Rensselaer

why not change the world?®

Regression & Classification Methods

Ahmed Eleish

Data Analytics ITWS/CSCI/MGMT - 4600/6600

February 3rd, 2026

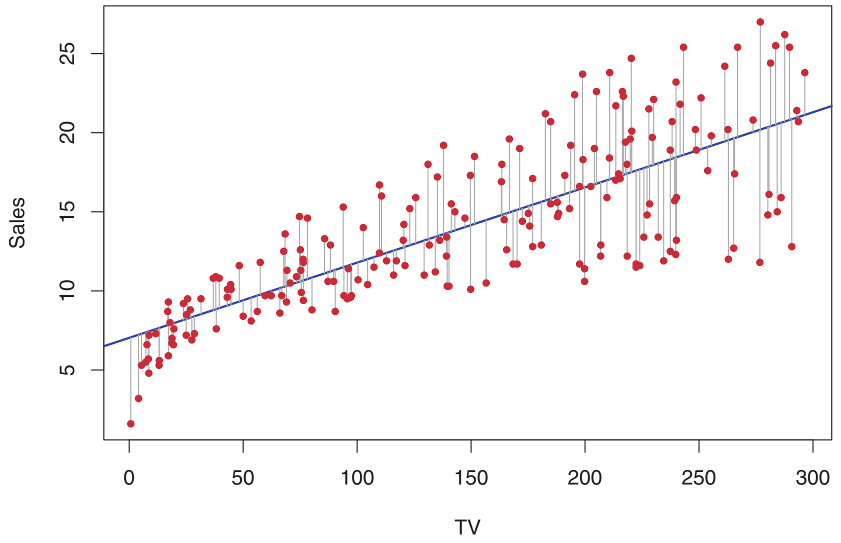
Tetherless World Constellation
Rensselaer Polytechnic Institute

Contents

Definitions

- **Independent variables/predictors/covariates:** Input variables used to predict or explain the outcome; the X variables in a model.
- **Dependent variable/target/response:** The output variable being predicted or explained; the Y variable in a model.
- **Model:** A mathematical function that maps input variables to output predictions:
  $\mathbf{Y = f(X) + \varepsilon}$ (where ε is a mean-zero random error term)
- **Feature space:** The n-dimensional space where each dimension represents a feature/predictor, and each data point is a location in this space.
- **Prediction error/residual:** The difference between the observed value and the predicted value:
  $\mathbf{e = y - \hat{y}}$

Regression

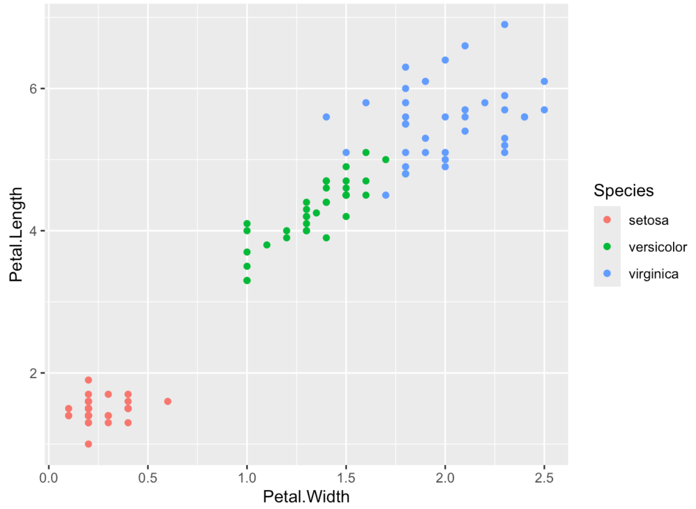


x-axis: independent numeric variable
y-axis: dependent numeric variable

Look for:
- Trend? direction?
- are points tightly grouped?

