TASK 2

Once again, the code was used to create a column named "FirstLanguage" which only lists the first language in the column "Languages" for each row. For example, if the column entry for "Languages" is "Spanish, German" – the FirstLanguage column entry is "Spanish." This was done under the assumption that the language listed first was the main language the movie was available in – and to simplify the process of generating regression tree.

	minsplit	maxdepth	cp	error
1	8	13	0.01	0.5205
2	9	9	0.01	0.5220
3	19	11	0.01	0.5240
4	20	10	0.01	0.5251
5	8	12	0.01	0.5255

The above table was generated to find out the best possible combination of 'minsplit' and 'maxdepth' that would yield the least error. Different regression tree models were mapped out, as can be seen in the code on the next page. I think the most important features for calculating Hidden Gem Score are Rotten Tomatoes Score and "FirstLanguage" hence "Languages" – as can be seen from the regression tree following. According to the regression tree, this prediction model works really well for some combinations of Rotten Tomatoes Score and Languages but not so well for others.

```
install.packages("rattle")

library(here) library(tidyverse) library(ggplot2) library(ggpubr) library(knitr) library(rpart) library(rpart.plot) library(rattle) library(RColorBrewer)

mydata <- read.csv("Final_Project_FlixGem.csv")

mydata <- mydata %>% select(Title, Languages, Series.or.Movie, Hidden.Gem.Score, Runtime, Director, IMDb.Score, Rotten.Tomatoes.Score, Metacritic.Score, Release.Date, Summary)

mydata <- mydata %>% filter(Series.or.Movie == 'Movie')

mydata <- na.omit(mydata)

testdata <- mydata %>% mutate(FirstLanguage = sub(",.*", "", mydata$Languages))

view(testdata)

hyper_grid <- expand.grid( minsplit = seq(5, 20, 1), maxdepth = seq(8, 15, 1) )

models <- list()

for (i in 1:nrow(hyper_grid)) {

# get minsplit, maxdepth values at row i minsplit <- hyper_grid$minsplit[i] maxdepth <- hyper_grid$maxdepth[i]

# train a model and store in the list models[[i]] <- rpart( formula = Hidden.Gem.Score ~ Runtime + IMDb.Score + Rotten.Tomatoes.Score + Metacritic.Score + FirstLanguage, data = testdata, method = "anova", control = list(minsplit = minsplit, maxdepth = maxdepth) ) }

get_cp <- function(x) { min <- which.min(x$cptable[, "xerror"]) cp <- x$cptable[min, "CP"] }

get_min_error <- function(x) { min <- which.min(x$cptable[, "xerror"]) xerror <- x$cptable[min, "xerror"] }

hyper_grid %>% mutate( cp = purrr::map_dbl(models, get_cp), error = purrr::map_dbl(models, get_min_error) ) %>% arrange(error) %>% top_n(-5, wt = error)

set.seed(123) train.index = sample(1:dim(mydata)[1],dim(mydata)[1]*0.7) train = mydata[train.index,] valid = mydata[-train.index,]

model <- rpart(formula = Hidden.Gem.Score ~ Runtime + IMDb.Score + Rotten.Tomatoes.Score + Metacritic.Score + FirstLanguage, data = testdata, method = "anova", control = list(minsplit = 8, maxdepth = 13, cp = 0.01))

rpart.plot(model, cex = 0.12)

default.model <- rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score, data = train) info.model <-
rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score, data = train, parms=list(split="information")) overfit.model <-
rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score, data = train,maxdepth= 5, minsplit=2,minbucket = 1)
one.rule.model<- rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score, data = train, maxdepth =1)
rpart.plot(one.rule.model, main="Single Rule Model") super.overfit.model <-
rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score, data = train, minsplit=2,minbucket = 1, cp = 0.0001)
rpart.plot(super.overfit.model, main = "Really Overfit") cost.driven.model <-
rpart(Hidden.Gem.Score~Languages+Runtime+IMDb.Score+Rotten.Tomatoes.Score+Metacritic.Score,data=train,parms=list(loss=matrix(c(0,1,5,0),byrow=TRUE,nrow=2)))
```