DATA3888: Data Science Capstone

Week 1 - Case study 1 + Classification

Andy Tran
22 February, 2023

THE UNIVERSITY OF SYDNEY

Why DATA3888?

Course Outcomes

- **Goals**
  - Exposure to different data types (5 case studies)
  - Learn practical data science skills
  - Develop collaboration and teamwork skills
- Lectures will introduce case studies and core data science skills. **Weeks 1-3**: Andy Tran, **Weeks 4-6**: Shila Ghazanfar
- Labs will have a detailed application of data science skills and case study exploration

Outline

- Case Study 1 - Reef data
- Representing spatial data
- Classification

Case study 1 - Reef data

- Coral bleaching is when coral reefs, under environmental stress (such as change in temperature), release algae in their tissues, causing them to turn white.
- Bleached corals become susceptible to disease and vulnerable to death.
- Study aim:
  - (i) Predicting which environmental factors is associated with bleaching event
  - (ii) Predicting which Reef is likely to be susecptible to bleaching
- A public dataset taken from the paper "A global analysis of coral bleaching over the past two decades." published in Nature communications in 2019.
- Authors are: Sully, S., Burkepile, D. E., Donovan, M. K., Hodgson, G., & Van Woesik, R.
- The author has curated coral bleaching events at 3351 sites in 81 countries from 1998 to 2017 and a suite of environmental variables at each site.

Data processing

- Data on global coral bleaching event can be found at the Reef Check website (reefcheck.org).
  - Also provided on GitHub repository for the Institute for Global Ecology
  - https://github.com/InstituteForGlobalEcology/Coral-bleaching-a-global-analysis-of-the-past-two-decades
- Data on environmental variables - Coral Reef Temperature Anomaly Database (CoRTAD Version 6) data
  - Available at NOAA's National Centers for Environmental Information (NCEI) webpage
  - https://data.nodc.noaa.gov/cortad/Version6/
- All sea surface temperature (SST) data used to determine the rate of SST change
  - Available in a downloadable file titled sst.mnmean.nc at NOAA's Earth Systems Research Laboratory (ESRL),
  - Physical Sciences Division (PSD) webpage
  - https://www.esrl.noaa.gov/psd/data/gridded/data.noaa.oisst.v2.html

Data processing - Extension

- All information on CoRTAD are stored in a special data structure called netCDF. Processing the netCDF and matching the coral bleaching records with environmental records requires advanced data wrangling skills.
- Details are in Github and also on Canvas.
- Curated data or integrated data contains
  - Coral bleaching at 3351 sites in 81 countries from 1998 to 2017
  - A suite of environmental covariates and temperature metrics

Understanding your data

- For a description of each column, please refer to Supplementary Table 1 of the aforementioned study.

```
reef <- read.csv("data/Reef_Check_with_cortad_variables_with_annual_rate_of_SST_change.csv")
dim(reef)
```

```
## [1] 9215   55
```

```
head(reef)
```

```
##                   Reef.ID        Reef.Name  Ocean  Country State.Province.Island
## 1 103.10.28.1E.10.50.46.1N Koh Mano (Minor) Indian Cambodia              Koh Kong
## 2   103.11.35.5E.10.49.32N Koh Mano (south) Indian Cambodia              Koh Kong
## 3  103.11.79.5E.10.48.2.7N      Koh Ta Team Indian Cambodia              Koh Kong
## 4   103.4.16.8E.11.3.36.2N   Koh Krosa Krao Indian Cambodia              Koh Kong
## 5   103.4.63.9E.11.3.58.3N Koh Krosa Kandal Indian Cambodia              Koh Kong
## 6  103.5.55.5E.10.53.48.5N        Koh Smach Indian Cambodia              Koh Kong
##   City.Town Year      Date Depth               Organism.Code S1 S2 S3 S4 Errors.
## 1           2003 19-Feb-03   4.5 Bleaching (% of population)  0  0  0  0   FALSE
## 2           2003 28-Feb-03   4.5 Bleaching (% of population)  0  0  0  0   FALSE
## 3           2003 24-Feb-03   5.0 Bleaching (% of population)  0  0  0  0   FALSE
## 4           2003 25-Feb-03   6.0 Bleaching (% of population)  0  0  0  0   FALSE
```

Data Dictionary

Supplementary Table 1. Ecological parameter and temperature metric descriptions and sources. Ecological parameters and temperature metrics used to predict coral bleaching prevalence and intensity across reefs worldwide. CoRTAD variables are from NOAA's Coral Reef Temperature Anomaly Database (www.nodc.noaa.gov/sog/cortad/). All CoRTAD variables are provided on a grid cell basis, of approximately 4 km resolution. The study time frame for CoRTAD data is from 1982-2017, with time units of 1 week. Rate of SST change is calculated from NOAA's Optimum Interpolation (OI) SST, using years 1984 and 2017. Depth, latitude, and year were recorded by divers when each Reef Check study was conducted. Diversity data was provided by J.E.N. Veron.

