Statistical Computing ~

Survival Analysis - ML | January , 2022

Emmanuelle Rodrigues Nunes

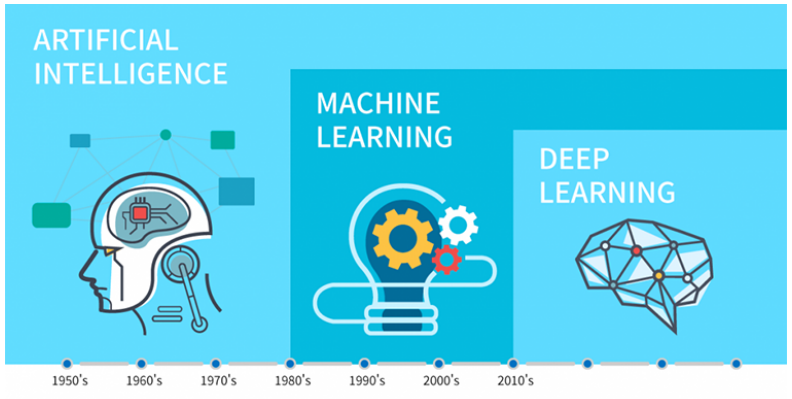
CIDRA-AIPD
African Institute for Professional Development

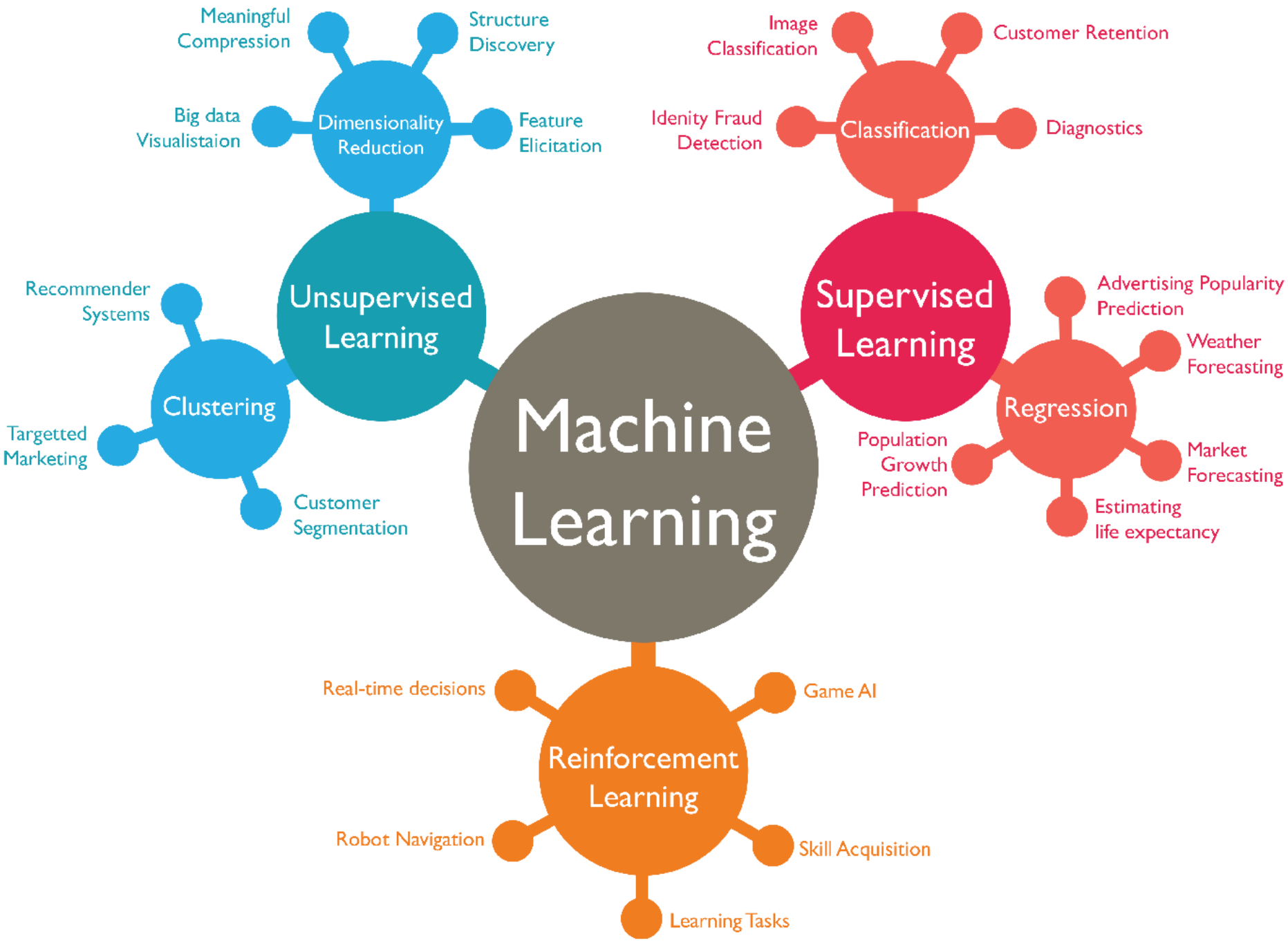
CIDRA-AIPD
African Institute
for Professional
Development

Machine Learning

Machine Learning - Concepts

- Machine Learning (ML) is a subset of Artificial Intelligence (AI)
- Algorithms that can improve automatically through experience and by the use of data without being explicit programmed, reason why we say that the algorithms learn.
- With ML algorithms we can build a model to make predictions or decisions.
- Machine learning algorithms are used many different applications, for example:
  - Medicine
  - Email filtering
  - Speech recognition
  - Computer vision





CIDRA-AIPD
African Institute for Professional Development

CIDRA-AIPD
African Institute for Professional Development

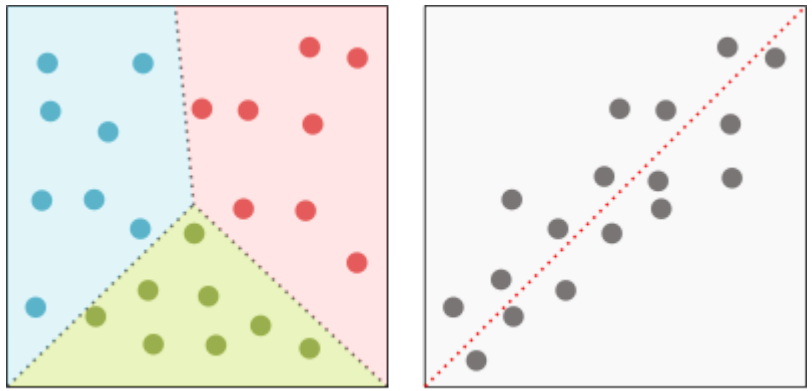
Supervised Learning

Supervised Learning

Supervised learning is where you have input variables (X) and an output variable (y) and you use an algorithm to learn the mapping function from the input to the output.

$$y = f(X)$$

- It is the most common type of Machine Learning problem
- It is called **supervised** because we have the label that tell us the correct information, and we are going to be corrected if we predict wrong.
- Supervised learning can be grouped into two problems:
    - **Regression:** The output variable is a real number, for example, weight
    - **Clustering:** The output variable is a category, for example, disease and no disease



Steps

CIDRA-AIPD
African Institute
for Professional
Development

To solve this kind of problem we have some steps to follow:

1. Data collection
2. Exploratory Data Analysis (EDA)
3. Data Processing
4. Data Modelling

I am going to briefly explain each of these steps that will be clarified when we have the example.

Data Collection

CIDRA-AIPD
African Institute for Professional Development

Data is very important for building a Machine learning model, as the model will learn from the dataset provided.

Collecting, **cleaning** and managing your data properly is a key factor for any ML project.

Garbage in, Garbage out

Exploratory Data Analysis

Understand the characteristics and distribution of the data to help us understand what kind of model and what type of learning we are dealing with.

Data Preprocessing

CIDRA-AIPD
African Institute
for Professional
Development

Data preprocessing is a process of cleaning the data and enabling them in the right format so they can be consumed by the algorithm it consists of data cleaning, missing values handling, feature selection and feature engineering

If the dataset is not good and if there are missing values, outliers, or the features are not presented in a corrected format, the ML model built from this data will probably be bad.

Methods usually used: encoding, normalization, imputation, outliers and NaN rejections, feature selection

Encoding

Encoding is a method to transform the categorical data into numbers before using it to fit the model. For example, if we have the categories "cat" and "dog" we can encode them to 0 and 1.

Feature scaling

CIDRA-AIPD African Institute for Professional Development

Some ML models need to be on the "same range" to fit the data properly without the addition of bias due to different scales. For example, if we have as features height as 150cm and salary as 40000, salary might be more important due to potentially being in the 1000's but that's not necessarily true.

