SENSORS DATA ANALYSIS

- Load data
  - Convert data into R readable format
  - Convert time into human readable format
- Clean data
  - Fill missing values
  - Select useful features
- Analyze data
  - Get a overview of the entire dataset
  - Separate different features and analyze individually
- Make prediction
  - Decision tree
  - Random forest
  - Gradient boosting tree
  - Logistic regression

LOAD DATA

- Use jsonlite package to convert the original file into R readable format i.e. a csv file.

```r
library(jsonlite)
library(plyr)

f <- readLines(con = "/Users/shanzhong/Desktop/sensorswww_data.txt", encoding = "UTF-8")
n <- length(f)
n # total number of lines
```

```
## [1] 75092
```

```r
data <- lapply(f[1:n-1], fromJSON) # the last line is incomplete
data <- lapply(data, as.data.frame)
df <- do.call(rbind.fill, data)

write.csv(df, "sensors_data.csv", row.names=FALSE, fileEncoding = "UTF-8")
df <- read.csv("sensors_data.csv", stringsAsFactors = FALSE)
dim(df)
```

```
## [1] 75091    70
```

- Convert time into human readable format and save it to new csv file

```
df.complete["time"] <- as.POSIXct(df.complete$time/1000, origin = "1970-01-01 00:00:00") # the raw time is in miliseconds
```

```
                time properties..first_visit_time
2017-03-06 09:04:10                        <NA>
2017-03-06 09:04:11                        <NA>
2017-03-06 09:04:16                        <NA>
2017-03-06 09:04:23                        <NA>
2017-03-06 09:04:58                        <NA>
2017-03-06 09:05:14                        <NA>
```

```
write.csv(df.complete, "sensors_userinfo.csv", row.names=FALSE)
```

CLEAN DATA

- Select the distinct id's appears in the dataset, delete the duplicated id's and then save the new dataset into another data frame

```r
# Select users whose first visit is recorded in this dataset
df.1sttime <- subset(df, properties..is_first_time == TRUE)
dim(df.1sttime) #9375  70
```

```r
id.1sttime<- df.1sttime$distinct_id

id.1sttime.unique <- unique(id.1sttime)
length(id.1sttime.unique) #9369
```

```r
# The same id shows up many times. Take a look at its entries
df.dup.id <- subset(df, distinct_id == "6be727db0adc4b0ea41431cc91c8a5e1481125ac")

# This user has duplicated "profile_set_once". Not useful information. Delete this user for simplicity
df <- subset(df, distinct_id != "6be727db0adc4b0ea41431cc91c8a5e1481125ac")

# dataframe of users with information since first visit
df.complete <- subset(df, distinct_id %in% id.1sttime.unique & !is.na(event))
```

ANALYSIS

- Overview of entire dataset

```r
#num of unique id's
length(unique(df.cleaned$distinct_id))
```

