Logistic Regression and Random Forest to Determine Credit Card Default

Travis Gubbe

June 13, 2025

In this session, I examine the Credit Card Clients data set found on the UCI Machine Learning Repository website to determine if a person will default on their credit card using logistic regression and random forest.

About the Data Set

The data set can be found on the UCI Machine Learning Repository site at the following link: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients.

The data contains 24 variables and a total of 30,000 individual instances. The variables are:

- Default Payment (0 = No, 1 = Yes)
- Amount of Given Credit
- Gender (1 = Male, 2 = Female)
- Education (1 = Graduate School, 2 = University, 3 = High School, 4 = Other, 5 = Unknown)
- Marital Status (1 = Married, 2 = Single, 3 = Others)
- Age
- History of Past Payment from April 2005 to September 2005 where -2=no consumption, -1=pay duly, 0=the use of revolving credit, 1=payment delay for one month, 2=payment delay for two months, … 9=payment delay for nine months and above
- Amount of Bill Statement from April 2005 to September 2005
- Amount of Previous Payment April 2005 to September 2005

There is a separate variable for each past payment, bill statement, and previous payment from April to September.

Exploratory Analysis

First, the data set is loaded into a data frame named "credit". The data frame is also viewed to see the columns and class types.

```
data <- read.csv("~/R datasets/Credit_Card.csv")
credit = subset(data, select = c("Limit_Bal", "Sex", "Education", "Marriage", "Age", "Pay_Sep",
                                 "Pay_Aug", "Pay_July", "Pay_June", "Pay_May", "Pay_April",
                                 "Bill_Amt_Sep", "Bill_Amt_Aug", "Bill_Amt_July",
                                 "Bill_Amt_June", "Bill_Amt_May", "Bill_Amt_April",
                                 "Pay_Amt_Sep", "Pay_Amt_Aug", "Pay_Amt_July","Pay_Amt_June",
                                 "Pay_Amt_May", "Pay_Amt_April", "Default"))
#Inspect the data frame
head(credit)
```

```
##   Limit_Bal Sex Education Marriage Age Pay_Sep Pay_Aug Pay_July Pay_June
## 1     20000   2         2        1  24       2       2       -1       -1
## 2    120000   2         2        2  26      -1       2        0        0
```

```
## 3     90000   2          2         2  34        0        0         0         0
## 4     50000   2          2         1  37        0        0         0         0
## 5     50000   1          2         1  57       -1        0        -1         0
## 6     50000   1          1         2  37        0        0         0         0
##   Pay_May Pay_April Bill_Amt_Sep Bill_Amt_Aug Bill_Amt_July Bill_Amt_June
## 1      -2        -2         3913         3102           689             0
## 2       0         2         2682         1725          2682          3272
## 3       0         0        29239        14027         13559         14331
## 4       0         0        46990        48233         49291         28314
## 5       0         0         8617         5670         35835         20940
## 6       0         0        64400        57069         57608         19394
##   Bill_Amt_May Bill_Amt_April Pay_Amt_Sep Pay_Amt_Aug Pay_Amt_July Pay_Amt_June
## 1            0              0           0         689            0            0
## 2         3455           3261           0        1000         1000         1000
## 3        14948          15549        1518        1500         1000         1000
## 4        28959          29547        2000        2019         1200         1100
## 5        19146          19131        2000       36681        10000         9000
## 6        19619          20024        2500        1815          657         1000
##   Pay_Amt_May Pay_Amt_April Default
## 1           0             0       1
## 2           0          2000       1
## 3        1000          5000       0
## 4        1069          1000       0
## 5         689           679       0
## 6        1000           800       0
```

```r
#Inspect the classes of the data frame
sapply(credit, class)
```

```
##      Limit_Bal            Sex      Education       Marriage            Age
##      "integer"      "integer"      "integer"      "integer"      "integer"
##        Pay_Sep        Pay_Aug       Pay_July       Pay_June        Pay_May
##      "integer"      "integer"      "integer"      "integer"      "integer"
##      Pay_April   Bill_Amt_Sep   Bill_Amt_Aug  Bill_Amt_July  Bill_Amt_June
##      "integer"      "integer"      "integer"      "integer"      "integer"
##   Bill_Amt_May Bill_Amt_April    Pay_Amt_Sep    Pay_Amt_Aug   Pay_Amt_July
##      "integer"      "integer"      "integer"      "integer"      "integer"
##   Pay_Amt_June    Pay_Amt_May  Pay_Amt_April        Default
##      "integer"      "integer"      "integer"      "integer"
```

```r
attach(credit)
```

Next, I want to see the amount of missing values and duplicates in the data frame.

```r
sum(is.na(credit))
```

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

```r
duplicates <- credit%>%duplicated()
duplicates_amount <- duplicates%>%(table)
duplicates_amount
```

```
## .
## FALSE  TRUE
## 29965    35
```

Since there are 35 duplicates in the data, the data frame is filtered to remove the duplicates.

```r
credit <- credit%>%distinct()
#Displays how many duplicates are present in the updated data frame.
duplicates_counts_unique <- credit%>%duplicated()%>%table()
duplicates_counts_unique
```

```
## .
## FALSE
## 29965
```

Next, the factor variables are converted from their numeric values to their actual names. This is done on a copy of the credit data frame.

```r
credit1 <- data.frame(credit)
head(credit1)
```

```
##   Limit_Bal Sex Education Marriage Age Pay_Sep Pay_Aug Pay_July Pay_June
## 1     20000   2         2        1  24       2       2       -1       -1
## 2    120000   2         2        2  26      -1       2        0        0
## 3     90000   2         2        2  34       0       0        0        0
## 4     50000   2         2        1  37       0       0        0        0
## 5     50000   1         2        1  57      -1       0       -1        0
## 6     50000   1         1        2  37       0       0        0        0
##   Pay_May Pay_April Bill_Amt_Sep Bill_Amt_Aug Bill_Amt_July Bill_Amt_June
## 1      -2        -2         3913         3102           689             0
## 2       0         2         2682         1725          2682          3272
## 3       0         0        29239        14027         13559         14331
## 4       0         0        46990        48233         49291         28314
## 5       0         0         8617         5670         35835         20940
## 6       0         0        64400        57069         57608         19394
##   Bill_Amt_May Bill_Amt_April Pay_Amt_Sep Pay_Amt_Aug Pay_Amt_July Pay_Amt_June
## 1            0              0           0         689            0            0
## 2         3455           3261           0        1000         1000         1000
## 3        14948          15549        1518        1500         1000         1000
## 4        28959          29547        2000        2019         1200         1100
## 5        19146          19131        2000       36681        10000         9000
## 6        19619          20024        2500        1815          657         1000
##   Pay_Amt_May Pay_Amt_April Default
## 1           0             0       1
## 2           0          2000       1
## 3        1000          5000       0
## 4        1069          1000       0
## 5         689           679       0
## 6        1000           800       0
```

```r
#Rename factor variables to their appropriate settings
credit1$Sex[credit$Sex %in% "1"] = "Male"
credit1$Sex[credit$Sex %in% "2"] = "Female"

credit1$Education[credit$Education %in% "1"] = "Grad School"
credit1$Education[credit$Education %in% "2"] = "College"
credit1$Education[credit$Education %in% "3"] = "High School"
credit1$Education[credit$Education %in% "4"] = "Other"
credit1$Education[credit$Education %in% "5"] = "Unknown"

credit1$Marriage[credit$Marriage %in% "0"] = "Unknown"
```

```r
credit1$Marriage[credit$Marriage %in% "1"] = "Married"
credit1$Marriage[credit$Marriage %in% "2"] = "Single"
credit1$Marriage[credit$Marriage %in% "3"] = "Other"

credit1$Default[credit$Default %in% "0"] = "No"
credit1$Default[credit$Default %in% "1"] = "Yes"
#See the change in the variable names
head(credit1)
```

