MACHINE LEARNING BASICS

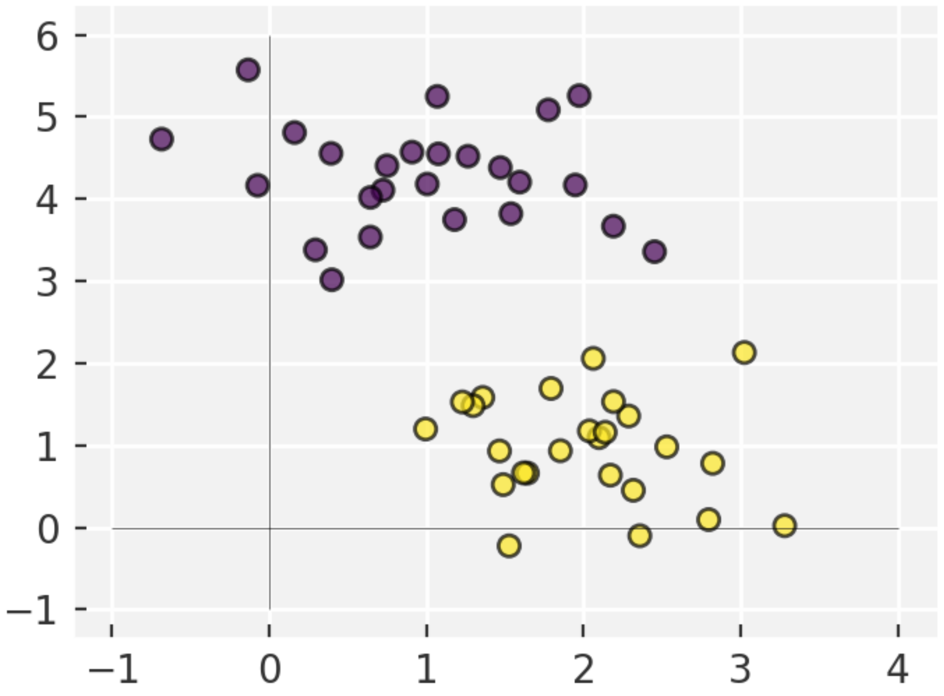
Keith Butler

Overview

- Types of ML
- Parameters and hyperparameters
- Features
- Decision trees
- Evaluation/metrics
- Overfitting
- Bagging and boosting

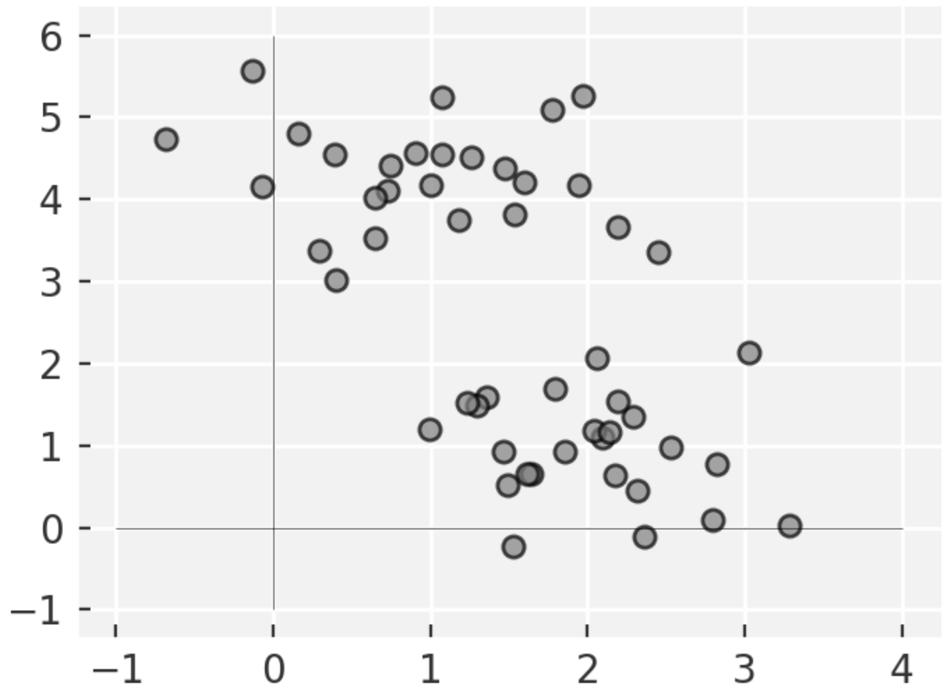
SUPERVISED ML

Learning a function that maps an input to an output based on example input-output pairs.



