MATH5743M: Statistical Learning:Assignment2

Surapot Nonpassopon

2024-04-27

Brexit Forecast by using Statistical Modeling.

Data understanding and preprocessing.

1. First, the data set file "Brexit.csv" must be inputted into R; after that, it could use function head() to observe the first five rows of data.

```
# call the necessary dataset
library(tidyverse)
library(ggplot2)
```

```
# input the data
setwd("~/Desktop/Semaster2/Stat Learning/Assignment/A2/Assignment2")
brexit <- read.csv('brexit.csv', stringsAsFactors = FALSE, encoding ='UTF-8')

# data observation
head(brexit)
```

```
##        abc1   notBornUK medianIncome medianAge withHigherEd voteBrexit
## 1 0.1336406 0.012605042    0.2525773 0.5000000   0.08552632       TRUE
## 2 0.1290323 0.113445378    0.1082474 0.2727273   0.11184210       TRUE
## 3 0.1612903 0.004201681    0.1288660 0.6363636   0.11842105       TRUE
## 4 0.3225806 0.046218487    0.2268041 0.4545455   0.21710526       TRUE
## 5 0.3456221 0.058823529    0.2010309 0.5454545   0.24342105       TRUE
## 6 0.2211982 0.012605042    0.2319588 0.4545455   0.09210526       TRUE
```

The explanation of each variable is provided in Table 1 below.

Table 1: The explanation of dataset variables.

Variable	Explanation
Abc1	The proportion of individuals who are in the ABC1 social classes which is the middle to upper class.
medianIncome	The median income of all people.
medianAge	The median age of all people.
withHigherEd	The proportion of people who have the level education at university level.
notBornUK	The proportion of people who were born outside the UK.

2. The missing value has been removed before the processing.

```
# remove the missing value
brexit <- na.omit(brexit)
```

3. After that, use the R function str () to understand the type of data in the dataset. All variables are numeric, while the "voteBrexit" is the logic that represents binary values TRUE and FALSE.

```r
# understand the dataset
str(brexit)
```

```
## 'data.frame':    344 obs. of  6 variables:
##  $ abc1        : num  0.134 0.129 0.161 0.323 0.346 ...
##  $ notBornUK   : num  0.0126 0.1134 0.0042 0.0462 0.0588 ...
##  $ medianIncome: num  0.253 0.108 0.129 0.227 0.201 ...
##  $ medianAge   : num  0.5 0.273 0.636 0.455 0.545 ...
##  $ withHigherEd: num  0.0855 0.1118 0.1184 0.2171 0.2434 ...
##  $ voteBrexit  : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

4. This is followed by using the function summary () to recheck the data that is already normalised, so the value should contain between 0 and 1.

```r
# check the structure of data
summary(brexit)
```

```
##       abc1          notBornUK        medianIncome      medianAge
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2857   1st Qu.:0.04622   1st Qu.:0.1804   1st Qu.:0.3636
##  Median :0.4194   Median :0.09244   Median :0.2577   Median :0.5455
##  Mean   :0.4315   Mean   :0.16068   Mean   :0.2923   Mean   :0.5103
##  3rd Qu.:0.5622   3rd Qu.:0.18067   3rd Qu.:0.3814   3rd Qu.:0.6818
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##   withHigherEd    voteBrexit
##  Min.   :0.0000   Mode :logical
##  1st Qu.:0.1908   FALSE:107
##  Median :0.2895   TRUE :237
##  Mean   :0.3162
##  3rd Qu.:0.4095
##  Max.   :1.0000
```

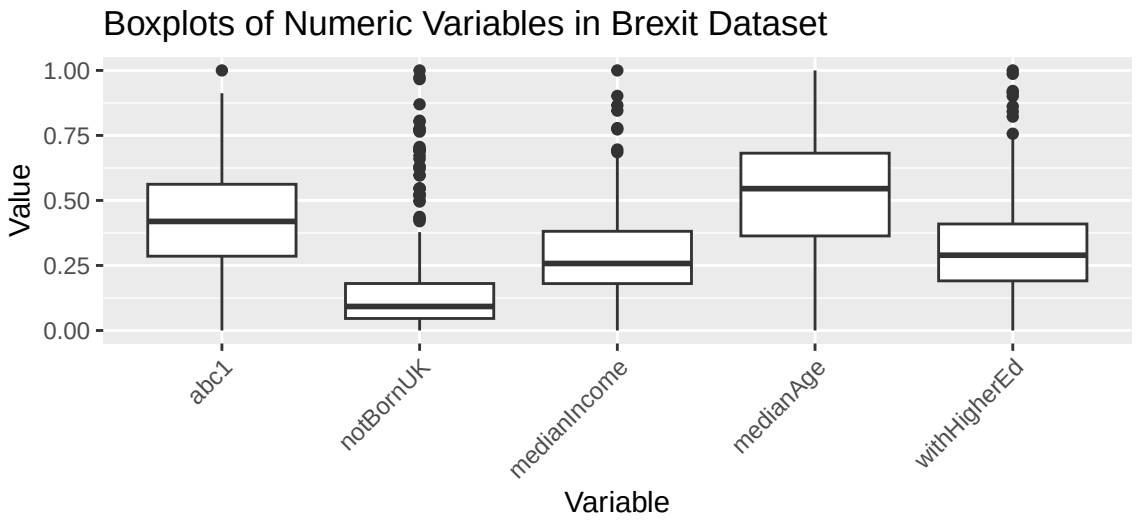

Figure 1: Boxplots of Numeric Variables in Brexit Dataset

5. This is followed by visualising the data to see its distribution. The box plot visualisation suggests that after normalising the data, the structure of 'notBornUK', 'medianIncome', and 'withHigherEd' still seems to be right-skewed and contain outliers. Meanwhile, the 'abc1' and 'medianAge' seem to be normally distributed.

Assigment Task

Task1: Use K-means algorithm (kmeans function in R) to cluster the inputs. Validate the optimal number of clusters. Justify your choice. Visualise the clusters and analyse the validated k-means model. Are the inferred clusters associated with or predictive of the outputs?

To perform the k-mean clustering and see the clustering of the data, first, the output variable that wants to be removed or observed, which is the "voteBrexit" variable, needs to be removed from the dataset.

```
# Task1: perform the K-mean clustering
# Exclude the voteBrexit variable that is the output variable that we want to predict
brexit_clustering <- brexit %>%
  select(-voteBrexit)
```

Secondly, before performing k-mean clustering algorithms in R, the value of k needs to be specified first. The value of K, in this case, means that it represents the number of the groups that algorithms will be identified, or cluster based on that variability from data.

