Class 9 Supervised Machine Learning and Tree-Based Models

Dr. Wei Miao

UCL School of Management

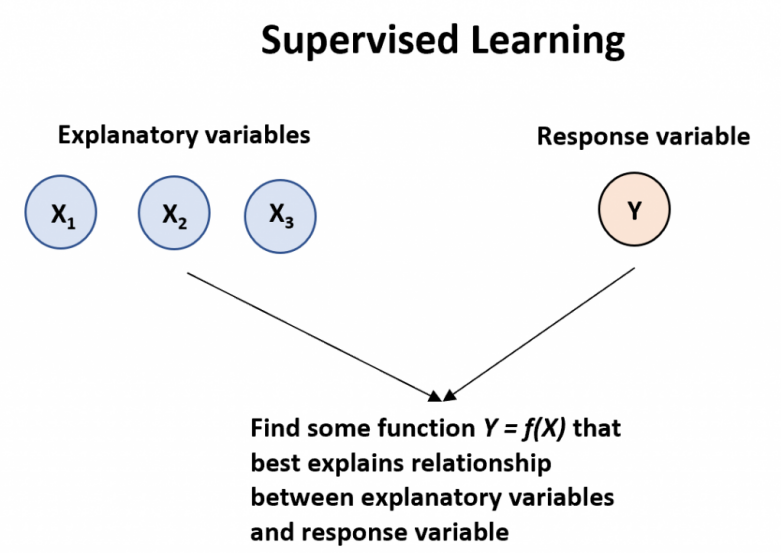
October 30, 2024

Section 1

Supervised Learning

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 learn the **underlying true relationship** between the **explanatory** variables and the **response** variable as accurately as possible.



Data Generating Process (DGP)

We use the following notations for supervised learning tasks:

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

- Y is the **response/outcome/target** variable to be predicted
- $X = (X_1, X_2, ..., X_p)$ are a set of **explanatory variables/features/predictors**
- $f(X; \theta) + \epsilon$ is the true relationship between X and Y, or DGP, which is never known to us[1]; ϵ is the **randomness term** or **error term**
- θ represents the set of **parameters** to be learned from the data

[1] "All model are wrong, but some are useful" – George Box. As business analysts, we need to use the "wrong models" correctly.

Types of Supervised Learning Algorithms

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

- **Classification tasks** if the outcome is **categorical**
  - Whether a customer responds to marketing offers (e.g., 1 for response, 0 for no response)
  - Whether a customer churns (e.g., 1 for churn, 0 for no churn)
  - Which product a customer purchases (e.g., 1 for product A, 2 for product B, etc.)
- **Regression tasks** if the outcome is **continuous**
  - Customer total spending in each period (e.g., $100, $200, etc.)
  - Demand forecasting such as the daily sales of a product (e.g., 100 units, 120 units, etc.)

Difference between Supervised and Unsupervised Learning

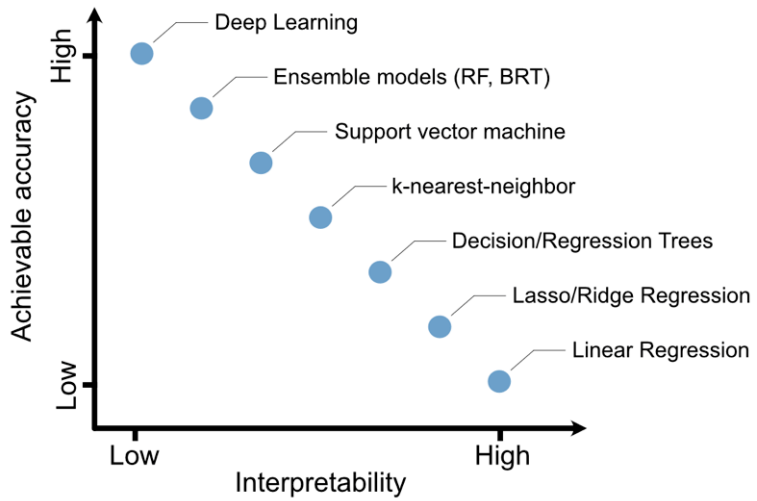
	Supervised Learning	**Unsupervised Learning**
Description	Estimate or predict an output based on one or more inputs.	Find structure and relationships from inputs. No "supervising" output.
Variables	Explanatory and Response variables	Explanatory variables only
Goal	(1) predict new values or (2) understand existing relationship between explanatory and response variables	place observations from a dataset into a specific cluster
Types of algorithms	(1) Regression and (2) Classification	Clustering

