R Software & Packages

Problem: Creating Summary of Analysis for English Premier League Standings for Sports Client

Solution: Web scraping for data and soft coding as per the output required using Packages

## EPL

### Loading libraries to use

library(tidyverse)

library(lubridate)

library(dplyr)

library(readr)

EPL\_Standings <- function(date\_I, season\_I){

### Input of Date and Season will be put up as an argument in this function

date\_I <- as.Date(mdy(date\_I), format = '%m/%d/%y')

### Using this date\_I input in the from of character type will be transformed to date type

season\_I <- paste0(substr(season\_I, 3, 4), substr(season\_I, 6, 7), '/')

### Using this season\_I input will be read as yyyy to be logged for the URL Link

primary\_data <- paste0('https://www.football-data.co.uk/mmz4281/',season\_I,'E0.csv')

### Accurately reading the file from the webpage and exports it to primary\_data

Table1 <- read\_csv(primary\_data)

### Using readr to make the data frame on Table1 directly from the source rather than local drive

Table2 <- Table1 %>%

select(Date, HomeTeam, AwayTeam, FTHG, FTAG, FTR) %>%

### Extracting out main columns out of the Table1 into Table2 using dplyr with pipe operater

### Now to take care of 'YYYY' format and 'YY' format we mutate year from Date column

mutate(year = ifelse(nchar(Date) == 10, substring(Date, 9, 10),

ifelse(nchar(Date) == 8, substring(Date, 7, 8), 'Check Date Format')),

Date = paste0(substring(Date, 1, 6), year),

M\_Date = as.Date(Date, format = '%d/%m/%y')) %>%

### Finally convertng the dates from Table1 into date type using Lubridate into Date\_M

### Now filtering out Table1 to include only primary\_data played up to the given date\_I input

filter(M\_Date <= date\_I) %>%

mutate(home\_point = ifelse(FTR == 'D', 1,

ifelse(FTR == 'H', 3,

ifelse(FTR == 'A', 0, NA))),

# mutating to list points at home for the given set

away\_point = ifelse(FTR == 'D', 1,

ifelse(FTR == 'H', 0,

ifelse(FTR == 'A', 3, NA))),

# mutating to list point on away for the given set

home\_win = ifelse(FTR == 'H', 1, 0),

# command counts 1 and adds if there is a win at home

away\_win = ifelse(FTR == 'A', 1, 0),

# command counts 1 and adds if there is a win on away

home\_draw = ifelse(FTR == 'D', 1, 0),

# command counts 1 and adds if there is a draw at home

away\_draw = ifelse(FTR == 'D', 1, 0),

# command counts 1 and adds if there is a draw on away

home\_loss = ifelse(FTR == 'A', 1, 0),

# command counts 1 and adds if there is a loss at home

away\_loss = ifelse(FTR == 'H', 1, 0))

# command counts 1 and adds if there is a loss on away

### By selection we make a data frame for creating the away set

set\_away <- Table2 %>%

select(Date, M\_Date, AwayTeam, FTHG, FTAG, FTR, away\_point, away\_win, away\_draw, away\_loss) %>%

group\_by(TeamName = AwayTeam) %>%

### Group using TeamName to summarize all summations we need

summarise(away\_count = n(),

# counting the total number of games played while away

away\_point = sum(away\_point),

# summing the total number of points made while away

away\_win = sum(away\_win),

# summing the total number of wins made while away

away\_draw = sum(away\_draw),

# summing the total number of draws made while away

away\_loss = sum(away\_loss),

# summing the total number of loss made while away

away\_goals\_for = sum(FTAG),

# summing the total number of goals made for while away

away\_goals\_against = sum(FTHG))

# summing the total number of goals made against while away

### By selection we make a data frame for creating the home set

set\_home <- Table2 %>%

select(Date, M\_Date, HomeTeam, FTHG, FTAG, FTR, home\_point, home\_win, home\_draw, home\_loss) %>%

group\_by(TeamName = HomeTeam) %>%

### Group using TeamName to summarize all summations we need

summarise(home\_count = n(),

# counting the total number of games played at home

home\_point = sum(home\_point),

# summing the total number of points made at home

home\_win = sum(home\_win),

# summing the total number of wins made at home

home\_draw = sum(home\_draw),

# summing the total number of draws made at home

home\_loss = sum(home\_loss),

# summing the total number of loss made at home

home\_goals\_for = sum(FTHG),

# summing the total number of goals made for at home

home\_goals\_against = sum(FTAG))

