Brexit

Marcus Sinclair

2022

Objectives

Given a dataset regarding EU referendum votes by electoral ward containing 6 attributes about 344 wards, our goal is apply logistic regression to determine *which input variables are relevant for explaining the output (Brexit vote)* by interpreting the model. We will compare such results to ones produced by the Guardian newspaper, who have analysed this exact dataset with the same objective to ours. To answer questions about the dataset we must first import the .csv file, saving the file as **Brexit**.

```
rm(list=ls())#clears all data from your current Global Environment
setwd("/Users/marcu/OneDrive/Desktop/Uni/MATH5743M/assessedPract2")
Brexit <- read.csv("brexit.csv")
```

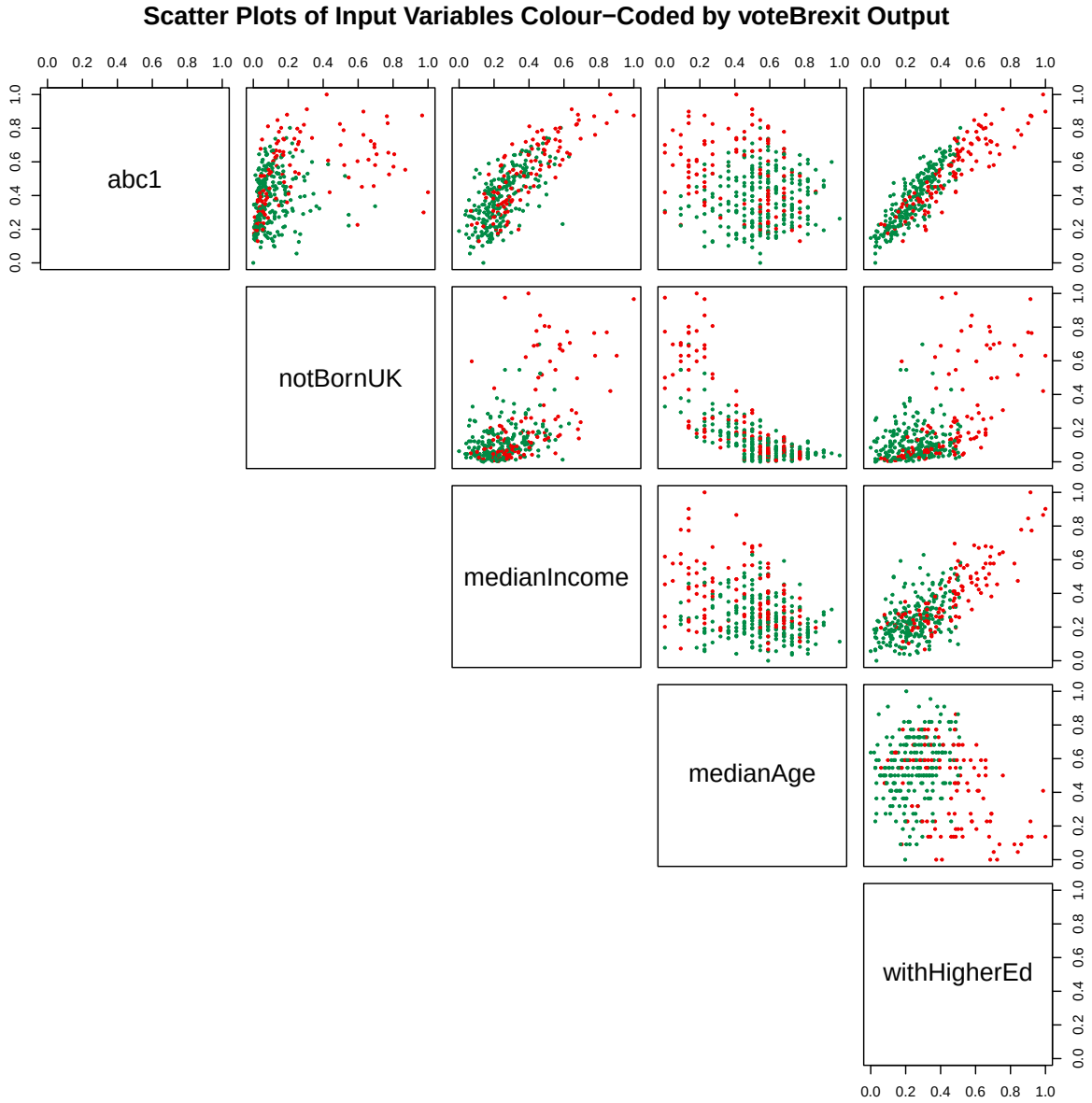
Data Understanding

Before we start applying logistic regression to the Brexit dataset, we must first understand each attribute and gain insight into their relations with each other. There are 6 attributes in the data. The output we consider is the Boolean variable **voteBrexit**, with output "TRUE" - the respective ward was pro Brexit, or "FALSE" - against Brexit. The 5 possible input variables are:

- **abc1**: proportion of individuals who are in the ABC1 social classes (middle to upper class)
- **medianIncome**: the median income of all residents
- **medianAge**: median age of residents
- **withHigherEd**: proportion of residents with any university-level education
- **notBornUK**: the proportion of residents who were born outside the UK

These are normalized so that the lowest value is zero and the highest value is one. To determine relations between input variables listed above, we plot the following scatter plots, with each possible combination shown.

```
Brexit$voteBrexit <- as.factor(Brexit$voteBrexit)
colours <- c("red2","springgreen4")
pairs(Brexit[,1:5],pch=19,cex=0.4,col=colours[Brexit$voteBrexit], lower.panel=NULL,
      main="Scatter Plots of Input Variables Colour-Coded by voteBrexit Output")
```



In the scatter plots above we denote data points constituting a ward with a "FALSE" voteBrexit output as a red point, and a "TRUE" voteBrexit output as a green point. This already tells us many interesting relations. For example, albeit not too surprising, wards with high proportions of residents who are not born in the UK almost always vote against Brexit - this is evident from the four scatter plots with a notBornUK input containing a vast amount of red data points at high notBornUK input values.

