March 14, 2018

Introduction to Data Science

Zeev Waks, Intel

Model evaluation

Advanced Analytics
We mine your business

Outline

- Quick ML review
- Model evaluation
  - Performance metrics
  - Hold-out evaluation
  - Cross validation
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature selection
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

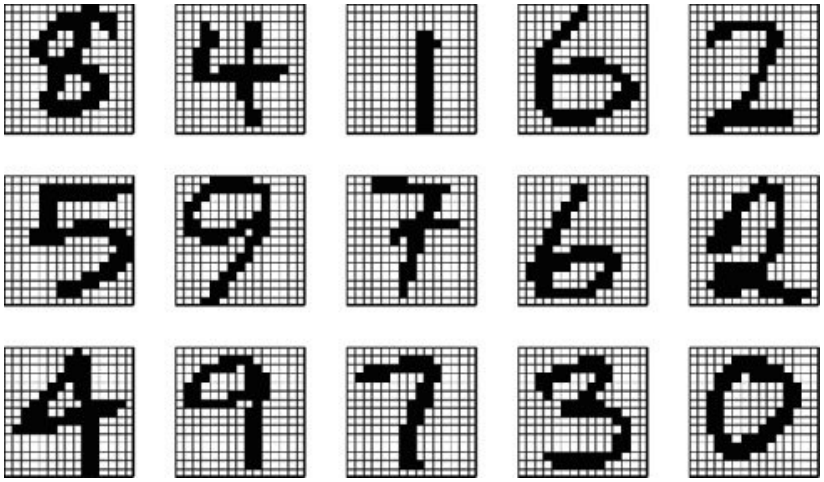
Agenda - Quick ML review

- Quick ML review

- Model evaluation
  - Performance metrics
  - Hold-out evaluation
  - Cross validation
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature selection
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

How do we acquire knowledge?

- Explicitly (Rule-based)
  - When you learn a second language you receive a grammar book with lots of rules and have vocabulary lessons.
- By learning



Types of learning

- Supervised – Making predictions from **labeled** examples

  Classification: Finite labels (predict gender, predict color)

  Regression: Ordered, continuous labels (predict height)

- Unsupervised – Detecting patterns from **unlabeled** examples

- Semi-supervised – Learning from labeled and unlabeled

Memorization and generalization

- Memorization:
  - Are we specifically only learning an algorithm on our data?
- Generalization
  - Refers to the capability of applying learned knowledge to previously unseen data
  - Without generalization there is no learning, just memorizing!

Features and labels

- **Features** (a.k.a covariates) are the variables that describe the data.

- **Label** (target variable/target covariate): The feature we want to predict in supervised learning.
  - So, a label is kind of like a feature

Supervised vs unsupervised examples

- Supervised - Predict housing prices using:
  - Features: number of bathrooms, floor, number of windows, kitchen size
  - Label: House prices

  *** We can switch around the label and the features

- Unsupervised - Find patterns in books using:
  - # of pages, # of printed copies, language, mean words per page
  - Label: there is none
    - Maybe we will ‘cluster’ the books into genres? Maybe into authors? Etc.

More examples of supervised vs unsupervised

Supervised:

1. Regression: Predict housing prices given the prices of 1,000 previously sold houses, with various **features** describing the houses (size, # of windows, …)

2. Classification: Predict if a gift is a book, a toy, or something else (three finite options only)

Unsupervised

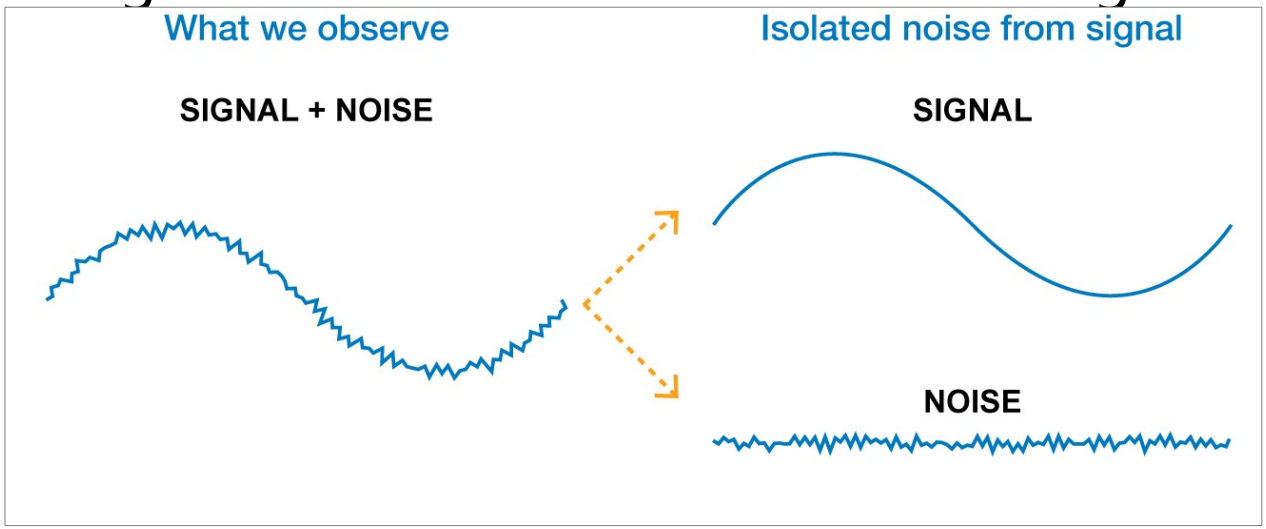
3. Get data on sports players and see if there are specific patterns.
   - Maybe the data has 3 ‘clusters’ that mean something– basketball players, soccer players, and sumo wrestlers?
   - Maybe it clusters into categories nobody has thought about?

What's in the data we learn from

The data usually has two components:

- Signal: information that is relevant to pattern detection or prediction
- Noise: information that is irrelevant to the future

Finding which is which is the challenge



Agenda – Performance metrics

- ~~Quick ML review~~
- Model evaluation
  - Performance metrics
  - Hold-out evaluation
  - Cross validation
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature se
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Why do we evaluate a model's performance?

1. To find the best preforming models
2. Part of parameter tuning

3. To report/publish results
4. To make research/business decisi
   - Can the model be used as is?
   - Do we need to try to improve?

Performance metrics

- Supervised
  - **Classification:**
    - Accuracy
    - Precision & Recall
    - ROC curves & AUC
  - **Regression:**
    - Mean square error (MSE) + Root MSE
    - Percent error
    - Mean absolute percent error
  - **Ranking (ordinal/discrete regression):**
    - Precision at N
- Unsupervised

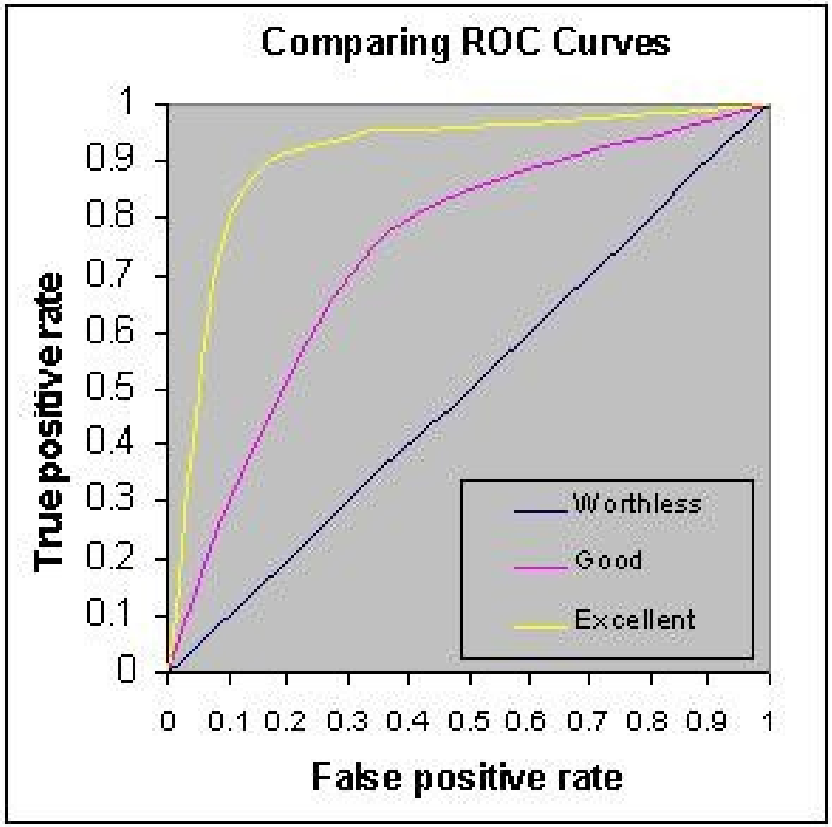
Classification: Precision and recall

- Precision (a.k.a PPV): What percent of our predictions are correct?
- Recall (a.k.a sensitivity): What percent of the accurate predictions did we capture?
- F1 score: A single number that combines the two values above. Good for ranking/sorting, and imbalanced classes

$$\mathrm{F1} = 2 * \frac{\mathrm{precision} * \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

- Accuracy: What percent of all our predictions (positive and negative) are correct?

