

INET 4061

Data Science I: Fundamentals

Lecture 5

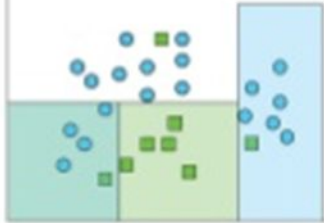
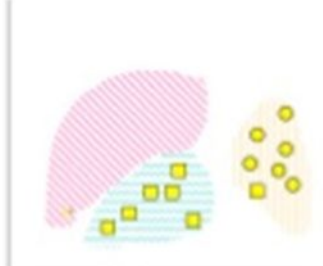
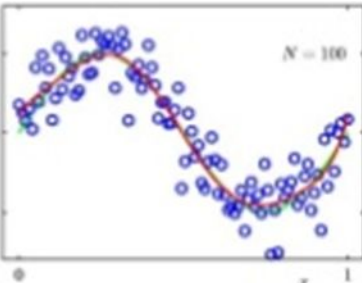

Classification

# Week 5: Overview

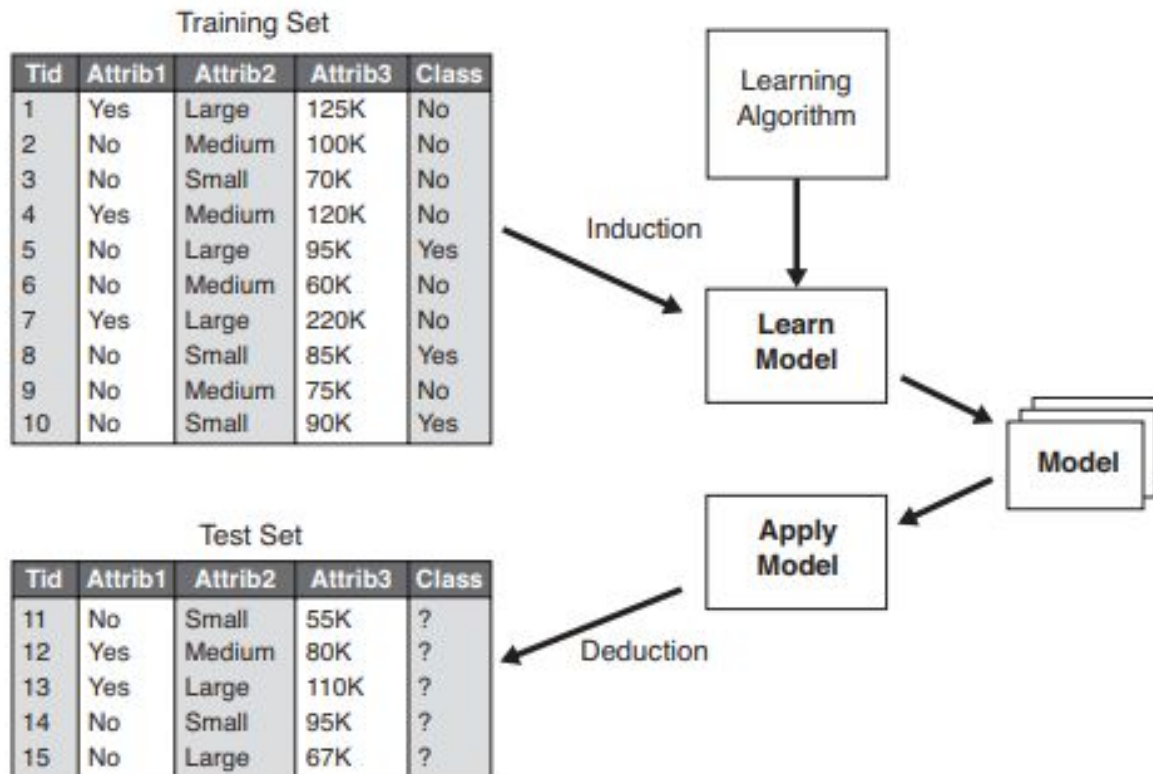
## Classification

- Logistic Regression
- Classification Tree
- Support Vector Machine
- K Nearest Neighbor
- Neural Network
- Classification output and comparisons

# So far ...

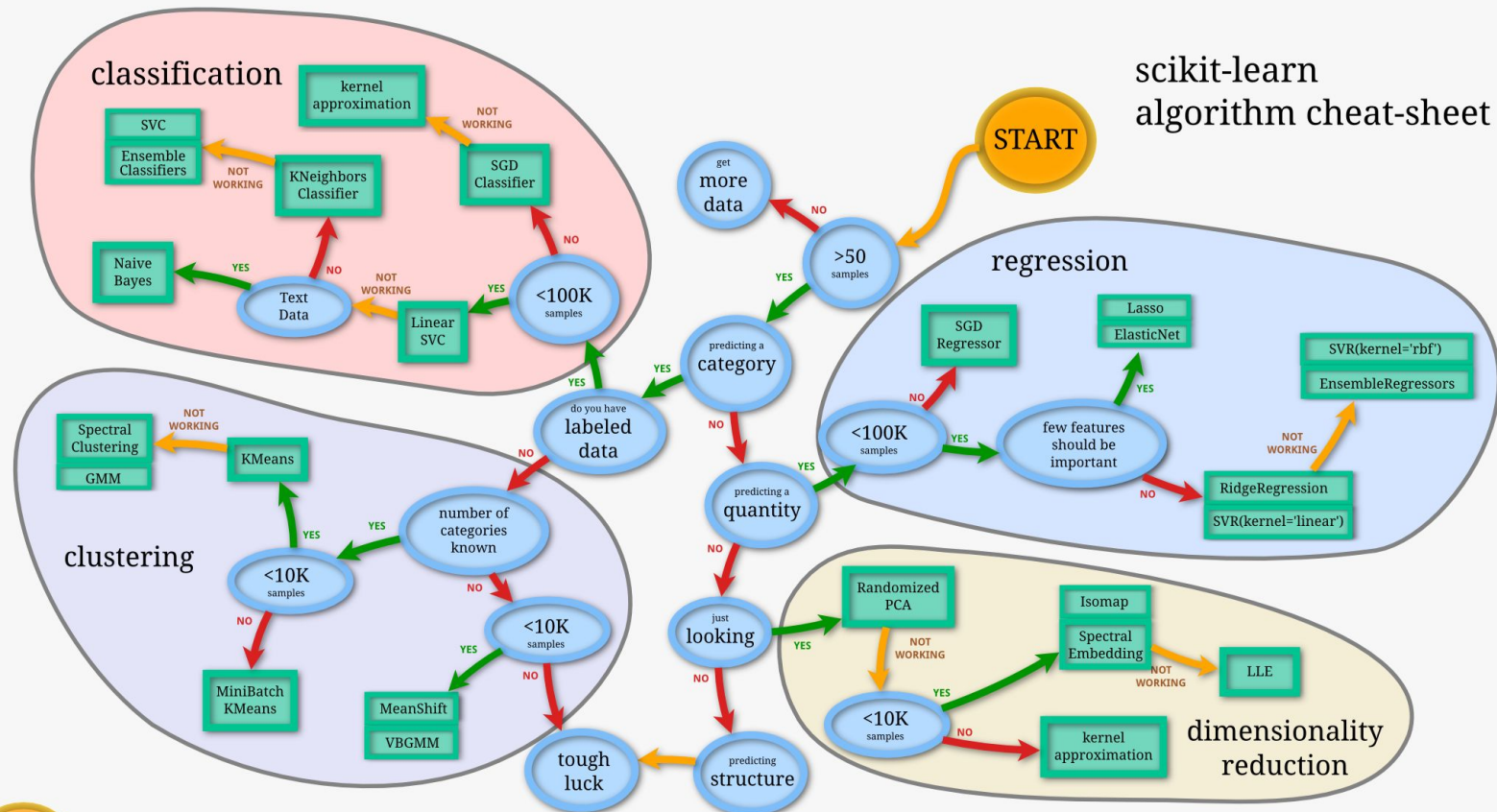
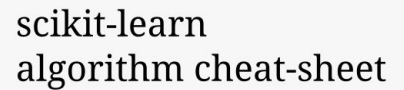
Predictive methods	Descriptive methods
<div><b>Classification</b></div>  <p>Learns a method for predicting the instance class from pre-labeled (classified) instances</p>	<div><b>Clustering</b></div>  <p>Finds "natural" grouping of instances given un-labeled data</p>
<div><b>Regression</b></div>  <p>An attempt to predict a continuous attribute</p>	<div><b>Association Rules</b></div>  <p>Method for discovering interesting relations between variables in large DBs</p>

