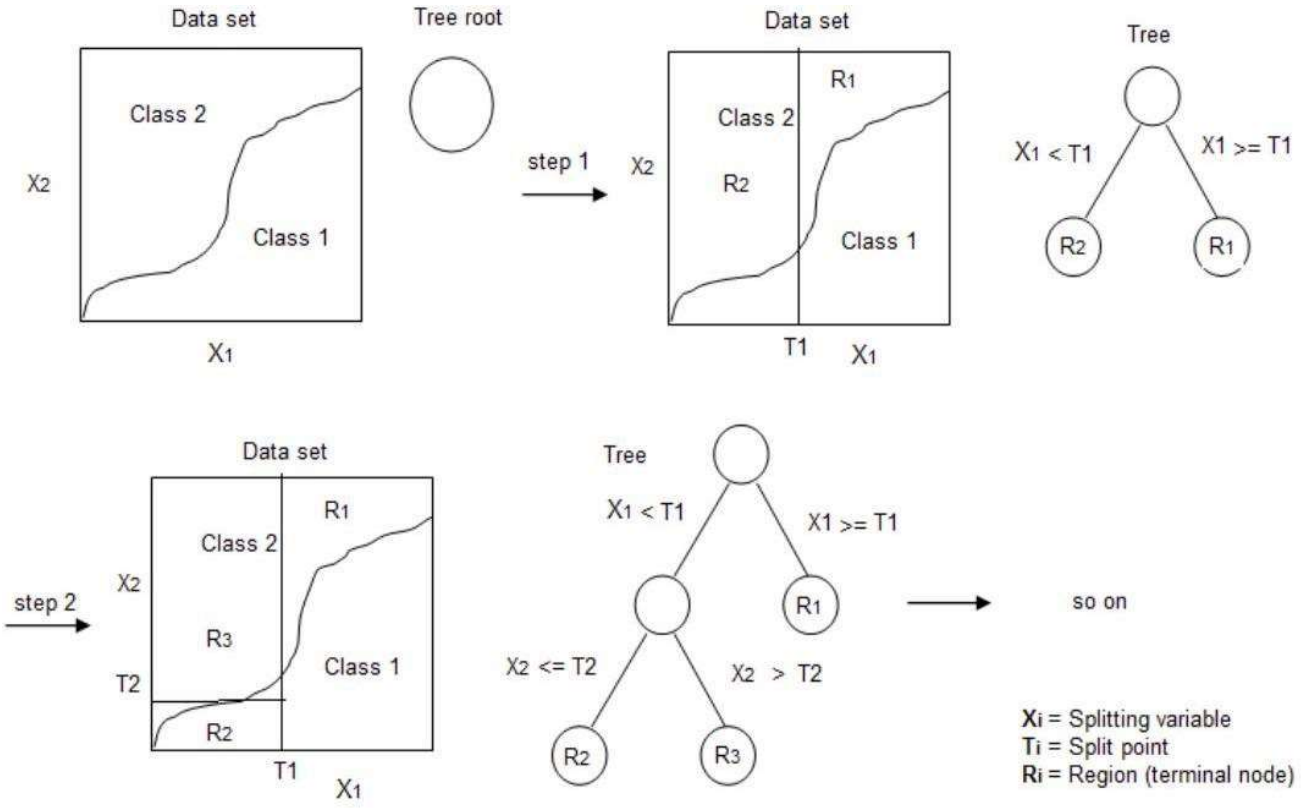
TREE BASED METHODS

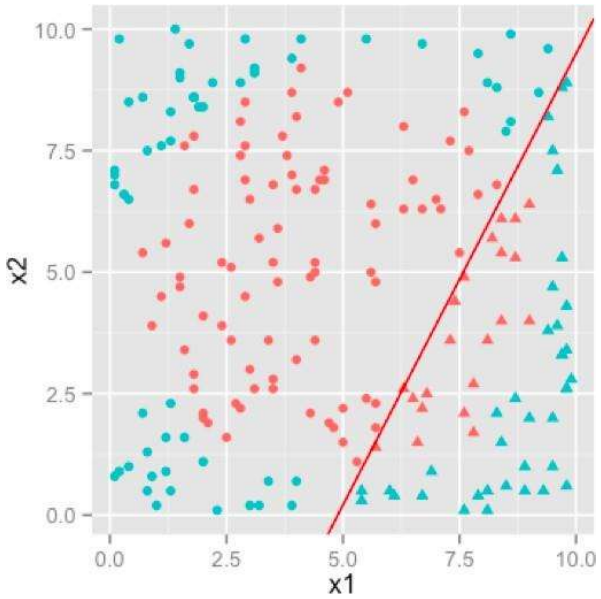
Definitions: Partition

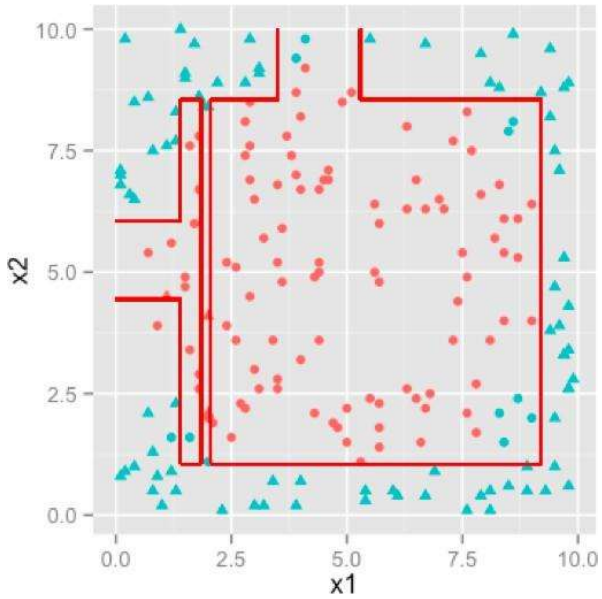
Overview

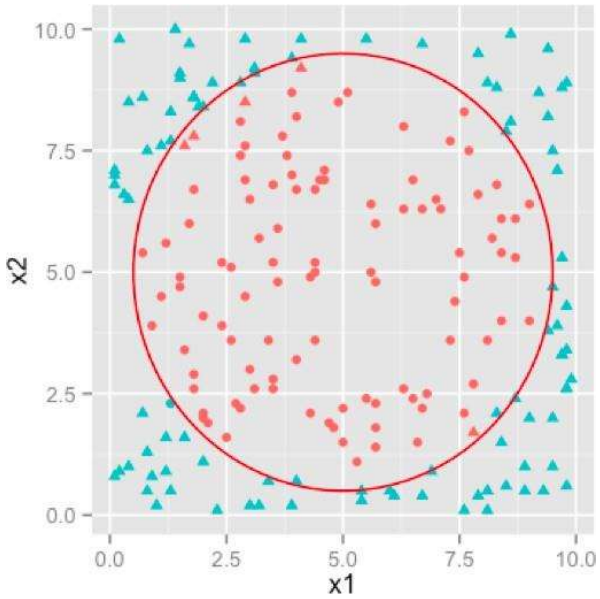


Definitions: Partition

Logistic Reg. vs. Trees vs. SVM

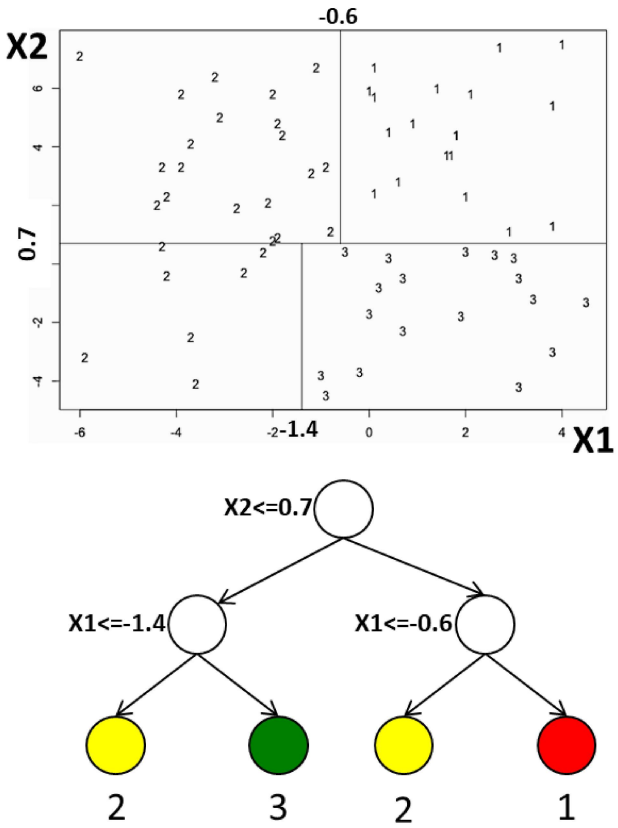






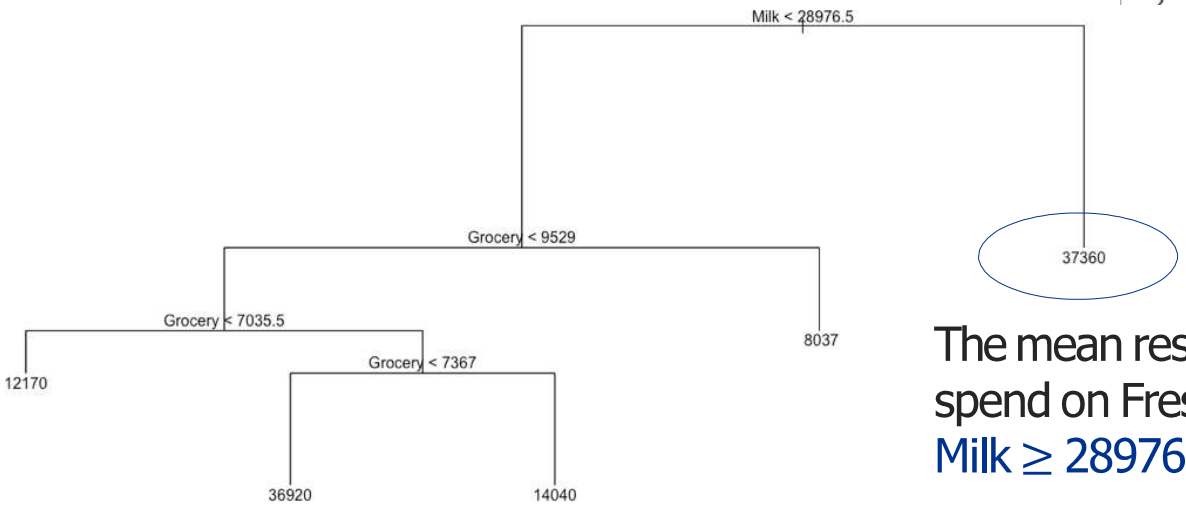
The tree-based model

- The result is:
  - The tree, which represents the recursive partition method
  - The leaves of the tree, which represent the simple model that applies to each final smaller data region
- Interpretation
  - Starting at the root node, a sequence of questions about the predictors are made (all questions are labeled within tree nodes)
  - The answers are labeled in the branches between nodes



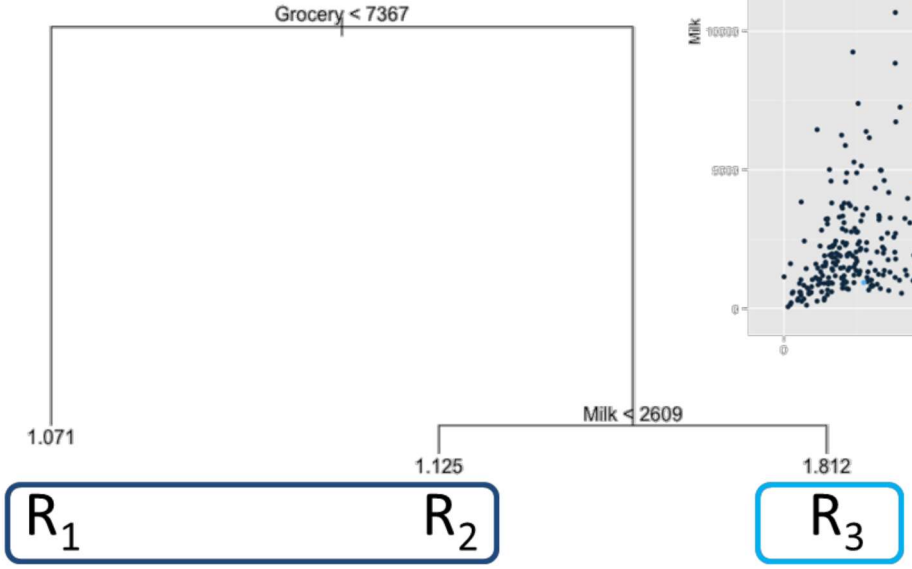
Regression Trees

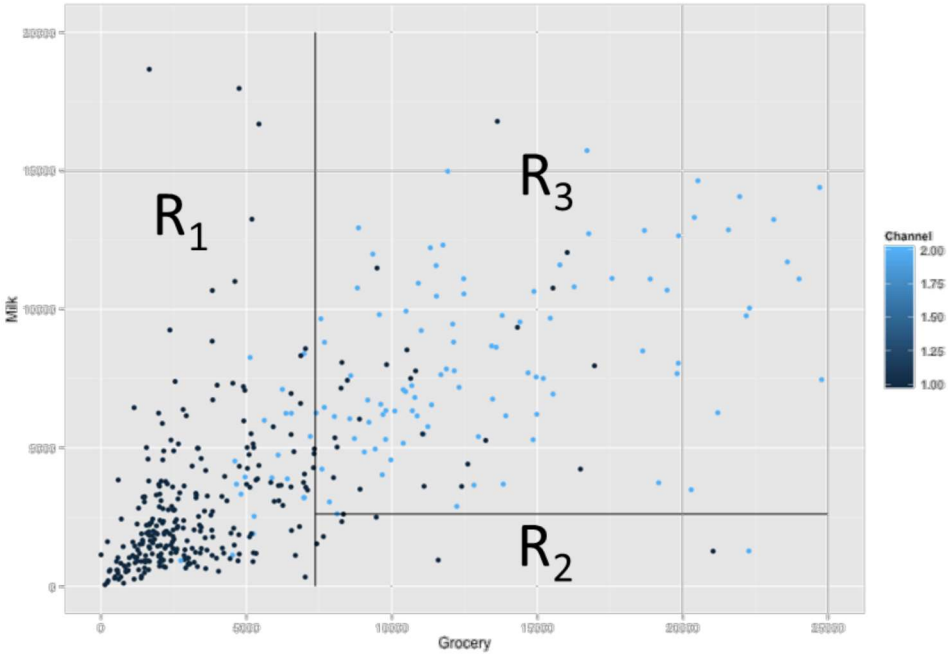
Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
2	3	12669	9656	7561	214	2674	1338
2	3	7057	9810	9568	1762	3293	1776
2	3	6353	8808	7684	2405	3516	7844
1	3	13265	1196	4221	6404	507	1788
2	3	22615	5410	7198	3915	1777	5185
2	3	9413	8259	5126	666	1795	1451
2	3	12126	3199	6975	480	3140	545
2	3	7579	4956	9426	1669	3321	2566



The mean response value for the annual spend on Fresh products in the data set with Milk ≥ 28976.5

Classification Trees





Cost function

- The goal is to find Regions that minimize the RSS:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- At each step, ALL possible predictors "***j***" and all possible cutpoint values "s", such as we found the pair of half-planes

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}$$

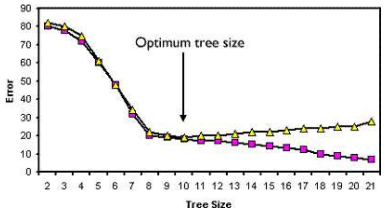
- We seek the values of "j" and "s" that minimize the expression

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Tree Pruning

- Trees normally overfit the data
- Simply making the tree a bit smaller lead to lower variance, though at the cost of a little more bias
- Best strategy to prune the tree?

a) Estimate cross-validation error for every possible tree size



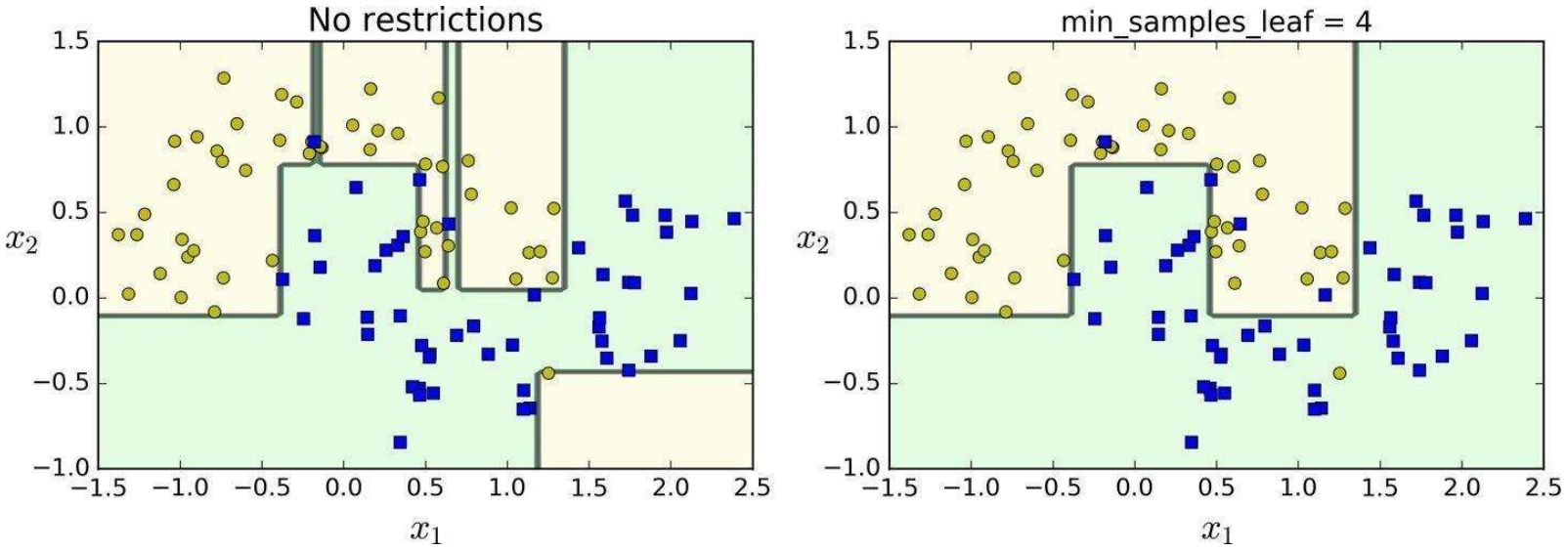
b) Use cost-complexity pruning: Introduce a regularization term in the loss function to penalize large trees

