Classification

Abbey Waldron

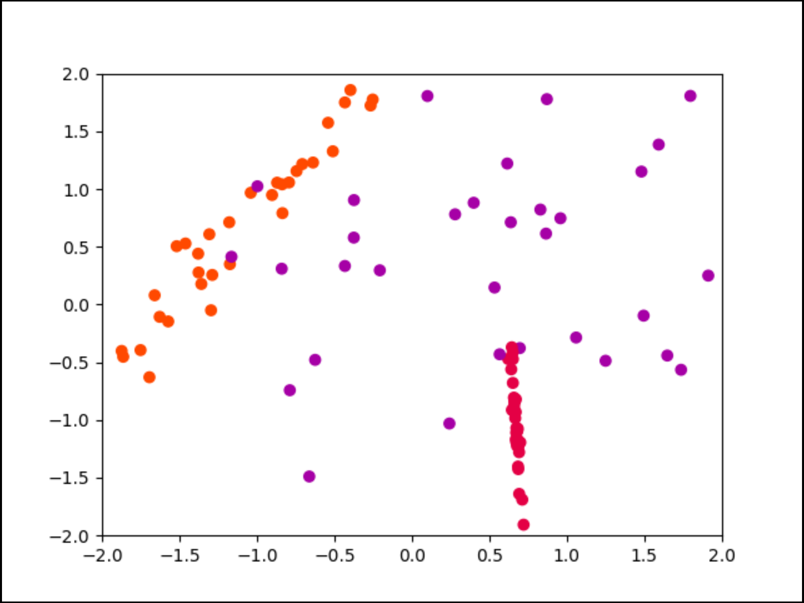
www.computerscienceretreat.com

Classification

Now we have discrete class values to predict! For example:

- Images $\rightarrow$ labels
- Sounds $\rightarrow$ words
- Demographic info $\rightarrow$ party voted for

Data is labelled



How to Evaluate Classification

Precision and Recall

```
from sklearn.metrics import classification_report
print(classification_report(y,y_pred))
```

Precision

$$\text{precision} = \frac{\text{true}_s}{\text{true}_s + \text{other}_s}$$

Where *s* means selected, true are the ones you are trying to select and other is anything else. Basically what fraction of the things you think you have found are really what you were looking for.

Recall

$$\text{recall} = \frac{\text{true}_s}{\text{true}_s + \text{true}_u}$$

Where *s* means selected, *u* means unselected. Basically what fraction of the things you are looking for you manage to find.

What to maximise?

$$F_\beta = (1+\beta^2)\frac{\text{precision} \times \text{recall}}{(\beta^2 \times \text{precision}) + \text{recall}}$$

Higher weight to recall $\rightarrow \beta = 2$

Higher weight to precision $\rightarrow \beta = 0.5$

Step 1: Scale data!

```
from sklearn.preprocessing import scale
X = scale(X)
```

Classification Algorithms

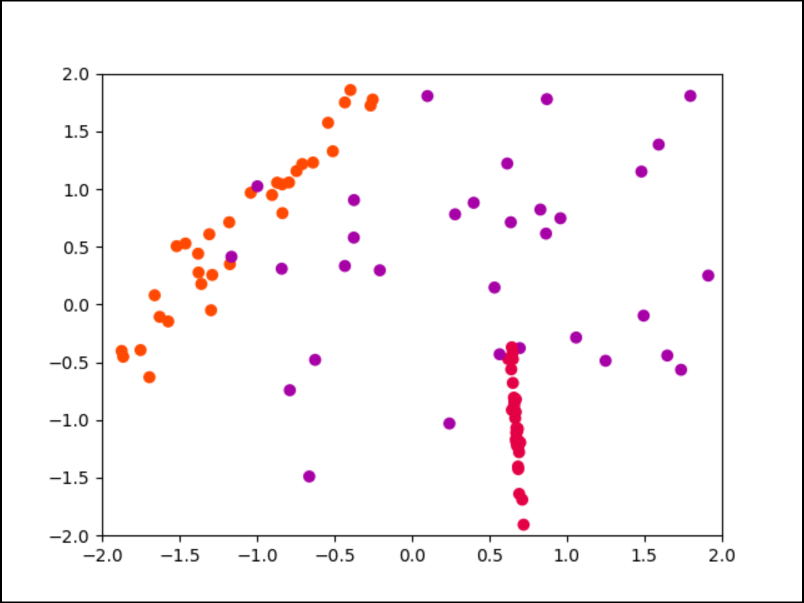
There are many!

