IOC Topic 5.1 - Introduction to Machine Learning

Transcript & Notes: PART 3

Author: Dr. Robert Lyon
Contact: robert.lyon@edgehill.ac.uk (www.scienceguyrob.com)
Institution: Edge Hill University
Version: 1.0

Topic 5, Module 1, Part 3

Introduction Slide

Hello and welcome back to Topic 5, Module 1, Introduction to machine learning. This is part 3 of the module. It will gently introduce you to some new machine learning algorithms. Part 3 will l take approximately 1 hour to complete. My name is Dr. Robert Lyon, and I'll be your guide during this module. Notes will be provided that accompany this content. I advise you to keep those near you as we move through the slides. Each slide is numbered, and this number corresponds to a page in the notes.

In the notes I'll sometimes encourage you to undertake some optional self-study. These self-study opportunities will help you understand the content presented. I'll also provide links and references in the notes that enrich the content being discussed – follow those up at your leisure.

Before proceeding I'd like to thank Dr. Swagat Kumar for writing and preparing the activities in this module.

Slide 2

Let's take a moment to review the topics this module will introduce…

- Useful terminology and key concepts. This will help you understand more advanced content as we progress.

This content has been covered in part 1.

- The mathematical background required to understand machine learning. This will be basic – the focus is on fostering an understanding, without worrying about complex equations. Our first automated learning system will follow shortly after.

This content has been covered in part 2.

- A number of machine learning algorithms from first principles, supported by examples you can try for yourself.

  This content will be covered here in part 3.

  The aim is to help you acquire the foundational knowledge required to apply machine learning in practice.

  Let's continue our journey by considering what automated machine learning is, at its core.

  Slide 3

  What we've talked about so far - this isn't really learning right? We can't say that decision stumps really learn at all.

  Instead we've reduced learning to a search for parameters, that minimise some error rate.

  Learning effectively reduced to a parameter search!

  When this works well it provides the illusion of intelligence – but this isn't how you or I learn. Let's carry on and consider some other classification algorithms.

  Slide 4

- Classification algorithms come in all shapes and sizes.
- Despite their differences, inside they are comprised of functions that attempt to perform the mapping from examples to labels. They mostly differ in how that mapping is actually done.
- What they have in common:
  - All are evaluated against some *feedback* metric, which guides their learning - call this the error rate.
  - They must choose (find) parameter values that minimise this error rate.

  Slide 5

  There are two main types of machine learning classifier.
  Discriminative approaches, and Generative approaches.

Discriminative Approaches

- Learn the differences between the examples in a training set.
- Use those differences to classify and make predictions.
- We often think similarly, like in the Fossa example.

Generative Approaches

- Learn a "model" of the classes in the training set.
- This model can be then compared against.
- We don't usually think this way, e.g. given all the cats we've seen, what is the probability that this animal is a cat?

There is an example in the notes that may help you better understand the difference between these approaches.

Additional Notes:
Below is a story that helps explain the difference between Discriminative and Generative learning. I didn't write this, so thank you to the author credited below.
Credit: https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3

A father has two kids, Kid A and Kid B. Kid A has a special character whereas he can learn everything in depth. Kid B have a special character whereas he can only learn the differences between what he saw.

One fine day, the father takes two of his kids (Kid A and Kid B) to a zoo. This zoo is a very small one and has only two kinds of animals, say a lion and an elephant. After they came out of the zoo, the father showed them an animal and asked both of them "**is this animal a lion or an elephant**?"

The Kid A, the kid suddenly draws the image of lion and elephant in a piece of paper based on what he saw inside the zoo. He compared both the images with the animal standing before and answered based on the **closest match** of image & animal, he answered: "The animal is Lion".

Here Kid B knows only the differences, based on **different properties learned**, he answered: "The animal is a Lion".

