Chicago Crime Report

Luiz Parlato

May 31, 2020

1 Overview

The reduction in crimes is the object of any developed society in the 21st century. Crimes are, for the most part, very delicate situations that involve a range of factors which influence the act of making an arrest or not.

For an understanding of this topic, we used the information on crimes in chicago (https://www.kaggle.com/armyaviator/chicago-crime-dataset-2001-present/version/1) from 2001 to 2018 where the **objective** was to classify the observations with respect to the variable `Arrest` in `TRUE` or `FALSE`.

1.1 Loading Necessary Packages

```
## loading packages
library(data.table) ## manipulation
library(ggplot2) ## visualization
library(GGally) ## Descriptive Visualization
library(plyr) ## aggregation
library(stringr) ## string manipulation
library(glmnet) ## Cross-Validation, Ridge and Lasso Regression
library(lubridate) ## working with dates
library(xtable) ## latex tables
library(caret) ## confusion Matrix
library(e1071) ## dependency of confusion matrix
```

1.2 Data

The data come from Kaggle Repository in a zipped format, it was uploaded on my github with name `crimes.zip`. Inside the zipped folder we find the file `Crimes_-_2001_to_present.csv`. The Following code unzip the folder, load the `.csv` file as data.table on R environment and then, remove the `.csv` file to save space on hard-drive.

```
crimeData <- fread(unzip("crimes.zip"))
file.remove("Crimes_-_2001_to_present.csv")
```

```
## [1] TRUE
```

To have a nice undersanting of the data, the first five observations are shown below:

ID <int>	Case Number <chr>	Date <chr>	Block <chr>	IUCR <chr>
11034701	JA366925	01/01/2001 11:00:00 AM	016XX E 86TH PL	1153
11162428	JA529032	11/28/2017 09:43:00 PM	026XX S CALIFORNIA BLVD	5131
11175304	JA545986	12/11/2017 07:15:00 PM	007XX N SACRAMENTO BLVD	031A
11227287	JB147188	10/08/2017 03:00:00 AM	092XX S RACINE AVE	0281
11227583	JB147595	03/28/2017 02:00:00 PM	026XX W 79TH ST	0620

5 rows

Primary Type <chr>	Description <chr>	Location Description <chr>	Arrest <chr>	Domestic <chr>
DECEPTIVE PRACTICE	FINANCIAL IDENTITY THEFT OVER $ 300	RESIDENCE	FALSE	FALSE
OTHER OFFENSE	VIOLENT OFFENDER: ANNUAL REGISTRATION	JAIL / LOCK-UP FACILITY	TRUE	FALSE
ROBBERY	ARMED: HANDGUN	SIDEWALK	TRUE	FALSE
CRIM SEXUAL ASSAULT	NON-AGGRAVATED	RESIDENCE	FALSE	FALSE
BURGLARY	UNLAWFUL ENTRY	OTHER	FALSE	FALSE

5 rows

Beat <int>	District <int>	Ward <int>	Community Area <int>	FBI Code <chr>	X Coordinate <int>	Y Coordinate <int>
412	4	8	45	11	NA	NA
1034	10	12	30	26	1158280	1886310
1221	12	27	23	03	1156092	1904769

Beat <int>	District <int>	Ward <int>	Community Area <int>	FBI Code <chr>	X Coordinate <int>	Y Coordinate <int>
2222	22	21	73	02	NA	NA
835	8	18	70	05	NA	NA

5 rows

Year <int>	Updated On <chr>	Latitude <dbl>	Longitude <dbl>	Location <chr>
2001	08/05/2017 03:50:08 PM	NA	NA	
2017	02/11/2018 03:54:58 PM	41.84378	-87.69464	(41.843778126, -87.694637678)
2017	02/11/2018 03:54:58 PM	41.89448	-87.70217	(41.894475919, -87.702169158)
2017	02/11/2018 03:57:41 PM	NA	NA	
2017	02/11/2018 03:57:41 PM	NA	NA	

5 rows

1.2.1 Adapting Data

We see from the tables before that `Date` variable is being read as `character` and should be considered as `Date` instead. `Arrest`, `Domestic`, `IUCR` and `FBI Code` are also being read as `character` while factor would be the best choice (they are all categorical variables).

```
#### Date
crimeData$Date <- as.Date(crimeData$Date, format = "%m/%d/%Y %I:%M:%S %p")
crimeData$year <- year(crimeData$Date)
crimeData$month <- month(crimeData$Date)
crimeData$day <- day(crimeData$Date)

#### Arrest, Domestic, IUCR and FBI Code as factor
crimeData$Arrest <- as.factor(crimeData$Arrest)
crimeData$Domestic <- as.factor(crimeData$Domestic)
crimeData$IUCR <- as.factor(crimeData$IUCR)
crimeData$`FBI Code` <- as.factor(crimeData$`FBI Code`)
```

1.2.2 Splitting Data

In order to be able to test our estimate models the data was first randomly divide into 3 data sets,

- **training set** with 80% of all observations, it is used to train possible models.
- **tuning set** with 10% of all observations, it is used to tune the threshold for the classification problem.
- **testing set** with 10% of all observations, it is used to test the performance of chosen model with chosen threshold.

Those sizes of each set was decided to be on proportion 8/1/1 in order to have enough data for modeling on training set and also enough data to tune and test our model.

```
set.seed(1234)
obs_out <- sample(1:nrow(crimeData), round(nrow(crimeData)*0.2))

obs_out_tunning <- obs_out[1:(length(obs_out)/2)]
obs_out_testing <- obs_out[((length(obs_out)/2) + 1):length(obs_out)]

training <- crimeData[!obs_out,]
testing <- crimeData[obs_out_testing,]
tuning <- crimeData[obs_out_tunning,]
```

1.2.3 Summary Statistics

Summary Statistics were generated on training set for evaluate parameters.

	ID <fctr>	Case Number <fctr>	Date <fctr>	Block <fctr>	IUCR <fctr>
X	Min. : 635	Length:5329232	Min. :2001-01-01	Length:5329232	0820 : 430893
X.1	1st Qu.: 3399067	Class :character	1st Qu.:2004-06-21	Class :character	0486 : 407800
X.2	Median : 6141319	Mode :character	Median :2008-03-11	Mode :character	0460 : 398300
X.3	Mean : 6157460	NA	Mean :2008-09-09	NA	1320 : 287518
X.4	3rd Qu.: 8741198	NA	3rd Qu.:2012-07-30	NA	1310 : 280538
X.5	Max. :11397866	NA	Max. :2018-07-24	NA	0810 : 274405
X.6	NA	NA	NA	NA	(Other):3249778

7 rows

	Primary Type <fctr>	Description <fctr>	Location Description <fctr>	Arrest <fctr>	Domestic <fctr>
X	Length:5329232	Length:5329232	Length:5329232	FALSE:3845719	FALSE:4631772
X.1	Class :character	Class :character	Class :character	TRUE :1483513	TRUE : 697460
X.2	Mode :character	Mode :character	Mode :character	NA	NA

3 rows

	Beat <fctr>	District <fctr>	Ward <fctr>	Community Area <fctr>	FBI Code <fctr>
X	Min. : 111	Min. : 1.0	Min. : 1.0	Min. : 0.0	06 :1116955
X.1	1st Qu.: 622	1st Qu.: 6.0	1st Qu.:10.0	1st Qu.:23.0	08B : 832229
X.2	Median :1111	Median :10.0	Median :22.0	Median :32.0	14 : 610615
X.3	Mean :1193	Mean :11.3	Mean :22.7	Mean :37.6	26 : 544160
X.4	3rd Qu.:1731	3rd Qu.:17.0	3rd Qu.:34.0	3rd Qu.:58.0	18 : 530757
X.5	Max. :2535	Max. :31.0	Max. :50.0	Max. :77.0	05 : 307496
X.6	NA	NA's :38	NA's :491884	NA's :492842	(Other):1387020

