32455038-final-assignment

June 14, 2024

1 FIT5197 2024 S1 Final Assessment

SPECIAL NOTE: Please refer to the assessment page for rules, general guidelines and marking rubrics of the assessment (the marking rubric for the kaggle competition part will be released near the deadline in the same page). Failure to comply with the provided information will result in a deduction of mark (e.g., late penalties) or breach of academic integrity.

YOUR NAME: Jeevan Rajagopal

STUDENT ID: 32455038

KAGGLE NAME/ID (See part 1, Question 5 or part 2, there are penalties if you don't enter it here!!!): j_rank1

Please also enter your details in this google form.

2 Part 1 Regression (50 Marks)

A few thousand people were questioned in a life and wellbeing survey to build a model to predict happiness of an individual. You need to build regression models to optimally predict the variable in the survey dataset called 'happiness' based on any, or all, of the other survey question responses.

You have been provided with two datasets, `regression_train.csv` and `regression_test.csv`. Using these datasets, you hope to build a model that can predict happiness level using the other variables. `regression_train.csv` comes with the ground-truth target label (i.e. happiness level) whereas `regression_test.csv` comes with independent variables (input information) only.

On the order of around 70 survey questions have been converted into predictor variables that can be used to predict happiness. We do not list all the predictor names here, but their names given in the data header can clearly be linked to the survey questions. e.g., the predictor variable 'iDontFeelParticularlyPleasedWithTheWayIAm' corresponds to the survey question 'I don't feel particularly pleased with the way I am.'

PLEASE NOTE THAT THE USE OF LIBRARIES ARE PROHIBITED IN THESE QUESTIONS UNLESS STATED OTHERWISE, ANSWERS USING LIBRARIES WILL RECEIVE 0 MARKS

2.1 Question 1 (NO LIBRARIES ALLOWED) (4 Mark)

Please load the `regression_train.csv` and fit a **multiple linear regression model** with 'happiness' being the target variable. According to the summary table, which predictors do you think

are possibly associated with the target variable (use the significance level of 0.01), and which are the **Top 5** strongest predictors? Please write an R script to automatically fetch and print this information.

NOTE: Manually doing the above tasks will result in 0 marks.

[147]:
```r
# ANSWER BLOCK

#Load the regression files to dataframes.
train_r <- read_csv('/kaggle/input/regression/regression_train.csv')
test_r <- read_csv('/kaggle/input/regression/regression_test.csv')
head(train_r)
```

```
Rows: 500 Columns: 43
  Column specification

Delimiter: ","
chr (10): gender, income, whatIsYourHeightExpressItAsANumberInMetresM,
doYou…
dbl (33): alwaysAnxious, alwaysStressed,
alwaysAccountableAndResponsibleForY…

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Rows: 90 Columns: 42
  Column specification

Delimiter: ","
chr (10): gender, income, whatIsYourHeightExpressItAsANumberInMetresM,
doYou…
dbl (32): alwaysAnxious, alwaysStressed,
alwaysAccountableAndResponsibleForY…

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

A tibble: 6 × 43

gender <chr>	income <chr>	whatIsYourHeightExpressItAsANumberInMetresM <chr>	doYouFeelASenseC <chr>
Male	200k above	165 - 170	Yes
Female	200k above	165 - 170	Yes
Male	120k - 150k	175 - 180	Yes
Male	120k - 150k	175 - 180	Yes
Male	80k - 120k	175 - 180	Yes
Male	80k - 120k	170 - 175	Yes

[148]:
```
summary(train_r)
```

```
   gender             income
Length:500         Length:500
Class :character   Class :character
Mode  :character   Mode  :character


whatIsYourHeightExpressItAsANumberInMetresM
Length:500
Class :character
Mode  :character


doYouFeelASenseOfPurposeAndMeaningInYourLife104
Length:500
Class :character
Mode  :character


howDoYouReconcileSpiritualBeliefsWithScientificOrRationalThinki
Length:500
Class :character
Mode  :character


howOftenDoYouFeelSociallyConnectedWithYourPeersAndFriends
Length:500
Class :character
Mode  :character


doYouHaveASupportSystemOfFriendsAndFamilyToTurnToWhenNeeded
Length:500
Class :character
Mode  :character


howOftenDoYouParticipateInSocialActivitiesIncludingClubsSportsV
Length:500
Class :character
```

```
Mode  :character


doYouFeelComfortableEngagingInConversationsWithPeopleFromDiffer
Length:500
Class :character
Mode  :character


