# Credit Card Fraud Detection Using Machine Learning

UCSB PSTAT131 Final Project

Sunrise Gao

## Contents



# Introduction

With the development of technology, online shopping has been unprecedentedly easy and convenient by many applications and browsers that can auto-fill your personal information and save it online. People no longer need to re-type every information that is needed to shop online. Only a few click to get the job done. However, this could rise a major problem that people's personal information such as credit card, home address get stolen, and is used unauthorizedly and illegally. This is called credit card frauds. Many shops and banks have developed a lot of way to protect customers from credit card fraud such as adding extra authorization step like Two-Factor Authentication(getting verify code from phone or email). But, there is

always new way fraudsters get your information: Lost or stolen credit cards. Skimming your credit card, such as at a gas station pump. Therefore, early credit card fraud detection is essentially important.

```
knitr::include_graphics("credit_cards.jpg")
```

### What is Credit Card Fraud Detection?

Credit card fraud detection is the collective term for the policies, tools, methodologies, and practices that credit card companies and financial institutions take to combat identity fraud and stop fraudulent transactions. As today's well-developed technology, there is so many new ways that people's information could get leaked or stolen. Analyzing and improving a newer and better model and algorithm for credit card fraud detection frequently is very important.

### About the Dataset

The dataset I will be using is from Kaggle.com called "Credit Card Fraud Detection". "The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions." (please see reference for source).

Now Let's get started. I will be doing exploratory data analysis first.

## Exploratory Data Analysis

### Loading Packages and Data

Check rmd file for full packages list.

#### Loading Data

```r
# read data
mydata <- read.csv("creditcard.csv")
head(mydata)
```

```
##   Time         V1          V2        V3         V4          V5          V6
## 1    0 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778
## 2    0  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081
## 3    1 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938
## 4    1 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317
## 5    2 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146
## 6    2 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755
##           V7          V8         V9         V10        V11         V12
## 1  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086
## 2 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531
## 3  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369
## 4  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823
## 5  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555
## 6  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384
##          V13        V14        V15        V16         V17         V18
## 1 -0.9913898 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058
## 2  0.4890950 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127
## 3  0.7172927 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931
## 4  0.5077569 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500
## 5  1.3458516 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479
## 6 -0.3580907 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315
##           V19         V20          V21          V22         V23         V24
## 1  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807
## 2 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648
## 3 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096
## 4 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533
## 5  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698
## 6 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658
##          V25        V26          V27         V28 Amount Class
## 1  0.1285394 -0.1891148  0.133558377 -0.02105305 149.62     0
## 2  0.1671704  0.1258945 -0.008983099  0.01472417   2.69     0
## 3 -0.3276418 -0.1390966 -0.055352794 -0.05975184 378.66     0
## 4  0.6473760 -0.2219288  0.062722849  0.06145763 123.50     0
## 5 -0.2060096  0.5022922  0.219422230  0.21515315  69.99     0
## 6 -0.2327938  0.1059148  0.253844225  0.08108026   3.67     0
```

### Checking Dimension

```r
# check original data dimension
dimension <- dim(mydata)
dimension
```

```
## [1] 284807     31
```

The data set has 284807 observations and 31 columns. There is 1 column for Time, 28 columns for PCA transformation, 1 column for purchase amount, and 1 column for class. Please check codebook for a detailed explanation for every variable.

### Checking Missing Value

```
# check missing value
sum(is.na(mydata))
```

```
## [1] 0
```

There is no missing value. We are good to go.

## Analyzing the Data

```
# genuine and fraudulent count
mydata %>%
  count(Class)
```

```
##   Class      n
## 1     0 284315
## 2     1    492
```

Here, we see that class = "0" is labeled as the transaction is genuine and class = "1" is labeled as fraudulent. So, there are 284315 valid transactions and 492 fraudulent cases. As we can see only about 0.17% fraudulent transaction out all the transactions. The data is highly unbalanced. so I will first apply models without balancing it, and if we don't get a good accuracy, we might have to find a way to balance this dataset.

```
# comparing summary of two transcation cases.
genuine_case <- mydata %>%
  filter(Class == 0)

describe(genuine_case$Amount)
```

```
##    vars      n  mean     sd median trimmed   mad min      max    range skew
## X1    1 284315 88.29 250.11     22   41.62 29.98   0 25691.16 25691.16   17
##    kurtosis   se
## X1   846.72 0.47
```

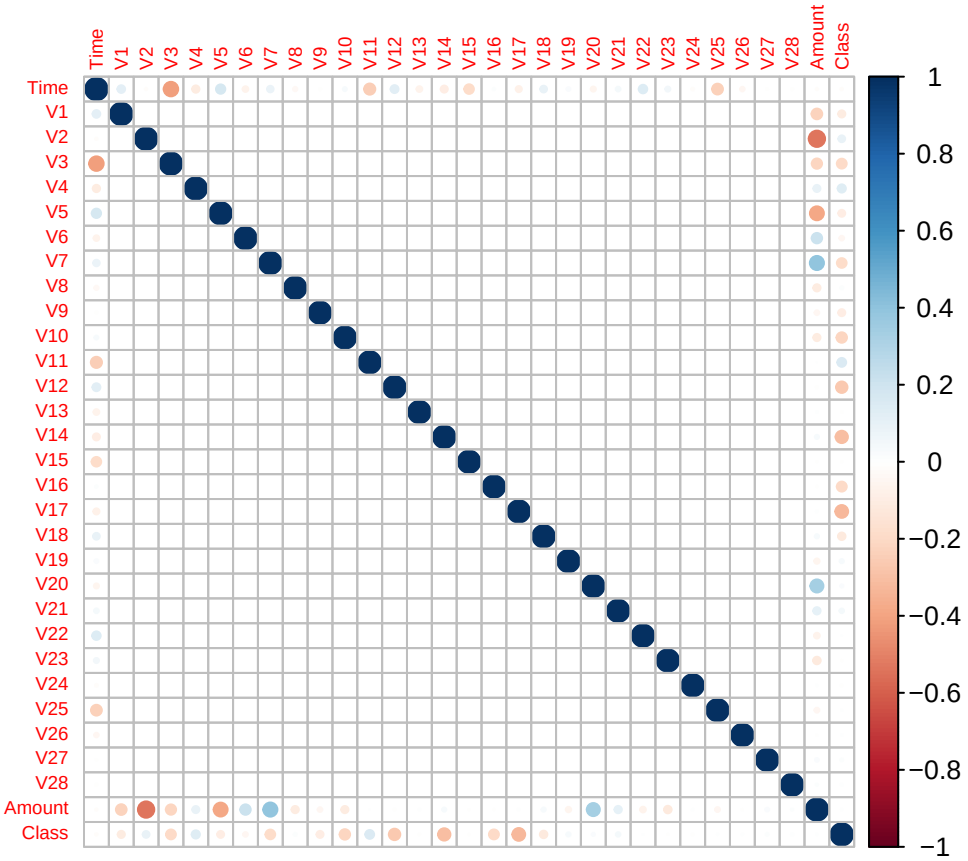
```
fraud_case <- mydata %>%
  filter(Class == 1)

describe(fraud_case$Amount)
```

```
##    vars   n   mean     sd median trimmed   mad min     max   range skew
## X1    1 492 122.21 256.68   9.25   58.83 13.71   0 2125.87 2125.87 3.73
##    kurtosis    se
## X1    17.47 11.57
```

