Machine Learning - Classification

Week 9 – Part 1 – Introduction to Classification

CS 457 - L1 Data Science

Zeehasham Rasheed

Chapter Objectives

- Define and describe the Classification process.
- Define and describe Classification techniques.

Classification: Definition

- Given a collection of records (training data)
- Each record contains a set of attributes (x), with one additional attribute which is the class/label (y)
- Training a model means to create a model from given data to predict the class (y) as a function of the values of attributes (x)
- Goal is to classify unseen/unknown/new records with no class/labels (y) but only contains attributes (x), as accurately as possible.
- For the evaluation of model, Test set is used to determine the accuracy of the model. Usually, the given data set is divided into Training and Test sets, with training set used to build the model and test set used to validate it.

Classification (Supervised)

- Classification is also called supervised learning because it uses known examples for learning.
-
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervised).

Applications (1)

- An emergency room in a hospital records 17 different attributes (for example blood pressure, age, weight etc.) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive care unit (ICU) or not.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Goal: to predict high-risk patients and separate them from low-risk patients.

Applications (2)

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant.
  - age
  - marital status
  - annual salary
  - outstanding debts
  - credit rating etc.
- Goal: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

Applications (3)

- A retailer wants to introduce new product in different regions. The past sales records for similar product contains the following information
  - Competitor's price
  - Advertising budget
  - Rating
  - Region population etc.
- Goal: to decide whether to launch a product in a specific region or not. The prediction can be profitable and not profitable.

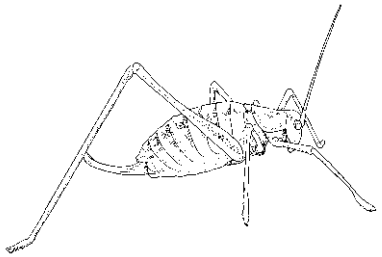
Classification Techniques

- There are many techniques/algorithms for performing classification.

- Commonly used techniques are
  - Nearest Neighbor
  - Decision Trees
  - Random Forests
  - Logistic Regression

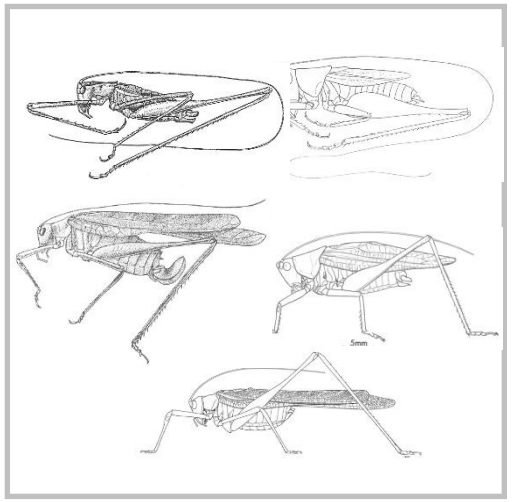
The Classification Problem (informal definition)

Given a collection of annotated data.
In this case 5 instances of **Katydids** and five of **Grasshoppers**, decide what type of insect the unlabeled example is.

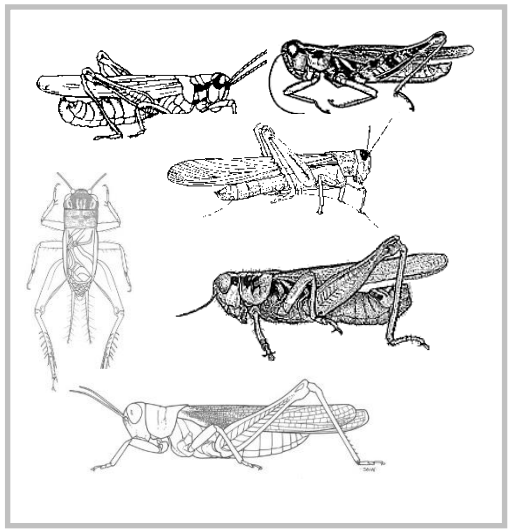


Katydid or **Grasshopper**?

Katydids



Grasshoppers

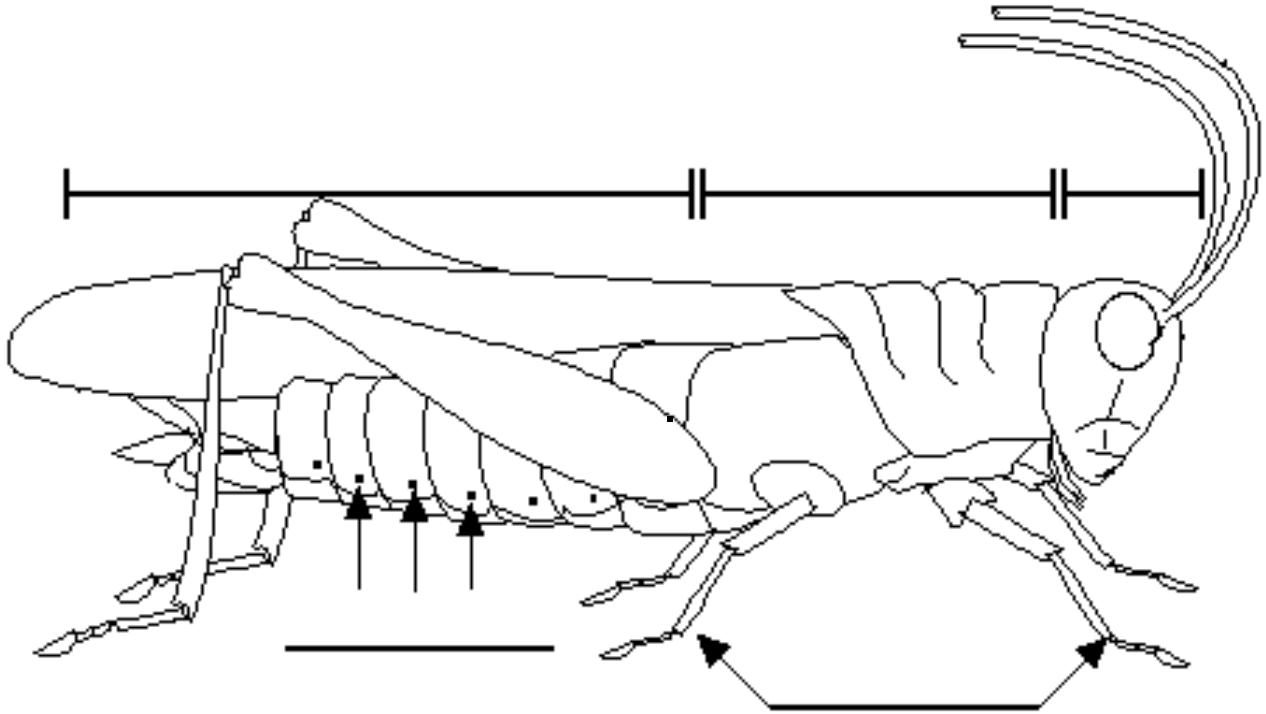


Features/Attributes

For any domain of interest, we can measure *features*

Color {**Green, Brown, Gray, Other**}

Has Wings?



Antennae Length

Mandible Size

Spiracle Diameter

Leg Length

Classification Dataset

We can store features in a database.

The classification problem can now be expressed as:

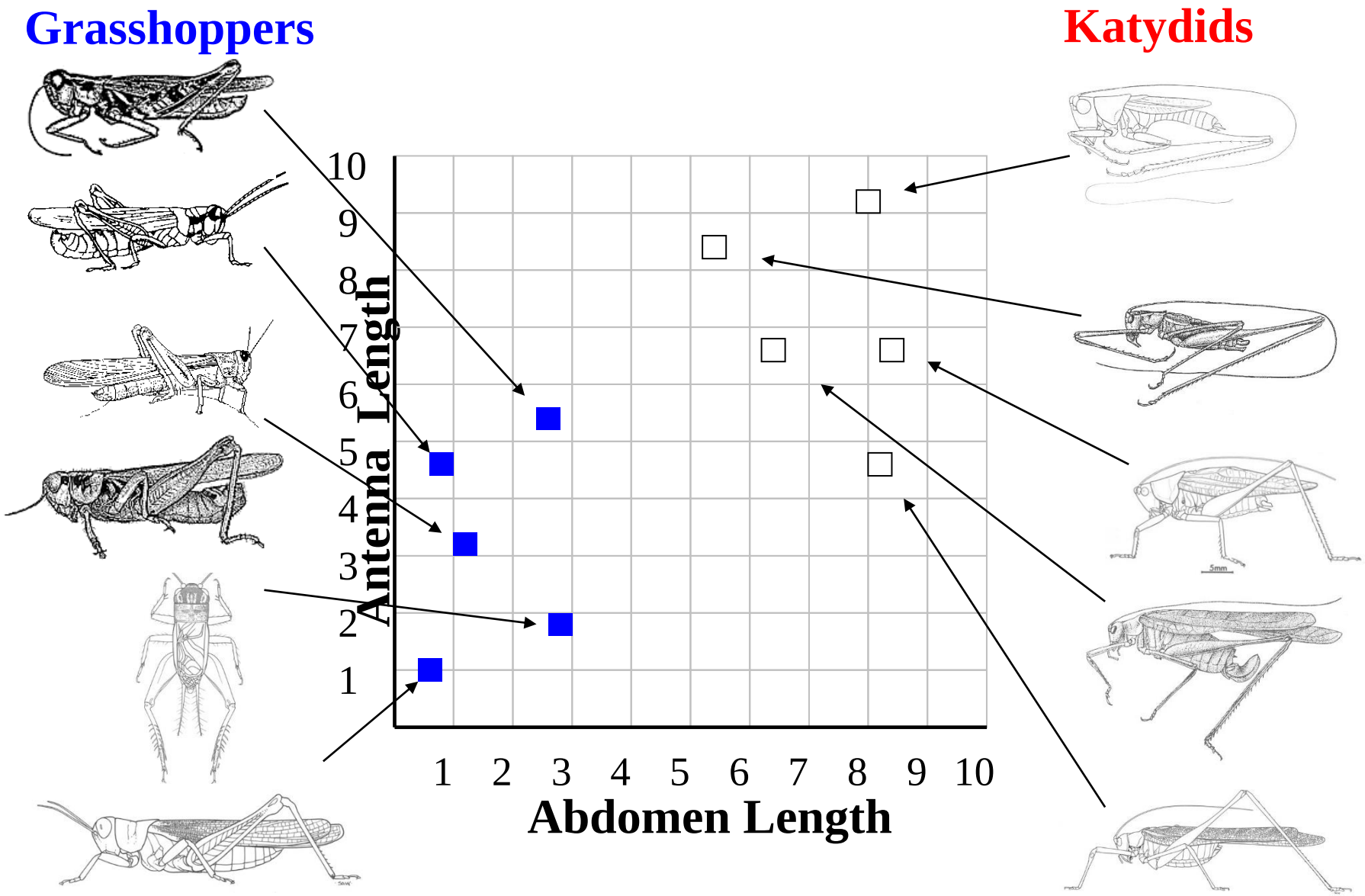
Given a training database (**My_Collection**), predict the **class** label of a previously unseen instance

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	**Grasshopper**
2	8.0	9.1	**Katydid**
3	0.9	4.7	**Grasshopper**
4	1.1	3.1	**Grasshopper**
5	5.4	8.5	**Katydid**
6	2.9	1.9	**Grasshopper**
7	6.1	6.6	**Katydid**
8	0.5	1.0	**Grasshopper**
9	8.3	6.6	**Katydid**
10	8.1	4.7	**Katydids**

previously unseen instance =

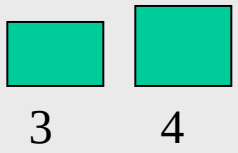
11	5.1	7.0	??????

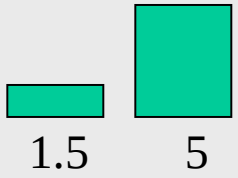
Visualizing Classification Data

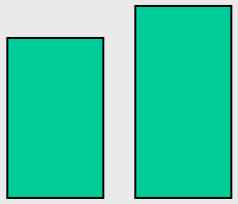


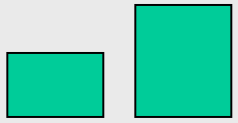
Pigeon Problem 1

Examples of class A

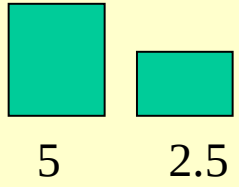


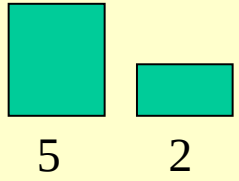


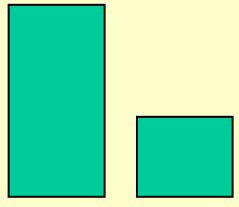


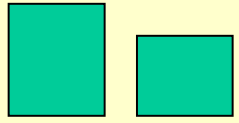


Examples of class B



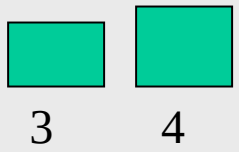


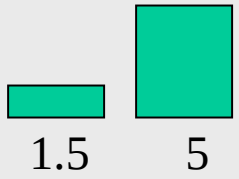


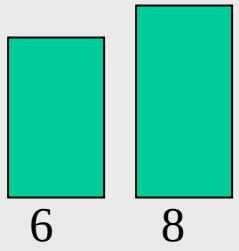


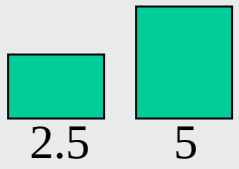
Pigeon Problem 1

Examples of class A

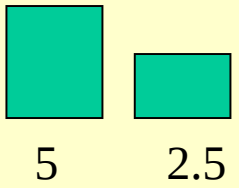


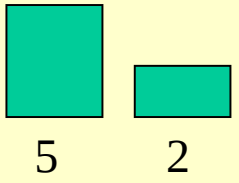


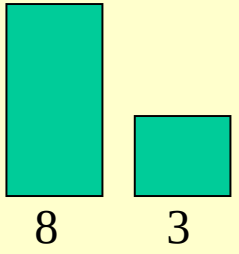


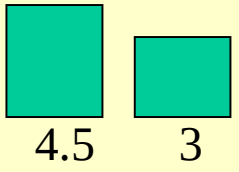


Examples of class B

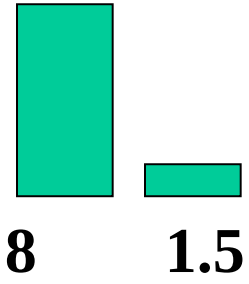


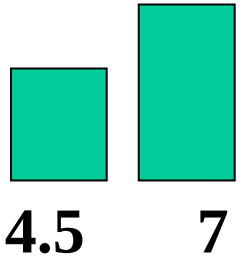






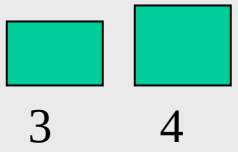
What class is this object?

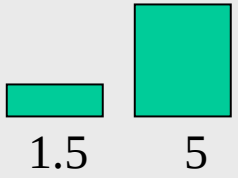


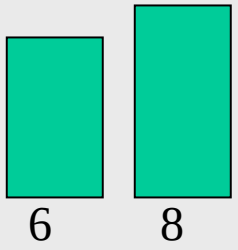


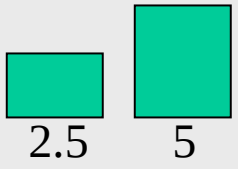
Pigeon Problem 1

Examples of class A

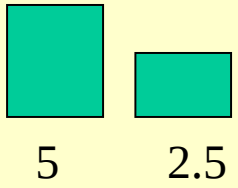


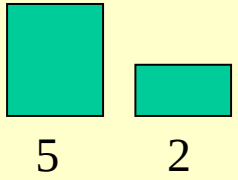


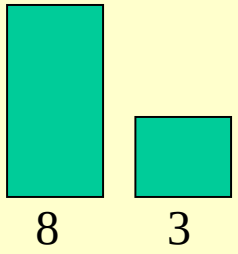


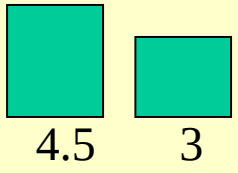


Examples of class B

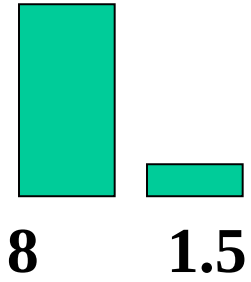


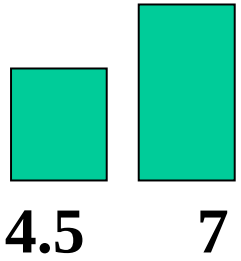






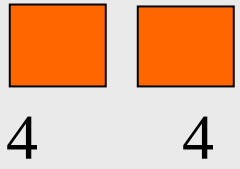
Here is the rule.
If the left bar is smaller than the right bar, it is an A, otherwise it is a B.

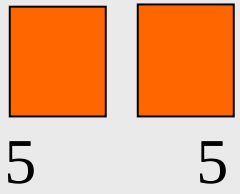


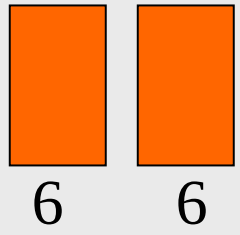


Pigeon Problem 2

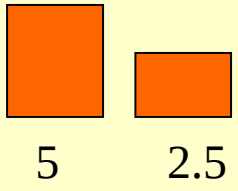
Examples of class A

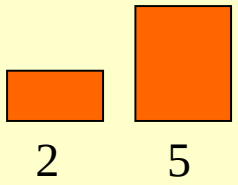


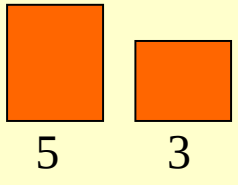


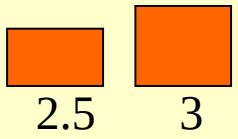


Examples of class B

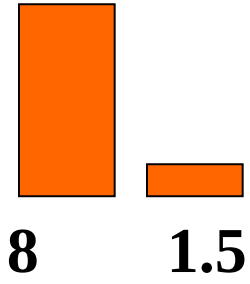


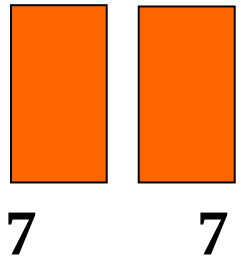






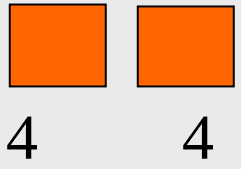
What class is this object?

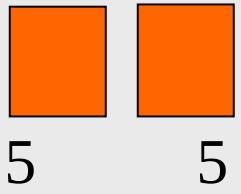


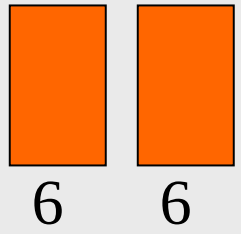


Pigeon Problem 2

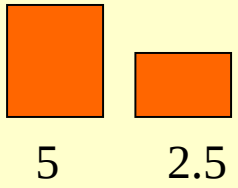
Examples of class A

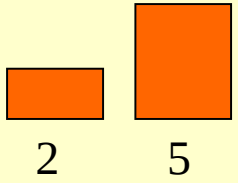


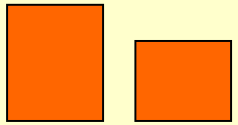


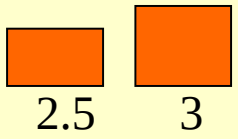


Examples of class B

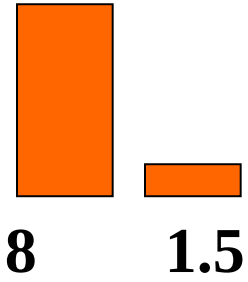


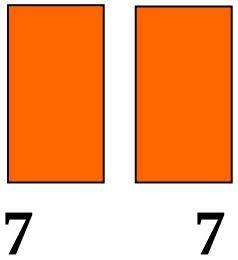






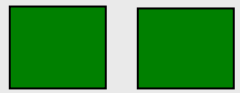
The rule is as follows, if the two bars are equal sizes, it is an A. Otherwise it is a B.