doYouFeelASenseOfPurposeAndMeaningInYourLife105 alwaysAnxious
Length:500                                       Min.   :-2.000
Class :character                                 1st Qu.:-2.000
Mode  :character                                 Median :-1.000
                                                 Mean   :-0.548
                                                 3rd Qu.: 0.000
                                                 Max.   : 2.000
alwaysStressed   alwaysAccountableAndResponsibleForYourActions   alwaysCalm
Min.   :-2.000   Min.   :-2.000                                  Min.   :-2
1st Qu.:-2.000   1st Qu.: 0.000                                  1st Qu.:-1
Median : 0.000   Median : 1.000                                  Median : 0
Mean   :-0.416   Mean   : 0.794                                  Mean   : 0
3rd Qu.: 1.000   3rd Qu.: 1.000                                  3rd Qu.: 1
Max.   : 2.000   Max.   : 2.000                                  Max.   : 2
myBodyIsHypermobileAndLovesToMove alwaysHaveFun    alwaysSerious
Min.   :-2.000                    Min.   :-2.000   Min.   :-2.000
1st Qu.:-1.000                    1st Qu.:-1.000   1st Qu.:-1.000
Median : 0.000                    Median : 0.000   Median : 0.000
Mean   :-0.154                    Mean   :-0.036   Mean   :-0.304
3rd Qu.: 1.000                    3rd Qu.: 1.000   3rd Qu.: 0.000
Max.   : 2.000                    Max.   : 2.000   Max.   : 2.000
alwaysDepressed alwaysLoveAndCareForYourself extremelyGoodAbilityToSense
Min.   :-2.00   Min.   :-2.000                Min.   :-2.0
1st Qu.:-2.00   1st Qu.: 0.000                1st Qu.: 0.0
Median :-1.00   Median : 0.000                Median : 1.0
Mean   :-0.83   Mean   : 0.204                Mean   : 0.5
3rd Qu.: 0.00   3rd Qu.: 1.000                3rd Qu.: 1.0
Max.   : 2.00   Max.   : 2.000                Max.   : 2.0
alwaysHaveDigestiveProblems iAmIntenselyInterestedInOtherPeople
Min.   :-2.000              Min.   :-2.00
1st Qu.:-2.000              1st Qu.:-1.00
Median :-1.000              Median : 0.00
Mean   :-0.664              Mean   :-0.17
3rd Qu.: 0.000              3rd Qu.: 1.00
Max.   : 2.000              Max.   : 2.00
iRarelyWakeUpFeelingRested iFindMostThingsAmusing
Min.   :-2.000             Min.   :-2.000
```

```
1st Qu.:-1.000             1st Qu.: 0.000
Median : 0.000             Median : 0.000
Mean   :-0.128             Mean   : 0.048
3rd Qu.: 1.000             3rd Qu.: 1.000
Max.   : 2.000             Max.   : 2.000
iAmAlwaysCommittedAndInvolved iDoNotThinkThatTheWorldIsAGoodPlace
Min.   :-2.000                Min.   :-2.000
1st Qu.: 0.000                1st Qu.:-1.000
Median : 0.000                Median :-1.000
Mean   : 0.228                Mean   :-0.482
3rd Qu.: 1.000                3rd Qu.: 0.000
Max.   : 2.000                Max.   : 2.000
iAmWellSatisfiedAboutEverythingInMyLife iFindBeautyInSomeThings
Min.   :-2.000                          Min.   :-1.000
1st Qu.:-1.000                          1st Qu.: 0.000
Median : 0.000                          Median : 1.000
Mean   :-0.216                          Mean   : 0.904
3rd Qu.: 1.000                          3rd Qu.: 1.000
Max.   : 2.000                          Max.   : 2.000
iAlwaysHaveACheerfulEffectOnOthers iFeelThatIAmNotEspeciallyInControlOfMyLife
Min.   :-2.000                     Min.   :-2.000
1st Qu.: 0.000                     1st Qu.:-1.000
Median : 0.000                     Median : 0.000
Mean   : 0.182                     Mean   :-0.342
3rd Qu.: 1.000                     3rd Qu.: 1.000
Max.   : 2.000                     Max.   : 2.000
iUsuallyHaveAGoodInfluenceOnEvents iDontHaveFunWithOtherPeople
Min.   :-2.000                     Min.   :-2.000
1st Qu.: 0.000                     1st Qu.:-2.000
Median : 0.000                     Median :-1.000
Mean   : 0.072                     Mean   :-0.756
3rd Qu.: 1.000                     3rd Qu.: 0.000
Max.   : 2.000                     Max.   : 2.000
alwaysEngageInPreparingAndUsingYourSkillsAndTalentsInOrderToGai
Min.   :-2.000
1st Qu.: 0.000
Median : 0.500
Mean   : 0.312
3rd Qu.: 1.000
Max.   : 2.000
alwaysMakingProgress extremelyGoodCommunicator
Min.   :-2.000       Min.   :-2.000
1st Qu.: 0.000       1st Qu.: 0.000
Median : 1.000       Median : 0.000
Mean   : 0.434       Mean   : 0.104
3rd Qu.: 1.000       3rd Qu.: 1.000
Max.   : 2.000       Max.   : 2.000
iDontFeelParticularlyPleasedWithTheWayIAm iFeelThatLifeIsVeryRewarding
```

```
Min.   :-2.00                                Min.   :-2.000
1st Qu.:-1.00                                1st Qu.: 0.000
Median : 0.00                                Median : 0.000
Mean   :-0.17                                Mean   : 0.262
3rd Qu.: 1.00                                3rd Qu.: 1.000
Max.   : 2.00                                Max.   : 2.000
iHaveVeryWarmFeelingsTowardsAlmostEveryone
Min.   :-2.000
1st Qu.:-1.000
Median : 0.000
Mean   : 0.122
3rd Qu.: 1.000
Max.   : 2.000
iAmNotParticularlyOptimisticAboutTheFuture   lifeIsGood       iLaughALot
Min.   :-2.000                               Min.   :-2.000   Min.   :-2.00
1st Qu.:-1.000                               1st Qu.: 0.000   1st Qu.: 0.00
Median : 0.000                               Median : 1.000   Median : 0.00
Mean   :-0.418                               Mean   : 0.474   Mean   : 0.19
3rd Qu.: 1.000                               3rd Qu.: 1.000   3rd Qu.: 1.00
Max.   : 2.000                               Max.   : 2.000   Max.   : 2.00
iDontThinkILookAttractive   happiness
Min.   :-2.000            Min.   :-43.9638
1st Qu.:-1.000            1st Qu.: -9.1106
Median : 0.000            Median : -1.4087
Mean   :-0.394            Mean   : -0.7136
3rd Qu.: 0.000            3rd Qu.:  7.5050
Max.   : 2.000            Max.   : 40.5404
```

[149]:
```r
# Fit a multiple linear regression model
model <- lm(happiness ~ ., train_r)
# Print the summary of the model
#summary(model_lm)
```

[150]:
```r
# Extract significant predictors at significance level of 0.01
sig_preds <- names(coef(model)[summary(model_lm)$coefficients[, "Pr(>|t|)"] < 0.
  ↪01])
# Print the significant predictors
print("Significant predictors at the 0.01 level:\n")
print(sig_preds, sep = "\n")
```