7 rows

	X Coordinate <fctr>	Y Coordinate <fctr>	Year <fctr>	Updated On <fctr>	Latitude <fctr>
X	Min. : 0	Min. : 0	Min. :2001	Length:5329232	Min. :36.62
X.1	1st Qu.:1152932	1st Qu.:1859190	1st Qu.:2004	Class :character	1st Qu.:41.77
X.2	Median :1165964	Median :1890516	Median :2008	Mode :character	Median :41.86
X.3	Mean :1164506	Mean :1885712	Mean :2008	NA	Mean :41.84
X.4	3rd Qu.:1176352	3rd Qu.:1909339	3rd Qu.:2012	NA	3rd Qu.:41.91
X.5	Max. :1205119	Max. :1951622	Max. :2018	NA	Max. :42.02
X.6	NA's :47336	NA's :47336	NA	NA	NA's :47336

7 rows

	Longitude <fctr>	Location <fctr>	year <fctr>	month <fctr>	day <fctr>
X	Min. :-91.69	Length:5329232	Min. :2001	Min. : 1.000	Min. : 1.00
X.1	1st Qu.:-87.71	Class :character	1st Qu.:2004	1st Qu.: 4.000	1st Qu.: 8.00
X.2	Median :-87.67	Mode :character	Median :2008	Median : 7.000	Median :16.00
X.3	Mean :-87.67	NA	Mean :2008	Mean : 6.504	Mean :15.62
X.4	3rd Qu.:-87.63	NA	3rd Qu.:2012	3rd Qu.: 9.000	3rd Qu.:23.00
X.5	Max. :-87.52	NA	Max. :2018	Max. :12.000	Max. :31.00
X.6	NA's :47336	NA	NA	NA	NA

7 rows

1.2.4 Selecting Features

First, variables with many `NA` values were dropped.

```
training <- training[,-c(13,14,16:22)]
testing <- testing[,-c(13,14,16:22)]
tuning <- tuning[,-c(13,14,16:22)]
```

Second, `Id` and `case number` which are just identities of each observation and are different for each observations do not need to be in any model. Also, `primary type` and `description` do not need to be as well since they are both expressed by `IUCR` column.

```
training <- training[,-c(1,2,6,7)]
testing <- testing[,-c(1,2,6,7)]
tuning <- tuning[,-c(1,2,6,7)]
```

Since `block` and `location description` have many classes and the information of them can be approximated using `Beat` and `District` , it was decided to use just the latters.

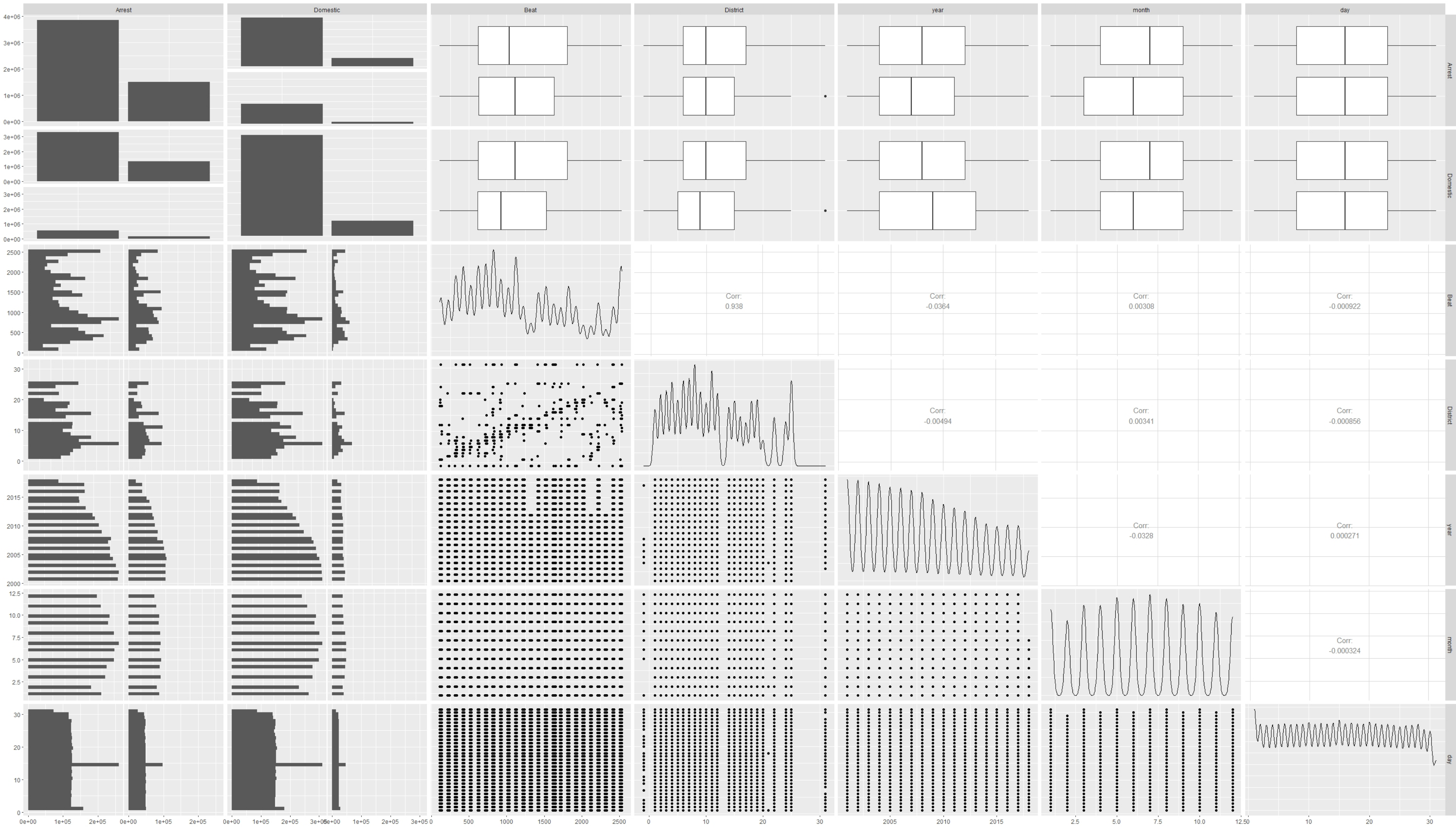
```
training <- training[,-c(1,2,4)]
testing <- testing[,-c(1,2,4)]
tuning <- tuning[,-c(1,2,4)]
```

Besides the data has 6661540 observations and just a little few does not have any value and then, I decided to treat them as -1.

```
training$District[is.na(training$District)] <- -1
testing$District[is.na(testing$District)] <- -1
tuning$District[is.na(tuning$District)] <- -1
```

1.2.4.1 Visualizing Data on Correlation Matrix

```
ggpairs(training[,-c(1,6)])
```



Correlation Matrix

The Correlation Matrix gives to us an idea about how the variables are correleted in pairs. It is specially useful to have an idea about the distribution of the `Arrest` label among each feature.