Section 2

Fundamental Tradeoffs

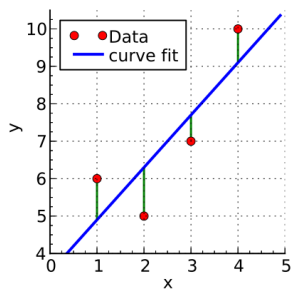
Accuracy-Interpretability Tradeoff

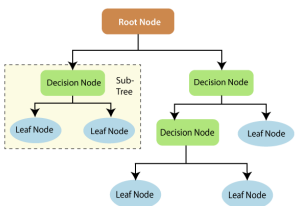
- Simpler models are easier to interpret but gives lower accuracy
- Complicated models can give better prediction accuracy but results are hard to interpret

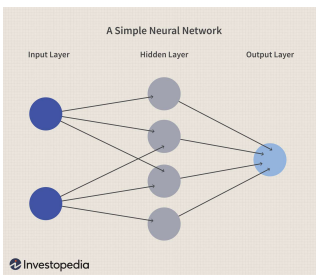


Comparison of Classic Supervised Learning Models

- Linear regression class models (easy to interpret, low accuracy)
  - Linear regression coefficients have economic interpretations but prediction accuracy is low
- **Tree-based Models (balance between interpretability and accuracy)**
  - **Decision tree**, **random forest**, and gradient boosting models
- Neural-network based models (hard to interpret, high accuracy)
  - Deep learning only give estimated weights that have no direct business interpretations





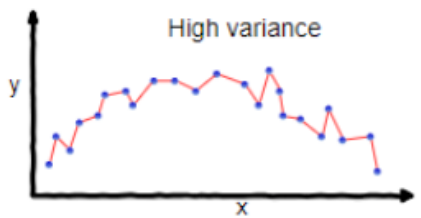


Bias Error and Variance Error

- After we have trained a machine learning model, we can test the model performance by looking at the errors of predictions.
  - **bias** is the prediction error of the model on the historical data
  - **variance** is the prediction error of the model on new data

Bias-Variance Tradeoff

- If a predictive model **fits historical data too well**, then it may not be flexible enough and thus have a higher chance of failing to make predictions for new data accurately. This problem is called **overfitting**.
- Overfitting leads to **low bias** but **high variance**. But this is not good for us because with supervised learning models, we want to have higher prediction accuracy for the future. Hence we face a **bias-variance tradeoff** or **bias-variance dilemma**.



