

# Data Mining (Week 1)

dm24s1

## Topic 06 : Data Modelling

### Part 01 : Data Modelling - Introduction

Preparation

Dr Bernard Butler

Department of Computing and Mathematics, WIT.  
(bernard.butler@setu.ie)

Data Handling

Exploring Data

Exploring Data 2

Building Models

Autumn Semester, 2024

#### Outline

- Components of a machine learning problem
- Machine learning concepts and notation
- Bias vs variance
- Learning curves
- Regularisation

Wrap up

# Data Mining (Week 6)

Introduction

Motivating Example

## Preparation

Data Handling

Exploring Data 1

Exploring Data 2

Building Models

## Prediction

Clustering

Regression  
1

Classification  
1

Regression  
2

Classification  
2

Wrap up

# Three Components of a Machine Learning Problem

It is easy to get lost among the multitude of choices one needs to make when given data mining problem.  
A good decomposition is the following:

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
$K$ -nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

---

A Few Useful Things to Know about Machine Learning, Domingos, 2012.

## Three Components of a ML Problem — Representation

Representation	Evaluation	Optimization
Instances <i>K</i> -nearest neighbor Support vector machines	Accuracy/Error rate Precision and recall Squared error	Combinatorial optimization Greedy search Beam search

**Representation** refers to formulating the problem as a machine learning problem — typically a **classification** problem, a **regression** problem or a **clustering** problem.

- How do we represent the input?
- What **features** to use?
- How do we learn additional features?
- With each type of problem, we have multiple subtypes:  
For example which classifier? a **decision tree**, a **neural network**, a **support vector machine**, etc.

## Three Components of a ML Problem — Evaluation

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
$K$ -nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search

**Evaluation** refers to an **objective function** or a **scoring function**, to distinguish a good model from a bad model.

- For a classification problem, we need this function to know if a given classifier is good or bad. A typical function can be based on the number of errors made by the classifier on a test set, using precision and recall.
- For a regression problem, it could be the squared error, or likelihood. Do we include regularisation? etc

# Three Components of a ML Problem — Optimisation

Representation	Evaluation	Optimization
Instances <i>K</i> -nearest neighbor Support vector machines	Accuracy/Error rate Precision and recall Squared error	Combinatorial optimization Greedy search Beam search

**Optimisation** is concerned with searching among the models in the language for the highest scoring model.

- How do we search among all the alternatives?
- Can we use some greedy approaches, branch and bound approaches, gradient descent, linear programming or quadratic programming methods.

# Data Modelling (aka Machine Learning)

As alternative to the three component (Representation / Evaluation / Optimisation) viewpoint we can think of a machine learning problem as

## Definition 1 (Machine Learning)

Study of algorithms that improve their performance  $P$  at some task  $T$  with experience  $E$ .

Well defined learning task:  $\langle P, T, E \rangle$

- What metric should be used to measure performance?
- What cost function should be used?
- What is the cost of incorrect prediction?
- Computational cost?

- How complex is the task?
- Task type: classification, regression, ...
- Linear vs nonlinear?
- What family of functions should be used?

- How many historical observations are needed?
- How accurate/noisy is the data?
- Do we have missing values?
- Is the data representative?

# Taxonomy of Machine Learning Models ...

## ...by Intuition/Motivation

- **Geometric models** use intuitions from geometry such as separating (hyper-)planes, linear transformations and distance metrics.
- **Probabilistic models** view learning as a process of reducing uncertainty, modelled by means of probability distributions.
- **Logical models** are defined in terms of easily interpretable logical expressions.

## ...by Algorithmic Properties

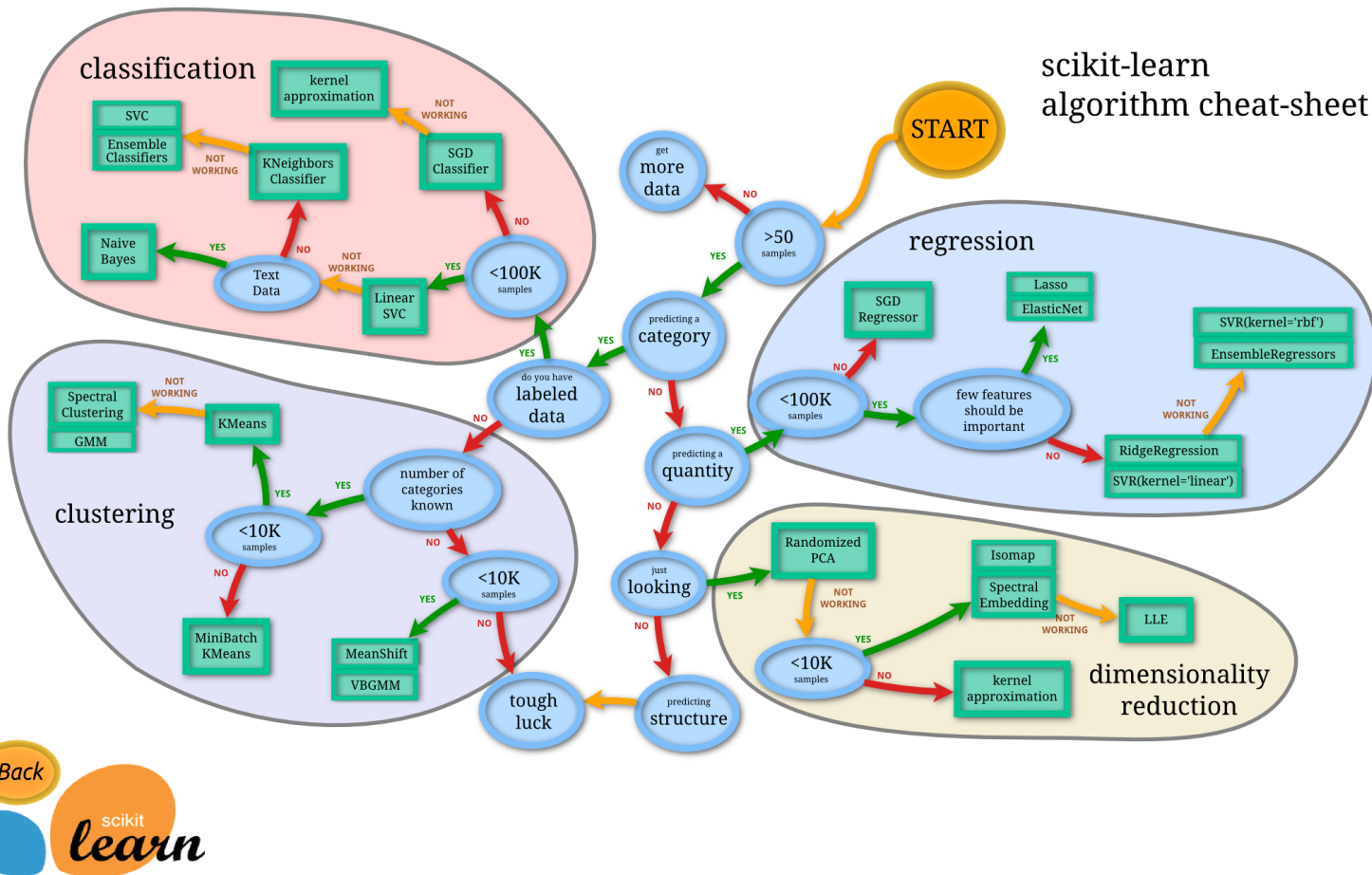
- **Regression models** predict a numeric output.
- **Classification models** predict a discrete class value.
- **Neural networks** learn based on a biological analogy
- **Local models predict** in the local region of a query instance.
- **Tree-based models** (recursively) partition the data to make predictions.
- **Ensembles** learn multiple models and combine their predictions.

