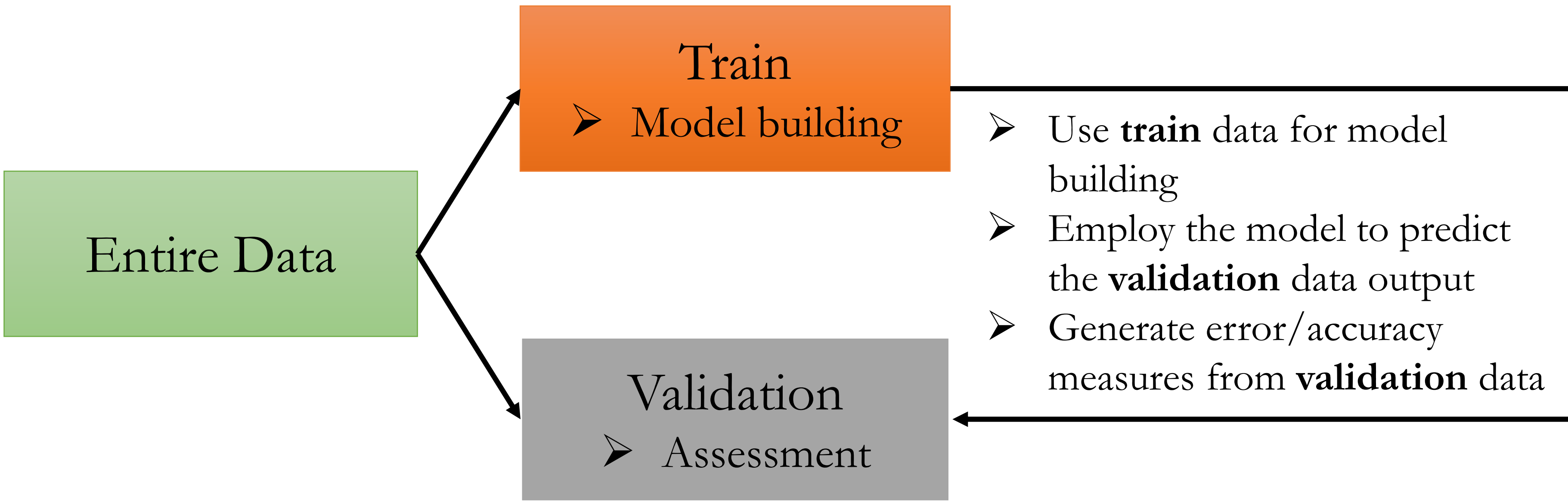
Model Evaluation & Accuracy Measures
Classification

Previous class

- Data partition : Train & Validation
- Model Evaluation and Accuracy measures for Regression
- Implementation in R/RStudio

