Flu shot learning

SusanSunny

1/7/2021

1 Introduction

This project is being completed as the last part of the Harvard X Data Science program.

1.1 Project Aim

For this project, I chose the data set from the "flu shot learning" competition published on drivendata.org. The aim of the challenge is to predict H1N1 and seasonal flu vaccination rates from a number of predictors. The primary evaluation metric is the area under the receiver operating characteristic curve (AUROC).

1.2 Project Overview

This report contains the result from two machine learning methods:

Firstly, random forests are constructed with the package xgboost.

Secondly, linear regression is performed using the package glmnet.

I chose to only predict the h1n1_vaccination rates for this report in order to keep it more clear. The prediction for the seasonal_vaccination rate can be performed analogously.

2 Methods and Analysis

2.1 Data Cleaning and Exploration

2.1.1 Download and overview of datasets

Since the data can only be downloaded from drivendata.org if logging in with an account, I have uploaded the data on my github account, from which the two datasets are downloaded in the R script.

The first dataset (flu_features) contains an ID for each entry (respondent_id) and 35 predictors (features), e.g. h1n1_concern, behavioral_face_mask, age_group, education, race, sex. Some predictors are numerical values and others are character strings. As we can already see from the first entries, flu_features contains NAs.

```
## 'data.frame':    26707 obs. of  36 variables:
##  $ respondent_id              : num  0 1 2 3 4 5 6 7 8 9 ...
##  $ h1n1_concern               : num  1 3 1 1 2 3 0 1 0 2 ...
##  $ h1n1_knowledge             : num  0 2 1 1 1 1 0 0 2 1 ...
##  $ behavioral_antiviral_meds  : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ behavioral_avoidance       : num  0 1 1 1 1 1 0 1 1 1 ...
##  $ behavioral_face_mask       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ behavioral_wash_hands      : num  0 1 0 1 1 1 0 1 1 0 ...
##  $ behavioral_large_gatherings: num  0 0 0 1 1 0 0 0 1 1 ...
##  $ behavioral_outside_home    : num  1 1 0 0 0 0 0 0 1 0 ...
##  $ behavioral_touch_face      : num  1 1 0 0 1 1 0 1 1 1 ...
##  $ doctor_recc_h1n1           : num  0 0 NA 0 0 0 0 1 0 0 ...
##  $ doctor_recc_seasonal       : num  0 0 NA 1 0 1 0 0 0 0 ...
##  $ chronic_med_condition      : num  0 0 1 1 0 0 0 1 0 1 ...
##  $ child_under_6_months       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ health_worker              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ health_insurance           : num  1 1 NA NA NA NA NA 1 NA 1 ...
##  $ opinion_h1n1_vacc_effective: num  3 5 3 3 3 5 4 5 4 4 ...
##  $ opinion_h1n1_risk          : num  1 4 1 3 3 2 1 2 1 2 ...
##  $ opinion_h1n1_sick_from_vacc: num  2 4 1 5 2 1 1 1 1 2 ...
##  $ opinion_seas_vacc_effective: num  2 4 4 5 3 5 4 4 4 4 ...
##  $ opinion_seas_risk          : num  1 2 1 4 1 4 2 2 2 2 ...
##  $ opinion_seas_sick_from_vacc: num  2 4 2 1 4 4 1 1 1 2 ...
##  $ age_group                  : chr  "55 - 64 Years" "35 - 44 Years" "18 - 34 Years" "65+ Years" ...
##  $ education                  : chr  "< 12 Years" "12 Years" "College Graduate" "12 Years" ...
##  $ race                       : chr  "White" "White" "White" "White" ...
##  $ sex                        : chr  "Female" "Male" "Male" "Female" ...
##  $ income_poverty             : chr  "Below Poverty" "Below Poverty" "<= $75,000, Above Poverty" "Bel
##  $ marital_status             : chr  "Not Married" "Not Married" "Not Married" "Not Married" ...
##  $ rent_or_own                : chr  "Own" "Rent" "Own" "Rent" ...
##  $ employment_status          : chr  "Not in Labor Force" "Employed" "Employed" "Not in Labor Force"
##  $ hhs_geo_region             : chr  "oxchjgsf" "bhuqouqj" "qufhixun" "lrircsnp" ...
##  $ census_msa                 : chr  "Non-MSA" "MSA, Not Principle  City" "MSA, Not Principle  City"
##  $ household_adults           : num  0 0 2 0 1 2 0 2 1 0 ...
##  $ household_children         : num  0 0 0 0 0 3 0 0 0 0 ...
##  $ employment_industry        : chr  NA "pxcmvdjn" "rucpziij" NA ...
##  $ employment_occupation      : chr  NA "xgwztkwe" "xtkaffoo" NA ...
```

The second dataset (flu_labels) also contains the respondent_id, so that the labels can be matched with the predictors and the response variables h1n1_vaccine and seasonal_vaccine.

Both dataset consist of 26707 rows.

```
## 'data.frame':    26707 obs. of  3 variables:
##  $ respondent_id   : num  0 1 2 3 4 5 6 7 8 9 ...
##  $ h1n1_vaccine    : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ seasonal_vaccine: num  0 1 0 1 0 0 0 1 0 0 ...
```

The next section explains how the NAs are removed.