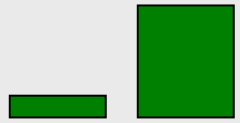


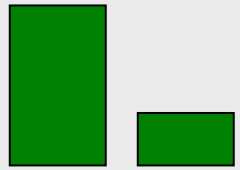


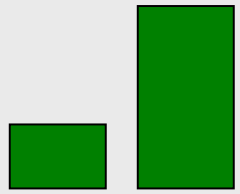
Pigeon Problem 3

Examples of class A

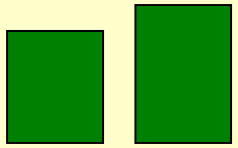

4 4

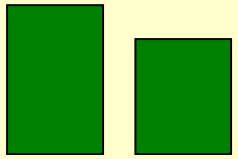

1 5

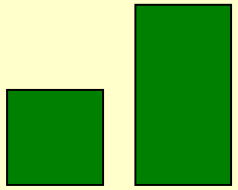

6 3

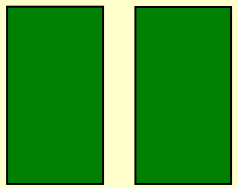

3 7

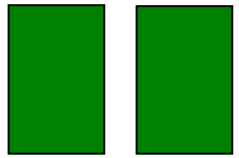
Examples of class B


5 6


7 5

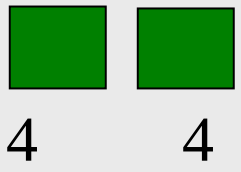

4 8

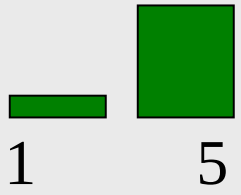

7 7

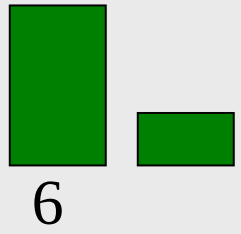

6 6

Pigeon Problem 3

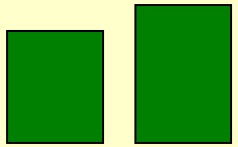
Examples of class A

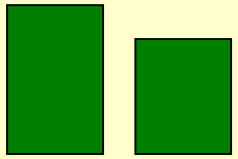


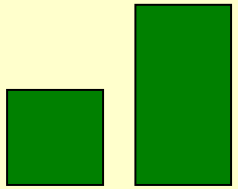


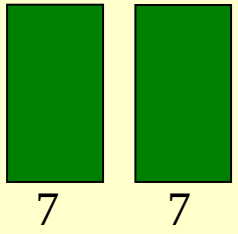


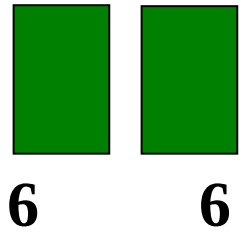
Examples of class B











The rule is as follows, if the square of the sum of the two bars is less than or equal to 100, it is an A. Otherwise it is a B.

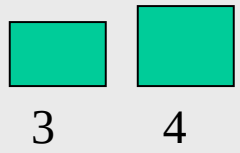
Why Pigeon Problem?

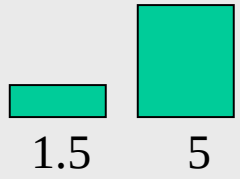
- Why did we spend so much time with this game?

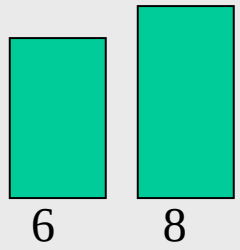
- Because we wanted to show that **almost all classification problems have a geometric interpretation**