By comparing summary of genuine cases and fraud cases, we can see that the mean of fraud case is higher than genuine cases which means the average money transaction for the fraudulent cases is more. This makes sense as credit card fraud is intended to steal as much money as it could as once. So, higher amount transactions can be a factor to detect fraudulent case.

### Visualizing the Data

```
# correlation matrix
correlation <- cor(mydata)
corrplot(correlation, tl.cex = 0.6, number.cex = 0.5, method = "circle", type = "full")
```




In the correlation matrix we can see that most of the values do not correlate to each other but there are some values that either has a positive or a negative correlation with each other. For example, V2 and V5 are highly negatively correlated with Amount. We may consider if the values reflect a importance to the target component but this definitely gives us a deeper understanding of the data.

I will be doing data splitting in next step.

## Data Splitting

### Convert class to factor

```
# convert class to factor
mydata <- mydata %>%
  mutate(Class = factor(Class, levels = c("1", "0"))) %>%
  select(-Time, )
```

**Initial Data Splitting**

```
# splitting
set.seed(2022)
cc_split <- initial_split(mydata, prop = 0.80, strain = Class)
cc_train <- training(cc_split)
cc_test <- testing(cc_split)
```

**Checking Dimension for Splitting Data**

```
# check dimension for training and testing
dim(cc_train)
```

```
## [1] 227845     30
```

```
dim(cc_test)
```

```
## [1] 56962    30
```

Here, we will have 227845 values for training set and 56962 values for testing set.

# Model Fitting

In this step, I will be fitting boosted tree, decision tree, logistic regression, and in total of 3 models.

**Creating Recipe & K-Fold**

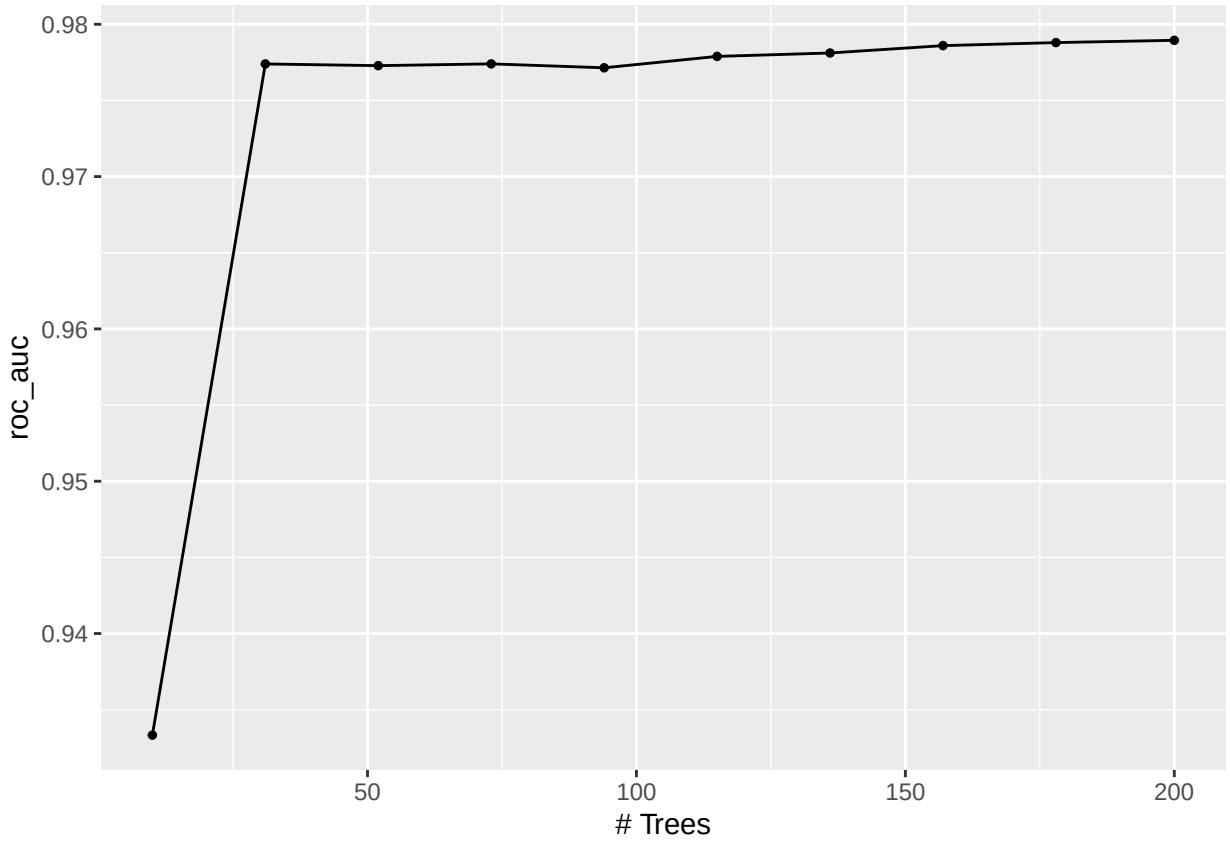
```
# create recipe and remove zero variance
cc_recipe <- recipe(Class ~ ., cc_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

# create k-fold
cc_folds <- vfold_cv(cc_train, v = 5, strain = Class)
```

## Model 1: Boosted Tree

First, I will do boosted tree model.