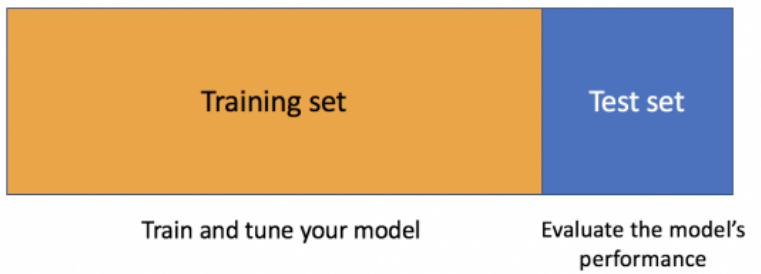
overfitting

Section 3

Overfitting and Underfitting

How to Mitigate Overfitting

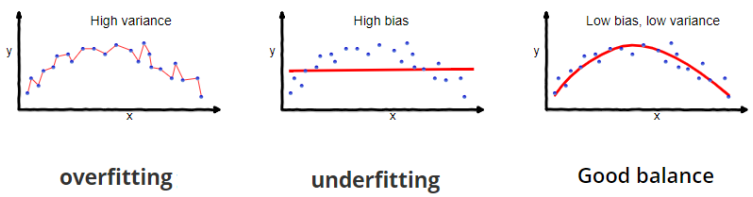
- To mitigate the overfitting problem, when training predictive models, we need to use the **cross-validation** technique by splitting the full **historical data** into a **training set** and a **test set**.
  - **A training set** (70% - 80% of labelled data)**:** we train the ML model based on the training set.
  - **A test set** (20% - 30% of labelled data)**:** Using the trained ML model from the training data, we can make predictions for the test data. However, we do observe the actual outcomes for the test set, so that we can evaluate the prediction accuracy by comparing the predicted outcomes versus the actual outcomes.



- 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).

Underfitting

- **Underfitting** occurs when a predictive model cannot sufficiently capture the DGP even on historical data.
- Underfitting leads to **high bias** as well as **high variance**. Thus, underfitting is the worst case, which should be avoided by all means.
- To mitigate the underfitting problem, we need to select more suitable models.

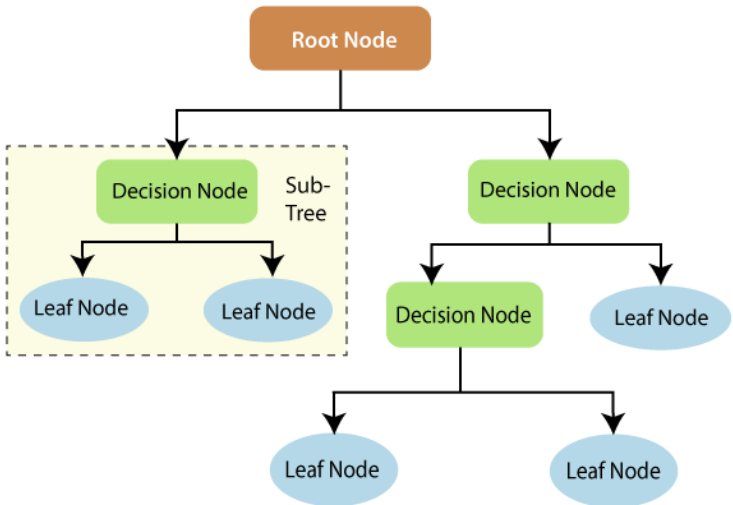


Section 4

Decision Tree

Introduction to Decision Tree

- A **decision tree** is a tree-like structure, which can be used for both classification and regression tasks.



Business Objective: Predict Customer Response to Marketing Offers

- M&S made marketing offers to customers in the data, and the variable `Response` represents whether or not customers responded to our offer in the previous similar marketing campaign.
- **Business objective**: Based on our historical data `data_full`, we want to train a decision tree model to predict the outcome variable `Response` based on `Recency` and `totalspending`.
- **Data collection and cleaning:**

```
pacman::p_load(dplyr, modelsummary)

data_full <- read.csv("https://www.dropbox.com/scl/fi/2q7ppqtyca0pd3j48
    mutate(totalspending = MntWines + MntFruits +
    MntMeatProducts + MntFishProducts +
    MntSweetProducts + MntGoldProds)
```