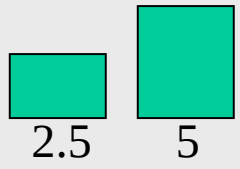
Pigeon Problem 1 Geometric Visual

Examples of class A

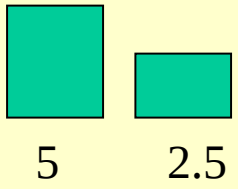


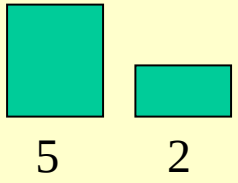


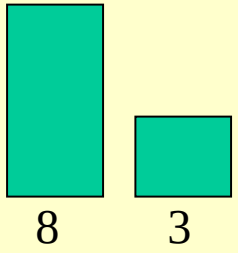


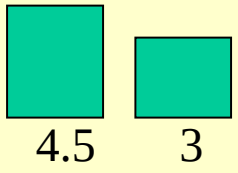


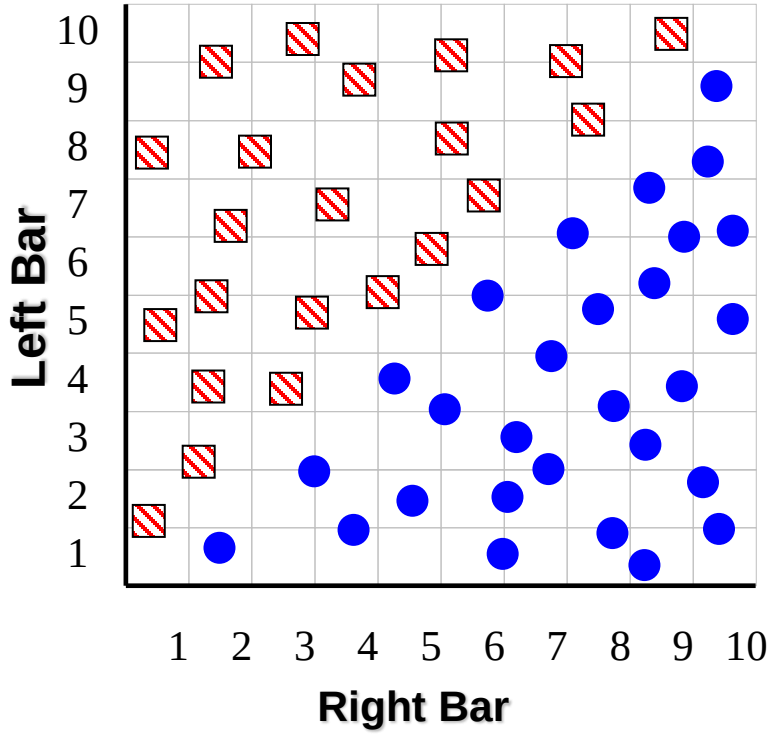
Examples of class B





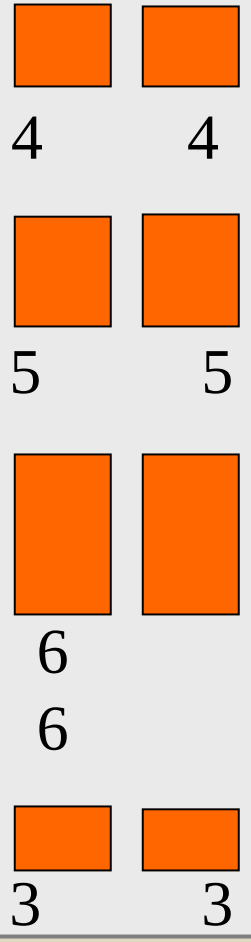




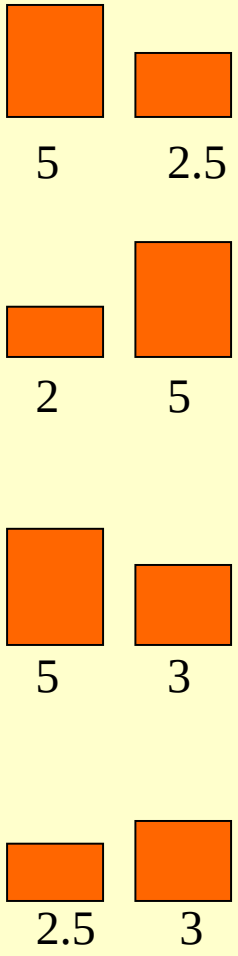


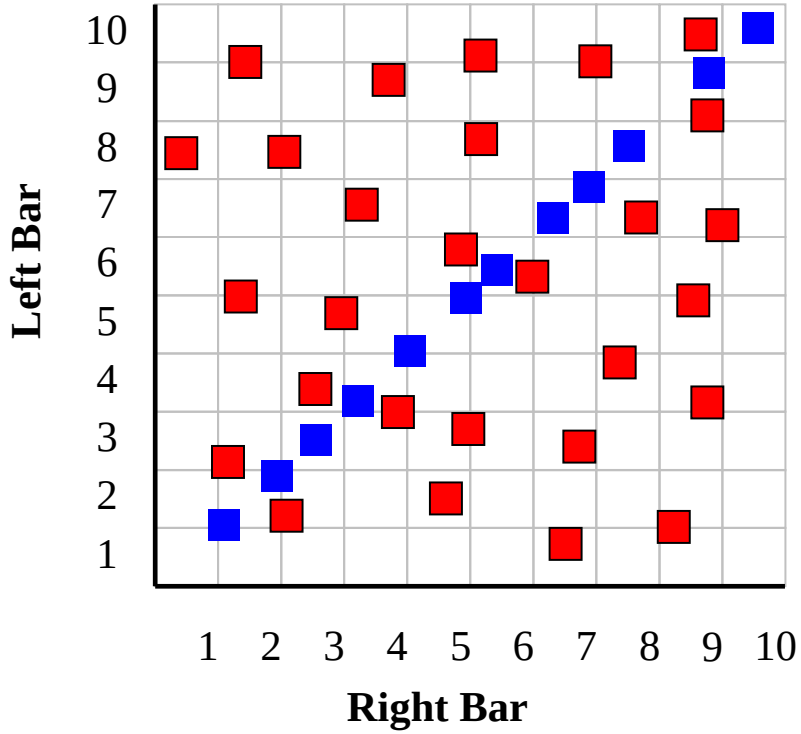
Pigeon Problem 2 Geometric Visual

Examples of class A

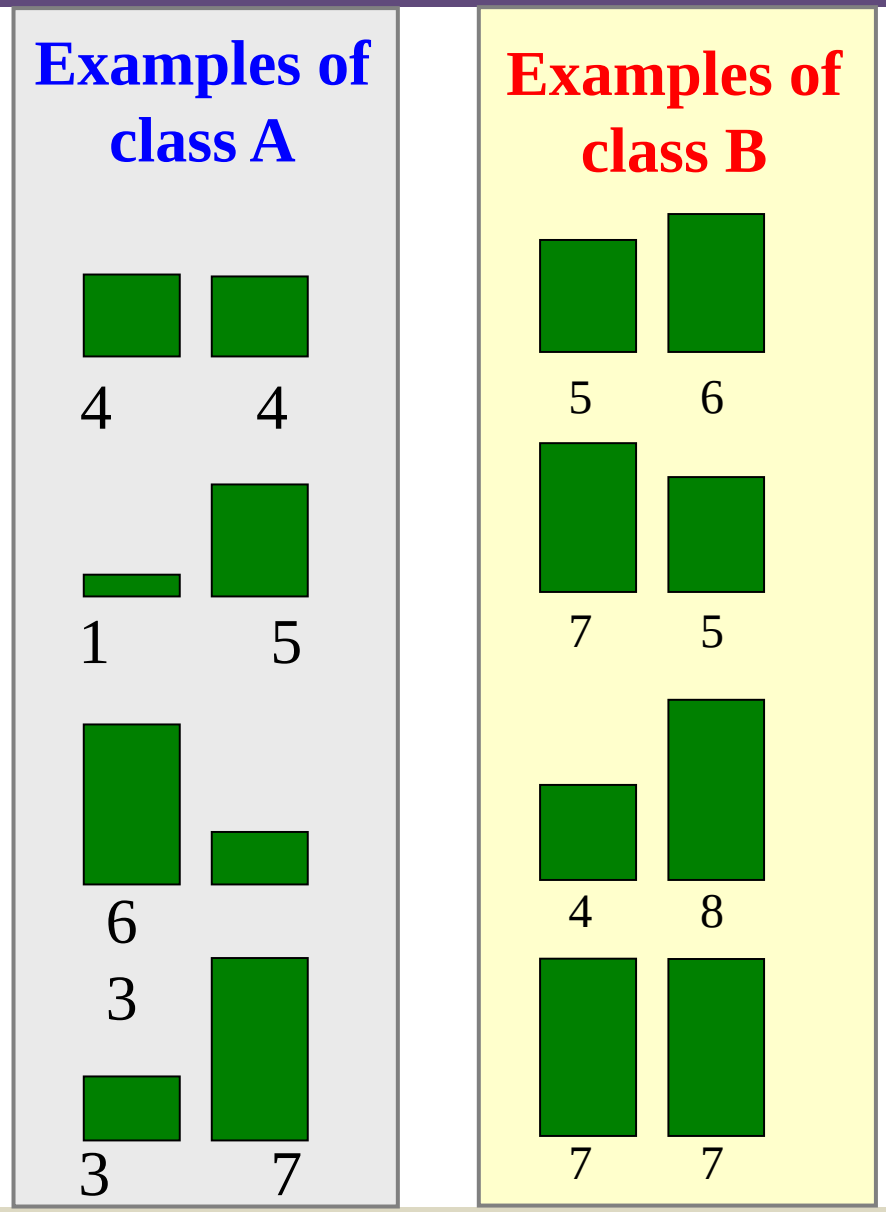


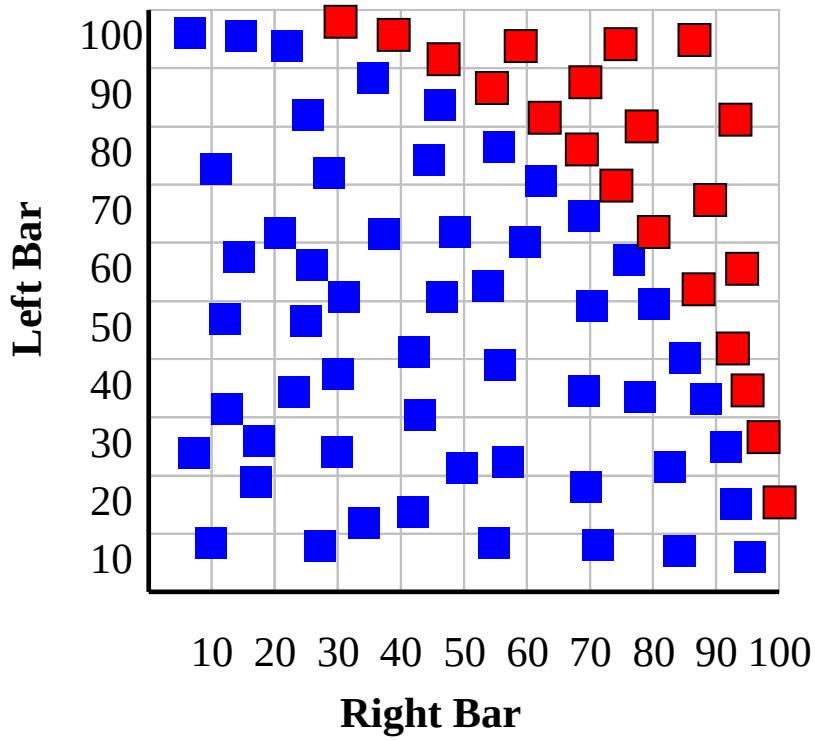
Examples of class B





Pigeon Problem 3 Geometric Visual





End of Part 1

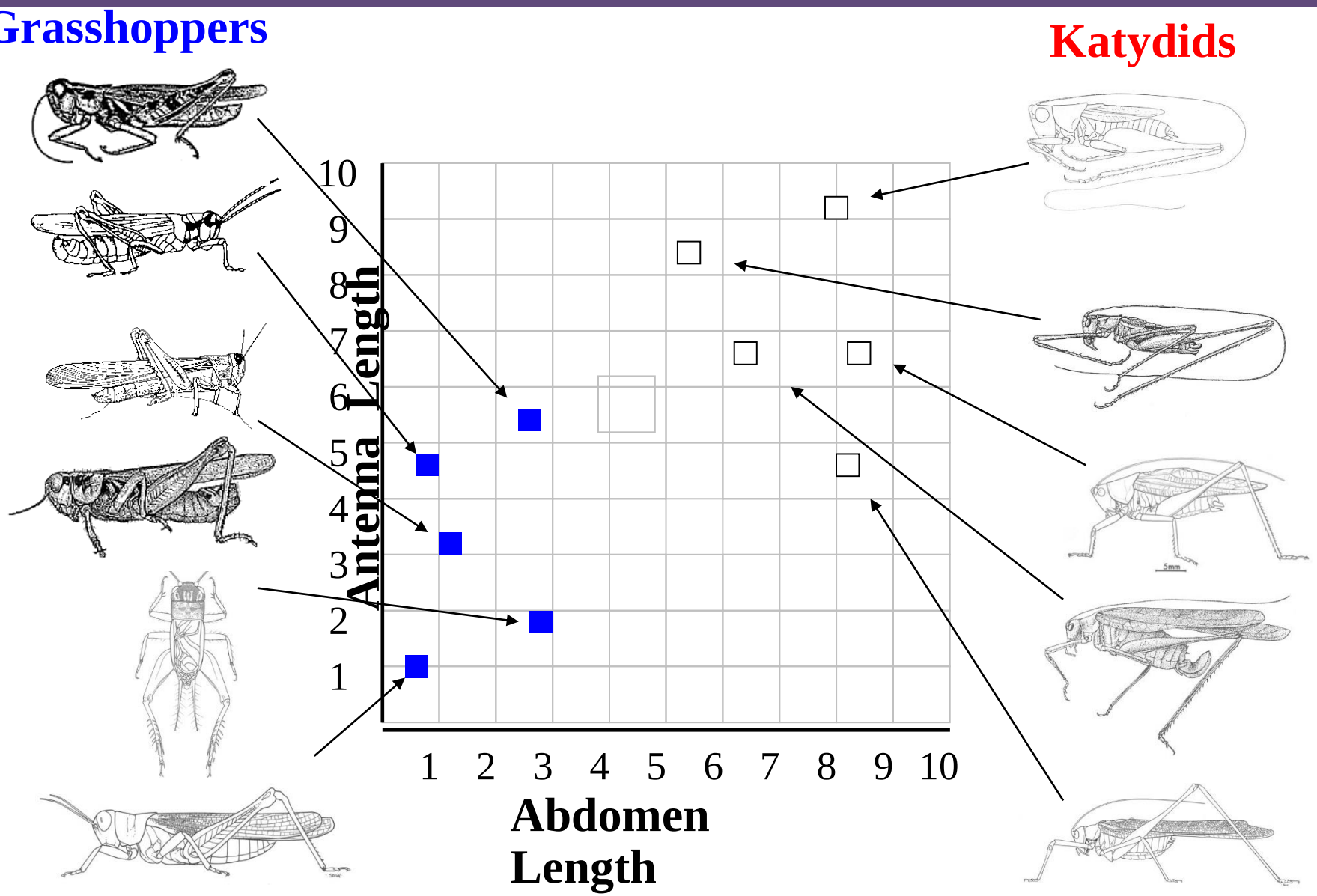
Machine Learning - Classification

Week 9 – Part 2 – Linear Classifiers

CS 457 - L1 Data Science

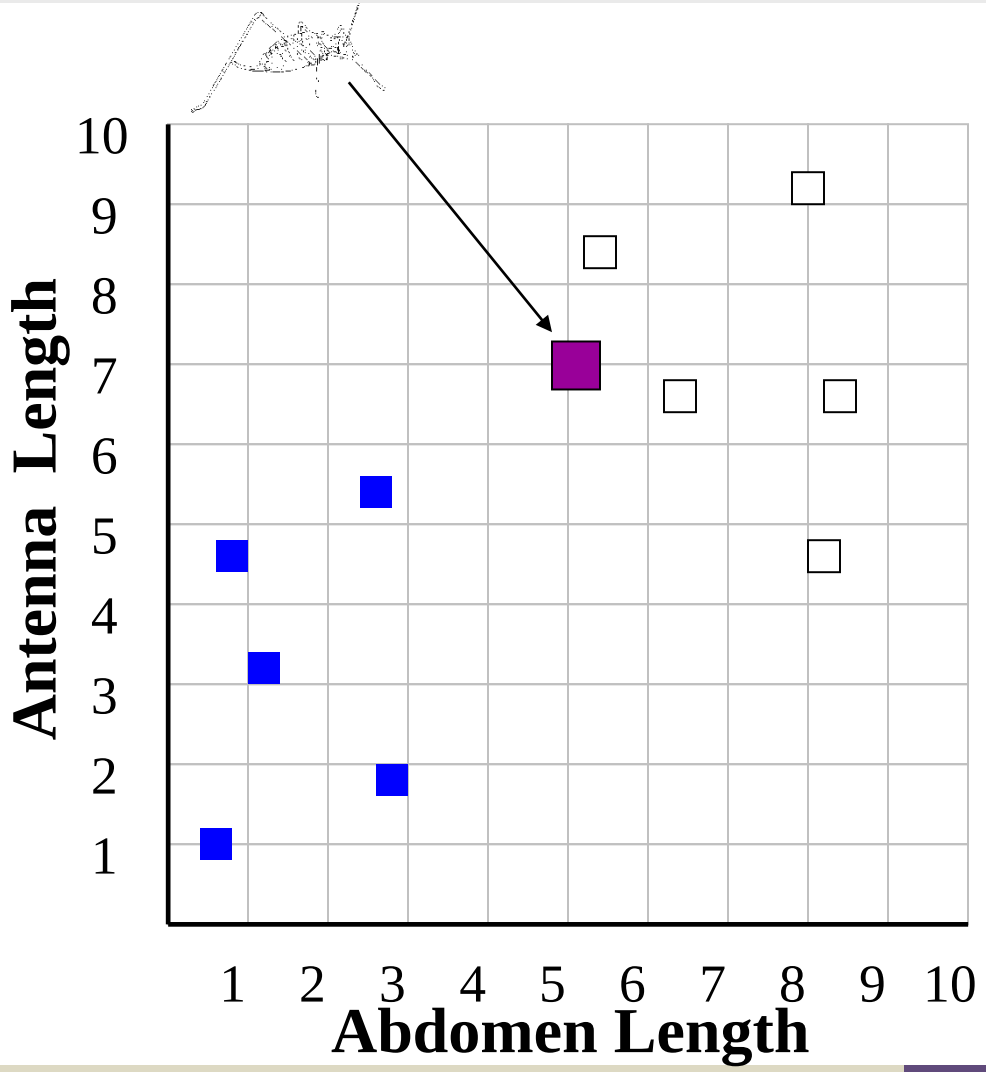
Zeehasham Rasheed

Geometric Visual of Classification Data



Geometric Visual for Unseen/New Data

previously unseen instance =	*11*	5.1	7.0	??????



We can "project" the previously unseen instance into the same space as the database.

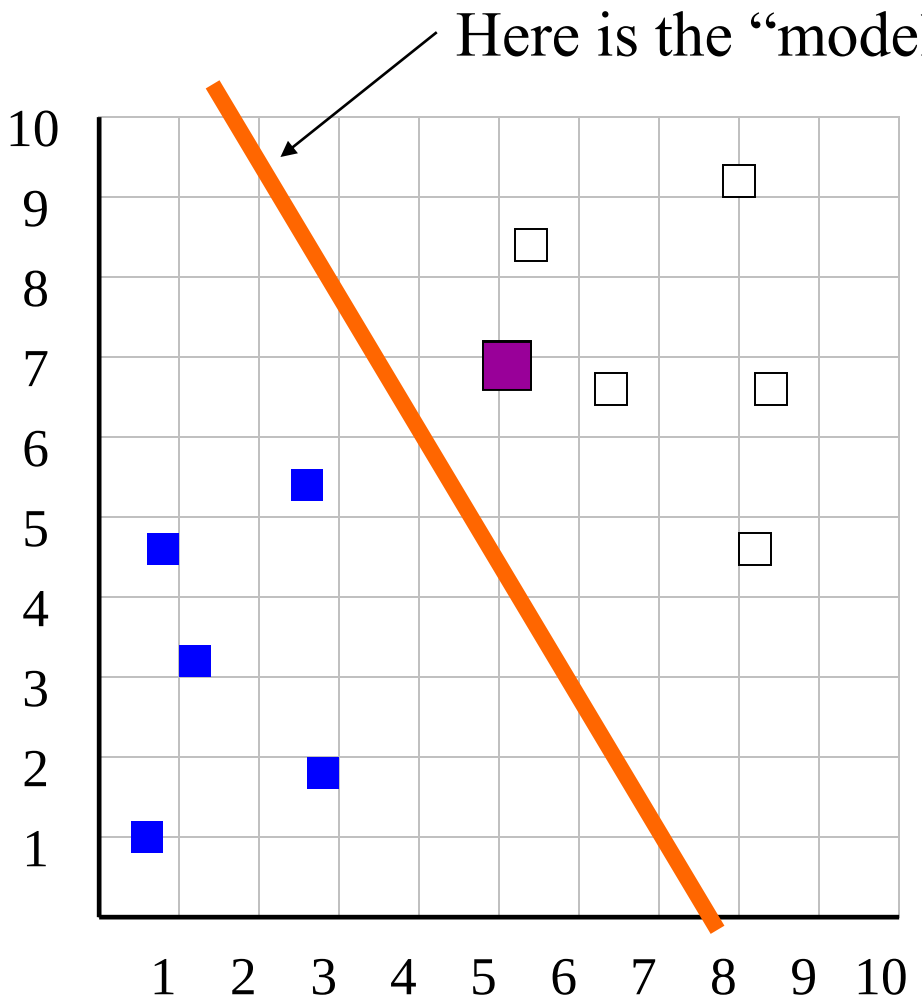
We have now abstracted away the details of our particular problem. It will be much easier to talk about points in space.

□ **Katydids**

■ **Grasshoppers**

Simple Linear Classifier

Linear Discriminant Analysis (LDA)



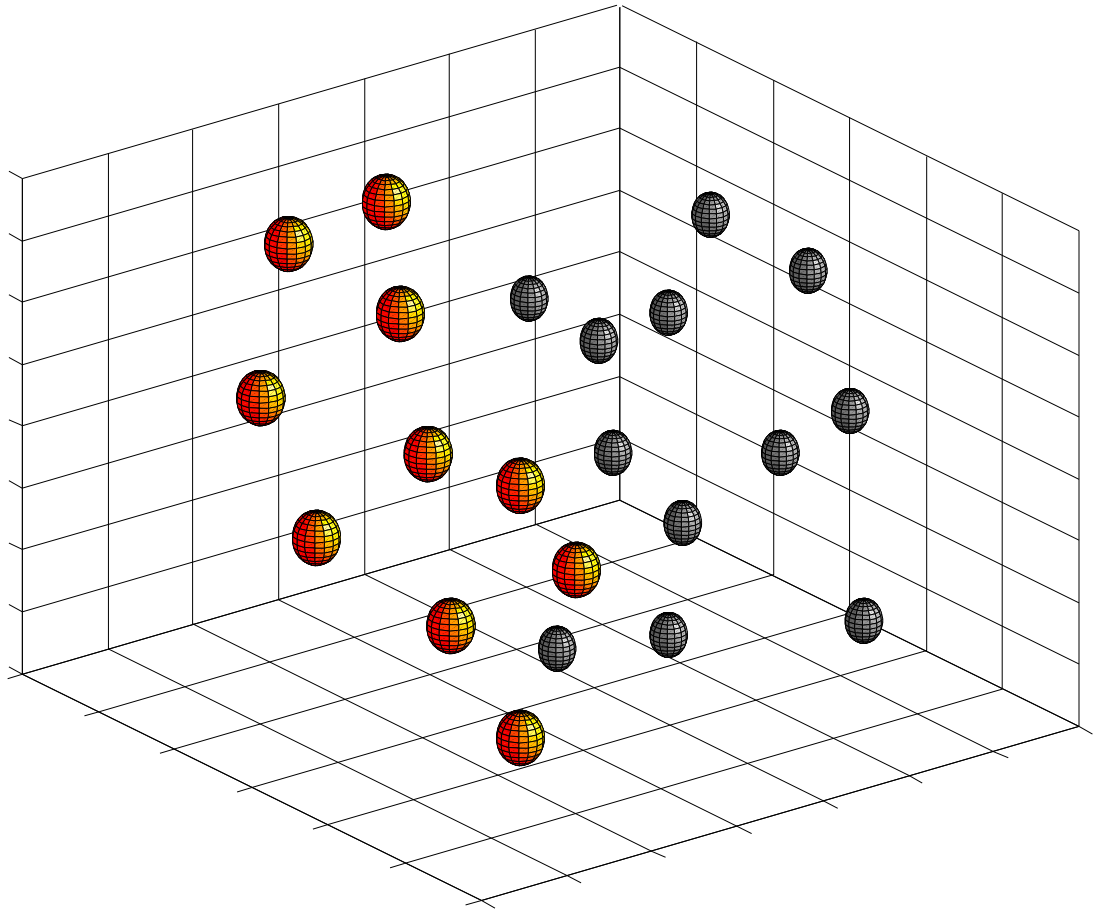
R.A. Fisher
1890-1962

If previously unseen instance above the line
then
class is **Katydid**
else
class is **Grasshopper**

□ **Katydids**
■ **Grasshoppers**

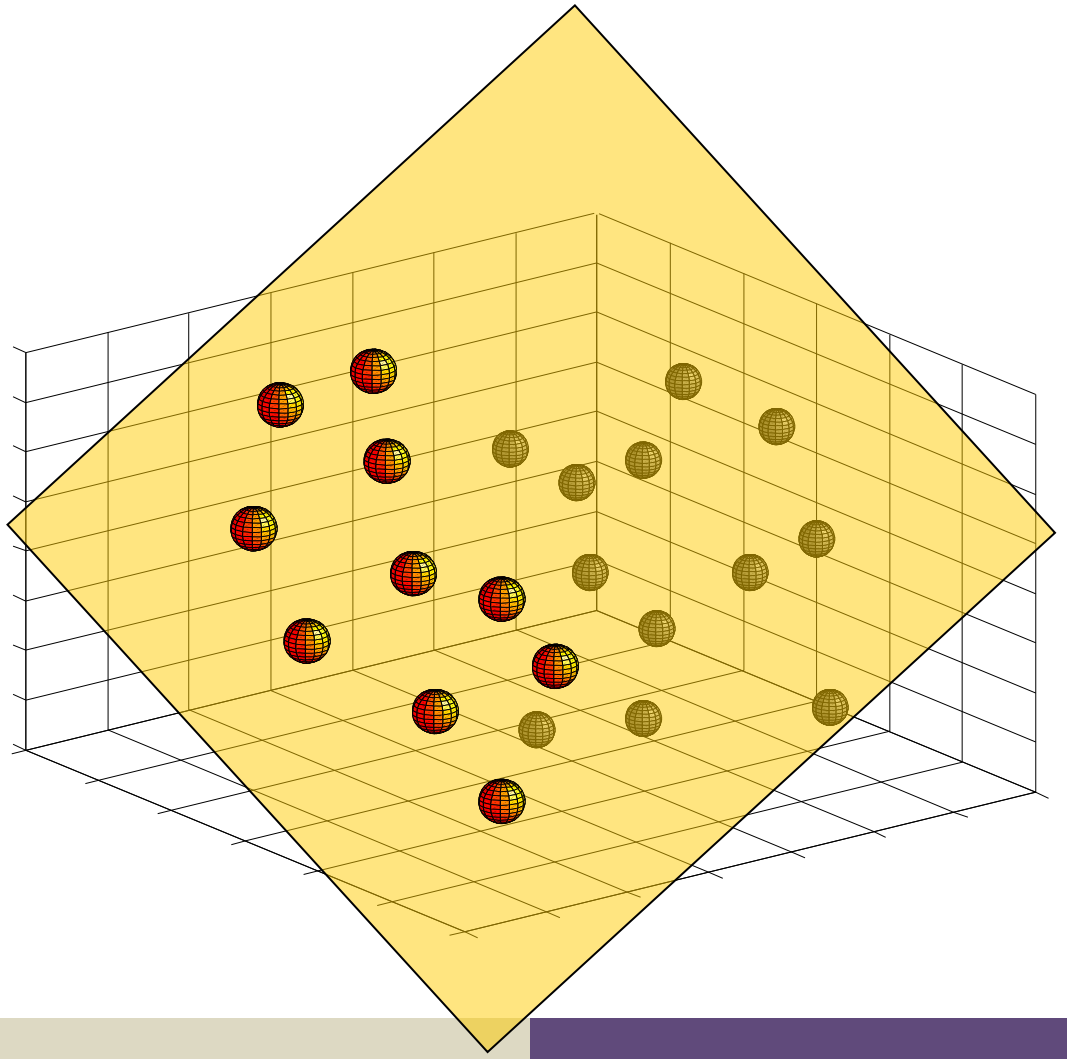
3 Dimension (3 Features)

- The simple linear classifier is defined for higher dimensional spaces



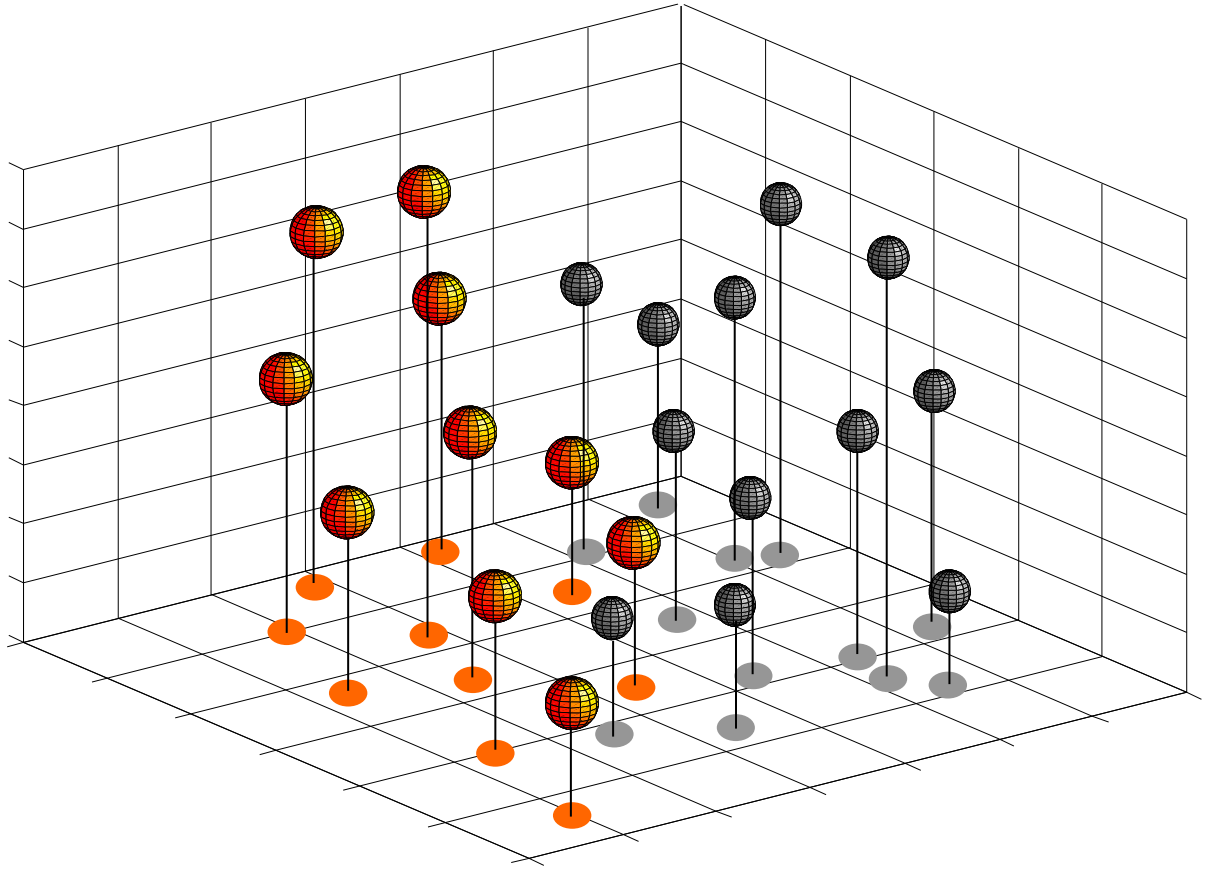
N Dimension (N Features)

- We can visualize it as being an n-dimensional hyperplane



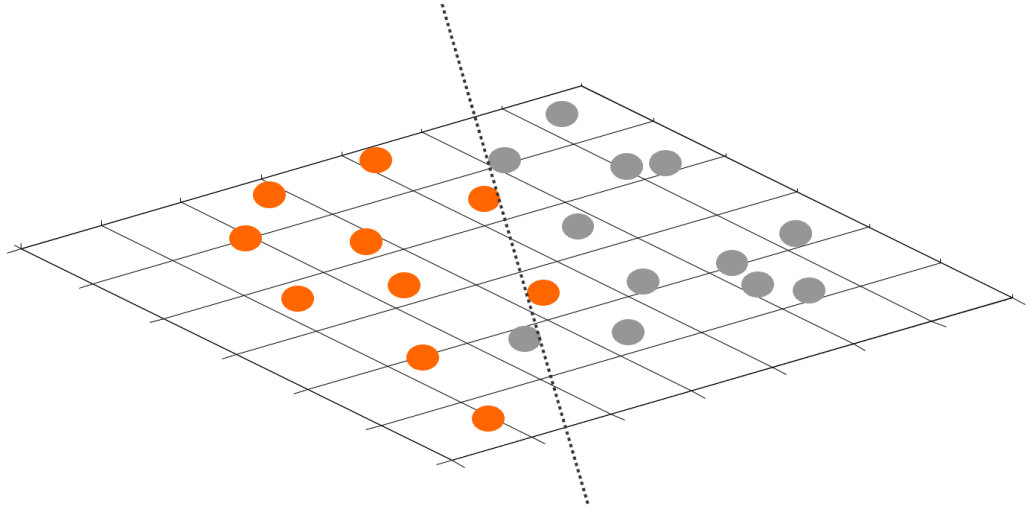
From 3 Dimension to 2 Dimension

- It is interesting to see this example in 2nd and 3rd dimensions

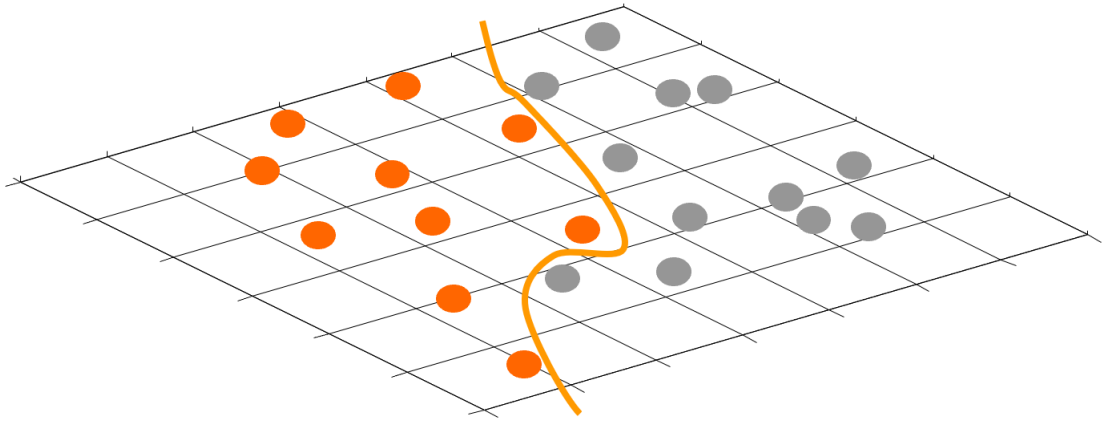


Linear vs Non-linear Classifier

We can no longer get perfect accuracy with the **simple linear classifier**



We could try to solve this problem by using a simple *quadratic* classifier or a **simple *cubic* classifier**