- kNN
- Logistic regression
- SVM and kernalised SVM
- Decision trees
- Ensembles such as random forests
- Neural Networks
- …

Check out the scikit learn documentation!

k Nearest Neighbours

Let's take our 2 dimensional example with three labelled classes.


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

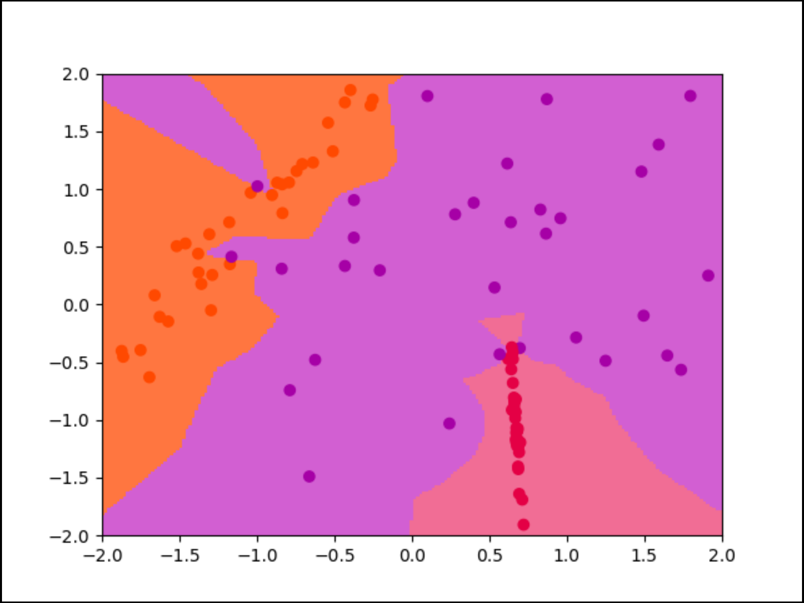
For each point in the space we want to assign a class value, 1, 2 or 3.

First consider $k = 1$ (one neighbour!)

For each point in the space we look for the closest training example (point on the plot).

That is its class.

k Nearest Neighbours, k=1



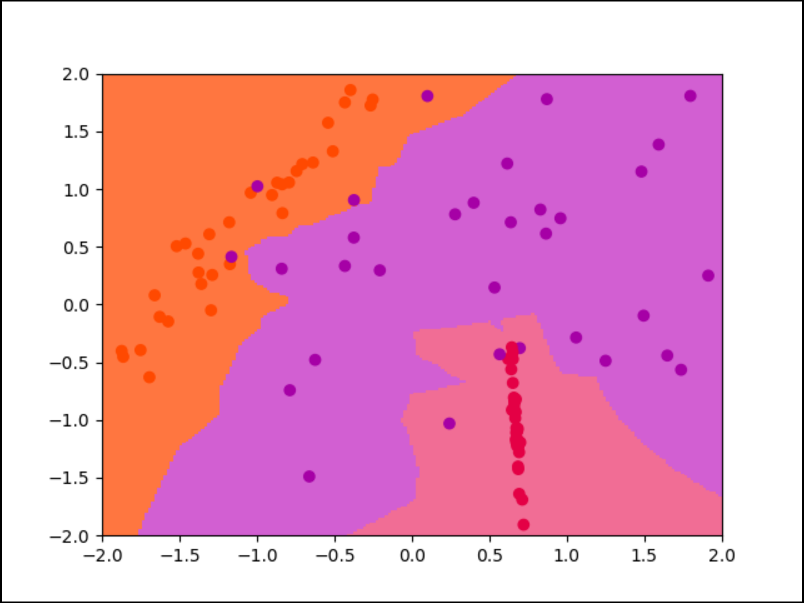
k should be higher

In general $k = 1$ will overfit our data.