Here, we can see both of them is finding the kind of animal, but the way of learning and the way of finding answer is entirely different. In Machine Learning, we generally call Kid A as a Generative Model & Kid B as a Discriminative Model.

Slide 6

Let's consider an example of Generative decision making. We're given some information regarding a disease, and then we're asked to build a classifier that can predict if someone is afflicted or not.

The prevalence of the disease in the group of patients described by the prior probability. The prior probability can be read as meaning the probability the disease occurring, without knowing any additional information.

$$P(disease) = 0.1$$

This prior is equal to 0.1 which corresponds to 10%.

Suppose there is some symptom associated with the disease. The symptom is "red spotty rash". The likelihood of having this symptom, irrespective of whether or not you have the disease, is 0.1 or 10%.

$$P(symptom) = 0.1$$

The likelihood of having this symptom, if afflicted with the disease, is given by the conditional probability. This can be read as, the probability of having the symptom, if afflicted by the disease. Here we see that 80% of the time, patients with the disease exhibit the symptom.

$$P(symptom|disease) = 0.8$$

So far, we have some basic information. But our classifier needs to determine if a patient has the disease, given some set of symptoms. How to do this? It will vary for each patient, as they may have different symptoms.

Thankfully there is a famous formula known as Bayes' Theorem, that allows us to calculate this. For now, we'll just use the theorem, but the notes contain resources that will help you understand it further.

$$P(hypothesis|data) = \frac{P(data|hypothesis) \times P(hypothesis)}{P(data)}$$

We can plug in our probabilities and begin computing the answer.

$$P(disease|symptoms) = \frac{0.8 \times 0.1}{0.1}$$
$$= 0.80 = 80\%$$

We can see that if you have a red spotty rash, there is an 80% chance that you have the disease in question.

Additional Notes:
A link to a useful resource that explains probability concepts: https://www.youtube.com/watch?v=OqmJhPQYRc8

Slide 7

If the prior is updated, we get a very different estimate for the likelihood of the disease.
Click
We're given a new prior for a specific age group cover 40 to 60 year olds. The probability of the disease in this group is low, at 0.1%.

$$P(disease) = 0.001 = 0.1\%$$

We can recompute our prediction using Bayes' theorem. We find that for this new age group, the likelihood of having this disease, even in the presence of a red spotty rash, is extremely low at just 0.8%.

$$P(disease|symptoms) = \frac{0.8 \times 0.001}{0.1}$$

$$= 0.8\%$$

Perhaps this makes more sense if I tell you that the mystery disease is the common childhood ailment, Chicken Pox.

Generative learning can be used very effectively, so long as the data provided is unbiased, complete and not misleading.
It may not be obvious from this example, but such methods can be easily used to make predictions. All you have to do is couple the outputs of Bayes' theorem, with an output threshold. For example, if the probability output by the theorem is greater than say, 70%, predict disease. Otherwise predict no disease.

We've actually just learned our next ML algorithm - Naïve Bayes. It makes predictions using probability in exactly this way. It learns by trying to find an optimal threshold over the output probability that achieves the minimum error rate.

Slide 8

- We'll try and explore how Naïve Bayes via some examples.
- We can work through 2 of the examples manually.
- The final example requires you to run some code using Python, Jupyter notebooks, and the Scikit-Learn API.

Slide 9

First a diversion. What is Scikit-Learn?

- Machine Learning software Library
- It can be used via interaction with the Python Programming language. We can see an example of scikit-learn being used in the video provided.
- Scikit-learn has been transformational for machine learning researchers. It's helped make our science results easily sharable, and reproducible. Scikit-learn doesn't stand on it's own. It's part of the wider Python software ecosystem.
- Requires some programming experience, but you don't need to be an expert.
- Pre-requisites can be tricky to understand for beginners. For this reason, we provide links to guides in the notes.

