Project 2

Amy Nguyen

Machine Learning

Summer 2018

Regression

The data set that I am going to use for regression is the pollution data. My goal is to predict O3.AQI or ozone air quality index. The three algorithms I will be using for my regression models are the basic linear regression algorithm, kNN, and decision tree. SVM and Neural Network unfortunately take too long for me to be able to run.

Importing Data

Pollution: https://www.kaggle.com/sogun3/uspollution -The data originally had more columns and over 1 million objects. Since this is a bit too much data for my laptop to handle I cleaned the data and randomly sampled it earlier to reduce the data size. I then exported using "write.csv()" and now I'll be reading in the edited data.

```
#Reading in edited pollution data
pollutionDF <- read.csv("C:/Users/Amy Nguyen/Desktop/edited pollution.csv")

#cleaning
pollutionDF <- pollutionDF[-c(1)]
```

Exploring the data

Exploring the data for better understanding:

```
str(pollutionDF)

## 'data.frame':    157199 obs. of  14 variables:
##  $ State.Code       : int  6 6 6 6 80 22 6 6 25 15 ...
##  $ County.Code      : int  13 83 13 73 2 33 67 13 25 3 ...
##  $ Site.Num         : int  3001 2004 2 1010 3 9 6 1004 42 10 ...
##  $ NO2.Mean         : num  7.13 6.17 4.46 22.68 29.35 ...
##  $ NO2.1st.Max.Value: num  14 13 8.3 45 53 14 14 16 47 5.1 ...
##  $ NO2.1st.Max.Hour : int  20 3 22 20 20 4 19 19 17 2 ...
##  $ NO2.AQI          : int  13 12 8 42 50 13 13 15 44 5 ...
##  $ O3.AQI           : int  28 37 44 33 33 23 22 11 16 33 ...
##  $ SO2.Mean         : num  1.227 0.214 0.557 1.409 1 ...
##  $ SO2.1st.Max.Value: num  3 0.6 1.5 2 2 4 0.4 2 10 4.9 ...
##  $ SO2.1st.Max.Hour : int  13 5 14 6 8 21 2 22 23 23 ...
##  $ CO.Mean          : num  0.33 0.196 0.274 0.564 0.957 ...
##  $ CO.1st.Max.Value : num  0.4 0.2 0.3 1.2 2.3 0.4 0.623 0.8 0.7 0.5 ...
##  $ CO.1st.Max.Hour  : int  4 0 7 7 6 6 0 19 21 6 ...
```

```r
head(pollutionDF)
```

```
##   State.Code County.Code Site.Num  NO2.Mean NO2.1st.Max.Value
## 1          6          13     3001  7.130435              14.0
## 2          6          83     2004  6.173913              13.0
## 3          6          13        2  4.460870               8.3
## 4          6          73     1010 22.681818              45.0
## 5         80           2        3 29.347826              53.0
## 6         22          33        9  7.863636              14.0
##   NO2.1st.Max.Hour NO2.AQI O3.AQI SO2.Mean SO2.1st.Max.Value
## 1               20      13     28 1.227273               3.0
## 2                3      12     37 0.214286               0.6
## 3               22       8     44 0.557143               1.5
## 4               20      42     33 1.409091               2.0
## 5               20      50     33 1.000000               2.0
## 6                4      13     23 1.162500               4.0
##   SO2.1st.Max.Hour  CO.Mean CO.1st.Max.Value CO.1st.Max.Hour
## 1               13 0.330435              0.4               4
## 2                5 0.195833              0.2               0
## 3               14 0.273913              0.3               7
## 4                6 0.563636              1.2               7
## 5                8 0.956522              2.3               6
## 6               21 0.261667              0.4               6
```

```r
tail(pollutionDF)
```

```
##        State.Code County.Code Site.Num  NO2.Mean NO2.1st.Max.Value
## 157194          6          95        4  8.465217              20.2
## 157195          6          37     1103 14.260870              25.0
## 157196          6          67        6  4.521739               8.0
## 157197          6          13     1002 15.256522              26.5
## 157198         35           1       23 16.394118              34.2
## 157199          6          75        5 16.478261              28.0
##        NO2.1st.Max.Hour NO2.AQI O3.AQI SO2.Mean SO2.1st.Max.Value
## 157194                0      19     42 0.614286               1.1
## 157195                5      24     25 0.000000               0.0
## 157196                0       8     25 0.514286               1.3
## 157197               20      25     20 1.763636               2.8
## 157198                7      32     32 0.333333               0.7
## 157199               17      26     20 0.800000               1.3
##        SO2.1st.Max.Hour  CO.Mean CO.1st.Max.Value CO.1st.Max.Hour
## 157194                8 0.369565              0.4               0
## 157195                2 0.182609              0.4               5
## 157196                8 0.143478              0.2               5
## 157197                5 0.495833              0.7               2
## 157198               14 0.230435              0.5               7
## 157199                2 0.456522              0.8              17
```