Implementation of Decision Tree in R

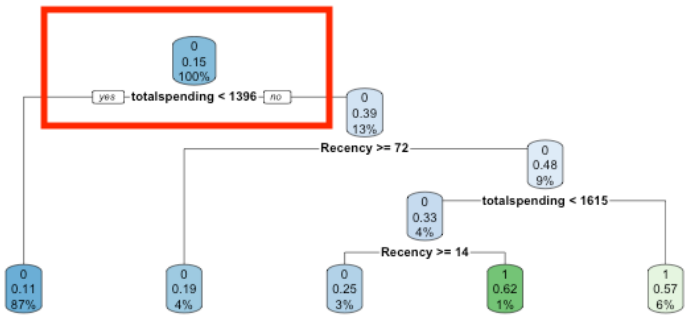
- Package `rpart` provides efficient implementation of decision trees in R; Package `rpart.plot` provides visualizations of decision trees
  - `formula`: `Response ~ Recency + totalspending` means that we want to predict the outcome variable `Response` based on the explanatory variables `Recency` and `totalspending`. In R, we use `~` to separate the outcome variable and the explanatory variables for all supervised learning tasks.
  - `data`: the dataset to train the model
  - `method`: "class" for classification tasks, "anova" for regression tasks

```
# Load the necessary packages
pacman::p_load(rpart,rpart.plot)

# Below example shows how to train a decision tree
tree1 <- rpart(
  formula = Response ~ Recency + totalspending,
  data    = data_full,
  method  = "class" # classification task; or 'anova' for regression
  )

# visualize the tree
rpart.plot(tree1)
```

How Decision Tree Works: Step 1



Step 1. Decision tree (DT) will try to split customers into 2 groups based on each unique value of each variable, and see which split can lead to customers being most different in terms of outcome `Response`.

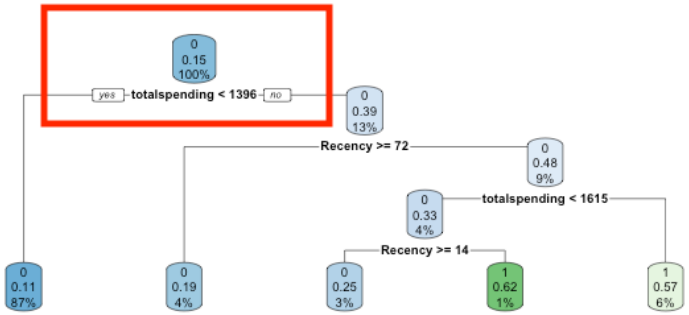
```
data_full$totalspending |> unique() |> sort() |> head(10)
```

```
 [1]  5  6  8  9 10 11 12 13 14 15
```

```
data_full$Recency |> unique() |> sort() |> head(10)
```

```
 [1] 0 1 2 3 4 5 6 7 8 9
```

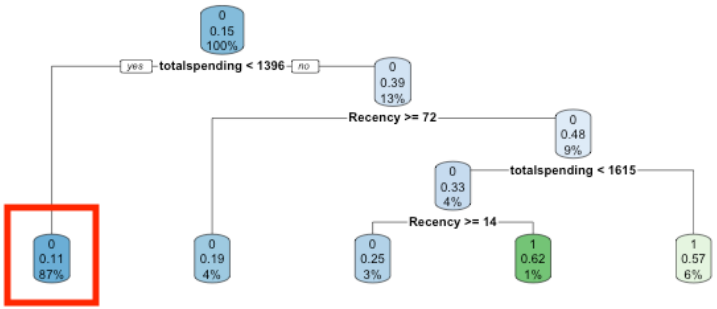
How Decision Tree Works: Step 1



Step 1. Decision tree (DT) will try to split customers into 2 groups based on each unique value of each variable, and see which split can lead to customers being most different in terms of outcome `Response`.