Methods:

- **Min-max normalization:** rescales the range of features to the new range in [0,1]

$$X_{normalised} = \frac{X - min(X)}{max(X) - min(x)}$$

- **Standardisation (Z-score normalisation):** Rescale the feature in a new range with a zero mean and a standard deviation is 1

$$X_{normalised} = \frac{X - \bar{X}}{\sigma_X}$$

Missing values

When we have missing data, the data won't be available, i.e., we have missing information. In this case we can:

- *Remove* the missing data
  - If enough data available
  - If the missing data does not have a pattern, i.e., is random
- *Inputation*: input the missing data using some statistical method
  - Use the mean or median of the variable to input missing values
  - Apply some algorithm, for example, linear or logistic regression to determine the likely value

Feature selection

Select the most relevant variables for the model. This can be used to:

- Simplify the model
- Reduce overfitting
- Reduce training time

Data Modelling

Split the data

Before applying the model we need to make sure that we have a dataset to train and evaluate the model and another one to test the accuracy of the model.

We need to have different datasets to train and test the model because we want the model to learn and be able to **generalise on unseen data**. If we use the same dataset for training and testing, we are "leaking" information to the test and we can't garantee that the model can generalise. Also, having this split help us with problems like **overfitting** and **underfitting**.

The splits of the data usually are:

- Training
- Validation
- Testing

We usually split the data into 80% for training, 10% for validation and 10% for testing.

Training set

It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

It should be large enough so that the model can “learn” correctly. Most of the data is used for training.

Validation Set

This set is used to validate our model performance during training. This validation process gives information that helps us tune the model’s hyperparameters and configurations accordingly.

- It tell us if the training is moving in the right direction or not.

The main idea of splitting the dataset into a validation set is to prevent our model from **overfitting** i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.

Test set

It tests the model after completing the training by providing an unbiased final model performance metric in terms of accuracy, precision, etc.

- It answers the question: "How well does the model perform?"

Modelling and evaluating the model

There are various machine learning models that you can choose according to the objective, for example, Linear Regression, Support Vector Machine (SVM), Decision Tree, Random Forest, K-Nearest Neighbors, Neural Network, K-means, etc.

We will see some of them when going through the examples.

Once we train the model, we need to evaluate it. Depending of your supervised learning problem, we have different metrics:

- **Classification metrics:** F1 score, precision, recall, accuracy
- **Regression metrics:** Mean Absolute Error, Mean Squared Error (MSE), Root Mean Square Error (RMSE), R^2 metric.

Prediction

When the best model is determined, it can be used to predict the new samples on the testing set.

CIDRA-AIPD
African Institute for Professional Development

Regression

Regression

CIDRA-AIPD
African Institute
for Professional
Development

Regression is a type of **supervised** learning. One example of algorithm that we can use for this kid of problem is a **Linear Regression**.

- Yes, it is the same Linear Regression as previously seem
  - In general, ML doesn't make any assumptions but the user should make sure that the data meet the assumptions of the linear regression model if he/she wants to use it

To explain how to solve a Regression problem, I am going to use the dataset *Medical insurance costs*. The data contains medical information and costs billed by health insurance companies. It contains 1338 rows of data and the columns: age, gender, BMI, children, smoker, region as features and insurance charges as the label.

I am going to use the R libraries `tidyverse` and `tidymodels`.

- `tidymodels` is a library for machine learning workflow

Regression example - Data collection

CIDRA-AIPD African Institute for Professional Development

The first step is to collect the data. The data is available at github and the link can be seen on the example.

```
# Load libraries
library(tidyverse)
library(tidymodels)

# Load data sets
url ← 'https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/insurance.csv'
insurance ← read_csv(url)
```

We can see some of the data by using `glimpse`. We can see that we have 1338 rows and 7 columns. Also, `sex`, `smoker` and `region` are characters that will be transformed to factors.

```
glimpse(insurance)
```

```
## Rows: 1,338
## Columns: 7
## $ age      <dbl> 19, 18, 28, 33, 32, 31, 46, 37, 37, 60, 25, 62, 23, 56, 27, 1…
## $ sex      <chr> "female", "male", "male", "male", "male", "female", "female",…
## $ bmi      <dbl> 27.900, 33.770, 33.000, 22.705, 28.880, 25.740, 33.440, 27.74…
## $ children <dbl> 0, 1, 3, 0, 0, 0, 1, 3, 2, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0…
## $ smoker   <chr> "yes", "no", "no", "no", "no", "no", "no", "no", "no", "no", …
## $ region   <chr> "southwest", "southeast", "southeast", "northwest", "northwes…
## $ charges  <dbl> 16884.924, 1725.552, 4449.462, 21984.471, 3866.855, 3756.622,…
```

Regression example - EDA

CIDRA-AIPD African Institute for Professional Development

I'll be using the package `dlookr` to help us with the EDA. We are displaying:

- na: Number of NA values
- mean, standard deviation and IQR
- p00: min value; p25: 1st quantile; p50: median; p75: third quantile; p100: max value

```
library(dlookr)
library(DT)
dlookr::describe(insurance, quantiles = c(0, 0.25, 0.5, 0.75, 1),
                 statistics = c("na", "mean", "sd", "IQR", "quantiles")) %>%
  DT::datatable(rownames = FALSE, options = list(dom = 't',scrollX = TRUE, scrollCollapse = TRUE)) %>% format
```

described_variables	n	na	mean	sd	IQR	p00	p25	p50	p75	
age	1,338.00	0.00	39.21	14.05	24.00	18.00	27.00	39.00	51.00	
bmi	1,338.00	0.00	30.66	6.10	8.40	15.96	26.30	30.40	34.69	
children	1,338.00	0.00	1.09	1.21	2.00	0.00	0.00	1.00	2.00	
charges	1,338.00	0.00	13,270.42	12,110.01	11,899.63	1,121.87	4,740.29	9,382.03	16,639.91	6

EDA - Sex

CIDRA-AIPD
African Institute for Professional Development

```
insurance %>%
  group_by(sex) %>%
  describe(quantiles = c(0, 0.25, 0.5, 0.75, 1),
           statistics = c("mean", "sd", "IQR", "quantiles")) %>%
  DT::datatable(rownames = FALSE, options = list(autoWidth = TRUE, dom = 't',scrollX = TRUE)) %>% formatRound
```

described_variables	sex	n	na	mean	sd	IQR	p00	p25	p50	p
age		662.00	0.00	39.50	14.05	24.75	18.00	27.00	40.00	5
age		676.00	0.00	38.92	14.05	25.00	18.00	26.00	39.00	5
bmi		662.00	0.00	30.38	6.05	8.19	16.82	26.13	30.11	3
bmi		676.00	0.00	30.94	6.14	8.58	15.96	26.41	30.69	3
charges		662.00	0.00	12,569.58	11,128.70	9,569.53	1,607.51	4,885.16	9,412.96	14,45
charges		676.00	0.00	13,956.75	12,971.03	14,370.46	1,121.87	4,619.13	9,369.62	18,98
children		662.00	0.00	1.07	1.19	2.00	0.00	0.00	1.00	2
children		676.00	0.00	1.12	1.22	2.00	0.00	0.00	1.00	2

EDA - Smoker

CIDRA-AIPD African Institute for Professional Development

```
insurance %>%
  group_by(smoker) %>%
  describe(quantiles = c(0, 0.25, 0.5, 0.75, 1),
           statistics = c("mean", "sd", "IQR", "quantiles")) %>%
  DT::datatable(rownames = FALSE, options = list(autoWidth = TRUE, dom = 't',scrollX = TRUE, scrollY = "300px
```

described_variables	smoker	n	na	mean	sd	IQR	p00	p25	p50
age		1,064.00	0.00	39.39	14.08	25.25	18.00	26.75	40.0
age		274.00	0.00	38.51	13.92	22.00	18.00	27.00	38.0
bmi		1,064.00	0.00	30.65	6.04	8.12	15.96	26.32	30.3
bmi		274.00	0.00	30.71	6.32	9.12	17.20	26.08	30.4
charges		1,064.00	0.00	8,434.27	5,993.78	7,376.45	1,121.87	3,986.44	7,345.
charges		274.00	0.00	32,050.23	11,541.55	20,192.96	12,829.46	20,826.24	34,456.3
children		1,064.00	0.00	1.09	1.22	2.00	0.00	0.00	1.0

EDA - Region

CIDRA-AIPD African Institute for Professional Development

```
insurance %>%
  group_by(region) %>%
  describe(quantiles = c(0, 0.25, 0.5, 0.75, 1),
           statistics = c("mean", "sd", "IQR", "quantiles")) %>%
  DT::datatable(rownames = FALSE, options = list(autoWidth = TRUE, scrollX = TRUE, scrollY = "300px")) %>% fc
```

