# Project Phase 2

## Team Rafalowski-Rabil

### Model Generation

#### Load Libraries

### Setup\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Load and Clean sharkstudent data set

#Read in Data set  
sharkstudent <- read\_csv("shark\_student.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

#Create New Season Variable  
sharkstudent<- mutate(sharkstudent,"Season" = str\_sub(sharkstudent$SeasonEpisode,12,14))  
  
# Renames columns without spaces and slash  
colnames(sharkstudent)[colnames(sharkstudent) == "Health / Wellness"] <- "HealthWellness"  
colnames(sharkstudent)[colnames(sharkstudent) == "Lifestyle / Home"] <- "LifestyleHome"   
colnames(sharkstudent)[colnames(sharkstudent) == "Software / Tech"] <- "SoftwareTech"   
colnames(sharkstudent)[colnames(sharkstudent) == "Children / Education"] <- "ChildrenEducation"   
colnames(sharkstudent)[colnames(sharkstudent) == "Fashion / Beauty"] <- "FashionBeauty"  
colnames(sharkstudent)[colnames(sharkstudent) == "Media / Entertainment"] <- "MediaEntertainment"  
colnames(sharkstudent)[colnames(sharkstudent) == "Fitness / Sports / Outdoors"] <- "FitnessSportsOutdoors"  
colnames(sharkstudent)[colnames(sharkstudent) == "Green/CleanTech"] <- "GreenCleanTech"  
colnames(sharkstudent)[colnames(sharkstudent) == "Uncertain / Other"] <- "UncertainOther"  
colnames(sharkstudent)[colnames(sharkstudent) == "Food and Beverage"] <- "FoodBeverage"  
colnames(sharkstudent)[colnames(sharkstudent) == "Business Services"] <- "BusinessServices"  
colnames(sharkstudent)[colnames(sharkstudent) == "Pet Products"] <- "PetProducts"  
  
#Convert all character variables to Factor  
sharkstudent = sharkstudent %>% mutate\_if(is.character,as\_factor)  
  
#Concert to factor and recode categories  
sharkstudent <- sharkstudent %>%   
 mutate(ReceiveOffer = as\_factor(ReceiveOffer)) %>%  
 mutate(ReceiveOffer = fct\_recode(ReceiveOffer, "Yes" = "1", "No" = "0")) %>%  
 mutate(RejectOffer = as\_factor(RejectOffer)) %>%  
 mutate(RejectOffer = fct\_recode(RejectOffer, "Yes" = "1", "No" = "0")) %>%  
 mutate(Deal\_Yes = as\_factor(Deal\_Yes)) %>%  
 mutate(Deal\_Yes = fct\_recode(Deal\_Yes, "Yes" = "1", "No" = "0")) %>%  
 mutate(Deal\_No = as\_factor(Deal\_No)) %>%  
 mutate(Deal\_No = fct\_recode(Deal\_No, "Yes" = "1", "No" = "0")) %>%  
 mutate(Eth1 = as\_factor(Eth1)) %>%  
 mutate(Eth1 = fct\_recode(Eth1, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 1" = "0")) %>%  
 mutate(Eth2 = as\_factor(Eth2)) %>%  
 mutate(Eth2 = fct\_recode(Eth2, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 2" = "0")) %>%  
 mutate(Eth3 = as\_factor(Eth3)) %>%  
 mutate(Eth3 = fct\_recode(Eth3, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 3" = "0")) %>%  
 mutate(Eth4 = as\_factor(Eth4)) %>%  
 mutate(Eth4 = fct\_recode(Eth4, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 4" = "0")) %>%  
 mutate(Eth5 = as\_factor(Eth5)) %>%  
 mutate(Eth5 = fct\_recode(Eth5, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 5" = "0")) %>%  
 mutate(Male1 = as\_factor(Male1)) %>%  
 mutate(Male1 = fct\_recode(Male1, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male2 = as\_factor(Male2)) %>%   
 mutate(Male2 = fct\_recode(Male2, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male3 = as\_factor(Male3)) %>%  
 mutate(Male3 = fct\_recode(Male3, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male4 = as\_factor(Male4)) %>%  
 mutate(Male4 = fct\_recode(Male4, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female1 = as\_factor(Female1)) %>%  
 mutate(Female1 = fct\_recode(Female1, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female2 = as\_factor(Female2)) %>%  
 mutate(Female2 = fct\_recode(Female2, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female3 = as\_factor(Female3)) %>%  
 mutate(Female3 = fct\_recode(Female3, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female4 = as\_factor(Female4)) %>%  
 mutate(Female4 = fct\_recode(Female4, "Yes" = "1", "No" = "0")) %>%  
 mutate(Novelties = as\_factor(Novelties)) %>%   
 mutate(Novelties = fct\_recode(Novelties, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`HealthWellness` = as\_factor(`HealthWellness`)) %>%   
 mutate(`HealthWellness` = fct\_recode(`HealthWellness`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FoodBeverage` = as\_factor(`FoodBeverage`)) %>%   
 mutate(`FoodBeverage` = fct\_recode(`FoodBeverage`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`BusinessServices` = as\_factor(`BusinessServices`)) %>%   
 mutate(`BusinessServices` = fct\_recode(`BusinessServices`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`LifestyleHome` = as\_factor(`LifestyleHome`)) %>%   
 mutate(`LifestyleHome` = fct\_recode(`LifestyleHome`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`SoftwareTech` = as\_factor(`SoftwareTech`)) %>%   
 mutate(`SoftwareTech` = fct\_recode(`SoftwareTech`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`ChildrenEducation` = as\_factor(`ChildrenEducation`)) %>%   
 mutate(`ChildrenEducation` = fct\_recode(`ChildrenEducation`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Automotive = as\_factor(Automotive)) %>%   
 mutate(Automotive = fct\_recode(Automotive, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FashionBeauty` = as\_factor(`FashionBeauty`)) %>%   
 mutate(`FashionBeauty` = fct\_recode(`FashionBeauty`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`MediaEntertainment` = as\_factor(`MediaEntertainment`)) %>%   
 mutate(`MediaEntertainment` = fct\_recode(`MediaEntertainment`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FitnessSportsOutdoors` = as\_factor(`FitnessSportsOutdoors`)) %>%   
 mutate(`FitnessSportsOutdoors` = fct\_recode(`FitnessSportsOutdoors`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`PetProducts` = as\_factor(`PetProducts`)) %>%   
 mutate(`PetProducts` = fct\_recode(`PetProducts`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Travel = as\_factor(Travel)) %>%   
 mutate(Travel = fct\_recode(Travel, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`GreenCleanTech` = as\_factor(`GreenCleanTech`)) %>%   
 mutate(`GreenCleanTech` = fct\_recode(`GreenCleanTech`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`UncertainOther` = as\_factor(`UncertainOther`)) %>%   
 mutate(`UncertainOther` = fct\_recode(`UncertainOther`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MalePresenter = as\_factor(MalePresenter)) %>%   
 mutate(MalePresenter = fct\_recode(MalePresenter, "Yes" = "1", "No" = "0" )) %>%  
 mutate(FemalePresenter = as\_factor(FemalePresenter)) %>%   
 mutate(FemalePresenter = fct\_recode(FemalePresenter, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MixedGenderPresenters = as\_factor(MixedGenderPresenters)) %>%   
 mutate(MixedGenderPresenters = fct\_recode(MixedGenderPresenters, "Yes" = "1", "No" = "0" )) %>%  
 mutate(CompanyState = as\_factor(CompanyState)) %>%   
 mutate(CompanyState = fct\_recode(CompanyState, "Yes" = "1", "No" = "0" )) %>%  
 mutate(BarbaraCorcoran = as\_factor(BarbaraCorcoran)) %>%   
 mutate(BarbaraCorcoran = fct\_recode(BarbaraCorcoran, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MarkCuban = as\_factor(MarkCuban)) %>%   
 mutate(MarkCuban = fct\_recode(MarkCuban, "Yes" = "1", "No" = "0" )) %>%  
 mutate(LoriGreiner = as\_factor(LoriGreiner)) %>%   
 mutate(LoriGreiner = fct\_recode(LoriGreiner, "Yes" = "1", "No" = "0" )) %>%  
 mutate(RobertHerjavec = as\_factor(RobertHerjavec)) %>%   
 mutate(RobertHerjavec = fct\_recode(RobertHerjavec, "Yes" = "1", "No" = "0" )) %>%  
 mutate(DaymondJohn = as\_factor(DaymondJohn)) %>%   
 mutate(DaymondJohn = fct\_recode(DaymondJohn, "Yes" = "1", "No" = "0" )) %>%  
 mutate(KevinOLeary = as\_factor(KevinOLeary)) %>%   
 mutate(KevinOLeary = fct\_recode(KevinOLeary, "Yes" = "1", "No" = "0" )) %>%  
 mutate(KevinHarrington = as\_factor(KevinHarrington)) %>%   
 mutate(KevinHarrington = fct\_recode(KevinHarrington, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Guest = as\_factor(Guest)) %>%   
 mutate(Guest = fct\_recode(Guest, "Yes" = "1", "No" = "0" ))