# summing the total number of goals made against at home

Table3 <- set\_home %>%

full\_join(set\_away, by = c('TeamName'))

### Using full\_join we combine sets created for home and away

Table3[is.na(Table3)] <- 0

### Catering to the NA possibilities as well by translatin it to 0

### Lastly we use mutation through dplyr to show all data for the requirements asked

Table5 <- Table3 %>%

mutate(MatchesPlayed = away\_count + home\_count,

# Adds the total number of matches played both home and away for the Team

Points = away\_point + home\_point,

# Adds the total number of points made both home and away for the Team

PPM = Points/MatchesPlayed,

# Calculates the total points made per total match played for the Team

PtPct = Points/(3\*MatchesPlayed),

# Calculates the total pointed made per total points possible for the Team

Wins = away\_win + home\_win,

# Adds the total wins both home and away for the Team

Draws = away\_draw + home\_draw,

# Adds the total draws both home and away for the Team

Loss = away\_loss + home\_loss,

# Adds the total losses both home and away for the Team

Record = paste0(Wins,'-',Loss,'-',Draws),

# Creates record of Wins-Losses-Draws for the Team

HomeRec = paste0(home\_win,'-',home\_loss,'-',home\_draw),

# Creates record at home of Wins-Losses-Draws for the Team

AwayRec = paste0(away\_win,'-',away\_loss,'-',away\_draw),

# Creates record while away of Wins-Losses-Draws for the Team

GS = home\_goals\_for + away\_goals\_for,

# Adds the total goals scored for both home and away for the Team

GSM = GS/MatchesPlayed,

# Calculates total goals scored for per total matches played

GA = home\_goals\_against + away\_goals\_against,

# Adds the total goals scored against both home and away for the Team

GAM = GA/MatchesPlayed)

# Calculates total goals scored against per total matches played

############################## Additional Problem

away\_last10 <- Table2 %>%

select(M\_Date, TeamName = AwayTeam, winA = away\_win, drawA = away\_draw, lossA = away\_loss)

### Creating another set for last 10 away data

home\_last10 <- Table2 %>%

select(M\_Date, TeamName = HomeTeam, winA = home\_win, drawA = home\_draw, lossA = home\_loss)

### Creating another set for last 10 home data

Table4 <- rbind(home\_last10, away\_last10)

### Using rbind we create a new data frame

### From the Table4 we get the top 10 games

Table5 <- Table4 %>%

group\_by(TeamName) %>%

top\_n(10, wt = M\_Date) %>%

# Please note the date is arranged in descending with arrange(desc(M\_Date)) %>%

summarise(Wins = sum(winA),

Loss = sum(lossA),

Draws = sum(drawA)) %>%

mutate(Last10 = paste0(Wins,'-',Loss,'-',Draws)) %>%

select(TeamName, Last10) %>%

# Now we summarize the Last 10 games wins losses and draws to mutate it into the Last10 field

inner\_join(Table5, by = c('TeamName')) %>%

# Then we join the Last10 selection with Table5 data frame we created

arrange(TeamName)

# Lastly we arrange Team Names in alphabetical order for the program to compute

Table6 <- Table5 %>%

arrange(desc(PPM), desc(Wins), desc(GSM), GAM) %>%

select(TeamName, Record, HomeRec, AwayRec, MatchesPlayed, Points, PPM, PtPct, GS, GSM, GA, GAM, Last10)

### Finally we feed everything is Table6 arranging in descending order from data frame Table5

return(Table6)}

### Our program is now complete to give the Output for the date and season input provided

############################## Trying for: "12/10/2020","2020/21"

output <- EPL\_Standings("12/10/2020","2020/21")

Problem: Analyzing independent variables for cause & effect on dependent for Research Client

Solution: Applying Linear Regression Model using Statistical Tests and varying factors and interactions

#Data shows values for 16 Whales in a study

# Dependent Variable (Y) being Weight [Kg] determined by

# Independent Variables (X) being Sleep [Daily Hours], FoodI [Intake Kg], BMI [Mass Ratio], FoodN [Nutrition Kg] and Type [Categorical]

# Read in MWhale.csv file

library(readr)

library(tidyverse)

library(ggpubr)

library(caret)

library(car)

theme\_set(theme\_pubr())

data = read.csv(file=file.choose(),header=TRUE)