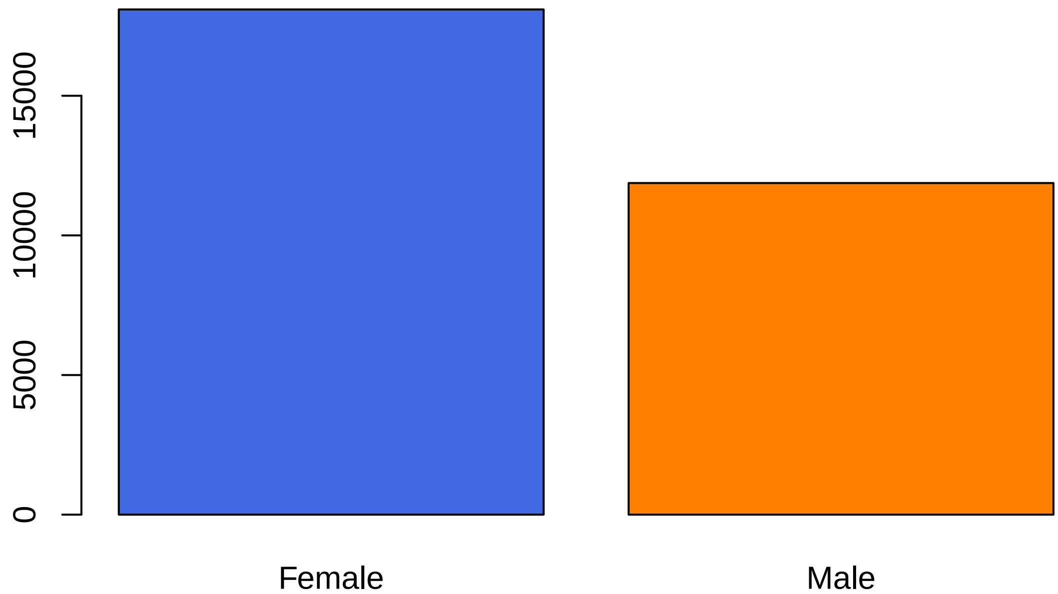
```
##   Limit_Bal    Sex   Education Marriage Age Pay_Sep Pay_Aug Pay_July Pay_June
## 1     20000 Female     College  Married  24       2       2       -1       -1
## 2    120000 Female     College   Single  26      -1       2        0        0
## 3     90000 Female     College   Single  34       0       0        0        0
## 4     50000 Female     College  Married  37       0       0        0        0
## 5     50000   Male     College  Married  57      -1       0       -1        0
## 6     50000   Male Grad School   Single  37       0       0        0        0
##   Pay_May Pay_April Bill_Amt_Sep Bill_Amt_Aug Bill_Amt_July Bill_Amt_June
## 1      -2        -2         3913         3102           689             0
## 2       0         2         2682         1725          2682          3272
## 3       0         0        29239        14027         13559         14331
## 4       0         0        46990        48233         49291         28314
## 5       0         0         8617         5670         35835         20940
## 6       0         0        64400        57069         57608         19394
##   Bill_Amt_May Bill_Amt_April Pay_Amt_Sep Pay_Amt_Aug Pay_Amt_July Pay_Amt_June
## 1            0              0           0         689            0            0
## 2         3455           3261           0        1000         1000         1000
## 3        14948          15549        1518        1500         1000         1000
## 4        28959          29547        2000        2019         1200         1100
## 5        19146          19131        2000       36681        10000         9000
## 6        19619          20024        2500        1815          657         1000
##   Pay_Amt_May Pay_Amt_April Default
## 1           0             0     Yes
## 2           0          2000     Yes
## 3        1000          5000      No
## 4        1069          1000      No
## 5         689           679      No
## 6        1000           800      No
```

Next, exploratory tables are made to view the distribution of the data set.

Data Distribution

Next, bar plots and distribution tables are created to see the proportion of the variables. This is done to see if the data is normally distributed. If the data is not normally distributed, it's advantageous to see how the data is skewed.

```r
#View the bar plots for the amount for each categorical variable
counts_Sex <- table(credit1$Sex)
barplot(counts_Sex, col = c("royalblue", "darkorange1"))
```



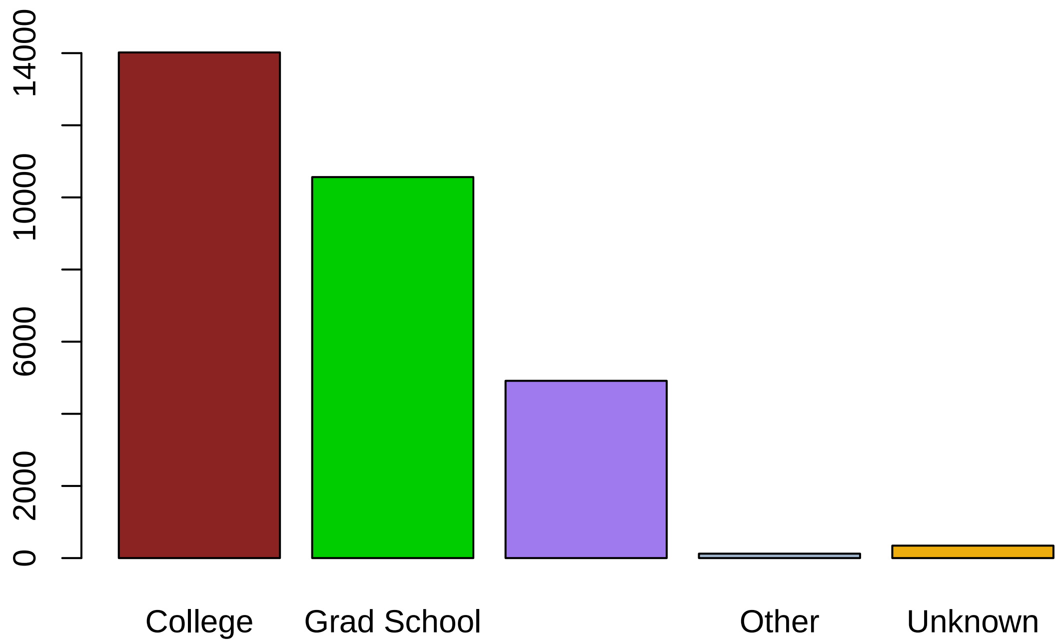

```r
#Basic table view of the amount of males and females
table(credit1$Sex)
```

```
##
## Female   Male
##  18091  11874
```

```r
#Proportion of each gender in table
prop.table(counts_Sex)
```

```
##
##    Female      Male
## 0.6037377 0.3962623
```

```r
counts_Education <- table(credit1$Education)
barplot(counts_Education, col = c("brown4", "green3", "mediumpurple2", "slategray3",
                                  "darkgoldenrod2"))
```



