Classification

Using Machine Learning Tools

Geron Chapter 3, 6

Last time …

- Steps you can take to improve the performance of your ML system:
  - replacing data
  - scaling data
- Pipelines for repeatable workflows
- Linear regression model
- Polynomial models
- Parameters and hyperparameters

Classification terminology

- **Classifier**
  - Uses a discrete set of possible outputs = classes
  - Can be supervised, semi-supervised or unsupervised (see week 7)
- **Target**
  - What we are predicting
  - Passed in as “y” to regression/classification method
  - Also called: label, ground truth, dependent variable, outcome variable, or response variable

Regression vs Classification

- Regression:
  - predicts real numbers (values)
  - on a numerical, ordered scale
  - the larger the difference, the worse
    - e.g. house price, wine quality

- Classification:
  - predicts classes (labels)
  - typically categorical, normally no meaningful order
  - all differences are usually treated equally
    - e.g. cancer vs healthy ; kangaroo vs pademelon vs quokka

Linear Model as a Binary Classifier

- Linear model used in regression:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- Apply a threshold:

If $\hat{y} < \text{threshold}$

Class 1

Else

Class 2

Stochastic Gradient Descent Classifier

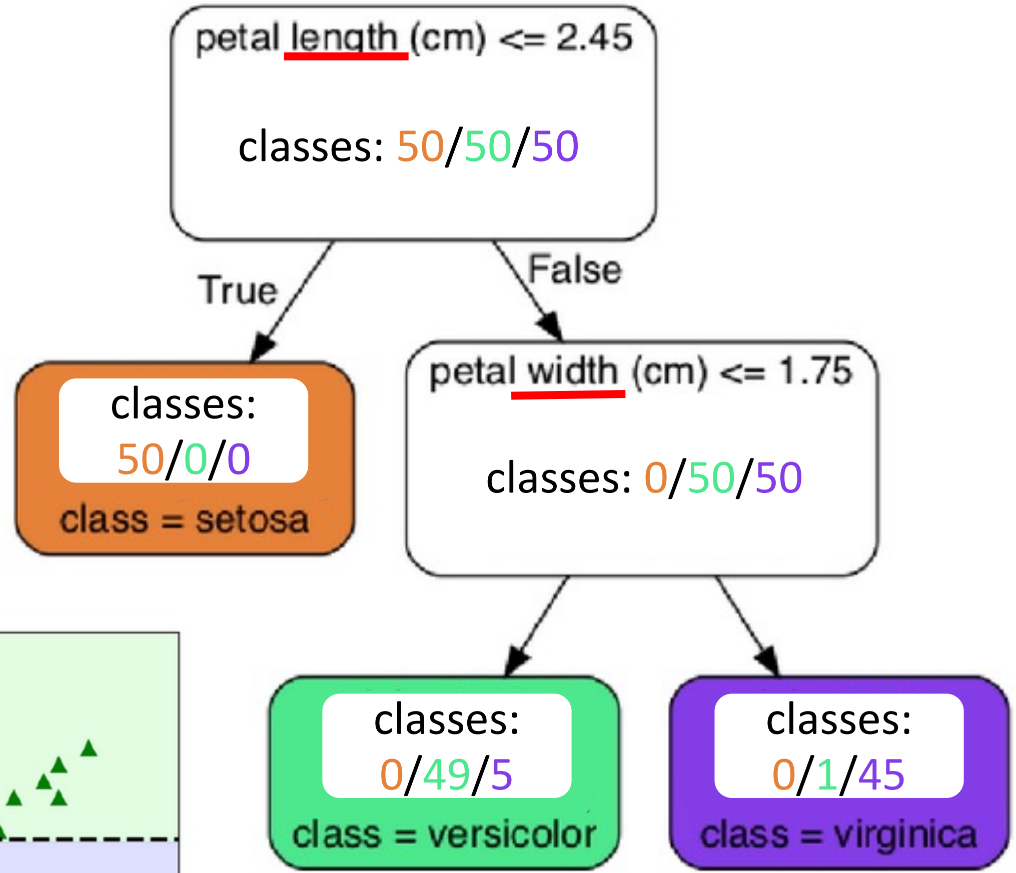
- Binary classifier using a linear model
- Stochastic Gradient Descent (SGD) is a fitting algorithm
  - Iteratively follows the gradient (derivative) of the loss function
  - Fast, scalable
- SGD classifier in scikit-learn is a linear model using SGD

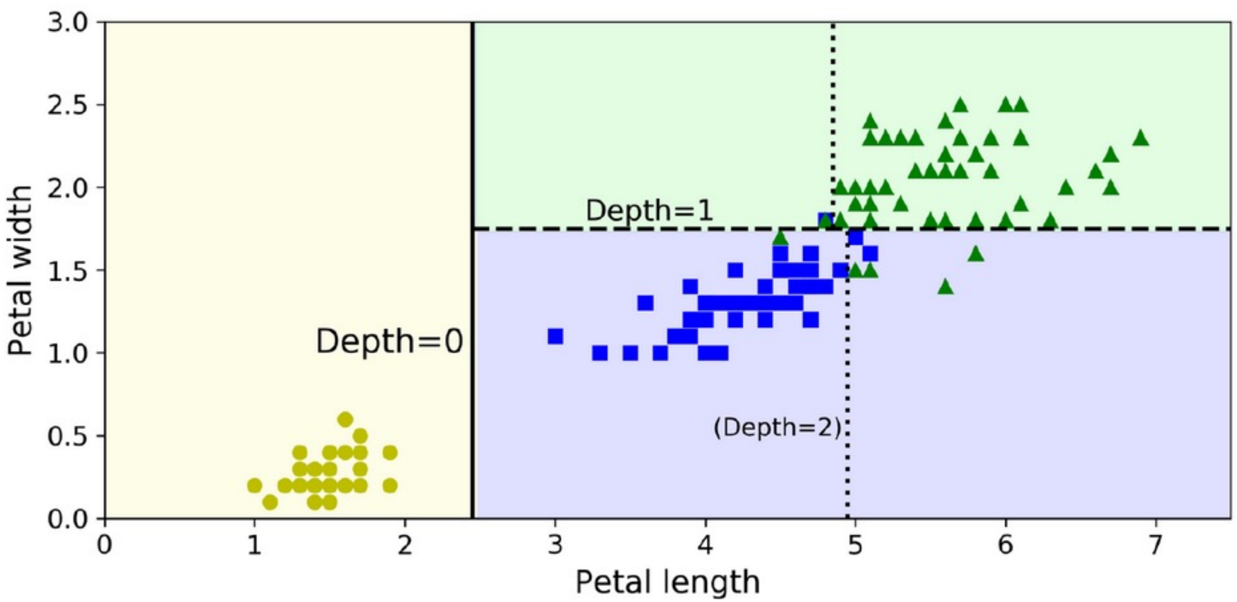
```
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier(random_state=42)
clf.fit(X_train, y_train)
```

