SEJ Analytix: Logistic Models for Daily Traffic in 2020

Machine Learning methods to classify 8 categories of Covid19 policies

Emanuela Ene

#Clear the space

```
rm(list = ls())
```

Packages and helper function

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(earth)     # fit MARS models
```

```
## Loading required package: earth
```

```
## Warning: package 'earth' was built under R version 4.3.3
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Warning: package 'plotmo' was built under R version 4.3.3
```

```
## Loading required package: plotrix
```

```
require(caret)     # automating the tuning process
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
require(vip)       # variable importance
```

```
## Loading required package: vip
```

```
## Warning: package 'vip' was built under R version 4.3.3
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
require(glmnet)
```

```
## Loading required package: glmnet
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
require(doParallel)
```

```
## Loading required package: doParallel
```

```
## Warning: package 'doParallel' was built under R version 4.3.3
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.3.3
```

```
## Loading required package: iterators
```

```
## Warning: package 'iterators' was built under R version 4.3.3
## Loading required package: parallel
```

```
setwd("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation")
```

Read in the data set.

```
trafficCovid = read.csv("C:/Users/Mama/Desktop/Customer_cases/CovidOnTransportation/categorical_and_numerical_Covid_and_impu
ted_TrafficDaily.csv")
```

```
which.min(trafficCovid$propTraffic)
```

```
## [1] 8026
```

```
trafficCovid[which.min(trafficCovid$propTraffic),]
```

```
##            date FIPS.State.Code Year.of.Data Month.of.Data Day.of.Data
## 8026 2020-12-05              23           20            12           5
##      stateTraffic Day.of.Week Week.of.Year priorTraffic offTraffic propTraffic
## 8026            0           7           49     0.878074  -0.878074           0
##      emer lock trv cls emergency lockDown travelBan closedStores
## 8026    Y    N   Y   R  1.159999 0.293556 0.6959995    0.4639997
```

```
positive_trafficCovid<-trafficCovid%>%filter(propTraffic>0)
which.min(positive_trafficCovid$propTraffic)
```

```
## [1] 17266
```

```
positive_trafficCovid[which.min(positive_trafficCovid$propTraffic),]
```

```
##             date FIPS.State.Code Year.of.Data Month.of.Data Day.of.Data
## 17266 2020-03-08              53           20             3           8
##       stateTraffic Day.of.Week Week.of.Year priorTraffic offTraffic
## 17266     0.000599           1           10     9.398967  -9.398368
##        propTraffic emer lock trv cls emergency lockDown travelBan closedStores
## 17266 6.373041e-05    Y    N   N   N  12.67592 3.351003  3.351003     3.351003
```

```
cats <- positive_trafficCovid%>%
  group_by(across(c(emer,lock,trv,cls))) %>%
  summarise(count = n(), .groups = 'drop') %>%
  select(-count)
print(cats)
```

```
## # A tibble: 14 × 4
##    emer  lock  trv   cls  
##    <chr> <chr> <chr> <chr>
##  1 N     N     N     N    
##  2 Y     N     N     N    
##  3 Y     N     N     R    
##  4 Y     N     N     Y    
##  5 Y     N     R     Y    
##  6 Y     N     Y     R    
##  7 Y     N     Y     Y    
##  8 Y     Y     N     R    
##  9 Y     Y     N     Y    
## 10 Y     Y     R     N    
## 11 Y     Y     R     Y    
## 12 Y     Y     Y     N    
## 13 Y     Y     Y     R    
## 14 Y     Y     Y     Y
```

```
cats<-cats%>%mutate(label=c("none", "NNN",'NNR', 'NNY','NRY','NYR', 'NYY', 'YNR', 'YNY', 'YRN','YRY','YYN', 'YYR','YYY' ))

traffic<-positive_trafficCovid %>% left_join(cats, by = c("emer", "lock", "trv", "cls"))%>%select(date,FIPS.State.Code,propTraffic,label)
traffic<-traffic%>%mutate(label=if_else(date<"2020-02-29", "before",label))%>%filter(label!="before", date<"2020-07-03")
```

#prepare data for classification

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
##
## Attaching package: 'tidyr'
```

```
## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

