Machine Learning & Data Mining

Classification

Kyung-Ah Sohn

Ajou University

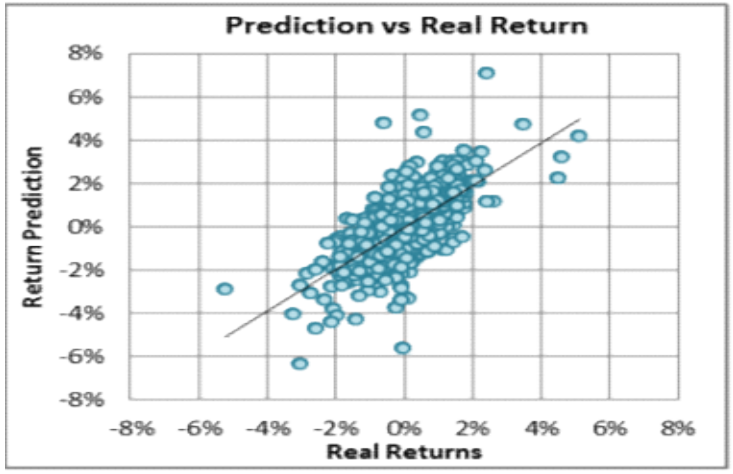
Content

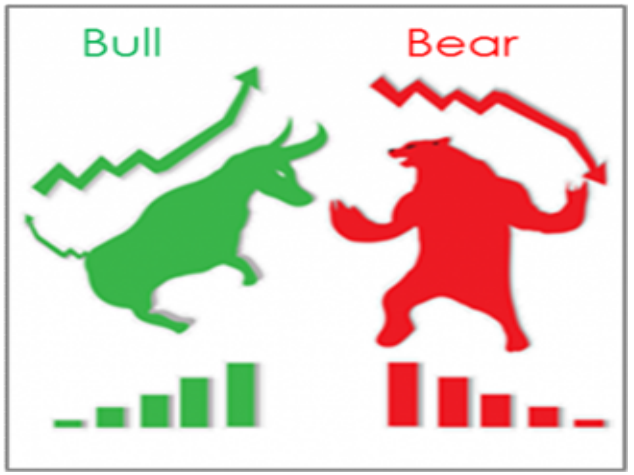
- Supervised approach
  - Classification
  - Regression

- Classification algorithms
  - KNN
  - Bayes classifier
  - Decision tree

Supervised learning

Regression vs **Classification**





Supervised Learning

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$

"Sports"
"News"
"Science"
...

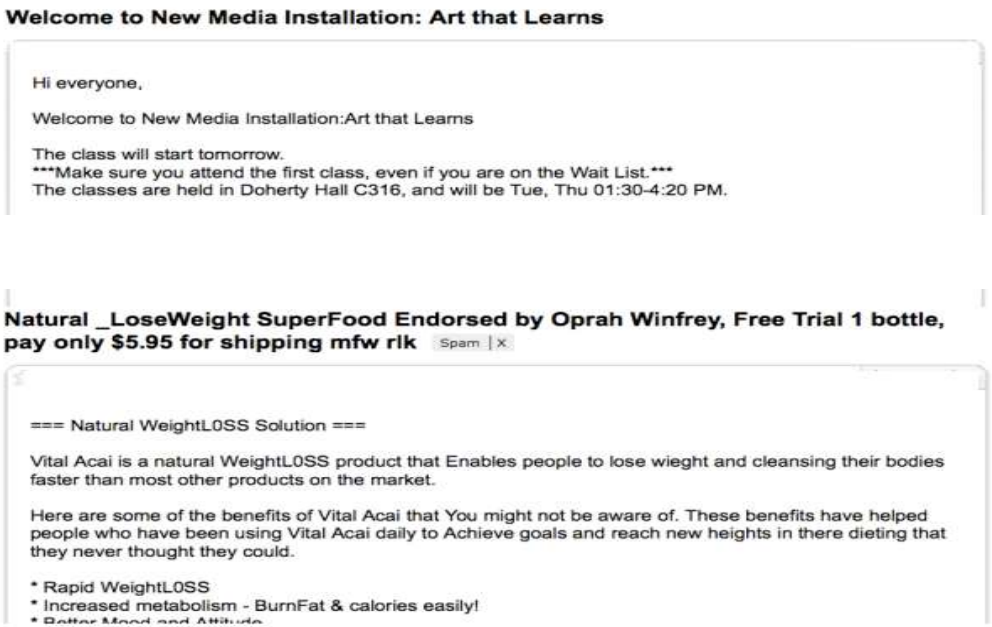
Discrete Labels
Classification

Share Price
"$ 24.50"