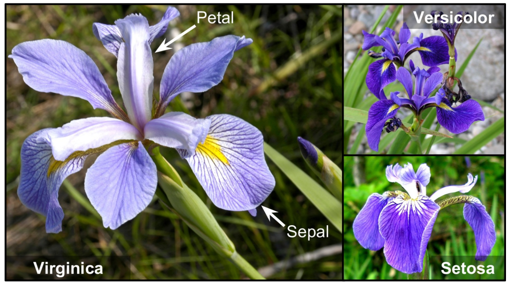
Decision Tree

Images: Geron, Hands On ML

- Iterative splitting
- Maximise class separation
- One feature at a time
- Up to maximum depth
- Prune to avoid overfitting



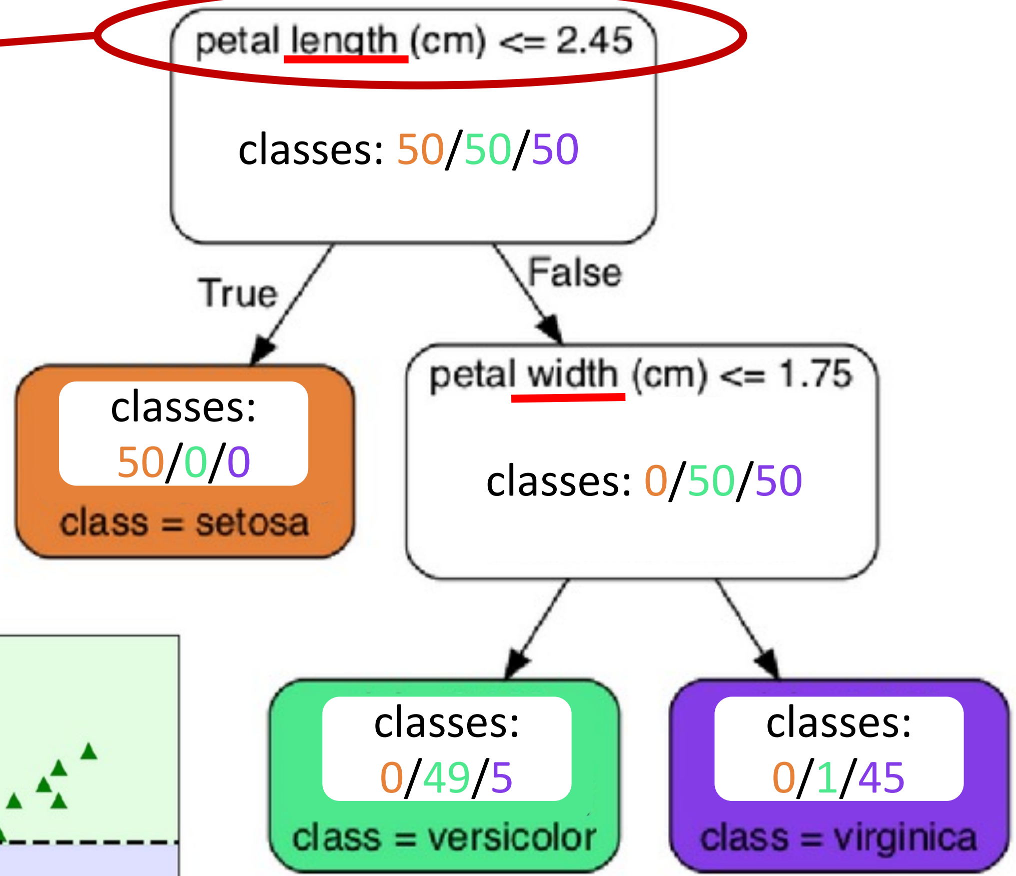


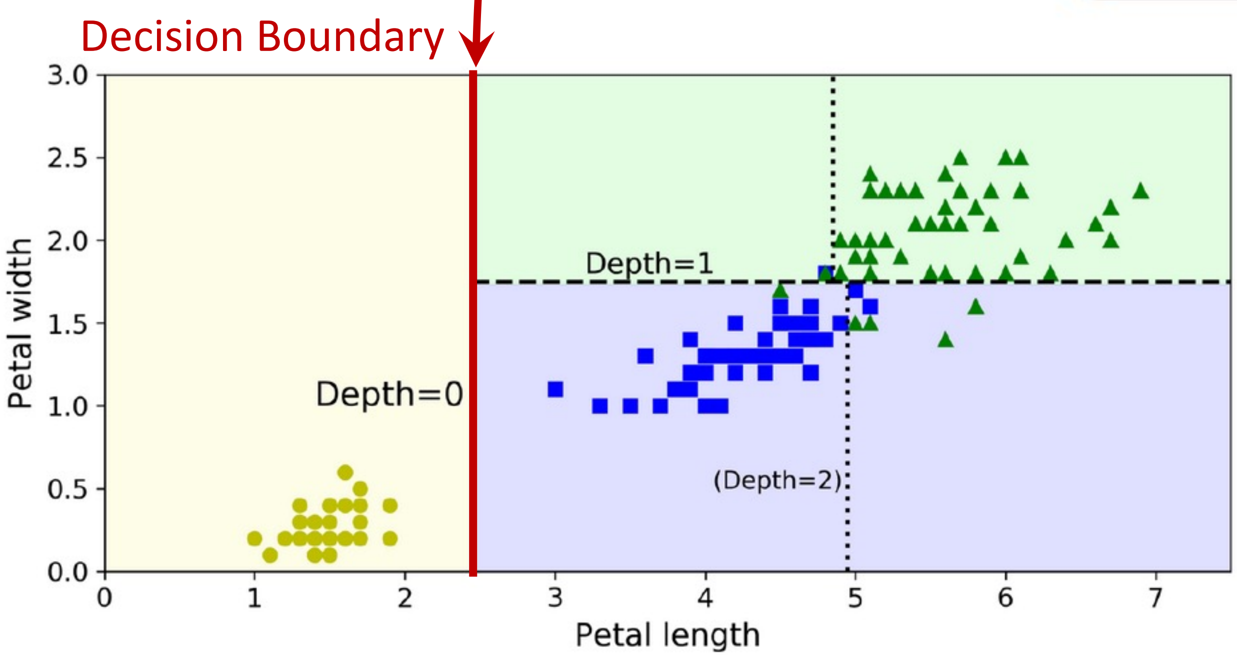


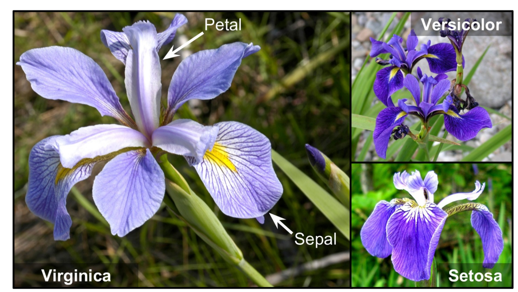
Decision Tree

Images: Geron, Hands On ML

- Iterative splitting
- Maximise class separation
- One feature at a time
- Up to maximum depth
- Prune to avoid overfitting