Data Partition : Training & Validation



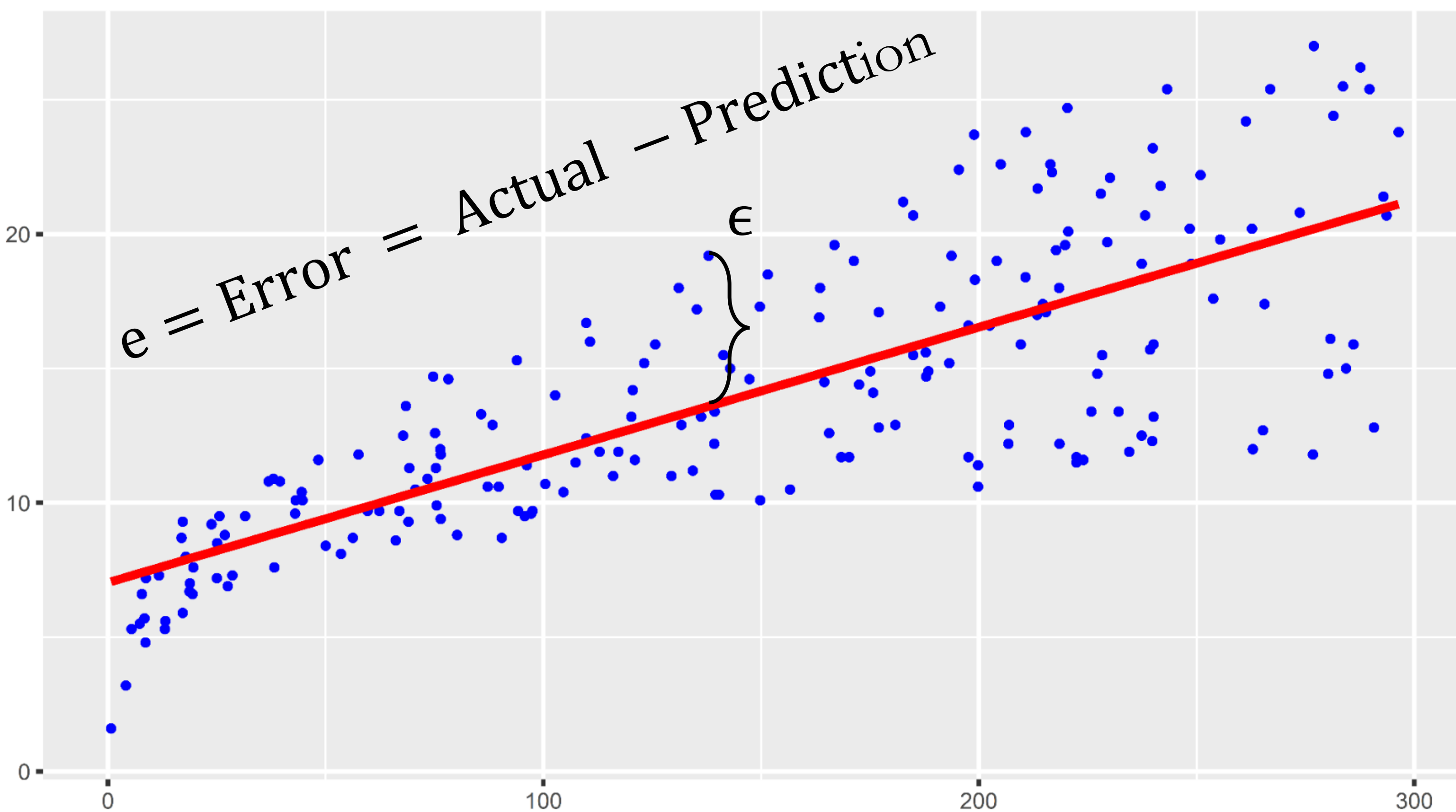
- Quantify & assess the model performance

Why this process?

Model built directly using the **entire** data does not demonstrate/quantify the predictive ability of new data

Error

- Error (e_i) for each observation i
- Error (e_i) : Difference between actual (Y_i) and predicted outcome ($\hat{Y}_i$)



Accuracy measures for Regression

- Mean Error (ME) : $\frac{1}{n}\sum_{i=1}^{n} e_i$
    - ➢ Indicates on-average predictions are over or under the outcome
- Mean Absolute Error (MAE) : $\frac{1}{n}\sum_{i=1}^{n}|e_i|$
    - ➢ Magnitude of average absolute error
- Mean Percentage Error (MPE) : $\left(\frac{1}{n}\sum_{i=1}^{n}\frac{e_i}{Y_i}\right) * 100$
    - ➢ Measure relative to the size of outcome Y_i
- MAPE (Mean Absolute Percentage Error) : $\left(\frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{Y_i}\right|\right) * 100$
- Root Mean Square Prediction Error (RMSE) : $\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$
    - ➢ This measure has same units as outcome Y_i

Error/Accuracy Measures

- We computed error measures for **validation** data
- Can they be computed for **training** data?
- What do the measures infer for each data?

Training

- Goodness-of-fit
- Additional measures - R^2, standard error
- Does not indicate predictive abilities

Validation

- Indicates predictive abilities
- Used to **compare across models** to assess their degree of prediction accuracy

- **Overfitting** can be detected by comparing the error measures between **training** and **validation** data
- Greater the difference in train & validation data error measures, greater the overfitting