##Problem 1: Build the Best Model

### Check Individual Models (SIMPLE LINEAR REGRESSION)

attach(data)

cor(Weight, Sleep)

cor(Weight, FoodI)

cor(Weight, BMI)

cor(Weight, FoodN)

cor(Weight, Type)

#Notice we can not get a direct correlation for Type as Type is not Numeric

#Notice strong Correlation FoodI & FoodN and relatively weaker for Sleep & BMI

modelSleep <- lm(Weight ~ Sleep, data=data)

modelFoodI <- lm(Weight ~ FoodI, data=data)

modelBMI <- lm(Weight ~ BMI, data=data)

modelFoodN <- lm(Weight ~ FoodN, data=data)

modelType <- lm(Weight ~ Type, data=data)

summary(modelSleep)

#F: 3.699 ~ Pv = 0.07502 i.e > 0.05 : Bad Model - Explains 0.209 (R^2) of Variation

#t: 1.923 ~ Pv = 0.075 i.e > 0.05 : Insignificant Relaition - b1 = 34.88 & b0 = 203.64

summary(modelFoodI)

#F: 120.9 ~ Pv = 2.85e-08 i.e < 0.05 : Good Model - Explains 0.8962 (R^2) of Variation

#t: 10.995 ~ Pv = 2.85e-08 i.e < 0.05 : Significant Relaition - b1 = 1.02330 & b0 = 138.42682

summary(modelBMI)

#F: 1.274 ~ Pv = 0.278 i.e > 0.05 : Bad Model - Explains 0.08341 (R^2) of Variation

#t: 1.129 ~ Pv = 0.27798 i.e > 0.05 : Insignificant Relaition - b1 = 28.38 & b0 = 363.03

summary(modelFoodN)

#F: 49.14 ~ Pv = 6.148e-06 i.e < 0.05 : Good Model - Explains 0.7783 (R^2) of Variation

#t: 7.01 ~ Pv = 6.15e-06 i.e < 0.05 : Significant Relaition - b1 = 1.4857 & b0 = 277.1877

### Define the Categories (NOMINAL: LINEAR REGRESSION)

table(Type)

#Notice 3 Types of Whales in this data

matrix <- model.matrix(~Type, data = data)

head(matrix[, -1])

#Notice that TypeBlue is Reference

summary(modelType)

#F: 20.91 ~ Pv = 8.665e-05 i.e < 0.05 : Good Model - Explains 0.7628 (R^2) of Variation

#b0: 542.26 = AvgWeight of Blue, b0 + b1: 526.42 = AvgWeight of Killer, b0 + b2: 386.78 = AvgWeight of Orca

### Build the Model (MULTIPLE LINEAR REGRESSION)

model0 <- lm(Weight ~ FoodI + FoodN, data = data)

summary(model0)

# Good Model which Explains 89.64% with b2 for FoodN being Insignificant

model1 <- lm(Weight ~ FoodI + FoodN + Sleep + Type + BMI, data = data)

summary(model1)

# Good Model which Explains 98.25% but with all the variables hence we run Anova

Anova(model1)

#Insignificant B's show us what can be ignored like FoodI & Sleep

model2 <- lm(Weight ~ FoodN + Type + BMI, data = data)

summary(model2)

# Good Model which Explains 97.38% but several one Insignificant b, Hence we check for Variation in Residuals

res2 <- resid(model2)

summary(res2)

plot(fitted(model2), res2, ylab="Residuals", xlab="Weights")

abline(0,0)

# Since the plot is not ideal we lastly assess any relation within the Variables

### Test for Explanation (INTERACTION: LINEAR REGRESSION)

model3 <- lm(Weight ~ BMI + FoodN + Type + BMI:FoodN + BMI:Type + FoodN:Type, data = data)

summary(model3)

Anova(model3)

# Good Model which Explains 99.56% Variation BUT actually with all Interaction b's Insignificant due to maybe redundancy

# Notice that in our best Multiple Regression Model 2 (Line 70) Type is causing issue: Hence we check for its Interactions only

model4 <- lm(Weight ~ BMI + FoodN + Type + BMI:Type + FoodN:Type , data = data)

summary(model4)

Anova(model4)

# Good Model which Explains 99.43% Variation AND actually has all b's all relatively significant now!

anova(model3,model4)