## ...by Fixed/Variable Number of Parameters

- **Parametric models** have a fixed number of parameters.
- In **non-parametric models** the number of parameters grows with the amount of training data.



# Aside: Scikit-learn Flowchart of Models (Shallow Learners)



A neural network with more than one hidden layer is called a **deep learner**, all other learners are **shallow learners**.

# Statistical Models vs Machine Learning Models

## Data

### Statistical Models

- Usually small ( $< 1000$  observations)
- Low dimension ( $< 10$  variables)
- Can have detailed understanding of data
- Data is clean — human has looked at each data point

## Models

- Simple models — complexity limited by theory
- Detailed/complex statistical assumptions re data
- Model known, and data is carefully examined to verify assumptions.

## Validation

- Evaluation based on theoretical estimates under stated statistical assumptions
- Analysis of errors using theoretical distributions

Statistics would be very different if it had been born after the computer instead of 100 years before

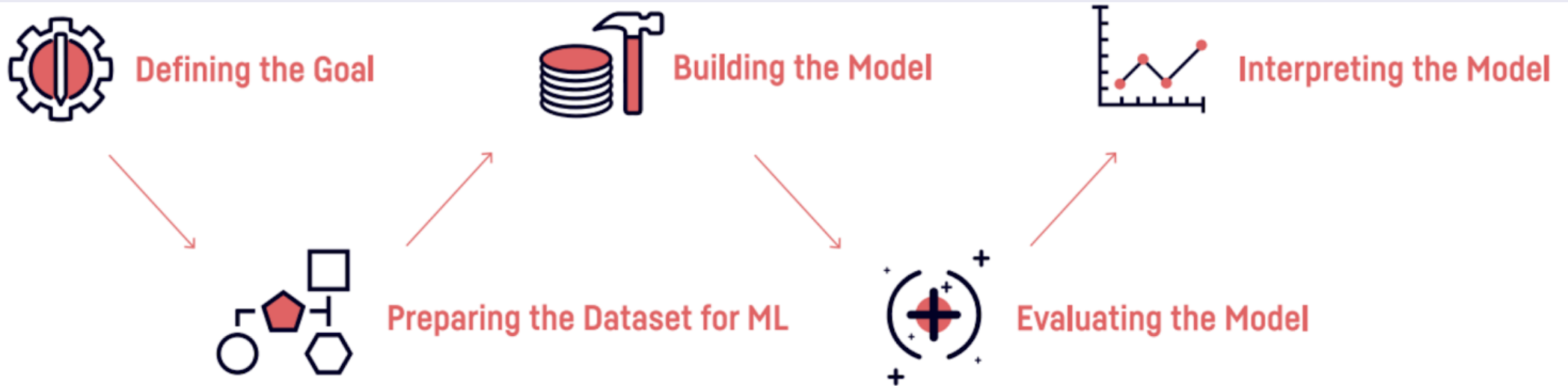
### ML Models

- Can be huge (million+ observations)
- Large dimension (1000+, more for vision)
- Too large for human to parse / understand
- Data not clean — humans can't afford to understand/fix each point
- “No” upper limit on model complexity
- Fewer statistical assumptions re data
- Don't know right model? No problem! have multiple models and vote/weight results
- Empirical evaluation methods instead of theory — how well does it work on **unseen** data?
- Don't calculate expected error, measure it from **unseen** data.

Splitting data into train+test(+validation) is vital

# The Pipeline Metaphor

## Model Building Pipeline



*Source: Dataiku*

## Comments

- We saw the first two stages in previous weeks
- This week we look at the remaining stages
- Of course this pipeline is a simplification. In reality it is iterative.