- After this step, DT finds that total spending is the best variable and 1396 is the best cutoff.
- DT therefore splits customers into 2 groups based on 1396.
- In each node, the 3 numbers are (1) predicted outcome (2) predicted probability of outcome being 1, and (3) share of customers in the node

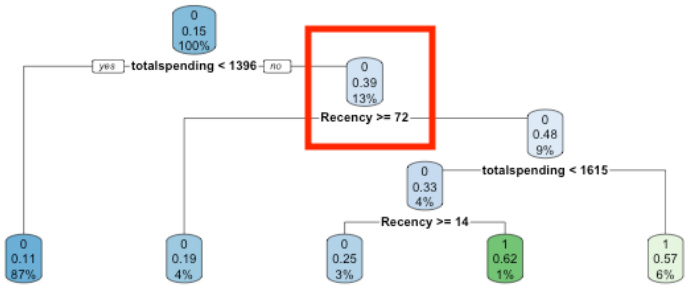
How Decision Tree Works: Step 2



Step 2. For customers in the left branch (`totalspending` < 1396), DT will continue to split based on each unique value of each variable, and see which split can result in the customers to be most different in terms of `Response`.

- However, DT couldn't find a cutoff that sufficiently differentiate customers, so DT stops in the left branch.

How Decision Tree Works: Step 3 ...



Step 3. For customers in the right branch (`totalspending` $>=$ 1396), DT will continue to split based on each unique value of each variable, and see which split can result in the customers to be most different in terms of Response.

- After this step, DT finds `Recency` is the best variable and 72 is the best cutoff. DT further splits customers into 2 groups.

Step 4. This process continues until DT determines that there is no need to further split customers.

How Decision Tree Works: Step 4

- Once the tree is fully grown, we can use the tree to make predictions on new customers.
- For a new customer, we can follow the tree from the root node to the leaf node, and the predicted outcome is the outcome of the leaf node.
- In R, we can use the `predict()` function to make predictions on new customers, which returns the predicted outcome of the new customers. Note that, the test data should have the same variable names as the training data.

```
# Make predictions on the mtcars
prediction_tree1 <- predict(tree1,
                            data = data_test)
```

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).

Section 5

Random Forest

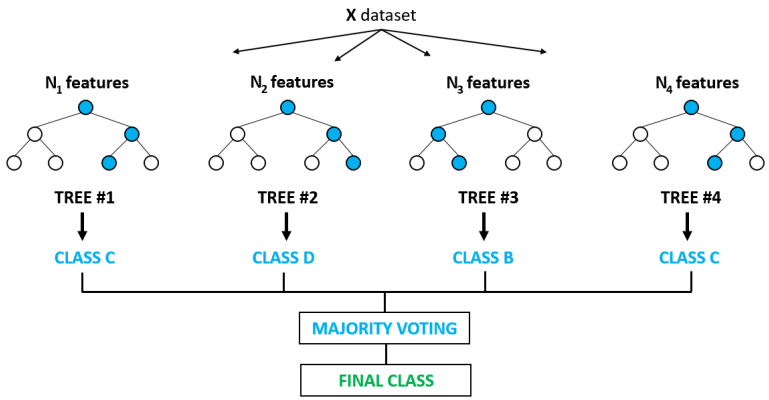
Disadvantages of Decision Trees

- Single regression trees tend to overfit, 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).
  - Instead of using all customers, each tree is grown to a **subsample** of customers instead of all customers (e.g., 70% of training data)
  - Instead of using all features for splitting, each tree is grown to a **subset** of features instead of all features (e.g., 3 out of 5 features)

Visualization of Random Forest



For a new customer,

- Each tree gives a prediction of the outcome
- Random forest takes the average (for regression tasks) or majority vote (for classification tasks) of all trees' predictions 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.

```
pacman::p_load(ranger)
randomforest1 <- ranger(
    formula   = Response ~ totalspending + Recency, # formula
    data      = data_full, # dataset to train the model
    num.trees = 500, # 500 decision trees
    seed = 888, # make sure of replication
    probability  = TRUE # to return predicted probabilities
  )
```

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

```
# Make predictions on the mtcars
prediction_rf <- predict(randomforest1,
                         data = data_full)

# Because prediction_rf is a list object
# Need to use $ to extract the predicted value as a numeric vector
prediction_rf$predictions
```

After-Class Reading

- (recommended) Decision tree in R
- (recommended) Random forest in R

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.