General

|  |  |
| --- | --- |
| ds <- read.csv(“filename”, [header=FALSE,] [stringAsFactor=FALSE]) |  |
| ds <- read.table(“filename”, header=FALSE, sep=”|”, quote=”\””) |  |
| str(x) |  |
| summary(x) |  |
| subset(ds, condition) | table(subset(CPS, Hispanic == 1)$Race) |
| table(x, y) | table(CPS$Region, is.na(CPS$Married)) |
| tapply(x, y, fun) |  |
| sort(x) | Can be applied to table(x) |
| merge(ds.x, ds.y, by.x=col, by.y = col, all.x=TRUE) |  |
| df$var = coredata(var) | Put var in df |
| df[is.na(df)] <- 0 |  |
| df = unique(df) |  |
| df$var = NULL |  |
| colnames(dtmTitle) = paste0("T", colnames(dtmTitle)) / c(“1”, “2”) | Add “T” to column names |
|  |  |
| grepl(“cat”,”dogs and cats”,fixed=TRUE) |  |
|  |  |
| wikiWords = cbind(wordsAdded, wordsRemoved) |  |
|  |  |
| matrix(c(1,2,3,4), byrow=TRUE, nrow=5) |  |
| as.matrix(table(….)) |  |
|  |  |
| nchar(str) | #chars in string |
|  |  |
| tail(sort(colMeans(df))) |  |
|  |  |
| as.matrix |  |
| as.vector |  |
| dim(vector) = c(x,y) | Vector to matrix |
|  |  |
| df’ <- na.omit(df) |  |
| df$var = relevel(df$var, “reference value”) | Use most frequent value as reference val |
|  |  |
| as.Date(strptime(mvt$Date, “%m/%d/%y %H:%M”)) |  |
| months(date), weekdays(date) |  |
| var <- lag(zoo(df$var), -2, na.pad=TRUE) | In package zoo, -2 represents lag |
|  |  |
| set.seed(88) | Split a set of observations |
| split = sample.split(df$dep\_var, SplitRatio = 0.75) | caTools |
| dfTrn = subset(df, split == TRUE) |  |
| trainSmall = train[sample(nrow(train), 2000), ] | random downsize |
| complete(mice(df)) | Replace NA’s with imputed values |
| strwrap(email) |  |
| CTRL-L: clear the console |  |
| getwd() |  |
| ls() |  |
| rm(x), rm(list=ls()) |  |
| Sys.setlocale(“LC\_ALL”, “C”) |  |
| Sys.setlocale(“LC\_ALL”, ‘en\_US.UTF-8’) |  |

Graphics

|  |  |
| --- | --- |
| plot(x,y, col=”red”, xlab=”…”, ylab=”…”, main=”…”, type=”l”, lty=2) |  |
| point(x, y, ….) | Plot on top of |
| lines(…) |  |
| hist(x) |  |
| boxplot(x,y) |  |
| abline(x), abline(v=as.Date(c(“2000-03-01”)), lwd=2) |  |
|  |  |
| image(matrix, axes=FALSE, col=gray(seq(0,1,length=256))) |  |

Liniar Regression

|  |  |
| --- | --- |
| lm(dep\_var ~ indep\_var1 + indep\_var2, data=trd) | ~ . takes all indep vars |
| summary(lm) | Significant: Pr < 0.05 |
| cor(trd) |  |
| lm$residuals | SSE = sum(lm$residuals^2)  RMSE = sqrt(SSE/nrow(lm)) |
| predict(lm, newdata=tsd) | SSE = sum((pd – tsd$dv)^2)  SST = sum((mean(trd$dv) –  tsd$dv)^2)  R2 = 1 – SSE / SST  RMSE =sqrt(SSE/nrow(tsd)) |
| df$indep\_var = relevel(as.factor(df$indep\_var), most frequent value) | Replace factor values |
| step(lm) |  |
|  | In case of non-liniar relation take log (Google homework) |

Logistic Regression

|  |  |
| --- | --- |
| glm(dep\_var ~ indep\_var1 +indep\_var2, data=trd, family=binomial) | ~ . takes all indep vars |
| summary(glm) | AIC: smaller is better |
| predict(glm, type=”response”, newdata =tsd) | probabilities |
| Odds = P(y=1)/P(y=0) = exp(b0 + b1\*x1 + b2\*x2 +…..) |  |
| Probability = P (y=1) = 1/(1+exp(-1\*(b+b1\*x1+b2\*x2+….))) |  |
| table(tsd$dep\_var, pd > threshold) | |  |  |  | | --- | --- | --- | | conf | pd = 0 | pd = 1 | | act = 0 | TN | FP | | act = 1 | FN | TP | |
|  | sensitivity = TP / (TP + FN)  specificity = TN / (TN + FP)  overall acc = (TP+TN)/N  overall err rt=(FP+FN)/N  FN err rt=FN/(TP+FN)  FP err rt=FP/(TN+FP) |
|  | threshold ~ specificity |
|  | threshold ~ - 1 \* sensitivity |
| ROCpred = prediction(pd, df$dep\_var)  ROCperf = performance(ROCpred, “tpr”, “fpr”)  plot(ROCperf, colorize =TRUE, print.cutoffs.at(seq(0,1,0.1),  test.adj=c(-0.2,1.7)) | ROCR, ROC curve: y=sensitivity, x=1-specificity |
| Multicollinearity: correlation between indep\_vars and signs of coefficients |  |
| as.numeric(performance(ROCpred, "auc")@y.values) | AUC (0..1), 1 is best |