Model Evaluation & Accuracy Measures
Classification

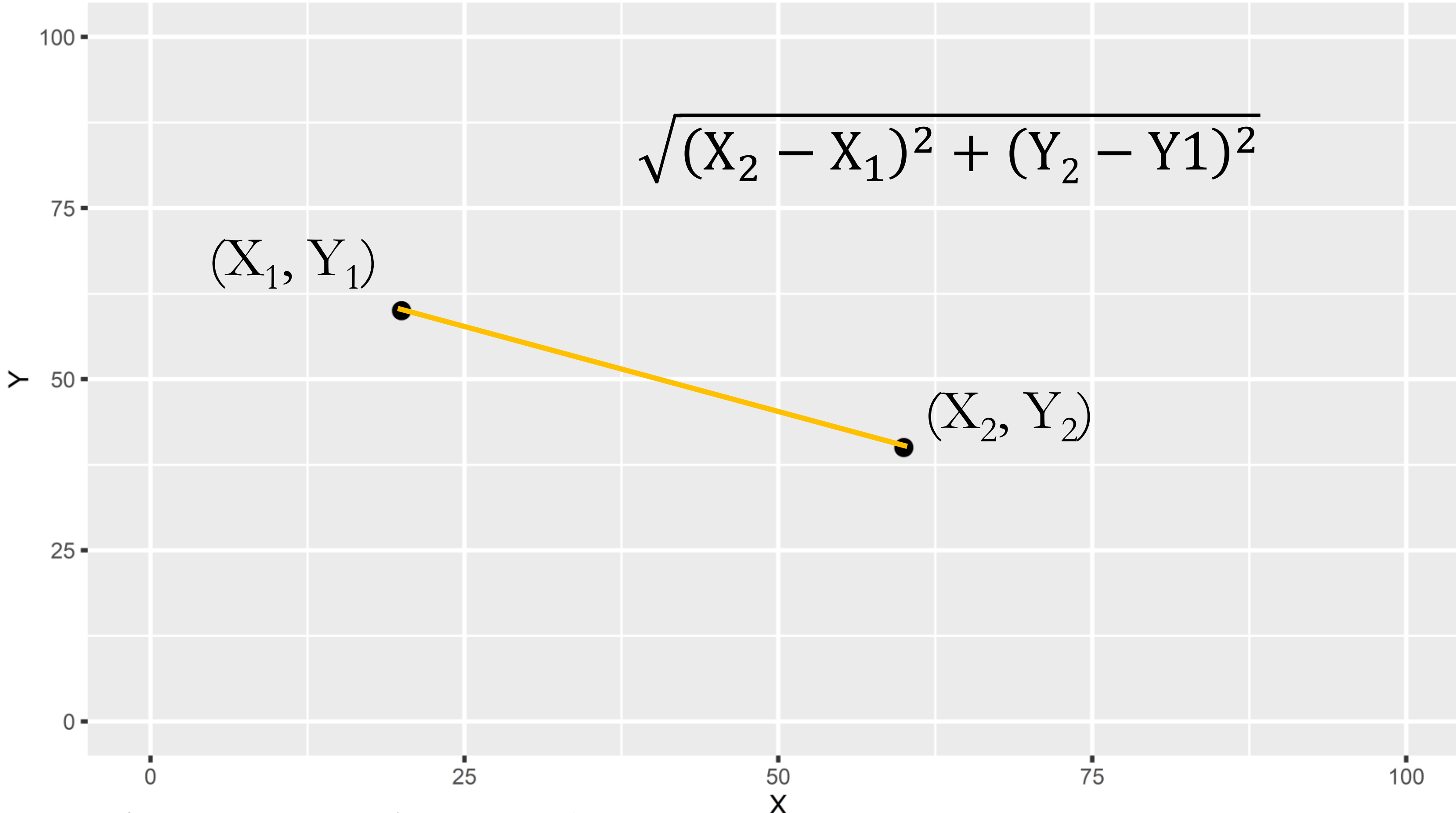
Recap of
k-Nearest Neighbor (*k*-NN)
as Classification

k-NN

- Simplest Machine Learning/Predictive algorithm
- Method relies on finding "**similar**" observations in the data
- Similar observations are also referred as "**Neighbors**"
- "**Neighbors**" are used to derive prediction for a new observation

Nearest ? Distance ?

