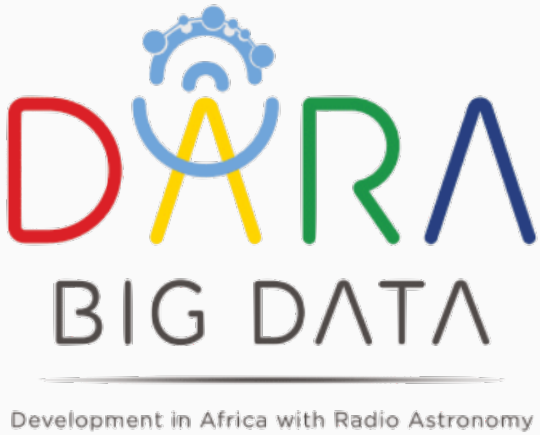
Python for Data Science

Classification



Anna Scaife

University of Manchester

Glossary: Types of Classification

Random Forest Uses multiple decision trees and bootstrapping

Use it for: Multi-class problems; mixed numerical and categorical data.

Python libraries: `scikit_learn`

Support Vector Machine (SVM) Uses geometric separation of labelled data points.

Use it for: Binary classification; fully numeric data.

Python libraries: `scikit_learn`

Convolutional Neural Network (CNN) Uses analogue of brain function firing neurons.

Use it for: Image data

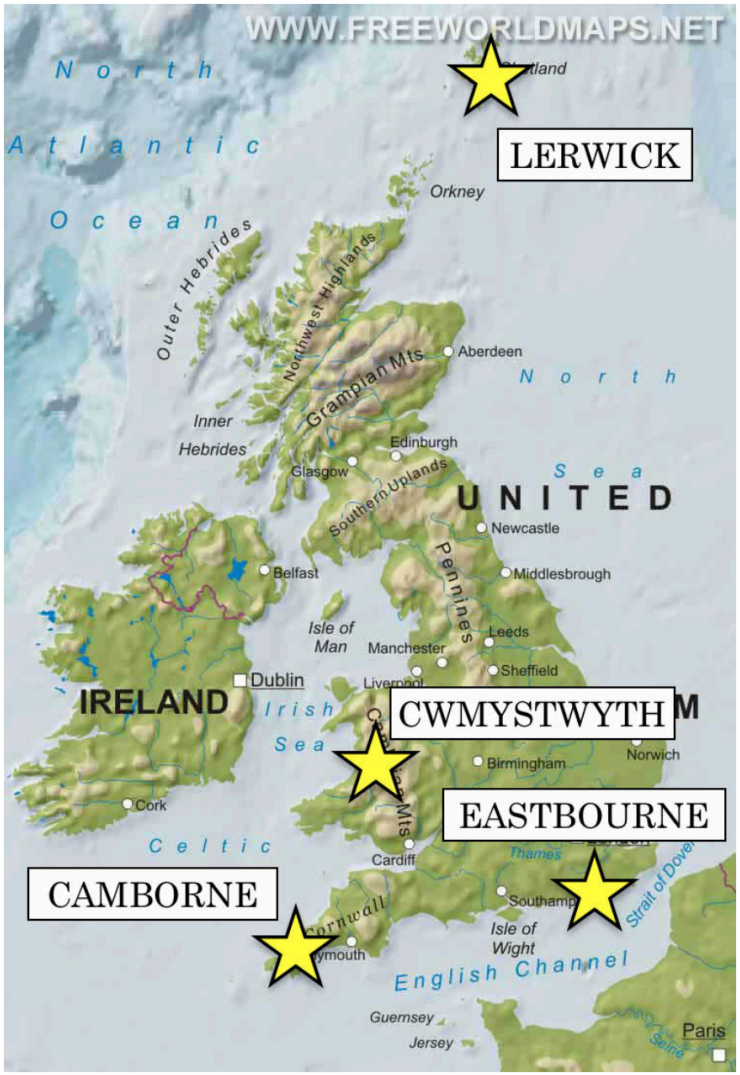
Python libraries: `pyTensorFlow`

Glossary: Classification terminology

Target (Class) The different types of thing we are trying to classify. This is an input for training a classifier and an output for our the test data.
Examples: [galaxy, star] [cat, chicken, horse] [calcification, pacemaker]

Features The labels for the data points that we're using as input to the classification.
Examples: [distance, size, brightness] [height, legs, tail length] [width, length, density]

Weather Example



Target (Class) [Camborne, Cwmystwyth, Eastbourne, Lerwick]

Features [maximum temperature, minimum temperature, air frost, rainfall, hours of sunshine]

