exercise_04

December 4, 2023

1 Exercise 4

1.1 Recap: Overfitting / Underfitting

1. Explain the bias / variance trade-off in your own words.
2. For the data below, plot `X` against `y_orig` and `X` against `y` (**hint**: `y` is a noisy variant of `y_orig`).
3. Split the data into train (80%) and test (20%) sets.
4. Fit a `DecisionTreeRegressor` to the train set.
5. Calculate the mean absolute error for train and test set.
6. Visualize the predictions for the train and test set.
7. Try different values for the parameter `max_depth` of the `DecisionTreeRegressor`. Can you underfit, fit well, and overfit?
8. How does this connect to the bias / variance trade-off?

[1]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import mean_absolute_error
```

[2]:

```
# generate X and y
np.random.seed(12)
n_steps = 4
X = np.arange(50 * n_steps).reshape((-1,1))
y_orig = np.repeat(np.arange(4), int(X.shape[0] / n_steps))
y = y_orig + (np.random.random(y_orig.size) - 0.5) * 2
```

[ ]:

```

```

1.2 Hyperparameter Optimization

1) What is the difference between parameters and hyperparameters of a model? Give examples!
2) What is hyperparameter optimzation?
3) Name two methods for hyperparameter optimization and compare them.
4) For the following dataset, report the mean absolute error for a `DecisionTreeRegressor` using 10-fold cross validation via the function `cross_val_score`. You can use the function's parameters `cv` and `scoring` to set the the number of cross validation splits and the validation

metric (e.g. neg_mean_absolute_error). There is no need for an initial train/test split in this example.

5) Now use GridSearchCV to optimize the max_depth parameter in the range of 1,2,3,4 and 5 for a DecisionTreeRegressor and report the same score. For this, run GridSearchCV with a 10-fold cross-valiation, cf. cv and param_grid parameters of GridSearchCV.
6) What do you think the notion of *“nested cross-validation”*, *“inner cross-validation”* and *”outer cross-validation* refer to in this example?

[15]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.metrics import mean_absolute_error
```

[16]:

```
# generate X and y
# generates X and y
np.random.seed(12)
n_steps = 4
X = np.arange(50 * n_steps).reshape((-1,1))
y_orig = np.repeat(np.arange(4), int(X.shape[0] / n_steps))
y = y_orig + (np.random.random(y_orig.size) - 0.5) * 2
```

1.3 Evaluation: Groups

1) Let’s consider a cancer dataset. The goal is to predict the progression of the cancer (e.g., stage 1, stage 2, …). Each patient appears multiple times in the dataset. Can you randomly split the data into train and test set?
2) Read the documentation of GroupedShuffleSplit, what does it do? In the process, explain what the groups parameter of the split method is used for!
3) Apply GroupedShuffleSplit on the data below and examine the results. The documentation of GroupedShuffleSplit also provides an example.
4) At home, have a look at different splitting strategies provided by scikit-learn.

[20]:

```
# code to generate data (skip this)
from sklearn.model_selection import GroupShuffleSplit
import numpy as np

rng = np.random.RandomState(1338)
cmap_data = plt.cm.Paired
cmap_cv = plt.cm.coolwarm
n_splits = 4

# Generate the class/group data
n_points = 20
X = rng.randn(n_points, 3)

y = rng.choice([0,1], n_points)
```

```
# Generate 10 uneven groups
# draw a prior for the likelihood of each group from the Dirichlet Distribution
# given that we observed elements of each group alpha_i - 1 time (here, alpha_i = 2)
group_prior = rng.dirichlet([2] * 10)
# assign a group [0,...,9] to each of the 20 points
groups = np.repeat(np.arange(10), rng.multinomial(n_points, group_prior))
```

[21]:

```
# use X, y, groups and apply Grouped
X, y, groups
```

[21]:

```
(array([[ 0.30339572,  0.06905289, -1.36994721],
        [-1.73542443,  0.92038986, -0.67328559],
        [ 0.31130278,  1.65990867, -0.38992731],
        [ 0.24670422,  1.26286911, -0.07143212],
        [ 0.21765819, -1.03871059, -0.57730003],
        [ 0.22515547, -1.11415101, -0.21211768],
        [-0.11681968, -0.23243284,  1.01017204],
        [-1.52260114, -0.03899189, -0.4953006 ],
        [-1.26511675,  1.30947599, -0.04807498],
        [ 0.65844446,  1.17818292,  0.92479273],
        [-0.1361633 , -0.43271388, -1.41218241],
        [-1.72601721, -0.12744758,  1.36070897],
        [ 1.80491419, -0.8311191 , -0.75247305],
        [-0.86747453, -1.09877395, -0.01585959],
        [ 1.76340557, -1.0951515 ,  0.65760157],
        [ 0.85346956, -1.34077646,  1.25792539],
        [ 1.40744721, -1.92822233,  0.34033999],
        [-0.82224763,  0.98701143, -0.40516663],
        [-0.47011842, -0.49503657, -0.38135189],
        [ 0.14000614, -0.99631156,  0.47125006]]),
 array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1]),
 array([2, 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 9, 9]))
```