- Euclidean Distance



- Predictors must be unit free
- Solution : **Standardization/Normalization**

Standardization/Normalization

- Subtract mean from each observation
- Divide the result by standard deviation

X
64
18
24
46
72

- **m = mean(c(64, 18, 24, 46, 72))**
- **s = sd(c(64, 18, 24, 46, 72))**
- X_norm = (X-m)/s

X_norm
0.8076
-1.1273
-0.8749
0.0505
1.1441

- Mean of normalized data is 0
- Standard deviation of normalized data is 1

Data on Riding Mowers

- Riding-mower manufacturer would like to find a way of classifying families in a city into an **Owner** or **Nonowner**
- Attributes
  - Income : Income of the household in thousand of dollars
  - Lot Size : Lot size in thousand of square foot
  - Ownership : Owner or Nonowner

Income	Lot_Size	Ownership
60	18.4	Owner
85.5	16.8	Owner
64.8	21.6	Owner
61.5	20.8	Owner

⋮

⋮

Today's class mandatory steps

- Create a folder name "**i.model evaluation_classification**" within the folder "**oba_455_555_ddpm_r/rproject**"
- Download "**model evaluation_classification_code.R**", and all **csv** files from canvas
- Place all downloaded files in
  "**oba_455_555_ddpm_r/rproject / i.model evaluation_classification**"
- Open RStudio project
- Open "**model evaluation_classification_code.R**" file within RStudio

Riding Mowers data partition

- How many observations result from 70% of riding mowers data?
- ~ 17 observations → Train data
- 7 observations → Validation data
- Few observations in validation data
- Let us use 60-40 partition due to few (24) observations
- **Train** : Randomly filter 60% of the main data
- **Validation** : Extract remaining 40% of the main data
- Let us build the k-NN classification model on train data for $k = 3$

k-NN as classification model in R

- Step 1 : Train data
  - Standardize the input numeric variables
  - Convert input character variables into dummy (binary) variables
- Step 2: Pick only standardized input numeric & dummy variables in train data
  - **Standardized train data**
- Step 3 : Validation data – prediction of interest
  - Standardize the input numeric variables
  - Convert input character variables into dummy variables
- Step 4: Pick only standardized input numeric & dummy variables in validation data
  - **Standardized validation data**
- Step 5 : Track the output variable in the train data
  - **Train data output**
- Step 6 : Execute the function "**knn**" to predict output in validation data

Prediction in the validation dataset

```
> validation %>%
+   select(Ownership_actual, Ownership_prediction)
# A tibble: 10 x 2
   Ownership_actual Ownership_prediction
   <fct>            <fct>
 1 Owner            Nonowner
 2 Owner            Owner
 3 Owner            Owner
 4 Owner            Owner
 5 Owner            Owner
 6 Owner            Nonowner
 7 Owner            Nonowner
 8 Nonowner         Nonowner
 9 Nonowner         Nonowner
10 Nonowner         Nonowner
```

- In Regression problem, actuals and predictions are numeric values
- Error (e_i) = Actual– Prediction ; We computed MAPE, ME, RMSE……
- How to measure accuracy measure here when you have categorical actuals and predictions?

Confusion/Classification Matrix

- Table/Matrix summarizes the correct and incorrect classifications

		Actual/Reference	
		C_1	C_2
Prediction	C_1	**Correct Classification (n_{11})**	**Incorrect Classification (n_{12})**
	C_2	**Incorrect Classification (n_{21})**	**Correct Classification (n_{22})**

- Rows & columns of matrix correspond to predicted and actual classes
- Diagonal cells (**upper left**, **lower right**) give correct classification
- Off-diagonal cells (**upper right**, **lower left**) give misclassification

Confusion/Classification Matrix

		Actual/Reference	
		C_1	C_2
Prediction	C_1	Correct Classification (n_{11})	Incorrect Classification (n_{12})
	C_2	Incorrect Classification (n_{21})	Correct Classification (n_{22})

- Total observations in **validation** data, $n = n_{11} + n_{12} + n_{21} + n_{22}$
- Estimated misclassification rate, $\text{err} = \frac{n_{12}+n_{21}}{n}$
- Accuracy $= 1 - \text{err} = \frac{n_{11}+n_{22}}{n}$

Confusion Matrix for validation data

		Actual/Reference	
		Nonowner	Owner
Prediction	Nonowner	**3 (n_{11})**	**3 (n_{12})**
	Owner	**0 (n_{21})**	**4 (n_{22})**

