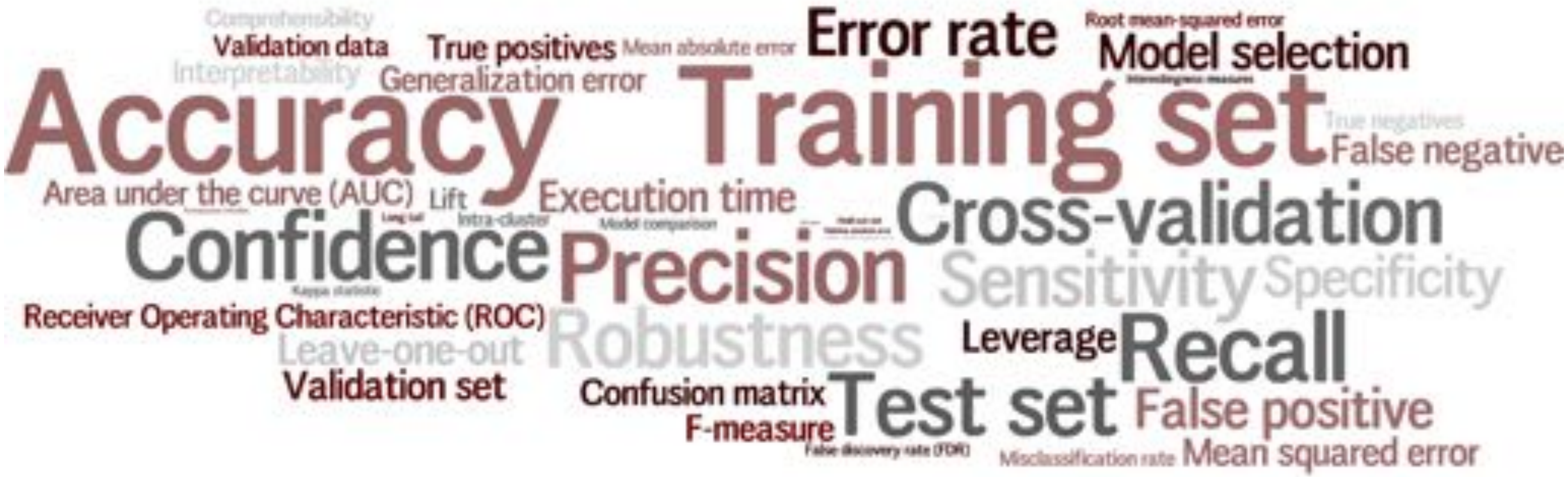
Methods for detecting cyber attacks Introduction To ML Part 2

Dr. Ran Dubin

Lecture slides were taken from Prof. Lior Rokach and Asaf Shabtai

Performance evaluation

Evaluating what's been learned



Introduction

- How to measure algorithm success?
- Compare between algorithms
- Training/validation/test sets
- Linear Regression
- Resampling methods: k-fold cross-validation
- Feature selection

Algorithm preference

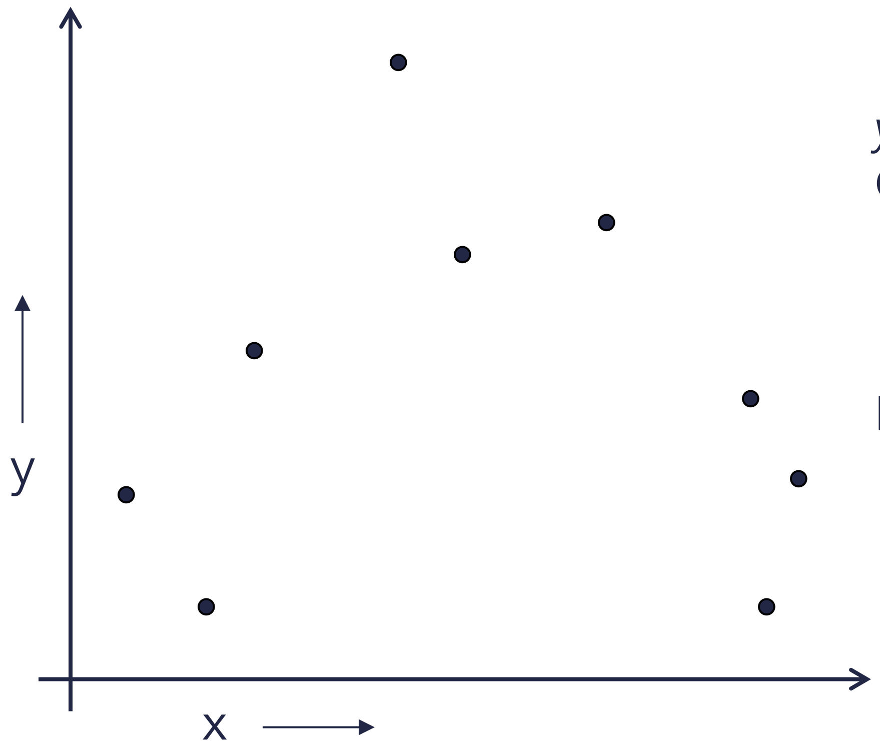
- Criteria (application-dependent):
  - Accuracy
  - Misclassification error, or risk (loss functions)
  - Training time/space complexity
  - Testing time/space complexity
  - Interpretability (model complexity, for instance number of hidden units)
  - Easy tuning
  - Easy programmability
  - Easy embedding

"Fast is fine, but accuracy is everything."
Xenophon
(Greek historian, **430 - 354 BC**)

Other criteria

- Robustness
- Missing values
- Noise
- Incremental
- Data type (e.g. numeric)
- Multi-class
- Imbalance
- Cost sensitive
- Availability

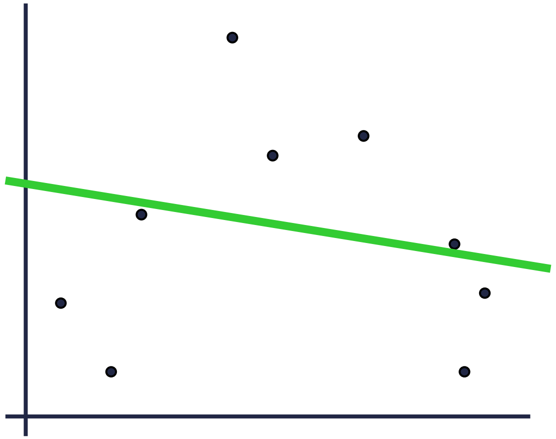
A Regression Problem



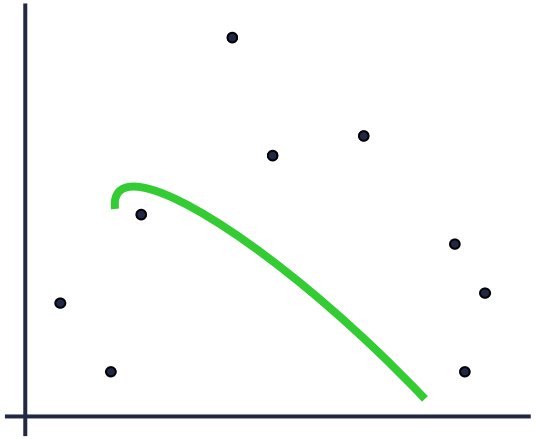
y = *f*(*x*) + *noise*
Can we learn f from this data?

Let's consider three methods…

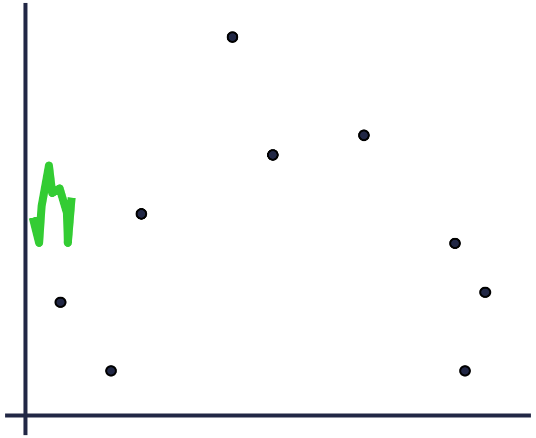
Which is best?



Linear regression



Quadratic regression

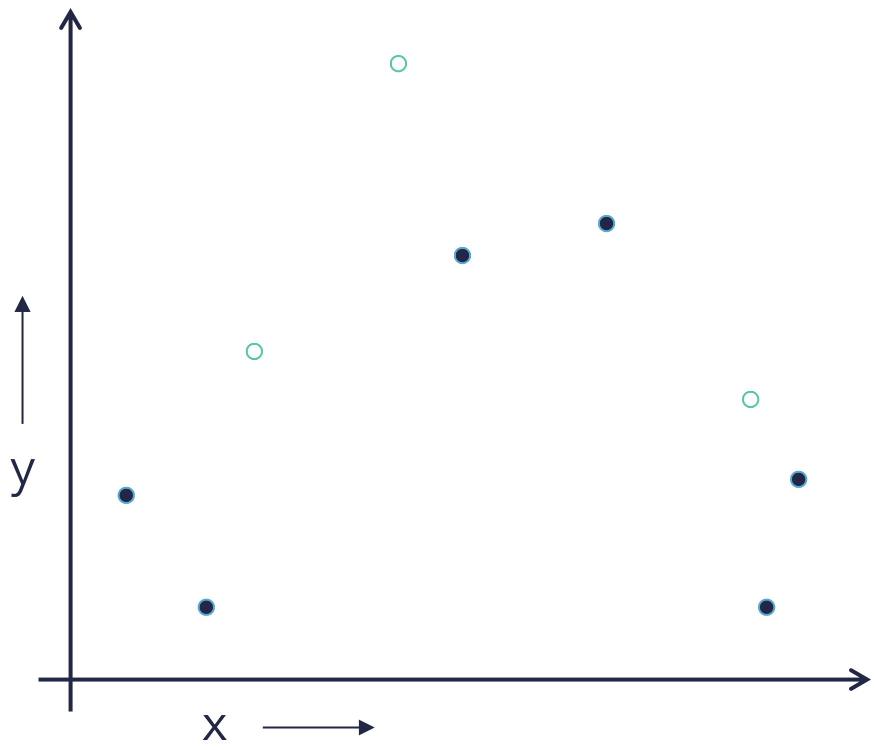


Piecewise linear
nonparametric regression

Why not choose the method with the best fit to the data?

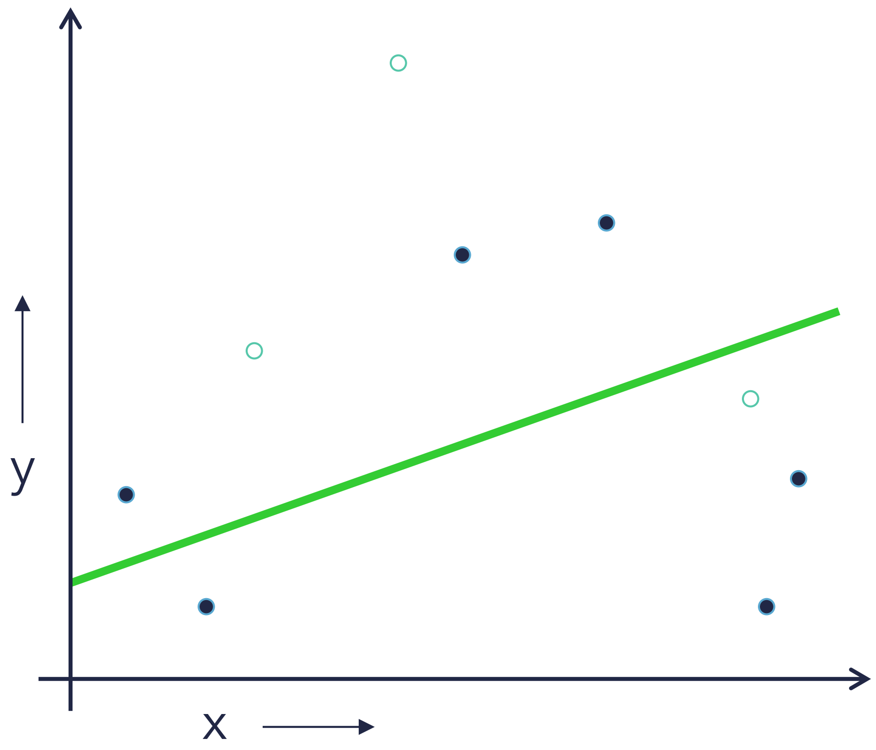
"How well are you going to predict future data drawn from the same distribution?"