1.4 Scaling

1) What is scaling?
2) Why is scaling important?
3) What are two comming examples of scaling? What do they do?
4) Apply these two scaling methods to the following data and plot the results. What do you observe (look at the axes)?
5) Add an outlier to the data and try again. What do you observe and why is this a problem?
6) At home, look at the different scaling procedures provided by `scikit-learn`. Which scalers would help aleviate the outlier issue?

[23]:

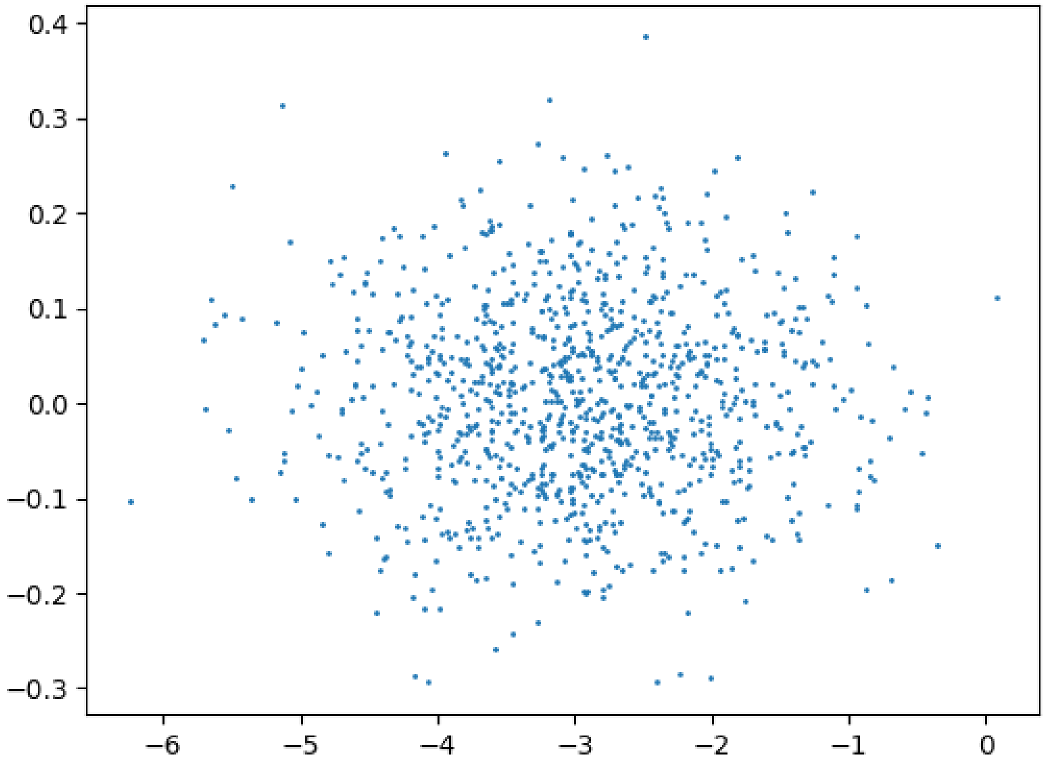
```
# generate the data
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(42)
X = np.random.multivariate_normal((0,0), [[1,0],[0,1]], 1000,)
X[:,0] = X[:,0] - 3
X[:,1] = X[:,1] / 10
```

[24]:

```
# plot the data
plt.scatter(X[:,0], X[:,1], s=1)
```

[24]:

```
<matplotlib.collections.PathCollection at 0x7f6fd071cc90>
```



[25]:

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
```

[ ]:

```

```

1.5 Preprocessing

Given the house prices dataset from Exercise 2:

1) Give some example features for the four different data types.
2) Name at least two issues in this dataset and two others that could occur when training a model.

3) Assume that you are given the postal code of a house - what other features could you add to the data? Why do you think I ask this question?

[31]:

```
import pandas as pd
import seaborn as sns
```

[32]:

```
data_houses = pd.read_csv(
    "_assets/house-prices-advanced-regression-techniques/train.csv",
    index_col="Id")
```

[ ]:

```

```