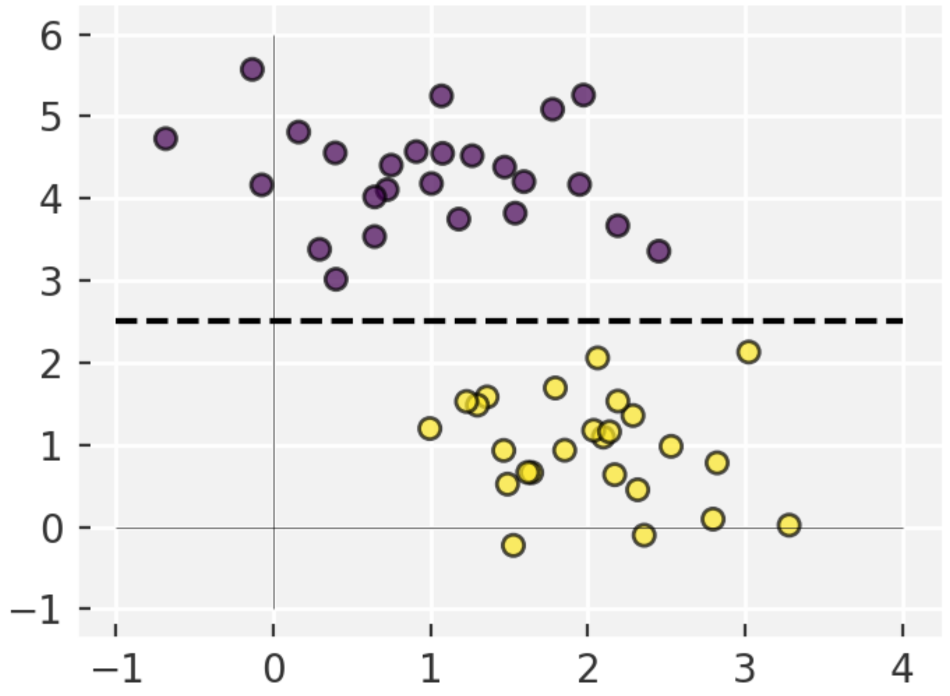
UNSUPERVISED ML

Data do not have labels, identifying trends in unlabelled datasets



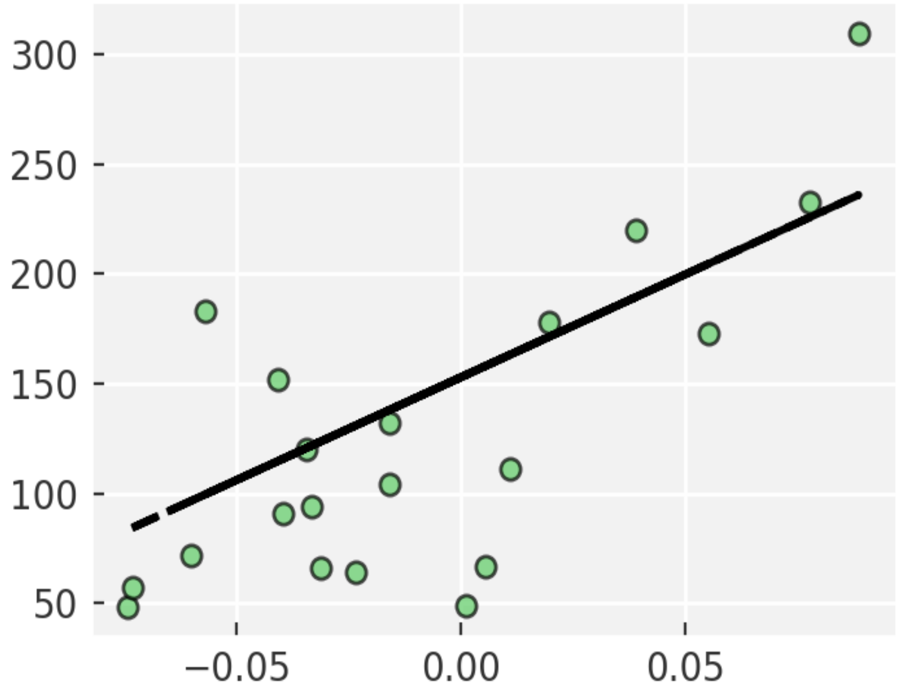
CLASSIFICATION

Identifying to which of a set of categories a new sample belongs, on the basis of a training set