The test set method



1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set

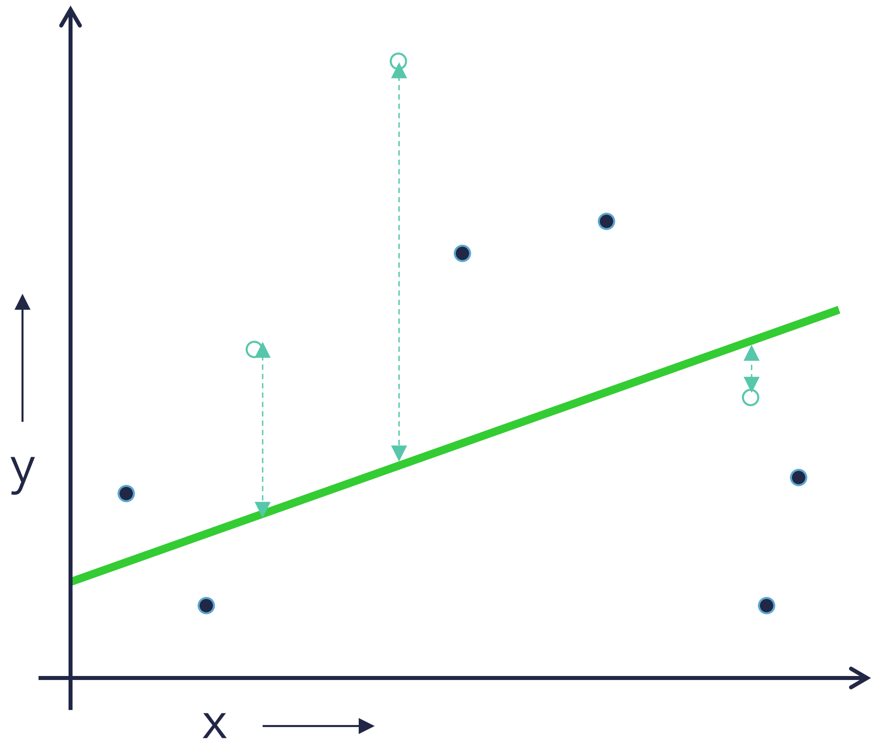
The test set method



Linear regression example

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set

The test set method



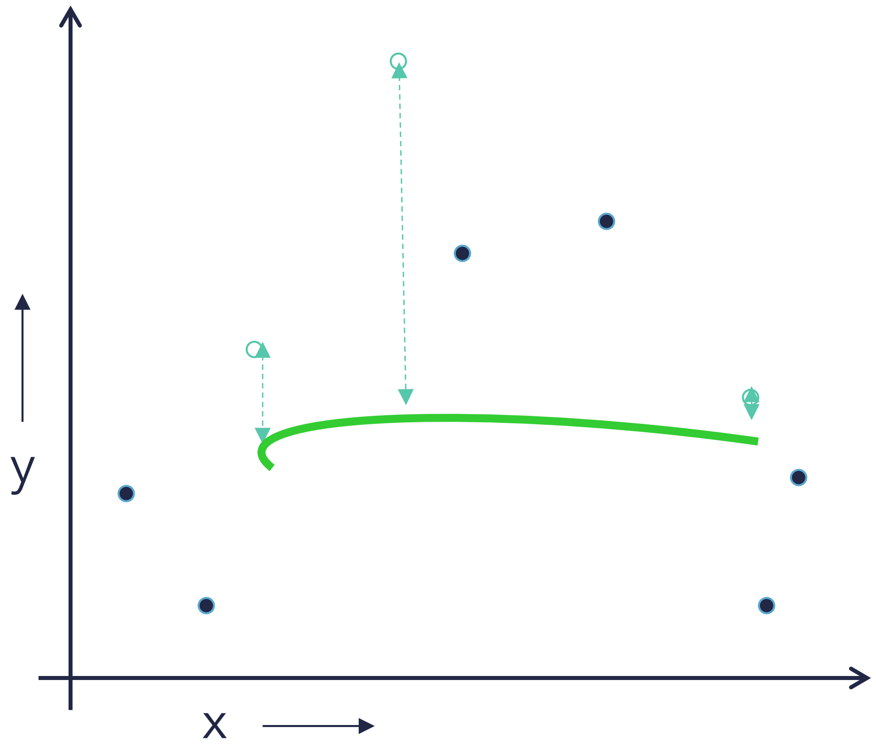
Linear regression example
Mean Squared Error = 2.4

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

Note: MSE ==0 perfect regression

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 :$$

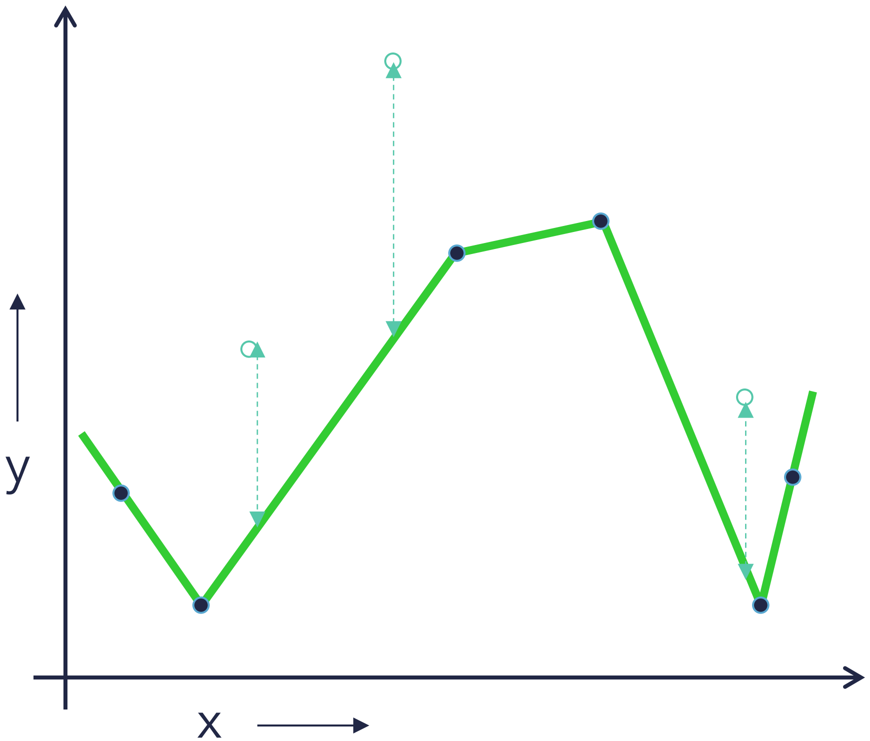
The test set method



Quadratic regression example
Mean Squared Error = 0.9

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

The test set method



Join-the-dots example
Mean Squared Error = 2.2

1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

The test set method

- Good news:
  - Simple
  - Can then simply choose the method with the best test-set score

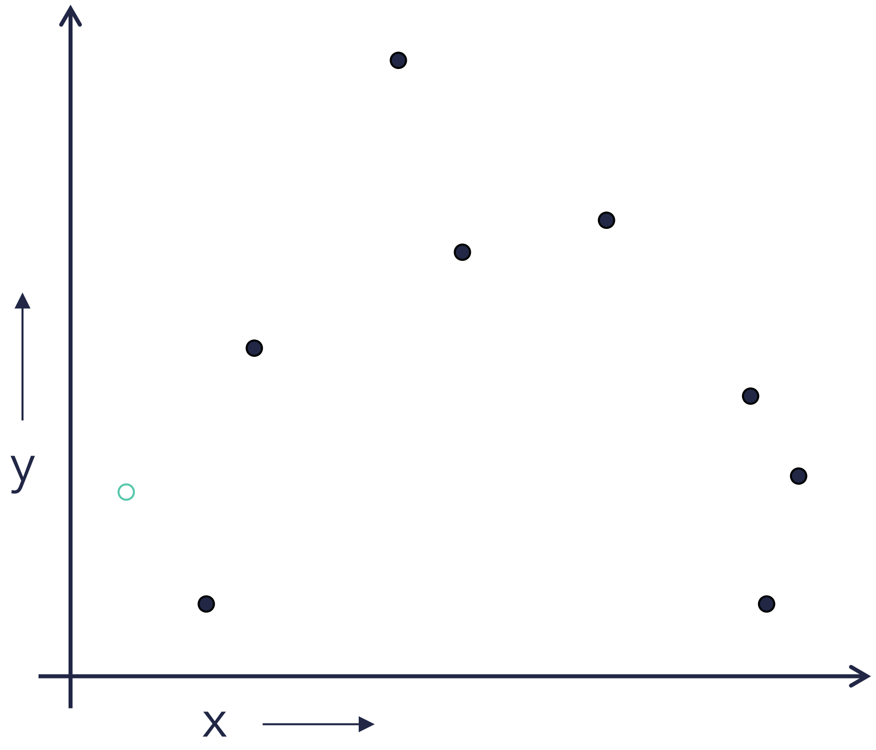
- Bad news:
  - What's the downside?

The test set method

- Good news:
  - Simple
  - Can then simply choose the method with the best test-set score

- Bad news:
  - Wastes data: we get an estimate of the best method to apply to 30% less data
  - If we don’t have much data, our test-set might just be lucky or unlucky
  - The “test-set estimator of performance has high variance”

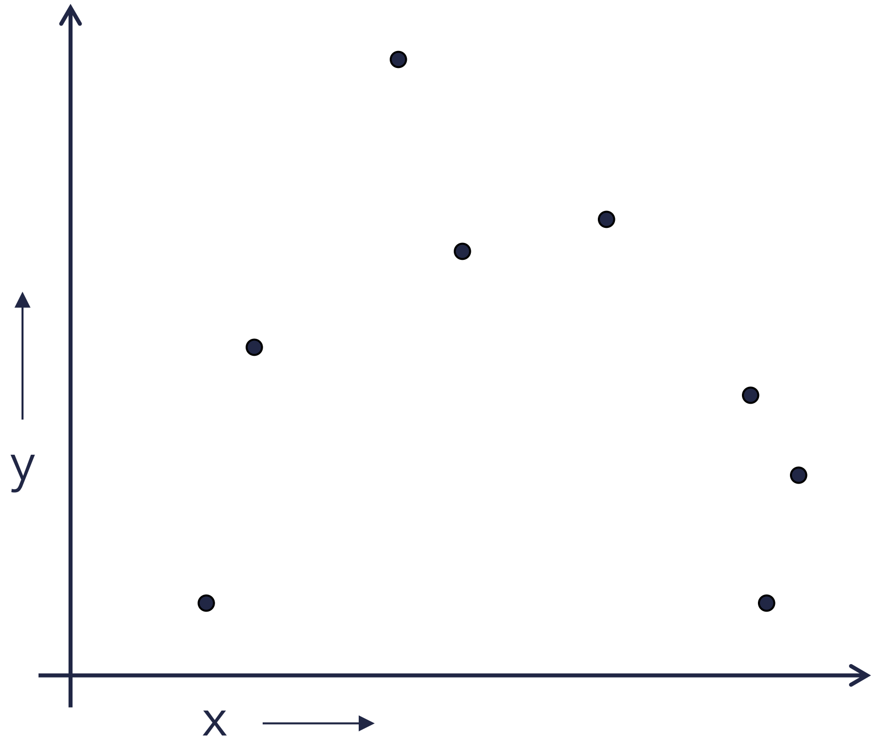
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record