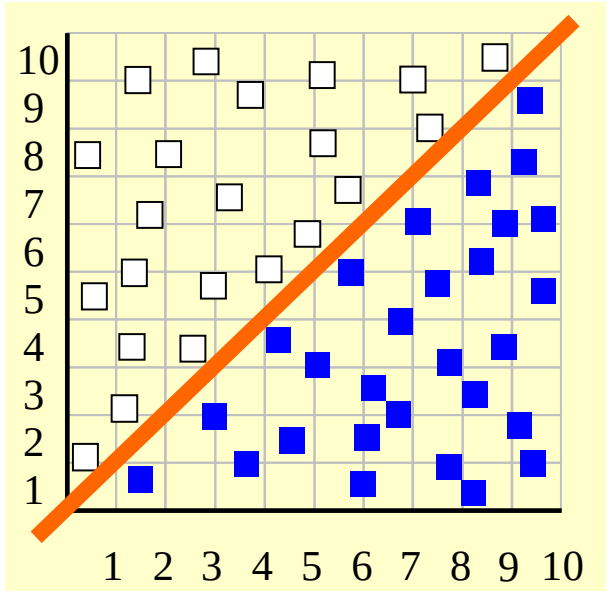


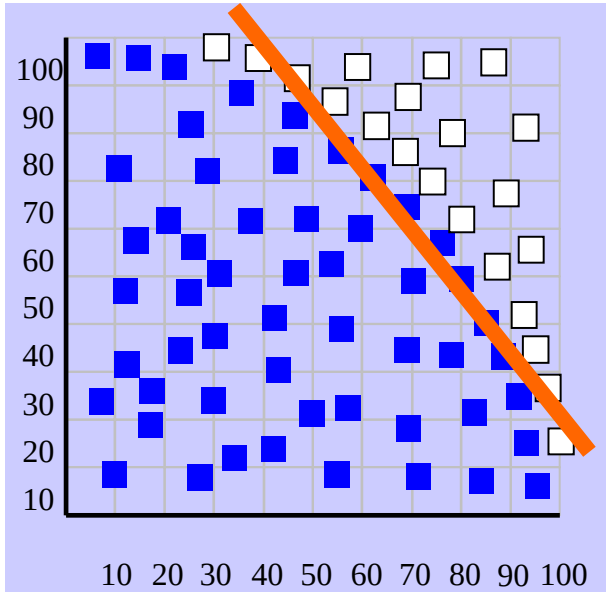
Having complex non-linear model is not a good idea to start with (not good for unseen/new data

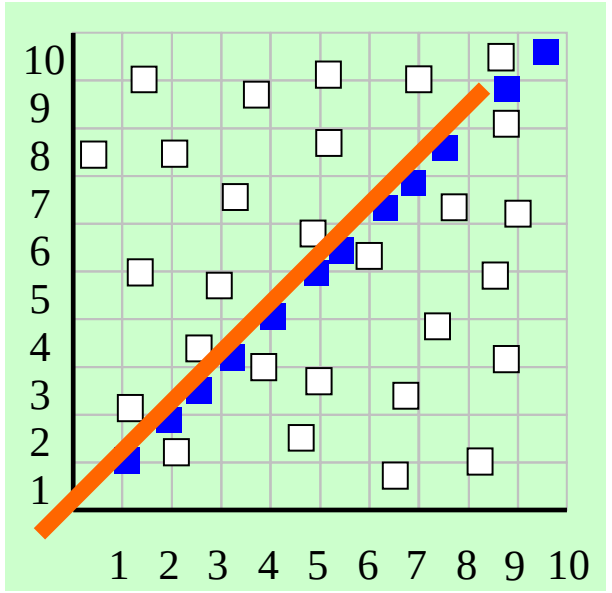
Visualizing Classifier Performance

Which of the "Pigeon Problems" can be solved by the Simple Linear Classifier?

1)Perfect
2)Useless
3)Pretty Good







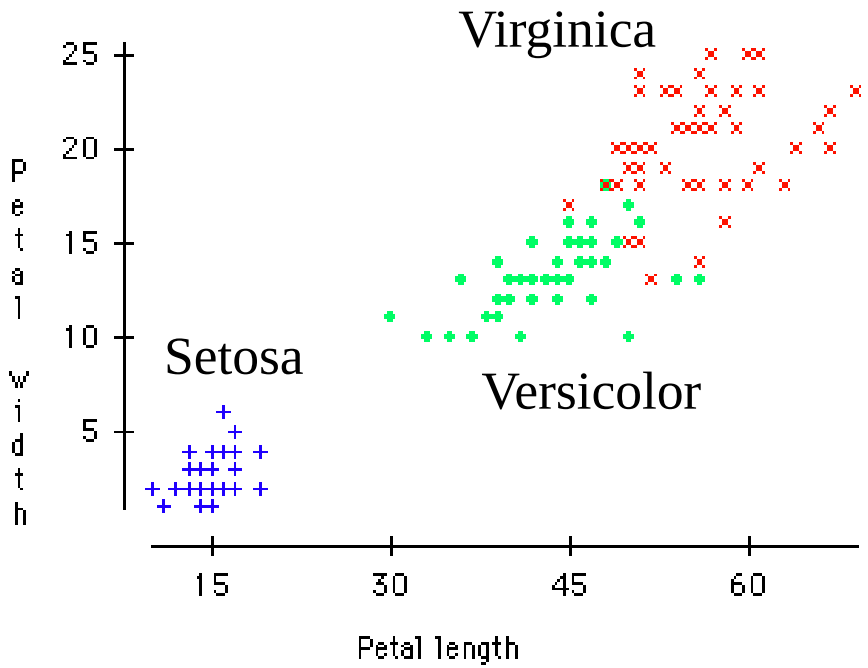
Problems that can be solved by a linear classifier are called **linearly separable**.

R. A. Fisher's Iris Dataset

A Famous Problem

3 classes
50 records of each class

The task is to classify Iris plants into one of 3 varieties using the Petal Length and Petal Width.



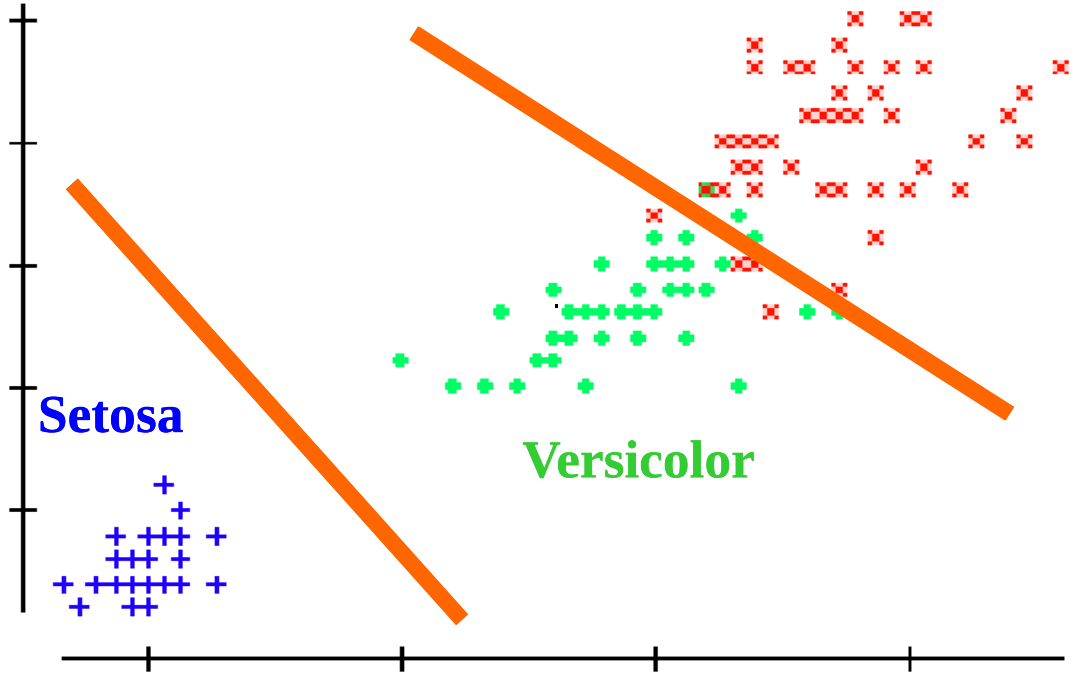
Iris Setosa

Iris Versicolor

Iris Virginica

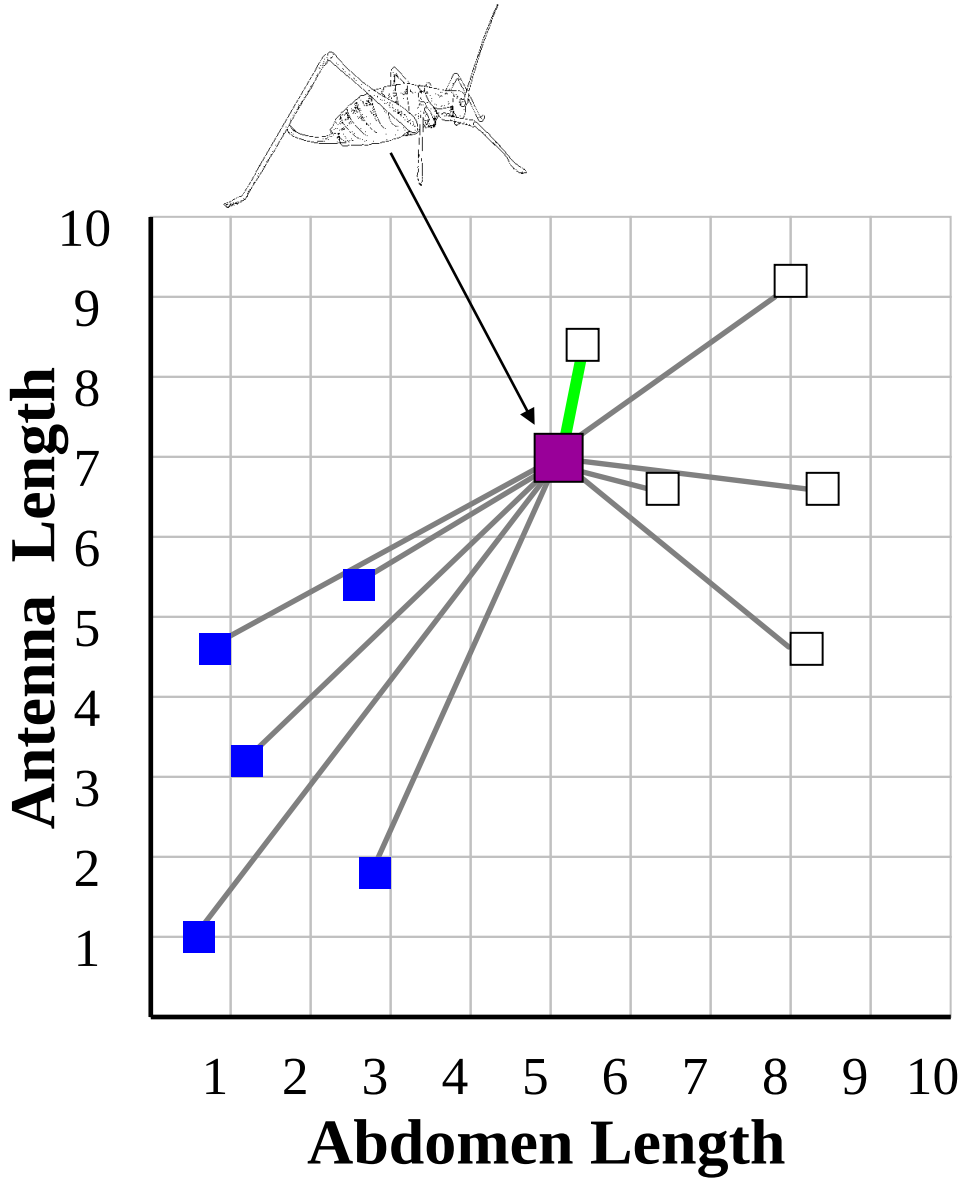
N Class Classification

We can generalize the piecewise linear classifier to **N classes**, **by fitting N-1 lines**. In this case we first learned the line to (perfectly) discriminate between **Setosa** and **Virginica**/**Versicolor**, then we learned to approximately discriminate between **Virginica** and **Versicolor**.



If petal width > 3.272 – (0.325 * petal length) **then** class = **Virginica Else if** petal width…

Nearest Neighbor Classifier

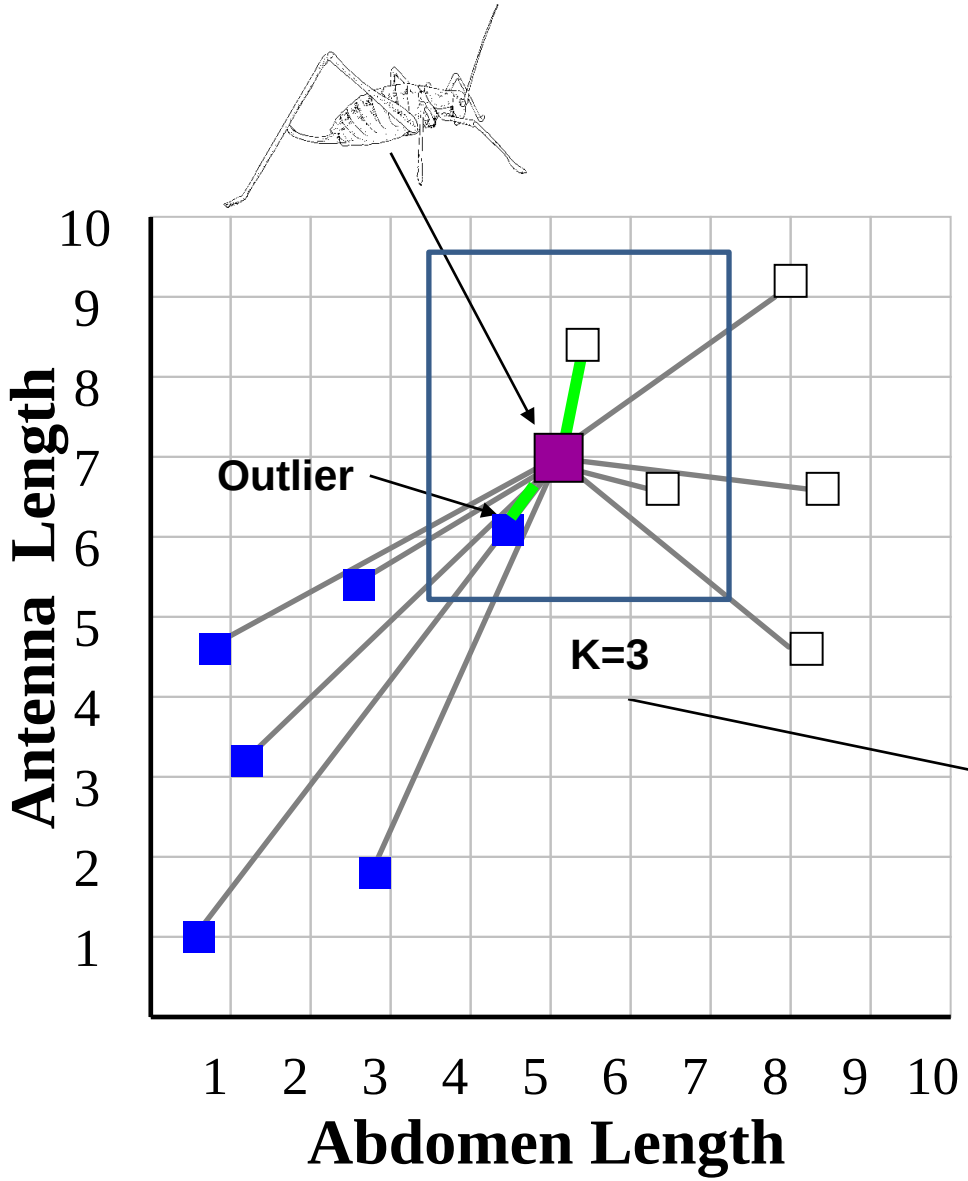


Nearest Neighbor Classifier

If the **nearest** instance to the previously unseen instance **is a Katydid**
    class is **Katydid**
else
    class is **Grasshopper**

□ **Katydids**
■ **Grasshoppers**

Nearest Neighbor Classifier



Nearest Neighbor Classifier

If the **nearest** instance to the previously unseen instance **is a Katydid**
class is **Katydid**
else
class is **Grasshopper**

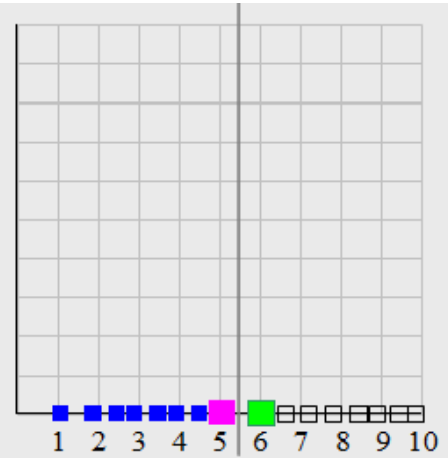
□ **Katydids**
■ **Grasshoppers**

Solution: Use K nearest neighbors instead, and take majority vote!

The nearest neighbor algorithm is sensitive to irrelevant features

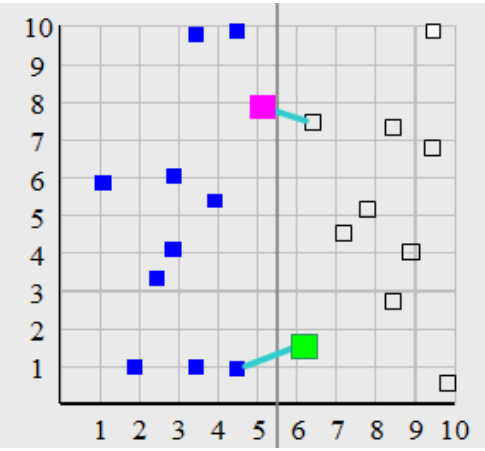
Suppose the following is true, if an insects antenna is longer than 5.5 it is a **Katydid**, otherwise it is a **Grasshopper**.

Using just the antenna length we get perfect classification!