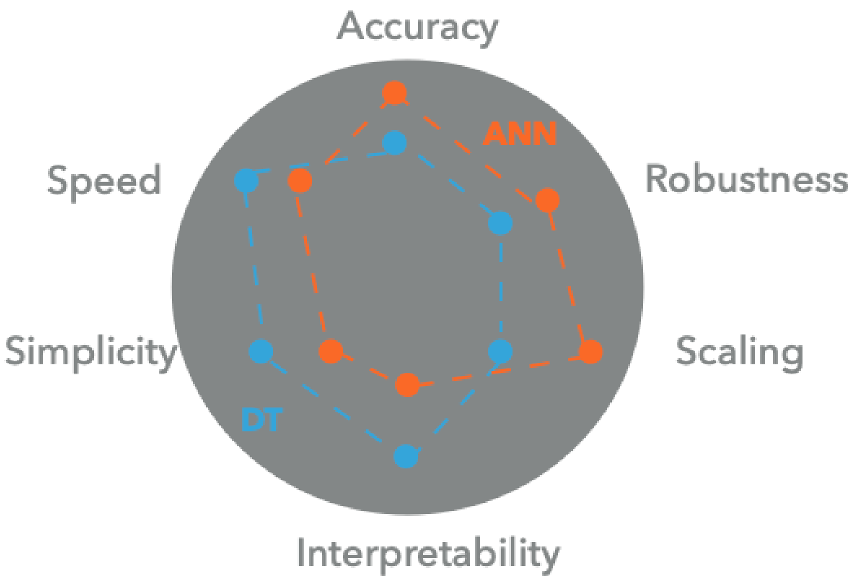
REGRESSION

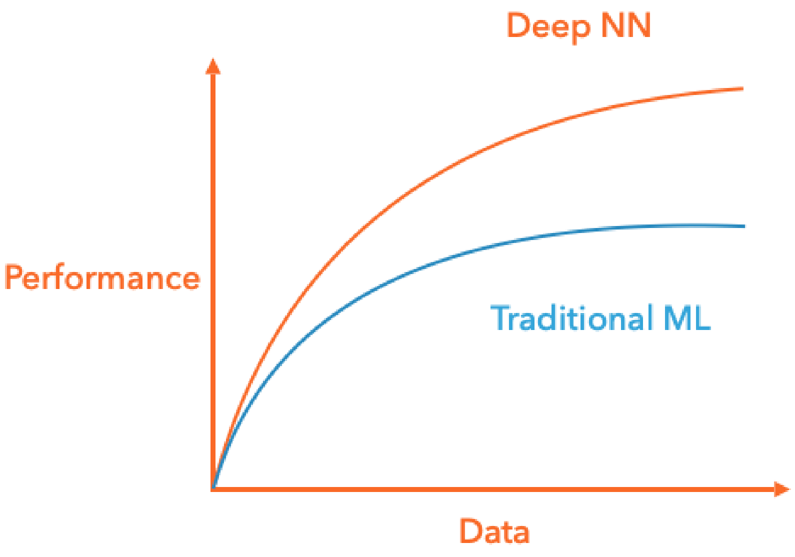
Models a target prediction value based on independent variables



CLASSICAL/DEEP METHODS

- Classical: linear regression, trees etc..
- Deep: neural network type models





PARAMETERS AND HYPER-PARAMETERS

- **Parameters** – properties of the model that are modified during training
- **Hyperparameters** – set of values that define the model and how it trains. Do not update during training
  - E.g. loss function, learning rate, number of parameters

FEATURES

- In ML approaches the data will typically consist of several or more features
- Features are simply the input variables for the model – x in $f(x) = y$

“…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”

"A Few Useful Things to Know About Machine Learning" by Pedro Domingos

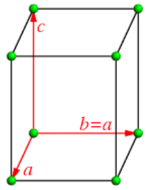
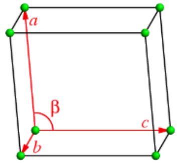
FEATURE ENGINEERING

- Transforming raw data into features that better represent the underlying problem
- Make inputs into things that an algorithm can understand
- E.g. Convert a date-time stamp into something more useful 2014-09-20T20:45:40Z -> Day: Tuesday; Year: 2014; Month: Sept
- Note that ‘Tuesday’ and ‘Sept’ are not particularly algorithm ready – how can we convert them to something more useful?