Parameter	Metric Description	Metric Source
Climatological SST (ClimSST)	Climatological sea surface temperature (SST) based on weekly SSTs for the study time frame, created using a harmonics approach	www.nodc.noaa.gov/sog/cortad/Version6/
Depth	Provided in meters	Reef Check
DHW (Degree Heating Weeks)	Defined as 1 °C above the long-term average for the warmest month in a climatology	www.nodc.noaa.gov/sog/cortad/Version6/
Diversity	The number of coral species confirmed present in an ecoregion	J.E.N Veron (personal communication) and at

What questions can we ask?

```
colnames(reef)
```

```
##  [1] "Reef.ID"                                  "Reef.Name"
##  [3] "Ocean"                                    "Country"
##  [5] "State.Province.Island"                    "City.Town"
##  [7] "Year"                                     "Date"
##  [9] "Depth"                                    "Organism.Code"
## [11] "S1"                                       "S2"
## [13] "S3"                                       "S4"
## [15] "Errors."                                  "What.errors."
## [17] "Average_bleaching"                        "ClimSST"
## [19] "Temperature_Kelvin"                       "Temperature_Mean"
## [21] "Temperature_Minimum"                      "Temperature_Maximum"
## [23] "Temperature_Kelvin_Standard_Deviation"    "Windspeed"
## [25] "SSTA"                                     "SSTA_Standard_Deviation"
## [27] "SSTA_Mean"                                "SSTA_Minimum"
## [29] "SSTA_Maximum"                             "SSTA_Frequency"
## [31] "SSTA_Frequency_Standard_Deviation"        "SSTA_FrequencyMax"
## [33] "SSTA_FrequencyMean"                       "SSTA_DHW"
## [35] "SSTA_DHW_Standard_Deviation"              "SSTA_DHWMax"
## [37] "SSTA_DHWMean"                             "TSA"
## [39] "TSA_Standard_Deviation"                   "TSA_Minimum"
## [41] "TSA_Maximum"                              "TSA_Mean"
## [43] "TSA_Frequency"                            "TSA_Frequency_Standard_Deviation"
## [45] "TSA_FrequencyMax"                         "TSA_FrequencyMean"
## [47] "TSA_DHW"                                  "TSA_DHW_Standard_Deviation"
## [49] "TSA_DHWMax"                               "TSA_DHWMean"
## [51] "Region"                                   "Diversity"
## [53] "rate_of_SST_change"                       "Longitude.Degrees"
## [55] "Latitude.Degrees"
```

Representing spatial information

THE UNIVERSITY OF SYDNEY

Representing spatial information

- There are several different R spatial formats to choose from. Your choice of format will largely be dictated by the package(s) and or function(s) used in your workflow.
- Since 2000, there is an increase of packages and tools to enable R to interface with geographic data.
- There are now many packages listed in the CRAN task view. It is a good overview of the variety of packages that deal with GIS for spatial data. https://cran.r-project.org/web/views/Spatial.html

Map data

```
library(maps) ## careful there are some legacy maps here
world_map <- map_data("world") ## low resolution map
head(world_map)
```

```
##        long      lat group order region subregion
## 1 -69.89912 12.45200     1     1  Aruba      <NA>
## 2 -69.89571 12.42300     1     2  Aruba      <NA>
## 3 -69.94219 12.43853     1     3  Aruba      <NA>
## 4 -70.00415 12.50049     1     4  Aruba      <NA>
## 5 -70.06612 12.54697     1     5  Aruba      <NA>
## 6 -70.05088 12.59707     1     6  Aruba      <NA>
```

```
dim(world_map)
```