Goal: predict continuous response variable

Classification

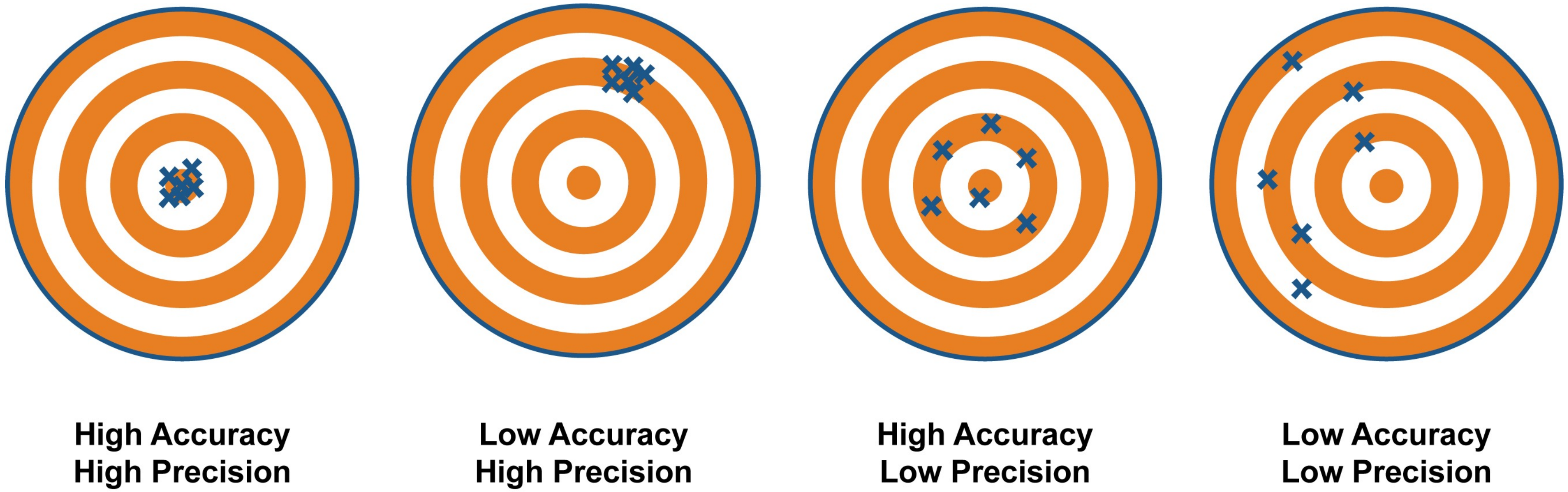


x-axis: numeric variable
y-axis: numeric variable

Look for:
- structure: groups? group separation/boundaries?

Goal: predict class label

Accurate vs. Precise



credit: climatica.org.uk (offline)

Regression –
Predicting Continuous Outcomes

Regression

- Supervised learning for continuous targets
- Models relationship between variables
- Produces numeric predictions

When to Use Regression?

- Target variable is continuous (price, temperature, sales)
- To understand variable relationships / forecast future values

e.g. Price prediction, demand forecasting

Linear Regression

- Fitting covariate and response data to a line is referred to as linear regression.

Definitions

Coefficient: numerical value that quantifies the relationship between predictor and response

Intercept: The expected value of the response variable when the value of the predictor variable is 0.
Slope: the average increase in Y associated with a one-unit increase in X.

Equation of line: $\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}x + \epsilon$

$\boldsymbol{\beta_0}$: intercept (Y when X = 0) • $\boldsymbol{\beta_1}$: slope (change in Y per unit change in X) • **ε**: error term

- Multiple methods for finding the best fit Line.

Some assumptions for linear regression

- **Linearity:** Relationship between predictor and response is linear
- **Independence:** Observations are independent (independently measured/sampled)
- **No multicollinearity:** Predictors not highly correlated
- **Homoscedasticity:** Constant variance of errors, i.e. errors/uncertainties are evenly spread out over range of inputs
- **Normality:** Errors are normally distributed

Least Squares Method

- Minimize sum of squared residuals
    - **Minimize $\Sigma(y_i - \hat{y}_i)^2$**
- Also called Ordinary Least Squares (OLS)
- Penalizes large errors more heavily
- Produces unique solution: line that minimizes total squared vertical distances from points

Least Squares Method

Equation of line: $\hat{y} = \widehat{\beta_0} + \widehat{\beta_1} x$

Let *n* be a positive integer. For a given data *(x_1,y_1), ..., (x_n,y_n)* $\in \mathbb{R}\times\mathbb{R}$,

- we obtain the intercept β_0 and slope β_1 using the least squares method.
- Residual Sum of Squares (RSS), the *i*th residual $e_i = y_i - \hat{y}_i$

$$\mathrm{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

Or

$$\mathrm{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

Least Squares Method

RSS can be re-written as:

$$\mathrm{L} \;=\; \sum_{i=1}^{n}(y_i \;-\widehat{\beta_0}-\widehat{\beta_1}x_i)^2$$

$\hat{y}_i$

$\downarrow$

Sum of squared distances between $(x_i,\, y_i)$ and $(x_i\;, \widehat{\beta_0}+\widehat{\beta_1}x_i\;)$ over i = 1,…,n

Least Squares Method

$$\mathrm{L} = \sum_{i=1}^{n} (y_i - \widehat{\beta_0} - \widehat{\beta_1} x_i)^2$$

- We partially differentiate *L* by β_0 and β_1 and let them be equal to zero, we obtain the following equations:

$$\frac{\partial L}{\partial \widehat{\beta_0}} = -2 \left(\sum_{i=1}^{n} \left(y_i - \widehat{\beta_0} - \widehat{\beta_1} x_i\right)\right) = 0 \qquad \text{Eq(1)}$$

$$\frac{\partial L}{\partial \widehat{\beta_1}} = -2 \left(\sum_{i=1}^{n} x_i \left(y_i - \widehat{\beta_0} - \widehat{\beta_1} x_i\right)\right) = 0 \qquad \text{Eq(2)}$$

Where the partial derivative is calculated by differentiating each variable and regarding the other variables as constants. In this case, β_1 and β_0 are regarded as constants when differentiating *L* by β_0 and β_1 respectively.

Least Squares Method

- By solving Eq (1) and Eq (2) when:

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 \neq 0 \qquad \text{Eq(3)}$$

i.e., $x_1 = x_2 = \cdots = x_N$ is not true.

where: $\bar{x}$ is the mean of X and $\bar{y}$ is the mean of Y.