So, we see that:

- The proportion of falses/trues of `Arrest` is about 3/1; (cell c(1,1) on Correlation Matrix)
- `Arrest` cases look to happen more often when the class of `Domestic` is false; (cell c(2,1) on Correlation Matrix)
- `Beat`, `District` and `Day` look to have the same distribution between trues and falses for `Arrest` variable; (cells c(1,3), c(1,4), c(1,7) on Correlation Matrix)
- `year` shows a decreasing aspect when `Arrest` is equal true but shows an encreasing aspect when `Arrest` is equal false; (cell c(5,1) on Correlation Matrix)

2 Analysis

First, The `training` set was splitted in `train` and `test` sets with 70% and 30% of observations each, respectively, so we could produce a **label vector** and **features matrix** originated from `train` set. The proportion of `train` and `test` sizes 7/3 was decided in order to evaluate how generic our model can be.

Second, The first model is generated using a binomial logistic regression and then, ridge regression and lasso regression with 10-fold cross-validation is performed in order to find the best lambda value that minimizes our classification error by F-Meausre.

Third, the `training` set is used in full for a cross-validation analysis where the data is split in 10 folds and each time we take one fold out and estimate coefficients using 9 folds predicting the one that was out. By this way, we will have coefficients and F-Mesures for 10 models, so we get the mean and the variance among all models of each beta to check if models are "generic" or "too especific" for the data.

Fourth, the model created with the mean of each beta is used to predict `Arrest` in the `tuning` set to evaluate the best threshold that segregate classes and then, the chosen threshold is used to predict `Arrest` in the `testing` set.

2.1 F-Measure

The F-Measure (or F-Score) is a measure used for model selection which gives a balance between precision and recall,

$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Where,

- **Recall.** Number of True Positives divided by the number of True Positives and the number of False Negatives.

- **Precision.** Number of True Positives divided by the number of True Positives and the number of False Positives.

2.2 Splitting Training Set

2.2.1 create second train and test set based on training

```
set.seed(4321)
obs_out2 <- sample(1:nrow(training), round(nrow(training)*0.3))

train <- training[!obs_out2,]
test <- training[obs_out2,]
```

2.2.2 separating label from features

```
cat_labels <- train[,2]
features <- train[,-2]
```

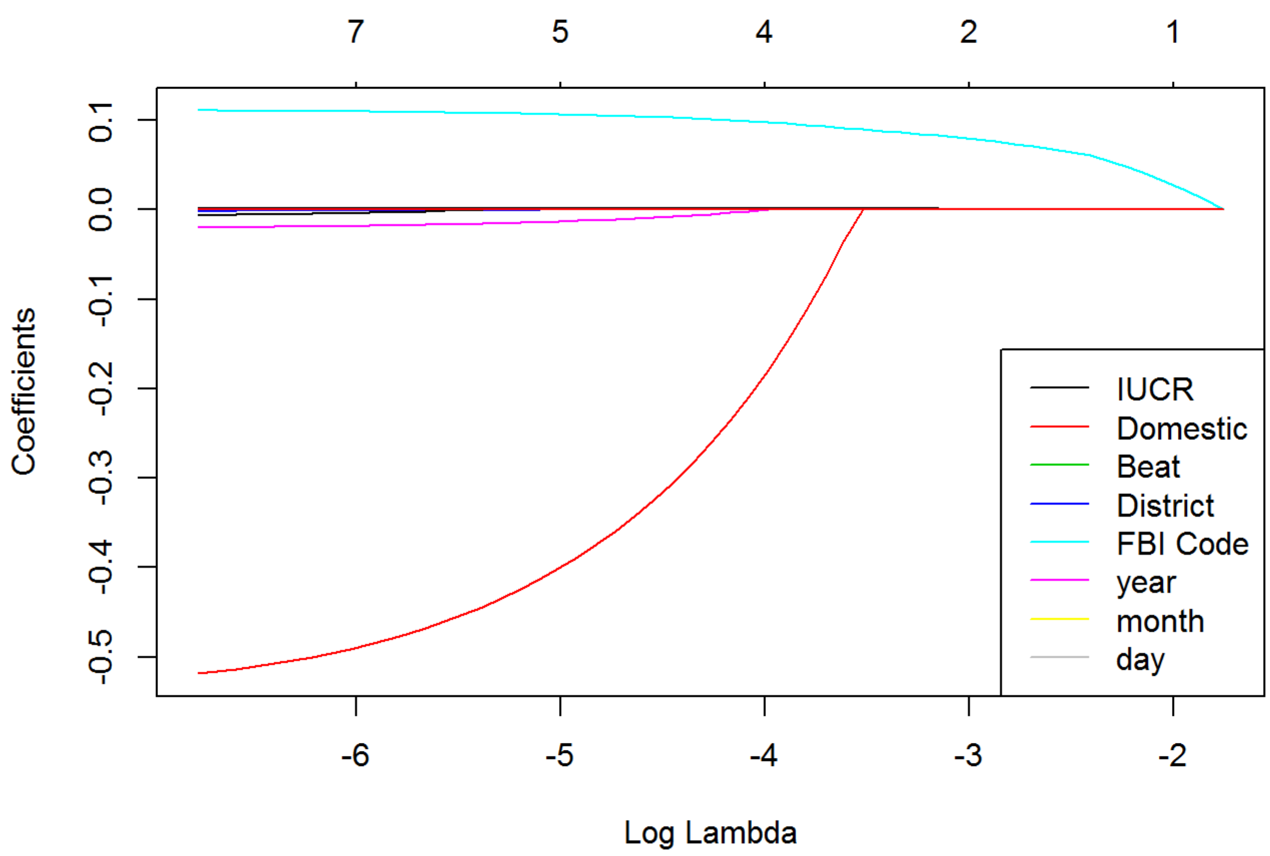
2.2.2.1 function for adding legend to fit

```
lbs_fun <- function(fit, ...) {
  L <- length(fit$lambda)
  x <- log(fit$lambda[L])
  y <- fit$beta[, L]
  labs <- names(y)
  legend('bottomright', legend=labs, col=1:length(labs), lty=1)
}
```

2.3 training binomial logistic regression for arrests

```
cvfit <- glmnet(data.matrix(features),
                cat_labels$Arrest,
                family = "binomial")

plot(cvfit, xvar = "lambda")
lbs_fun(cvfit)
```




```
predict_min_lambda <- predict(cvfit, newx = data.matrix(test[,-2]), s= 0.01, type = "class")
```

The plot shows that `Domestic` is highly affected by low values of Log Lambda in comparison to the others. `FBI Code` also shows to range for low values of Log Lambda. `year` is just a few affected while the others coefficients all almost go to zero.

2.3.1 calculation on F-measure for the prediction

```
cm_regular <-confusionMatrix(as.factor(predict_min_lambda), reference = as.factor(test$Arrest))
f_byclass_regular <- cm_regular[["byClass"]][["F1"]]

cm_regular ## checking how good was the model using whole train set
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   FALSE    TRUE
##      FALSE 1049076  284276
##      TRUE   104991  160427
##
##                Accuracy : 0.7565
##                  95% CI : (0.7559, 0.7572)
##     No Information Rate : 0.7218
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3079
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9090
##             Specificity : 0.3608
##          Pos Pred Value : 0.7868
##          Neg Pred Value : 0.6044
##              Prevalence : 0.7218
##          Detection Rate : 0.6562
##    Detection Prevalence : 0.8340
##       Balanced Accuracy : 0.6349
##
##        'Positive' Class : FALSE
##
```

```
f_byclass_regular ##  f-measure
```

```
## [1] 0.8435057
```