Continuous Labels
Regression

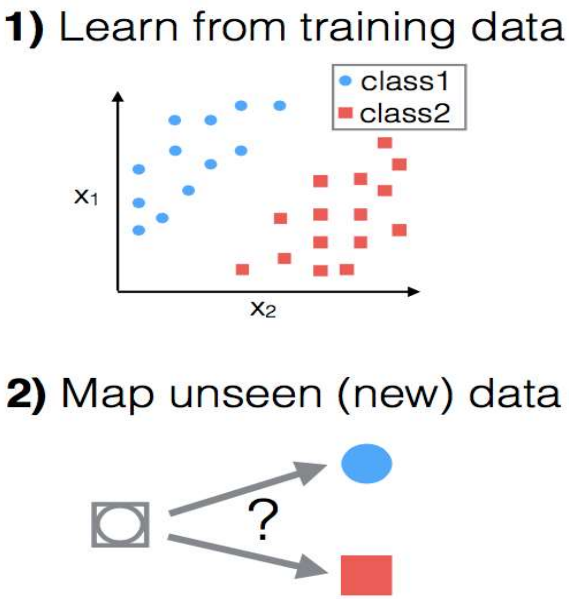
Task: Given $X \in \mathcal{X}$, predict $Y \in \mathcal{Y}$.

Spam Filtering



Spam/
Not spam

Classification: Training vs. Test



Training vs. Test data

- Training data: to learn a classifier (using known labels)

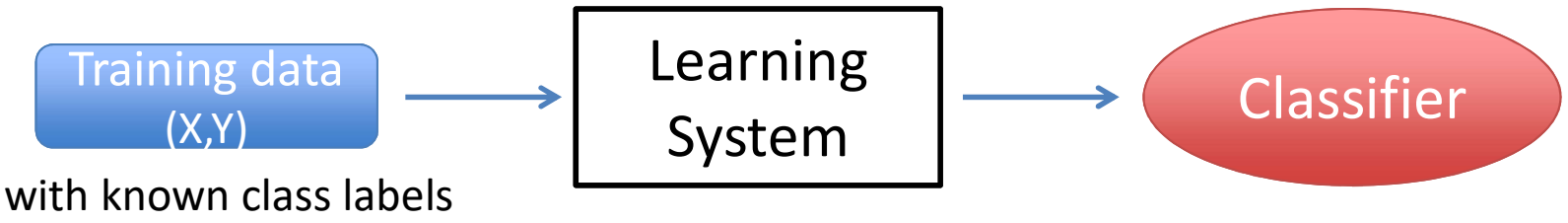
	team	coach	play	ball	score	game	win	goal	label
Doc_1	3	9	0	2	5	7	0	0	**1**
Doc_2	0	7	5	4	0	1	2	1	**0**
…				…					**…**
Doc_n	0	1	0	0	1	2	2	0	**1**

- Test data: to predict a label or to evaluate the classifier

	team	coach	play	ball	score	game	win	goal	label
Doc_new	3	9	0	2	5	7	0	0	**?**

Classification

- **Learning**: Induce classifiers from training data



- **Prediction**: use the learned classifier to predict labels for new data

Explain whether each scenario is a classification or regression problem, and provide n (# of samples) and p (# of features).

(a) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.

(b) We are considering launching a new product and wish to know whether it will be a *success* or a *failure*. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

(c) We are interesting in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.

Classification algorithms

- K-Nearest Neighbor
- Support Vector Machine (SVM)
- Decision trees
- Naïve Bayes
- Logistic regression
- Neural networks
- Bayesian networks
- …

K-NEAREST NEIGHBOR ALGORITHM

Classification example

Training data

No.	v_1=Strength	v_2=Smooth	y=Quality
x_1	5	5	Good
x_2	1	2	Bad
x_3	6	7	Good
x_4	3	5	Bad

Test data

x_5	2	3	?

k-Nearest Neighbors (k-NN) algorithm

- *Instance-based learning*
- 1-NN
  - Predict the same class label as the *nearest* instance in the training set

- In general, k-NN
  - Find the *k* closest training points (according to some distance measures, e.g. Euclidean, …)
  - Predicted class: majority vote of k-neighbors

k-NN classification

- For a given test sample
  1. Calculate distances of all training samples to the test sample
  2. Pick *k* closest training samples
  3. Calculate majority