Classificaiton: ROC curve



Area under curve (the ROC curve)

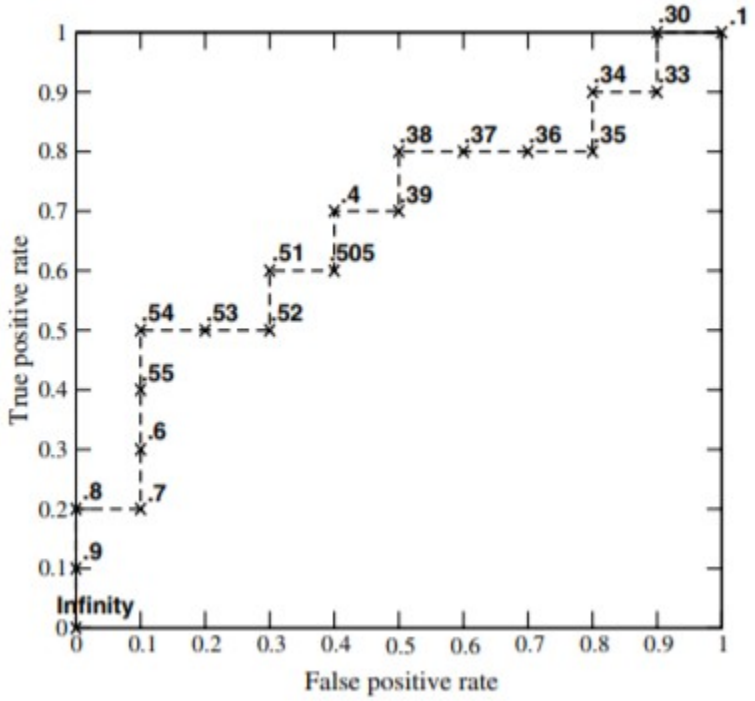
A Receiver Characteristic Curve (ROC) plots the True positive rate (TPR) vs. the False positive rate (FPR). The maximum area under the curve (AUC) is 1. Completely random predictions have an AUC of 0.5. The advantage of this metric is that it is continuous.

Constructing a ROC curve

Sort predictions

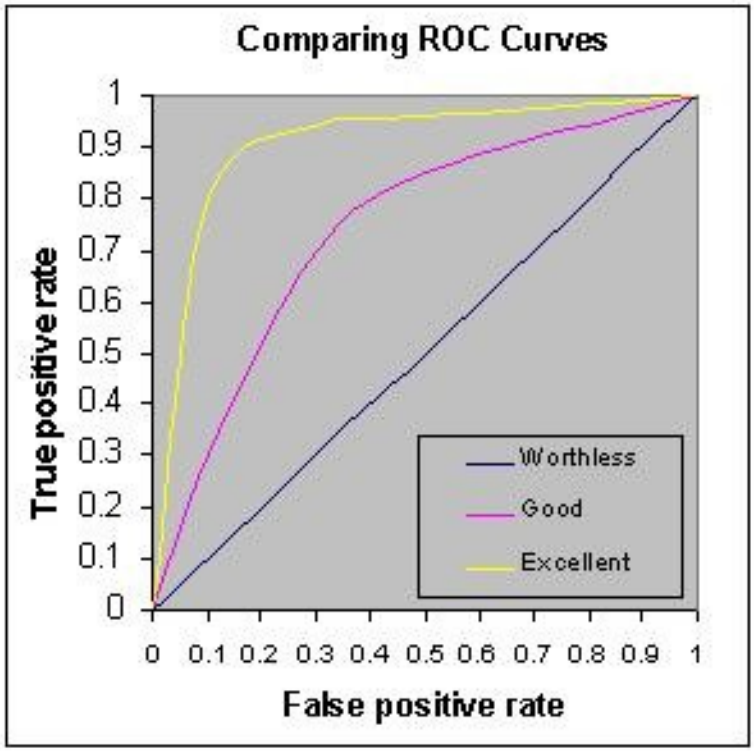
Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Plot



AUC interpretation

- The AUC is equal to the probability that a random positive example will be ranked above a random negative example.



Classificaiton: Confusion matrix

Confusion Matrix

Total		Predicted		
		Iris-setosa	*Iris-versicolor*	*Iris-virginica*
Actual	*Iris-setosa*	12	1	1
	Iris-versicolor	0	16	0
	Iris-virginica	0	1	16

- Used in classification

- Show Actual vs predicted results

- Enables visualizing performance and calculating performance metrics

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

The confusion matrix – list of statistical metrics that can be calculated

- Many performance terms: Precision, recall, sensitivity, specific, true negative rate, true positive rate, PPV, NPV, type 1 error, type 2 error

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	**Specificity** = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$	

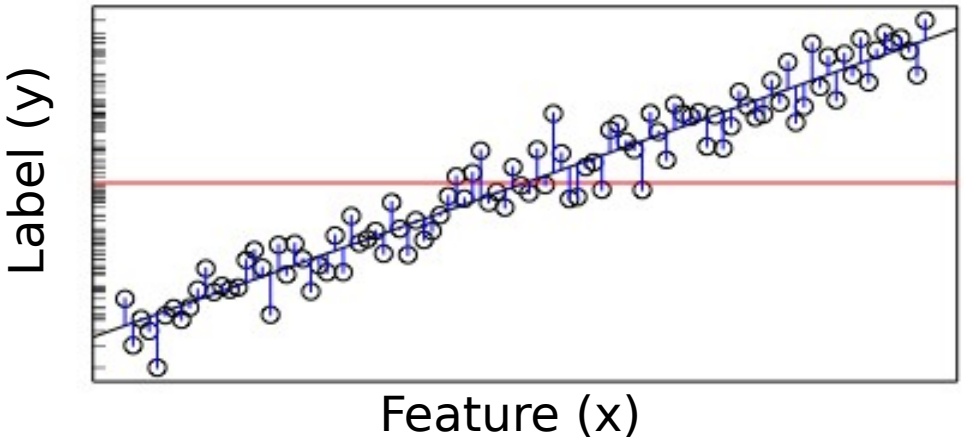
$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

Ranking – precision at N

- Precision at N
  - How many accurate examples did we capture in our top N ranked examples. This is often used in information (document) retrieval

Regression – performance metrics

Blue lines are the error



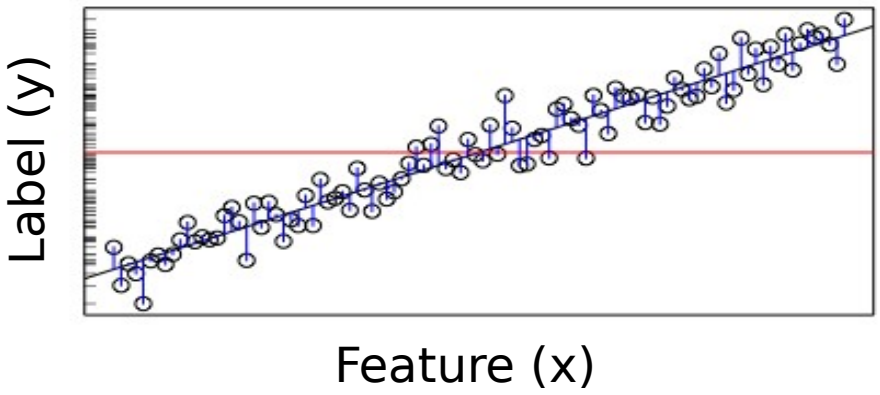
For every example, we have:

- Error $=(y_{predicted} - y_{true})$
- Percent error $=(y_{predicted} - y_{true})/y_{true}$

Now lets aggregate the errors and get a single value (next slide)

Regression – performance metrics

Blue lines are the error



- Mean square error (MSE) and Root MSE (RMSE):
  - MSE = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$
  - Gives high weight to outliers, most common metric

- MAPE: Mean absolute percent error
  - Percentages are easy to understand. Simply the mean of the |Percent error|

$$\text{MAPE} = \frac{\sum \frac{|\text{A-F}|}{\text{A}} \times 100}{\text{N}}$$

Agenda – Hold-out evaluation

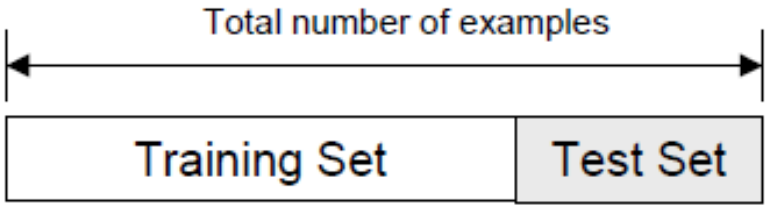
- ~~Quick ML review~~
- Model evaluation
  - ~~Performance metrics~~
  - Hold-out evaluation
  - ~~Cross validation~~
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature se
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

The basic rule in model validation

Don’t test your model on data it’s been trained with!