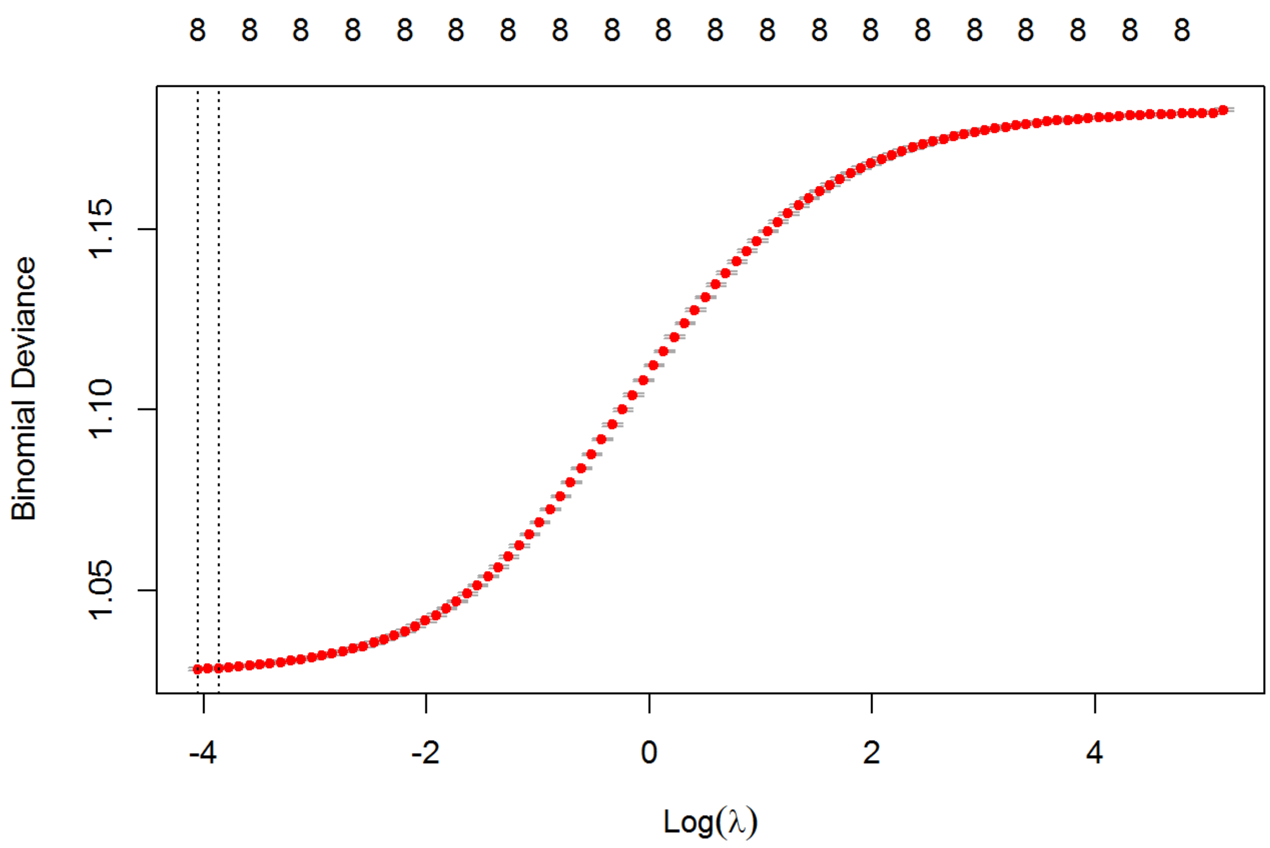
We see an F-Measure equal to 0.8435 with Sensitivity equal to 0.9090 and Specificity equal to 0.3608. So, it is doing a good job finding true positives, however, it is not as good as finding true negatives.

2.4 Cross-Validating with 10-Fold for Ridge or Lasso

A cross-validation is performerd in order to estimate the best **lambda** for Ridge and Lasso Classification. ### training binomial logistic regression ridge with cross-validation 10-fold

```
cvfit10 <- cv.glmnet(data.matrix(features),
                     cat_labels$Arrest,
                     nfolds = 10,
                     alpha = 0,
                     family = "binomial")

plot(cvfit10)
```




```
cvfit10$lambda.min
```

```
## [1] 0.0173234
```

```
cvfit10$lambda.1se
```

```
## [1] 0.02086609
```

The plot shows that the best lambda is around 0 with 8 parameters. This idea is reinforced by the value of lambda.min which is 0.0173 and the value of lambda.1se which is 0.0208.

2.4.0.1 calculation on F-measure for the prediction

```
predict_min_lambda <- predict(cvfit10, newx = data.matrix(test[,-2]),
                              s= "lambda.min", type = "class")
cm_ridge <- confusionMatrix(as.factor(predict_min_lambda),
                            reference = as.factor(test$Arrest))
f_byclass_ridge <- cm_ridge[["byClass"]][["F1"]]

cm_ridge ## checking model
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   FALSE    TRUE
##      FALSE 1052066  255332
##      TRUE   102001  189371
##
##                Accuracy : 0.7765
##                  95% CI : (0.7758, 0.7771)
##     No Information Rate : 0.7218
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3775
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9116
##             Specificity : 0.4258
##          Pos Pred Value : 0.8047
##          Neg Pred Value : 0.6499
##              Prevalence : 0.7218
##          Detection Rate : 0.6580
##    Detection Prevalence : 0.8178
##       Balanced Accuracy : 0.6687
##
##        'Positive' Class : FALSE
##
```

```
f_byclass_ridge ## f-measure
```

