Lecture: classification in Python
Learning objectives

In this lecture we will...
- Demonstrate how to set up classification problems in Python
- Introduce the `LogisticRegression` model from the `sklearn` library
We'll look at a simple dataset from the UCI repository: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data
• This dataset is concerned with which (Polish) companies go bankrupt
Reading the data

- Data is in CSV format, but first contains a header that we need to skip

```python
In [1]: f = open("datasets/bankruptcy/5year.arff", 'r')

In [2]: while not '@data' in f.readline():
   pass

  Header ends and the "real" data begins after we see the "@data" tag

- Next we read the CSV data. We (a) skip rows with missing entries; (b) convert all fields to floats; and (c) convert the label to a bool

```
Processing the data

• Next let's look at some simple statistics about our data

In [4]: len(dataset)

Out[4]: 3031  ➔ Number of samples (after discarding missing values)

In [5]: sum([x[-1] for x in dataset])

Out[5]: 102  ➔ Number of positive samples

• Next we extract our features (X) and labels (y), much as we would do for a regression problem

In [6]: X = [values[:-1] for values in dataset]

In [7]: y = [values[-1] for values in dataset]

True/False labels
Concept: The **sklearn** library

The **sklearn** library contains a number of different regression and classification models

For example:

- `linear_model.LinearRegression()` - linear regression
- `linear_model.LogisticRegression()` - logistic regression
- `svm.SVC` - Support Vector Classifier

- In this lecture we'll use the **LogisticRegression** module
Fitting the logistic regression model

- First we import the library and create an instance of the model, before fitting it to data

```python
In [8]: from sklearn import linear_model
In [9]: model = linear_model.LogisticRegression()
In [10]: model.fit(X, y)
```

- Note that this function doesn't produce any output, rather it just updates the class instance to store the model
Making predictions

• Make predictions from the data:

```python
In [11]: predictions = model.predict(X)
```

```python
In [12]: predictions
```

```
Out[12]: array([False, False, False, ..., False, False, False, False])
```

• Check whether they match the labels

```python
In [13]: correctPredictions = predictions == y
```

```python
In [14]: correctPredictions
```

```
Out[14]: array([ True,  True,  True, ..., False, False, False])
```

• And compute the error

```python
In [15]: sum(correctPredictions) / len(correctPredictions)
```

```
Out[15]: 0.9663477400197954
```
We achieved fairly high accuracy using a simple classifier "off the shelf"

- But note that we're evaluating our classifier on the same data that was used to train it
- How can we be sure that our classifier will work well on unseen data?
- This is something we'll cover in the next course, when we look at training, testing, and validation
Other classification algorithms in sklearn

This example showed how to use **logistic regression**, but other classifiers are available in sklearn and have a similar interface:

- sklearn.svm.SVC: **Support Vector Classifier**
- sklearn.tree.DecisionTreeClassifier: **Decision trees**
  - sklearn.naive_bayes: **Naïve Bayes**
- sklearn.neighbors.KNeighborsClassifier: **Nearest Neighbors**

Summary of concepts

- Introduced the sklearn library
- Showed how to set up a simple classification problem in Python

On your own...

- Try to set up a similar classification problem using another of the UCI datasets – look for classification datasets that have numerical attributes (i.e., datasets similar to the one used for this exercise)