Furthermore, we observe relational trends. For instance, wards with large proportions of highly educated residents tend to contain high proportions of individuals who are in the ABC1 social class - see the top right plot. Many more inferences can be made from the plots above but for this report we shall now focus on applying the logistic model to the dataset to determine the most relevant input variables for explaining the output.

Logistic Regression Modelling

A reasonable starting point is to output the logistic regression model using all inputs. That is, our functional form of the model is defined as:

$$\phi(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}},$$

with $\mathbf{x} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$ denoting a linear combination of the inputs:

- A scalar constant - β_0.
- The first input x_1 representing a ward's abc1 value, multiplied by a second constant - $\beta_1 x_1$.
- The second input x_2 representing a ward's notBornUK value, multiplied by a third constant - $\beta_2 x_2$.
- The third input x_3 representing a ward's medianIncome value, multiplied by a fourth constant - $\beta_3 x_3$.
- The fourth input x_4 representing a ward's medianAge value, multiplied by a fifth constant - $\beta_4 x_4$.
- The fifth input x_5 representing a ward's withHigherEd value, multiplied by a sixth constant - $\beta_5 x_5$.

To output the model we use the **glm** command, stating the family of the model as binomial instead of the default Gaussian value the glm function takes. The outputs of the coefficients can be seen in the **summary table** below.

```
myglm = glm(voteBrexit ~ ., family=binomial, data=Brexit)
summary(myglm)$coefficients
```

```
##                 Estimate Std. Error    z value     Pr(>|z|)
## (Intercept)   -0.1385963  0.8476664 -0.1635033 8.701222e-01
## abc1          17.5779980  2.9114177  6.0376077 1.564157e-09
## notBornUK      5.6861383  1.8033339  3.1531256 1.615323e-03
## medianIncome  -6.3857396  1.9217008 -3.3229625 8.906690e-04
## medianAge      5.9208767  1.4065616  4.2094684 2.559722e-05
## withHigherEd -26.7442592  3.5761868 -7.4784291 7.521625e-14
```

The estimated constants β_0 to β_5 can be seen via the estimate column, with β_0 = -0.1385963 the initial intercept value going down in order to β_5 = -26.7442592, the withHigherEd constant. The model produces such estimates in the following fashion. The sigmoidal function defined above outputs a value between zero and one. The function classifies the output voteBrexit as "FALSE" if the function takes values less than one half and "TRUE" for the complement of this. The coefficients are constructed such that the likelihood of the resulting outputs for each of the 344 ward inputs are maximized.