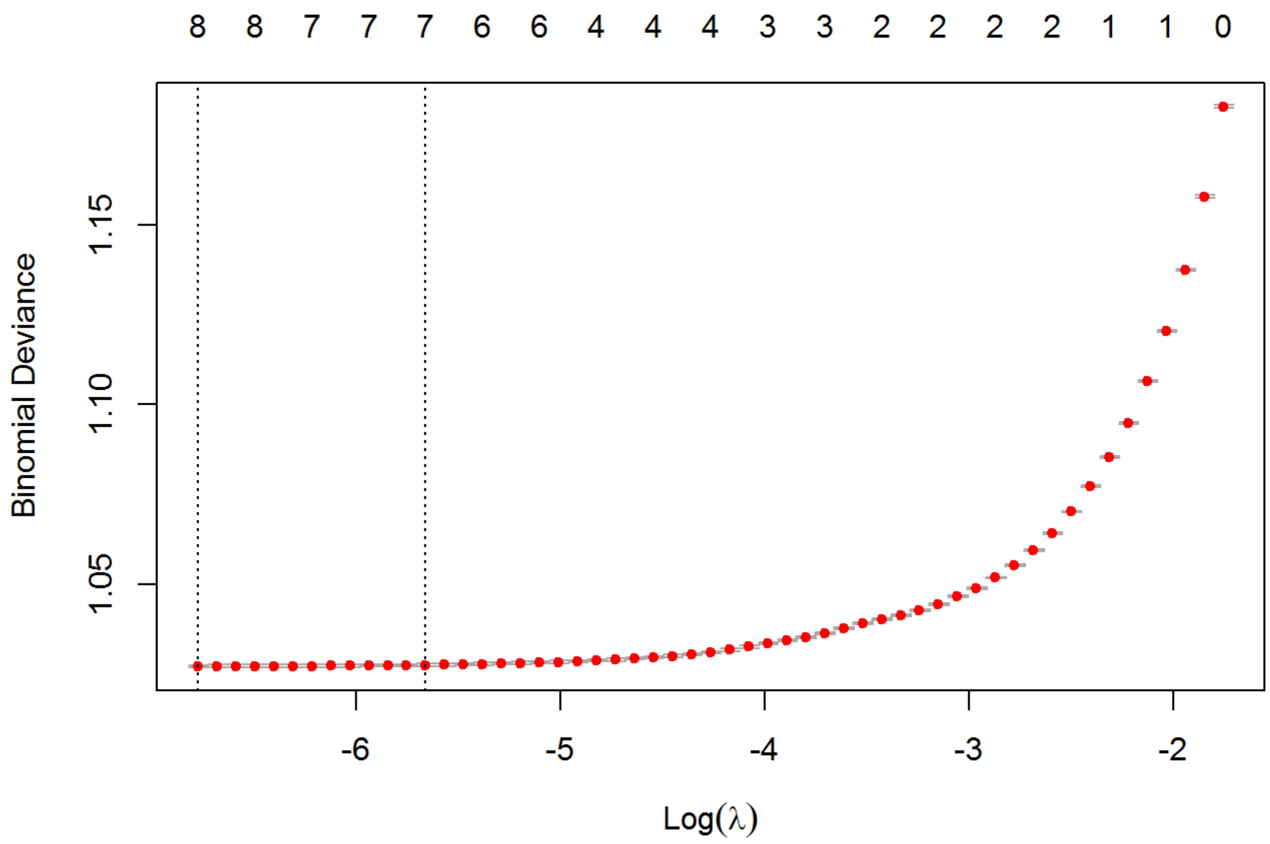
```
## [1] 0.8548291
```

We see an F-Measure equal to 0.8548, Sensitivity equal 0.9116 and Specificity equal 0.4258, all three measures were a little bit higher than in the previous model.

2.4.1 training multinomial logistic regression lasso with cross-validation 10-fold

```
cvfit10_lasso <- cv.glmnet(data.matrix(features),
                           cat_labels$Arrest,
                           nfolds = 10,
                           alpha = 1,
                           family = "binomial")

plot(cvfit10_lasso)
```




```
cvfit10_lasso$lambda.min
```

```
## [1] 0.001139764
```

```
cvfit10_lasso$lambda.1se
```

```
## [1] 0.003480674
```

The plot shows that the best lambda is around 0 with 8 or 7 parameters. This idea is reinforced by the value of lambda.min which is 0.0011 and the value of lambda.1se which is 0.0035.

2.4.1.1 calculation on F-measure for the prediction

```
predict_min_lambda <- predict(cvfit10_lasso, newx =data.matrix(test[,-2]),
                              s= "lambda.min", type = "class")
cm_lasso <- confusionMatrix(as.factor(predict_min_lambda),
                            reference = as.factor(test$Arrest))
f_byclass_lasso <- cm_lasso[["byClass"]][["F1"]]

cm_lasso ## checking model
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    FALSE    TRUE
##      FALSE 1050004  247150
##      TRUE   104063  197553
##
##                Accuracy : 0.7803
##                  95% CI : (0.7797, 0.781)
##     No Information Rate : 0.7218
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3929
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9098
##             Specificity : 0.4442
##          Pos Pred Value : 0.8095
##          Neg Pred Value : 0.6550
##              Prevalence : 0.7218
##          Detection Rate : 0.6568
##    Detection Prevalence : 0.8113
##       Balanced Accuracy : 0.6770
##
##        'Positive' Class : FALSE
##
```

```
f_byclass_lasso ## f-measure
```

```
## [1] 0.8567192
```

We see an F-Measure equal to 0.8567, Sensitivity equal to 0.9098 and Specificity equal 0.4442. Among all of three models until now, this one estimated by cross validation using lasso was with the highest F-Measure and Specificity, while it gets almost that same value for Sensitivity than the other two models.

Since not only predicting true positives but also true negatives is desirable, we chose the sigma-lambda minimum for the model regularized by lasso that gives to us the highest Specificity.

2.5 Overal Performance

```
rbind(f_byclass_regular,f_byclass_ridge,f_byclass_lasso)
```

```
##                        [,1]
## f_byclass_regular 0.8435057
## f_byclass_ridge   0.8548291
## f_byclass_lasso   0.8567192
```

Parcial Points:

1- The result shows an improvement when using ridge or lasso 'selection' instead of using the regular one.

2- Ridge and Lasso perfomerd well.

3- Among Ridge and Lasso, Laso was slightly better.

4- From the above, the chosen model was using lasso and lambda = 0.0011. (`cvfit10_lasso$lambda.min`)

2.6 10-Fold with all training set

After chosing lambda.min from `cvfit_lasso` as the best value for lambda, a 10-fold cross validation using at this time the whole `training` to produce estimates of parameters for 10 models.

```
data2 <- training ## backup

set.seed(54321)
shuffle_data <- data2[sample(nrow(data2)),] ## shuffleling data to create random folds

size9 <- ceiling(nrow(shuffle_data)/10) ## size of each fold
size10 <- nrow(shuffle_data) - size9*9
shuffle_data$fold <- c(rep(1:9, each = size9),rep(10,size10))

beta_i <- list()
f_byclass <- list()

for(i in 1:10){
  train_dummy <- shuffle_data[shuffle_data$fold != i,]
  test_dummy <- shuffle_data[shuffle_data$fold == i,]

  fit_dummy <- glmnet(data.matrix(train_dummy[,-c(2,10)]),
                      train_dummy$Arrest,
                      alpha = 1,
                      lambda = cvfit10_lasso$lambda.min,
                      family = "binomial")

  beta_i[[i]] <- coef(fit_dummy, s = "lambda.min")

  predict_dummy <- predict(fit_dummy, newx = data.matrix(test_dummy[,-c(2,10)]), type = "class")
  cm_dummy <- confusionMatrix(as.factor(predict_dummy), reference = as.factor(test_dummy$Arrest))
  f_byclass_dummy <- cm_dummy[["byClass"]][["F1"]]

  f_byclass[[i]] <- f_byclass_dummy
}
```

2.6.1 Mean for F-Meausre among all folds

```
f_data <- t(as.data.frame(f_byclass))
f_mean <- colMeans(f_data, na.rm = T)

f_mean
```

```
## [1] 0.8567073
```

The F-Measure for the model using the whole data was equal to 0.8567073. This value, besides is very close to the one calculated to the previous model, was a little higher.

2.6.2 Mean among folds for estimated betas

```
beta_mean <- data.frame(matrix(ncol = 10, nrow = 9))

for(i in 1:10){
    list_dummy <- as.matrix(beta_i[[i]])
    beta_mean[i] <- data.matrix(list_dummy)
}

## means beta
beta_final <- as.matrix(beta_mean) %>% apply(1,mean)

## variation on beta_final
var_beta_final <- as.matrix(beta_mean) %>% apply(1,var)

data.frame(variable = c("intercerpt",colnames(training[,-2])), beta_estimated = beta_final, var_beta_estimated = var_beta_fi
nal)
```

variable <fctr>	beta_estimated <dbl>	var_beta_estimated <dbl>
intercerpt	3.731126e+01	1.537522e-02
IUCR	1.632304e-03	7.375484e-11
Domestic	-5.206818e-01	1.156458e-06
Beat	-4.069875e-05	2.253924e-12
District	-1.365937e-03	2.455484e-08
FBI Code	1.112193e-01	1.856167e-08
year	-1.962435e-02	3.827002e-09
month	-6.059575e-03	1.218017e-08
day	2.873573e-04	1.429960e-09

9 rows

`beta_final` shows the values of the mean between all models created by the cross-validation of each estimate of parameter while `var_beta_final` shows the variance of the estimate.

It is interesting from the above that the parameters have very little variance which means that the parameters estimates were not too dependent of the data used to estimate them, so it is a generic model.