```
# setting the centre of clustering K as 2
# because we want to cluster the result in to binary output
set.seed(123456)
k <- 2
```

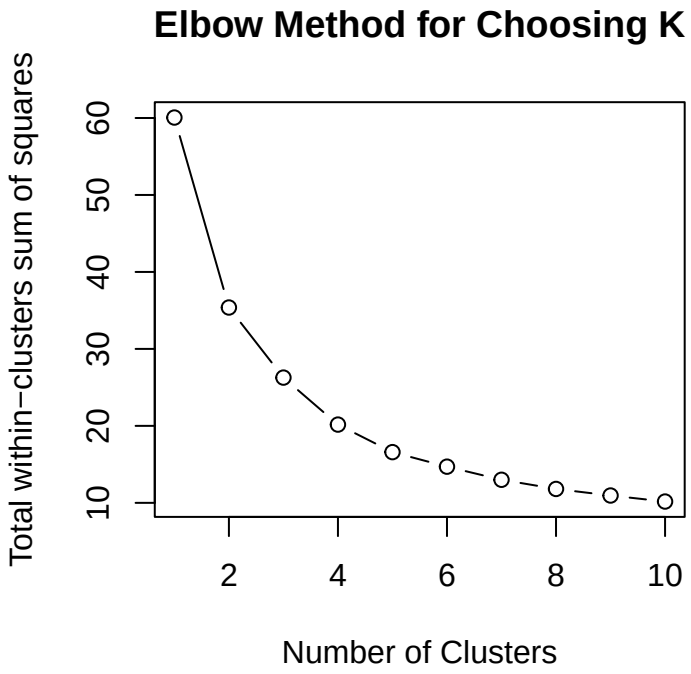

Figure 2: Elbow Method for Choosing K

Justify the results of using K as 2

In this algorithm, the number of K will be set as 2. To justify the result of this:

1. The outcome variable, "vote Brexit," represents the binary variable value, including "true" or "false," to present that more than 50% of people voted to leave or remain. Thus, setting the K as equal to 2 makes sense in classifying it into two distinct categories.
2. Moreover, further study could use the standard method to validate the optimal number of clusters by conducting the elbow plot. This plot could suggest the optimal number of clusters to the sign from a

clear elbow from the graph. Figure 2 is the elbow plot, which shows a clear and significant decrease in elbow sign at k=2 and k=3; therefore, using k = 2 might be a good choice for classifying the binary variable than k=3.

Thus, these two reasons clearly and well-justified the use of k equal to 2 in this k mean clustering process.

Thirdly, when fitting the data with the kmean function in R, the first argument is the Brexit dataset with the output variables(voteBrexit) excluded, and the second argument is the k value, which in this case is set to 2. After processing the k-mean clustering, the result from clustering will be shown below.

```
# process k-mean clustering
kclus <- kmeans(brexit_clustering, centers = k)
print(kclus)
```

```
## K-means clustering with 2 clusters of sizes 71, 273
##
## Cluster means:
##        abc1  notBornUK medianIncome medianAge withHigherEd
## 1 0.6618420 0.42289028    0.5168433 0.3072983    0.5660675
## 2 0.3715501 0.09248315    0.2339224 0.5631036    0.2512531
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   2   1   2   2   2   2   2   2   2   1   2   2   2   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   2   2   2   2   2   2   2   2   2   2   2   2   1   1   2   2   2   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   2   1   2   2   2   2   1   2   2   2   2   2   1   1   1   1   1   2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   1   1   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   1
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   1   1   1   1   1   1   1   2   2   2   2   1   2   2   2   2   1   2   2   2
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   2   2   2   2   2   2   1   2   2   2   2   2   1   2   2   2   2   2   2   2
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   1   2   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   1   2   2   2   2
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   1   2   2   2   2   2
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2
```

```
## 341 342 343 344
##   2   2   2   2
##
## Within cluster sum of squares by cluster:
## [1] 12.38362 22.99400
##  (between_SS / total_SS =  41.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

The result shows that the clusters are grouped into groups, cluster1 and cluster2, with the first group containing 71 elements and the second one containing 273 elements.

Before creating the plot of k-mean clustering, the contingency table is created to see how the output(voteBrexit) relates to the clusters.

```
# Add the cluster to the original dataframe
brexit_kclus <- brexit %>%
  mutate(cluster = kclus$cluster)

# create a contingency table to see the output(voteBrexit) relates to the clusters.
table(brexit_kclus$voteBrexit, brexit_kclus$cluster)
```

```
##
##           1   2
##   FALSE  55  52
##   TRUE   16 221
```

The contingency table comparing the Brexit vote preference to the cluster identified in the dataset:

- Cluster 1 has 71 members, 55 of whom voted "False" to remain in the EU, while the other 16 voted "True" to leave the EU.
- Cluster 2 has 273 members, 52 of whom voted "False" to remain in the EU, while the other 221 voted "True" to leave the EU.

This contingency table suggests that cluster 2 contains the majority of the dataset's people who vote to leave the EU, while the majority of cluster 1 contains the people who want to remain in the EU.

Visualisation

Perform the scatter plots to visualise the clusters and validate the k-means model clustering.

```
# plot k-mean clustering
plot(brexit_clustering, col = kclus$cluster)
```

From the scatter plot matrix in Figure1, the analysis from the validated k-means model is:

1. Cluster Separation: The matrix shows that the data are clustered in two groups (represented by two colours), as setting the K=2 in the setting model at first. The clearly separate 2 groups show that some features better distinguish between the two clusters than others.
2. The impact of features: This matrix shows that features like medianAge might show more distinct groupings than others. This indicates that the age features might significantly impact the clustering process of the two data groups.
3. The overlapping of results: Many features, such as the income rate and the education levels, still significantly overlap after clustering. This suggests that these features alone may not be sufficient to distinctly separate the data into the chosen number of clusters (k=2).

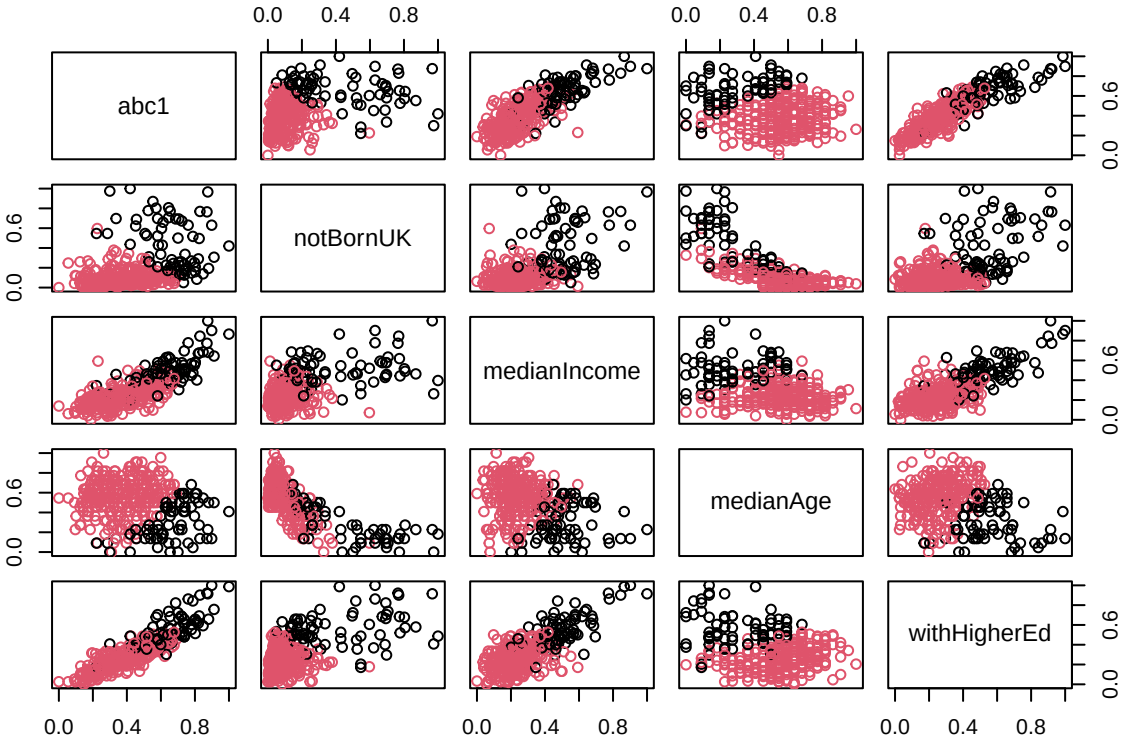

Figure 3: K-Means Clustering analysis on Brexit voting demographics

In conclusion, the k-means clustering visualisation, provided in Figure 3, shows that the k-means clustering model can separate the data into two clusters. Some features effectively separate the data into two groups; however, it still has some overlapping features, suggesting the data are not clearly or well separated into two groups.