Interpretation of the coefficients is as follows. The sign of the coefficient estimate determines the input variables relation to the voteBrexit output. Negative sign indicates a "FALSE" relation i.e. the input is against Brexit, positive sign indicates a "TRUE" relation i.e. the input is pro Brexit. This is evident by checking the order of the **contrasts** function of our voteBrexit output with the following code:

```
contrasts(Brexit$voteBrexit)
```

```
##       TRUE
## FALSE    0
## TRUE     1
```

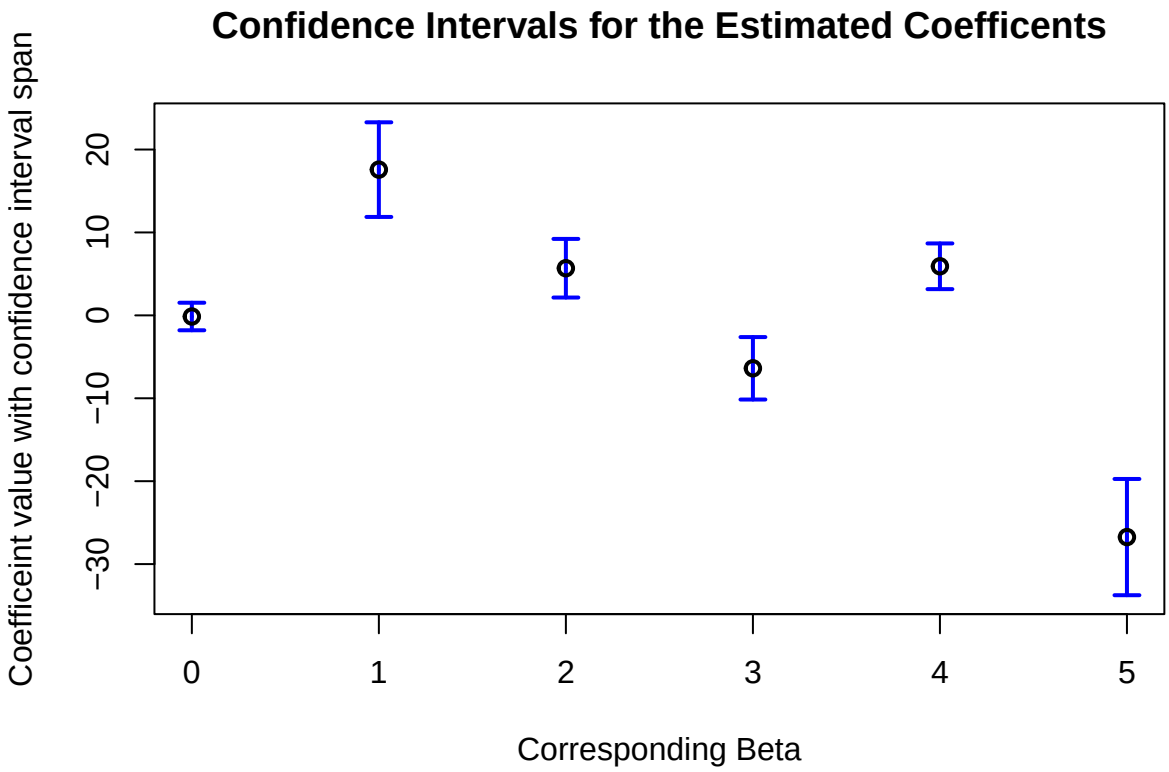
The size of the coefficients determines how dominant the respective input is compared to other inputs. The higher the absolute value, the greater weight the corresponding input has on determining the output classification. For instance, withHigherEd dominates the model, with the largest absolute coefficient value of $|\beta_5| = 26.7442592$ - analogous to this, the intercept has little to no impact on the model classification as seen by the relatively low value of $|\beta_0| = 0.1385963$.

