Schedule

Block 2 (11:00 am - 12:30 pm)

(3) Data preprocessing in Python

(4) Evaluating & tuning machine learning classifiers

Lunch break (12:30 - 1:30 pm)

Machine Learning vs Deep Learning

Supervised learning dataset

Unstructured data

Input:

https://en.wikipedia.org/wiki/Iris_setosa#/media/File:Irissetosa1.jpg

Label: **Iris-setosa**

Supervised learning dataset

Unstructured data

Structured data

Inputs:

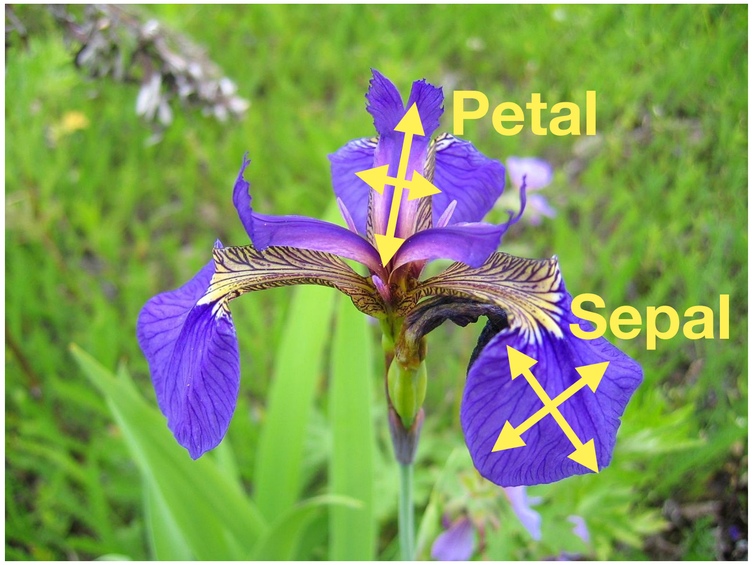
Sepal length	Sepal width	Petal length	Petal length
5.1	3.5	1.4	0.3
4.9	3	1.4	0.2
5.9	3	5.1	1.8
…	…	…	…

Labels:

Label
Iris-setosa
Iris-setosa
Iris-virginica
…

Supervised learning dataset

"Manual" feature engineering



Sepal length	Sepal width	Petal length	Petal length
5.1	3.5	1.4	0.3
4.9	3	1.4	0.2
5.9	3	5.1	1.8
…	…	…	…

Supervised learning dataset

Deep learning

“Traditional” machine learning

Sepal length	Sepal width	Petal length	Petal length
5.1	3.5	1.4	0.3
4.9	3	1.4	0.2
5.9	3	5.1	1.8
...	...	...	...

Stratified Splits

```
from sklearn.model_selection import train_test_split

X_temp, X_test, y_temp, y_test = \
        train_test_split(X, y, test_size=0.2,
                         shuffle=True, random_state=123, stratify=y)
np.bincount(y_temp)
```

```
array([40, 40, 40])
```

```
X_train, X_valid, y_train, y_valid = \
        train_test_split(X_temp, y_temp, test_size=0.2,
                         shuffle=True, random_state=123, stratify=y_temp)
X_train.shape
```

```
(96, 4)
```