Holdout test set: The naïve approach

- Randomly split the entire dataset into:
  - Training set: A dataset used for training the model
  - Test set (a.k.a validation set): Data only used for testing the model



Train model here

Estimate model performance here

Holdout test set – considerations

- Can the data be split randomly without bias?
  - Test set and training set should be interchangeable, except for data size. Otherwise test results are not a true reflection of performance.
  - In other words, test and train data should be from the same distribution

- Do we have enough data?
  - Example: Predict if a car is American or Japanese based on 50 examples?

Will the test set be big enough to accurately report our model performance?

What is biased/unbiased data?

- Unbiased data sampling:
  - Data that represents the diversity in our target distribution of data on which we want to make predictions.
  - I.e., a random, representative sample of the entire distribution.

- Biased data:
  - Data that does not accurately reflect the target distribution.
  - Examples:
    - **Elections**: we only asked people with glasses who they will vote for, instead of asking a random sample of the population.
    - **Simulated data**: We cannot get real data, so we create simulated/synthetic data
  - May be addressed using transfer learning/domain adaptation

The three-way split

- **Training set**

A set of examples used for learning

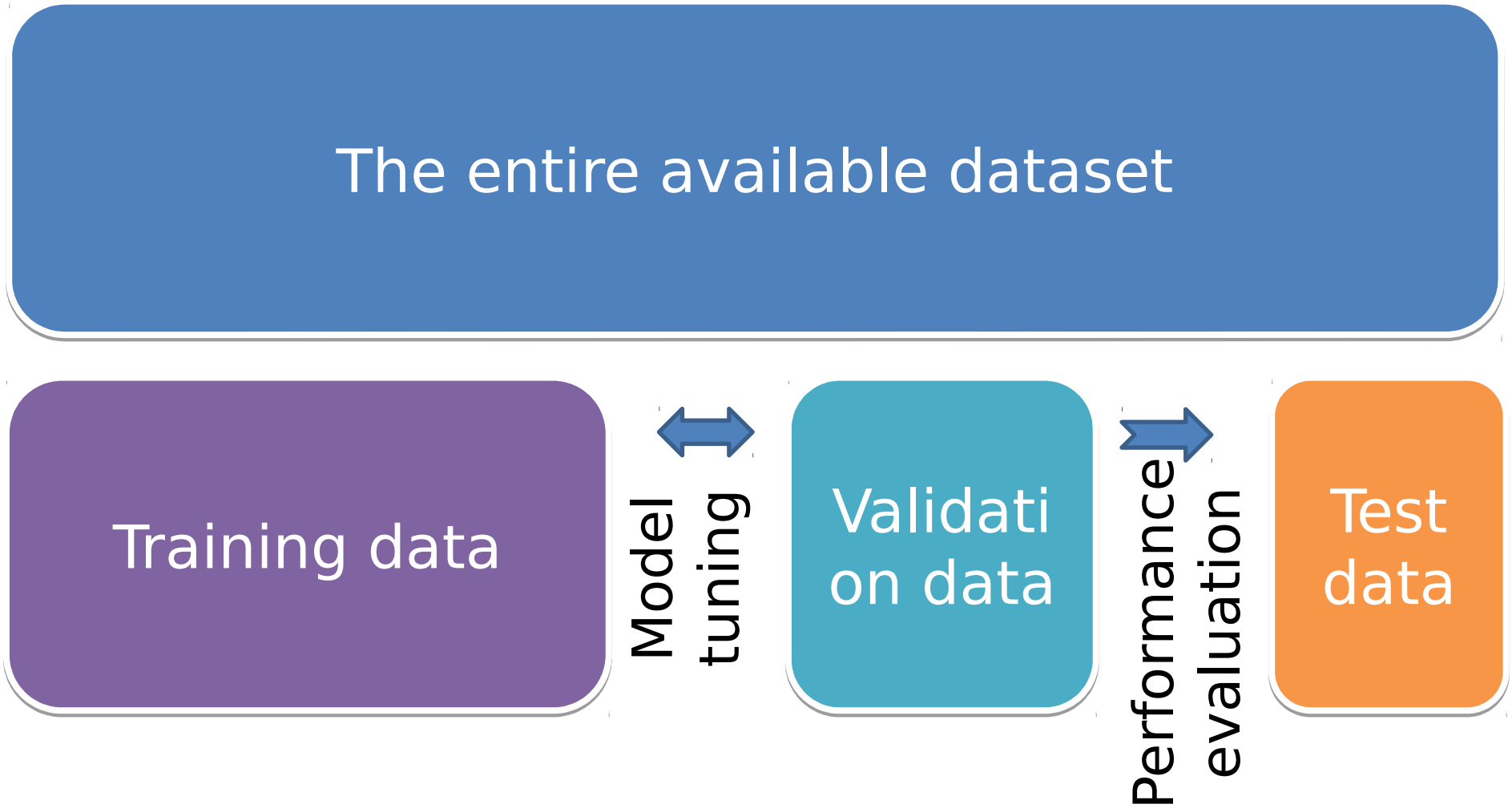
- **Validation (a.k.a development) set**

A set of examples used to tune the parameters of a classifier

- **Test set**

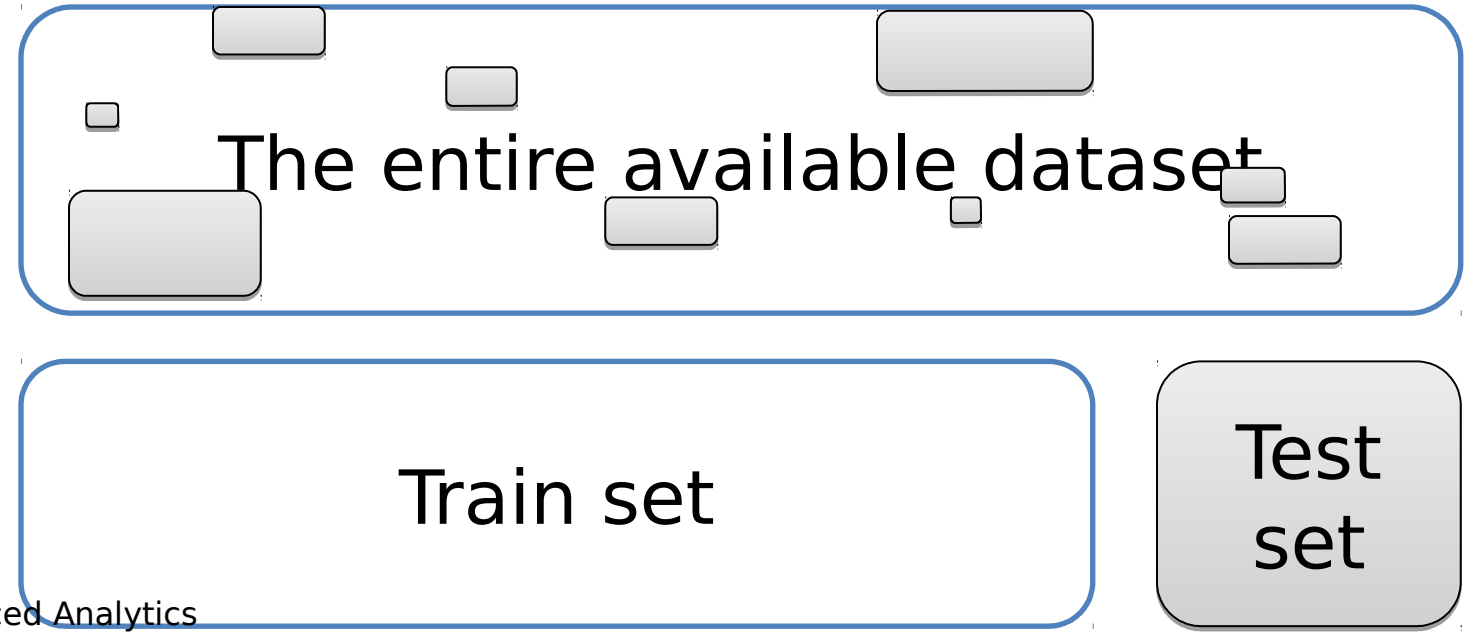
A set of examples used only to assess the performance of fully-trained classifier. After assessing the model with the test set, YOU *MUST NOT* further tune your model (that's the theory anyway… - in order to prevent 'learning the test set' and 'overfitting')

The three-way split



How to perform the split?