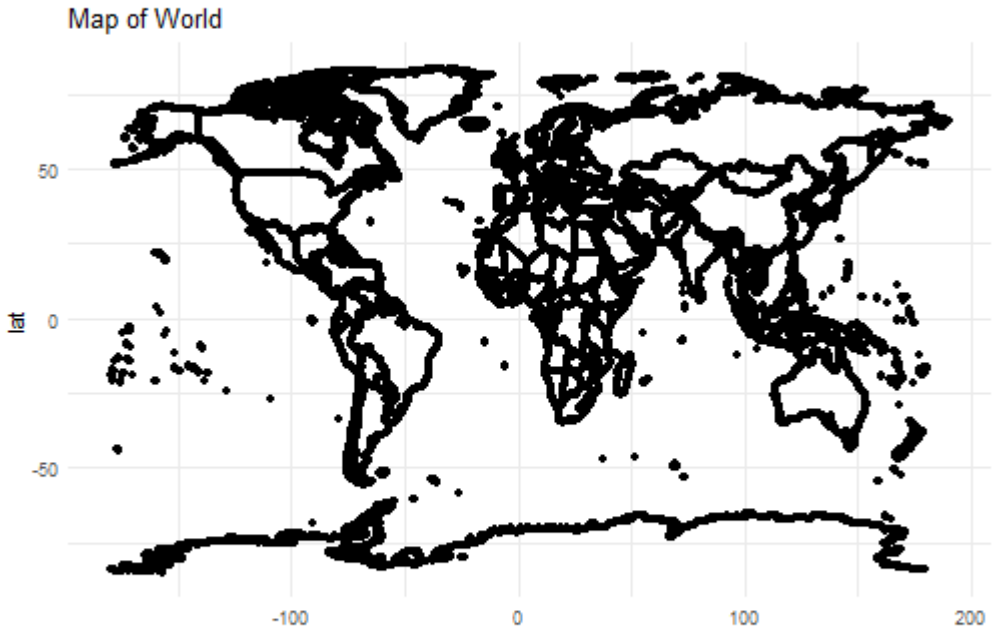
```
## [1] 99338     6
```

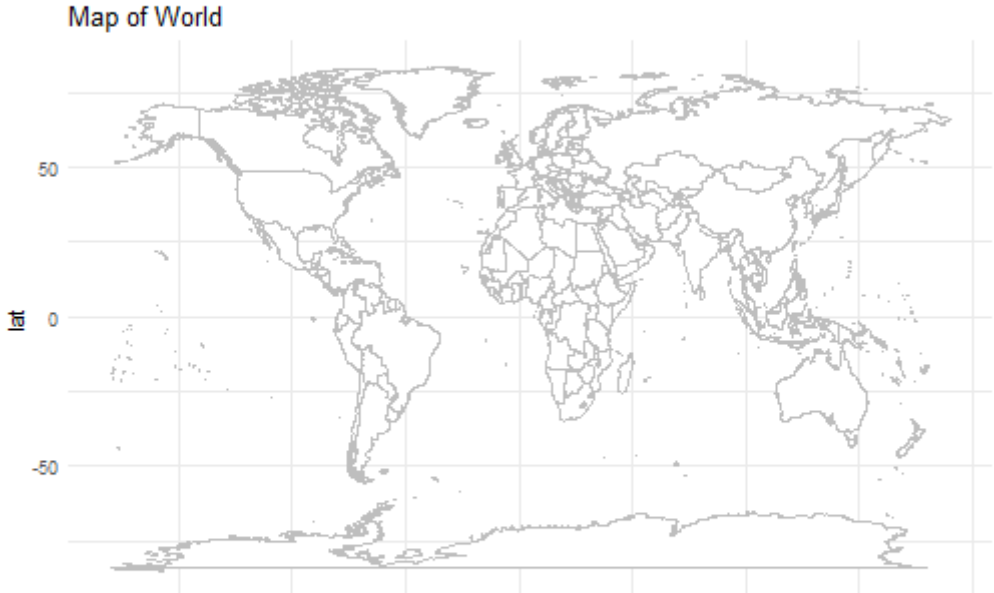
The function `map_data` turns data from the maps package in to a data frame suitable for plotting with `ggplot2`.

Map data

```
ggplot(world_map, aes(x = long, y = lat)) +
  geom_point() + theme_minimal() +
  theme(legend.position = "bottom",
        aspect.ratio = 0.6) +
  ggtitle("Map of World")
```