# Confirming through the Anova test we can see that there is no significant difference between the to as

# Pv = 0.2315 i.e > 0.05. Hence we do not reject H0

# So we will now choose Model 4 as BEST which has the lesser number of Variables.

##Problem 2: Predict Weight of Orca whose Food Intake is 150 and Nutrition in that food is 80

# Interval for a single run

predval <- data.frame(BMI=3.75, FoodN=80, Type='Orca')

predict(model4, newdata=predval, interval="prediction", level=0.95)

# Interval for the average of a series of runs

predict(model4, newdata=predval, interval="confidence", level=0.95)

Problem: Predicting Delay in Flights for Airline Client

Solution: Supervised Regression with Subset Selection and testing with least MSE.

library(caret)

library(tidyverse)

install.packages("dplyr")

#step0 Perform EDA

flight\_data = read.csv(file=file.choose(),header=TRUE)

library(skimr)

summaryStats <- skim(flight\_data)

summaryStats

boxplot(flight\_data$Arr\_Delay~flight\_data$Carrier, ylab="Arrival Delay")

### Step 1: Partition our Data

#We first change categorical variables into dummy variables using model.matrix (we will also use one-hot encoding in the future for non-linear models). The function model.matrix() can automatically convert categorical predictor variables to dummy variables. It also creates a column of 1s, which we don’t need at this time. That column of 1 is used for estimating intercept if you write algorithm by yourself, but most available functions automatically creates that column during estimation.

flight\_predictors\_dummy <- model.matrix(Arr\_Delay~ ., data = flight\_data)#create dummy variables expect for the response

flight\_predictors\_dummy<- data.frame(flight\_predictors\_dummy[,-1]) #get rid of intercept and make data frame

flight\_data <- cbind(Arr\_Delay=flight\_data$Arr\_Delay, flight\_predictors\_dummy)

#We randomly split the data into training (80%) and testing (20%) datasets:

set.seed(99) #set random seed

index <- createDataPartition(flight\_data$Arr\_Delay, p = .8,list = FALSE)

flight\_train <- flight\_data[index,]

flight\_test <- flight\_data[-index,]

###Step 2: Train or Fit Model

#We train the model using the train function. In this train function

#In the first argument, provide a formula with the response variable ~ predictor variables or use response variable ~ . to include all predictor variables

#In the data argument, provide the training data set

#In the method argument, provide the method or machine learning model to use We must also load the relevant libraries for this machine learning model before running the train function. For the linear regression we use method = “lm”

#In the trControl argument, this “none” means fit one model to the entire training set. We will discuss more advance validation approaches later.

library(MASS)

subset\_model <- train(Arr\_Delay ~ .,

data = flight\_train,

method = "glmStepAIC",

direction = "backward",

trControl =trainControl(method = "none"))

#Get results

coef(subset\_model$finalModel)

#Step 3: Get Predictions using Testing Set Data

#For regression problems, we want to get the predicted median house price for all observations in the test set.

subset\_pred<-predict(subset\_model, flight\_test)

#Step 4: Evaluate Model Performance

#For regression problems, we evaluate performance of models using the mean squared error of the test set.

#Calculate the mean squared error for this model.

MSE<-mean((subset\_pred- flight\_test$Arr\_Delay)^2)

MSE

Problem: Predicting whether Customer Will Default or Not on Loan for Bank Client

Solution: Supervised Classification with Logistic Regression using Lasso for testing with most AUC.

#install.packages("caret") #this may take a while

library(caret)

library(tidyverse)

loan\_data <- read.csv(file = "loan\_default\_dataset.csv", header=T)

loan\_data = read.csv(file=file.choose(),header=TRUE)

table(loan\_data$Default)

# we have 1000 observations and 700 are non-default and 300 are default

#boxplot(loan\_data$~ loan\_data$ )

#install.packages("skimr")

library(skimr)

summaryStats <- skim(loan\_data)

summaryStats

boxplot(loan\_data$Term, ylab= 'Term')

boxplot(loan\_data$Checking\_amount, ylab='Checking\_amount' )

boxplot(loan\_data$Age, ylab= 'Age')

boxplot(loan\_data$Credit\_score, ylab= 'Credit\_score')

boxplot(loan\_data$Amount, ylab= 'Amount')

boxplot(loan\_data$Saving\_amount, ylab= 'Saving\_amount')

boxplot(loan\_data$No\_of\_credit\_acc, ylab= '# Credit Account')

boxplot(loan\_data$Emp\_duration, ylab= 'Emp\_duration')

boxplot(loan\_data$No\_of\_credit\_acc~loan\_data$Default)

boxplot(loan\_data$Amount ~loan\_data$Default)

boxplot(loan\_data$Amount ~loan\_data$Emp\_status)