$$\sum_{j=1}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

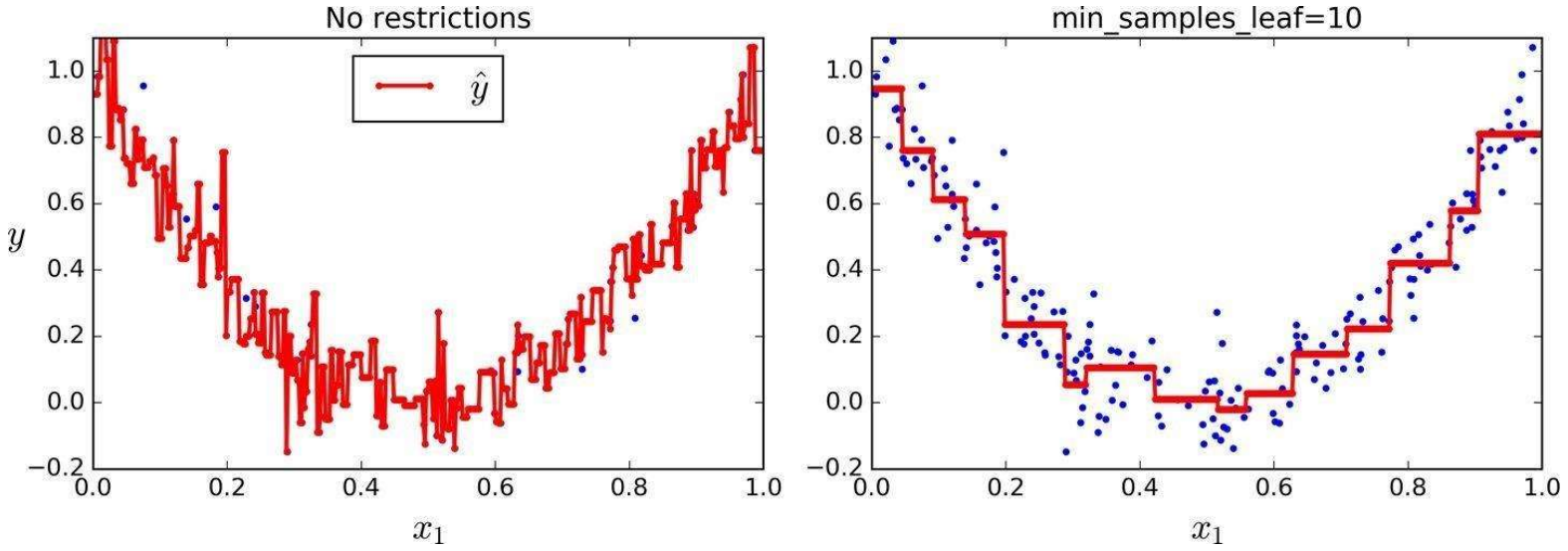
# of leaves

Tuning parameter

Classification Tree Pruning

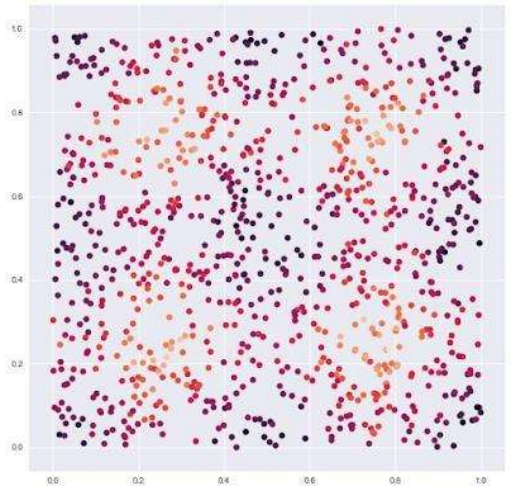


Regression Tree Pruning

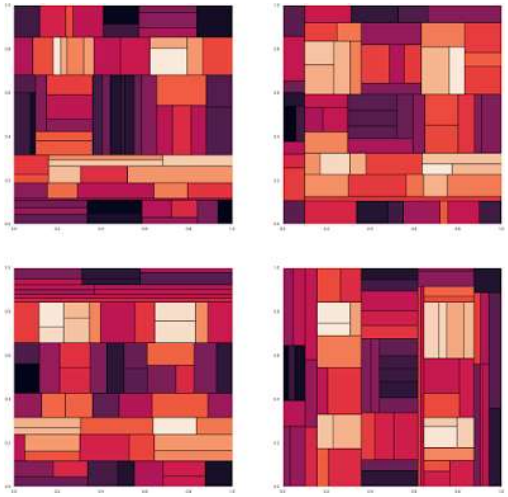


Ensemble Models

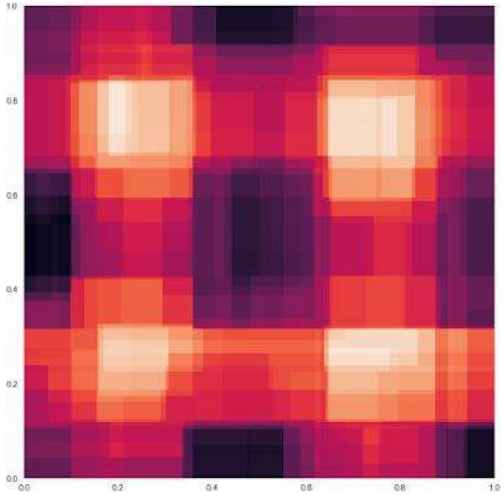
Bagging



Dataset



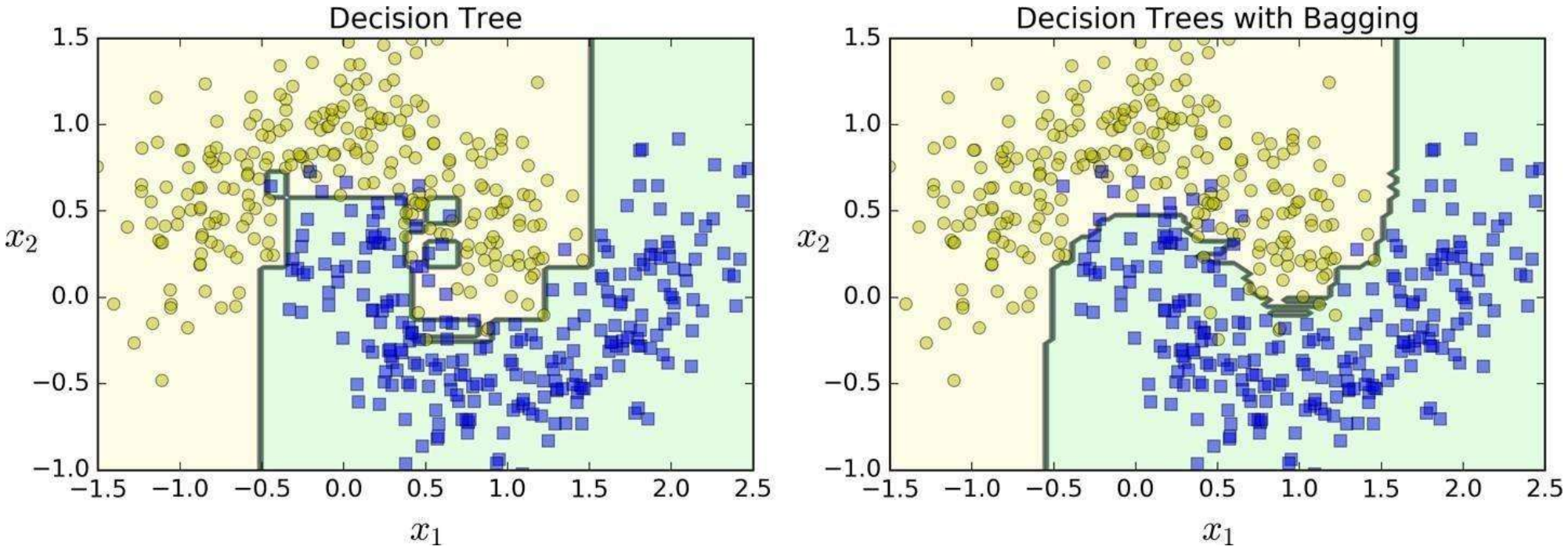
Individual
Trees



Bagging

Source: http://structuringtheunstructured.blogspot.com/2017/11/coloring-with-random-forests.html

Bagging


Decision Tree
Decision Trees with Bagging
x_1
x_2
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
2.0
2.5

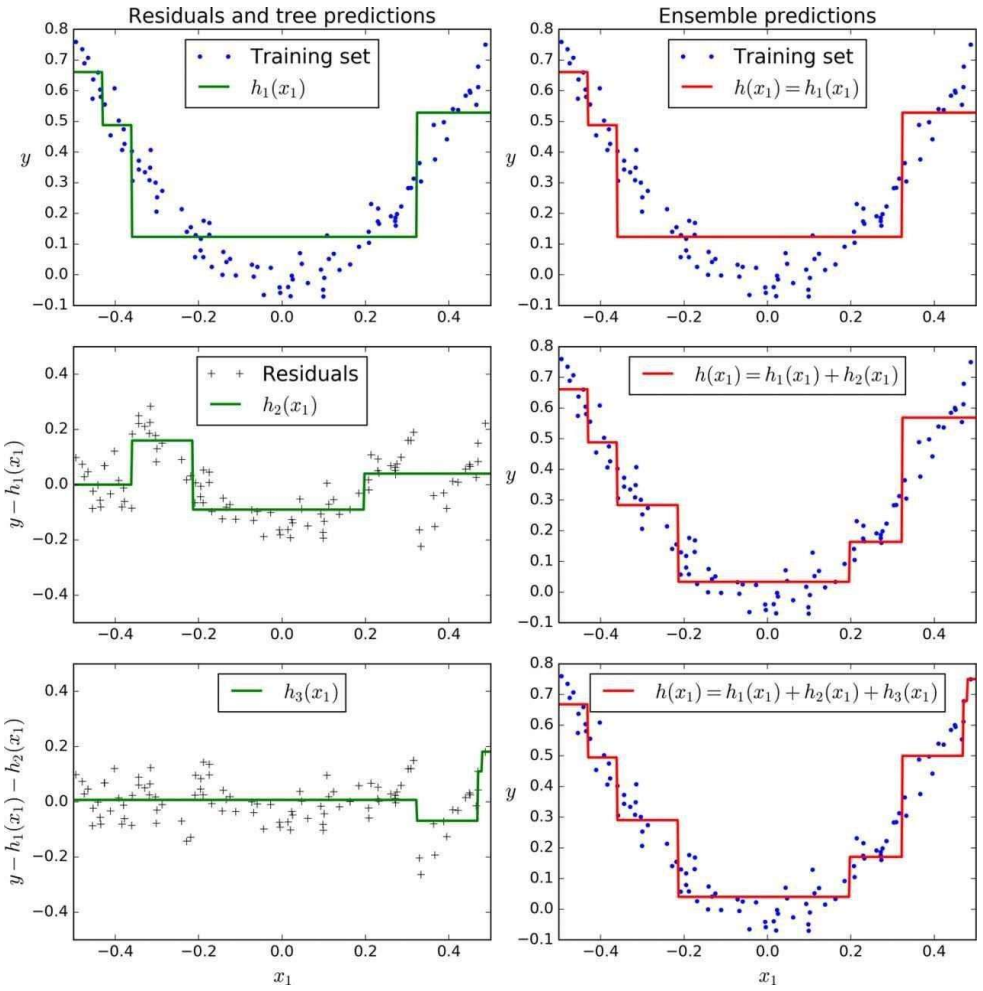
Bagging: Random Forest

- Differ from bagging in the number of predictors used at each split.
  - Instead of using all predictors, the algorithm is not allowed to consider most of the available predictors.
  - Only $m < M$ features are considered for each node for a split
  - m is a hyper-parameter to be tuned in the training process ($\sqrt{|M|}$ is a good value to start with)
- Why? Strong predictors cause most of the trees look similar.
- This technique decorrelate the trees.

Boosting

- Instead of fitting a large decision tree to the data, boosting tries to learn slowly from the errors.
  - Regression: The model fits a regression tree from the residuals
  - Classification: Gradient Boosting or AdaBoost to build additive tree models.
  - These small trees are added into the fitted function, slowly improving our estimate in areas where it does not perform well.
- Tuning parameters
  - Number of trees (B)
  - Shrinkage parameter (λ): rate at which boosting learns (typically: 0.01, 0.001)
  - The number of splits in each tree

Boosting


Residuals and tree predictions
Ensemble predictions
Training set
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
Residuals
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

Boosting: XGBoost

- Pros:
  - Extremely fast (parallel computation) and Highly efficient.
  - Versatile (Can be used for classification, regression or ranking).
  - Can be used to extract variable importance.
  - Do not require feature transformation (missing values imputation, scaling and normalization)

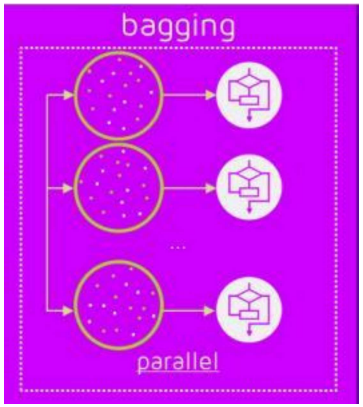
- Cons:
  - Leads to overfitting if hyperparameters are not tuned properly.

Boosting: XGBoost

Additional Resources:

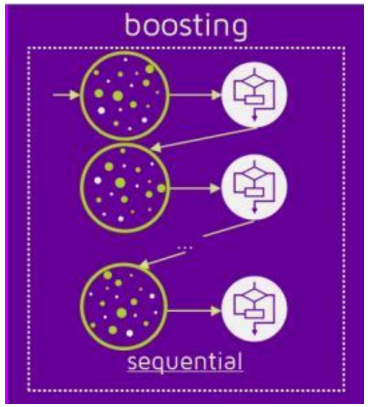
- Deeper explanation of the Gradient Boosting idea: link
- Talk of Tianqi Chen, the creator of the library: https://www.youtube.com/watch?v=Vly8xGnNiWs

Bagging vs. Boosting



Random Forest

- Independent Classifiers
- Reduce Variance
- Handles Overfitting

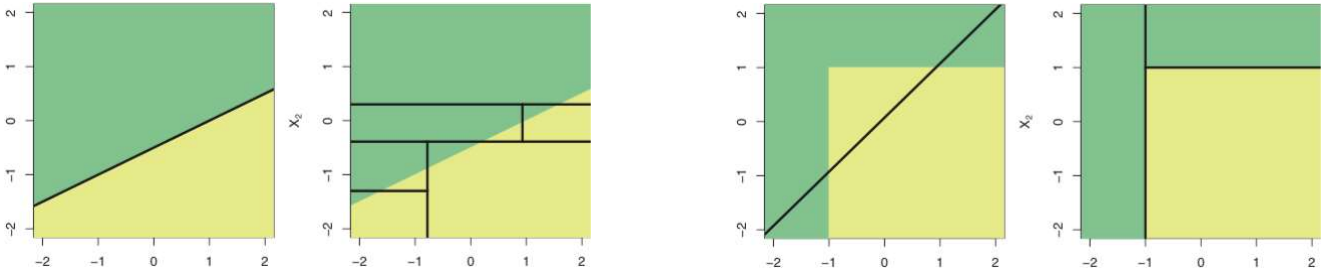


XGBoost

- Sequential Classifiers
- Reduce Bias
- Can (and do) overfit

When to use Decision Trees

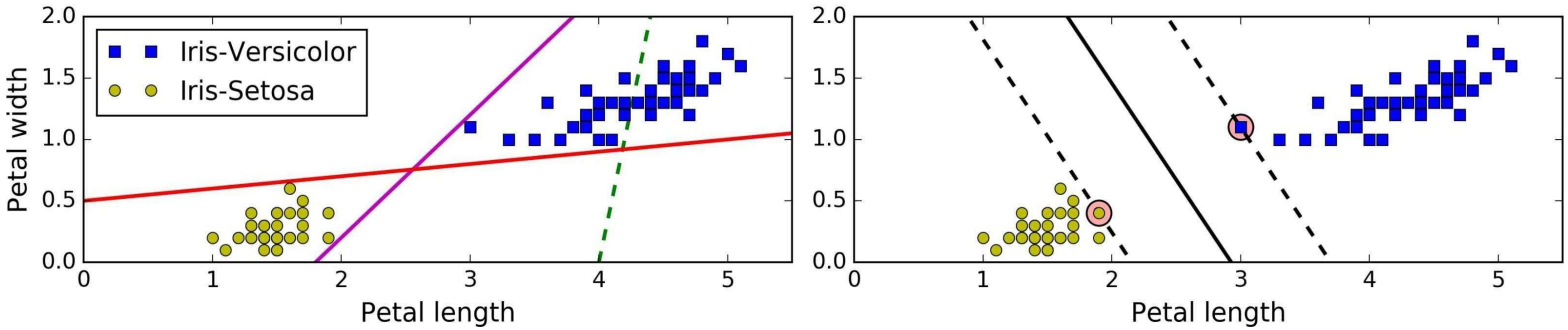
- Regression doesn't work



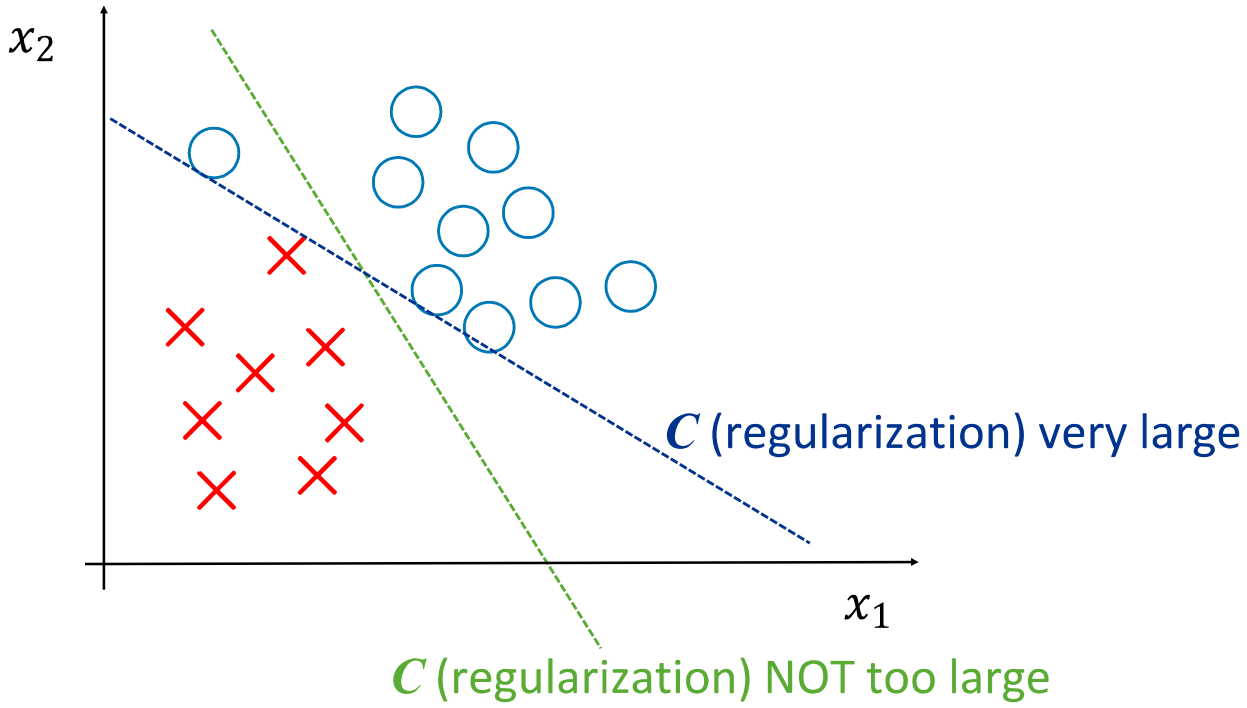
- Model intelligibility is important
- Problem does not depend on many features
- Linear combinations of features not critical
- Speed of learning is important
- Medium to large training sets