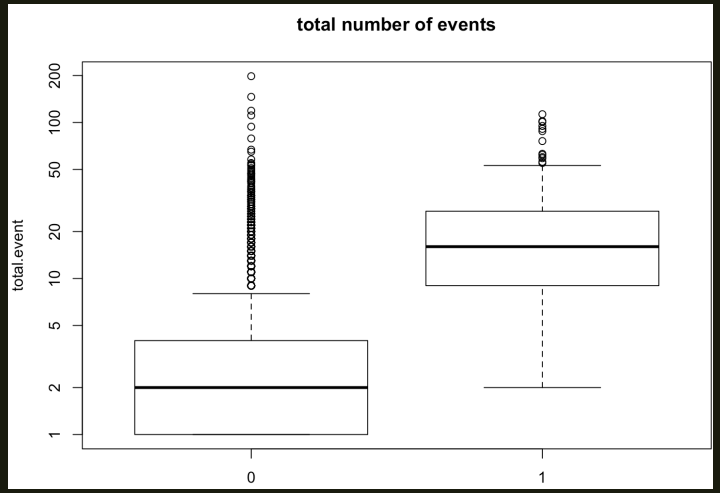
```
## [1] 9368
```

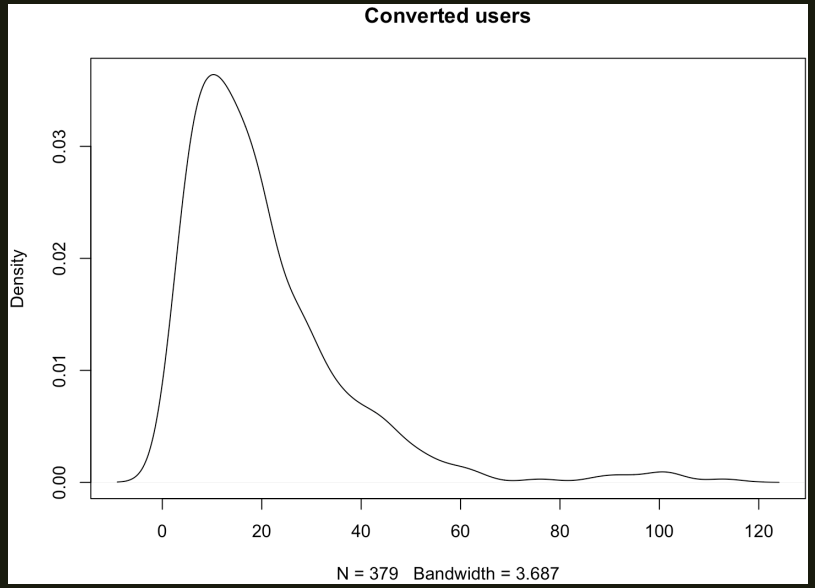
```r
#time span of data
range(df.cleaned$time) #8 days
```

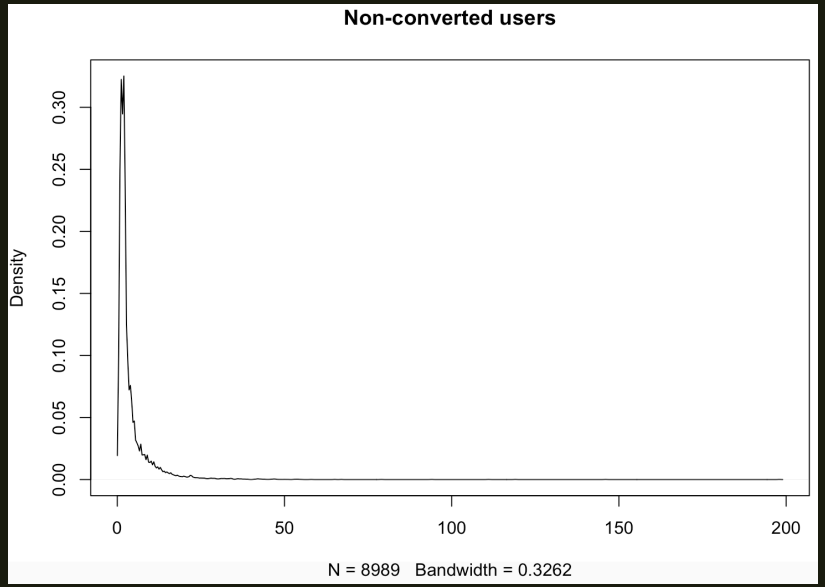
```
## [1] "2017-03-06 09:04:10" "2017-03-14 18:45:48"
```

```r
#unique events
events <- unique(df.cleaned$event) # 12 events
events #12 events
```

```
##  [1] "$pageview"              "btnClick"
##  [3] "click_send_cellphone"   "verify_cellphone_code"
##  [5] "index_leave"            "clickSubmit"
##  [7] "demo_leave"             "about_leave"
##  [9] "courses_leave"          "formSubmit"
## [11] "page_close"             "courses_play_leave"
```

