Common R command to know during the PW

Replace the missing data with the average of the feature in which the data is missing

|  |
| --- |
| dataset$Age = ifelse(is.na(dataset$Age),  ave(dataset$Age, FUN = function (x)mean(x, na.rm = TRUE)),  dataset$Age) |

Encode dataframe column using 1 and 2 integers

|  |
| --- |
| dataset$class = factor(dataset$class,  levels = c('No', 'Yes'),  labels = c(0, 1)) |

Split the dataset into training and test sets.

|  |
| --- |
| library(caTools)# required library for data splition  set.seed(123)  split = sample.split(dataset$Purchased, SplitRatio = 0.8)# returns true if observation goes to the Training set and false if observation goes to the test set.    #Creating the training set and test set separately  training\_set = subset(dataset, split == TRUE)  test\_set = subset(dataset, split == FALSE) |

|  |
| --- |
| library(caret)  # create a list of 80% of the rows in the original dataset we can use for training  validation\_index <- createDataPartition(dataset$Species, p=0.80, list=FALSE)  # select 20% of the data for validation  validation <- dataset[-validation\_index,]  # use the remaining 80% of data to training and testing the models  dataset <- dataset[validation\_index,] |

Scale dataset

|  |
| --- |
| dataset = scale(dataset) |

summarize your dataset :

|  |
| --- |
| dim, summary, sapply, head, levels. |

Create box plots :

|  |
| --- |
| boxplot(x,…) |

Create a barplot :

|  |
| --- |
| plot(y) |

scatterplot matrix

|  |
| --- |
| featurePlot(x=x, y=y, plot="ellipse") |

box and whisker plots for each attribute

|  |
| --- |
| featurePlot(x=x, y=y, plot="box") |

Report on the accuracy of each model by first creating a list of the created models and using the summary function. (knn, svm and rf are already trained models)

|  |
| --- |
| knn ← train(…)  svm ← train(…)  fr ← train(…)  results <- resamples(list(knn=fit.knn, svm=fit.svm, rf=fit.rf))  summary(results) |

Create a plot of the model evaluation results and compare the spread and the mean accuracy of each model.

|  |
| --- |
| dotplot(results) |

Scatterplot 2 variables, eg lstat vs. Medv

|  |
| --- |
| plot(training\_data$lstat, training\_data$medv) |

Plot the regression model.

|  |
| --- |
| plot(training\_data$lstat, training\_data$medv,  xlab = "Houshold with Low Socioeconomic Income",  ylab = "Median House Value",  col = "red",  pch = 20)  # Make the line color blue, and the line's width =3 (play with the width!)  abline(model, col = "blue", lwd =3) |

scatterplot of 2 variables with a smoothed line.

|  |
| --- |
| ggplot(marketing, aes(x = youtube, y = sales)) +  geom\_point() +  stat\_smooth() |

It’s also possible to compute the correlation coefficient between the two variables using the R function cor()

scatterplot of 2 variables with a regression smoothed line.

|  |
| --- |
| ggplot(marketing, aes(youtube, sales)) +  geom\_point() +  stat\_smooth(method = lm) |

Plot different graphe of a model :

|  |
| --- |
| par(mfrow=c(2,2))  plot(model) |

Visualize the correlations

|  |
| --- |
| corrplot.mixed(cor(data)) |

Plot Purchased in function of Age and add the curve of the obtained logistic regression model.

|  |
| --- |
| classifier = glm(…)  plot(training\_set$Age,training\_set$Purchased)  curve(predict(classifier, data.frame(Age=x), type="response"), add=TRUE) |

|  |
| --- |
| library(ggplot2)  ggplot(training\_set, aes(x=Age, y=Purchased)) +  geom\_point() +  stat\_smooth(method="glm", method.args=list(family="binomial"), se=FALSE) |

To get confusion matrix, you can use table(…)

Calculate the accuracy, specificity, sensitivity and the precision of the model.

|  |
| --- |
| CM = table(test\_set[,5], y\_pred)  metrics <- function(CM) {  acc = (CM[1,1]+CM[2,2])/(CM[1,1]+CM[1,2]+CM[2,1]+CM[2,2])  spc = CM[1,1]/(CM[1,1]+CM[1,2])  ses = CM[2,2]/(CM[2,2]+CM[2,1])  my\_list <- list("accuracy" = acc, "specificity" = spc, "sensitivity" = ses)  return(my\_list)}  print(paste("specificity: ",metrics(cm)$specificity))  print(paste("sensitivity: ",metrics(cm)$sensitivity)) |