2.1.2 Removal of NA's

The following table shows that there are NA's in nearly all columns of the data set. In some columns around half of the rows contain NAs.

```
##                              number_of_NAs
## respondent_id                            0
## h1n1_concern                            92
```

```
## h1n1_knowledge                      116
## behavioral_antiviral_meds            71
## behavioral_avoidance                208
## behavioral_face_mask                 19
## behavioral_wash_hands                42
## behavioral_large_gatherings          87
## behavioral_outside_home              82
## behavioral_touch_face               128
## doctor_recc_h1n1                   2160
## doctor_recc_seasonal               2160
## chronic_med_condition               971
## child_under_6_months                820
## health_worker                       804
## health_insurance                  12274
## opinion_h1n1_vacc_effective         391
## opinion_h1n1_risk                   388
## opinion_h1n1_sick_from_vacc         395
## opinion_seas_vacc_effective         462
## opinion_seas_risk                   514
## opinion_seas_sick_from_vacc         537
## age_group                             0
## education                          1407
## race                                  0
## sex                                   0
## income_poverty                     4423
## marital_status                     1408
## rent_or_own                        2042
## employment_status                  1463
## hhs_geo_region                        0
## census_msa                            0
## household_adults                    249
## household_children                  249
## employment_industry               13330
## employment_occupation             13470
```

Furthermore, only 6437 of the 26707 rows contain no NA's at all.