One hot encoding

Vector of length = number of categories

Each element is the probability that the data represents a given class

Material	Ortho	Rhomb
	1	0
	0	1

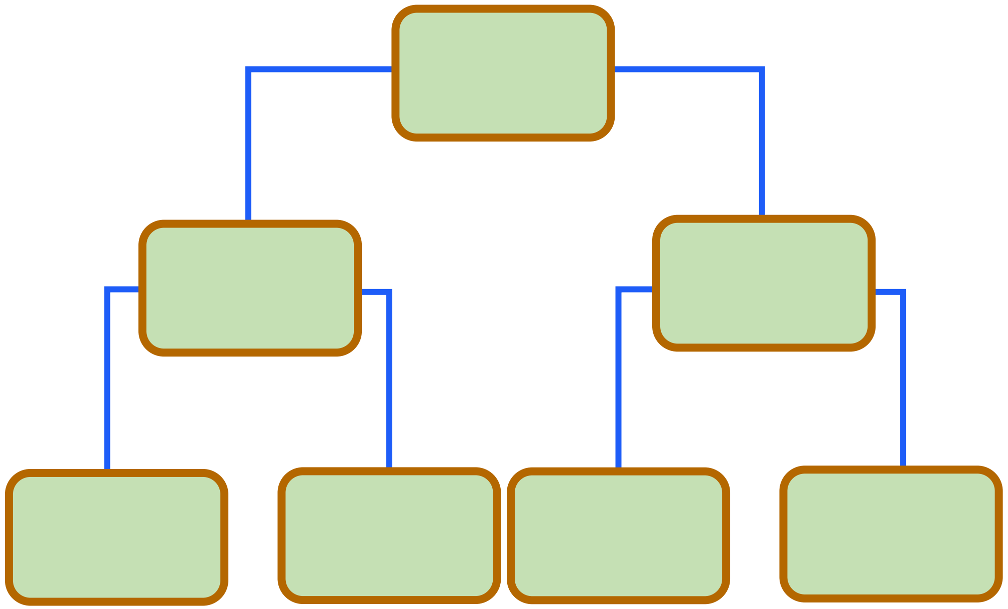
BUILDING BLOCK: ONE HOT ENCODER

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print(integer_encoded)

onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
```

DECISION TREES



Data is split by features. E.g. brightness of a pixel
Splits are arranged such that the data splits as evenly as possible at each point.

DECISION TREES

$$Q_{left}(\theta) = (x, y) | x_f \leq t_j$$
$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta)$$

Data is split according to a threshold value tj.

$$C(Q, \theta) = \frac{n_{left}}{N_j} H(Q_{left}(\theta)) + \frac{n_{right}}{N_j} H(Q_{right}(\theta))$$

The cost of the split is calculated based on some impurity function H() e.g. RMSD of the data.

$$\theta^* = \underset{\theta}{\operatorname{argmin}} C(Q, \theta)$$

The splitting parameters are chosen to minimise C at each split.

GO TO NOTEBOOK

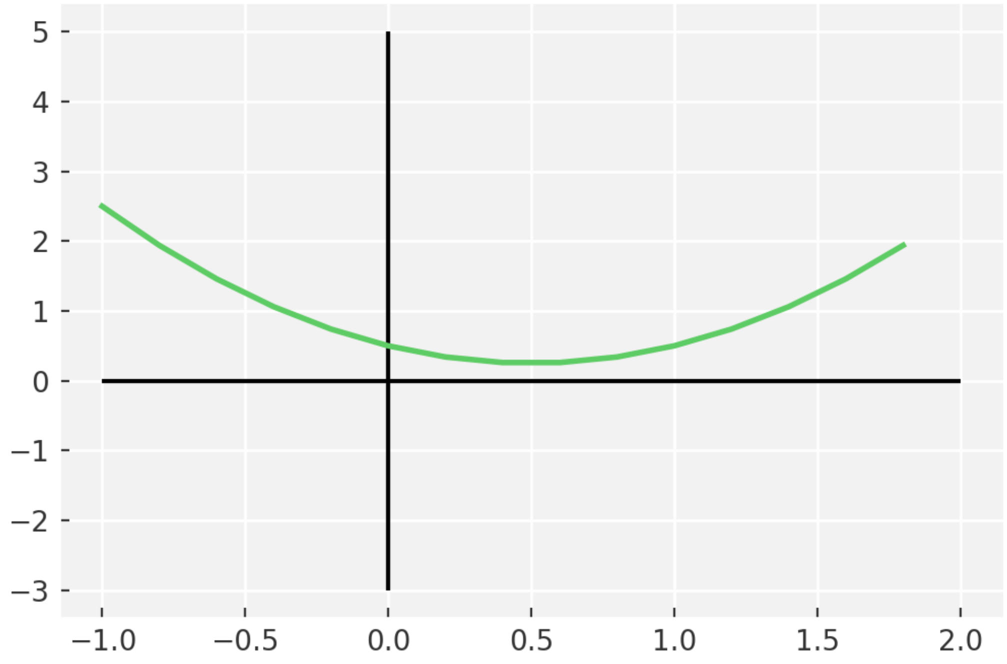
OPTIMISATION/EVALUATION

- Evaluation
  - Objective function or scoring function.
  - Distinguish good from bad models.
- Objective function = loss function = cost function
  - Must faithfully represent the “goodness” of a model in a single number

EVALUATION METRICS

$$MSE = \frac{1}{N}\sum(f_i - y_i)^2$$

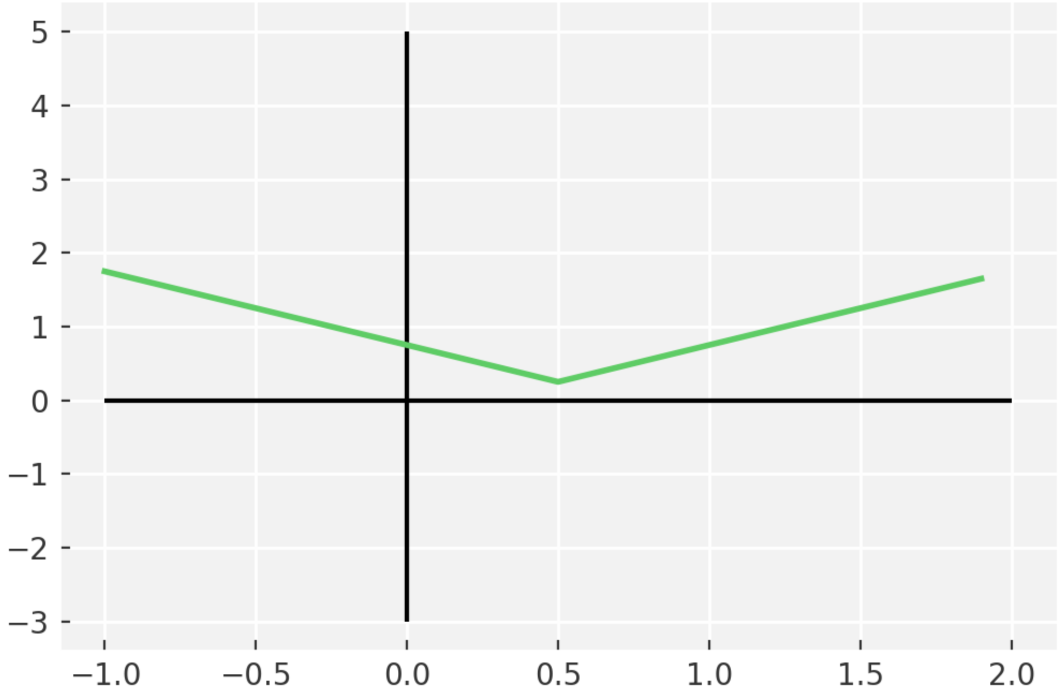
- Mean squared error
- Used in regression
- Square endures a single minimum
- Avoids local minima trapping
- Easy to calculate



EVALUATION

$$MAE = \frac{1}{N}\sum|f_i - y_i|$$

- Mean Absolute Error