Plot the ROC curve and calculate AUC value.

|  |
| --- |
| library(ROCR)  score <- prediction(prob\_pred,test\_set[,5])  performance(score,"auc")  auc = as.numeric( performance(score,"auc")@y.values)  plot(performance(score,"tpr","fpr"),col="green")  abline(0,1,lty=8) |

Plot the obtained tree from rpart

|  |
| --- |
| plot(Boston\_tree)  text(Boston\_tree, pretty = 0)  title(main = "Regression Tree") |

|  |
| --- |
| rpart.plot(Boston\_tree) |

|  |
| --- |
| prp(Boston\_tree) |

Print the CP table and plot result

|  |
| --- |
| printcp(Boston\_tree)  plotcp(Boston\_tree) |

prune a decision tree using the cp of smallest tree that is within one standard deviation of the tree with the smallest xerror.

|  |
| --- |
| best\_cv\_error <- Boston\_tree$cptable[which.min(Boston\_tree$cptable[,4]),4]  # achieved at :  index\_best <- which.min(Boston\_tree$cptable[,4])  Boston\_tree$cptable[index\_best,4]  Boston\_tree$cptable[index\_best,5]  Boston\_tree <- prune(Boston\_tree,cp= 0.01665273)  rpart.plot(Boston\_tree)  par(mfrow=c(1,1))  X <- Boston\_train$lstat  Y <- Boston\_train$rm  Z <- Boston\_train$medv  ZZ = abs(Z)/max(Z)  plot(Y~X, col=rgb(0, 0, ZZ), main="trainins set")  abline(v=5.3,lty=2,col=4)  abline(h=7.4,lty=2,col=4) |

visually compare the performance of two models (decision tree and linear model) by plotting the Actual (reality) response values against the predicted values.

|  |
| --- |
| par(mfrow=c(1,2))  plot(Boston\_tree\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Predicted vs Actual: Single Tree, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2)  plot(Boston\_lm\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Predicted vs Actual: Linear Model, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2) |

Plot bagging tree :

|  |
| --- |
| Boston\_bagging = randomForest(…)  plot(Boston\_bagging, col = "#cd0050", lwd = 2, main = "Bagged Trees: Error vs Number of Trees") |

to see the most important predictors in the obtained random forest model.

|  |
| --- |
| importance(Boston\_forest, type = 1)  varImpPlot(Boston\_forest) |

Plot the importance of the predictors to the model

|  |
| --- |
| varImpPlot(Boston\_forest, type = 1) |

Summarty a model

|  |
| --- |
| summary(model) |

Construct a final plot to compare the four trained models

|  |
| --- |
| par(mfrow=c(2,2))  plot(Boston\_tree\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Predicted vs Actual: Single Tree, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2)  plot(Boston\_bagging\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Bagging, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2)  plot(Boston\_forest\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Random Forest, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2)  plot(Boston\_boost\_pred, Boston\_test$medv,  xlab = "Predicted", ylab = "Actual",  main = "Boosting, Test Data",  col = "#cd0050", pch = 20)  grid()  abline(0, 1, col = "dodgerblue", lwd = 2) |

Tuning tree

|  |
| --- |
| library(randomForest)  library(mlbench)  library(caret)  library(e1071)  #Random forest  control <- trainControl(method='oob',  number=4,  search='grid')  #create tunegrid with 15 values from 1:15 for mtry to tunning model. Our train function will change number of entry variable at each split according to tunegrid.  tunegrid <- expand.grid(.mtry = (1:5))  rf\_gridsearch <- train(spam ~ .,  data = training\_set,  method = 'rf',  metric = 'Accuracy',  tuneGrid = tunegrid)  plot(rf\_gridsearch)  #gbm  control <- trainControl(method='cv',  number=5)  #Metric compare model is Accuracy  metric <- "Accuracy"  tunegrid <- expand.grid(.shrinkage =c(0.001,0.01),.n.trees = c(2,5,20),.interaction.depth = c(2,3),.n.minobsinnode =c( 4,1) )  gbm\_cv <- train(spam~.,  data=training\_set,  method='gbm',  metric='Accuracy',  tuneGrid=tunegrid,  trControl=control, verbose=FALSE)  print(gbm\_cv$bestTune)  plot(gbm\_cv) |