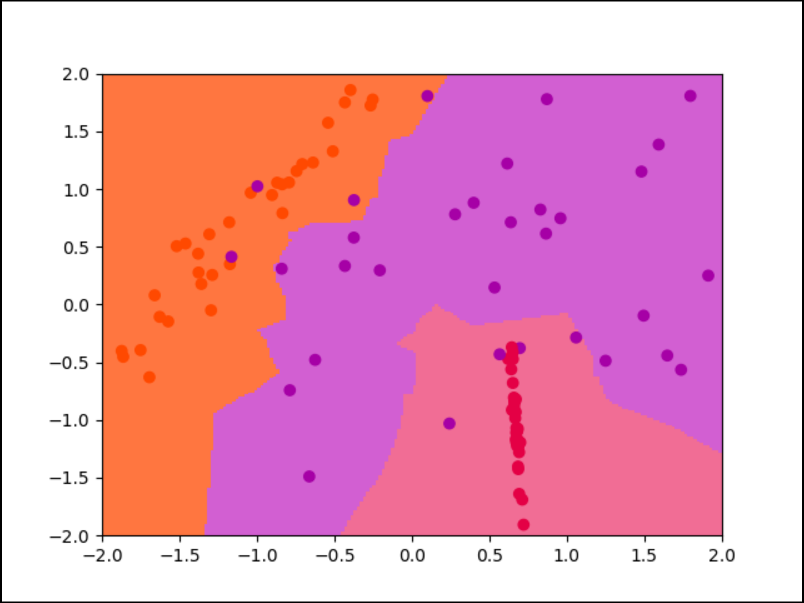
Let's increase the value of k to three, then consider the three closest points, and vote for a winning class!

Odd numbers make sure you'll get a winner.