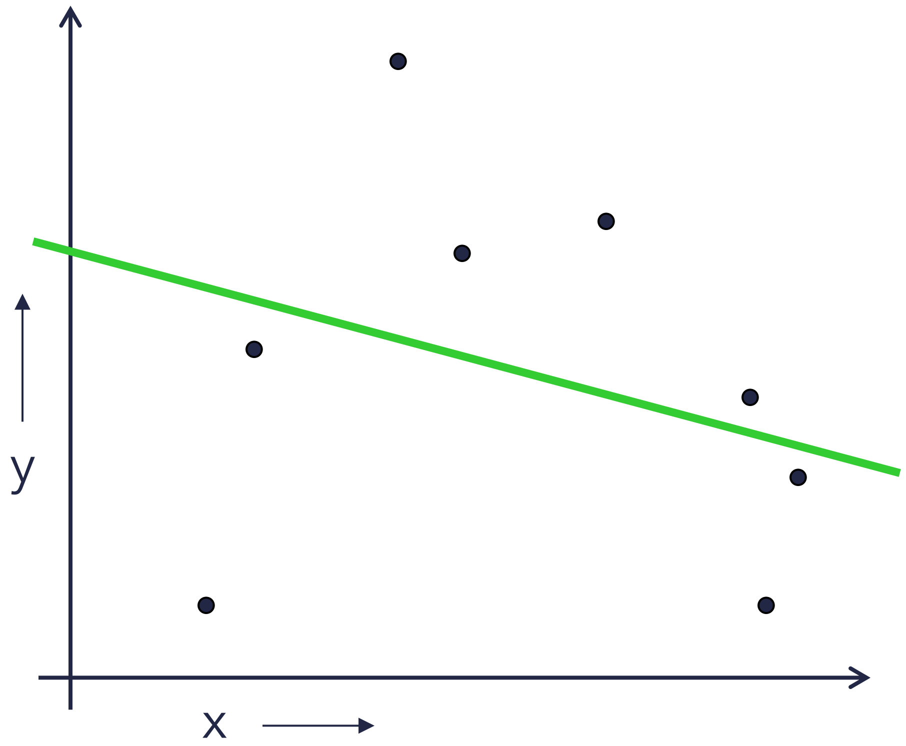
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset

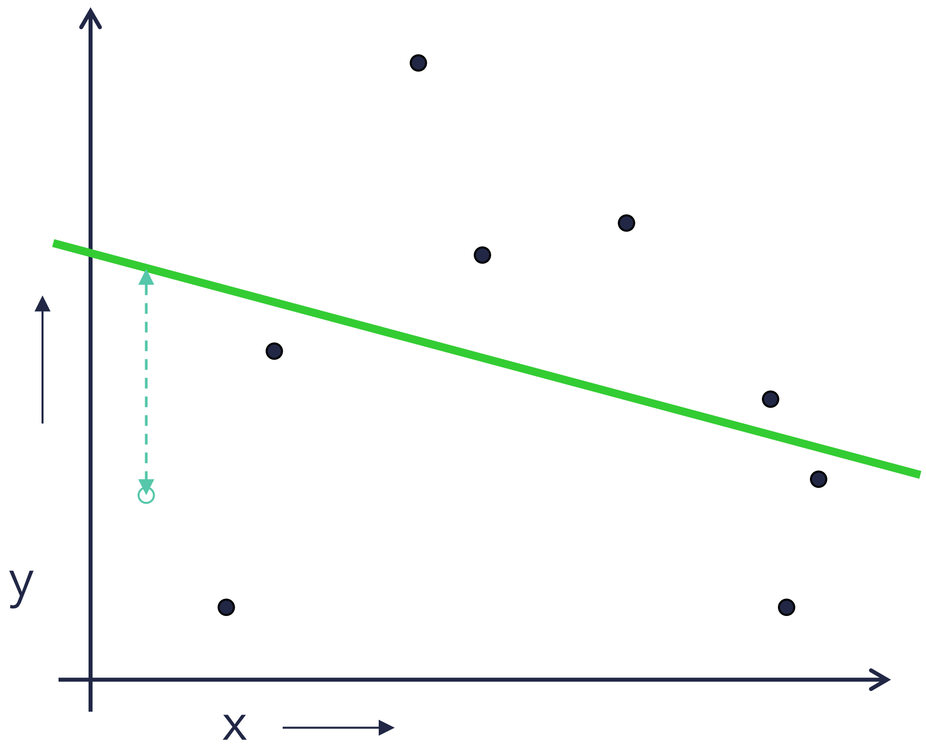
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining (R-1) datapoints

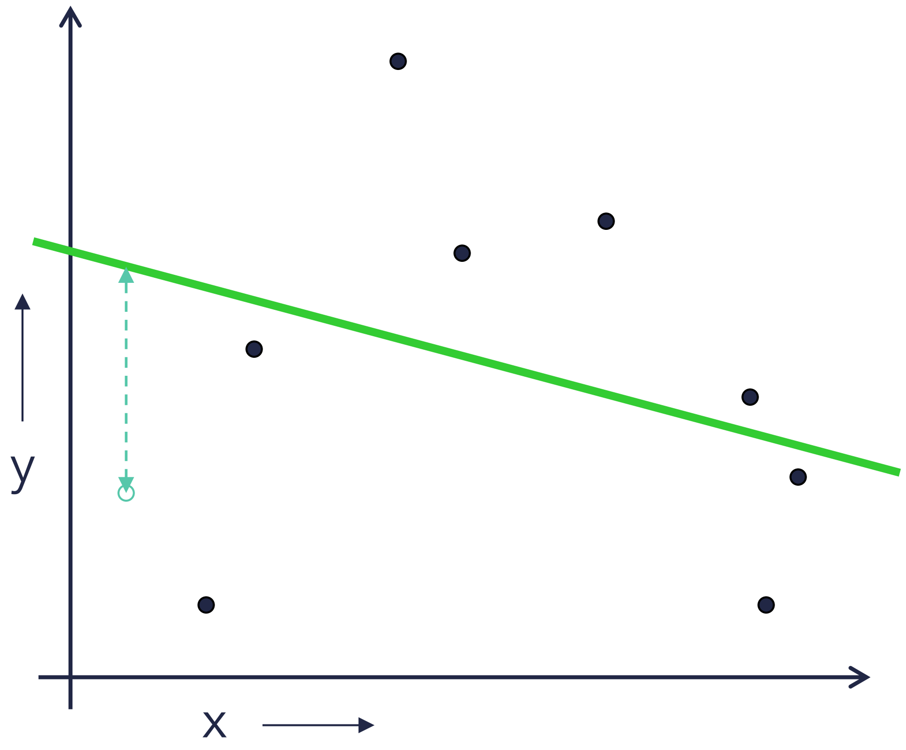
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining (R-1) datapoints
4. Note your error (x_k, y_k)

LOOCV (Leave-one-out Cross Validation)

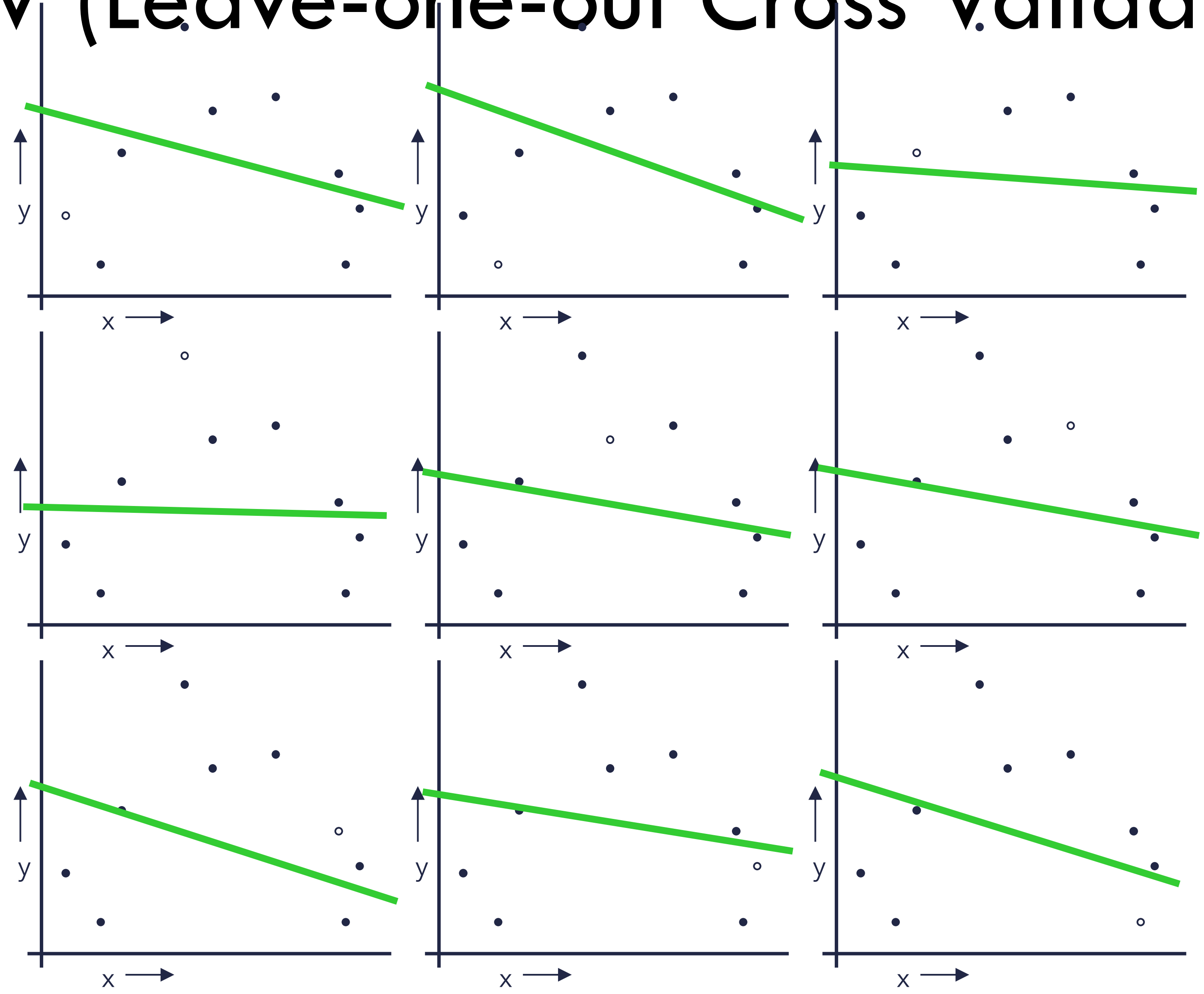


For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining (R-1) datapoints
4. Note your error (x_k, y_k)

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

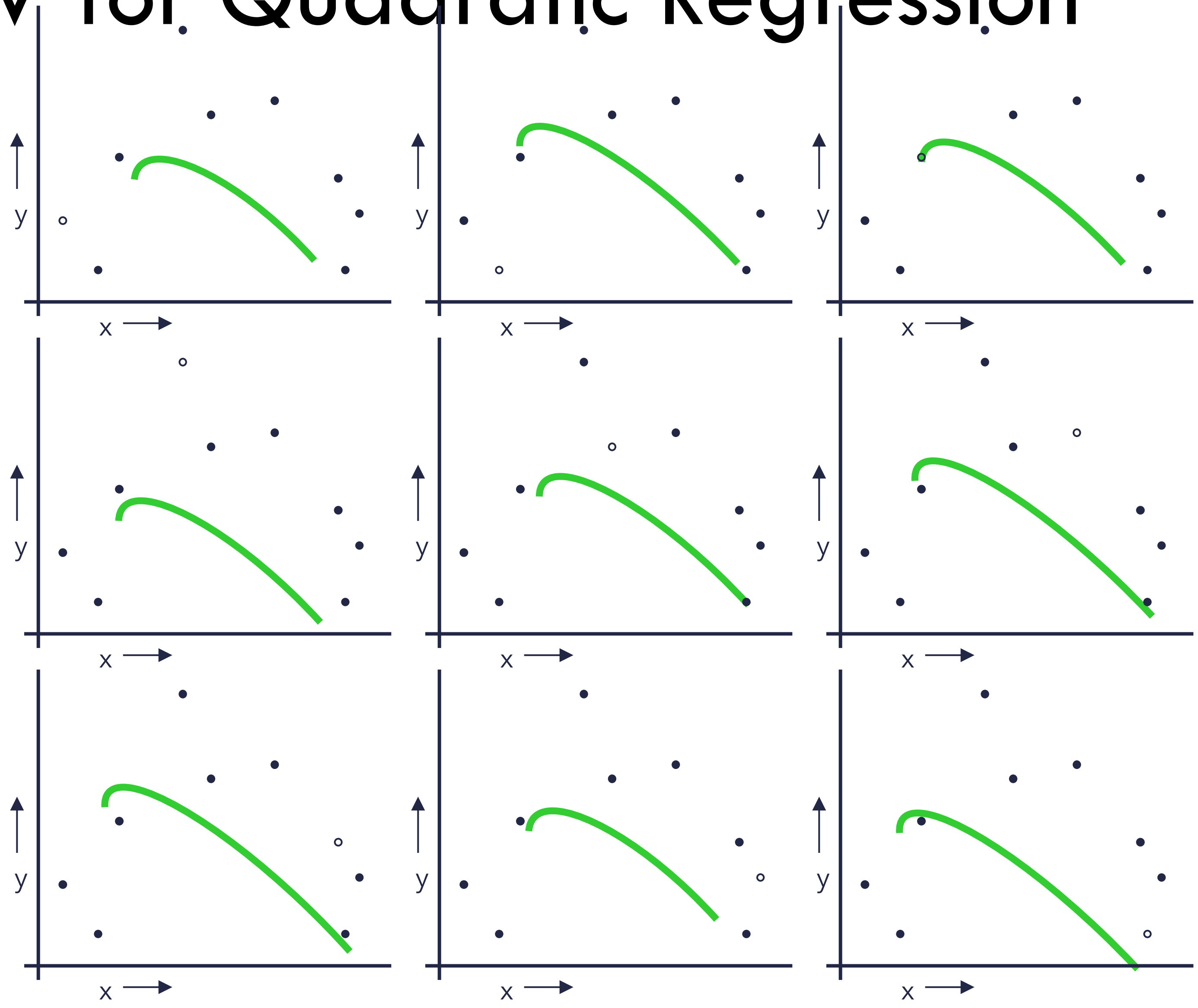
LOOCV (Leave-one-out Cross Validation)