Association of inferred clusters with the prediction of the outputs

The inferred clusters of red and black groups from Figure 3 show varying degrees of separation across different variables, which have separate and overlapping results. However, it might be helpful to suggest that this dataset could separate the output into two groups. That indicates that the association with predicting the outputs of voting Brexit, binary variable, is shown only in cluster some of the variables, such as age and income, that seem to separate reasonably well, which might indicate that these variables are good predictors of the binary outcome. However, the overlapping results still imply that the clusters might be predictive for binary outputs but less effective. Therefore, the inferred clusters, as the clusters are two groups, seem to be associated with predicting the outputs with the binary variable.

Task2: Consider a logistic regression model for the data. Use the model with all inputs to find which input variables are relevant to explain the output by interpreting the model. Identify the direction and magnitude of each input effect from the fitted coefficients. Which inputs would you say have strong effects? Order the inputs in terms of decreasing effect. Justify your reasoning.Compare your findings with the plots shown on the Guardian website. Do your findings agree with these plots? Comment on your findings.

Fitting the model of logistic regression

To progress the logistic regression and evaluate the performance of the model this time, the data set has been random and split into 2 sets: the training set at 80% and the test set at 20% of Brexit data.

```
# Task2: fitting the model logistic regression
# Set seed for sampling
set.seed(123)

# Split the dataset of Brexit first to trainning and test set.
# split into training (80 percent),
# and test data (20 percent)
brexit_size <- floor(0.80 * nrow(brexit))
training <- sample(seq_len(nrow(brexit)), brexit_size)
brexit_train <- brexit[training, ]
brexit_test <- brexit[-training, ]
```

After that fitting the model by using the glm () function

1. The first argument is a formula and will be specific by the relevant input and outputs from our models, in this case, setting the output as a vote as "voteBrexit" variable and setting all the variables left as input.
2. The second argument is family setting as the binomial because the likelihood of the logistic regression is the form of the binomial probability that the output is either 0 or 1.
3. The third argument is the dataset that is Brexit dataset.

```
# Fit the logistic model to the training data.
# Specific the glm to perform logistic regression
# setting the output as voteBrexit
# setting the input as allvariable
logis_glm_train <- glm(voteBrexit ~ ., family = binomial, data = brexit_train)
summary(logis_glm_train)
```

```
##
## Call:
## glm(formula = voteBrexit ~ ., family = binomial, data = brexit_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.5456     0.9434  -0.578  0.56300
## abc1          16.4429     3.1406   5.236 1.65e-07 ***
## notBornUK      5.6805     1.9377   2.932  0.00337 **
## medianIncome  -5.2370     2.0677  -2.533  0.01132 *
## medianAge      6.3647     1.5335   4.150 3.32e-05 ***
## withHigherEd -25.7200     3.8321  -6.712 1.92e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 343.25  on 274  degrees of freedom
## Residual deviance: 201.53  on 269  degrees of freedom
## AIC: 213.53
##
## Number of Fisher Scoring iterations: 6
```

Interpreting the logistic regression model results

The strong coefficient with providing in the decreasing order of the logistic model and the direction are provided in Table 2.

Table 2: Summarizes the coefficient that affects the output of the logistic regression model.

Variable	Coefficient	Direction	Standard Error
withHigherEd	25.72	Negative	3.832
abc1	16.443	Positive	3.140
medianAge	6.365	Positive	1.534
notBornUK	5.681	Positive	1.938
medianIncome	5.237	Negative	2.068

The reason that is used to justify the input value that has a strong or weak effect on the output is the magnitude of the coefficient. By the properties of the logistic function based on the equation

$$\phi(x) = \frac{1}{1 + \exp(-x)}$$

When x is the linear combination of the coefficients and the predictor variables, which the coefficient that can be received from the estimate of the model. The large negative number of coefficients can make the $\exp(-x)$ will lead to a very large value; this is the result of the $\phi(x)$ logistic regression tend to be the 0. The 0 in the case of voting Brexit means people vote to remain in the EU. In terms of variables in the Brexit dataset, the withHigherEd is considered as the strong negative variable because the magnitude of the variable of the negative impact of withHigherEd is very high compared to medianIncome that more than around 5 times. Thus, withHigherEd should give the strong negative effect by it high magnitude.

When x is the linear combination of the coefficients and the predictor variables, which the coefficient that can be received from the estimate of the model. The large positive number of coefficients can make the $\exp(-x)$ will lead to a very low value; this is the result of the $\phi(x)$ logistic regression tend to be the 1. The 1 in the case of voting Brexit means people vote to leave the EU. In terms of variables in the Brexit dataset, the abc1 is considered as the strong positive variable because the magnitude of the variable of the positive impact of abc1 is very high compared to notBornUK and medianAge that more than around 3 times. Thus, abc1 should give the strong positive effect by it high magnitude.

The summaries of the direction and magnitude of each input effect have been provided in Table 2. That already orders the magnitude of the effect in the decreasing levels. To elaborate on each variable in the table:

- The "withHigherEd", this variable has the strongest negative impact on the outcome, which makes the outcome more likely to be 0, as said before, which means people who have a higher education might not vote to remain in the EU.
- The "abc1" variable is the second strong impact effect, and this one has a strong positive impact. It could make the outcome more likely to be 1, which means people who are in the ABC1 social classes or highers might vote to leave the EU.

- The "medianAge" variable has a small positive impact on the outcome, which means that people of older ages might vote to leave the EU with a small impact.
- The "notBornUK" variable also has a small positive impact on the outcome variable, which means that people who were born outside the UK might vote to leave the EU with a small impact.
- The "meadianIncome" variable has quite a small negative impact on the outcome, which means people with a high median income might vote to remain in the EU with a small impact.

To summarise, the two significant input variables that impact the output variable are education level, which has a strong negative impact, and social class, which has a strong positive impact. The other three variables (age, income, and birth location) have quite a small impact on the output of those who vote to leave the EU or not.