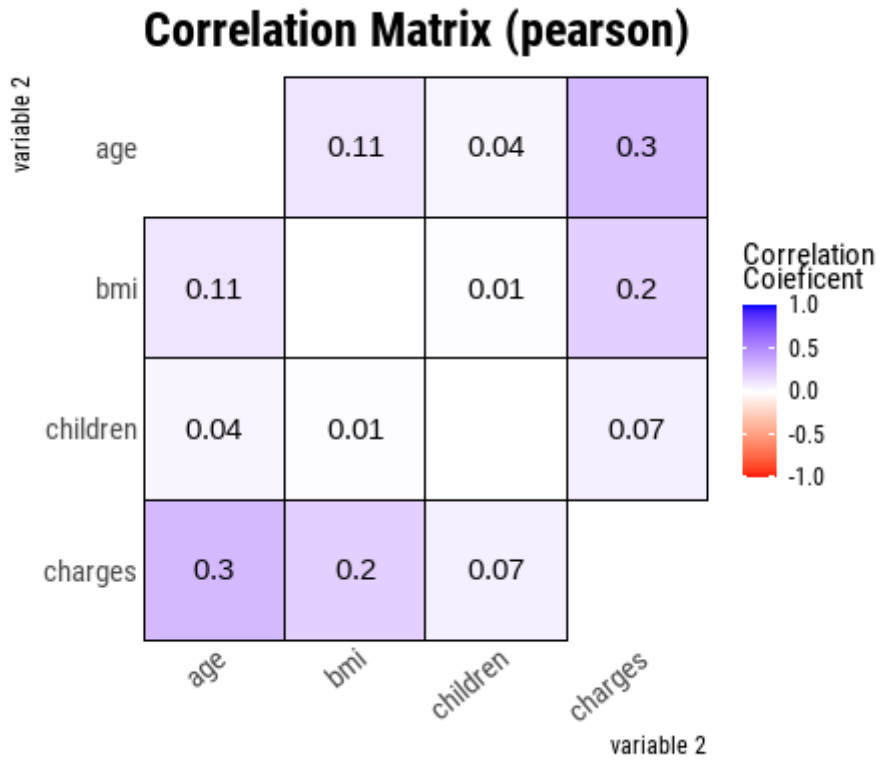
Show 10 entries Search:

described_variables	region	n	na	mean	sd	IQR	p00	p25	p50
age		324.00	0.00	39.27	14.07	24.00	18.00	27.00	39.50
age		325.00	0.00	39.20	14.05	25.00	19.00	26.00	39.00
age		364.00	0.00	38.94	14.16	24.25	18.00	26.75	39.00
age		325.00	0.00	39.46	13.96	24.00	19.00	27.00	39.00
bmi		324.00	0.00	29.17	5.94	8.03	15.96	24.87	28.88
bmi		325.00	0.00	29.20	5.14	7.03	17.39	25.75	28.88
bmi		364.00	0.00	33.36	6.48	9.24	19.80	28.57	33.33

Showing 1 to 10 of 16 entries

EDA - correlation

```
correlate(insurance) %>%
  plot()
```



Regression example - Feature engineering

We need to transform the characters values to factors.

```
insurance ← insurance %>%
  mutate_if(is.character, as.factor)

glimpse(insurance)
```

```
## Rows: 1,338
## Columns: 7
## $ age      <dbl> 19, 18, 28, 33, 32, 31, 46, 37, 37, 60, 25, 62, 23, 56, 27, 1…
## $ sex      <fct> female, male, male, male, male, female, female, female, male,…
## $ bmi      <dbl> 27.900, 33.770, 33.000, 22.705, 28.880, 25.740, 33.440, 27.74…
## $ children <dbl> 0, 1, 3, 0, 0, 0, 1, 3, 2, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0…
## $ smoker   <fct> yes, no, no, no, no, no, no, no, no, no, no, yes, no, no, yes…
## $ region   <fct> southwest, southeast, southeast, northwest, northwest, southe…
## $ charges  <dbl> 16884.924, 1725.552, 4449.462, 21984.471, 3866.855, 3756.622,…
```

Ordinary least squares is invariant to the scale of numerical variables, while methods such as lasso or ridge regression are not.

- For invariant methods there is no real need for standardisation
- For non-invariant methods you should standardise.

Regression - Data Splitting

CIDRA-AIPD African Institute for Professional Development

We will be using the `initial_split()` function to partition the data into training and test sets. The main arguments of the function are: `data` and `prop` that represents the training split proportion.

- The split will be 80/20 for training/test.

After creating the partition, we need to apply the `training()` and `testing()` functions to have the partitions.

> Remember to always use `set.seed()` to ensure your results are reproducible.

```
set.seed(88)

# create split object
insurance_split ← initial_split(insurance, prop = 0.8)

# Build training data set
insurance_train ← insurance_split %>% training()
cat("Insurance train dimension:", nrow(insurance_train), "rows and", ncol(insurance_train), "columns")
```

```
## Insurance train dimension: 1070 rows and 7 columns
```

```
# Build testing data set
insurance_test ← insurance_split %>% testing()
cat("Insurance test dimension:", nrow(insurance_test), "rows and", ncol(insurance_test), "columns")
```

Regression - Modelling

The next step in the process is to build a linear regression model to fit our training data.

For every model type, such as linear regression, there are numerous packages (or *engines*) that can be used.

For example, we can use the `lm()` function from base R or the `stan_glm()` function from the `rstanarm` package. Both of these functions will fit a linear regression model to our data with slightly different implementations.

The `parsnip` package from `tidymodels` acts like an aggregator across the various modeling engines within R. This makes it easy to implement machine learning algorithms from different R packages with one unifying syntax.

To specify a model object with parsnip, we must:

1. Pick a model type
2. Set the engine
3. Set the mode (either regression or classification)

Regression - Modelling

CIDRA-AIPD
African Institute
for Professional
Development

Linear regression is implemented with the `linear_reg()` function in `parsnip`. To the set the engine and mode, we use `set_engine()` and `set_mode()`.

Let's create a linear regression model object with the `lm` engine.

- This is the default engine for most applications.

```
lm_model ← linear_reg() %>%
           set_engine('lm') %>% # adds lm implementation of linear regression
           set_mode('regression')

# View object properties
lm_model
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Regression - Fit model to data

We can fit the model by using the `fit()` function from `parsnip`. The function has the following arguments:

- parnsip model
- model formula
- data frame with the training data

In our formula, we have specified that `charges` is the response variable and age, sex, bmi, children, smoker and region are our predictor variables.

```
lm_fit ← lm_model %>%
          fit(charges ~ ., data = insurance_train)

# View lm_fit properties
lm_fit
```

```
## parsnip model object
##
##
## Call:
## stats::lm(formula = charges ~ ., data = data)
##
## Coefficients:
##     (Intercept)              age          sexmale              bmi
##        -12582.4            257.7             -271.9            359.6
##        children        smokeryes  regionnorthwest  regionsoutheast
```

Regression - Exploring the model

CIDRA-AIPD African Institute for Professional Development

To obtain the results from our trained model in a data frame, we can use the `tidy()` and `glance()` functions from the `broom` package.

The `tidy()` function takes a linear regression object and returns a data frame of the estimated model coefficients and their associated F-statistics and p-values.

```
library(broom)
tidy(lm_fit) # Data frame of estimated coefficients
```

```
## # A tibble: 9 × 5
##   term            estimate std.error statistic   p.value
##   <chr>              <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)      -12582.    1063.    -11.8   1.92e- 30
## 2 age                 258.      12.6    20.5   8.98e- 79
## 3 sexmale            -272.     354.     -0.769 4.42e-  1
## 4 bmi                 360.      30.9    11.7   1.26e- 29
## 5 children            484.     145.      3.33  9.06e-  4
## 6 smokeryes         23550.     434.     54.2   4.35e-308
## 7 regionnorthwest    -528.     508.     -1.04  2.99e-  1
## 8 regionsoutheast   -1132.     504.     -2.24  2.50e-  2
## 9 regionsouthwest   -1230.     502.     -2.45  1.45e-  2
```

Regression - Exploring the model

CIDRA-AIPD African Institute for Professional Development

The `glance()` function returns performance metrics obtained on the training data such as the R2 value (r.squared) and the RMSE (sigma).

```
# Performance metrics on training data
glance(lm_fit)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic p.value    df  logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>   <dbl>  <dbl>  <dbl>
## 1     0.769         0.767 5755.      441.       0     8 -10778. 21575. 21625.
## # … with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Regression - Predict on the test set

CIDRA-AIPD African Institute for Professional Development

To assess the performance of the model, we must use the test set to predict the charge value and then compare to the real value.