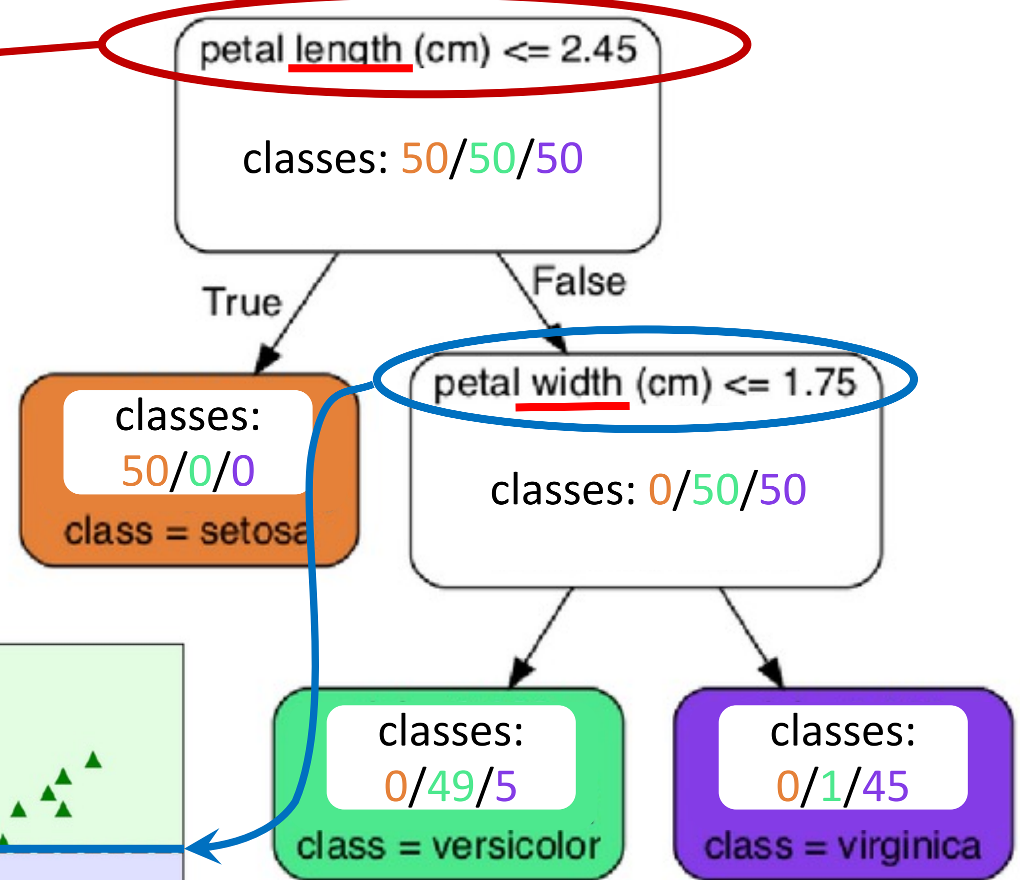




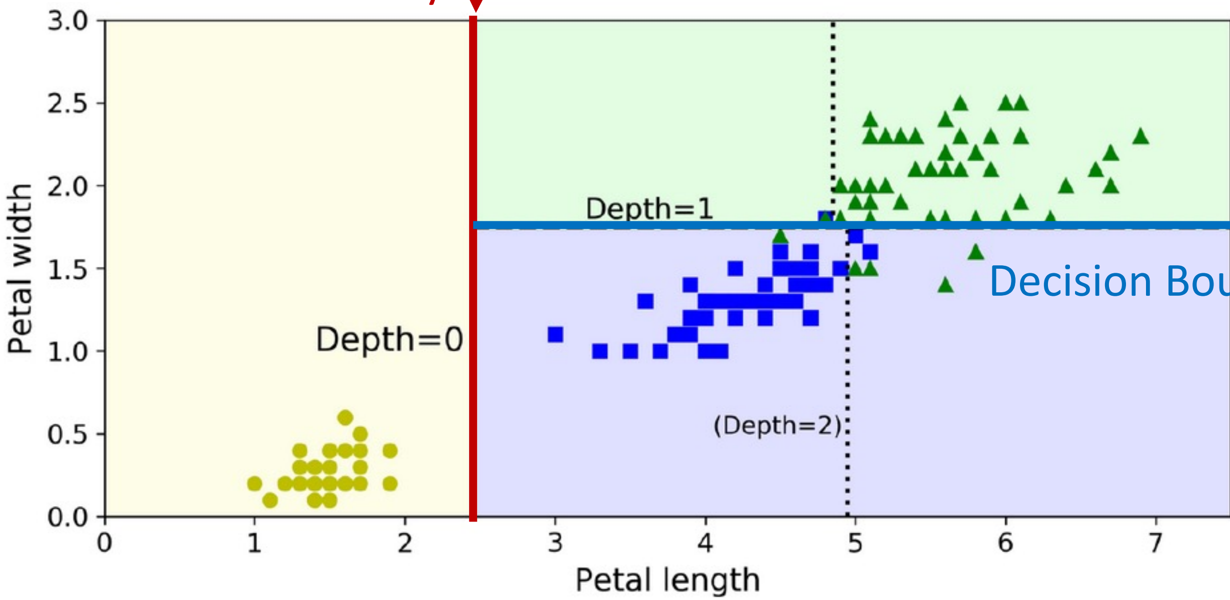
Decision Tree

Images: Geron, Hands On ML

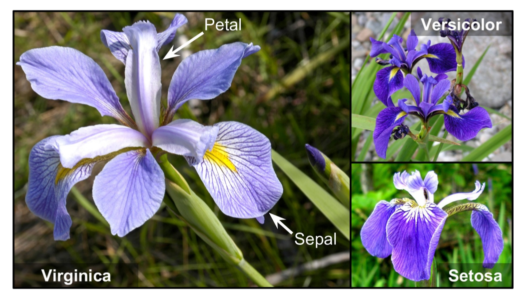
- Iterative splitting
- Maximise class separation
- One feature at a time
- Up to maximum depth
- Prune to avoid overfitting



Decision Boundary



Decision Boundary



Inner workings → white-box

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- Gini impurity metric measures the class distribution in a node
  - best = only one class (pure) → G = 0 for impurity
  - worst = completely evenly spread → $G_{max} = 1 - 1/N$ for impurity