```
allStateCodes<-unique(traffic$FIPS.State.Code)
 traffic_wide<-tibble()
 for (state in allStateCodes){
 data<-traffic%>%filter(FIPS.State.Code==state)

 data_wide <- data %>% pivot_wider(names_from = date, values_from = propTraffic,values_fill = 1e-06 )
 traffic_wide<-bind_rows(traffic_wide,data_wide)
 }

mydata<-as.data.frame(traffic_wide)%>%select(-FIPS.State.Code)
table(mydata$label)
```

```
##
##  NNN  NNR  NNY none  NRY  NYR  NYY  YNR  YNY  YRY  YYR  YYY
##   50    1   15   50    8    2   13    2   17    8    2   14
```

```
data<-mydata%>%filter(!(label%in%c('NNR','NYR','YNR','YYR')))
table(data$label)
```

```
##
##  NNN  NNY none  NRY  NYY  YNY  YRY  YYY
##   50   15   50    8   13   17    8   14
```

```
Y = make.names(data$label)
X = select(data,-label)
Y = as.factor(Y)

set.seed(2)
trainSplit = createDataPartition(y = Y, p = 0.8, list = FALSE)

Ytrain = Y[trainSplit]
Xtrain = X[trainSplit,]
XtrainMat = as.matrix(Xtrain)
Ytest  = Y[-trainSplit]
Xtest  = X[-trainSplit,]
XtestMat = as.matrix(Xtest)
```

Let’s look at a fitting the logistic elastic net

```
set.seed(1)
K = 2
trainControl = trainControl(method = "cv", number = K)
tuneGrid = expand.grid('alpha'=c(.5, 1),'lambda' = seq(0.0001, .01, length.out = 5))
elasticOut = train(x = Xtrain, y = Ytrain,
method = "glmnet", family = 'multinomial',
trControl = trainControl, tuneGrid = tuneGrid)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
elasticOut$bestTune
```

```
##   alpha   lambda
## 7     1 0.002575
```

Using these selected tuning parameters, let’s get some predictions on the test digits data

```
glmnetOut         = glmnet(x = XtrainMat, y = Ytrain,
                           alpha = elasticOut$bestTune$alpha, family = 'multinomial')
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