```
[1] "Significant predictors at the 0.01 level:\n"
 [1] "(Intercept)"
 [2] "income10k - 15k"
 [3] "income120k - 150k"
 [4] "income150k - 200k"
 [5] "income15k - 20k"
 [6] "income200k above"
 [7] "income20k - 50k"
```

```
 [8] "income50k - 80k"
 [9] "income80k - 120k"
[10] "whatIsYourHeightExpressItAsANumberInMetresM175 - 180"
[11] "whatIsYourHeightExpressItAsANumberInMetresM180 - 185"
[12] "alwaysStressed"
```

[151]:
```r
# Sort to get the Top 5 predictors.
top_5_preds<- names(sort(abs(coef(model_lm)),decreasing = TRUE)[1:6])
# Print the top 5 strongest predictors
cat("\nTop 5 significant predictors are:\n")
cat(top_5_preds, sep = "\n")
```

```
Top 5 significant predictors are:
income80k - 120k
income50k - 80k
(Intercept)
income20k - 50k
income200k above
income15k - 20k
```

We ignore the "(Intercept)" above as it is not a valid predictor.

2.2 Question 2 (2 Mark)

R squared from the summary table reflects that the full model doesn't fit the training dataset well; thus, you try to quantify the error between the values of the ground-truth and those of the model prediction. You want to write a function to predict 'happiness' with the given dataset and calculate the root mean squared error (rMSE) between the model predictions and the ground truths. Please test this function on the full model and the training dataset.

[152]:
```r
# ANSWER BLOCK

# Funtion to calculate RMSE
RMSE_Calc <- function(model,data)
{
prediction <- predict(model,  data)
rmse <- sqrt(mean((data$happiness - prediction)^2))
return(rmse)}
```

[153]:
```r
#Store the RMSE Value by calling the function
rmse_value <- RMSE_Calc(model, train_r)
#Print the RMSE Value.
cat("RMSE:",rmse_value)
```

```
RMSE: 6.672557
```

2.3 Question 3 (2 Marks)

You find the full model complicated and try to reduce the complexity by performing bidirectional stepwise regression with BIC.

Calculate the **rMSE** of this new model with the function that you implemented previously. Is there anything you find unusual? Explain your findings in 100 words.

[154]:
```
#devtools::install_github('IRkernel/repr')
```

[173]:
```
# ANSWER BLOCK
# Perform bidirectional stepwise regression for the linear with BIC and␣
 ↪calculate the RMSE.
bi_model <- step(model,direction = "both",scope=formula(model))
rmse_bi_step_model <- RMSE_Calc(bi_model,train_r)

#bi_model
```

2.4 Question 4 (2 Mark)

Although stepwise regression has reduced the model complexity significantly, the model still contains a lot of variables that we want to remove. Therefore, you are interested in lightweight linear regression models with ONLY TWO predictors. Write a script to automatically find the best lightweight model which corresponds to the model with the least **rMSE** on the training dataset. Compare the **rMSE** of the best lightweight model with the **rMSE** of the full model - `lm.fit` - that you built previously. Give an explanation for these results based on consideration of the predictors involved.

[ ]:
```
# ANSWER BLOCK

# ANSWER BLOCK
# Create lightweightmodel.
best_lightweight_model <- NULL
best_rmse_lightweight <- Inf
# Get list of all predictor combinations.
predictor_dataset <- combn(names(train_r)[!names(train_r) %in% "happiness"], 2)
```

[ ]:
```
# Iterate through each combination of predictors
for (i in 1:ncol(predictor_dataset)) {
predictors <- predictor_dataset[, i]
# Create the lightweight model.
lightweight_model <- lm(happiness ~ ., data = train_r[, c(predictors,␣
 ↪"happiness")])
# Lightweightmodel RMS
rmse_lightweight <- calculate_RMSE(lightweight_model, train_r)
# Check for the lower RMS value amobg the two
if (rmse_lightweight  < best_rmse_lightweight)
{best_lightweight_model <- lightweight_model
 best_rmse_lightweight <- rmse_lightweight}}
```

[ ]:
```r
# RMSE for the lightweight model.
cat("RMSE for the best lightweight model :", best_rmse_lightweight)
# RMSE of the linear model.
cat("\nRMSE for the full linear model :", rmse_value)
```

Compare both the RMSE values of the lightweight and full models above. We see that the RMSE value for the light weight value is higher. This maybe because the significant predictors are not used in the lightweight model.

The full model with max number of predictors is providing better fit , as the model is better at predicting with more number of significant predictors. It is very useful to get the list of significant predictors to create a model which can have the best fit corresponding to lower RMSE values.

2.4.1 ANSWER (TEXT)

2.5 Question 5 (Libraries are allowed) (40 Marks)

As a Data Scientist, one of the key tasks is to build models **most appropriate/closest** to the truth; thus, modelling will not be limited to the aforementioned steps in this assignment. To simulate for a realistic modelling process, this question will be in the form of a Kaggle competition among students to find out who has the best model.

Thus, you **will be graded** by the **rMSE** performance of your model, the better your model, the higher your score. Additionally, you need to describe/document your thought process in this model building process, this is akin to showing your working properly for the mathematic sections. If you don't clearly document the reasonings behind the model you use, we will have to make some deductions on your scores.

This is the video tutorial on how to join any Kaggle competition.

When you optimize your model's performance, you can use any supervised model that you know and feature selection might be a big help as well. Check the non-exhaustive set of R functions relevant to this unit for ideas for different models to try.

Note Please make sure that we can install the libraries that you use in this part, the code structure can be:

```
install.packages("some package", repos='http://cran.us.r-project.org')

library("some package")
```

Remember that if we cannot run your code, we will have to give you a deduction. Our suggestion is for you to use the standard `R version 3.6.1`

You also need to name your final model `fin.mod` so we can run a check to find out your performance. A good test for your understanding would be to set the previous **BIC model** to be the final model to check if your code works perfectly.

[ ]:
```r
#Load the required libraries.
library(tidyverse)
library(dplyr)
library(plotly)
library(xgboost)
```

```
library(performance)
library(visreg)
library(fastDummies)
library(caret)
library(nnet)
library(neuralnet)
library(xgboost)

library(tidymodels)
#library(adagio)
library(mlr3)
library(mlr3tuning)
library(mlr3learners)
library(mlr3verse)
library(caTools)
library(catboost)
library(lightgbm)
```

[ ]:
```
#Dummy encoding.
#Prepare the data for the model by dummy encoding all the categorical values.
train_model <- fastDummies::dummy_cols(train_r, select_columns =␣
 ↪c("gender","doYouFeelASenseOfPurposeAndMeaningInYourLife104",
                                                                ␣
 ↪"howDoYouReconcileSpiritualBeliefsWithScientificOrRationalThinki",
                                                                ␣
 ↪"howOftenDoYouFeelSociallyConnectedWithYourPeersAndFriends",␣
 ↪"doYouHaveASupportSystemOfFriendsAndFamilyToTurnToWhenNeeded",
                                                                ␣
 ↪"howOftenDoYouParticipateInSocialActivitiesIncludingClubsSportsV","doYouFeelComfortableEnga
                                                                ␣
 ↪"doYouFeelASenseOfPurposeAndMeaningInYourLife105"),
                                        remove_first_dummy = TRUE)

#Do the dummy encoding for the test data as well.
test_model <- fastDummies::dummy_cols(test_r, select_columns =␣
 ↪c("gender","doYouFeelASenseOfPurposeAndMeaningInYourLife104",
                                                              ␣
 ↪"howDoYouReconcileSpiritualBeliefsWithScientificOrRationalThinki",
                                                              ␣
 ↪"howOftenDoYouFeelSociallyConnectedWithYourPeersAndFriends",␣
 ↪"doYouHaveASupportSystemOfFriendsAndFamilyToTurnToWhenNeeded",
                                                              ␣
 ↪"howOftenDoYouParticipateInSocialActivitiesIncludingClubsSportsV","doYouFeelComfortableEnga
                                                              ␣
 ↪"doYouFeelASenseOfPurposeAndMeaningInYourLife105"),
                                      remove_first_dummy = TRUE)
```

```r
head(train_model)
```

[ ]:
```r
#Income and Height mapping.
#Convert the 'income' and Height -␣
 ↪"whatIsYourHeightExpressItAsANumberInMetresM" variables from category to a␣
 ↪numerical value
#based on it's 'Group mean value', eg- if income is "0-10k" we set it as 5,000␣
 ↪which is it's mean value.

income_mapping <- list(
  "0 - 10k" = 5000,
  "10k - 15k" = 12500,
  "15k - 20k" = 17500,
  "20k - 50k" = 35000,
  "50k - 80k" = 65000,
  "80k - 120k" = 100000,
  "120k - 150k" = 135000,
  "150k - 200k" = 175000,
  "200k above" = 200000
)

#Train data income mapping.
train_model$income_numeric <- sapply(train_model$income, function(x)␣
 ↪income_mapping[[x]])
#train_model5$income_numeric <- as.numeric(unlist(train_model5$income_numeric))

#Test data income mapping.
test_model$income_numeric <- sapply(test_model$income, function(x)␣
 ↪income_mapping[[x]])

#Height mapping.

height_mapping <- list(
  "140 - 150" = 145,
  "150 - 155" = 152.5,
  "155 - 160" = 157.5,
  "160 - 165" = 162.5,
  "165 - 170" = 167.5,
  "170 - 175" = 172.5,
  "175 - 180" = 177.5,
  "180 - 185" = 182.5,
  "185 - 190" = 187.5,
  "190 above" = 190
)

#Train data height mapping.
```

```r
train_model$height_numeric <-␣
 ↪sapply(train_model$whatIsYourHeightExpressItAsANumberInMetresM, function(x)␣
 ↪height_mapping[[x]])

#Test data height mapping.
test_model$height_numeric <-␣
 ↪sapply(test_model$whatIsYourHeightExpressItAsANumberInMetresM, function(x)␣
 ↪height_mapping[[x]])
```

[ ]:
```r
#Remove Unwanted COlumns.
train_boost <- train_model[, -(1:10)]
#Convert the rest of the columns of 'train_boost' dataset to numeric.
train_boost[] <- sapply(train_boost, as.numeric)
```

[ ]:
```r
#Check for outliers in the Happiness Column.
#Create a boxplot for the column of interest
boxplot(train_boost$happiness, main = "Boxplot of Happiness", ylab = "Value")

# Calculate mean and standard deviation of target variable -'happiness'.
mean_happiness <- mean(train_boost$happiness, na.rm = TRUE)
sd_happiness <- sd(train_boost$happiness, na.rm = TRUE)