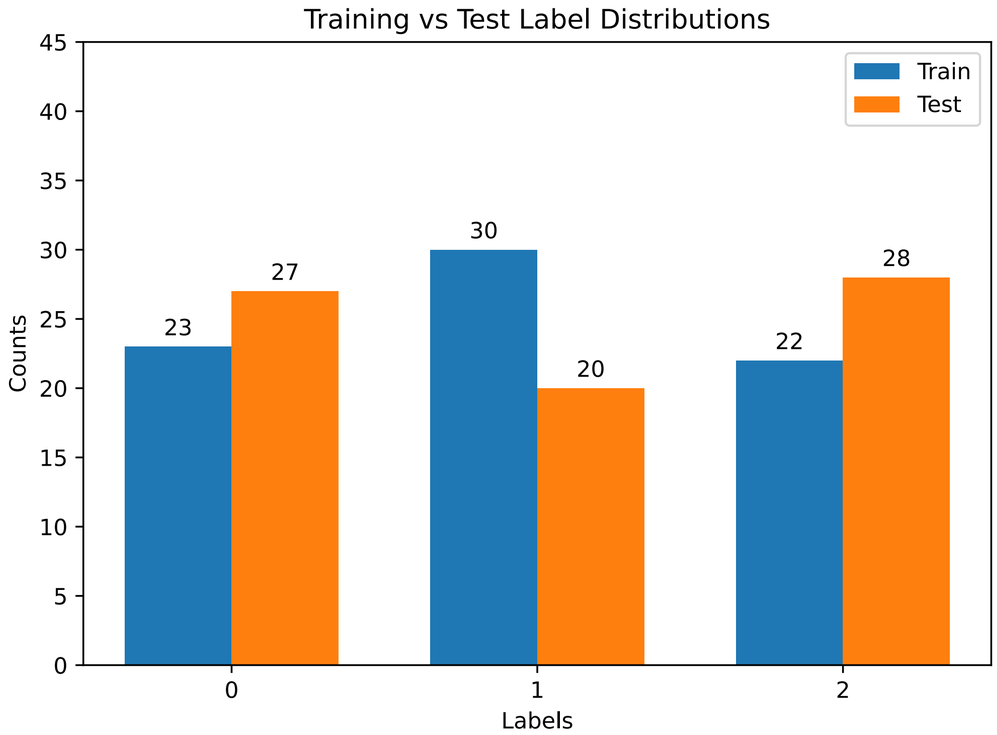
Stratified random train/test splits

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.5, random_state=123)
```

```
import numpy as np

print("Training labels", np.bincount(y_train))
print("Test labels", np.bincount(y_test))
```

```
Training labels [23 30 22]
Test labels [27 20 28]
```



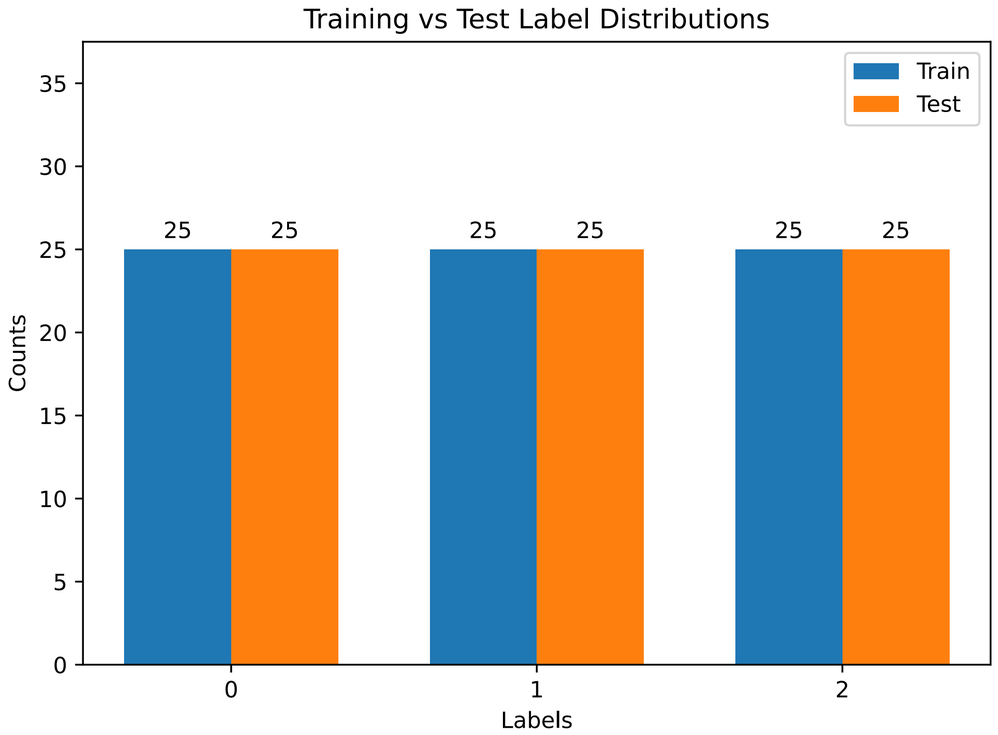
Stratified random train/test splits