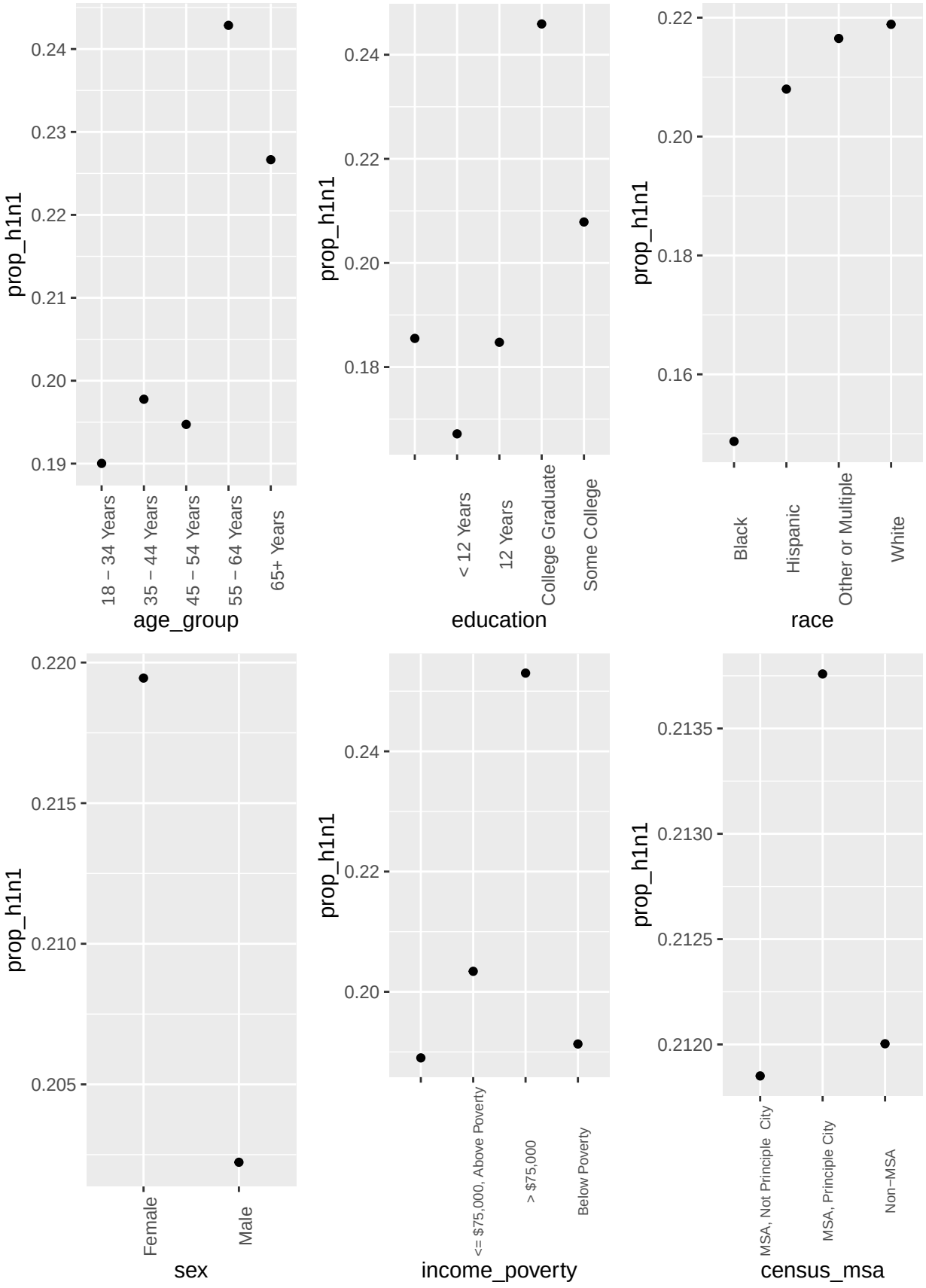
```
## [1] 6437
```

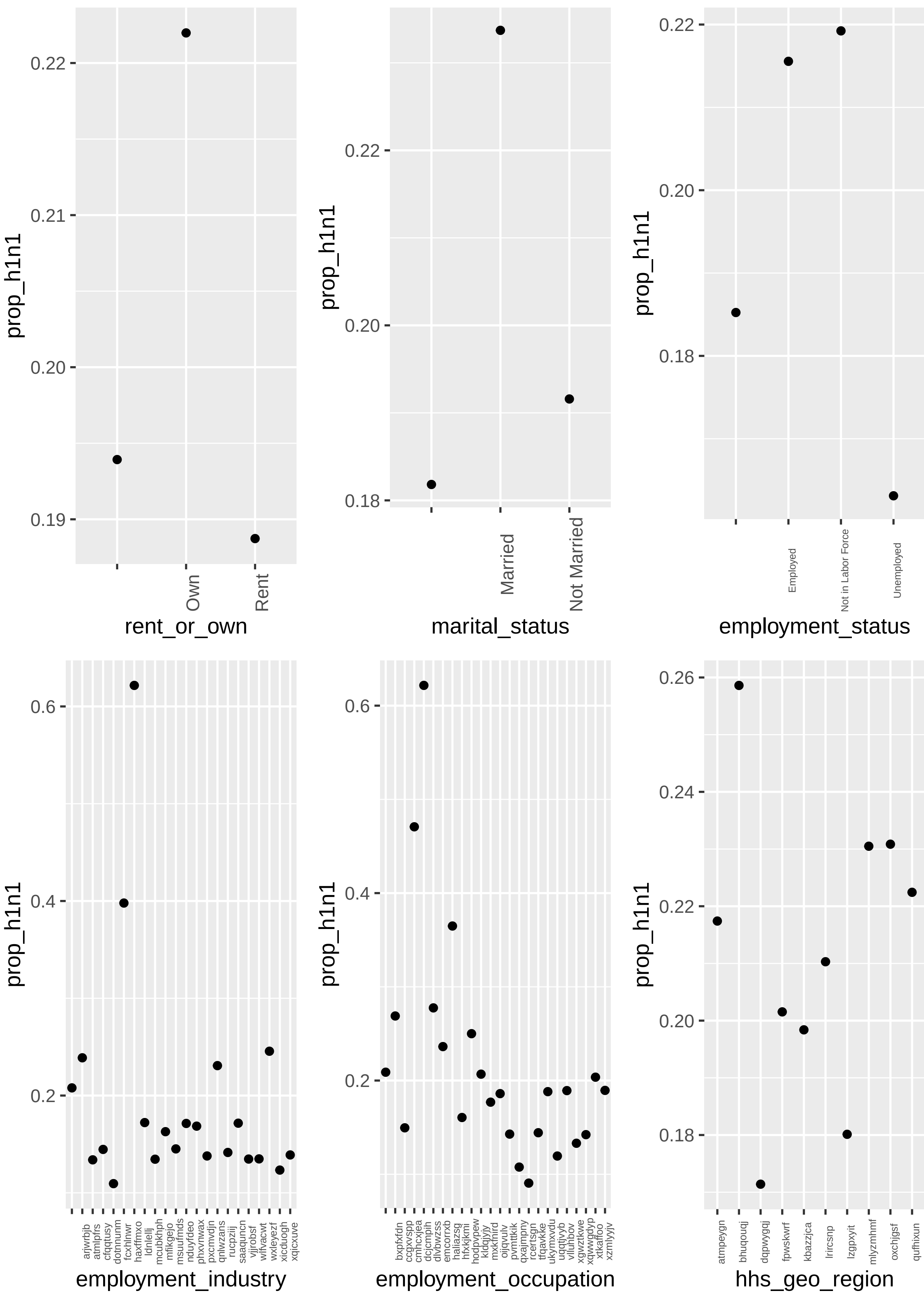
Since the number of rows without NA's is so small, it seems useful to replace the NAs with suitable values. I decided to replace the NA's in numerical columns with the column mean value. Whereas the NA's in the character string columns, I treated as a different value by replacing it with an empty string.

2.1.3 Factors and ordering

For both, the purpose of fitting a linear model and for the purpose of fitting a random forest, it is useful to choose an ordering for the factors representing the character strings which is compatible with the mean vaccination rate per factor level.

As we can see from the following plots, the original ordering does not show a monotone dependency of vaccination on factor level.


prop_h1n1
0.24
0.23
0.22
0.21
0.20
0.19
18 – 34 Years
35 – 44 Years
45 – 54 Years
55 – 64 Years
65+ Years
age_group
prop_h1n1
0.24
0.22
0.20
0.18
< 12 Years
12 Years
College Graduate
Some College
education
prop_h1n1
0.22
0.20
0.18
0.16
Black
Hispanic
Other or Multiple
White
race
prop_h1n1
0.220
0.215
0.210
0.205
Female
Male
sex
prop_h1n1
0.24
0.22
0.20
<= $75,000, Above Poverty
> $75,000
Below Poverty
income_poverty
prop_h1n1
0.2135
0.2130
0.2125
0.2120
MSA, Not Principle City
MSA, Principle City
Non−MSA
census_msa


prop_h1n1
0.19
0.20
0.21
0.22
Own
Rent
rent_or_own
prop_h1n1
0.18
0.20
0.22
Married
Not Married
marital_status
prop_h1n1
0.18
0.20
0.22
Employed
Not in Labor Force
Unemployed
employment_status
prop_h1n1
0.2
0.4
0.6
arjwrbjb
atmlpfrs
cfqqtusy
dotnnunm
fcxhlnwr
haxffmxo
ldnlellj
mcubkhph
mfikgejo
msuufmds
nduyfdeo
phxvnwax
pxcmvdjn
qnlwzans
rucpziij
saaquncn
vjjrobsf
wlfvacwt
wxleyezf
xicduogh
xqicxuve
employment_industry
prop_h1n1
0.2
0.4
0.6
bxpfxfdn
ccgxvspp
cmhcxjea
dcjcmpih
dlvbwzss
emcorrxb
haliazsg
hfxkjkmi
hodpvpew
kldqjyjy
mxkfnird
oijqvulv
pvmttkik
qxajmpny
rcertsgn
tfqavkke
ukymxvdu
uqqtjvyb
vlluhbov
xgwztkwe
xqwwgdyp
xtkaffoo
xzmlyyjv
employment_occupation
prop_h1n1
0.18
0.20
0.22
0.24
0.26
atmpeygn
bhuqouqj
dqpwygqj
fpwskwrf
kbazzjca
lrircsnp
lzgpxyit
mlyzmhmf
oxchjgsf
qufhixun
hhs_geo_region