# What does a (supervised learning) model look like?

## Definition 2 (Linear Model)

General form of linear model used in this module looks like

$$y_i \sim f_i^{(1)} + f_i^{(2)} + \dots + f_i^{(n)}$$

where  $y_i$  is the value of the response variable for observation  $i$ , and  $f_i^{(j)}$ ;  $j = 1, \dots, n$  is the value of the  $j^{\text{th}}$  feature for that observation.

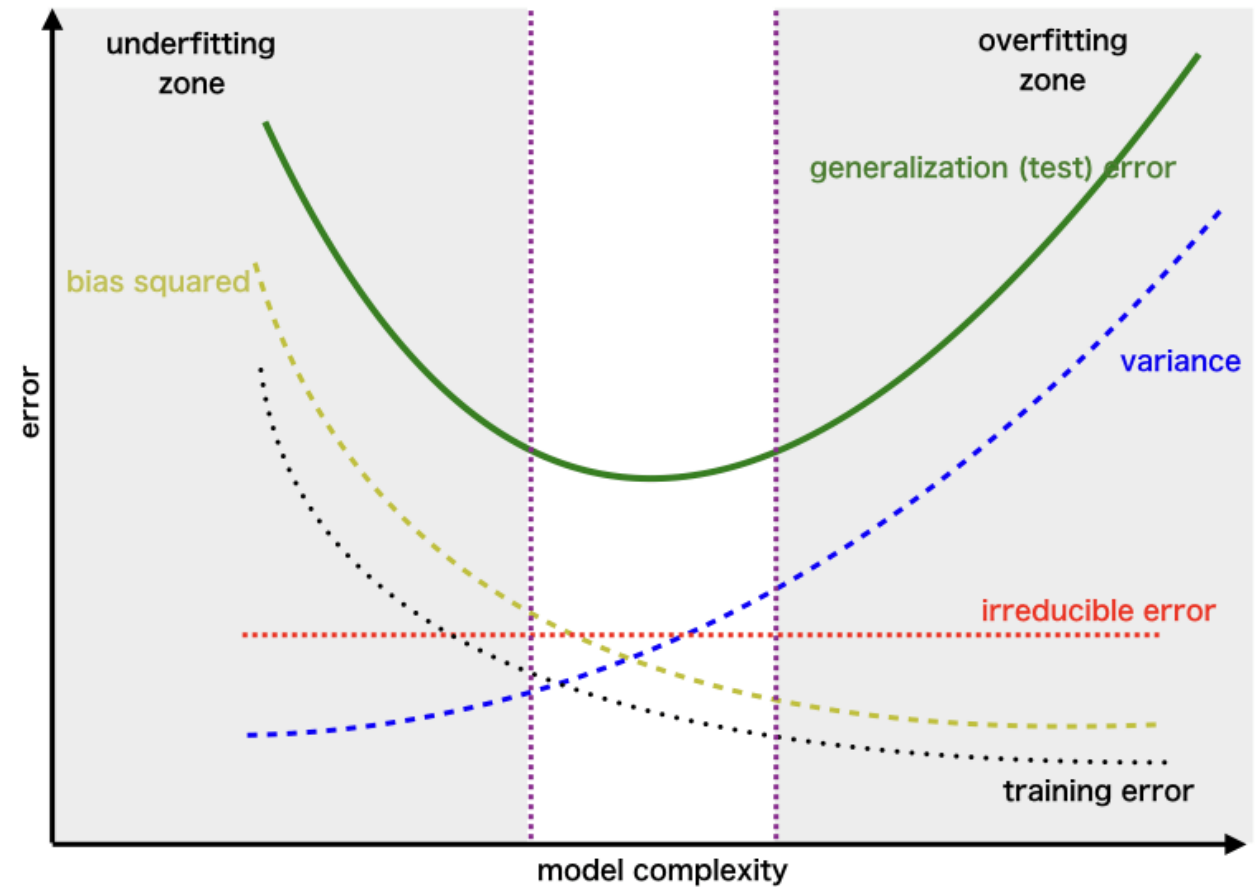
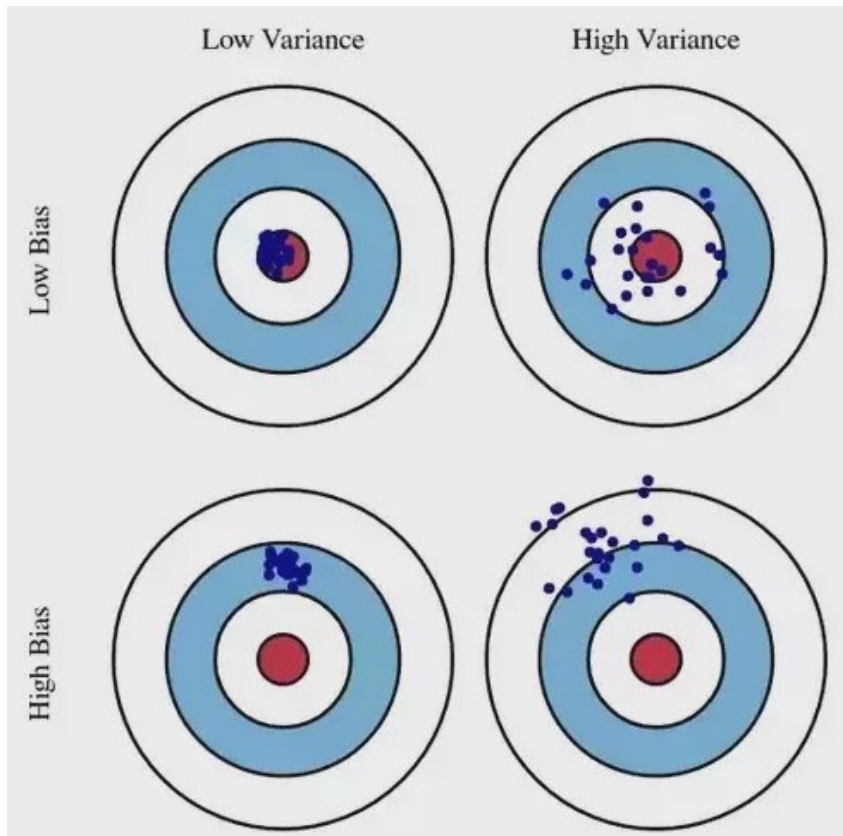
The model is linear in the sense that it can be turned into the following linear equation:

$$y_i = a_0 + a_1 f_i^{(1)} + a_2 f_i^{(2)} + \dots + a_n f_i^{(n)} + \varepsilon_i$$

Note that the features  $f$  can be nonlinear but the model parameters  $a$  must appear linearly.