- Similar to MSE
- No quadric term
- More robust to outliers
- MSE penalises large differences much more than MAE
- Large gradients close to zero - slow to optimise



EVALUATION

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

- Huber loss
- Quadratic close to the minimum
- Linear far from the minimum
- Overcomes problems of MSE and MAE
- More expensive to calculate

EVALUATION

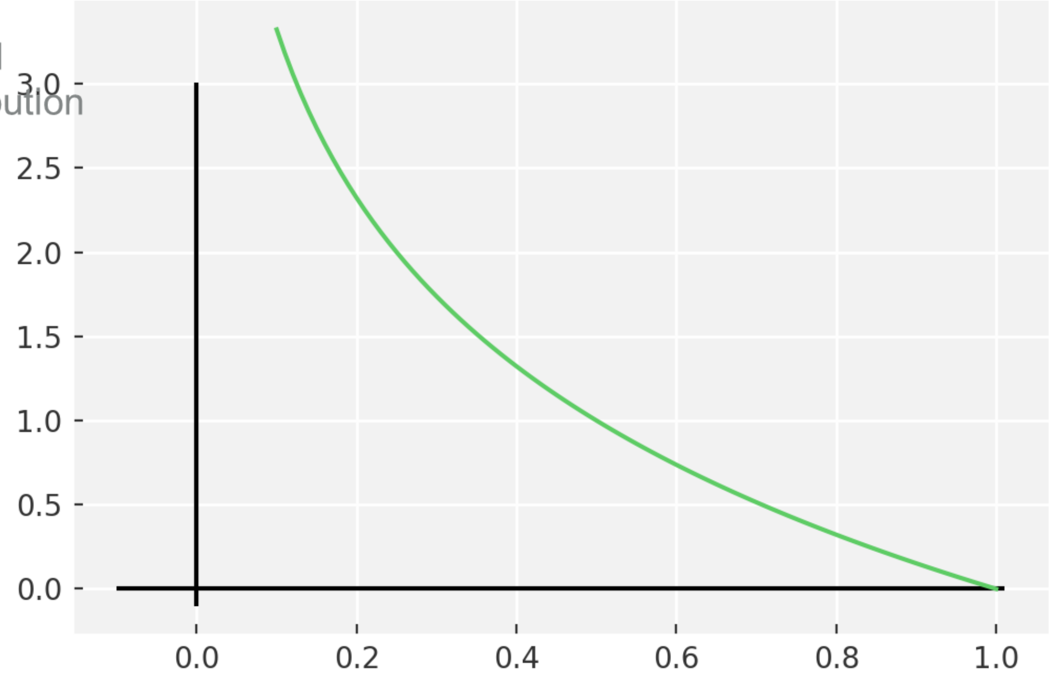
- Cross entropy
- Used for classification problems
- Tells us how similar our model distribution is to the true distribution
- Penalises all errors, but especially those that are most inaccurate

0	0	1	0
0.15	0.25	0.5	0.1

True distribution

$$H(p,q) = -\sum p_i \log_2(q_i)$$

Model distribution



EVALUATION

- Hinge loss
- Used for classification
- Does not seek to reproduce the distribution of data
- 0 as long as the classification is correct

Label(+/-1)

$$L = max(0, 1 - t \cdot y)$$

Prediction

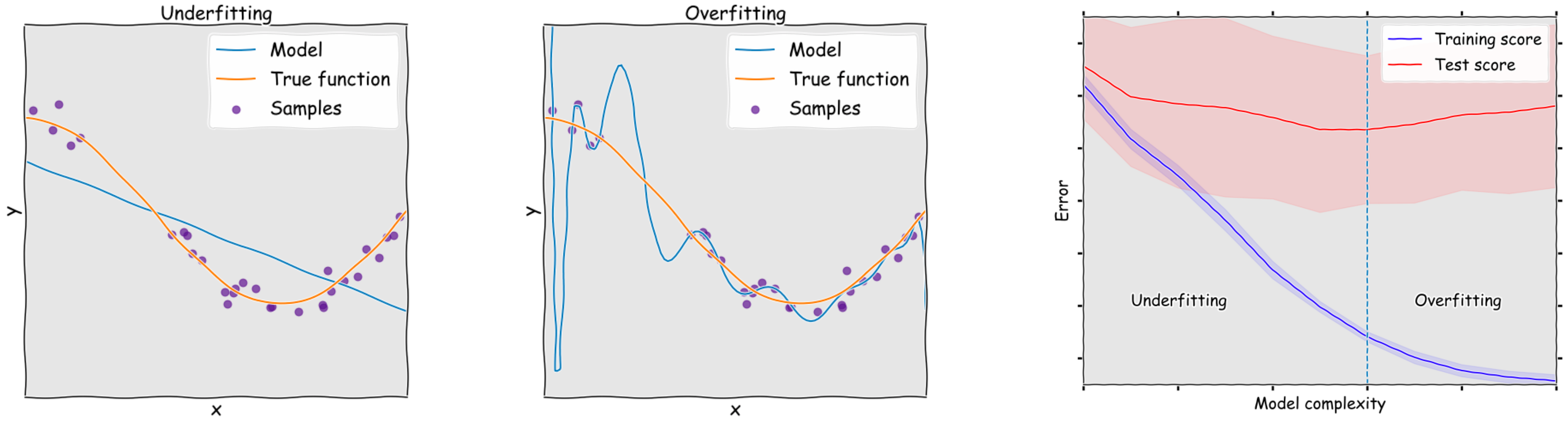
EVALUATION: TABLE OF CONFUSION

		CONDITION determined by "Gold Standard"			
	TOTAL POPULATION	CONDITION POS	CONDITION NEG	PREVALENCE $\frac{\text{CONDITION POS}}{\text{TOTAL POPULATION}}$	
TEST OUT-COME	TEST POS	True Pos TP	*Type I Error* False Pos FP	*Precision* Pos Predictive Value PPV = $\frac{\text{TP}}{\text{TEST P}}$	False Discovery Rate FDR = $\frac{\text{FP}}{\text{TEST P}}$