- We can obtain:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})} \qquad \text{Eq(4)}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x} \qquad \text{Eq(5)}$$

$\longrightarrow$ Equation of line: $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$

Full derivation: https://www.youtube.com/watch?v=ewnc1cXJmGA

Linear Models are Estimates

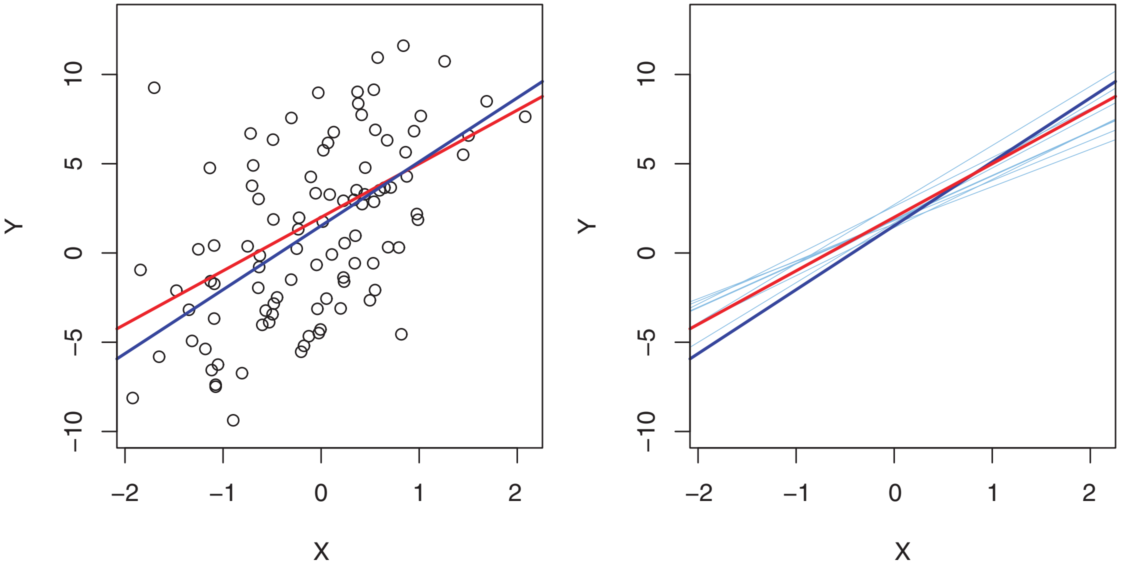
True relationship between X and Y: ***Y =2 + 3X + ε***

- Where ϵ is a mean-zero random error

Dark Red: true relationship

Dark Blue: least squares regression line

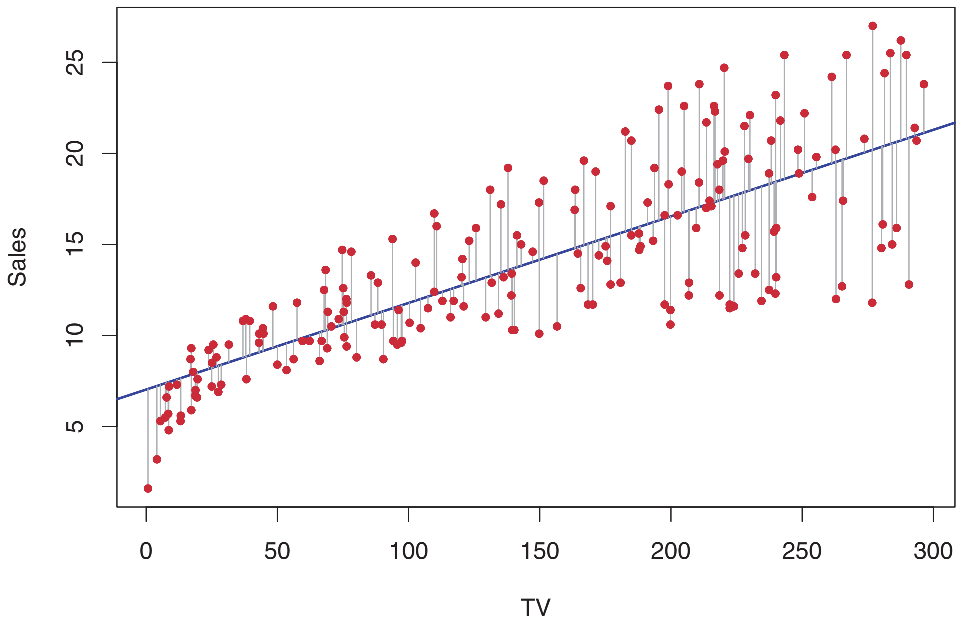
Light Blue: least squares regression lines for multiple new random datasets generated using the same model



*Remember, we are estimating models based on samples to learn about populations…

Evaluating Linear Models

- Sales vs. TV ad spending
- Sales in 1000s of units
- TV ad spending in 1000s of $



Evaluating Linear Models

1. Assessing the accuracy of coefficient estimates

1.1 Values of coefficients >> their Std. errors

1.2. High t-statistic

1.3. Low p-value

	Coefficient	Std. error	t-statistic	p-value
Intercept	7.0325	0.4578	15.36	< 0.0001
TV	0.0475	0.0027	$\boxed{17.67}$	$\boxed{< 0.0001}$

$$t = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)}$$

Hypothesis (more TV ads → more sales)

H0 : There is no relationship between X and Y

Ha : There is some relationship between X and Y

Reject the null hypothesis!

Evaluating Linear Models

2. Assessing the accuracy of the model

Residual Standard Error

- Mean sales ≈ 14,000 units
- RSE = 3.26 = 3,260 units
  good/bad?