Additional Notes:

You may be wondering how to install Python/Scikit-learn on your personal machine. In recent years the Anaconda Package manager has made use of such tools even easier. It's for this reason that I can't stress enough that it's a no-brainer to take advantage of these tools. To make the most of scikit-learn, I recommend:

- Downloading and installing Anaconda for which ever version of Python your familiar with.
- Using Anaconda to install Numpy, Scipy, Matplotlib, and scikit-learn packages. Anaconda will sort this out for your, and automatically install/configure all prerequisites.
- Install jupyter notebooks (if not already installed by Anaconda).
- Use Jupyter notebooks to interact with scikit-learn.

Slide 10

Before we run some code, let's explore some of the ideas from the last slide.

- Suppose we have a bag containing 8 marbles.
- 3 marbles are white, 3 are blue and 2 are yellow.

1. What is the chance of randomly picking out a blue marble, i.e. what is $\boldsymbol{P(marble = blue)}$? Well, this is $\frac{3}{8}$.
2. What is the chance of picking a white marble, if we already picked a yellow marble out the bag: $\boldsymbol{P(marble = white|\ yellow\ marble\ picked)}$?

   Well to answer this we need to know:

   $$P(yellow) \times\ P(white|yellow\ marble\ picked).$$

   If the probability of yellow, $P(yellow) = \frac{2}{8}$

   and the chance of picking a white marble after a yellow marble has been picked is $P(white|yellow\ marble\ picked) = \frac{3}{7}$.

   then we can work this out as $\frac{2}{8} \times \frac{3}{7} = \frac{3}{28} = 0.107\ to\ 3.d.p.\ = approx\ 10\%$ chance.

   Additional notes:
   Useful link that describes conditional probability:
   https://www.mathsisfun.com/data/probability-events-conditional.html

Slide 11

We have data describing the use of sports facilities, along with the corresponding weather. In particular it records if there was play on the tennis court, according to the weather.

This data is shown in the table on the far left of the slide. We can see it contains two features, weather and play. Weather is a categorical feature with 3 possible values – Sunny, Overcast, Rainy. Play is a binary feature with two possible values – yes or no. There are 14 examples in the table.

We can create a frequency table, containing the number of times there was play for each type of weather. This shows for example, that when the weather was overcast, there was never any play. We can also deduce from this table how many times each type of weather occurred. It was Overcast 4 times, Rainy 5 times, and Sunny 5 times.

We can also deduce that there was play 9 times, and no play 5 times.

From these counts, we can calculate the likelihood of specific weather, as well as play vs. no play. These are the prior probabilities. We can see the prior probability of rain is 0.36 or 36%.

Suppose it's sunny today. What is the chance that there will be play today: $P(play|sunny)$?

We can use Bayes theorem to determine this.

$P(play|sunny) = \frac{P(sunny|play) \times P(Play)}{P(sunny)}$

We can fill in the values. We know that the prior probability of sunny weather is $\frac{5}{14}$, the prior probability of play is $\frac{9}{14}$, and the conditional probability of it being sunny if there is play going on is $\frac{3}{9}$.

Now we plug in the values and calculate the answer.

$$\frac{\frac{3}{9} \times \frac{9}{14}}{\frac{5}{14}} = \frac{0.333 \times 0.643}{0.357} =$$

$0.600\ to\ 3.d.p. = approx\ 60\%$ chance.

We find that the probability of play happening if it is sunny, is 60%. So based on that information, do we predict play if we have sunny weather?

Additional Notes:

Conditional probability link: https://www.mathsisfun.com/data/probability-events-conditional.html

Slide 12

Before we decide, let's check the probability of no play, if it's sunny outside.

We know that the prior probability of sunny weather is $\frac{5}{14}$, the prior probability of no play is $\frac{5}{14}$, and the conditional probability of it being sunny if there is no play going on is $\frac{2}{5}$.

Now we plug in the values and calculate the answer.

$$\frac{\frac{2}{5} \times \frac{5}{14}}{\frac{5}{14}} = \frac{0.400 \times 0.357}{0.357} =$$

$0.400\ to\ 3.d.p. = approx\ 40\%$ chance.