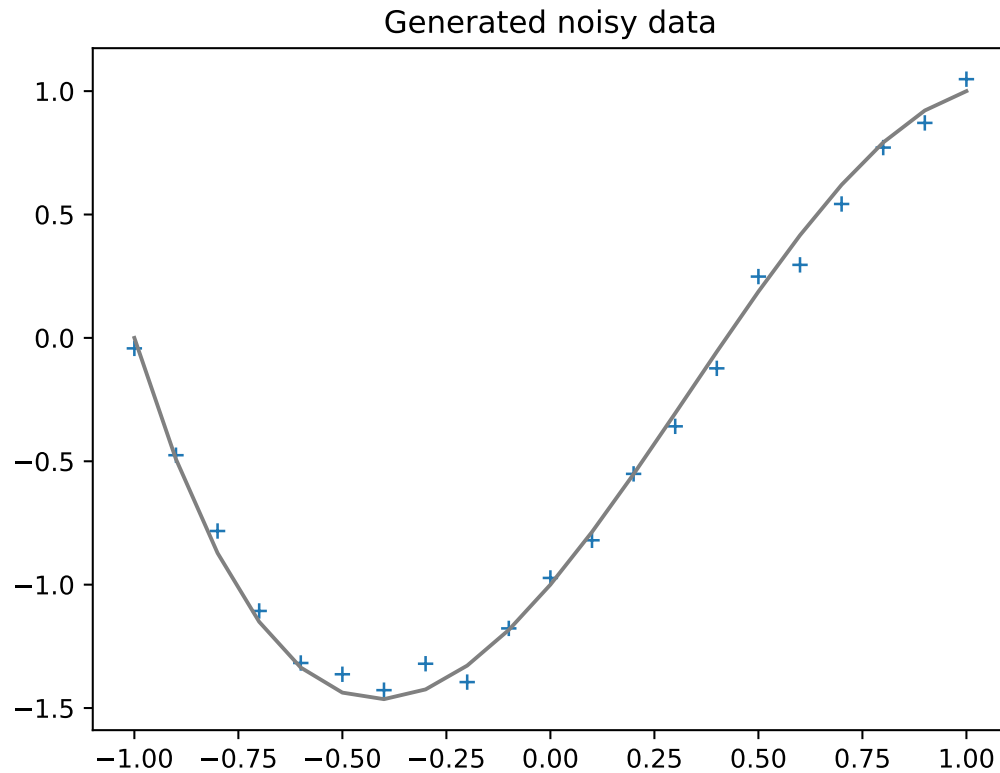
The goal of modelling is to find  $a$  so that the *prediction error* is a minimum.

# Bias-Variance and Total Error



Look for  $a$  that minimise the generalization error (estimated using the test set)

## Example: Noisy data



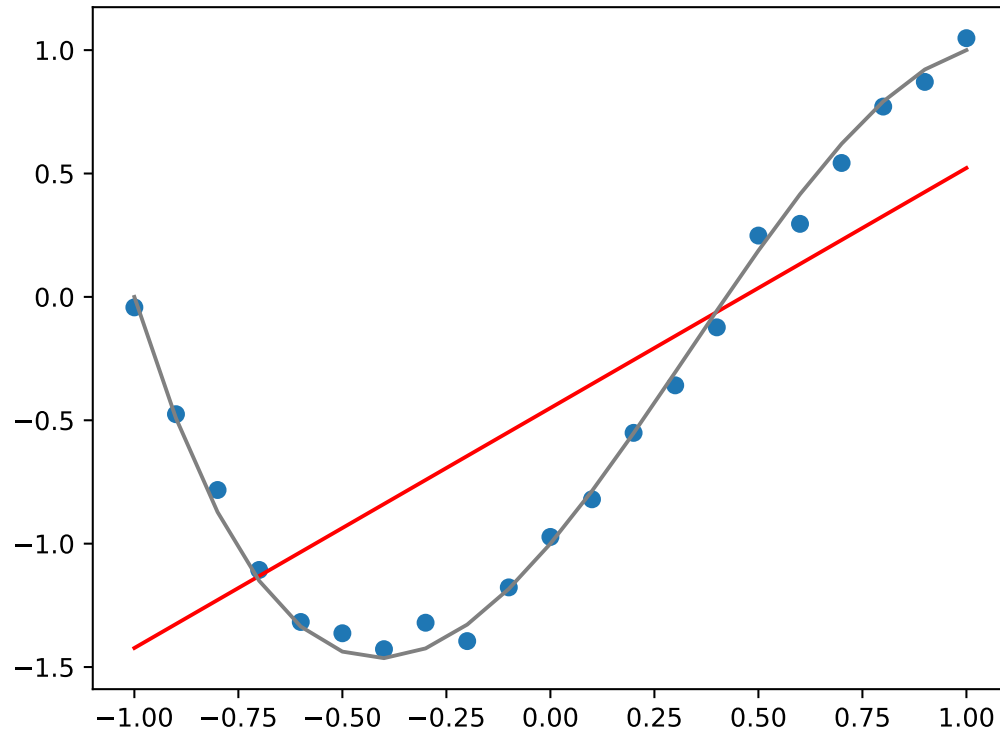
### Comments

- Given data with some error (noise)
- Expected underlying model is indicated by the grey curve
- In the next slides we will compare different models, indicated by red curves
- The models have different numbers of *features*
- The values predicted by each model lie on the red curve
- The **loss function** is an estimate of how much the grey and red curves differ