R^2

- measures the proportion of the variability in *Y* that can be explained using *X*
- has a value between 0,1

Quantity	Value
Residual standard error	3.26
R^2	0.612
F-statistic	312.1

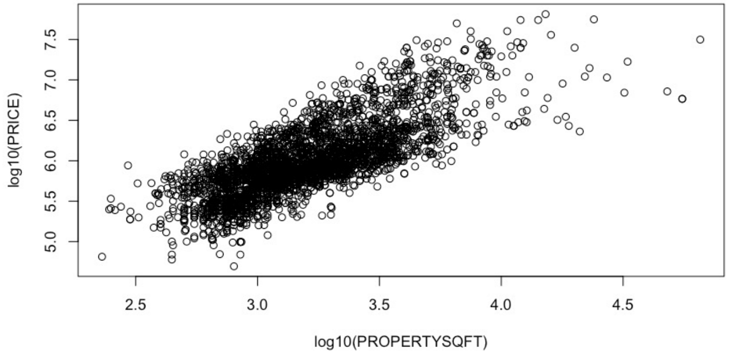
$$\mathrm{RSE} = \sqrt{\frac{1}{n-2}\mathrm{RSS}} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

$$\mathrm{TSS} = \sum(y_i - \bar{y})^2$$

Exercise: Linear models with the NY Housing Dataset

```
## read dataset
NY_House_Dataset <- read_csv("~/Courses/Data
Analytics/Spring26/datasets/NY-House-Dataset.csv")

dataset <- NY_House_Dataset

lmod1 <- lm(log10(PRICE)~log10(PROPERTYSQFT), data =
dataset)

## print model output
summary(lmod1)

## scatter plot of 2 variables with best fit line
plot(log10(PRICE)~log10(PROPERTYSQFT), data = dataset)
abline(lmod1)
```



Code: https://rpi.box.com/s/ysgt4r7ttajlygdxh63v72lfs4besypt

Classification – Predicting Class Labels

k-Nearest Neighbors

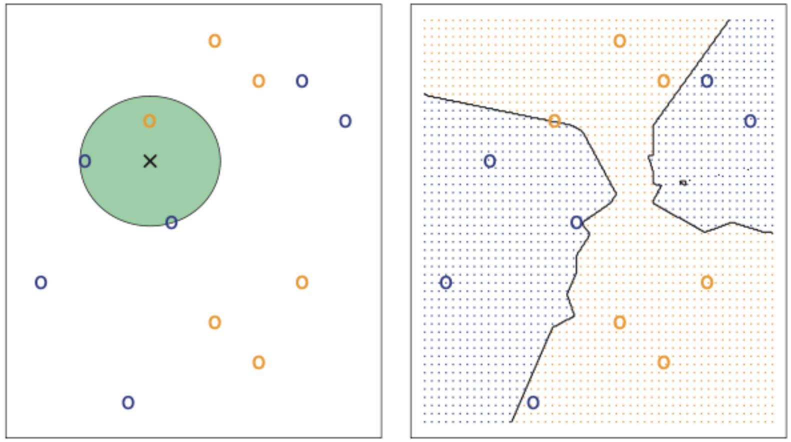
k-Nearest Neighbors

- Supervised learning for categorical targets
- Binary classification:
    - Two classes (Yes/No, Cat/NotACat)
    - Multi-class: More than two classes (Low/Medium/High)

e.g. email spam detection, disease diagnosis, customer segmentation

k-Nearest Neighbors

- In the figure a dataset is shown consisting 6 blue and 6 orange observations.
- Our goal is to make a prediction for the point labeled by the X
- Suppose we choose *k*=3, then KNN will first identify the three observations that are closest to the X as shown in the figure.
- This neighborhood is shown as a circle. It consist of 2 blue points and 1 orange point, resulting in estimated probabilities of 2/3 for the blue class and 1/3 for the orange class.
- Hence, kNN will predict that the X belongs to the blue class.



k = 3 neighborhood and decision boundaries

k-Nearest Neighbors

Pros:

- Simple and intuitive
- No training phase (lazy learning)
- Non-parametric (no assumptions about data)
- Effective with sufficient data

Cons:

- Computationally expensive for large datasets
- Sensitive to feature scaling
- Requires choosing k
- Storage intensive (must keep all training data)

Exercise: kNN with Iris dataset

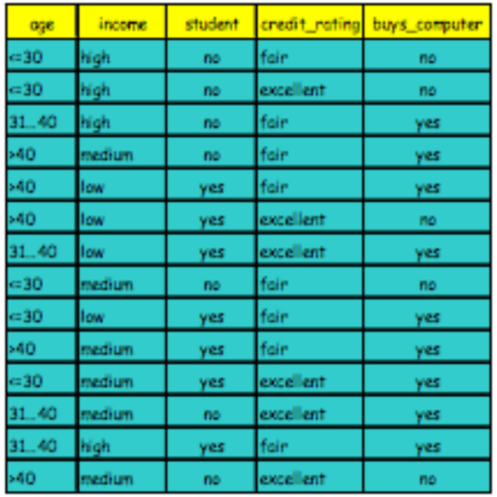
```
iris.data <- iris

s.train <- sample(150,100)

# creat training and testing sets
iris.train <-iris[s.train,]
iris.test <-iris[-s.train,]

## kNN Model
knn.predicted <- knn(iris.train[,1:4], iris.test[,1:4],
iris.train[,5], k=3)

## confusion matrix/contingency table
table(knn.predicted, iris.test[,5],
dnn=list('predicted','actual'))
```