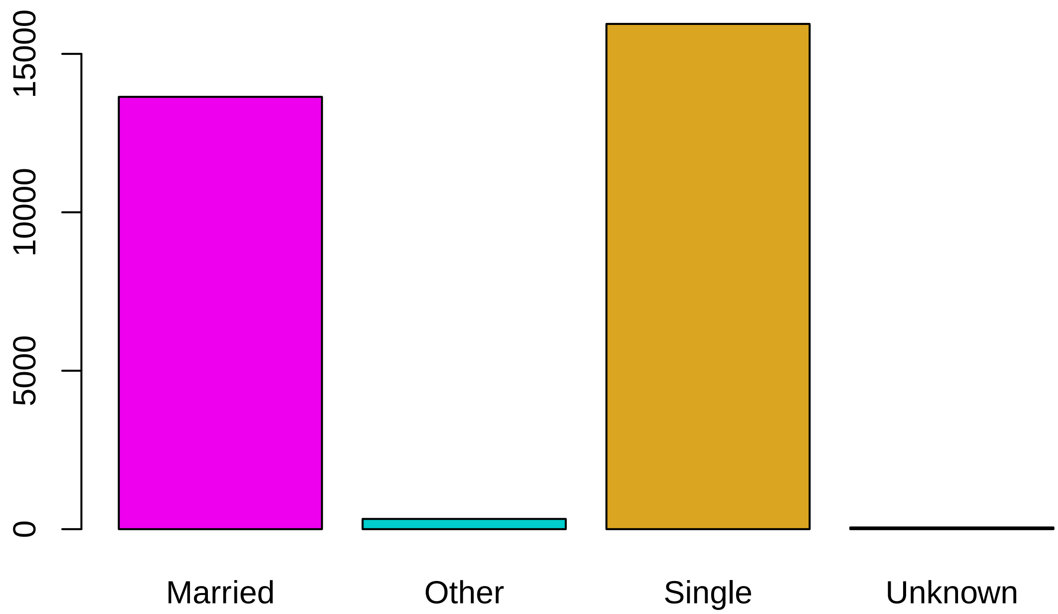

```r
table(credit1$Education)
```

```
##
##     College Grad School High School       Other     Unknown
##       14019       10563        4915         123         345
```

```r
#Proportion of each education level in table
prop.table(counts_Education)
```

```
##
##     College Grad School High School       Other     Unknown
## 0.467845820 0.352511263 0.164024695 0.004104789 0.011513432
```

```r
counts_Marriage <- table(credit1$Marriage)
barplot(counts_Marriage, col = c("magenta2", "Cyan3", "goldenrod"))
```



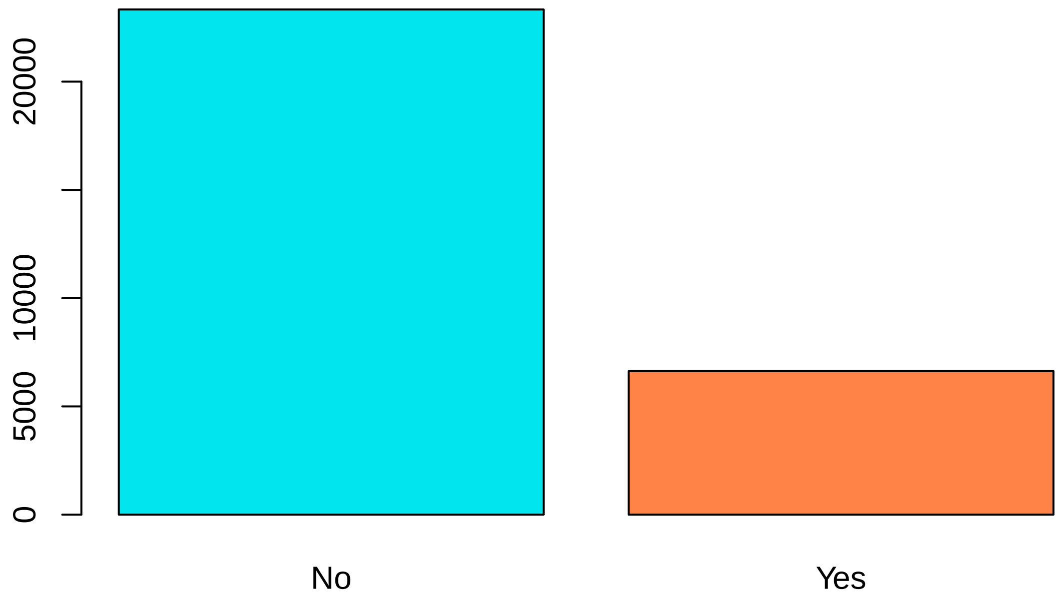

```
table(credit$Marriage)
```

```
##
##     0     1     2     3
##    54 13643 15945   323
```

```
#Proportion of each marriage status in table
prop.table(counts_Marriage)
```

```
##
##     Married       Other      Single     Unknown
## 0.455297847 0.010779242 0.532120808 0.001802102
```

```
counts_Default <- table(credit1$Default)
barplot(counts_Default, col = c("turquoise2", "sienna1"))
```




```r
table(credit$Default)
```

```
##
##     0     1
## 23335  6630
```

```r
prop.table(counts_Default)
```

```
##
##        No       Yes
## 0.7787419 0.2212581
```

```r
table.default_gender <- table(credit1$Default, credit$Sex)
prop.table(table.default_gender, 2)
```

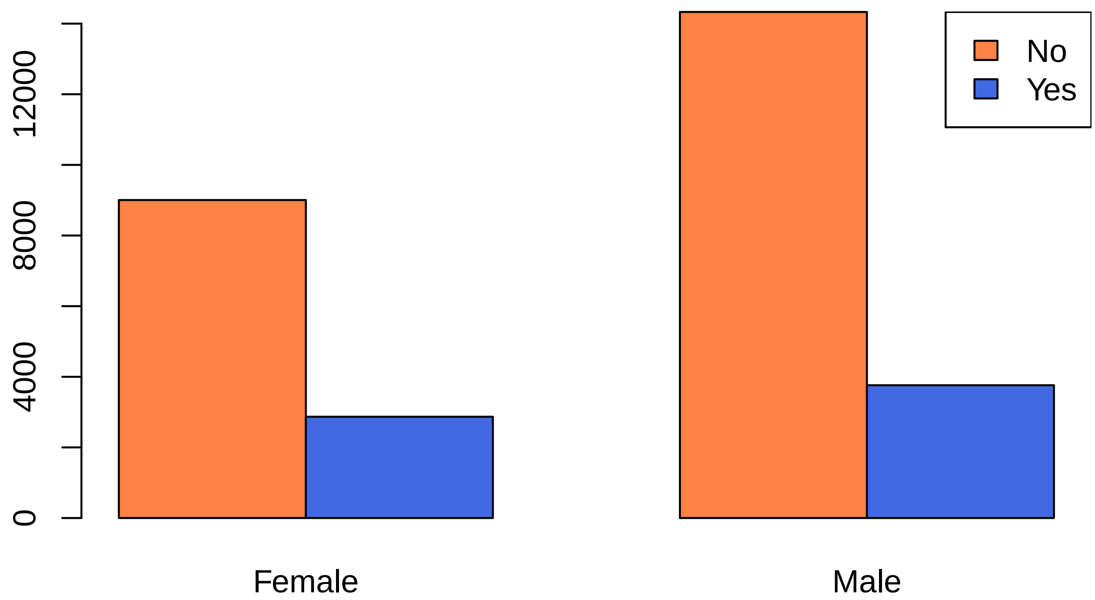
```
##
##                 1         2
##   No  0.7583797 0.7921066
##   Yes 0.2416203 0.2078934
```

```r
prop.table(table.default_gender, 1)
```