Support Vector Machines

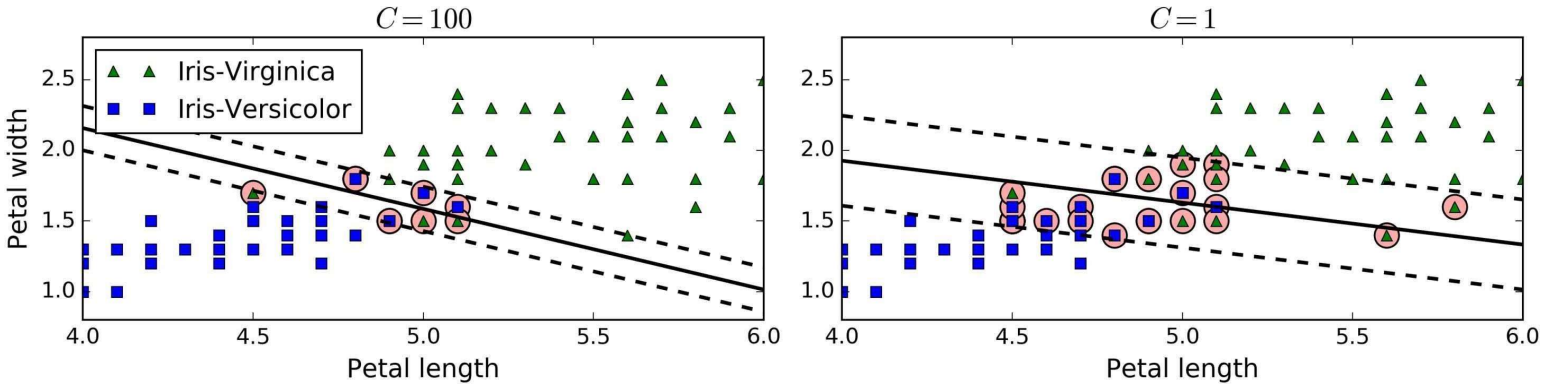
Large Margin Classifier



SVM Decision boundary



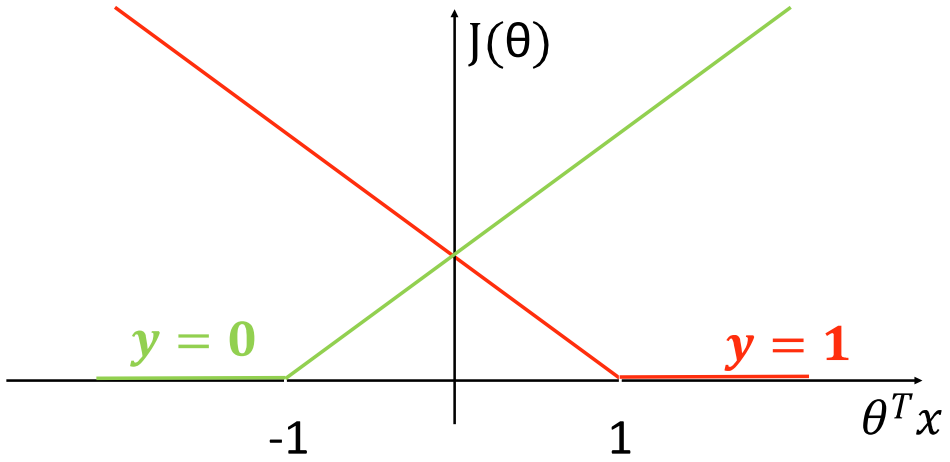
SVM Decision boundary



SVM: Algorithm

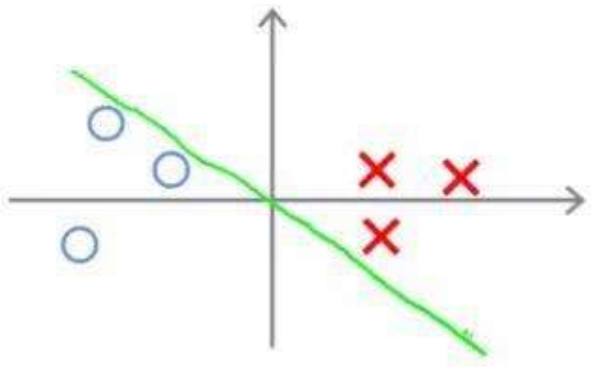
Large Margin Classifier

$$\min_{\theta} C\left[\sum_{i=1}^{m} y^{(i)} cost_1\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) cost_0\left(\theta^T x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$



SVM: Algorithm

Decision Boundary

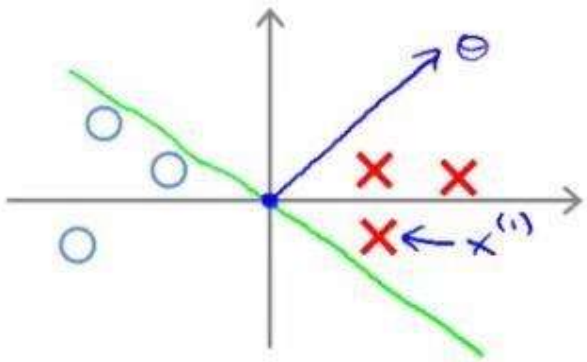


- SVM would not choose this line
- Decision boundary comes very close to examples

http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

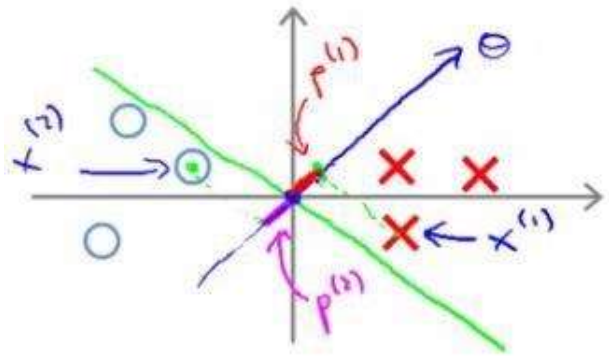
Decision Boundary



- Look at first example (x_1)

http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

Decision Boundary

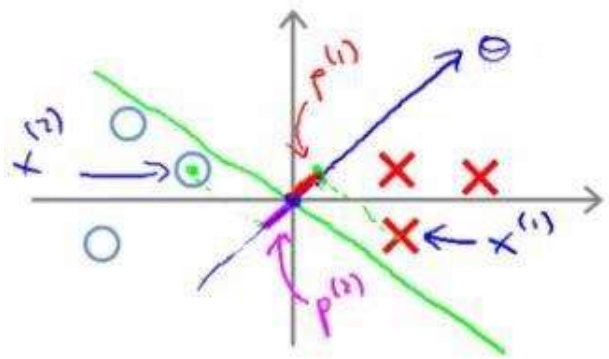


- Look at first example (x_1)
  - Project a line from x_1 on to the $\boldsymbol{\theta}$ vector (so it hits at 90 degrees)
  - The distance between the intersection and the origin is (p_1)
- Similarly, look at second example (x_2)
  - Project a line from x_2 into to the $\boldsymbol{\theta}$ vector
  - This is the magenta line, which will be negative (p_2)

http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

Decision Boundary

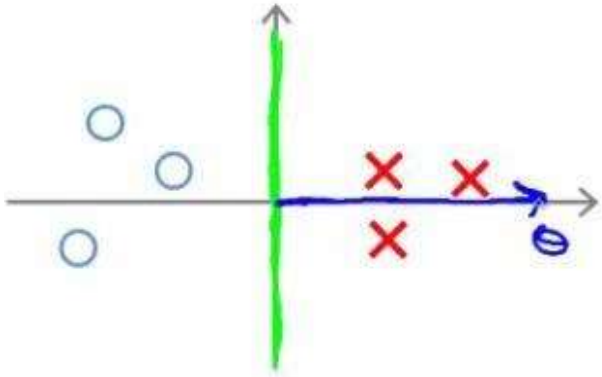


- p are small
- Look back at our optimization objective
- $p_1 * ||\theta||$ needs to be ≥1 for positive examples
  - If p is small Means that ||θ|| must be pretty large
- $p_2 * ||\theta||$ needs to be ≤−1 for negative examples
  - p2 is small ||θ|| must be a large number
- Why is this a problem?
- The optimization objective is trying to find a set of parameters where ||θ|| is small
  - This doesn't seem like a good direction because as p values get smaller ||θ|| must get larger to compensate
  - We should make p values larger which allows ||θ|| to become smaller

http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

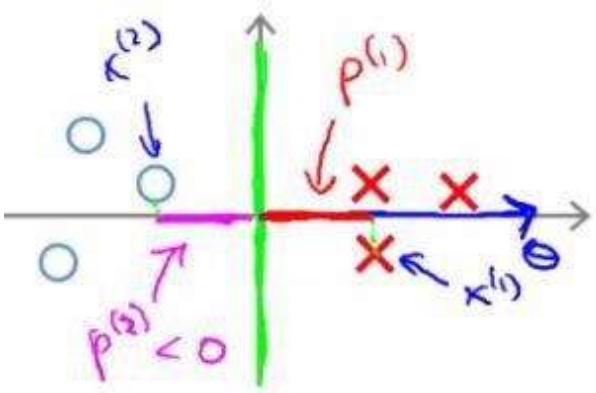
Decision Boundary



http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

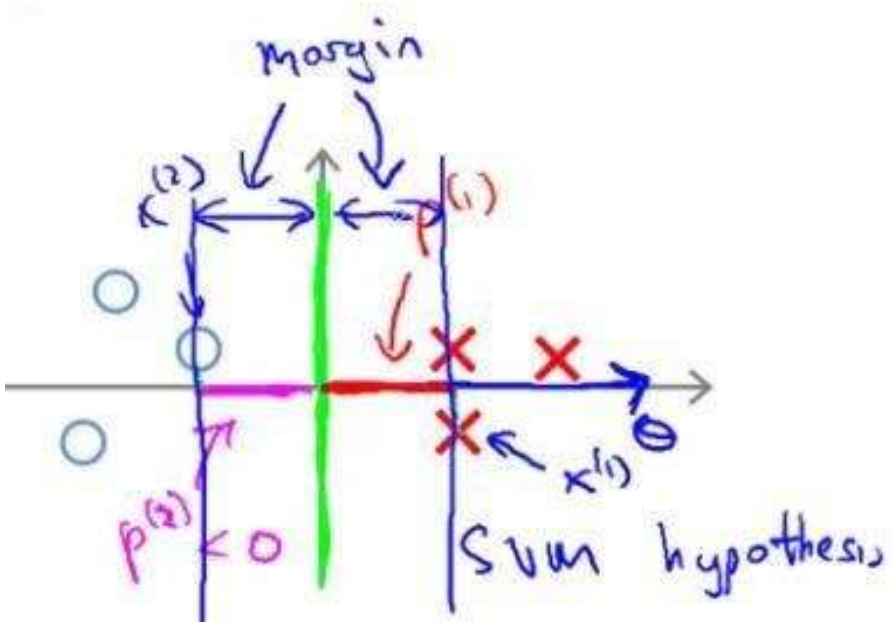
Decision Boundary



http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

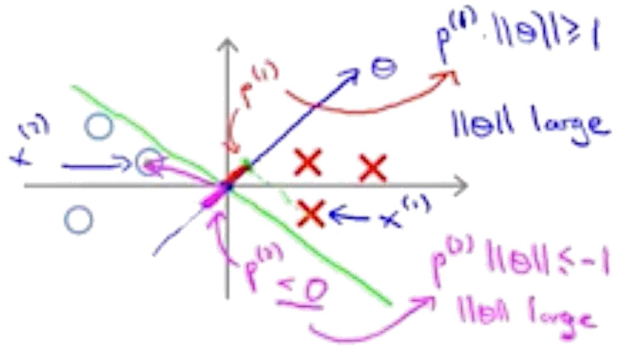
Decision Boundary

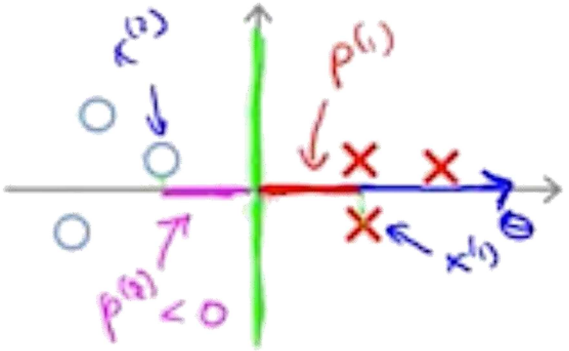


http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

SVM: Algorithm

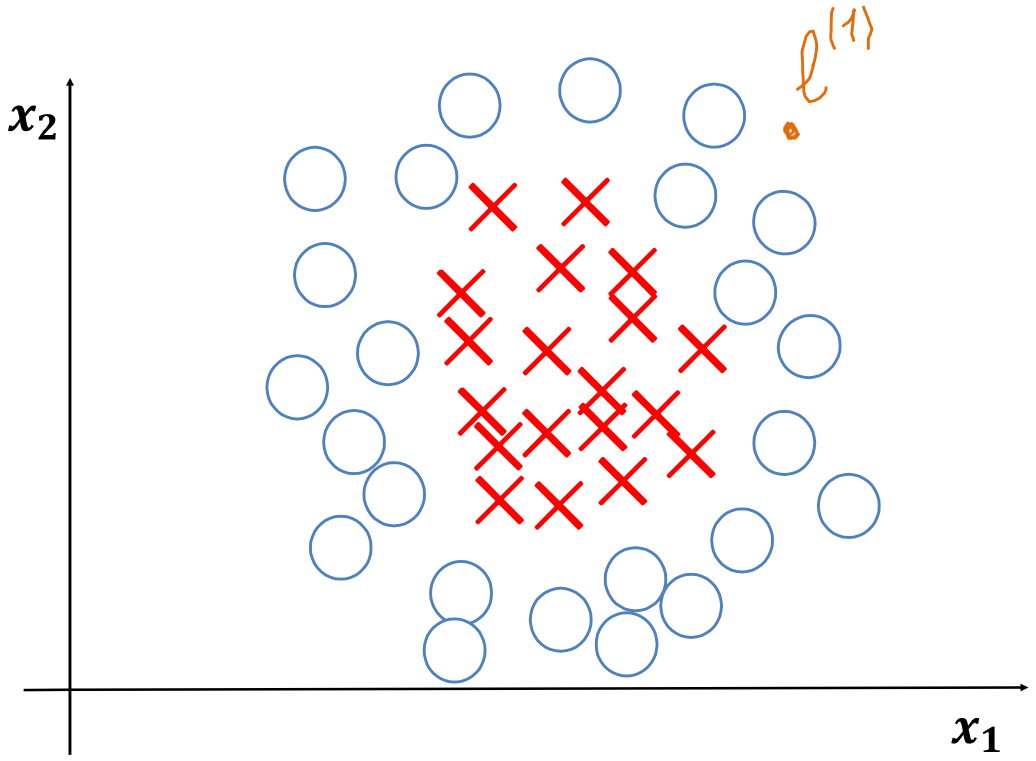
Decision Boundary

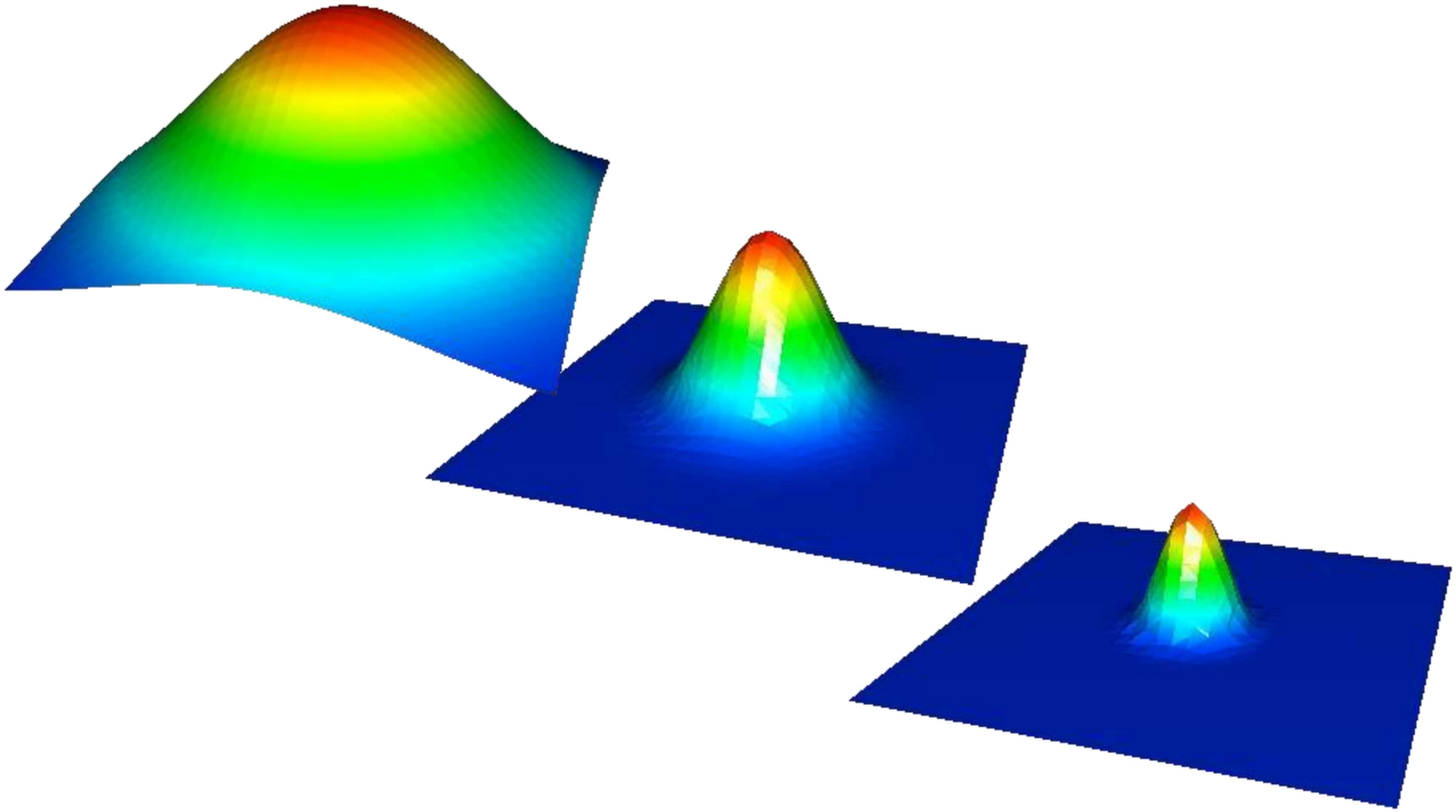




http://www.holehouse.org/mlclass/12_Support_Vector_Machines.html

Kernel intuition





Summary

SVMs Pros and Cons

SVM

Pros:

- Can handle large feature space
- Can handle non-linear feature interactions

Cons:

- Not very efficient with large number of observations
- It can be tricky to find appropriate kernel sometimes

Decision Trees

Pros:

- Intuitive Decision Rules
- Can handle non-linear features
- Consider variable interactions

Cons:

- Highly biased to training set [Random Forests solve this]
- No ranking score as direct result

Logistic Regression

Pros:

- Convenient probability scores
- Efficient implementations available
- Widespread industry comfort for logistic regression solutions

Cons:

- Doesn't perform well when feature space is too large
- Doesn't handle large number of categorical features/variables well
- Relies on transformations for non-linear features

EVALUATION METRICS

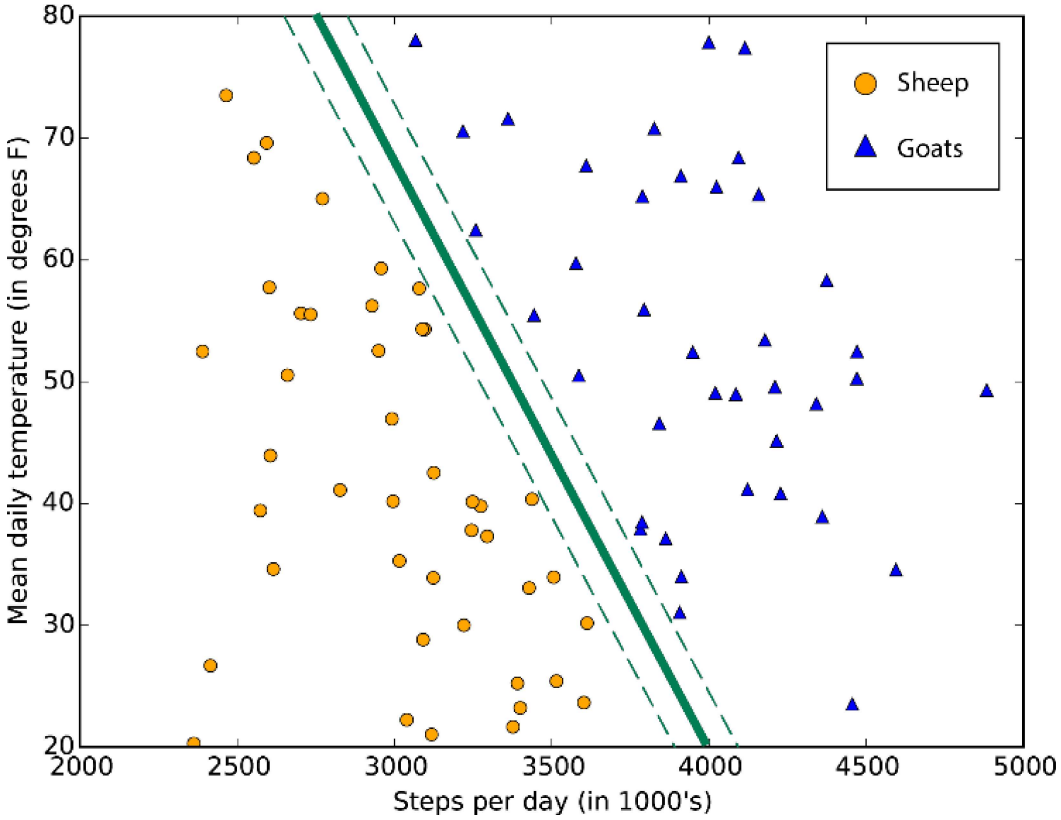
Definitions

- Model Evaluation: performance of a model, from a data science point of view, and being able to translate that into the business goals aimed at with its construction.
  - Different problems, different models, different performance measures.
- Model Validation: Measure how sure we're that the model will work in production (new, unseen data) as well as when it was trained.
  - Namely, do we have enough training data? Is it representative?

Methods for feature (and model) selections

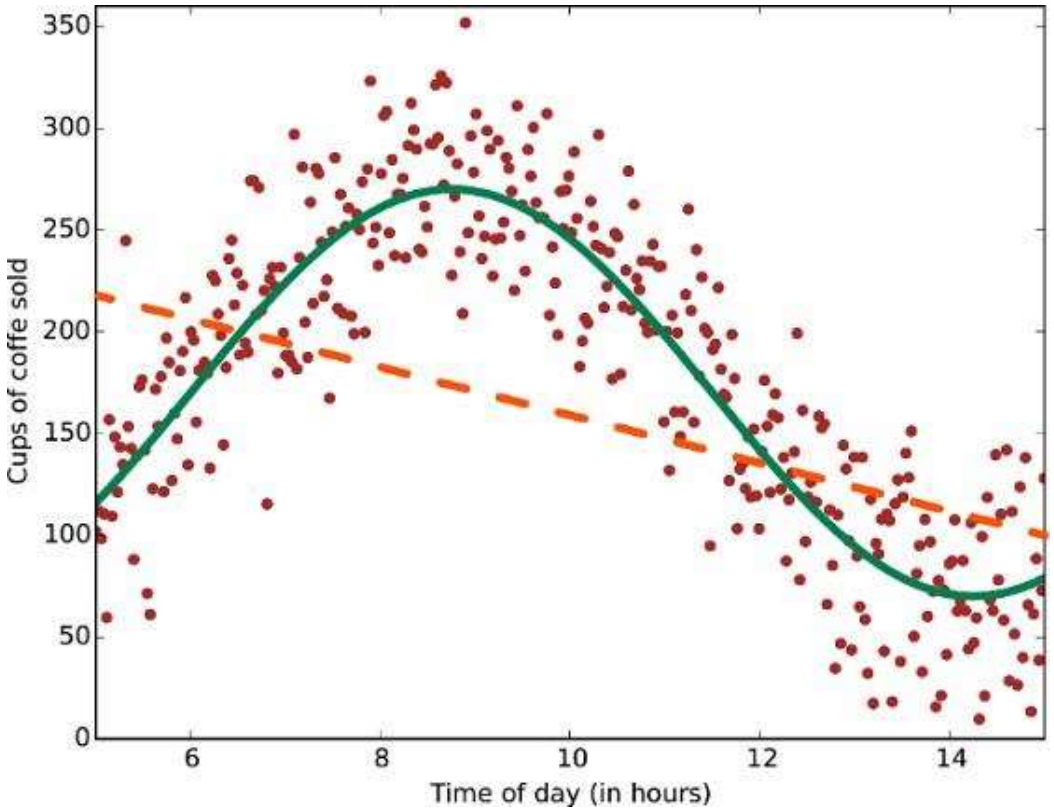
- Classification
- Scoring
- No-target methods

Classification Problems



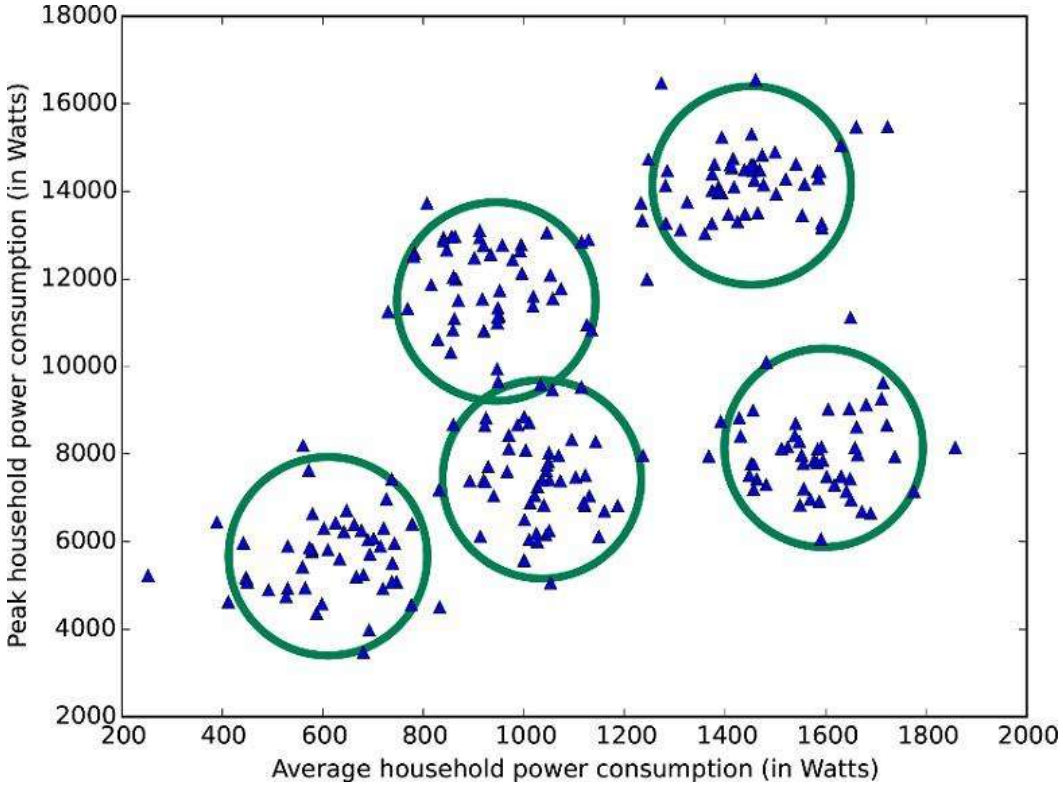
- Assign labels (categories) to untrained observations (objects)
- Can be multicategory (multinomial) or two-category (binomial)
  - We can always turn a binary classifier into a multicategory classifier.

Scoring Problems (Regression)



- Predict the output for a new set of values at the input or estimate the probability of an event.
  - Fraud Detection
  - Predict increase in sales for a particular marketing campaign.
  - Predict a value, given a known set of past observations.

No-Target Problems



- There's no outcome we want to predict
- Identify patterns or relationships in the data.
- Methods:
  - Clustering
    - Useful when we don't know what we're looking for.
    - Ambiguous.
  - Apriori algorithms
    - Recommendation systems, association rules (market basket analysis).
  - Nearest neighbor
    - Supervised classification method

Model Evaluation Metrics

- Classification
- Scoring
- Probability Estimation
- Clustering

Evaluation Classification Models

Confusion Matrix: Table of the counts at each combination of factors

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Accuracy

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- Most common measure of performance for classifiers
- Definition: #of items categorized correctly divided by the total nr. of items

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}$$

- Accuracy tell us how 'accurate' is our model predicting categories for unseen data and will also tell us what will be the expected error rate.
- Caveat: DO NOT use accuracy for unbalanced classes (i.e.: predict rare events).

Precision and Recall

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

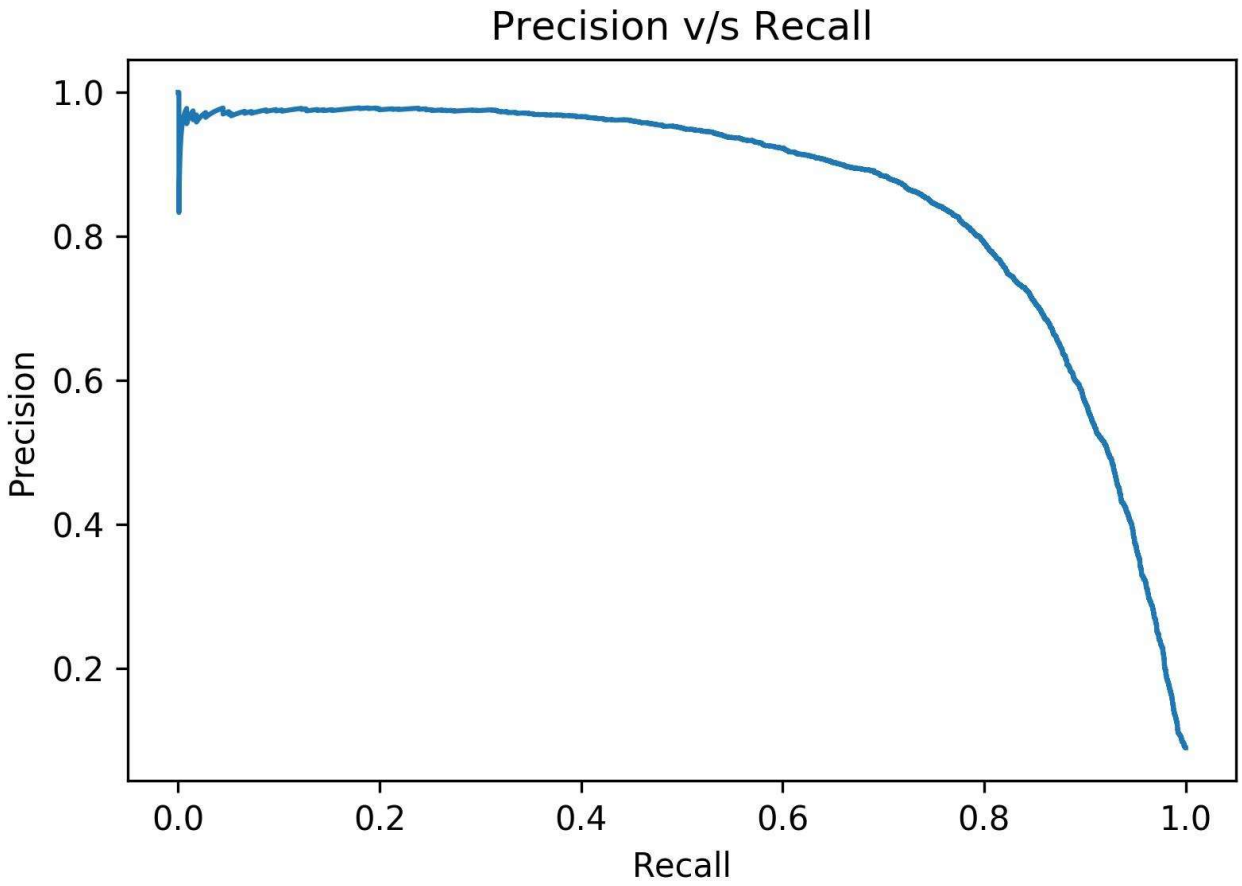
- **Precision**: fraction of predictions that actually are in the class
  - **Measure of confirmation:** how often my model predictions are correct

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

- **Recall:** fraction of observations in the class that actually are detected.
  - **Measure of utility:** how much my model finds what it must find

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

Precision-Recall Tradeoff



F-measure

- Combination of precision and recall

$$F = \frac{2 * precision * recall}{precision + recall}$$

- Any model compromising any of them will lower its F1 score.
- Evaluate models where we want to find a balance between precision and recall
- NOT a suitable method if there is a class imbalance.

Matthew's Correlation Coefficient

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- MCC measures the quality of a binary classifier.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(FN + TN)(FP + TN)(TP + FN)}}$$

- Range of -1 to 1:
  - -1 indicates a completely wrong binary classifier
  - +1 indicates a completely correct binary classifier
- It is a fair measure that can be used with unbalanced classes

Compare Accuracy, Recall and MCC

		Prediction	
		Positive	Negative
Actual	Positive	0	24
	Negative	FP	327

Accuracy: 0.932
MCC: 0.0

		Prediction	
		Positive	Negative
Actual	Positive	24	0
	Negative	327	0

Recall: 1.0
MCC: 0.0

		Prediction	
		Positive	Negative
Actual	Positive	24	0
	Negative	0	327

MCC: 1.0

Cohen's Kappa

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- Is your classifier is performing better than simply guessing at random according to the frequency of each class?

$$K = \frac{P(o) - P(e)}{1 - P(e)}$$

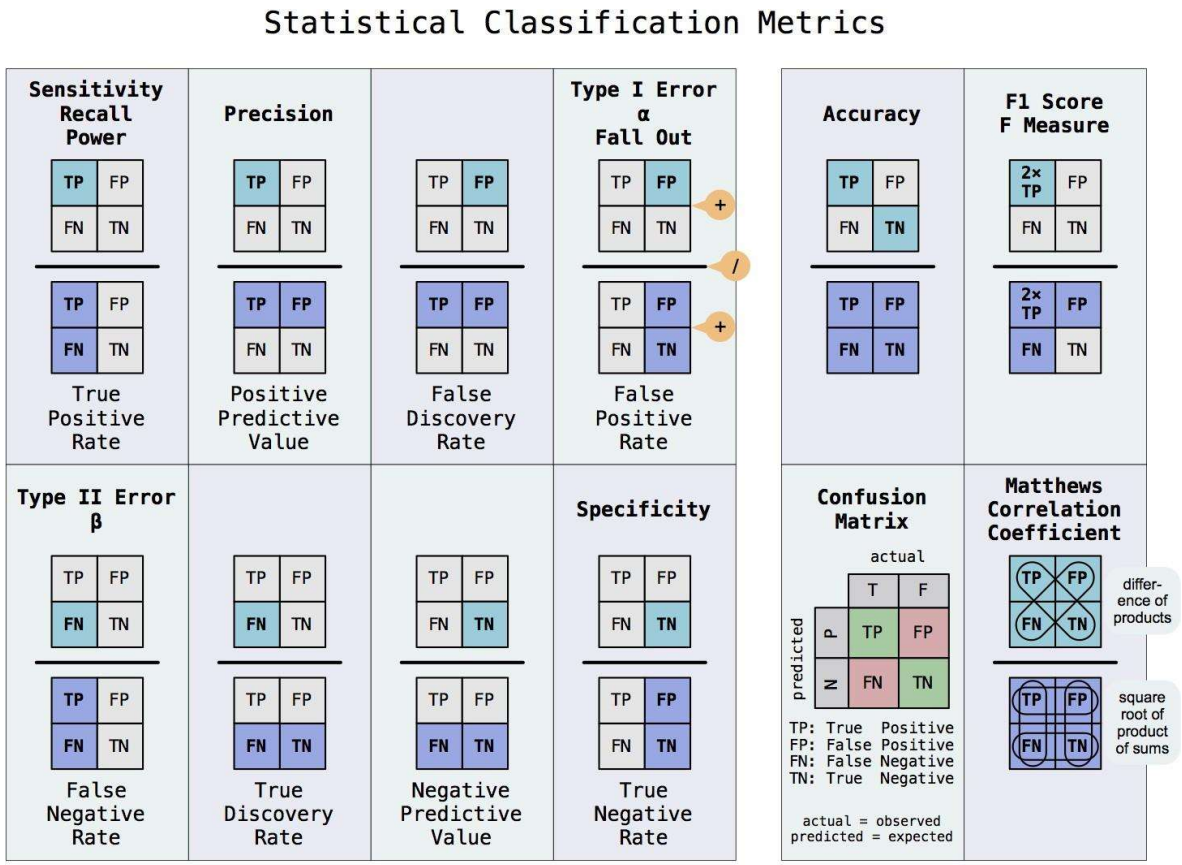
- Range: 0 to 1
- It can be used with unbalanced classes

What to do with imbalanced dataset?

- Collect more data
- Change Metric: use MCC, Kappa or the confusion matrix directly.
- Resample with bootstrapping
- Generate synthetic samples
- Change the algorithm (try with trees)
- Change approach: instead of classifying, maybe you should try anomaly detection.