## Warning: Unknown levels in `f`: 1, 3, 4

## Warning: Unknown levels in `f`: 1, 3, 4

## Warning: Unknown levels in `f`: 1

## Warning: Unknown levels in `f`: 1, 0

# Remove weak or non-required variables  
sharkstudent = sharkstudent %>% select(-CompanyState)   
sharkstudent = sharkstudent %>% select(-Deal\_No)   
sharkstudent = sharkstudent %>% select(-ReceiveOffer)  
sharkstudent = sharkstudent %>% select(-RejectOffer)  
sharkstudent = sharkstudent %>% select(-Male4)  
# sharkstudent = sharkstudent %>% select(-Female4)  
# sharkstudent = sharkstudent %>% select(-Eth4)  
# sharkstudent = sharkstudent %>% select(-Eth5)  
sharkstudent = sharkstudent %>% select(-SeasonEpisode)  
sharkstudent = sharkstudent %>% select(-Company)  
sharkstudent = sharkstudent %>% select(-X1)

#### Load and Clean sharkcompetition data set

#Read in Data set  
sharkcompetition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

#Create New Season Variable  
sharkcompetition<- mutate(sharkcompetition,"Season" = str\_sub(sharkcompetition$SeasonEpisode,12,14))  
  
# Renames columns without spaces and slash  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Health / Wellness"] <- "HealthWellness"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Lifestyle / Home"] <- "LifestyleHome"   
colnames(sharkcompetition)[colnames(sharkcompetition) == "Software / Tech"] <- "SoftwareTech"   
colnames(sharkcompetition)[colnames(sharkcompetition) == "Children / Education"] <- "ChildrenEducation"   
colnames(sharkcompetition)[colnames(sharkcompetition) == "Fashion / Beauty"] <- "FashionBeauty"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Media / Entertainment"] <- "MediaEntertainment"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Fitness / Sports / Outdoors"] <- "FitnessSportsOutdoors"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Green/CleanTech"] <- "GreenCleanTech"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Uncertain / Other"] <- "UncertainOther"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Food and Beverage"] <- "FoodBeverage"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Business Services"] <- "BusinessServices"  
colnames(sharkcompetition)[colnames(sharkcompetition) == "Pet Products"] <- "PetProducts"  
  
#Convert all character variables to Factor  
sharkcompetition = sharkcompetition %>% mutate\_if(is.character,as\_factor)  
  
#Concert to factor and recode categories  
sharkcompetition <- sharkcompetition %>%   
 mutate(Eth1 = as\_factor(Eth1)) %>%  
 mutate(Eth1 = fct\_recode(Eth1, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 1" = "0")) %>%  
 mutate(Eth2 = as\_factor(Eth2)) %>%  
 mutate(Eth2 = fct\_recode(Eth2, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 2" = "0")) %>%  
 mutate(Eth3 = as\_factor(Eth3)) %>%  
 mutate(Eth3 = fct\_recode(Eth3, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 3" = "0")) %>%  
 mutate(Eth4 = as\_factor(Eth4)) %>%  
 mutate(Eth4 = fct\_recode(Eth4, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 4" = "0")) %>%  
 mutate(Eth5 = as\_factor(Eth5)) %>%  
 mutate(Eth5 = fct\_recode(Eth5, "African American" = "1", "White" = "2", "Asian" = "3", "Latino" = "4", "No presenter 5" = "0")) %>%  
 mutate(Male1 = as\_factor(Male1)) %>%  
 mutate(Male1 = fct\_recode(Male1, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male2 = as\_factor(Male2)) %>%   
 mutate(Male2 = fct\_recode(Male2, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male3 = as\_factor(Male3)) %>%  
 mutate(Male3 = fct\_recode(Male3, "Yes" = "1", "No" = "0")) %>%  
 mutate(Male4 = as\_factor(Male4)) %>%  
 mutate(Male4 = fct\_recode(Male4, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female1 = as\_factor(Female1)) %>%  
 mutate(Female1 = fct\_recode(Female1, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female2 = as\_factor(Female2)) %>%  
 mutate(Female2 = fct\_recode(Female2, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female3 = as\_factor(Female3)) %>%  
 mutate(Female3 = fct\_recode(Female3, "Yes" = "1", "No" = "0")) %>%  
 mutate(Female4 = as\_factor(Female4)) %>%  
 mutate(Female4 = fct\_recode(Female4, "Yes" = "1", "No" = "0")) %>%  
 mutate(Novelties = as\_factor(Novelties)) %>%   
 mutate(Novelties = fct\_recode(Novelties, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`HealthWellness` = as\_factor(`HealthWellness`)) %>%   
 mutate(`HealthWellness` = fct\_recode(`HealthWellness`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FoodBeverage` = as\_factor(`FoodBeverage`)) %>%   
 mutate(`FoodBeverage` = fct\_recode(`FoodBeverage`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`BusinessServices` = as\_factor(`BusinessServices`)) %>%   
 mutate(`BusinessServices` = fct\_recode(`BusinessServices`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`LifestyleHome` = as\_factor(`LifestyleHome`)) %>%   
 mutate(`LifestyleHome` = fct\_recode(`LifestyleHome`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`SoftwareTech` = as\_factor(`SoftwareTech`)) %>%   
 mutate(`SoftwareTech` = fct\_recode(`SoftwareTech`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`ChildrenEducation` = as\_factor(`ChildrenEducation`)) %>%   
 mutate(`ChildrenEducation` = fct\_recode(`ChildrenEducation`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Automotive = as\_factor(Automotive)) %>%   
 mutate(Automotive = fct\_recode(Automotive, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FashionBeauty` = as\_factor(`FashionBeauty`)) %>%   
 mutate(`FashionBeauty` = fct\_recode(`FashionBeauty`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`MediaEntertainment` = as\_factor(`MediaEntertainment`)) %>%   
 mutate(`MediaEntertainment` = fct\_recode(`MediaEntertainment`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`FitnessSportsOutdoors` = as\_factor(`FitnessSportsOutdoors`)) %>%   
 mutate(`FitnessSportsOutdoors` = fct\_recode(`FitnessSportsOutdoors`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`PetProducts` = as\_factor(`PetProducts`)) %>%   
 mutate(`PetProducts` = fct\_recode(`PetProducts`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Travel = as\_factor(Travel)) %>%   
 mutate(Travel = fct\_recode(Travel, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`GreenCleanTech` = as\_factor(`GreenCleanTech`)) %>%   
 mutate(`GreenCleanTech` = fct\_recode(`GreenCleanTech`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(`UncertainOther` = as\_factor(`UncertainOther`)) %>%   
 mutate(`UncertainOther` = fct\_recode(`UncertainOther`, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MalePresenter = as\_factor(MalePresenter)) %>%   
 mutate(MalePresenter = fct\_recode(MalePresenter, "Yes" = "1", "No" = "0" )) %>%  
 mutate(FemalePresenter = as\_factor(FemalePresenter)) %>%   
 mutate(FemalePresenter = fct\_recode(FemalePresenter, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MixedGenderPresenters = as\_factor(MixedGenderPresenters)) %>%   
 mutate(MixedGenderPresenters = fct\_recode(MixedGenderPresenters, "Yes" = "1", "No" = "0" )) %>%  
 mutate(CompanyState = as\_factor(CompanyState)) %>%   
 mutate(CompanyState = fct\_recode(CompanyState, "Yes" = "1", "No" = "0" )) %>%  
 mutate(BarbaraCorcoran = as\_factor(BarbaraCorcoran)) %>%   
 mutate(BarbaraCorcoran = fct\_recode(BarbaraCorcoran, "Yes" = "1", "No" = "0" )) %>%  
 mutate(MarkCuban = as\_factor(MarkCuban)) %>%   
 mutate(MarkCuban = fct\_recode(MarkCuban, "Yes" = "1", "No" = "0" )) %>%  
 mutate(LoriGreiner = as\_factor(LoriGreiner)) %>%   
 mutate(LoriGreiner = fct\_recode(LoriGreiner, "Yes" = "1", "No" = "0" )) %>%  
 mutate(RobertHerjavec = as\_factor(RobertHerjavec)) %>%   
 mutate(RobertHerjavec = fct\_recode(RobertHerjavec, "Yes" = "1", "No" = "0" )) %>%  
 mutate(DaymondJohn = as\_factor(DaymondJohn)) %>%   
 mutate(DaymondJohn = fct\_recode(DaymondJohn, "Yes" = "1", "No" = "0" )) %>%  
 mutate(KevinOLeary = as\_factor(KevinOLeary)) %>%   
 mutate(KevinOLeary = fct\_recode(KevinOLeary, "Yes" = "1", "No" = "0" )) %>%  
 mutate(KevinHarrington = as\_factor(KevinHarrington)) %>%   
 mutate(KevinHarrington = fct\_recode(KevinHarrington, "Yes" = "1", "No" = "0" )) %>%  
 mutate(Guest = as\_factor(Guest)) %>%   
 mutate(Guest = fct\_recode(Guest, "Yes" = "1", "No" = "0" ))

