

## Class 7 Supervised Learning for STP (II): Supervised Learning

Dr Wei Miao

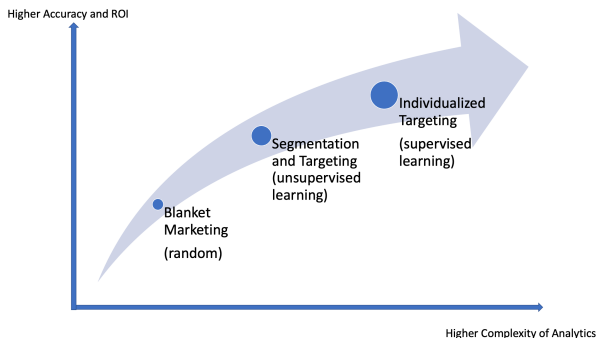
UCL School of Management

Thu, Oct 27 2022

## Section 1

# Supervised Learning

# Motivation: Why Supervised Learning for Business?



# Data Generating Process

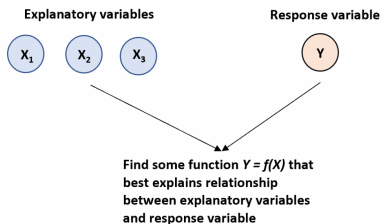
$$Y = f(X; \theta) + \epsilon$$

- $f$  is the function that characterizes the true relationship between  $X$  and  $Y$ 
  - $f$  is often called Data Generating Process (DGP), which is NEVER known to us
- $Y$  the **response/outcome/explained** variable to be predicted
  - e.g., whether customer responds to our offer (1/0)
- $X = (X_1, X_2, \dots, X_p)$  are a set of **predictors/features/explanatory variables**
  - customers' past purchase history (e.g., spending in each category)
  - demographic variables (e.g., income, age, kids, etc.)
- $\theta$  represents the set of **function parameters** of the DGP  $f$
- $\epsilon$  is the random **error term**, with mean zero.

# Supervised Learning

- A **supervised learning model** is used when we have one or more explanatory variables and a response variable and we would like to find some function that describes the DGP between the explanatory variables and the response variable as accurately as possible.

## Supervised Learning



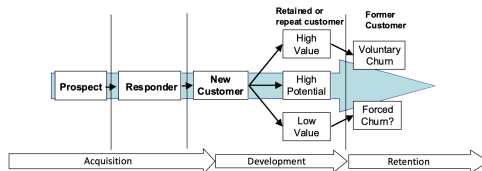
- Since we never know the true DGP: All models are wrong, but some are useful<sup>1</sup>

<sup>1</sup>A famous [quote](#) by statistician George Box.

# Types of Supervised Learning Tasks for Customer Relationship Management

Depending on the type of the **response variable**, supervised learning tasks can be divided into two major groups

- **Classification tasks:** outcome is a categorical variable
  - Whether a customer responds to marketing offers (acquisition)
  - Whether a customer churns (retention)
- **Regression tasks:** outcome is a continuous variable
  - Probability of customers responding to marketing offers (acquisition)
  - Customer total spending in each period (development)
  - Probability of customer churn (retention)



# Difference between Supervised and Unsupervised Learning

	Supervised Learning	Unsupervised Learning
<b>Description</b>	Involves building a model to estimate or predict an output based on one or more inputs.	Involves finding structure and relationships from inputs. There is no “supervising” output.
<b>Variables</b>	Explanatory and Response variables	Explanatory variables only
<b>End goal</b>	Develop model to <b>(1)</b> predict new values or <b>(2)</b> understand existing relationship between explanatory and response variables	Develop model to <b>(1)</b> place observations from a dataset into a specific cluster or to <b>(2)</b> create rules to identify associations between variables.
<b>Types of algorithms</b>	<b>(1)</b> Regression and <b>(2)</b> Classification	<b>(1)</b> Clustering and <b>(2)</b> Association

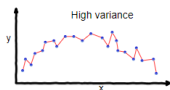
# Commonly Used Supervised Learning Models

- Linear regression class models (easy to interpret, low accuracy)
  - OLS, Lasso, Ridge regressions
- **Tree-based Models (good balance between interpretability and accuracy)**
  - Decision tree, random forest
- Neural-network based models (hard to interpret, high accuracy)
  - Deep learning models

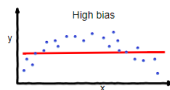


# Overfitting and Underfitting

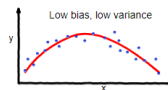
- **Overfitting** means a predictive model corresponds too closely to **historical data**, and therefore fails to predict **new data** reliably.
  - Overfitting leads to high **variance error**, which is the error from being unable to predict for new data points
- **Underfitting** occurs when a predictive model cannot adequately capture the underlying structure of **historical data**.
  - Underfitting leads to high **bias error**, which means high prediction error on the historical data, as the model fails to sufficiently capture the relations between  $X$  and  $Y$
  - An underfitting model also tends to have poor predictive performance for future data points.



overfitting



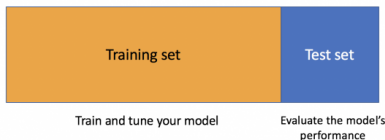
underfitting



Good balance

# Mitigate the Underfitting and Overfitting Problems

- To mitigate the underfitting problem, select better models and tune the parameters.
- To mitigate the overfitting problem, when building predictive models, we need to divide the **full labelled historical data** into different sets. The percentage of split depends on the context.
  - **Training set** (70% - 80% of labelled data): train the model parameters
  - **Test set** (20% - 30% of labelled data): evaluate the prediction accuracy



- For more complicated models with hyper-parameters such as deep learning models, we may even need to split our data into 3 sets (training, validation, and test sets). You will learn more details in term 2 from ML specialization modules.