```python
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
```

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.5, random_state=123, stratify=y)
```

```python
import numpy as np

print("Training labels", np.bincount(y_train))
print("Test labels", np.bincount(y_test))
```

```
Training labels [25 25 25]
Test labels [25 25 25]
```



Normalization: Z-score standardization

$$x_{std}^{[i]} = \frac{x^{[i]} - \mu_x}{\sigma_x}$$

```
x = np.arange(6).astype(float)
x
```

```
array([0., 1., 2., 3., 4., 5.])
```

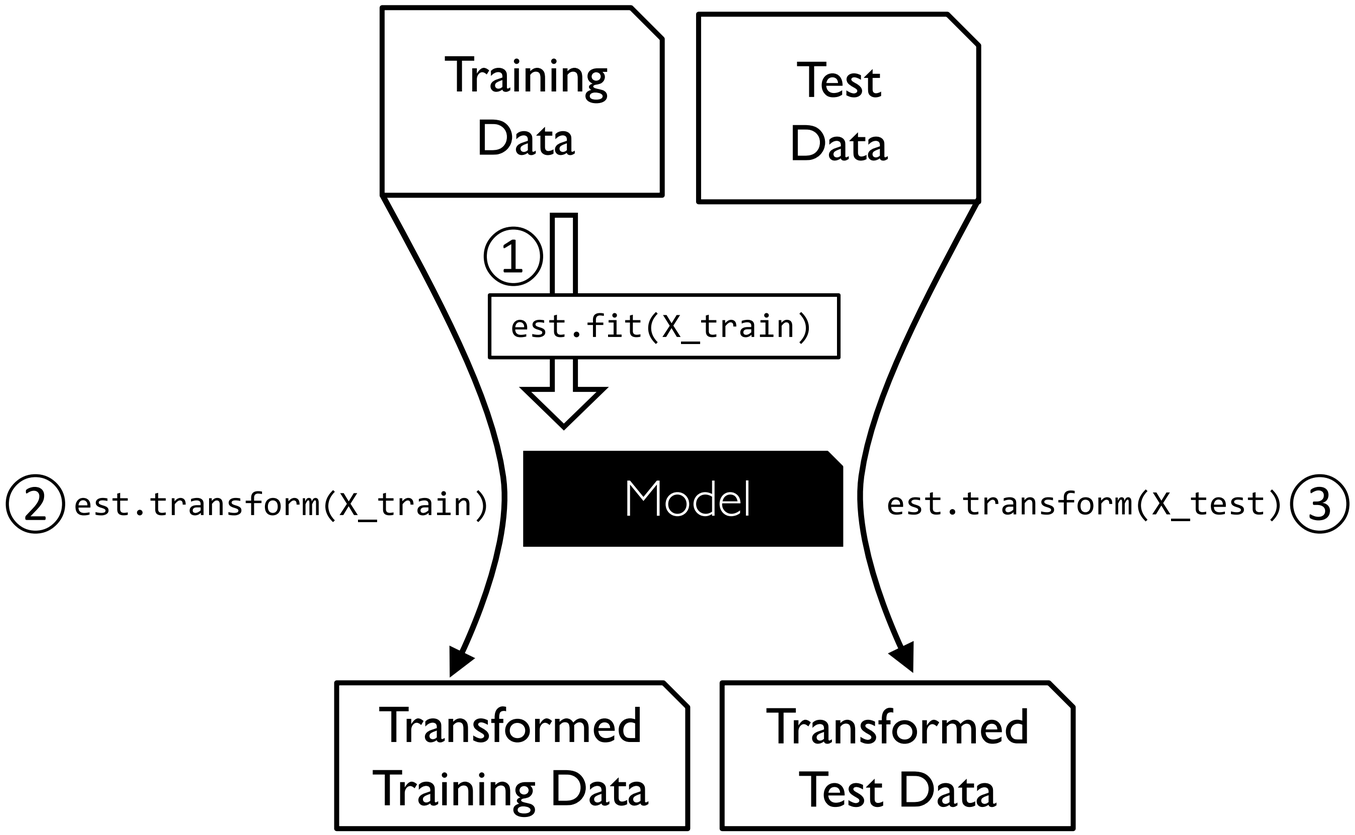
```
x_std = (x - x.mean()) / x.std()
x_std
```

```
array([-1.46385011, -0.87831007, -0.29277002,  0.29277002,  0.87831007,
        1.46385011])
```