- How many examples in each data set?
  - Training: Typically 60-80% of data
  - Test set: Typically 20-30% of your trained set
  - Validation set: Often 20% of data
- Examples
  - 3 way: Training: 60%, CV: 20%, Test: 20%
  - 2 ways: Training 70%, Test: 30%



How to perform the split?

- Time based – If time is important

Can you use Jul 2011 stock price to try and predict Apr 2011 stock price?

How to perform the split?

- Some datasets are effected by seasonality

Is it a good test set?

Holdout summary

- Good news:
  - Intuitive
  - Usually easy to perform
  - Considered the ideal method for evaluation

- Drawbacks:
  - In small datasets you don’t have the luxury of setting aside a portion of your data
  - The performance will be misleading if we had unfortunate split
  - Often requires planning ahead of time to prevent training on test set or getting insights from it)
  - External evaluation is ideal

Agenda – Cross validation

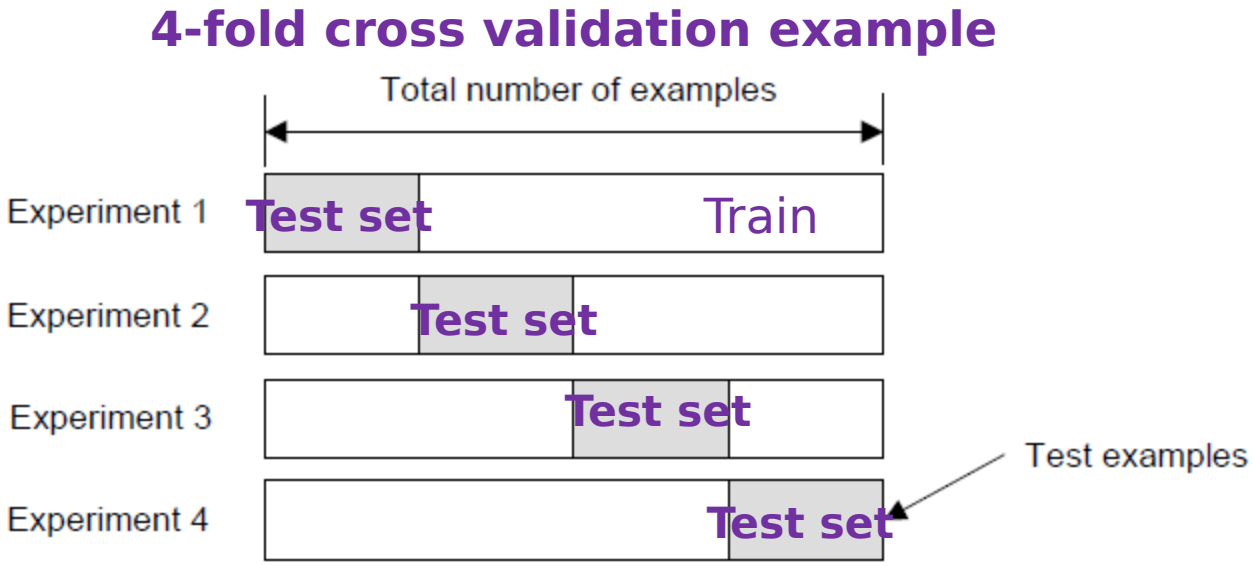
- ~~Quick ML review~~
- Model evaluation
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - Cross validation
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature selection
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Cross validation

- Used when:
  - a hold-out plan is difficult to design or was not designed at the beginning
  - the dataset is small, and you want to use all the data for good model training

Cross - Validation

- Create a K-fold partition of the dataset
  - For each of the K experiments, use K-1 folds for training and the remaining one for testing



Cross - Validation

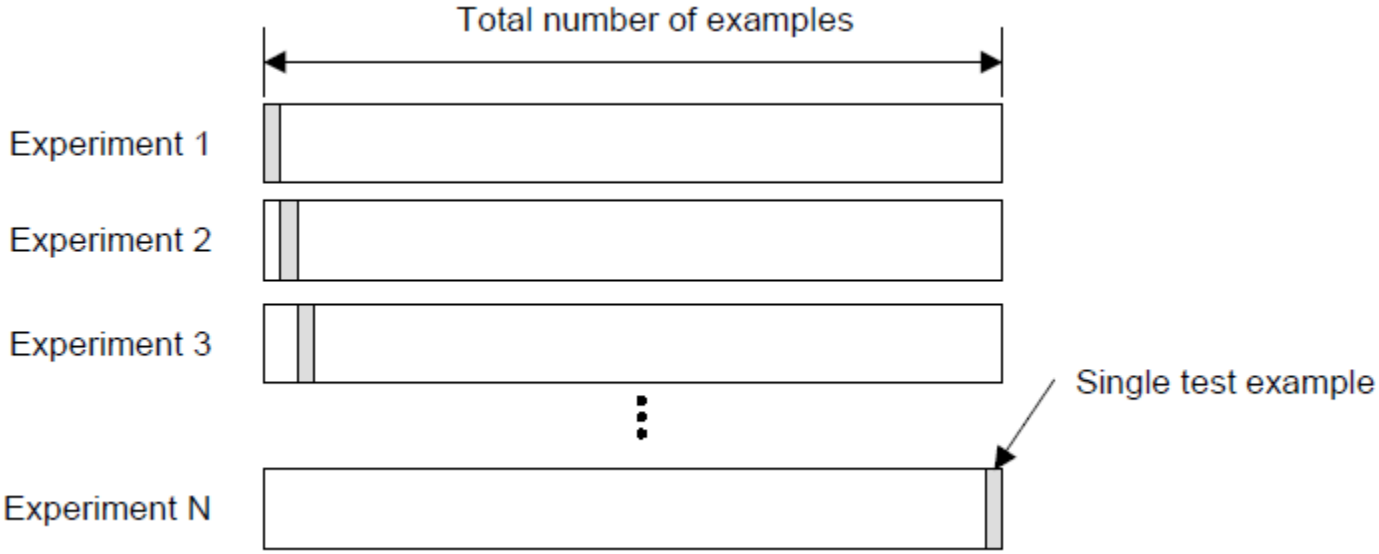
- How do you summarize the performance?
  - **Average:** Usually average of performance between experiments