Code: https://rpi.box.com/s/ysgt4r7ttajlygdxh63v72lfs4besypt

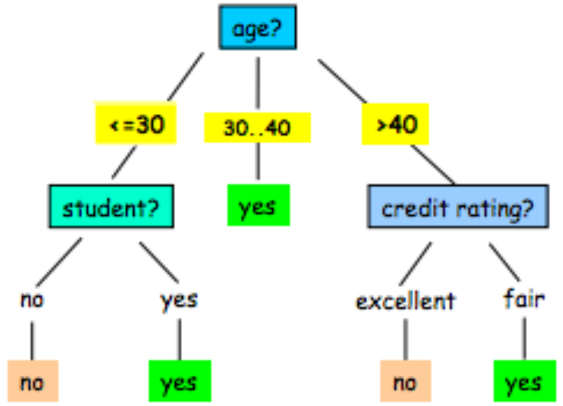
Decision Trees

Decision tree classifier

Classification by Decision Tree Induction



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



buys_computer ?

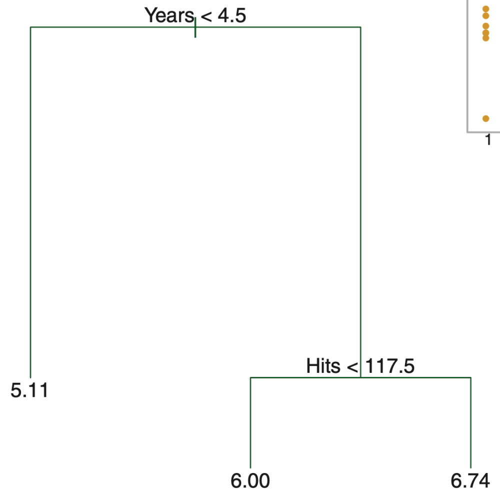
More on this later in Group 2 ...

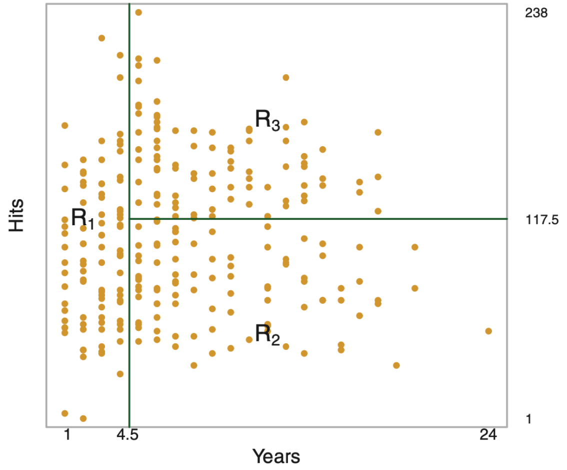
Decision Tree

- A decision tree has a hierarchical structure with “nodes” and “directed edges”
- The top node is defined as the “root node” and the nodes at the bottom are called as “leaf nodes”
- Nodes that are neither root node nor the leaf nodes are identified as “internal nodes” in the decision tree.
- There is a “class label” or numerical value associated with each leaf node
- Decision trees can be applied to both regression and classification problems

Decision Tree - Regression

e.g. predicting a player's figure salary (order of magnitude) based on their experience, stats, etc.