2.2 Analysis

2.2.1 Random Forests with xgboost

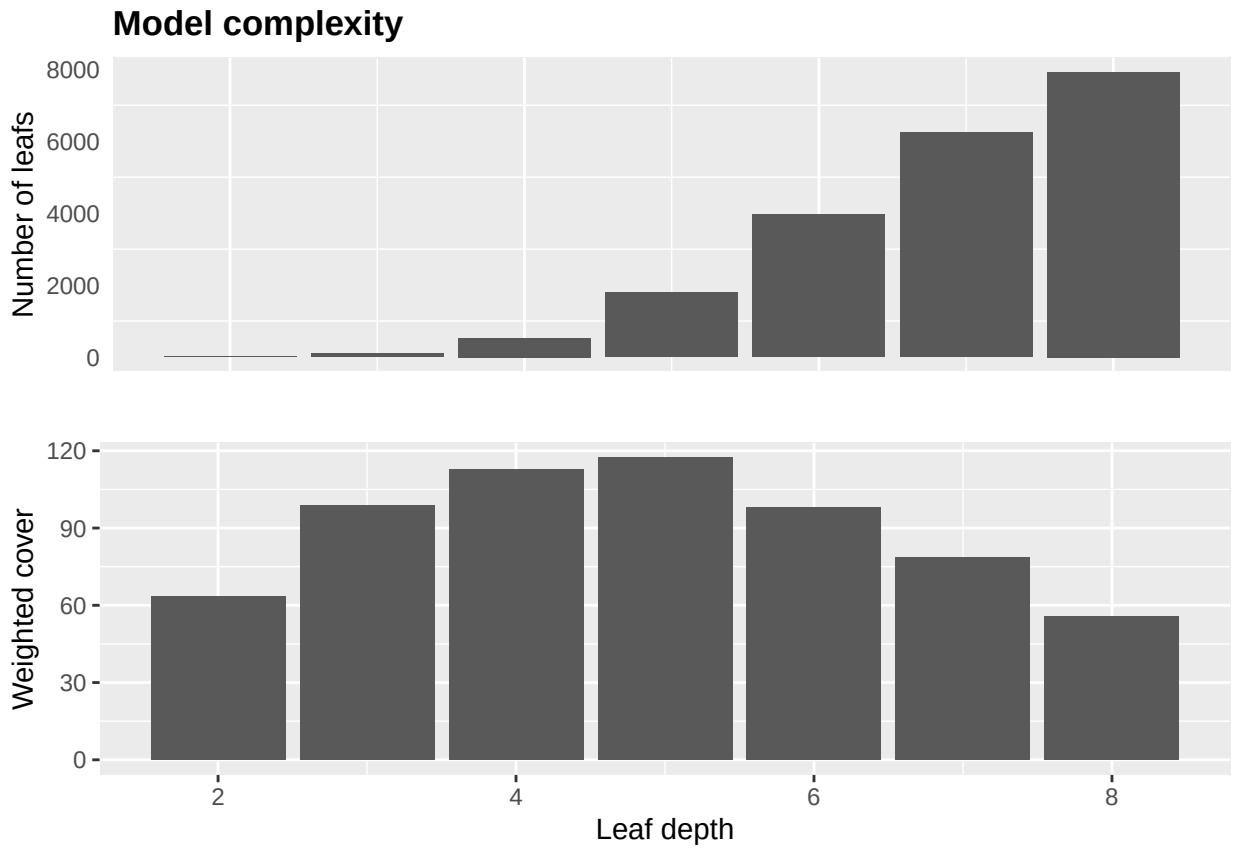
Xgboost is a powerful R package which can be used to build ensembles of decision trees (random forests) using gradient boosting algorithms. Random Forests are a machine learning method in which a large number of decision trees are built, each fitted with a random portion of the data. The prediction of the ensemble is built as a weighted sum of all the trees in the model.

I used the function xgb.cv to optimize the parameters for the xgboost model. Experimentation showed that a learning rate of 0.02 roughly leads to 500 trees per forest which was still manageable on a normal PC. An even lower learning rate might have led to even better models, but 0.02 seems to be a good compromise. It is important that every decision tree is fitted on a subset of the data so that the decision trees vary from each other enough. Therefore, the parameter subsample needs to be chosen smaller than 1. On the other hand the decision trees should not be fitted on a too small sample size, otherwise only simple decision trees could be fitted to the small sample size. Eventually I decided for a subsample of 0.9. Starting with a colsample_bytree of 0.7, I tried different values of gamma and maxdepth and chose gamma = 4, max_depth = 7. Given a gamma of 4 and maxdepth of 7, colsample_bytree = 0.5 led to a reasonable prediction performance. Running the function xgb.cv for many different values for the parameters takes quite some time, so that it is not included in this report.