This is done with the `predict()` function. This function takes two important arguments:

- Trained model
- new_data to generate predictions

It's best to combine the test data set and the predictions into a single data frame.

```
insurance_test_results ← predict(lm_fit, new_data = insurance_test) %>%
                           bind_cols(insurance_test)

# View results
insurance_test_results %>% head()
```

```
## # A tibble: 6 × 8
##    .pred   age sex      bmi children smoker region    charges
##    <dbl> <dbl> <fct>  <dbl>    <dbl> <fct>  <fct>       <dbl>
## 1  3285.    33 male    22.7        0 no     northwest  21984.
## 2  3529.    31 female  25.7        0 no     southeast   3757.
## 3 11641.    60 female  25.8        0 no     northwest  28923.
## 4 31690.    34 female  31.9        1 yes    northeast  37702.
## 5  1518.    18 female  26.3        0 no     northeast   2198.
## 6  8413.    31 female  36.6        2 no     southeast   4950.
```

Regression - Evaluate the model

CIDRA-AIPD African Institute for Professional Development

To evaluate the model, I'll use RMSE and R^2 that can be obtained by using the functions `rmse()` and `rsq()` functions. Both functions take the following arguments:

- `data`: dataframe with real and predicted values
- `truth`: column with the real labels
- `estimate`: columns with the predictions

```
# RMSE on test set
rmse(insurance_test_results, truth = charges, estimate = .pred) %>%
  # R2 on the test set
  bind_rows(rsq(insurance_test_results, truth = charges, estimate = .pred))
```

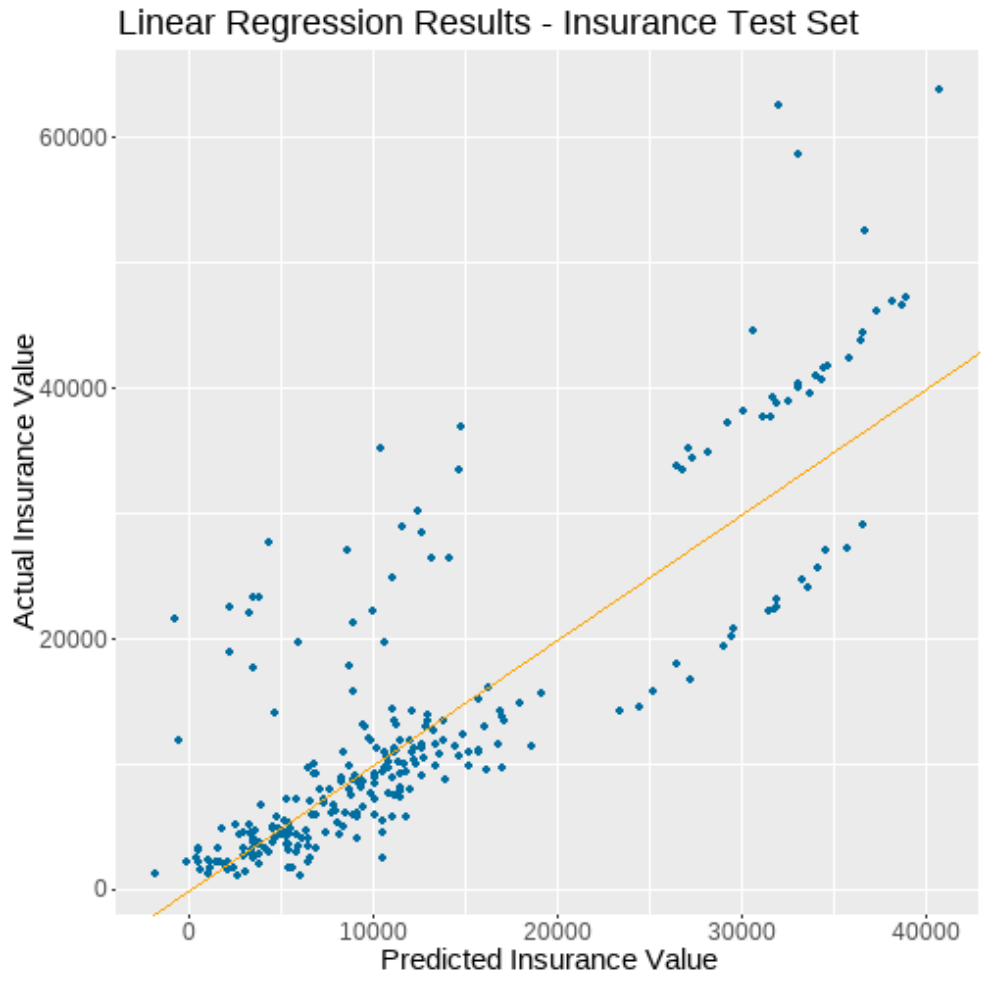
```
## # A tibble: 2 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard    7185.
## 2 rsq     standard       0.695
```

The closes to zero the RMSE value is the best your model is, and we can see that the RMSE of this model is very high, which indicates that this model is not a very good model to predict insurance charges.

Regression - Plot R2

We can visualise the fit of the model by plotting the R^2.

```
ggplot(data = insurance_test_results,
       mapping = aes(x = .pred, y = charges)) +
  geom_point(color = '#006EA1') +
  geom_abline(intercept = 0,
              slope = 1,
              color = 'orange') +
  labs(title = 'Linear Regression Results -
       Insurance Test Set',
       x = 'Predicted Insurance Value',
       y = 'Actual Insurance Value')
```



Classification

Classification

CIDRA-AIPD
African Institute for Professional Development

Classification is a type of **supervised** learning where we categorise data into classes. There are many different algorithms that can help us solve this kind of problems.

Classification requires a training dataset with many examples of inputs and outputs from which to learn. It can be categorised in two types of problems:

- **Binary classification:** The outcome has only **two** labels, for example, disease and not disease.
  - Some popular algorithms are: Logistic Regression, Decision Tree, K-Nearest Neighbour (KNN)
- **Multi-label classification:** The outcome has multiple labels, for example, dog, cat, bird and other.
  - Some popular algorithms: KNN, Decision Tree, Random Forest, Naive Bayes

To evaluate the model performance we can make use of ROC, confusion matrix, etc. We need to be aware of **class imbalance** problems.

Evaluation metrics

When thinking about classification, some common metrics to understand performance involves:

- Accuracy
- Precision
- Recall
- ROC

Most of this metrics can be determined by creating a **confusion matrix**, a contingency table with two dimensions (*actual* and *predicted*) that help us determine the performance of a classification model when we know the true classification of the data.

		Predicted	
		Negative (**N**) -	Positive (**P**) +
Actual	Negative -	True Negatives (**TN**)	False Positives (**FP**) **Type I error**
	Positive +	False Negatives (**FN**) **Type II error**	True Positives (**TP**)

Evaluation metrics - Confusion Matrix

CIDRA-AIPD African Institute for Professional Development

From the confusion matrix, we can get the following performance metrics:

- **Accuracy:** Proportion of correct predictions among the total number of cases examined.

$$\mathrm{Acc} = \frac{TP + TN}{N + P}$$

- **Precision or Positive Predictive Value (PPV):** Proportion of positive predictions that was actually correct

$$\mathrm{Pr} = \frac{TP}{TP + FP}$$

- **Sensitivity or recall or True Positive Rate (TPR):** Proportion of correct predictions among the positive instances.

$$\mathrm{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score:** Harmonic mean of precision and recall.