k Nearest Neighbours, k=3



k Nearest Neighbours, k=5



Code for kNN

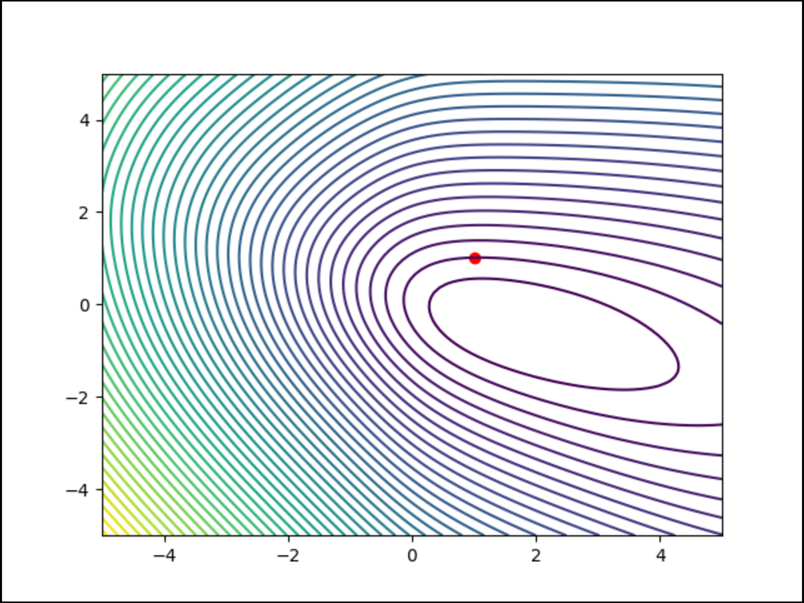
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import scale
X = scale(X)
model = KNeighborsClassifier(n_neighbors=5)
model = model.fit(X,y)
```

Logistic Regression

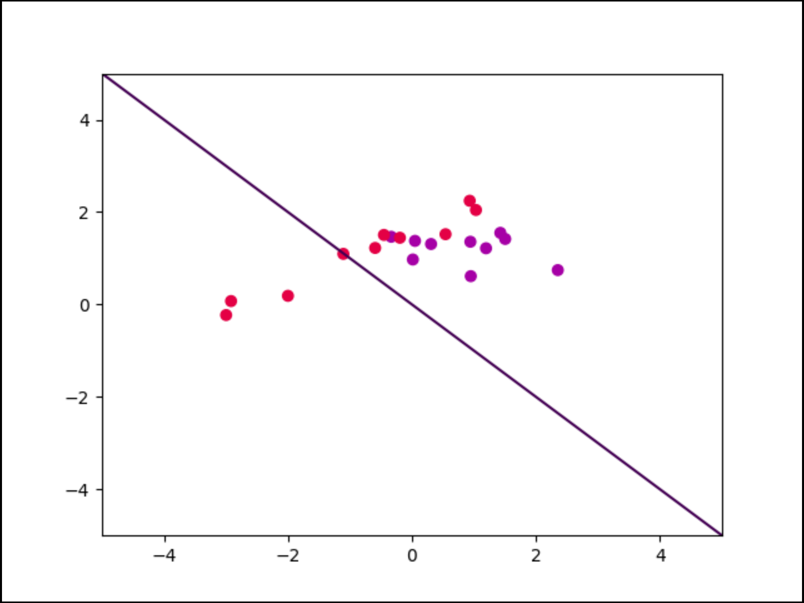
$$h_\theta(x) = \frac{1}{1 + e^{-\theta x}}$$

$$J\left(h_\theta(x), y\right) = -\left(y \log h_\theta(x) + (1 - y) \log\left(1 - h_\theta(x)\right)\right)$$

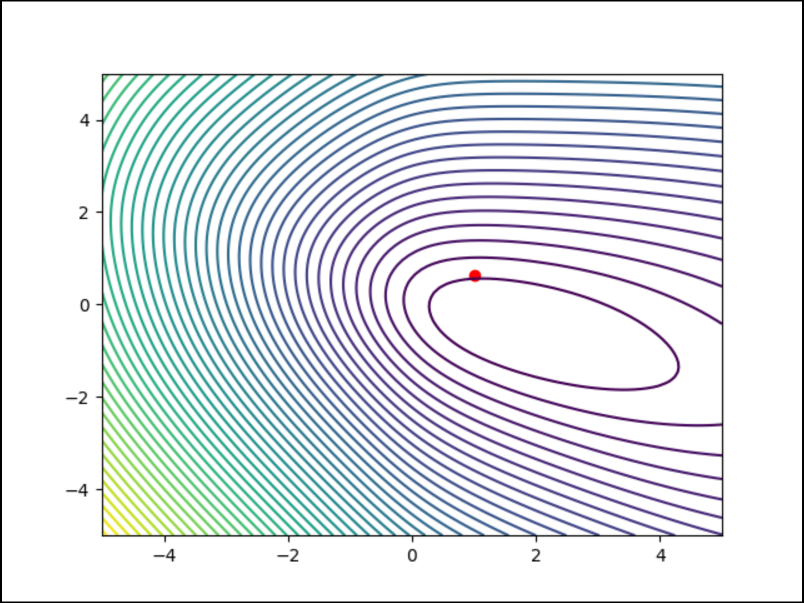