**Setting up**

```r
# setup
bt_model <- boost_tree(trees = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# define grid
bt_grid <- grid_regular(trees(c(10,200)),levels = 10)

# workflow
bt_workflow <- workflow() %>%
  add_model(bt_model) %>%
  add_recipe(cc_recipe)

bt_res <- tune_grid(
  bt_workflow,
  resamples = cc_folds,
  grid = bt_grid,
  metrics = metric_set(roc_auc),)

autoplot(bt_res)
```




The roc_auc keeps increasing until reaches the peak around 0.978 with around 30 tress.

### Fitting the Model

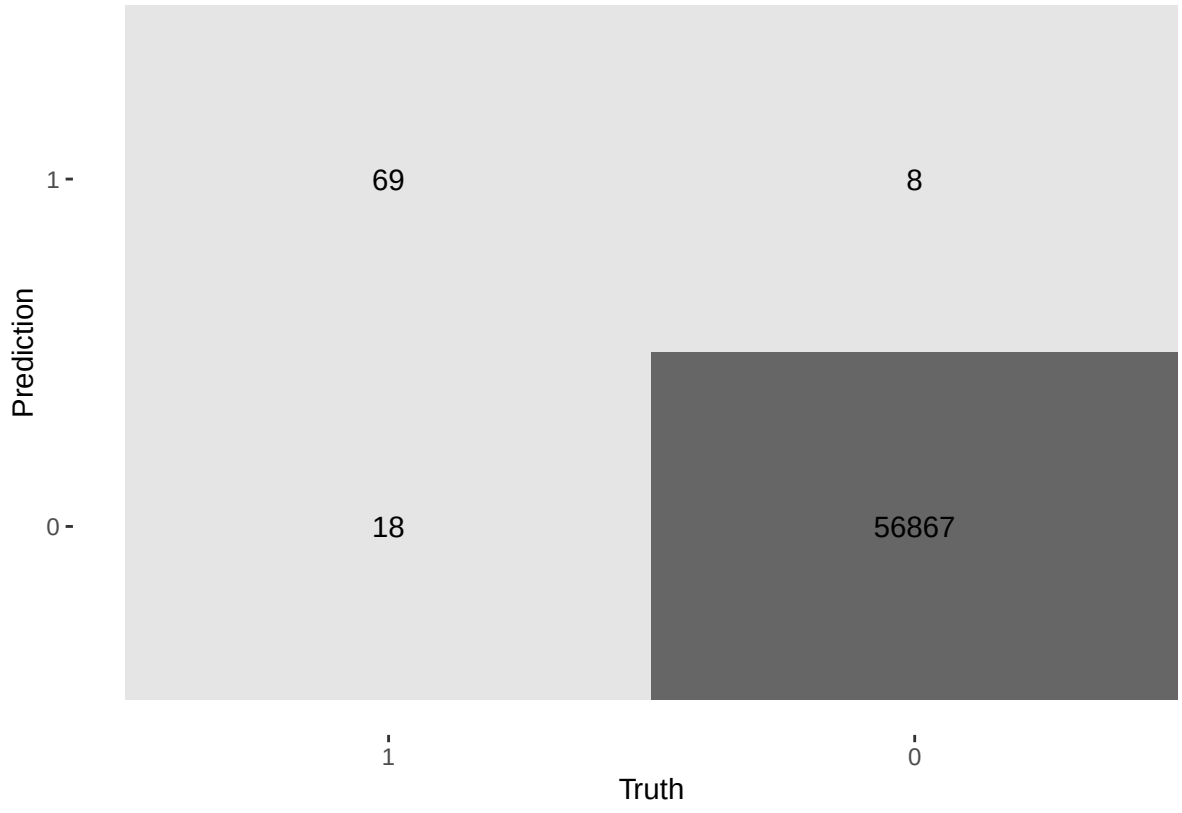
```
# select best tree and fit
best_tree <- select_best(bt_res)
bt_final <- finalize_workflow(bt_workflow, best_tree)
bt_final_fit <- fit(bt_final, data = cc_train)

augment(bt_final_fit, new_data = cc_test)%>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          1.00
```

From the result above, the best boosted tree has 0.9995436 accuracy that almost equal to 1. (Notice: the exact result might by rounded when knitting, please see code for the exact value if needed.)

### Heat Map

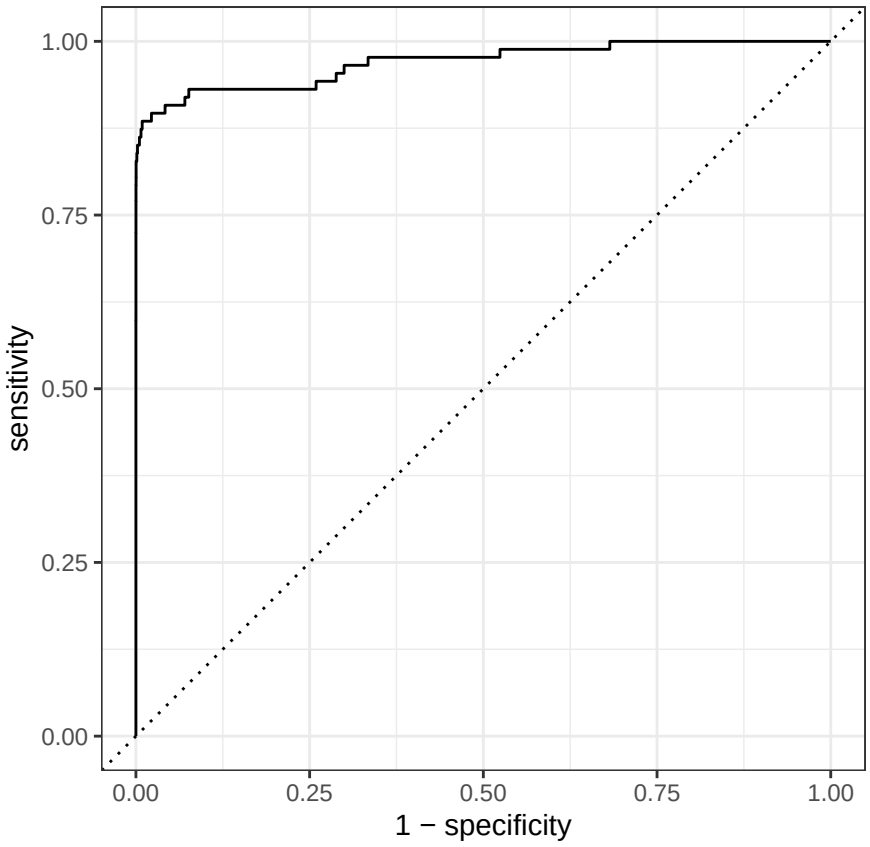
```
# heat map
augment(bt_final_fit, new_data = cc_test) %>%
  conf_mat(truth = Class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



The best boosted tree model has successful predicted 69 of 87 observations with 0.9995435 accuracy.

**ROC & AUC**