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

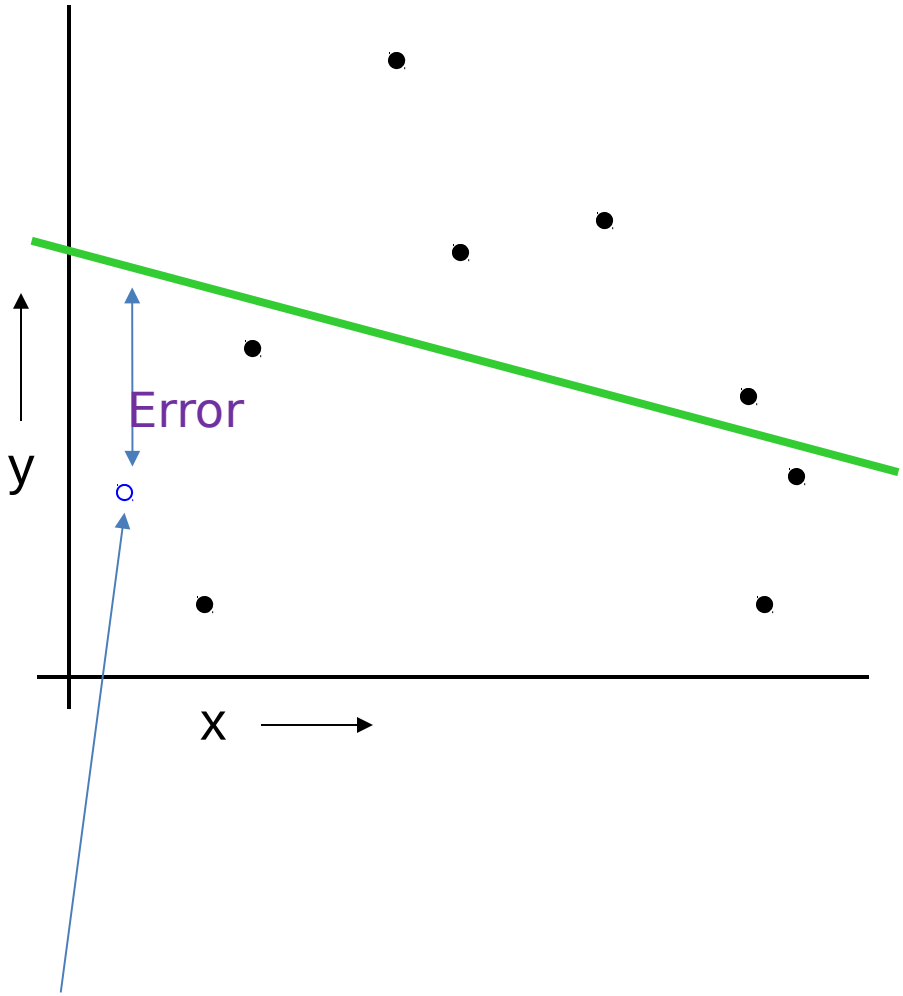
- How many folds are needed?
  - Common choice: 5-fold or 10-fold cross validation (probably since these numbers sound nice)
  - Large datasets even a 3-Fold cross validation will be quite accurate
  - Smaller datasets we will prefer bigger K
  - Leave-One-Out approach (K=n). In this case the number of folds K is equal to the number of examples n. Used for smaller datasets

Leave-One-Out Cross Validation (LOOCV)

- Leave-One-Out is the degenerate case of K-Fold cross validation, where K is chosen as the total number of examples



LOOCV in action



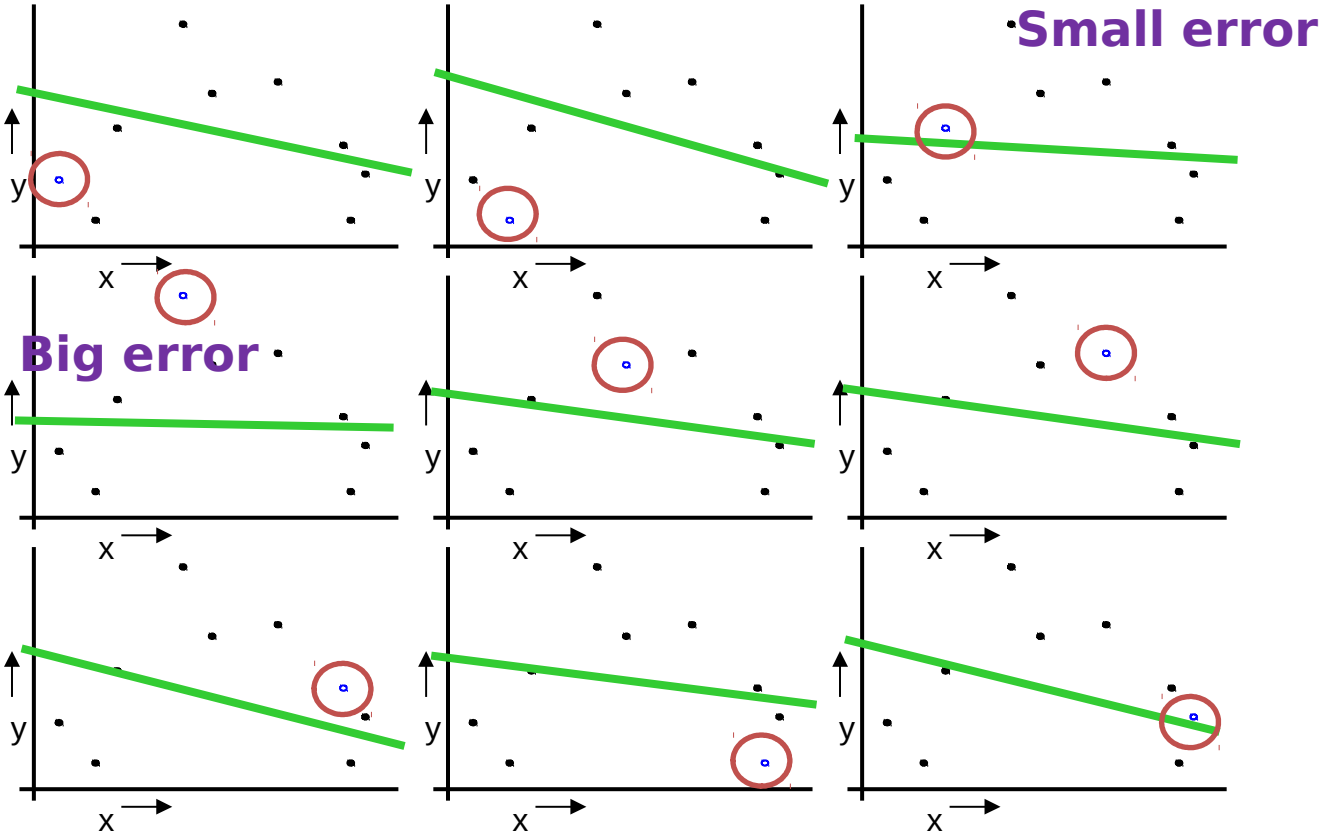
For k=1 to R, do:

1. *Let (x_k,y_k) be the k'th record*
2. *Remove (x_k,y_k) from the dataset*
3. *Train on the remaining R-1 datapoints*
4. *Measure your error (x_k,y_k)*

When you've done all points, report the mean error.

I was left out □.

LOOCV in action



$$MSE_{LOOCV} = 2.12$$

Cross-Validation summary

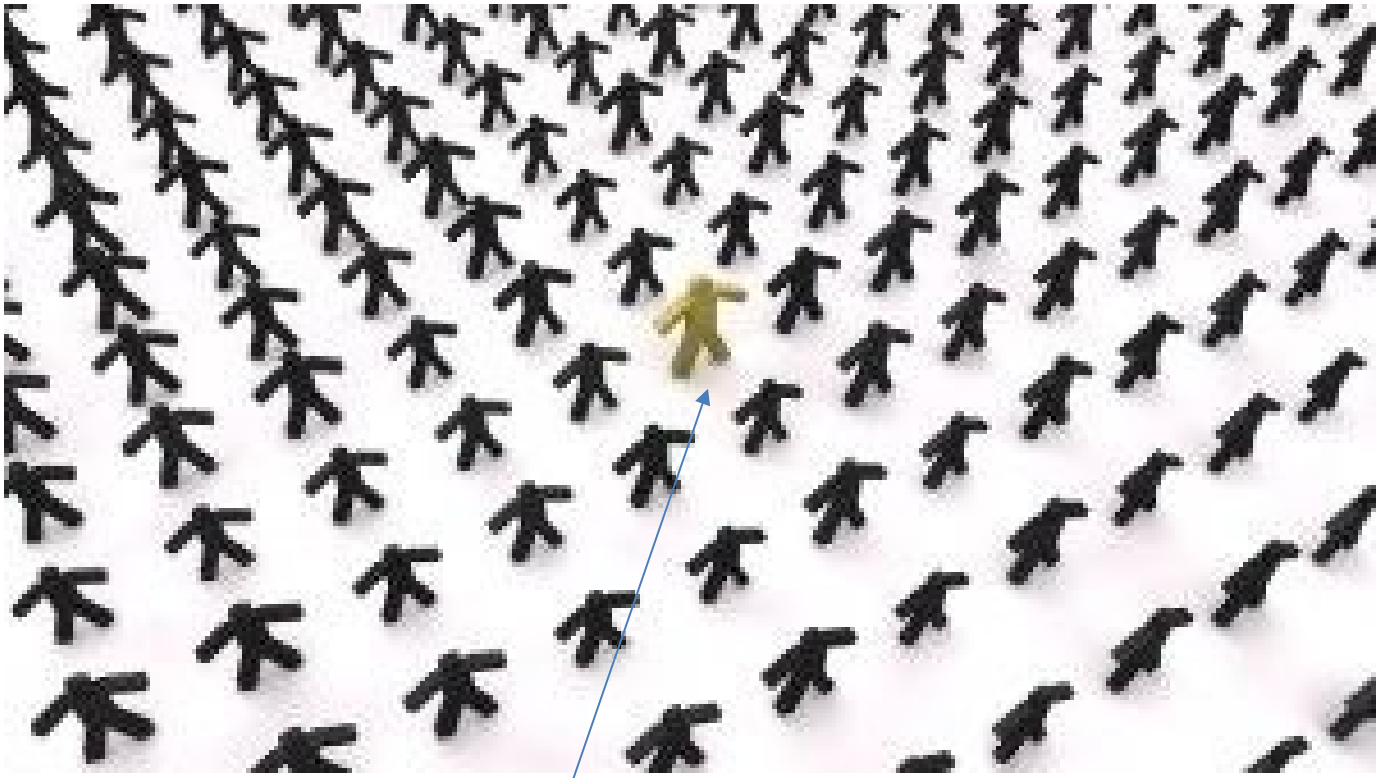
- Good news:
  - Good for small datasets - training on all the data
  - No need to plan hold-out dataset(s), which can be challenging sometimes

- Drawbacks:
  - Partitioning is not always easy to do
  - Can be less believable than hold-out data (especially 3rd party holdout).