$MSE_{LOOCV} = 2.12$

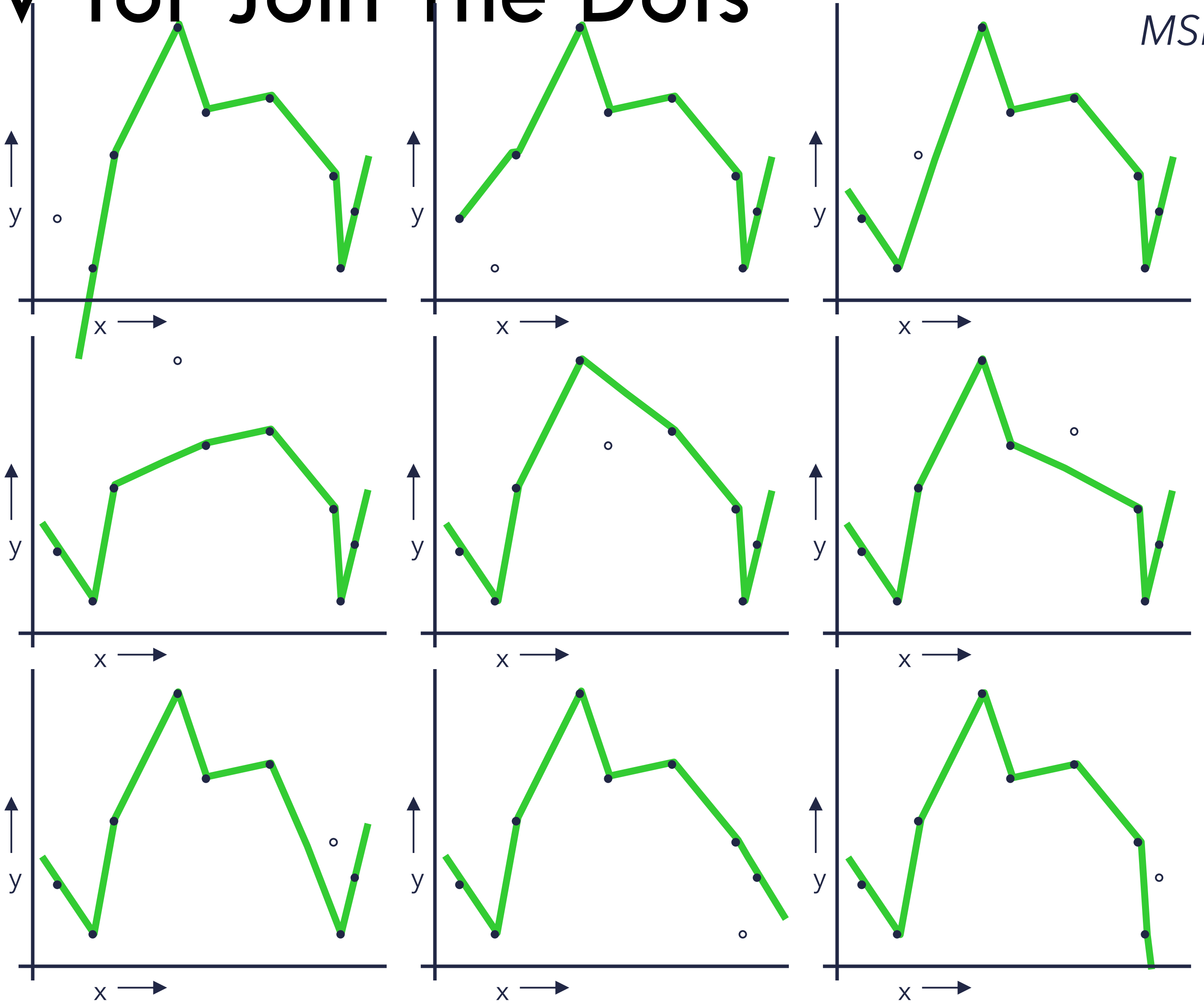
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

LOOCV for Quadratic Regression



$MSE_{LOOCV} = 0.962$

LOOCV for Join The Dots



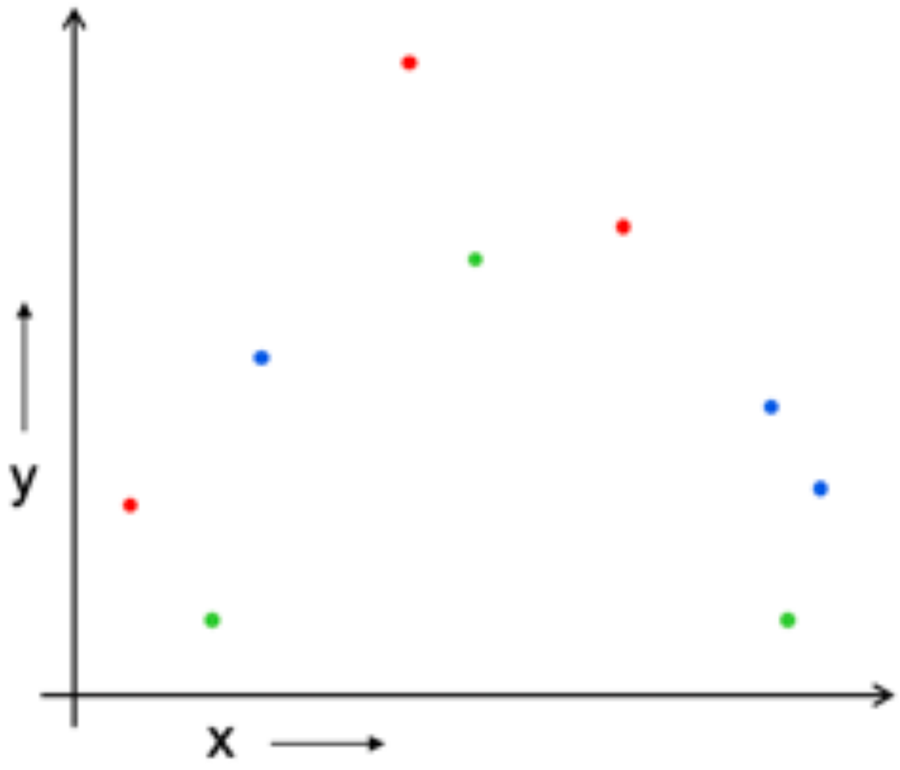
$MSE_{LOOCV} = 3.33$

Which kind of Validation?

	Downside	Upside
Test-set 70%-30%	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive	Doesn't waste data

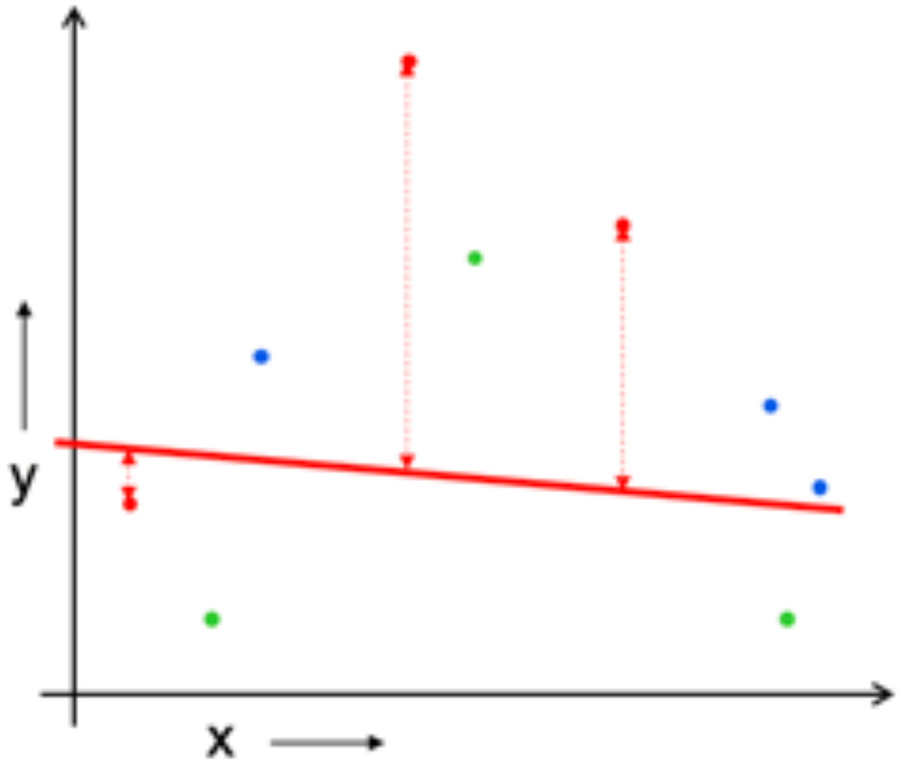
…can we get the best of both worlds?

k-fold Cross Validation



Randomly break the dataset into *k* partitions (in our example we'll have *k*=3 partitions colored Red, Green and Blue)

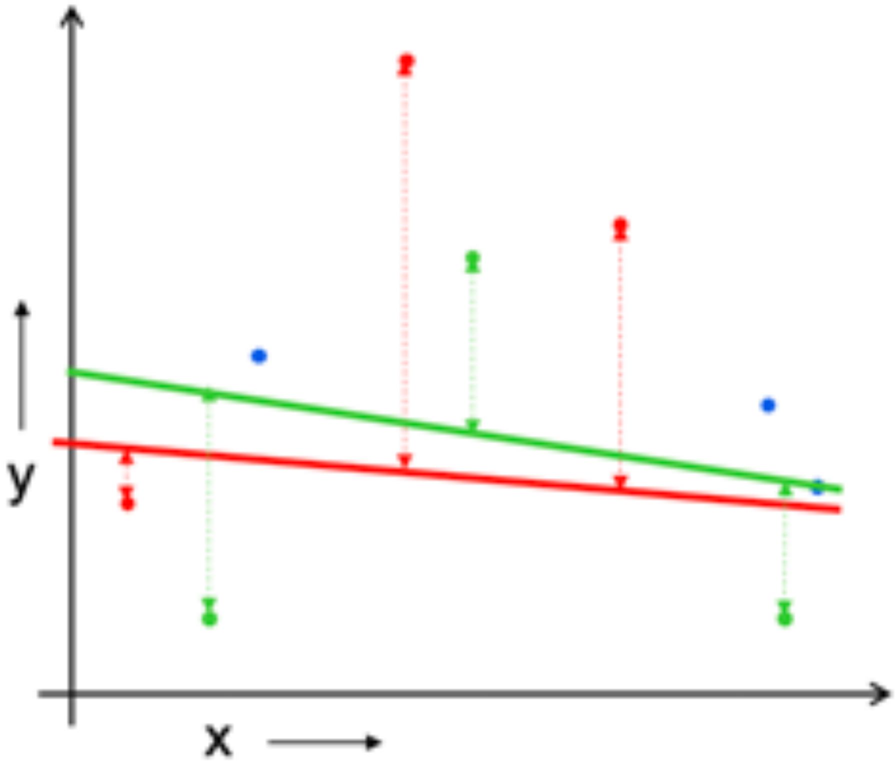
k-fold Cross Validation



Randomly break the dataset into *k* partitions (in our example we'll have *k*=3 partitions colored Red, Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