Compare the findings with the Guardian Websites

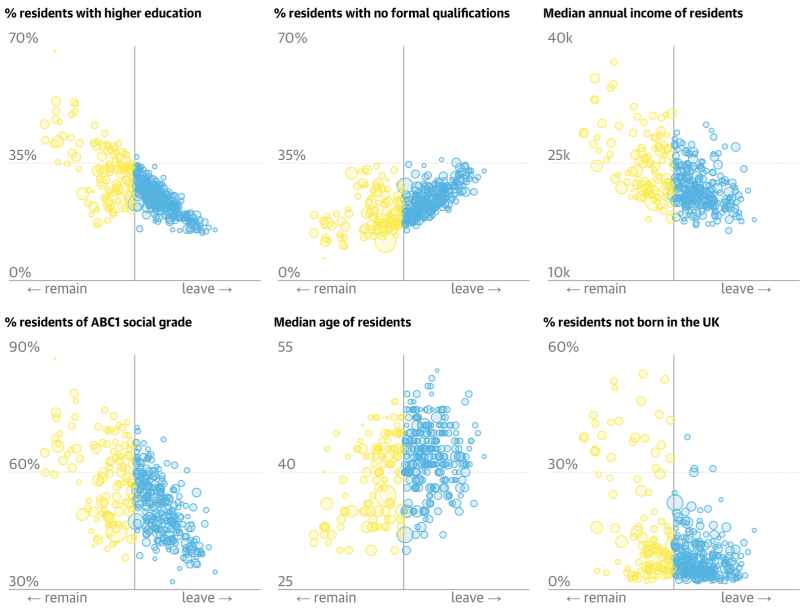

Figure 4: The analysis of Brexit data from the Guardian website.

I agree with this plot in Figure 4 in some features, with the logistic regression model providing the evidence to support it. However, it has some features that are still opposite to the result of the model, so that could be the question about the accuracy of the Guardian website or the wrong of the model.

In terms of education level, the results from the logistic regression model, which shows that people with higher education might vote to remain, are the same as the plot on the Guardian website. The insight that was obtained from the model could be used as support evidence.

In terms of social class, the logistic model results show that people in higher social classes might vote to leave the EU, which seems to be opposite to the plot from the Guardian website, which shows people in high social classes would like to remain in the EU. Thus, this might be the question about how accurate this plot is because it has evidence from the logistic model to argue.

Regarding age, it is also coming to the same conclusion as the logistic regression model and Guardian plot that people getting older might want to leave the EU.

As regards the proportion of residents who were not born in the UK, the result from the logistic regression model argues the Guardian plot, which the Gaudian plot shows people who are not born in the UK would

like to remain in the EU. However, the result from the model suggests people who were born outside the UK might vote to leave the EU.

In terms of income rate, it is coming to the same conclusion as the logistic regression model and Guardian plot: that people who have high income rates might want to remain in the EU.

To summarise, The Guardian plot and models suggest the same result in the features of education level, income rate and the age of people. Nevertheless, there are still some arguments from the model regarding the accuracy of the Gaudian plot on the features of social class and the proportion of residents who were not born in the UK.

Task3: Discuss factors that may affect interpretability of the regression coefficients of the fitted model. Based on your discussion, explain whether you can reliably determine which inputs are relevant for modelling the output and order the input variables based on their relevance in decreasing order. Justify your reasoning. Use BAGGING for logistic regression for Task 2 to analyse variability of the regression coefficients to support your answer.

The factor that may affect the interpretability of the coefficients of the fitted model To discuss the factors that could potentially impact the interpretability of the fitted model regression coefficient. There are some factors that might affect that are:

1. The scale of the variable: If the data set uses the data in different units or scales, sometimes this might affect the coefficient of the logistic regression model. However, this Brexit dataset already scales the value from 0 to 1, which is helpful for fitting the model process.
2. Outliers: The outliers could skew the coefficient of the fitted model away from the true effect of predictors. As this data also has outliers from the distribution in Figure 1 which might affect the coefficients of the fitted model.
3. Sample size: A small sample size could have more variance of that data, which might make it unstable in the estimation of the coefficient.
4. The dataset's correlation: If the dataset has two or more highly correlated variables, it might be challenging to determine each predictor's individual influence, which could lead to the logistic regression model's unstable coefficient.
5. Non-monotonic data: The figure below shows a non-monotonic data pattern that might affect the logistic regression coefficient.

```
# checking the pattern of dataset
plot(brexit_test$withHigherEd, brexit_test$voteBrexit)
```

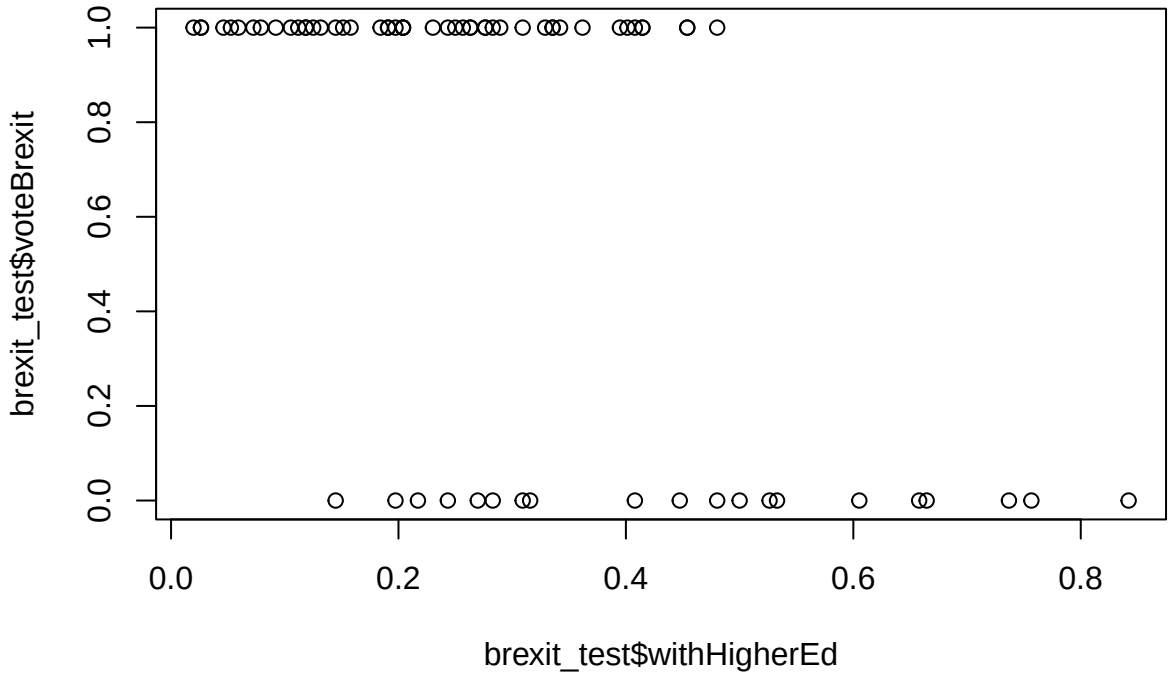

Figure 5: Plot of relationship between higher education and voting behavior

Using the BAGGING for logistic regression to analyse the variability of the regression coefficients

The factor that can affect the instability of the fitted model coefficient could be handled by using the method that is BAGGING for logistic regression; the bagging method is the method to generate the new data from the original dataset by sampling it uniformly with replacement to fitting the model. This process allows the model to be fitted multiple times, which can help stabilise the coefficients by averaging the results across these multiple samples to generate the model that might have the stability of the coefficient.