- CART = Classification and Regression Tree

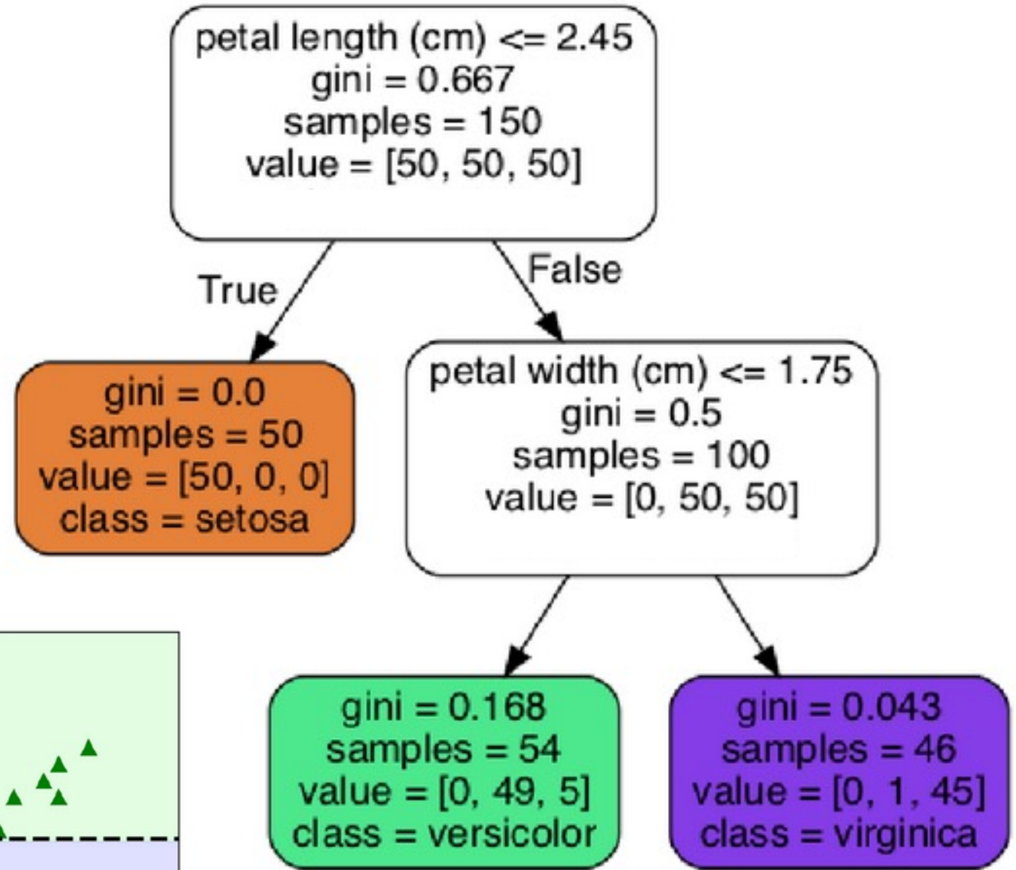
$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

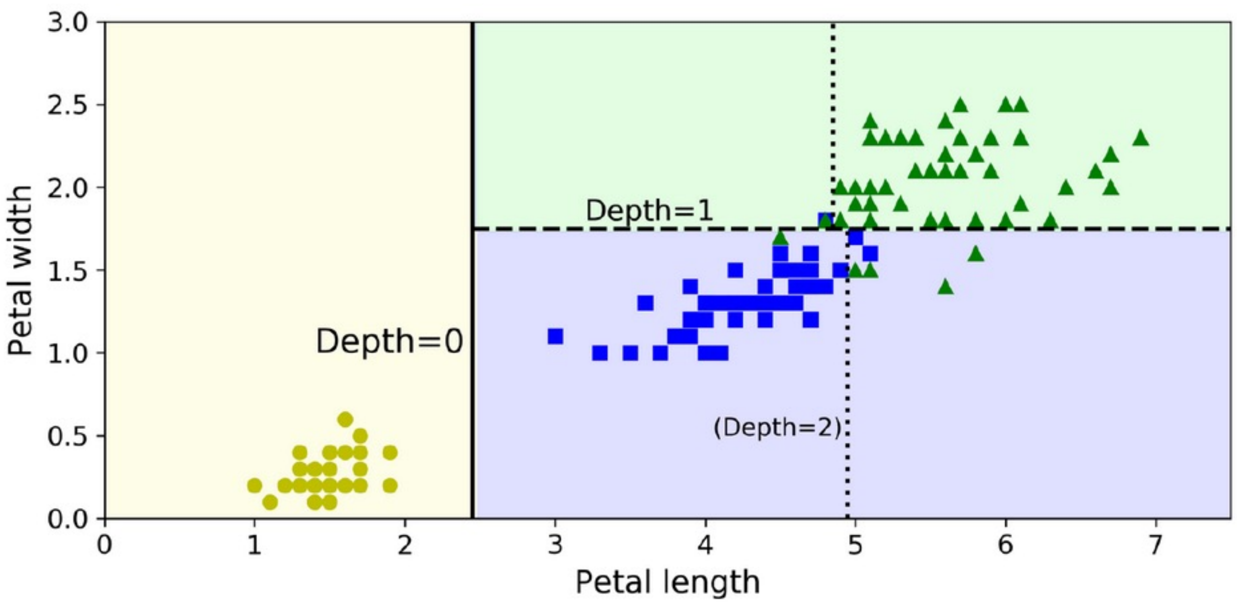
- A loss function is minimised to set threshold → min impurity
  - it is based on a weighted sum of the impurities from each branch
- Search for best threshold and branch out
- Iteratively grow tree until maximum depth reached