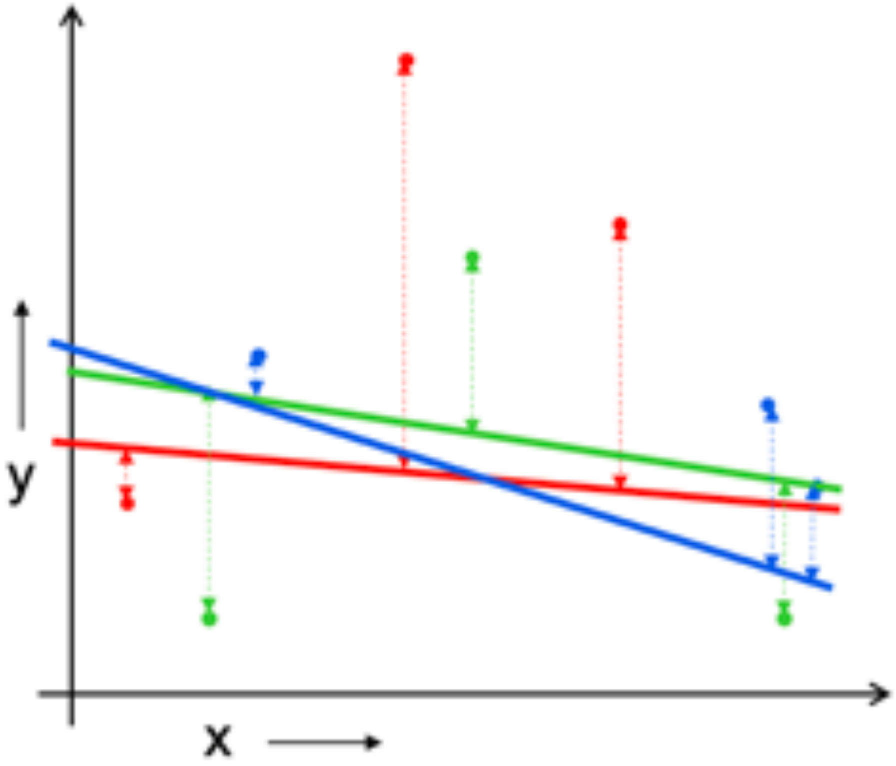
k-fold Cross Validation



Randomly break the dataset into *k* partitions (in our example we'll have *k*=3 partitions colored Red, Green and Blue)

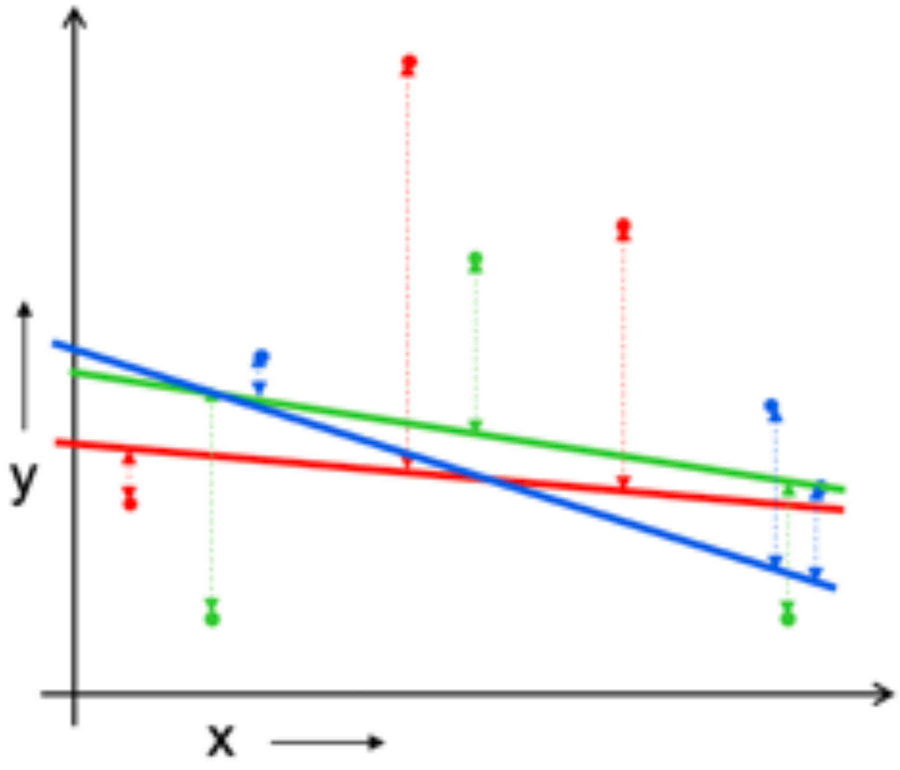
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

k-fold Cross Validation



Randomly break the dataset into *k* partitions (in our example we'll have *k*=3 partitions colored Red, Green and Blue)
For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

k-fold Cross Validation

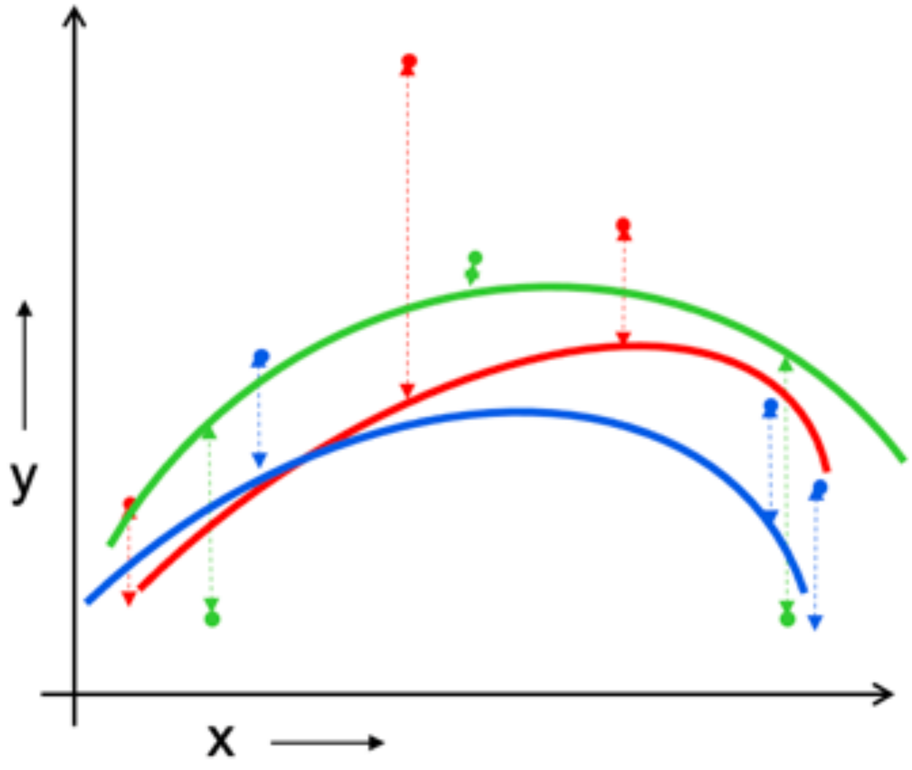


Linear Regression MSE_{3FOLD}=2.05

Randomly break the dataset into *k* partitions (in our example we'll have *k*=3 partitions colored Red, Green and Blue)

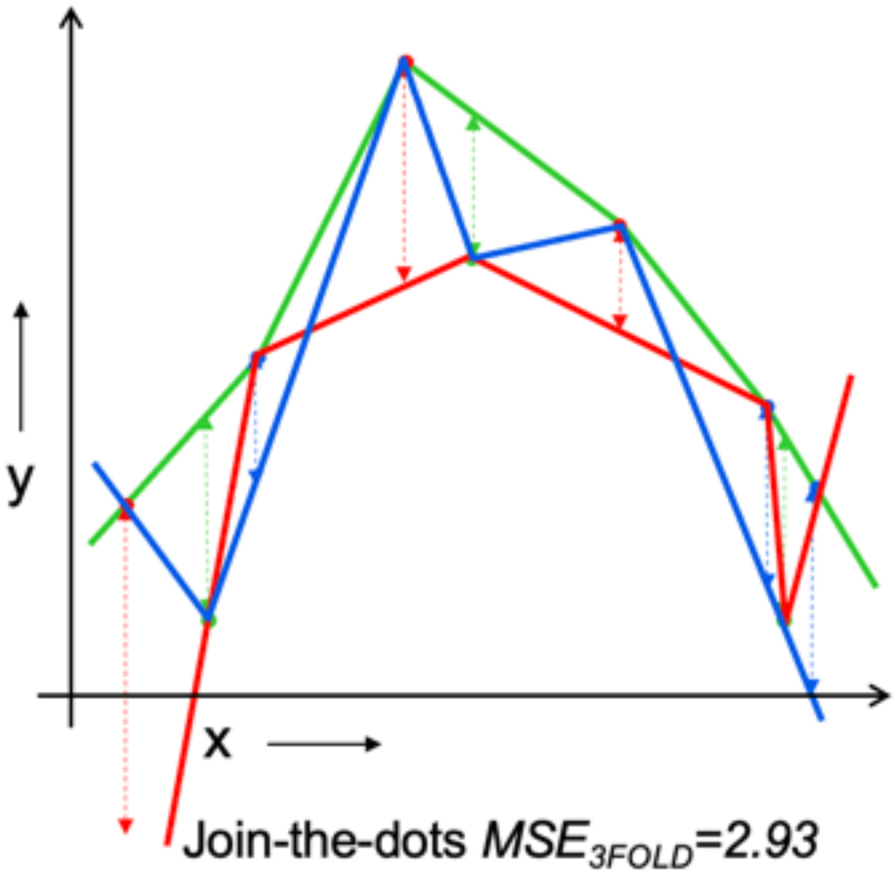
Then report the mean error.

k-fold Cross Validation



Quadratic Regression $MSE_{3FOLD}=1.11$

k-fold Cross Validation

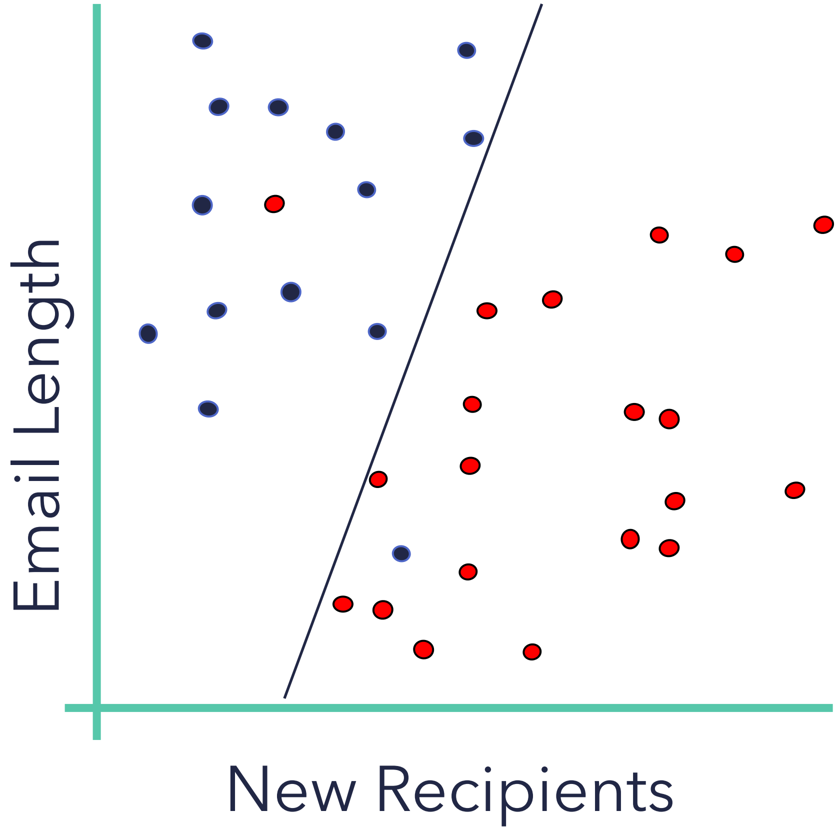


Which kind of Cross Validation?

	Downside	Upside
Test-set	Variance: unreliable estimate of future performance.	Cheap.
Leave-one-out	Expensive. Has some weird behavior.	Doesn't waste data.
10-fold	Wastes 10% of the data. 10 times more expensive than test set.	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastes more data than 10-fold. More Expensive than test set.	Slightly better than test-set.
R-fold	Identical to Leave-one-out.	