To progress the BAGGING for logistic regression by making the 1000 bootstrap samples in R as the code below, it needs to create the 1000 different samples training data first, build the logistic regression model for each and make the prediction to test the data, these prediction results for each will be added together to use to find the average of prediction results from 10000 samples. Additionally, the model, before looping the coefficient matrix, has been created to store the value of the coefficient of all models to use to calculate the average of the coefficient value from the 1000 bootstrap sampling.

```
# Task3: fitting the model logistic regression and doing bagging for Bootstrap Aggregating
# doing bagging for Bootstrap Aggregating
# make the 1000 bootstrap of the samples and fitting the logistic model to each
# record the test data and aggregate
bagged_predictions <- rep(0, nrow(brexit_test))
coefficients_matrix <- matrix(NA, nrow = 1000, ncol = length(coef(logis_glm_train)))

for (i in 1:1000) {
  bootstrap_index <- sample(seq_len(nrow(brexit_train)), replace = TRUE)
  bs_data <- brexit_train[bootstrap_index, ]
  # Fit a logistic model to the bootstrap
  bs_model <- glm(voteBrexit ~ ., data = bs_data, family = binomial)
  # Store the coefficients of this model
  coefficients_matrix[i, ] <- coef(bs_model)
  bagged_predictions <- bagged_predictions + predict(bs_model, newdata = brexit_test, type = "response")
}
# Finding the overall
bagged_predictions <- bagged_predictions / 1000

# Calculate the average coefficient for each variable
average_coefficients <- apply(coefficients_matrix, 2, mean, na.rm = TRUE)
print(average_coefficients)
```

```
## [1]  -0.5938537  17.3341760   6.1010042  -5.7908272   6.6519116 -26.8609695
```

After the looping finish, the coefficients of all models, stored in the coefficient, are used to find the average value of all. The results are clearly represented in Table 3 for the clearly seen.

Table 3: Summarizes the coefficients affecting the logistic regression model output. 'Coefficient (Before)' refers to the coefficient of the model before bagging and 'Coefficient (After)' refers to the average coefficient after bagging for 1000 bootstrap aggregations.

Variable	Coefficient (Before)	Coefficient (After)	Direction
withHigherEd	25.720	26.861	Negative
abc1	16.443	17.334	Positive
medianAge	6.365	6.652	Positive
notBornUK	5.681	6.101	Positive
medianIncome	5.237	5.791	Negative

After the ordering, the magnitude effect in the decreasing order is shown in Table 3, and the result suggests the same order as Task 2. The strongest negative direction seems to be the education level, with a coefficient as high as 26.861, which might make the outcome of logistic regression more likely to be 0, which means people who have a higher education might vote to remain in the EU. In term of the strongest positive seems to be social class, which might make the outcome of logistic regression more likely to be 1 which means people who are in the ABC1 social classes or higher might vote to leave the EU. With following by the other 3 variables have quite a small effect, which are age, birth location and income rate, respectively.

Ordering the reliable coefficient to the variance model after bootstrapping 1000 times.

```
# Calculate the average coefficient variance each variable
coefficient_variances <- apply(coefficients_matrix, 2, var, na.rm = TRUE)
print(coefficient_variances)
```

```
## [1]  0.6323373  9.9128032  4.0212352  5.5596736  1.8489332 17.9213606
```

Table 4: Variance of coefficients after 1000 bootstrap samples for logistic regression analysis.

Variable	Variance After 1000 Bootstraps
withHigherEd	17.921
abc1	9.913
medianIncome	5.560
notBornUK	4.021
medianAge	1.849

From the processing of bootstrapping 1000 times, the higher variance suggests that the variables are more unstable and unreliable in this Brexit data, suggesting the withHigherEd coefficient after fitting the bagged-logistic model is the most unstable and unreliable with the highest variance at around 16. Followed by the abc1 approximately 10. Meanwhile, the other three variables, median income, notBornUK and medianAge, have a low variance across the bootstrap samples, suggesting a more reliable and stable variable in the output especially in the "medianAge".

Impact of input, which is relevant for modelling in the output by decreasing order.

In conclusion, the result from Table 3 and Table 4, the variable with have strongest impact on the output and ranking as the [1st ranking] are 'withHigerEd'. However, this variable has the highest unstable coefficient from the highest variance after bootstrapping.

The [2nd ranking] is the "abc1" variable with a high positive impact on the output. However, this variable has an unstable coefficient from the high variance of around 10 after bootstrapping.

The [3rd ranking] is coming with "medianAge" from the coefficient around 6. There are some unstable coefficients when bootstrapping, but it is lower than the 1st and 2nd ranks.

The [4th ranking] is the "notBornUK", which has a small positive effect on the output variable. This variable has a low variance of coefficient in the bootstrapping process, suggesting that this coefficient could be stable and reliable.

The [5th ranking] is the "medianIncome", which has a small negative effect on the output variable. This variable also has a low variance of coefficient in the bootstrapping process, suggesting that this coefficient could be stable and reliable.

Justify the reasons for using the BAGGING for logistic regression.

To summarise, the reason for using BAGGING for logistic regression is to handle the variability of the regression.

1. The Bagging method could reduce the variability of the regression because it averages the coefficients across multiple samples. This Bagging method might be used to make the more stable of the coefficients. To elaborate, the other important thing that can be seen from the table is the change in magnitude after bagging. It can be that the magnitude after bagging is slightly increased, which shows the result after it has been fitting the model with a new, different training set 1000 times and has averaged the magnitude of the coefficient across the multiple samples into a more stable and reliable number.
2. The Bagging method could help reduce the overfitting because the Bagging step trains the model from many random subsets of data, which could help the model better understand the new unseen data.
3. In general, using the bagging method might improve the performance of the model when observing the model evaluation of log-likelihood. Evaluating the log-likelihood value that could be used to show how well the model can predict the actual outcomes indicated by the value close to 0 might be the better model. From the result below, the logistic regression model has a log-likelihood result of -23.341, while the bagged logistic regression model has a log-likelihood result slightly worse at around -23.355. Thus, it could be seen that using the bagging methods in this data set might not improve the performance of the model.

```
# Compared the result before/after Bagging methods.
# Making the prediction the probability of the test data being 1
logis_glm_prediction <- predict(logis_glm_train, newdata = brexit_test, type = 'response')
# Evaluate the predictive log-likelihood
testLL <- sum(log(logis_glm_prediction[brexit_test$voteBrexit == 1])) +
  sum(log(1 - logis_glm_prediction[brexit_test$voteBrexit == 0]))
print(testLL)
```

```
## [1] -23.34117
```

```
# Evaluate  predictive likelihood of the bagging model
bstestLL <- sum(log(bagged_predictions[brexit_test$voteBrexit == 1])) +
  sum(log(1 - bagged_predictions[brexit_test$voteBrexit == 0]))
print(bstestLL)
```

```
## [1] -23.35458
```

4. The single case of the logistic regression model cannot capture the well in the probability of the pattern, which is non-monotonic, while the Bagging method is used to learn from different subsets of data that might capture the relationship of data better. However, this dataset seems to have little change in bagging capture of the relationship of data as Figures 6 and 7 below might not suggest a significant improvement in capturing the pattern, which is non-monotonic.

```
# Plot the predictions
plot(brexit_test$withHigherEd, logis_glm_prediction)
```

```
plot(brexit_test$withHigherEd, bagged_predictions)
```