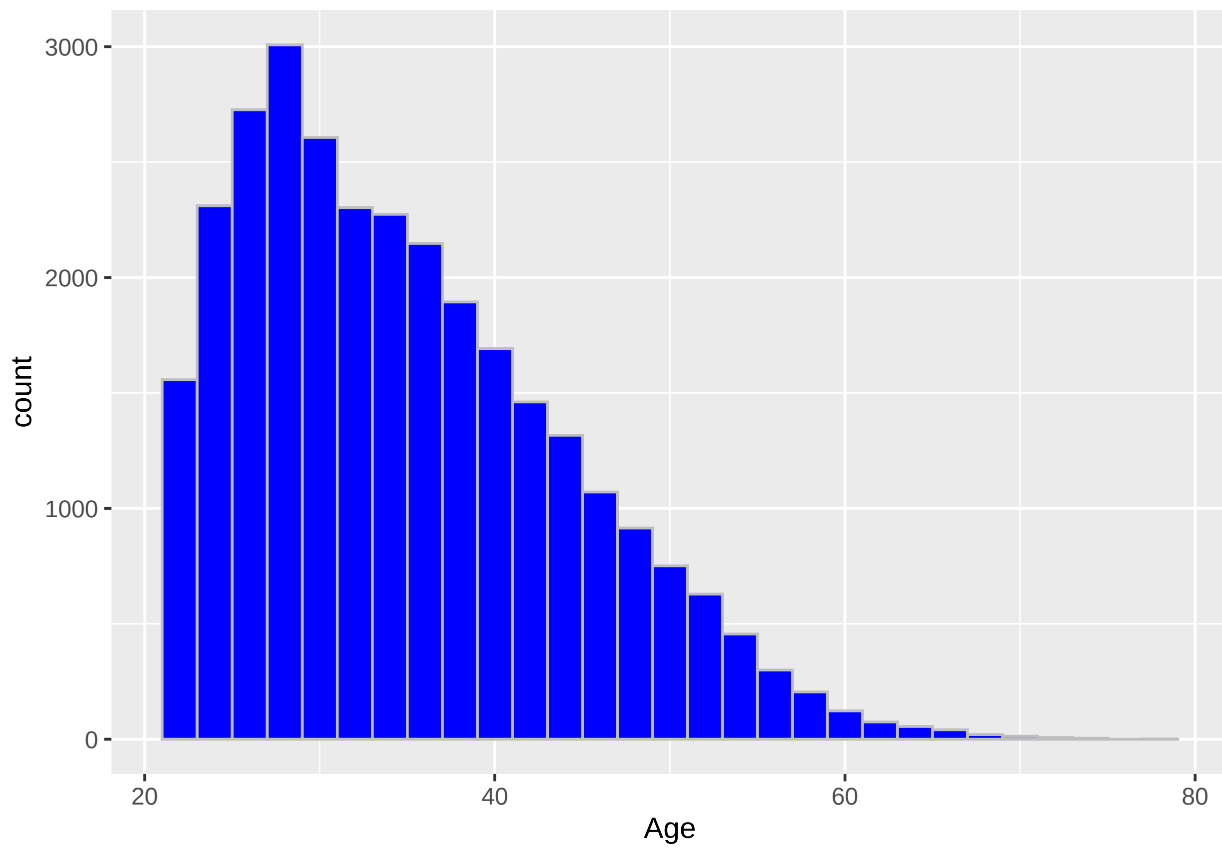
```
##
##                1        2
##   No  0.385901 0.614099
##   Yes 0.432730 0.567270
```

```r
barplot(table.default_gender, col = c("sienna1", "royalblue"), beside = T,
        names.arg = c("Female", "Male"))
```

```r
legend("topright", legend = c("No", "Yes"), fill = c("sienna1", "royalblue"))
```

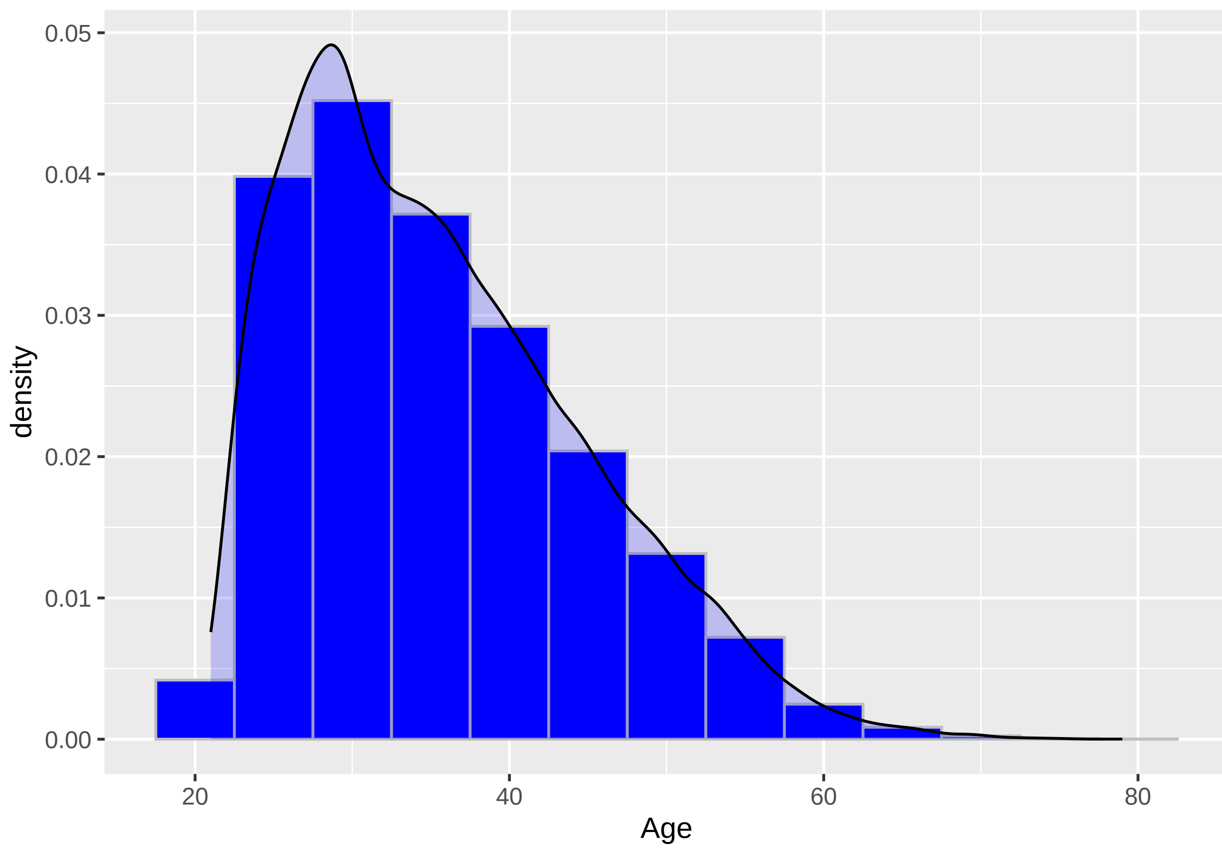



```r
ggplot(data = credit, aes(x = Age)) + geom_histogram(fill = "Blue", col = "Grey", bins = 30)
```




```
ggplot(data = credit, aes(x = Age)) + geom_histogram(aes(y = ..density..), fill = "Blue", col = "Grey",
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
mean(credit1$Age)
```

```
## [1] 35.48797
```

Looking at the created charts and tables, the data has more females than males. In addition, the age distribution is skewed to the right, meaning the data is represented by younger participants. As such, it may be easier to predict credit card default for females or for younger participants compared to males or older participants.

Scaling the Data

Before setting up the prediction model, all variables except for Default (the variable we are trying to predict) are scaled so the data is standardized.

```
credit = credit %>%
  mutate(across(1:23, scale))
```

Train and Test Sets

Before creating prediction models, training and testing data sets are created. A training data set is a subset of examples used to train the model, while the testing data set is a subset used to test the training model.

```
#Initializes number generator.
set.seed(123)
#New sample created for the training and testing data sets. The data is split with 75% in training and 
sample <- sample(c(TRUE, FALSE), nrow(credit1), replace = TRUE, prob = c(0.75, 0.25))
train_set <- credit[sample, ]
test_set <- credit[!sample, ]
```

Sampling the Data

From the bar plots, it is clear there is an imbalance between those who default and those who did not in the data. This could cause issues in creating a prediction model, which would most likely skew towards predicting much more "No" answers since there are more within the sampled data. To solve this issue, oversampling and undersampling the training set data can be done. Oversampling duplicates random samples from the minority class, while undersampling randomly reduces samples from the majority class. Doing both helps to "even out" the bias and possibly improve the model's overall performance.