The Data

Camborne
Location 162700E 40700N, Lat 50.218 Lon -5.327, 87m amsl
Estimated data is marked with a * after the value.
Missing data (more than 2 days missing in month) is marked by ---.
Sunshine data taken from an automatic Kipp & Zonen sensor marked with a #, otherwise sunshine data taken from a Campbell Stokes recorder.

yyyy	mm	tmax	tmin	af	rain	sun
		degC	degC	days	mm	hours
1978	9	17.5	11.3	0	26.7	---
1978	10	15.6	10.7	0	20.4	---
1978	11	12.6	7.6	0	56.3	---
1978	12	9.2	5.0	5	276.7	---
1979	1	6.5	0.9	13	134.8	---
1979	2	6.7	1.9	5	133.0	---
1979	3	8.8	3.6	2	143.8	105.0
1979	4	10.6	5.5	0	65.9	161.1
1979	5	12.4	5.8	0	82.3	227.0
1979	6	16.0	10.4	0	43.2	192.5
1979	7	18.2	12.1	0	27.9	198.9
1979	8	17.3	11.8	0	78.6	186.6
1979	9	16.3	10.9	0	40.0	145.3
1979	10	14.8	9.6	0	141.6	106.4
1979	11	11.3	7.1	0	83.1	65.4
1979	12	9.9	5.6	2	215.2	43.5
1980	1	7.3	2.0	9	120.3	69.4
1980	2	10.0	5.8	0	160.2	79.3

The Data

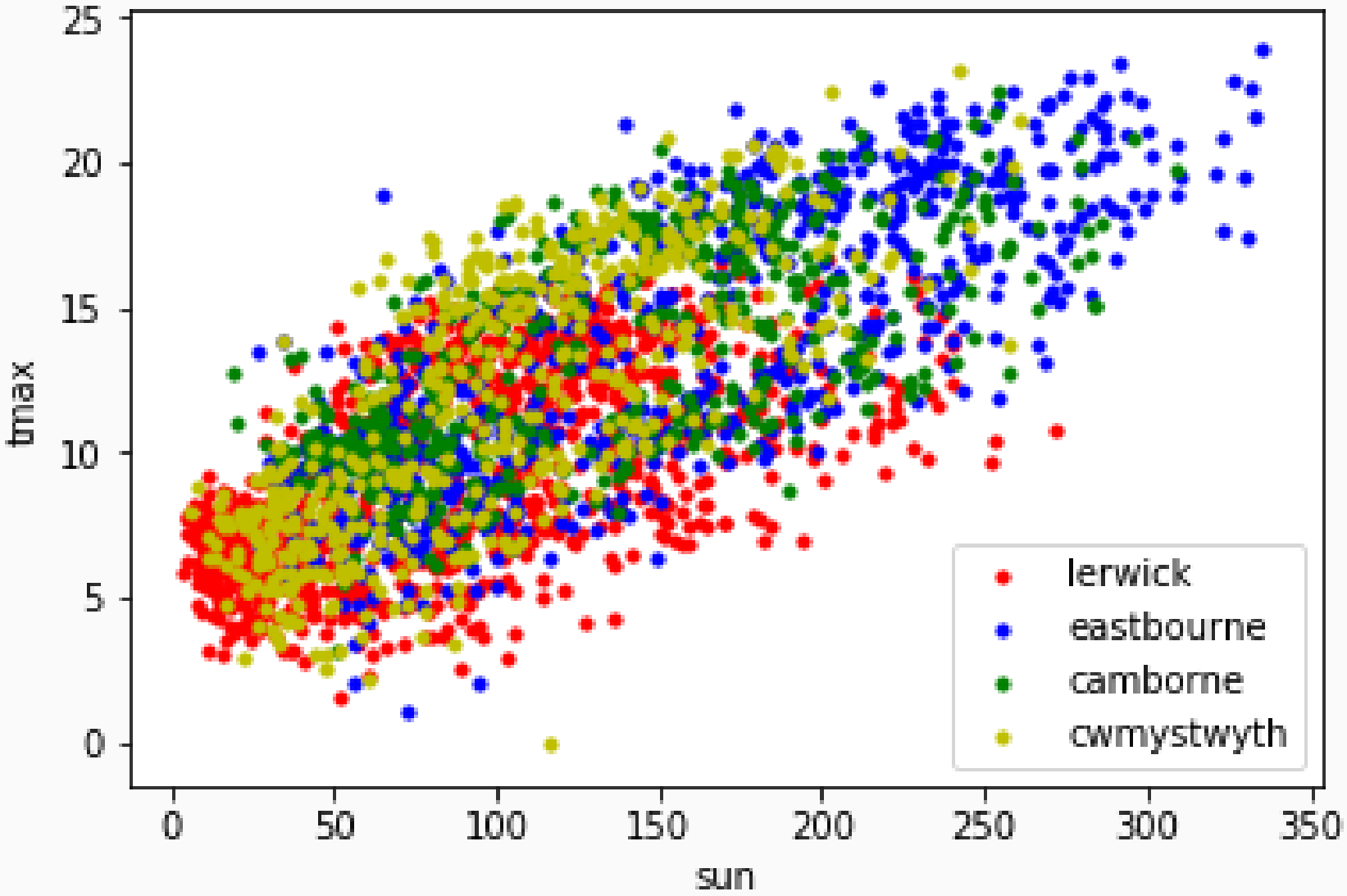
Camborne
Location 162700E 40700N, Lat 50.218 Lon -5.327, 87m amsl
Estimated data is marked with a * after the value.
Missing data (more than 2 days missing in month) is marked by ---.
Sunshine data taken from an automatic Kipp & Zonen sensor marked with a #, otherwise sunshine data taken from a Campbell Stokes recorder.