Source: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Classification Metrics Summary



Application

Measure	Typical business need	Follow-up question
Accuracy	"We need most of our decisions to be correct."	"Can we tolerate being wrong 5% of the time? And do users see mistakes like spam marked as non-spam or non-spam marked as spam as being equivalent?"
Precision	"Most of what we marked as spam had darn well better be spam."	"That would guarantee that most of what is in the spam folder is in fact spam, but it isn't the best way to measure what fraction of the user's legitimate email is lost. We could cheat on this goal by sending all our users a bunch of easy-to-identify spam that we correctly identify. Maybe we really want good specificity."
Recall	"We want to cut down on the amount of spam a user sees by a factor of 10 (eliminate 90% of the spam)."	"If 10% of the spam gets through, will the user see mostly non-spam mail or mostly spam? Will this result in a good user experience?"
Sensitivity	"We have to cut a lot of spam, otherwise the user won't see a benefit."	"If we cut spam down to 1% of what it is now, would that be a good user experience?"
Specificity	"We must be at least *three nines* on legitimate email; the user must see at least 99.9% of their non-spam email."	"Will the user tolerate missing 0.1% of their legitimate email, and should we keep a spam folder the user can look at?"

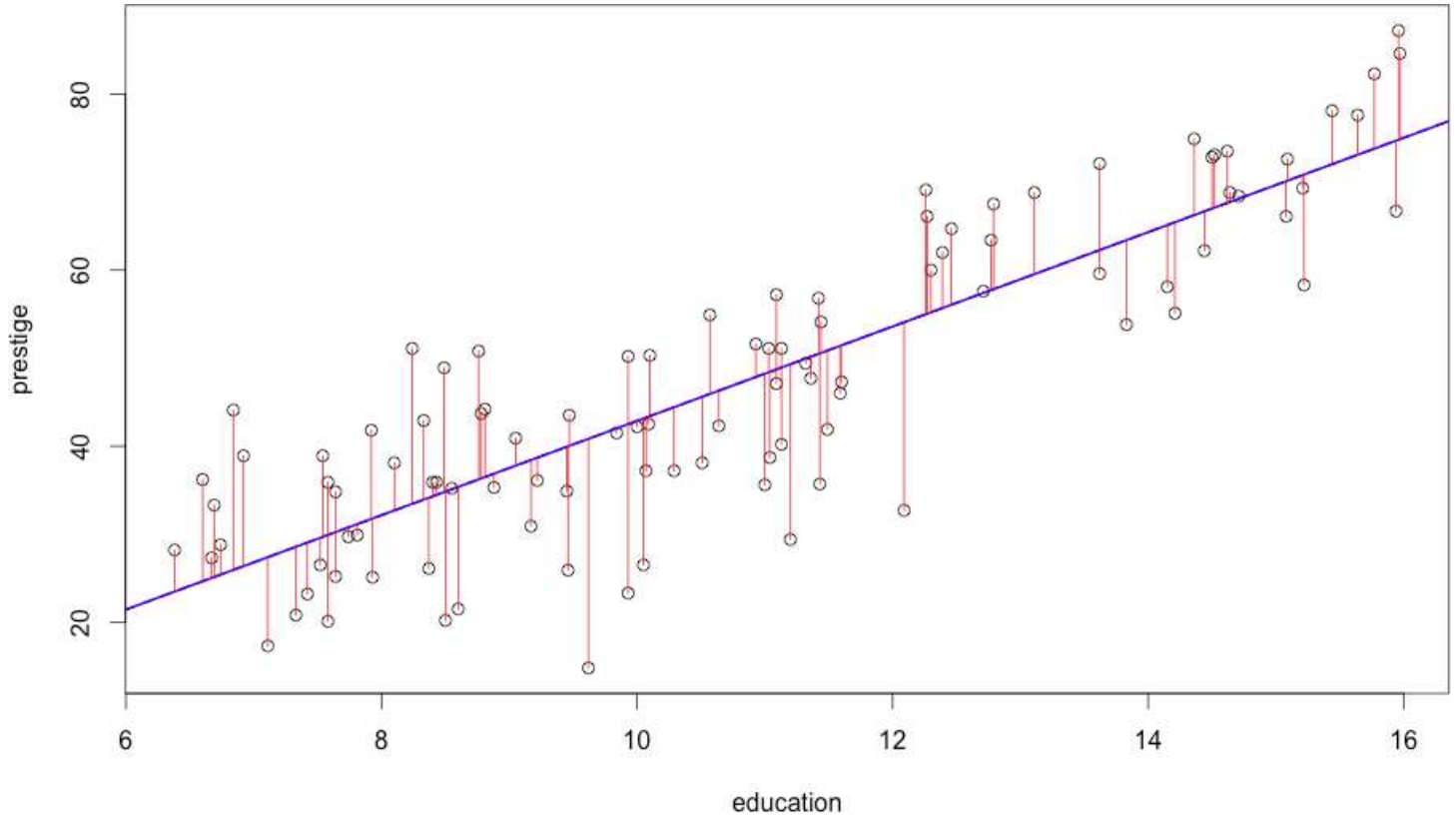
Source: Practical Data Science with R. pg. 98.

Intuition

Measure	Formula	Example	Intuition
Accuracy	$\frac{(TP+TN)}{(TP+FP+TN+FN)}$	0.9214	Overall, my model is predicting the correct class in 92,14% of the cases, or missing in 7.86% of the cases
Precision	$\frac{TP}{(TP+FP)}$	0.9187	In 8.13% of the cases I'm including false predictions of the positive class.
Recall	$\frac{TP}{(TP+FN)}$	0.8778	I'm missing 12.22% of the positive cases when predicting it with my model.
Specificity	$\frac{TN}{(TN+FP)}$	0.9496	I'm missing 5.04% of the negative cases when predicting it with my model.

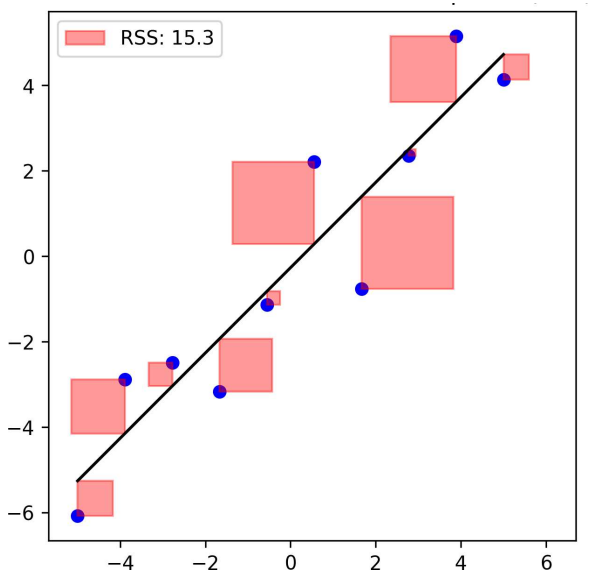
Evaluating Scoring Models

Difference between predictions and the actual outcomes (residuals)



Root Mean Square Error

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$



- The most common goodness of fit.
- Interpreted as a standard deviation of the prediction
- In the same units as y.

R-Squared (Coefficient of determination)

$$\boldsymbol{TSS} = \boldsymbol{SS_{tot}} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$\boldsymbol{RSS} = \boldsymbol{SS_{res}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2$$

$$R^2 = \frac{\boldsymbol{TSS} - \boldsymbol{RSS}}{\boldsymbol{TSS}} = 1 - \frac{\boldsymbol{RSS}}{\boldsymbol{TSS}} = 1 - \frac{\boldsymbol{SS_{res}}}{\boldsymbol{SS_{tot}}}$$

- What fraction of the y variation is explained (can be predicted) by the model.
- Best possible value =1.
  Near 0, bad.
- Difficult to explain to the business.

Absolute Error

$$Abs.Error = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

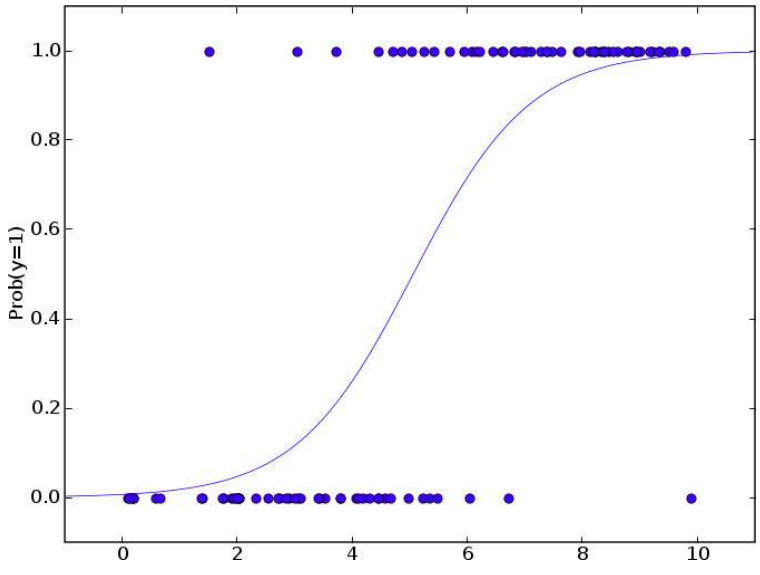
$$Mean\ Abs.Error = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

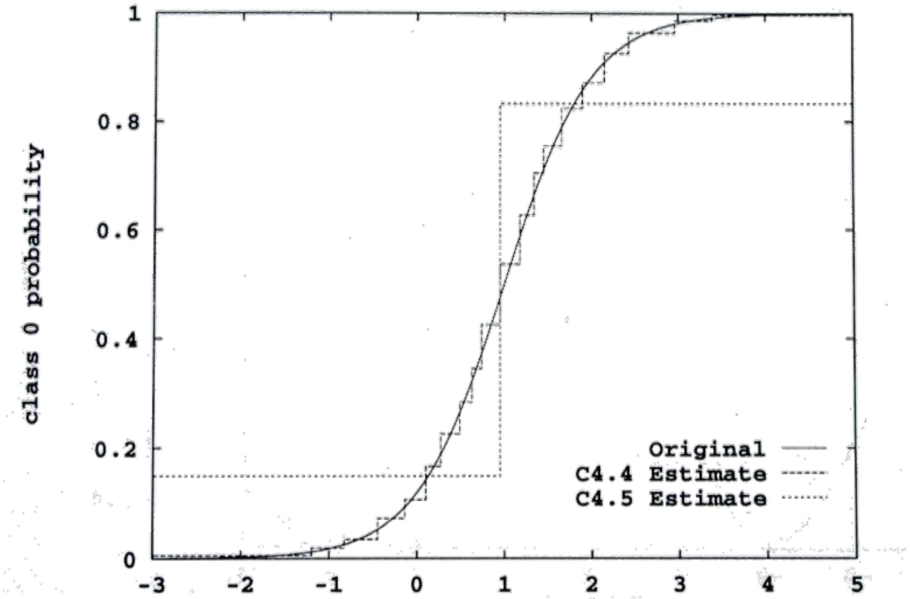
$$Abs.Error = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|}$$

- These measures are OK to be reported, but…
- …don't make them the project goal or to attempt to optimize them.
- Absolute error tend not to "get aggregates right" or "roll up reasonably" as most of the squared errors do.

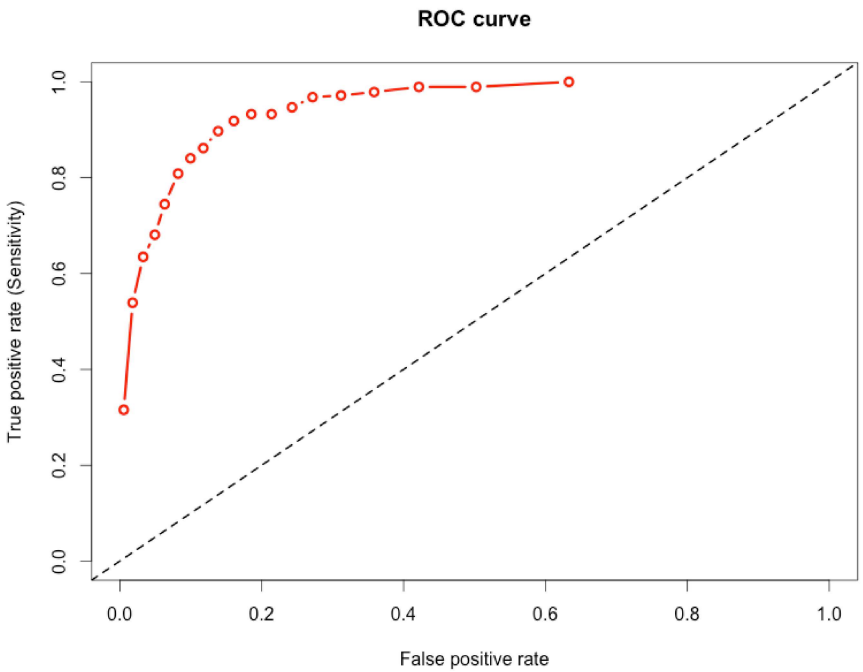
Evaluating Probability Models

Probability Models return a class where each observation belongs, together with an estimated probability (confidence) of the item being in that class.





ROC



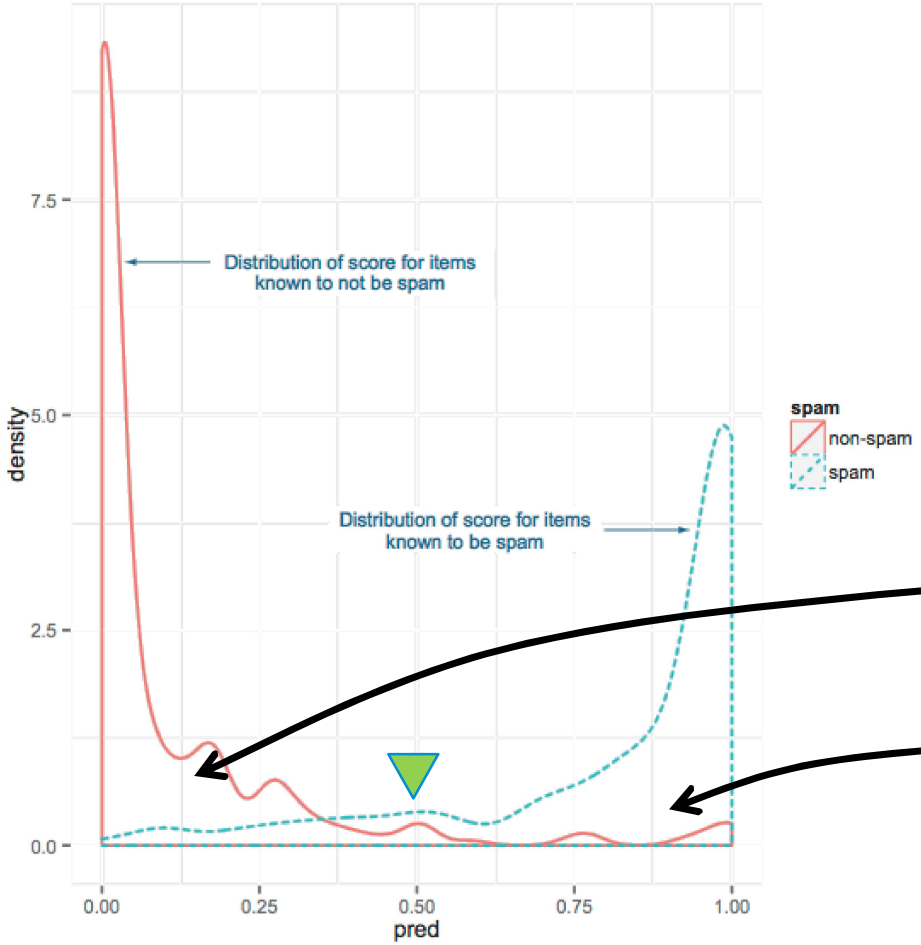
- a.k.a. The True Operating Characteristic Curve.

- How to:
  - Compute TRUE positive rate, and FALSE positive rate for a RANGE of score thresholds.
  - Compute the Area Under the Curve (AUC) for the previous values

ROC (remarks)

- The AUC does not have as straightforward a business intuition. Difficult to interpret.
- Single value summary expectation of the classifier performance.
  - If you're about to use a single value (plot), use Precision and Recall.
  - F1 is also a good single-value measure of the classifier quality.
  - In case of doubt, combine them to take decisions.
- Your decision, from the threshold chosen in the ROC curve, are irrelevant to the design of other classifiers on the same problem.
- Double density plots are easier to explain.

Double Density Plots



If I set the threshold here (▽), I will misclassify:

- All the items known to be spam (blue) before that point.
- All the items known NOT to be spam (red) beyond that point.

Evaluating Clustering Models (1/2)

- Clustering implies trying with different values of 'k'. The problem is to decide which one is the best.
- Useful checks:
  - Clusters with very few samples (individual samples)
  - Clusters with too many samples (nothing in common among the samples)
- Compactness: compare the distance between items in the same cluster to the distance between items from different clusters.
  - Mean intra-cluster must be smaller than inter-cluster.

Evaluating Clustering Models (2/2)

- As classification
  - Do not use the labels for clustering, and measure the overall performance assigning observations with the same label to the same cluster.
- Silhouette
  - Silhouette measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
  - Ranges from -1 to 1
- Try different algorithms
  - K-Medians, K-Medoids, Mixture of Gaussians, Density based clustering, etc.

Model Validation

- Model Problems
- Train-Test Splitting
- Cross Validation
- Significance Testing

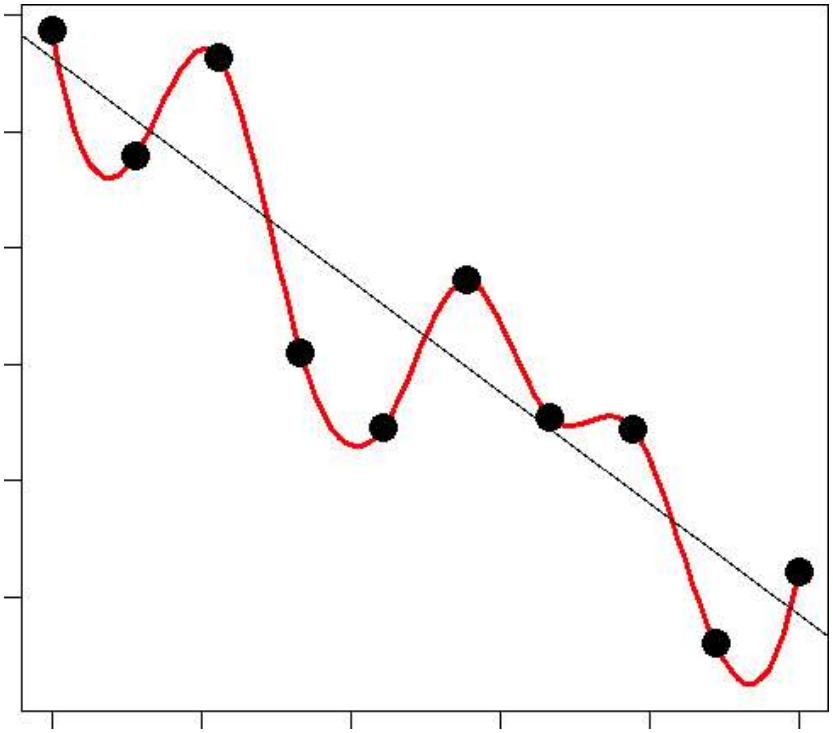
Model Problems

- Bias
  - Systematic error in the model
- Variance
  - Oversensitivity of the model to small variations in the data
- Overfit
  - Features of the model that arise from relations that are in the training data, but not representative of the general population.
- Nonsignificance
  - A model built on the assumption of an important relation when in fact the relation may not hold in the general population.