We find that the probability of no play happening if it is sunny, is 40%.

Given that the probability of no play, is less than the probability of play, in this instance we predict play.

Slide 13/14

- Go to the provided link: https://colab.research.google.com/
- Login using your Gmail ID.
- Try to open a new file. Look for the file option and then click "New Python 3 Notebook".
- Familiarize yourself with Jupyter Notebooks (if you haven't already). Go through the Jupyter tutorials – the links are provided in the module notes for this slide. if unfamiliar with Jupyter. Also learn about markdown language.
- We'll use the Python Scikit APIs mentioned earlier, to build the machine learning models in this exercise.
- Execute the examples given in the file: IOC-ML-Exp.pynb You'll find the link to this file in the module notes for this slide.
- Note that all the all practical examples will be in this one file.
  Additional Notes:
  Here are the useful links you'll need to pursue the practical examples. Take your time moving through these. You may not pick things up right away, but that's entirely normal.
  Working environment: https://colab.research.google.com/
  Jupyter: https://jupyter.org/
  Jupyter notebook tutorial: https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook
  Another jupyter tutorial: https://www.dataquest.io/blog/jupyter-notebook-tutorial/
  Markdown guide: https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed
  Scikit-learn API: https://scikit-learn.org/stable/tutorial/basic/tutorial.html
  IOC-5-1-Intro-to-ML.ipynb:
  https://colab.research.google.com/drive/18oD8TPaGlaNJiGsE43TQ_8iMicHQflA7

Slide 15

- The notebook contains code that allows you to build a Naïve Bayes Classifier for a real dataset. Scroll down to the Naïve Bayes section in the notebook.
- The data we'll use contains details of the length and width of petals and sepals for various flowers. The goal is to correctly classify the petals assigning them their correct class label - setosa (0), versicolor (1), virginica (2).
- There are four features plus a class label for each example.

- Move through the cells in the notebook, executing them as you go.
- Each cell takes you through a stage in the ML process:

1. Loading the data.
2. Preparing the test / training data sets – here we split the data to build these.
3. Building the model.
4. Evaluating the model.
5. Applying the model to new data.

- Alter the settings for yourself, experiment with what happens. Change the training/test dataset split – what's the effect?

Additional Notes:
More information on the IRIS dataset: https://archive.ics.uci.edu/ml/datasets/Iris/
For more information on the classifier used in this activity, please visit the following link: https://scikit-learn.org/stable/modules/naive_bayes.html

Slide 16

- Naïve Bayes is very fast to train and predict.
- Easy to understand.
- It has few parameters that require tuning.

Slide 18

Suppose we have a dataset containing 2 classes:

- Stars
- Circles

Our goal is the separate them as accurately as possible.

One way to do this, is the expand on the idea of the decision stump. Suppose we combine multiple stumps. This would allow us to combine multiple linear boundaries.

We can see the first boundary has partitioned the data into circles and stars using some feature. Now we can add another stump to the previous to create a second boundary.

The second boundary lets us create two separate regions in the data. But our stump no longer looks like it did before – it's acquiring more of a tree-like structure.

We continue to add stumps, partitioning the data further.

With the addition of the 4th stump, we achieved something interesting. We've now created two pure regions in the top left of the plot. These regions are pure because they contain only 1 type of class. This means in this area; the new stump is now perfectly separating circles and stars. It achieves 100% accuracy on both.
We can keep adding more stumps. With each addition, we able to better separate the data. This way of learning, which involves recursively adding stumps together is known as tree learning. The combined system is known as a Decision Tree.

As you can see from the diagram, decisions trees are non-linear models. This is because they can produce non-linear separators.

Decision trees successively discriminate labelled training data into classes via partitioning. Each partition forms an extra decision boundary.