The random oversampling and undersampling is performed below:

```
credit_balance_train <- ovun.sample(Default ~., data = train_set, method = "over")$data
#credit_balance_train <- ovun.sample(Default ~., data = train_set, N = nrow(train_set), seed = 1, metho

#credit_balance <- ovun.sample(Default ~., data = credit1, N = nrow(credit1),
#                              p = 0.5, seed = 1, method = "both")$data
```

Now that the training and testing data sets are created and have been randomly sampled, prediction analysis methods such as logistic regression and random forest can be completed.

Logistic Regression

First, logistic regression is done to find the probability of default for an individual. Logistic regression models the probability that a response variable (Y) belongs to a particular category. This method uses maximum likelihood to fit the model in the range between 0 and 1.

Logistic regression is a classification method great for a yes/no response. A number closer to 1 represents "Yes", while a number closer to 0 represents "No".

A logistic regression model is created below, which is then used to predict the probabilities of credit card default for three individuals:

```
# With Training Set
#fit_glm <- glm(Default ~ ., data = credit_balance_train, family = binomial())
#Displays summary of the logistic regression model. Use step AIC to narrow logistic model based on stat
#summary(fit_glm)
#stepAIC(fit_glm)

#Next logistic regression model, removing variables based on AIC and statistical significance.
fit_glm2 <- glm(Default ~Limit_Bal+Sex+Education+Marriage+Age+Pay_Sep+Pay_Aug+
                  Pay_July+Bill_Amt_Sep+Bill_Amt_July+Pay_Amt_Sep+Pay_Amt_Aug+
                  Pay_Amt_June+Pay_Amt_May+Pay_Amt_April,
                  data = credit_balance_train, family = binomial())

summary(fit_glm2)
```

```
##
## Call:
## glm(formula = Default ~ Limit_Bal + Sex + Education + Marriage +
##     Age + Pay_Sep + Pay_Aug + Pay_July + Bill_Amt_Sep + Bill_Amt_July +
##     Pay_Amt_Sep + Pay_Amt_Aug + Pay_Amt_June + Pay_Amt_May +
##     Pay_Amt_April, family = binomial(), data = credit_balance_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.20469    0.01213 -16.876  < 2e-16 ***
## Limit_Bal    -0.09177    0.01523  -6.027 1.67e-09 ***
## Sex          -0.02481    0.01171  -2.120  0.03402 *
```

```
## Education     -0.07410    0.01264  -5.862 4.57e-09 ***
## Marriage      -0.11086    0.01287  -8.612  < 2e-16 ***
## Age            0.06778    0.01292   5.245 1.56e-07 ***
## Pay_Sep        0.58536    0.01505  38.898  < 2e-16 ***
## Pay_Aug        0.11379    0.01863   6.108 1.01e-09 ***
## Pay_July       0.09232    0.01709   5.403 6.56e-08 ***
## Bill_Amt_Sep  -0.28548    0.03578  -7.979 1.47e-15 ***
## Bill_Amt_July  0.20913    0.03750   5.577 2.44e-08 ***
## Pay_Amt_Sep   -0.24585    0.02471  -9.950  < 2e-16 ***
## Pay_Amt_Aug   -0.24590    0.02903  -8.471  < 2e-16 ***
## Pay_Amt_June  -0.05234    0.01630  -3.212  0.00132 **
## Pay_Amt_May   -0.02527    0.01486  -1.701  0.08890 .
## Pay_Amt_April -0.02306    0.01571  -1.468  0.14213
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48270  on 34819  degrees of freedom
## Residual deviance: 42681  on 34804  degrees of freedom
## AIC: 42713
##
## Number of Fisher Scoring iterations: 5
```

```r
#Third attempt at logistic regression model
fit_glm3 <- glm(Default ~Limit_Bal+Sex+Education+Marriage+Age+Pay_Sep+Pay_Aug+
                  Pay_July+Bill_Amt_Sep+Bill_Amt_July+Pay_Amt_Sep+Pay_Amt_Aug+
                  Pay_Amt_June, data = credit_balance_train, family = binomial())

summary(fit_glm3)
```

```
##
## Call:
## glm(formula = Default ~ Limit_Bal + Sex + Education + Marriage +
##     Age + Pay_Sep + Pay_Aug + Pay_July + Bill_Amt_Sep + Bill_Amt_July +
##     Pay_Amt_Sep + Pay_Amt_Aug + Pay_Amt_June, family = binomial(),
##     data = credit_balance_train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.20414    0.01212 -16.840  < 2e-16 ***
## Limit_Bal     -0.09720    0.01504  -6.461 1.04e-10 ***
## Sex           -0.02473    0.01170  -2.113 0.034594 *
## Education     -0.07423    0.01264  -5.873 4.27e-09 ***
## Marriage      -0.11145    0.01287  -8.660  < 2e-16 ***
## Age            0.06801    0.01292   5.264 1.41e-07 ***
## Pay_Sep        0.58663    0.01504  38.997  < 2e-16 ***
## Pay_Aug        0.11297    0.01863   6.065 1.32e-09 ***
## Pay_July       0.09234    0.01709   5.404 6.52e-08 ***
## Bill_Amt_Sep  -0.29532    0.03551  -8.317  < 2e-16 ***
## Bill_Amt_July  0.21464    0.03741   5.738 9.57e-09 ***
## Pay_Amt_Sep   -0.25152    0.02469 -10.187  < 2e-16 ***
## Pay_Amt_Aug   -0.25311    0.02894  -8.745  < 2e-16 ***
## Pay_Amt_June  -0.05493    0.01634  -3.361 0.000778 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48270  on 34819  degrees of freedom
## Residual deviance: 42687  on 34806  degrees of freedom
## AIC: 42715
##
## Number of Fisher Scoring iterations: 5
```

```
stepAIC(fit_glm3)
```

```
## Start:  AIC=42714.68
## Default ~ Limit_Bal + Sex + Education + Marriage + Age + Pay_Sep +
##     Pay_Aug + Pay_July + Bill_Amt_Sep + Bill_Amt_July + Pay_Amt_Sep +
##     Pay_Amt_Aug + Pay_Amt_June
##
##                 Df Deviance   AIC
## <none>                42687 42715
## - Sex            1    42691 42717
## - Pay_Amt_June   1    42699 42725
## - Age            1    42714 42740
## - Pay_July       1    42716 42742
## - Bill_Amt_July  1    42721 42747
## - Education      1    42721 42747
## - Pay_Aug        1    42723 42749
## - Limit_Bal      1    42729 42755
## - Bill_Amt_Sep   1    42760 42786
## - Marriage       1    42762 42788
## - Pay_Amt_Aug    1    42788 42814
## - Pay_Amt_Sep    1    42829 42855
## - Pay_Sep        1    44288 44314
```

```