Setting Train and Test

The same train and test sets will be used for each algorithm to ensure fair comparison

```
set.seed(1111)
i <- sample(1:nrow(pollutionDF), nrow(pollutionDF)*0.70, replace=FALSE)
train1 <- pollutionDF[i,]
test1 <- pollutionDF[-i,]
```

Linear Regression

Creating the Linear regression model

```
lm <- lm(O3.AQI~., data=train1)
summary(lm)

##
## Call:
## lm(formula = O3.AQI ~ ., data = train1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -113.681  -10.403   -3.332    5.031  187.461
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.443e+01  1.788e-01 192.560  < 2e-16 ***
## State.Code        -5.601e-03  3.638e-03  -1.540    0.124
## County.Code        7.261e-03  7.320e-04   9.919  < 2e-16 ***
## Site.Num           9.876e-04  2.904e-05  34.013  < 2e-16 ***
## NO2.Mean          -1.387e+00  1.513e-02 -91.647  < 2e-16 ***
## NO2.1st.Max.Value  1.750e-01  4.431e-02   3.949 7.85e-05 ***
## NO2.1st.Max.Hour  -2.084e-01  7.764e-03 -26.838  < 2e-16 ***
## NO2.AQI            7.929e-01  4.713e-02  16.824  < 2e-16 ***
## SO2.Mean           1.585e-01  3.920e-02   4.043 5.27e-05 ***
## SO2.1st.Max.Value  1.347e-01  1.364e-02   9.877  < 2e-16 ***
## SO2.1st.Max.Hour   4.424e-02  8.623e-03   5.130 2.90e-07 ***
## CO.Mean           -5.780e-01  3.938e-01  -1.468    0.142
## CO.1st.Max.Value  -4.454e+00  1.939e-01 -22.970  < 2e-16 ***
## CO.1st.Max.Hour   -1.717e-01  7.854e-03 -21.860  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.57 on 110025 degrees of freedom
## Multiple R-squared:  0.1345, Adjusted R-squared:  0.1344
## F-statistic:  1315 on 13 and 110025 DF,  p-value: < 2.2e-16
```

As you can see everything other than State.Code and CO.mean were good predictors as indicated by a low p-value.

Predict and evaluate in test data

```
#Predicting
predl1 <- predict(lm, newdata = test1)

#Correlation
corl1 <- cor(predl1, test1$O3.AQI)
print(paste("Linear Regression Correlation: ", corl1))

## [1] "Linear Regression Correlation:  0.368194260190988"

#Mean Squared Error
msel1 <- mean((predl1 - test1$O3.AQI)^2)
print(paste("Linear Regression mse: ", msel1))

## [1] "Linear Regression mse:  337.037357483883"

#Root Mean Squared Error
rmsel1 <- sqrt(msel1)
print(paste("Linear Regression rmse: ", rmsel1))

## [1] "Linear Regression rmse:  18.358577218398"
```

Unfortunately, our correlation is not too great as it's not near +/- 1. We'll use MSE and RMSE for comparison of our other models.

kNN Regression

Although kNN would be better scaled in order to keep consistency for each model, we will be leaving data unscaled.

