National College of Ireland

Configuration Manual

MSc Research Project
MSc. in Data Analytics

Richard Burke
Student ID: 15034097

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Richard Burke
Student ID:	15034097
Programme:	MSc. In Data Analytics **Year:** 2020
Module:	Research Project
Lecturer:	Catherine Mulwa
Submission Due Date:	17/08/2020
Project Title:	The Significance of External Factors on an Individual's Health Determination
Word Count:	1027 **Page Count:** 11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Richard Burke
Date:	17/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Richard Burke
15034097

1 Introduction

The objective of this configuration manual is to offer the reader an understanding of the software and hardware requirements required for the delivery of the research project: 'Extraneous Factors in an Individual's Health Determination'. Furthermore, the reader will be able to implement independently post completion of this document.

2 System Configuration

2.1 Hardware

Model: Dell Latitude 7480, 2018, 256 GB
Processor: 2.8 GHz Intel Core i7-7600U
RAM: 16 GB 2904 MHz
Graphics: Intel HD Graphics 620 Dynamic ram allocation
Operating System: Microsoft Windows 10

2.2 Software

R Studio: Version 1.2.1335
R: R version 3.6.1 (2019-07-05)

3 Project Development

Following a Cross Industry Standard Process for Data Mining (CRISP- DM)
Methodology, the data collection, data transformation, data processing, and data visualisation are outlined below.

3.1 Datasets

Deprivation index data received from Jonathan Pratschke. Ireland shapefile and CSO data downloaded from the Irish Governmental Open Data site.[1]

Data sets:

- Playgrounds (2011)
- Streetlights (2010)
- Parks (2011)
- Schools (2011)
- Health Centres (2011)
- Leisure Sites (2009)
- Trees (2011)
- Census 2016 Small Area Population Statistics
- Ireland Census 2016 Boundary File

Jonathan Pratschke & Trutz Hasse (RIP):

- SAP Level Deprivation Index data – 2016[2]

3.2 Data Preparation

Data preparation is completed in R using R Studio. To generate a meaningful response variable, the self-assessed health determination from the census was ranked based on the proportion of total residents considering their health to be fair, bad, or very bad.

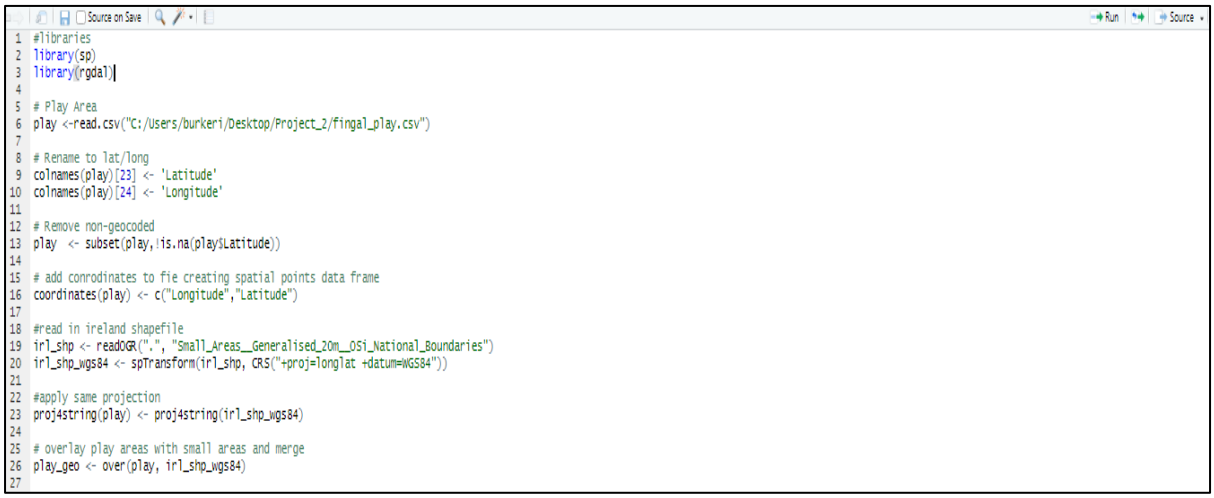
These were then split into five equal-sized groups based on their ratio and assigned categories.

```
#libraries
library(Hmisc)
library(rbin)

# Creating Ratio

shp_fingal$Health_Ratio <- (
  shp_fingal$T12_3_FT + shp_fingal$T12_3_BT + shp_fingal$T12_3_VBT)/
  (shp_fingal$T12_3_FT + shp_fingal$T12_3_BT + shp_fingal$T12_3_VBT + shp_fingal$T12_3_VGT + shp_fingal$T12_3_GT
   )

# Splitting equally by Ratio

Fing$Bucket <- as.numeric(cut2(Fing$Health_Ratio, g=5))

# Labelling Assigned

Fing$Assigned <-  ifelse(Fing$Bucket == 1,
                         'Higher',
                         ifelse(Fing$Bucket == 2,
                                'Above',
                                ifelse(Fing$Bucket == 3,
                                       'Middle',
                                       ifelse(Fing$Bucket == 4,
                                              'Below',
                                              ifelse(Fing$Bucket == 5,
                                                     'Poor',
                                                     "" )))))
```