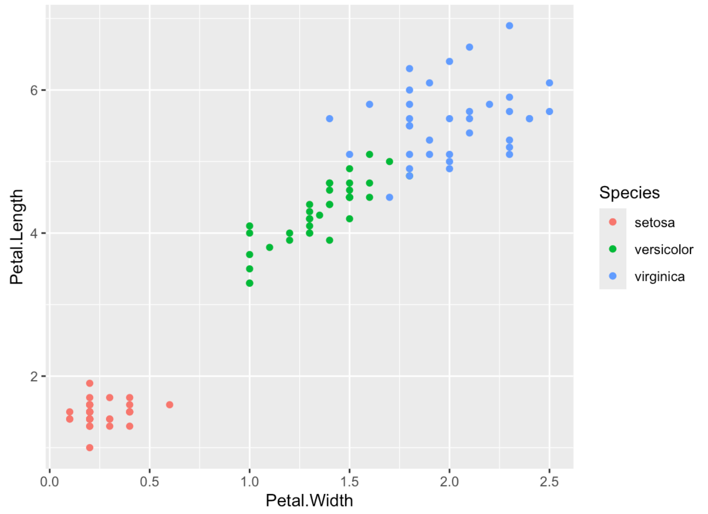


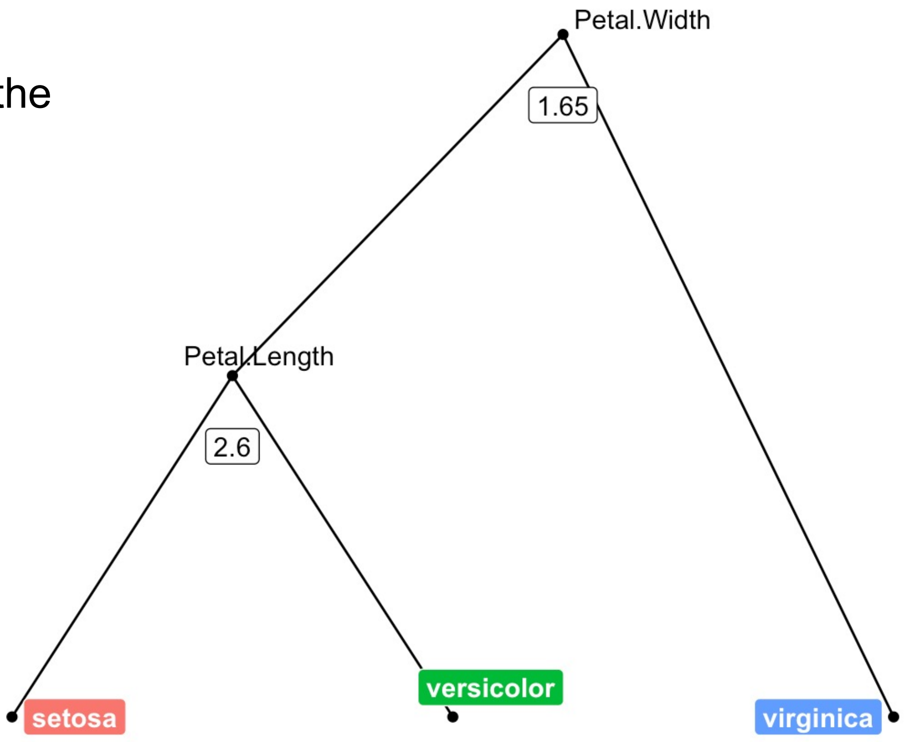


Reference/Resources: Introduction to Statistical Learning with R -7 Edition: Chapter 8

Decision Tree – Classification

e.g. predicting a flower's species based on the measurements of its petals, sepals, etc.





Decision Tree - Classification

• When we implement decision trees for classification, the idea is to split the data into subsets. So that each subset belongs to one particular class.

• In other words, splitting the data into regions, that are separated by decision boundaries, where each region's samples have only one class.

• Classification decisions are made by traversing the decision tree

• Traversing starts from the root node (from the top of the tree).

• **When a leaf node is reached through traversing, the category of the leaf node determines the classification**.

Decision Tree

- The **depth is measured from the root node** and the **depth at the root node is zero**.

- **The depth of the decision tree**: Tree Depth is calculated by counting the number of edges in the longest path from the root node to a leaf node.

- Number of nodes in the decision tree determine the size of the tree.

- **The decision tree constructing algorithm is referred to as a tree induction algorithm**.

Decision Tree

Pros:

• Trees can be displayed graphically, and easily interpreted even non-experts (especially if the tree is small) can understand.
• Some people believe decision trees are more closely mirror human decision-making.

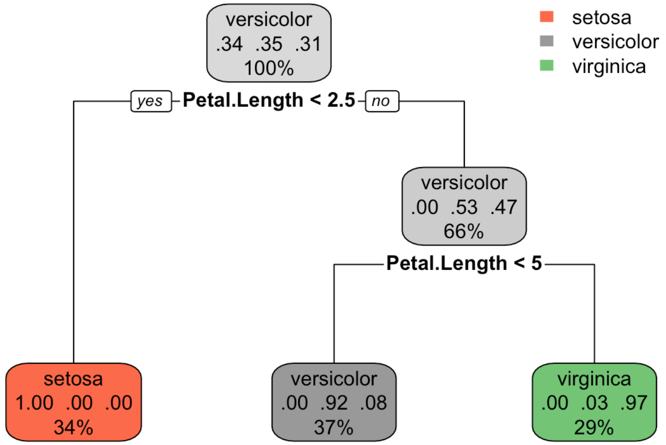
Cons:

• Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.

•Trees can be very non-robust. In other words, a small changes in the data can cause a large change in the final estimated tree

Exercise: Decision Tree with Iris dataset

```
iris.data <- iris

s.train <- sample(150,100)

# creat training and testing sets
iris.train <-iris[s.train,]
iris.test <-iris[-s.train,]

# Decision tree model
tree.model <- rpart(Species~., iris.train, method = "class")

#plotting the decision tree model using rpart.plot() function
rpart.plot(tree.model)

tree.model.predicted <-  predict(tree.model, iris.test, type = "class")

## confusion matrix/contingency table
table(tree.model.predicted, iris.test$Species, dnn=list('predicted','actual'))
```



Code: https://rpi.box.com/s/2wg4obl8ajrc1qm12rirdffylz96yn1d

Random Forest(s)

Random Forest

- Random Forest is based on decision trees.
- Random Forest improves on decision trees by using Bootstrap Aggregation (Bagging)
- The original algorithm was created in 1995 by Tin Kam Ho.
- An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006.

http://www.stat.berkeley.edu/~breiman/RandomForests/

Bootstrap Aggregating (Bagging)

- Build multiple decision trees
  - Each tree trained on random subset of data
  - Each split considers random subset of features
  - Final prediction: majority vote (classification) or average (regression)
- Reduces variance through averaging
  - Less prone to overfitting
  - More stable predictions
  - Captures different patterns with different trees
- Sample with replacement