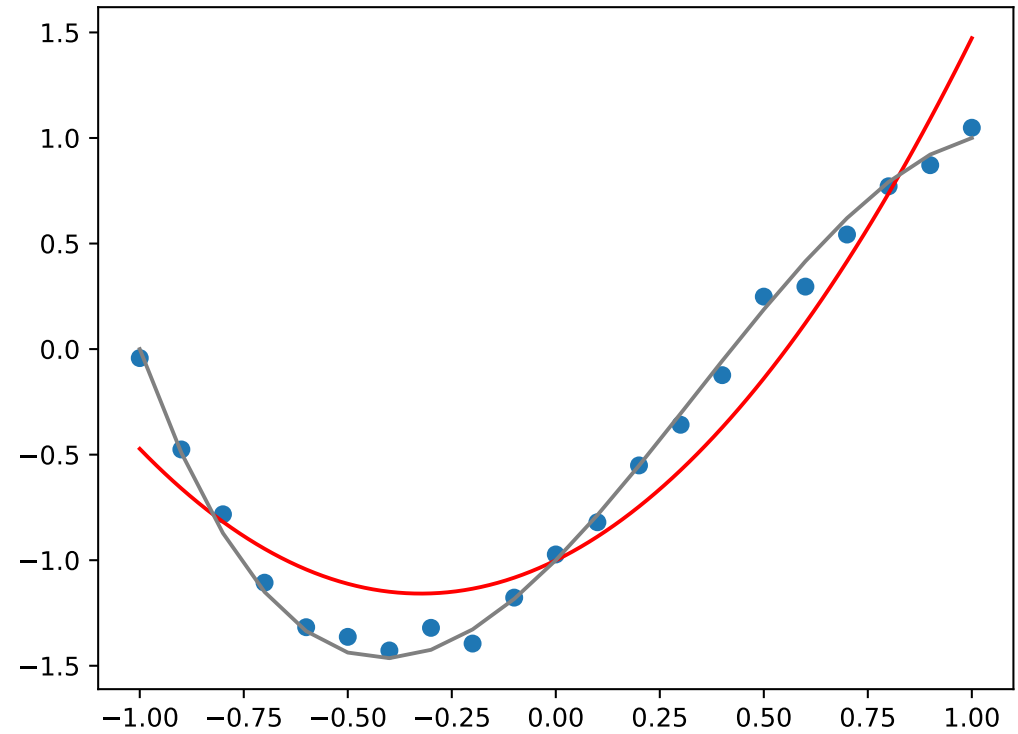
➤ Look for the number of features that minimise the loss function