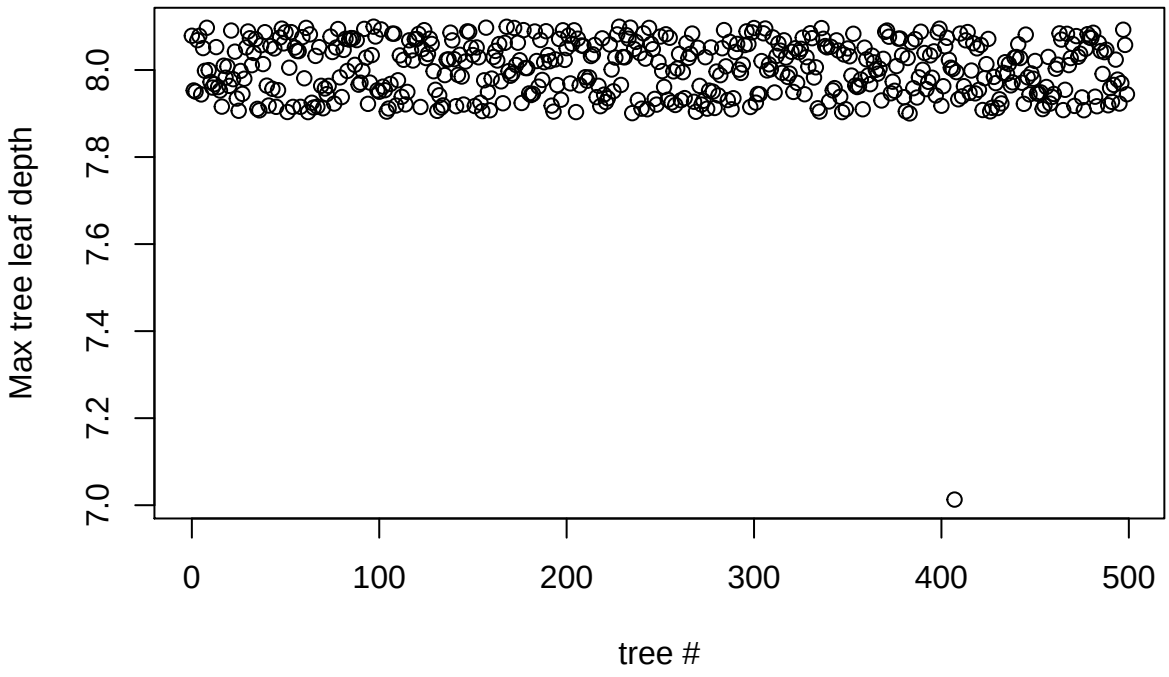
The following code builds the xgboost model xgb:

```
##xgboost

#factors as integers
flu_features_int1<- flu_features
flu_features_int1[string_cols]<-lapply(flu_features[string_cols], function(x) as.integer(x))
#remove respondent_id
flu_features_int<- flu_features_int1[,2:36]

#construct matrix for model
data=xgb.DMatrix(data=as.matrix(flu_features_int), label = flu_labels$h1n1_vaccine)
#train model
set.seed(1)
xgb<-xgb.train(data=data, objective= "binary:logistic", eval_metric="logloss",
          nrounds=500, eta = 0.02, gamma = 4, subsample = 0.9,
          colsample_bytree = 0.5, max_depth=7)
```

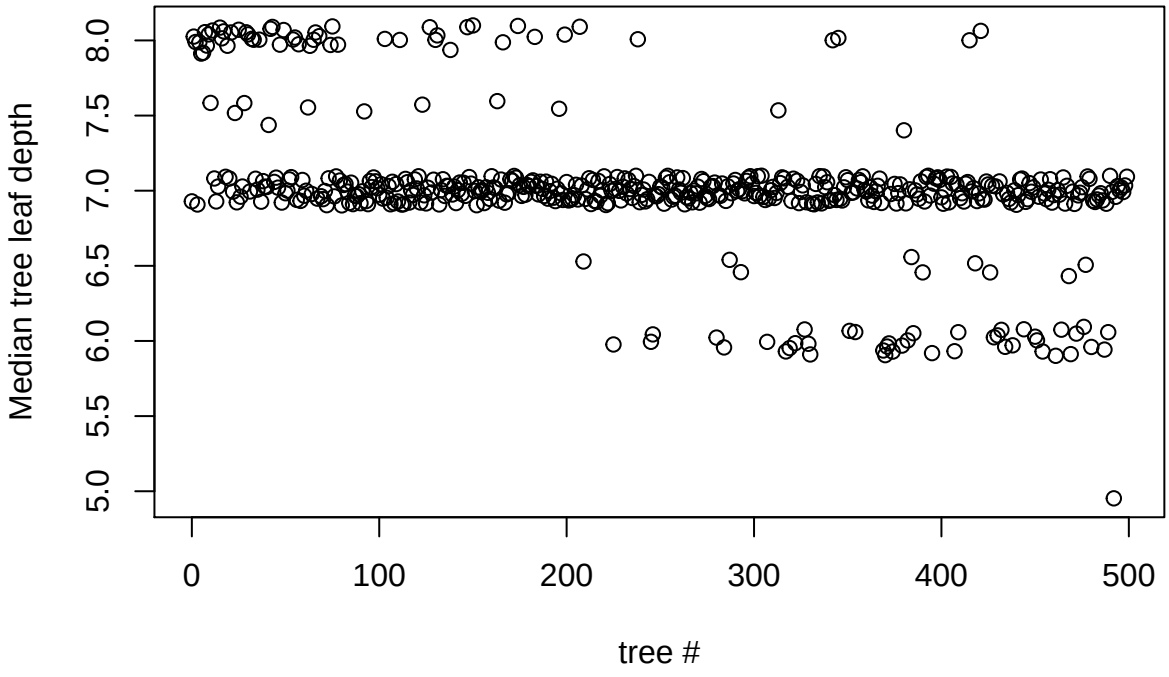
As can be seen from the upper graph, the higher the leaf depth, the more leafs a tree has. The plot below shows how many predictions are decided at each leaf depth.