Random Forest

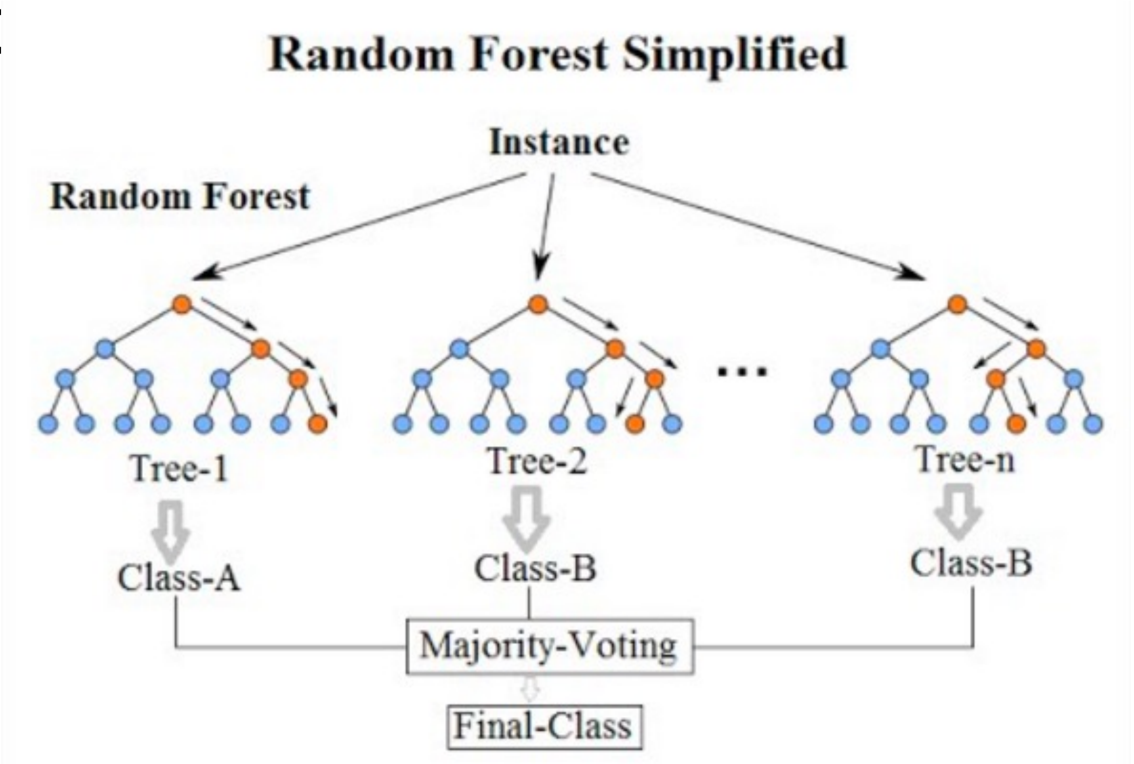


Image Resource: https://commons.wikimedia.org/wiki/File:Random_forest_diagram_complete.png

Random Forest

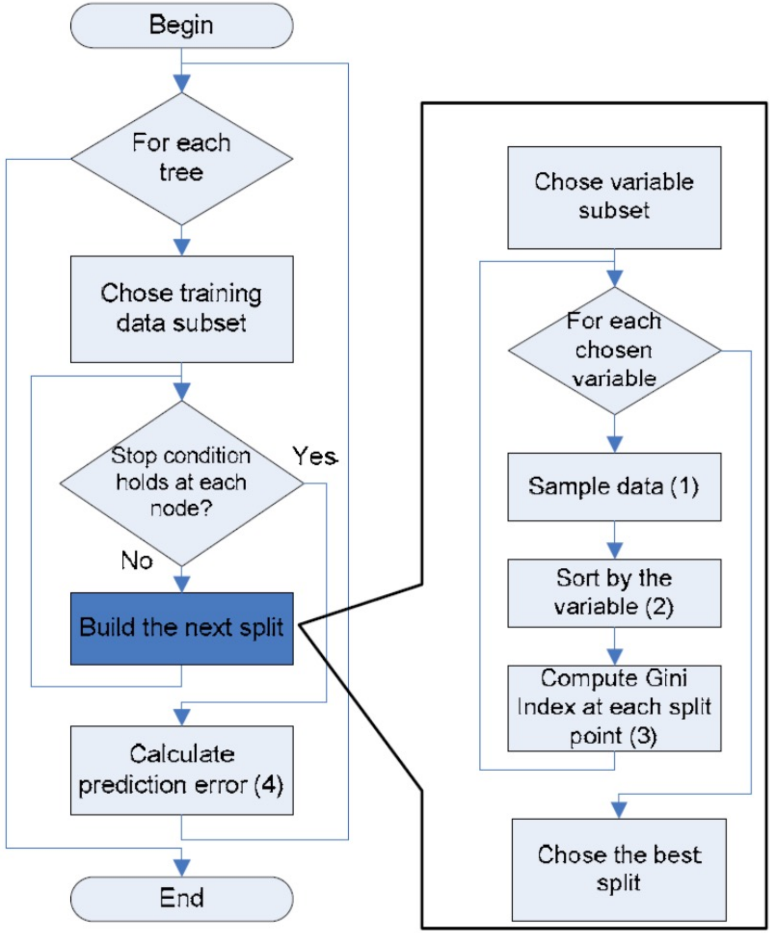
- Decide number of trees to build (e.g., 100, 500)
- Decide number of features to consider at each split

For each tree:
- Bootstrap sample: randomly select *n* samples from training data WITH replacement

Grow the Tree:
- At each node, randomly select a subset of features
- Find the best split among only those selected features
- Split the node based on best feature
- Repeat recursively until stopping criteria met:
    - Maximum depth reached
    - Minimum samples in node
    - Node is pure (all same class)

Store the Tree
- Save the completed tree as part of the forest

End Training



Image/ Photo Credit: Albert A. Montillo

When to Stop splitting the nodes?