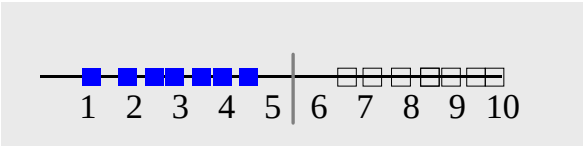


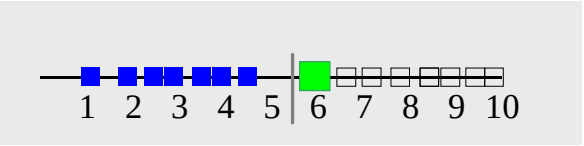
Suppose however, we add in an **irrelevant** feature, for example the insects mass.

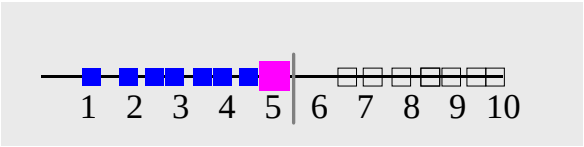
Using both the antenna length and the insects mass with the 1-NN algorithm we get the wrong classification!

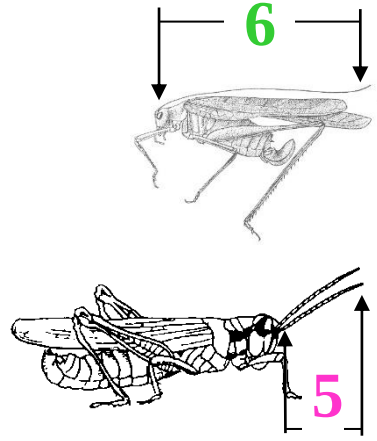


Training data









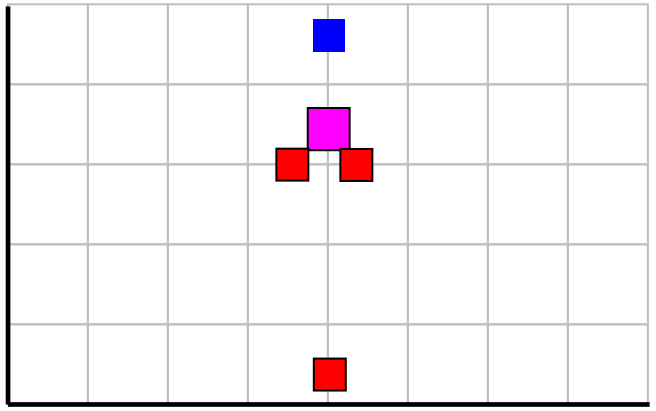
Handling Irrelevant Features

- How do we mitigate the nearest neighbor algorithms sensitivity to irrelevant features?

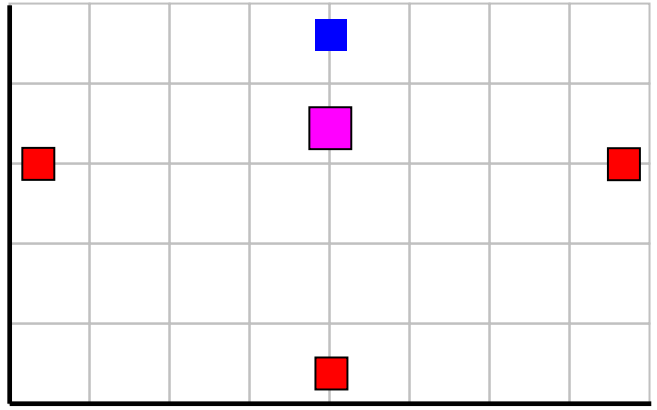
  - Use more training instances
  - Ask an expert what features are relevant to the task
  - Use statistical tests to try to determine which features are useful (remember p-values, can be calculated using **hypothesis testing methods**)

Units of Measurement

- The nearest neighbor algorithm is sensitive to the units of measurement



X axis measured in **centimeters**
Y axis measure in dollars
The nearest neighbor to the **pink** unknown instance is **red**.



X axis measured in **millimeters**
Y axis measure in dollars
The nearest neighbor to the **pink** unknown instance is **blue**.

- One solution is to normalize the units to pure numbers.

Advantages/Disadvantages of Nearest Neighbor

- **Advantages:**
  - Simple to implement
  - Handles correlated features (Arbitrary class shapes)
  - Defined for any distance measure
  - Handles streaming data trivially
- **Disadvantages:**
  - Very sensitive to irrelevant features.
  - Slow classification time for large datasets
  - Works best for real valued datasets
  - Does not build a model explicitly

End of Part 2

Machine Learning - Classification

Week 9 – Part 3 – Decision Trees

CS 457 - L1 Data Science

Zeehasham Rasheed

Decision Tree

- **Decision Tree Classifier**

  - You can think of the decision tree algorithm in terms of a flow chart
  - You ask the first question, then depending on the answer you either move down and left or down and right to a different second question
  - Each question defines a place where “branches” form a fork
  - For a given point, if we follow each sequence of answers down to the end of the branch, it will tells us the final answer for how we should classify the data.

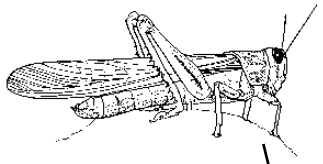
Decision Trees Details

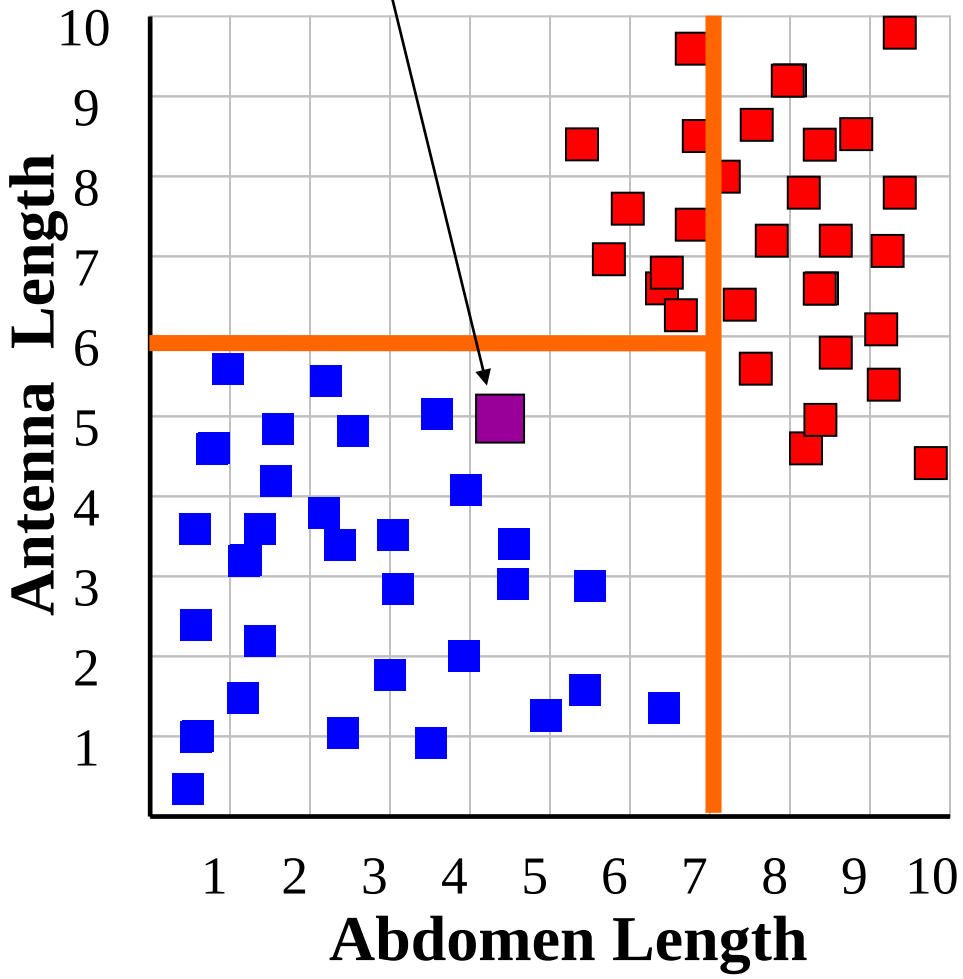
- Decision tree learning is one of the most widely used techniques for classification
- **A decision tree is a tree with the following properties:**
  - An inner node (decision node) represents an attribute. First node is called root
  - An edge represents a test (rule) on the attribute of the parent node
  - A leaf node represents the class/label od an example/record

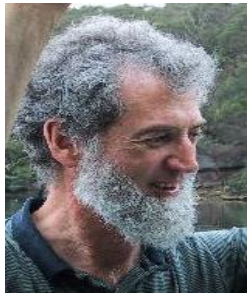
- **Construction of a decision tree:**
  - Based on the training data
  - Top-Down strategy

- **Functions of a decision tree**
  - Traversal of the decision tree from the root to one of the leaves (class)
  - Assignment of the records to a particular class by traversing from root to leaf
  - Decision rules can be interpreted as “if, then” statements.

Decision Tree Classifier

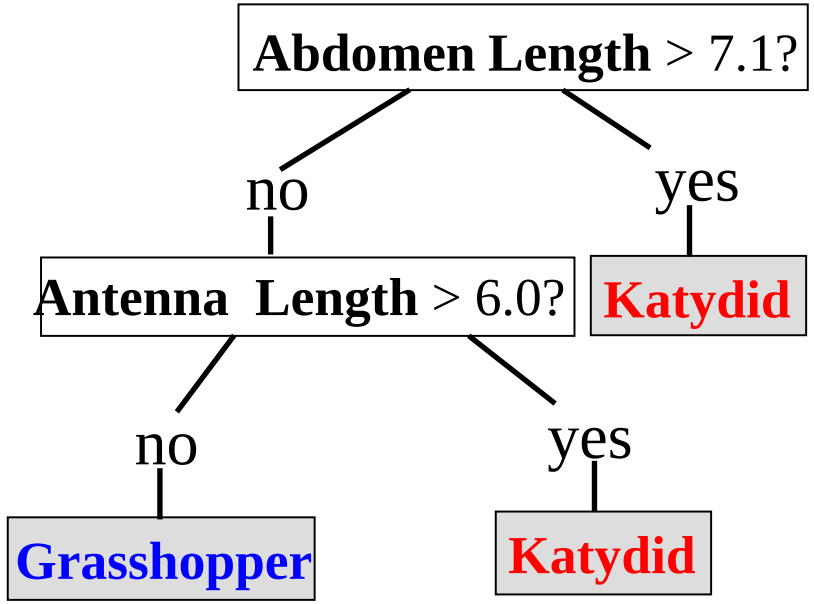




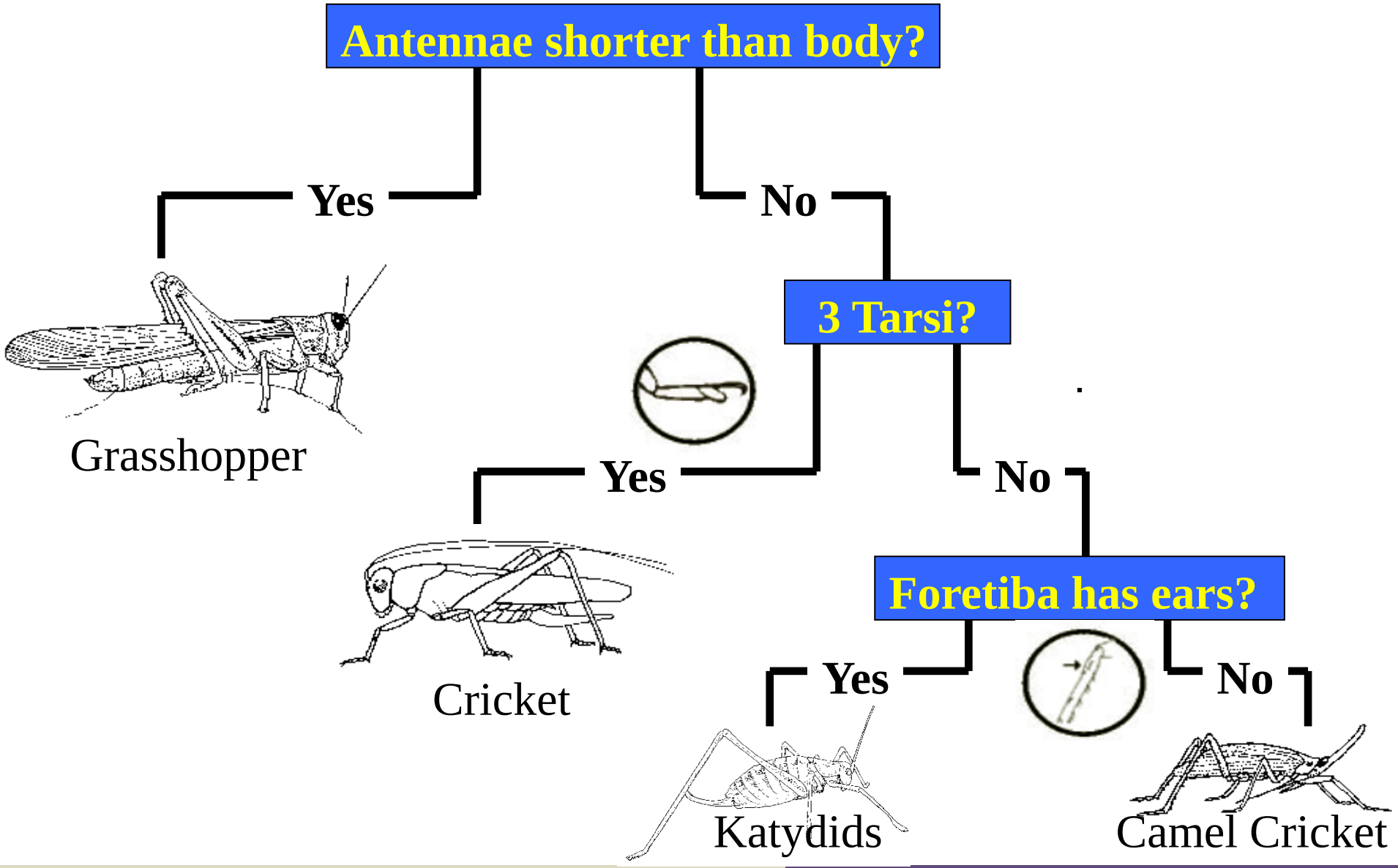


Ross Quinlan

(Classification Tree)



Decision Trees Predate Computers



Decision Tree Classification

- **Decision tree**
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- **Decision tree generation consists of two phases**
  - Tree construction
  - At start, all the training examples are at the root
  - Partition examples recursively based on selected attributes
- **Tree pruning**
  - Identify and remove branches that reflect noise or outliers
  - Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

How do we construct the decision tree?

- **Basic Approach**
  - Tree is constructed in a top-down manner
  - At start, all the training examples are at the root
  - At each step, partition the set of examples based on selected attribute.
  - Attributes are selected based on some measure, e.g. **information gain**
    - **Top most attribute at the root is the most important attribute for model prediction**

- **When do we stop partitioning a node?**
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Information Gain as a Splitting Criteria

- Select the attribute with the **highest information gain** (information gain is the expected reduction in entropy).
  - High Information Gain > Low Entropy
- Assume there are two classes, A and B
- Let the set of examples **S** contain x elements of class A and y elements of class B
- The amount of information, needed to decide if an arbitrary example in S belongs to A or B is defined as

$$E(S) = -\frac{x}{x+y}\log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y}\log_2\left(\frac{y}{x+y}\right)$$

Entropy (A Measure of Impurity)

- **Lets say 2 classes, A and B**
- Let the set of examples S contain x elements of class A and y elements of class B.

$$E(S) = -\frac{x}{x+y}\log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y}\log_2\left(\frac{y}{x+y}\right)$$

- For example: say our initial population is composed of 14 cases of class “Churn” and 16 cases of class “Not Churn”

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

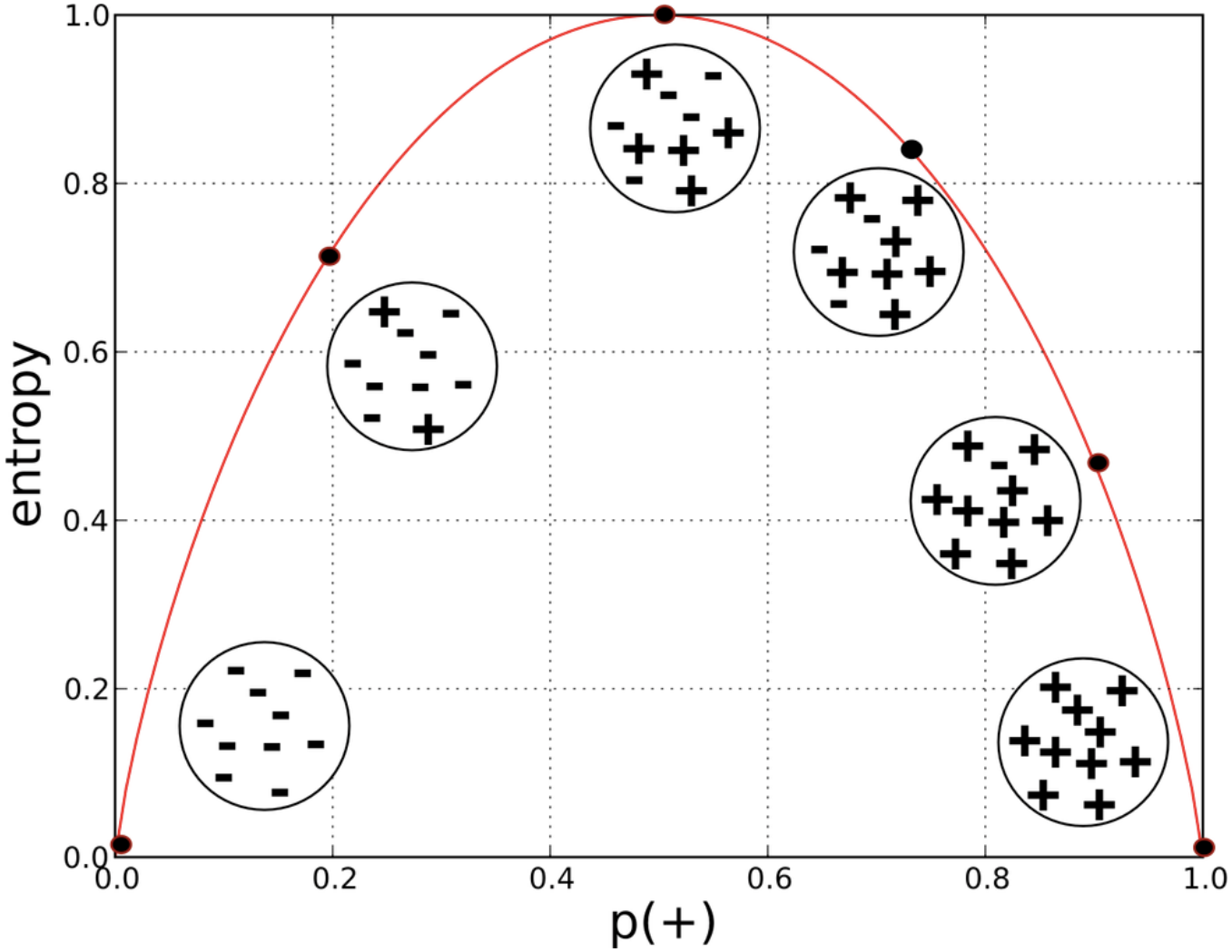
- We can have more two classes (just add appropriate terms to the formula)

$$Entropy = p_1\log(p_1) - p_2\log(p_2) - p_3\log(p_3) - \ldots$$

p_i is the proportion of class i in the data

Entropy Visualization

- **Entropy (Impurity)**



Information Gain in Decision Tree Induction

- Assume that using attribute A, a current set will be partitioned into some number of child sets
- The encoding information that would be gained by branching on A

Summation, total of

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

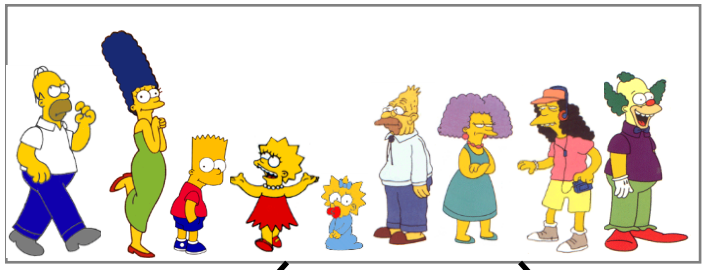
- Select the attribute with the highest information gain and use it to split the node.