# Classification Model



**Figure 4.3.** General approach for building a classification model.

# Algorithm Cheat Sheet



# Logistic Regression

- Calculates Classification Probabilities
- S-shaped function
  - Closely related to exponential distributions
  - Linear model for a transformation of the probability
    - Generalized Linear Model

$$g(p) = \beta_0 + x \cdot \beta$$

# Logistic Regression Formula

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta$$

$$p = \frac{e^{\beta_0 + x \cdot \beta}}{1 + e^{\beta_0 + x \cdot \beta}} = \frac{1}{1 + e^{-(\beta_0 + x \cdot \beta)}}$$

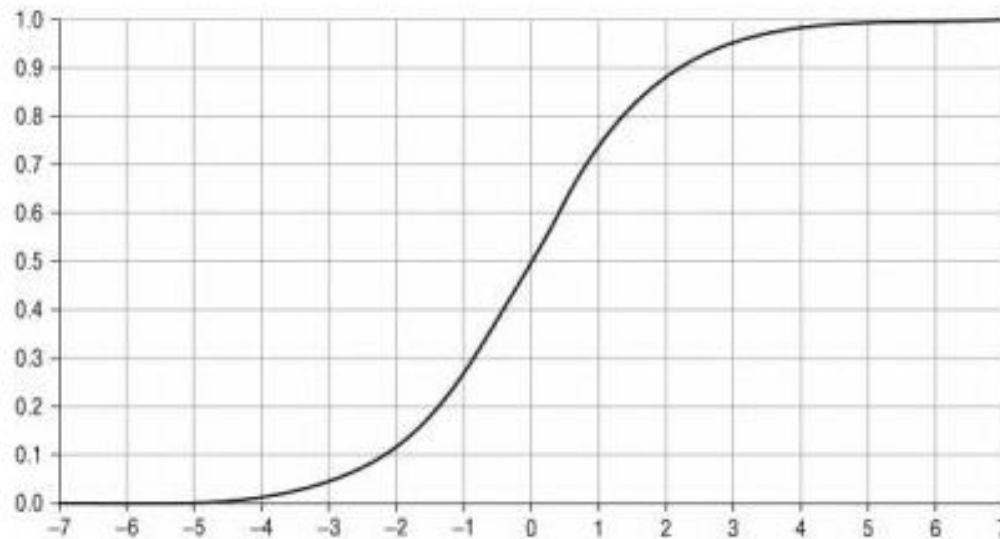


Figure 6.13: The logistic function goes from 0 to 1 just like a probability.

# Classification Tree

---

Chapter 8 Tree Based Methods

<http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf>

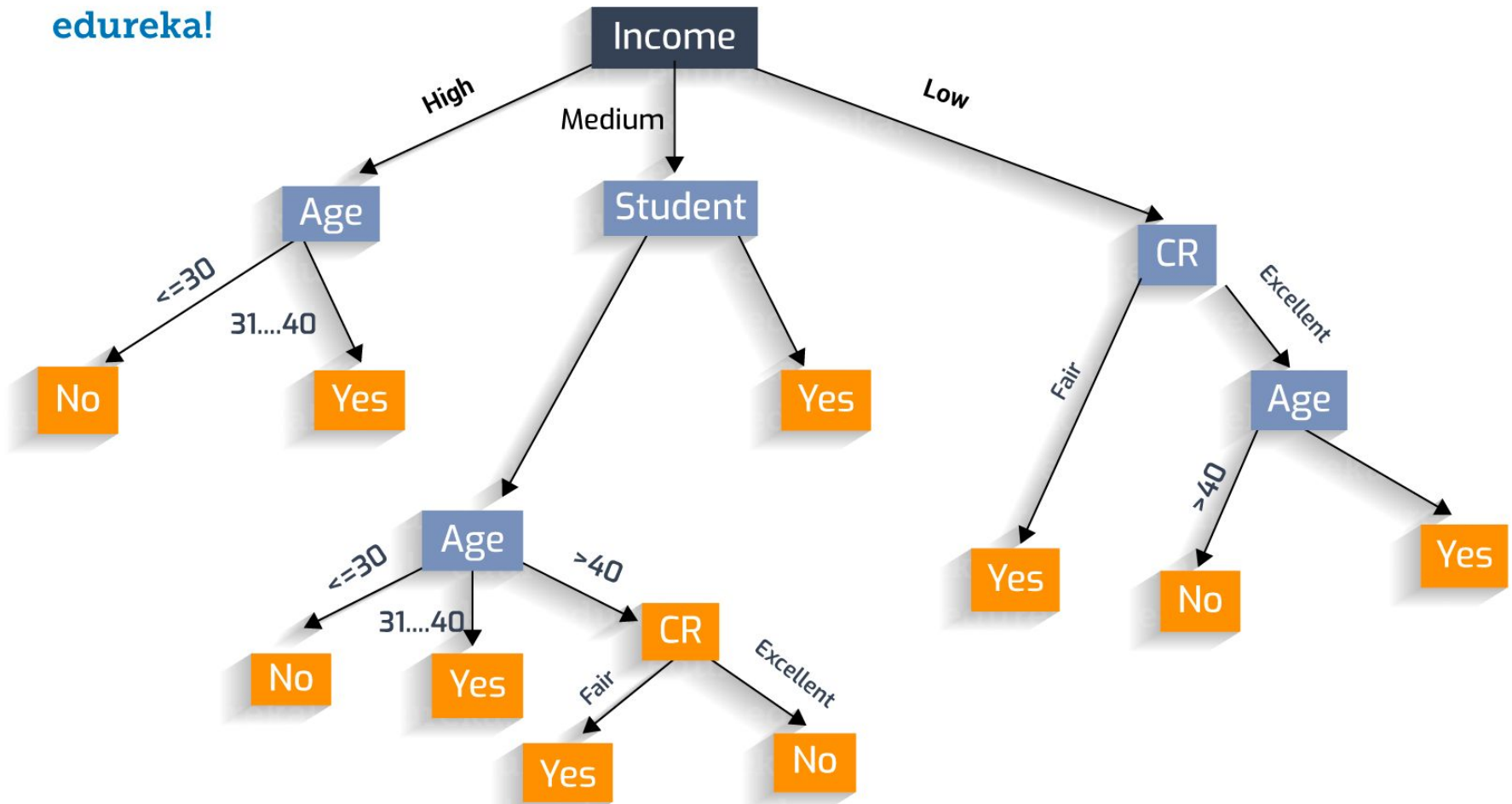


# What Is A Decision Tree?

- Algorithm to predict behavior (directed mining) or explain how underlying data attributes are distributed (undirected mining)
- Uses a tree structure to represent the data-mining model
- Each node can be expressed as a set of rules that describe the function of that node as well as the nodes that led to that node
- Classification Trees
  - Classifies cases into specific groups
  - Different Leaves may make the same classification, but for a different reason
- Regression Trees
  - Estimate the value of a target variable (numeric)

# Tree Structure

edureka!