Evaluating what has been learned

1. We randomly select a portion of the data to be used for training (the training set)
2. Train the model on the training set
3. Once the model is trained, we run the model on the remaining instances (the test set) to see how it performs



Confusion Matrix

Classified as

Actual

	Black	Red
Black	7	1
Red	0	5

Not by Accuracy alone

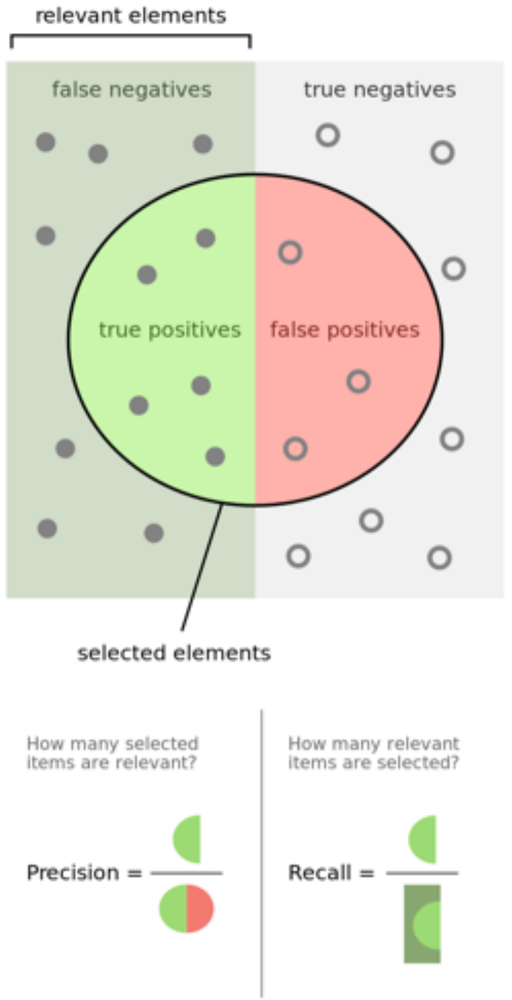
- Higher accuracy does not necessarily simply better performance on target task
- Classes are often very skewed
  - Malicious insider is rare
  - What if 99:1 split?
  - Always say "no" -- 99% default accuracy
- Errors have different costs
  - Deleting an email vs. machine infected

Other

	Predicted class	
True Class	Yes	No
Yes	TP: True Positive	FN: False Negative
No	FP: False Positive	TN: True Negative

- Error rate = # of errors / # of instances = (FN+FP) / N
- Recall (TPR) = # of found positives / # of positives
- = TP / (TP+FN) = sensitivity = hit rate
- Precision = # of found positives / # of found = TP / (TP+FP)
- Specificity (TNR) = TN / (TN+FP)
- False alarm rate (FPR) = FP / (FP+TN) = 1 - Specificity

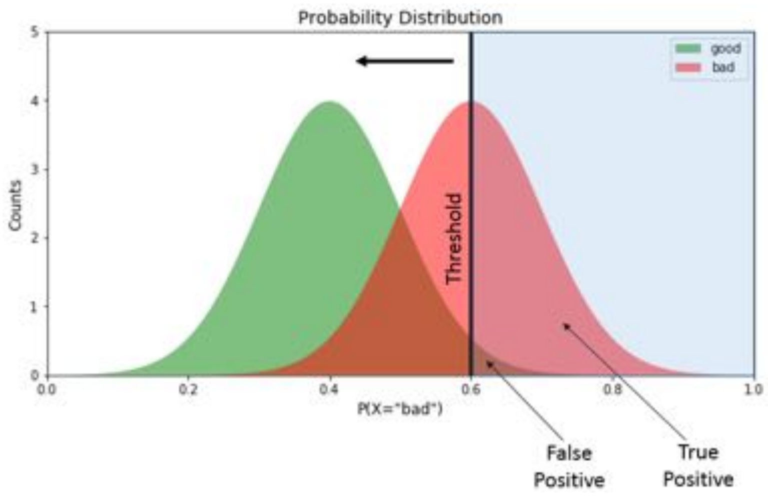
Precision & Recall



Taken from: https://en.wikipedia.org/wiki/Precision_and_recall

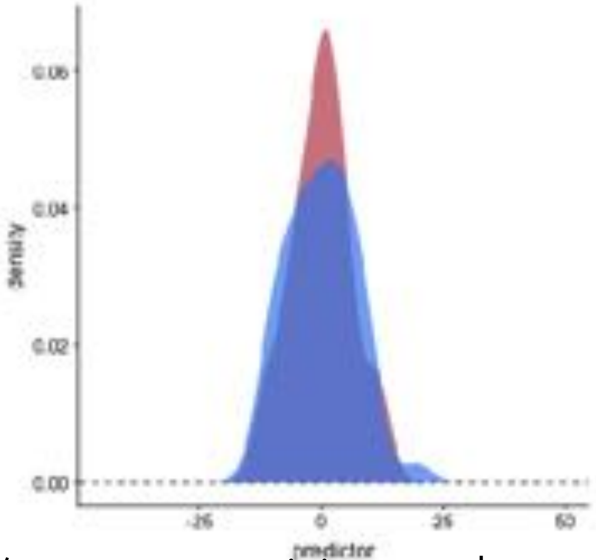
ROC Curve

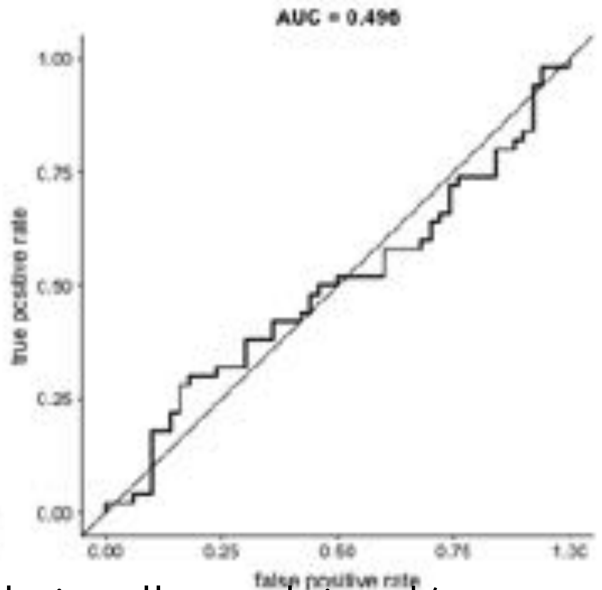
- Receiver Operating Characteristic (ROC) curve:
    - A technique for visualizing, organizing and selecting classifiers based on their performance
    - Two-dimensional graph in which the TPR (Recall) is plotted on the Y axis and the FPR (False alarm rate) is plotted on the X axis
    - Depicts relative tradeoffs between benefits (true positives) and costs (false positives)



$$TPR = \frac{True\ Positive}{Total\ Positive}$$

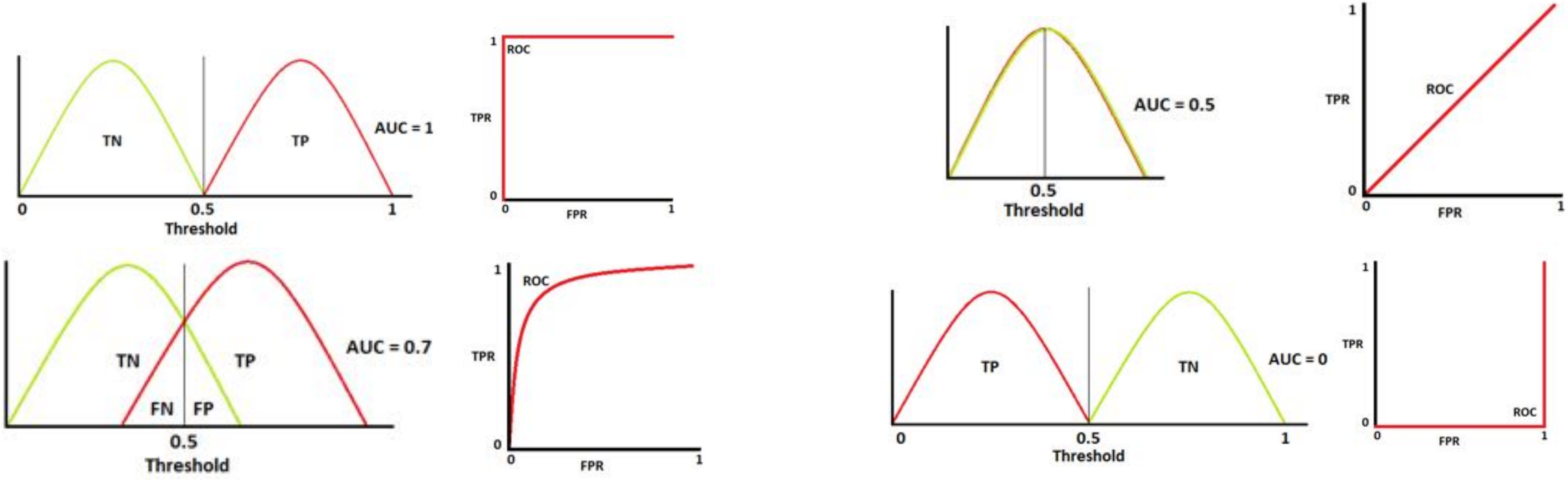
$$FPR = \frac{False\ Positive}{Total\ Negative}$$





Source: https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/

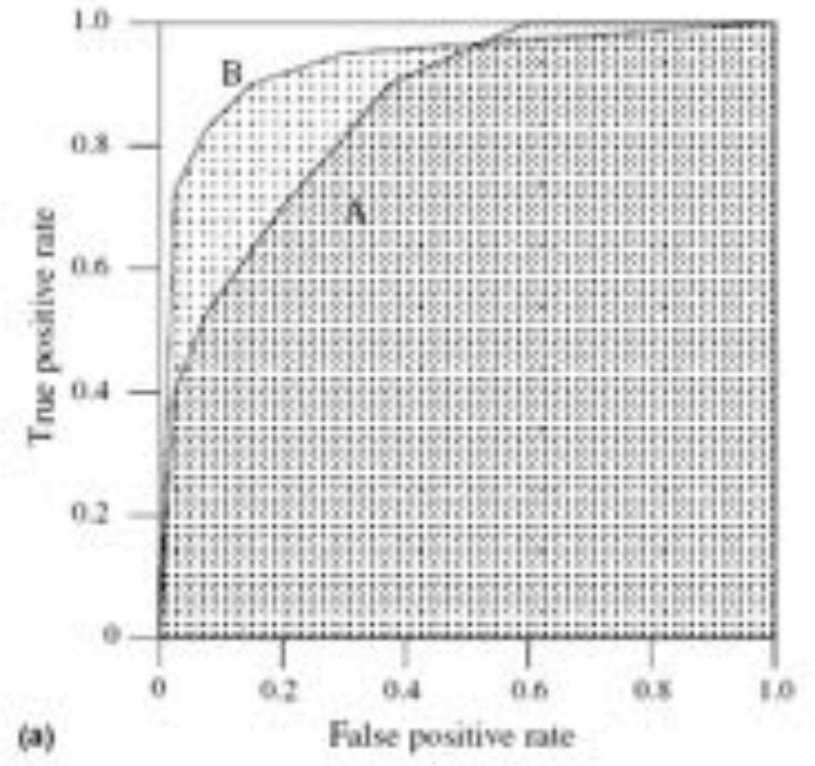
ROC Curve For Binary Classification



Source: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Area under ROC curve (AUC)

- Comparing two ROC curves:
- The graph represents the areas under two ROC curves, A and B
- Classifier B has greater area and therefore better average performance