# Calculate z-scores on the happiness distribution based on mean and SD.
z_scores <- (train_boost$happiness - mean_happiness) / sd_happiness

#Define the threshold for removing outlier, SD=3.
threshold <- 3

# Identify outliers
outliers <- train_boost$happiness[abs(z_scores) > threshold]
outliers
```

As you can see from above we have only one outlier in the target column 'happiness' with a value of -43.9638, we will not be removing this as I have tried removing it and fitting the model which lead to worse RMSE scores than we get by keeping it. Hence we will be keeping the outlier for better performance of the model.

[ ]:
```r
#Split the dataset into 80:20 for testing and training purpose respectively.
set.seed(24)  # Setting seed for reproducability.
split_boost <- caret::createDataPartition(train_boost$alwaysAnxious, p = 0.8,␣
 ↪list=FALSE)

#Make the train-test split.
train_xgboost <- train_boost[split_boost,]
test_xgboost <- train_boost[-split_boost,]
```

```
[ ]: set.seed(24) # Set seed for reproducability.
     #XGBoost applied to regression.
     # Grid Parameters for xgb-tuning.
     grid_tune <- expand.grid(
       nrounds=c(800),                # No of rounds that the model needs to run.
       max_depth = c(3),              # Depth of the tree.
       eta=c(0.05), gamma = c(0.05), #The larger the gamma , the more␣
      ↪converservative/generlised the model is going to be.
       colsample_bytree = c(0.85),    #Lesser value colsample_bytree of prevent␣
      ↪overfitting.
       min_child_weight = c(0.9),
       subsample = c(0.9))            #The model takes a ratio of the total columns␣
      ↪for each iteration/round.
```

We have finetuned the gridtune above after running numerous iterations with different values of nrounds,max_depth,eta,colsample_bytree, min_child_weight and subsample and found that the above parameters to be the besttune parameters. We have previously trained the model using the below grid_tune values.

grid_tune <- expand.grid(nrounds=c(800,1000,1500,2000), max_depth = c(2,3,4,5,6), #objective = "reg:squarederror", eta=c(0.05,0.1,0.15,0.2), gamma = c(0.05,0.1), colsample_bytree = c(0.85,0.9,0.95), min_child_weight = c(0.8,0.9,0.95), subsample = c(0.85,0.9,0.95)).

```
[ ]: #The model was finetuned using the below grid parameters.


     #grid_tune <- expand.grid(
     #  nrounds=c(800,1000,1500,2000),
     #  max_depth = c(2,3,4,5,6),
     #  eta=c(0.05,0.1,0.15,0.2), gamma = c(0.05,0.1),
     #  colsample_bytree = c(0.85,0.9,0.95),
     #  min_child_weight = c(0.8,0.9,0.95),
     #  subsample = c(0.85,0.9,0.95)).
```

```
[ ]: ### The below is a NOTE, no need to run it.
     #The best tuned Parameters for Xgboost that we got were the below:-

     xgb_tune$bestTune
     **  nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
        800         3 0.05  0.05                0.85                0.9       0.9**
     ###
```

```
[ ]: #Set the cross validation parameters for XG_Boost.
     train_control <- trainControl(method="cv",number=3,verboseIter =␣
      ↪FALSE,allowParallel = TRUE)

     #Create the training and testing data for the model.
```

```
x <- subset(train_boost, select = -happiness)
y <- train_boost$happiness
```

[ ]:
```
set.seed(24)
xgb_tune <- train(x=x,
                  y=y,
                  trControl=train_control, #Pass the train control as defined above.
                  tuneGrid=grid_tune,      #Pass the tuned grid parameters as train control as defined above.
                  method="xgbTree",
                  objective = "reg:squarederror",
                  #early_stopping_rounds=10,
                  verbose=FALSE,  # Change to FALSE to not get the training console messages.
                  set.seed(24))   # Set seed for reporducability.
xgb_tune
```

XGBoost is chosen above for our regression anaylsis as it is robust on multi-class(numerical and categorical) data handling, easy to tune and built in regularisation features and is not prone to overfitting.

I have also built a number of linear models before using Xgboost and got a better score in XGBoost and hence going ahead with it.

[167]:
```
#Predict the splitted test data and calculate the RMSE for it.
predictions <- predict(xgb_tune, test_xgboost)

#Print the RMSE.
#This is just to check the RMSE on sample is close to zero or not..
RMSE(predictions, test_xgboost$happiness)
```

1.23836349594447

[168]:
```
# Build your final model here, use additional coding blocks if you need to
fin.mod <- xgb_tune
```

[169]:
```
# Create the test data for model prediction.
test_model_cut <- test_model[, -(1:10)]
```

[170]:
```
# Load in the test data.
#test <- read.csv("regression_test.csv")
test <- test_model_cut
# If you are using any packages that perform the prediction differently, please change this line of code accordingly.
pred.label <- predict(fin.mod, test)
# put these predicted labels in a csv file that you can use to commit to the Kaggle Leaderboard
```