```
# ROC
augment(bt_final_fit, new_data = cc_test)%>%
  roc_curve(Class, .pred_1) %>%
  autoplot()
```




```
# AUC
augment(bt_final_fit, new_data = cc_test) %>%
  roc_auc(Class, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.970
```

Boosted tree model looks great and has very high accuracy as it is the first model we fit. Let's see if other models can do best!

## Model 2: Logistic Regression

For the third model, I will be fitting logistic regression model.

### Setting Up

```
# setup
log_model <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

# workflow
log_workflow <- workflow() %>%
  add_model(log_model) %>%
  add_recipe(cc_recipe)
```

### Fitting the model

```
# fit the training data
log_fit_train <- fit(log_workflow, cc_train)

# predict
predict(log_fit_train, new_data = cc_train, type = "class") %>%
  bind_cols(cc_train %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.999
```

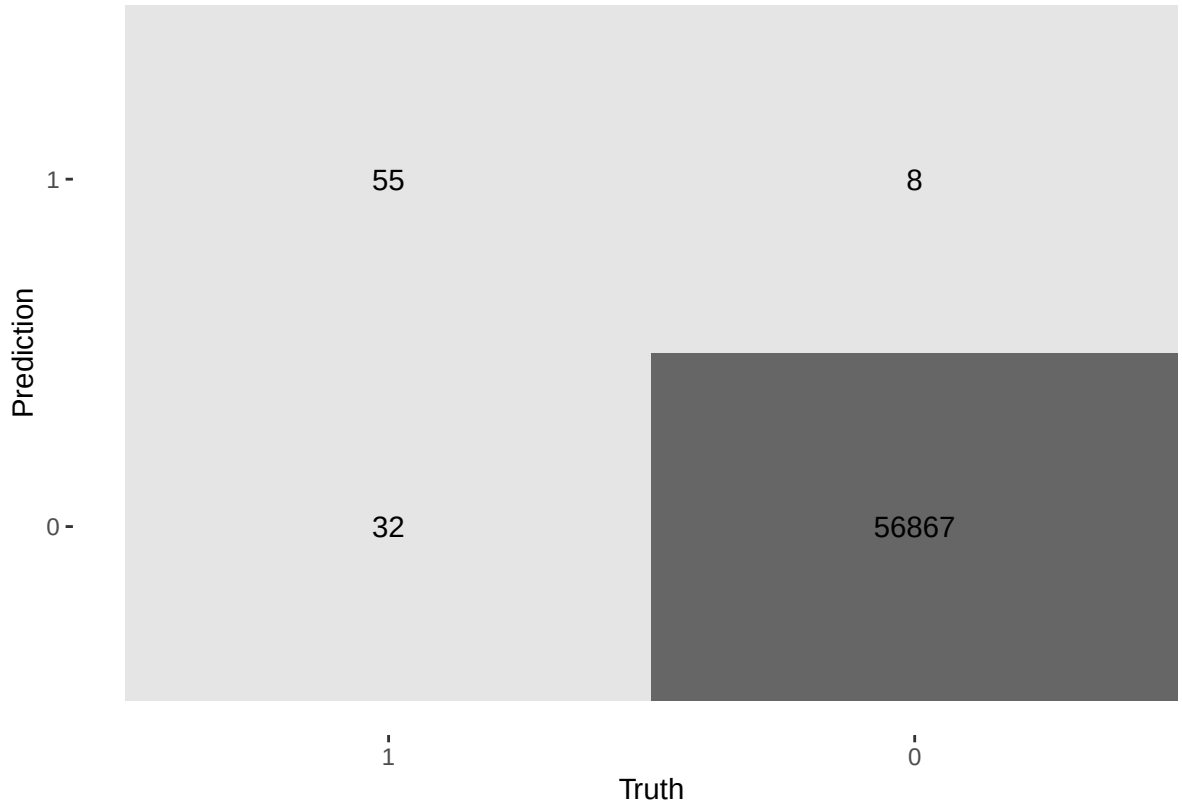
```
# fit the testing data
log_fit_test <- fit(log_workflow, cc_test)

# predict
predict(log_fit_test, new_data = cc_test, type = "class") %>%
  bind_cols(cc_test %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.999
```

As the result above, logistic regression model also has a very high accuracy with both training and testing data, and accuracy for testing data is 0.9992978.

### Heat Map

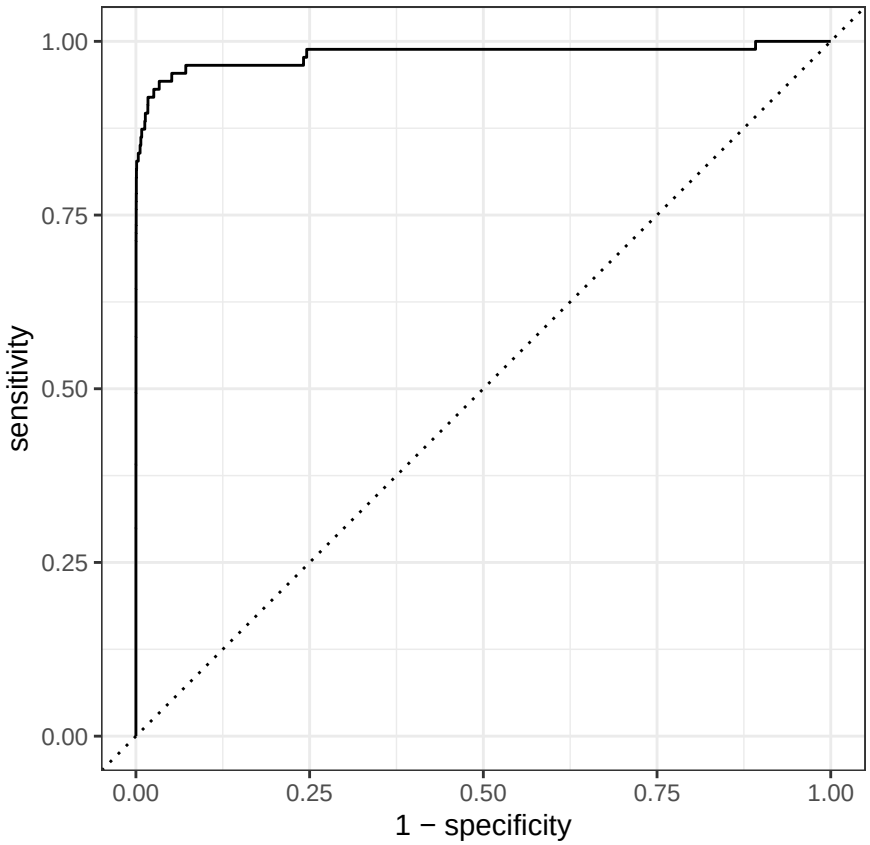
```
# heat map
augment(log_fit_test, new_data = cc_test) %>%
  conf_mat(truth = Class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




Logistic regression model successful predicted 55 out of 87 observations.