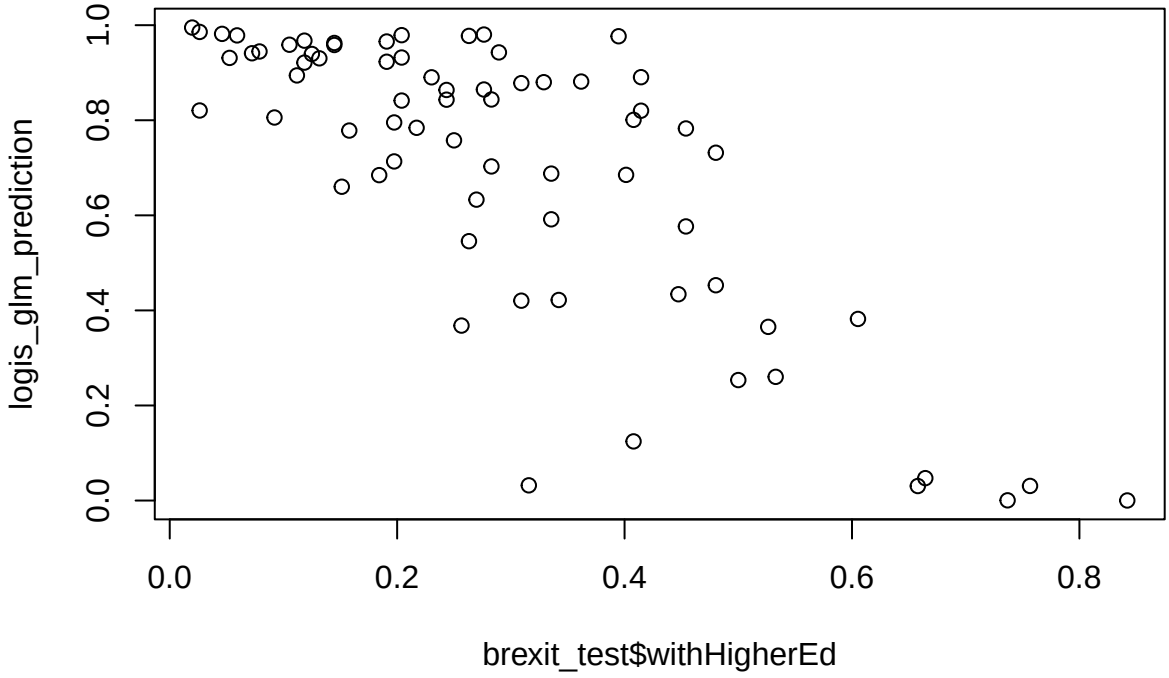

Figure 6: Scatter plot of logistic regression predictions vs. proportion with higher education

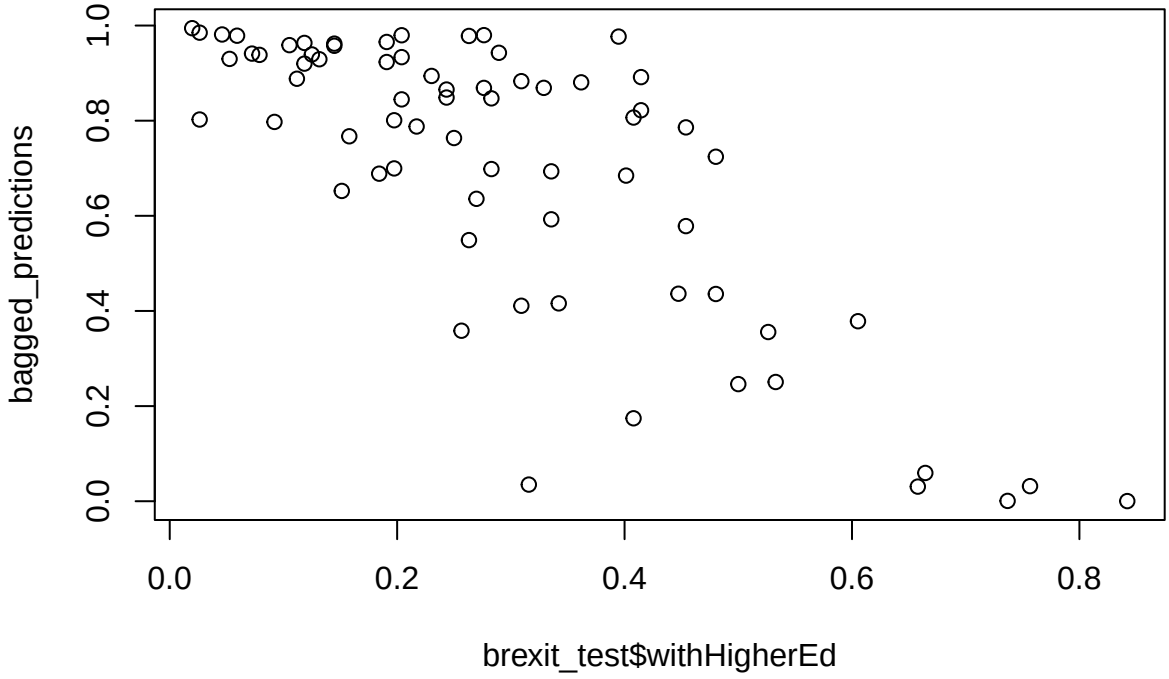

Figure 7: scatter plot of bagged logistic regression predictions vs. proportion with higher education

To summarise, the use of bagging or bootstrap aggregating methods, when applied to the logistic regression model, could be used to improve the stability of the coefficient in the model by averaging the coefficients across the multiple bootstrap samples, which is used to help reduce the variability and prevent the overfitting of the model. Furthermore, it could be used to improve performances in the capture of non-monotonic patterns in data. However, this dataset does not give the result well after evaluating the performance of the model by log-likelihood that the value seems slightly lower than the normal logistic model, suggesting that using the bagged method might not improve the fitting-performance and accuracy model.

Task4: Based on your discussion for Task 3, present and carry out an alternative logistic regression approach to carry out the analysis for Task 2. Discuss the benefits and disadvantages of your approach.

First idea: feature selection for logistic regression.

For this method, we will try to reduce the variable that seems to have less impact on the output variable to give the model a focus on the variables which might have more impact on the variable; thus it would be helpful to reduce the unstable of some coefficient that solve the problem which discussion in the task 3. So, in this approach, we will try to evaluate the result after doing the feature selection methods.