Information Gain = Impurity (parent) – [Impurity (children)]

Note: Entropy is at its minimum if the collection of objects is completely uniform/similar

Classification Example Dataset

Person		Hair Length	Weight	Age	Class
	Homer	0”	250	36	M
	Marge	10”	150	34	F
	Bart	2”	90	10	M
	Lisa	6”	78	8	F
	Maggie	4”	20	1	F
	Abe	1”	170	70	M
	Selma	8”	160	41	F
	Otto	10”	180	38	M
	Krusty	6”	200	45	M
	Comic	8”	290	38	?



$$E(S) = -\frac{y}{y+n}\log_2\left(\frac{y}{y+n}\right) - \frac{n}{y+n}\log_2\left(\frac{n}{y+n}\right)$$

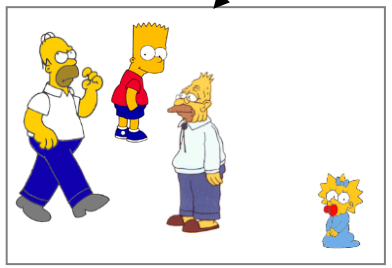
Entropy(4**F**,5**M**) = -(4/9)log2(4/9) - (5/9)log2(5/9)

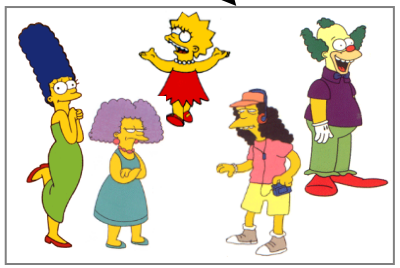
= **0.9911**

yes

no

Hair Length <= 5?





Entropy(1**F**,3**M**) = -(1/4)log2(1/4) - (3/4)log2(3/4)

= **0.8113**

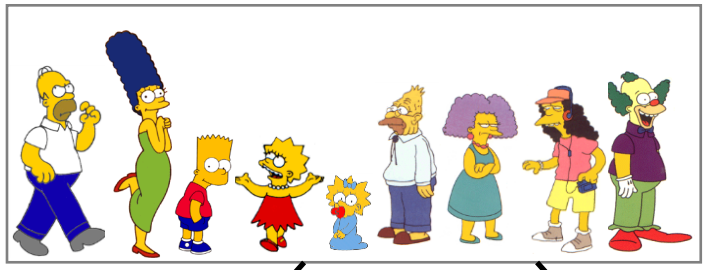
Let us try splitting on *Hair length*

Entropy(3**F**,2**M**) = -(3/5)log2(3/5) - (2/5)log2(2/5)

= **0.9710**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Hair Length <= 5) = **0.9911** – (4/9 * **0.8113** + 5/9 * **0.9710**) = **0.0911**



$$E(S) = -\frac{y}{y+n}\log_2\left(\frac{y}{y+n}\right) - \frac{n}{y+n}\log_2\left(\frac{n}{y+n}\right)$$

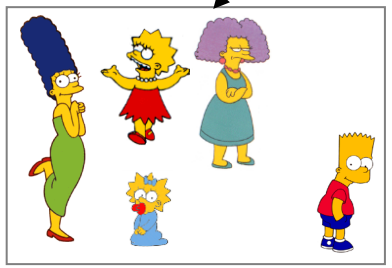
Entropy(4**F**,5**M**) = -(4/9)log2(4/9) - (5/9)log2(5/9)

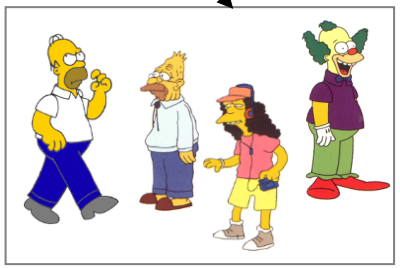
= **0.9911**

yes

no

Weight <= 160?





Let us try splitting on *Weight*

Entropy(4**F**,1**M**) = -(4/5)log2(4/5) - (1/5)log2(1/5)

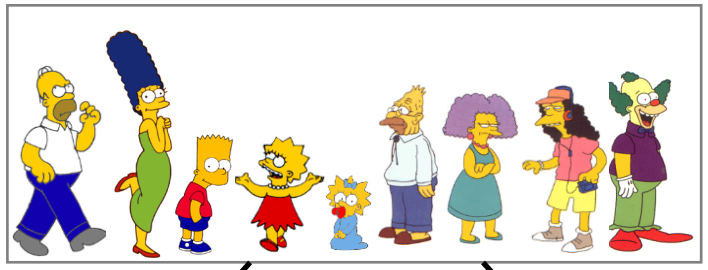
= **0.7219**

Entropy(0**F**,4**M**) = -(0/4)log2(0/4) - (4/4)log2(4/4)

= **0**

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Weight <= 160) = **0.9911** – (5/9 * **0.7219** + 4/9 * **0**) = **0.5900**



$$E(S) = -\frac{y}{y+n}\log_2\left(\frac{y}{y+n}\right) - \frac{n}{y+n}\log_2\left(\frac{n}{y+n}\right)$$

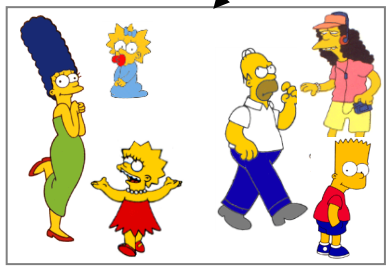
Entropy(4**F**,5**M**) = -(4/9)log2(4/9) - (5/9)log2(5/9)

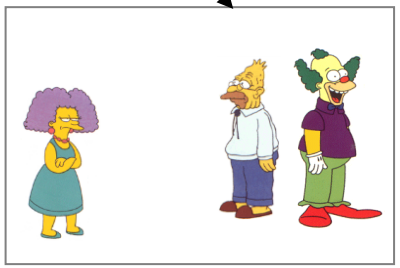
= **0.9911**

yes

no

age <= 40?





Let us try splitting on *Age*

Entropy(3**F**,3**M**) = -(3/6)log2(3/6) - (3/6)log2(3/6)

= **1**

Entropy(1**F**,2**M**) = -(1/3)log2(1/3) - (2/3)log2(2/3)

= **0.9183**

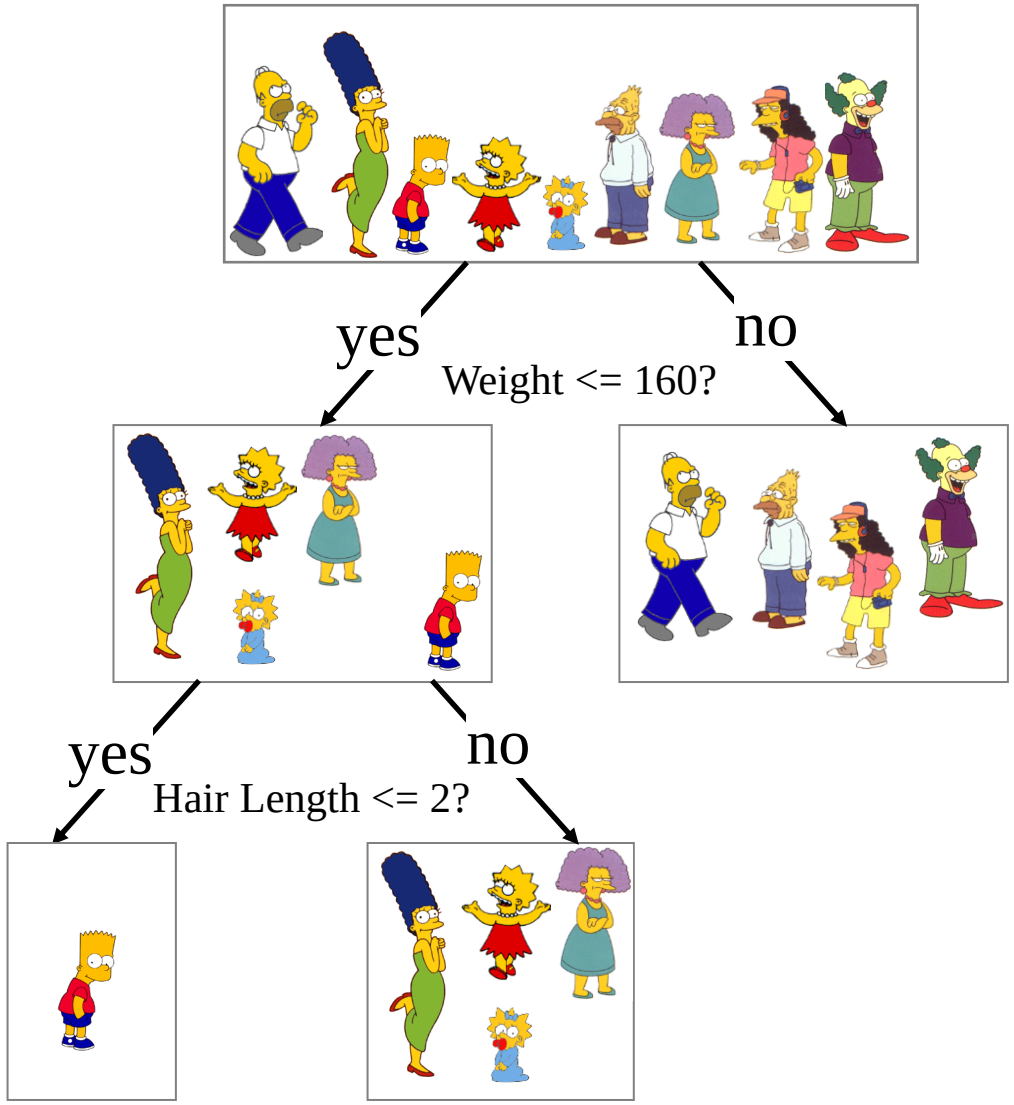
$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

Gain(Age <= 40) = **0.9911** – (6/9 * **1** + 3/9 * **0.9183**) = **0.0183**

Recursiveness

Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified
So we simply recurse!

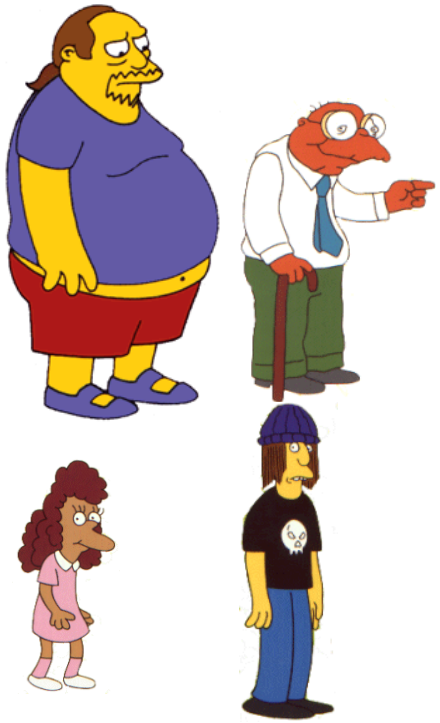
This time we find that we can split on *Hair length*, and we are done!

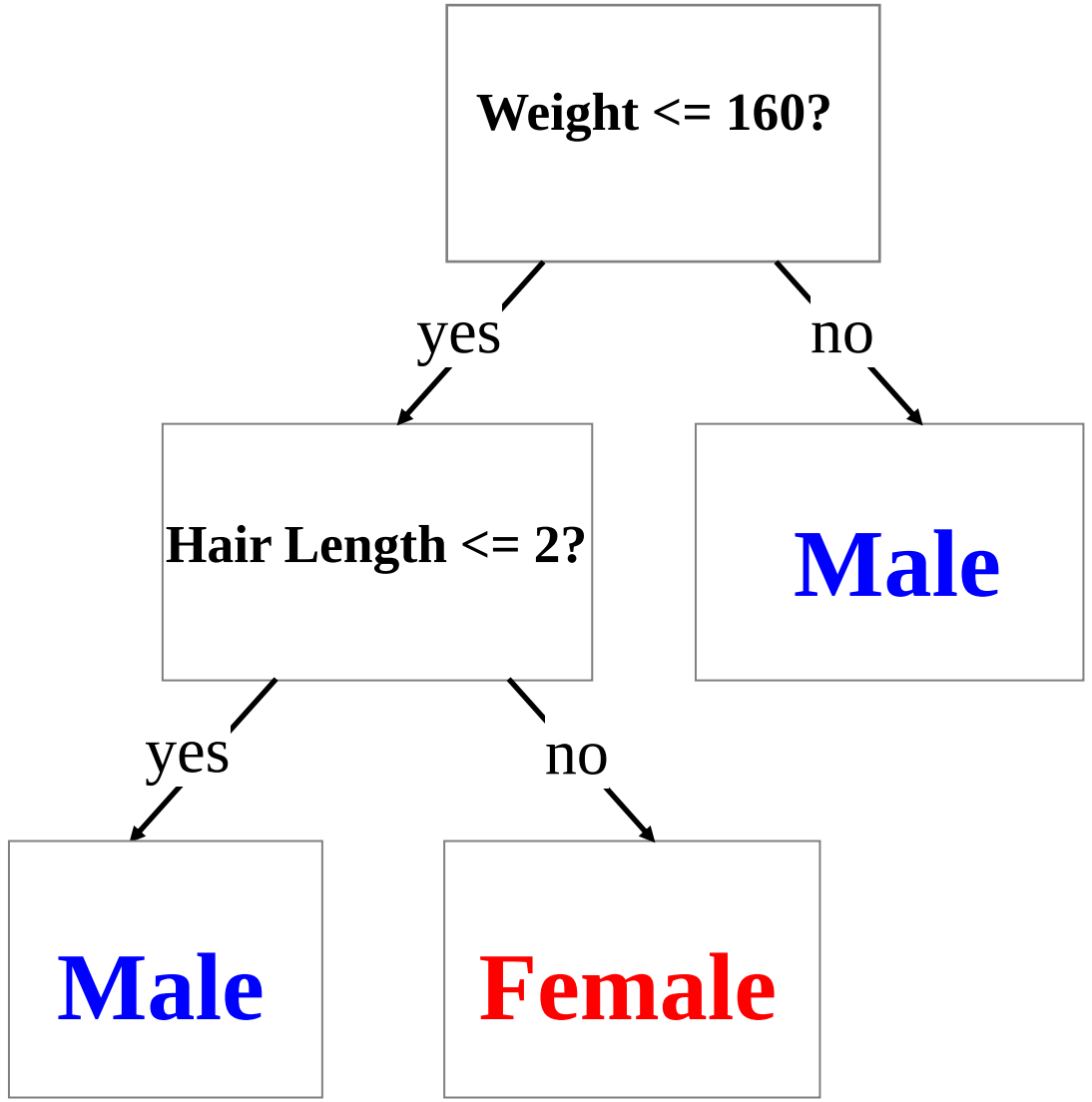


Classifying Unknown Data

We don't need to keep the data around, just the test conditions.

How would these people be classified?



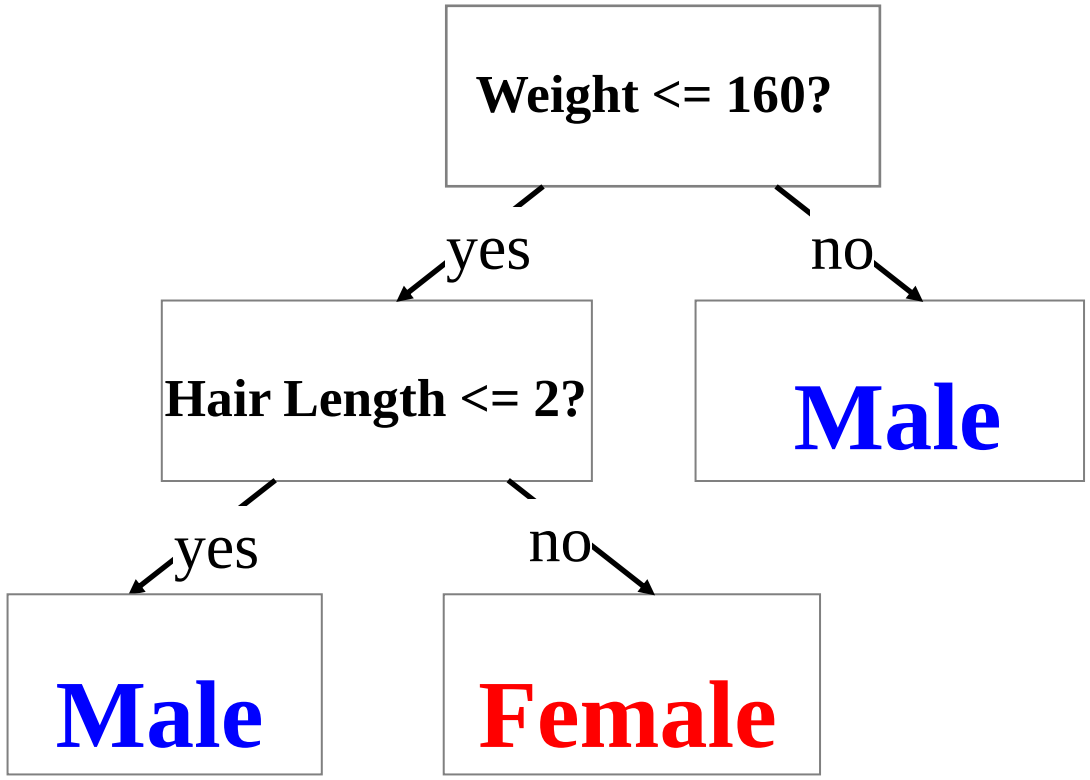


Decision Trees to Rules

It is trivial to convert Decision Trees to rules

Rules to Classify Males/Females

If *Weight* **greater than** 160, classify as **Male**
Elseif *Hair Length* **less than or equal** to 2, classify as **Male**
Else classify as **Female**