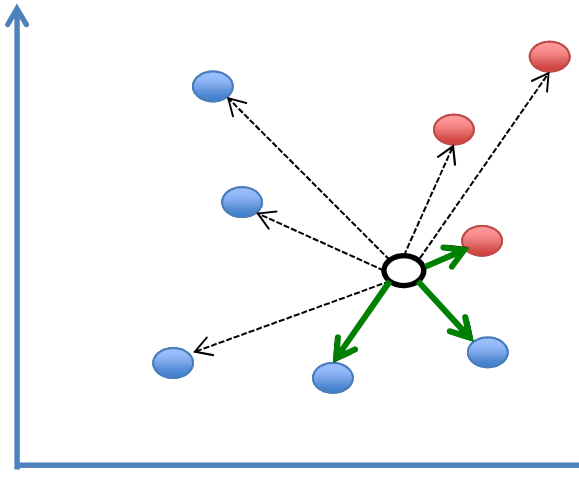
Classifiction example

- Compute the distance between test data and all the training samples

No.	v_1=Strength	v_2=Smooth	y=Quality
x_1	5	5	Good
x_2	1	2	Bad
x_3	6	7	Good
x_4	3	5	Bad
x_5	2	3	?

$$d(\mathbf{x}_1, \mathbf{x}_5) = \sqrt{(5-2)^2 + (5-3)^2} = 3.6$$
$$d(\mathbf{x}_2, \mathbf{x}_5) = \sqrt{(1-2)^2 + (2-3)^2} = 1.4$$
$$d(\mathbf{x}_3, \mathbf{x}_5) = \sqrt{(6-2)^2 + (7-3)^2} = 5.6$$
$$d(\mathbf{x}_4, \mathbf{x}_5) = \sqrt{(3-2)^2 + (5-3)^2} = 2.2$$

Example



Training data
Bad
Good

k=1 ➔ class=Good

K=3 ➔ class=Bad

Neighbor size k

- Choice of k
  - Smaller k ➔ higher variance (less stable)
  - Larger k ➔ higher bias (less precise)

  - Proper choice of k depends on the dataset
    - Heuristics
    - Cross-validation

k-NN

- No explicit training or model, “lazy learning”
- One of the simplest classification methods
- *Gives the maximum likelihood estimation of the class posterior probabilities*
- Can be used as a baseline model
- Many extensions

K-NN (Pros)

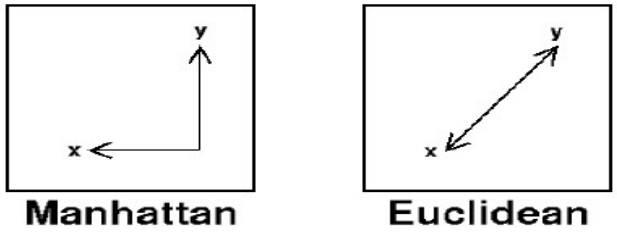
+ Easy to understand and program
+ Explicit reject option
  - If there is no majority agreement
+ Easy handing of missing values
+ Close to optimal
  - *Asymptotic misclassification rate (as the number of data points goes to infinity) is bounded above by twice the Bayes error rate*

K-NN (Cons)

- *Sensitive to noise, irrelevant features*
- Computationally expensive O(*np*)
- *Large memory requirements*
- More frequent classes dominate result
- Curse of dimensionality
  - “nearest” neighbor may be very far
  - In high dimensions, “nearest” becomes meaningless

Choice of Distance Measure

- Euclidean distance
- Manhattan distance
- 1 - Correlation
- 1 - Cosine similarity
- L_p norm
- Mahalanobis distance
- …



Data and Application-dependent

Euclidean Distance

- n objects with d attributes
  - $\boldsymbol{x}_i = (x_1(i), x_2(i), \ldots, x_d(i))$, i=1,…,n

- Most common distance metric is Euclidean distance

$$d_E(i,j) = \left(\sum_{k=1}^{d} (x_k(i) - x_k(j))^2 \right)^{\frac{1}{2}}$$

Example

No.	v_1=Strength	v_2=duration	y=Quality
1	5	5000	Good
2	1	4900	Bad
3	6	2900	Good
4	3	5200	Bad
5	2	3300	?

Euclidean distance??

Euclidean distance

- Makes sense in the case where the different measurements are commensurate; each variable measured in the same units

- If the measurements are different, say length in different units, Euclidean distance is not necessarily meaningful
  - Standardization!

Standardization

- When variables are not commensurate, we can standardize them by subtracting the mean and then dividing by the standard deviation

- Mean $$\bar{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Variance $$\hat{\sigma}^2 = \frac{\sum_i^n (x_i - \bar{x})^2}{n-1}$$

- Standard deviation $$\hat{\sigma} = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n-1}}$$

Normalization!!