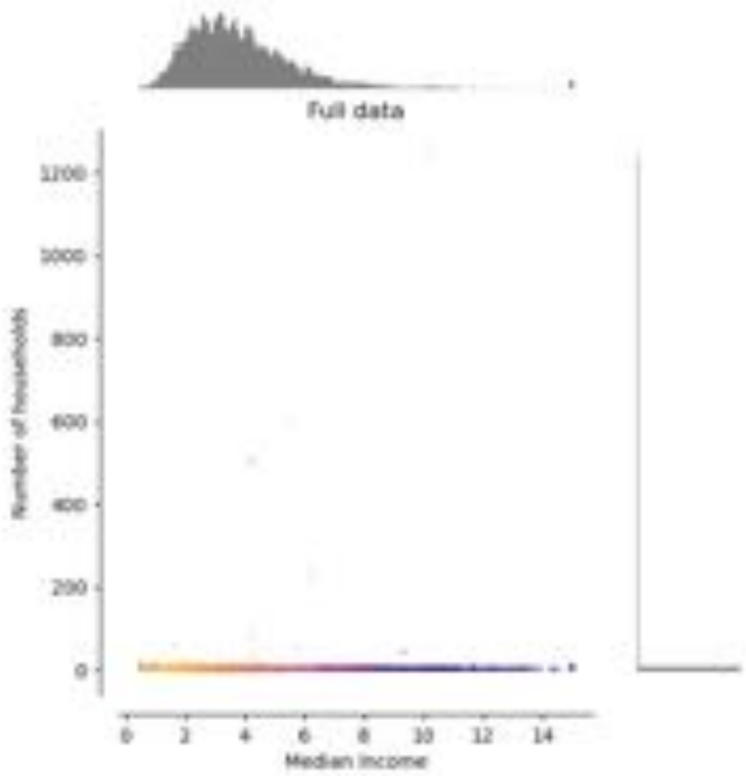


Normalization

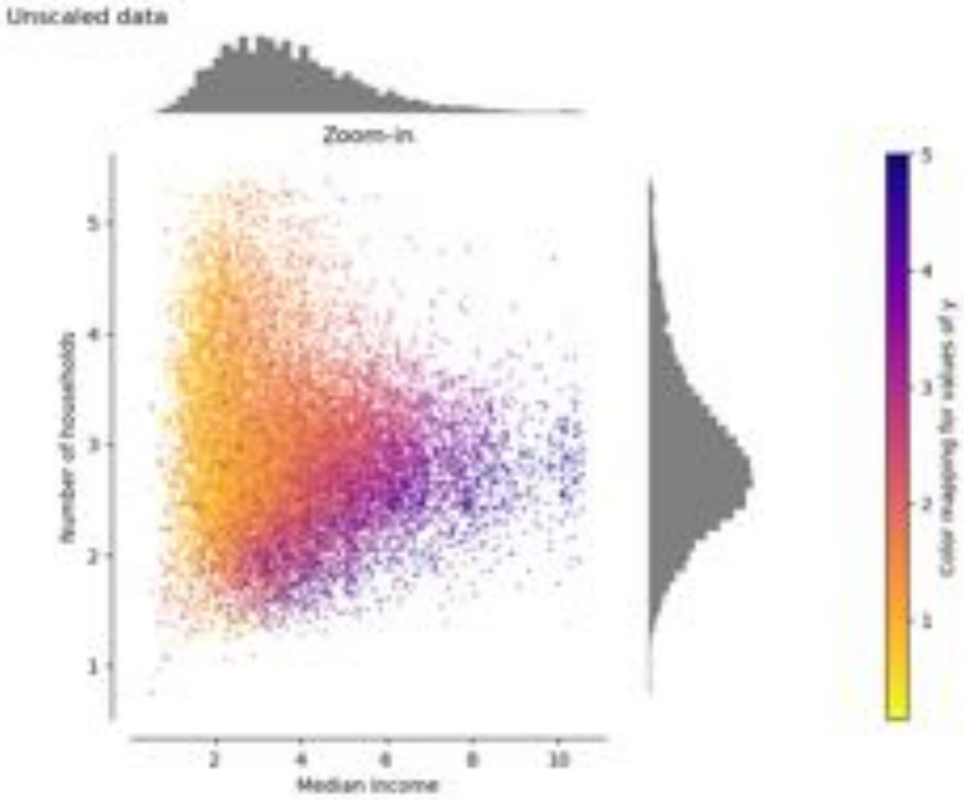
Please also check the IPYTHON

Normalization

Original Data



Zoom in without outliers



Source: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py

StandardScaler

```python
from sklearn.preprocessing import StandardScaler
import numpy as np

# 4 samples/observations and 2 variables/features
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

print(data)
[[0, 0],
 [1, 0],
 [0, 1],
 [1, 1]])

print(scaled_data)
[[-1. -1.]
 [ 1. -1.]
 [-1.  1.]
 [ 1.  1.]]
```

- Removes the mean and scales the data to unit variance.
  - Unit variance means that the standard deviation of a sample as well as the variance will tend towards 1 as the sample size tends towards infinity.
  - This is done per dimension --> each dim will have average $\mu = 0$ and $\sigma = 1$.
- The scaling shrinks the range of the feature values as shown in the left figure below. However, the outliers have an influence when computing the empirical mean and standard deviation.
- StandardScaler therefore cannot guarantee balanced feature scales in the presence of outliers.

Standardization:

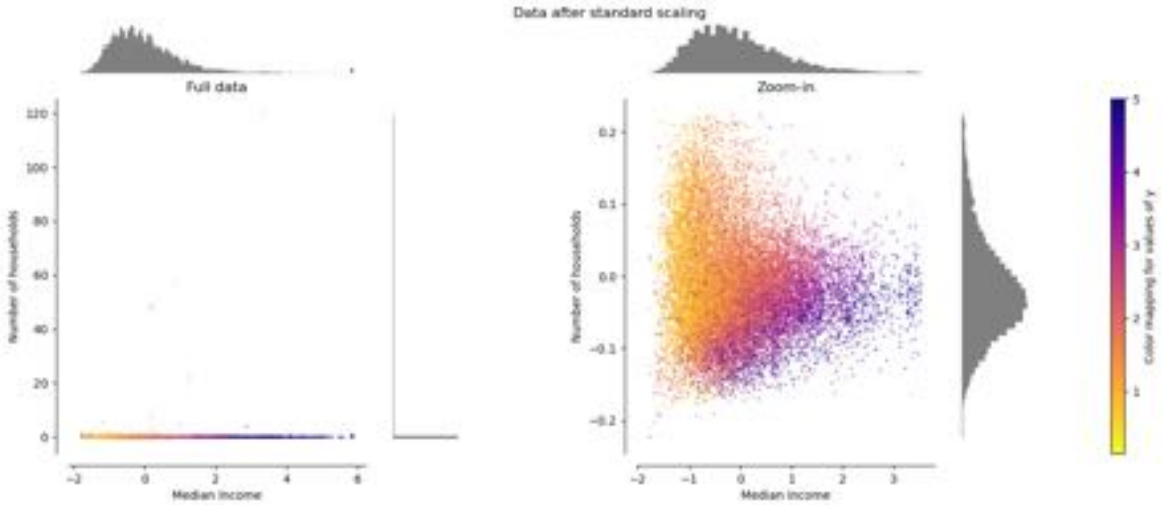
$$z = \frac{x-\mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

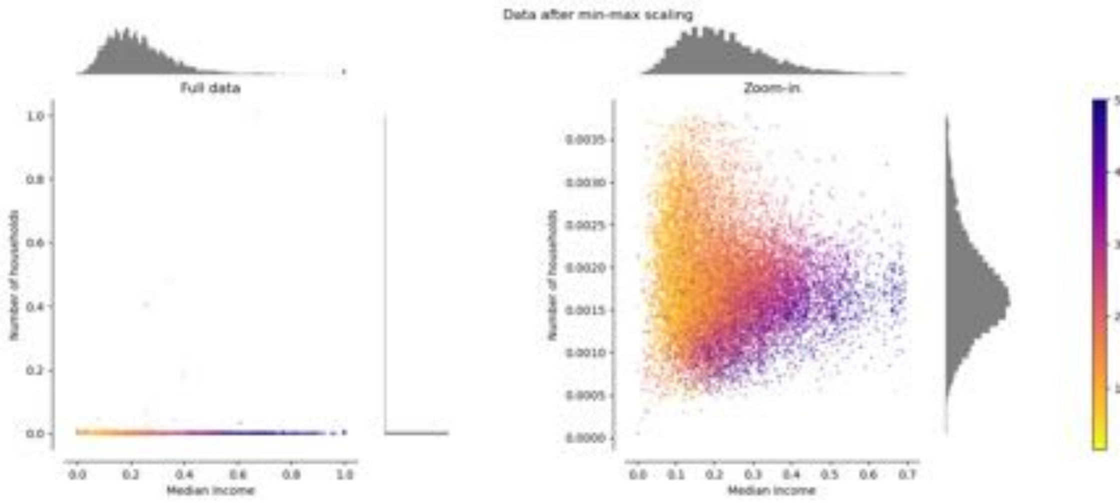
and standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$$



MinMaxScaler

- rescales the data set such that all feature values are in the range [0, 1] as shown in the right panel below.
- However, this scaling compresses all inliers into the narrow range [0, 0.005] for the transformed number of households.
- Both StandardScaler and MinMaxScaler are very sensitive to the presence of outliers.
- X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
  X_scaled = X_std * (max - min) + min



Feature Selection

Why feature selection?

- A large number of extracted features, some of which redundant or irrelevant, present several problems
- Misleading the learning algorithm
- Over-fitting and therefore reducing generality
- Increasing model complexity and processing time
- Large number of features requires more examples
- Requires more resources

Goal

- Select a subset of relevant features that eventually will lead to a better, faster, and easier to understand model

Number of new Recipients	Email Length (K)	Country (IP)	Customer Type	Email Type
0	2	Germany	Gold	Ham
1	4	Germany	Silver	Ham
5	2	Nigeria	Bronze	Spam
2	4	Russia	Bronze	Spam
3	4	Germany	Bronze	Ham
0	1	USA	Silver	Ham
4	2	USA	Silver	Spam

Why feature selection?

- Assume *m* features
- We want to select *k*

$$\binom{m}{k} = \frac{m!}{k!\,(m-k)!}$$

- Example: *m*=20 *k*=5 ➔ 15,504 combinations

Feature selection methods

- Univariate vs. Multivariate
- Univariate - selecting the best features based on univariate statistical tests.
  - It can be seen as a preprocessing step to an estimator.

Univariate method

- Assign a rank to each feature independently
- Indicates how good the feature is in discriminating between classes
- Differ in the ranking metric

- Remove useless

Low Variance

- Simple baseline approach to feature selection
- It **removes** all **features** whose **variance** doesn't meet some **threshold**
- It removes all zero-variance features, i.e. features that have the same value in all samples.

- **Example**: remove features that are 1 or 0 in 80% if the samples.
  - **Boolean** features are **Bernoulli random** variables, and the variance of such variables is given by:

$$\mathrm{Var}[X] = p(1-p)$$

  - Thr = 0.8*(1-0.8)

```
>>> from sklearn.feature_selection import VarianceThreshold
>>> X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
>>> sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
>>> sel.fit_transform(X)
array([[0, 1],
       [1, 0],
       [0, 0],
       [1, 1],
       [1, 0],
       [1, 1]])
```

  - As expected the VarianceThreshold has removed the first column which has probability p= 5/6 >*0.8 of containing zeros*

SelectKBest

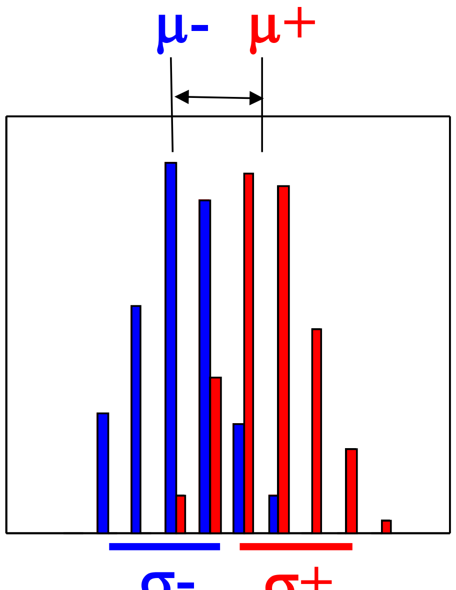
- Removes all but the K highest scoring features
- Select based on different functions
  - Mutual information
  - Chi2
  - Note different functions can work better for specific use cases
- Return univariate scores and p-values (for most variants)