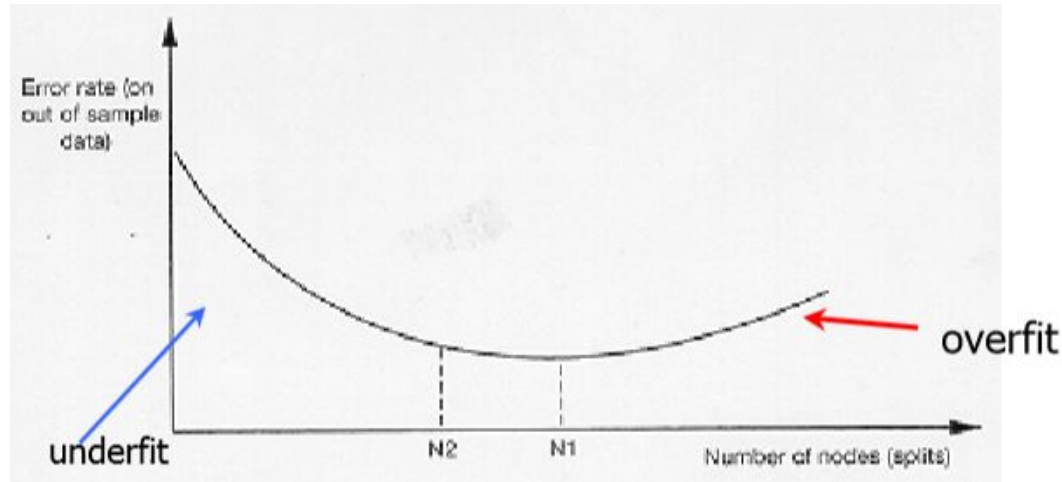
# When To Use Trees

- Segment a set of data that represents a group of potential customers
- Identify possible interactive relationships among variables to understand how a change in one variable can affect another
- Represent relationships among variables visibly
- Simplify attributes and categories to focus on ones that impact predictions
- Explore data to identify significant variables in a data set that can be used as a target

# Decision Tree Characteristics

- Simplicity –follow the tree to understand rules and classifications (provides reasons)
- Decision trees represent:
  - Rules to select records
  - Series of questions
    - Root question should best differentiate among target classes
    - Leaf node determines classification
- Effectiveness is percentage of correct classifications for known records
- Underlying Structure: most algorithms -binary tree
- Process: Split at decision points
- Validation Method: Cross validation (test subsets of known data to determine how well the model correctly predicts values based on that data)

# Underfit and Overfit



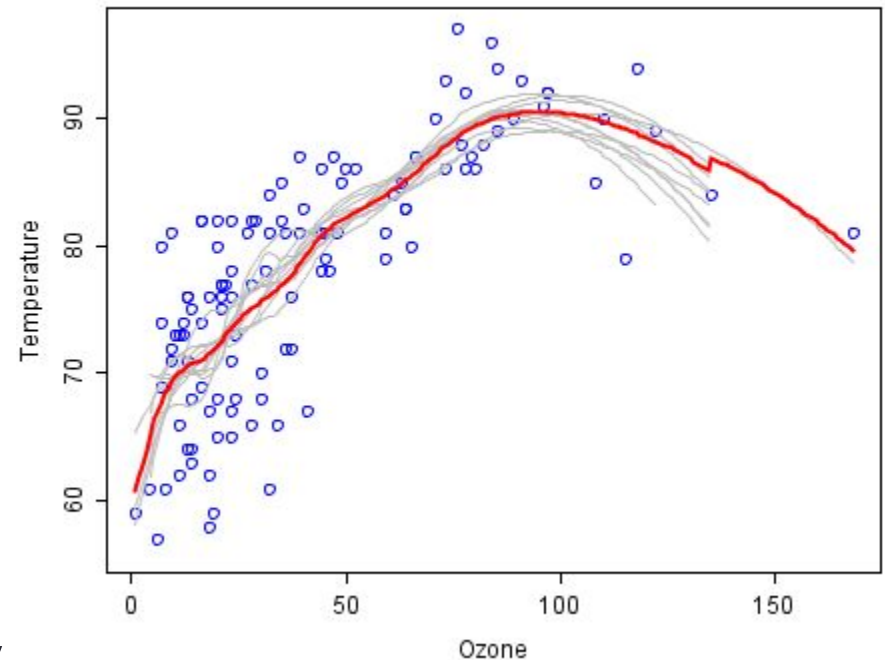
Underfit – model has not been trained with enough cases and is too simple  
Overfit – model over trained to training set and does not generalize well

# Improve Accuracy

- Don't want to over train (over fit tree)
- Bagging, Random Forests, Boosting
  - Combine multiple trees to produce a single prediction
  - Often improves prediction accuracy
  - Some loss in interpretation

# Bagging

- Bootstrap aggregating
- Model averaging
- Improve stability and accuracy
  - Reduce variance
  - Helps to avoid overfitting
- Use: neural nets, classifiers, regression trees



# Boosting

- Reduce bias in supervised learning
- Multiple weak learners to create one strong learner
  - Weak learner: classifier only slightly correlated with the true classification
  - Weight weak learner by accuracy
  - After add weak learner, reweight data
- ex. AdaBoost



# Building a Decision Tree

## Recursive partitioning

- 1. Determine which attribute is the most relevant –splits population cases by that attribute
- 2. Each partitioned group of cases is a node
- 3. Process repeated for each subgroup until a stopping condition is met
  - 1. All cases in the node meet the criteria;
  - 2. No more attributes to split on;
  - 3. Not enough data for statistically significant results

# Problem Trees

- Node has no predominate value
- -> Over-fit (finding patterns that are coincidental)
- -> Under-fit (model does not have enough attributes to discover important patterns)
- All cases have about the same chance of occurrence
  - Consider removing node
- Derive incorrect outcomes
  - Consider pruning tree (remove nodes)
  - Bonsai or stunting 0technique –determine if split is statistically significant before creating a split
- Certain attributes have little impact on outcome
- Careful with default values so don't predict incorrect outcomes

# Pure Tree

- Tree that has only pure nodes is a pure tree
- Each pure leaf node contains 100% of a given type of case
- Suspect over-fitting
- Only one attribute actually determines probability of reaching a target
  - Example:
    - Surf conditions: waves, temperature, coolness, break quality
    - Only relevant attribute: waves