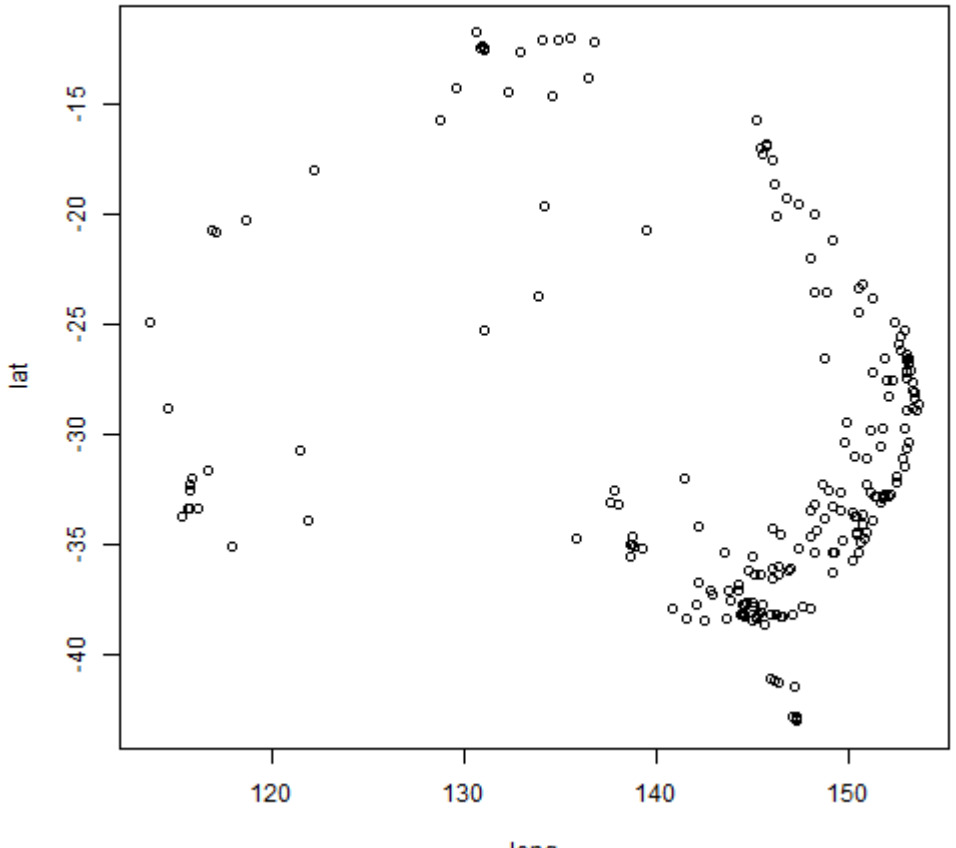
```
ggplot(world_map, aes(x = long, y = lat, group
  geom_polygon(fill = "white", colour = "gray")
  theme_minimal() + ggtitle("Map of World") +
  theme(legend.position = "bottom",
        aspect.ratio = 0.6)
```





Map data - Australia

```
df <- world.cities[world.cities$country.etc == "Australia",]
plot(df[, c("long", "lat")])
```



Using Google Maps - extension

- There are a series of functions that are based on the Google Geocoding API.
- To use Google's Geocoding API, you must first enable the API in the Google Cloud Platform Console. See ?register_Google.
- To obtain an API key and enable services, go to https://cloud.google.com/maps-platform/.
- The `mutate_geocode()` function uses Google Maps to find the longitude and latitude of each location.
- The `get_map()` function is a smart wrapper that queries the Google Maps, OpenStreetMap, Stamen Maps or Naver Map servers for a map.

```
## not tested
library(ggmap)
myMap <- get_map(location = "Australia", zoom = 4)
ggmap(myMap) +
geom_point(data = df[, c("long","lat", "pop")], aes(x=long, y = lat, colour = pop > 1000000))
```

Leaflet - basic usage

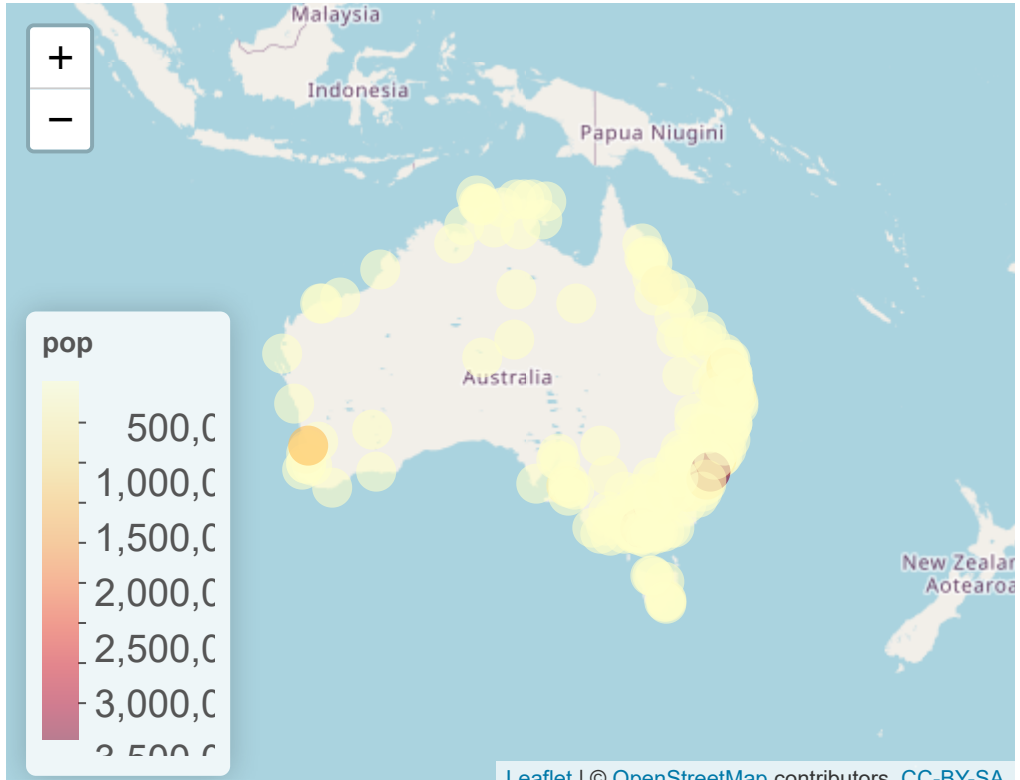
- Here is another option.