## Section 2

# Decision Tree and Random Forest

# Introduction to Decision Tree

- There are many methodologies for constructing regression trees but one of the oldest and most commonly used is known as the **classification and regression tree (CART)** approach developed by Breiman et al. (1984).

## Motivation Example: Predicting mpg from mtcars

- From mtcars, we want to predict the outcome variable mpg based on cyl and hp
  - Predictors  $X$  include cyl and hp
  - Outcome variable  $Y$  is mpg

```
1  pacman::p_load(dplyr,modelsummary)
2  data("mtcars")
3  # Generate a new data for the task
4  data_decision_tree <- mtcars %>%
5    select(mpg,cyl,hp)
6  # check first 4 rows
7  data_decision_tree %>%
8    head(n = 4)
```

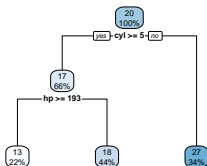
	mpg	cyl	hp
Mazda RX4	21.0	6	110
Mazda RX4 Wag	21.0	6	110
Datsun 710	22.8	4	93
Hornet 4 Drive	21.4	6	110

# Implementation of Decision Tree in R

- Package `rpart` provides implementation of decision trees in R
  - `rpart()` is the function in the package to train a decision tree; refer to its help function for more details.
- Package `rpart.plot` provides nice visualizations of decision trees

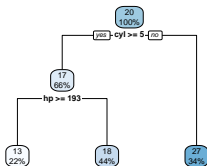
```
1 # Load the necessary packages
2 pacman::p_load(rpart,rpart.plot)
3 # Below example shows how to train a decision tree
4 tree1 <- rpart(
5   formula = mpg ~ cyl + hp,
6   data    = data_decision_tree,
7   method  = "anova"
8 )
```

# Intuition behind Decision Trees



- 1 First, we find that `cyl` matters more than `hp` in terms of predicting `mpg`, so we split two branches according to `cyl`. We find that  $\{4\}$  and  $\{6,8\}$  split can best differentiate car models.
  - `cyl = 4`, go to the left branch
  - `cyl = 6 or 8`, go to the right branch

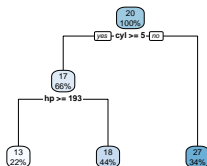
# Intuition behind Decision Trees



- ② For car models with 4 cylinders, we try all possible splits based on `hp`, and find that 193 can best differentiate car models.
- `hp < 193`, use the average `mpg` in that **terminal node**, 13, as the predicted `mpg`
  - `hp >= 193`, use the average `mpg` in that **terminal node**, 18, as the predicted `mpg`



# Intuition behind Decision Trees



- ④ For car models with more than 5 cylinders, we find that `hp` does not differentiate car models, so we do not further split.
  - Use the average `mpg` in that **terminal node**, 27, as the predicted `mpg`

# Pros and Cons of Decision Trees

## Advantages of Decision Trees

- They are very interpretable.
- Making predictions is fast.
- It's easy to understand what variables are important in making the prediction. The internal nodes (splits) are those variables that most largely reduce the SSE (criteria for split).

## Disadvantages of Decision Trees

- Single regression trees have high variance (overfitting), resulting in unstable predictions
- Due to the high variance, single regression trees tend to have poor predictive accuracy.

# Random Forest

To overcome the overfitting tendency of a single decision tree, random forest has been developed by (Breiman 2001).

## ① Bootstrap subsampling

- Each tree is grown to a bootstrapped subsample

## ② Split-variable randomization

- each time a split is to be performed, the search for the split variable is limited to a random subset of  $m$  out of the  $p$  variables. For regression trees, typical default values are  $m = p/3$ .

# Visualization of Random Forest



- Each tree gives a prediction
- Random forest takes the majority of voting as the final prediction

# Implementation of Random Forest in R

- Package ranger provides implementation of random forest in R.
- `ranger()` is the function in the package to train a random forest; refer to its help function for more details.
- The following code shows how to train a random forest consisting of 500 decision trees, where the outcome variable is mpg, and the predictors are 5 car attribute variables.

```
1  pacman::p_load(ranger)
2  randomforest1 <- ranger(
3    formula    = mpg ~ hp + cyl + disp + wt + gear,
4    data       = mtcars, # dataset to train the model
5    num.trees  = 500, # 500 decision trees
6    seed      = 888 # make sure of replication
7  )
```

# Make Predictions from Random Forest

- After we train the predictive model, we can use `predict()` function to make predictions
  - The 1st argument is the trained model object
  - The 2nd argument is the dataset to make predictions on

```
1 # Make predictions on the mtcars
2 prediction_rf <- predict(randomforest1,
3                           data = mtcars)
4
5 # Because prediction_rf is a list object
6 # Need to use $ to extract the predicted value as a numeric vector
7 prediction_rf$predictions
```

```
[1] 20.80814 20.80526 24.13373 20.14718 17.50231 18.98494 14.73153 24.
[9] 22.84680 18.73638 18.73638 16.26754 16.26130 16.17000 11.96695 11.
[17] 13.35940 29.91299 30.71855 31.04063 22.58045 16.30967 16.48256 14.
[25] 17.42216 29.60584 26.18860 28.29520 15.87313 20.14866 14.97831 22.
```

## After-Class Reading

- (optional) Varian, Hal R. "Big data: New tricks for econometrics." Journal of Economic Perspectives 28, no. 2 (2014): 3-28
- (recommended) [Decision tree in R](#)
- (recommended) [Random forest in R](#)