MNIST Dataset

Classification, Logistic Regression

Last few weeks

Aurélien Géron, ***Hands-on-Machine Learning***

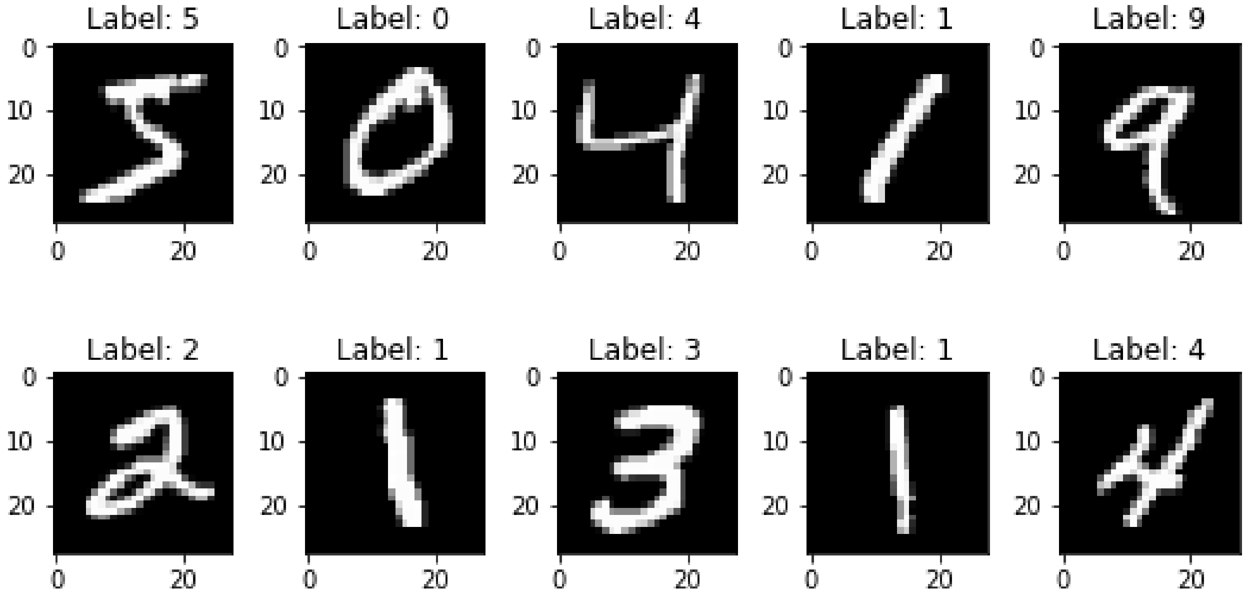
1. Look at the big picture
2. Get the data and set aside a test set
3. Discover and visualise the data to gain insights
4. Prepare the data for Machine Learning algorithms
5. Identify a suitable metric for evaluating the task
6. Select a model and train it
7. Fine-tune your model
8. Present your solution -> **Final Assignment!**
9. Launch, monitor and maintain your system -> **if ready!**

This week

- Classification
- Logistic Regression
- Performance Metrics for classification
- Classification using MNIST dataset (hand-drawn digits)
- Sklearn API

Classification deals with categorical data

- We’re used to applying **regression models** to **continuous data** to make predictions (e.g. house prices could be from 100k to 10 mil.).
- But **classification** lends itself to **categorical data** – to classify as a particular category (a value that is one of : yes/no, true/false, north/south, A/B/C/D, 0/1/2/3/4 etc)



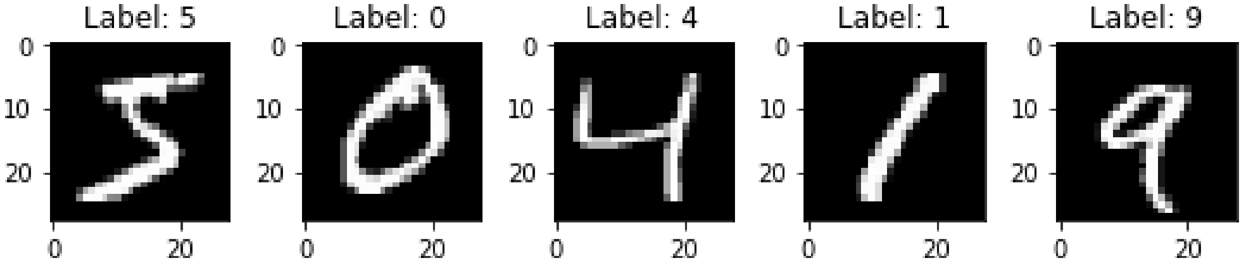
Classification tasks

- **Binary classification:** two classes (0 or 1)

Dog

Cat

- **Multi-class:** (exclusive: can only be one: this is a '9')

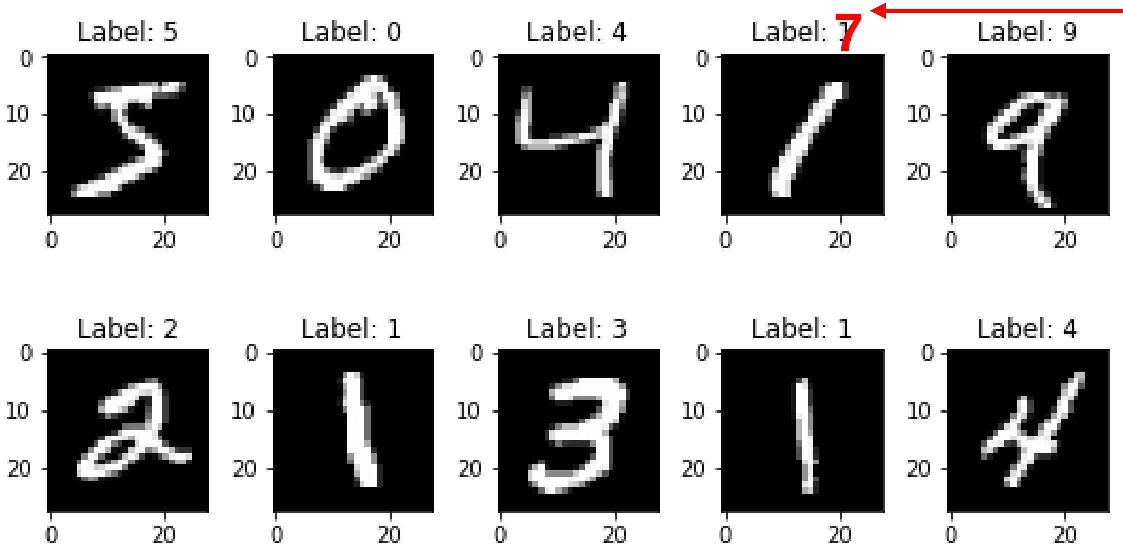


- **Multi-label:** (can belong to more than one label: Action AND Sci-fi)

5. Metrics: Accuracy

- Performance metrics are tricker for classification.
- Accuracy would be the most intuitive obvious one to measure

$$accuracy = \frac{\#\ correctly\ predicted\ records}{\#\ total\ records}$$

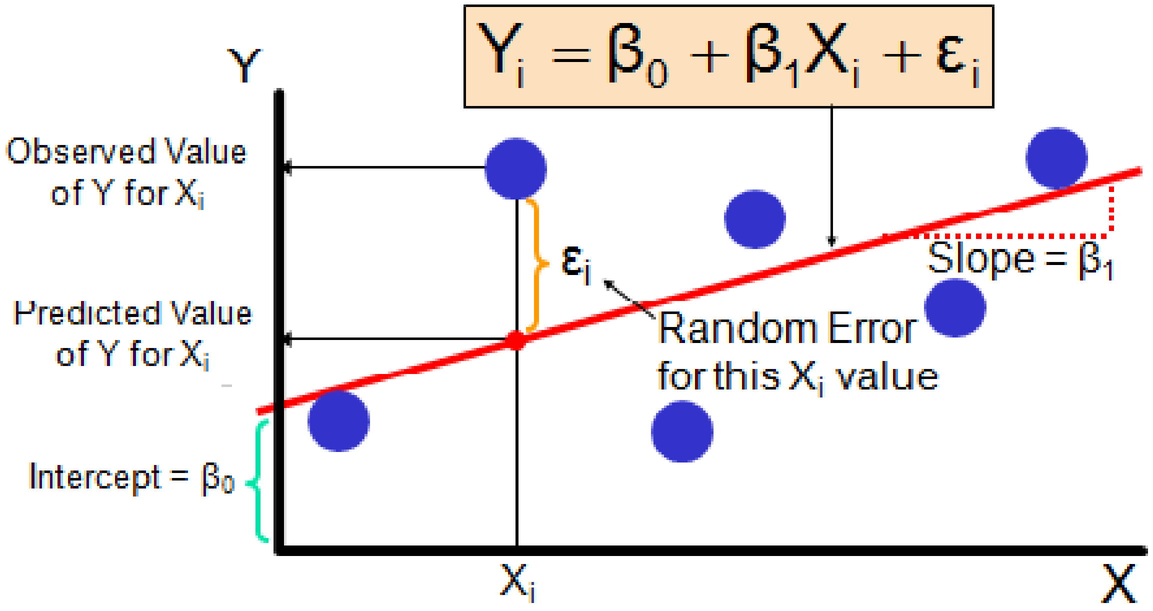


Let's say that the model incorrectly classified this image as a 7 rather than a 1

$$accuracy = \frac{9}{10} = 90\%$$

5. Identify a metric for evaluation

- **Metrics for Regression tasks**:
  - Mean Squared Error (MSE)
  - Root Mean Squared Error (RMSE)
  - Mean Absolute Error (MAE)
  - R^2 (variance explanation)