# High Bias, Low variance

Data and its fit with 2 features



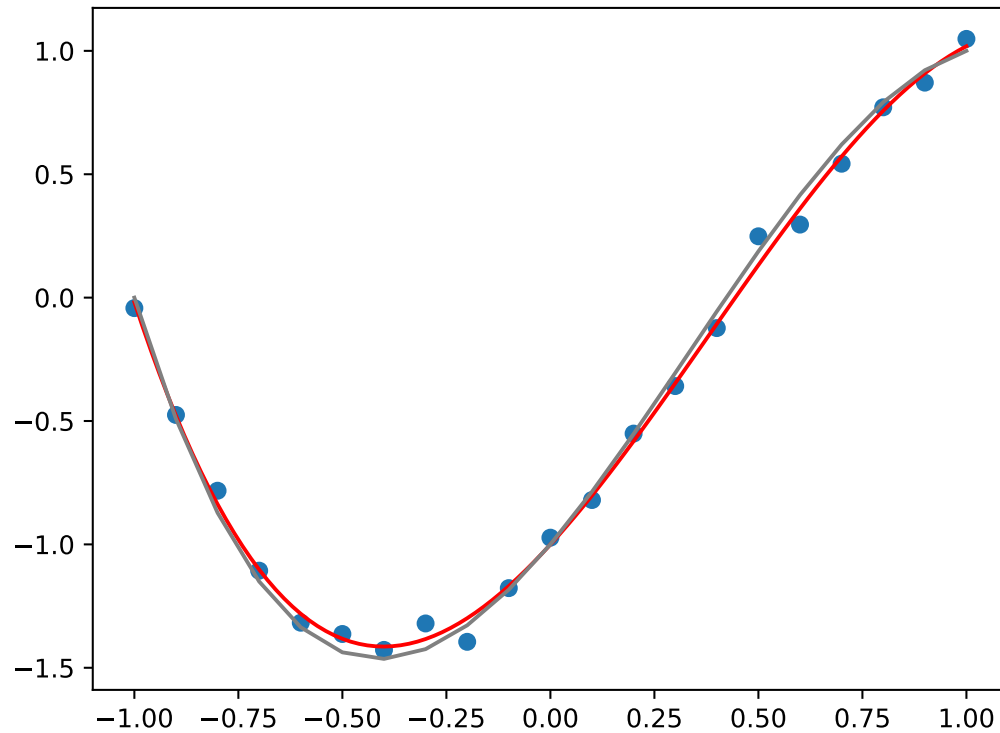
Data and its fit with 3 features



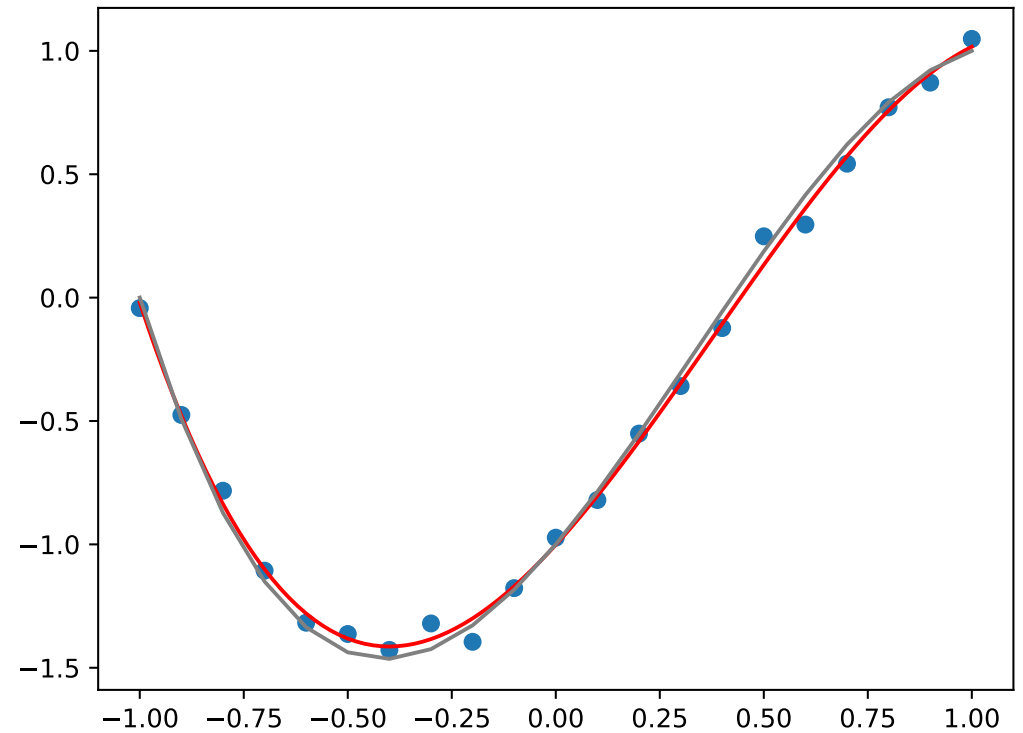
Need more features...

# Low Bias, Low variance

Data and its fit with 4 features



Data and its fit with 5 features

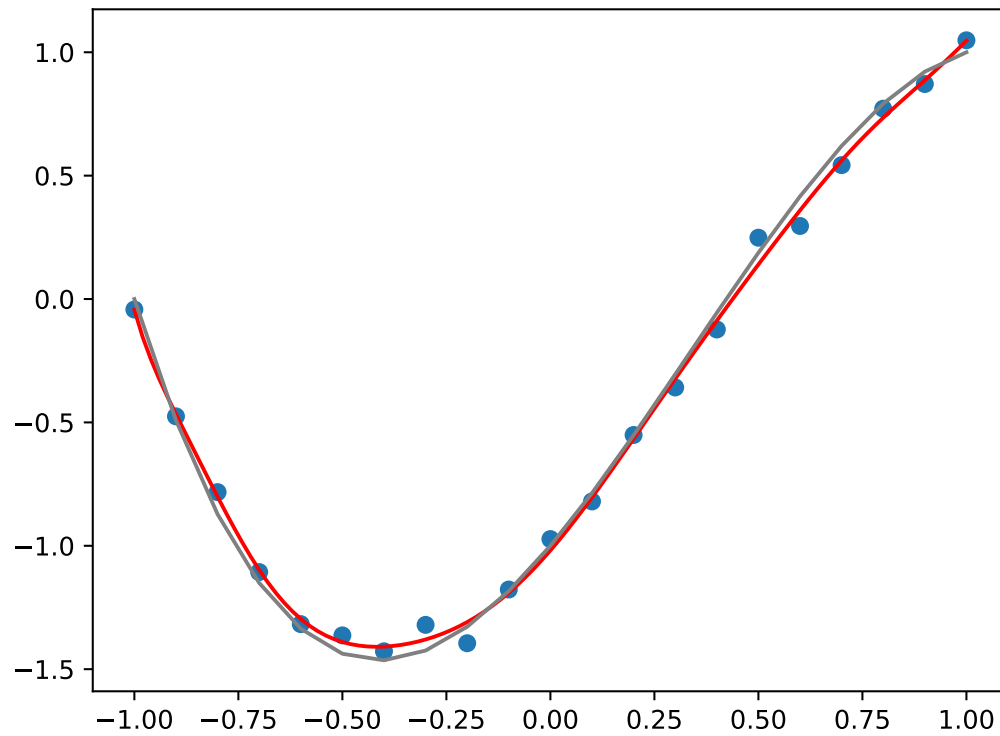


About the right number of features...