The inputs in descending order are:

- **Strong effects**: withHigherEd, abc1, medianIncome, medianAge and notBornUK
- **Impacts**: pro Brexit- abc1, medianAge and notBornUK
- **Impacts**: against Brexit- withHigherEd and medianIncome

Note, these coefficient values are only estimates. It is useful to quantify how precise these estimates are in order to gain insight into their statistical significance on the model. An approach to quantify precision is via calculating **confidence intervals** for each coefficient estimate $\beta_0 - \beta_5$. A 95% confidence interval for our estimates is defined as: **estimate** $\pm$ **standard error** $\times z_{2.5\%}$. Our results as shown in the plot below.

```
library("plotrix")
CILow <-c(NA,NA,NA,NA,NA,NA)
CIHigh <-c(NA,NA,NA,NA,NA,NA)
for (i in 1:length(CILow)){
  CILow[i] <- summary(myglm)$coefficients[i,1]-summary(myglm)$coefficients[i,2]*qnorm(0.975)
  CIHigh[i] <- summary(myglm)$coefficients[i,1]+summary(myglm)$coefficients[i,2]*qnorm(0.975)
}
plotCI(x = seq(0,5,length.out=6),y = summary(myglm)$coefficients[,1],li = CILow,ui = CIHigh,
       scol="blue",main="Confidence Intervals for the Estimated Coefficents",
       xlab="Corresponding Beta",ylab="Coefficeint value with confidence interval span",lwd=2)
```



The span of values taken by the blue whiskers indicates a 95% certainty that the true value of beta situates in that domain. The larger the whisker the greater the uncertainty in the coefficient value. It is worth mentioning the confidence intervals themselves will be approximate since the standard errors are approximated by assuming that the likelihood is normally distributed (central limit theorem is applied). $\beta_0 \rightarrow [-1.8, 1.52]$, i.e. has whiskers than span through zero and thus is statistically insignificant on the model output. Although $\beta_2 \rightarrow [2.15, 9.22]$ and $\beta_3 \rightarrow [-10.15, -2.62]$ do not span through zero, they are relatively close compared to the remaining coefficients- their significance on the output could be questioned.

An important observation of our model must be made. From the summary table we see the coefficient weights for abc1, $\beta_1 = 17.577998$, and withHigherEd, $\beta_5 =$ -26.7442592 are negatively related. However, from data understanding, we find strong correlation between this two inputs as shown by the abc1 vs withHigherEd scatter plot. For large values both inputs are against Brexit and for small values both inputs and pro Brexit.

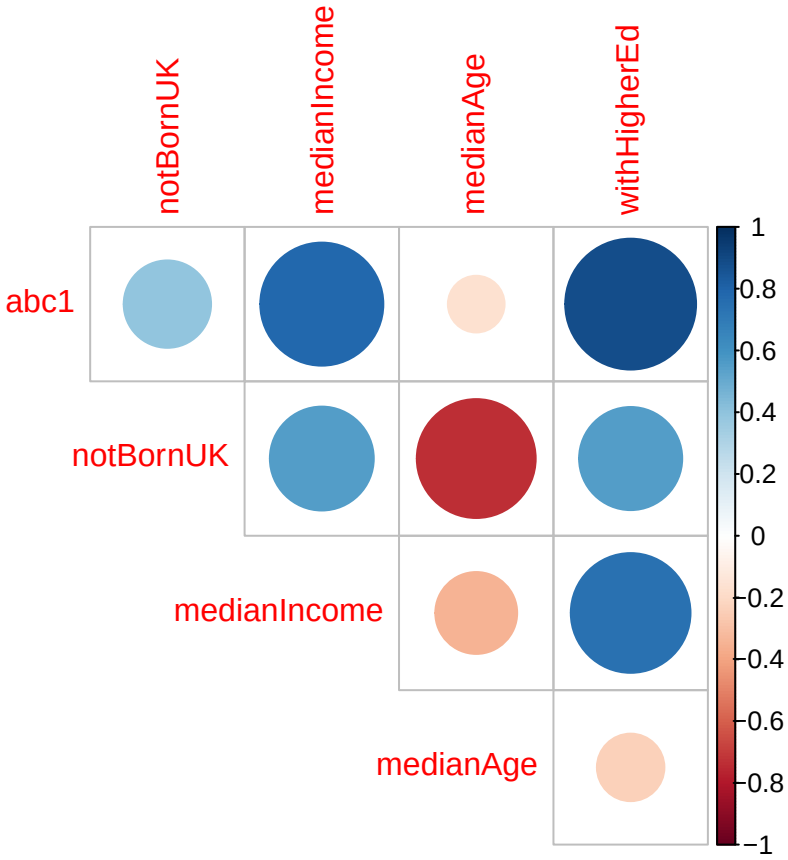
Both inputs are expected to have negative coefficients. The model fails due to the inputs being correlated. Our goal is to quantify which inputs have a higher enough correlation to warrant disregarding them from the model. A correlation matrix can be seen below.

