exercise_03

November 30, 2023

1 Exercise 3

1.1 Evaluation

1.1.1 Classification

Prediction possibilities For a binary classification problem, what are the four prediction possibilities? List their names and briefly explain what they represent.

[ ]:

Define the accuracy measure

[ ]:

Define precision and recall

[ ]:

Properties of precision and recall What can a classifier predicting the class `0` or `1` do to

1) always get a *precision* of `1`
2) always get a *recall* of `1`

Application of precision and recall In what applications is precision more important than recall, and in which applications is recall more important than precision?

[ ]:

Combine precision and recall

- What is a measure that combines precision and recall?
- Define it.
- Why do we use the harmonic rather than the arithmetic mean?

[ ]:

Default values Consider the following, label set:

- `y`: roughly the same amount of cases (`1`) and controls (`0`)

Now, calculate the *accuracy*, *precision*, *recall*, and *ROC AUC*.

1) For a classifier that returns random labels. What do you observe?
2) For a classifier that always returns `1`. What do you observe?

Hints: * You can simulate these classifiers without input data, i.e., by generating their predictions manually. * You can use `numpy` and `scikit-learn` to show these cases instead of answering theoretically.

[139]:
```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪roc_auc_score, classification_report

np.random.seed(42)
n = 1000

# labels 1 (random)
y = np.random.choice([0,1], p=(0.5, 0.5), size=n)
```

[ ]:
```
```

Imbalanced data Consider the following, label set:

- `y_imbalanced`: only a small set of cases (`1`) compated to controls (`0`)

Now, calculate the *accuracy*, *precision*, *recall*, and *ROC AUC*.

1) For a classifier that returns random labels. What do you observe?
2) For a classifier that always predicts the majority class (`0`). What do you observe and why could this be an issue in practice (particularly if we only consider `accuracy`)?
3) Which class label does `scikit-learn` consider to be a "case", i.e., the class of interest?

Hints: * You can simulate these classifiers without input data, i.e., by generating their predictions manually. * You can use `numpy` and `scikit-learn` to show these cases instead of answering theoretically.

[163]:
```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪roc_auc_score, classification_report

np.random.seed(42)
n = 1000

# labels 2 (imbalanced, i.e., small sets of `cases` (label=1))
y_imbalance = np.random.choice([0,1], p=(0.9, 0.1), size=n)
```

[ ]:
```
```

1.1.2 Regression

MAE vs MSE

1. Write down the formulas for MAE and MSE.

2. When would you use MAE and when would you use MSE?
3. Which functions correspond to these measures in `scikit-learn`?

```
[ ]:
```

R^2

1. Write down the formula for the R^2 measure.
2. Look at the formula and try to understand what the R^2 measure intuitively measures. **Hint:** Consider the case were we only predict the mean $\bar{y}$.
3. Which functions corresponds to R^2 in `scikit-learn`?

```
[ ]:
```

1.2 Train / Test

1. Use `train_test_split` to shuffle and split the given data `(X, y)` into a train and a test dataset with a 80:20 ratio. (**Hint:** you can use the parameters `shuffle` and `test_size`)
2. Train the `DecisionTreeClassifier` on the training dataset.
3. Calculate the ROC AUC score for the training and the test dataset.
4. What do you observe?
5. Why should we always have a test set?
6. Why should we NEVER fit a model before we define a test set?

```
[1070]: np.random.seed(42)

n_samples = 1000
n_features = 100
X = np.random.random((n_samples, n_features))
y = np.random.choice([0, 1], p=(0.5, 0.5), size=n_samples)
```

```
[1071]: from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
[ ]:
```

1.3 Cross Validation

Consider the following dataset `(X, y)`. We already split it into a train `(X_train, y_train)`, `(X_val, y_val)`, and a test set `(X_test, y_test)`.

1. Fit a `DecisionTreeClassifier` on the train dataset.
2. Calculate the ROC AUC score on the train, validation, and test dataset. What do you observe?
3. How would you apply cross validation and how would it help you?
4. Apply cross validation appropriately and report the mean and standard deviation of the ROC AUC scores.
5. In addition to the ROC AUC score on the test set, why would you also always report the mean and standard deviation of the cross validations scores?

6. Why would you be a bit suspicious of the current AUC SCORE on the test set?
7. BONUS: Why are we observing these results based on the data we use?

[1685]:
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

[1686]:
```
np.random.seed(42)

n_samples = 200 * 6
X, y = make_classification(n_samples=n_samples, flip_y=0.01, n_redundant=2,␣
↪n_informative=2)
X[-2*int(n_samples / 6):-int(n_samples / 6),:] = np.random.
↪random((int(n_samples / 6), X.shape[1]))
```

[1687]:
```
# define the test set
X_intermediate, X_test, y_intermediate, y_test = train_test_split(X, y,␣
↪test_size=1/6, shuffle=False)
```

[1688]:
```
# use the remaining data to define the train and validation set
X_train, X_val, y_train, y_val = train_test_split(X_intermediate,␣
↪y_intermediate, test_size=0.2, shuffle=False)
```

[1689]:
```
X_train, y_train
X_val, y_val
X_test, y_test
;
```

[1689]:
```
''
```

1.4 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` and calculate the mean absolute error for train and test set
5. Visualize the predictions for train and test.
6. Try different `max_depth`. Can you underfit, fit well, and overfit?
7. How does this connect to the bias / variance trade-off?

[1692]:
```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
```

[3]:
```
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
```

[ ]:

```

```