There are several criteria that can be used to determine the when a node shouldn’t be split into subsets:

- W**hen all samples in the node have the same class label**.

- Since getting pure subsets is difficult to archive with real world data, **the stopping criteria can be modified to a certain percentage of the samples in the node. i.e 95% of have the same class label.**

- When the **number of samples** in the node falls below a certain **minimum number**.

- When the improvement in **impurity measure** is way too small to measure (too small to make a much difference in classification result).

- The algorithm can also stop expanding when it reaches **maximum tree-depth**.

Impurity Measure

- A decision tree will select the split that minimize the Gini-index.

Gini = 1 - $\Sigma(p_i)^2$

where p_i is the proportion of samples belonging to class *I*

e.g. Binary Classification (Classes A & B)

Pure Node
100 samples, all Class A
$p_1 = 1.0$, $p_2 = 0.0$
Gini = $1 - (1.0^2 + 0.0^2) = 1 - 1 =$ **0** (perfectly pure)

Maximum Impurity
100 samples, 50 Class A, 50 Class B
$p_1 = 0.5$, $p_2 = 0.5$
Gini = $1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) =$ **0.5** (maximum impurity for binary)

Impurity Measure

• Besides the Gini-index, there are other impurity measures available such as:

- entropy or information gain
- misclassification rate

• A tree will test all variables to determine the best way to split a node using a purity measure such as Gini-index to compare different possibilities

• **Tree induction algorithms repeatedly split nodes to get more and more homogeneous subsets**.

Random Forest

Pros

- Higher accuracy than single tree
- Handles large datasets efficiently
- Estimates feature importance
- Handles missing values well

Cons

- Speed - with larger more complex datasets
- Interpretability – many trees are difficult (may be impossible) to explain or visualize collectively
- May overfit with noisy datasets

Errors in Classification

- In classification, the model's output is the predicted class label for the input variables and the true class label is the target.
- **If the predicted class label is different from the actual class label (true class) then there is an error with that classification**.
- The error rate is the percentage of errors made over the entire dataset.
- Error rate is also known as the misclassification rate or simply called the error.

Error = (*Number of Misclassifications*)/(Total Number of Samples)

Accuracy = (*Number of Correct Classifications*)/(Total Number of Samples)

Confusion Matrix / Contingency Table

e.g.

predicted ↓ / actual →	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	16	2
virginica	0	0	17

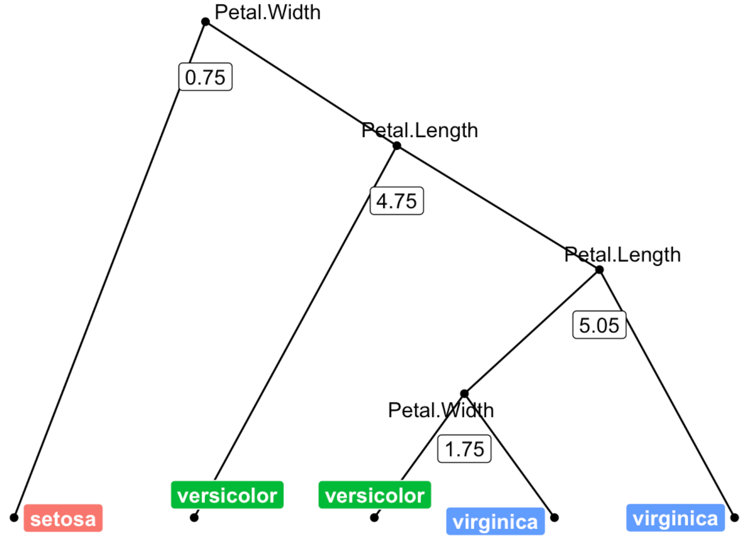
- Evaluating a kNN model trained on 2/3 of observations (100) in the Iris dataset and tested on the remaining 50

Error = 2/50 = 0.04 = 4%

Accuracy = 48/50 = 0.96 = 96%

Exercise: Decision Tree with Iris dataset

```
iris.data <- iris

s.train <- sample(150,100)

# creat training and testing sets
iris.train <-iris[s.train,]
iris.test <-iris[-s.train,]

## Random Forest Model
rf.model <- randomForest(Species~., data=iris.train, proximity=TRUE)

## predict class labels
rf.predicted <- predict(rf.model, iris.test)

## confusion matrix/contingency table
table(rf.predicted,iris.test$Species, dnn=list('predicted','actual'))
```



Code: https://rpi.box.com/s/ysgt4r7ttajlygdxh63v72lfs4besypt

Read More:

ISLR Seventh Printing
https://rpi.box.com/s/qvr6phv8iuwdn0xwrnwy3kh1ardokn77

- Chapter 2 + first section from chapters 3 & 4

Next Class: Friday Feb. 6th

Lab 2

Thanks!