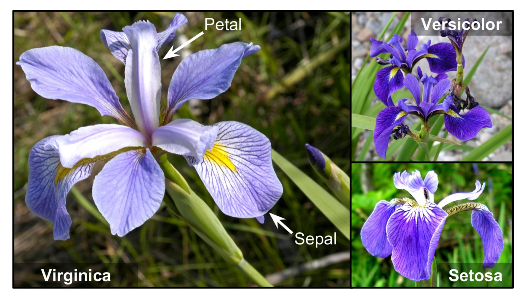
Decision Tree

Images: Geron, Hands On ML

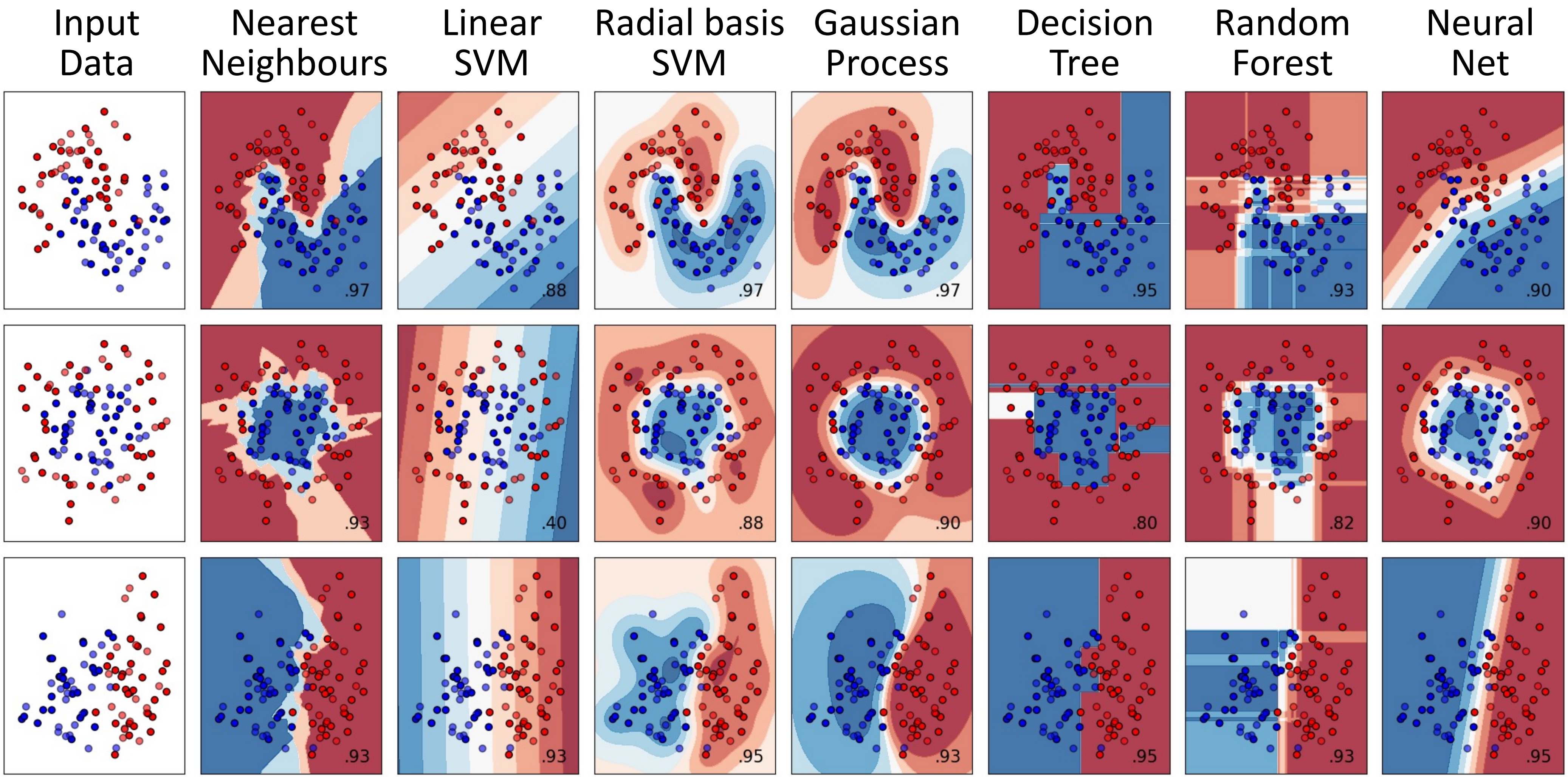
- Iterative splitting
- Maximise class separation
- One feature at a time
- Up to maximum depth
- Prune to avoid overfitting







Comparison of Classification Approaches



Note the different decision boundary shapes

Image: © 2007 - 2019, scikit-learn developers (BSD 3-clause License)

Evaluation / Performance Metrics

Evaluation / Performance Metrics

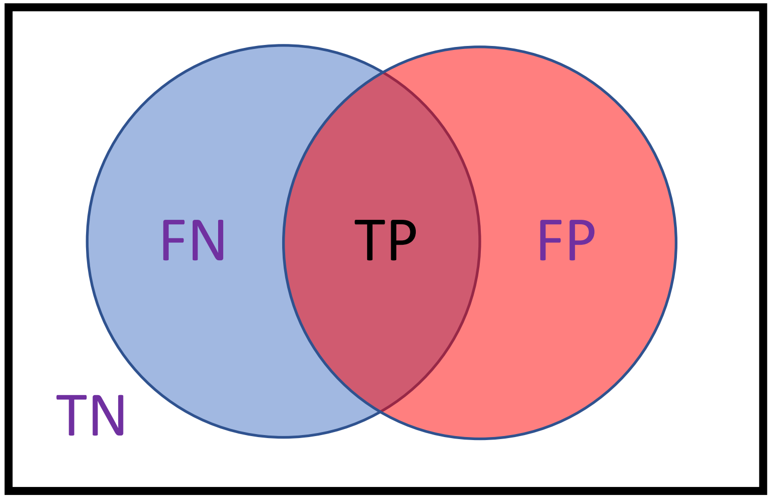
- Regression:
  - (Root) Mean Squared Error = L^2 norm = $\frac{1}{N}\sum_i |x_i - yi|^2 = \|\mathbf{x} - \mathbf{y}\|_2$
  - Mean Absolute Error = L^1 norm = $\frac{1}{N}\sum_i |x_i - y_i| = \|\mathbf{x} - \mathbf{y}\|_1$
  - Median/Max Absolute Error
  - R^2 / correlation coefficient $\rho = r = \frac{(\mathbf{x}-\bar{x})\cdot(\mathbf{y}-\bar{y})}{|\mathbf{x}-\bar{x}|\,|\mathbf{y}-\bar{y}|} = \cos\theta$

Evaluation / Performance Metrics

- Regression:
  - (Root) Mean Squared Error = L^2 norm = $\frac{1}{N}\sum_i |x_i - y_i|^2 = \|\mathbf{x} - \mathbf{y}\|_2$
  - Mean Absolute Error = L^1 norm = $\frac{1}{N}\sum_i |x_i - y_i| = \|\mathbf{x} - \mathbf{y}\|_1$
  - Median/Max Absolute Error
  - R^2 / correlation coefficient $\rho = r = \frac{(\mathbf{x}-\bar{x})\cdot(\mathbf{y}-\bar{y})}{|\mathbf{x}-\bar{x}|\,|\mathbf{y}-\bar{y}|} = \cos\theta$

- Classification:
  - Accuracy / Confusion Matrix
  - Precision / Recall
  - ROC Curve / Area Under Curve (AUC)
  - F_1 Score

Positive & Negative / True & False



Intersection = TP

All Real "Positives" = TP + FN

All Positive Detections = TP + FP

- Positives/Negatives (P/N) are from model predictions (above/below threshold)
- True is when the prediction agrees with the ground truth (labels)
- A true positive is a positive that agrees with ground truth
- A false negative is something of interest in the ground truth that is missed by the model
- Both FP and FN are bad, but usually not equally bad
- Sometimes P/N is obvious (e.g. detecting an object) but sometimes it is arbitrary (e.g. male/female) or even counter-intuitive (e.g. disease = positive)

Precision and Recall

$$Precision = \frac{TP}{TP + FP} =$$

What fraction of positive predictions are correct?