##
## Call:  glm(formula = Default ~ Limit_Bal + Sex + Education + Marriage +
##     Age + Pay_Sep + Pay_Aug + Pay_July + Bill_Amt_Sep + Bill_Amt_July +
##     Pay_Amt_Sep + Pay_Amt_Aug + Pay_Amt_June, family = binomial(),
##     data = credit_balance_train)
##
## Coefficients:
##   (Intercept)      Limit_Bal            Sex      Education       Marriage
##      -0.20414       -0.09720       -0.02473       -0.07423       -0.11145
##           Age        Pay_Sep        Pay_Aug       Pay_July   Bill_Amt_Sep
##       0.06801        0.58663        0.11297        0.09234       -0.29532
## Bill_Amt_July    Pay_Amt_Sep    Pay_Amt_Aug   Pay_Amt_June
##       0.21464       -0.25152       -0.25311       -0.05493
##
## Degrees of Freedom: 34819 Total (i.e. Null);  34806 Residual
## Null Deviance:       48270
## Residual Deviance: 42690     AIC: 42710
```

```
#VIF of the 3rd logistic regression model. VIF scores for Bill_Amt_Sep and Bill_Amt_July are a little h
vif(fit_glm3)
```

```
##     Limit_Bal           Sex      Education       Marriage            Age
##      1.493893      1.023597       1.133318       1.233396       1.284692
```

```
##       Pay_Sep       Pay_Aug      Pay_July  Bill_Amt_Sep Bill_Amt_July
##      1.557327      2.609168      2.240797      8.829168      9.558931
##   Pay_Amt_Sep   Pay_Amt_Aug  Pay_Amt_June
##      1.182104      1.259967      1.073581
```

```
fit_glm4 <- glm(Default ~Limit_Bal+Education+Marriage+Age+Pay_Sep+Pay_Aug+
                Pay_July+Pay_Amt_Sep+Pay_Amt_Aug+Pay_Amt_June,
                data = credit_balance_train, family = binomial())

summary(fit_glm4)
```

```
##
## Call:
## glm(formula = Default ~ Limit_Bal + Education + Marriage + Age +
##     Pay_Sep + Pay_Aug + Pay_July + Pay_Amt_Sep + Pay_Amt_Aug +
##     Pay_Amt_June, family = binomial(), data = credit_balance_train)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.20685    0.01214 -17.042  < 2e-16 ***
## Limit_Bal    -0.13874    0.01410  -9.838  < 2e-16 ***
## Education    -0.08429    0.01257  -6.707 1.99e-11 ***
## Marriage     -0.11104    0.01282  -8.664  < 2e-16 ***
## Age           0.07100    0.01278   5.557 2.74e-08 ***
## Pay_Sep       0.57781    0.01494  38.686  < 2e-16 ***
## Pay_Aug       0.08664    0.01834   4.725 2.30e-06 ***
## Pay_July      0.10033    0.01689   5.941 2.83e-09 ***
## Pay_Amt_Sep  -0.26384    0.02547 -10.357  < 2e-16 ***
## Pay_Amt_Aug  -0.22072    0.02801  -7.881 3.26e-15 ***
## Pay_Amt_June -0.07356    0.01663  -4.422 9.77e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48270  on 34819  degrees of freedom
## Residual deviance: 42786  on 34809  degrees of freedom
## AIC: 42808
##
## Number of Fisher Scoring iterations: 5
```

```
stepAIC(fit_glm4)
```

```
## Start:  AIC=42808.2
## Default ~ Limit_Bal + Education + Marriage + Age + Pay_Sep +
##     Pay_Aug + Pay_July + Pay_Amt_Sep + Pay_Amt_Aug + Pay_Amt_June
##
##                Df Deviance   AIC
## <none>               42786 42808
## - Pay_Aug       1    42809 42829
## - Pay_Amt_June  1    42809 42829
## - Age           1    42817 42837
## - Pay_July      1    42822 42842
## - Education     1    42831 42851
## - Marriage      1    42861 42881
```

```
## - Pay_Amt_Aug   1    42869 42889
## - Limit_Bal     1    42884 42904
## - Pay_Amt_Sep   1    42935 42955
## - Pay_Sep       1    44362 44382
```

```
##
## Call:  glm(formula = Default ~ Limit_Bal + Education + Marriage + Age +
##     Pay_Sep + Pay_Aug + Pay_July + Pay_Amt_Sep + Pay_Amt_Aug +
##     Pay_Amt_June, family = binomial(), data = credit_balance_train)
##
## Coefficients:
##  (Intercept)     Limit_Bal     Education      Marriage           Age
##     -0.20685      -0.13874      -0.08429      -0.11104       0.07100
##      Pay_Sep       Pay_Aug      Pay_July   Pay_Amt_Sep   Pay_Amt_Aug
##      0.57781       0.08664       0.10033      -0.26384      -0.22072
## Pay_Amt_June
##     -0.07356
##
## Degrees of Freedom: 34819 Total (i.e. Null);  34809 Residual
## Null Deviance:       48270
## Residual Deviance: 42790     AIC: 42810
```

```r
#VIF Values are much better.
vif(fit_glm4)
```

```
##    Limit_Bal    Education     Marriage          Age      Pay_Sep      Pay_Aug
##     1.312723     1.124306     1.227232     1.261724     1.538249     2.546716
##     Pay_July  Pay_Amt_Sep  Pay_Amt_Aug Pay_Amt_June
##     2.206446     1.118786     1.089299     1.061608
```

```r
#Fit_glm4 seems to be the best model.
```

```r
pred_probs <- predict.glm(fit_glm4, newdata = test_set, type = "response")
#Displays the predictions for a few values.
head(pred_probs)
```

```
##         2         4         5         8        11        16
## 0.3723262 0.5442417 0.3462276 0.4228482 0.4649700 0.5577874
```

```r
#Sorts predictions into their respective class (0 or 1) depending on their value.
pred <- ifelse(pred_probs<0.5, 0,1)
#Creates and displays the confusion matrix table based on the actual and predicted values.
confusion_table <- table(test_set$Default, pred)
confusion_table
```

```
##    pred
##        0    1
##   0 4235 1613
##   1  610 1030
```

```r
#Creates the confusion matrix statistics for the logistic regression model.
cm_log <- confusionMatrix(confusion_table, positive = '1', mode = "everything")
#Saves the accuracy, precision, and recall values.
log_accuracy = accuracy(test_set$Default, pred)
log_precision = cm_log$byClass['Precision']
log_recall = cm_log$byClass['Recall']
log_pos_precision = cm_log$byClass['Neg Pred Value']
#Prints the accuracy, precision, and recall values.
```

```r
print(paste("Accuracy: ", round(log_accuracy,3)))
```

```
## [1] "Accuracy:  0.703"
```

```r
print(paste("Precision: ", round(log_precision,3)))
```

```
## [1] "Precision:  0.628"
```

```r
print(paste("Recall: ", round(log_recall,3)))
```

```
## [1] "Recall:  0.39"
```

```r
print(paste("Default Precision: ", round(log_pos_precision,3)))
```

```
## [1] "Default Precision:  0.724"
```