```
write.csv(
    data.frame("RowIndex" = seq(1, length(pred.label)), "Prediction" = pred.
 ↪label),
    "RegressionPredictLabel.csv",
    row.names = F
)
```

[171]:
```
#Check the prediction outputs.
pred.label
```

1. -10.8883171081543 2. -27.3217811584473 3. 16.1269836425781 4. 14.9535083770752
5. 2.48074102401733 6. 28.9451084136963 7. 11.8567457199097 8. 8.02546691894531
9. 12.0862770080566 10. -1.5354038476944 11. 2.58320546150208 12. -14.9477729797363
13. -6.60607814788818 14. 28.6163902282715 15. 4.33915710449219 16. -7.21403455734253
17. 2.85440897941589 18. -14.963641166687 19. -3.75865268707275 20. -13.4068012237549
21. -20.1677913665771 22. -8.88114261627197 23. -7.8122820854187 24. 14.7771816253662
25. -7.88863611221313 26. 6.94033193588257 27. -16.8186454772949 28. -8.17624378204346
29. 3.34523940086365 30. -13.5519561767578 31. 6.68086910247803 32. 10.8345222473145
33. 7.02801084518433 34. -7.67852258682251 35. -2.84179210662842 36. -7.28657197952271
37. 9.80325031280518 38. 11.6161584854126 39. -15.2007265090942 40. 30.4065589904785
41. -17.3048934936523 42. 17.7602672576904 43. -17.0334739685059 44. -17.2007675170898
45. 0.527075111865997 46. -7.34668016433716 47. 4.09672164916992 48. -18.2491569519043
49. -9.82156848907471 50. -4.33360958099365 51. -9.46744155883789 52. 21.838306427002
53. 12.7477960586548 54. -5.6918511390686 55. 21.1182670593262 56. -21.0869026184082
57. -9.57925224304199 58. 19.389799118042 59. 1.75990200042725 60. -8.15780544281006
61. 13.5988597869873 62. -9.89347171783447 63. -8.17624378204346 64. 29.133810043335
65. -0.787096679210663 66. 19.9617156982422 67. 23.0419902801514 68. -17.7292385101318
69. 7.05905294418335 70. -10.9103937149048 71. 29.6210670471191 72. 31.8343715667725
73. 33.601863861084 74. -4.87533855438232 75. -5.5538477897644 76. 4.13018798828125
77. -5.60927104949951 78. -3.701096534729 79. -5.76682329177856 80. 5.85784912109375
81. -17.1170539855957 82. -0.947741508483887 83. 24.611457824707 84. -4.04473543167114
85. 2.03597974777222 86. -11.4526119232178 87. -16.2340602874756 88. 8.88313293457031
89. -26.0855274200439 90. -14.1710844039917

[ ]:
```
## PLEASE DO NOT ALTER THIS CODE BLOCK, YOU ARE REQUIRED TO HAVE THIS CODE␣
 ↪BLOCK IN YOUR JUPYTER NOTEBOOK SUBMISSION
## Please skip (don't run) this if you are a student
## For teaching team use only

tryCatch(
    {
        source("../supplimentary.R")
    },
    error = function(e){
        source("supplimentary.R")
    }
)
```

```
truths <- tryCatch(
    {
        read.csv("../regression_test_label.csv")
    },
    error = function(e){
        read.csv("regression_test_label.csv")
    }
)


RMSE.fin <- rmse(pred.label, truths$x)
cat(paste("RMSE is", RMSE.fin))
```

3 Part 2 Classification (50 Marks)

A few thousand people were questioned in a life and wellbeing survey to build a model to predict happiness of an individual, but this time we want to predict a categorical score for perfect mental health, rather than a continuous score. You need to build 5-class classification models to optimally predict the variable in the survey dataset called 'perfectMentalHealth' based on any, or all, of the other survey question responses.

You have been provided with two datasets, `classification_train.csv` and `classification_test.csv`. Using these datasets, you hope to build a model that can predict 'perfectMentalHealth' using the other variables. `classification_train.csv` comes with the ground-truth target label (i.e. 'perfectMentalHealth' happiness classes) whereas `classification_test.csv` comes with independent variables (input information) only.

On the order of around 70 survey questions have been converted into predictor variables that can be used to predict 'perfectMentalHealth'. We do not list all the predictor names here, but their names given in the data header can clearly be linked to the survey questions. E.g. the predictor variable 'iDontFeelParticularlyPleasedWithTheWayIAm' corresponds to the survey question 'I don't feel particularly pleased with the way I am.'

This question will also be in the form of a Kaggle competition among students to find out who has the best model.

[156]:
```
# Load in the train and test classification data.
train_c <- read_csv('/kaggle/input/classification/classification_train.csv')
test_c <- read_csv('/kaggle/input/classification/classification_test.csv')

head(train_c)
```

```
Rows: 500 Columns: 43
  Column specification

Delimiter: ","
chr (10): gender, income, whatIsYourHeightExpressItAsANumberInMetresM,
doYou…
```

```
dbl (33): perfectMentalHealth, alwaysAnxious, alwaysStressed,
alwaysAccounta…

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Rows: 95 Columns: 42
 Column specification

Delimiter: ","
chr (10): gender, income, whatIsYourHeightExpressItAsANumberInMetresM,
doYou…
dbl (32): alwaysAnxious, alwaysStressed,
alwaysAccountableAndResponsibleForY…

 Use `spec()` to retrieve the full column specification for this
data.
 Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

A tibble: 6 × 43

gender <chr>	income <chr>	whatIsYourHeightExpressItAsANumberInMetresM <chr>	doYouFeelASenseC <chr>
Female	10k - 15k	155 - 160	Yes
Female	120k - 150k	165 - 170	Yes
Male	0 - 10k	160 - 165	Yes
Male	80k - 120k	160 - 165	Yes
Female	120k - 150k	170 - 175	Yes
Female	0 - 10k	160 - 165	Yes

