The UK's European university

University of Kent

COMP8270 / PROGRAMMING FOR ARTIFICIAL INTELLIGENCE

Fernando Otero
febo@kent.ac.uk
cs.kent.ac.uk/people/staff/febo

overview:

1. Classification

2. Python coding:
   - Decision Trees and Random Forests

overview:

1. **Classification**

2. **Python coding:**
   - **Decision Trees and Random Forests**

Classification task (1):

- Supervised learning

- Consist of finding a model that is able to predict the value of a **target** (class) attribute of an example based on the values of a set of **predictor** attributes

Classification task (2):

Predictor Attributes | *Class Attribute*

breast	tumor size	age	irradiated	recurrence
left	4	29	no	no
right	29	40	no	yes
right	14	38	yes	no
right	49	64	no	yes
left	55	32	yes	no
right	24	52	yes	yes

Classification task (3):

- Goal: discover a relationship which allows us to predict the class of an example, given its predictor attributes
- This relationship is discovered by using a training set, where the class of examples is known …
- … and then the relationship is used to predict the class of examples in the test set, whose class is unknown

Classification task (4):

Training set
(known-class examples)

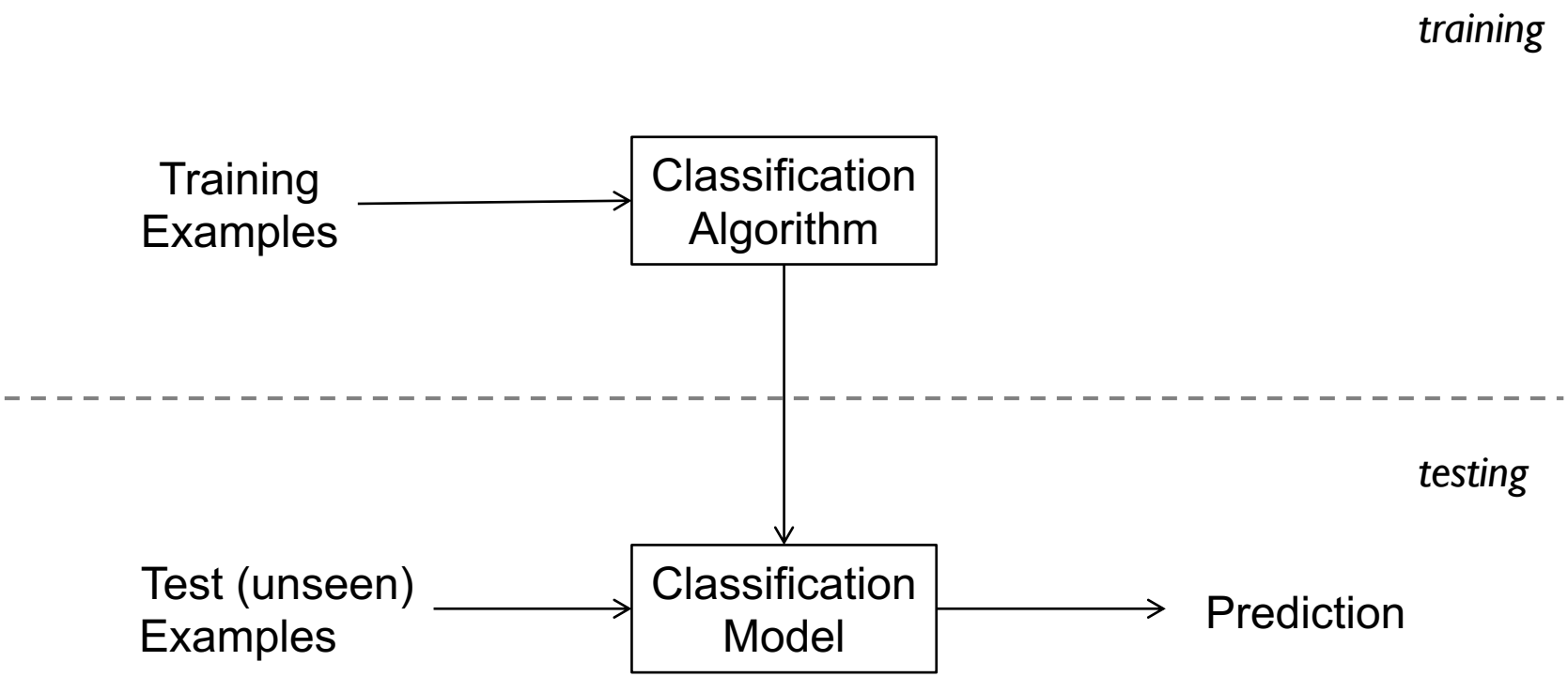
	. . .		recurrence?
			no
			yes
			no
			yes
			no
			yes