Figure 1: Assigned Categories in R Studio

[1] https://data.gov.ie/organization/fingal-county-council
[2] http://trutzhaase.eu/deprivation-index/the-2016-pobal-hp-deprivation-index-for-small-areas/

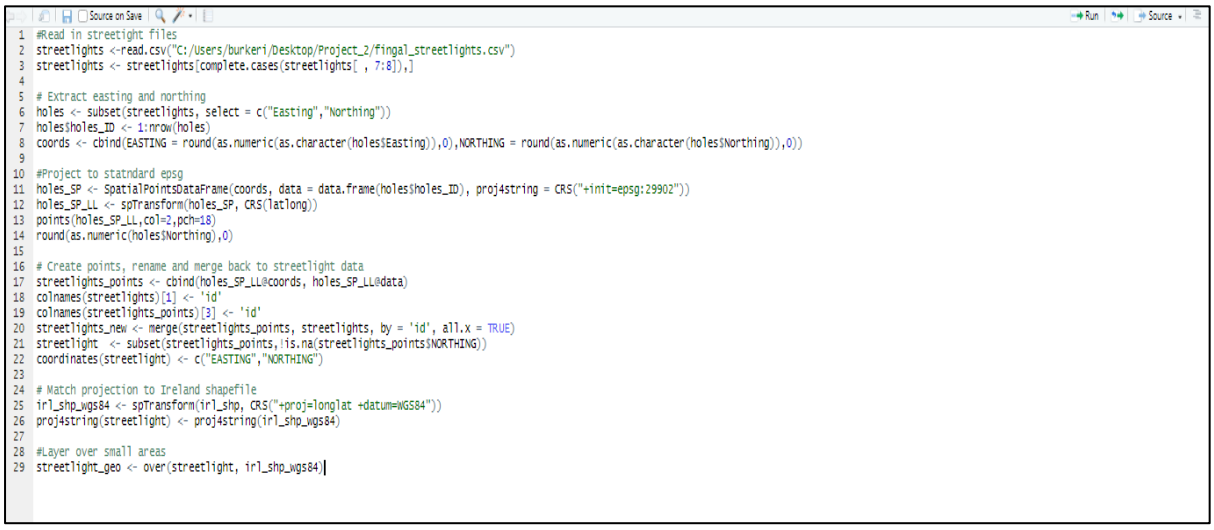
Every facility or amenity needed to be mapped to a small area to generate aggregate statistics. These are completed in R Studio using the spatial packages (sp and rgdal). This was completed for each amenity separately before being amalgamated. Example of transformation for play areas included in Figure 2.



```r
#libraries
library(sp)
library(rgdal)

# Play Area
play <-read.csv("C:/Users/burkeri/Desktop/Project_2/fingal_play.csv")

# Rename to lat/long
colnames(play)[23] <- 'Latitude'
colnames(play)[24] <- 'Longitude'

# Remove non-geocoded
play  <- subset(play,!is.na(play$Latitude))

# add conrodinates to fie creating spatial points data frame
coordinates(play) <- c("Longitude","Latitude")

#read in ireland shapefile
irl_shp <- readOGR(".", "Small_Areas__Generalised_20m__OSi_National_Boundaries")
irl_shp_wgs84 <- spTransform(irl_shp, CRS("+proj=longlat +datum=WGS84"))

#apply same projection
proj4string(play) <- proj4string(irl_shp_wgs84)

# overlay play areas with small areas and merge
play_geo <- over(play, irl_shp_wgs84)
```