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectKBest
>>> from sklearn.feature_selection import chi2
>>> X, y = load_iris(return_X_y=True)
>>> X.shape
(150, 4)
>>> X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
>>> X_new.shape
(150, 2)
```

Univariate method

Fisher Score

- **Calculates** the difference, described in terms of **mean** and **standard deviation**, between the **positive and negative examples** relative to a certain feature.
- **Features with high quality** should assign **similar values** to instances in the **same class** and **different values** to instances from **different classes**.

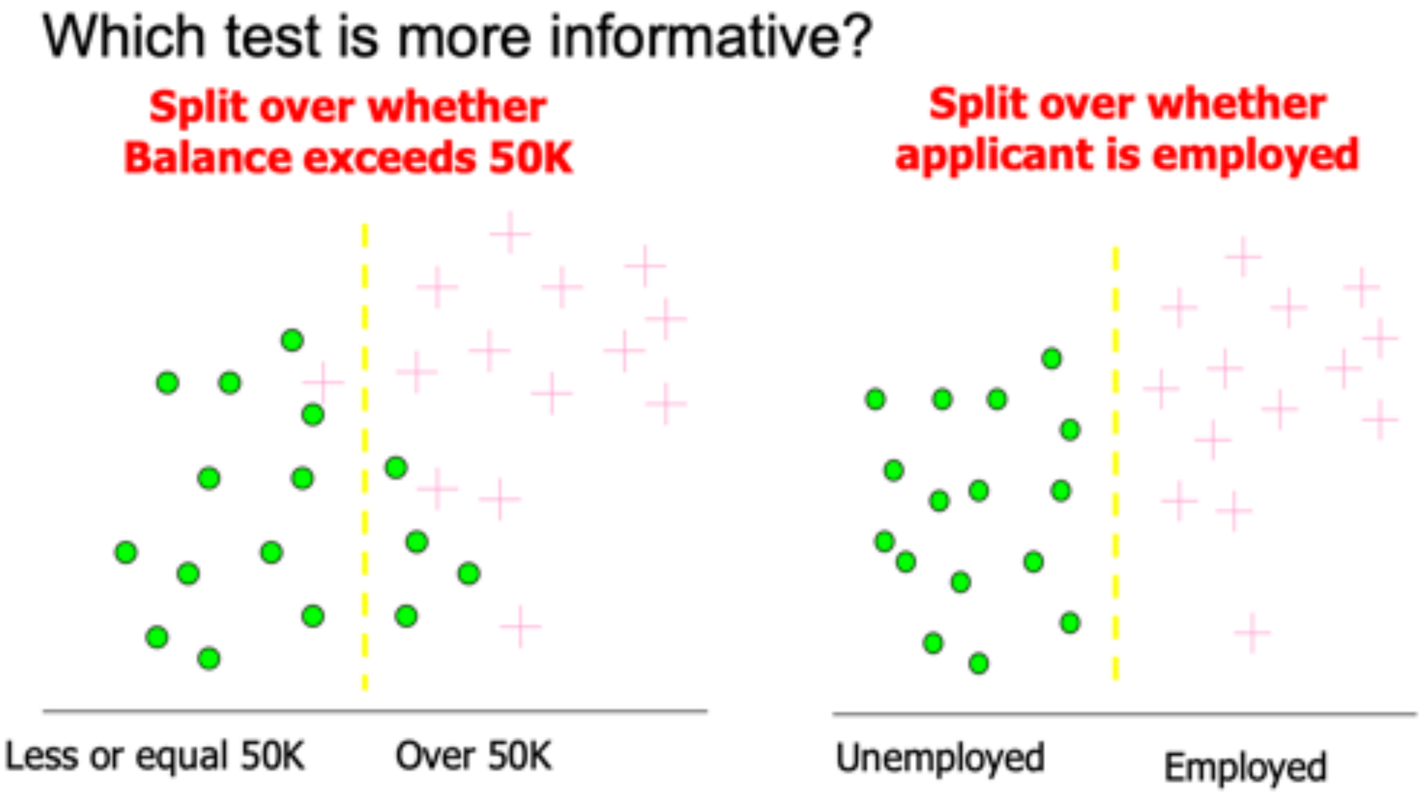


$$R^{(i)} = \frac{\left|\mu_+^{(i)} - \mu_-^{(i)}\right|}{\sigma_+^{(i)} + \sigma_-^{(i)}}$$

Entropy & Information Gain Explained

- Information Gain & Entropy slides taken from Prof. Linda Shapiro
- Source: https://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf

Entropy

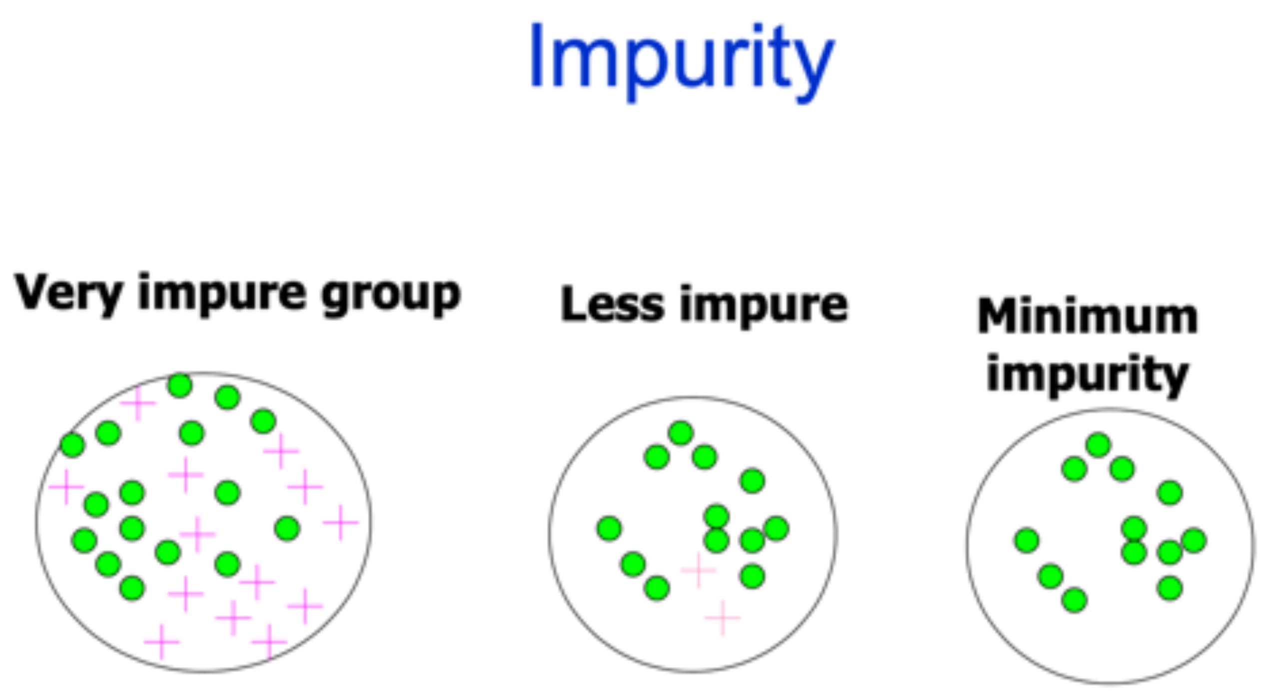

Which test is more informative?
Split over whether Balance exceeds 50K
Split over whether applicant is employed
Less or equal 50K
Over 50K
Unemployed
Employed

Entropy

Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

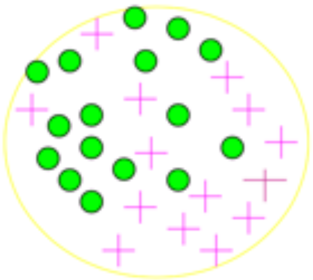
Entropy



Entropy

$\text{Log}_2 1 = 0$

- Entropy = $\sum_i -p_i \log_2 p_i$



p_i is the probability of class i
Compute it as the proportion of class i in the set.

16/30 are green circles; 14/30 are pink crosses
$\log_2(16/30)$ = -.9; $\log_2(14/30)$ = -1.1
Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

- Entropy comes from information theory. The higher the entropy the more the information content.

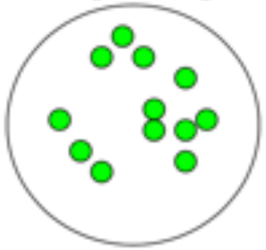
What does that mean for learning from examples?

Entropy – 2 class cases

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0

  not a good training set for learning

Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

  good training set for learning

Maximum impurity

Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

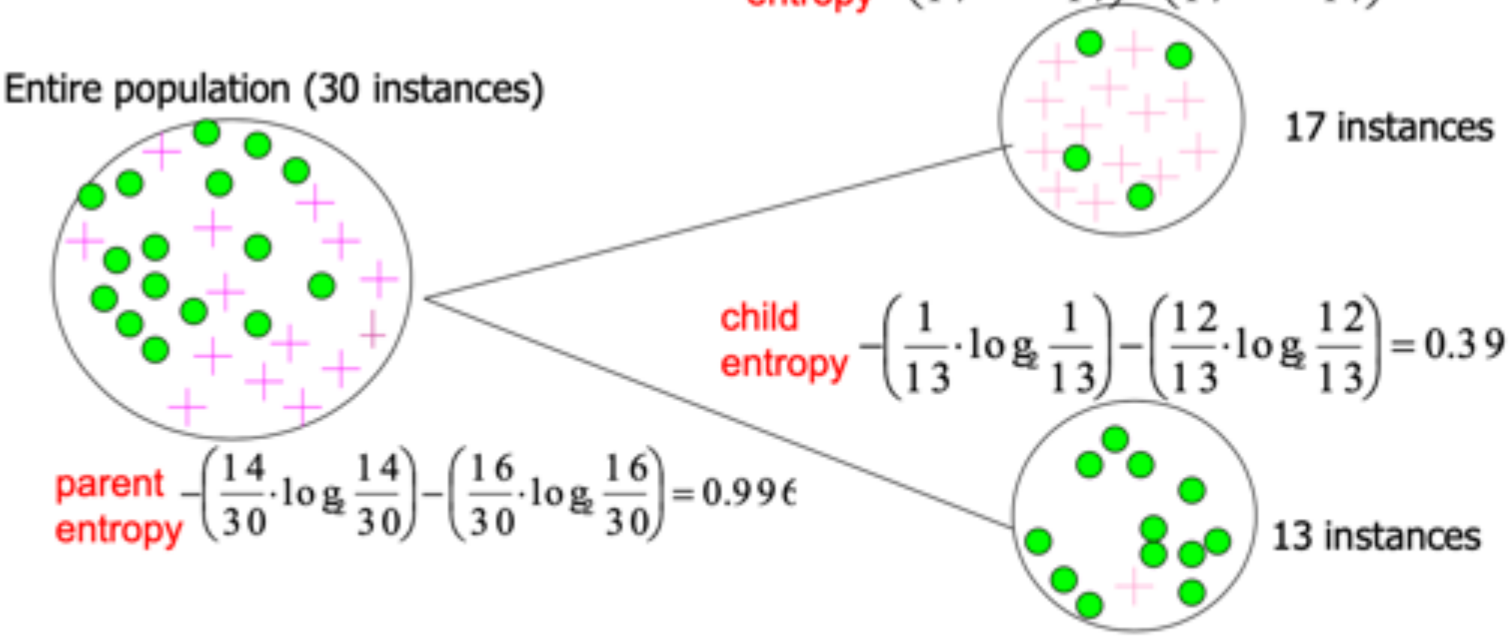
- Information gain tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38 **for this split**

Information Gain Example

Training Set: 3 features and 2 classes

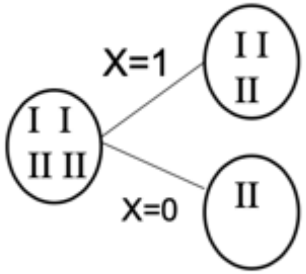
X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Information Gain Example

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute X

If X is the best attribute,
this node would be further split.



$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= .5284 + .39$
$= .9184$

$E_{child2} = 0$

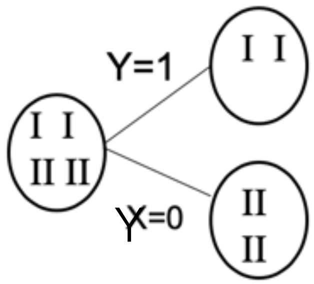
$E_{parent} = 1$

GAIN = 1 – (3/4)(.9184) – (1/4)(0) = .3112

Information Gain Example

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Y



$E_{child1} = 0$

$E_{child2} = 0$

$E_{parent} = 1$

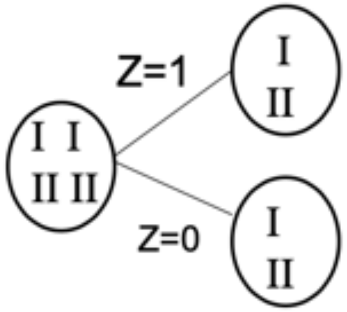
GAIN = 1 –(1/2) 0 – (1/2)0 = 1; BEST ONE

Note: The entropy here is 0 but Information Gain Is 1

Information Gain Example

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Z



$E_{child1} = 1$

$E_{child2} = 1$

$E_{parent} = 1$

GAIN = 1 – (1/2)(1) – (1/2)(1) = 0 ie. NO GAIN; WORST

Note: Here the Entropy is 1
But the Information Gain is
0