	TEST NEG	*Type II Error* False Neg FN	True Neg TN	False Omission Rate FOR = $\frac{\text{FN}}{\text{TEST N}}$	Neg Predictive Value NPV = $\frac{\text{TN}}{\text{TEST N}}$
	ACCURACY ACC ACC = $\frac{\text{TP + TN}}{\text{TOT POP}}$	Sensitivity (SN), Recall Total Pos Rate TPR TPR = $\frac{\text{TP}}{\text{CONDITION POS}}$	Fall-Out False Pos Rate FPR FPR = $\frac{\text{FP}}{\text{CONDITION NEG}}$	Pos Likelihood Ratio LR + LR + = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic Odds Ratio DOR DOR = $\frac{\text{LR +}}{\text{LR -}}$
		Miss Rate False Neg Rate FNR FNR = $\frac{\text{FN}}{\text{CONDITION POS}}$	Specificity (SPC) True Neg Rate TNR TNR = $\frac{\text{TN}}{\text{CONDITION NEG}}$	Neg Likelihood Ratio LR - LR - = $\frac{\text{TNR}}{\text{FNR}}$	

https://en.wikipedia.org/wiki/Confusion_matrix

EVALUATION

If a model is not expressive enough or is too expressive it is detrimental to predictive power

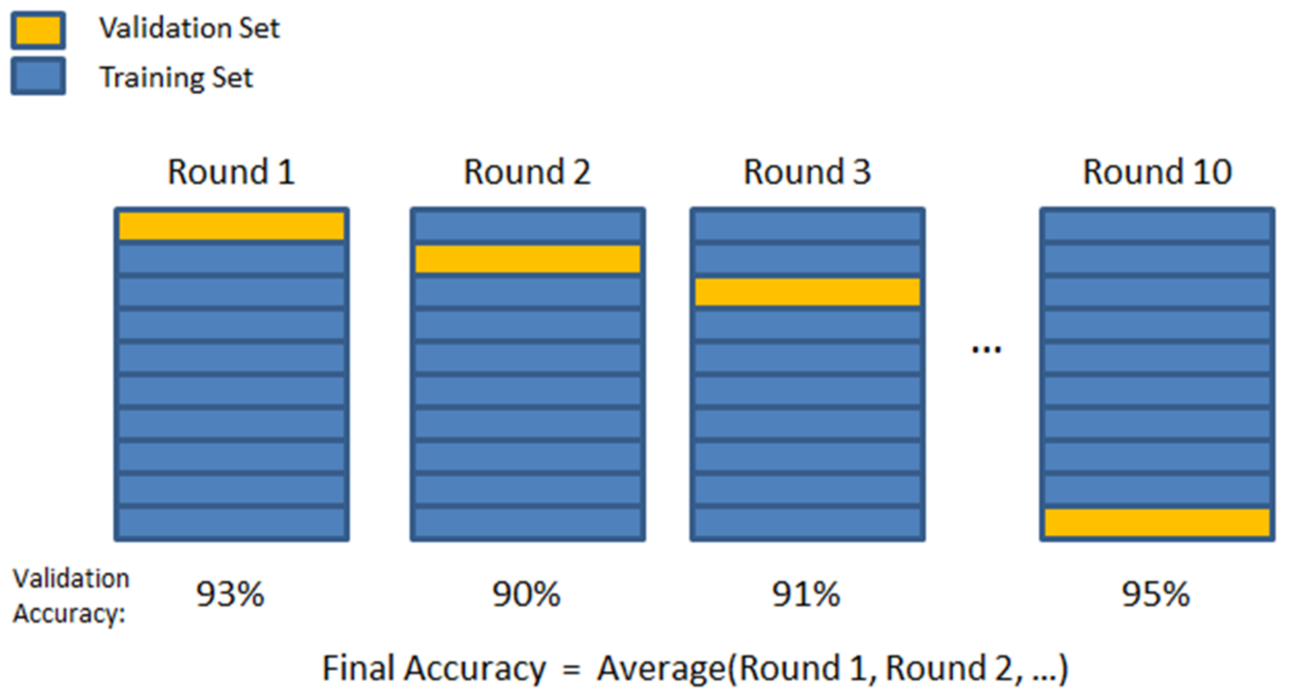


TEST AND VALIDATION SETS

- The model must always be validated on data not used for testing
- Often something like 20% of data is used for validation
- Make sure that validation and training distributions are the same (usually)

EVALUATION

n-fold cross validation allow us to use all the data for training and gives statistical errors