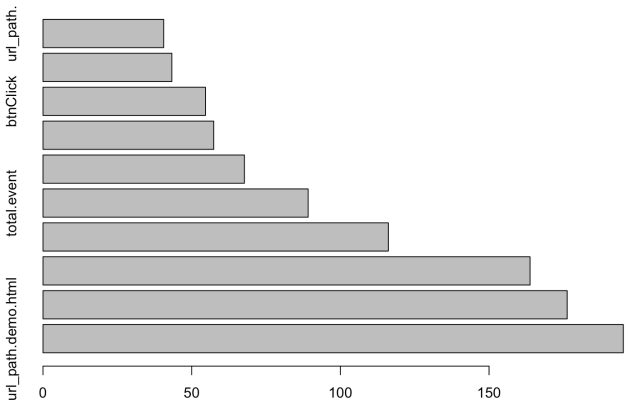


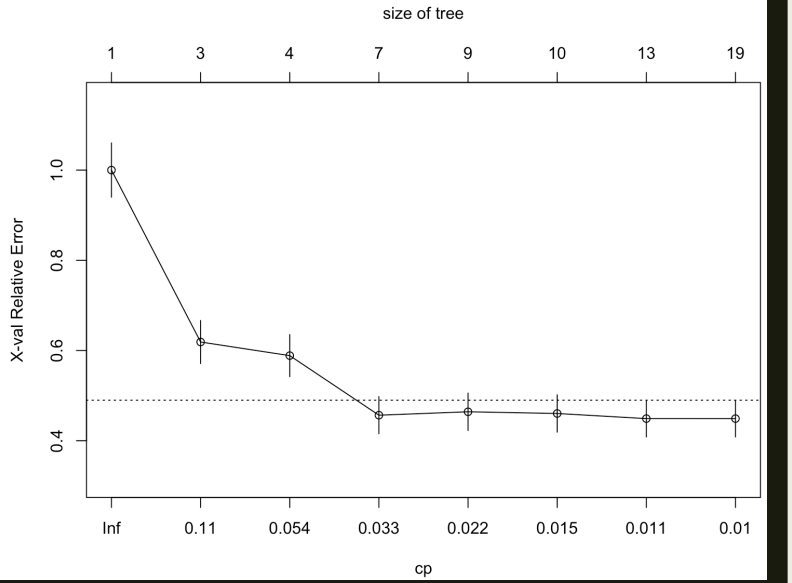

Converted users
Density
0.00
0.01
0.02
0.03
0
20
40
60
80
100
120
N = 379 Bandwidth = 3.687


Non-converted users
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
N = 8989 Bandwidth = 0.3262

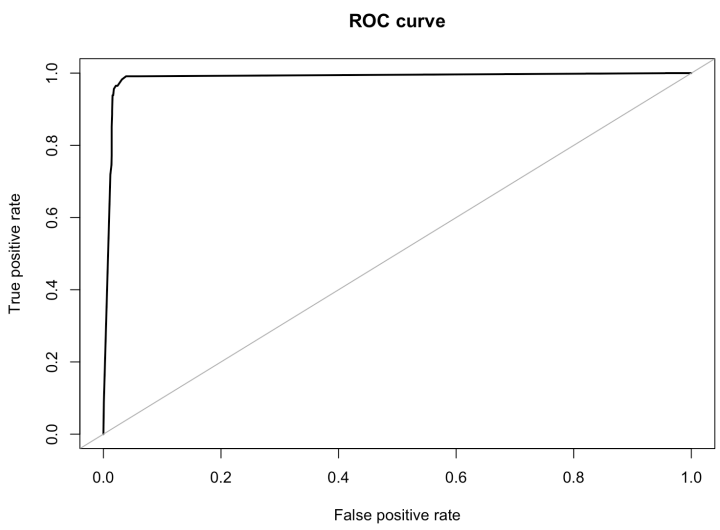
PREDICTION

- Predict user conversion
- Decision Tree





- ACC for decision tree



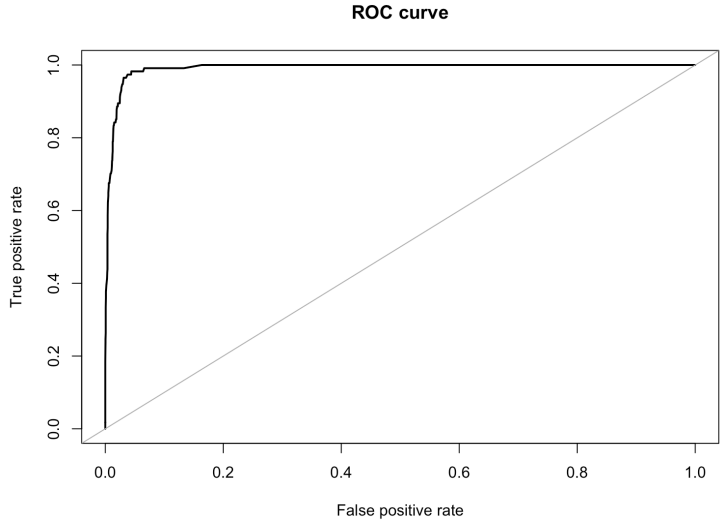

```
## Area under the curve (AUC): 0.987
```

- Random Forest

```
Call:
 randomForest(formula = as.factor(converted) ~ ., data = train,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 18

        OOB estimate of  error rate: 1.69%
Confusion matrix:
     0   1 class.error
0 6261  31 0.004926891
1   80 185 0.301886792
```