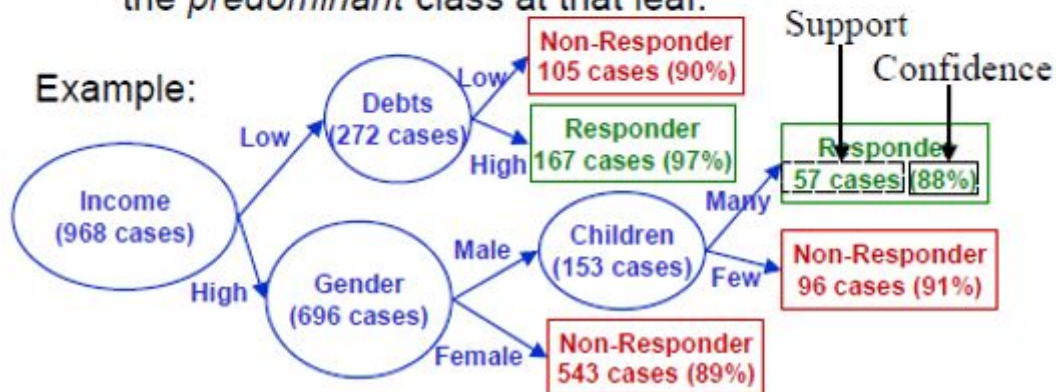
# Support And Confidence

## Building a Tree\*

Support and Confidence

Some trees write the following information alongside each node:

- **support**: the number of training examples that reached that leaf (irrespective of their actual class)
- **confidence**: the percentage of training examples in the predominant class at that leaf.



# **Support Vector Macines**

# Support Vector Classifier

Discriminative classifier

- Optimal hyperplane
- Categorizes unknown data

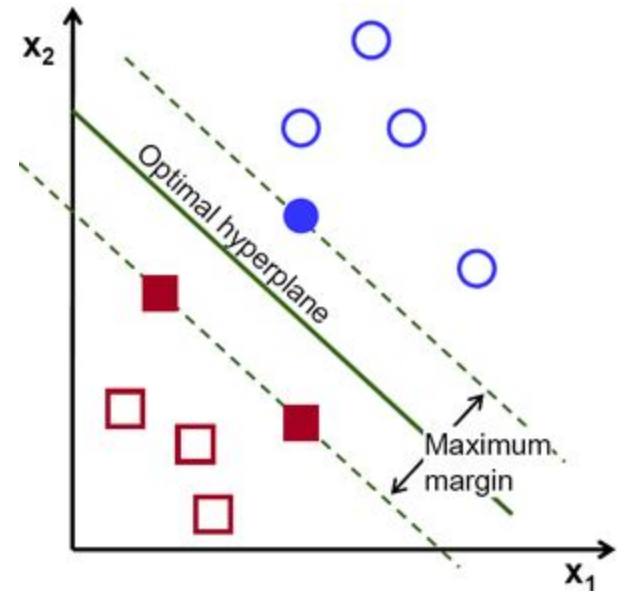
$$f(x) = \beta_0 + \beta^T x$$

Weight vector  $\beta$

Bias  $\beta_0$

Soft margin –slack variables allow observations to be on the wrong side of the margin or hyperplane

2D simplification:



# Support Vector Machine Kernel

- Kernel –generalization of the inner product

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

- Linear Kernel

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}.$$

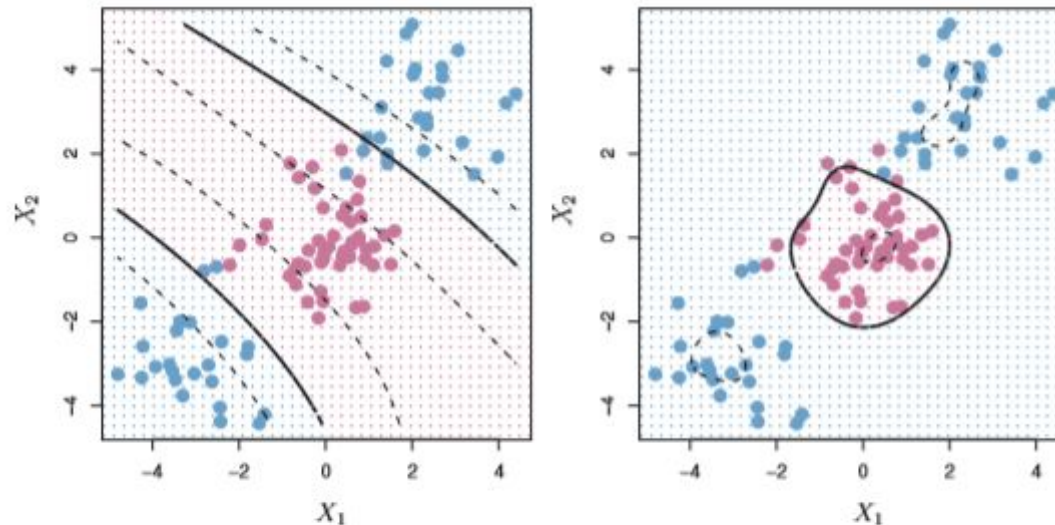
- Polynomial Kernel

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^p x_{ij} x_{i'j})^d.$$

- Radial Kernel

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2)$$

# Polynomial vs. Radial Kernel



**FIGURE 9.9.** Left: An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule. Right: An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.



# K Nearest Neighbor

---

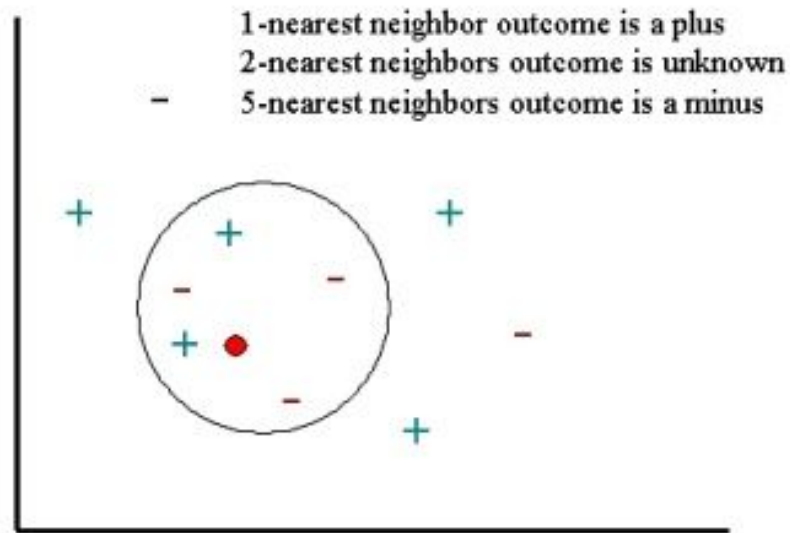
<http://www.statsoft.com/Textbook/k-Nearest-Neighbors>

<https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/>

# K-nearest neighbor

- Supports memory-based model
  - Memory Based Reasoning
  - Similarity
    - Remember past cases
    - New Case –find similar past cases
- Instances with known outcome
  - Set of independent variables and dependent outcomes
  - Regression
    - Continuous dependent variables
  - Classification
    - Categorical or discrete dependent variable

# Classify a New Point



Classify the outcome based on a selected number of its nearest neighbors.