## Warning: Unknown levels in `f`: 3

## Warning: Unknown levels in `f`: 1, 3, 4

## Warning: Unknown levels in `f`: 1, 2, 3, 4

## Warning: Unknown levels in `f`: 1

## Warning: Unknown levels in `f`: 1, 0

# Remove weak or non-required variables  
sharkcompetition = sharkcompetition %>% select(-CompanyState)   
sharkcompetition = sharkcompetition %>% select(-Male4)  
# sharkcompetition = sharkcompetition %>% select(-Female4)  
# sharkcompetition = sharkcompetition %>% select(-Eth4)  
# sharkcompetition = sharkcompetition %>% select(-Eth5)  
sharkcompetition = sharkcompetition %>% select(-SeasonEpisode)  
sharkcompetition = sharkcompetition %>% select(-Company)  
sharkcompetition = sharkcompetition %>% select(-X1)

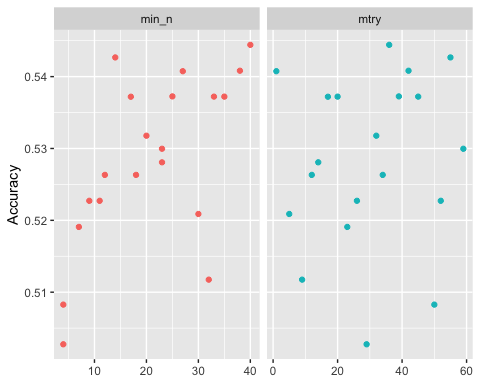
## Random Forests\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

set.seed(123)  
rf\_folds = vfold\_cv(sharkstudent, v = 5)

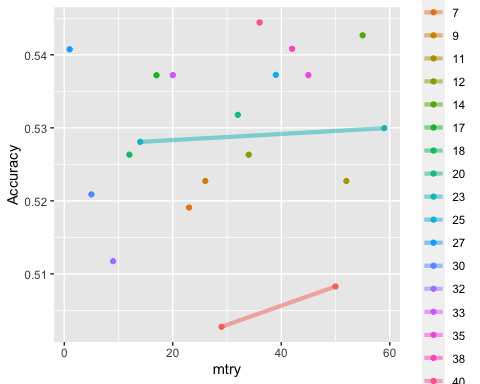
set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~., sharkstudent) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
set.seed(123)  
rf\_res = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = 20 #try 20 different combinations of the random forest tuning parameters  
)

## i Creating pre-processing data to finalize unknown parameter: mtry

#Plot for Tuning  
rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



#### Random Forest Tuning\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

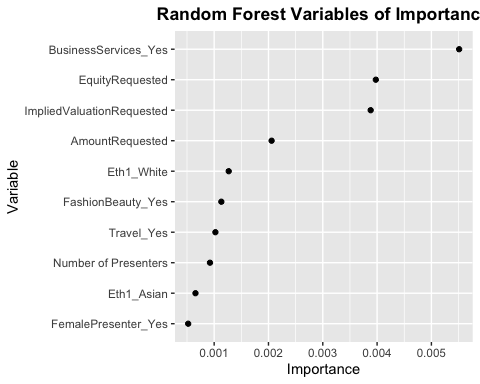
#RANDOM FOREST  
set.seed(123)  
rf\_folds = vfold\_cv(sharkstudent, v = 5)  
  
set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~., sharkstudent) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
  
rf\_grid = grid\_regular(  
 mtry(range = c(2, 30)), #these values determined through significant trial and error  
 min\_n(range = c(15, 25)), #these values determined through significant trial and error  
 levels = 5  
)  
  
set.seed(123)  
rf\_res\_tuned = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = rf\_grid #use the tuning grid  
)

#### Random Forest Var of Importance\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 shark\_wflow,  
 best\_rf  
)  
  
final\_rf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 9  
## trees = 100  
## min\_n = 22  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

set.seed(123)  
final\_rf\_fit = fit(final\_rf, sharkstudent)  
  
set.seed(123)  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point", mapping=aes\_string(fill="Variable")) + labs(title="Random Forest Variables of Importance", y="Importance", x="Variable") + theme(plot.title = element\_text(hjust = 0.5,face="bold"))



### Split the Data

set.seed(123)   
shark\_split = initial\_split(sharkstudent, prob = 0.70, strata = Deal\_Yes)  
train = training(shark\_split)  
test = testing(shark\_split)