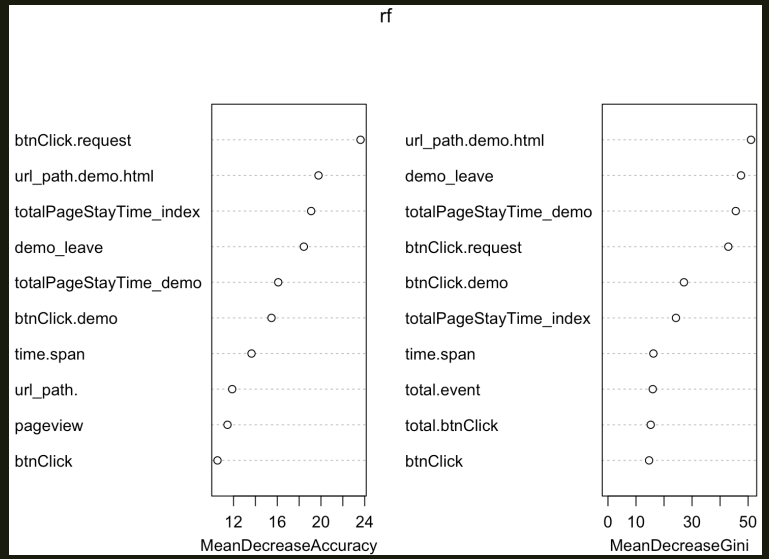
```
Call:
accuracy.meas(response = test$converted, predicted = predict(rf,
    test, type = "prob")[, 2])

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.748
recall: 0.702
F: 0.362
```

- ACC for random forest



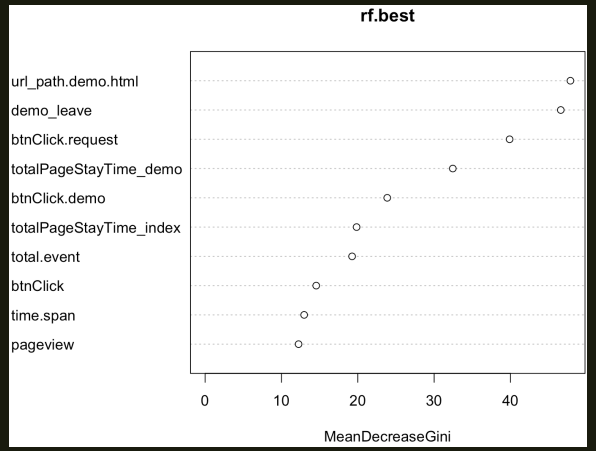

```
## Area under the curve (AUC): 0.991
```

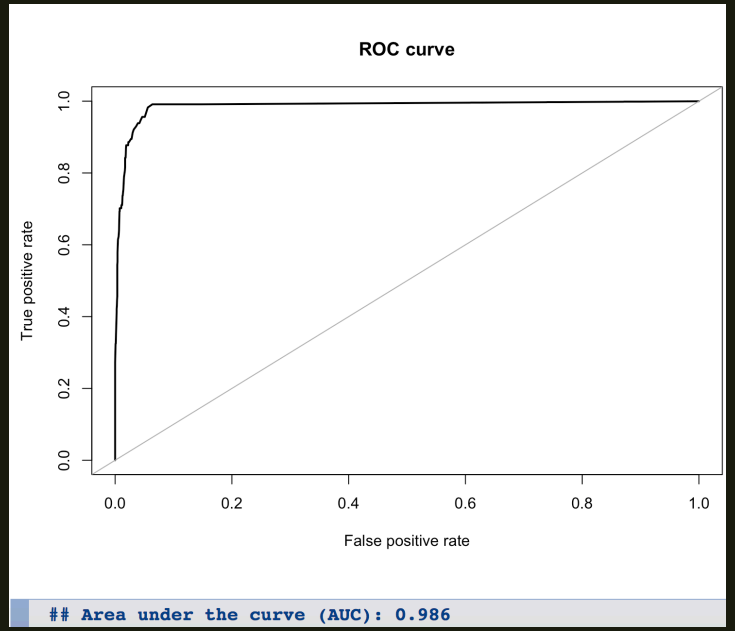
- Feature Importance



- ACC for random forest after tuning parameters

	ntree.model	nodesize.model	f.test
1	80	8	0.3594470
2	80	10	0.3720930
3	80	12	0.3644860
4	100	8	0.3594470
5	100	10	0.3720930
6	100	12	0.3615023
7	200	8	0.3665158
8	200	10	0.3669725
9	200	12	0.3686636