- Usually assume features should contribute equally.
  - Min-max normalization

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

  - Z-score standardization

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$

Extensions

- L_p metric:

$$dist(i,j) = \left(\sum_{k=1}^{d} | x_k(i) - x_k(j) |^p \right)^{\frac{1}{p}}$$ where p >=1

- Manhattan, city block or L_1 metric:

$$dist(i,j) = \sum_{k=1}^{d} |x_k(i) - x_k(j)|$$

- L_∞

$$dist(i,j) = \max_k |x_k(i) - x_k(j)|$$

Testing alternative values of k

k value	# false negatives	# false positives	Percent classified Incorrectly
1	1	3	4 percent
5	2	0	2 percent
11	3	0	3 percent
15	3	0	3 percent
21	2	0	2 percent
27	4	0	4 percent

- create several sets of 100 patients at random and repeatedly retest the result
- Or do cross-validation (will be discussed later)

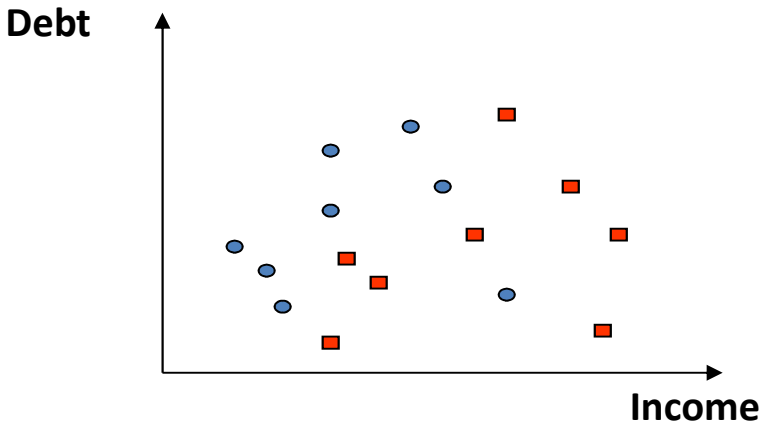
DECISION TREE

Understanding decision trees

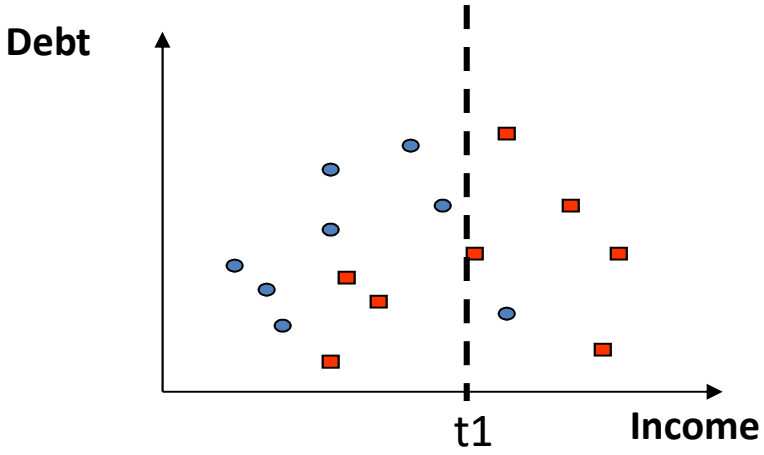
- A model in the form of a **tree structure**
- Nodes indicate a decision to be made on the attribute
- Branches: indicate the decision's choice
- Leaf nodes denote the result of a combination of decisions

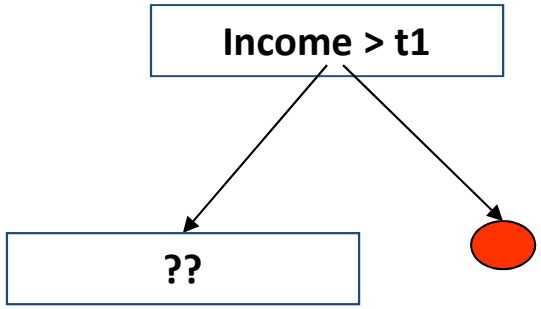
- Decision trees are built using a divide and conquer approach.