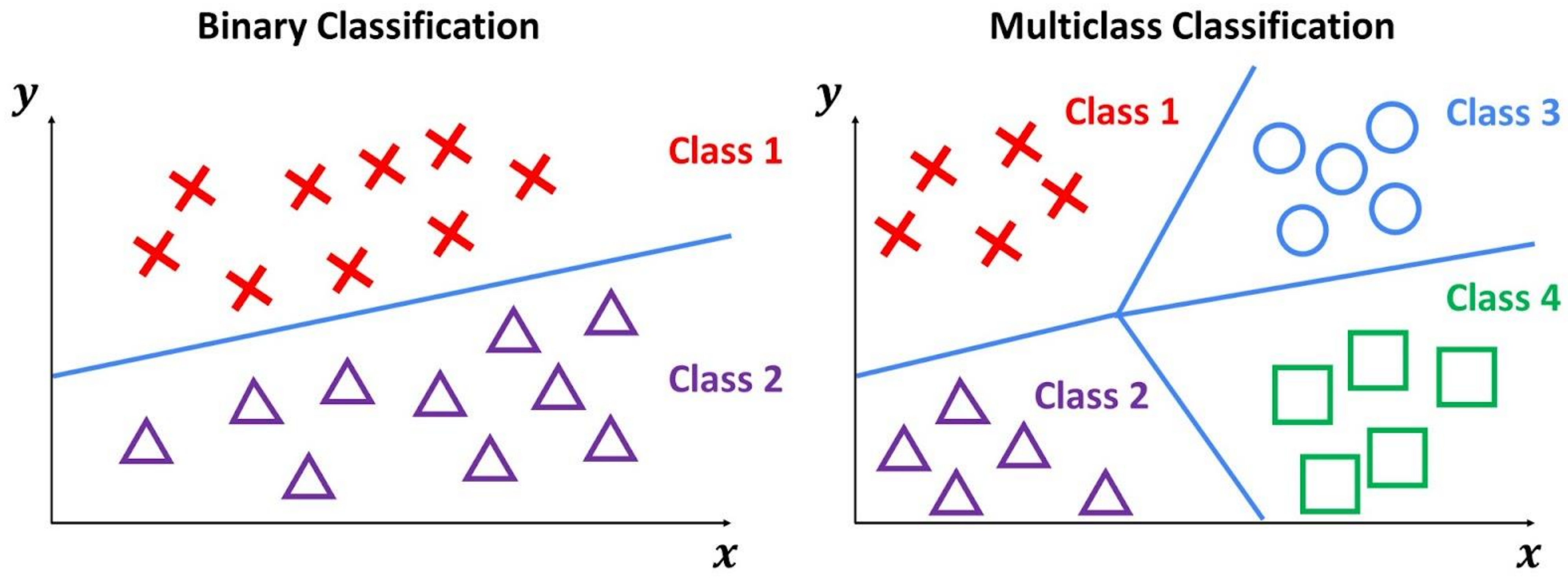


- **Metrics for Classification tasks**:
  - Precision -> F1-score
  - Recall
  - **Accuracy** (percentage correct)
  - Confusion Matrix (type I and II errors)

Actual \ **Predicted**	Spam	Non-spam
Spam	**600** (True positive)	**300** (False negative)
Non-spam	**100** (False positive)	**9000** (True negative)

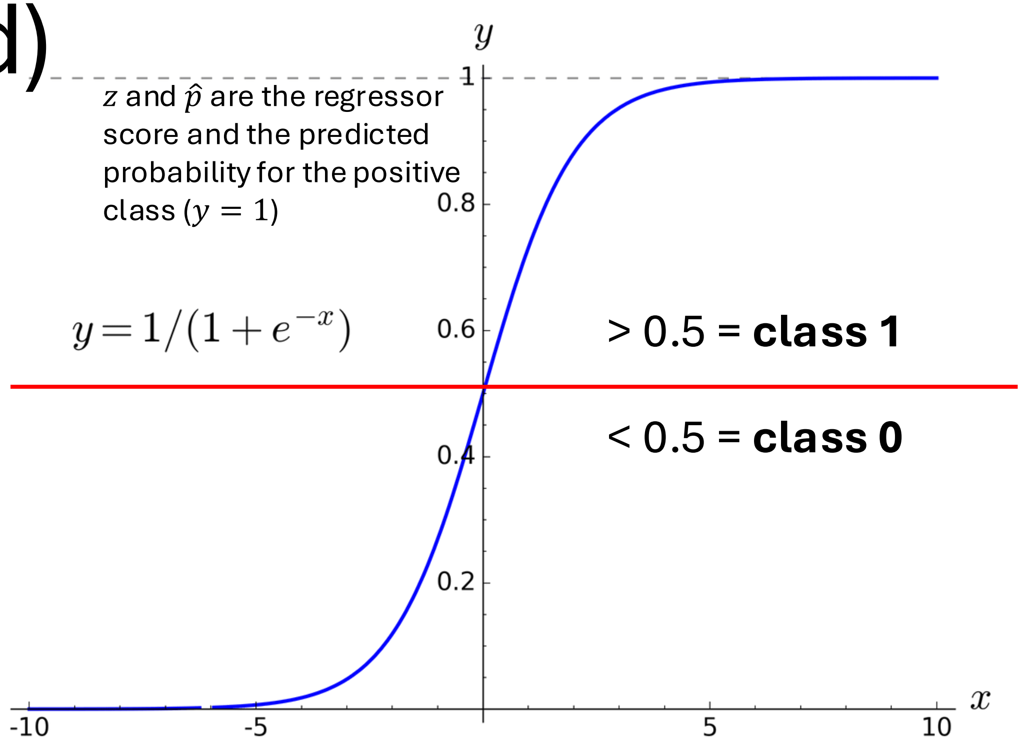
Logistic Regression

- In classification we would typically use:
  - **sigmoid** (for binary/multi-label)
  - **softmax** (for multi-class) to output probabilities.



$$\hat{p}(\boldsymbol{\theta}) = \hat{p}(\boldsymbol{w}, b) = \frac{1}{1+ e^{-z}} = \frac{1}{1+ e^{-(\boldsymbol{x}^T \boldsymbol{w}+b)}}$$

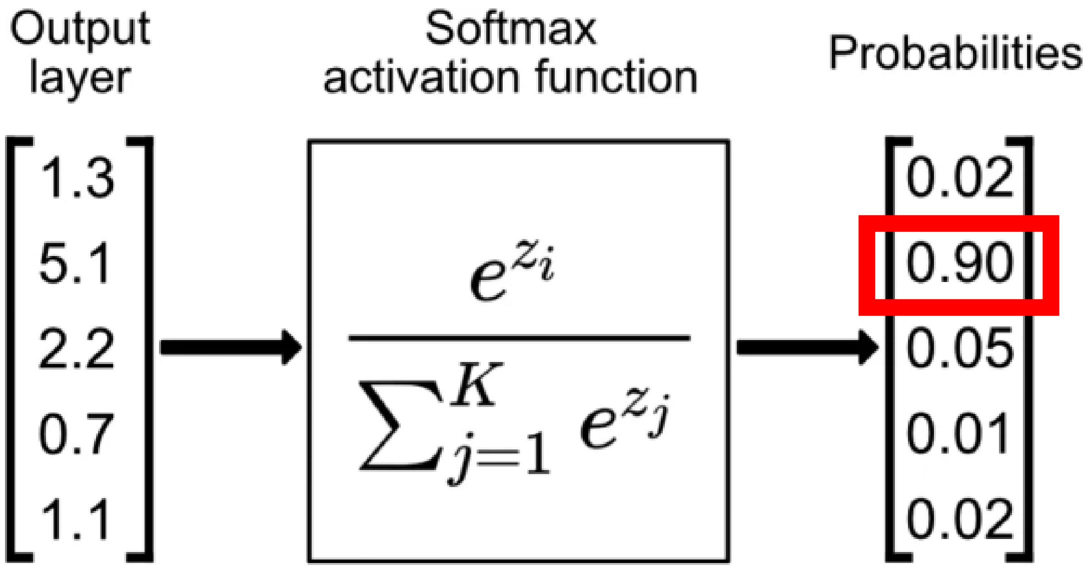
Logistic Regression (Sigmoid)