### ROC & AUC

```
# RUC
augment(log_fit_test, new_data = cc_test) %>%
  roc_curve(Class, .pred_1) %>%
  autoplot()
```




```
 # AUC
augment(log_fit_test, new_data = cc_test) %>%
  roc_auc(Class, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.981
```

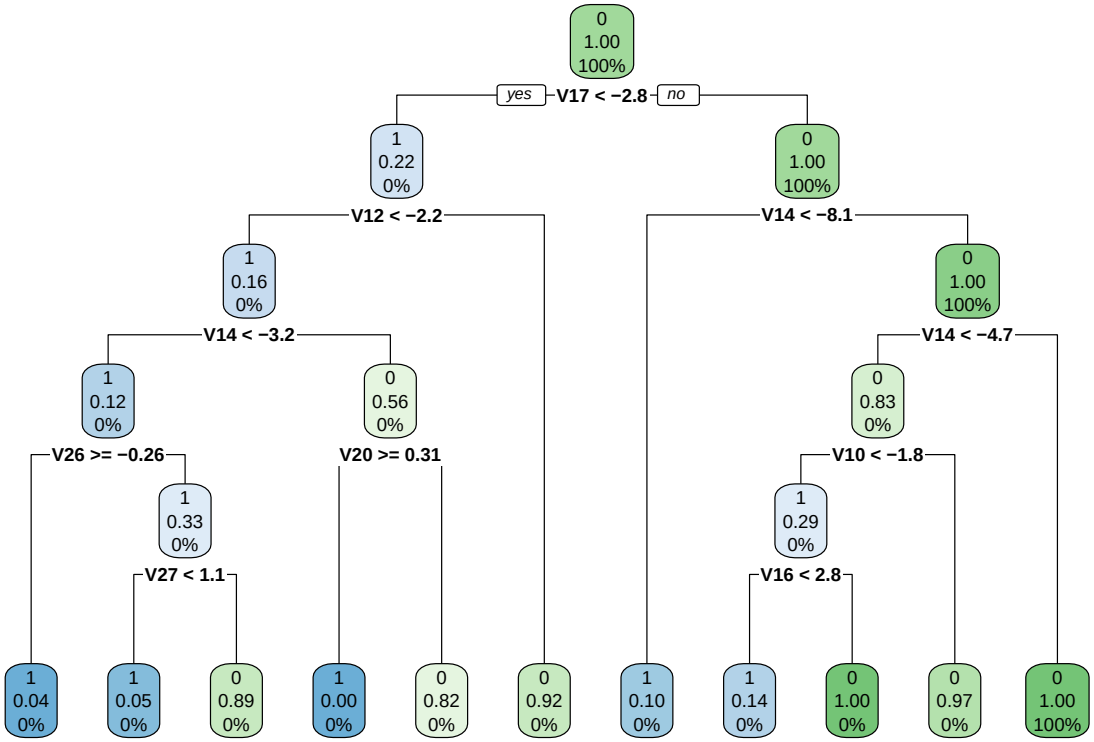
## Model 3: Decision Tree Model

For the third model, I will do decision tree model.

### Setting Up

```
# setup
dt_model <- decision_tree() %>%
  set_engine("rpart")

dt_model_class <- dt_model %>%
  set_mode("classification")

# fit training data
```

```
dt_model_fit <- dt_model_class %>%
  fit(Class ~ ., data = cc_train)

dt_model_fit %>%
  extract_fit_engine() %>%
  rpart.plot(roundint=FALSE)
```




**Fiting the Model**

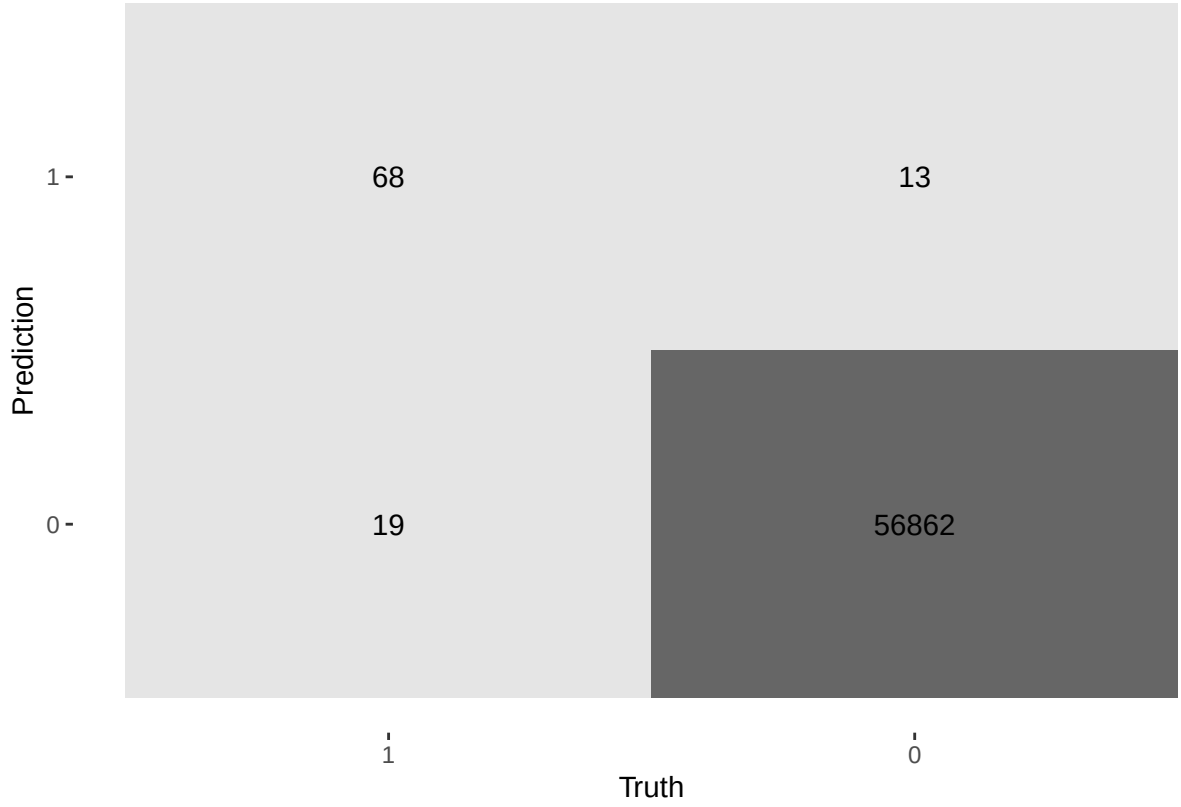
```
# fit testing data
augment(dt_model_fit, new_data = cc_test) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.999
```