2.7 Predicting on tuning set for chosing threshold

The `tuning` set is now used for evaluate best threshold that segregates classes.

```
predict_tuning <- beta_final[1] + beta_final[2]*as.numeric(tuning$IUCR) +
                                 beta_final[3]*as.numeric(tuning$Domestic) +
                                 beta_final[4]*tuning$Beat +
                                 beta_final[5]*tuning$District +
                                 beta_final[6]*as.numeric(tuning$`FBI Code`) +
                                 beta_final[7]*tuning$year +
                                 beta_final[8]*tuning$month +
                                 beta_final[9]*tuning$day

summary(predict_tuning)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.3500 -1.7737 -1.4856 -1.1040 -0.4427  1.0185
```

The predicted values for `tuning` set using the parameters given by `beta_final` range from -3.35 to 1.0185 where 75% of the values where lower than -0.44, which makes interesting to test for `median`, `mean`, `3rd Qu`.

Also, would be interesting to see the percentage of arrest on both training and tuning set:

```
data_prop <- count(training$Arrest)
data_prop2 <- count(tuning$Arrest)
predict_tuning <- as.data.frame(predict_tuning)

quantile(predict_tuning$predict_tuning,
         data_prop$freq[data_prop$x == FALSE] / sum(data_prop$freq))
```

```
##  72.16272%
## -0.5164168
```

```
quantile(predict_tuning$predict_tuning,
         data_prop2$freq[data_prop2$x == FALSE] / sum(data_prop2$freq))
```

```
##  72.13317%
## -0.5171714
```

Both sets have around 72% of cases without any arrest made, so a possible threshould would be given by the percentile 72.

2.7.1 Testing Thresholds

```
## testing threshold = median
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -1.4856] <- TRUE

cm_tuning_median <- confusionMatrix(as.factor(predict_tuning$true), reference = as.factor(testing$Arrest))
f_byclass_tuning_median <- cm_tuning_median[["byClass"]][["F1"]]

## testing threshold = mean
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -1.104] <- TRUE

cm_tuning_mean <- confusionMatrix(as.factor(predict_tuning$true), reference = as.factor(testing$Arrest))
f_byclass_tuning_mean <- cm_tuning_mean[["byClass"]][["F1"]]


## testing threshold = -0.51 given by proprotion
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -0.51] <- TRUE

cm_tuning_51 <- confusionMatrix(as.factor(predict_tuning$true), reference = as.factor(testing$Arrest))
f_byclass_tuning_51 <- cm_tuning_51[["byClass"]][["F1"]]

## testing threshold = -0.44
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -0.44] <- TRUE

cm_tuning_44 <- confusionMatrix(as.factor(predict_tuning$true), reference = as.factor(testing$Arrest))
f_byclass_tuning_44 <- cm_tuning_44[["byClass"]][["F1"]]

xtable(cm_tuning_median$table, caption = "Median = -1.4856")
```

	FALSE <int>	TRUE <int>
FALSE	240348	92726
TRUE	240460	92620

2 rows

```
xtable(cm_tuning_mean$table, caption = "Mean = -1.10")
```

	FALSE <int>	TRUE <int>
FALSE	296111	114163
TRUE	184697	71183

2 rows

```
xtable(cm_tuning_51$table, caption = "Percentile 72 = -0.51")
```

	FALSE <int>	TRUE <int>
FALSE	348169	134351
TRUE	132639	50995

2 rows

```
xtable(cm_tuning_44$table, caption = "Percentile 75 = -0.44")
```

	FALSE <int>	TRUE <int>
FALSE	361013	139154
TRUE	119795	46192

2 rows

```r
xtable(rbind(f_byclass_tuning_median, f_byclass_tuning_mean, f_byclass_tuning_51, f_byclass_tuning_44))
```

	x <dbl>
f_byclass_tuning_median	0.5906212
f_byclass_tuning_mean	0.6646100
f_byclass_tuning_51	0.7228462
f_byclass_tuning_44	0.7360290

4 rows

The Confusion Matrices above show that increasing in the value of the threshould gave a better classification.

3 Results

3.1 Predicting on testing set using Beta Mean Model and threshold (-0.44)

```r
predict_final <- beta_final[1] + beta_final[2]*as.numeric(testing$IUCR) +
                                 beta_final[3]*as.numeric(testing$Domestic) +
                                 beta_final[4]*testing$Beat +
                                 beta_final[5]*testing$District +
                                 beta_final[6]*as.numeric(testing$`FBI Code`) +
                                 beta_final[7]*testing$year +
                                 beta_final[8]*testing$month +
                                 beta_final[9]*testing$day

## testing threshold = -0.44
predict_final <- as.data.frame(predict_final)
predict_final$true <- FALSE
predict_final$true[predict_final$predict_final >= -0.51] <- TRUE

cm_final<- confusionMatrix(as.factor(predict_final$true), reference = as.factor(testing$Arrest))
f_byclass_final <- cm_final[["byClass"]][["F1"]]

cm_final$table
```

```
##           Reference
## Prediction  FALSE   TRUE
##      FALSE 410245  72418
##      TRUE   70563 112928
```

3.2 Predicting on testing using Ridge Regression Model

```r
## testing cvfit10 on testing
predict_min_lambda <- predict(cvfit10, newx =data.matrix(testing[,-2]), s= 'lambda.min', type = "class")
cm2 <- confusionMatrix(as.factor(predict_min_lambda), reference = as.factor(testing$Arrest))
f_byclass_ridge2 <- cm2[["byClass"]][["F1"]]

cm2$table
```

```
##           Reference
## Prediction  FALSE   TRUE
##      FALSE 438303 106466
##      TRUE   42505  78880
```

3.3 Predicting on testing using Lasso Regression Model

```r
## testing cvfit10_lasso on testing
predict_min_lambda <- predict(cvfit10_lasso, newx =data.matrix(testing[,-2]),  s= 'lambda.min', type = "class")
cm_lasso2 <- confusionMatrix(as.factor(predict_min_lambda), reference = as.factor(testing$Arrest))
f_byclass_lasso2 <- cm_lasso2[["byClass"]][["F1"]]

cm_lasso2$table
```

```
##           Reference
## Prediction  FALSE   TRUE
##      FALSE 437379 102947
##      TRUE   43429  82399
```

3.4 Predicting on testing using GLM Classification Model

```
## testing cvfit on testing
predict_min_lambda <- predict(cvfit, newx =data.matrix(testing[,-2]), s = 0.01, type = "class")
cm1 <- confusionMatrix(as.factor(predict_min_lambda), reference = as.factor(testing$Arrest))
f_byclass_1 <- cm1[["byClass"]][["F1"]]

cm1$table
```

```
##           Reference
## Prediction  FALSE   TRUE
##      FALSE 436968 118505
##      TRUE   43840  66841
```

3.5 Overall Performance

```
rbind(f_byclass_final, f_byclass_lasso2, f_byclass_ridge2, f_byclass_1)
```

```
##                       [,1]
## f_byclass_final  0.8515980
## f_byclass_lasso2 0.8566535
## f_byclass_ridge2 0.8547442
## f_byclass_1      0.8433388
```

4 Conclusions

- **Brief Summary.** This report searched to understand the scope of crimes in Chicago and to propose a model for classification of either an Arrest was made or not for each case of crime. Our findings showed that he best model was a logistic regression with lasso regularization generated by 10-fold cross-validation with lambda value equal to 0.0011 and threshold equal to -0.44. Their coefficents are:

```
xtable(data.frame(variable = c("intercerpt",colnames(training[,-2])), beta_estimated = beta_final, var_beta_estimated = var_
beta_final))
```

variable <fctr>	beta_estimated <dbl>	var_beta_estimated <dbl>
intercerpt	3.731126e+01	1.537522e-02
IUCR	1.632304e-03	7.375484e-11
Domestic	-5.206818e-01	1.156458e-06
Beat	-4.069875e-05	2.253924e-12
District	-1.365937e-03	2.455484e-08
FBI Code	1.112193e-01	1.856167e-08
year	-1.962435e-02	3.827002e-09
month	-6.059575e-03	1.218017e-08
day	2.873573e-04	1.429960e-09

9 rows

The column `beta_estimated` shows us that the features `IUCR`, `Beat`, `Disctirct` and `Day` were all shrinked to zero. Also, it shows that `Domestic` feature has the highest absolute coefficient, followed by `FBI Code` which is about 5 times smaller in magnitude. In a very small intesity, `year` and `month` affects in inverse magnitude when changing one unit.