# k-Nearest Neighbor Prediction

Distance Function

Examples

$$D(x, p) = \begin{cases} \sqrt{(x - p)^2} & \text{Euclidean} \\ (x - p)^2 & \text{Euclidean squared} \\ \text{Abs}(x - p) & \text{Cityblock} \\ \text{Max}(|x - p|) & \text{Chebyshev} \end{cases}$$

Voting scheme

Winner determines the classification

# Finding k

- Finding k
  - small  $k \rightarrow$  large variance in predictions
  - large  $k \rightarrow$  bias
- Cross Validation
  - Set  $k$  large enough to minimize the probability of misclassification and small enough (with respect to the number of cases in the example sample) so that the  $K$  nearest points are close enough

# How does k-nearest neighbor work?

- Given a new case of dependent values (query point), estimate the outcome based on the KNN examples
  - Find K examples closest in distance to query point
    - For example
      - Regression –average outcomes of the K nearest neighbors
      - Classification -majority of votes
- Cross validation
  - Test subsets of known data to determine how well the model correctly predicts values based on that data

# Artificial Neural Network (ANN)

---

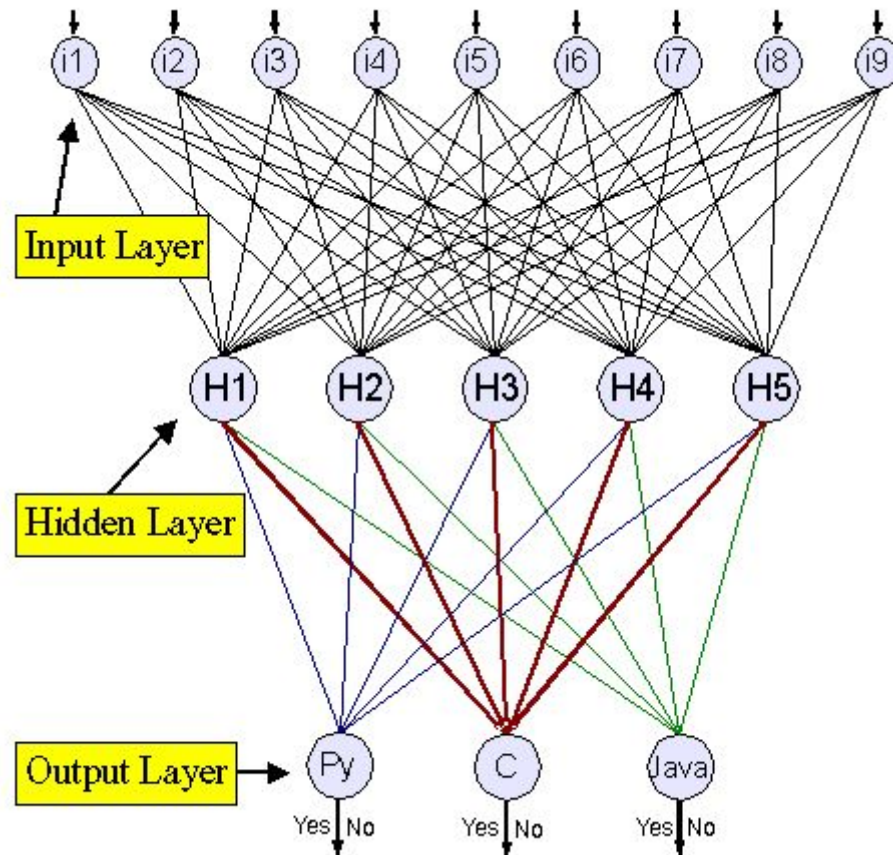
<https://www.ibm.com/developerworks/library/l-neural/>

Chapter 8 -Neural Networks Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, Third Edition

by Gordon S. Linoff and Michael J.A. Berry

John Wiley & Sons © 2011 (888 pages) Citation ISBN:9780470650936

# Neural Network Structure





# What is a neural network?

- Analogy -biological neuron
- Sums weights distributed over nodes
- Node “fires” based on weighted inputs
  - Activation function
    - $\hat{A}_a = (X1 * W1)+(X2 * W2)+...+(Xi * Wi)+...+ (Xn * Wn)$
    - input an array of weighted quantities
    - sums them
    - if the sum meets or surpasses some threshold, output a quantity
- Build the model -Supervised Training
  - Forward Propagation
  - Backward Propagation
  - Readjust weights
- Unsupervised Training (SOM + ART models)

# Model Comparison: Strengths and Weaknesses

---

# Logistic Regression

## Pros

- Probability scores for outputs
- Efficient implementations available
- Can parallelize
  - Batch gradient descent
  - Estimation technique to transform estimators into linear estimate of log odds
  - Piece-wise linear techniques
- Multi-collinearity is not really an issue and can be countered with L2 regularization to some extent
- Wide spread acceptance for logistic regression solutions
- Does not make many of the key assumptions of linear regression and general linear models that are based on ordinary least squares algorithms –particularly regarding linearity, normality, homoscedasticity, and measurement level

## Cons

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

<http://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2>

<http://www.statisticssolutions.com/assumptions-of-logistic-regression>

# Decision Trees

## Pros

- Provides histogram information for paths of rules and patterns
- Understandable decision rules-Can construct a rule for a classification by following a single series of tree branches
- Visual results–view as a tree, showing splits and result distributions
- Predictive–provides rules to predict the likelihood of new cases fitting the model
- Can handle nonlinear features
- Takes into account variable interactions
- Handles missing data
- Variable “importance” –shows which attributes provide the best splits •Good performance –binary tree search

## Cons

- Can be highly biased to training set
- Use Random Forest
- No ranking score as direct result
- Use leaf node proportions or mean/median (for regression trees) for score and use for ranking
- Can spread too thin –too many classifications causes too few cases to fit the final category, resulting in false conclusions
- Overfitting (fits training set too well; doesn't generalize as well)
- Training performance –Node can be sorted multiple times to determine best split, which can be slow
- Never discovers rules that involve relationships between variables
- Distribution information is lost
- Categorical variables may cause bushy trees
- Error prone when number of training cases per category is small
- Difficult to understand rules when complex trees with many leaves

# SVM

## Pros

- Can handle large feature space
- Can handle non-linear feature interactions
- Does not rely on entire dataset
- Can handle
  - nonlinear decision boundaries
  - missing data for “obvious” cases
  - large feature spaces
    - ex. text analysis.

## Cons

- Not very efficient for large number of observations
- Can be difficult to find appropriate kernel
- For non linear kernels, SVMs can be very costly to train on large datasets

# kNN

## Pros

- Handles noisy data
- Large training data sets
- Accuracy improves over time
- Incremental learning from past cases
- add new cases and solutions to model

## Cons

- Euclidian distance
- Scaling
- Weighting
- Mapping
- Lazy evaluation
- Wait until the need to classify a new case

# ANN

## Pros

- Work well with noisy data
- Appropriate for applications with time
- Perform well for many problems
- Supervised learning and unsupervised clustering

## Cons

- No Explanation of Outputs
- Inputs –numerical values -1 to 1 or 0 to 1
- Difficult Decisions
  - What input attributes will be used to build the network?
  - How will the network output be represented?
  - How many hidden layers should the network contain?
  - How many nodes should there be in each hidden layer?
  - What condition will terminate network training?