Creating kNN model with best k and finding correlation and mse

```
library(caret)

## Warning: package 'caret' was built under R version 3.5.1

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.1

library(DMwR)

## Warning: package 'DMwR' was built under R version 3.5.1

## Loading required package: grid

ck <- rep(0,20)
msek <- rep(0,20)
rmsek <- rep(0,20)
```

```
i <- 1
for (k in seq(1, 50, 5)){
  fitk <- knnreg(train1[,1:7, 9:14],train1[,8], k=k)
  predk <- predict(fitk, test1[,1:7, 9:14])
  ck[i] <- cor(predk, test1$O3.AQI)
  msek[i] <- mean((predk - test1$O3.AQI)^2)
  rmsek[i] <- sqrt(msek[i])
  print(paste("k=", k, "cor=", ck[i], "mse=", msek[i], "rmse= ", rmsek[i]))
  i <- i + 1
}

## [1] "k= 1 cor= 0.465407662249587 mse= 417.006475925468 rmse=  20.420736419
7638"
## [1] "k= 6 cor= 0.508867083756186 mse= 298.8671634024 rmse=  17.28777496968
31"
## [1] "k= 11 cor= 0.520368314124304 mse= 287.248666062029 rmse=  16.94841190
38342"
## [1] "k= 16 cor= 0.522505585565726 mse= 284.673588904668 rmse=  16.87227278
42063"
## [1] "k= 21 cor= 0.523742169684185 mse= 283.546834294147 rmse=  16.83884895
98947"
## [1] "k= 26 cor= 0.518833979249868 mse= 285.286083328504 rmse=  16.89041394
78138"
## [1] "k= 31 cor= 0.516554362122603 mse= 286.075810058298 rmse=  16.91377574
81379"
## [1] "k= 36 cor= 0.51329799226757 mse= 287.351447160732 rmse=  16.951443807
5561"
## [1] "k= 41 cor= 0.509974535916536 mse= 288.668170550472 rmse=  16.99023750
71825"
## [1] "k= 46 cor= 0.507225787655282 mse= 289.776346397925 rmse=  17.02281840
3482"
```

From our list, that has finally loaded in after some time, the best k value is k=21. It has the highest correlation value and lowest mse and rmse value.

Thus, the metrics for our unscaled kNN regression model are: Correlation = 0.5237 mse = 283.5468 rmse = 16.8388

Comparing to the Linear regression model we can see that our unscaled kNN model performs much better. Our correlation is much higher and our mse and rmse values are lower. A possible reason why linear regression many have performed so poorly is because it may be too simplified for my scenario.

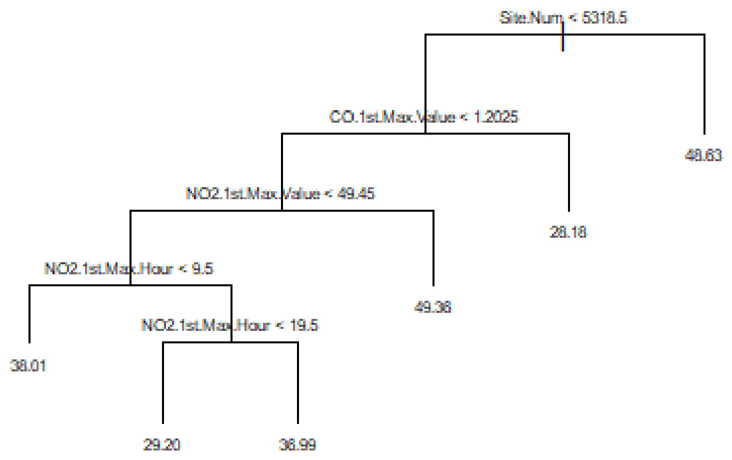
Decision Tree

Creating the tree

```
library(tree)

## Warning: package 'tree' was built under R version 3.5.1

treel1<- tree(O3.AQI~., data=train1)
summary(treel1)

##
## Regression tree:
## tree(formula = O3.AQI ~ ., data = train1)
## Variables actually used in tree construction:
## [1] "Site.Num"          "CO.1st.Max.Value"  "NO2.1st.Max.Value"
## [4] "NO2.1st.Max.Hour"
## Number of terminal nodes:  6
## Residual mean deviance:  370 = 40710000 / 110000
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -49.360 -10.990  -3.012   0.000   5.818 181.000
```