[158]:
```
#Onehot Dummy encoding.
#Prepare the data for OHE of all the categorical variables in test and train␣
 ↪datasets.
xgb_train_OHE <- fastDummies::dummy_cols(train_c, select_columns =␣
 ↪c("gender","doYouFeelASenseOfPurposeAndMeaningInYourLife104",
                                                                  ␣
 ↪"howDoYouReconcileSpiritualBeliefsWithScientificOrRationalThinki",
                                                                  ␣
 ↪"howOftenDoYouFeelSociallyConnectedWithYourPeersAndFriends",␣
 ↪"doYouHaveASupportSystemOfFriendsAndFamilyToTurnToWhenNeeded",
                                                                  ␣
 ↪"howOftenDoYouParticipateInSocialActivitiesIncludingClubsSportsV","doYouFeelComfortableEnga
                                                                  ␣
 ↪"doYouFeelASenseOfPurposeAndMeaningInYourLife105"),
                                        remove_first_dummy = TRUE)
```

```
xgb_test_OHE <- fastDummies::dummy_cols(test_c, select_columns =␣
 ↪c("gender","doYouFeelASenseOfPurposeAndMeaningInYourLife104",
                                                                  ␣
 ↪  "howDoYouReconcileSpiritualBeliefsWithScientificOrRationalThinki",
                                                                  ␣
 ↪  "howOftenDoYouFeelSociallyConnectedWithYourPeersAndFriends",␣
 ↪"doYouHaveASupportSystemOfFriendsAndFamilyToTurnToWhenNeeded",
                                                                  ␣
 ↪ ␣
 ↪"howOftenDoYouParticipateInSocialActivitiesIncludingClubsSportsV","doYouFeelComfortableEnga
                                                                  ␣
 ↪  "doYouFeelASenseOfPurposeAndMeaningInYourLife105"),
                                          remove_first_dummy = TRUE)

head(xgb_train_OHE)
```

A tibble: 6 × 62

gender <chr>	income <chr>	whatIsYourHeightExpressItAsANumberInMetresM <chr>	doYouFeelASenseO <chr>
Female	10k - 15k	155 - 160	Yes
Female	120k - 150k	165 - 170	Yes
Male	0 - 10k	160 - 165	Yes
Male	80k - 120k	160 - 165	Yes
Female	120k - 150k	170 - 175	Yes
Female	0 - 10k	160 - 165	Yes

[160]:
```
#Income and Height mapping for train and test data.
#Convert the 'income' and Height -␣
 ↪"whatIsYourHeightExpressItAsANumberInMetresM" variables from category to a␣
 ↪numerical value
#based on it's 'Group mean value', eg- if income is "0-10k" we set it as 5,000␣
 ↪which is it's mean value.

income_mapping <- list(
  "0 - 10k" = 5000,
  "10k - 15k" = 12500,
  "15k - 20k" = 17500,
  "20k - 50k" = 35000,
  "50k - 80k" = 65000,
  "80k - 120k" = 100000,
  "120k - 150k" = 135000,
  "150k - 200k" = 175000,
  "200k above" = 200000
)

#Train data income mapping.
xgb_train_OHE$income_numeric <- sapply(xgb_train_OHE$income, function(x)␣
 ↪income_mapping[[x]])
```

```
#train_model5$income_numeric <- as.numeric(unlist(train_model5$income_numeric))

#Test data income mapping.
xgb_test_OHE$income_numeric <- sapply(xgb_test_OHE$income, function(x)␣
 ↪income_mapping[[x]])

#Height mapping.

height_mapping <- list(
  "140 - 150" = 145,
  "150 - 155" = 152.5,
  "155 - 160" = 157.5,
  "160 - 165" = 162.5,
  "165 - 170" = 167.5,
  "170 - 175" = 172.5,
  "175 - 180" = 177.5,
  "180 - 185" = 182.5,
  "185 - 190" = 187.5,
  "190 above" = 190
)

#Train data income mapping.
xgb_train_OHE$height_numeric <-␣
 ↪sapply(xgb_train_OHE$whatIsYourHeightExpressItAsANumberInMetresM,␣
 ↪function(x) height_mapping[[x]])

#Test data income mapping.
xgb_test_OHE$height_numeric <-␣
 ↪sapply(xgb_test_OHE$whatIsYourHeightExpressItAsANumberInMetresM, function(x)␣
 ↪height_mapping[[x]])
```

[161]:
```
#Remove unwanted columns
#As we have done the conversion to numeric through income mapping and height␣
 ↪mapping above we can remove the "income" and␣
 ↪"whatIsYourHeightExpressItAsANumberInMetresM" columns.
#As we have done the One Hot- Dummy encoding for columns 1- Gender and column␣
 ↪numbers 4-10 we can remove them as well.
xgb_train_OHE <- xgb_train_OHE[, -(1:10)]
xgb_test_OHE <- xgb_test_OHE[, -(1:10)]
```

[162]:
```
#Convert the target variable 'perfectMentalHealth' to a factor.
xgb_train_OHE$perfectMentalHealth <- as.
 ↪factor(xgb_train_OHE$perfectMentalHealth)
```

[163]:
```
# Set seed for reproducibility of the code.
set.seed(10)
```

```
#Set Parameters for Train Control for the XGBoost Model.
train_control_c <- trainControl(method="cv",number=5,verboseIter =␣
↪TRUE,allowParallel = TRUE)

#Set Parameters for grid_tune of XGBoost model.
grid_tune_c <- expand.grid(
  nrounds=c(200),
  max_depth = c(10),
  eta=c(0.05), gamma = c(0.2), #The larger the gamma , the more converservative/
↪generlised the model is going to be.
  colsample_bytree = c(0.9), #Lesser to prevent overfitting.
  min_child_weight = c(5),
  subsample = c(0.85))
```

- I have previously run the Xgboost classifier with a number of hyperparameters as found below and selected the Best-Tune hyperparamters to be used in the 'grid_tune_c' above.
- The parameters to train and fix the value of grid_tune_c were as below in the commented code , the grid_cv model took about 8 hours to run in my system.