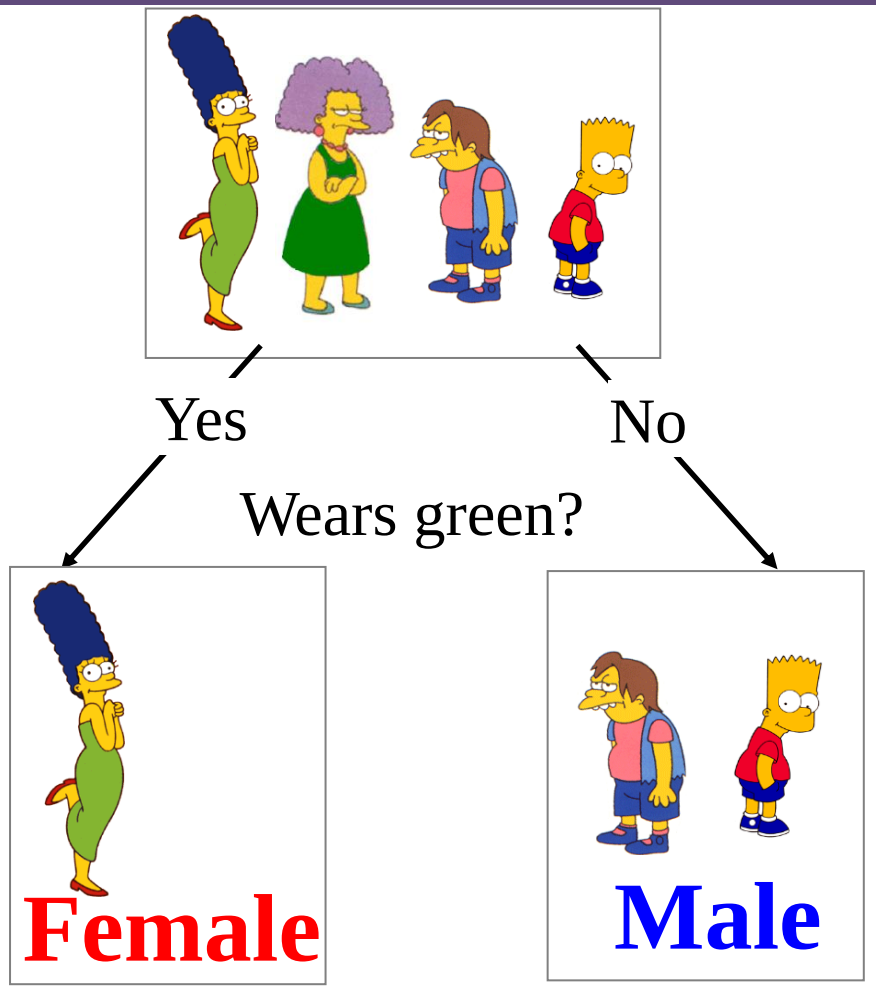
Problem with Limited Data

With small datasets there is a **great danger of overfitting**

- When you have few datapoints, there are many possible splitting rules that perfectly classify the data, but **will not generalize** to future datasets (very deep/large tree trying to classify each training record correctly)

And a very simple decision tree leads to a problem of **underfitting**

- This is because the model is unable to capture the relationship between the input examples



For example, the rule “Wears green?” perfectly classifies the data, so does “Has blue shoes”…

Decision Tree Summary

- **For classification modeling, decision tree is one of the most popular data mining tools. It is:**
  - Easy to understand (Doctors love them!)
  - Easy to implement
  - Easy to use
  - Computationally cheap

- **Works remarkably well**
  - (not the most accurate, by the way)

- **Almost all Machine Learning packages include decision tree algorithms**
- Has advantages for model comprehensibility, which is important for:
  - model evaluation
  - communication to non-ML-savvy stakeholders

Handling Overfitting

- **Avoid Overfitting in Classification**

- The generated tree may overfit the training data
  - **Too many branches**, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples (test data)

- **General approaches to avoid/handle overfitting**
  - Tree pruning

Example: Complexity control in tree induction

- For decision tree: **Pruning**
  - Reduce size of tree to (try to) reduce overfitting
  - choose “right size” of tree based on fitting curve
    - Choose the right **height/depth** of the tree
    - Resulting in better performance (reducing classification error on test data)
  - eliminate subtrees if statistically estimated error is not increasing

Real World Example

- **Retailer Example**
  - A retailer wants to introduce new product in different locations. The past sales records for similar product contains the following information
    - CompPrice, Income, Advertising, Population, Price, Quality, Age, Education, Urban, US, SalesCat
    - SalesCat is the outcome

- Goal: to predict whether the sale of a product would be “high” or “low” in a new location.

Dataset Details

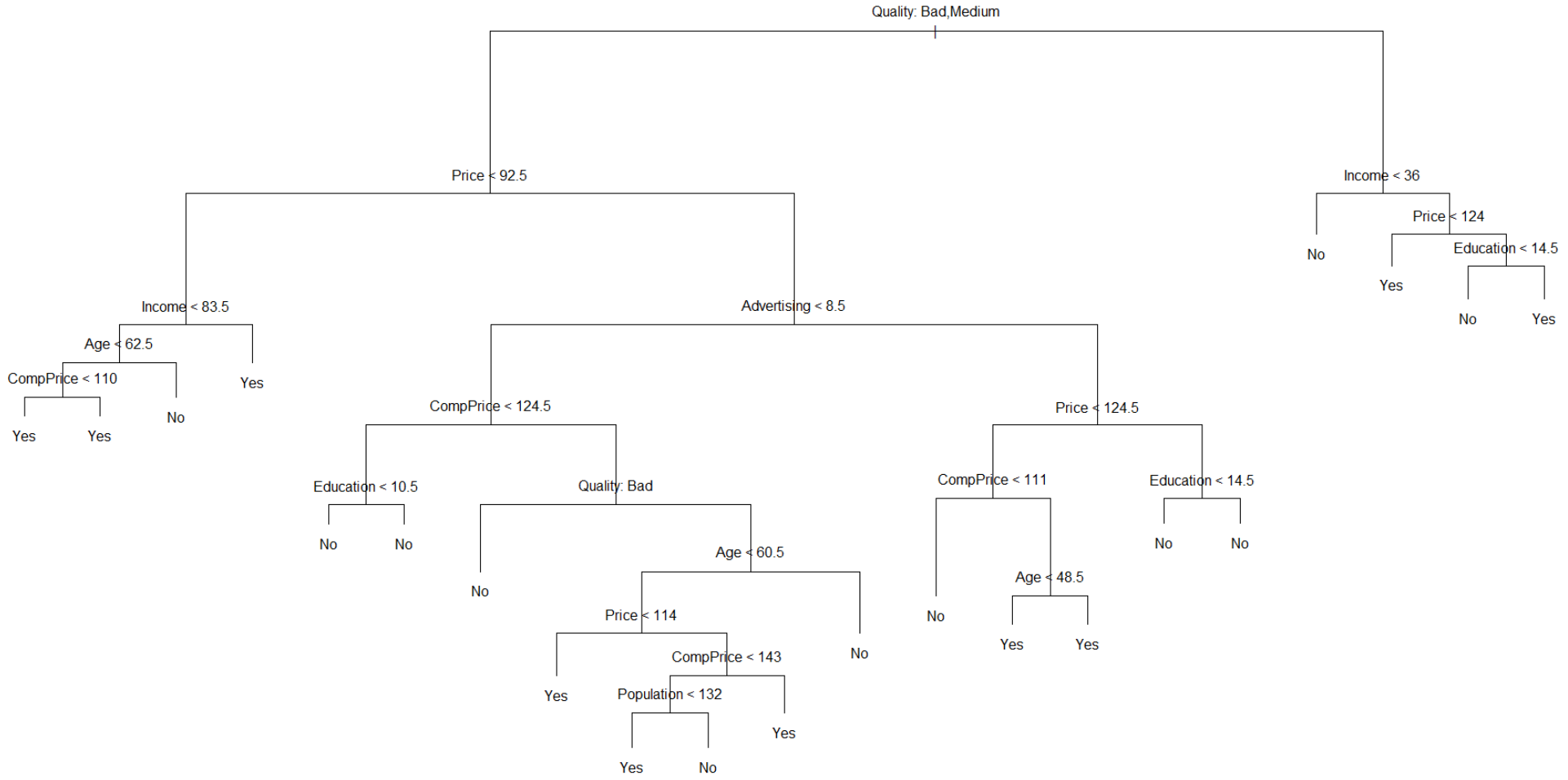
- CompPrice: Price charged by competitor at each location
- Income: Community income level (in thousands of dollars)
- Advertising: Local advertising budget for company at each location (in thousands of dollars)
- Population: Population size in region (in thousands)
- Price: Price company charges for a product at each location
- Quality: A value with levels Bad, Good and Medium indicating the quality of the product at each location
- Age: Average age of the local population
- Education: Education level at each location
- Urban: A value of No and Yes to indicate whether the store is in an urban or rural location
- US: A value of No and Yes to indicate whether the store is in the US or not
- **SalesCat: A value of Low and High to show the outcome of sale trend at each location (**response variable)

R code: Creating tree model

- Pick top 250 sample for training.
- Rest of the sample is used for testing (-train)
- error is the total misclassification rate.

```
# A library for Decision Tree
library(tree)
# Read data file
product <- read.csv('salesdata.csv')
# Separate training data
train <- c(1:250)
# Building Decision Tree model
dt <- tree(SalesCat ~ ., data = product, subset = train)
plot(dt)
text(dt, pretty = 0)
# Prediction on Test data
dt.pred <- predict(dt, product[-train, ], type = "class")
# Calculate misclassification error
error <- mean(dt.pred != product$SalesCat[-train])
```

Decision Tree from the data



Some Analysis on Tree

- Tree starts from root (top) and ends at leaves (bottom)
- Root contains the most important attribute (here important means that attribute gives highest information for classification)
- Large number of levels and nodes make the tree more complex.
- Large and complex trees leads to a problem called “over fitting” in classification.

Questions

Question	Answer
How many terminal nodes ? (use summary function)	
Who is the most important attribute ?	
Which attribute is missing in the tree ?	
What does missing attribute tells us ?	
What is the percentage of misclassification error ?	
Is the tree looks complex and big (w.r.t terminal nodes) ?	
What is the problem related to large trees called ?	

Answers

Question	Answer
How many terminal nodes ? (use summary function)	21
Who is the most important attribute ?	Quality
Which attribute is missing in the tree ?	Urban
What does missing attribute tells us ?	Least important attribute
What is the percentage of misclassification error ?	30.6 %
Is the tree looks complex and big (w.r.t terminal nodes) ?	Yes
What is the problem related to large trees called ?	Over fitting

Decision Tree – Over Fitting

- Over fitting is a significant practical problem in decision tree models and many other predictive models.
- Over fitting happens when the learning algorithm continues to grow tree (or model) that reduce training set error at the cost of an increased test set error.
- Overall goal is to develop a better model which is small, simple and predicts data with good accuracy
- A good model has better generalization performance (simple yet accurate on both train and test set)

Pruning

- There are several approaches to avoiding over fitting in building decision trees.
  - Pre-pruning that stop growing the tree earlier, before it perfectly classifies the training set.
  - Post-pruning that allows the tree to perfectly classify the training set, and then post prune the tree.

- Practically, the second approach of post-pruning is more successful (because it is not easy to precisely estimate when to stop growing the tree at runtime)

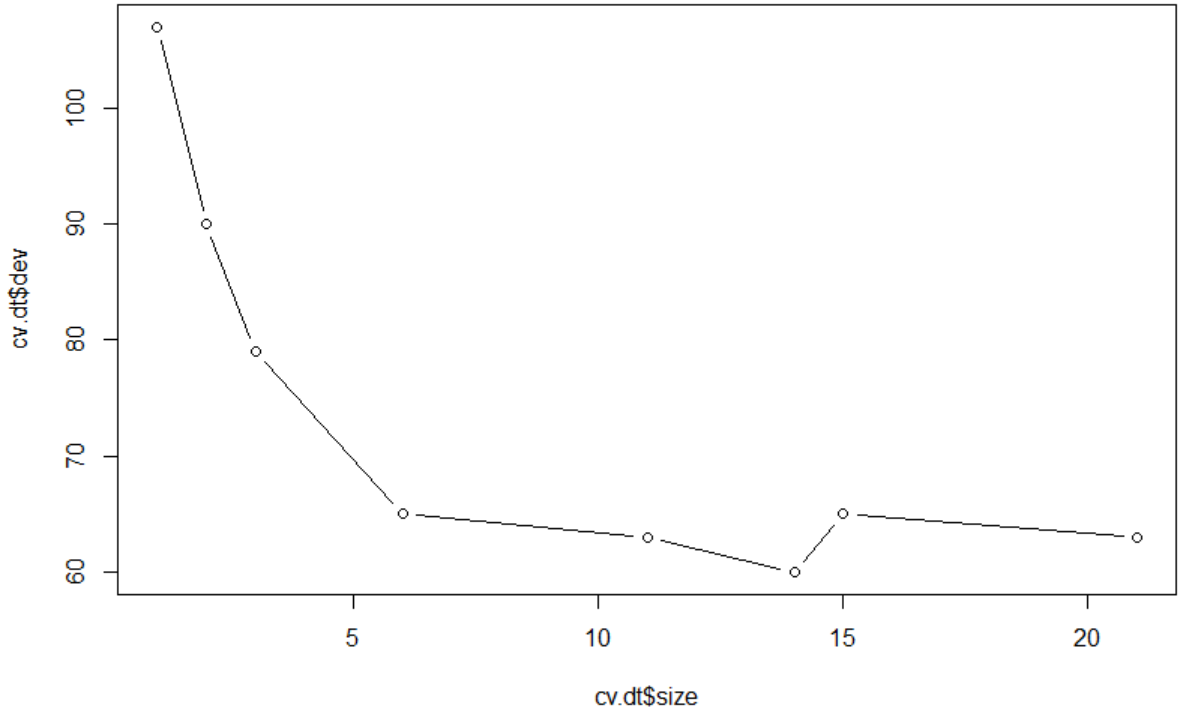
Pruning Method

- The important step of tree pruning is to define a criterion to determine the correct final tree size
- In this method, the available data is separated into two sets: a training set, which is used to build the decision tree, and a validation set or test set, which is used to evaluate the impact of pruning nodes from the tree.
- The method stops when a particular size of tree gives the best performance (in terms of classification accuracy)

Size vs Classification Error

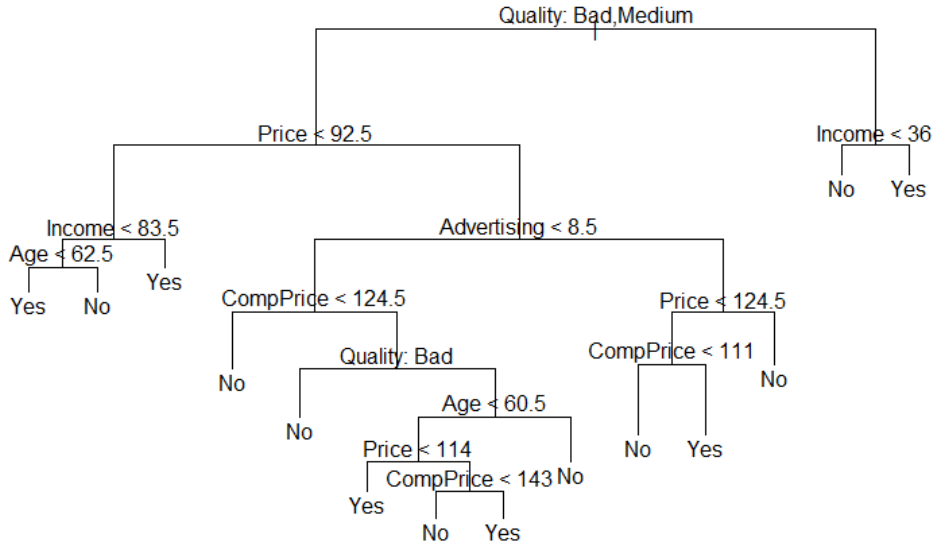
```
# Cross-validation for Pruning
cv.dt = cv.tree(dt, FUN = prune.misclass)
plot(cv.dt$size, cv.dt$dev, type='b')
```

- cv.dt$size is the size of tree
- cv.dt$dev is deviance or error rate
- Size = 14 seems to be the best choice



R code: Creating best tree model

- **Results:**
  - Size is 14
  - Percentage of Misclassification Error is 28% (from 30
  - Thus Pruning helps reducing misclassification error




```
# Using best node size and get the Best tree
prune.dt = prune.misclass(dt, best = 14)
  plot(prune.dt) text(prune.dt, pretty = 0)
# Prediction on Test data using Best tree
prune.pred <- predict(prune.dt, product[-train, ],
  type = "class")
# Calculate misclassification error
error <- mean(prune.pred != product$SalesCat[-train])
```

Can we further improve?

- Yes, you can use **Random Forest !**
- Random Forest is an ensemble learning algorithm.
- **Ensemble learning** is the process by which multiple models (classifiers) are strategically generated and combined to solve a particular classification problem.
- Ensemble learning is primarily used to improve the classification or prediction performance of a model, or reduce the likelihood of an unfortunate selection of a poor model.

End of Part 3

Machine Learning - Classification

Week 9 – Part 4 – Random Forests

CS 457 - L1 Data Science

Zeehasham Rasheed

Random Forest (RF)

- The basic principle is that a group of “weak learners” can come together to form a “strong learner”.

- The word “forest” comes from the fact that the algorithm combines a large number of (decision) trees.

Construction of RF

- First, construct trees in two ways:
  - 1. Add an element of randomness to the process. For example, we **randomly select which input attributes** are used to create tree (selecting columns)
  - Also choose best split criteria that maximizes Information Gain.
  - 2. Another way to add randomness is a procedure called bootstrapping or bagging, in which we train the decision trees on a random subset sampled from the original dataset. (selecting rows)
  - Widely used method

Construction of RF (2)

- Second, each tree is grown to the largest extent possible. There is no pruning.

- Third, standard way to **combine the trees output** is through a voting process.
  - For example, given a new test data point, we run it through each decision tree in the forest
  - Record how many trees vote for it to be a “Yes” label and how many vote for it to be a “No” label.
  - Which ever label **gets the most votes**, that is the decision Random Forest makes in the end

Why Random Forest?

- RF effectively deals with two issues
  - **Over fitting:** Avoid complex and large trees by choosing subset of dataset to create tree
  - **Under fitting:** Avoid too simple trees that are unable to capture the structure and patterns in the dataset (by creating multiple trees and avoiding pruning step)

Apply RF

- Lets apply the RF on the same retail example and compare with decision trees.
- ntree is the number of decision tree used in random forest

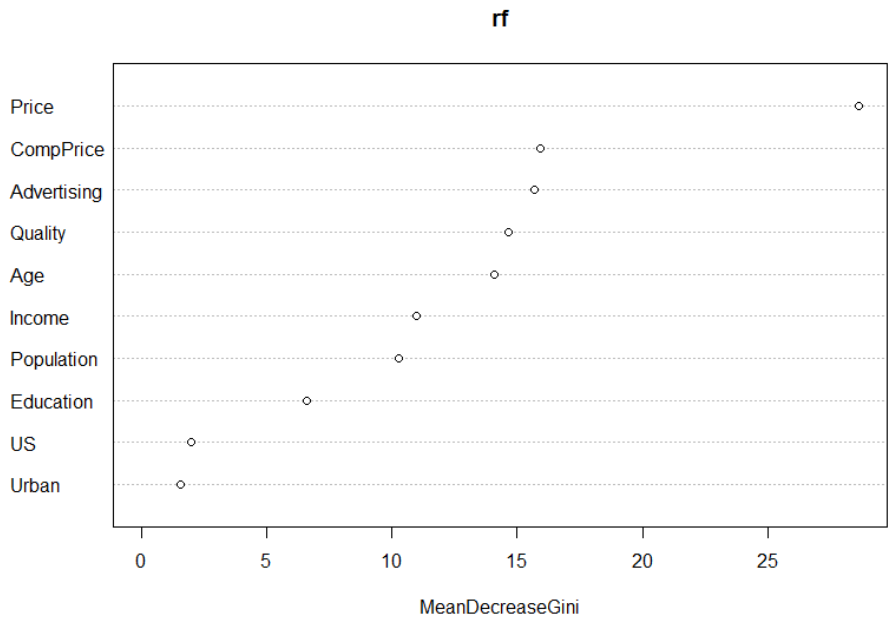
```
# A library for Random Forest
library(randomForest)
# Read data file
product <- read.csv('salesdata.csv')
attach(product)
# Separate training data
train <- c(1:250)
# Building Random Forest model
rf <- randomForest(SalesCat ~ ., data = product,
  subset = train, ntree = 100)
```

RF Important Attributes

- Find the important attributes in RF model for prediction

```
# Some information about model
importance(rf)
##             MeanDecreaseGini
## CompPrice             15.559
## Income                11.772
## Advertising           15.391
## Population            10.397
## Price                 28.450
## Quality               15.311
## Age                   14.862
## Education              6.197
## Urban                  1.114
## US                     1.801
```

```
# Importance Plot
varImpPlot(rf)
```



RF Prediction on Test Data

- Apply RF model on test data to calculate misclassification error

```
# Prediction on Test data
rf.test.pred <- predict(rf,newdata=product[-train, ])
# Calculate misclassification error
error <- mean(rf.test.pred != product$SalesCat[-train])
```

- Misclassification error is 19.3% (error value and importance plot might change because of random sampling in Random Forest. But the overall trend should look the same)

- RF performs much better (19.3% misclassification error) than Decision Trees (28% misclassification error) on same dataset.