$$F_1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

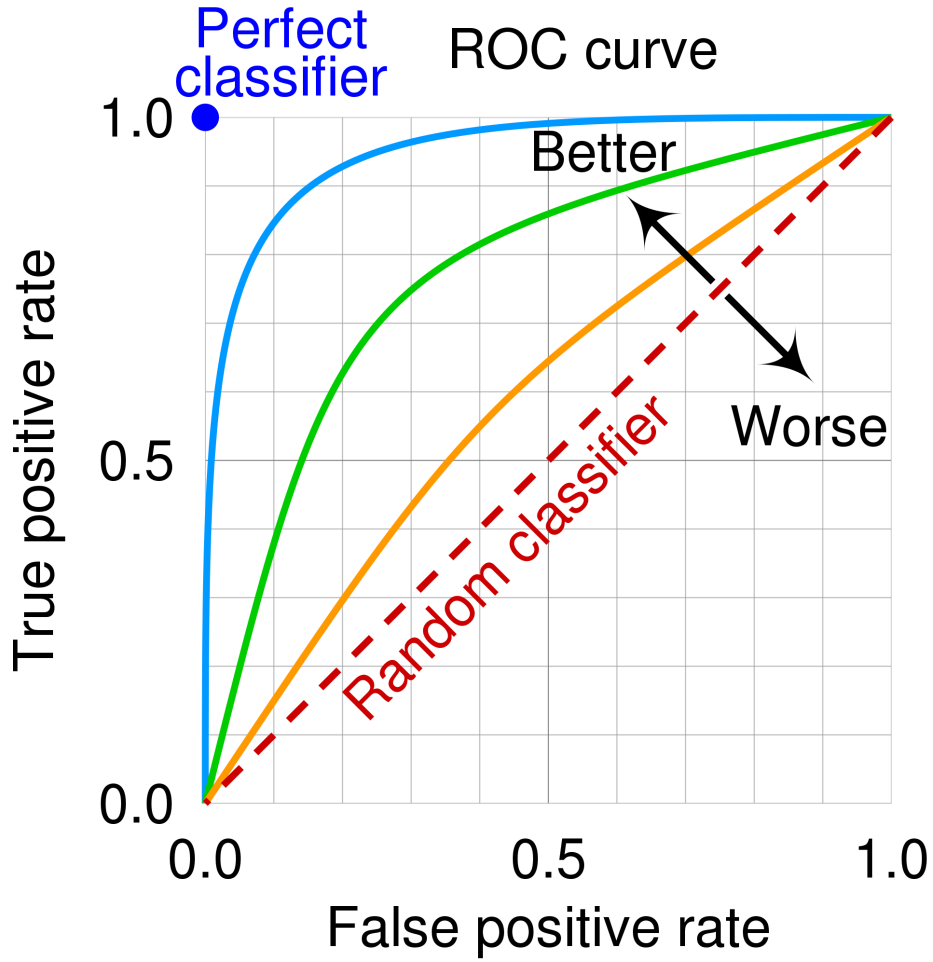
Evaluation metric - ROC curve

ROC curve is a performance measurement for the classification problems at various threshold settings. This curve plots two parameters:

- True Positive Rate
- False Positive Rate = 1 - Specificity

From the ROC curve we can get the **Area Under the Curve (AUC)** metric that ranges from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

- When AUC is 0.5, it means the model has no class separation capacity whatsoever.



Class Imbalance problem

CIDRA-AIPD
African Institute for Professional Development

An **imbalanced classification** problem is a problem where the distribution of examples across the known classes is biased or skewed.

- The distribution can vary from a slight bias to a severe imbalance.

Most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class.

- This results in models that have poor predictive performance, specifically for the minority class
- Typically, the minority class is more important

For example, we want to detect fraudulent transaction, and if every 1000 transactions only 1 is fraudulent, it is very easy for the model to classify everything as non-fraud, so the accuracy of the model will be almost 1, but the actual class that we wanted to determine, it was wrongfully classified.

- We need to look at other metrics that can tell us how the prediction performed for different classes

> Using **accuracy** as a peformance metric is **not recommended**.

We need to use techniques that will help us undersampling the majority class or oversampling the minority class, for example, Random undersampling and SMOTE oversampling.

Classification - Example

CIDRA-AIPD
African Institute
for Professional
Development

The dataset we are going to use to perform the classification problem is `BreastCancer: Wisconsin Breast Cancer Database`. The data can be downloaded from the `mlbench` package.

The data has 699 observations on 11 variables, one being a character variable, 9 being ordered or nominal, and 1 target class.

```
library(mlbench)
data(BreastCancer)
BreastCancer ← BreastCancer %>% select(-Id)
glimpse(BreastCancer)
```

```
## Rows: 699
## Columns: 10
## $ Cl.thickness    <ord> 5, 5, 3, 6, 4, 8, 1, 2, 2, 4, 1, 2, 5, 1, 8, 7, 4, 4, …
## $ Cell.size       <ord> 1, 4, 1, 8, 1, 10, 1, 1, 1, 2, 1, 1, 3, 1, 7, 4, 1, 1,…
## $ Cell.shape      <ord> 1, 4, 1, 8, 1, 10, 1, 2, 1, 1, 1, 1, 3, 1, 5, 6, 1, 1,…
## $ Marg.adhesion   <ord> 1, 5, 1, 1, 3, 8, 1, 1, 1, 1, 1, 1, 3, 1, 10, 4, 1, 1,…
## $ Epith.c.size    <ord> 2, 7, 2, 3, 2, 7, 2, 2, 2, 2, 1, 2, 2, 2, 7, 6, 2, 2, …
## $ Bare.nuclei     <fct> 1, 10, 2, 4, 1, 10, 10, 1, 1, 1, 1, 1, 3, 3, 9, 1, 1, …
## $ Bl.cromatin     <fct> 3, 3, 3, 3, 3, 9, 3, 3, 1, 2, 3, 2, 4, 3, 5, 4, 2, 3, …
## $ Normal.nucleoli <fct> 1, 2, 1, 7, 1, 7, 1, 1, 1, 1, 1, 1, 4, 1, 5, 3, 1, 1, …
## $ Mitoses         <fct> 1, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 4, 1, 1, 1, …
## $ Class           <fct> benign, benign, benign, benign, benign, malignant, ben…
```

Classification example - Summary

CIDRA-AIPD African Institute for Professional Development

```
summary(BreastCancer)
```

```
##   Cl.thickness   Cell.size     Cell.shape  Marg.adhesion  Epith.c.size
##  1       :145   1       :384   1       :353   1       :407   2       :386
##  5       :130   10      : 67   2       : 59   2       : 58   3       : 72
##  3       :108   3       : 52   10      : 58   3       : 58   4       : 48
##  4       : 80   2       : 45   3       : 56   10      : 55   1       : 47
##  10      : 69   4       : 40   4       : 44   4       : 33   6       : 41
##  2       : 50   5       : 30   5       : 34   8       : 25   5       : 39
##  (Other):117   (Other): 81   (Other): 95   (Other): 63   (Other): 66
##   Bare.nuclei   Bl.cromatin  Normal.nucleoli    Mitoses           Class
##  1       :402   2       :166   1       :443     1       :579   benign   :458
##  10      :132   3       :165   10      : 61     2       : 35   malignant:241
##  2       : 30   1       :152   3       : 44     3       : 33
##  5       : 30   7       : 73   2       : 36     10      : 14
##  3       : 28   4       : 40   8       : 24     4       : 12
##  (Other): 61   5       : 34   6       : 22     7       :  9
##  NA's    : 16   (Other): 69   (Other): 69     (Other): 17
```

```
BreastCancer %>% group_by(Class) %>% count() %>% ungroup() %>% mutate(prop = n / sum(n))
```

```
## # A tibble: 2 × 3
##   Class         n  prop
##   <fct>     <int> <dbl>
## 1 benign      458 0.655
## 2 malignant   241 0.345
```

Classification - Data Preprocessing

CIDRA-AIPD African Institute for Professional Development

As we have missing data, we will drop the rows that are incomplete.

```
BreastCancer %>% anyNA()
```

```
## [1] TRUE
```

```
cat("Number of rows before dropping missing values: ", nrow(BreastCancer))
```

```
## Number of rows before dropping missing values:  699
```

```
BreastCancer ← BreastCancer %>% drop_na()
cat("Number of rows after dropping missing values: ", nrow(BreastCancer))
```

```
## Number of rows after dropping missing values:  683
```

As all the variables of the dataset, except Class, can be numerical features, I'll change the variable type.

```
BreastCancer ← BreastCancer %>%
  mutate_at(vars(-("Class")), as.numeric)
```

Classification - Data Splitting

CIDRA-AIPD African Institute for Professional Development

The split will be 80/20 for training/test.

```
set.seed(88)

# create split object
bcancer_split ← initial_split(BreastCancer, prop = 0.8)

# Build training data set
bcancer_train ← bcancer_split %>% training()
cat("Breast cancer train dimension:", nrow(bcancer_train), "rows and", ncol(bcancer_train), "columns")
```

```
## Breast cancer train dimension: 546 rows and 10 columns
```