Agenda – Imbalanced classes

- ~~Quick ML review~~
- Model evaluation
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross-validation~~
  - Training with imbalanced classes
  - Overfitting/underfitting
- Dimensionality and feature se
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Dealing with imbalanced classes



Class 1: Short people
Class 2: Tall people

Can we build a classifier that can find the needles in the haystack?

Imbalanced Class Sizes

- Classes frequently have very unequal frequency
- Creates training and evaluation challenges
- Examples:
  - Security: >99.99% of Americans are not terrorists
  - Cancer diagnosis: 99% healthy, 1% disease
    - If we ‘straight-up’ predict that nobody has cancer, then 99% of our predictions are correct, but we will never detect cancer.
    - If we misclassify 1% of healthy patients 50% of people we tell have cancer are actually cancer-free!

Good metrics for imbalanced classes

- Good metrics
  - AUC
  - F1 score
- Not ideal (bad) metrics:
  - Accuracy
  - Only recall or only precision

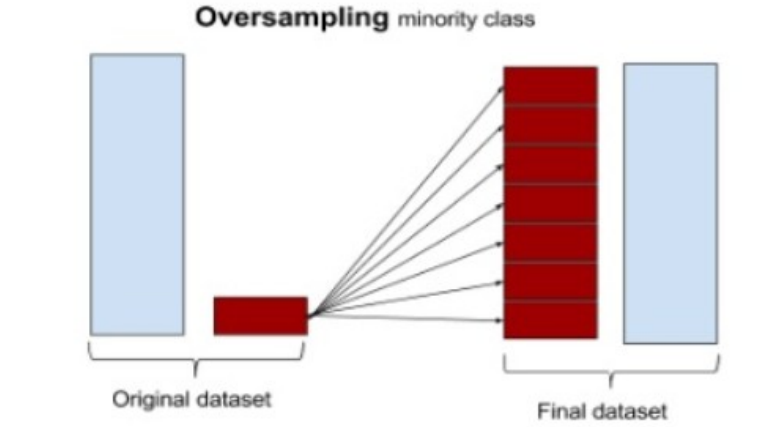
Learning with imbalanced datasets is often done by balancing the classes

- Create a balanced data set by:
  - Down-sampling the majority class
  - Up-sampling the minority class
  - Assign larger weights to the minority class samples

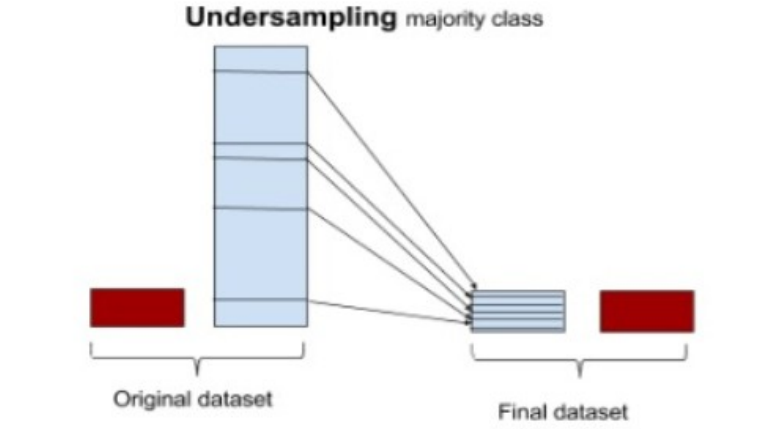
Important: Estimate the final results using an imbalanced held-out (test) set

Up-sampling and Down-sampling

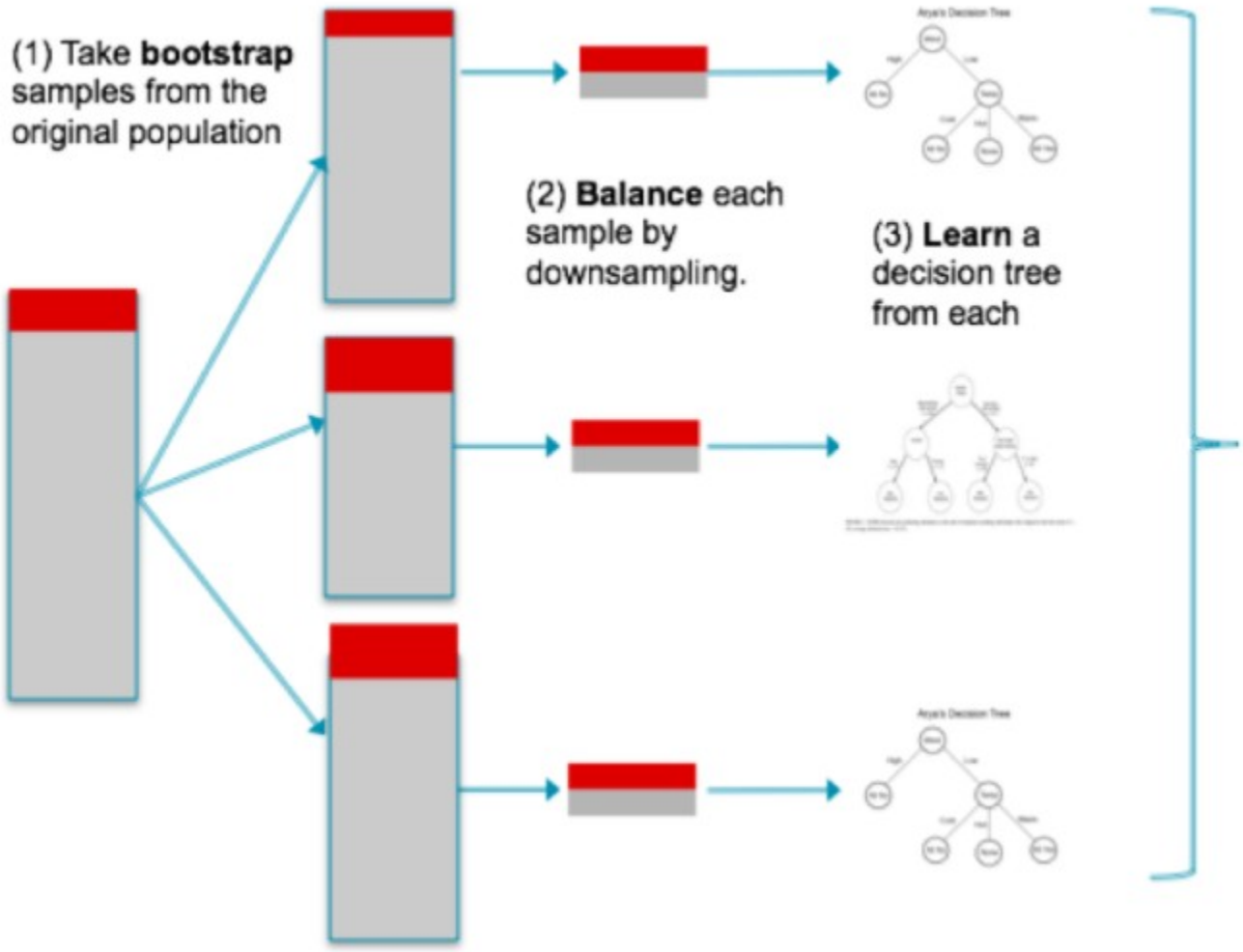
- Up-sample:
  - Repeat minority points
  - Synthetic Minority Oversampling Technique (SMOTE)



- Down-sample:
  - Sample (randomly choose) a **random subset of points** in the majority class
  - Bootstrapping (next slide)