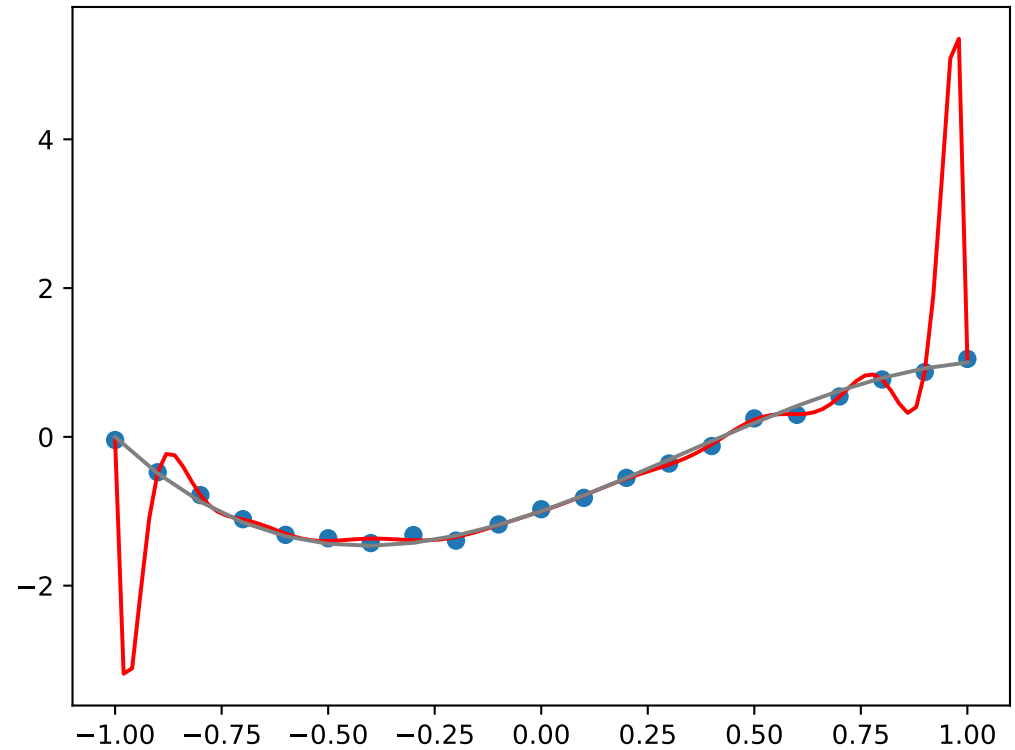


# Low Bias, High variance

Data and its fit with 12 features



Data and its fit with 18 features



Too many features...

# Example Model Types

Model	Applications	Concerns
Logistic Regression	X-ray classification	Regression with transformed variable
Fully connected networks	Classification	Classical ANN: choose encoding and size
Convolutional Neural Networks	Image processing	deep learning - choose segmentation
Recurrent Neural Networks	Voice recognition	ANN with feedback - how much?
Random Forest	Fraud Detection	Ensemble method - how many?
Reinforcement Learning	Learning by trial and error	Choose goal and penalties
Generative Models	Image creation	Choose parameters
K-means	Segmentation	Choose distance function and $k$
k-Nearest Neighbors	Recommendation systems	Choose distance function and $k$
Bayesian Classifiers	Spam and noise filtering	Deal with imbalances

# Before you start...

Does a *pre-trained* model exist?

## Transfer Learning

- Building a model from scratch is resource-intensive
- Open source data and model exist, particularly for deep learning (not in this module)
- Most frameworks provide example models that can be used as a template
  - Select a similar model
  - Prune it (remove unnecessary terms)
  - Train using the pruned model as a starting point

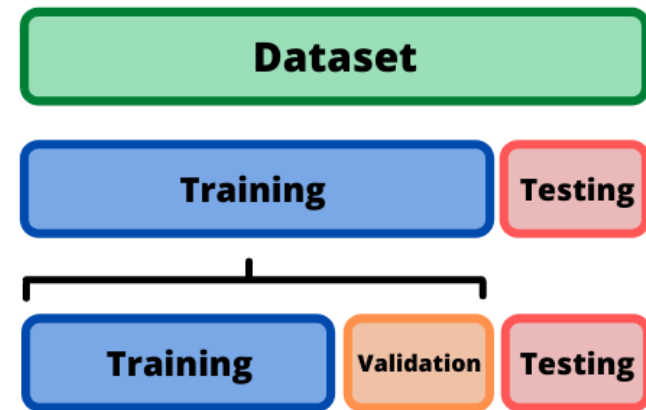
# Training, test and valuation subsets: 3-way Holdout

## Why Split?

Hold back some data to check how the model is doing.

- **Training** data is sample used to fit the model parameters.
- **Test** data is sample used to test the final model fitted to the training data.
- **Validation** data is sample used to test each interim model while tuning it.

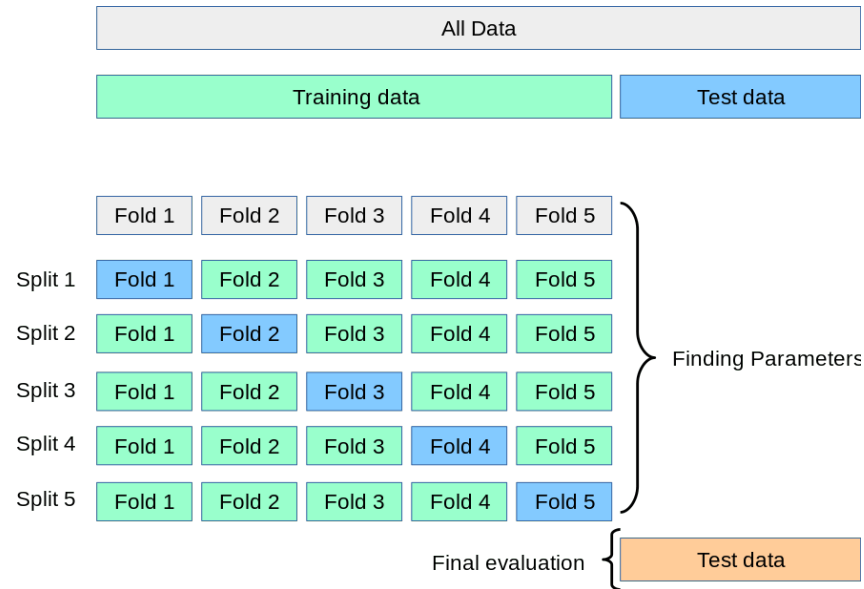
## Typical Splits



## sklearn example

```
from sklearn.model_selection import train_test_split
trainVal, test = train_test_split(df, test_size=0.2, seed=42)
train, validation = train_test_split(trainVal, test_size=0.1)
```

# K-fold cross validation



Source: [https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)