Model complexity
Number of leafs
8000
6000
4000
2000
0
Weighted cover
120
90
60
30
0
2
4
6
8
Leaf depth

There is only one tree with maximum leaf depth 7, nearly all trees have a leaf depth of 8:



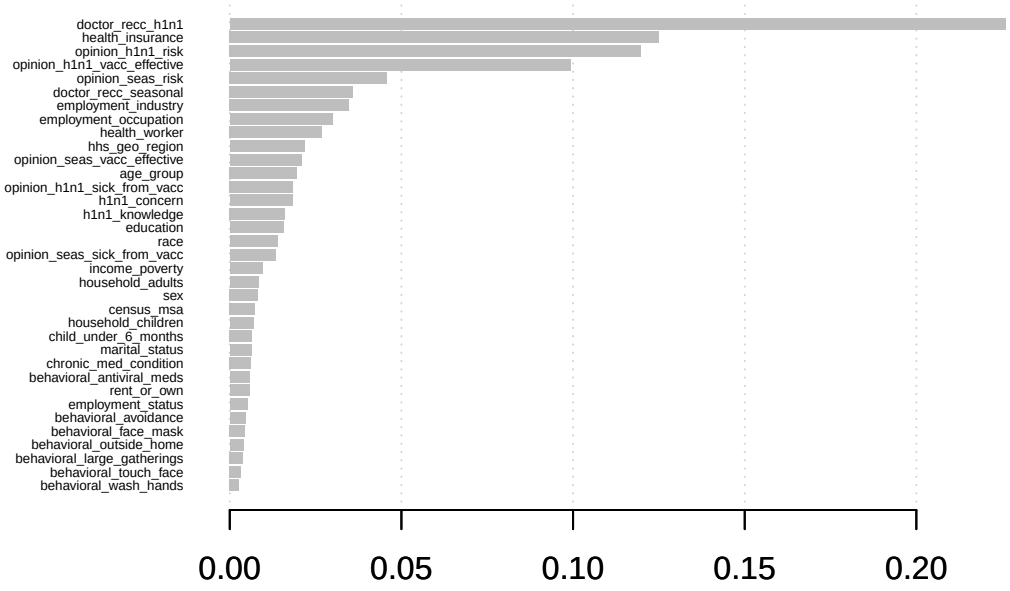
The median leaf depths range from 5 to 8. The most common median leaf depth is 7.



This table and diagram show the importance of each variable. We can see that for the h1n1_vaccine the most important predictors are the doctor_recc_h1n1, health_insurance, opinion_h1n1_risk and opinion_h1n1_vacc_effective.

```
##                         Feature        Gain       Cover   Frequency
##  1:            doctor_recc_h1n1 0.225925269 0.078126101 0.027160125
##  2:            health_insurance 0.125005476 0.091183701 0.032184251
##  3:           opinion_h1n1_risk 0.119746342 0.094253536 0.049644332
##  4: opinion_h1n1_vacc_effective 0.099293572 0.092780792 0.044968413
##  5:           opinion_seas_risk 0.045834185 0.043881491 0.042729941
##  6:        doctor_recc_seasonal 0.035934712 0.045721501 0.033228871
##  7:         employment_industry 0.034700741 0.054550522 0.059294633
##  8:       employment_occupation 0.029968439 0.050036880 0.058896682
##  9:              health_worker 0.026808006 0.036662726 0.023329851
## 10:             hhs_geo_region 0.021702106 0.027563189 0.056061284
## 11: opinion_seas_vacc_effective 0.020901397 0.032094821 0.035168880
## 12:                   age_group 0.019590255 0.033269457 0.042182759
## 13: opinion_h1n1_sick_from_vacc 0.018401389 0.038108896 0.043227379
## 14:                h1n1_concern 0.018287241 0.030854666 0.039297617
## 15:             h1n1_knowledge 0.016036033 0.021474426 0.031836044
## 16:                   education 0.015670226 0.024722586 0.035218624
## 17:                        race 0.013935455 0.022685416 0.031189375
## 18: opinion_seas_sick_from_vacc 0.013338612 0.023571117 0.034721186
## 19:             income_poverty 0.009508090 0.017358012 0.025618067
## 20:           household_adults 0.008391855 0.012756998 0.025369348
```

```
## 21:                         sex 0.008224099 0.017774764 0.018902651
## 22:                 census_msa 0.007364549 0.010759807 0.022384719
## 23:          household_children 0.006845094 0.010248792 0.021041636
## 24:        child_under_6_months 0.006540674 0.012987446 0.016664179
## 25:              marital_status 0.006410098 0.009433043 0.016266229
## 26:       chronic_med_condition 0.006174473 0.008931297 0.016415460
## 27:   behavioral_antiviral_meds 0.005746367 0.009858724 0.016017510
## 28:                 rent_or_own 0.005678643 0.008229216 0.016962642
## 29:           employment_status 0.005206245 0.006477127 0.015321096
## 30:        behavioral_avoidance 0.004523598 0.004876440 0.013530319
## 31:        behavioral_face_mask 0.004422451 0.006715926 0.012336467
## 32:     behavioral_outside_home 0.004032799 0.004231339 0.012585186
## 33: behavioral_large_gatherings 0.003891132 0.008772050 0.011540566
## 34:       behavioral_touch_face 0.003327978 0.003139359 0.010247227
## 35:       behavioral_wash_hands 0.002632399 0.005907834 0.008456449
##                         Feature        Gain       Cover   Frequency
```