Plotting

```
plot(treel1)
text(treel1, cex=0.5, pretty=0)
```



Evaluation the tree

```
#Predicting
predl2 <- predict(treel1, newdata=test1)

#Correlation
corl2 <- cor(predl2, test1$O3.AQI)
print(paste("Decision Tree Correlation: ", corl1))

## [1] "Decision Tree Correlation:  0.368194260190988"

#Mean Squared Error
msel2 <- mean((predl2 - test1$O3.AQI)^2)
print(paste("Decision Tree mse: ", msel2))

## [1] "Decision Tree mse:  363.552074247546"

#Root Mean Squared Error
rmsel2 <- sqrt(msel2)
print(paste("Decision Tree rmse: ", rmsel2))

## [1] "Decision Tree rmse:  19.0670415704048"
```

As you can see out correlation is still not the best and in comparison, to the other algorithms it performed the worse. Linear regression's correlation is slightly higher, and it's mse and rmse values are slightly lower. That being said decision trees have a reputation of having low accuracy so I'm not surprised to see that it also has low correlation.

Evaluating the algorithms

After going through the three algorithms I found that kNN worked the best for predicting Ozone Air Quality Index (O3.AQI). That being said although it performed the best the correlation of 0.5237 was still not that high. In order to improve results, I would scale the data for the kNN regression and test even more k values. I would also try random forest and bagging for the decision tree to see if that would yield better results.

Classification

The data I'm going to be using for classification is this fraud data. I hope to detect whether or not the activity was or was not fraudulent based on the other variables. For classification the 3 algorithms that I will be using are logistic regression, Naive Bayes, and decision trees once again. Unfortunately, my laptop just cannot handle SVM or Neural Networks and crashes RStudio.

Importing the data

Fraud: https://www.kaggle.com/ntnu-testimon/banksim1#bsNET140513_032310.csv - Just like the previous data I had to clean and sample the data to a smaller size in order to be able to use the data. The algorithms took too long or crashed RStudio previously.

```
#Reading in fraud data
fraudDF <- read.csv("C:/Users/Amy Nguyen/Desktop/edited fraud.csv")

#Cleaning
fraudDF <- fraudDF[-c(1:3,5:8)]
fraudDF <- na.omit(fraudDF)

fraudDF$fraud <- as.factor(fraudDF$fraud)
```

Exploring the data

In order to get a better idea of our dataset let's explore it.

```
#Exploring fraudDF
str(fraudDF)

## 'data.frame':    178392 obs. of  4 variables:
##  $ age     : Factor w/ 8 levels "'0'","'1'","'2'",..: 3 4 3 3 5 3 3 4 5 2
...
##  $ category: Factor w/ 15 levels "'es_barsandrestaurants'",..: 13 13 13 11
13 13 13 6 13 13 ...
##  $ amount  : num  35.7 23.3 53 151.8 6.6 ...
##  $ fraud   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

head(fraudDF)

##   age              category amount fraud
## 1 '2' 'es_transportation'  35.67     0
## 2 '3' 'es_transportation'  23.33     0
## 3 '2' 'es_transportation'  53.05     0
## 4 '2'   'es_sportsandtoys' 151.85     0
## 5 '4' 'es_transportation'   6.60     0
## 6 '2' 'es_transportation'  11.84     0

tail(fraudDF)

##        age              category amount fraud
## 178387 '4' 'es_transportation'  21.23     0
## 178388 '2' 'es_transportation'  13.09     0
## 178389 '3' 'es_transportation'   6.73     0
## 178390 '3' 'es_transportation'   0.61     0
## 178391 '4' 'es_transportation'  22.01     0
## 178392 '3' 'es_transportation'  45.72     0
```

Setting Train and Test

The same train and test sets will be used for each algorithm to ensure fair comparison

```
set.seed(2222)
i <- sample(1:nrow(fraudDF), nrow(fraudDF)*0.70, replace=FALSE)
train2 <- fraudDF[i,]
test2 <- fraudDF[-i,]
```