Classification & Trees

|  |  |
| --- | --- |
| CART |  |
| rpart(DepVar ~ IndepVar + IndepVar, data=dfTrn, method=”class” [, minbucket=25] [, cp=0.05] [, parms=list(loss=panaltyMatrix)]) | rpart  cp ~ 1/(#splits) (0…1) |
| prp(CARTmodel) | rpart.plot |
| pd1 = predict(CARTmodel, newdata=dfTst, type=”class”) | threshold = 0.5 |
| table(df$DepVar, pd1) | confusion matrix |
| pd2 = predict(CARTmodel, newdata=df$tst) | probabilities |
| rpd2 = prediction(pd2[,2], df$DepVar) | ROCR |
| prf = performance(rpd2, “tpr”, “fpr”) |  |
| plot(prf) |  |
|  |  |
| RandomForest |  |
| randomForest(DepVar ~ IndepVar + IndepVar, data=df, nodesize=25, ntree=200) | randomForest |
| pd1 = predict(ForestModel, newdata=df [,type=”prob”]) | Delivers probabilities ([,2]) |
| table(df$DepVar, pd1) | Confusion matrix |
| vu = varUsed(rf, count=TRUE) | Freq of vars in splits |
| vusorted = sort(vu, decreasing = FALSE, index.return = TRUE) |  |
| dotchart(vusorted$x, names(rf$forest$xlevels[vusorted$ix])) |  |
| varImpPlot(rf) | Impurity decrease ~ most used split |
|  |  |
| CrossValdation | Determine optimal cp |
| numFolds = trainControl(method="cv",number=10) | caret, e1071 |
| cpGrid = expand.grid(.cp=seq(0.01, 0.5, 0.01)) |  |
| train(DepVar ~ IndepVar + IndepVar, data=df, method="rpart", trControl=numFolds, tuneGrid=cpGrid) |  |
| rpart(DepVar ~ IndepVar + IndepVar, data=df, method="class", cp=0.18) | Result of train…. |

Text processing

|  |  |
| --- | --- |
| read.csv(…., stringsAsFactors=FALSE) |  |
| corpus = Corpus(VectorSource(tweets$Tweet)) | tm, SnowballC |
| corpus corpus[[1]]$content | inspect |
| corpus = tm\_map(corpus, content\_transformer(tolower)) |  |
| corpus = tm\_map(corpus, PlainTextDocument) |  |
| corpus = tm\_map(corpus, removePunctuation) |  |
| corpus = tm\_map(corpus, removeWords, c("apple", stopwords("english"))) |  |
| corpus = tm\_map(corpus, stemDocument) |  |
| stopwords("english")[1:10] |  |
|  |  |
| frequencies = DocumentTermMatrix(corpus) |  |
| frequencies |  |
| inspect(frequencies[1000:1005,505:515]) |  |
| findFreqTerms(frequencies, lowfreq=20) |  |
| sparse = removeSparseTerms(frequencies, 0.995) |  |
| Sparse |  |
| tweetsSparse = as.data.frame(as.matrix(sparse)) |  |
| colnames(tweetsSparse) = make.names(colnames(tweetsSparse)) |  |

Clustering

|  |  |
| --- | --- |
| distances = dist(df[2:20], method=”euclidean”) |  |
| cluster = hclust(distances, method=”ward.D”) |  |
| plot(cluster) |  |
| rect.hclust(cluster, k=3, border=”red”) |  |
| clusterGroups = cutree(cluster, k=10) |  |
| tapply(df$var, clusterGroups, mean) |  |
|  |  |
| kmc = kmeans(vector, centers=k, iter.max=1000) |  |
| str(kmc) |  |
|  |  |
| kcca = as.kcca(kmc, healthyVector) | flexclust |
| tumorClusters = predict(kcca, [newdata=tumorVector]) |  |
|  |  |
| preproc = preProcess(airlines)  airlinesNorm = predict(preproc, airlines) | caret – normalize data |

Attention points

|  |  |
| --- | --- |
| Null values – imputation [Assignment 3, 3: Predicting Loan Repayment] | mice |
| Multicolliniairity | cor(indep\_var, indep\_var) |
| Unordered Factors – replace by n-1 binary factors and a reference value (= most frequent value) [assignment 2, 2: Reading Test Scores] | relevel  letters$isB = as.factor(letters$letter == "B") |
| Not a linear relationship, try log(indep\_var) [assignment 2, 3: Detecting Flu Epidemics] |  |
| Time lag [assignment 2: 3, Detecting Flu Epidemics] | zoo, lag |
| AUC = The probability the model can correctly differentiate |  |
| Computing profitability of an investment [assignment 3, 3: Predicting Loan Repayment] |  |
| Cross Validation selecting cp [assignment 4, 3: Predicting earnings from] |  |
| Random Forest, variables used, impurity [assignment 4, 3: Predicting earnings from] | varUsed |
| Penalty matrix [Unit4, 2: Keeping an Eye…, Video 7: Basline model and Penalty Matrix] |  |
| Normalize data before clustering [assignment 6, 3: Predicting stock returns] | caret - preproc |

Other

|  |  |
| --- | --- |
| rm(list = ls()) |  |
| format(x, digits, nsmall, scientific, width, justify = c("left", "right", "centre", "none")) |  |
|  |  |
| cn <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=MANGO;database=Ruud;trusted\_connection=yes;") | RODBC |
|  |  |
| drv <- JDBC("org.apache.phoenix.jdbc.PhoenixDriver", "C:\\Users\\Ruud\\Documents\\Squirrel\\lib\\phoenix-4.4.0.2.4.1.0-327-client.jar", "'") |  |
| cn <- dbConnect(drv, "jdbc:phoenix:10.254.1.4:2181:/hbase-unsecure") |  |
|  |  |
| df <- sqlFetch(cn, 'stock', colnames=FALSE) |  |
| df <- ts(df, frequency=365, start=c(2007,36)) |  |
|  |  |