## sklearn example

```
from sklearn.model_selection import cross_val_score
# clf is some classifier, X and y are the features and target of the training set
scores = cross_val_score(clf, X, y, cv=5)
```

`scores` is a  $k = 5$  element array, can be used to estimate the prediction error (or other score) while building a model

# Featuring engineering 1: Scaling of numerical variables

## Scaling - what it does

- If numeric features have different scales, e.g.  $[-0.005, -0.003]$  and  $[10000, 10001]$  some terms dominate, others are “lost”
- Better: transfer the scaling from the feature to the model parameter
- A min-max scaling is often a good choice:

$$\tilde{X} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

- Note that  $X$  is in the range  $[X_{\min}, X_{\max}]$  but  $\tilde{X}$  is in the range  $[0, 1]$ .
- Other options include StandardScaler (subtract mean and divide by standard deviation) and a max-abs scaler (scales to  $[-1, 1]$ )

## sklearn example

```
from sklearn.preprocessing import MinMaxScaler
# df is a dataframe with numeric features
scaler = MinMaxScaler()
dfScaled = scaler.fit(df)
```

`dfScaled` can be used instead of `df` with the advantage that the fitted parameters are more accurate.

## Feature Engineering 2: Choice of Features

---

- How many to include? Use metrics to decide. Will see some when considering regression and classification.
- How do we handle different feature types? Need to encode categorical variables.
- Can we derive new numeric features? Yes,  $f' = \log(f)$  etc. is possible

# Using Categorical Features in (Logistic) Regression

How can Categorical-valued features participate in linear models?

Given the following fragment of a dataset, where the goal is to predict the salary of employees in a large organisation:

```
df = pd.read_csv('data/team.csv',\
                 index_col="Name")
df
```

Role Skilled Salary			
Name			
Alice	Designer	Yes	40000
Bob	Programmer	No	25000
Carol	Tester	No	30000

How can this data be represented by a linear model, where all quantities must take numeric values?



## Using pandas .getdummies() on a binary-valued column

```
dfSkilledDummies = pd.get_dummies(df['Skilled'],\n    prefix='Skilled',\n    dtype=int)\ndfSkilledDummies
```

	Skilled_No	Skilled_Yes
Name		
Alice	0	1
Bob	1	0
Carol	1	0

Note that a binary-valued column becomes 2 dummy columns

## Reducing redundancy (by 1) in 2 dummy columns

```
dfSkilledIndicators = pd.get_dummies(df['Skilled'],\
    prefix='Skilled',\
    drop_first=True,\
    dtype=int)\
    .rename(columns={"Skilled_Yes": "IsSkilled"})
dfSkilledIndicators
```

IsSkilled	
Name	
Alice	1
Bob	0
Carol	0

➤ A single indicator column can replace a group of 2 dummy columns

## Using pandas .getdummies() on a multi-valued column

```
dfRoleDummies = pd.get_dummies(df['Role'],\n                                prefix='Role',\n                                dtype=int)\ndfRoleDummies
```

	Role_Designer	Role_Programmer	Role_Tester
Name			
Alice	1	0	0
Bob	0	1	0
Carol	0	0	1

➤ Note that an  $n$ -valued column becomes  $n$  dummy columns

## Reducing redundancy (by 1) in $n$ dummy columns

```
dfRoleIndicators = pd.get_dummies(df['Role'],\
    prefix='Role',\
    drop_first=True,\
    dtype=int)\
    .rename(columns={\
        "Role_Programmer": "IsProgrammer",\
        "Role_Tester": "IsTester"})
```

dfRoleIndicators

	IsProgrammer	IsTester
Name		
Alice	0	0
Bob	1	0
Carol	0	1

$n - 1$  indicator columns can replace a group of  $n$  dummy columns

# Deriving and using dummy/indicator features

- Identify potential categorical features in EDA Pass 1
- Identify whether each feature is (potentially) *usable* in EDA Pass 2
- Identify whether each feature is (potentially) *useful* in EDA Pass 3
- Add all potentially usable and useful features (regardless of type) to a list  $F$
- For each categorical feature  $f_j$  in  $F$  having  $n$  levels
  - Derive  $n - 1$  indicator features  $\tilde{f}_j^k$ , where  $k = 1, \dots, n - 1$
  - Replace the original categorical feature  $f_j$  in  $F$  with the derived indicator features  $\tilde{f}_j^k$ .
- Build the model using the features in  $F$ .

# Summary

---

- We have reviewed different types of models and considered their general form.
- We looked at the goals of modelling: minimise predictive error.
- We considered how feature engineering can help.
  - Scaling numerical features, so that variation is treated fairly between features.
  - Choosing a subset of features (more to come in future weeks...), looking for the sweet spot between under- and over-fitting.
  - Encoding categorical features as numerical dummy features (more to come in future weeks...), so they can participate in linear models
- In subsequent weeks we will put this theory into practice.

# Resources

---

- **A Summary of the Basic Machine Learning Models**

[towardsdatascience.com/a-summary-of-the-basic-machine-learning-models-e0a65627ecbe](https://towardsdatascience.com/a-summary-of-the-basic-machine-learning-models-e0a65627ecbe)

- **Train-Test Split for Evaluating Machine Learning Algorithms**

[https://machinelearningmastery.com/](https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms)

[train-test-split-for-evaluating-machine-learning-algorithms](https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms)

This week I have focused on the theory rather than its (python) implementation. This is a nice article that covers the implementation side of things.

- **Cross-Validation: Estimator Evaluator**

[medium.com/swlh/cross-validation-estimator-evaluator-897d28afb4ff](https://medium.com/swlh/cross-validation-estimator-evaluator-897d28afb4ff)

Nice article that covers cross-validation in a lot more detail — we will be using many of these variants in later weeks, especially k-fold stratified.