Using cross-validation to try to evaluate the model through the different of selecting input variable by selecting the variable based on the coefficients from Table 3:

- **The first model**: selecting all variables (withHigherEd + abc1 + medianAge + notBornUK + medianIncome)
- **The second model**: cut 1 less effect variable (withHigherEd + abc1 + medianAge + notBornUK)
- **The third model**: cut 2 less effect variables (withHigherEd + abc1 + medianAge)
- **The fourth model**: cut 3 less effect variables (withHigherEd + abc1)
- **The fifth model**: cut 4 less effect variables (withHigherEd)

Fitting the four models in the logistic regression equation by looping it can calculate the log-likelihood estimate for each model to compare the performance.

```
# cross-validation
# list the model that want to evaluate
formulas = c("voteBrexit ~.", "voteBrexit ~ withHigherEd + abc1 + medianAge + notBornUK",
             "voteBrexit ~withHigherEd + abc1 + medianAge",
             "voteBrexit ~withHigherEd + abc1",
             "voteBrexit ~withHigherEd")

predictive_log_likelihood = rep(NA, length(formulas))

for (i in 1:length(formulas)){
  #First fit logistic regression model with the training data
  model = glm(formula = formulas[i], family = binomial, data = brexit_train)
  fs_predictions = predict(model, newdata = brexit_test, type = "response")
  predictive_log_likelihood[i] <- sum(log(fs_predictions[which(brexit_test$voteBrexit == 1)]))
    + sum(log(1 - fs_predictions[brexit_test$voteBrexit == 0]))
}
```

Interpret the result from cross-validation of feature selection by plotting the graph of log-likelihood.

```r
plot(1:length(formulas), predictive_log_likelihood,
     xlab="Model Number", ylab="Log Probability")
```

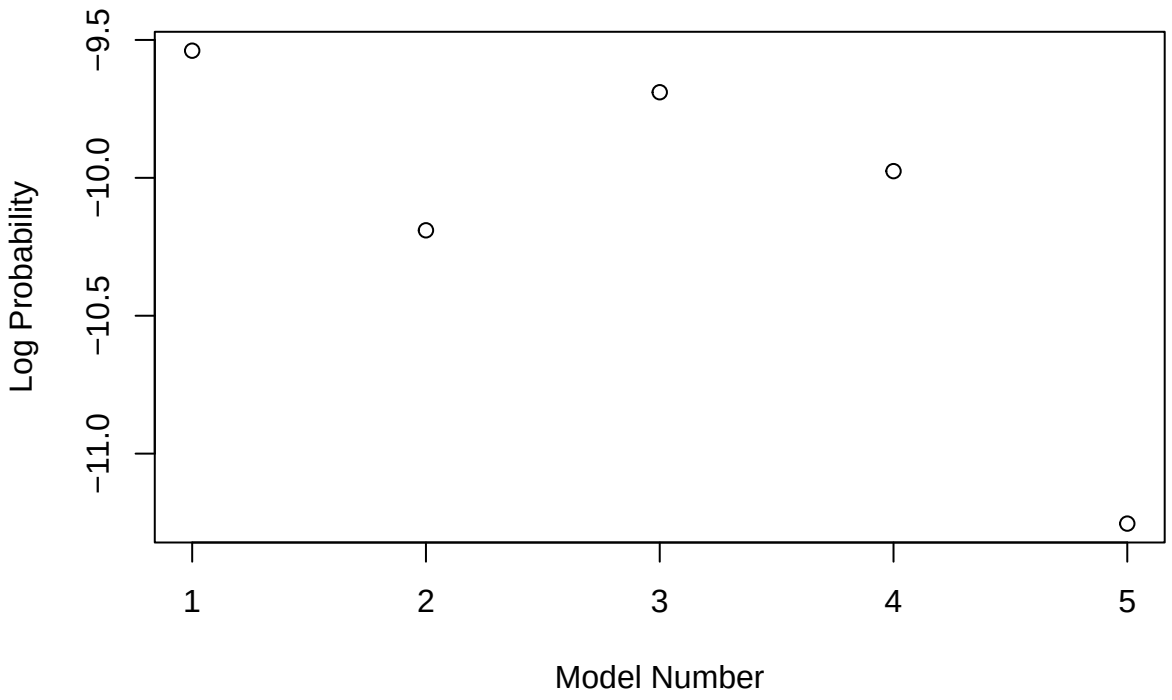

Figure 8: Compared of log-likelihood in different feature selection modes

To summaries, the result: from the plot, the first model which selected all inputs show the highest log probability of the input, making it the best performance model. The model of 2 which cut the variable of "medianAge" seem to performance drop. In terms of Model 3, which suggesting cut the unnecessary variable as 'medianage"' seem to have significantly increase performance than Models 2. As regards models 4 and 5, which cut a lot of variables, the model seems to give the worst performance. Thus, it could be confirmed that the best model is the model which inputs all of the variables as model1.

After that, repeat the process of cross-validation 100 times to count the model that can give the highest loglikelihood value as the figure below also confirms that model 1, which includes all of the variables as input (same as task), is the best performance model.

```r
# repeat the process 100 times
winner = rep(NA, 100)
for (iteration in 1:100){

#Make a new random training data - test data split
  brexit_size <- floor(0.80 * nrow(brexit))
  training <- sample(seq_len(nrow(brexit)), brexit_size)
  brexit_train <- brexit[training, ]
  brexit_test <- brexit[-training, ]

  predictive_log_likelihood = rep(NA, length(formulas))

  for (i in 1:length(formulas)){
    #First fit logistic regression model with the training data
    model = glm(formula = formulas[i], family = binomial, data = brexit_train)
    fs_predictions = predict(model, newdata = brexit_test, type = "response")
```

```
    predictive_log_likelihood[i] <- sum(log(fs_predictions[which(brexit_test$voteBrexit == 1)])) + sum(
  }

  #Find the winning model for this iteration
  winner[iteration] = which.max(predictive_log_likelihood)
}

#Plot a histogram of how often each model wins
hist(winner, breaks = seq(0.5, 5.5, 1), xlab='Model', ylab='Frequency', main='histrogram of cross valida
```

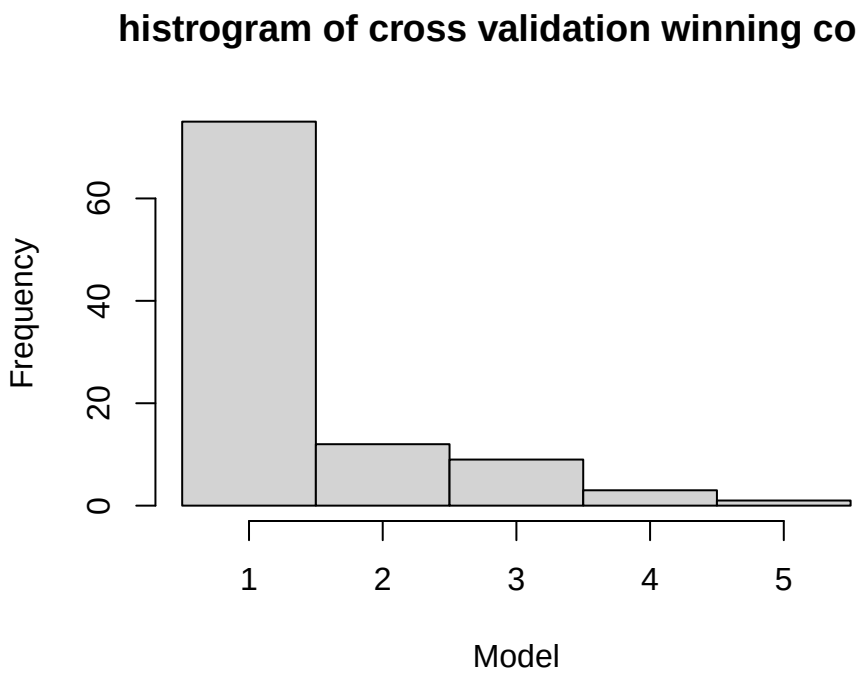

Figure 9: histrogram of cross validation winning count

Summaries the benefits and disadvantages of this approach.