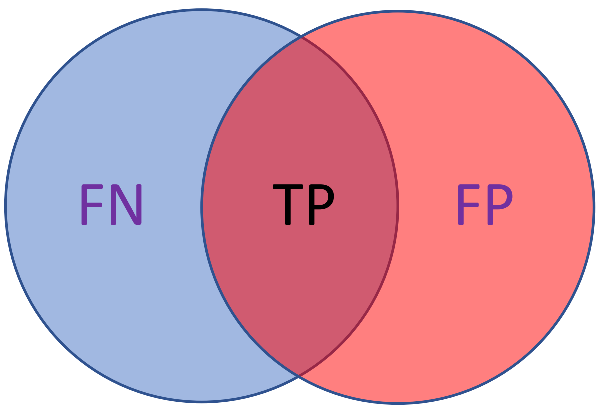
= Positive predictive value

$$Recall = \frac{TP}{TP + FN} =$$

What fraction of the real positive class are detected?

= True positive rate

= Sensitivity



Intersection = TP

All Real “Positives” = TP + FN

All Positive Detections = TP + FP

Precision and Recall

$$Precision = \frac{TP}{TP + FP}$$

What fraction of positive predictions are correct?

= Positive predictive value

$$Recall = \frac{TP}{TP + FN}$$

What fraction of the real positive class are detected?

= True positive rate

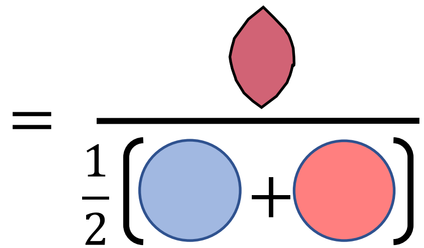
= Sensitivity

- Both ignore True Negatives (TN)
- Compare with Accuracy = (TP+TN)/(TP+TN+FP+FN)
- Sometimes accuracy is better, sometimes not

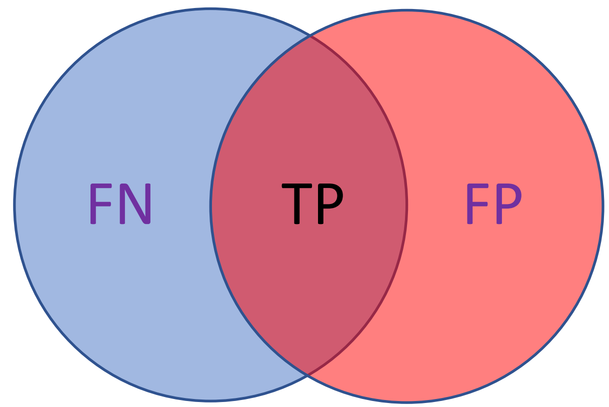
```
from sklearn.metrics import precision_score, recall_score
precision_score(y_train, y_train_pred)
recall_score(y_train, y_train_pred)
```

F_1 Score

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{precision \times recall}{\frac{1}{2} * (precision + recall)} = \frac{TP}{\frac{1}{2} * (FN + FP + 2 * TP)}$$



- Harmonic mean of precision and recall
- Ignores True Negatives (TN)
- Evenly weights precision and recall
- F_β weights differently, using β value



Intersection = TP

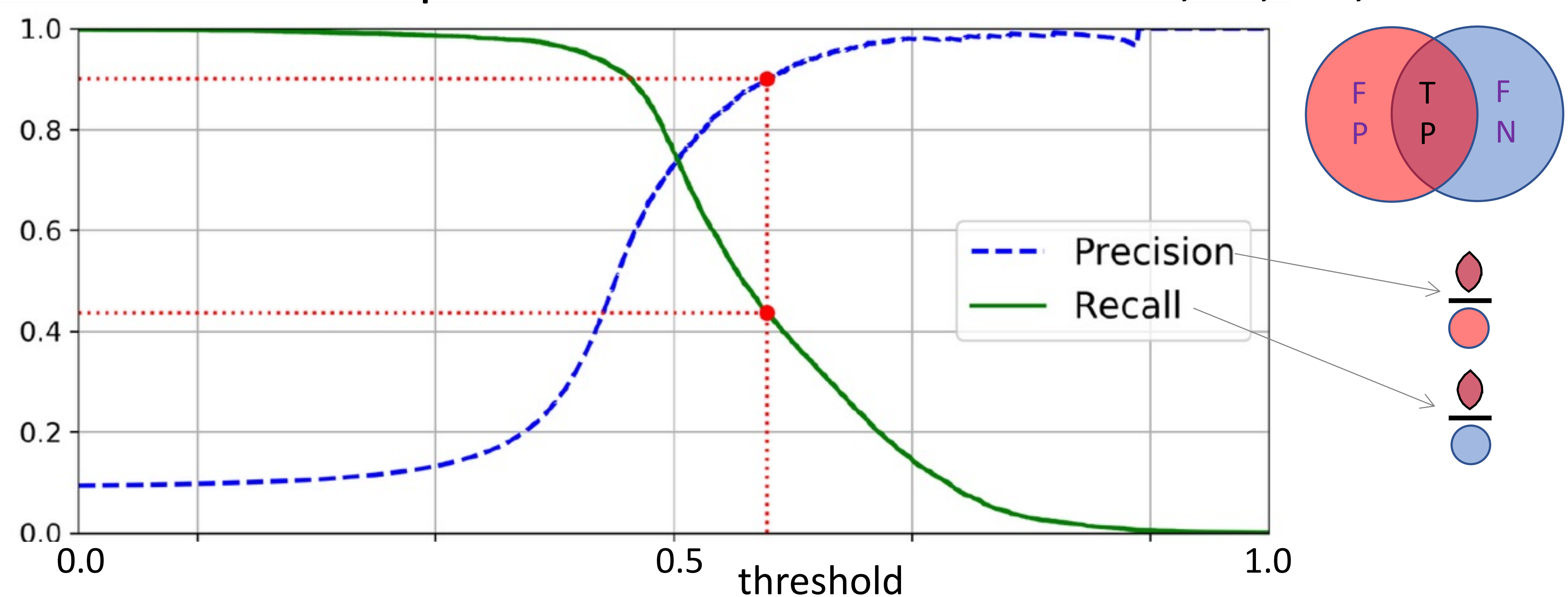
All Real Positives = TP + FN

All Positive Detections = TP + FP

```
from sklearn.metrics import f1_score
f1_score(y_train, y_train_pred)
```

Precision – Recall Trade-Off

Take real output value and threshold it → TP, FP, FN, TN



Low threshold → everything is positive, FN=0, = , Recall → 1

High threshold → everything is negative, FP=0, = , Prec. → 1

Image: Geron, Hands On ML

Receiver-Operating-Characteristic (ROC)

- Display trade-off between true positive rate and false positive rate
- Top left corner is best: TP=1, FP=0
- Each threshold → one point
- Use all thresholds of the output to get curve (min→ max)
- Output of method (before thresholding) can be called a "probability"
- Can also be used to look at hyper-parameter changes (instead of threshold)