Scaling validation and test sets

```
mu, sigma = X_train.mean(axis=0), X_train.std(axis=0)

X_train_std = (X_train - mu) / sigma
X_valid_std = (X_valid - mu) / sigma
X_test_std = (X_test - mu) / sigma
```

The scikit-learn transformer API



The scikit-learn transformer API

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X_train)

X_train_std = scaler.transform(X_train)
X_valid_std = scaler.transform(X_val)
X_test_std = scaler.transform(X_test)
```

Working with categorical data

```
import pandas as pd

df = pd.read_csv("data/categoricaldata.csv")
df.head()
```

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XXL	15.3	class1

Mapping ordinal data

```
mapping = {
    "S":   1,
    "M":   2,
    "L":   3,
    "XL":  4,
    "XXL": 5
}

df["size"] = df["size"].map(mapping)
df
```

	color	size	price	classlabel
0	green	2	10.1	class1
1	red	3	13.5	class2
2	blue	5	15.3	class1

Encoding nominal label categories

```
from sklearn.preprocessing import LabelEncoder


lenc = LabelEncoder()

lenc.fit(df["classlabel"]);
```

```
df["class_label"] = lenc.transform(df["classlabel"])
df
```

	color	size	price	classlabel	class_label
0	green	2	10.1	class1	0
1	red	3	13.5	class2	1
2	blue	5	15.3	class1	0

Encoding nominal feature categories

	color	size	price	classlabel
0	green	2	10.1	0
1	red	3	13.5	1
2	blue	5	15.3	0

Encoding nominal feature categories

	color	size	price	classlabel
0	green	2	10.1	0
1	red	3	13.5	1
2	blue	5	15.3	0

```
from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder(drop="first")
ohe.fit(df[["color"]]);
```

```
ohe.transform(df[["color"]]).toarray()
```

```
array([[1., 0.],
       [0., 1.],
       [0., 0.]])
```

Encoding nominal feature categories

	color	size	price	classlabel
0	green	2	10.1	0
1	red	3	13.5	1
2	blue	5	15.3	0

```
from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder(drop="first")
ohe.fit(df[["color"]]);
```

```
ohe.transform(df[["color"]]).toarray()
```

```
array([[1., 0.],
       [0., 1.],
       [0., 0.]])
```

Encoding nominal feature categories

```
encoded_data = ohe.transform(df[["color"]]).toarray()
```

```
# The new features
encoded_df = pd.DataFrame(
    encoded_data, columns=ohe.get_feature_names_out(["color"]))

# Concatenate the DataFrames along axis 1
df = pd.concat([df, encoded_df], axis=1)

# Drop the original column, if desired
df = df.drop(columns="color")

df
```

	size	price	classlabel	color_green	color_red
0	2	10.1	0	1.0	0.0
1	3	13.5	1	0.0	1.0
2	5	15.3	0	0.0	0.0

Dealing with missing data

```
import pandas as pd

df = pd.read_csv("data/missingdata.csv")
df.head()
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

Dealing with missing data

```
import pandas as pd

df = pd.read_csv("data/missingdata.csv")
df.head()
```

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	10.0	11.0	12.0	NaN

```
# drop rows with missing values:
df.dropna(axis=0)
```

	A	B	C	D
0	1.0	2.0	3.0	4.0

```
# drop columns with missing values:
df.dropna(axis=1)
```

	A	B
0	1.0	2.0
1	5.0	6.0
2	10.0	11.0

Exercise:

Data preparation, scaling, and classification

🎮 https://github.com/rasbt/posit2023-python-ml