Bootstrapping

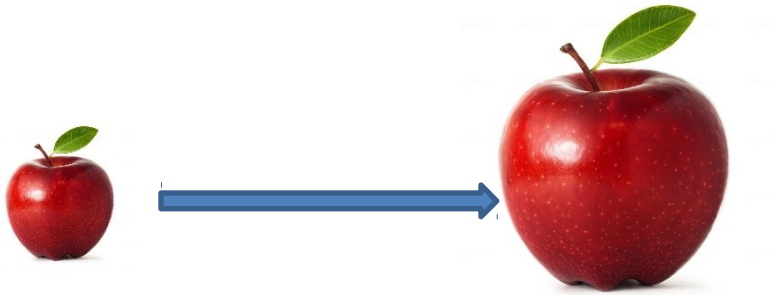


Bootstrapping

- Sample **m** sets from the majority so that each sets size is equal to the minority set.
- The **downsampling** is done with replacement (repeats allowed).
- Train a model of minority vs. bootstrapped sample for each bootstrap iteration
- This gives us **m** different models this is the basis of ensemble models like Random forest

Reweighting

- Idea: Assign larger weights to samples from the smaller class



- A commonly used weighting scheme is linearly by class size:
  - Where is the size of the class *c* and is the total sample size

Imbalanced data can be harnessed

- The Viola/Jones Face Detector

Faces

Non-Faces

Training data

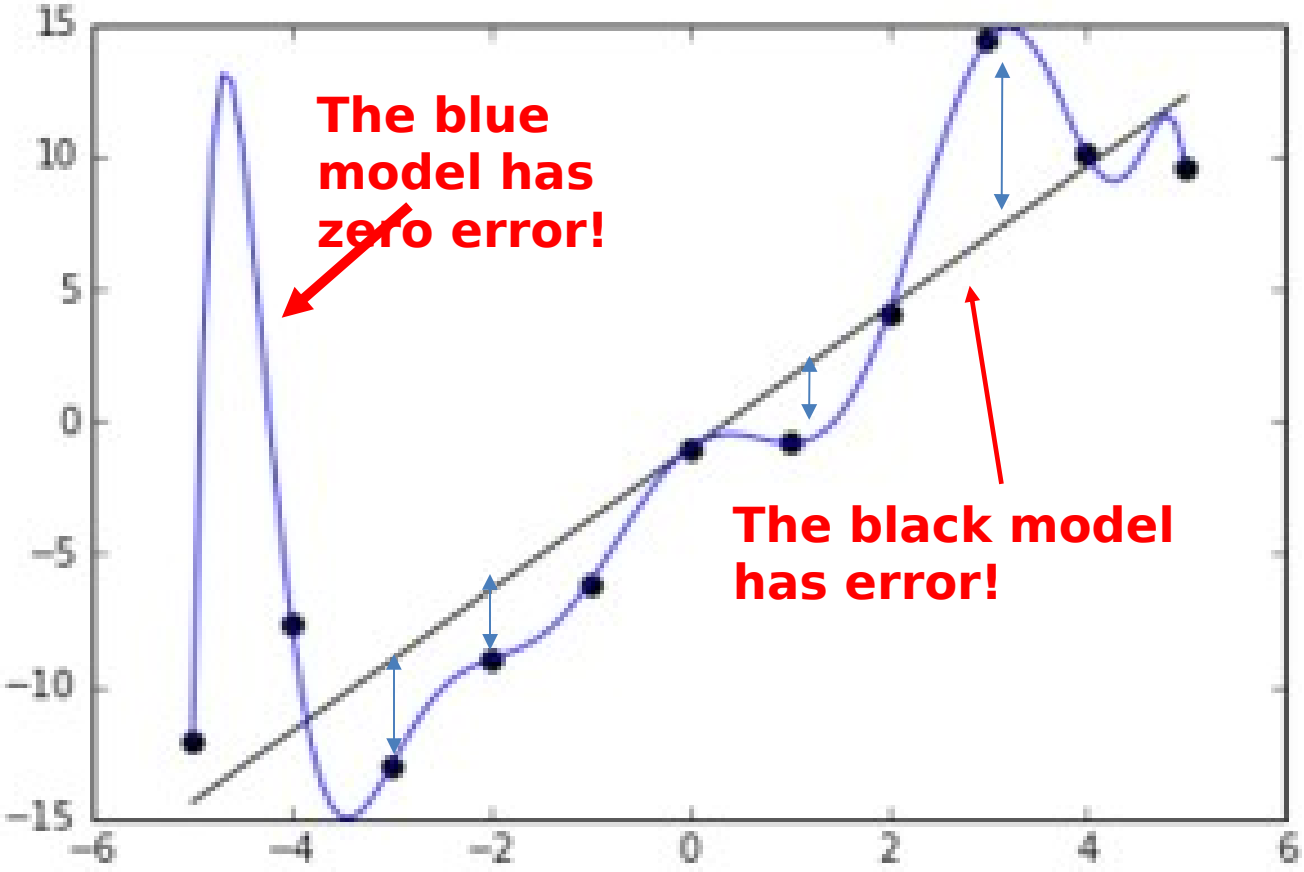
5000 faces 10^8 non faces

- P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features”, CVPR, 2001
- P. Viola and M. Jones, “Robust real-time face detection”, IJCV 57(2), 2004

Agenda – Overfitting/underfitting

- ~~Quick ML review~~
- Model evaluation
  - ~~Performance metrics~~
  - ~~Hold-out evaluation~~
  - ~~Cross validation~~
  - ~~Training with imbalanced classes~~
  - Overfitting/underfitting
- Dimensionality and feature selection
  - Curse of dimensionality
  - Filter feature selection
  - Wrapper feature selection
  - Principal Component Analysis (dimensionality reduction)

Overfitting and underfitting



So which model is better : blue or black?

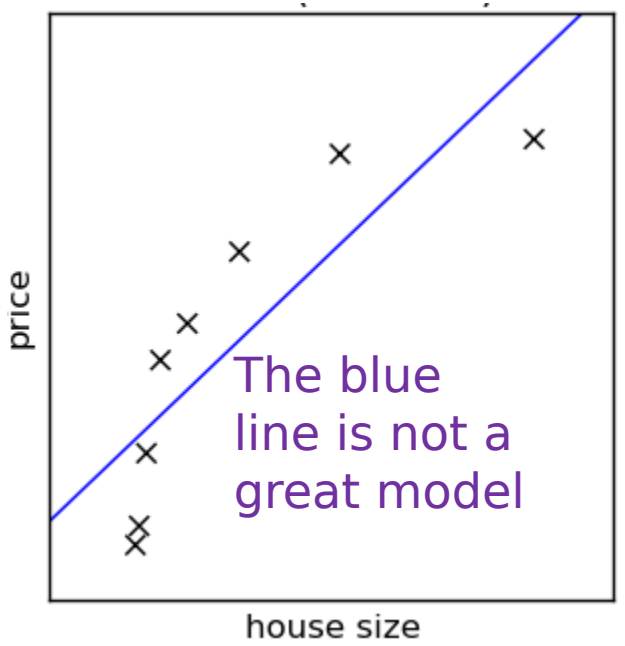
Underfitting (high bias)

- In underfitting, the training error and test error are high

- Caused by too ‘simple’ models
  - Too few features
  - Use of features is not ‘complex’

- Examples
  - Can you accurately detect spam emails using only the word ‘free’?

  - Can you accurately predict housing prices using only the year it was built?

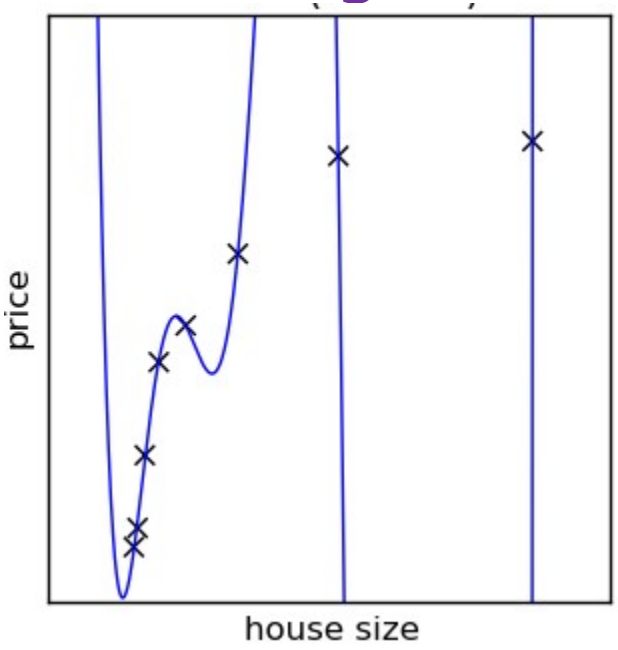


Probably not. More features (words) are needed)

Probably not - More features are needed: size, location, #windows, etc.