Logistic Regression

Build logistic model

```
glm1 <- glm(fraud~., data= train2, family = "binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(glm1)

##
## Call:
## glm(formula = fraud ~ ., family = "binomial", data = train2)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.774   0.000   0.000   0.000   4.388
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -4.706e+00  5.537e-01  -8.499  < 2e-16 ***
## age'1'                         -4.235e-01  5.182e-01  -0.817  0.41373
## age'2'                         -2.703e-01  5.054e-01  -0.535  0.59281
## age'3'                         -5.463e-01  5.079e-01  -1.076  0.28211
## age'4'                         -3.416e-01  5.089e-01  -0.671  0.50198
## age'5'                         -4.845e-01  5.184e-01  -0.935  0.35003
## age'6'                         -5.531e-01  5.397e-01  -1.025  0.30543
## age'U'                         -6.872e+00  1.586e+01  -0.433  0.66482
## category'es_contents'          -1.923e+01  5.634e+03  -0.003  0.99728
## category'es_fashion'           -1.773e-01  3.168e-01  -0.560  0.57562
## category'es_food'              -1.906e+01  1.065e+03  -0.018  0.98573
## category'es_health'             2.624e-01  2.541e-01   1.033  0.30167
## category'es_home'               2.336e-01  3.162e-01   0.739  0.45996
## category'es_hotelservices'      1.448e+00  2.929e-01   4.942 7.73e-07 ***
## category'es_hyper'              1.243e+00  2.719e-01   4.569 4.90e-06 ***
## category'es_leisure'            4.506e+00  4.729e-01   9.529  < 2e-16 ***
## category'es_otherservices'      1.839e+00  3.277e-01   5.611 2.01e-08 ***
## category'es_sportsandtoys'      2.623e+00  2.561e-01  10.240  < 2e-16 ***
## category'es_tech'              -3.662e-01  3.479e-01  -1.053  0.29249
## category'es_transportation'    -1.888e+01  2.430e+02  -0.078  0.93808
## category'es_travel'            -4.779e+00  6.627e-01  -7.211 5.55e-13 ***
## category'es_wellnessandbeauty'  7.591e-01  2.554e-01   2.972  0.00296 **
## amount                          1.445e-02  4.166e-04  34.673  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17192.2  on 124873  degrees of freedom
## Residual deviance:  4646.9  on 124851  degrees of freedom
## AIC: 4692.9
```

```
##
## Number of Fisher Scoring iterations: 22
```

So, from our data we can see a few interesting things. Our best predictors are the categories; hotel services, hyper, leisure, other service, sports and toys, travel, and the amount of money in the transaction. That being said, our null deviance is not too bad considering how much data we're dealing with but our residual deviance is much smaller so we might have a good model on our hands.

Evaluate on test data

```
library(caret)

probc1 <- predict(glm1, newdata= test2, type="response")
predc1 <- ifelse(probc1>0.5, 1, 0)

accc1 <- mean(predc1==test2$fraud)
print(paste("accuracy = ", accc1))

## [1] "accuracy =  0.994712059494002"

table(predc1, test2$fraud)

##
## predc1     0     1
##      0 52814   249
##      1    34   421
```

As you can see we have a very high accuracy.

Naive Bayes

Created model and predict

```
library(e1071)

nb1 <- naiveBayes(fraud~., data=train2)
nb1

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          0          1
## 0.98712302 0.01287698
##
```

```
## Conditional probabilities:
##    age
## Y             '0'          '1'          '2'          '3'          '4'
##   0 0.0041941817 0.0982347119 0.3149043532 0.2473431441 0.1835542648
##   1 0.0074626866 0.0939054726 0.3351990050 0.2375621891 0.1946517413
##    age
## Y             '5'          '6'          'U'
##   0 0.1053331819 0.0444567034 0.0019794591
##   1 0.0920398010 0.0385572139 0.0006218905
##
##    category
## Y   'es_barsandrestaurants' 'es_contents' 'es_fashion'    'es_food'
##   0            0.0108951373  0.0015576071 0.0105462982 0.0443431279
##   1            0.0130597015  0.0000000000 0.0161691542 0.0000000000
##    category
## Y    'es_health'    'es_home' 'es_hotelservices'   'es_hyper' 'es_leisure'
##   0 0.0251326400 0.0027420375       0.0021417098 0.0102948096 0.0000649003
##   1 0.2356965174 0.0398009950       0.0771144279 0.0435323383 0.0652985075
##    category
## Y   'es_otherservices' 'es_sportsandtoys'    'es_tech' 'es_transportation'
##   0       0.0012249931       0.0036425292 0.0038453426        0.8595151948
##   1       0.0304726368       0.2804726368 0.0192786070        0.0000000000
##    category
## Y    'es_travel' 'es_wellnessandbeauty'
##   0 0.0002677137           0.0237859588
##   1 0.0740049751           0.1050995025
##
##    amount
## Y        [,1]      [,2]
##   0  31.82246  31.05934
##   1 522.74993 793.42871
```