- Binary classification (class 0 or 1)
- Estimates the probability that a sample belongs to a certain class by training a (linear) regressor that will return scores in the $(-\infty, +\infty)$ interval (continuous)
- Then passing the output of the regressor to a **logistic (sigmoid) function so that the** output will be a value **between 0 and 1.**
- If the **output is > 0.5** assign to **class 1**
- If the **output is < 0.5**, otherwise assign to **class 0**

Logistic Regression (Softmax)

- Multi-class classification (classes 0, 1, 2, 3, 4 etc.)
- **The softmax function** is used in multi-class classification to convert raw scores (logits) into **probabilities** that sum to **1**.
- Then use **argmax()** to find class with highest probability score

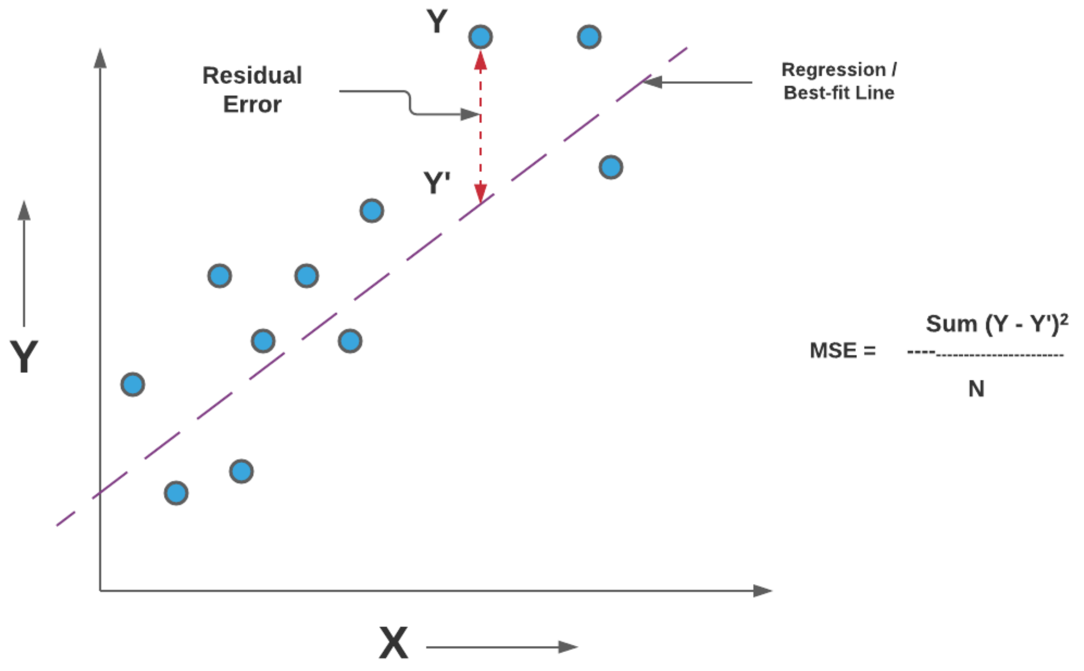


argmax(prob) = 1 (**class 1**)

5. Metrics: Mean Squared Error (MSE)

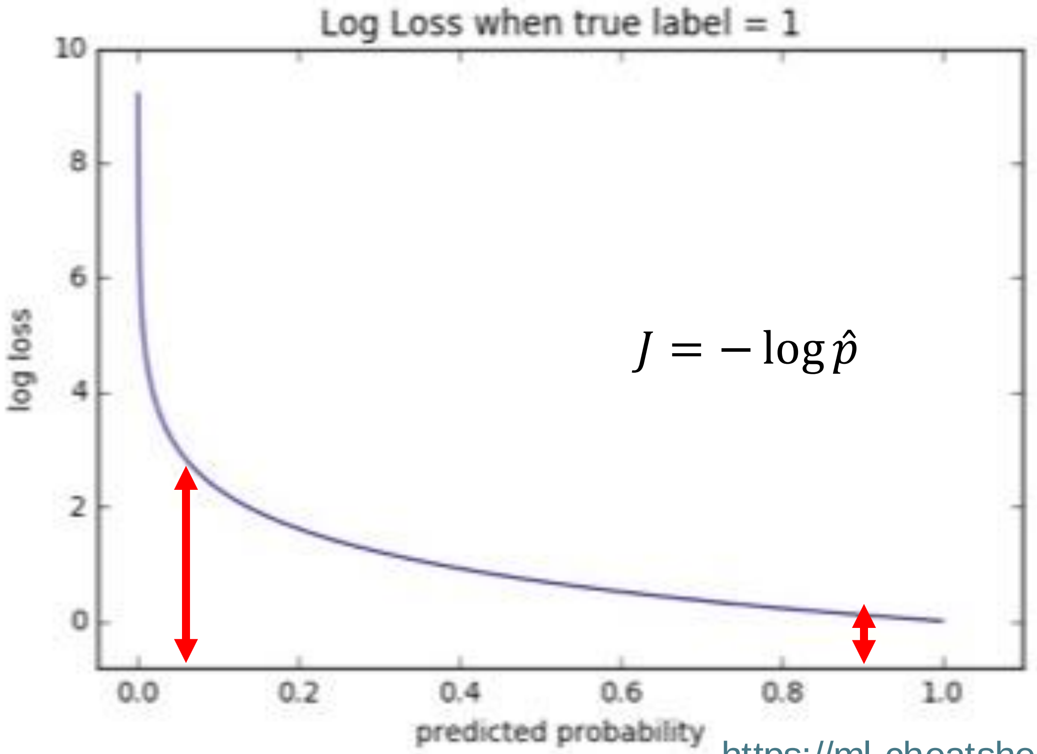
- This is the mean of the squared errors.
- Larger errors are noted more than with MAE, making MSE more popular.

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$



Log loss cost function

- Log loss creates a **logarithmic gradient**, meaning the **penalty for incorrect predictions increases sharply when the model is wrong** and decreases when it is closer to the correct probability.



https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

$$\text{crossentropy} = \mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k}^{K} y_k \log \hat{p}_k$$

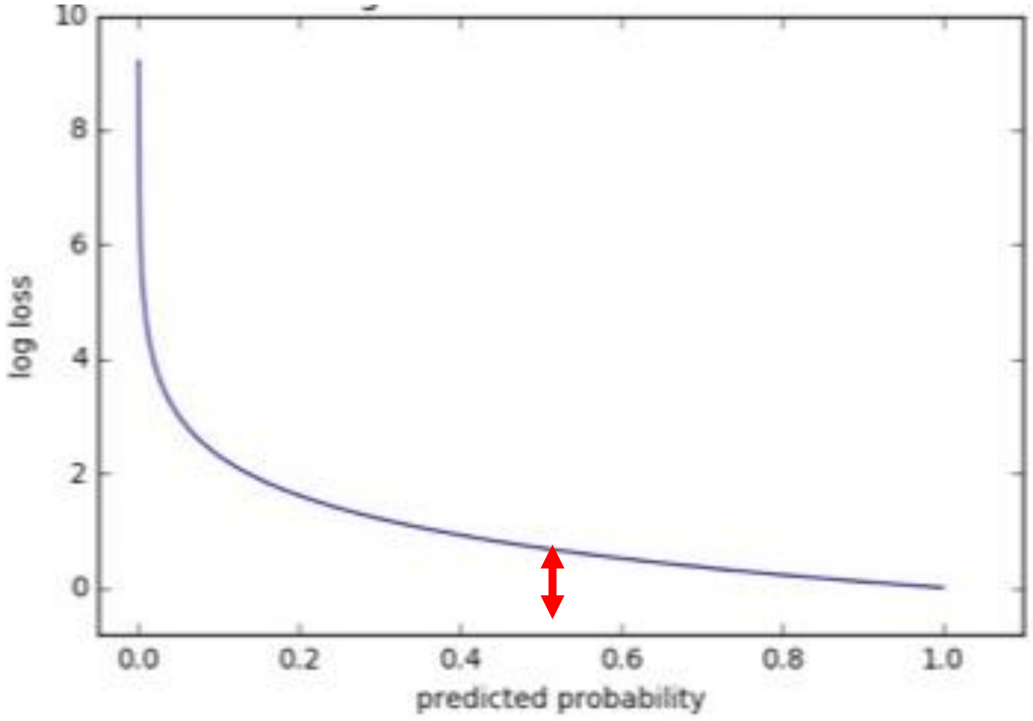
Softmax regression: cross entropy cost function

- The cost function used for softmax regression is a multi-class extension of the **log-loss** function, called **cross-entropy**.
- It applies **log-loss** to the highest probability (and selected class).