Figure 2: Overlay play areas and small areas in R Studio

The streetlight data set was mapped using easting and northing rather the longitude and latitude. This required a spatial transformation to generate their respective longitude and latitude references.



```r
#Read in streetight files
streetlights <-read.csv("C:/Users/burkeri/Desktop/Project_2/fingal_streetlights.csv")
streetlights <- streetlights[complete.cases(streetlights[ , 7:8]),]

# Extract easting and northing
holes <- subset(streetlights, select = c("Easting","Northing"))
holes$holes_ID <- 1:nrow(holes)
coords <- cbind(EASTING = round(as.numeric(as.character(holes$Easting)),0),NORTHING = round(as.numeric(as.character(holes$Northing)),0))

#Project to statndard epsg
holes_SP <- SpatialPointsDataFrame(coords, data = data.frame(holes$holes_ID), proj4string = CRS("+init=epsg:29902"))
holes_SP_LL <- spTransform(holes_SP, CRS(latlong))
points(holes_SP_LL,col=2,pch=18)
round(as.numeric(holes$Northing),0)

# Create points, rename and merge back to streetlight data
streetlights_points <- cbind(holes_SP_LL@coords, holes_SP_LL@data)
colnames(streetlights)[1] <- 'id'
colnames(streetlights_points)[3] <- 'id'
streetlights_new <- merge(streetlights_points, streetlights, by = 'id', all.x = TRUE)
streetlight  <- subset(streetlights_points,!is.na(streetlights_points$NORTHING))
coordinates(streetlight) <- c("EASTING","NORTHING")

# Match projection to Ireland shapefile
irl_shp_wgs84 <- spTransform(irl_shp, CRS("+proj=longlat +datum=WGS84"))
proj4string(streetlight) <- proj4string(irl_shp_wgs84)

#Layer over small areas
streetlight_geo <- over(streetlight, irl_shp_wgs84)
```

Figure 3: Reprojection of streetlights to latitude longitude

The points for all amenities were aggregated by each small area in Fingal.

```
library(dplyr)

play_count <- play_geo %>% count(SMALL_AREA)
colnames(play_count)[2] <- 'Play_Area'

streetlight_count <- streetlight_geo %>% count(SMALL_AREA)
colnames(streetlight_count)[2] <- 'Streetlight_Area'

parks_count <- parks_geo %>% count(SMALL_AREA)
colnames(parks_count)[2] <- 'Parks_Area'

school_count <- schools_geo %>% count(SMALL_AREA)
colnames(school_count)[2] <- 'School_Area'

health_count <- health_geo %>% count(SMALL_AREA)
colnames(health_count)[2] <- 'Health_Area'

leisure_count <- leisure_geo %>% count(SMALL_AREA)
colnames(leisure_count)[2] <- 'Leisure_Area'

tree_count <- tree_geo %>% count(SMALL_AREA)
colnames(tree_count)[2] <- 'Tree_Area'


Fingal_Areas <- as.data.frame(Fing$SMALL_AREA)
colnames(Fingal_Areas)[1] <- 'SMALL_AREA'
```