From the `var_beta_estimated` we have the information that all 10 models fitted on by cross-validating the `training` set had parameters estimated very closely one to another, so have evidence to believe that our estimates are independent from the data used to train it.

- **Potential Impact.** Be able to predict if the case is going to result an `Arrest` using the parameters described in *Brief Summary* as regressors.
- **Limitations.** Features are too general and more individual characteristics of each case could improve classification.
- **Future Work.** A nice future work would be evaluating threshold for classification, also, would be trying other n-fold cross-validations to check for under or over fitting.

5 Code

```
## installing packages (Not using if(!require()) since I am using for sure all of them)
install.packages("data.table") ## manipulation
install.packages("ggplot2") ## visualization
install.packages("GGally") ## Descriptive Visualization
install.packages("plyr") ## aggregation
install.packages("stringr") ## string manipulation
install.packages("glmnet") ## Cross-Validation, Ridge and Lasso Regression
install.packages("lubridate") ## working with dates
install.packages("xtable") ## latex tables
install.packages("caret") ## confusion Matrix
install.packages("e1071") ## dependency of confusion matrix

## loading packages
library(data.table)
library(ggplot2)
library(GGally)
library(plyr)
library(stringr)
library(glmnet)
library(lubridate)
library(xtable)
library(caret)
library(e1071)


## loading data
crimeData <- fread(unzip("crimes.zip"))
file.remove("Crimes_-_2001_to_present.csv")

## Visualizing top 5 observations
xtable(crimeData[1:5,1:7])
xtable(crimeData[1:5,8:15])
xtable(crimeData[1:5,16:22])

set.seed(1234)
## adapting date
crimeData$Date <- as.Date(crimeData$Date, format = "%m/%d/%Y %I:%M:%S %p")
crimeData$year <- year(crimeData$Date)
crimeData$month <- month(crimeData$Date)
crimeData$day <- day(crimeData$Date)

## change block dimension to street dimension
crimeData$Block <- str_sub(crimeData$Block, start = 7)

## Arrest, Domestic, IUCR and FBI Code as factor
crimeData$Arrest <- as.factor(crimeData$Arrest)
crimeData$Domestic <- as.factor(crimeData$Domestic)
crimeData$IUCR <- as.factor(crimeData$IUCR)
crimeData$`FBI Code` <- as.factor(crimeData$`FBI Code`)

## training set, tunning set and test set (80% training, 10% tunning, 10% testing)
obs_out <- sample(1:nrow(crimeData), round(nrow(crimeData)*0.2))

obs_out_tunning <- obs_out[1:(length(obs_out)/2)]
obs_out_testing <- obs_out[((length(obs_out)/2) + 1):length(obs_out)]

training <- crimeData[!obs_out,]
testing <- crimeData[obs_out_testing,]
tuning <- crimeData[obs_out_tunning,]

############################
## Descriptive Statistics ##
############################

str(training) ## structure of variables
summary(training) ## summary of values

## Removing variables with NAs and not going to be used
training <- training[,-c(13,14,16:22)]
testing <- testing[,-c(13,14,16:22)]
tuning <- tuning[,-c(13,14,16:22)]

## drop id, case number, (primary type and description - using IUCR)
training <- training[,-c(1,2,6,7)]
testing <- testing[,-c(1,2,6,7)]
tuning <- tuning[,-c(1,2,6,7)]

## drop block and location description since usinb Beat and District
training <- training[,-c(1,2,4)]
testing <- testing[,-c(1,2,4)]
tuning <- tuning[,-c(1,2,4)]

## visualizing correlations of Date and Domestic with Arrest (Label)
```

```r
ggpairs(training[,-c(1,6)])

## Visualizing IUCR
ggplot(training) + geom_bar(aes(factor(IUCR))) +
  theme(axis.text.x = element_text(angle = 90))

## Visualizing FBI Code
ggplot(training) + geom_bar(aes(factor(`FBI Code`)))

####################
## Modeling Data ###
####################

## handling NA observations for Disctrit
training$District[is.na(training$District)] <- -1
testing$District[is.na(testing$District)] <- -1
tuning$District[is.na(tuning$District)] <- -1

## create second train and test set based on training
set.seed(4321)
obs_out2 <- sample(1:nrow(training), round(nrow(training)*0.3))

train <- training[!obs_out2,]
test <- training[obs_out2,]

## separating label from features
cat_labels <- train[,2]
features <- train[,-2]

## function for adding legend to fit
lbs_fun <- function(fit, ...) {
  L <- length(fit$lambda)
  x <- log(fit$lambda[L])
  y <- fit$beta[, L]
  labs <- names(y)
  legend('bottomright', legend=labs, col=1:length(labs), lty=1) # <<< ADDED BY ME
}


## training binomial logistic regression for arrests
cvfit <- glmnet(data.matrix(features),
                cat_labels$Arrest,
                family = "binomial", trace.it = 1)

plot(cvfit, xvar = "lambda")
lbs_fun(cvfit)

coef(cvfit, s = 0.01)

predict_min_lambda <- predict(cvfit,
                              newx = data.matrix(test[,-2]), s= 0.01, type = "class")

### calculation on F-measure for the prediction
cm_regular <-confusionMatrix(as.factor(predict_min_lambda),
                             reference = as.factor(test$Arrest))
f_byclass_regular <- cm_regular[["byClass"]][["F1"]]

cm_regular ## checking how good was the model using whole train set
f_byclass_regular ## micro f-measure

############### Cross-Validating with 10-Fold. Why 10? To have enough data on each training
############### Checking if Ridge or Lasso could improve precision


## training binomial logistic regression ridge with cross-validation 10-fold
cvfit10 <- cv.glmnet(data.matrix(features),
                     cat_labels$Arrest,
                     nfolds = 10,
                     alpha = 0,
                     family = "binomial", trace.it = 1)

plot(cvfit10)

coef(cvfit10, s = "lambda.min")

predict_min_lambda <- predict(cvfit10,
                              newx = data.matrix(test[,-2]),
                              s= "lambda.min", type = "class")
cm_ridge <- confusionMatrix(as.factor(predict_min_lambda),
                            reference = as.factor(test$Arrest))
f_byclass_ridge <- cm_ridge[["byClass"]][["F1"]]

cm_ridge ## checking model for under or overfitting
```

```
f_byclass_ridge ## micro f-measure

## training multinomial logistic regression lasso with cross-validation 10-fold
cvfit10_lasso <- cv.glmnet(data.matrix(features),
                           cat_labels$Arrest,
                           nfolds = 10,
                           alpha = 1,
                           family = "binomial", trace.it = 1)

plot(cvfit10_lasso)

coef(cvfit10_lasso, s = "lambda.min")

predict_min_lambda <- predict(cvfit10_lasso, newx =data.matrix(test[,-2]),
                              s= "lambda.min", type = "class")
cm_lasso <- confusionMatrix(as.factor(predict_min_lambda),
                            reference = as.factor(test$Arrest))
f_byclass_lasso <- cm_lasso[["byClass"]][["F1"]]

cm_lasso ## checking model for under or overfitting
f_byclass_lasso ## micro f-measure

########