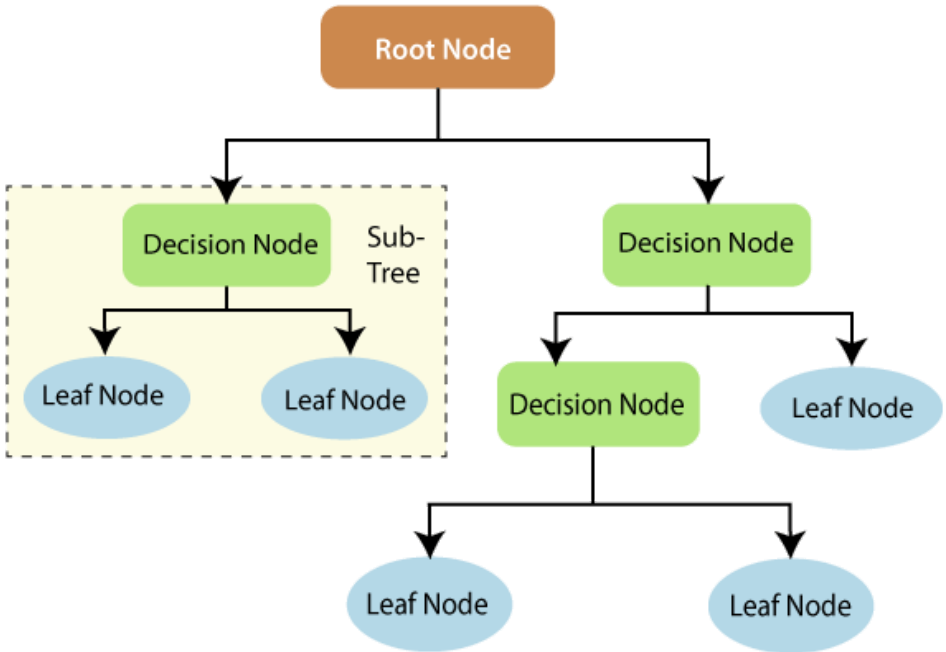
```
# Build testing data set
bcancer_test ← bcancer_split %>% testing()
cat("Breast cancer test dimension:", nrow(bcancer_test), "rows and", ncol(bcancer_test), "columns")
```

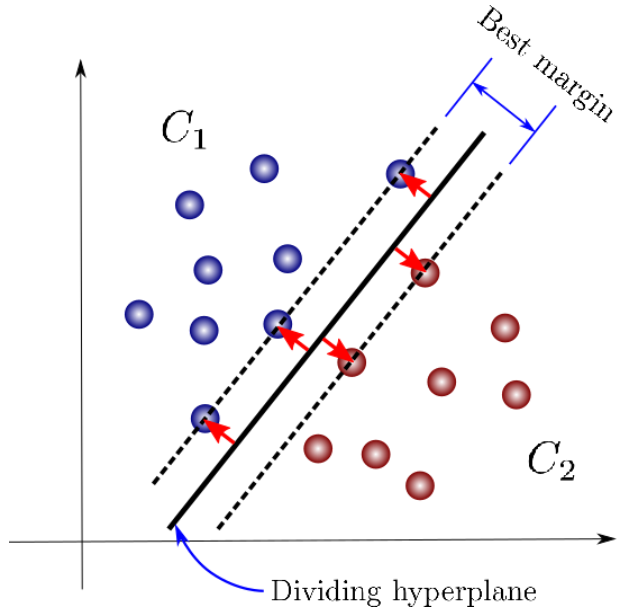
```
## Breast cancer test dimension: 137 rows and 10 columns
```

Classification - Modelling

I will be using and comparing the results of two models:

- Decision Tree
- Suport Vector Machine (SVM)





Classification - Modelling

CIDRA-AIPD
African Institute
for Professional
Development

Decision Tree and SVM are implemented with the `decision_tree()` and `svm_linear()` functions in `parsnip`. To the set the engine and mode, we use `set_engine()` and `set_mode()`.

Let's create this models using the following engines:

- `rpart` for the decision tree that needs the `rpart` package installed
- `LiblineaR` for the SVM that needs to have the package `LiblineaR` installed

```
tree_model ← decision_tree() %>%
              set_engine('rpart') %>%
              set_mode('classification')

svm_model ← svm_linear() %>%
             set_engine('LiblineaR') %>%
             set_mode('classification')
```

Classification - Fit model to data

CIDRA-AIPD African Institute for Professional Development

`Class` is the response variable and all the other variables are our predictor variables.

```
tree_fit ← tree_model %>% fit(Class ~ ., data = bcancer_train)
svm_fit ← svm_model %>% fit(Class ~ ., data = bcancer_train)

# View fit properties
svm_fit %>% broom::tidy()
```

```
## # A tibble: 10 × 2
##    term            estimate
##    <chr>              <dbl>
##  1 Cl.thickness     -0.124
##  2 Cell.size        -0.0537
##  3 Cell.shape       -0.0611
##  4 Marg.adhesion    -0.0494
##  5 Epith.c.size      0.0217
##  6 Bare.nuclei      -0.136
##  7 Bl.cromatin      -0.104
##  8 Normal.nucleoli  -0.0392
##  9 Mitoses          -0.136
## 10 Bias              2.37
```

Classification - Predict on the test set

CIDRA-AIPD African Institute for Professional Development

To assess the performance of the model, we must use the test set to predict the charge value and then compare to the real value.

```
bcancer_test_results_tree ← predict(tree_fit, new_data = bcancer_test) %>%
                            bind_cols(bcancer_test)
bcancer_test_results_tree %>% head(2)
```

```
## # A tibble: 2 × 11
##   .pred_class Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
##   <fct>              <dbl>     <dbl>      <dbl>         <dbl>        <dbl>
## 1 benign                 3         1          1             1            2
## 2 malignant              8        10         10             8            7
## # … with 5 more variables: Bare.nuclei <dbl>, Bl.cromatin <dbl>,
## #   Normal.nucleoli <dbl>, Mitoses <dbl>, Class <fct>
```

```
bcancer_test_results_svm ← predict(svm_fit, new_data = bcancer_test) %>%
                           bind_cols(bcancer_test)
bcancer_test_results_svm %>% head(2)
```

```
## # A tibble: 2 × 11
##   .pred_class Cl.thickness Cell.size Cell.shape Marg.adhesion Epith.c.size
##   <fct>              <dbl>     <dbl>      <dbl>         <dbl>        <dbl>
## 1 benign                 3         1          1             1            2
## 2 malignant              8        10         10             8            7
## # … with 5 more variables: Bare.nuclei <dbl>, Bl.cromatin <dbl>,
```

Classification - Evaluate the models

CIDRA-AIPD African Institute for Professional Development

To evaluate the models, we will use the `confusionMatrix` function from `caret`.

- Decision Tree

```
library(caret)
confusionMatrix(bcancer_test_results_tree$.pred_cla
                bcancer_test_results_tree$Class) %>
  tidy() %>%
  filter(term %in% c("precision", "recall", "specif
  select(term, class, estimate)
```

```
## # A tibble: 5 × 3
##   term        class  estimate
##   <chr>       <chr>     <dbl>
## 1 accuracy    <NA>      0.949
## 2 specificity benign    0.952
## 3 precision   benign    0.978
## 4 recall      benign    0.947
## 5 f1          benign    0.963
```

- SVM

```
confusionMatrix(bcancer_test_results_svm$.pred_clas
                        bcancer_test_results_svm$Cla
  tidy() %>%
  filter(term %in% c("precision", "recall", "specif
  select(term, class, estimate)
```

```
## # A tibble: 5 × 3
##   term        class  estimate
##   <chr>       <chr>     <dbl>
## 1 accuracy    <NA>      0.956
## 2 specificity benign    0.905
## 3 precision   benign    0.959
## 4 recall      benign    0.979
## 5 f1          benign    0.969
```

CIDRA-AIPD
African Institute
for Professional
Development

Unsupervised Learning

Unsupervised Learning

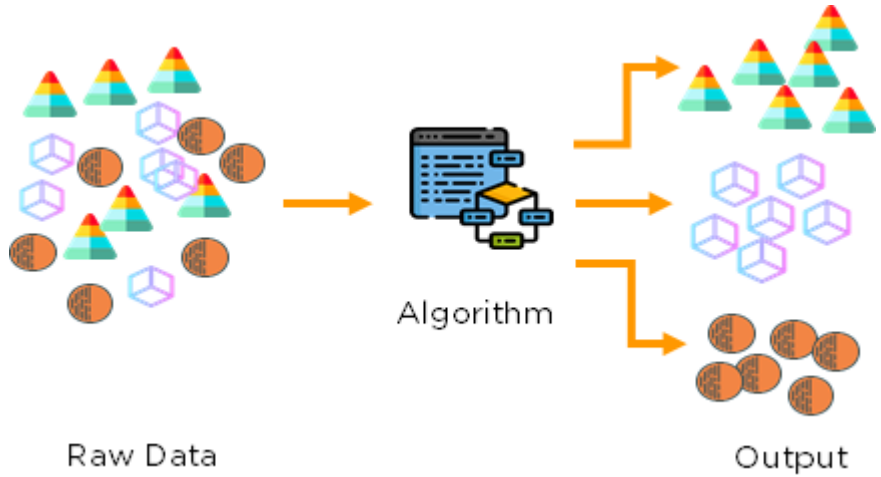
This type of problem is called **Unsupervised** because, unlike supervised learning, we don't have a label.