Gradient Descent



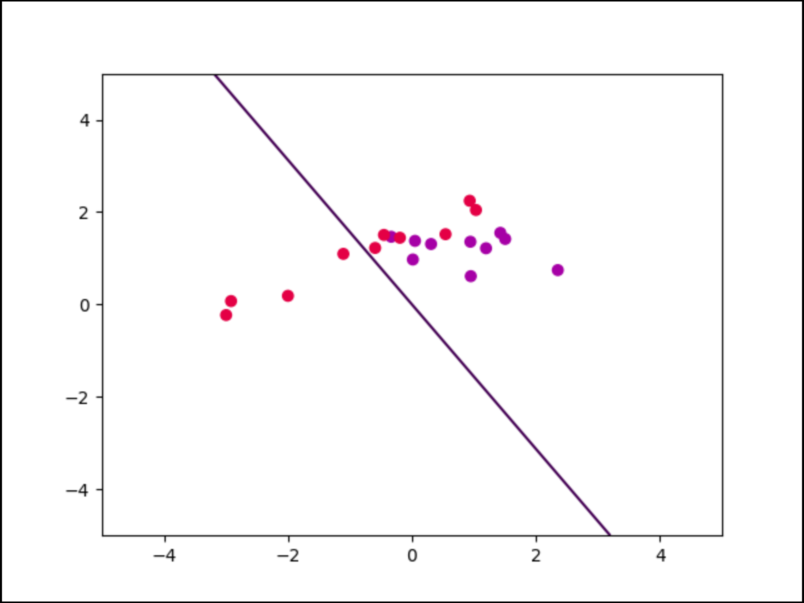
Gradient Descent



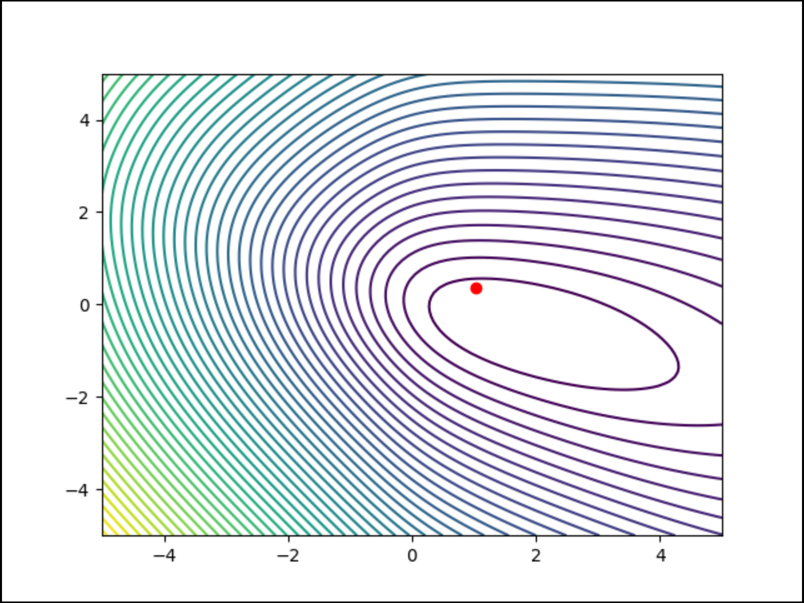
Gradient Descent



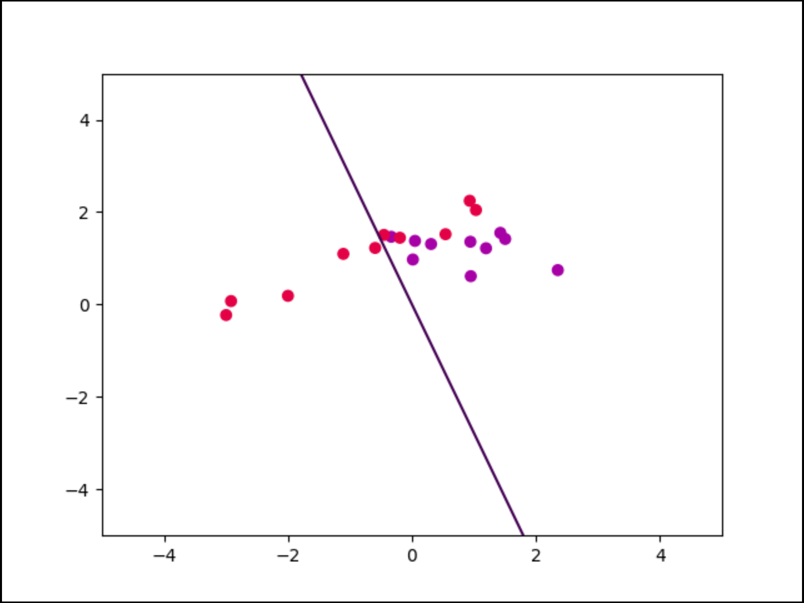
Gradient Descent



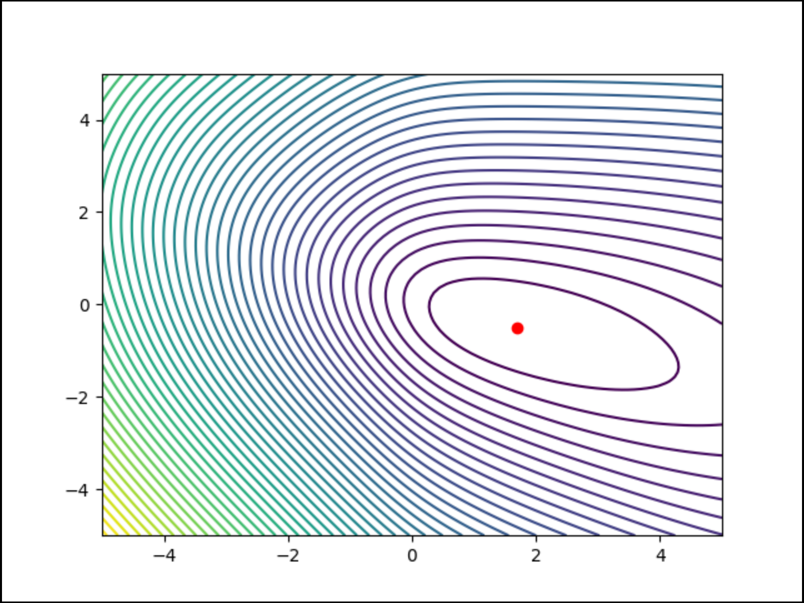
Gradient Descent



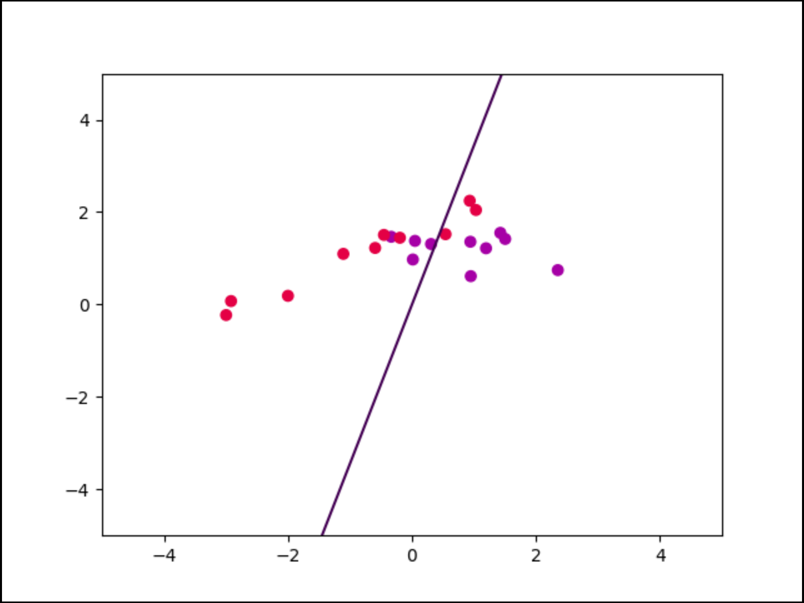
Gradient Descent



Gradient Descent



Gradient Descent



Code for Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import scale
X = scale(X)
model = LogisticRegression()
model = model.fit(X,y)
```