Meta-data

yyyy	mm	tmax degC	tmin degC	af days	rain mm	sun hours
1978	9	17.5	11.3	0	26.7	---
1978	10	15.6	10.7	0	20.4	---
1978	11	12.6	7.6	0	56.3	---
1978	12	9.2	5.0	5	276.7	---
1979	1	6.5	0.9	13	134.8	---
1979	2	6.7	1.9	5	133.0	---
1979	3	8.8	3.6	2	143.8	105.0
1979	4	10.6	5.5	0	65.9	161.1
1979	5	12.4	5.8	0	82.3	227.0
1979	6	16.0	10.4	0	43.2	192.5
1979	7	18.2	12.1	0	27.9	198.9
1979	8	17.3	11.8	0	78.6	186.6
1979	9	16.3	10.9	0	40.0	145.3
1979	10	14.8	9.6	0	141.6	106.4
1979	11	11.3	7.1	0	83.1	65.4
1979	12	9.9	5.6	2	215.2	43.5
1980	1	7.3	2.0	9	120.3	69.4
1980	2	10.0	5.8	0	160.2	79.3

Data

The Data


25
20
15
10
5
0
tmax
0
50
100
150
200
250
300
350
sun
lerwick
eastbourne
camborne
cwmystwyth

Classification with scikit_learn

```
from sklearn.ensemble import RandomForestClassifier as RFC
```

```
import pandas as pd  # for data formatting
```

```
feature_names=np.array(['year','month','tmax','tmin','af',
                                                'rain','sun'])
target_names=np.array(['lerwick','eastbourne','camborne',
                                                'cwmystwyth'])
```

Classification with scikit_learn

```
df = pd.DataFrame(data, columns=feature_names)
```

```
df['places'] = pd.Categorical.from_codes(target, target_names)
```

Glossary: Classification terminology

Training Data data used to train the weights on the classifier (70%)

Test Data data used to test the trained classifier and confirm the predictive power (10%)

Validation Data data used to check for overfitting (20%)

Classification with scikit_learn

```
frac = 0.75
df['is_train'] = np.random.uniform(0, 1, len(df)) <= frac
```

```
train, test = df[df['is_train']==True],
                                    df[df['is_train']==False]
```

Classification with scikit_learn

The first two data columns are the year and month of the measurement. They probably (*) aren't relevant to the classification, so we'll exclude them:

```
features = df.columns[2:7]
```

```
forest = RFC(n_jobs=2,n_estimators=100)
```

Classification with scikit_learn

```
y, _ = pd.factorize(train['places'])
```

```
forest.fit(train[features], y)
```

```
Out[28]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=2,
                                oob_score=False, random_state=None, verbose=0,
                                warm_start=False)
```

Classification with scikit_learn

```
preds = target_names[forest.predict(test[features])]
```

Confusion matrix:

True → Predicted ↓	Camborne	Cwmystwyth	Eastbourne	Lerwick
Camborne	41	1	27	22
Cwmystwyth	2	95	4	10
Eastbourne	35	7	113	26
Lerwick	30	19	46	218
	108	**122**	**190**	**276**

Classification with scikit_learn

		Truth	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive (FP) Type I Error
	Negative	False Negative (FN) Type II Error	True Negative (TN)

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

...and many others.