- Total observation in **validation** data $n = n_{11} + n_{12} + n_{21} + n_{22} = 10$
- Estimated misclassification rate, $err = \frac{n_{12}+n_{21}}{n} = \frac{3}{10} = 30\%$
- Overall Accuracy $= 1 - err = \frac{n_{11}+n_{22}}{n} = \frac{7}{10} = 70\%$
- Remember all these results are for $k = 3$

Unequal importance of classes

- Sometimes it is **more important** to predict a membership correctly in class C_1 than in class C_2
- Example : Predicting financial status (bankrupt/solvent) of firms
- Predicting **bankrupt** status is more important than **solvent**
- Overall Accuracy is not a good measure under unequal importance of classes
- Measures : **Sensitivity** and **Specificity**

Confusion Matrix

		Actual/Reference	
		C_1	C_2
Prediction	C_1	Correct Classification (n_{11})	Incorrect Classification (n_{12})
	C_2	Incorrect Classification (n_{21})	Correct Classification (n_{22})

- Let's say the important class is C_1
- **Sensitivity** : Ability to **detect** the important class members correctly

  ➢ $$\frac{n_{11}}{n_{11} + n_{21}}$$

- **Specificity** : Ability to **rule out** non-important class members correctly

  ➢ $$\frac{n_{22}}{n_{22} + n_{12}}$$

Confusion Matrix output from RStudio

```
Confusion Matrix and Statistics

          Reference
Prediction Nonowner Owner
  Nonowner        3     3
  Owner           0     4

               Accuracy : 0.7
                 95% CI : (0.3475, 0.9333)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.6496

                  Kappa : 0.4444

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.5714
         Pos Pred Value : 0.5000
         Neg Pred Value : 1.0000
             Prevalence : 0.3000
         Detection Rate : 0.3000
   Detection Prevalence : 0.6000
      Balanced Accuracy : 0.7857

       'Positive' Class : Nonowner
```

Confusion Matrix

Accuracy

Sensitivity and Specificity

```
Confusion Matrix and Statistics

          Reference
Prediction Nonowner Owner
  Nonowner        3     3
  Owner           0     4

               Accuracy : 0.7
                 95% CI : (0.3475, 0.9333)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.6496

                  Kappa : 0.4444

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.5714
         Pos Pred Value : 0.5000
         Neg Pred Value : 1.0000
             Prevalence : 0.3000
         Detection Rate : 0.3000
   Detection Prevalence : 0.6000
      Balanced Accuracy : 0.7857

       'Positive' Class : Nonowner
```

What k to use ?

- Remember we predicted the output in the validation data using the model built on train data with $k = 3$
- Retrieve the accuracy values by varying from $k = 1$ to 14
- Choose k with the highest accuracy

```
"The accuracy for k =  1 is 0.7"
"The accuracy for k =  2 is 0.7"
"The accuracy for k =  3 is 0.7"
"The accuracy for k =  4 is 0.5"
"The accuracy for k =  5 is 0.7"
"The accuracy for k =  6 is 0.6"
"The accuracy for k =  7 is 0.7"
"The accuracy for k =  8 is 0.6"
"The accuracy for k =  9 is 0.6"
"The accuracy for k =  10 is 0.3"
"The accuracy for k =  11 is 0.3"
"The accuracy for k =  12 is 0.3"
"The accuracy for k =  13 is 0.3"
"The accuracy for k =  14 is 0.3"
```

Recap of the Process

- **Step 1** : Partition the entire data into two parts
  - **Train** : Randomly filter X % of the main data
  - **Validation** : Extract the remaining (1-X%)
- **Step 2** : Build the model on the **train** data
- **Step 3** : Compute accuracy measures on the **validation** data
- What is the **drawback** in the above process?
- Model building & performance evaluation based on **one** random partition
- That one partition can result in excellent model build and performance
- What happens for a different random partition?
- How to overcome this drawback?

Resampling

- Indispensable tool in Statistics/Machine Learning
- Idea
    - Repeatedly draw sample from the data
    - Fit model of interest on each sample
- Example
    - Fit Linear Regression on each repeated sample
    - Examine the extent to which results/accuracy measures differ across multiple validation datasets
- Computationally expensive
- Methods : **Cross-Validation** and **Bootstrap**

Next Class

- Cross-Validation
- Logistics Regression

Thank You