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.2
```

```
## corrplot 0.92 loaded
```

```
c <- cor(Brexit[-6])
corrplot(c,type="upper",diag=F)
```



The greater the size of the circle the more correlated the two inputs are. Using this graph we will constrain ourselves to disregarding models including the combinations:

- **abc1** and **medianIncome**
- **abc1** and **withHigherEd**
- **notBornUK** and **medianAge**
- **medianIncome** and **withHigherEd**

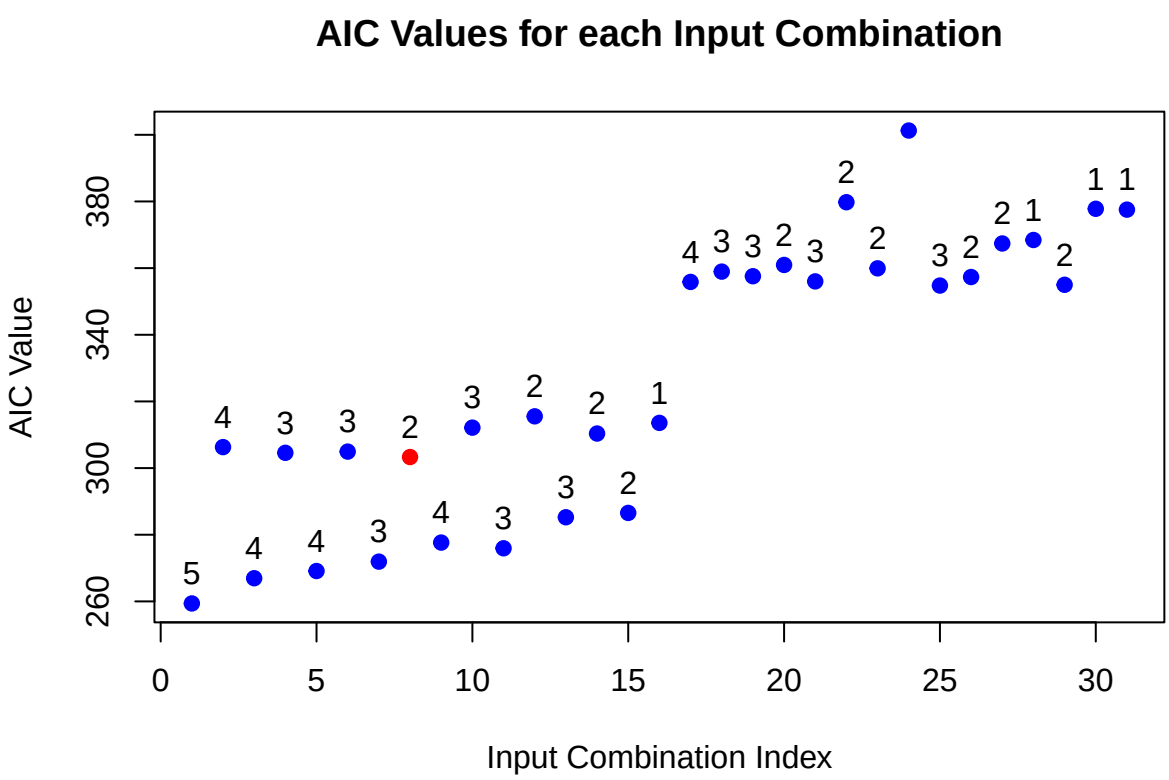
Combinations

Let us now consider which inputs combination performs best. A brute force approach (disregarding our input constraints for now) is to compute the logistic regression model for each possible input combination, quantify each combination's performance and choose the combination of inputs with the most optimum performance. We will quantify model performance via AIC values - the lower value the better. This statistic has the added benefit of penalizing models with large amounts of inputs which is beneficial to prevent overfitting.

Our dataset consists of 5 inputs thus all the possible combinations (excluding the possibility of having zero

inputs) are $\binom{5}{1} + ... + \binom{5}{5} = 31$ combinations. Here is the code that outputs the model AIC value for each combination of inputs.