- The algorithm learns patterns from unlabeled data.

Common tasks:

- **Clustering:** Technique that groups unlabeled data based on their similarities or differences.
  - Example: KMeans
- **Dimensionality Reduction**: Reduce the number of features while also preserving the integrity of the dataset as much as possible.
  - Example: Principla Components Analysis (PCA)

Some challenges can occur when it allows machine learning models to execute without any human intervention. For example:



CIDRA-AIPD
African Institute
for Professional
Development

Clustering

Clustering

Clustering can be considered the most important unsupervised learning problem.

It involves discovering groups in data. Unlike supervised learning, clustering algorithms only interpret the input data and find natural groups or clusters in feature space.

A cluster is a collection of objects which are *similar* between them and are *dissimilar* to the objects belonging to other clusters.

Given a set of points, with some distance between points, grouping the points into some number of clusters, such that

- **internal (within the cluster)** distances should be small i.e members of clusters are close/similar to each other.
- **external (intra-cluster)** distances should be large i.e. members of different clusters are dissimilar.

Distance metrics

CIDRA-AIPD African Institute for Professional Development

We need to define a **distance** or **proximity** metric for 2 points.

- How similar or dissimilar 2 points are
- **Similarity** $s(x_i, x_k)$: Large if x_i, x_k are similar
- **Dissimilarity (or distance)** $d(x_i, x_k)$: Small if x_i, x_k are similar

Examples:

- **Euclidean Distance:** $d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$
- **Mahalanobis Distance:** $d_M(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{p} - \mathbf{y})^T S^{-1} (\mathbf{p} - \mathbf{y})}$
- **Jaccard Distance:** $J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$

A *good* proximity measure is **very** application dependent.

Clustering - Example

CIDRA-AIPD African Institute for Professional Development

The steps to perform an unsupervised learning task are very similar to an supervised one, with the difference that we don't need to predict anything.

For the example, we will use the the built-in R data set `USArrests`, which contains statistics in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. It includes also the percent of the population living in urban areas.

```
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization

us_arrests ← USArrests
glimpse(us_arrests)
```

```
## Rows: 50
## Columns: 4
## $ Murder   <dbl> 13.2, 10.0, 8.1, 8.8, 9.0, 7.9, 3.3, 5.9, 15.4, 17.4, 5.3, 2.…
## $ Assault  <int> 236, 263, 294, 190, 276, 204, 110, 238, 335, 211, 46, 120, 24…
## $ UrbanPop <int> 58, 48, 80, 50, 91, 78, 77, 72, 80, 60, 83, 54, 83, 65, 57, 6…
## $ Rape     <dbl> 21.2, 44.5, 31.0, 19.5, 40.6, 38.7, 11.1, 15.8, 31.9, 25.8, 2…
```

Clustering - Data Preprocessing

We are going to check and remove NA's if the exist in the data

```
us_arrests %>% anyNA()
```

```
## [1] FALSE
```

As we don't want the clustering algorithm to depend to an arbitrary variable unit, we start by scaling/standardizing the data using the R function `scale`:

```
us_arrests_scaled ← us_arrests %>%
  scale()

head(us_arrests_scaled)
```

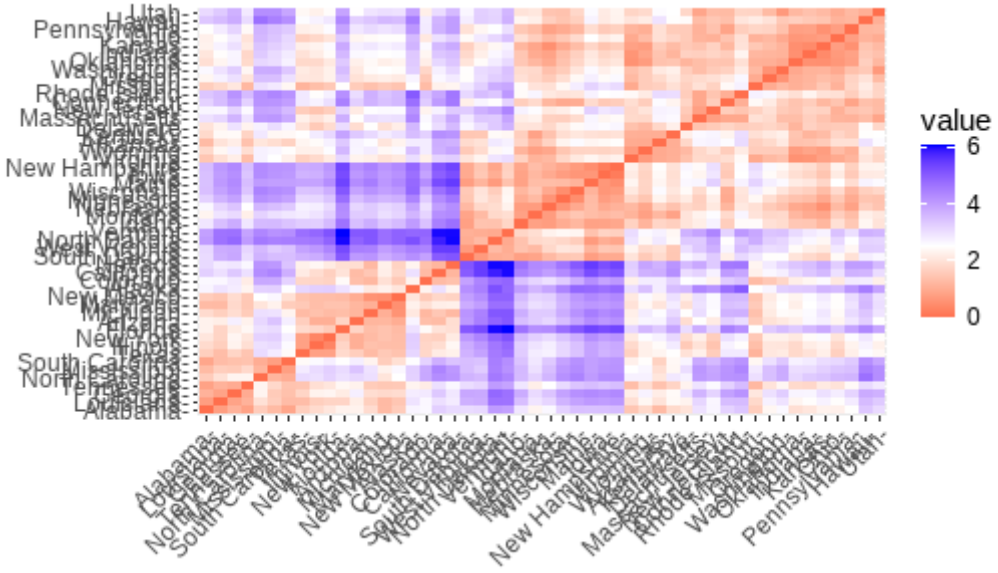
```
##               Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

Clustering - Distance

We can compute and visualise the distance matrix using the functions `get_dist` and `fviz_dist` from the `factoextra` package.

- `get_dist`: Computing a distance matrix between the rows of a data matrix.
  - The default distance computed is the **Euclidean**
- `fviz_dist`: Visualise the distance matrix

```
distance ← get_dist(us_arrests_scaled)
fviz_dist(distance)
```



Clustering - K-Means

K-means clustering is the most commonly used unsupervised machine learning algorithm for partitioning a given data set into a set of k groups determine by the analyst. In k-means clustering, each cluster is represented by its center (i.e, centroid) which corresponds to the mean of points assigned to the cluster.

Determining the optimal number of clusters

As the number of clusters is user specified, we would like to use the optimal number of clusters. The three most popular methods for determining the optimal clusters are:

- **Elbow method:** Minimise the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square)
  - Can be calculated using the function `fviz_nbclust` using method `wss`
- **Silhouette method:** Measures the quality of a clustering, i.e., it determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering.
  - Can be calculated using the function `fviz_nbclust` using method `silhouette`
- **Gap statistic:** The approach can be applied to any clustering method. The gap statistic compares the total intracluster variation for different values of k with their expected values under null reference distribution of the data (i.e. a distribution with no obvious clustering).
  - Can be visulaised using the function `fviz_gap_stat` using the statistic generated by `clusGap`

Clustering - Optimal number of clusters

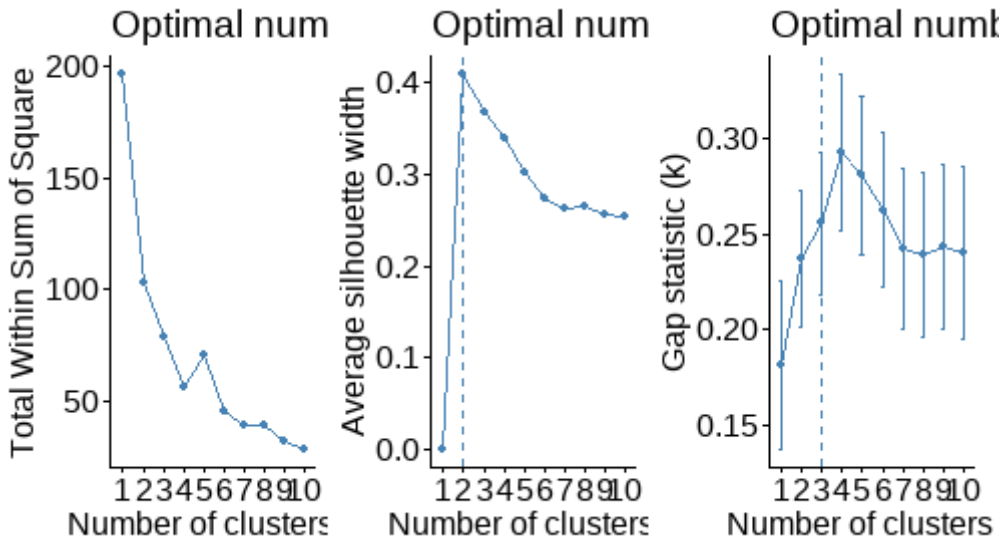
CIDRA-AIPD
African Institute for Professional Development

```
library(patchwork)

set.seed(123)

elbow_method ← fviz_nbclust(us_arrests_scaled, kmeans, method = "wss")
sill ← fviz_nbclust(us_arrests_scaled, kmeans, method = "silhouette")
gap_stat ← clusGap(us_arrests_scaled, FUN = kmeans, nstart = 25, K.max = 10, B = 50)
gap_plot ← fviz_gap_stat(gap_stat)

elbow_method + sill + gap_plot
```



Clustering - KMeans

With most of these approaches suggesting 3 as the number of optimal clusters, we can perform the final analysis and extract the results using 3 clusters.