[ ]:
```
#Set Parameters for grid_tune of XGBoost model.
#grid_tune_c <- expand.grid(
 # nrounds=c(200,400,600),
 # max_depth = c(6,8,10,12),
 # eta=c(0.05,0.1,0.2), gamma = c(0.2,0.4), #The larger the gamma , the more␣
↪converservative/generlised the model is going to be.
 # colsample_bytree = c(0.9,1), #Lesser to prevent overfitting.
 # min_child_weight = c(5,10,15),
 # subsample = c(0.85,0.9,1.0))
```

Reasons for choosing XGBoost for classifier is because XGBoost is an good choice for classification :- * It includes built-in regularization techniques (L1 and L2) to prevent overfitting and supports parallel processing. * Though it takes a lot of time to fine-tune it gives good results. * It handles various data types, including numerical, categorical, and missing values, making it versatile. * XGBoost is robust to any outliers in data providing good accuracy.

[164]:
```
#Train the XGBoost classifier model
set.seed(10)
xgb_classifier <- train(x = subset(xgb_train_OHE, select =␣
↪-perfectMentalHealth),
                        y = xgb_train_OHE$perfectMentalHealth,
                        trControl= train_control_c,
                        tuneGrid=grid_tune_c,
                        method="xgbTree",
                        eval_metric = "merror",          #Evaluation metric␣
↪multi-class classification error rate.
                        objective = 'multi:softprob',    #"multi:softprob"␣
↪objective used for multi-class classification problems.
```

```
                        verbose = TRUE,                 #Set the console output␣
 ↪to False, change if needed.
                        set.seed(10))                    #Set seed for␣
 ↪reproducability.
xgb_classifier
```

```
+ Fold1: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
```

```
Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"
```

```
- Fold1: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
+ Fold2: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
```

```
Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"
```

```
- Fold2: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
+ Fold3: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
```

```
Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"
```

```
- Fold3: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
+ Fold4: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
```

```
Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
```

```
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"

- Fold4: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
+ Fold5: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85

Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"

- Fold5: nrounds=200, max_depth=10, eta=0.05, gamma=0.2, colsample_bytree=0.9,
min_child_weight=5, subsample=0.85
Aggregating results
Fitting final model on full training set

Warning message:
"Setting row names on a tibble is deprecated."
Warning message in check.booster.params(params, …):
"The following parameters were provided multiple times:
        objective
  Only the last value for each of them will be used.
"

eXtreme Gradient Boosting

500 samples
 53 predictor
  5 classes: '-2', '-1', '0', '1', '2'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 399, 401, 399, 400, 401
Resampling results:

  Accuracy   Kappa
  0.3477468  0.1467144

Tuning parameter 'nrounds' was held constant at a value of 200
Tuning

Tuning parameter 'min_child_weight' was held constant at a value of 5
```

```
Tuning parameter 'subsample' was held constant at a value of 0.85
```

[165]:
```r
# Load in the train and test classification data.
#train <- read_csv('../kaggle/input/classification/classification_train.csv')
#test <- read_csv('../kaggle/input/classification/classification_test.csv')

# Build your final model here, use additional coding blocks if you need to
fin.mod <- xgb_classifier

# If you are using any packages that perform the prediction differently, please␣
 ↪change this line of code accordingly.
pred.label <- predict(fin.mod, xgb_test_OHE)
# put these predicted labels in a csv file that you can use to commit to the␣
 ↪Kaggle Leaderboard
write.csv(
    data.frame("RowIndex" = seq(1, length(pred.label)), "Prediction" = pred.
 ↪label),
    "ClassificationPredictLabel.csv",
    row.names = F
)
```

[166]:
```r
#CHECK THE PREDICTED OUTPUTS.
pred.label
```

1. 1 2. 1 3. 1 4. 1 5. 1 6. 1 7. -2 8. -1 9. 1 10. -2 11. -1 12. 1 13. 1 14. 1 15. 0 16. 0 17. 0 18. 1 19. 0 20. 2 21. 0 22. 2 23. 2 24. 0 25. 0 26. 1 27. -1 28. -1 29. 0 30. 0 31. 2 32. 0 33. 1 34. -1 35. 0 36. 0 37. 0 38. 2 39. 2 40. 0 41. 1 42. 1 43. -1 44. 1 45. 2 46. 0 47. -1 48. 0 49. 0 50. 0 51. 0 52. 0 53. 1 54. 0 55. -1 56. 0 57. 0 58. -2 59. 1 60. 1 61. 0 62. -1 63. -1 64. 0 65. 0 66. 1 67. 0 68. 0 69. 1 70. -1 71. 0 72. 1 73. -1 74. 1 75. 0 76. 0 77. 0 78. 1 79. 1 80. 0 81. 0 82. 0 83. 0 84. 0 85. 0 86. 0 87. 1 88. 2 89. 0 90. 0 91. 0 92. 0 93. -1 94. 1 95. 1

Levels: 1. '-2' 2. '-1' 3. '0' 4. '1' 5. '2'

[ ]:
```r
## PLEASE DO NOT ALTER THIS CODE BLOCK, YOU ARE REQUIRED TO HAVE THIS CODE␣
 ↪BLOCK IN YOUR JUPYTER NOTEBOOK SUBMISSION
## Please skip (don't run) this if you are a student
## For teaching team use only

truths <- tryCatch(
    {
        read.csv("../classification_test_label.csv")
    },
    error = function(e){
        read.csv("classification_test_label.csv")
    }
)

f1_score <- F1_Score(truths$x, pred.label)
```

```
cat(paste("f1_score is", f1_score))
```