Overfitting (high variance) is when the model learns the noise and signal

- If the model overfits
  - It cannot generalize well to new data
  - It memorizes the training data
  - It has a low training error, and high test error

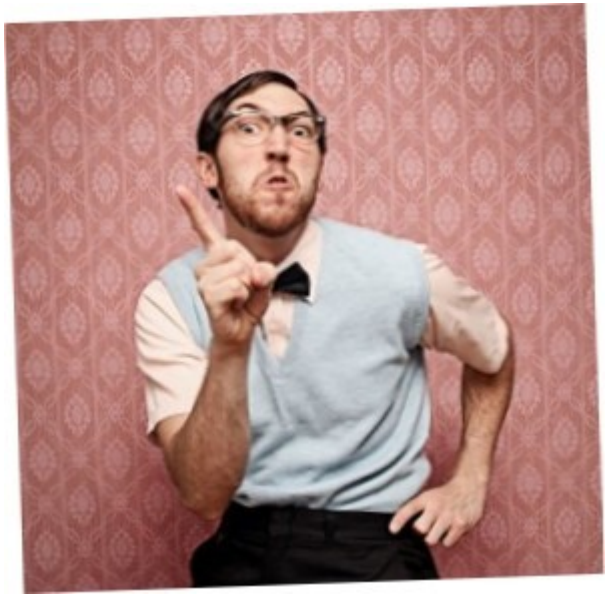


The model has no error, but it does not seem to represent the data well

Overfitting is caused by:

- Too much model complexity
  - Typically too many features

- Little or not diverse data
  - Big data: The more data we have, the more complex a model we can use
  - Diversity: Redundant data does not add information and thus does not improve the model



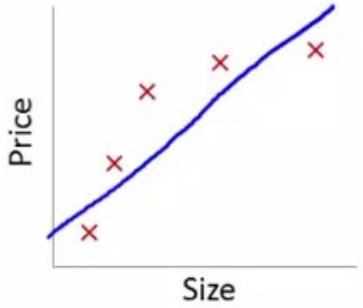
Overfitting due to the choice of too many features

- We can use many features to try and predict the price of a house

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

Overfitting due to poor model choice

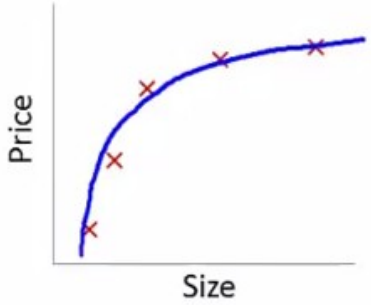
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$

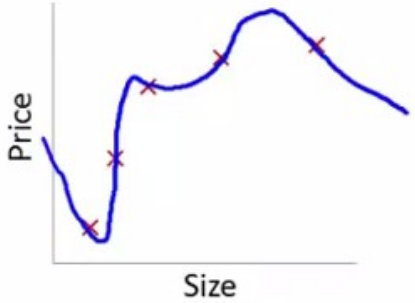
"Underfit" "High bias"

Not many features (only x1)



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

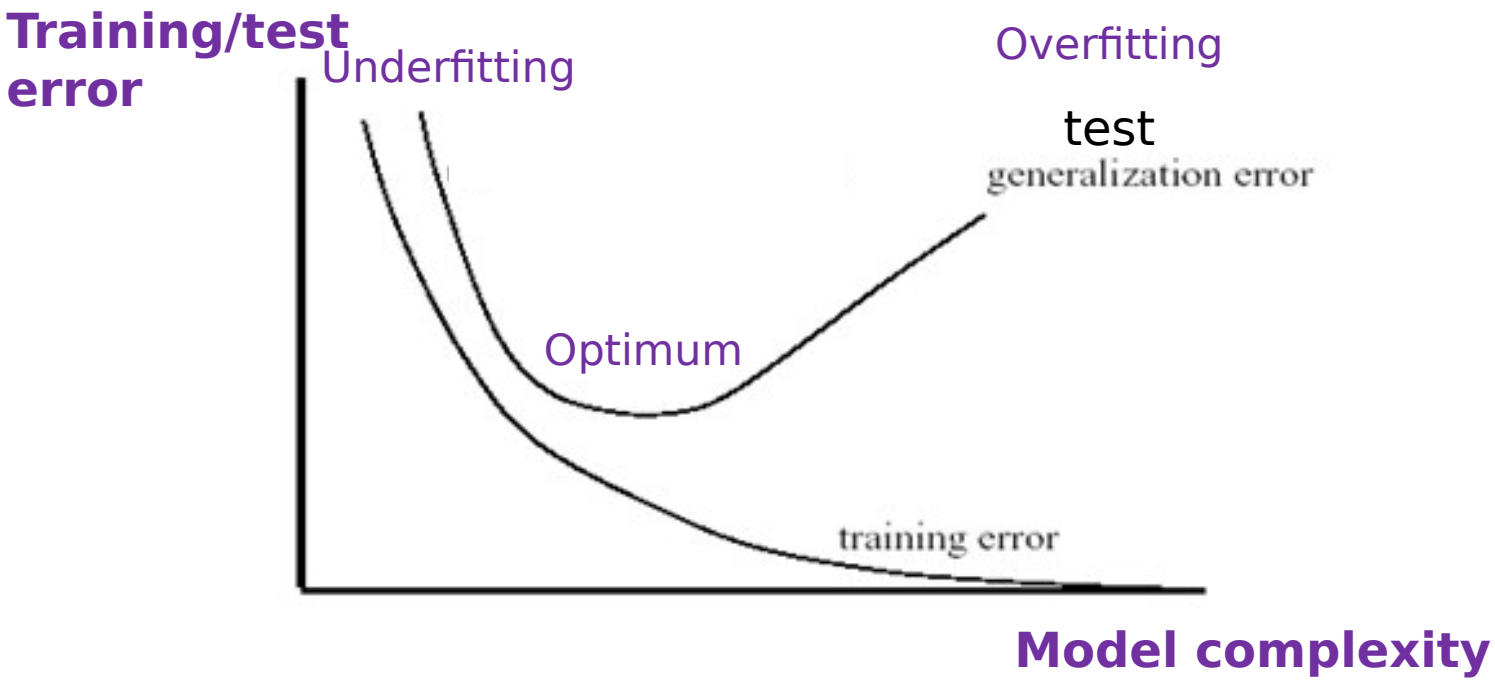
"Overfit" "High variance"

Too many features (x1 to x4)

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Andrew Ng

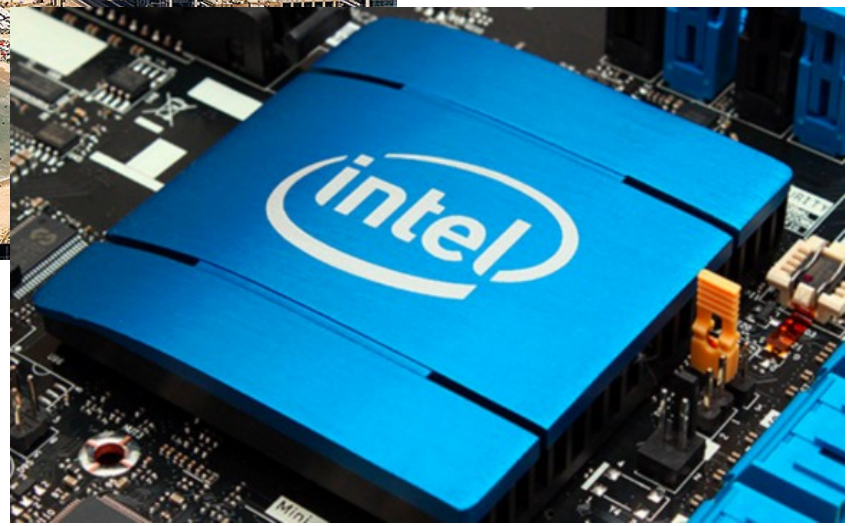
We can plot learning curves to spot overfitting



Addressing Overfitting

- More data and more diverse data if possible
- Reduce # of features/dimensionality
- Regularization
  - Keep all the features but penalize some features/ values of parameters
  - This is particularly useful when we have a lot of features, each contributing a bit to the prediction

Thank you and enjoy life



Zeev Waks, zeev.waks@intel.com
Data Science Lead
Intel – Advanced Analytics