### Predictions and Confusion Matrix for Sharkstudent

predictionrf = predict(final\_rf\_fit, sharkstudent)  
head(predictionrf)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 Yes   
## 4 Yes   
## 5 No   
## 6 No

confusionMatrix(predictionrf$.pred\_class, sharkstudent$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 189 19  
## Yes 61 282  
##   
## Accuracy : 0.8548   
## 95% CI : (0.8226, 0.8832)  
## No Information Rate : 0.5463   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7029   
##   
## Mcnemar's Test P-Value : 4.563e-06   
##   
## Sensitivity : 0.9369   
## Specificity : 0.7560   
## Pos Pred Value : 0.8222   
## Neg Pred Value : 0.9087   
## Prevalence : 0.5463   
## Detection Rate : 0.5118   
## Detection Prevalence : 0.6225   
## Balanced Accuracy : 0.8464   
##   
## 'Positive' Class : Yes   
##

### Predictions and Confusion Matrix for Train

predictionrf2 = predict(final\_rf\_fit, train)  
head(predictionrf2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(predictionrf2$.pred\_class, train$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 139 14  
## Yes 48 211  
##   
## Accuracy : 0.8495   
## 95% CI : (0.8113, 0.8826)  
## No Information Rate : 0.5461   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6917   
##   
## Mcnemar's Test P-Value : 2.777e-05   
##   
## Sensitivity : 0.9378   
## Specificity : 0.7433   
## Pos Pred Value : 0.8147   
## Neg Pred Value : 0.9085   
## Prevalence : 0.5461   
## Detection Rate : 0.5121   
## Detection Prevalence : 0.6286   
## Balanced Accuracy : 0.8405   
##   
## 'Positive' Class : Yes   
##

### Predictions and Confusion Matrix for Test

predictionrf3 = predict(final\_rf\_fit, test)  
head(predictionrf3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(predictionrf3$.pred\_class, test$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 50 5  
## Yes 13 71  
##   
## Accuracy : 0.8705   
## 95% CI : (0.8031, 0.9214)  
## No Information Rate : 0.5468   
## P-Value [Acc > NIR] : 2.753e-16   
##   
## Kappa : 0.7359   
##   
## Mcnemar's Test P-Value : 0.09896   
##   
## Sensitivity : 0.9342   
## Specificity : 0.7937   
## Pos Pred Value : 0.8452   
## Neg Pred Value : 0.9091   
## Prevalence : 0.5468   
## Detection Rate : 0.5108   
## Detection Prevalence : 0.6043   
## Balanced Accuracy : 0.8639   
##   
## 'Positive' Class : Yes   
##

### Predictions for sharkcompetition dataset

pred\_competition = predict(final\_rf\_fit, sharkcompetition, type = "class")  
head(pred\_competition)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 No   
## 4 Yes   
## 5 Yes   
## 6 No

#### Combine predicitons with shark competion ID

competition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

kaggle = competition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_1 = bind\_cols(kaggle, pred\_competition) #here, you would put your predictions object  
  
colnames(kaggle\_1)[colnames(kaggle\_1) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_1

## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <fct>   
## 1 1 No   
## 2 2 Yes   
## 3 3 No   
## 4 4 Yes   
## 5 5 Yes   
## 6 6 No   
## 7 7 No   
## 8 8 Yes   
## 9 9 Yes   
## 10 10 Yes   
## # … with 226 more rows

Now we can write this dataframe out to a CSV file. This is file that you submit to Kaggle.

write.csv(kaggle\_1, "kaggle\_submit.csv", row.names=FALSE)

#### Random Forest Model Tuned 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Split the Data

set.seed(123)   
shark\_split = initial\_split(sharkstudent, prob = 0.70, strata = Deal\_Yes)  
train = training(shark\_split)  
test = testing(shark\_split)

set.seed(123)  
rf\_folds = vfold\_cv(train, v = 5)

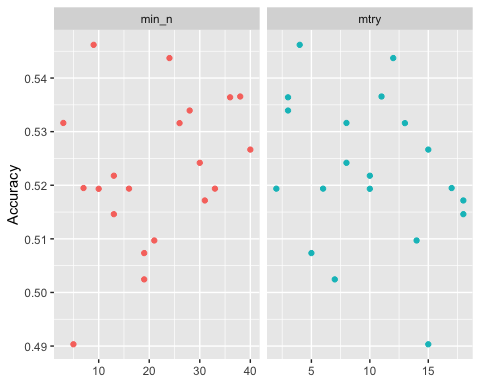
### RF with 10 important variables from Phase 1

set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ Season + BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
set.seed(123)  
rf\_res = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = 20 #try 20 different combinations of the random forest tuning parameters  
)

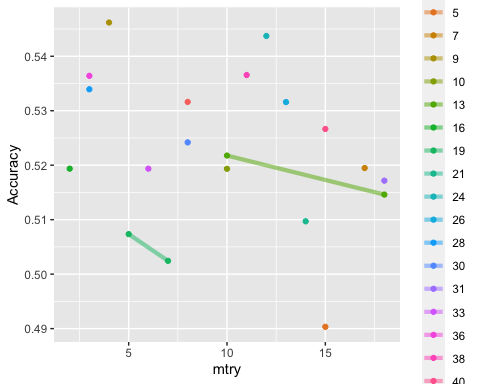
## i Creating pre-processing data to finalize unknown parameter: mtry

### Visual for Tuning

#Plot for Tuning  
rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



### Tuning

#RANDOM FOREST  
set.seed(123)  
rf\_folds = vfold\_cv(train, v = 5)  
  
set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ Season + BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
  
rf\_grid = grid\_regular(  
 mtry(range = c(5, 13)), #these values determined through significant trial and error  
 min\_n(range = c(24, 34)), #these values determined through significant trial and error  
 levels = 5  
)  
  
set.seed(123)  
rf\_res\_tuned = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = rf\_grid #use the tuning grid  
)

best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 shark\_wflow,  
 best\_rf  
)  
  
final\_rf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 9  
## trees = 100  
## min\_n = 24  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

set.seed(123)  
final\_rf\_fit = fit(final\_rf, train)

####Predictions and Confusion Matrix for Train Set

trainpredrf = predict(final\_rf\_fit, train)  
head(trainpredrf)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 No   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(trainpredrf$.pred\_class, train$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 137 20  
## Yes 50 205  
##   
## Accuracy : 0.8301   
## 95% CI : (0.7903, 0.8651)  
## No Information Rate : 0.5461   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6526   
##   
## Mcnemar's Test P-Value : 0.0005279   
##   
## Sensitivity : 0.9111   
## Specificity : 0.7326   
## Pos Pred Value : 0.8039   
## Neg Pred Value : 0.8726   
## Prevalence : 0.5461   
## Detection Rate : 0.4976   
## Detection Prevalence : 0.6189   
## Balanced Accuracy : 0.8219   
##   
## 'Positive' Class : Yes   
##

### Predictions and Confusion Matrix for Test

testpredrf = predict(final\_rf\_fit, test)  
head(testpredrf)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(testpredrf$.pred\_class, test$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 28 20  
## Yes 35 56  
##   
## Accuracy : 0.6043   
## 95% CI : (0.5179, 0.6862)  
## No Information Rate : 0.5468   
## P-Value [Acc > NIR] : 0.10026   
##   
## Kappa : 0.1851   
##   
## Mcnemar's Test P-Value : 0.05906   
##   
## Sensitivity : 0.7368   
## Specificity : 0.4444   
## Pos Pred Value : 0.6154   
## Neg Pred Value : 0.5833   
## Prevalence : 0.5468   
## Detection Rate : 0.4029   
## Detection Prevalence : 0.6547   
## Balanced Accuracy : 0.5906   
##   
## 'Positive' Class : Yes   
##