rbind(f_byclass_regular,f_byclass_ridge,f_byclass_lasso)

#####################################
##### Parcial Points:
##
## 1- The result shows an improvement when using ridge or
##    lasso 'selection' instead of using the regular one.
## 2- Ridge and Lasso perfomerd well.
## 3- Among Ridge and Lasso, Laso was slightly better.
##    From the above, the chosen model was using lasso and
##    lambda = 0.001141191 (cvfit10_lasso$lambda.min)

#########################
#### 10-fold all data ###
#########################

data2 <- training ## backup

set.seed(54321) ## useful for creating simulations that can be reproduced
shuffle_data <- data2[sample(nrow(data2)),] ## shuffleling data to create random folds

size9 <- ceiling(nrow(shuffle_data)/10) ## size of each fold
size10 <- nrow(shuffle_data) - size9*9
shuffle_data$fold <- c(rep(1:9, each = size9),rep(10,size10))

beta_i <- list()
f_byclass <- list()

for(i in 1:10){
  train_dummy <- shuffle_data[shuffle_data$fold != i,]
  test_dummy <- shuffle_data[shuffle_data$fold == i,]

  fit_dummy <- glmnet(data.matrix(train_dummy[,-c(2,10)]),
                      train_dummy$Arrest,
                      alpha = 1,
                      lambda = cvfit10_lasso$lambda.min,
                      family = "binomial", trace.it = 1)

  beta_i[[i]] <- coef(fit_dummy, s = "lambda.min")

  predict_dummy <- predict(fit_dummy,
                           newx = data.matrix(test_dummy[,-c(2,10)]), type = "class")
  cm_dummy <- confusionMatrix(as.factor(predict_dummy),
                              reference = as.factor(test_dummy$Arrest))
  f_byclass_dummy <- cm_dummy[["byClass"]][["F1"]]

  f_byclass[[i]] <- f_byclass_dummy
}

#as.data.frame(beta_i[[1]]$`1`)

f_data <- t(as.data.frame(f_byclass))
f_mean <- colMeans(f_data, na.rm = T)

f_mean
f_byclass_lasso

beta_mean <- data.frame(matrix(ncol = 10, nrow = 9))

for(i in 1:10){
```

```
    list_dummy <- as.matrix(beta_i[[i]])
    beta_mean[i] <- data.matrix(list_dummy)
}

## means beta
beta_final <- as.matrix(beta_mean) %>% apply(1,mean)
beta_final

## variation on beta_final
var_beta_final <- as.matrix(beta_mean) %>% apply(1,var)
var_beta_final

## comparing coef
coef(cvfit10_lasso)
as.matrix(beta_final)

##########################################################################
## Predicting on tuning set using beta_final for tuning threshold class ##
##########################################################################

predict_tuning <- beta_final[1] + beta_final[2]*as.numeric(tuning$IUCR) +
                                 beta_final[3]*as.numeric(tuning$Domestic) +
                                 beta_final[4]*tuning$Beat +
                                 beta_final[5]*tuning$District +
                                 beta_final[6]*as.numeric(tuning$`FBI Code`) +
                                 beta_final[7]*tuning$year +
                                 beta_final[8]*tuning$month +
                                 beta_final[9]*tuning$day

summary(predict_tuning)

## testing threshold = median
predict_tuning <- as.data.frame(predict_tuning)
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -1.4856] <- TRUE

cm_tuning_median <- confusionMatrix(as.factor(predict_tuning$true),
                                    reference = as.factor(testing$Arrest))
f_byclass_tuning_median <- cm_tuning_median[["byClass"]][["F1"]]

## testing threshold = mean
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -1.104] <- TRUE

cm_tuning_mean <- confusionMatrix(as.factor(predict_tuning$true),
                                  reference = as.factor(testing$Arrest))
f_byclass_tuning_mean <- cm_tuning_mean[["byClass"]][["F1"]]

## testing threshold = proportional true/false on training set
data_prop <- count(training$Arrest)
data_prop2 <- count(tuning$Arrest)
quantile(predict_tuning$predict_tuning,
         data_prop$freq[data_prop$x == FALSE] / sum(data_prop$freq))
quantile(predict_tuning$predict_tuning,
         data_prop2$freq[data_prop2$x == FALSE] / sum(data_prop2$freq))

## testing threshold = -0.51 given by proprotion
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -0.51] <- TRUE

cm_tuning_51 <- confusionMatrix(as.factor(predict_tuning$true),
                                reference = as.factor(testing$Arrest))
f_byclass_tuning_51 <- cm_tuning_51[["byClass"]][["F1"]]

## testing threshold = -0.44 (3rd Qu) (approx. prop TRUE/FALSE in t
## raining$Arrest and tuning$Arrest
predict_tuning$true <- FALSE
predict_tuning$true[predict_tuning$predict_tuning >= -0.44] <- TRUE

cm_tuning_44 <- confusionMatrix(as.factor(predict_tuning$true),
                                reference = as.factor(testing$Arrest))
f_byclass_tuning_44 <- cm_tuning_44[["byClass"]][["F1"]]

#######################################################################
## Predicting on testing set using beta_final with threshold = -0.44 ##
#######################################################################

predict_final <- beta_final[1] + beta_final[2]*as.numeric(testing$IUCR) +
  beta_final[3]*as.numeric(testing$Domestic) +
  beta_final[4]*testing$Beat +
  beta_final[5]*testing$District +
  beta_final[6]*as.numeric(testing$`FBI Code`) +
  beta_final[7]*testing$year +
```

```r
  beta_final[8]*testing$month +
  beta_final[9]*testing$day

summary(predict_final)

## testing threshold = -0.44
predict_final <- as.data.frame(predict_final)
predict_final$true <- FALSE
predict_final$true[predict_final$predict_final >= -0.51] <- TRUE

cm_final<- confusionMatrix(as.factor(predict_final$true),
                           reference = as.factor(testing$Arrest))
f_byclass_final <- cm_final[["byClass"]][["F1"]]

## testing cvfit10 on testing
predict_min_lambda <- predict(cvfit10, newx =data.matrix(testing[,-2]),
                              s= 'lambda.min', type = "class")
cm2 <- confusionMatrix(as.factor(predict_min_lambda),
                       reference = as.factor(testing$Arrest))
f_byclass_ridge2 <- cm2[["byClass"]][["F1"]]

f_byclass_ridge2 ## micro f-measure


## testing cvfit10_lasso on testing
predict_min_lambda <- predict(cvfit10_lasso, newx =data.matrix(testing[,-2]),
                              s= 'lambda.min', type = "class")
cm_lasso2 <- confusionMatrix(as.factor(predict_min_lambda),
                             reference = as.factor(testing$Arrest))
f_byclass_lasso2 <- cm_lasso2[["byClass"]][["F1"]]

f_byclass_lasso2 ## micro f-measure

## testing cvfit on testing
predict_min_lambda <- predict(cvfit, newx =data.matrix(testing[,-2]),
                              s = 0.01, type = "class")
cm1 <- confusionMatrix(as.factor(predict_min_lambda),
                       reference = as.factor(testing$Arrest))
f_byclass_1 <- cm1[["byClass"]][["F1"]]

f_byclass_1 ## micro f-measure

## Overall Performance before and after splitting manually on testing set
rbind(f_byclass_final, f_byclass_lasso2, f_byclass_ridge2, f_byclass_1)
```