Benefits

1. It could reduce some unimportant variables so the model can focus only on the important variable that might improve the performance of the model; however, this dataset suggests that all input variables seem the best.
2. Reduce the risk of overfitting by reducing some non-impact variables helps to reduce the complexity of the model, which decreases the likelihood of overfitting, making the model better for predicting the new data.
3. reduce the time of computation because fewer values of the features help to reduce the times of training models, which would be helpful when working with large datasets.

Disadvantages

1. If the feature selection process is not well managed, sometimes it might exclude the important variable that is very useful to predict the result, leading to the reduce of performance for the prediction of models like model 5 that reduces the social class variable, which has a high impact and the result shown that the model give the significantly performance drop.

2. If the feature selection is biased, it can significantly impact the fairness and the performance of the model selection. So, it should have a reasonable assumption to cut the model variable.

Second idea: Bayesian logistic regression

Based on Task 3, which discusses the factors that may affect the interpretability of the regression coefficient of fitting the models, task 4 will focus on the alternative logistic regression that could be used to handle the problem of unstable coefficient in fitting the logistic regression model.

This seconde idea will perform the model of Bayesian logistic regression as finding information from this model could be used to address the unstable of coefficients by including the prior knowledge about the coefficients, which helps to stabilise the coefficients. It would give a range of the possible value of each coefficient in the data. Thus, this method can make the model predictions more reliable. The factor that this model could be used to handle are:

- Outliers: Bayesian logistic regression manages outliers by using the prior information to limit the impact of outliers in the model, resulting in more stable and reliable estimates.
- Small sample size: the Bayesian logistic regression handles the small sample size of Brexit data by using prior knowledge to strengthen the model's estimates to help with the lack of data and improve stability.
- High correlation of data: the Bayesian logistic regression addresses the high correlation among the predictors by using the priors that can account for and adjust the influence of the correlated variables, leading to a more stable model.
- Non-monotonic data: the Bayesian logistic regression can effectively handle the non-monotonic data using the flexible specifications of the model and prior knowledge that can help to understand the complex relationship between the variables. This could help to bettering capture the non-monotonic data more accurately.

In the Bayesian logistic regression, the equation remains the same as the logistic regression as equation below; the coefficient β is updated with the prior beliefs about the coefficients based on the data.

$$\phi(x) = \frac{1}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k)}$$

where β is the coefficient of the logistic regression model updated with the prior knowledge. And x is the predictor variable.

Fitting the Bayesian logistic regression model

First, it needs to install the package that is necessary to use to progress the Bayesian logistic regression model.

```
# install necessary package
library("rstanarm")
```

Second, the model is fitted by using the function stan_glm(), setting the first argument as the voteBrexit, the output variable, and the other variable as input. The second argument is setting the model as binomial to classify the output of the model as binomial. And input the training data as argument three. The prior arguments are set as default using the function of 'rstanarm', which is designed to be weakly informative and provide stability to the estimate process without overconstating the outcome.

```
# Fitting the model
bayesian_model <- stan_glm(voteBrexit ~ .,
                           family = binomial,
                           data = brexit_train,
                           prior = default_prior_coef(family))
```

```
##
```

```
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.24 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.105 seconds (Warm-up)
## Chain 1:                0.104 seconds (Sampling)
## Chain 1:                0.209 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.097 seconds (Warm-up)
## Chain 2:                0.107 seconds (Sampling)
## Chain 2:                0.204 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
```

```
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.102 seconds (Warm-up)
## Chain 3:                0.09 seconds (Sampling)
## Chain 3:                0.192 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 8e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.106 seconds (Warm-up)
## Chain 4:                0.105 seconds (Sampling)
## Chain 4:                0.211 seconds (Total)
## Chain 4:
```

```r
# summary the model
summary(bayesian_model)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      voteBrexit ~ .
##  algorithm:    sampling
```

```
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 275
##  predictors:   6
##
## Estimates:
##                mean   sd    10%   50%   90%
## (Intercept)    0.3    0.9  -0.9   0.3   1.5
## abc1          16.8    3.0  13.1  16.8  20.7
## notBornUK      4.2    1.9   1.7   4.2   6.6
## medianIncome  -6.5    2.1  -9.1  -6.5  -3.8
## medianAge      4.6    1.5   2.7   4.6   6.5
## withHigherEd -24.2    3.6 -28.9 -24.1 -19.7
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.7    0.0  0.6   0.7   0.7
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  2001
## abc1          0.1  1.0  1093
## notBornUK     0.1  1.0  1297
## medianIncome  0.0  1.0  1917
## medianAge     0.0  1.0  1442
## withHigherEd  0.1  1.0  1175
## mean_PPD      0.0  1.0  3577
## log-posterior 0.0  1.0  1526
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

Third, summarising the model outcome has provided the posterior outcome of the average of the coefficient, which might be more stable than the normal logistic regression. These are represented in the table below in decreasing order of strong of magnitude:

Table 5: Coefficient Magnitudes of Variables

Variables	Coefficient Magnitudes
withHigherEd	-24.2
abc1	16.8
medianIncome	-6.5
medianAge	4.6
notBornUK	4.2

To summarise the result, the ranking of the effect of input that affects the output result is changing from Table 3, which could be said that the "withHigherEd" and "abc1" still have a strong impact on the target variable while the other three variables provide a lower impact. However, the ranking of magnitude of "medianIncome" from the lowest changing to the third rank followed by the "medianAge" and "notBornUK".

To see the performance of the model through the log-likelihood estimation:

```r
# Making the prediction the probability of the test data being 1
logis_bys_prediction <- predict(bayesian_model, newdata = brexit_test, type = 'response')
# Evaluate the predictive log-likelihood
bysLL <- sum(log(logis_bys_prediction[which(brexit_test$voteBrexit == 1)])) + sum(log(1 - logis_bys_pre
print(bysLL)
```

```
## [1] -22.54715
```

Table 6: Comparison of Method Performance Based on Log-likelihood Estimates

Compared the Performance of Methods	Log-likelihood Estimate
Normal logistic regression	-23.341
Bagged-logistic regression	-23.355
Bayesian logistic regression	-22.547

Table 6 illustrates the performance of different logistic models shown the most performance is likely to be the bayesian logistic regression model, with slightly improved from the normal logistic regression and bagged-logistic regression.

Summaries the benefits and disadvantages of this approach.

Benefits

1. Making the coefficient more stable through the Bayesian methods that use prior knowledge about the coefficient.
2. This model prevents the overfitting problem because it integrates new data with prior knowledge. Thus, making the model more robust to the new dataset.
3. Handle the non-monotonic data set by data using the flexible specifications of the model and prior knowledge that can help to understand the complex relationship between the variables. This could help to bettering capture the non-monotonic data more accurately. Thus, it slightly suggests the improvement of the performance of loglikelihood in Table 6.

Disadvantages

1. Need to know or assume the knowledge of the prior of the data which sometimes the result could be sensitive to the choice of prior in model setting.
2. It might be challenging to handle data which is very large because this model needs to learn from prior knowledge, so it might require a high-performance computer to handle the large dataset.