The evaluation metric AUROC is calculated with the cross-validation function of the xgboost package and the built-in evaluation metric "auc".

```
#Calculate AUROC
set.seed(1)
xbgcv<- xgb.cv(data=data, verbose=TRUE, print_every_n=100, nfold=5,
               objective="binary:logistic", eval_metric="auc", nrounds=500,
               eta=0.02, gamma=4, max_depth=7, subsample=0.9, colsample_bytree=0.5)
```

```
## [1]  train-auc:0.820400+0.031092 test-auc:0.809026+0.027731
## [101]    train-auc:0.885831+0.002001 test-auc:0.865689+0.007840
## [201]    train-auc:0.897502+0.001418 test-auc:0.870083+0.007531
## [301]    train-auc:0.906116+0.001248 test-auc:0.871928+0.007305
## [401]    train-auc:0.913065+0.001127 test-auc:0.872459+0.007105
## [500]    train-auc:0.918852+0.001148 test-auc:0.872524+0.006965

##    iter train_auc_mean train_auc_std test_auc_mean test_auc_std
## 1:  500       0.918852   0.001147772     0.8725238  0.006964715

## [1] "out-of-sample AUC:0.8725238"
```

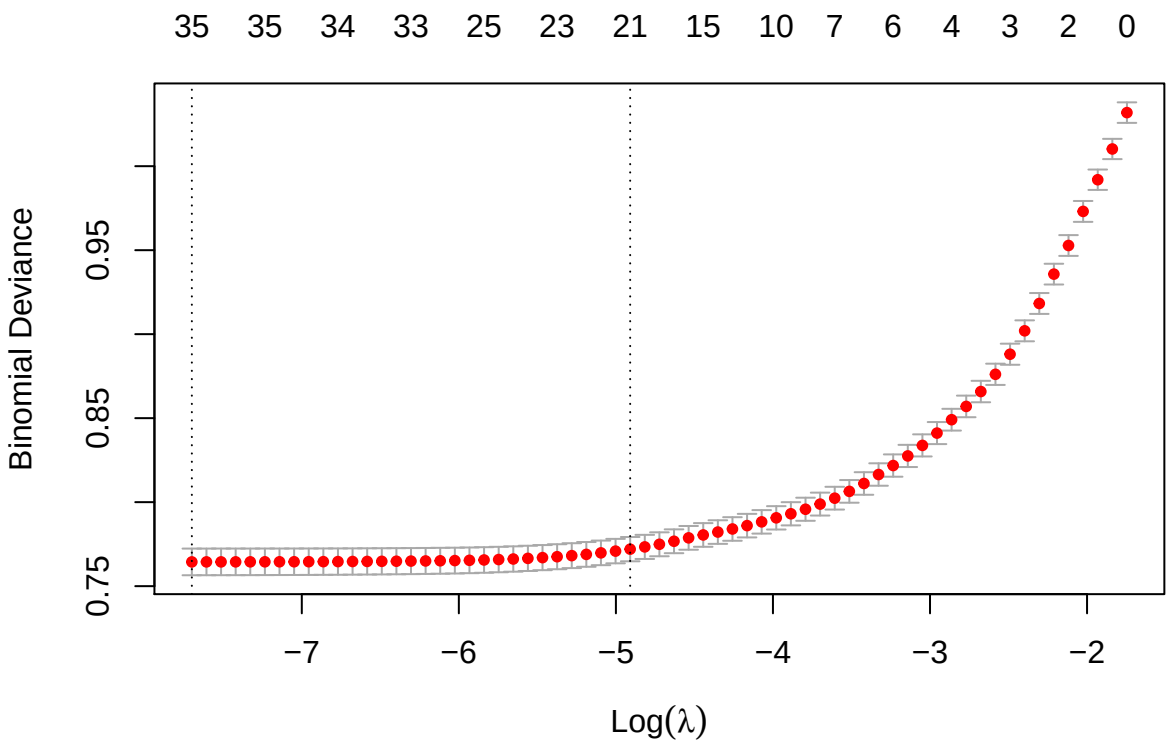
The out-of-sample AUC is 0.8725238.

2.2.2 Glmnet without one-hot encoding of the variables

The linear model is built using the package glmnet.

```
##Linear Model with glmnet
##without one-hot-encoding

#Create test set
set.seed(1) #set seed to always get the same result
test_index <- createDataPartition(y = flu_labels$h1n1_vaccine, times = 1, p = 0.1, list = FALSE)
test_index<- as.vector(test_index)
x_train <- flu_features_int[-test_index,]
x_test <- flu_features_int[test_index,]
y_train <- flu_labels[-test_index,]
y_test <- flu_labels[test_index,]

##without one-hot-encoding (assume linearity because I ordered the variables by the mean response)
#create matrix for cv.glmnet
glm_matrix1<-as.matrix(x_train)
#model with 10-fold cross-validation
set.seed(1)
alpha<- 0.9
cvg1<-cv.glmnet(glm_matrix1, y_train$h1n1_vaccine, family = "binomial", alpha=alpha)
```

Here, the log(lambda) is plotted against the logloss (prediction error). The smaller the penalty parameter lambda, the more variables are included in the model. In order to achieve the very best prediction error we could choose a lambda with the optimal prediction error. However, I choose the largest value of lambda, so that the error is within 1 standard error of the minimum, for a more robust model.