Create a Leaflet map with these basic steps:

- Create a map widget by calling `leaflet()`.
- Add layers (i.e., features) to the map by using layer functions (e.g. `addTiles`, `addMarkers`, `addPolygons`) to modify the map widget.
- Repeat previous steps as required.
- Print the map widget to display it.

Map data - Australia population, extension

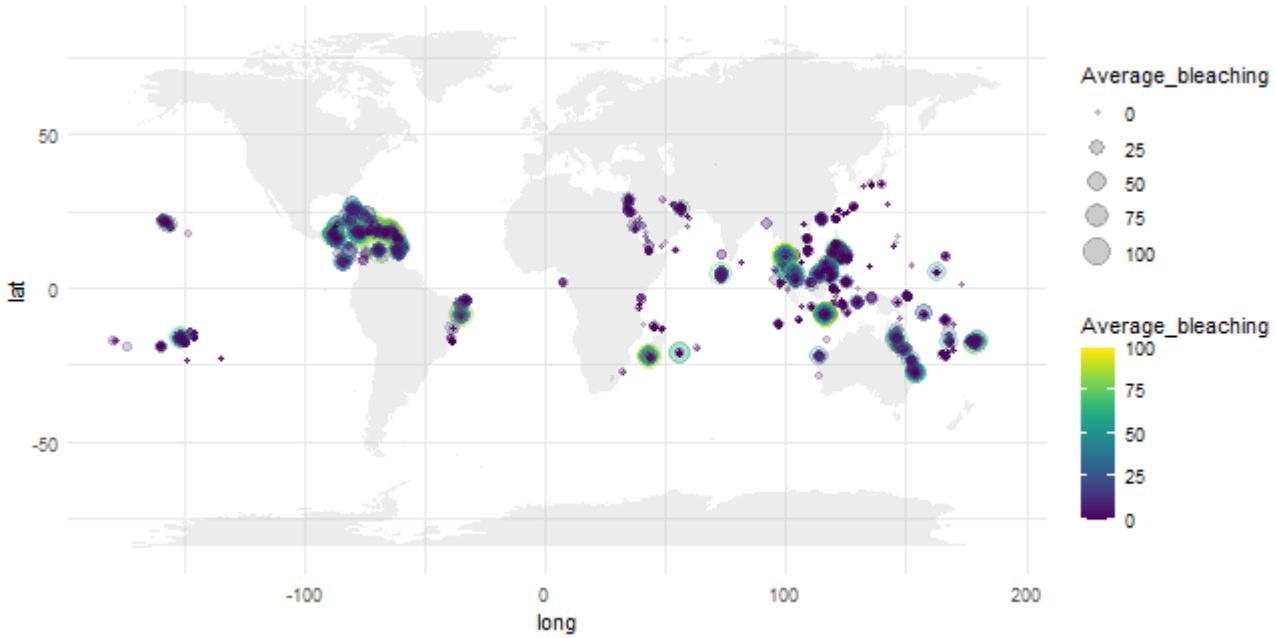
```
library(leaflet)
pal <- colorNumeric(palette = "YlOrRd", domain = df$pop)
leaflet(data = df) %>% addTiles() %>%
    addCircleMarkers(lat = ~lat, lng = ~long, popup = ~name,
                     color = ~pal(pop), stroke = FALSE, fillOpacity = 0.6) %>%
    addLegend(position = "bottomleft", pal = pal, values = ~pop)
```



IDA - Back to the task

- **Task:** To visualise the level of coral bleaching in various reefs on a world map

```
world_map <- map_data("world")
ggplot()+geom_polygon(data =world_map, aes(x=long, y = lat, group = group), fill="grey", alpha=0.3) +
  geom_point(data =reef, alpha = 0.2, aes(y=Latitude.Degrees, x= Longitude.Degrees,
  size=Average_bleaching, color=Average_bleaching)) + scale_colour_viridis() + theme_minimal()
```



Classification

THE UNIVERSITY OF SYDNEY

Can we predict Coral Bleaching?