Decision Tree Example

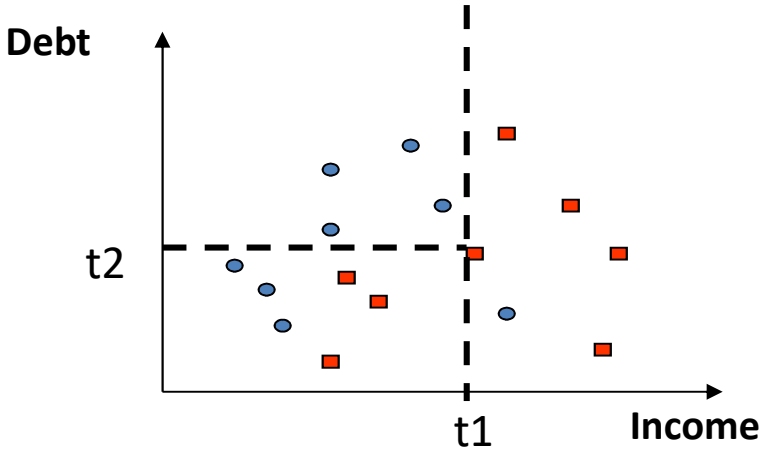


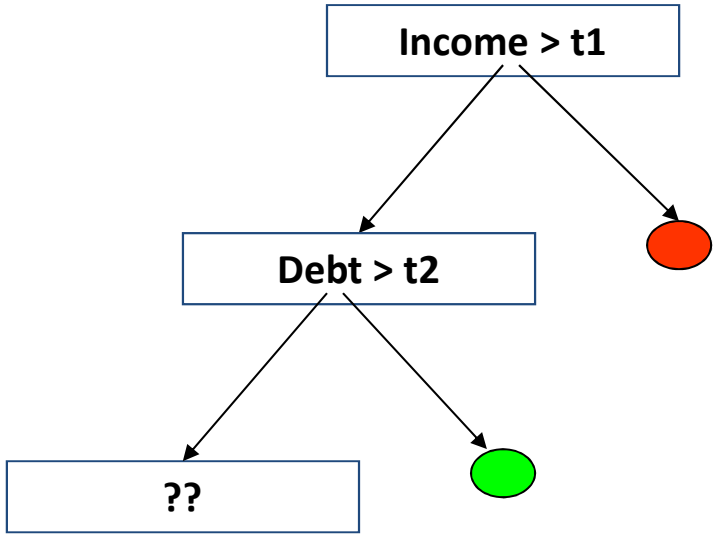
Decision Tree Example



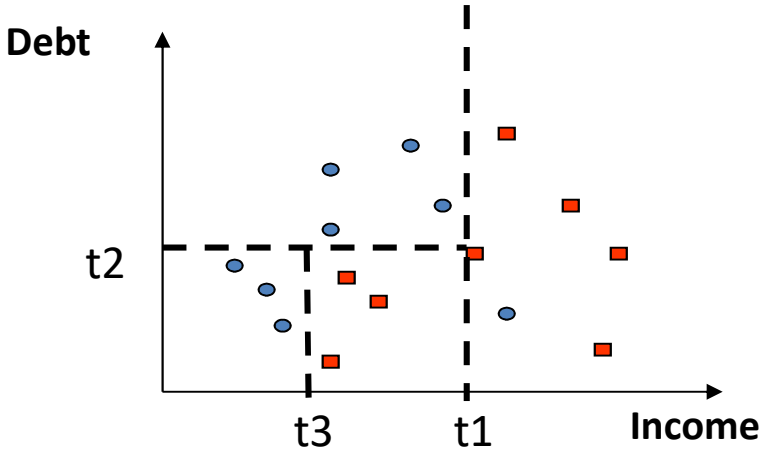


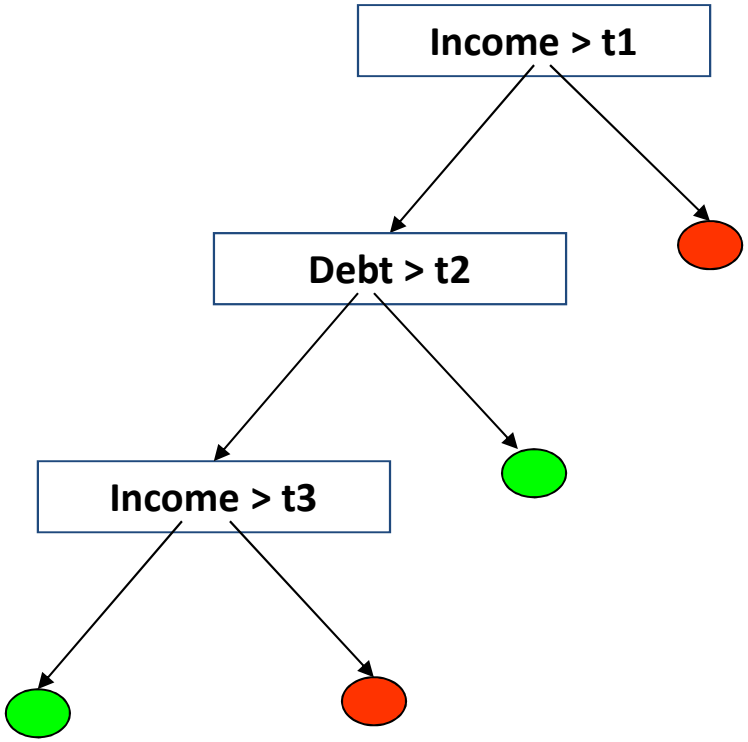
Decision Tree Example



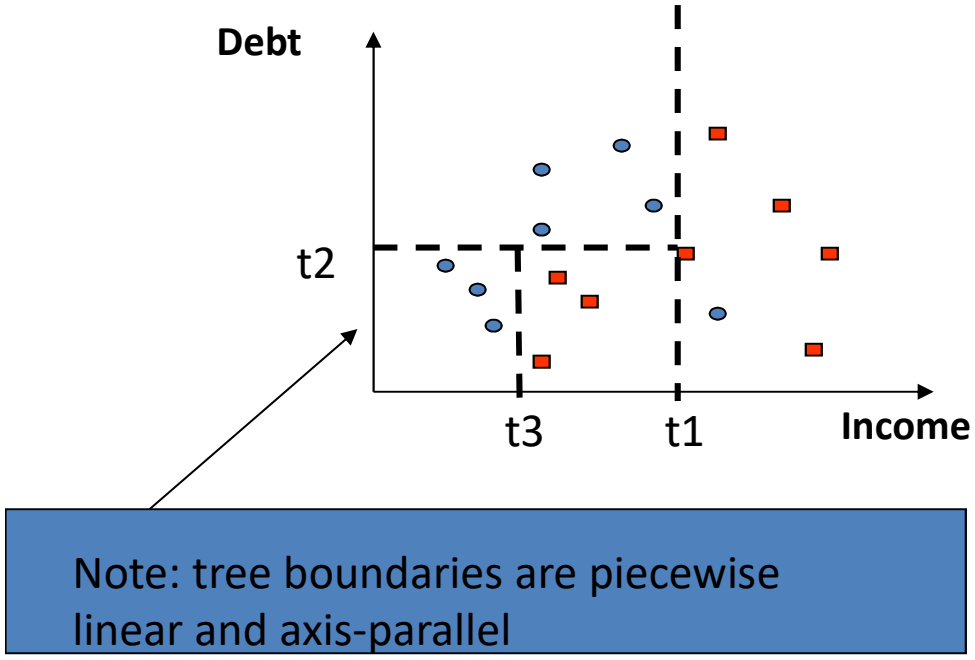


Decision Tree Example

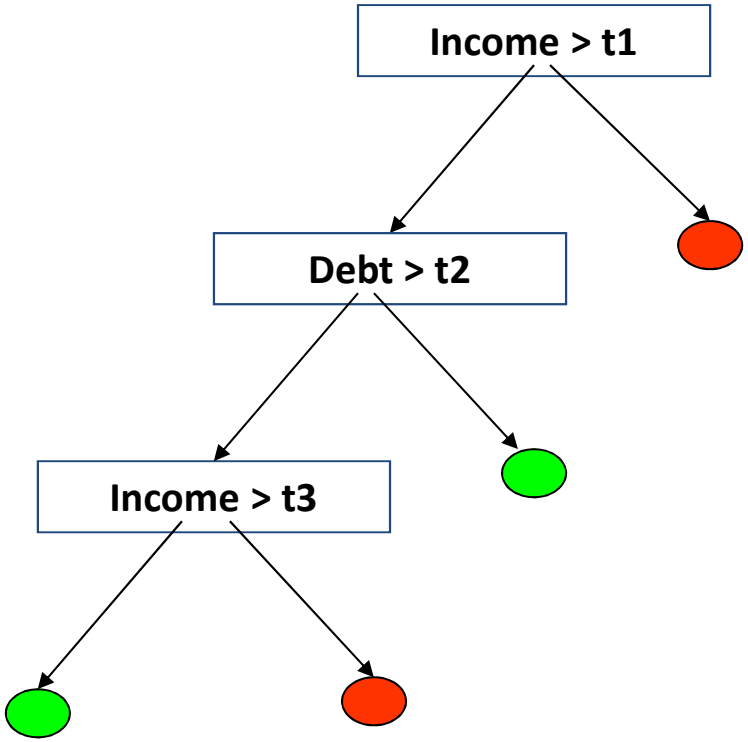




Decision Tree Example



Note: tree boundaries are piecewise linear and axis-parallel



Partitioning Up the Predictor Space

- One way to make predictions in a classification/regression problem is to divide the predictor space (i.e. all the possible values for for $X_1, X_2, \ldots, X_p$) into distinct regions, say $R_1, R_2, \ldots, R_k$

- Then for every X that falls in a particular region (say R_j) we make the same prediction

Some Natural Questions

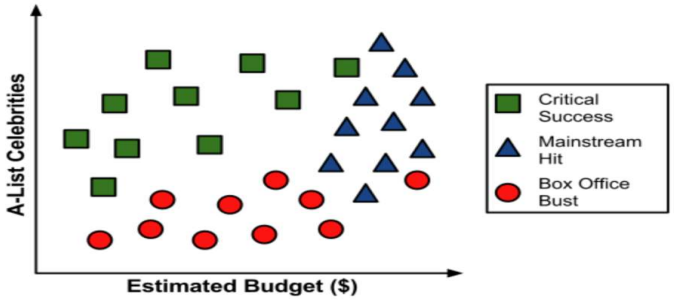
1. Where to split? i.e. how do we decide on what regions to use i.e. R_1, R_2,…,R_k or equivalently what tree structure should we use?