To build Decision Trees there are some key questions to answer. Which feature should each stump split upon? What split value should be used for each feature? How deep should the tree be allowed to go?

Additional Notes:
To learn more about decision trees, check the following link: https://en.wikipedia.org/wiki/Decision_tree_learning

Slide 19

At each spilt point, decision trees find the feature that maximises the separation between the classes in the data.

This is done by considering the statistical distribution of the features. The aim is the find a distribution that exhibits "separability". What does this mean? Well, we have three feature distributions shown in the top left of the slide. These distributions show the probability of a feature value occurring, for both the circle and star classes.

Feature 1's distribution is fairly separable. Hence the probability distributions for the circle and star classes don't overlap much. You can see that a linear separator easily fits between them.
The distributions for features 2 and 3 show much more overlap. They are not as separable. Of the three features, we choose the best feature for the first stump.

Just because the first feature was the most separable for the first stump, doesn't mean it will be for the second. Or for the third. Each time a new tree split is considered, all features are revaluated again for separability.
Perhaps you'll understand why if you think back to the probability we introduced for Naïve Bayes. Feature 1 works well for the first split, as without any additional information, it is the best separator.

But now as we move further down the tree, our decisions can be conditioned on previous steps. An example may help. Suppose we want to split Felines into their respective families.

The best separator at first, is mass. It allows us to efficiently split house cats form Tigers, Lions, Jaguars etc. However, some species of large cat are of similar size.

At this level trying to split Jaguars from Mountain Lions on mass won't work, as these distributions greatly overlap.

At this stage in the decision-making process, a feature such as length is more useful, because this has greater separability given what we already know – that these two types of animal are similar in terms of mass.

Decision trees search for the best feature and best feature split-point, via a mathematical optimisation method. This searches for the parameters that yield the lowest error rate.

Additional Notes:
A link with a tutorial on probability distributions: https://www.analyticsvidhya.com/blog/2017/09/6-probability-distributions-data-science/

Slide 20

We can visualise how this works. As we try different split point values, we impact the error rate. As we start to approach the optimal spilt point, the error rate drops off.

If we keep going past the optimal value, then we'll start encountering more errors again.

Slide 21/22 – practical activity

- The notebook contains code that allows you to build a decision tree for a real dataset. Scroll down to the Decision tree section.

- Here we apply the decision tree to the to the Breast Cancer Dataset.
- Step through the cells in the notebook.
- You'll find that accuracy on the test set increases after restricting the depth of the tree to 4 layers.
- Play around with various user defined parameters and see how they impact the accuracy.
- Decision trees can be susceptible to overfitting – once finished with the Decision Tree section, scroll down to the Random Forest section.
- A Random Forest is an algorithm that combines multiple Decision Trees to achieve better results.
- Execute the Random Forest cells.
- You might find that the Random Forest can overcome some of the decision tree's inherent deficiencies.

Additional Notes:

More information on the dataset:
https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)

Scikit-learns Decision Tree algorithm: https://scikit-learn.org/stable/modules/tree.html

Slide 23 – practical activity

That can be seen when studying the decision boundaries of the trees we've trained, versus the boundary produced by the Random Forest as a whole.

In this image we can see the decision boundaries of 6 different classifiers. In the top row we have boundaries produces by 3 different Decision Trees. In the bottom row from left to right we have two boundaries produced by Decision trees, and 1 produced by a Random Forest algorithm.

If you look closely, you'll find that the Random Forest has produce the better decision boundary compared to the Decision trees. You should be able to confirm this by evaluating the algorithms and determining their precise levels of accuracy.

Slide 24 – practical activity

- Compare the results of the Random Forest with those obtained using Decision trees for the Breast Cancer data.

- Compare the feature importance in both cases. One can see that more features participate in Random Forest decision making.