Once the model was run, the accuracy, precision, and recall were found for the prediction model. Accuracy describes how often the model is correct in its overall prediction. Precision identifies how often the model identifies those who default on their credit card out of all who do so, while recall identifies how often the model correctly identifies those who default on their credit card.Another way of describing precision and recall is precision is a measure of quality, while recall is a measure of quantity.

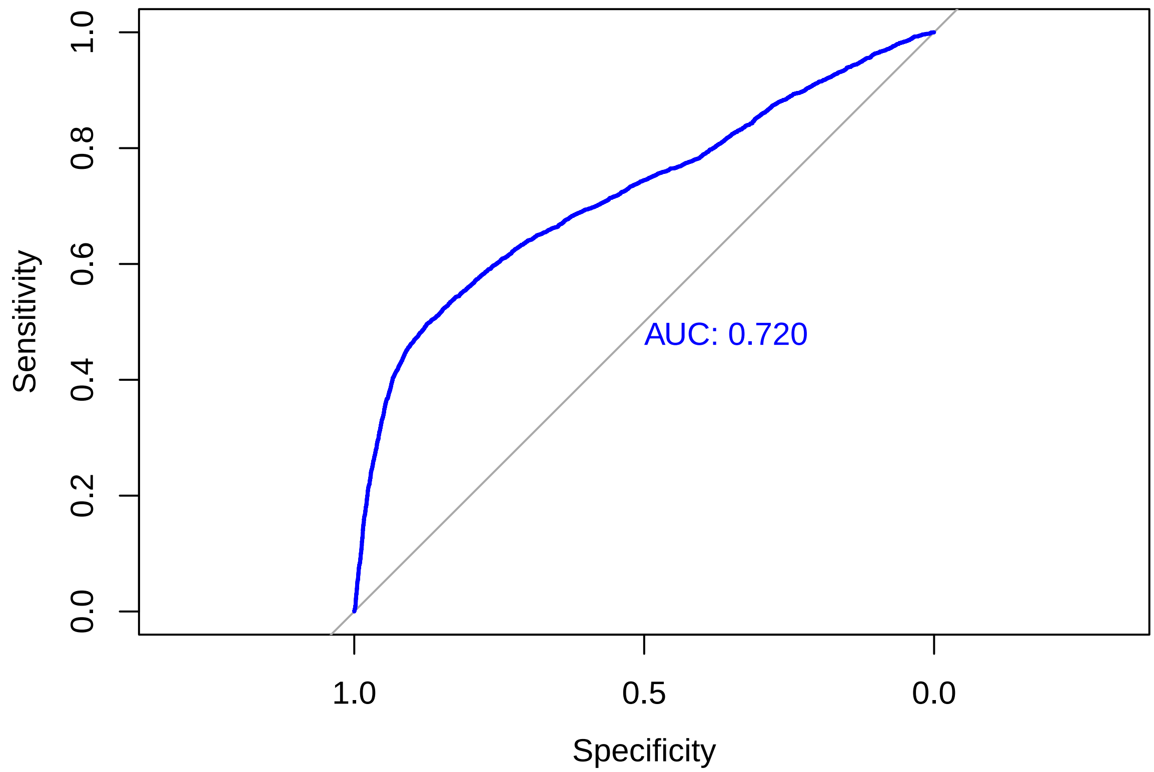
In the case of the logistic regression model, the accuracy was 70.3%, precision was 62.8%, and recall was 39%. The precision for predicting actual default cases correctly was 72.4%. Overall, the logistic regression model was fairly decent at its predicting whether a client would default.

The regression model's accuracy, specificity, and sensitivity can be improved by optimizing the cutoff point for the model. One way of doing so is using the Receiver Operating Characteristic (ROC) curve, which plots the true positive (sensitivity) against the false positive rate against various thresholds. The AUC curve can be used to measure the performance of the model, with a higher AUC number demonstrating better model performance. An AUC 0.8 and above indicates good model performance. The model has an AUC score of 0.721, which indicates acceptable model performance. This will be a factor to keep in mind for potential model improvements.

```r
prob <- predict(fit_glm4, type = "response")
# Create ROC curve
roc_obj <- roc(credit_balance_train$Default, prob, plot = TRUE, col = "blue", print.auc = TRUE)
```

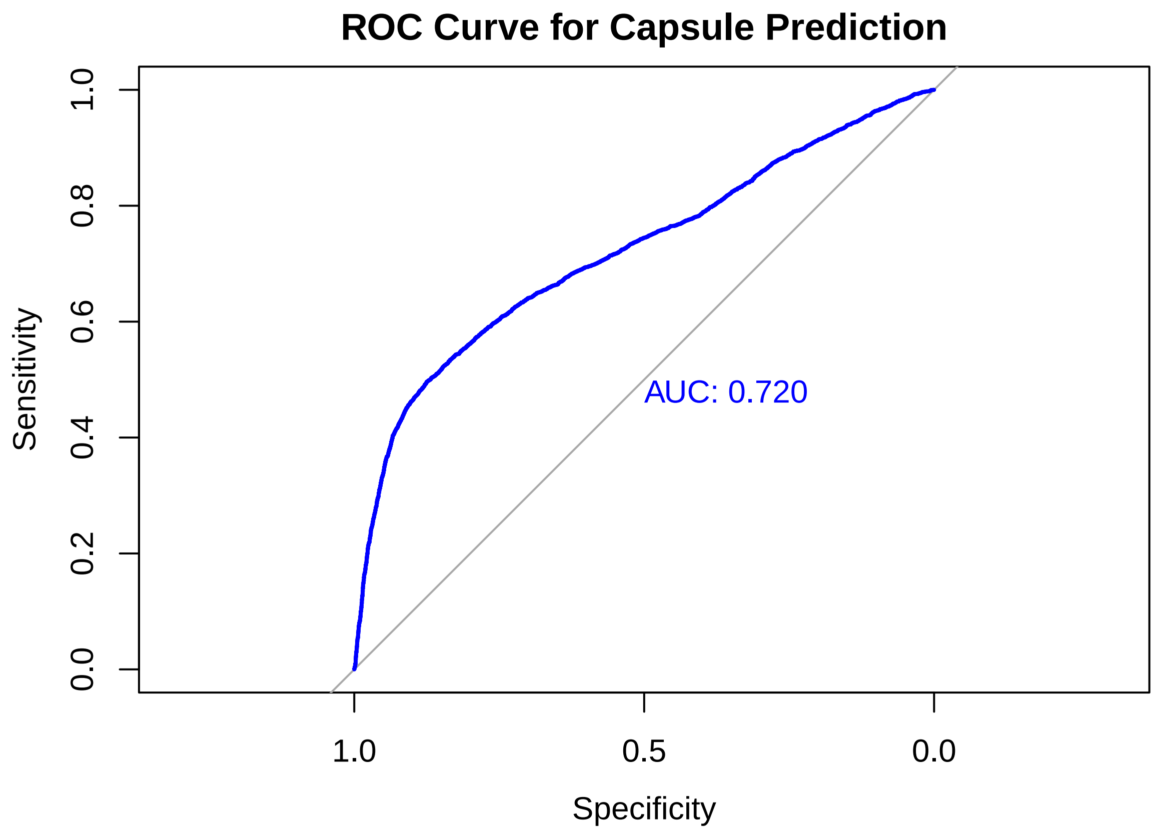
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```r
# Find optimal cutoff (maximizes sensitivity + specificity)
#AUC performance = 0.72

#Plot ROC curve with AUC value
plot(roc_obj, col = "blue", print.auc = TRUE, main = "ROC Curve for Capsule Prediction")
```




```r
opt <- coords(roc_obj, "best", ret = c("threshold", "sensitivity", "specificity"))
print(opt)
```

```
##   threshold sensitivity specificity
## 1  0.566382   0.4963942   0.8749929
```

```r
#threshold = 0.566
#sensitivity = 0.496
#specificity = 0.875
```

The ROC curve supplies various cutoff values for the logistic regression model. From the ROC curve, the optimal cutoff point to maximize all three measurements (accuracy, specificity, and sensitvity) is 0.566, while the cutoffs to maximize sensitivity and specificity are 0.496 and 0.875, respectively. The optimal cutoff from the AUC curve, which maximizes both sensitivity and specificity, is 0.72. There is a tradeoff for each cutoff point, so it is up to the user to determine which is best for the purpose of the model. I chose to use 0.566 since I want to maximize both the overall accuracy of the model while accurately identifying those who will default on their credit card.