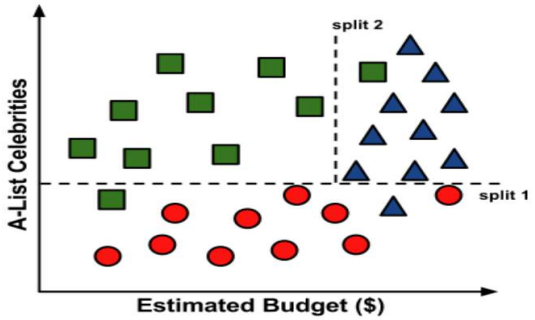
2. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

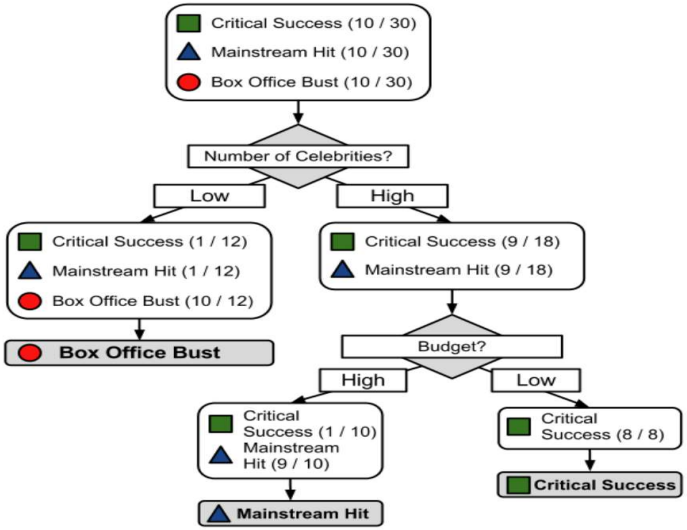
1. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

- Simple!
- For region R_j, the best prediction is simply
  - the most common category among the training data within that region (in case of classification)
  - the average of all the responses from our training data that fell in region R_j (in case of regression)

2. Where to Split?







look for feature values that split the data in such a way that partitions contain examples primarily of a single class.