Figure 4: Aggregation to small area

Due to the sparse nature of the data sets (apart from trees and streetlights), secondary factors were created to represent the nearest distance from each polygon (small area) to the amenity.

```
########### Park_Distance  #############

# Define global reference ellipsoid and a gravity field model
utmStr <- "+proj=utm +zone=%d +datum=NAD83 +units=m +no_defs +ellps=GRS80"
crs <- CRS(sprintf(utmStr, 29))

# Apply CRS
proj4string(parks) <- CRS(proj4string(Fingal_Shp_File))
parks <- spTransform(parks, crs)
Fingal_Shp_File <- spTransform(Fingal_Shp_File, crs)

# Determine distance
a <- gDistance(Fingal_Shp_File,parks,byid=TRUE)
park_name <- as.data.frame(row.names(a))
polygon_name <- as.data.frame(colnames(a))
# Vecotr of column names and min distance
vec = colnames(a)
min_dist_park <- as.data.frame(apply(a[,vec],2,min))
colnames(min_dist_park)[1] <- 'min_dist_park'
min_dist_park_num <- as.data.frame(apply( a, 2, which.min))
min_dist_park$row_id <- row.names(min_dist_park)
Fingal_Shp_Frm <- Fingal_Shp_File@data
Fingal_Shp_Frm$row_id <- row.names(Fingal_Shp_Frm)

#merge back to main shapefile
dist_to_park <- merge(min_dist_park, Fingal_Shp_Frm[, c('SMALL_AREA', 'row_id')], by = 'row_id')
```

Figure 5: Derive minimum distance between points and polygons

Finally, combine all the data derived data sets ready for processing and visualisation.

```r
#Merge data sets
HP_Index <- read.csv("C:/Users/burkeri/Desktop/Project_2/Copy of HP Index 2006-2016 HP Index Scores

HP_Index$Small_Area_New <- ifelse(HP_Index$SmallArea2017 == '',
                                  as.character(HP_Index$SmallArea2011),
                                  as.character(HP_Index$SmallArea2017))

Fingal_Areas <-  merge(Fingal_Areas, play_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, streetlight_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, parks_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, school_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, health_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, leisure_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <-  merge(Fingal_Areas, tree_count, by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas <- merge(Fingal_Areas, dist_to_health, by = 'SMALL_AREA')
Fingal_Areas$row_id <- NULL
Fingal_Areas <- merge(Fingal_Areas, dist_to_schools, by = 'SMALL_AREA')
Fingal_Areas$row_id <- NULL
Fingal_Areas <- merge(Fingal_Areas, dist_to_playg, by = 'SMALL_AREA')
Fingal_Areas$row_id <- NULL
Fingal_Areas <- merge(Fingal_Areas, dist_to_leisure, by = 'SMALL_AREA')
Fingal_Areas$row_id <- NULL
Fingal_Areas <- merge(Fingal_Areas, dist_to_park, by = 'SMALL_AREA')
Fingal_Areas$row_id <- NULL
Fingal_Areas <-  merge(Fingal_Areas, Fing[,c(1, 32)], by = 'SMALL_AREA', all.x = TRUE)
Fingal_Areas[is.na(Fingal_Areas)] <- 0
Fingal_Areas_2 <- merge(Fingal_Areas, HP_Factors[,c(1:3)], by = "SMALL_AREA", all.x = TRUE)
```

Figure 6: Merge the derived points and aggregations

4 Visualisation and Inspection

The aggregated data sets were inspected to produce visual and tabular depictions showing the dispersion of areas with below-average health determination across Fingal. The polygons must be fortified to produce the shapes and merged back to the derived data.

The plots are built using ggplot2 and colour coding polygons based on their health categorisation.

```r
# in order to plot polygons, first fortify the data
Fingal_Shp_File@data$id <- rownames(Fingal_Shp_File@data)
Fingal_Shp_File.df <- fortify(Fingal_Shp_File, region='id')


#Merge to original data
Fingal_Shp_File.df <- merge (Fingal_Shp_File.df,as.data.frame(Fingal_Shp_File[,c('id','Assigned')]), by = "id")

#Make reposnse a factor
Fingal_Shp_File.df$Assigned <- factor(Fingal_Shp_File.df$Assigned, levels = c("Higher", "Above", "Middle", "Below", "Poor"))

# build plot
Fingal_Shp_FilegMap1 <- ggplot(data = Fingal_Shp_File.df, aes(long, lat, group=id))
Fingal_Shp_FilegMap1 <- Fingal_Shp_FilegMap1 + geom_polygon(aes(fill = Assigned))
##Fingal_Shp_FilegMap1 <- Fingal_Shp_FilegMap1 + geom_path(colour = 'gray', linestyle = 2)
Fingal_Shp_FilegMap1 <- Fingal_Shp_FilegMap1 + scale_fill_brewer(palette='YlGnBu')
Fingal_Shp_FilegMap1 <- Fingal_Shp_FilegMap1 + labs(title="Fingal by Health Categorisation")
```