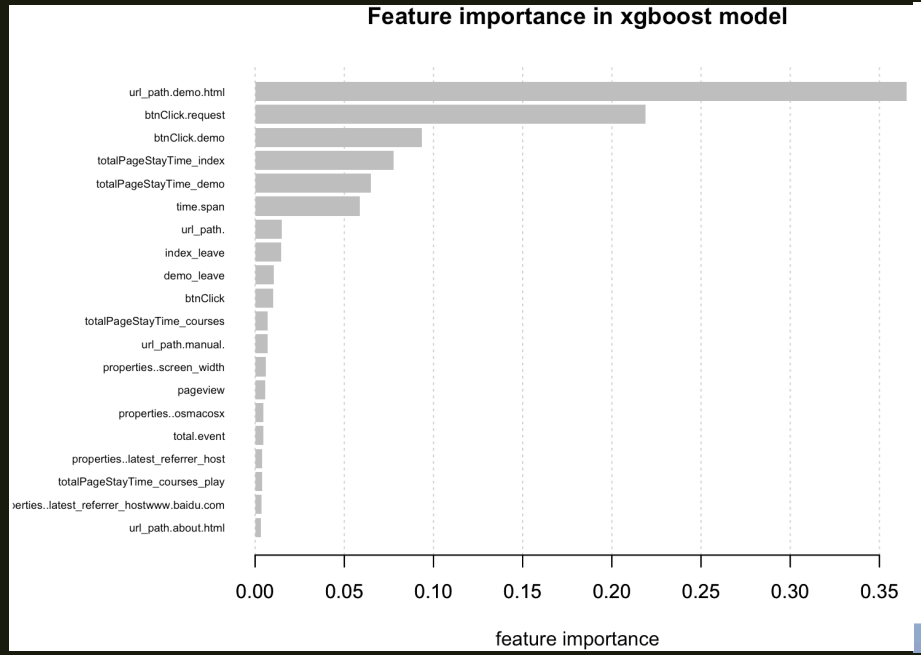

```
## Area under the curve (AUC): 0.986
```

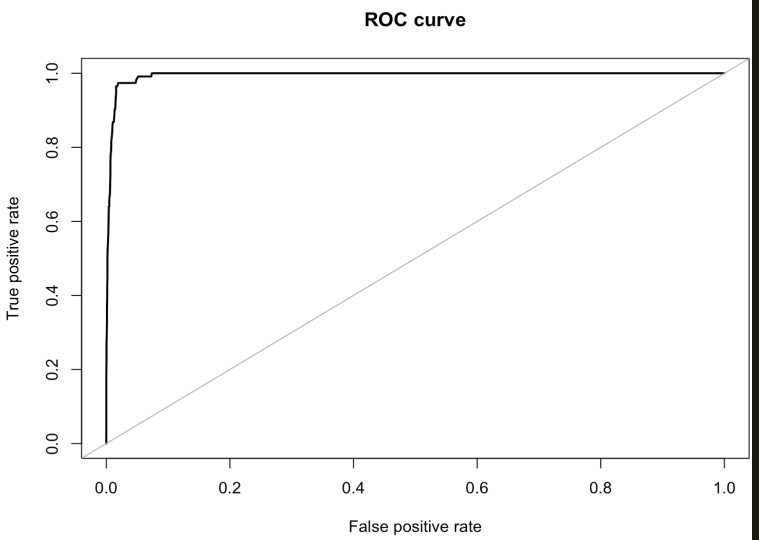
- Gradient Boosting Tree

```
Call:
accuracy.meas(response = test.y, predicted = predict(xgb.best,
    test.matrix))

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.773
recall: 0.868
F: 0.409
```

	max_depth.model	eta.model	test_auc
1	4	0.1	0.9553183
2	4	0.2	0.9553183
3	4	0.3	0.9553183
4	4	0.4	0.9553183
5	5	0.1	0.9560259
6	5	0.2	0.9560259
7	5	0.3	0.9560259
8	5	0.4	0.9560259
9	6	0.1	0.9560559
10	6	0.2	0.9560559
11	6	0.3	0.9560559
12	6	0.4	0.9560559
13	7	0.1	0.9561228
14	7	0.2	0.9561228
15	7	0.3	0.9561228
16	7	0.4	0.9561228