After fitting the model, the accuracy of desicion tree model is 0.9994382 which is very high as well.

**Heat map**

```
augment(dt_model_fit, new_data = cc_test) %>%
  conf_mat(truth = Class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




As the result shown above, decision tree model has predicted 68 out of 87 observations.

**AUC**

```
augment(dt_model_fit, new_data = cc_test) %>%
  roc_auc(Class, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.908
```

## Model 4: LDA

Finally, I will be doing LDA and QDA model.

### Setting Up

```
#setup
lda_model <- discrim_linear() %>%
  set_mode("classification") %>%
  set_engine("MASS")

# workflow
lda_workflow <- workflow() %>%
  add_model(lda_model) %>%
  add_recipe(cc_recipe)
```

### Fitting the model

```
# fit the training data
lda_fit_train <- fit(lda_workflow, cc_train)

# predict for training data
predict(lda_fit_train, new_data = cc_train, type = "class") %>%
  bind_cols(cc_train %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.999
```

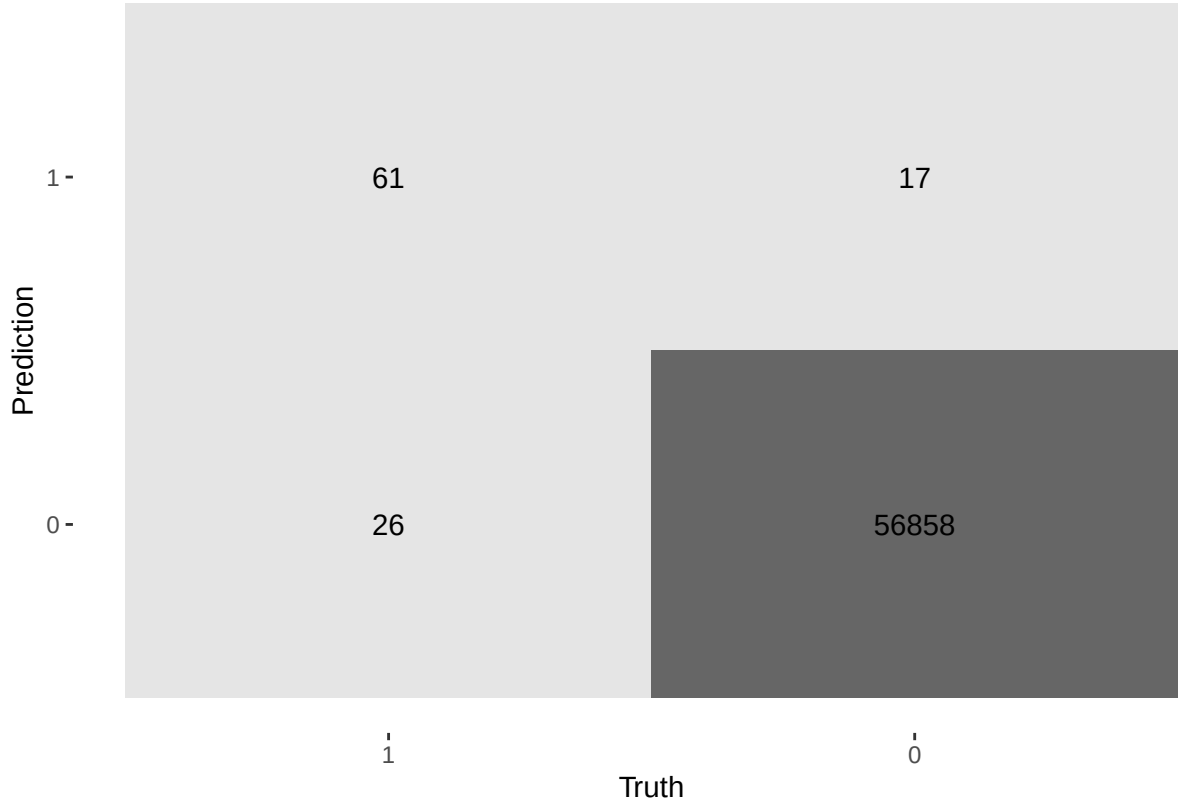
```
# fit the testing data
lda_fit_test <- fit(lda_workflow, cc_test)