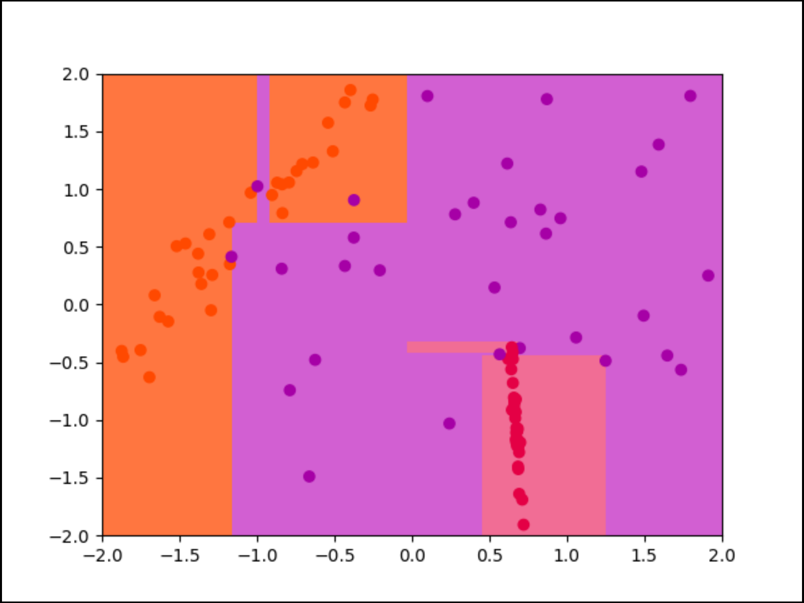
Decision Tree

Imagine writing a series of if/else statements in your code to separate your classes.

`If` $\texttt{x}_0 < 5$ `and` $\texttt{x}_1 > 7$ `then…`

End up with lots of horizontal and vertical decision boundaries

Decision Tree



Code for Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model = model.fit(X,y)
```

Tips for Classification

If things are going wrong…

- Check you have scaled your data
- Make sure to visualise your results (always make a plot)