Figure 7: Depict visually using ggplot2

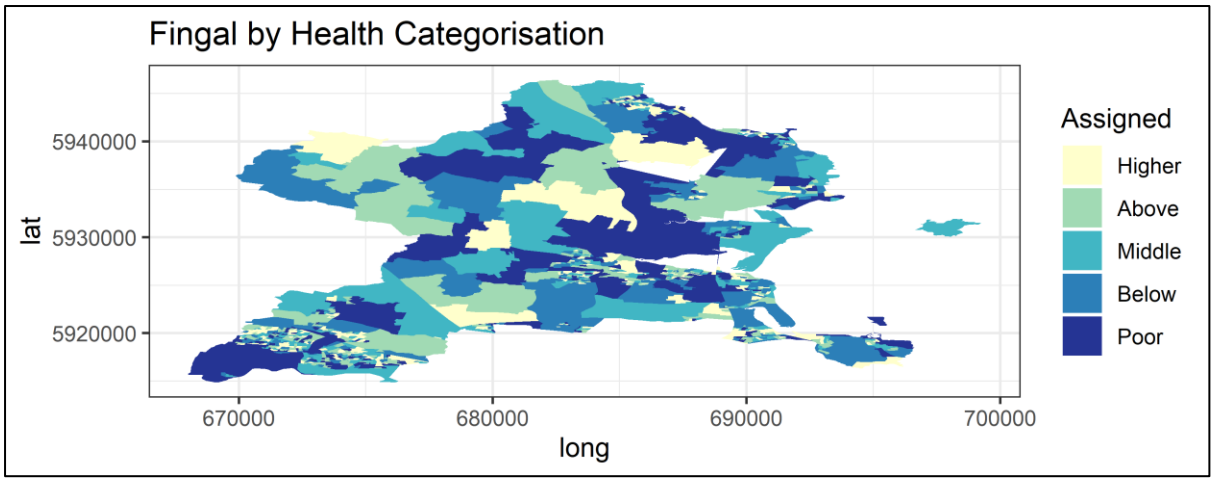


Figure 8: Dispersion of health categorisation across Fingal

The data was further aggregated by Assigned category and Electoral district to view of the underlying data points for each entity.

```
159
160 summed_tables <- aggregate(.~Assigned, Fingal_Areas, sum)
161 summed_ED_tables <- aggregate(.~EDNAME, Fingal_Areas, sum)
162 mean_tables <- aggregate(.~Assigned, Fingal_Areas, mean)
163 mean_ED_tables <- aggregate(.~EDNAME, Fingal_Areas, mean)
164
165
```

Figure 9: Aggregate to ED and Category

Table 1: Lowest 5 by categorisation

Assigned	*EDNAME*	*n*	*Total*	*Percentage*
Poor	Blanchardstown-Corduff	10	12	83.33
Poor	Balbriggan Urban	15	28	53.57
Poor	Blanchardstown-Roselawn	3	6	50.00
Poor	Blanchardstown-Tyrrelstown	3	6	50.00
Poor	Kilsallaghan	3	6	50.00