Cross-validation on a model

|  |
| --- |
| control <- trainControl(method="cv", number=10)  # kNN  set.seed(7)  fit.knn <- train(Species~., data=dataset, method="knn", metric="Accuracy", trControl=control)  # c) advanced algorithms  # SVM  set.seed(7)  fit.svm <- train(Species~., data=dataset, method="svmRadial", metric="Accuracy", trControl=control)  # Random Forest  set.seed(7)  fit.rf <- train(Species~., data=dataset, method="rf", metric="Accuracy", trControl=control) |

Extract individuals (line 1 to 23) and variables (column 1 to 10) from data:

|  |
| --- |
| data[1:23, 1:10] |

Print the quality of representation of the variables and plot them from PCA

|  |
| --- |
| res.pca <- PCA(dataset, graph = FALSE)  head(res.pca$var$cos2, 4) |

Specify supplementary individuals and variables, the function PCA().

|  |
| --- |
| res.pca <- PCA(decathlon2, ind.sup = 24:27,  quanti.sup = 11:12, quali.sup = 13, graph=FALSE) |

Predict results (coordinates, correlation and cos2) for the supplementary quantitative variable.

|  |
| --- |
| res.pca$quanti.sup  fviz\_pca\_var(res.pca) |

Predict results for the supplementary individuals (ind.sup) and visualize all individuals (active and supplementary ones).

|  |
| --- |
| res.pca$ind.sup  p <- fviz\_pca\_ind(res.pca, col.ind.sup = "blue", repel = TRUE)  p <- fviz\_add(p, res.pca$quali.sup$coord, color = "red")  p |

To explore how the 3 different flower classes are distributed along the 4 different features, visualize them via histograms using the ggplot packages through the geom\_histogram plot.

|  |
| --- |
| library(ggplot2)  require(gridExtra)  # histogram of sepal\_length  plot1 = ggplot(iris, aes(x=sepal\_length, fill=class)) + geom\_histogram(binwidth=.2, alpha=.5)  # histogram of sepal\_width  plot2 =ggplot(iris, aes(x=sepal\_width, fill=class)) + geom\_histogram(binwidth=.2, alpha=.5)  # histogram of petal\_length  plot3 =ggplot(iris, aes(x=petal\_length, fill=class)) + geom\_histogram(binwidth=.2, alpha=.5)  # histogram of petal\_width  plot4 = ggplot(iris, aes(x=petal\_width, fill=class)) + geom\_histogram(binwidth=.2, alpha=.5)  grid.arrange(plot1, plot2, plot3, plot4, ncol=2) |

plot the clusters centroids and add the names of the observations.

|  |
| --- |
| plot(pointsCards, col = km$cluster)  points(km$centers,col=c("red","black"),pch=3,cex=3,lwd=3)  text(x = pointsCards, labels = rownames(pointsCards), col = km$cluster, pos = 3, cex = 0.75) |

Visualize the “within groups sum of squares” of the k-means clustering results

|  |
| --- |
| mydata <- pointsCards  wss <- (nrow(mydata)-1)\*sum(apply(mydata,2,var))  for (i in 2:15){  wss[i] <- sum(kmeans(mydata,centers=i)$withinss)  }  plot(1:15, wss, type="b", xlab="Number of Clusters",  ylab="Within groups sum of squares") |

visualize the ‘between\_SS / total\_SS

|  |
| --- |
| for (i in 1:15){  wss[i] <- kmeans(mydata,centers=i)$betweenss/kmeans(mydata,centers=i)$totss  }  plot(1:15, wss, type="b", xlab="Number of Clusters",ylab="between\_SS / total\_SS") |

Visualize the teams on the first two principal components and color them with respect to their cluster.

|  |
| --- |
| #based on factoextra library  fviz\_cluster(km.ligue1, data = ligue1, # km.ligue1 is where you stored your kmeans results  palette = c("red", "blue", "green"), # 3 colors since 3 clusters  ggtheme = theme\_minimal(),  main = "Clustering Plot"  ) |

Chek that there is no missing data

|  |
| --- |
| NA\_Check <- is.na(customer\_data)  summary(NA\_Check) |

Compute the Hopkins statistic

|  |
| --- |
| library(magrittr)  gradient.color <- list(low = "steelblue", high = "white")  customer\_data[, -1] %>% # Remove column 1 (ID)  scale() %>% # Scale variables  get\_clust\_tendency(n = 50, gradient = gradient.color) |