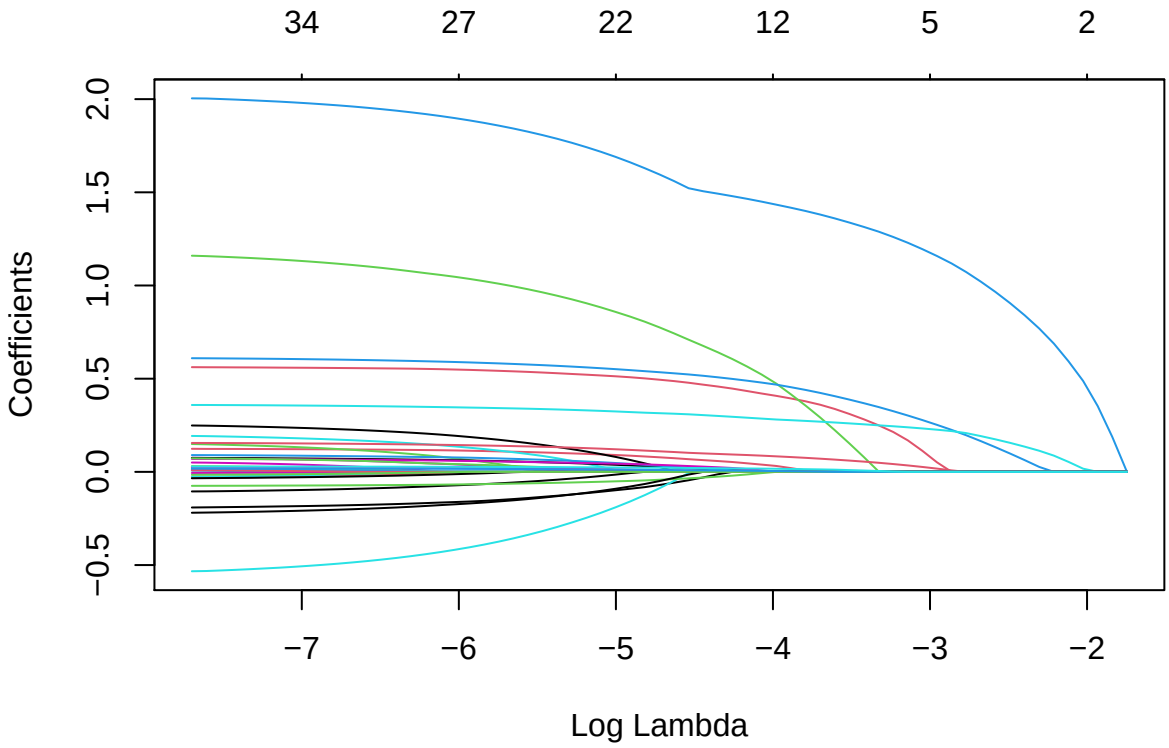

```
## [1] "lambda with optimal prediction error: -7.7001305308078"
```

```
## [1] "largest value of lambda such that error is within 1 standard error of the minimum: "
```

```
## [1] "-4.90911829947496"
```

```
#build model
glm_model1<-glmnet(glm_matrix1, y_train$h1n1_vaccine, family="binomial", alpha = 0.9)
```

If we look at the lambdas against the coefficients we can see how much influence each variable has depending on the lambda. One variable seems to have a high positive impact on the prediction. The majority of the variables have only a small influence.



In order to calculate the AUROC I used the method auc from the pROC package.

```
#make prediction
test_matrix<- as.matrix(x_test)
pred1<-predict(glm_model1, test_matrix, type="response", s=cvg1$lambda.1se)
#calculate AUC
auc_glm1<-auc(y_test$h1n1_vaccine, as.vector(pred1))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_glm1
```

```
## Area under the curve: 0.8383
```

2.2.3 Glmnet with one-hot-encoding

A linear model assumes a linear relationship between the variables. However, in our data set we also have variables that might not have a linear relationship. If we don't want to assume a linear relationship between the variables, we can use one-hot-encoding of factor levels. This way each value of each factor level is made to a separated predictor.

```
##Linear Model with glmnet
##with one-hot-encoding

#one-hot-encoding with makeX
glm_matrix2<- makeX(x_train)
```

Now, we build the model using the same modeling approach, but with the one-hot-encoded matrix:

```
#build model
glm_model2<-glmnet(glm_matrix2, y_train$h1n1_vaccine, family="binomial", alpha = 0.9)
```

Indeed, the prediction is only slightly better than without one-hot-encoding.

```
#make prediction
test_matrix<- makeX(x_test)
pred2<-predict(glm_model2, test_matrix, type="response", s=cvg2$lambda.1se)
#calculate AUC
auc_glm2<-auc(y_test$h1n1_vaccine, as.vector(pred2))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_glm2
```

```
## Area under the curve: 0.8372
```

3 Results

The following table summarizes the results for the AUC using the different models:

```
##                           Model       AUC
## 1                       xgboost 0.8725238
## 2                        glmnet 0.8382935
## 3 glmnet with one-hot-encoding 0.8372061
```

We can see that the xgboost model shows a better result than the glmnet models. Using one-hot-encoding for the glmnet did not improve the AUC. It even resulted in a slightly lower AUC. Thus, it seems that assuming a linear relationship between the factor levels within the variables is suitable in this data set.

The best public score at the moment (January 6th of 2021) for predicting h1n1_vaccine and seasonal_vaccine is at 0.8658. We can not directly compare our value since our result is only for predicting the variable h1n1_vaccine, whereas the competition's score refers to the average of the AUC metric of the h1n1_vaccine and the seasonal_vaccine prediction tasks. However, our result seems to lie in a good range.

4 Conclusion

In conclusion, Xgboost and glmnet are suitable packages to tackle this prediction problem. The results for the xgboost model are very depending on the parameters. Thus, a more comprehensive optimization of the parameters (eta, gamma, subsample, max_depth, nrounds, colsample_bytree and others) could potentially further improve the results. It would be interesting to calculate the result for the seasonal_vaccine as well in order to be able to better compare the result.