Test set
(unknown-class examples)

	. . .		recurrence?
			?
			?
			?
			?
			?
			?

Classification involves induction from training set (**"the past"**), in order to predict the class of test examples (in **"the future"**)

Classification task (5):



overview:

1. Classification

2. Python coding:
   - Decision Trees and Random Forests

Classification with Python:

1. Choose your data

2. Pre-processing
   - Check missing values
   - Attribute transformation?

3. Create a classification model

4. Evaluate the performance

Loading the data:

- pandas can load directly from file/remote:

```
import pandas as pd

# make the column names nice
columns = ["sepal length",
           "sepal width",
           "petal length",
           "petal width",
           "class"]

iris_data = pd.read_csv(
    'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
    header=None,
    names=columns)
```

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Loading the data:

- pandas can load directly from file/remote:

```
import pandas as pd

# make the column names nice
columns = ["sepal length",
           "sepal width",
           "petal length",
           "petal width",
           "class"]

iris_data = pd.read_csv(
    'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
    header=None,
    names=columns)
```

predictor attributes

target attribute

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Pre-processing:

- Do we need any at this point?

```
# information regarding the attributes
iris_data.info()

>  #   Column        Non-Null Count  Dtype
> ---  ------        --------------  -----
>  0   sepal length  150 non-null    float64
>  1   sepal width   150 non-null    float64
>  2   petal length  150 non-null    float64
>  3   petal width   150 non-null    float64
>  4   class         150 non-null    object


# see the class distribution
iris_data['class'].groupby(iris_data['class']).count()
```

Model creation (1):

- Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier

# prepares the training data
X_train = iris_data.loc[:, 'sepal length':'petal width']
y_train = iris_data['class']

# builds the classifier
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
```

Model creation (2):

- Visualising the model:

```
from matplotlib import pyplot as plt
from sklearn.tree import export_text
from sklearn.tree import plot_tree

# text representation
text_representation = export_text(tree)
print(text_representation)

# graphic representation
fig = plt.figure(figsize=(25,20))
_ = plot_tree(tree, feature_names=columns, class_names=y_train.values, filled=True)
```

Evaluation (1):

- Accuracy:

$$\frac{\#\ correct\ classification}{\#\ total\ examples}$$

```
# accuracy score [0, 1]
tree.score(X_train, y_train)
```

Evaluation (2):

- Confusion matrix:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


predictions = tree.predict(X_train)
cm = confusion_matrix(y_train, predictions, labels=tree.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=tree.classes_)

# displays the matrix
_ = disp.plot()
```

…but:

- We have been evaluating the model on the training data!

- It does not measure whether the model generalises to unknown data well or not.

Evaluation (3):

- train + test partition:

```
from sklearn.model_selection import train_test_split

X = iris_data.loc[:, 'sepal length':'petal width']
y = iris_data['class']

# splits the data in training + testing
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# builds the classifier
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)


# evaluates on the test data
tree.score(X_test, y_test)

# displays the confusion matrix
predictions = tree.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=tree.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=tree.classes_)
_ = disp.plot()
```

Another classifier:

- Easy to use other classifiers:

```
from sklearn.ensemble import RandomForestClassifier


# ... as before

# instead of tree use a forest
forest = RandomForestClassifier(n_estimators=10, random_state=0)

# ... as before
```

Next lecture:

- Regression