#Step 1 partition and preprocessing

library("tidyverse")

dummies\_model<-dummyVars(Default~.,data=loan\_data)

Default\_predictors\_dummy<-data.frame(predict(dummies\_model,newdata=loan\_data))

loan\_data<-cbind(Default=loan\_data$Default,Default\_predictors\_dummy)

####Logistic Regression####

####Logistic Regression####

loan\_data$Default<-as.factor(loan\_data$Default)

loan\_data$Default<-fct\_recode(loan\_data$Default, notDefault = "0", Default = "1")

loan\_predictors\_dummy <- model.matrix(Default~ ., data = loan\_data)#create dummy variables expect for the response

loan\_predictors\_dummy<- data.frame(loan\_predictors\_dummy[,-1]) #get rid of intercept

loan\_data <- cbind(Default=loan\_data$Default, loan\_predictors\_dummy)

library(caret)

set.seed(99) #set random seed

index <- createDataPartition(loan\_data$Default, p = .8,list = FALSE)

loan\_train <-loan\_data[index,]

loan\_test <- loan\_data[-index,]

##Step 2: Train or Fit LASSO Logistic Regression Model

# install and load packages for machine learning model

library(e1071)

library(glmnet)

library(Matrix)

set.seed(10)#set the seed again since within the train method the validation set is randomly selected

loan\_model <- train(Default ~ .,

data = loan\_train,

method = "glmnet",

standardize =T,

tuneGrid = expand.grid(alpha =1, #lasso

lambda = seq(0.0001, 1, length = 20)),

trControl =trainControl(method = "cv",

number = 5,

classProbs = TRUE,

summaryFunction = twoClassSummary),

metric="ROC")

loan\_model

#list coefficients selected

coef(loan\_model$finalModel, loan\_model$bestTune$lambda)

##Step 3: Get Predictions using Testing Set Data

#First, get the predicted probabilities of the test data.

predprob\_lasso<-predict(loan\_model , loan\_test, type="prob")

#Step 4: Evaluate Model Performance

#install.packages("ROCR")

library(ROCR)

pred\_lasso <- prediction(predprob\_lasso$Default, loan\_test$Default,label.ordering =c("notDefault","Default") )

perf\_lasso <- performance(pred\_lasso, "tpr", "fpr")

plot(perf\_lasso, colorize=TRUE)

#Get the AUC

auc\_lasso<-unlist(slot(performance(pred\_lasso, "auc"), "y.values"))

auc\_lasso

Problem: Predicting whether Customer Will Churn or Not for Client Assessment

Solution: Supervised Classification with Decision Tree using K-fold Cross Validation for testing with most AUC.

loan\_data$Default<-as.factor(loan\_data$Default)

levels(loan\_data$Default)

loan\_data$Default<-fct\_recode(loan\_data$Default, notDefault = "0", Default = "1")

#modeling

set.seed(99) #set random seed

index <- createDataPartition(loan\_data$Default, p = .8,list = FALSE)

Default\_train <-loan\_data[index,]

Default\_test <- loan\_data[-index,]

#train model

library(rpart)

set.seed(12)

Default\_model <- train(Default ~ .,

data = Default\_train,

method = "rpart",

trControl =trainControl(method = "cv",number = 5,

## Estimate class probabilities

classProbs = TRUE,

#needed to get ROC

summaryFunction = twoClassSummary),

metric="ROC")

Default\_model

plot(Default\_model)

plot(varImp(Default\_model))

library(rpart.plot)

rpart.plot(Default\_model$finalModel,type=5)

#get predictions

predprob\_Default<-predict(Default\_model,Default\_test,type="prob")

#create ROC AUC

install.packages("ROCR")

library(ROCR)

pred\_tree<-prediction(predprob\_Default[,2],Default\_test$Default,

label.ordering = c("notDefault","Default"))

perf\_tree <- performance(pred\_tree, "tpr", "fpr")

plot(perf\_tree, colorize=TRUE)

auc\_tree<-unlist(slot(performance(pred\_tree, "auc"), "y.values"))

auc\_tree

Problem: Predicting Customer Response based on Demographics & Buying for Food Delivery Client