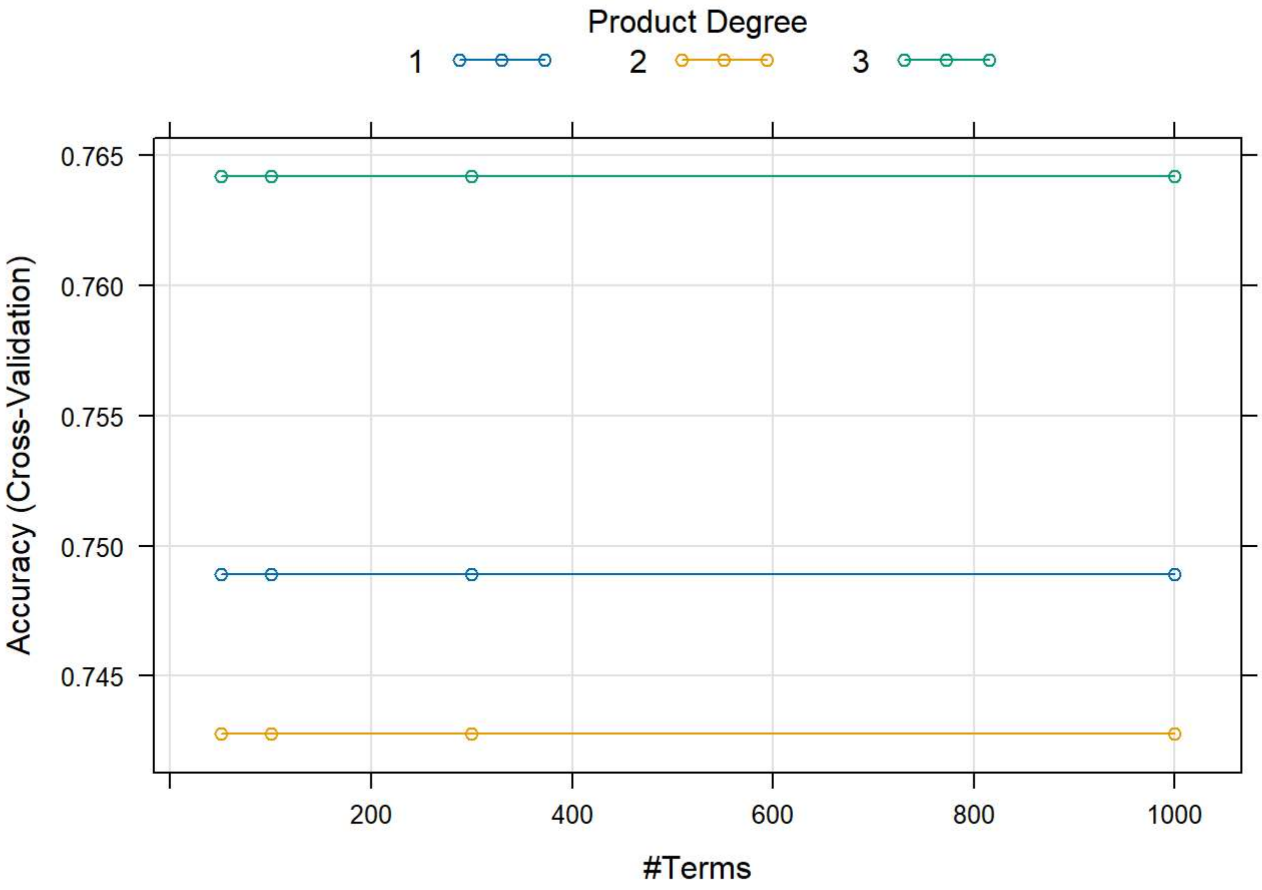
```
probHatTestGlmnet = predict(glmnetOut, XtestMat, s=elasticOut$bestTune$lambda, type = 'response')

YhatTestGlmnet = apply(probHatTestGlmnet,1,which.max) %>% "["(colnames(probHatTestGlmnet),.)
```

#Apply MARS to the normalized state traffic on days

```
# Enable parallel processing

library(doParallel)
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)

fdaOut = train(x = Xtrain,
                y = Ytrain,
                method = 'fda',
                metric = 'Accuracy',
                tuneGrid=expand.grid(degree=1:3,nprune=c(50, 100,300,1000)),
                trControl = trainControl(method='CV',number = 5, classProbs = TRUE))

fdaOut
```