Visually comparing each ED to its overall mean in ggplot2 by transposing the wide data sets to long format and generating group means to add as points.

```r
library(ggplot2)
library(data.table)

#Change mean ED tables from wide to long
df_melted2 <-
  melt(mean_ED_tables[,c(1, 21:31)]
       , id.vars="EDNAME", variable.name="Resource", value.name="Count")

# View lowest areas by mean assigned against overall mean
df_melted2 %>%
  group_by(Resource) %>%
  summarise(Count = mean(Count))
df_melted2 <- df_melted2[order(-df_melted2$Count),]

# Plot with ggplot
p <- ggplot(df_melted2  %>%
              filter(EDNAME %in% c('Blanchardstown-Corduff', 'Blanchardstown-Roselawn',
                                   'Swords Village','Balbriggan Urban', 'Blanchardstown-Tyrrelstown', 'Kilsallaghan')), aes(Resource, Count, fill=Resource)) +
  geom_bar(stat="identity", position = position_dodge()) +
  facet_wrap(~ EDNAME, scales = "free") +
  coord_flip() +
  theme(panel.background = element_blank())  +
  geom_point(data = df_melted2 %>%

               group_by(Resource) %>%
               summarise(Count = mean(Count))) +
  labs(
    title = "Lowest 6 Areas by Health Determination",
    x = "Count",
    y = "Resource"
  )

```

Figure 10: Plotting ED and overall means

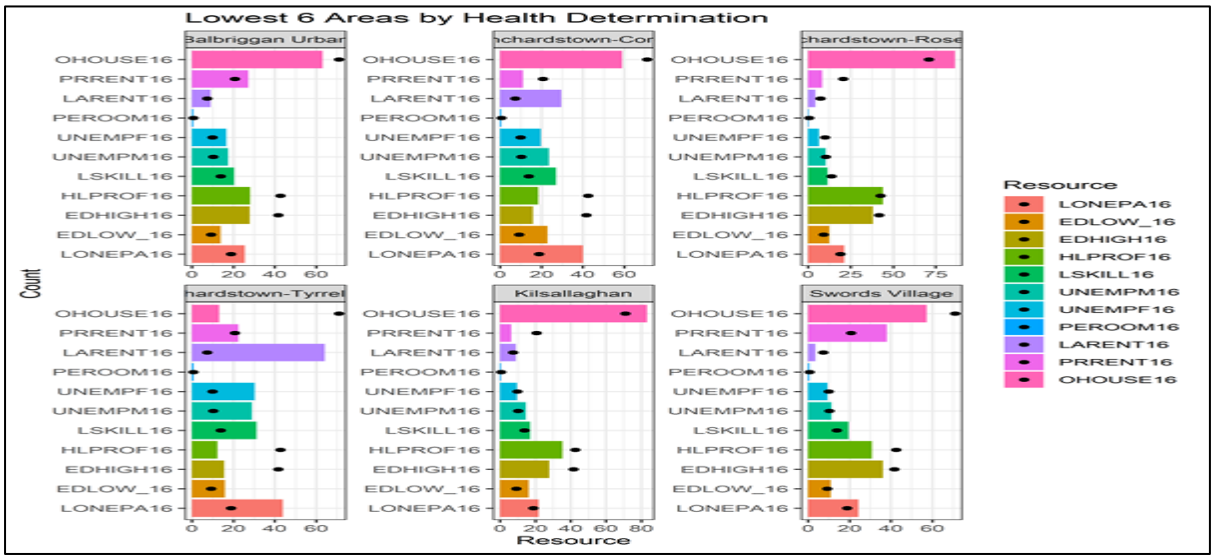


Figure 11: Lowest 6 by self-determined health

5 Modelling

The package Caret in R was used to implement all the models (excluding Auto ML) and to develop optimised parameters for XGBoost and Random Forest. Classification accuracy, precision, and recall were used as measures of a model's performance on a random 25% holdout set.