C5.0 decision tree algorithm

- C5.0 uses entropy for measuring purity

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506
```

- Which feature to split?

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

the difference between the entropy before the split (*S1*) and after the split (*S2*):

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

the sum of entropy of each of the n partitions weighted by its proportion (w_i).

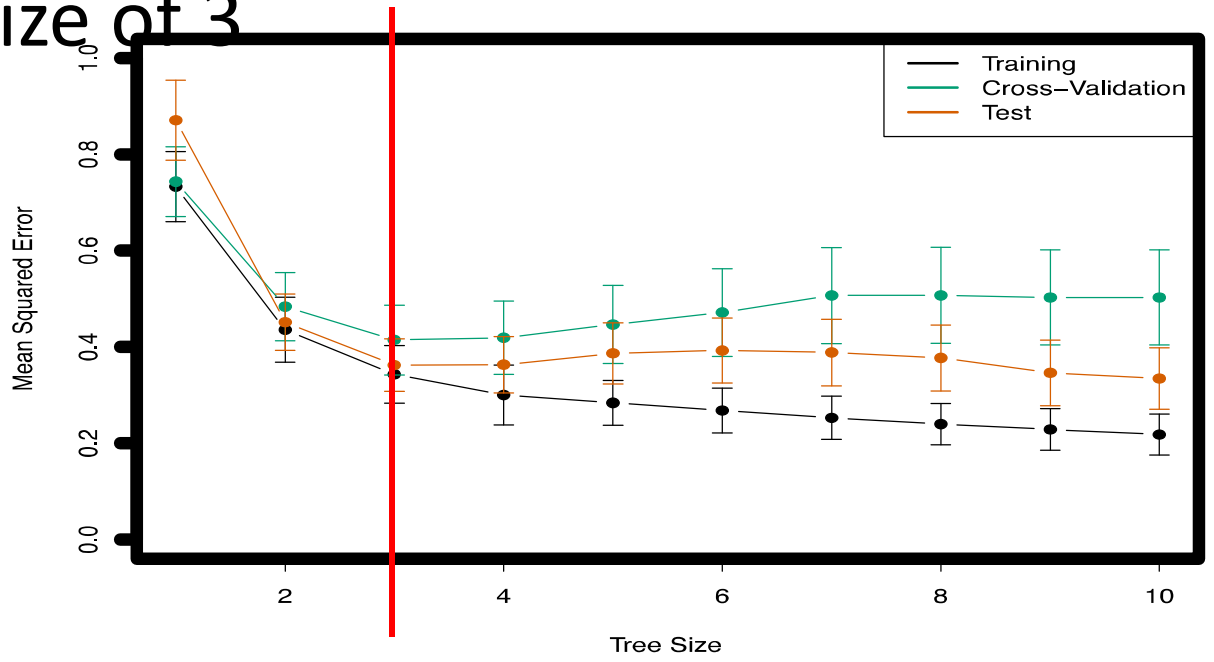
  - The higher the information gain is, the better a feature is at creating homogeneous groups

Improving Tree Accuracy

- A large tree (i.e. one with many terminal nodes) may tend to over fit the training data
- Generally, we can improve accuracy by “pruning” the tree i.e. cutting off some of the terminal nodes.
- How do we know how far back to prune the tree? We use **cross validation** to see which tree has the lowest error rate.

Example: Baseball Players' Salaries

- The minimum (cross validation) error occurs at a tree size of 3



Why trees are useful in Practice

- Can handle high dimensional data
  - Builds a model using 1 dimension at a time
  - But do not scale well to massive data sets (e.g. N in millions)
    - repeated access of subsets of the data
- Can handle any type of input variables
  - Most other methods require data of a single type
- Trees are (somewhat) interpretable
  - Trees are very easy to explain to people (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert

Issues with Decision Trees

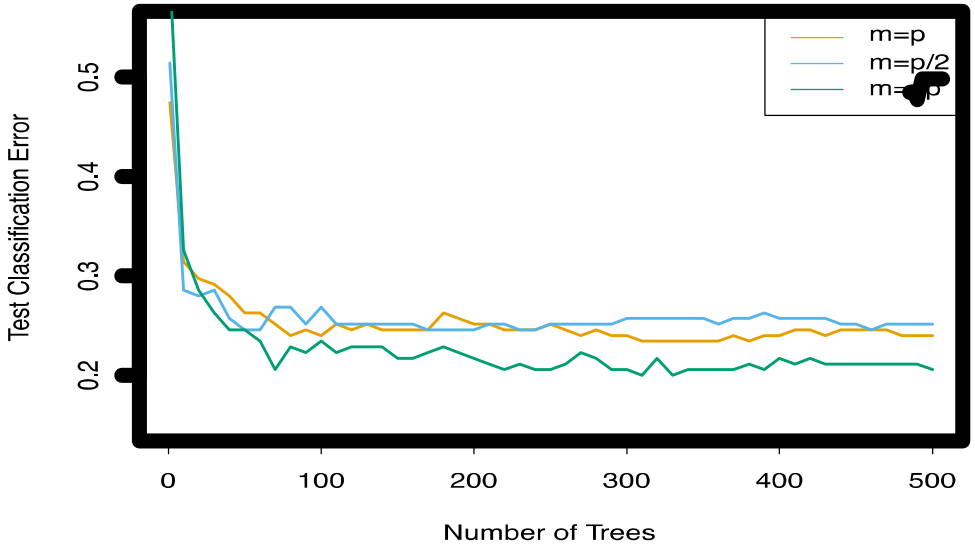
- Binary splits?
  - Multiway splits can be used, but cause fragmentation
- Linear combination splits?
  - May produce small improvements
  - Optimization is much more difficult (how to get the weights, etc.)
  - Much harder to interpret
- Model instability
  - A small change in the data can lead to a completely different tree

Random Forests

- Ensemble methods: by combining multiple weaker learners, a stronger learner is created

- Random forests: *decision tree forests*
  - Build a number of decision trees on *bootstrapped* training samples (:repeat resampling the observed dataset, each of which is obtained by random sampling with replacement from the original dataset)
  - but when building these trees, a random sample of *m* features is chosen from the full set of *p* features (usually $m \approx \sqrt{p}$)

Random Forest with different values of “m”



Random Forests

- Strengths
  - An all-purpose model that performs well on most problems
  - Can handle noisy or missing data; categorical or continuous features
  - Selects only the most important features
  - Can be used on data with an extremely large number of features or examples

- Weaknesses
  - Unlike a decision tree, the model is not easily interpretable
  - May require some work to tune the model to the data

Other classification algorithms

- Support Vector Machine
- Neural network and deep learning
- Will be discussed later