The Challenge

High Variance

Drawing a curve through every training observation

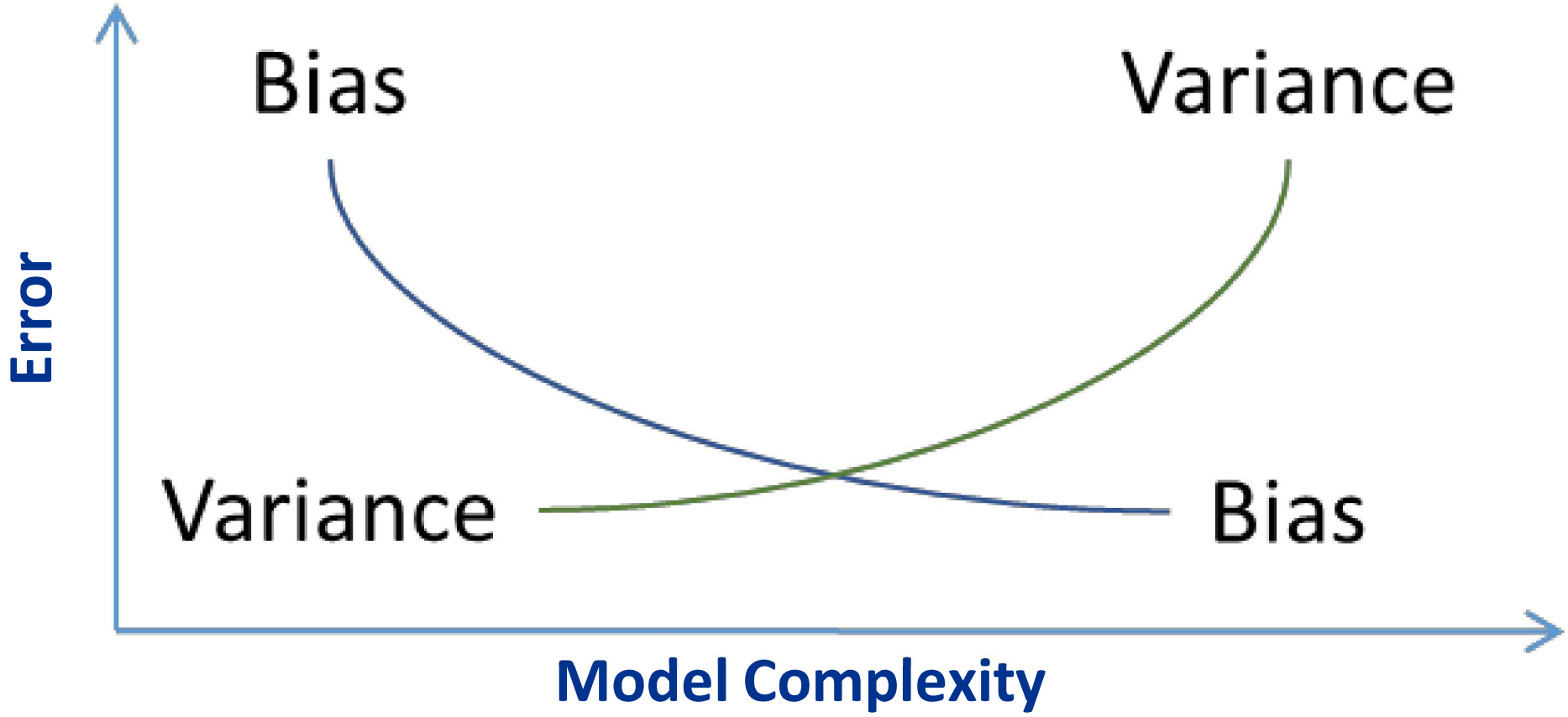


High Bias

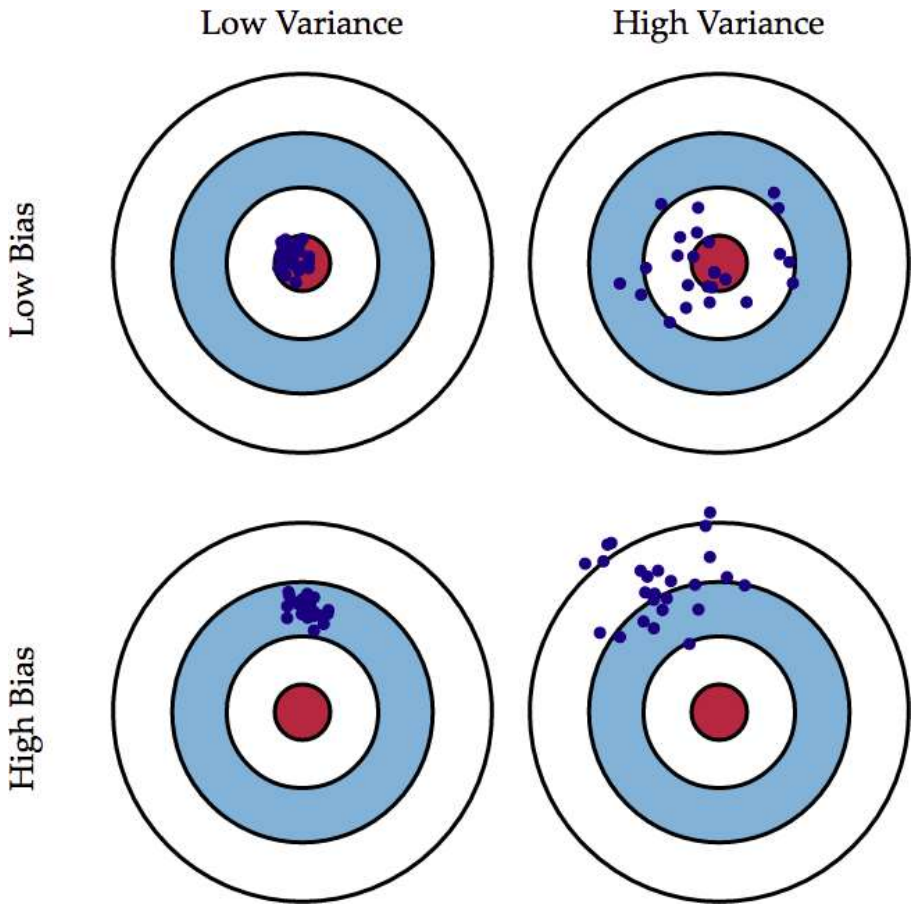
Fitting a straight horizontal line to the data

The challenge is finding a model with low variance and bias

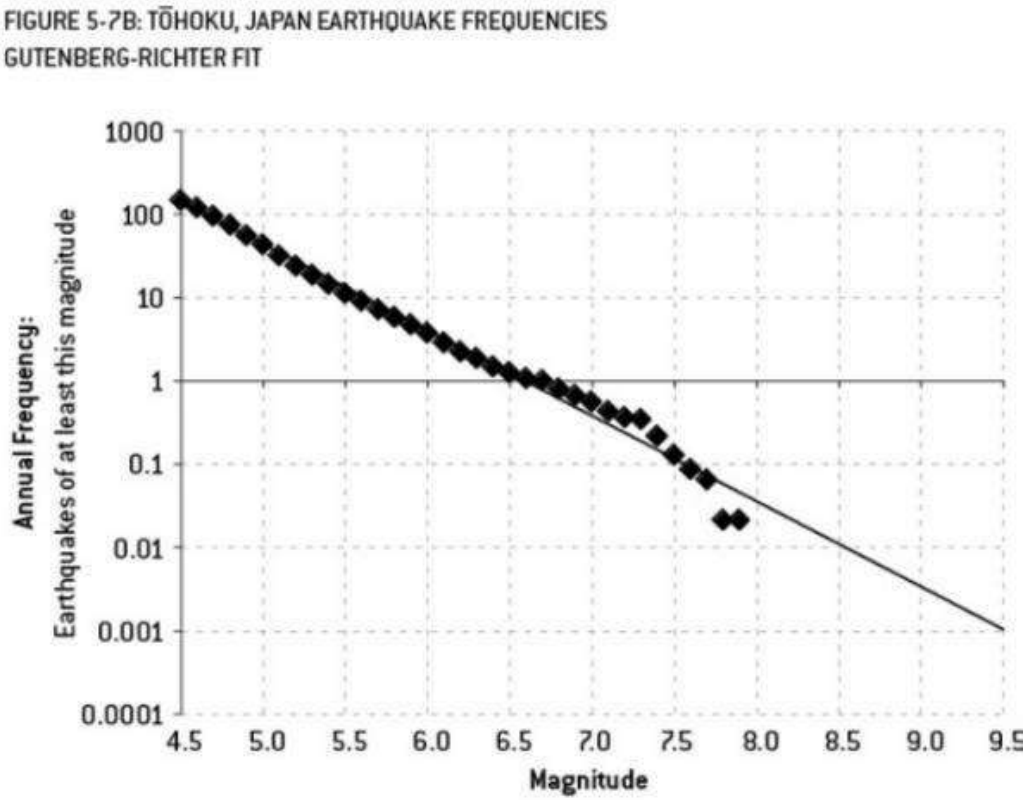
Bias-Variance Trade-Off



Bias-Variance Trade-Off

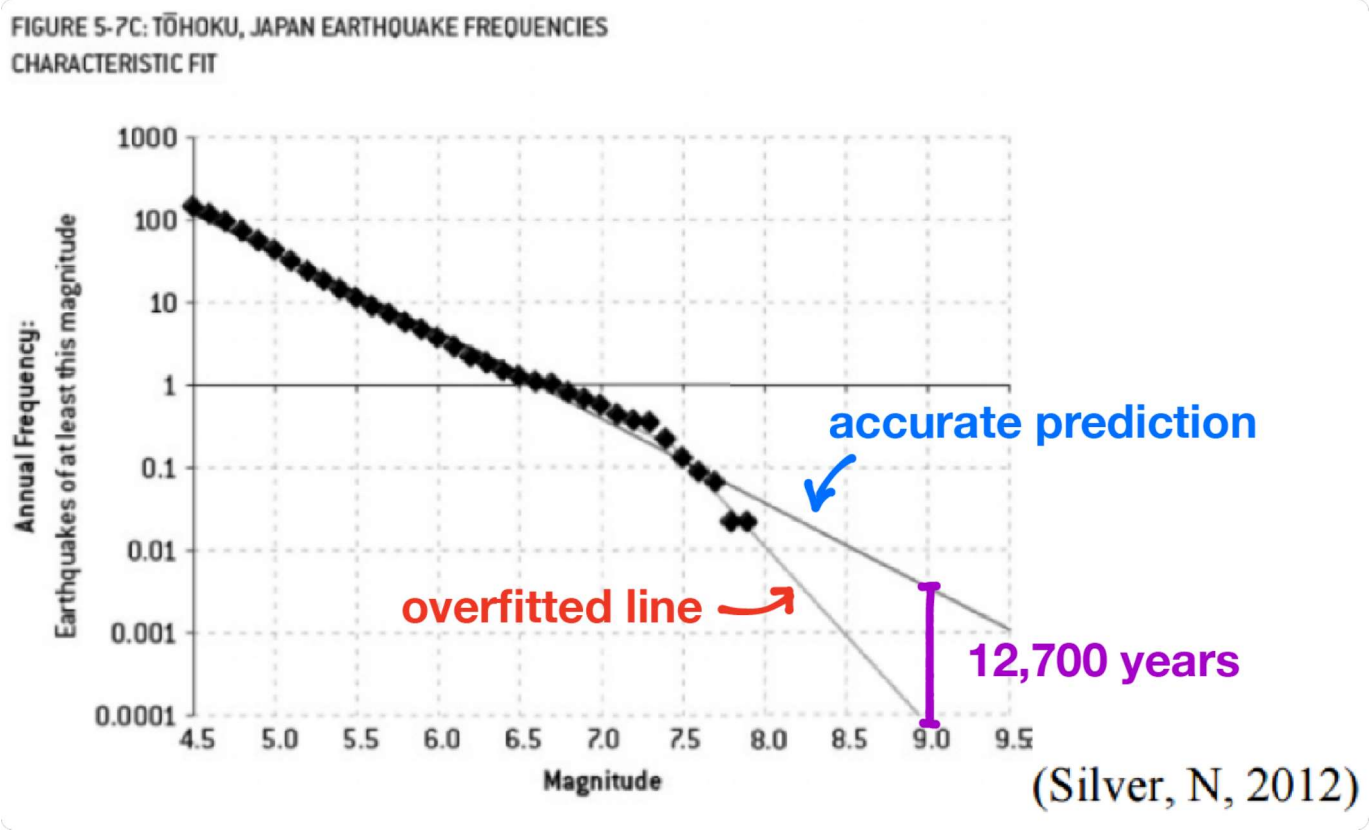


Bias-Variance Trade-Off



https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/

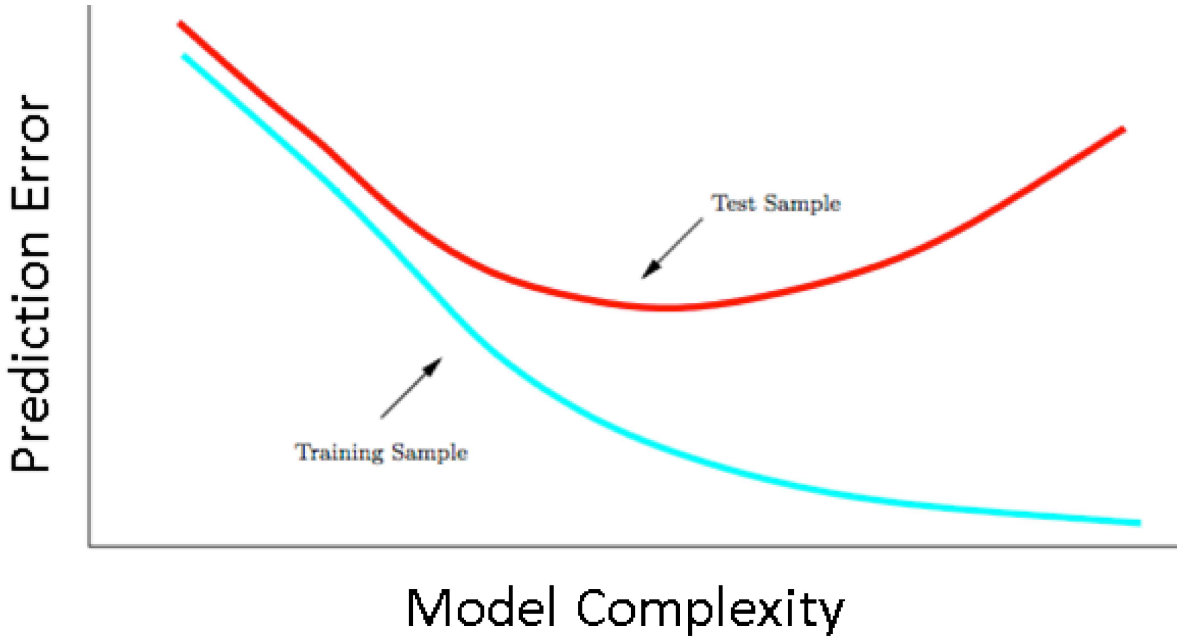
Bias-Variance Trade-Off



https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/

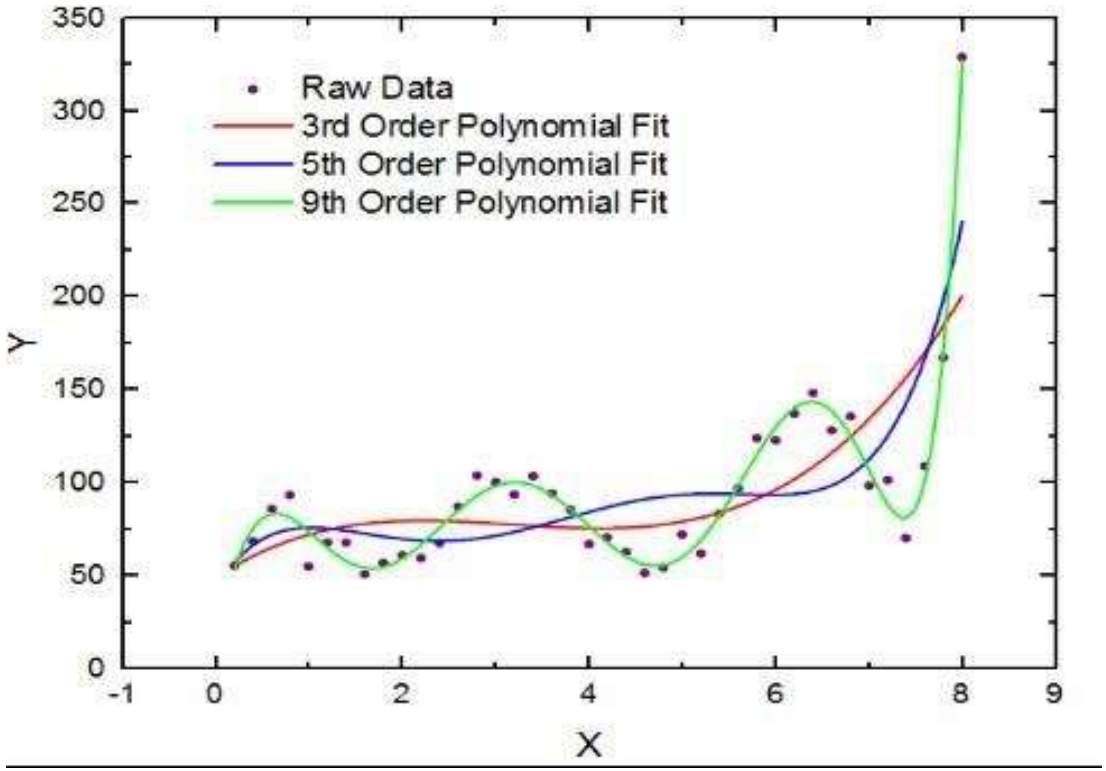
Errors

Training error can dramatically underestimate the Test Error



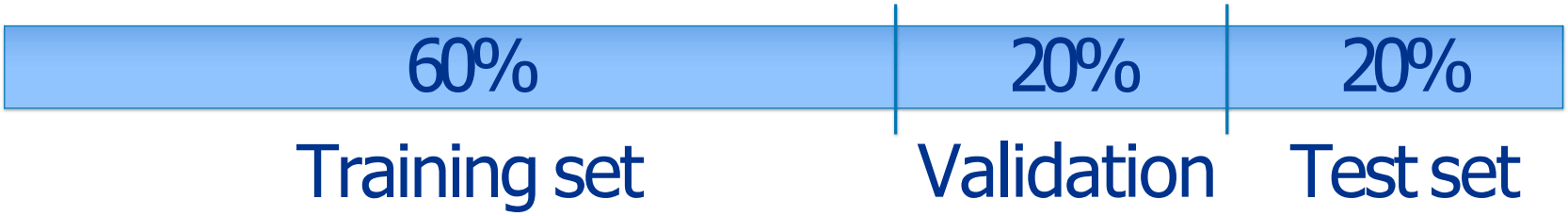
Errors

Splitting datasets into training and test is NOT enough!