### Predictions for sharkcompetition data

#Generates Predictions for competition set  
predictionsrf = predict(final\_rf\_fit, sharkcompetition, type = "class")  
head(predictionsrf)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 No   
## 6 No

### Combine predicitons with shark competion ID

competition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

kaggle = competition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_3 = bind\_cols(kaggle, predictionsrf) #here, you would put your predictions object  
  
colnames(kaggle\_3)[colnames(kaggle\_3) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_3

## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <fct>   
## 1 1 No   
## 2 2 Yes   
## 3 3 Yes   
## 4 4 Yes   
## 5 5 No   
## 6 6 No   
## 7 7 No   
## 8 8 No   
## 9 9 Yes   
## 10 10 No   
## # … with 226 more rows

write.csv(kaggle\_3, "kaggle\_submit2.csv", row.names=FALSE)

#### Random Forest Model Tuned 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

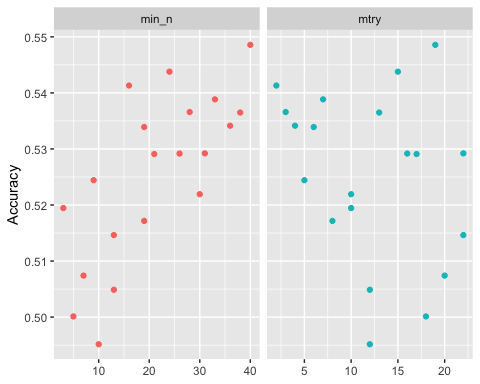
### RF with important variables + 3 more variables from Var of Importance Visual

set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ Season + BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome + FashionBeauty + Travel + `Number of Presenters` + FemalePresenter, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
set.seed(123)  
rf\_res = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = 20 #try 20 different combinations of the random forest tuning parameters  
)

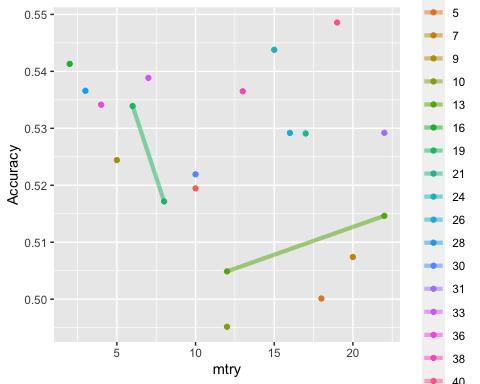
## i Creating pre-processing data to finalize unknown parameter: mtry

### Visual for Tuning

#Plot for Tuning  
rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



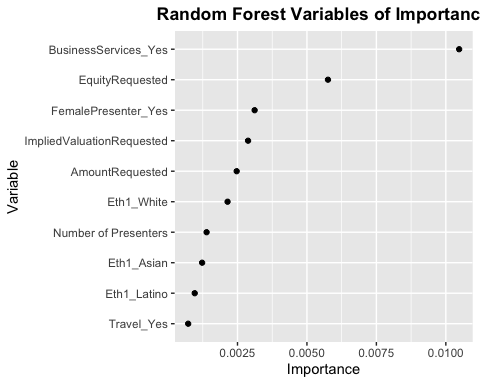
### Tuning

#RANDOM FOREST  
set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ Season + BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome + FashionBeauty + Travel + `Number of Presenters` + FemalePresenter, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
  
rf\_grid = grid\_regular(  
 mtry(range = c(0, 10)), #these values determined through significant trial and error  
 min\_n(range = c(15, 35)), #these values determined through significant trial and error  
 levels = 5  
)  
  
set.seed(123)  
rf\_res\_tuned = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = rf\_grid #use the tuning grid  
)

best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 shark\_wflow,  
 best\_rf  
)  
  
final\_rf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 10  
## trees = 100  
## min\_n = 30  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

set.seed(123)  
final\_rf\_fit = fit(final\_rf, train)  
  
set.seed(123)  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point", mapping=aes\_string(fill="Variable")) + labs(title="Random Forest Variables of Importance", y="Importance", x="Variable") + theme(plot.title = element\_text(hjust = 0.5,face="bold"))



### Predictions and Confusion Matrix for Train set

trainpredrf2 = predict(final\_rf\_fit, train)  
head(trainpredrf2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(trainpredrf2$.pred\_class, train$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 140 18  
## Yes 47 207  
##   
## Accuracy : 0.8422   
## 95% CI : (0.8034, 0.8761)  
## No Information Rate : 0.5461   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6775   
##   
## Mcnemar's Test P-Value : 0.0005147   
##   
## Sensitivity : 0.9200   
## Specificity : 0.7487   
## Pos Pred Value : 0.8150   
## Neg Pred Value : 0.8861   
## Prevalence : 0.5461   
## Detection Rate : 0.5024   
## Detection Prevalence : 0.6165   
## Balanced Accuracy : 0.8343   
##   
## 'Positive' Class : Yes   
##

### Predictions and Confusion Matrix for Test

testpredrf2 = predict(final\_rf\_fit, test)  
head(testpredrf2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(testpredrf2$.pred\_class, test$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 22 23  
## Yes 41 53  
##   
## Accuracy : 0.5396   
## 95% CI : (0.453, 0.6244)  
## No Information Rate : 0.5468   
## P-Value [Acc > NIR] : 0.60177   
##   
## Kappa : 0.0477   
##   
## Mcnemar's Test P-Value : 0.03359   
##   
## Sensitivity : 0.6974   
## Specificity : 0.3492   
## Pos Pred Value : 0.5638   
## Neg Pred Value : 0.4889   
## Prevalence : 0.5468   
## Detection Rate : 0.3813   
## Detection Prevalence : 0.6763   
## Balanced Accuracy : 0.5233   
##   
## 'Positive' Class : Yes   
##

### Predictions for sharkcompetition data

predictionsrf2 = predict(final\_rf\_fit, sharkcompetition, type = "class")  
head(predictionsrf2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 No

#### Combine predicitons with shark competion ID

competition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

kaggle = competition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_5 = bind\_cols(kaggle, predictionsrf2) #here, you would put your predictions object  
  
colnames(kaggle\_5)[colnames(kaggle\_5) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_5

## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <fct>   
## 1 1 No   
## 2 2 Yes   
## 3 3 Yes   
## 4 4 Yes   
## 5 5 Yes   
## 6 6 No   
## 7 7 No   
## 8 8 Yes   
## 9 9 Yes   
## 10 10 Yes   
## # … with 226 more rows

Now we can write this dataframe out to a CSV file. This is file that you submit to Kaggle.

write.csv(kaggle\_5, "kaggle\_submit3.csv", row.names=FALSE)

#### Random Forest Model Tuned 3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

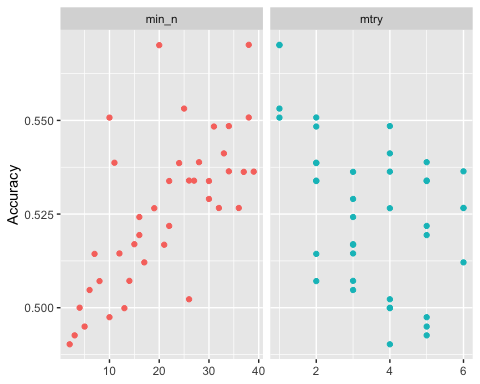
### RF with Quantitative Variables + Important Categorical Variables

set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ AmountRequested + EquityRequested + ImpliedValuationRequested + MalePresenter + FemalePresenter + BusinessServices, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
set.seed(123)  
rf\_res = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = 40 #try 20 different combinations of the random forest tuning parameters  
)