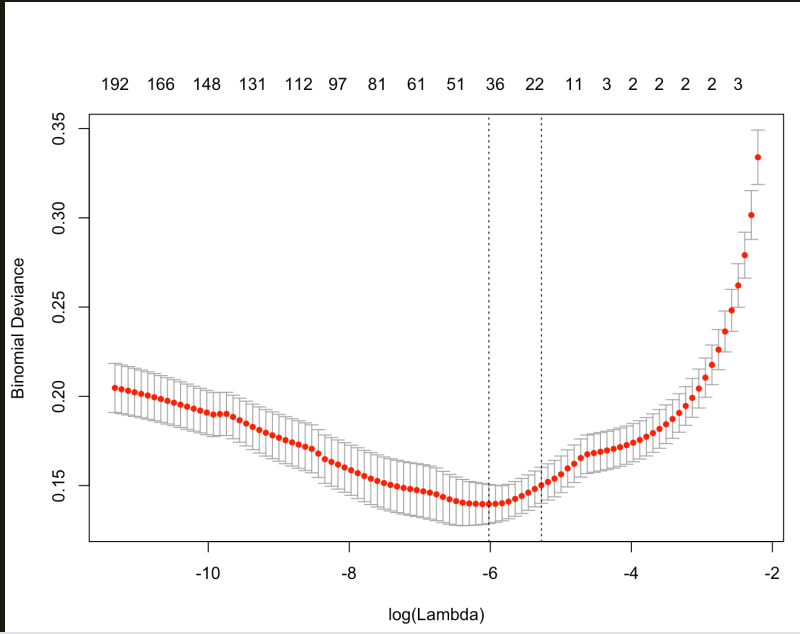
- ACC for xgboost



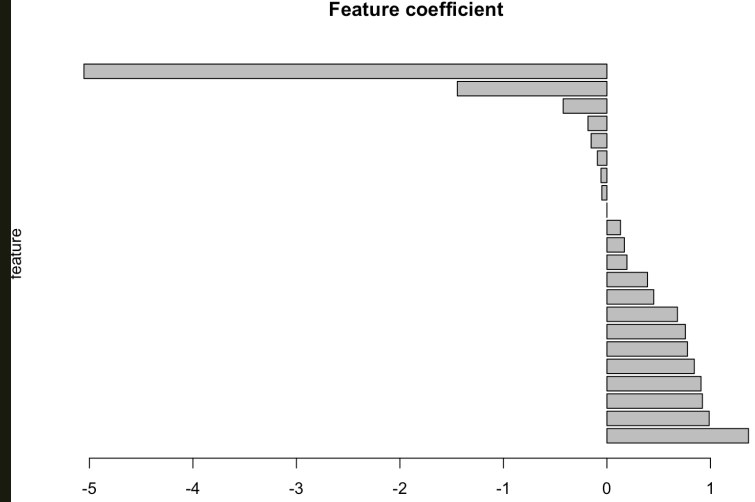


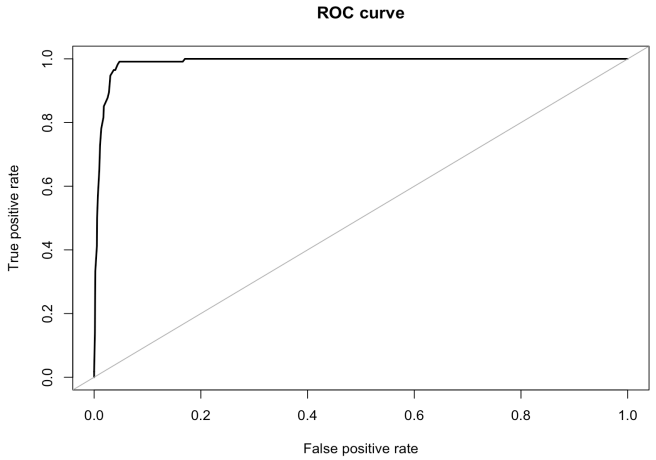

```
## Area under the curve (AUC): 0.995
```

- Logistic Regression




```
Call:
accuracy.meas(response = test.y, predicted = predict(logistic.cv,
    test.matrix, s = "lambda.1se"))

Examples are labelled as positive when predicted is greater than 0.5

precision: 0.774
recall: 0.421
F: 0.273
```






```
## Area under the curve (AUC): 0.989
```

The End