Softmax()

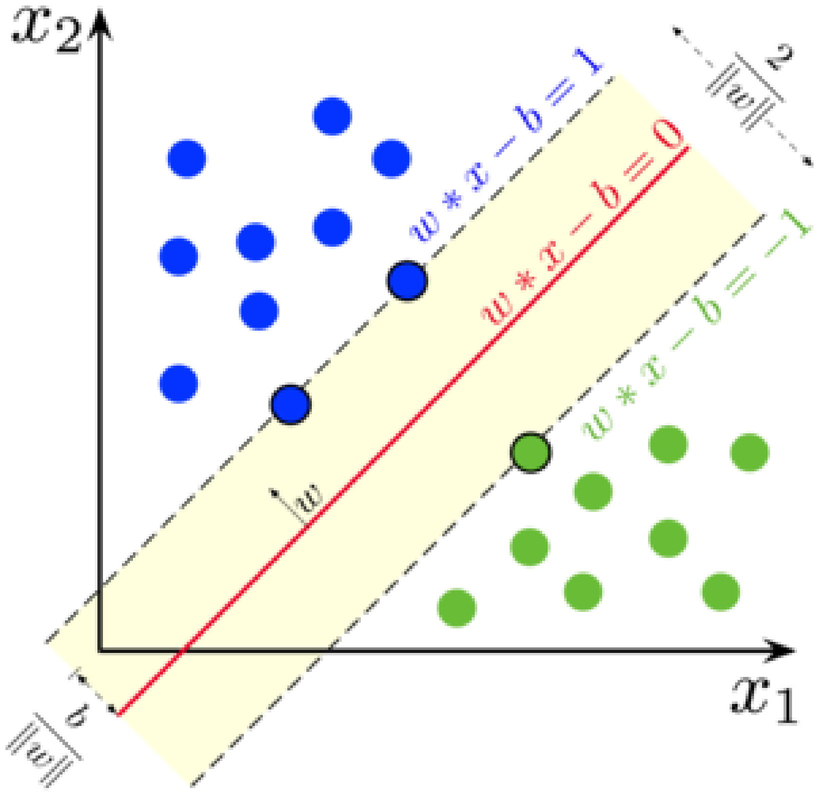
0.2
0.3
0.5

L = - log (**0.5**) = 0.693



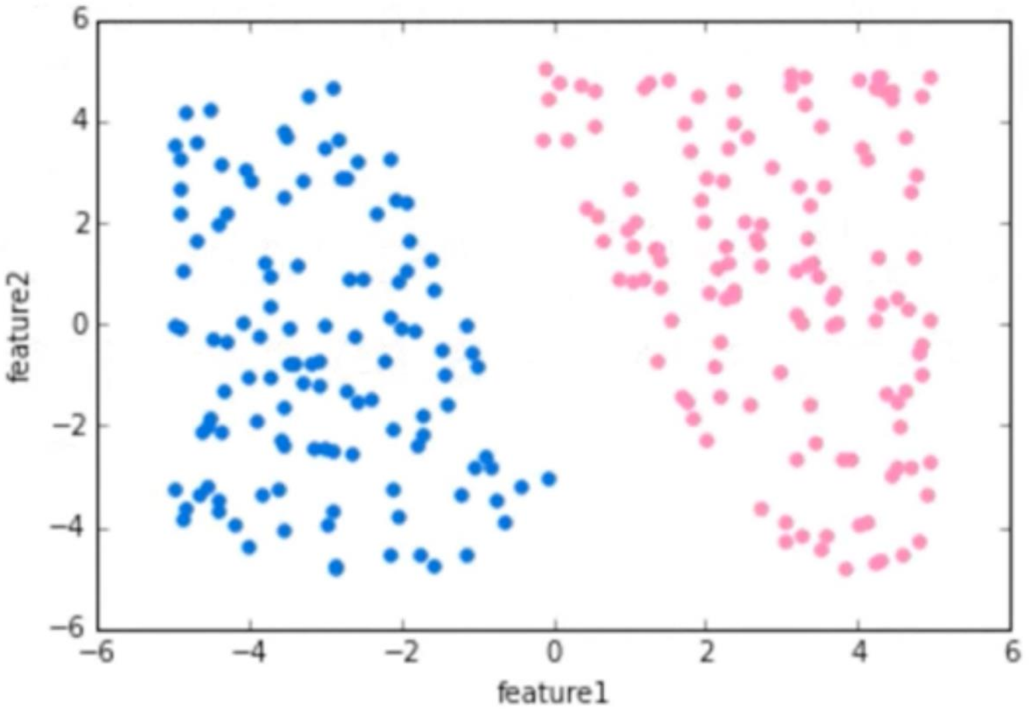
Support Vector Machines (SVMs)

- An SVM constructs a hyper-plane (or set of hyper-planes) in a high dimensional space, which can be used for classification (or regression or other tasks).
- Intuitively, a good separation is achieved by the hyper-plane that has the **largest distance** to the **nearest** training **data points** of any class (so-called **functional margin**),
- In general, the **larger the margin, the lower the generalization error of the classifier**
- Let's look at an example...



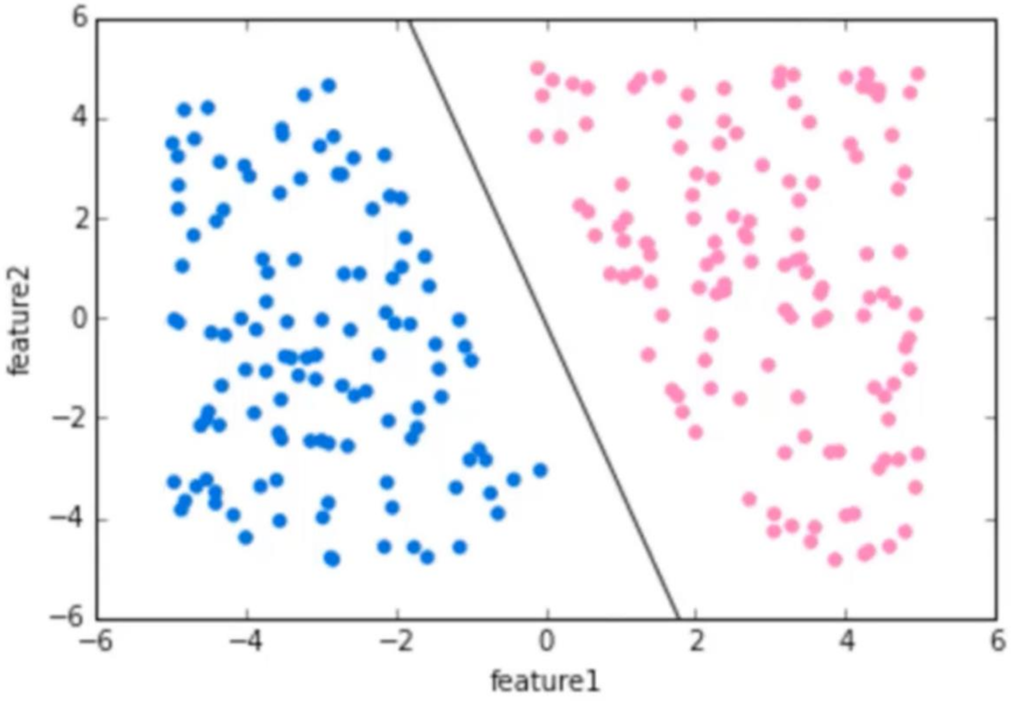
SVM Example:

- Here are two dimensions, and we have two classes / labels



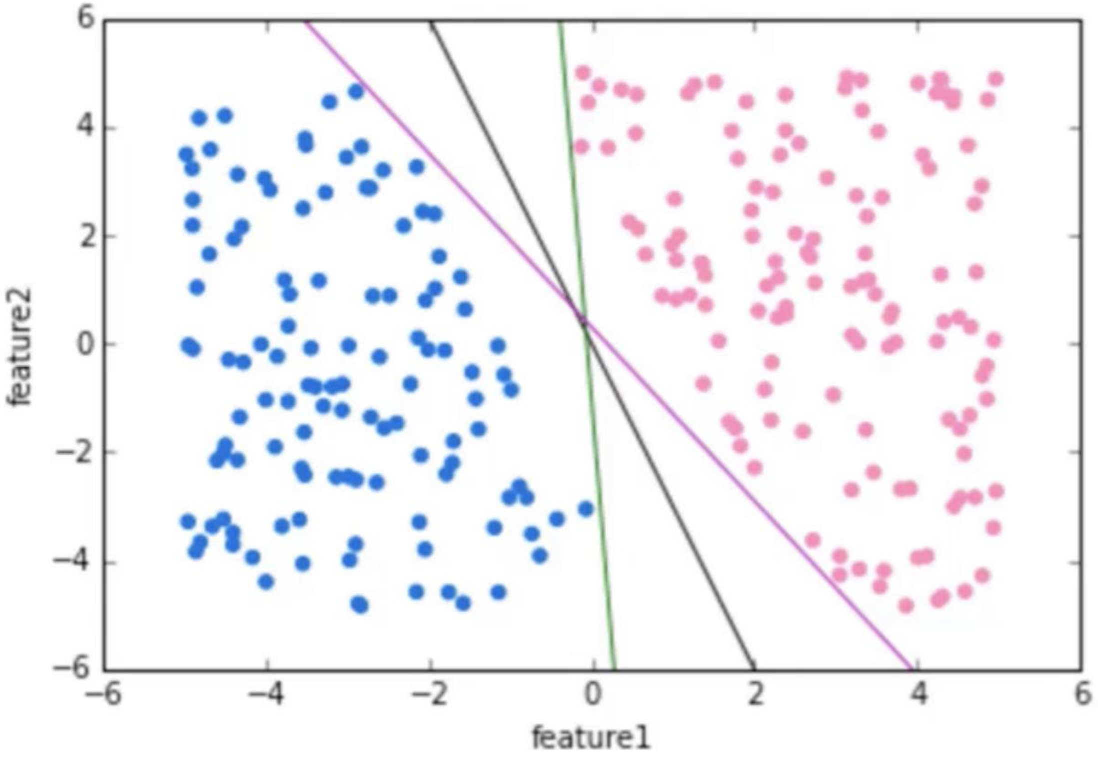
SVM Example:

- We could draw a regression dividing line to help us classify



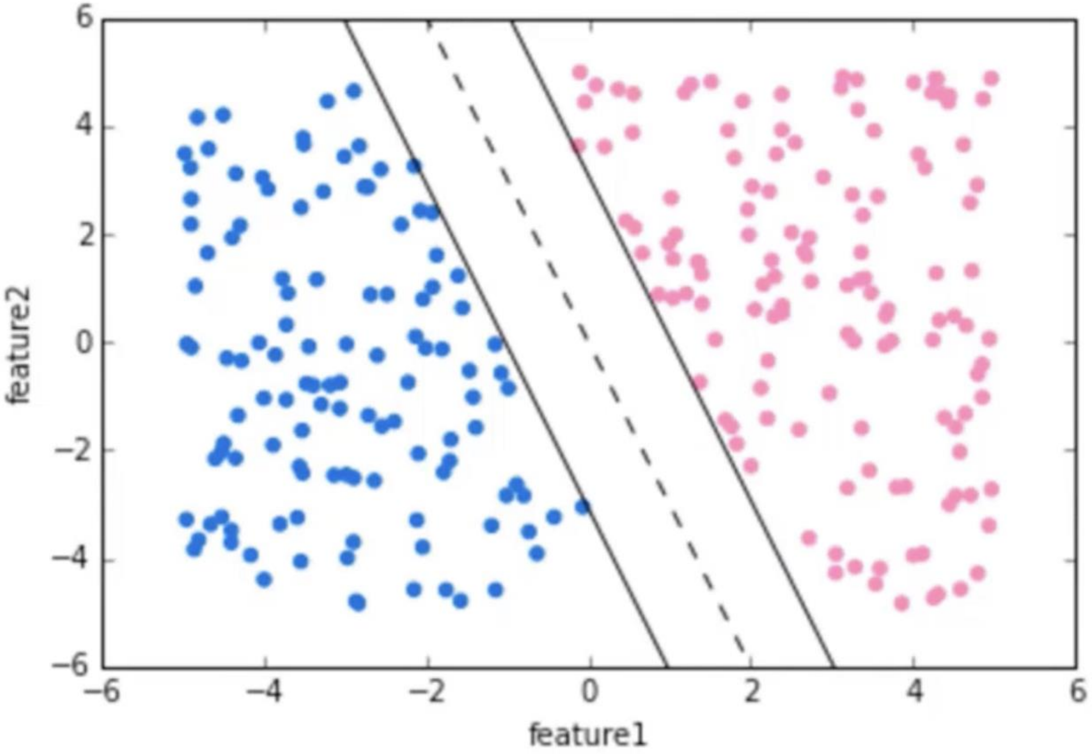
But there are many options…

- Which slope/gradient should we go with?



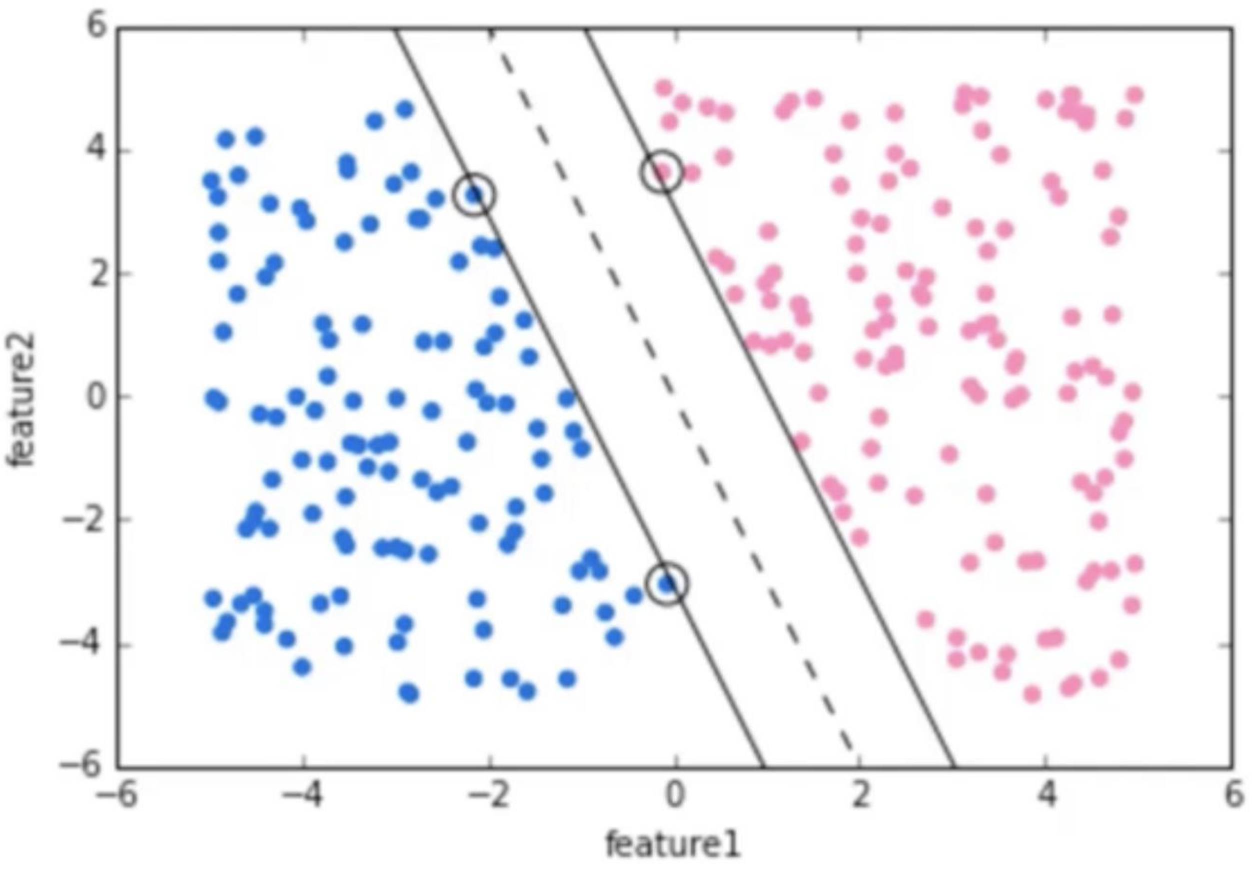
SVM maximises margin between classes

- We would like to choose a hyperplane that maximises the margin between classes – reduces the likelihood of error



