with `"# YOUR CODE"` comments.
- Note that you may use 1 or more line of code for replacing each `"# YOUR CODE"` comment.

For example, imagine there is a question asking you to implement a threshold function that should return 1 if the input `x` is greater than 0.5 and otherwise. This could appear as follows in the the exercise:

```
def threshold_func(x):
    # YOUR CODE
```

A valid answer could be

```
def threshold_func(x):
    if x > 0.5:
        return 1
    else:
        return 0
```

Another valid solution could be

```
def threshold_func(x):
    return int(x > 0.5)
```

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

```
Roshan
last updated: 2020-11-25

CPython 3.8.3
IPython 7.16.1

numpy 1.18.5
scipy 1.5.0
matplotlib 3.2.2
sklearn 0.23.1
```

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

1. Hyperparameter Tuning and Model Selection

1.1 Using Grid Search for Hyperparameter Tuning

In this exercise, you will be working with the *Pima Indians Diabetes Database* database by Vincent Sigillito (vgs@aplcen.apl.jhu.edu), which is available from the UCI database (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes)) or 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)

1.1.1) Load the Dataset [2 Pts]

Use pandas to load the dataset from the `dataset_37_diabetes.csv` CSV file located in the HW folder.

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

import pandas as pd


df = pd.read_csv('dataset_37_diabetes.csv')
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 [2 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 [2 Pts]

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

In [15]:
```
#------------------#
##### 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, train_size=.7, stratify=y, random_state=123)
```

1.1.4) Gridsearch and model selection [4 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,
                  n_jobs=-1,
                  refit=True,
                  scoring='accuracy',
                  cv=10)

gs.fit(X_train, y_train)

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

```
Best Accuracy: 73.91%
```

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` i
print('Test Accuracy: %.2f%%' % ((sum((gs.best_estimator_.predict(X_test) == y_te
```

```
Best Params: {'criterion': 'gini', 'max_depth': 2}
Test Accuracy: 71.00%
```

1.2 Estimate the Generalization Performance using Bootstrap Methods

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 [4 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

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

from mlxtend.evaluate import bootstrap_point632_score
import numpy as np


# Compute Out-of-bag Bootstrap
scores = bootstrap_point632_score(gs.best_estimator_, X_train, y_train, method='o


# 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: 74.37%
95% Confidence interval: [0.68, 0.80]
```

In [12]:
```
#------------------#
##### STUDENTS #####
#------------------#

from mlxtend.evaluate import bootstrap_point632_score
import numpy as np


# Compute .632 Bootstrap
scores = bootstrap_point632_score(gs.best_estimator_, X_train, y_train, n_splits=


# 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: 75.79%
95% Confidence interval: [0.72, 0.79]
```

In [13]:
```
#------------------#
##### STUDENTS #####
#------------------#

from mlxtend.evaluate import bootstrap_point632_score
import numpy as np


# Compute .632+ Bootstrap
scores = bootstrap_point632_score(gs.best_estimator_, X_train, y_train, method='


# 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: 75.38%
95% Confidence interval: [0.71, 0.79]
```

1.2.2 Analyzing the different bootstrap results

- 1) [2 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).

According to the lecture that showed 95% confidence intervals for bootstrapping methods, the OOB is pessimistically biased. Similarly, we can see that the .632 bootstrap is overly optimistic. The .632+ bootstrap was designed to not suffer from the same optimistic bias that .632 suffers from, so I expect .632+ bootstrap to have the least bias.

- 2) [2 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.

The oob on the training set was closest to the test set accuracy, although all bootstrapping methods in my case were overly optimistic. This is reasonabe because there are only ~800 records in our full dataset and we are using a high number of splits.

- 3) [2 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.

The results I found were surprising. I expected the oob to be pessimistic. It turned out to just be the least optimistically biased. Perhaps if we used a different number of splits, or a larger dataset, we would be able to see more significant results. Overall, even though all the bootstrap methods seemed optimistically biased, the ordering of the scores they produced was expected (oob < .632+ < .632).