  Additional Notes:
  Some resources describing Random Forests:
  https://en.wikipedia.org/wiki/Random_forest
  https://www.youtube.com/watch?v=D_2LkhMJcfY
  https://towardsdatascience.com/understanding-random-forest-58381e0602d2
  https://www.datascience.com/resources/notebooks/random-forest-intro

  Slide 26 – unsupervised learning - activity

  To make a prediction for an unlabelled data point, unsupervised algorithms first group them together with the points closest to them in the training set – i.e. group according to the "nearest neighbours".

  Closest / nearest neighbours are computed using distance measures e.g. the Euclidean distance:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + \ \ldots + (q_n - p_n)^2}$$

  The Euclidean distance is easy to compute once you understand the background. Here we can see two points represented in a 2-d "Euclidean" plane. Each point can be represented by a set of coordinates. In particular, coordinates corresponding to their position on the x-axis, and position on the y-axis. The point p sits at position 1 on the x axis and position 1 on the y-axis. These coordinates can be written as (x, y) pairs that you're already familiar with.

  There is a formula that enables you to compute the Euclidean distance between two points. We can just treat this as a black box, plug in some values, and compute distances as required.

  In this example we can easily compute the distance between p and q.

$$= \sqrt{(3-1)^2 + (3-1)^2}$$
$$= \sqrt{(2)^2 + (2)^2} = \sqrt{4+4} = 2.828 \text{ to 3. d. p}$$

  The distance is around 2.828 to three decimal places. Note that I've provided links in the notes that will help you understand these concepts more clearly.

  Once we find the nearest neighbours of a data point via a method like above, we determine the most popular label amongst its labelled nearest neighbours.

  We assign the majority label of the neighbours to the unlabelled data point.

We just need to determine how many of its nearest neighbours, $\boldsymbol{k}$, we should use to decide.

Additional Notes:
Some links that help explain the background of the k-nearest neighbours' algorithm:
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
A link that helps explain the Euclidean distance:
https://en.wikipedia.org/wiki/Euclidean_distance
Video explaining Euclidean distance and geometry:
https://www.youtube.com/watch?v=nyZuite17Pc

Slide 27

We have some data containing two classes, circles and stars. We start out with two stars, and one circle described. We also have a new data point, represented by the question mark. It is unlabelled. However, using the k-nn approach, we can assign a likely label. To do this we assign the unknown point the majority label of its 3 nearest neighbours.

Thus, we predict star as the likely label for $k = 3$.

If we increase the number of neighbours under consideration to 5, things change. Now the majority label of the 5 nearest neighbours is circle.

Thus, we predict circle as the likely label for $k = 5$.

If we again increase the number of neighbours to 7, things change once more. Now the majority label of the 7 nearest neighbours is star again.

Thus we predict star as the likely label for $k = 7$.
This approach is simple, but it has proven very effective for numerous real-world problems. As you will discover, finding the best value for k is a trial and error process.
Additional Notes:
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Slide 28/29 – practical activity

- The notebook contains code that allows you to build a k-NN classifier for the IRIS dataset.
- Step through the cells in the notebook.

The program has the following parts

1. Load the data.

2. Analyse the dataset.
3. Prepare the training/ test datasets.
4. Create & Train classifier Model.
5. Evaluate the model using test set.

Additional Notes:
More information on the IRIS dataset: https://archive.ics.uci.edu/ml/datasets/Iris/

Slide 30

k-NN is easy to implement and easy to understand. Only two parameters need to be selected - number of neighbours (k) and distance measure (e.g. Euclidean distance)

It is a good baseline method to try before considering more advanced techniques.

The prediction process can be slow for very large datasets, or datasets with large numbers of features (>100).

It does not perform well with sparse datasets (where most feature values are 0).

Slide 32

Let's visualise the evaluation of machine learning algorithm. Suppose we have a labelled test dataset consisting of stars and circles as shown.