BUILDING BLOCK CROSS VALIDATION

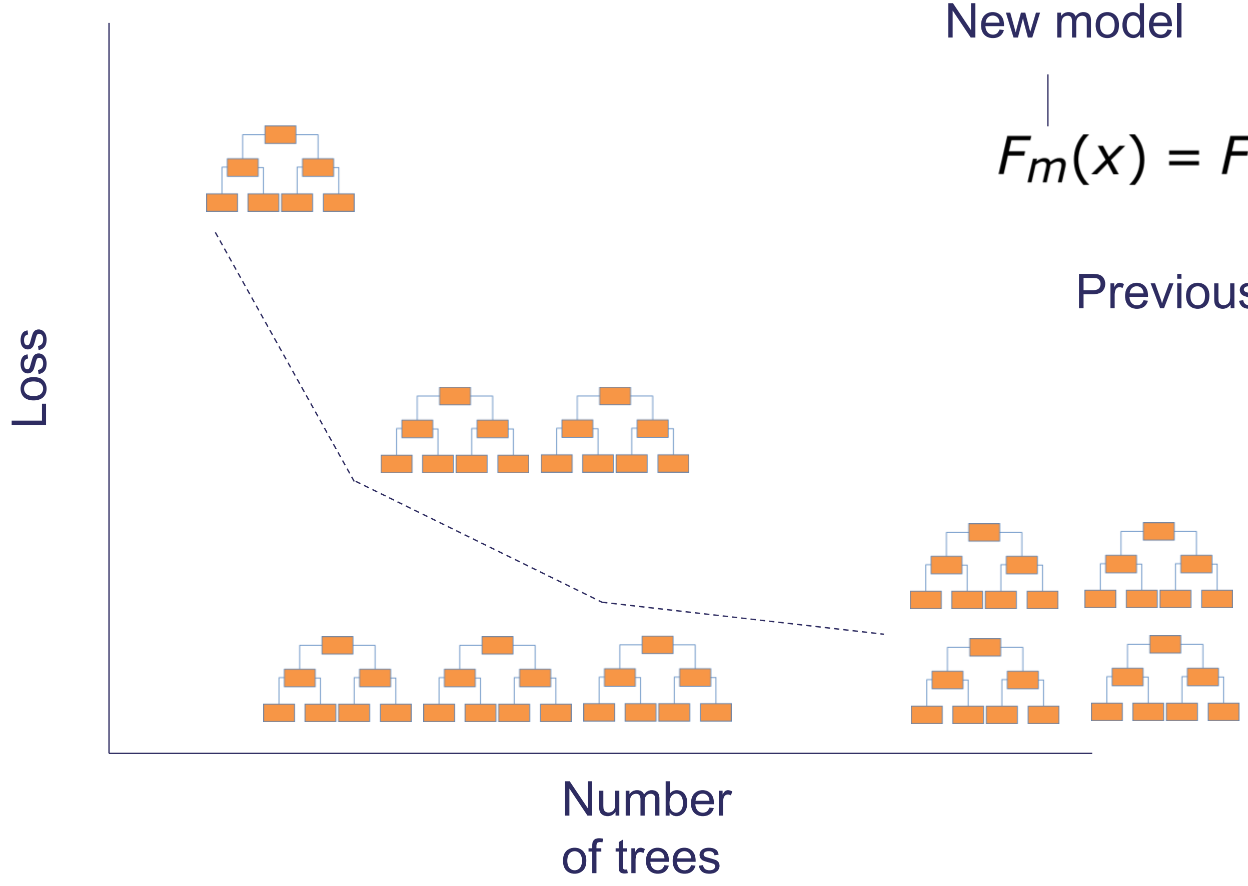
```
from sklearn.model_selection import
cross_val_score
clf = svm.SVC(kernel='linear', C=1)
scores = cross_val_score(clf, X, y, cv=5)
```

GO TO NOTEBOOK

BOOSTING + BAGGING

- To overcome the limitations of a weak learner we can use booting or bagging.
- Both methods use an ensemble of weak learners to build a strong learner
- Boosting – choose next learner based on the errors of the last learner (gradient boosted decision trees)
- Bagging – stochastically choose next learners (random forests)

BOOSTED DECISION TREES



New model

Learning rate

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

Previous model

Sub model of errors

BUILDING BLOCK: BOOSTED DECISION TREE

```
from sklearn import ensemble

gbr = ensemble.GradientBoostingRegressor(loss='lad', max_depth
= 10, learning_rate = 0.015, min_samples_split = 50,
min_samples_leaf = 1, max_features = len(cols), subsample =
0.9, n_estimators = 300)

gbr.fit(X, y)
```

GO TO NOTEBOOK

CONCEPT CHECKLIST

Supervised/unsupervised machine learning

Classical machine learning/deep learning

Parameters/hyperparameters

Features and feature engineering

Decision trees

Overfitting

Evaluation/metrics

Test/train split, cross-validation

Bagging and boosting

THANK YOU

mdi-group.github.com