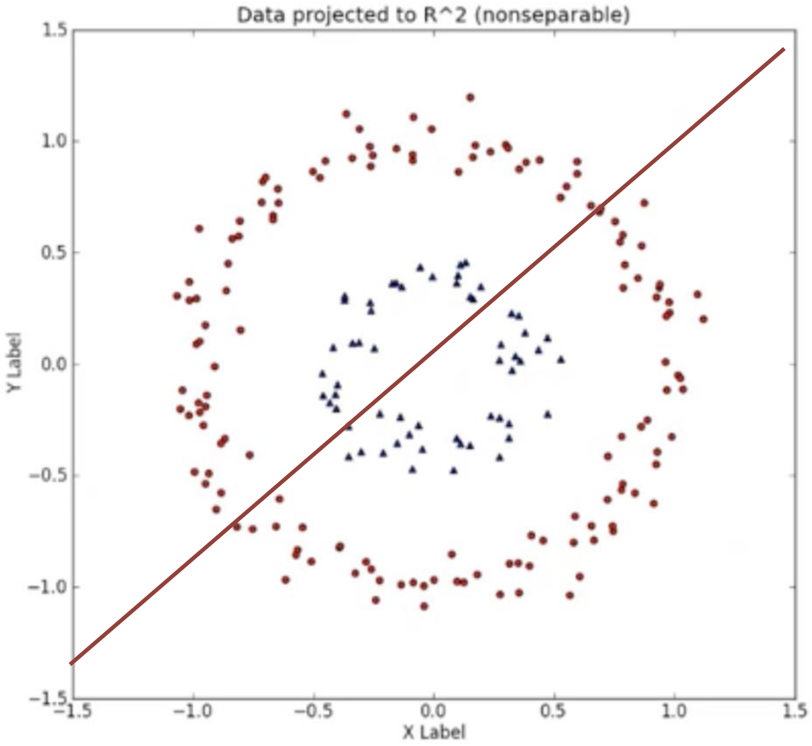
Support Vectors

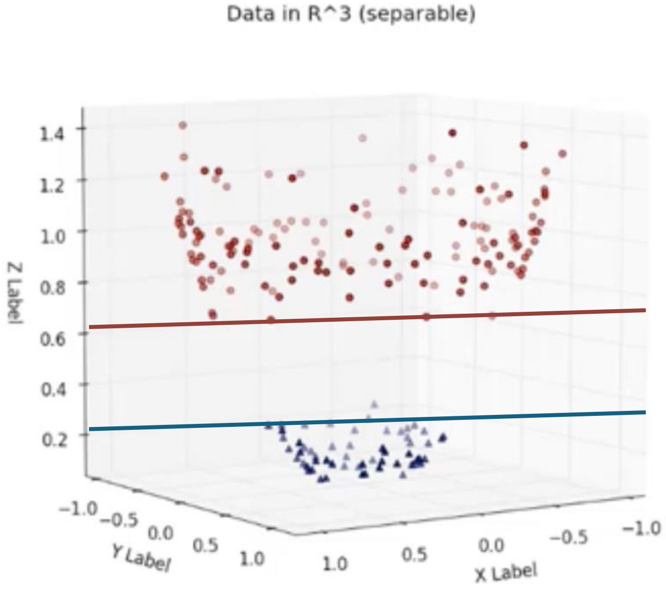
- **Points** that support the margin lines are known **as Support Vectors**



Support Vectors: the ‘Kernel Trick’

- This concept can expand to **non-linear data** (that isn’t separable with a straight line) through a ‘kernel-trick’
- In 2D, a straight cannot be drawn to separate the classes, but in 3D, it can!!



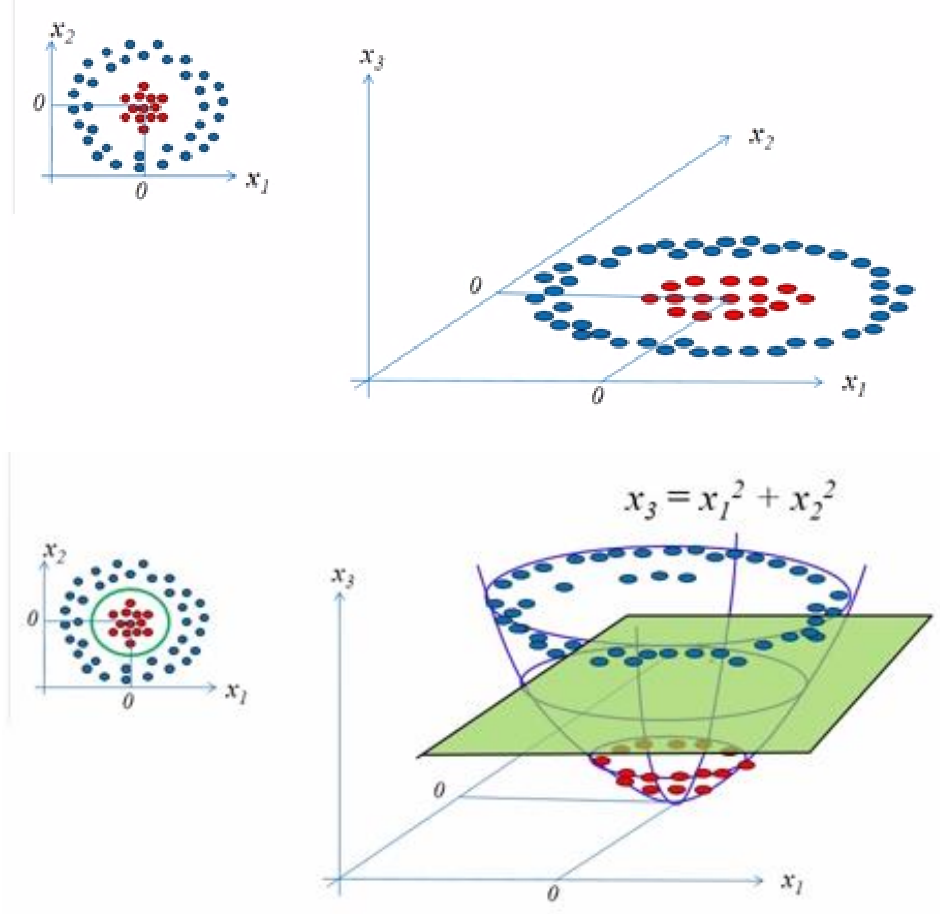


Support Vectors: the ‘Kernel Trick’

- To solve a nonlinear problem with SVM:

1. We **transform** the training data onto a **higher dimensional feature space** via a **mapping function** ϕ.
2. We **train a linear SVM model** to classify the data in **this new feature space**.
3. Then, we can **use the same mapping function** ϕ to **transform unseen data** to classify it using the linear SVM model.

The **kernel trick** avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary.



https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Support_Vector_Machines_SVM_2.php

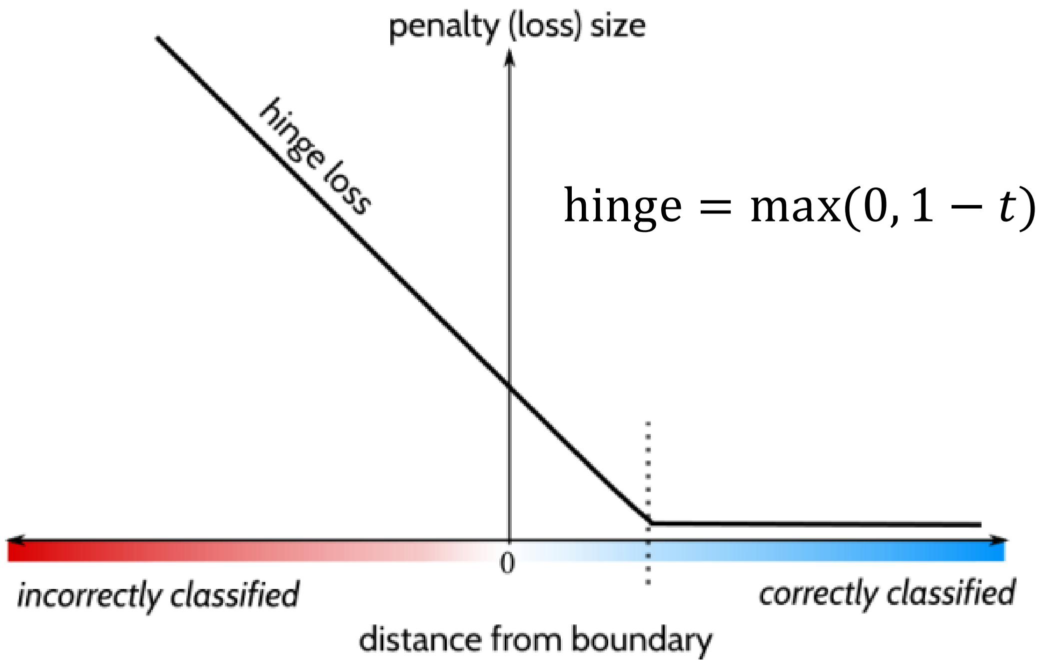
Online SVMs with the hinge loss

$$J(\theta) = J(\boldsymbol{w}, b) = \underbrace{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}_{\text{regularization term}} + \underbrace{C\sum_{1=1}^{m} \max(0, 1 - t_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))}_{\text{hinge loss term}}$$

Traditional SVM are trained offline (batch-training)

However, we can train online SVMs using gradient descent, just like we train logistic or softmax regression classifiers

Rather than using the **log loss** we use the **hinge function** as our cost function



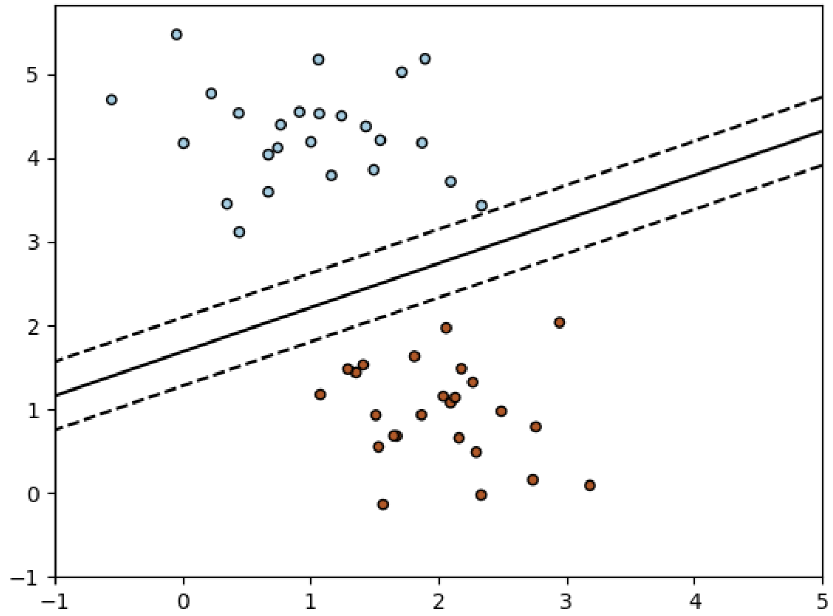
$$\text{hinge} = \max(0, 1 - t)$$

https://www.analyticsvidhya.com/blog/2021/05/top-15-questions-to-test-your-data-science-skills-on-sv

Stochastic Gradient Descent Classification

- The class **SGDClassifier** implements a plain **stochastic gradient descent learning** routine which supports different loss functions and penalties for classification.
- The default function scikit-learn is the **Linear SVM decision function**

https://scikit-learn.org/stable/modules/sgd.html



https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_separating_hyperplane.html

Confusion Matrix

- **Accuracy** = Total correctly classified (True Positive and True Negative) out of all predictions (total population)
- **Precision** = Measure of **predictive positive cases** (**T**rue **P**ositive and **F**alse **P**ositive). Precision focusses on quality

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- **Recall (Sensitivity)** = Measure of **actual positive cases** (**T**rue **P**ositive and **F**alse **N**egative)

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- **F1 Score** = Harmonic mean of precision and recall. Provides a balanced score.