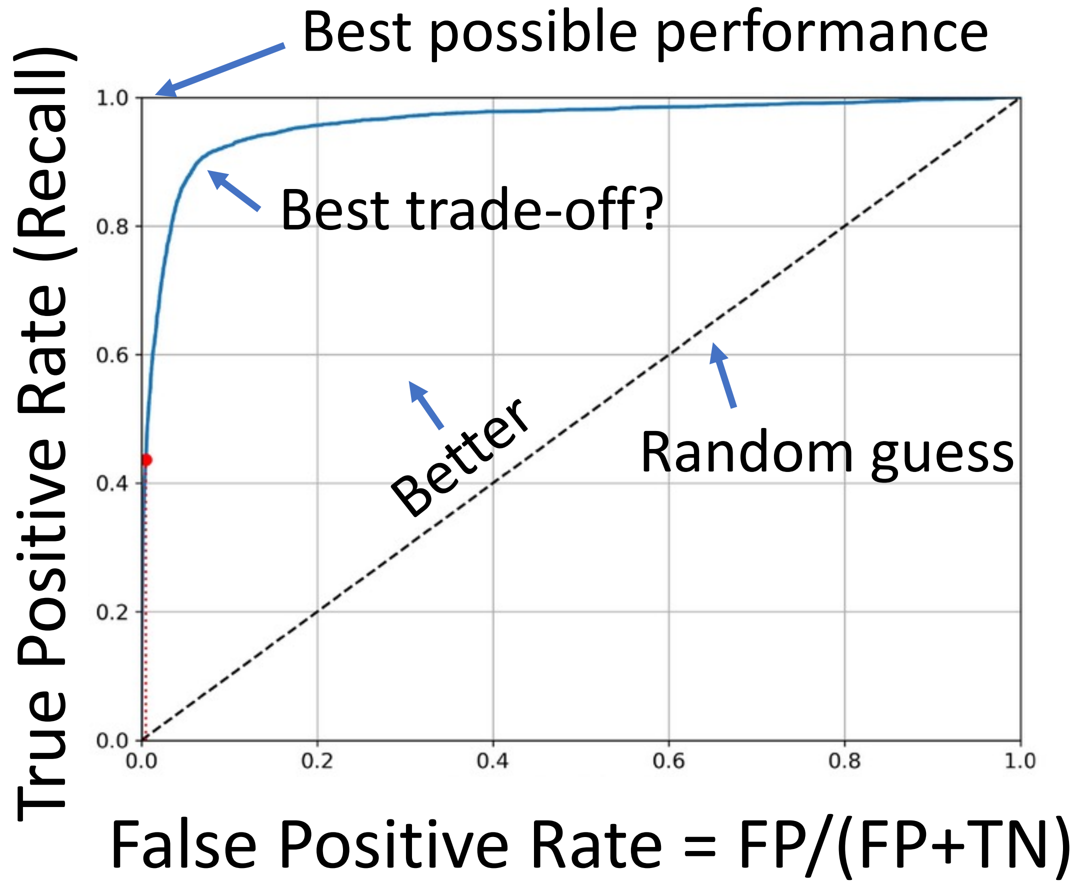


Image: Geron, Hands On ML, added annotations

Area-Under-Curve (AUC)

- Single scalar value to compare classifiers or hyperparameters
- Depends on performance across all thresholds or hyperparameter settings
- Usually use whole curve

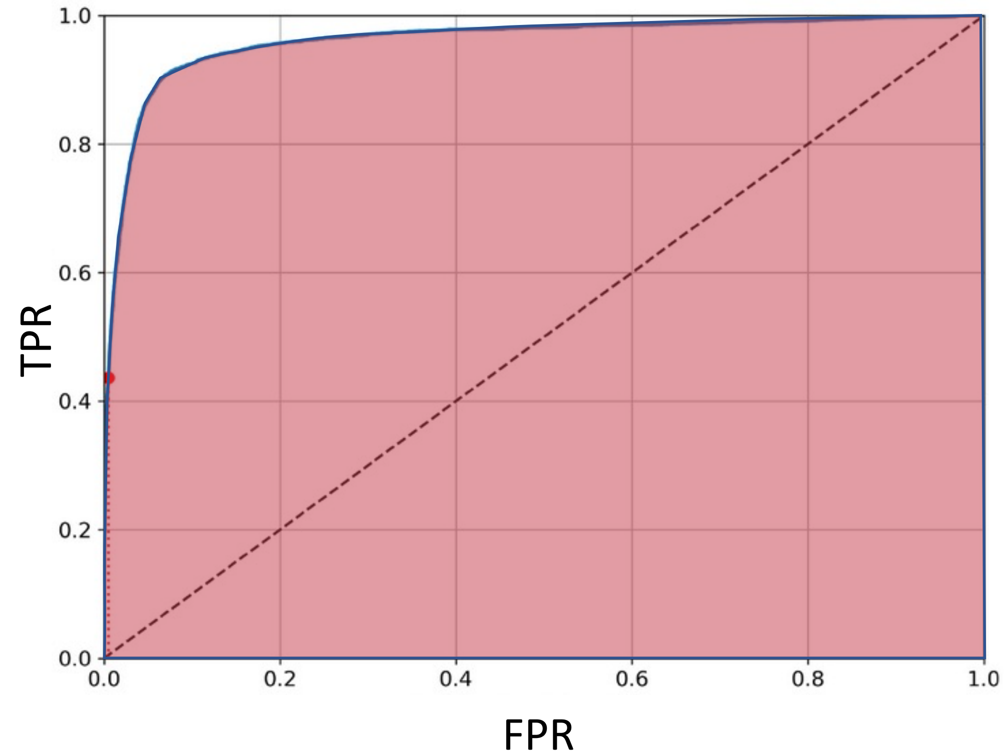


Image: Geron, Hands On ML, added annotations

Area-Under-Curve (AUC)

- Single scalar value to compare classifiers or hyperparameters
- Depends on performance across all thresholds or hyperparameter settings
- Usually use whole curve
  - but a smaller range of FPR can also be used (based on acceptable performance range)