Validation

- Validation set to avoid optimistic error estimates.



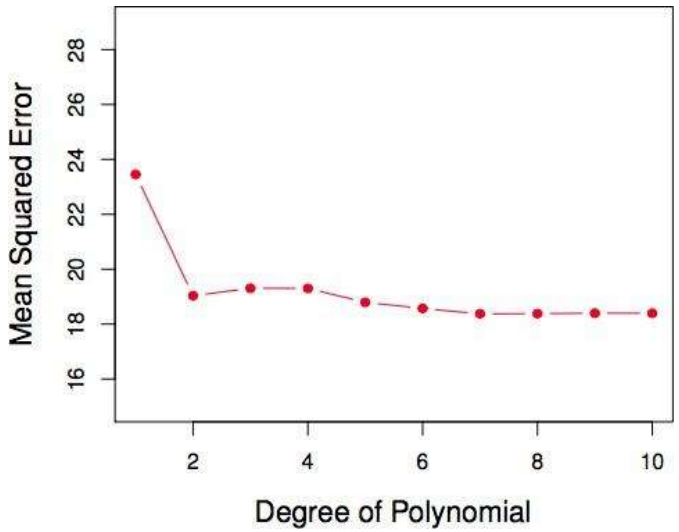
- Now, the approach is:

1. Fit a model to the training set for each polynomial degree (d).
2. Find the polynomial degree (d) with the least error using the Cross Validation set.
3. Estimate the generalization error using the test set

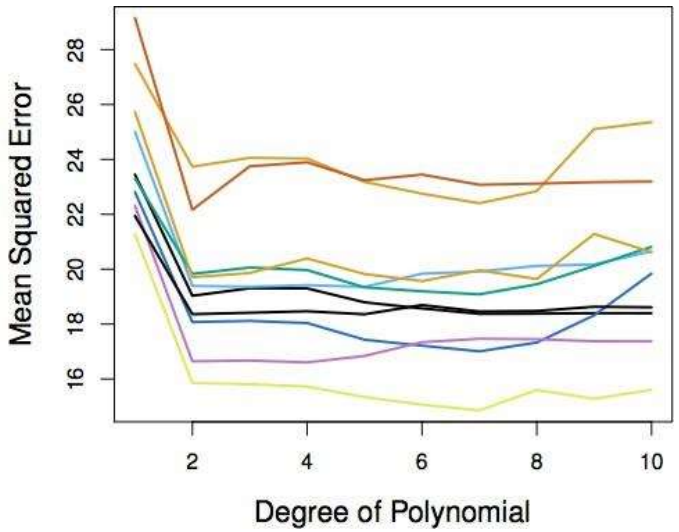
Single vs. Multiple splits

Validation set error may tend to overestimate the test error for the model fit on the entire data set.

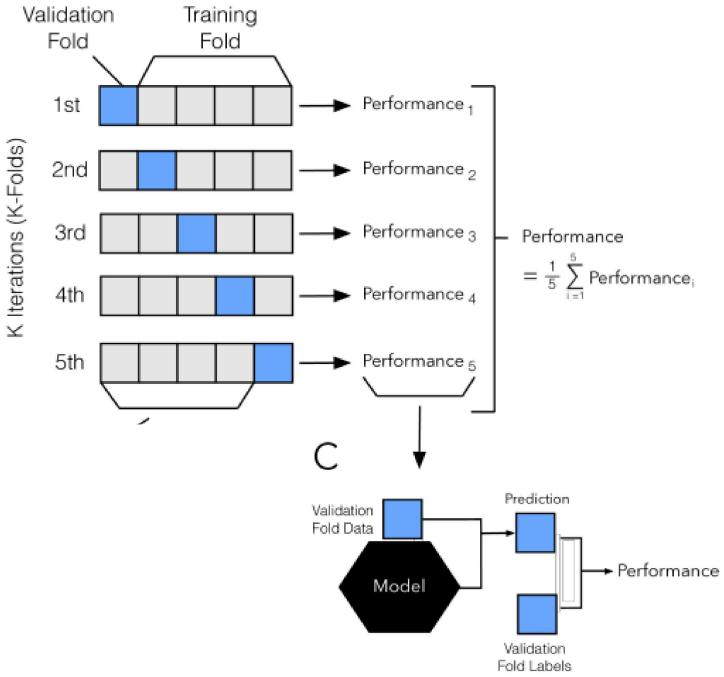
Single Split

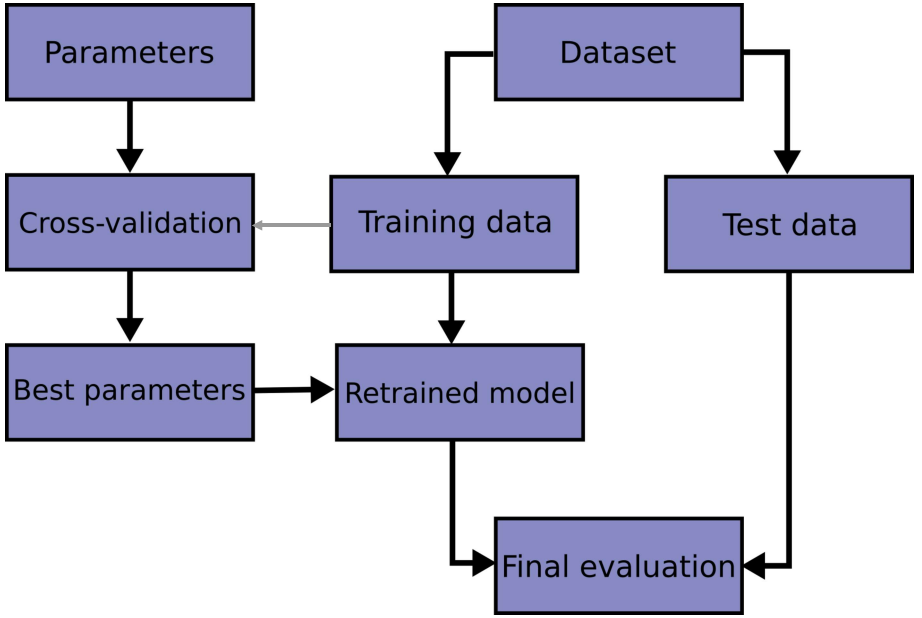


Multiple Splits



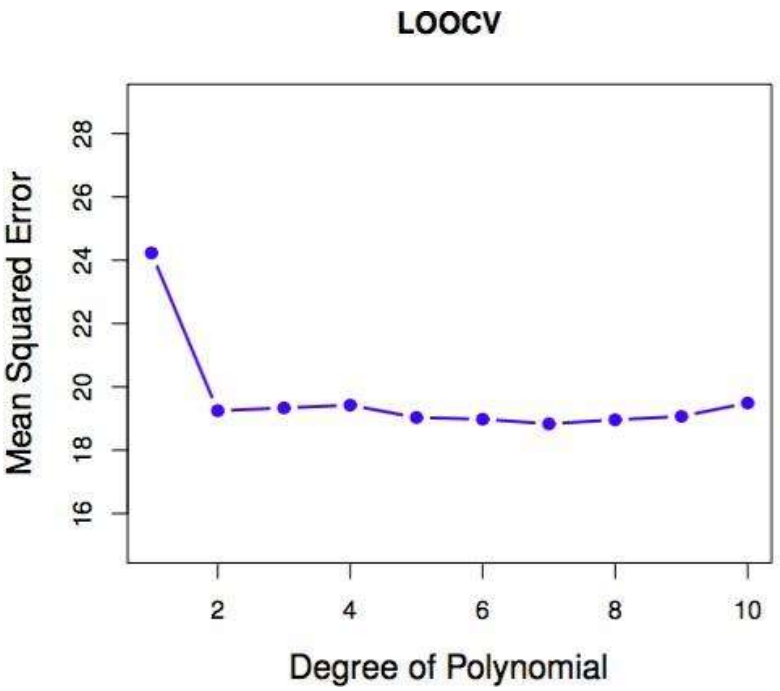
Cross Validation resampling

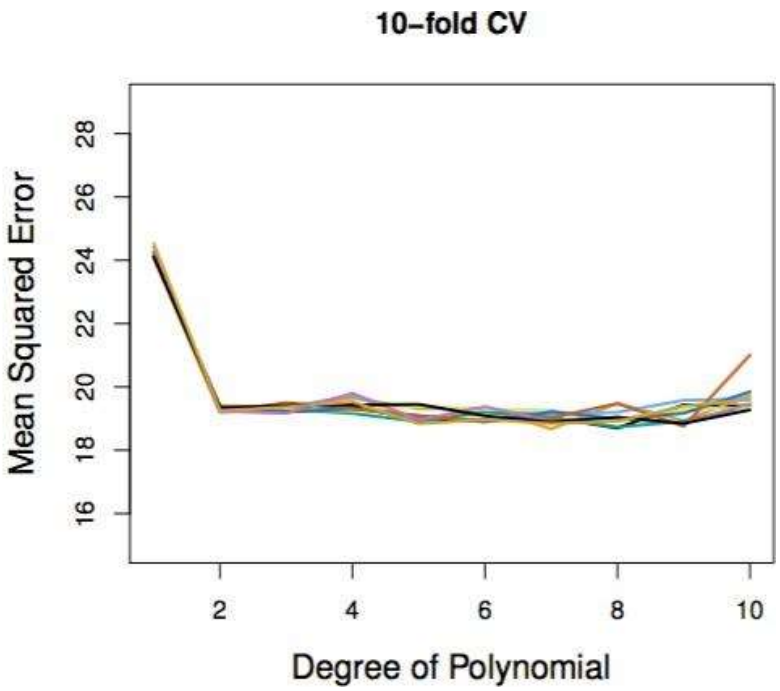




https://scikit-learn.org/stable/modules/cross_validation.html

LOOCV & K-Fold

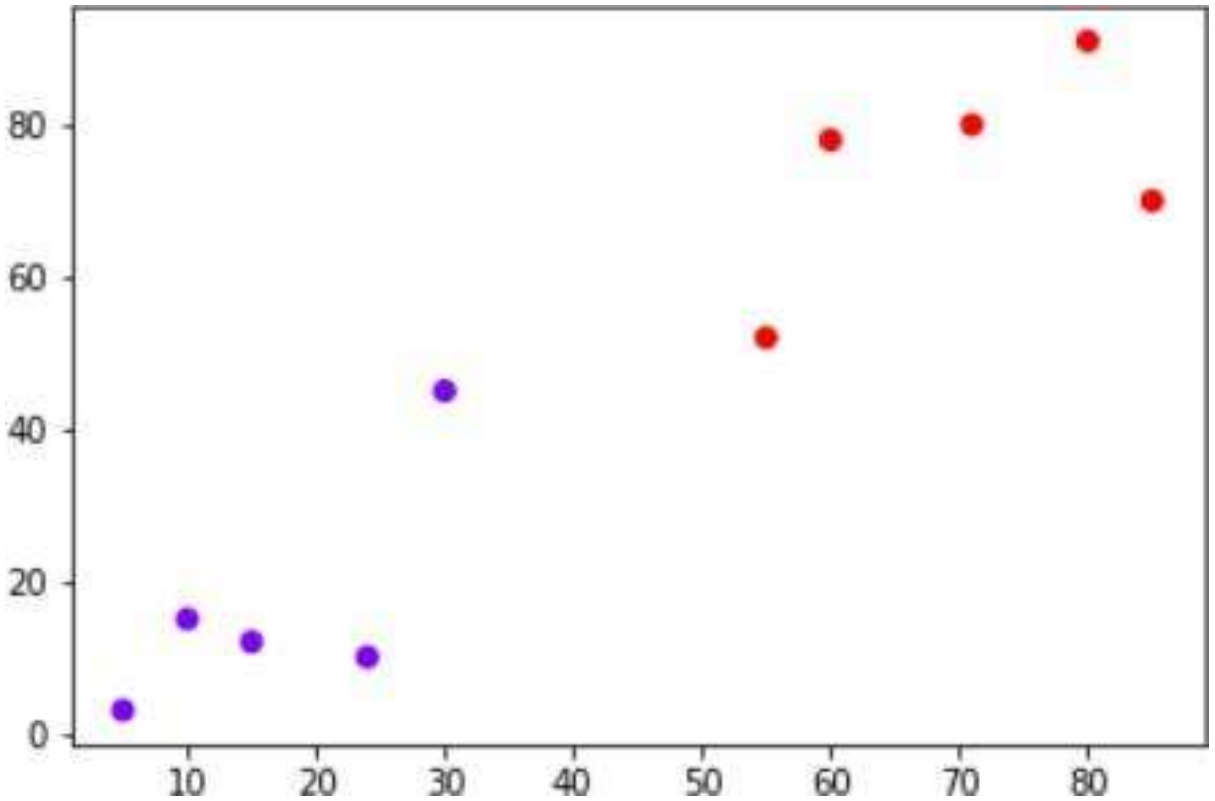




Each 10-fold is run nine times, each with a different random split of the data into 10 parts.

Nearest Neighbor Methods

"Instance based" classification



"Instance based" classification

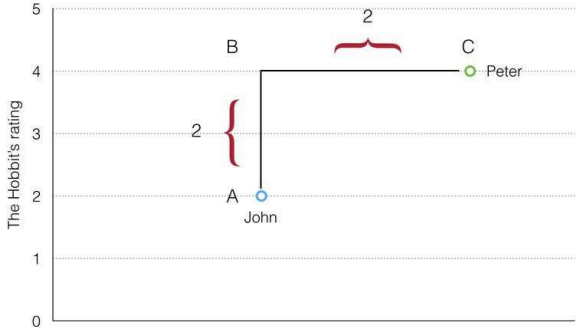
- Idea: Find an existing instance that is "most similar" to the new instance, that we're trying to classify.
  - "Most similar" is a measure of the distance or similarity between two instances
- To classify, select the class/label of (pick one):
  - The most similar one
  - The most frequent among the nearest k-neighbors
  - A weighted majority

Distance/Similarity

- The term "distance" is really important.
- Distance-based ML methods find items that are similar based on that "distance" definition. This is, items that are close to each other.
- Let's see how it works