Evaluating on test

```
predc2 <- predict(nb1, test2$fraud)

accc2 <- mean(predc2==test2$fraud)
print(paste("accuracy = ", accc2))

## [1] "accuracy =  0.987480847565305"

table(predc2, test2$fraud)

##
## predc2     0     1
##      0 52848   670
##      1     0     0
```

As you can see we also got a large accuracy. However, our accuracy is lower than the accuracy of the logistic model. This may be because we are dealing with a larger dataset and Naive Bayes is best for smaller datasets.

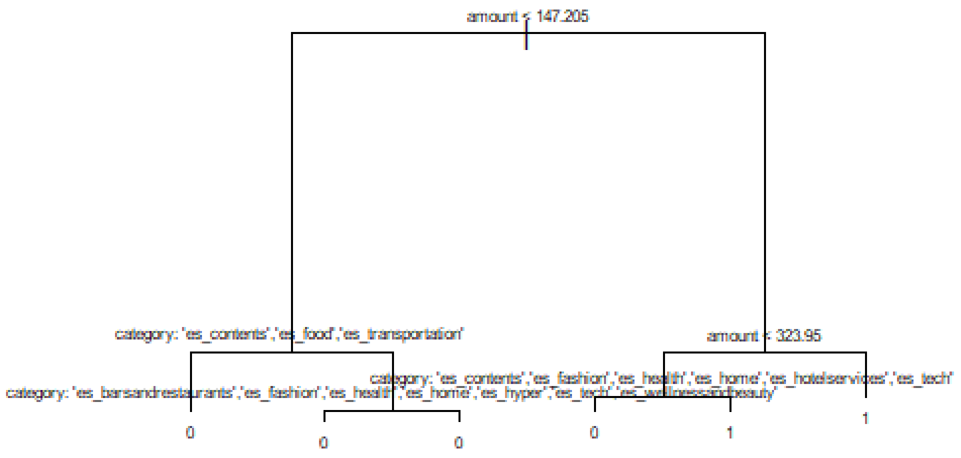
Decision Tree

Creating the Tree

```
library(tree)

ctree <- tree(fraud~., data= train2)
summary(ctree)

##
## Classification tree:
## tree(formula = fraud ~ ., data = train2)
## Variables actually used in tree construction:
## [1] "amount"   "category"
## Number of terminal nodes:  6
## Residual mean deviance:  0.03771 = 4709 / 124900
## Misclassification error rate: 0.005974 = 746 / 124874
```

Graphing

```
plot(ctree)
text(ctree, cex=0.5, pretty=0)
```


amount < 147.205
category: 'es_contents','es_food','es_transportation'
amount < 323.95
category: 'es_contents','es_fashion','es_health','es_home','es_hotelservices','es_tech'
category: 'es_barsandrestaurants','es_fashion','es_health','es_home','es_hyper','es_tech','es_wellnessandbeauty'
0
0
0
0
1
1

Evaluating

```
predc3 <- predict(ctree, newdata = test2, type="class")

accc3 <- mean(predc3==test2$fraud)
print(paste("accuracy = ", accc3))

## [1] "accuracy =  0.994300982846893"

table(predc3, test2$fraud)

##
## predc3     0     1
##      0 52749   206
##      1    99   464
```

The decision tree also performed very well. However, it did not perform as well as the logistic regression. Once again this is most likely due to its reputation for low accuracy. That being said, with bagging, random forest, or even pruning the decision tree may yield better results.

Evaluating the Algorithms

After going through the three algorithms, the logistic regression came out as the winner with the best accuracy. As stated before this might be because Naive Bayes is better suited for smaller datasets and Decision Tree are known to be inaccurate. Furthermore, we did not apply pruning, bagging, or random forest.