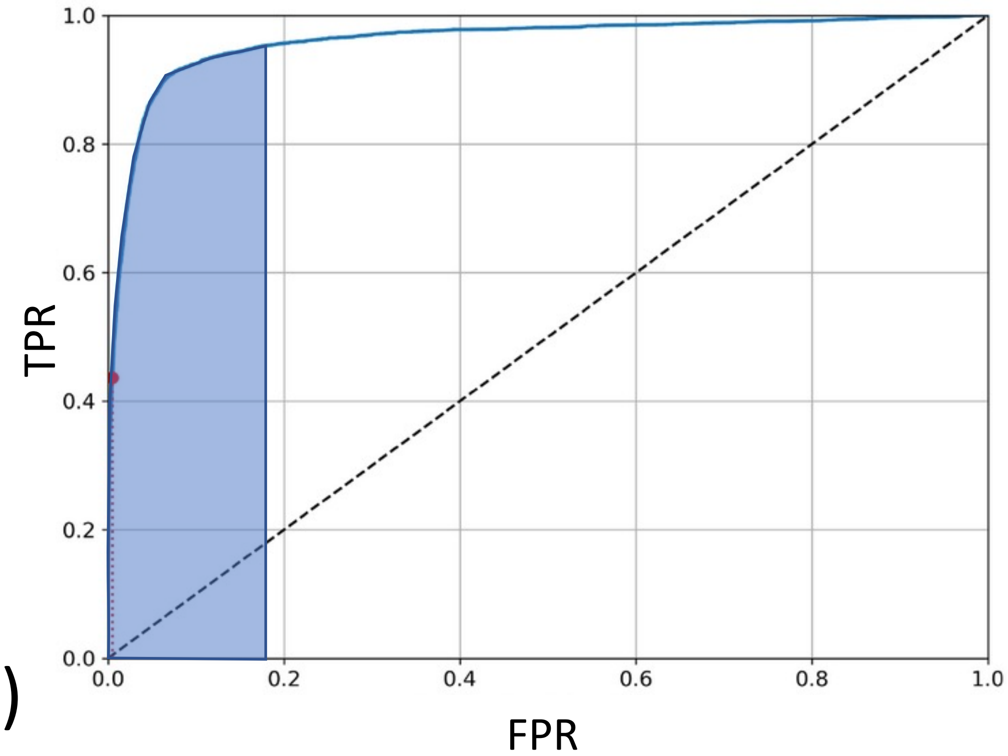


Image: Geron, Hands On ML, added annotations

Multiclass Classification

- More than two classes
- Targets can be:
  - arbitrary integers / names
  - one-hot encoding
    - N vectors ; each with one element = 1, others = 0

vector for class 1 . . . vector for class 5

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

- Can use binary classifiers for multiclass problems
  - One-vs-Rest strategy (default)
  - One-vs-One strategy
  - Scikit-learn classifiers do this automatically

Confusion Matrix

- Displays predicted vs actual class
  - run on the test set
- Diagonal elements are good
- Off-diagonals show errors
- Can have any number of classes
- Shows where most common errors occur

Predicted: Actual:	Class 1 (Negative)	Class 2 (Positive)
Class 1	True Negative count	False Positive count
Class 2	False Negative count	True Positive count

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
y_train_pred = cross_val_predict(clf, X_train, y_train)
confusion_matrix(y_train, y_train_pred)
```

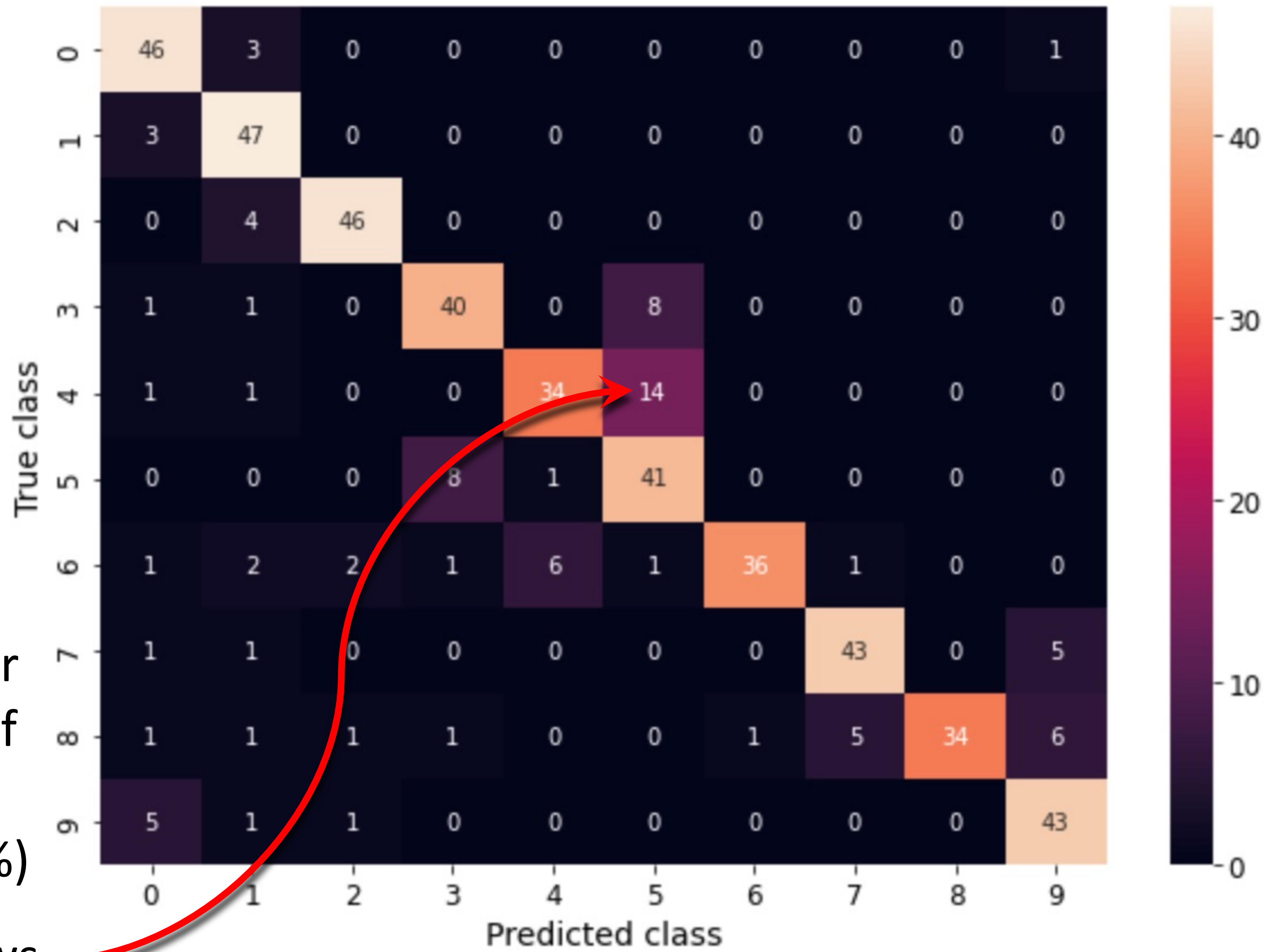
Confusion Matrix

Sum within a row = number of true samples of that class

Sum within col = number of predictions of that class

Can show raw counts or normalised (e.g. as % of true class samples so each row sums to 100%)

Large off-diagonal shows common error: here the true class 4 is predicted as class 5




```
import seaborn as sn
sn.heatmap(array, annot=True)
```

Summary

- Classification vs. regression
- Range of classifier approaches
  - SGD classifier
  - Decision Tree
- White box vs. Black box
  - Decision boundaries
- Several performance metrics
  - Precision, recall
  - ROC, AUC
  - Confusion matrix
- Multi-class classification
  - Different algorithms for combining 2-class classification
  - Different options for representing labels (esp. one-hot encoding)