Let us try to predict the presence of coral bleaching using the following variables:

- `ClimSST` - Sea surface temperature
- `SSTA` - Sea surface temperature anomaly
- `SSTA_DHW` - Sea surface temperature "Degree Heating Weeks"
- `TSA` - Thermal stress anomaly

We only consider the Atlantic Ocean for simplicity, and perform some data cleaning.

```
reef_clean = reef %>%
  filter(Ocean == "Atlantic") %>%
  select(ClimSST, SSTA, SSTA_DHW, TSA, Average_bleaching) %>%
  na.omit %>%
  mutate(Average_bleaching = factor(Average_bleaching > 0))
```

Review: Classification

- Classification is the problem of identifying which of a set of categories (sub-populations) an observation (or observations) belongs to, using a set of properties (explanatory variables)
- This is a category of supervised learning (where examples are labelled)
- There are wide range of statistical classifiers with different advantages and disadvantages. We will look at the following:
  - Logistic regression
  - Decision tree
  - Random Forest
  - k Nearest Neighbours
  - Linear Discriminant Analysis
  - Support Vector Machine

Classification

- Let us split the data into 70% training and 30% testing sets.
- This allows us to test the performance of our models on **unseen** data.

```
set.seed(10)
trainingIndex = sample(1:nrow(reef_clean), round(nrow(reef_clean)*0.7))
training_data <- reef_clean[trainingIndex,]
testing_data <- reef_clean[-trainingIndex,]

X_train <- training_data %>% select(-Average_bleaching)
X_test <- testing_data %>% select(-Average_bleaching)
y_train <- training_data$Average_bleaching
y_test <- testing_data$Average_bleaching
```

Logistic Regression

- Modelling binary data since Y_i is either 0 or 1, it is natural to model Y as a Bernoulli random variable: $Y_i|\mathbf{x}_i \sim Bernoulli(p(\mathbf{x'}_i, \beta))$, where the probability that $Y_i = 1$ is given by some function of our predictors, $p(x'_i, \beta)$.
- A logistic regression model begins with,

$$Y_i|\mathbf{x}_i \; Bernoulli\left(\frac{exp(\mathbf{x'}_i\beta)}{1 + exp(\mathbf{x'}_i\beta)}\right)$$

- If we had a new observation vector $\mathbf{x}_0$ and we knew the vector β, we could calculate the probability that the corresponding $Y = 1$:

$$P(Y = 1|\mathbf{x}_0) = \frac{exp(\mathbf{x'}_0\beta)}{1 + exp(\mathbf{x'}_0\beta)}$$

- One classification rule is: if this probability is greater than 0.5, we would make the prediction $\hat{Y} = 1$, otherwise we would predict $\hat{Y} = 0$.

Logistic Regression in R

- We use the `glm` function to fit the logistic regression model.
- We use the `predict` function to use the model to make predictions on new data
- The output of the prediction is a **probability**. We round it to convert it to a classification.

```
trained_glm <- glm(Average_bleaching ~ . , data = training_data, family = binomial)
probs_glm <- predict(trained_glm, X_test, type = "response")
predicted_glm <- ifelse(probs_glm > 0.5, "TRUE", "FALSE")
table(y_test, predicted_glm)
```

```
##         predicted_glm
## y_test  FALSE TRUE
##   FALSE   189   95
##   TRUE    130  152
```

```
round(mean(y_test == predicted_glm),3)
```

```
## [1] 0.602
```

Decision Tree

- Decision tree is a type of supervised learning algorithm that can be used in both regression and classification problems. It works for both categorical and continuous input and output variables.
- A decision tree determine the predicted outcome based on series of questions and conditions.
- Creation of Binary Decision Trees involves two key components:
  - What features do we use to make our decisions?
  - What is the threshold for classifying each decision into a **yes** or **no** answer?
- Binary tree structured classifiers are constructed by repeated splits of subsets (nodes) of the measurement space X into two descendant subsets (starting with X itself)
- Each terminal subset is assigned a class label; the resulting partition of X corresponds to the classifier

Decision Tree in R

- We use the `rpart` function to fit a decision tree.

```
library(rpart)
trained_tree = rpart(Average_bleaching ~ . , data = training_data)
predicted_tree = predict(trained_tree, X_test)
predicted_tree = ifelse(predicted_tree[,"TRUE"] > 0.5, "TRUE", "FALSE")
table(y_test, predicted_tree)
```

```
##         predicted_tree
## y_test   FALSE TRUE
##   FALSE    164  120
##   TRUE     132  150
```