5.1 Auto ML

Specifying the max number of model's limits run time and ensures model selection can be completed.

```r
# Load library
library(h2o)
library(dplyr)
library(caret)


cols.num <- c("POPCHG16", "AGEDEP16", "LONEPA16", "EDLOW_16","EDHIGH16","HLPROF16",
              "LSKILL16" ,"UNEMPM16",  "UNEMPF16","PEROOM16" ,"LARENT16","PRRENT16","OHOUSE16" )
Fingal_Areas_2[cols.num] <- sapply(Fingal_Areas_2[cols.num],as.numeric)


Fingal_Areas_2$Assigned <- recode(Fingal_Areas_2$Assigned, 'Higher'=5, 'Above'=4, 'Middle'=3, 'Below'=2, 'Poor' = 1)
Fingal_Areas_2$Assigned <- as.factor(Fingal_Areas_2$Assigned)

TrainXG2 <- createDataPartition(Fingal_Areas_2$Assigned, p=0.75, list=FALSE)
trainingXG2 <- Fingal_Areas_2[ TrainXG2, ]
testingXG2 <-  Fingal_Areas_2[ -TrainXG2, ]


# start h2o cluster
invisible(h2o.init())
=
# convert data as h2o type
train_h = as.h2o(trainingXG2)
test_h = as.h2o(testingXG2)

# set label type
y = 'Assigned'
pred = setdiff(names(trainingXG2), y)

#convert variables to factors
trainingXG2[,y] = as.factor(trainingXG2[,y])
testingXG2[,y] = as.factor(testingXG2[,y])

# Run AutoML for 20 base models
aml = h2o.automl(x = pred, y = y,
                 training_frame = train_h,
                 max_models = 20,
                 seed = 1,
                 max_runtime_secs = 3600
)
```

Figure 12: AutoML implementation

The output includes a set of models as a part of a leader board and allows the user to select which model to implement or allow default selection. Predictions of the testing data set can be made within the code, and a confusion matrix demonstrated using caret's in-built function.

AutoML using the package h2o requires pre-processing of the data to execute. This involves recoding the textual response variable to numeric. For this problem, all other variables were set as numeric with a 75%/25% split between training and testing data. The h2o cluster is initialised, and the data sets uploaded.

```
)


# AutoML Leaderboard
lb = aml@leaderboard

aml@leader

# prediction result on test data
prediction = h2o.predict(aml@leader, test_h[,-5]) %>%
  as.data.frame()

prediction$predict
#prediction$predict <- as.factor(round(prediction$predict,0))
# create a confusion matrix
confusion_matrix_h2O <- caret::confusionMatrix(testingXG2$Assigned, prediction$predict)

# close h2o connection
h2o.shutdown(prompt = F)
```

Figure 13: AutoML output

5.2 Naïve Bayes

Naive Bayes was implemented using caret and its in-built functions. The default grid search and parameter optimisation were selected by caret. As with AutoML, the training set was split into a test and train set with a 25% and 75% split respectively.

```
#### Naive Bayes Basic

start.time <- Sys.time()
Assigned_nb <- train(Assigned ~., data = trainingXG2, method = "nb", metric = "Accuracy",importance = TRUE, preProc = c("center", "scale"))
system.time()

library(caret)

nb2_VapIMP <- varImp(Assigned_nb)

test_predict_nb2 <- predict(Assigned_nb, testingXG2)

summary(test_predict_nb2)

confustion_matrix_nb2 <- confusionMatrix(test_predict_nb2, as.factor(testingXG2$Assigned))
confustion_matrix_nb2

end.time <- Sys.time()

time.taken_nb2 <- end.time - start.time
```

Figure 14: Naïve Bayes implementation

5.3 Random Forest

Additional parameters were specified for the Random Forest model to expand the tuning grid and parameter selection. This was implemented using caret on the same test and train data sets.

```
############ Random Forest Tune

start.time <- Sys.time()

 # Specifying cross validation -  10 folds.
trctrl <- trainControl(method = "cv", number = 10)

#SPecify randomly slected predictors
mtry <- sqrt(ncol(trainingXG2))

tunegrid <- expand.grid(.mtry=mtry)

# fit model

rf_fit <- train(Assigned ~., data = trainingXG2, method = "rf",
                metric='Accuracy',
                trControl=control,
                tuneLength = 13
                )
# variable importance

rf_VapIMP <- varImp(rf_fit)


# predict on new data

test_predict_rf <- predict(rf_fit, testingXG2)

#confusion matrix

confusion_matrix_rf <- confusionMatrix(test_predict_rf, as.factor(testingXG2$Assigned))

end.time <- Sys.time()

time.taken_rf <- end.time - start.time
```