```
# Compute k-means clustering with k = 3
set.seed(123)
kmeans ← kmeans(us_arrests_scaled, 3, nstart = 25)
print(kmeans)
```

```
## K-means clustering with 3 clusters of sizes 20, 13, 17
##
## Cluster means:
##       Murder    Assault   UrbanPop       Rape
## 1  1.0049340  1.0138274  0.1975853  0.8469650
## 2 -0.9615407 -1.1066010 -0.9301069 -0.9667633
## 3 -0.4469795 -0.3465138  0.4788049 -0.2571398
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              3              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              1              3              3              1              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              2              1              3              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              2              1              2              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
```

Clustering - Visualise clusters

CIDRA-AIPD African Institute for Professional Development

We can visualise the clusters using the function `fviz_cluster`.

```
fviz_cluster(kmeans, data = us_arrests_scaled)
```

Clustering - Final step

CIDRA-AIPD African Institute for Professional Development

And we can extract the clusters and add to our initial data to do some descriptive statistics at the cluster level:

```
us_arrests %>%
  mutate(cluster = kmeans$cluster) %>%
  group_by(cluster) %>%
  summarise_all("mean")
```

```
## # A tibble: 3 × 5
##   cluster Murder Assault UrbanPop  Rape
##     <int>  <dbl>   <dbl>    <dbl> <dbl>
## 1       1  12.2    255.      68.4  29.2
## 2       2   3.6     78.5     52.1  12.2
## 3       3   5.84   142.      72.5  18.8
```

CIDRA-AIPD
African Institute
for Professional
Development

Principal Components Analysis

Principal Components Analysis (PCA)

CIDRA-AIPD African Institute for Professional Development

Principal Components Analysis (PCA) is an unsupervised machine learning technique that seeks to find *principal components*, i.e., linear combinations of the original predictors, that explain a large portion of the variation in a dataset.

- The goal of PCA is to explain most of the variability in a dataset with fewer variables than the original dataset.

For the example we will use the dataset `fat` available in the `faraway` library. The dataset contains 18 physical measurements.

```
library(faraway)
attach(fat)
glimpse(fat)
```

```
## Rows: 252
## Columns: 18
## $ brozek  <dbl> 12.6, 6.9, 24.6, 10.9, 27.8, 20.6, 19.0, 12.8, 5.1, 12.0, 7.5,…
## $ siri    <dbl> 12.3, 6.1, 25.3, 10.4, 28.7, 20.9, 19.2, 12.4, 4.1, 11.7, 7.1,…
## $ density <dbl> 1.0708, 1.0853, 1.0414, 1.0751, 1.0340, 1.0502, 1.0549, 1.0704…
## $ age     <int> 23, 22, 22, 26, 24, 24, 26, 25, 25, 23, 26, 27, 32, 30, 35, 35…
## $ weight  <dbl> 154.25, 173.25, 154.00, 184.75, 184.25, 210.25, 181.00, 176.00…
## $ height  <dbl> 67.75, 72.25, 66.25, 72.25, 71.25, 74.75, 69.75, 72.50, 74.00,…
## $ adipos  <dbl> 23.7, 23.4, 24.7, 24.9, 25.6, 26.5, 26.2, 23.6, 24.6, 25.8, 23…
## $ free    <dbl> 134.9, 161.3, 116.0, 164.7, 133.1, 167.0, 146.6, 153.6, 181.3,…
## $ neck    <dbl> 36.2, 38.5, 34.0, 37.4, 34.4, 39.0, 36.4, 37.8, 38.1, 42.1, 38…
## $ chest   <dbl> 93.1, 93.6, 95.8, 101.8, 97.3, 104.5, 105.1, 99.6, 100.9, 99.6…
```

PCA - Preprocessing

CIDRA-AIPD African Institute for Professional Development

PCA is **influenced** by the magnitude of each variable; therefore, the results obtained when we perform PCA will also depend on whether the variables have been individually scaled.

```
apply(fat, 2, var)
```

```
##       brozek         siri      density          age       weight       height
## 6.007576e+01 7.003582e+01 3.621955e-04 1.588114e+02 8.637227e+02 1.341651e+01
##       adipos         free         neck        chest        abdom          hip
## 1.330871e+01 3.323928e+02 5.909339e+00 7.107292e+01 1.162747e+02 5.132372e+01
##        thigh         knee        ankle       biceps      forearm        wrist
## 2.756200e+01 5.816801e+00 2.872664e+00 9.128095e+00 4.083193e+00 8.715808e-01
```

PCA works best with numerical data, so in case you have categorical features on the dataset, you need to "dummyfy" the features.

PCA - Example

We can apply PCA with the function `prcomp`. You will also set two arguments, center and scale, to be TRUE.

```
fat_pca ← prcomp(fat, center = TRUE, scale. = TRUE)
summary(fat_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.2807 1.6586 1.04664 0.88222 0.81474 0.75617 0.56521
## Proportion of Variance 0.5979 0.1528 0.06086 0.04324 0.03688 0.03177 0.01775
## Cumulative Proportion  0.5979 0.7508 0.81163 0.85487 0.89175 0.92351 0.94126
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.51350 0.48876 0.43012 0.36381 0.28051 0.24204 0.20448
## Proportion of Variance 0.01465 0.01327 0.01028 0.00735 0.00437 0.00325 0.00232
## Cumulative Proportion  0.95591 0.96918 0.97946 0.98681 0.99118 0.99444 0.99676
##                          PC15    PC16    PC17    PC18
## Standard deviation     0.1899 0.12595 0.07834 0.01527
## Proportion of Variance 0.0020 0.00088 0.00034 0.00001
## Cumulative Proportion  0.9988 0.99965 0.99999 1.00000
```

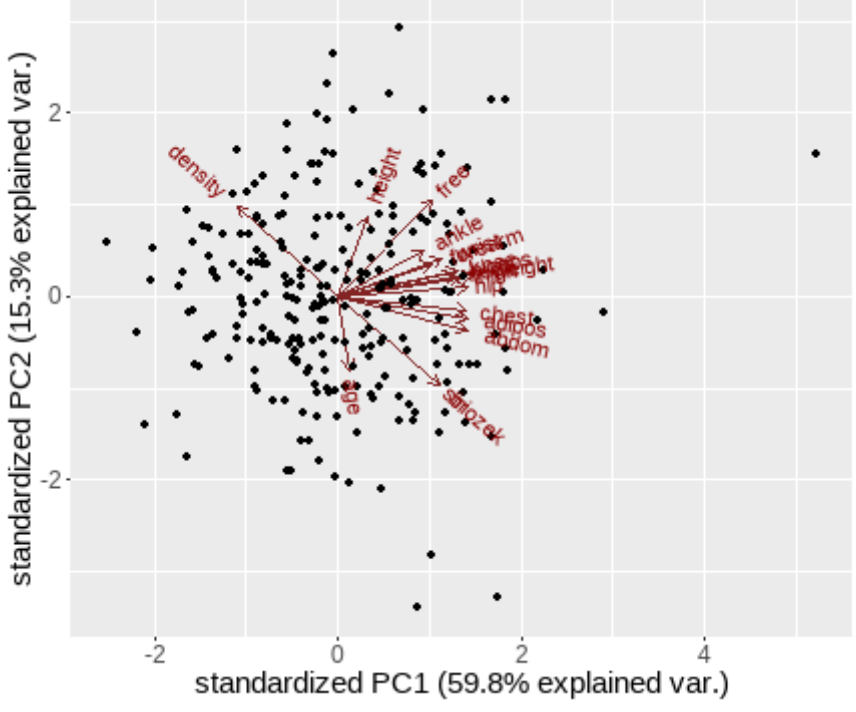
You obtain 18 principal components. Each of these explains a percentage of the total variation in the dataset.

- PC1 explains 59.8% of the total variance, which means that almost half of the information in the dataset can be encapsulated by just that one Principal Component.
- PC2 explains 15% of the variance. With just PC1 and PC2 we can explain ~74% of the variance.

PCA - Plotting

We can plot PCA using the `ggbiplot` library. We need to install this library from github using the function `install_github` from `devtools` giving the github link as parameter of the function.

```
#devtools::install_github("vqv/ggbiplot")
library(ggbiplot)
ggbiplot(fat_pca)
```



Reference

https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv

The Elements of Statistical Learning

Machine Learning: A Probabilistic Perspective