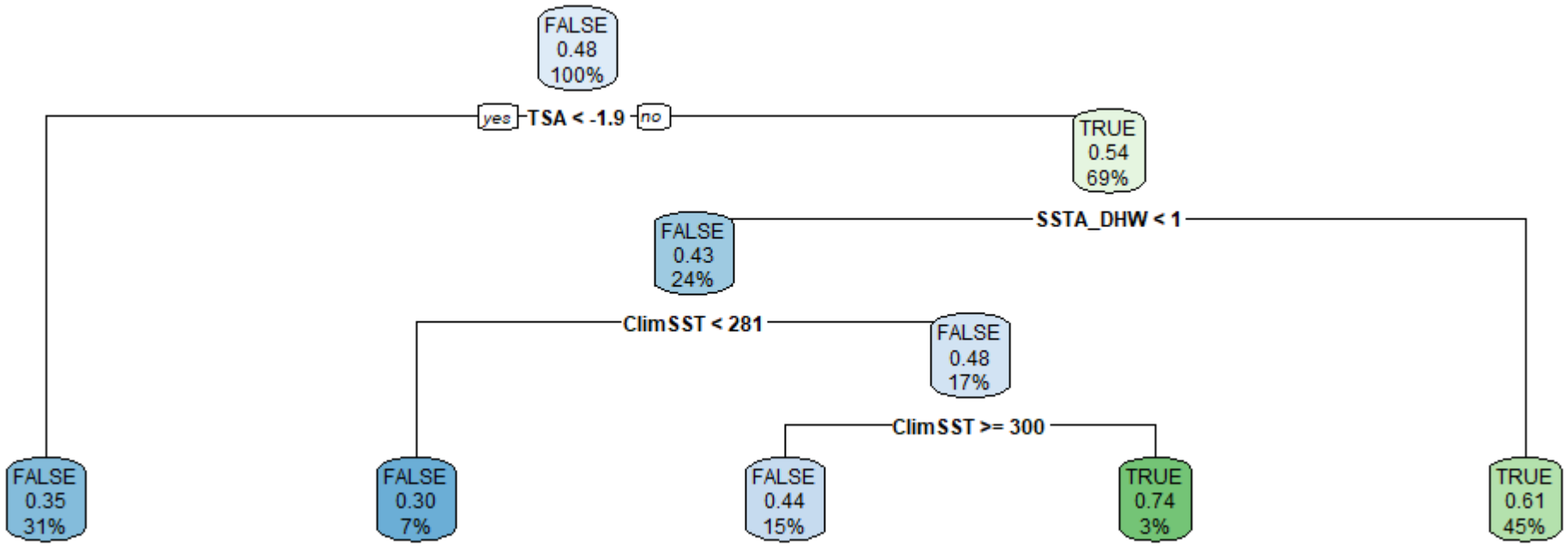
```
round(mean(y_test == predicted_tree),3)
```

```
## [1] 0.555
```

Decision Tree in R

- We use the `rpart.plot` function to visualise a decision tree.

```
library(rpart.plot)
rpart.plot(trained_tree)
```



Random Forest

- A random forest is a set of decision trees which each guess (predicts) the class of each sample.
- The class which has the most common "guesses" for a particular sample is predicted as that sample's class.
- The decision boundary is often non-linear. For example, IF value < 5 AND IF value > 11 THEN Class 1.

Random Forest in R

- We fit a random forest using the function `randomForest`
- Random forests have a large number of options. Here, the default values are used.

```
library(randomForest) ##install.packages("randomForest")
trained_rf = randomForest(X_train, y_train)
predicted_rf = predict(trained_rf, X_test)
table(y_test, predicted_rf)
```

```
##         predicted_rf
## y_test   FALSE TRUE
##    FALSE   190   94
##    TRUE    100  182
```

```
round(mean(y_test == predicted_rf), 3)
```

```
## [1] 0.657
```

k Nearest Neighbours

- k-nearest neighbours (kNN) is a non-parametric algorithm (doesn’t assume the data follows any particular shape).
- In kNN, we vote on the class of a new data point by looking at the majority class of the k nearest neighbours.

k Nearest Neighbours in R

- We use the `knn` function to perform k nearest neighbours.
- As there is no "fitted model", we need to use the entire training data and labels to make the predictions.

```
library(class)
predicted_knn5 = class::knn(train = X_train, test = X_test, cl = y_train, k = 5)
table(y_test, predicted_knn5)
```

```
##         predicted_knn5
## y_test   FALSE TRUE
##    FALSE   168  116
##    TRUE    110  172
```

```
round(mean(y_test == predicted_knn5),3)
```

```
## [1] 0.601
```

Maximum likelihood (ML) discriminant rule

- A maximum likelihood estimator (MLE) chooses the parameter value that makes the chance of the observations the highest.
- For known class conditional densities $p_k(\mathbf{x})$, the maximum likelihood (ML) discriminant rule predicts the class of an observation $\mathbf{x}_0$ by $C(X) = argmax_k p_k(\mathbf{x})$
- For quadratic discriminant analysis, we assume the data is have a multivariate Gaussian (normal) distribution. In practice, population mean vectors μ_k and covariance matrices Σ_k are estimated by corresponding sample quantities
- Diagonal Linear Discriminant Analysis (DLDA) is special version of Linear Discriminant Analysis which assumes no covariance between the features. If there were only two variables used, the separation line would be a straight line. In higher dimensions, it is a plane.