Solution: Supervised Classification with XgBoost using Feature Engineering for testing with most AUC and Analysing with SHAP plots.

library(caret)

library(lattice)

library(ggplot2)

library(tidyverse)

library(dplyr)

library(forcats)

fasteats = read.csv(file=file.choose(),header=TRUE)

str(fasteats)

library(skimr)

skim(fasteats)

summary(fasteats)

sum(is.na(fasteats$Income))

fasteats <- fasteats %>%

mutate(Year\_Birth = ifelse(Year\_Birth < 1940, median(fasteats$Year\_Birth), Year\_Birth))

library(lubridate)

fasteats$Dt\_Customer <- as.Date(fasteats$Dt\_Customer, format = "%m/%d/%y")

max\_length <- max(nchar(fasteats$ID))

fasteats$ID <- as.numeric(sprintf("%0\*d", max\_length, as.integer(fasteats$ID)))

# Removing the ID col

fasteats <- select(fasteats, -ID)

fasteats$Marital\_Status <- factor(fasteats$Marital\_Status)

fasteats$Marital\_Status <- recode(fasteats$Marital\_Status, "Alone" = "Single")

fasteats$Marital\_Status <- fct\_lump(fasteats$Marital\_Status, n = 5, other\_level = "Other")

#fasteats$Response<-as.factor(fasteats$Response)

#fasteats$Response<-fct\_recode(fasteats$Response, No = "0", Yes = "1")

fasteats$Education <- as.factor(fasteats$Education)

levels(fasteats$Education)

preProcess\_missingdata\_model <- preProcess(fasteats, method='medianImpute')

preProcess\_missingdata\_model

fasteats <- predict(preProcess\_missingdata\_model,newdata=fasteats)

skim(fasteats)

fasteats\_predictors\_dummy <- model.matrix(Response~ ., data = fasteats)

fasteats\_predictors\_dummy <- data.frame(fasteats\_predictors\_dummy[,-1])

fasteats <- cbind(Response = fasteats$Response, fasteats\_predictors\_dummy)

fasteats$Response <- as.factor(fasteats$Response)

fasteats$Response <- fct\_recode(fasteats$Response, noresponse = "0",response = "1")

fasteats$Response<- relevel(fasteats$Response, ref = "response")#this makes the first level "response"

set.seed(99)

index <- createDataPartition(fasteats$Response, p = .8,list = FALSE)

fasteats\_train <- fasteats[index,]

fasteats\_test <- fasteats[-index,]

##if we don't replace my missing values before converting them to dummy values I am missing some rows.

##preProcess\_missingdata\_model <- preProcess(fasteats, method='medianImpute')

##preProcess\_missingdata\_model

##fasteats\_train <-predict(preProcess\_missingdata\_model,newdata=fasteats\_train)

##fasteats\_test <-predict(preProcess\_missingdata\_model,newdata=fasteats\_test)

set.seed(10)

model\_fasteats<-train(Response~.,

data= fasteats\_train,

method="xgbTree",

tuneGrid= expand.grid(nrounds = c(50,200),

eta = c(0.025, 0.05, 0.1),

max\_depth = c(2, 3, 4, 5),

gamma = 0,

colsample\_bytree = 1,

min\_child\_weight = 1,

subsample = 1),

trControl=trainControl(method = "cv",

number = 5)

)

model\_fasteats

model\_fasteats$bestTune

plot(varImp(model\_fasteats))

fasteats\_predictions<- predict(model\_fasteats, fasteats\_test, type = "prob")

library(ROCR)

pred\_tree<-prediction(fasteats\_predictions$response,fasteats\_test$Response,

label.ordering = c("noresponse","response"))

perf\_tree <- performance(pred\_tree, "tpr", "fpr")

plot(perf\_tree, colorize=TRUE)

auc\_tree<-unlist(slot(performance(pred\_tree, "auc"), "y.values"))

auc\_tree

install.packages("SHAPforxgboost")

library(SHAPforxgboost)

Xdata<-as.matrix(select(fasteats\_train,-c(Response))) # change data to matrix for plots

# Crunch SHAP values

shap <- shap.prep(model\_fasteats$finalModel, X\_train = Xdata)

# SHAP importance plot

shap.plot.summary(shap)