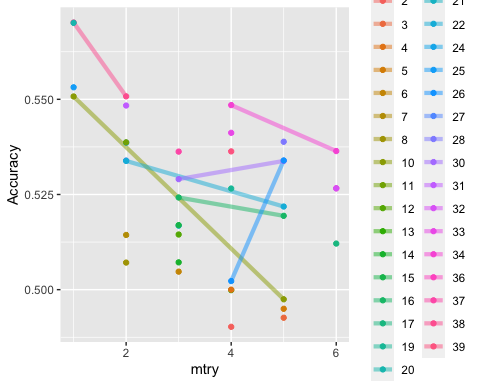
## i Creating pre-processing data to finalize unknown parameter: mtry

### Visual for Tuning

#Plot for Tuning  
rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, min\_n, mtry) %>%  
 pivot\_longer(min\_n:mtry,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



rf\_res %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 mutate(min\_n = factor(min\_n)) %>%  
 ggplot(aes(mtry, mean, color = min\_n)) +  
 geom\_line(alpha = 0.5, size = 1.5) +  
 geom\_point() +  
 labs(y = "Accuracy")



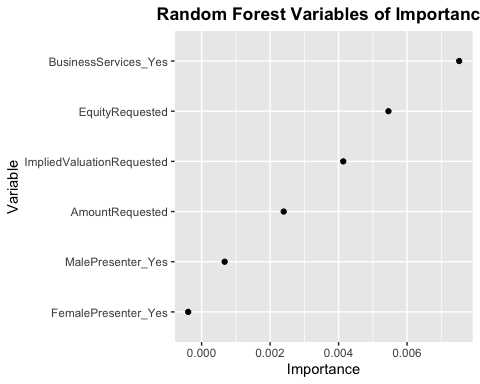
### Tuning

#RANDOM FOREST  
set.seed(123)  
shark\_recipe = recipe(Deal\_Yes ~ AmountRequested + EquityRequested + ImpliedValuationRequested + MalePresenter + FemalePresenter + BusinessServices, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
 # step\_other()  
 # step\_novel()  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 100) %>% #add tuning of mtry and min\_n parameters  
 #setting trees to 100 here should also speed things up a bit, but more trees might be better  
 set\_engine("ranger", importance = "permutation") %>% #added importance metric  
 set\_mode("classification")  
  
shark\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(shark\_recipe)  
  
  
rf\_grid = grid\_regular(  
 mtry(range = c(0, 5)), #these values determined through significant trial and error  
 min\_n(range = c(20, 40)), #these values determined through significant trial and error  
 levels = 5  
)  
  
set.seed(123)  
rf\_res\_tuned = tune\_grid(  
 shark\_wflow,  
 resamples = rf\_folds,  
 grid = rf\_grid #use the tuning grid  
)

best\_rf = select\_best(rf\_res\_tuned, "accuracy")  
  
final\_rf = finalize\_workflow(  
 shark\_wflow,  
 best\_rf  
)  
  
final\_rf

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 1  
## trees = 100  
## min\_n = 40  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

set.seed(123)  
final\_rf\_fit = fit(final\_rf, train)  
  
set.seed(123)  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point", mapping=aes\_string(fill="Variable")) + labs(title="Random Forest Variables of Importance", y="Importance", x="Variable") + theme(plot.title = element\_text(hjust = 0.5,face="bold"))



###Predictions and Confusion Matrix for Train

trainpredrf3 = predict(final\_rf\_fit, train)  
head(trainpredrf3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(trainpredrf3$.pred\_class, train$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 71 26  
## Yes 116 199  
##   
## Accuracy : 0.6553   
## 95% CI : (0.6072, 0.7012)  
## No Information Rate : 0.5461   
## P-Value [Acc > NIR] : 4.306e-06   
##   
## Kappa : 0.2753   
##   
## Mcnemar's Test P-Value : 8.098e-14   
##   
## Sensitivity : 0.8844   
## Specificity : 0.3797   
## Pos Pred Value : 0.6317   
## Neg Pred Value : 0.7320   
## Prevalence : 0.5461   
## Detection Rate : 0.4830   
## Detection Prevalence : 0.7646   
## Balanced Accuracy : 0.6321   
##   
## 'Positive' Class : Yes   
##

#Predictions and Confusion Matrix for Test

testpredrf3 = predict(final\_rf\_fit, test)  
head(testpredrf3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(testpredrf3$.pred\_class, test$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 15 13  
## Yes 48 63  
##   
## Accuracy : 0.5612   
## 95% CI : (0.4745, 0.6451)  
## No Information Rate : 0.5468   
## P-Value [Acc > NIR] : 0.4002   
##   
## Kappa : 0.0704   
##   
## Mcnemar's Test P-Value : 1.341e-05   
##   
## Sensitivity : 0.8289   
## Specificity : 0.2381   
## Pos Pred Value : 0.5676   
## Neg Pred Value : 0.5357   
## Prevalence : 0.5468   
## Detection Rate : 0.4532   
## Detection Prevalence : 0.7986   
## Balanced Accuracy : 0.5335   
##   
## 'Positive' Class : Yes   
##

### Predictions for sharkcompetition data

predictionsrf3 = predict(final\_rf\_fit, sharkcompetition, type = "class")  
head(predictionsrf3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 No

#### Combine predicitons with shark competion ID

competition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

kaggle = competition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_6 = bind\_cols(kaggle, predictionsrf3) #here, you would put your predictions object  
  
colnames(kaggle\_6)[colnames(kaggle\_6) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_6

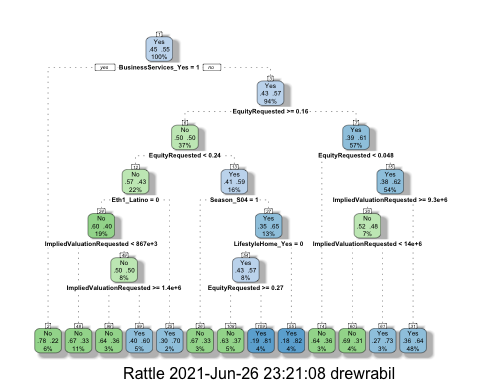
## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <fct>   
## 1 1 Yes   
## 2 2 Yes   
## 3 3 Yes   
## 4 4 Yes   
## 5 5 Yes   
## 6 6 No   
## 7 7 No   
## 8 8 Yes   
## 9 9 Yes   
## 10 10 Yes   
## # … with 226 more rows

Now we can write this dataframe out to a CSV file. This is file that you submit to Kaggle.

write.csv(kaggle\_6, "kaggle\_submit4.csv", row.names=FALSE)

## Classification Tree\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

library(rpart) #for classification trees  
library(rpart.plot)  
  
sharkcf\_recipe = recipe(Deal\_Yes ~ Season +BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome, train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree() %>%  
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
sharkcf\_wflow =workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(sharkcf\_recipe)  
  
sharkcf\_fit = fit(sharkcf\_wflow, train)  
  
tree = sharkcf\_fit %>% pull\_workflow\_fit() %>% pluck("fit")  
  