# Model Selection and Classification Output Comparison

---



# Classification Model Selection

- Build several models
  - Cross validation to select best model
  - Ensemble methods
- Better data often beats better algorithms
- Design good features
- For large datasets, consider classification performance
  - Select algorithm based on speed or ease of use instead
- How large is the training set?
  - If your training set is small, high bias/low variance classifiers (e.g., Naive Bayes) have an advantage over low bias/high variance classifiers (e.g., kNN), since the latter will overfit.
  - Low bias/high variance classifiers start to win out as the training set grows (they have lower asymptotic error), since high bias classifiers aren't powerful enough to provide accurate models.

# Comparison –Different Kind of Scores

- Model Evaluation
  - Scores – quality measures
  - Estimation methods – estimate value of a score
  - Statistical test – comparison among different solutions
- What does best quality mean -> different kinds of scores
  - What are we interested in?
  - What do we want to optimize?
  - What are the characteristics of the problem?
  - What are the characteristics of the data set?
- The measure you optimize makes a difference
- Use measure appropriate for problem
- Accuracy is often not sufficient or sometimes not appropriate

# Examples of Scores for Classification

Accuracy

Precision

Recall

F-Score

ROC /AUC

# Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

# Confusion Matrix as a Binary Classifier

n=165	Predicted Yes	Predicted No
Actual Yes	100	5
Actual No	10	50

**Accuracy:** how often is the classifier correct?

$$(TP + TN) / \text{total} = (100 + 50) / 165 = 0.91$$

**Misclassification Rate:** how often is the classifier wrong?

$$(FP + FN) / \text{total} = (10 + 5) / 165 = 0.09$$

- equivalent to 1 minus Accuracy
- also known as "Error Rate"

**True Positive Rate:** When it's actually yes, how often does it predict yes?

$$TP / \text{actual yes} = 100 / 105 = 0.95$$

- also known as "Sensitivity" or "Recall"

**False Positive Rate:** When it's actually no, how often does it predict yes?

$$FP / \text{actual no} = 10 / 60 = 0.17$$

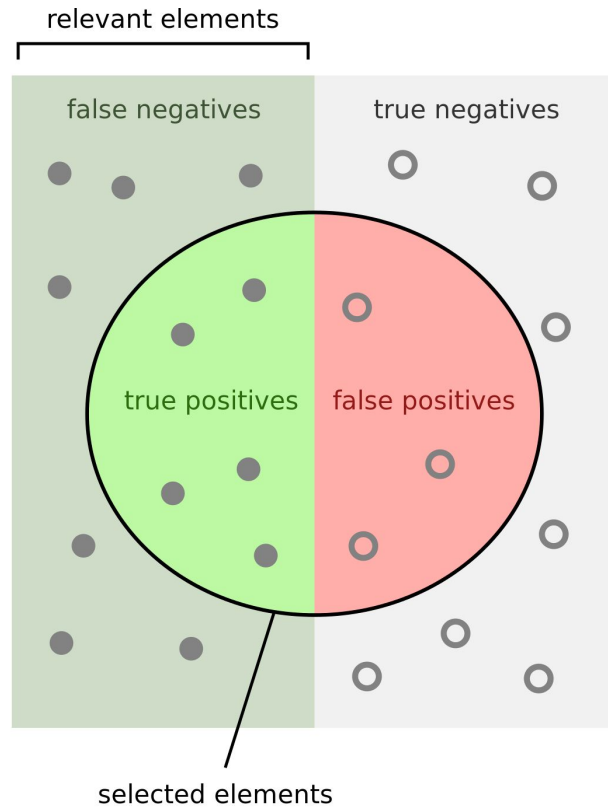
**Recall:** Of the yes's, how accurate are we at predicting them?

$$TP / (TP + FN) = 100 / 105 = 0.95$$

**Precision:** When we predict yes, are accurate are we?

$$TP / (TP + FP) = 100 / 110 = 0.91$$

**F Score:** weighted average of the true positive rate (recall) and precision



$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}$$

How many selected items are relevant?

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

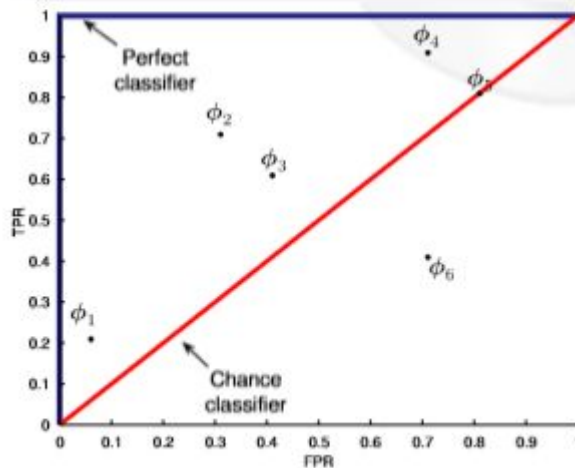
How many relevant items are selected?

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

# Receiver Operating Characteristics (ROC)

## ROC Space

Coordinate system used for visualizing classifiers performance where  $TPR$  is plotted on the  $Y$  axis and  $FPR$  is plotted on the  $X$  axis.



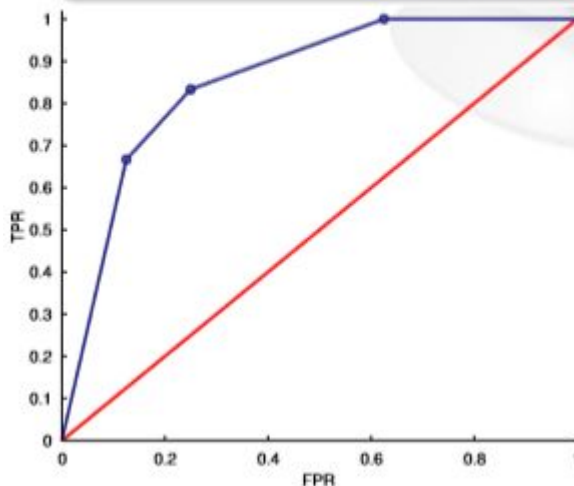
- $\phi_1$ : kNN
- $\phi_2$ : Neural network
- $\phi_3$ : Naive Bayes
- $\phi_4$ : SVM
- $\phi_5$ : Linear regression
- $\phi_6$ : Decision tree

# ROC Curve

- Insensitive to skew class distribution
- Insensitive to misclassification cost
- Each point on the curve represents a different tradeoff (cost ratio) between FP and FN

## ROC Curve

For a probabilistic/fuzzy classifier, a ROC curve is a plot of the TPR vs. FPR as its discrimination threshold is varied



$p(c x)$	$T = 0,2$	$T = 0,5$	$T = 0,8$	$C$
0,99	$c^+$	$c^+$	$c^+$	$c^+$
0,90	$c^+$	$c^+$	$c^+$	$c^+$
0,85	$c^+$	$c^+$	$c^+$	$c^+$
0,80	$c^+$	$c^+$	$c^+$	$c^+$
0,78	$c^+$	$c^+$	$c^-$	$c^+$
0,70	$c^+$	$c^+$	$c^-$	$c^-$
0,60	$c^+$	$c^+$	$c^-$	$c^+$
0,45	$c^+$	$c^-$	$c^-$	$c^-$
0,40	$c^+$	$c^-$	$c^-$	$c^-$
0,30	$c^+$	$c^-$	$c^-$	$c^-$
0,20	$c^+$	$c^-$	$c^-$	$c^+$
0,15	$c^-$	$c^-$	$c^-$	$c^-$
0,10	$c^-$	$c^-$	$c^-$	$c^-$
0,05	$c^-$	$c^-$	$c^-$	$c^-$