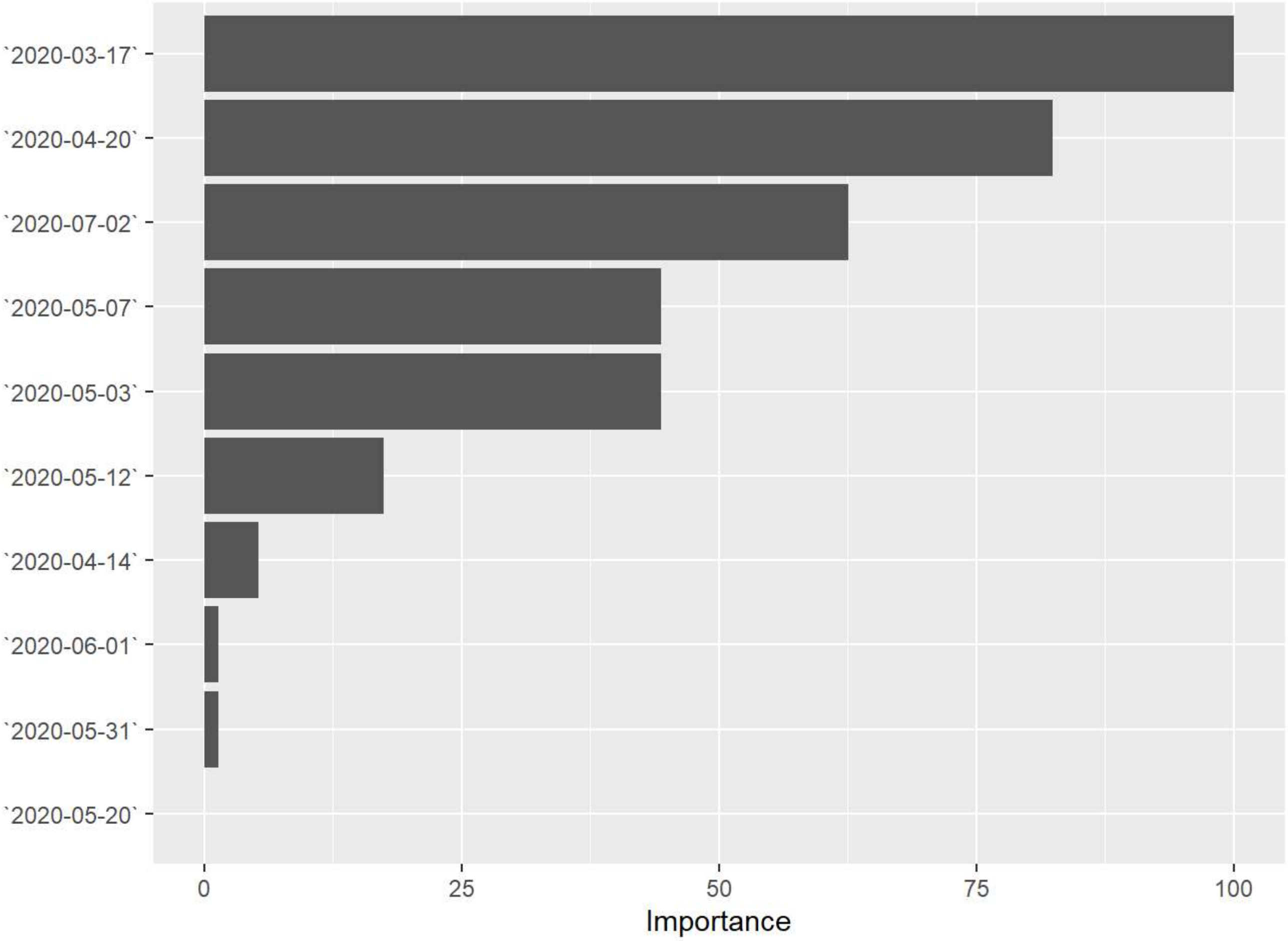
```
## Flexible Discriminant Analysis
##
## 143 samples
## 125 predictors
##   8 classes: 'NNN', 'NNY', 'none', 'NRY', 'NYY', 'YNY', 'YRY', 'YYY'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 116, 112, 115, 115, 114
## Resampling results across tuning parameters:
##
##   degree  nprune  Accuracy   Kappa
##   1         50    0.7488847  0.6895604
##   1        100    0.7488847  0.6895604
##   1        300    0.7488847  0.6895604
##   1       1000    0.7488847  0.6895604
##   2         50    0.7427636  0.6823946
##   2        100    0.7427636  0.6823946
##   2        300    0.7427636  0.6823946
##   2       1000    0.7427636  0.6823946
##   3         50    0.7641921  0.7089764
##   3        100    0.7641921  0.7089764
##   3        300    0.7641921  0.7089764
##   3       1000    0.7641921  0.7089764
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 3 and nprune = 50.
```

```
plot(fdaOut)
```