```
AICs = rep(NA, length(formulas))
for (i in 1:length(formulas)){
  current_model <- glm(formulas[i], family=binomial, data=Brexit)
  AICs[i] <- current_model$aic
}
NumbInputs <- lengths(regmatches(formulas, gregexpr("\\+", formulas))) + 1
Indx <- seq(1,31)
plot(Indx,AICs,pch=19,col=ifelse(Indx==8,"red","blue"),main="AIC Values for each Input Combination",
     xlab="Input Combination Index", ylab="AIC Value")
text(Indx,AICs,labels=NumbInputs,pos=3)
```



From the scatter plot above, we see all AIC values for each of the 31 input combinations. They are labelled with the number of inputs used. The model with greatest performance under our input constraints is voteBrexit ~ medianAge + withHigherEd with AIC value 303.3090828. This can be seen by the red point on the plot above.

Cross-Validation

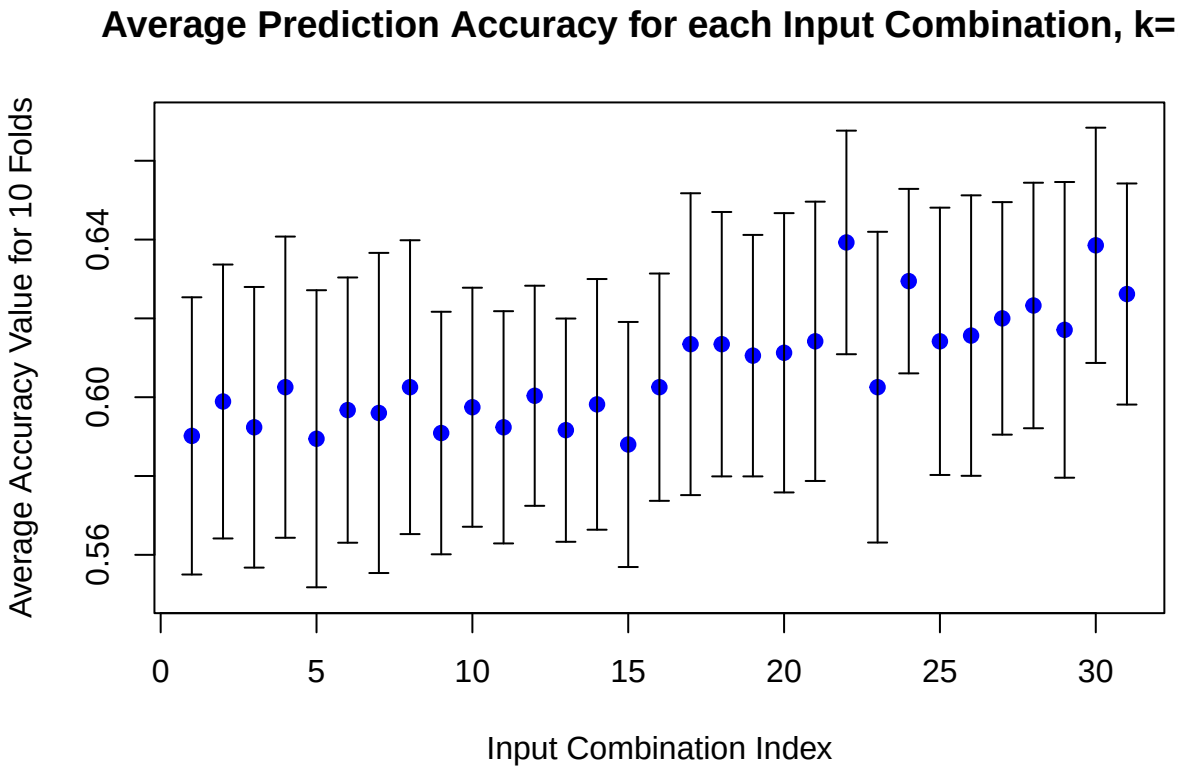
Instead of using the entire dataset to train the logistic regression model, we now apply a cross-validation strategy. The idea is to partition our dataset into a training set and a testing set. We train our model in the usual fashion (glm function optimizes beta parameters) but now we are able to test the logistic regression model, quantifying the model performance via testing statistics. We also apply this strategy $k = 10$ (10-fold cross-validation) times whereby each partition contains wards that are independent from other partitions. This strategy is used for each of the 31 input combinations (we disregard disallowed input combinations after computation), thus the total number of model outputs is $10 \times 31 = 310$ models produced. We quantify

a model's performance via its accuracy percentage - the proportion of correct predictions. The higher the accuracy the greater the model performance. Here is the code that performs cross-validation:

```
set.seed(121)
#folds
k = 10
#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=31, ncol=k)
#consider fold 1 to k
for (j in 1:k){
#creates testing and training sets randomly, test=20% of dataset
idx <- sample(1:344, 69)
train_data <- Brexit[-idx, ]
test_data <- Brexit[idx, ]
#applies glm model, predicts using test data and outputs accuracy
for (i in 1:length(formulas)){
 current_model <- glm(formulas[i], family=binomial, data=train_data)
 raw_probabilities <- predict(current_model, test_data, type="response")
 predicted_output <- rep("FALSE", dim(train_data)[1])
 predicted_output[raw_probabilities > .5] = "TRUE"
 table <- table(predicted_output, train_data$voteBrexit)
 Single_accuracy[i,j] <- sum(diag(table))/sum(table)
}

}
Average_accuracy <- rowSums(Single_accuracy)/k
```

```
#creates whiskers for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
plot(Indx,Average_accuracy,pch=19,col="blue",
     main="Average Prediction Accuracy for each Input Combination, k=10",
     xlab="Input Combination Index", ylab="Average Accuracy Value for 10 Folds",
     ylim=c(0.55,0.67))
arrows(x0=Indx, y0=Average_accuracy-accuracy_sd, x1=Indx, y1=Average_accuracy+accuracy_sd,
     code=3, angle=90, length=0.05)
```



The scatter plot above shows the prediction accuracy average value for each of the 31 input combinations. This value is computed as follows.

1. Randomly sample a training and testing set, we choose a 4:1 split for this approach.
2. Train the logistic regression model at input combination 1.
3. Test the model with testing set and store predictive accuracy value.
4. Repeat steps 1-3 k times.
5. Consider the next input combination 2 and repeat 1-4.
6. Take the average of the 10 accuracy predictions for each input combination.

The blue points are the average accuracy values and the whiskers indicate the standard deviation of the 10 computed accuracy values for each input combination. The larger the whiskers the greater the deviation in accuracy values thus the greater instability in the model. The model with the highest accuracy value under our constraints is voteBrexit ~ notBornUK with a average predictive accuracy of 0.6385455. We interpret this as follows. If we use this model to classify whether a new ward will vote for or against Brexit, the model will choose correct 63.85% of the time.

Upon reflection, given the large deviation in accuracy values, although this performance approach gives good insight into our model, it is not ideal to determine an optimal model combination. My choice would be voteBrexit ~ medianAge + withHigherEd with summary table:

```
Optglm <- glm(formulas[8], family=binomial, data=Brexit)
summary(Optglm)$coefficients
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)   2.539996  0.5236610  4.850458 1.231765e-06
## medianAge     2.413670  0.7047741  3.424743 6.153811e-04
## withHigherEd -8.745590  1.1099425 -7.879318 3.291719e-15
```

We have explained the interpretation of the coefficients and their uncertainty at the start of our report. The same concepts can be applied to this model. withHigherEd is most dominant - as the input increases the model is more likely to classify "FALSE". As medianAge increases the model is more likely to classify "TRUE". All inputs are statistically significant as evident by the small p-values - confidence intervals could be made using the same method shown prior.

Looking at the Guardians statistical learning applied to the dataset, we both agree withHigherEd is the best predictor for a ward to vote remain or not. They found geographical outliers to this in Scotland so I assume they had access to a similar dataset but with geo-locations of the wards. notBornUK seems to be a reasonable predictor too as it has an acceptable lack of correlation between withHigherEd. Further analysis could be too consider a model both with these two inputs and compete using test data against my choice of voteBrexit ~ medianAge + withHigherEd.