#plot the tree  
fancyRpartPlot(tree)



sharkcf\_fit$fit$fit$fit$cptable

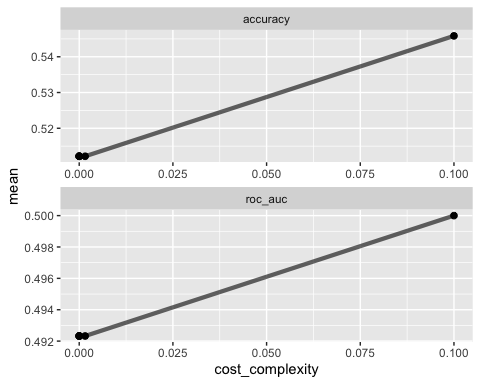
## CP nsplit rel error xerror xstd  
## 1 0.06951872 0 1.0000000 1.0000000 0.05404081  
## 2 0.03208556 1 0.9304813 0.9518717 0.05376847  
## 3 0.02139037 3 0.8663102 1.0374332 0.05417999  
## 4 0.01604278 5 0.8235294 1.0588235 0.05423121  
## 5 0.01336898 8 0.7754011 1.0855615 0.05426639  
## 6 0.01069519 10 0.7486631 1.0855615 0.05426639  
## 7 0.01000000 12 0.7272727 1.1390374 0.05424080

set.seed(123)  
folds = vfold\_cv(train, v = 5)

sharkcf\_recipe = recipe(Deal\_Yes ~ Season +BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome, train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 6)   
  
sharkcf\_wflow =workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(sharkcf\_recipe)  
  
tree\_res =   
 sharkcf\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )

Borrowed code from: <https://www.tidymodels.org/start/tuning/>

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.1 Preprocessor1\_Model6

final\_wf =   
 sharkcf\_wflow %>%   
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

Predictions on training set

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred$.pred\_class,train$Deal\_Yes,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 0 0  
## Yes 187 225  
##   
## Accuracy : 0.5461   
## 95% CI : (0.4966, 0.5949)  
## No Information Rate : 0.5461   
## P-Value [Acc > NIR] : 0.5203   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.5461   
## Neg Pred Value : NaN   
## Prevalence : 0.5461   
## Detection Rate : 0.5461   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

Predictions on testing set

treepred\_test = predict(final\_fit, test, type = "class")  
head(treepred\_test)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test$.pred\_class,test$Deal\_Yes,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 0 0  
## Yes 63 76  
##   
## Accuracy : 0.5468   
## 95% CI : (0.4602, 0.6313)  
## No Information Rate : 0.5468   
## P-Value [Acc > NIR] : 0.535   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 5.662e-15   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.5468   
## Neg Pred Value : NaN   
## Prevalence : 0.5468   
## Detection Rate : 0.5468   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

## Naive Bayes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Data set is split 70% into a training set and 30% into a test set  
set.seed(123)  
shark\_split = initial\_split(sharkstudent, prop = 0.70, strata = Deal\_Yes)  
train = training(shark\_split)  
test = testing(shark\_split)  
  
folds = vfold\_cv(train,v=5)

#### Removes quantitative variables, variables with multiple non-binary categoreis, and Female 4 variable (train df did not have a “yes”)

trainnb=train %>%   
 select(-`Number of Presenters`) %>%   
 select(-Eth1) %>%   
 select(-Eth2) %>%   
 select(-Eth3) %>%   
 select(-Eth4) %>%   
 select(-Eth5) %>%   
 select(-AmountRequested) %>%   
 select(-EquityRequested) %>%   
 select(-ImpliedValuationRequested) %>%   
 select(-Season)   
   
 testnb=test %>%   
 select(-`Number of Presenters`) %>%   
 select(-Eth1) %>%   
 select(-Eth2) %>%   
 select(-Eth3) %>%   
 select(-Eth4) %>%   
 select(-Eth5) %>%   
 select(-AmountRequested) %>%   
 select(-EquityRequested) %>%   
 select(-ImpliedValuationRequested) %>%   
 select(-Season)

#### Set up Naive Bayes Data

#create objects x which holds the predictor variables and y which holds the response variables  
x = trainnb[,-1]  
y = trainnb$Deal\_Yes

#### Runs Naive Bayes Model on train

model = train(x,y,'nb',trControl=trainControl(method='cv',number=10))

## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 17

## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 17

## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 20

## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 10

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 20

## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 23

## Warning: Setting row names on a tibble is deprecated.

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 23

## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.  
  
## Warning: Setting row names on a tibble is deprecated.

#### Generate Confusion Matrix on train and test data

Predicttrain <- predict(model,newdata = trainnb) #Get the confusion matrix to see accuracy value and other parameter values ]

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 172

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 188

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 226

confusionMatrix(Predicttrain, trainnb$Deal\_Yes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 103 78  
## Yes 72 132  
##   
## Accuracy : 0.6104   
## 95% CI : (0.5597, 0.6594)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.005886   
##   
## Kappa : 0.2165   
##   
## Mcnemar's Test P-Value : 0.683091   
##   
## Sensitivity : 0.5886   
## Specificity : 0.6286   
## Pos Pred Value : 0.5691   
## Neg Pred Value : 0.6471   
## Prevalence : 0.4545   
## Detection Rate : 0.2675   
## Detection Prevalence : 0.4701   
## Balanced Accuracy : 0.6086   
##   
## 'Positive' Class : No   
##

Predicttest <- predict(model,newdata = testnb) #Get the confusion matrix to see accuracy value and other parameter values ]

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 24

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 51

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 161

confusionMatrix(Predicttest, testnb$Deal\_Yes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 33 43  
## Yes 42 48  
##   
## Accuracy : 0.488   
## 95% CI : (0.4097, 0.5666)  
## No Information Rate : 0.5482   
## P-Value [Acc > NIR] : 0.9489   
##   
## Kappa : -0.0325   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.4400   
## Specificity : 0.5275   
## Pos Pred Value : 0.4342   
## Neg Pred Value : 0.5333   
## Prevalence : 0.4518   
## Detection Rate : 0.1988   
## Detection Prevalence : 0.4578   
## Balanced Accuracy : 0.4837   
##   
## 'Positive' Class : No   
##

### Submission Prep\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Prepare Shark Competitition df for Prediction For Submission

#Generates Predictions for competition set  
  
sharkcompnb=sharkcompetition %>%   
 select(-`Number of Presenters`) %>%   
 select(-Eth1) %>%   
 select(-Eth2) %>%   
 select(-Eth3) %>%   
 select(-Eth4) %>%   
 select(-Eth5) %>%   
 select(-AmountRequested) %>%   
 select(-EquityRequested) %>%   
 select(-ImpliedValuationRequested) %>%   
 select(-Season)

#### Generate predictions for sharkcompetition

pred\_competition = predict(model, sharkcompnb, type = "prob")

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 18

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 23

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with  
## observation 221

head(pred\_competition)

## No Yes  
## 1 0.6852159 0.3147841  
## 2 0.3975377 0.6024623  
## 3 0.5185938 0.4814062  
## 4 0.3560919 0.6439081  
## 5 0.6852159 0.3147841  
## 6 0.8323064 0.1676936

pred\_competitiondf <- as.data.frame(pred\_competition)

#### Converts Probabilities to Yes and No

pred\_competition<- mutate(pred\_competition,"Deal\_Yes"=case\_when(Yes>0.5 ~ "Yes", Yes<0.5 ~ "No"))

#### Combine predicitons with shark competion ID

#competition <- read\_csv("shark\_competition.csv")  
kaggle = sharkcompetition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_2 = bind\_cols(kaggle, pred\_competition) #here, you would put your predictions object  
  
colnames(kaggle\_2)[colnames(kaggle\_2) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_2 = select(kaggle\_2,ID,Deal\_Yes)  
  
kaggle\_2

## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <chr>   
## 1 1 No   
## 2 2 Yes   
## 3 3 No   
## 4 4 Yes   
## 5 5 No   
## 6 6 No   
## 7 7 No   
## 8 8 Yes   
## 9 9 Yes   
## 10 10 Yes   
## # … with 226 more rows