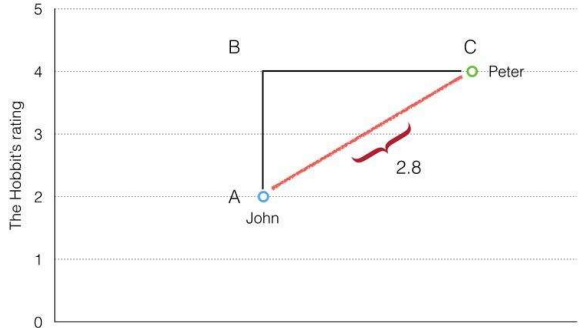
Manhattan Distance

$$d(x,y) = \sum_{k=1}^{n} | x_k - y_k |$$

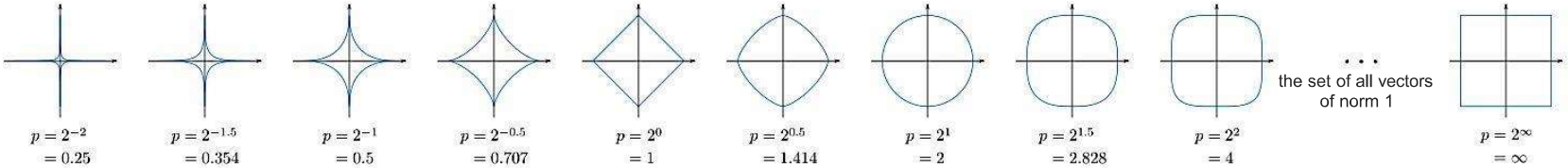


Euclidean Distance

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

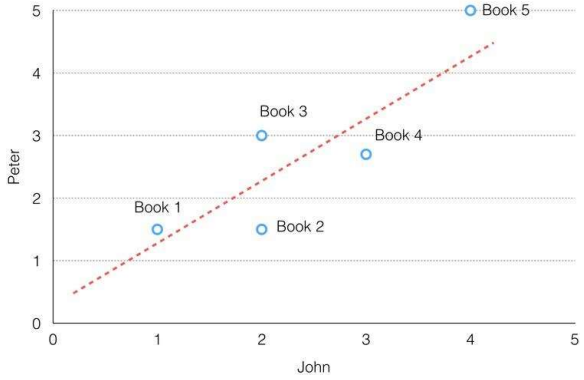


Minkowsky



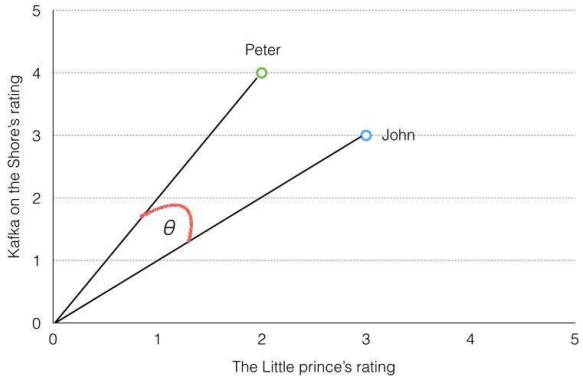
Pearson Correlation Score

$$Pearson(x,y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}}$$

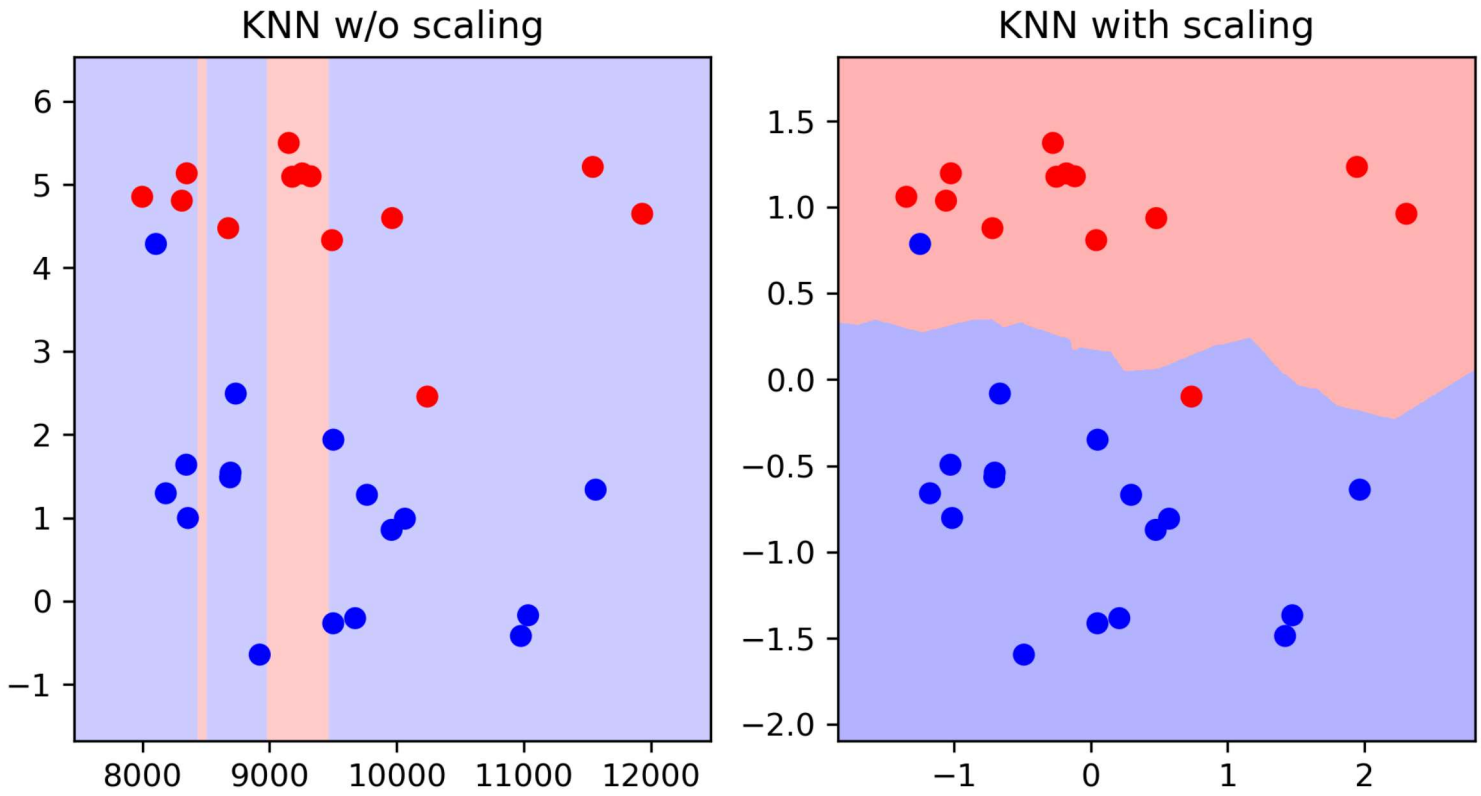


Cosine Distance

$$cos(\theta) = \frac{x \cdot y}{||x|| * ||y||}$$



Notes on Scaling



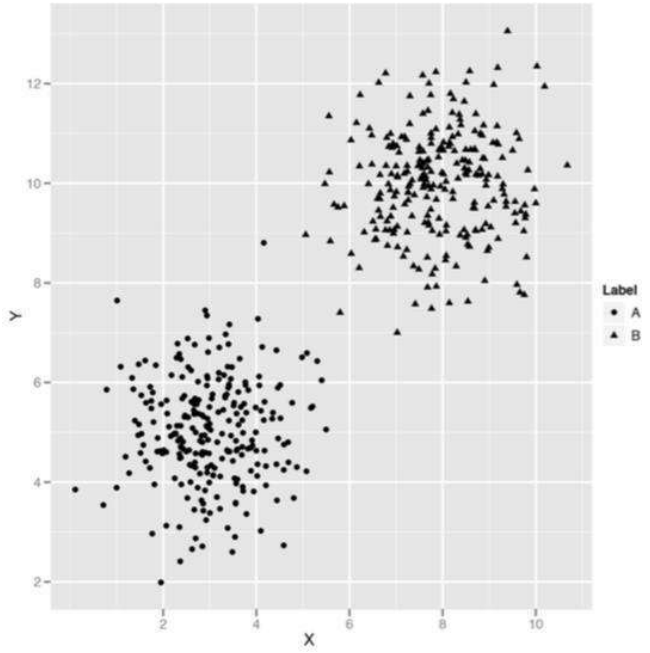
Source: Andreas Mueller – Applied Machine Learning

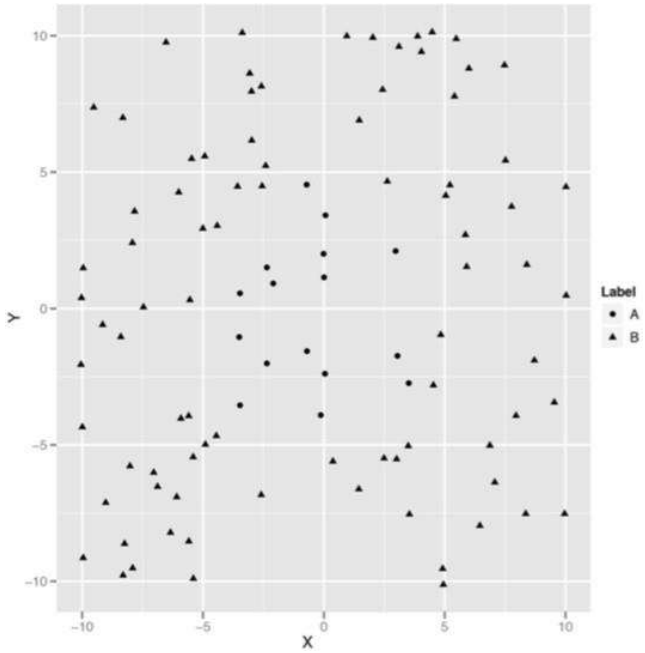
K-Nearest Neighbor

- Memory based
  - learning by memorization: memorize all previously encountered instances.
  - given a new instance, find one from the memorized set that most closely "resembles" the new one and assign new instance to the same class as the "nearest neighbor".
  - how do we define "resembles"? Many options for distance and similarity functions.
- KNN is "lazy"
  - defers all the real work until new instance is obtained; no attempt is made to learn a generalized model from the training set.
  - less data preprocessing and model evaluation, but more work must be done at classification time.

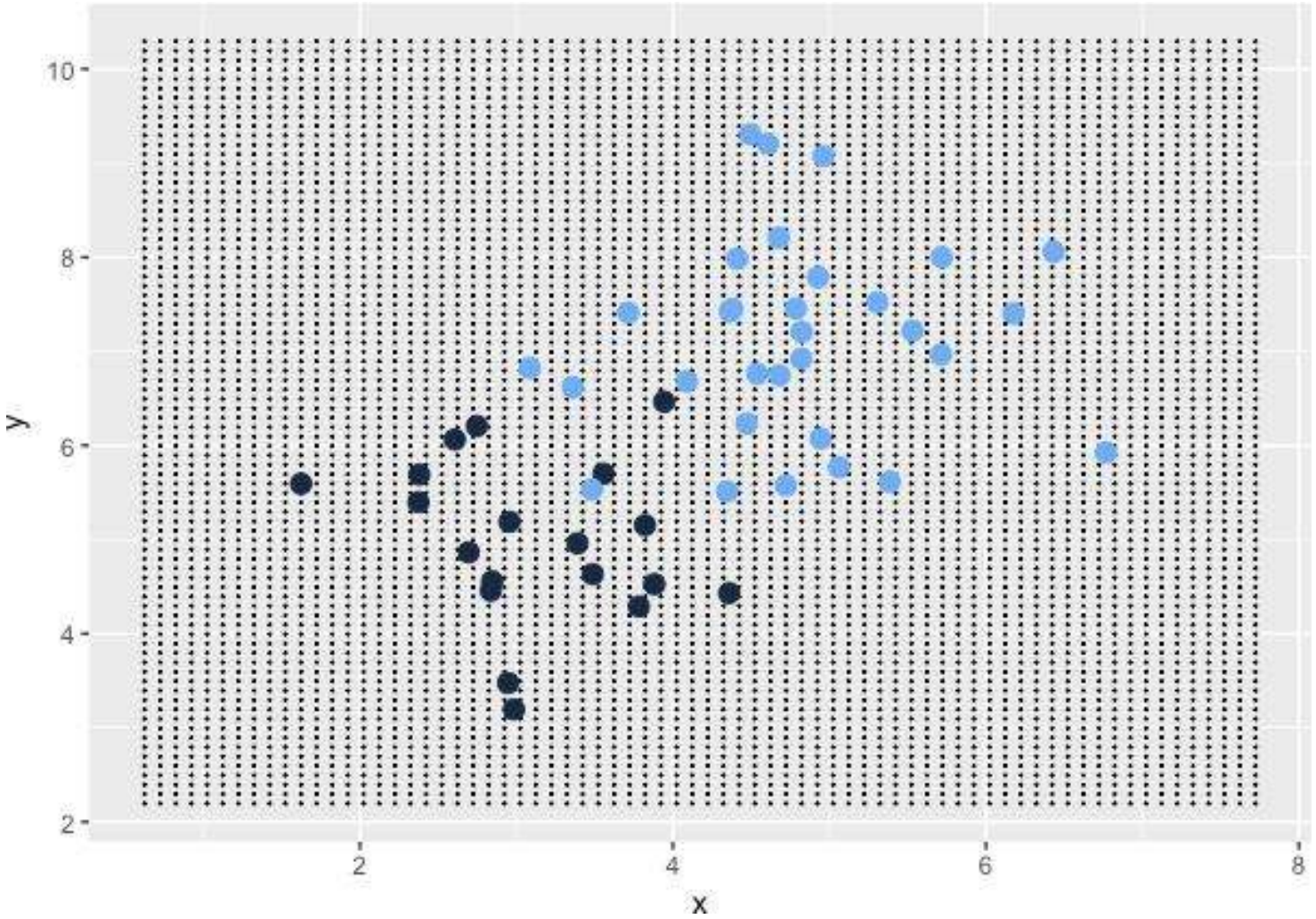
Decision boundary

How could you try to build a classifier that would match a complicated decision boundary (on the right)?

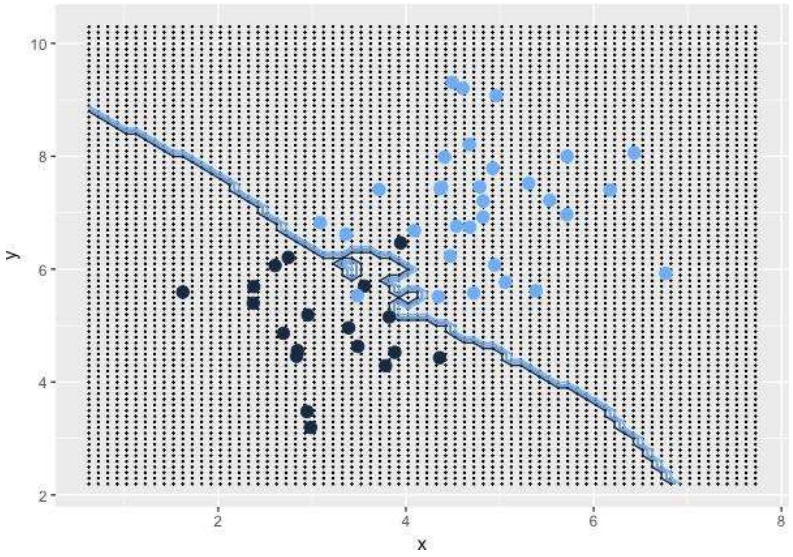




…use the points nearest the point you're trying to classify to make your guess

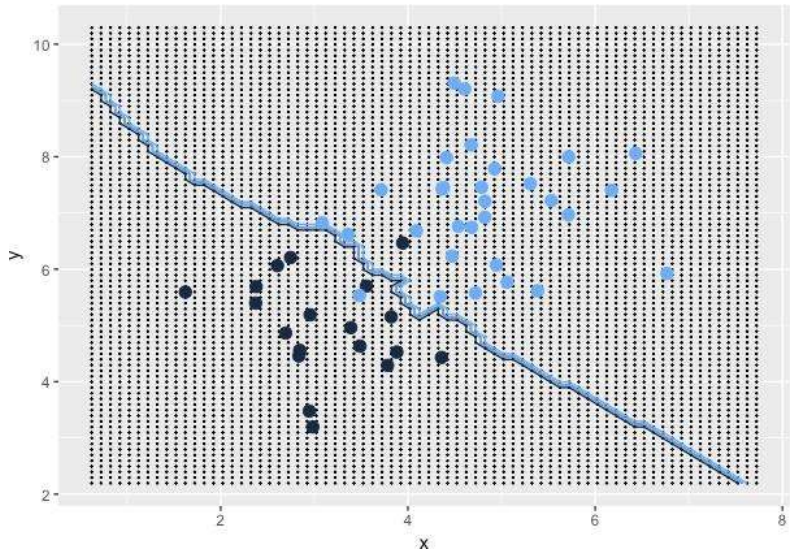

10
8
6
4
2
y
2
4
6
8
x

K =3

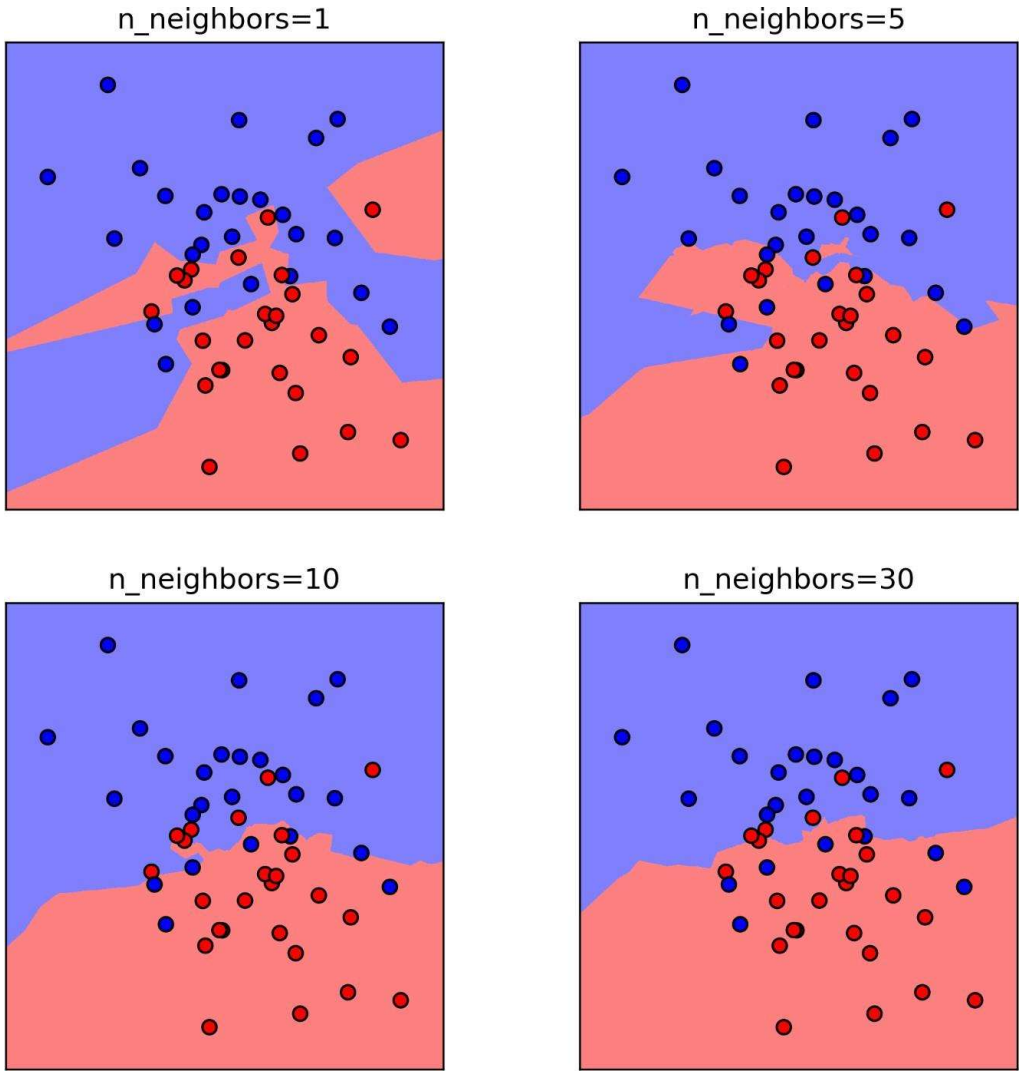


Increasing k reduces variance, increases bias

Choose the K that reduces the number of errors (cross-validated)



K =5


n_neighbors=1
n_neighbors=5
n_neighbors=10
n_neighbors=30

K-Nearest-Neighbor Strategy

- Given object x, find the k most similar objects to x
  - The k nearest neighbors
  - Variety of distance or similarity measures can be used to identify and rank neighbors
  - Note that this requires comparison between x and all objects in the database
- Classification
  - Find the class label for each of the k neighbors
  - Use a voting or weighted voting approach to determine the majority class among the neighbors (a combination function)
    - Weighted voting means the closest neighbors count more
  - Assign the majority class label to x
- Prediction/Regression
  - Identify the value of the target attribute for the k neighbors
  - Return the weighted average as the predicted value of the target attribute for x

Combination Functions

- Voting: the "democracy" approach
  - poll the neighbors for the answer and use the majority vote
  - the number of neighbors (k) is often taken to be odd in order to avoid ties
    - works when the number of classes is 2
    - If there are more than two classes, take k to be the number of classes plus 1

- Impact of k on predictions
  - different values of k affect the outcome of classification
  - we can associate a confidence level with predictions (this can be the %of neighbors that agree)
  - problem is that no single category may get a majority vote
  - strong variations in results for different choices of k, this an indication that the training set is not large enough

KNN and Collaborative Filtering

- Collaborative Filtering Example
  - A movie rating system
  - Ratings scale: 1 ="hate it"; 7 ="love it"
  - Historical DB of users includes ratings of movies by Sally, Bob, Chris, and Lynn
  - Karen is a new user who has rated 3 movies, but has not yet seen "Independence Day"; should we recommend it to her?

Will Karen like "Independence Day"?

	Sally	Bob	Chris	Lynn	Karen
Star Wars	7	7	3	4	7
Jurassic Park	6	4	7	4	4
Terminator II	3	4	7	6	3
Independence Day	7	6	2	2	?

KNN and Collaborative Filtering

	Star Wars	Jurassic Park	Terminator 2	Indep. Day	*Average*	Cosine	Distance	Euclid	Pearson
Sally	7	6	3	7	5.33	0.983	2	2.00	0.85
Bob	7	4	4	6	5.00	0.995	1	1.00	0.97
Chris	3	7	7	2	5.67	0.787	11	6.40	-0.97
Lynn	4	4	6	2	4.67	0.874	6	4.24	-0.69
Karen	7	4	3	?	4.67	1.000	0	0.00	1.00

K	Prediction
1	6
2	6.5
3	5

K is the number of nearest neighbors used in to find the average predicted ratings of Karen on Independence Day.

Summary of KNN

- To make a prediction, KNN identifies those K training observations that are closest to X.
- And then, X is assigned to the class to which the plurality of these observations belong.
- Therefore
  - KNN is superior to LDA or Logistic Regression when the decision boundary is highly non-linear
  - However, KNN does not tell anything about what predictors are important.

Summary of KNN

- Interpretation:
  - The whole model is difficult to interpret beyond the mere similarity between neighbors.
  - This makes KNN appropriate for recommendation systems
- Dimensionality/efficiency
  - KNN considers all features (predictors) in the problem space.
  - Solution
    - perform feature selection
    - inject domain knowledge into similarity calculation by setting your own distance function.

Summary of KNN

- When to use it?
  - Less than 20 features
  - Lots of training data
  - Always scale/standardize feature values.
- Advantages
  - Simple algorithm: easy to program, interpret and explain
  - Learn complex target function
  - No optimization or training required
  - Classification accuracy can be very good; can outperform more complex models.
- Disadvantages
  - Slow at query time
  - Easily fooled by irrelevant attributes and the scale of the data