Figure 15: Random Forest implementation

5.4 XGBoost

XGBoost (gradient boosted decision trees implementation) was developed using carets manual tuning options to create an expanded grid search for the optimised parameters.

```
####### XGB Tuned

# Specify Train COntrol

trctrl <- trainControl(method = "cv", number = 10)

# Specify expanded tuning grid

tune_grid <- expand.grid(nrounds = 500,
                         max_depth = 30,
                         eta = 0.01,
                         gamma = 1,
                         colsample_bytree = 1,
                         min_child_weight = 100,
                         subsample = 1)

# fit model


xgb_fit <- train(Assigned ~., data = trainingXG2, method = "xgbTree",
                 trControl=trctrl,
                 tuneGrid = tune_grid,
                 tuneLength = 5
                 )

# Variable importance
xgb_var_imp <- varImp(xgb_fit)

# predict
xgb_test_predict <- predict(xgb_fit, testingXG2)

# COnfusion Matrix

confuson_matrix_xgb <- confusionMatrix(xgb_test_predict, as.factor(testingXG2$Assigned))
```

Figure 16: XGBoost implementation

5.5 Multinomial Regression

Multinomial regression was implemented utilising the same parameter optimisation features. However, decay is the critical parameters, and an appropriate search grid was selected. The goal of the decay parameter is to discourage overfitting on data sets and increase reproducibility.

```
#### MLM

tuneGrid_mnl <- expand.grid(decay = seq(0, 1, by = 0.1))

mlm_fit <- train(Assigned ~., data = trainingXG2, method = "multinom",
                 trControl=trctrl
                 ,tuneGrid = tuneGrid_mnl
                 ,tuneLength = 13
)


mlm_var_imp <- varImp(mlm_fit)

mlm_test_predict <- predict(mlm_fit, testingXG2)

summary(mlm_test_predict)

confuson_matrix_mlm <- confusionMatrix(mlm_test_predict, as.factor(testingXG2$Assigned))
confuson_matrix_mlm

```

Figure 17: Multinomial Regression Implementation

6 Model Results

The variable importance (where available) is computed for each model in the processing steps. They are aggregated and visualised. Each model produces predictions on the test data set and the creation of a confusion matrix to allow comparisons of the accuracy, sensitivity, and specificity of each model.

```
# Variable Importance
library(data.table)
# Varimpo Output to Dataframe
mlmVarImp <- as.data.frame(setDT(mlmVarImp, keep.rownames = TRUE)[])
xgbVarImp <- as.data.frame(setDT(xgbVarImp, keep.rownames = TRUE)[])
rfVarImp <- as.data.frame(setDT(rfVarImp, keep.rownames = TRUE)[])
colnames(rfVarImp)[1] <- 'Predictor'
colnames(xgbVarImp)[1] <- 'Predictor'
colnames(mlmVarImp)[1] <- 'Predictor'
colnames(rfVarImp)[2] <- 'Overall'

# Merge to single tabural view
VarImps <- merge(rfVarImp, mlmVarImp, by = 'Predictor')
VarImps <- merge(VarImps, xgbVarImp, by = 'Predictor')

```

Figure 18: Variable Importance

The confusion matrices were amalgamated to show the differential in performance across the accuracy, sensitivity, specificity, and kappa.

```

results <- rbind(confusion_matrix_rf$overall,
confuson_matrix_mlm$overall,
confuson_matrix_xgb2$overall,
confustion_matrix_nb$overall,
confusion_matrix_h2O$overall)
results <- as.data.frame(results)

colnames(Names)[1] <- "Model"
Names <- as.data.frame(c(as.character('Random Forest'),
                         as.character('Multinomial Regression'),
                         as.character('XGBoost'),
                         as.character('Naive Bayes'),
                         as.character('H2o')))

Results <- cbind(Names, results)

```

Figure 19: Confusion Matrices