- The stars are the target items enclosed by the yellow dashed circle. We want to separate stars and circles, but only because we're trying to find stars.
- The white circle with a shaded region represents those examples some classifier has labelled as stars.
- This central region, where the two circles overlap, contains the relevant items that the classifier has correctly identified in the test set. These are known as, "True Positives".
- The left most region of the yellow dashed circle, not covered by the shaded area, contains relevant items that were not "retrieved". These represent errors. Here the classifier has predicted circle, when it should have predicted star. Such mistakes are known as "False Negatives".
- The right most region of the white circle, that doesn't overlap with the yellow dashed circle, contains irrelevant items that were "retrieved". These represent errors. Here the classifier has predicted star, when it should have predicted circle. Such mistakes are known as "False Positives".

- The remaining examples are the irrelevant items not retrieved. That is, the classifier correctly predicted circle for these items. These are known as, “True Negatives”.

By counting the outcomes, we can evaluate performance. To do this we use evaluation “metrics”. These include,

- $\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}}$
- $\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$
- $\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$

To test our algorithms, we must evaluate them using useful metrics. In machine learning we use standard metrics during evaluation. Accuracy is the most easily understood metric and is sometimes a good indicator of performance. But there are many more metrics. Here I simply introduce the three most common, along with a visual explanation for how they are derived.

Additional Notes:

Read a little more about precision and recall:
https://en.wikipedia.org/wiki/Precision_and_recall

Now try this for yourself. Suppose We classify 100 items. Of the 100, we label 80 as stars, and 20 as circles. We are then told that:
We correctly classified 75 stars as stars (True Positives = 75)
We correctly classified 18 circles as circles (True Negatives = 18)
We incorrectly classified 5 stars as circles (False Negatives = 5)
We incorrectly classified 2 circles as stars (False Positives = 2)
What is the recall, precision, and accuracy?

$$Recall = \frac{TP}{TP + FN}$$
$$= \frac{75}{75 + 5}$$
$$= 0.9375 = 93.75\%$$
$$Precision = \frac{TP}{TP + FP}$$
$$= \frac{75}{75 + 2}$$
$$= 0.974 \; to \; 3.d.p. = 94.4\%$$
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
$$= \frac{75 + 18}{75 + 2 + 18 + 5}$$

$$= 0.93 = 93\%$$

That's pretty good!

Slide 33

Suppose we have the following labelled dataset containing two classes. We want to use it for training and testing. However, we want to do this in a robust way, and we don't want our results to be overly specific to any training/test split that we may choose. So how to proceed.

1. First we randomly split the labelled data in to n equally sized portions called "folds". We ensure each fold has the correct proportion of examples for each class, when compared to the complete labelled data set. So if the ratio of stars to circles is 50:50 in the complete data set, each fold should maintain this ratio.
2. Label the folds using numbers.
3. We then use the first fold to train a model, and test on the remaining folds. In this case we train on 1 fold and test on 7.
4. Record the result achieved for the first test.
5. Now use the second fold to train the model. All other folds (including the first) are now used for testing.
6. Repeat this process until all folds have been used for training.
7. Aggregate the results on for each training fold, and compute averages. This allows a more rounded impression of performance.
   This process is known as N-fold Stratified Cross Validation.

Additional Notes:

Machine learning algorithms can be evaluated in many ways, using many different evaluation metrics. Perhaps the most trusted "standard" evaluation strategy, is Stratified n-fold cross validation. This involves testing an algorithm n times, on available data partitioned into n disjoint folds. If n=5 is used, then the data is split into 5 (maintaining the proportion of examples belonging to each class in a fold). Four of the folds are then used to train an algorithm, and the fifth fold is used to test it. The results are recorded, then the procedure repeated until all folds have been used for testing. The results on each fold are aggregated to give averaged results - hopefully indicative of real-world performance.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Slide 34

Let's summarise the main phases involved in the application of machine learning.

First, we collect our data. This step may require data normalization, data cleaning, data labelling, or even removing personal details according to ethics requirements.