Visualize feature importance

```
fdaVip = vip(fdaOut,num_features = 25, bar = FALSE, metric = "Accuracy")
plot(fdaVip)
```



```
head(coef(fdaOut$finalModel))
```

```
##                                              [,1]       [,2]       [,3]
## (Intercept)                           -23.7505542  0.6664888  4.3142581
## h(0.60571-`2020-03-17`)                54.1307411 -5.1856839 -2.9237637
## h(0.60571-`2020-03-17`)*`2020-04-20`   -0.2927487 22.1588806  8.0977819
## h(0.537715-`2020-05-03`)               -0.6934318 -2.8327676  3.5508055
## `2020-05-03`*h(0.869302-`2020-05-07`)  -1.8440657  5.2572424  1.4483704
## h(1.03552-`2020-05-20`)                -0.1632763 -0.1149881  0.3324705
##                                             [,4]       [,5]       [,6]
## (Intercept)                           -0.7499315 -0.1594524 -2.3592351
## h(0.60571-`2020-03-17`)                0.1480197  0.1046893  0.1136816
## h(0.60571-`2020-03-17`)*`2020-04-20`  -5.2252562 -0.9117866  0.3530192
## h(0.537715-`2020-05-03`)               1.1716123  8.1592222  9.1757452
## `2020-05-03`*h(0.869302-`2020-05-07`) 19.8326231 12.0330296  7.0376765
## h(1.03552-`2020-05-20`)               -0.7247700  3.2443717 -2.8510321
##                                             [,7]
## (Intercept)                           -0.2185078
## h(0.60571-`2020-03-17`)               -0.3372721
## h(0.60571-`2020-03-17`)*`2020-04-20`  -0.6879373
## h(0.537715-`2020-05-03`)               3.4332204
## `2020-05-03`*h(0.869302-`2020-05-07`)  5.6336649
## h(1.03552-`2020-05-20`)               -0.6054813
```

Visualize the importance object we computed with *vip*

```
important    = fdaVip$data$Variable[fdaVip$data$Importance > 1e-16]
importantVal = fdaVip$data$Importance[fdaVip$data$Importance > 1e-16]

importantIndex = sapply(strsplit(important,'day'),function(x){return(as.numeric(x[2])+1)})
importantDigit = rep(0,125**2)

df <- data.frame(Important = important, Importance_Metric = importantVal)
print(df)
```

```
##      Important Importance_Metric
## 1 `2020-03-17`        100.000000
## 2 `2020-04-20`         82.463444
## 3 `2020-07-02`         62.618268
## 4 `2020-05-03`         44.362249
## 5 `2020-05-07`         44.362249
## 6 `2020-05-12`         17.380579
## 7 `2020-04-14`          5.213366
## 8 `2020-05-31`          1.320635
## 9 `2020-06-01`          1.320635
```

```
probHatTestFDA = predict(fdaOut$finalModel, Xtest, type='posterior')
YhatTestFDA = apply(probHatTestFDA,1,which.max) %>% "["(colnames(probHatTestGlmnet),.)
```

```
# Stop parallel processing
#stopCluster(cl)
#registerDoSEQ()
```

The confusion matrices

```
(tabGlmnet = table(YhatTestGlmnet, Ytest)) #### Answer 2.3.2
```

```
##                Ytest
## YhatTestGlmnet NNN NNY none NRY NYY YNY YRY YYY
##           NNN   10   0    0   0   0   0   0   0
##           NNY    0   3    0   1   0   0   0   0
##           none   0   0   10   0   0   0   0   0
##           NYY    0   0    0   0   2   0   0   0
##           YNY    0   0    0   0   0   2   1   1
##           YYY    0   0    0   0   0   1   0   1
```

```
(tabFDA    = table(YhatTestFDA, Ytest)) #### Answer 2.3.3
```

```
##             Ytest
## YhatTestFDA NNN NNY none NRY NYY YNY YRY YYY
##        NNN   10   0    0   0   0   1   0   0
##        NNY    0   2    0   0   0   0   0   0
##        none   0   0   10   0   0   0   0   0
##        NRY    0   0    0   1   0   0   0   0
##        NYY    0   1    0   0   2   0   0   0
##        YNY    0   0    0   0   0   1   1   1
##        YYY    0   0    0   0   0   1   0   1
```