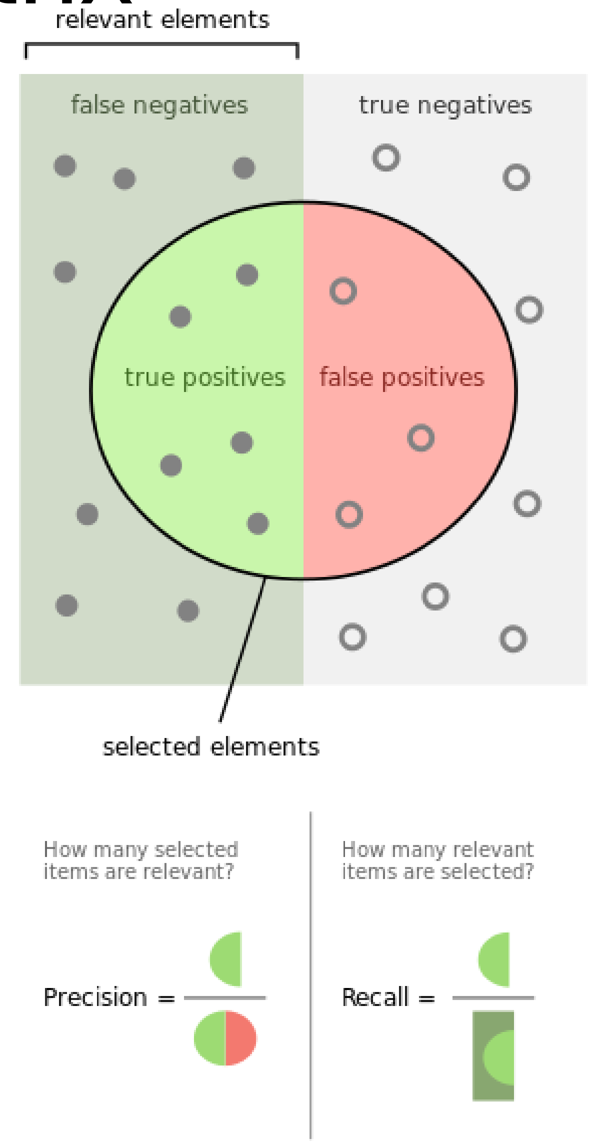
Performance Metrics: Confusion Matrix

$$\boldsymbol{accuracy} = \frac{correct\ predictions}{total\ predictions} = \frac{\boldsymbol{TP + TN}}{TP + TN + FP + FN}$$

$$\boldsymbol{precision} = \frac{TP}{TP + \boldsymbol{FP}}$$

$$\boldsymbol{recall} = \frac{TP}{TP + \boldsymbol{FN}}$$

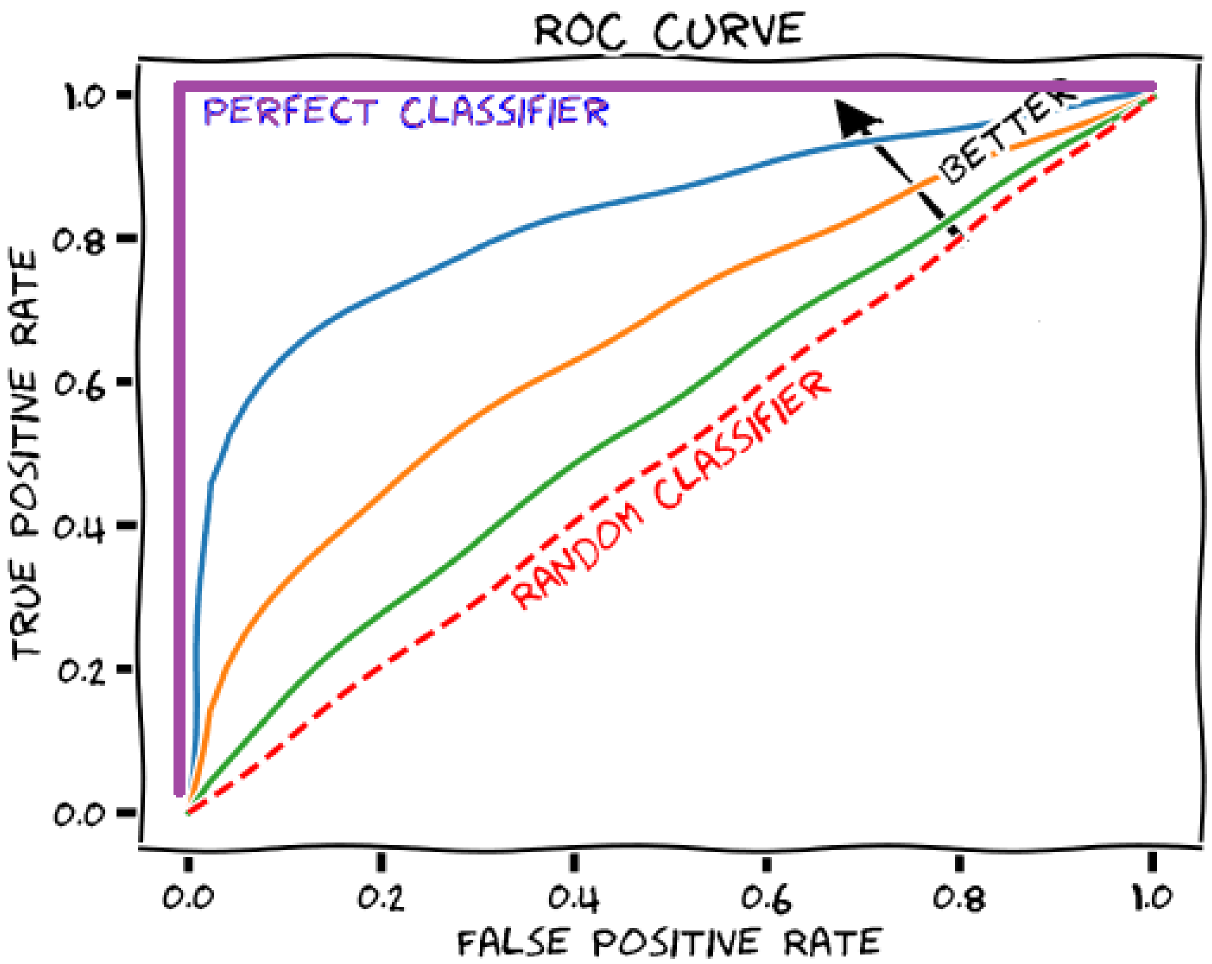
$$\boldsymbol{F1} = 2\,\frac{precision \times recall}{precision + recall}$$



Performance Metrics: Area under the ROC curve

$$TPR = recall = \frac{TP}{TP + FN}$$

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



https://medium.datadriveninvestor.com/understanding-auc-roc-clearly-explained-74c53d292a02

MNIST Dataset
Sklearn

Images are Numbers

```
[[  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   3.  18.  18.  18. 126. 136. 175.  26.]
 [ 94. 154. 170. 253. 253. 253. 253. 253. 225. 172.]
 [253. 253. 253. 253. 253. 253. 253. 251.  93.  82.]
 [253. 253. 253. 253. 198. 182. 247. 241.   0.   0.]
 [107. 253. 253. 205.  11.   0.  43. 154.   0.   0.]
 [  1. 154. 253.  90.   0.   0.   0.   0.   0.   0.]
 [  0. 139. 253. 190.   2.   0.   0.   0.   0.   0.]
 [  0.  11. 190. 253.  70.   0.   0.   0.   0.   0.]
 [  0.   0.  35. 241. 225. 160. 108.   1.   0.   0.]
 [  0.   0.   0.  81. 240. 253. 253. 119.  25.   0.]
 [  0.   0.   0.   0.  45. 186. 253. 253. 150.  27.]
 [  0.   0.   0.   0.   0.  16.  93. 252. 253. 187.]
 [  0.   0.   0.   0.   0.   0.   0. 249. 253. 249.]
 [  0.   0.   0.   0.  46. 130. 183. 253. 253. 207.]
 [  0.   0.  39. 148. 229. 253. 253. 253. 250. 182.]
 [ 24. 114. 221. 253. 253. 253. 253. 201.  78.   0.]
 [213. 253. 253. 253. 253. 198.  81.   2.   0.   0.]
 [253. 253. 253. 195.  80.   9.   0.   0.   0.   0.]
 [253. 244. 133.  11.   0.   0.   0.   0.   0.   0.]
 [132.  16.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.]]
```