Whatever the exact steps are, we should end up with a dataset ready for learning. There should be enough samples in the data set to enable it to be split into disjoint training, and test sets. The data should be representative of the concepts we're trying to learn, and we should aim to minimize the potential for bias where possible.

The next step involves feature design and extraction. This involves designing features for the data, that we believe will help us achieve accurate classification results. If we're given a dataset of animal pictures, we can extract many possible features, e.g. ***has legs***, ***number of legs***, ***number of eyes***, ***colour***, ***size*** etc.

After you've designed the features, you must extract them from the dataset, and evaluate them. You can evaluate features by quantifying their information content. Some features have higher information content than others, whilst a subset features may reveal previously hidden information when combined with seemingly unrelated features. In both cases it's good to evaluate the features to discover these relationships.

Additional Notes:
https://scikit-learn.org/stable/modules/feature_selection.html

Slide 35

Once the features have been evaluated, you can use them to start exploring which machine learning methods work best for your problem. It is desirable to try different types of learning models, evaluating their performance. Choose a model, and train it.

Evaluate the trained model on separate test data set - ensure the performance is still good. If it isn't, try a different algorithm. Repeat this exploration phase until performance is satisfactory.

When satisfied that the machine learning model works well, then you can finally deploy it upon new data.

Slide 36

We've covered a great deal in this module,

- Data sets, features, and class labels.
- Labelled and unlabelled datasets.
- Different types of learning that we're capable of.
- Bias.
- The concept of classification.
- Generalisation, and under/over fitting.
- Functions.
- Different ML algorithms: Decision Stumps, Decision Trees, Naïve Bayes, k-nearest neighbours.
- How we evaluate our algorithms.
- How to run and build some example algorithms.

We hope this inspires you to learn more. Perhaps you'll continue reading and researching the subject! To help with that, here are some resources you may find interesting.

Slide 37

There's no escaping it - Machine learning is a difficult subject. It's an area of study comprising many subfields that dip into really diverse (and often unexpected) areas. To truly understand the key concepts and apply them for yourselves, further research will be needed. To try and help you on that journey, here are links to resources that I personally have found useful in the past.

The book called "Machine Learning" by Mitchell, really helped me when I began working in this field. "Data Mining" by Witten et. al. provides a practical perspective that I also found very valuable. The book "Pattern Recognition and Machine Learning" by Bishop is an exceptional text, but math heavy. Only look here if you want an in-depth understanding of the mathematics underpinning many ML algorithms.

Here are links to the toolkits used during this module, and some we didn't have time to explore. I encourage you to research these in your own time.

Finally, I'd like to share one of my own resources that you might enjoy. This is a link to a tutorial I wrote that teaches you how to process radio astronomy data. It's written for the beginner. Perhaps after trying this tutorial, you'll be able to help astronomers find new and interesting stars with your newly acquired knowledge!

https://github.com/astro4dev/OAD-Data-Science-Toolkit/tree/master/Teaching%20Materials/Machine%20Learning/Supervised%20Learning/Examples/PPC

Slide 38

Here are some links you might find useful if wishing to learn more about machine learning. In particular there are groups geared toward supporting women to become data science experts. These include,

- Women in Machine Learning, https://wimlworkshop.org/
- Girl Geeks, https://www.girlgeeks.uk/
- Women in Data, https://womenindata.co.uk/
- Women in data science, https://www.turing.ac.uk/research/research-projects/women-data-science-and-ai
- Her + Data, http://herplusdata.org/chapters/

Maybe you'll look in to these communities in your own time.

Thank you for taking an interest in this course. I hope you enjoyed it.

Before you go, I'd like to ask if any aspects of this module were unclear. Where there any topics that you simply didn't understand? If you could let me know, I would very much like your valuable feedback - I'm keen to improve my teaching material. A contact email address is provided in the notes (robert.lyon@edgehill.ac.uk).

Best of look with the rest of your studies.