Questions

Question	Answer
How many trees used in RF?	
Which is the first most important attribute ?	
Which is the second most important attribute ?	
Which is the third most important attribute ?	
What is the percentage of misclassification error ?	
Is RF performs better than decision tree ?	
What is the effect of increasing the number of trees ?	

Answers

Question	Answer
How many trees used in RF?	100
Which is the first most important attribute ?	Price
Which is the second most important attribute ?	CompPrice
Which is the third most important attribute ?	Advertising
What is the percentage of misclassification error ?	19.3 % (may vary)
Is RF performs better than decision tree ?	Yes
What is the effect of increasing the number of trees ?	Performs better to some extent

Conclusion From Exercise

- From the results, we see that Random Forest performs much better than Decision Trees
- If a retailer uses Decision Tree to introduce a product in a new location, retailer will fail 28% of the times to get the high sales
- If a retailer uses Random Forest to introduce a product in a new location, retailer will fail about 19% of the times to get the high sales.
- **RF is a better choice !**

Logistic Regression

- **Logistic Regression -** Another technique for classification

- Logistic regression is used to analyze relationships between a dependent variable (output) and independent variables (inputs)
  - Same Linear Regression process

- Logistic regression combines the independent variables to estimate the probability that a particular event will occur. This event is defined by the dependent variable.

LR prediction

- The output value produced by logistic regression is a probability value between 0.0 and 1.0

  - If the probability produced by logistic regression for one class is above some threshold (the default is 0.5), the input is predicted to be a member of that output class.
  - If the probability is below the threshold, the subject is predicted to be a member of other class.

End of Part 4

Machine Learning - Classification

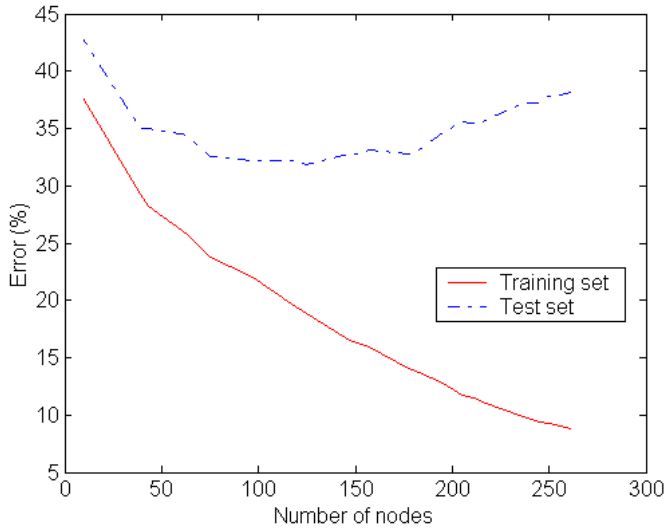
Week 9 – Part 5 – Model Evaluation

CS 457 - L1 Data Science

Zeehasham Rasheed

Model Evaluation

- We are interested in **generalization**
  - the performance on data **not used for training**
  - Training data is also called Ground Truth



- Given a data set, we hold out some data for evaluation
  - **holdout set** for final evaluation is called **test set**

- Accuracy on training data is sometimes called **“training” accuracy or “in-sample”** accuracy
- Accuracy on test data is called **“testing” accuracy or “out-of-sample”** accuracy on test data
  - a.k.a. “holdout accuracy” (an estimate of generalization accuracy)

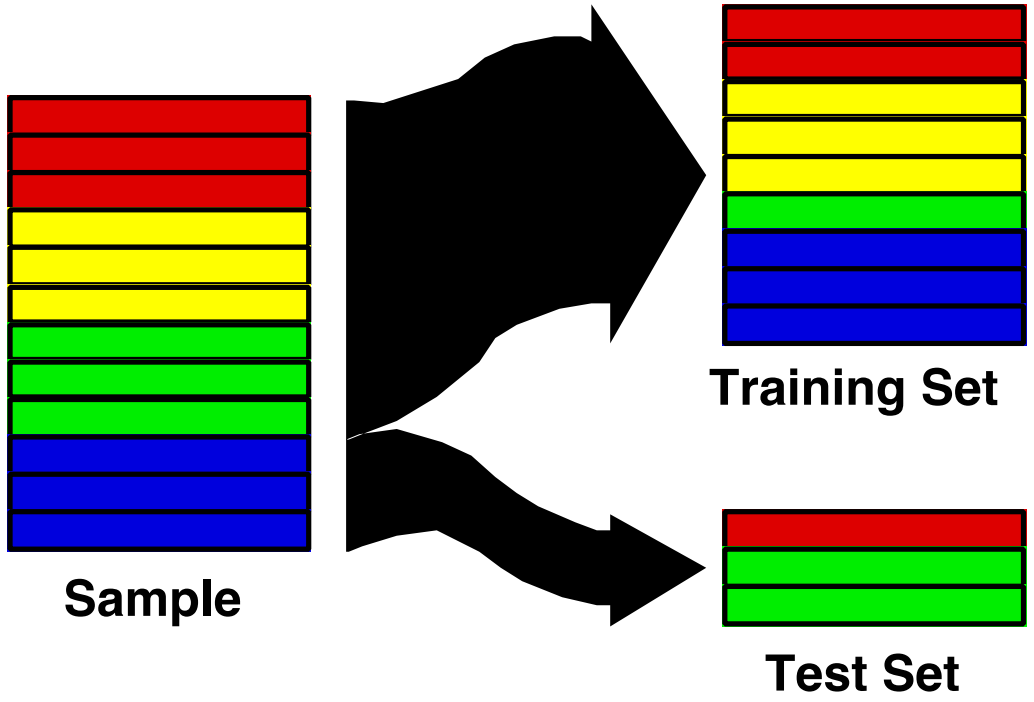
Hold-out validation: simple holdout set

- Partition data into training set and test set

  challenges:

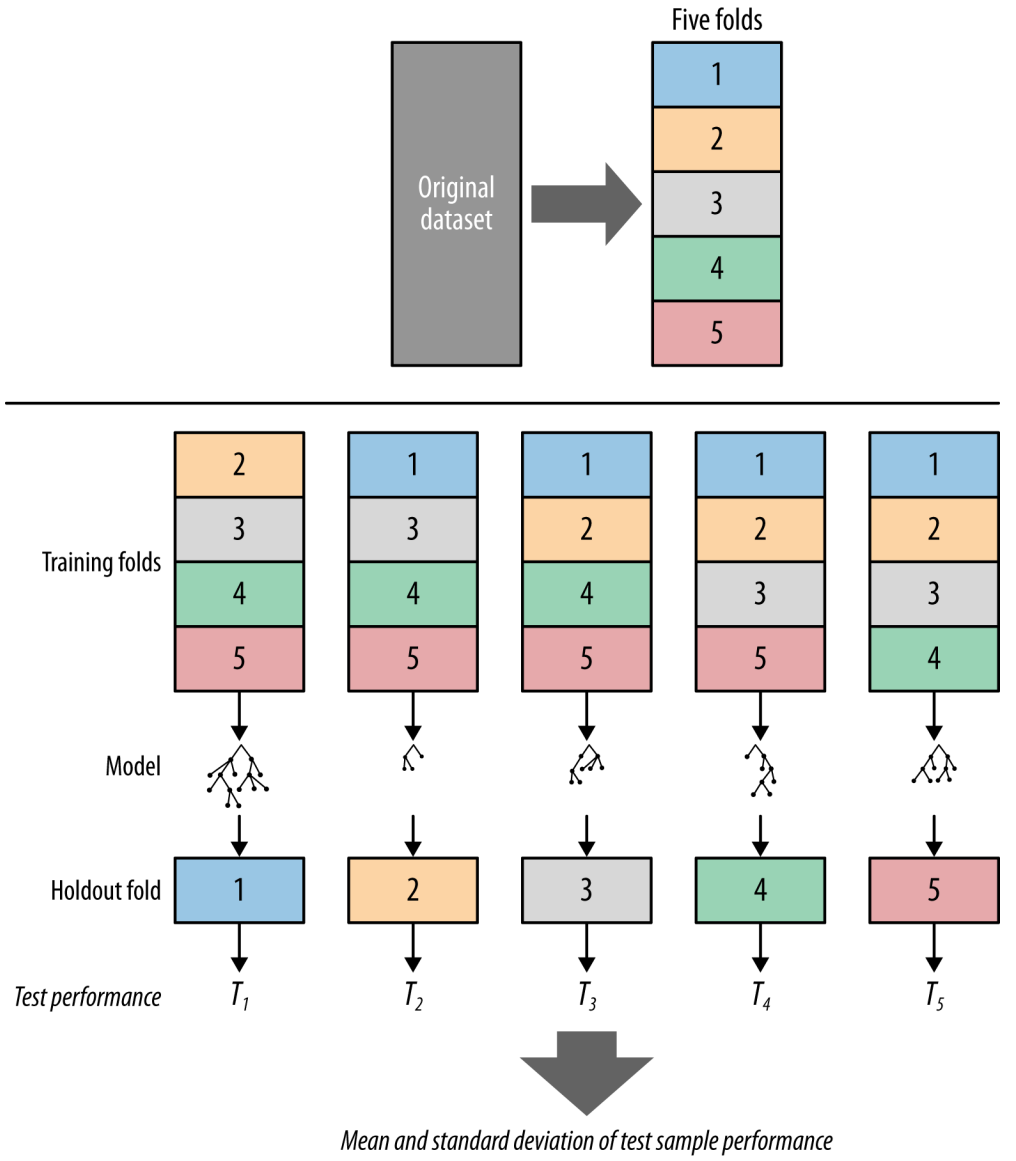
  1) what if by accident you selected a particularly easy/hard test set?

  2) do you have an idea of the variation in model accuracy due to training?



Holdout validation: Cross-validation (CV)

- Train and Test on every record
- Better method of evaluation compared to simple holdout set



Metrics for Performance Evaluation

- Focus on the predictive capability of a model

- Confusion Matrix:
  - a: TP (true positive)
  - b: FN (false negative)
  - c: FP (false positive)
  - d: TN (true negative)

	Predicted	
Actual	Class=Yes	Class=No
Class=Yes	a	b
Class=No	c	d

- Most widely-used metric

	Predicted	
Actual	Class=Yes	Class=No
Class=Yes	a	b
Class=No	c	d

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Computing Cost

- Computing Cost of Classification

	Predicted	
Actual	Class=Yes	Class=No
Class=Yes	-1	100
Class=No	1	0

Cost of misclassifying class Yes as class No and so on

Model M1	Predicted	
Actual	Class=Yes	Class=No
Class=Yes	150	40
Class=No	60	250

Accuracy = 80%
Cost = 3910

Model M2	Predicted	
Actual	Class=Yes	Class=No
Class=Yes	250	45
Class=No	5	200

Accuracy = 90%
Cost = 4255

Things about Decision Trees

- Easy to understand
- Can be represented using flow charts
- Easy to implement
- Sometimes gives very complex model and falls into over fitting problem. Required pruning
- Carefully choosing the important attribute is the key to good model

Things about Random Forests

- Accurate learning algorithm.
- Handles over fitting problem.
- No pruning is required.
- Complex model and hard to implement
- Parameter such as number of decision trees must be carefully selected.

End of Part 5

Machine Learning - Classification

Week 9 – Part 6 – Recommendation Systems

CS 457 - L1 Data Science

Zeehasham Rasheed

Recommendation using Decision Trees

- Your friend likes some whiskey.
- Can you recommend a new whiskey that makes your friend happy?
- How do you know if your friend will like the new whiskey?
- **Hypothesis/Assumption:** Any new whiskey that tastes similar to your friend's choice of whiskey would work.

Decision Tree as a Recommender

- You can use Decision Tree !
- Consider a whiskey data collected from survey. People rated each attribute of whiskey at a scale from1 to 5.
- The attributes of whiskey are Distillery (whiskey name), Body, Sweetness, Smoky, Medicinal, Tobacco, Honey, Spicy, Winey, Nutty, Malty, Fruity and Floral
- Distillery (whiskey name) can be considered as response variable (class)

Create a Recommender

- The idea is to produce a tree that fits the data perfectly, set

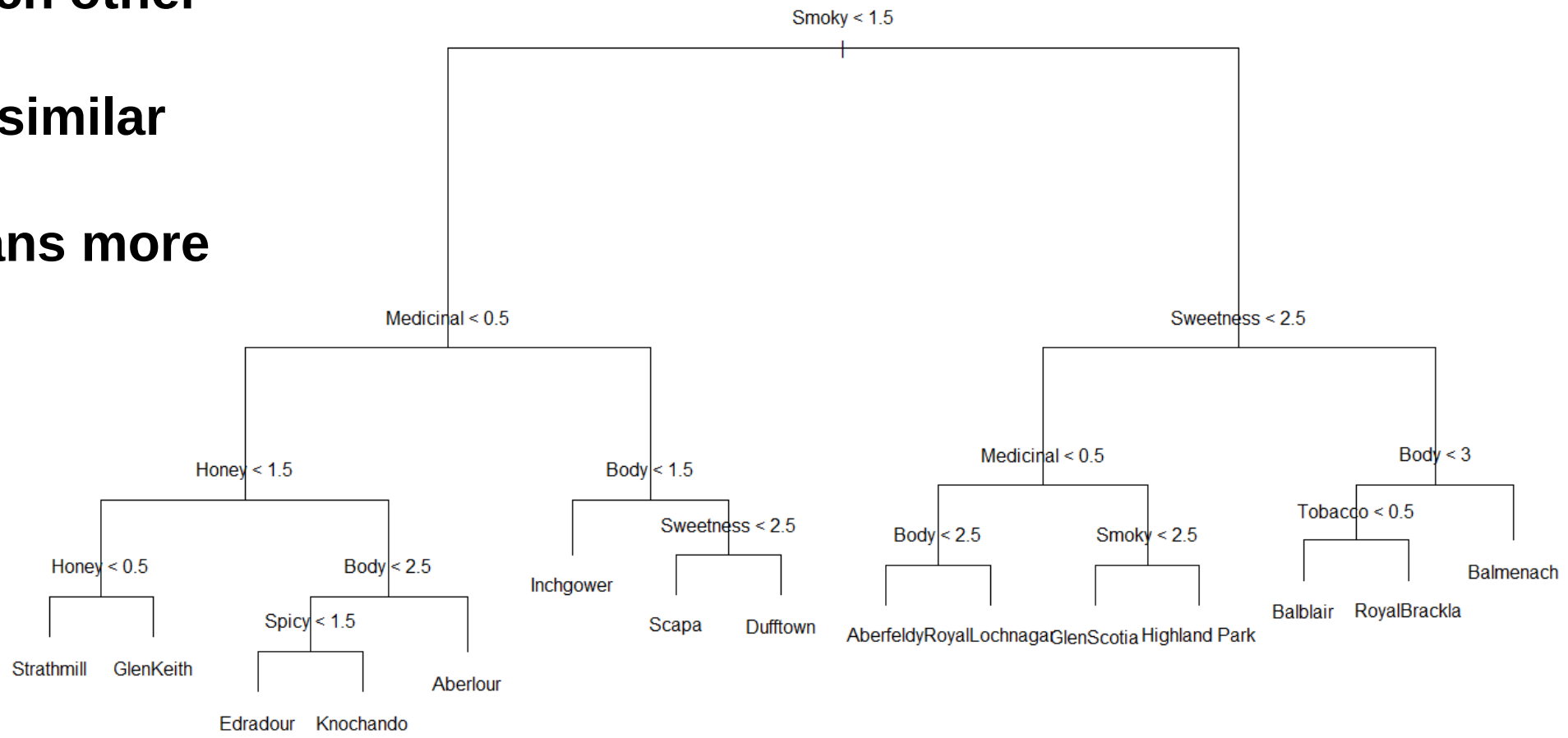
- minsize = 2 (smallest node size)
- mincut = 1 (minimum number of observation in child node)

```
library(tree)
data <- read.csv('whiskeydataset.csv')
dt <- tree(Distillery~., data=data, control = tree.control(nrow(data), mincut = 1,
  minsize = 2))
plot(dt)
text(dt,pretty=0)
```

Recommender Tree

Leaves (classes) closer to each other in a tree are considered similar

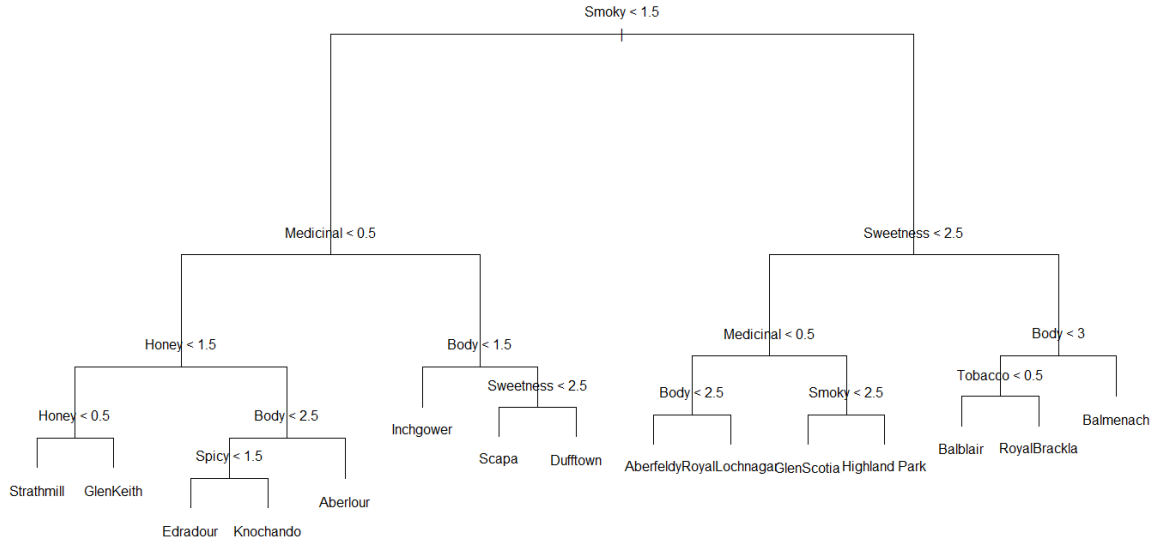
(farther means more different)



How to Make a Decision?

- Rule of Thumb: In decision trees, nodes close to each other are similar in their attributes
  - Therefore, Whiskeys closer to each other in a particular node down the tree are similar.

Question	Answer
Is Edradour similar to Knochando?	
Is Scapa similar to Dufftown?	
Is Strathmill similar to Balmenach?	
Is Strathmill similar to Dufftown?	



End of Part 6