# predict for testing data
predict(lda_fit_test, new_data = cc_test, type = "class") %>%
  bind_cols(cc_test %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.999
```

As the result above, LDA's accuracy is quite similar to logistic regression with 0.9992451 for testing data.

### Heat Map

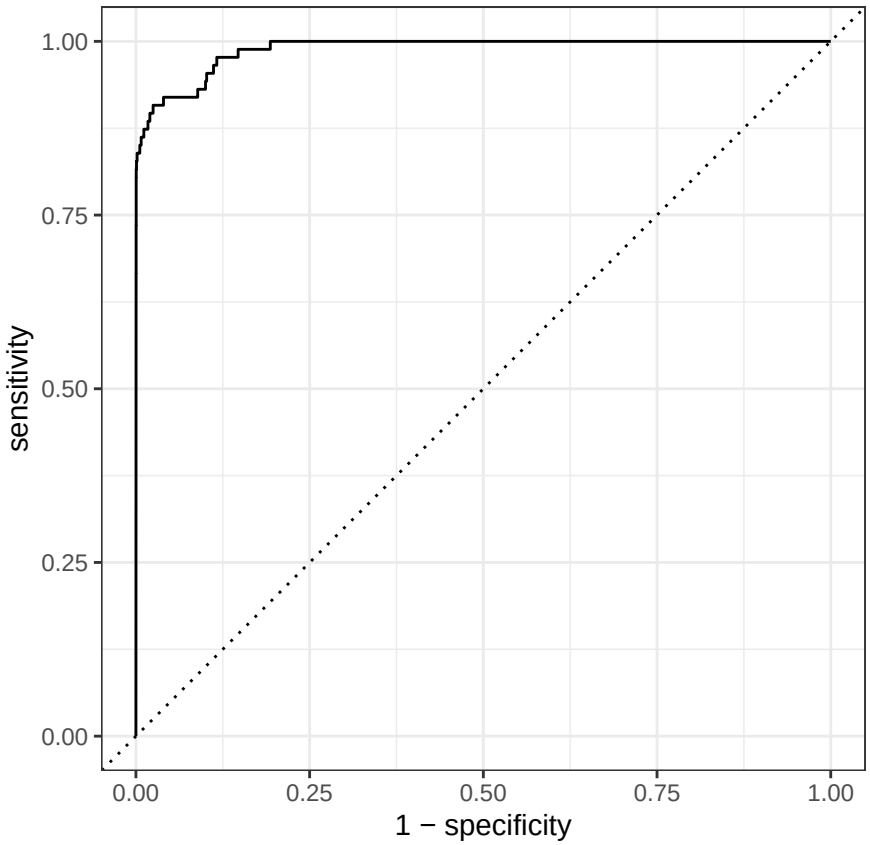
```
# heat map
augment(lda_fit_test, new_data = cc_test) %>%
  conf_mat(truth = Class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




LDA model has predicted 61 out 87 observations.

### ROC & AUC

```
# ROC
augment(lda_fit_test, new_data = cc_test) %>%
  roc_curve(Class, .pred_1) %>%
  autoplot()
```




```
 # AUC
augment(lda_fit_test, new_data = cc_test) %>%
  roc_auc(Class, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.989
```

## Model 5: QDA

Here, I am going do the QDA model as the last one.

### Setting Up

```
# setup
qda_model <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

# workflow
qda_workflow <- workflow() %>%
```

```
  add_model(qda_model) %>%
  add_recipe(cc_recipe)
```

### Fitting the model

```
# fit the training data
qda_fit_train <- fit(qda_workflow, cc_train)

# predict for training data
predict(qda_fit_train, new_data = cc_train, type = "class") %>%
  bind_cols(cc_train %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.975
```

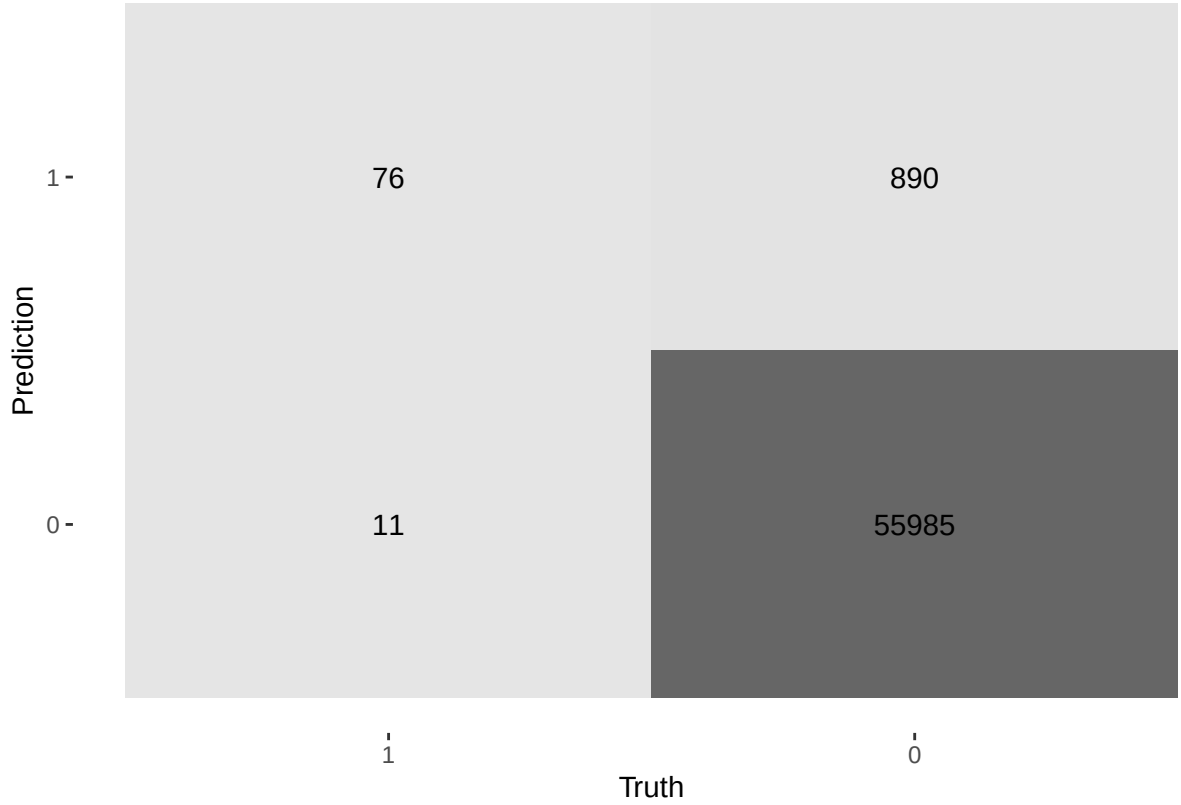
```
# fit the testing data
qda_fit_test <- fit(qda_workflow, cc_test)

# predict for testing data
predict(qda_fit_test, new_data = cc_test, type = "class") %>%
  bind_cols(cc_test %>% select(Class)) %>%
  accuracy(truth = Class, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.984
```

QDA model performed well with another high accuracy 0.9841824 for testing data, though it is not as higher as logistic regression and LDA.