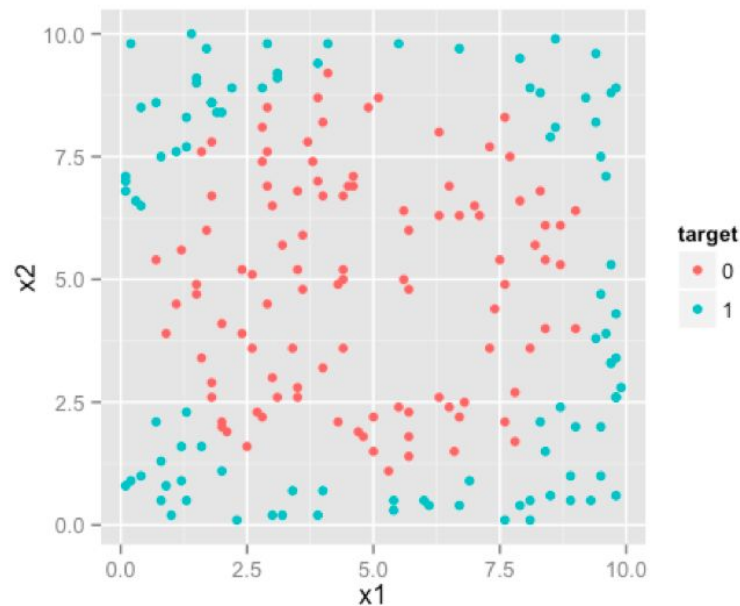
**Classification**

**Model Comparisons**

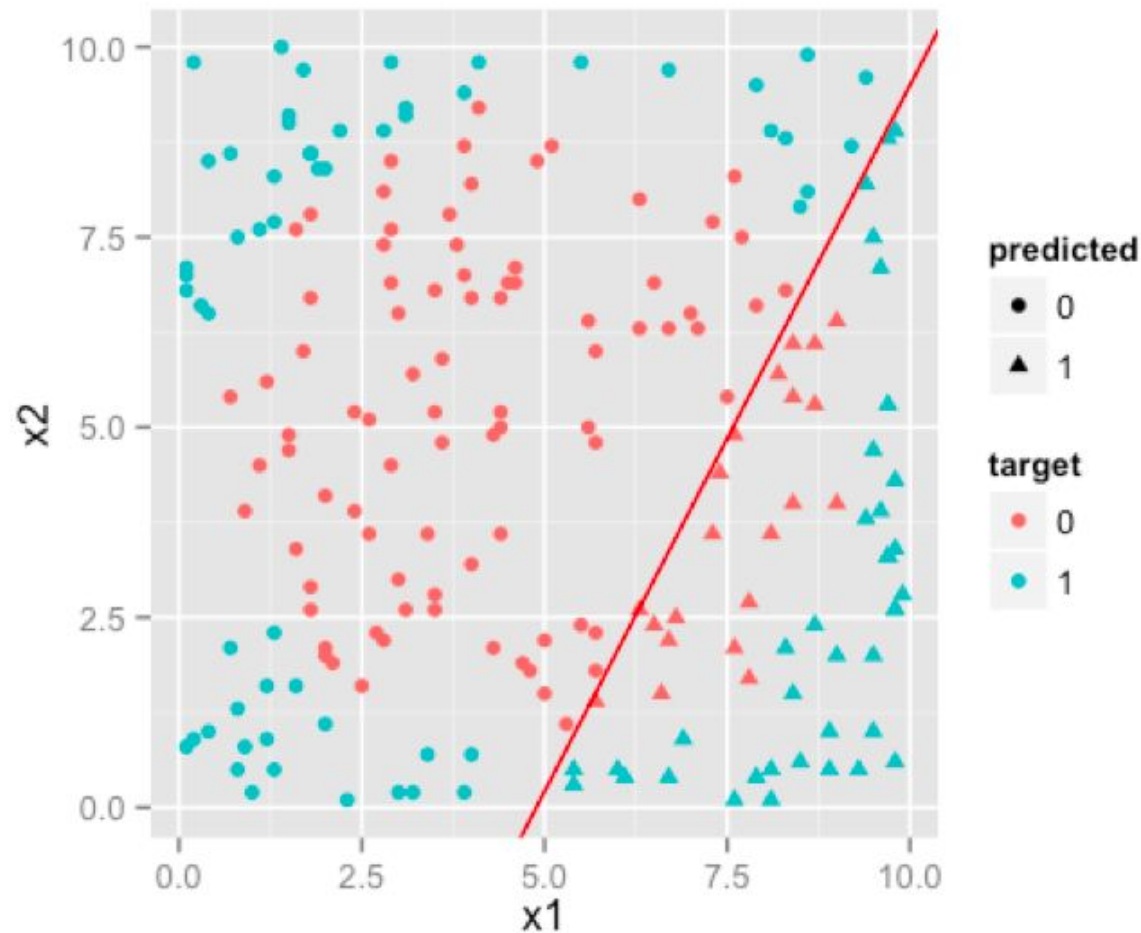
# Example Dataset – How to Classify

<http://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part1/>

---

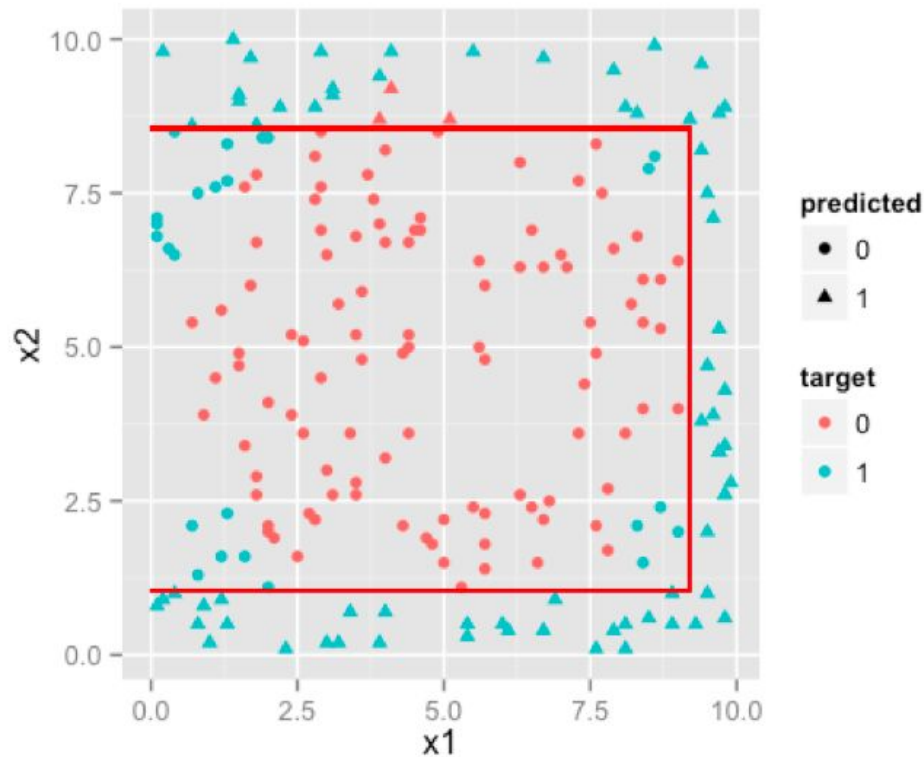


# Logistic Regression –Linear Boundary

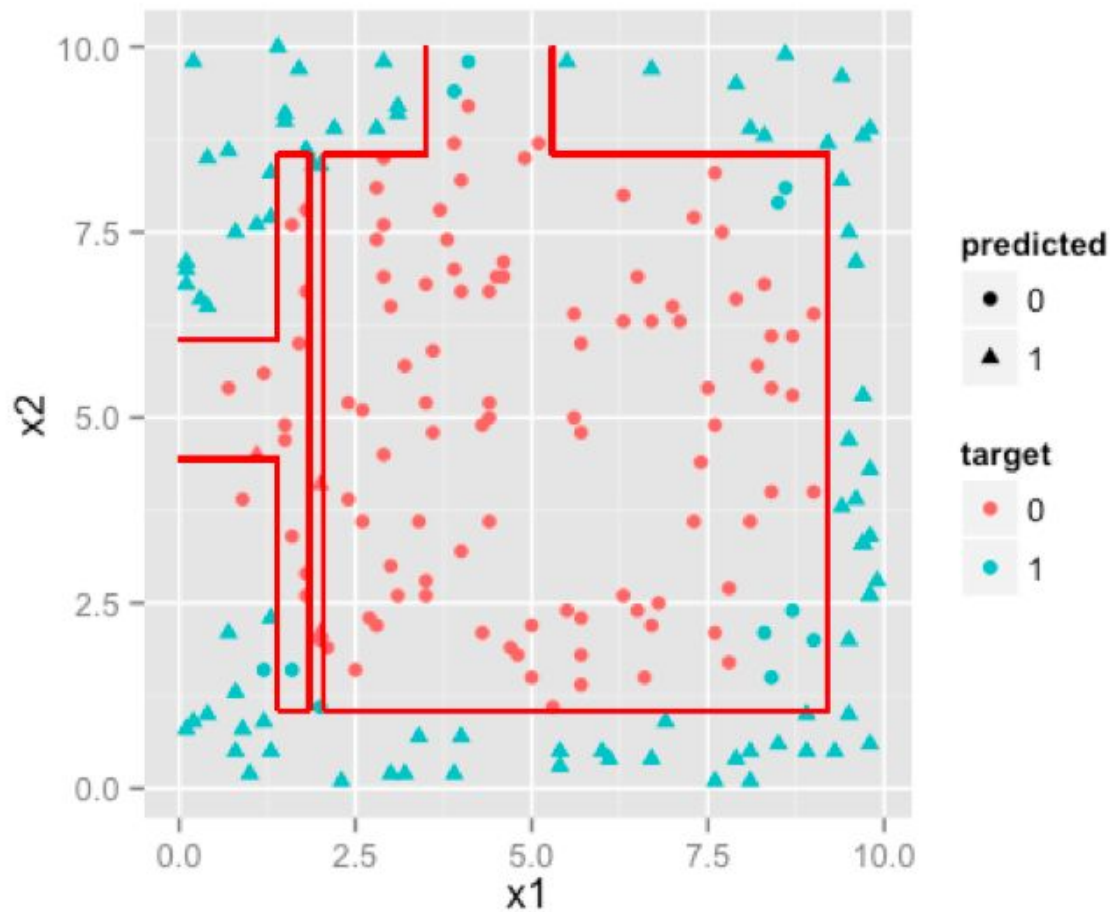


# Decision Tree –rules with constants

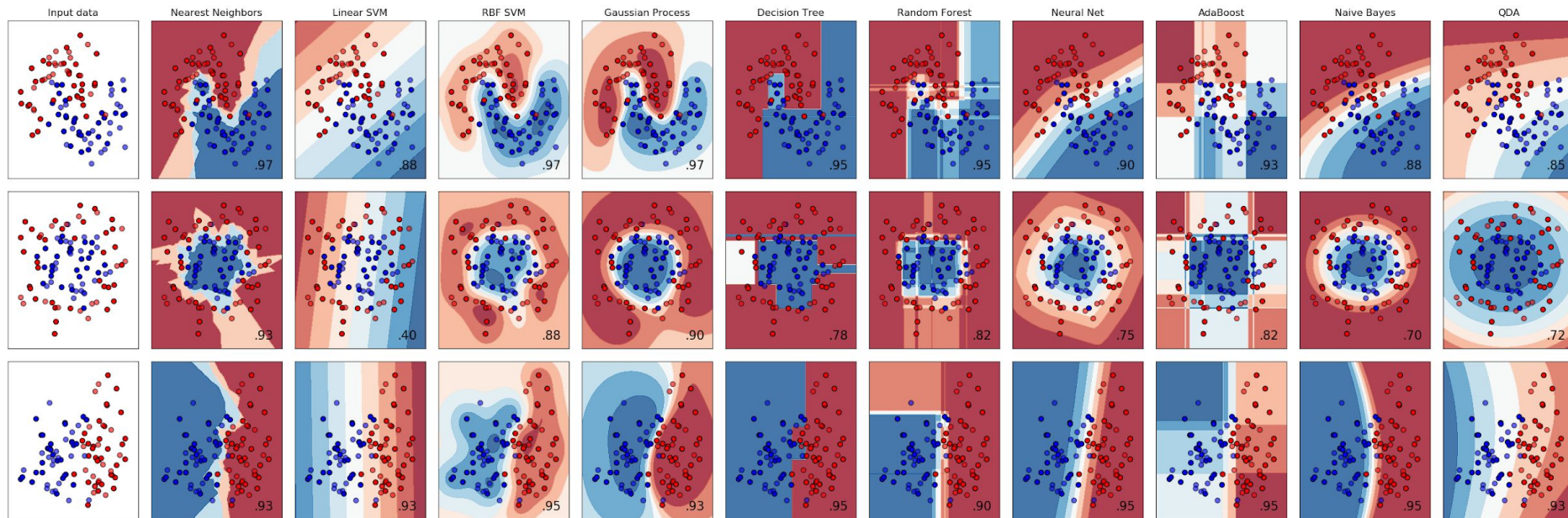
$x < \text{constant}$  or  $x > \text{constant}$



# Decision Tree –more complex rules



# Classifier Comparison



# Suggested Steps

1. Start with logistic regression
  - Use model performance as baseline
2. See if decision trees (Random Forests) provide significant improvement
  - Even if you do not end up using the model, you can use random forest results to remove noisy variables
3. Try SVM if you have large number of features and number of observations are not a limitation for available resources and time
4. Try ensemble models
5. Good data beats any algorithm.
  - Always see if you can find a good feature by using domain knowledge.
  - Try various iterations of ideas while experimenting with feature creation

**THIS CONCLUDES THE PRESENTATION.**

Thank you.



UNIVERSITY OF MINNESOTA

**Driven to Discover<sup>SM</sup>**