- Here is a hand drawn digit 5 from the MNIST library [28 x 28] px
- An image is just a matrix of numbers [0,255]   0 for white, 255 for black

Logistic Regression with sklearn

- from **sklearn.model_selection** import **train_test_split**
- from **sklearn.linear_model** import **LogisticRegression**
- from **sklearn.metrics** import **confusion_matrix**
- from **sklearn.metrics** import **classification_report**

Logistic Regression – fit the model to the data

from **sklearn.linear_model** import **LogisticRegression**

log_model = LogisticRegression()
log_model.**fit**(X_train, y_train)

log_model**.classes_**

```
array([0, 1])
```

Logistic Regression – let's test the model

predictions = log_model.**predict**(X_test)

```
array([0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 0])
```

Logistic Regression – let’s evaluate the model

from **sklearn.metrics** import **confusion_matrix**

confusion_matrix(y_test,predictions)

```
array([[90, 19],
       [48, 43]])
```

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Logistic Regression – let's evaluate the model

from **sklearn.metrics** import **classification_report**

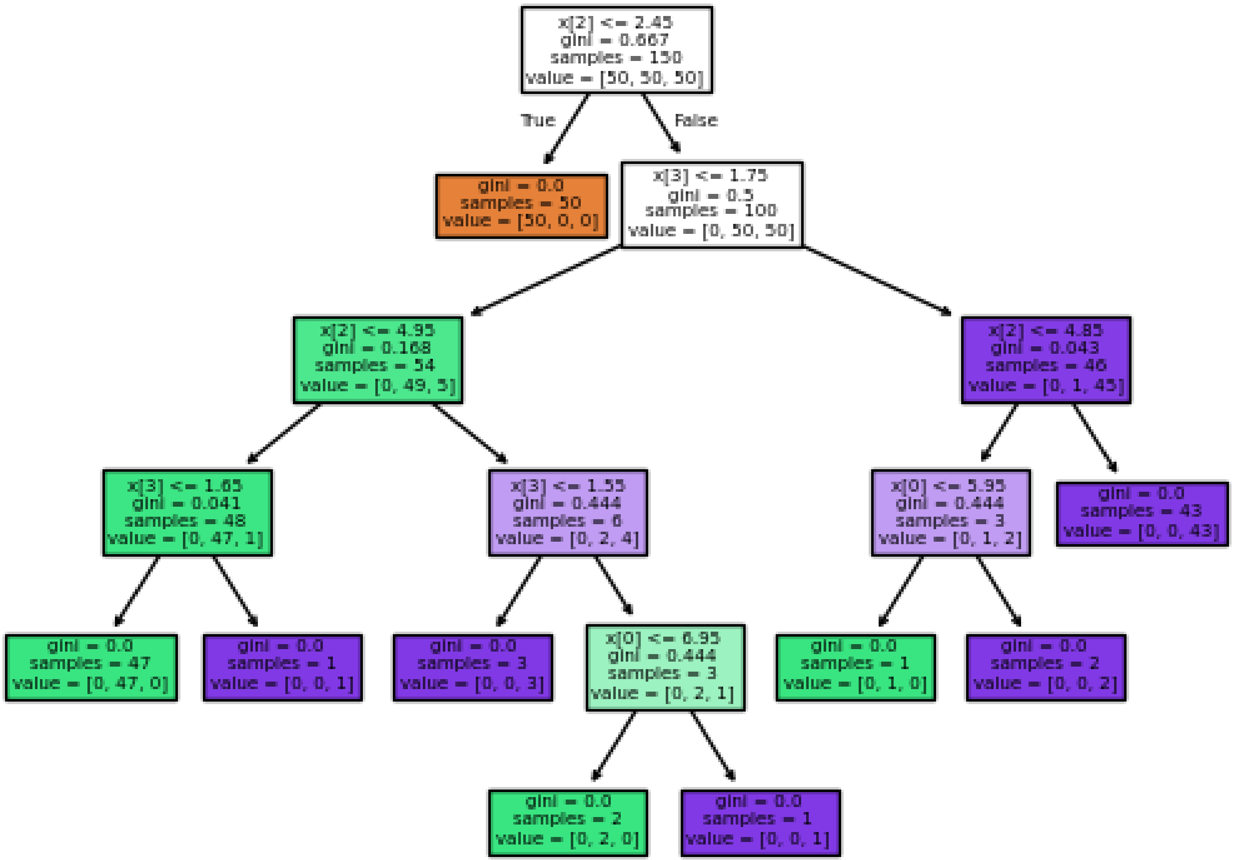
print(**classification_report**(y_test,predictions))

	precision	recall	f1-score	support
0	0.69	0.92	0.79	101
1	0.88	0.58	0.70	99
accuracy			0.75	200
macro avg	0.78	0.75	0.74	200
weighted avg	0.78	0.75	0.74	200

Coming up: Decision Trees

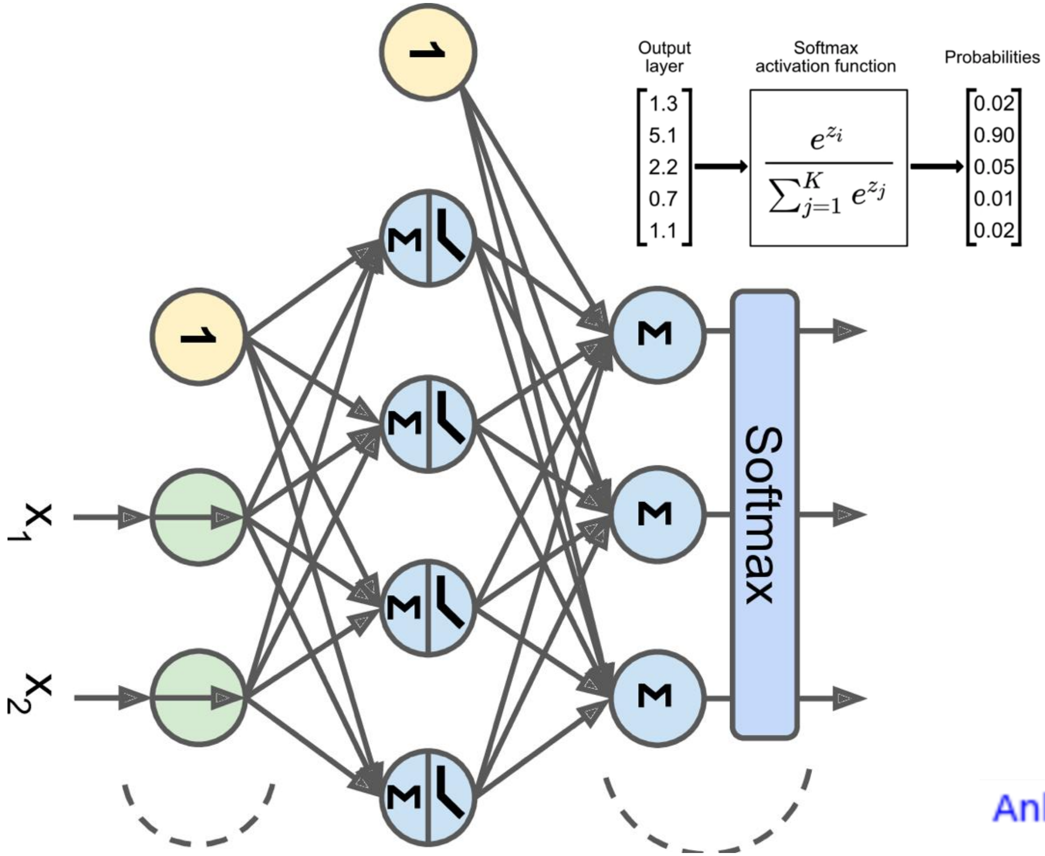
- Decision Trees (DTs) are a non-parametric supervised learning method used for classification (and regression).
- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision tree trained on all the iris features



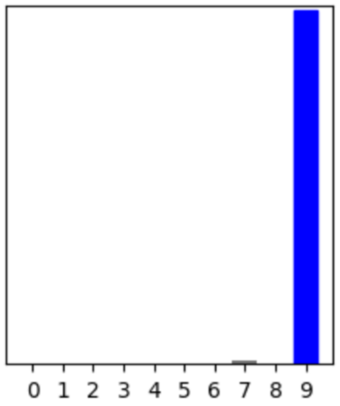
Coming up: Neural Networks for classification

Input



Hidden layer(s)

argmax



Ankle boot 99% (Ankle boot)