### Heat Map

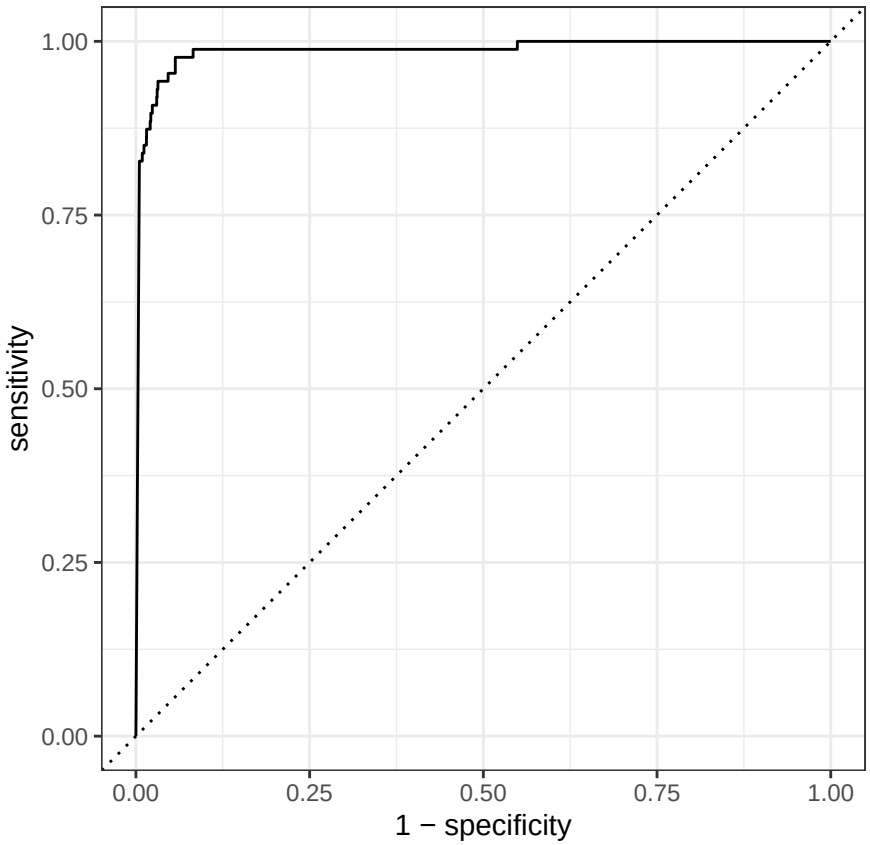
```
# heat tmap
augment(qda_fit_test, new_data = cc_test) %>%
  conf_mat(truth = Class, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




Surprisingly, QDA without the highest accuracy predicted 76 out of 87 which is the most out of all 5 models we have done so far.

### ROC & AUC

```
# ROC
augment(qda_fit_test, new_data = cc_test) %>%
  roc_curve(Class, .pred_1) %>%
  autoplot()
```




```
 # AUC
augment(qda_fit_test, new_data = cc_test) %>%
  roc_auc(Class, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.986
```

# Conclusion

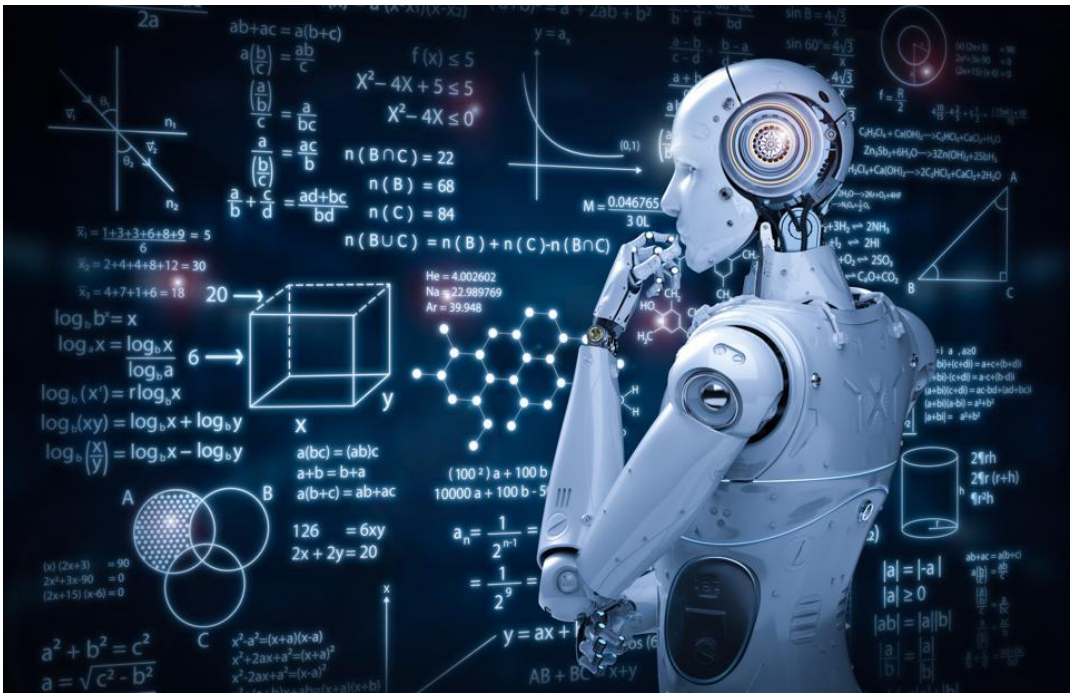
Let's do a quick recap.

We have done 4 model, all of the them have very high accuracy, but still there is only that perform the best prediction which is.

| Model | Accuracy | AUC | Prediction |
|---|---|---|---|
| Boosted Tree | 0.9995436 | 0.9698348 | 69 out of 87 |
| Logistic Regression | 0.9992978 | 0.980975 | 55 out of 87 |
| Decision Tree | 0.9994382 | 0.9078385 | 68 out of 87 |
| LDA | 0.9992451 | 0.9885043 | 61 out of 87 |
| QDA | 0.9885043 | 0.9864512 | 76 out of 87 |

In every model, we have compared their accuracy, AUC as well as prediction they have predicted by heat map. As the table shown above, with the data set is highly unbalanced as fraudulent transactions are way less than genuine transactions, QDA model surprisingly predicted the most truth values without the highest accuracy. On the other hand, boosted tree model has the highest accuracy and predicted 69 truth value out of 87. Although QDA model predicted more truth values than boosted tree model, but was the highest accuracy of all models we have fit, I would still pick boosted tree model as final model to do prediction. I believe that in the futures there will be a better method and faster algorithm to detect credit card fraud quickly and correctly though it is going to be a long way, but it is a learning process and it is all worth it at the end! Thank you for reading!

```
knitr::include_graphics("ml.jpeg")
```



# Reference


**Dataset**
Credit Card Detection

**Articles**
Credit Card Fraud Detection: Top ML Solutions in 2022

Steps to Take if You Are a Victim of Credit Card Fraud

Credit Card Fraud Detection: Everything You Need to Know