DLDA: A linear boundary classifier

- We use the function `DLDA` from the package `sparsediscrim`.

```
library(sparsediscrim)
trained_dlda = lda_diag(Average_bleaching ~ . , data = training_data)
predicted_dlda = predict(trained_dlda, X_test)
table(y_test, predicted_dlda)
```

```
##         predicted_dlda
## y_test  FALSE TRUE
##   FALSE   175  109
##   TRUE    136  146
```

```
round(mean(y_test == predicted_dlda),2)
```

```
## [1] 0.57
```

Support vector machine

- An SVM classifier may either use a linear or non-linear boundary, depending on the user's choice of *kernel*.
- Understanding how it works requires an understanding of convex optimisation, so that is not covered in this course.
- The default kernel used is the radial basis function kernel which calculates a non-linear boundary.

Support vector machine in R

- We use the `svm` function from the `e1071` to fit a decision tree.

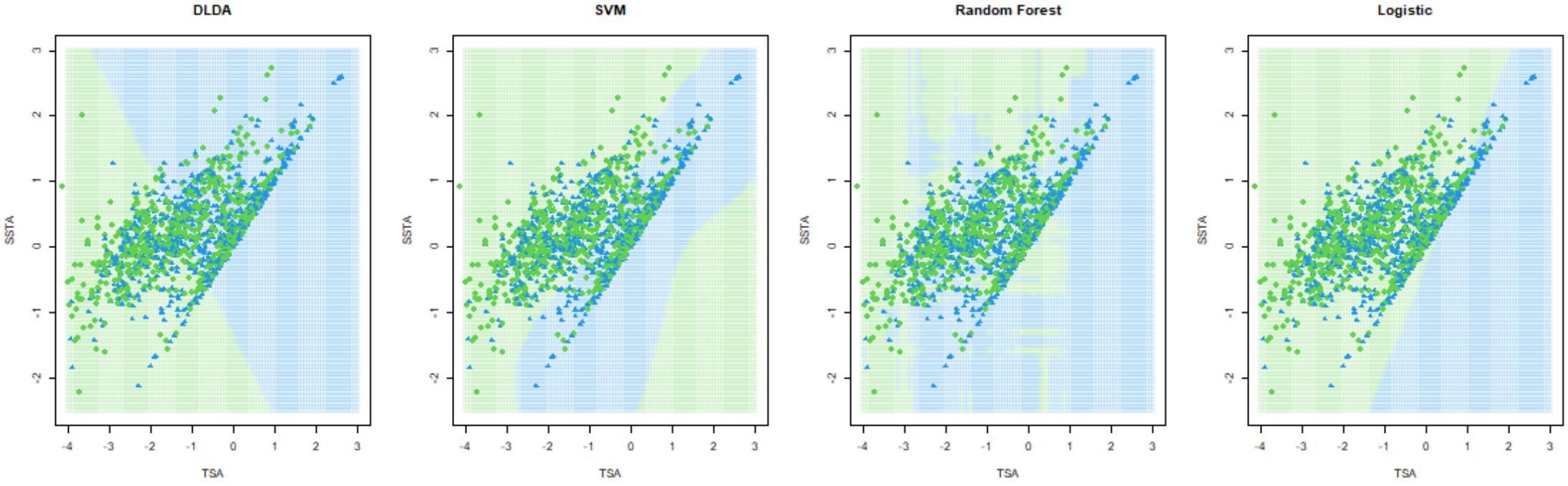
```
library(e1071)
trained_svm = svm(X_train, y_train, kernel ="radial")
predicted_svm = predict(trained_svm, X_test)
table(y_test, predicted_svm)
```

```
##         predicted_svm
## y_test   FALSE TRUE
##    FALSE   183  101
##    TRUE    131  151
```

```
round(mean(y_test== predicted_svm), 2)
```

```
## [1] 0.59
```

Comparison: decision boundaries



Summary

- There are a wide range of different statistical classifiers:
  - Logistic regression
  - Decision tree
  - Random Forest
  - k Nearest Neighbours
  - Linear Discriminant Analysis
  - Support Vector Machine
  - Many more!
- Each has different advantages and disadvantages which will have different performance on different data!