#### Finalize

Now we can write this dataframe out to a CSV file. This is file that you submit to Kaggle.

write.csv(kaggle\_2, "kagglenb\_submit.csv", row.names=FALSE)

## Log Reg\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Data set is split 70% into a training set and 30% into a test set  
set.seed(123)  
shark\_split = initial\_split(sharkstudent, prop = 0.70, strata = Deal\_Yes)  
train = training(shark\_split)  
test = testing(shark\_split)  
  
folds = vfold\_cv(train,v=5)

#Creates Logistic regression model for predictor variable state for response variable violator  
shark\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
shark\_recipe = recipe(Deal\_Yes~Season + BusinessServices + EquityRequested + ImpliedValuationRequested + AmountRequested + Eth1 + ChildrenEducation + Automotive + LifestyleHome, train) %>%  
 step\_other(BusinessServices,ChildrenEducation,Automotive,LifestyleHome, threshold = 0.05) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted %<%  
   
logreg\_wf = workflow() %>%  
 add\_recipe(shark\_recipe) %>%   
 add\_model(shark\_model)  
  
shark\_fit = fit(logreg\_wf, train)  
  
summary(shark\_fit$fit$fit$fit)

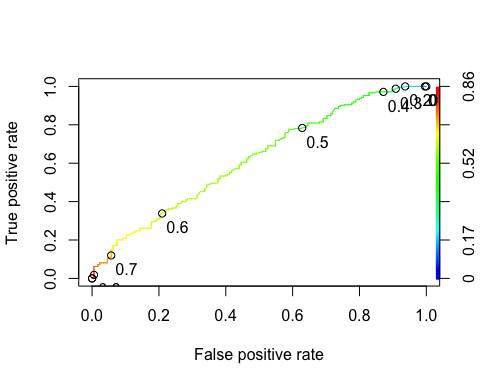
##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9718 -1.2224 0.7952 1.0740 1.7778   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.337e+01 5.354e+02 -0.025 0.9801   
## EquityRequested -9.601e-01 1.288e+00 -0.746 0.4559   
## ImpliedValuationRequested -1.450e-08 3.729e-08 -0.389 0.6974   
## AmountRequested -2.035e-07 4.048e-07 -0.503 0.6151   
## Season\_S02 -3.226e-01 6.947e-01 -0.464 0.6424   
## Season\_S03 -3.128e-01 5.921e-01 -0.528 0.5973   
## Season\_S04 -8.723e-01 5.330e-01 -1.637 0.1017   
## Season\_S05 -7.558e-01 5.290e-01 -1.429 0.1531   
## Season\_S06 -5.840e-01 5.377e-01 -1.086 0.2774   
## Season\_S07 -3.631e-01 5.310e-01 -0.684 0.4942   
## Season\_S08 -2.791e-01 5.665e-01 -0.493 0.6222   
## Season\_S09 -4.374e-01 5.499e-01 -0.796 0.4263   
## BusinessServices\_other -1.288e+00 6.026e-01 -2.137 0.0326 \*  
## Eth1\_African.American 1.429e+01 5.354e+02 0.027 0.9787   
## Eth1\_White 1.423e+01 5.354e+02 0.027 0.9788   
## Eth1\_Asian 1.527e+01 5.354e+02 0.029 0.9773   
## Eth1\_Latino 1.391e+01 5.354e+02 0.026 0.9793   
## ChildrenEducation\_Yes 7.389e-01 3.792e-01 1.949 0.0513 .  
## Automotive\_other 5.444e-01 9.085e-01 0.599 0.5490   
## LifestyleHome\_Yes 3.111e-01 2.929e-01 1.062 0.2881   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 530.54 on 384 degrees of freedom  
## Residual deviance: 505.85 on 365 degrees of freedom  
## AIC: 545.85  
##   
## Number of Fisher Scoring iterations: 12

#### Predictions and Analysis

#Isolate Violator probabilities  
predictions = predict(shark\_fit, train, type="prob")[2]  
#head(predictions)

Threshold selection

#Prepares object for ROC curve or threshold  
ROCRpred = prediction(predictions, train$Deal\_Yes)   
  
#Plots ROC:  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



Area under the curve (AUC). AUC is a measure of the strength of the model. Values closer to 1 are better. Can be used to compare models.

as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.62

#Determine threshold to balance sensitivity and specificity  
#DO NOT modify this code  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.6333333  
## specificity 0.5314286  
## cutoff 0.5428840

#### Confusion matrix

#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
sharktest <- table(train$Deal\_Yes,predictions > 0.542)  
sharktest

##   
## FALSE TRUE  
## No 92 83  
## Yes 77 133

Accuracy

(sharktest[1,1]+sharktest[2,2])/nrow(train)

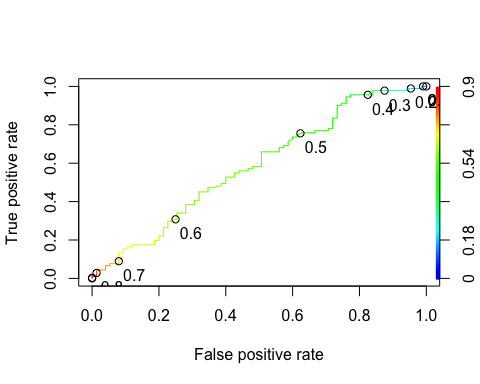
## [1] 0.5844156

#### Run Predictions for Test Data

#Isolate Violator probabilities  
predictions = predict(shark\_fit, test, type="prob")[2]  
#head(predictions)

Threshold selection

#Prepares object for ROC curve or threshold  
ROCRpred = prediction(predictions, test$Deal\_Yes)   
  
#Plots ROC:  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.5843223

#Determine threshold to balance sensitivity and specificity  
#DO NOT modify this code  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.6593407  
## specificity 0.4933333  
## cutoff 0.5320759

#### Confusion matrix

#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
sharktest <- table(test$Deal\_Yes,predictions > 0.532)  
sharktest

##   
## FALSE TRUE  
## No 37 38  
## Yes 31 60

Accuracy

(sharktest[1,1]+sharktest[2,2])/nrow(test)

## [1] 0.5843373

### Submission for Kaggle

#### Predictions on competition set

#Generates Predictions for competition set  
pred\_competition = predict(shark\_fit, sharkcompetition, type = "class")  
head(pred\_competition)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 No

#### Combine predicitons with shark competion ID

competition <- read\_csv("shark\_competition.csv")

## Warning: Missing column names filled in: 'X1' [1]

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## .default = col\_double(),  
## Company = col\_character(),  
## SeasonEpisode = col\_character(),  
## CompanyState = col\_character()  
## )  
## ℹ Use `spec()` for the full column specifications.

kaggle = competition %>% rowid\_to\_column("ID") %>% select(ID) #creating a data frame with just the ID number from competition  
  
kaggle\_2 = bind\_cols(kaggle, pred\_competition) #here, you would put your predictions object  
colnames(kaggle\_2)[colnames(kaggle\_2) == ".pred\_class"] <- "Deal\_Yes"  
  
kaggle\_2

## # A tibble: 236 x 2  
## ID Deal\_Yes  
## <int> <fct>   
## 1 1 Yes   
## 2 2 Yes   
## 3 3 Yes   
## 4 4 Yes   
## 5 5 Yes   
## 6 6 No   
## 7 7 No   
## 8 8 Yes   
## 9 9 Yes   
## 10 10 Yes   
## # … with 226 more rows

Now we can write this dataframe out to a CSV file. This is file that you submit to Kaggle.

write.csv(kaggle\_2, "kaggleLog\_submit.csv", row.names=FALSE)