```r
pred_new <- ifelse(pred_probs<0.566, 0,1)
#Creates and displays the confusion matrix table based on the actual and predicted values.
confusion_table_new <- table(test_set$Default, pred_new)
confusion_table_new
```

```
##    pred_new
##        0    1
##   0 5162  686
##   1  812  828
```

```
#Creates the confusion matrix statistics for the logistic regression model.
cm_log_opt <- confusionMatrix(confusion_table_new, positive = '1', mode = "everything")
#Saves the accuracy, precision, and recall values.
log_accuracy_opt = accuracy(test_set$Default, pred_new)
log_precision_opt = cm_log_opt$byClass['Precision']
log_recall_opt = cm_log_opt$byClass['Recall']
log_pos_precision_opt = cm_log_opt$byClass['Neg Pred Value']
#Prints the accuracy, precision, and recall values.
print(paste("Accuracy: ", round(log_accuracy_opt,3)))
```

```
## [1] "Accuracy:  0.8"
```

```
print(paste("Precision: ", round(log_precision_opt,3)))
```

```
## [1] "Precision:  0.505"
```

```
print(paste("Recall: ", round(log_recall_opt,3)))
```

```
## [1] "Recall:  0.547"
```

```
print(paste("Default Precision: ", round(log_pos_precision_opt,3)))
```

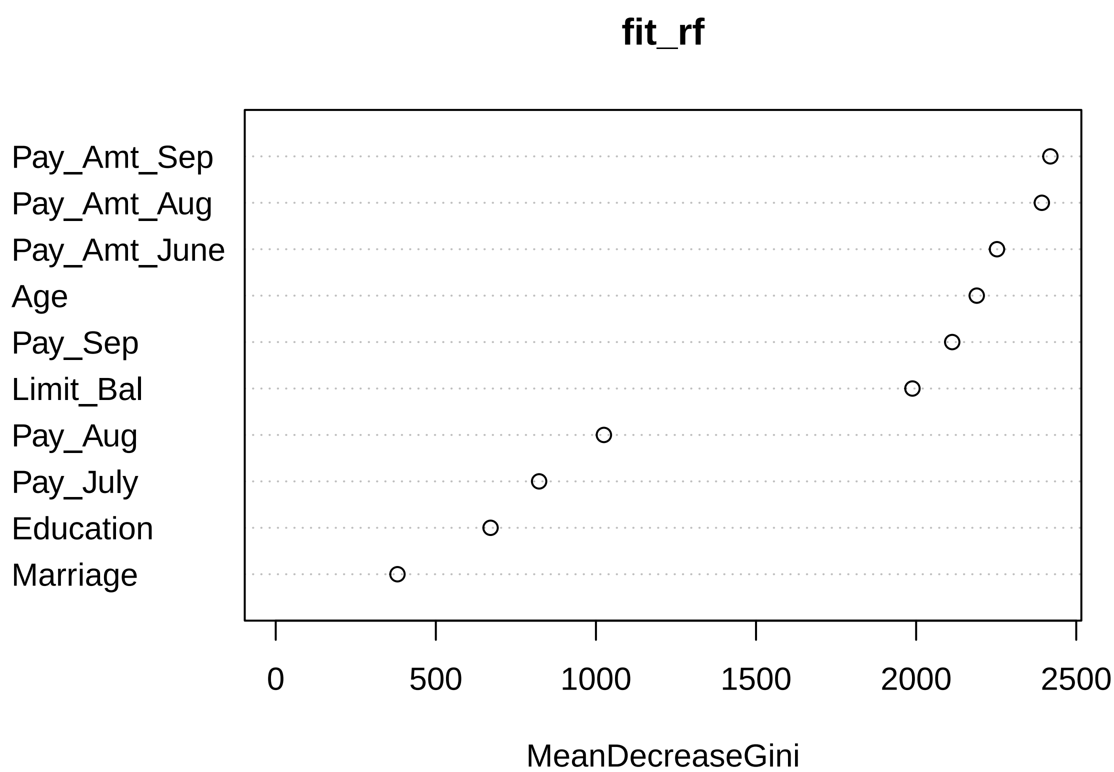
```
## [1] "Default Precision:  0.883"
```

Random Forest

Another prediction model used is random forest. Random forest is a classifying method consisting of many decision trees. By creating a "forest" of decision trees, the classifying model hopes to select it's best model by running many different decision trees and "takes the majority" to determine classification. To do so, random forest uses out-of-bag sampling.

A random forest model is created to determine the probability of credit card default:

```
set.seed(123)
#Random Forest for variables. mtry = 3 since there are 10 variables (square root of 10 is close to 4).
fit_rf <- randomForest(factor(Default) ~Limit_Bal+Education+Marriage+Age+Pay_Sep+Pay_Aug+
                  Pay_July+Pay_Amt_Sep+Pay_Amt_Aug+Pay_Amt_June,
                  mtry = 3, data = credit_balance_train)
varImpPlot(fit_rf)
```




```r
#Predicts values in the test set.
predict_rf <- predict(fit_rf, test_set)
#Creates the confusion matrix table for the random forest model.
confusion_table_rf <- table(test_set$Default, predict_rf)
#Creates and displays the confusion matrix statistics for the random forest model.
cm_rf <- confusionMatrix(confusion_table_rf, positive = '1', mode = "everything")
cm_rf
```

```
## Confusion Matrix and Statistics
##
##    predict_rf
##        0    1
##   0 5236  612
##   1  864  776
##
##                Accuracy : 0.8029
##                  95% CI : (0.7937, 0.8118)
##     No Information Rate : 0.8146
##     P-Value [Acc > NIR] : 0.9955
##
##                   Kappa : 0.3901
##
##  Mcnemar's Test P-Value : 6.435e-11
##
##             Sensitivity : 0.5591
##             Specificity : 0.8584
```

```
##          Pos Pred Value : 0.4732
##          Neg Pred Value : 0.8953
##               Precision : 0.4732
##                  Recall : 0.5591
##                      F1 : 0.5125
##              Prevalence : 0.1854
##          Detection Rate : 0.1036
##    Detection Prevalence : 0.2190
##       Balanced Accuracy : 0.7087
##
##        'Positive' Class : 1
##
```

```r
#Saves the accuracy, precision, and recall values.
rf_accuracy = accuracy(test_set$Default, predict_rf)
rf_precision = cm_rf$byClass['Precision']
rf_recall = cm_rf$byClass['Recall']
rf_pos_precision = cm_rf$byClass['Pos Pred Value']
#Prints the accuracy, total precision, recall, and default precision values.
print(paste("Accuracy: ", round(rf_accuracy,3)))
```

```
## [1] "Accuracy:  0.803"
```

```r
print(paste("Precision: ", round(rf_precision,3)))
```

```
## [1] "Precision:  0.473"
```

```r
print(paste("Recall: ", round(rf_recall,3)))
```

```
## [1] "Recall:  0.559"
```

```r
print(paste("Default Precision: ", round(rf_pos_precision,3)))
```

```
## [1] "Default Precision:  0.473"
```

From the input printed and the plot provided, it is seen that the pay amount and bill amount in September, as well as limit balance are important variables in determining credit card default. It can also be argued age, bill amount in August and bill amount in July are important variables in determining credit card default.

Looking at the confusion matrix, the random forest model's accuracy was 81.1%, precision was 44.6%, and recall was 59.0%. The precision for predicting actual default cases correctly was 44.6%. Overall, the random forest model was slightly better in its accuracy and recall. However, the logistic regression model performed better than the random forest model when predicting actual default cases.

Conclusion

In this project, logistic regression and random forest models were created to predict if an individual would default on their credit card.

First, the data was cleaned for accuracy and manipulated to view distributions and trends in the data. From the tables and plots created, the data had more females than males and was skewed in age, with participants below the age of 40 much more prevalent than participants over the age of 40.

Next, prediction models were created to predict an individual's chances of defaulting on their credit card. The first model used was a logistic regression model. This model was used to predict if an individual would default on their credit card based on their information. This model is great for predicting a Yes/No classification for individuals. From the model created, three individuals were created with their unique information. In the example above, all three individuals created had a good chance of not defaulting on their credit card.

A random forest model was also created to determine the most important variables in a prediction model, as well as to see the accuracy of the created model. From the results, the random forest model could accurately predict someone not defaulting on their credit card, but had a more difficult time accurately predicting when someone would default on their credit card.

When comparing the two models, the following table was created:

```
set.caption("Performance for Logistic Regression and Random Forest Models")
data.table = rbind(c(log_accuracy_opt, log_precision_opt, log_recall_opt, log_pos_precision_opt), c(rf_a
colnames(data.table) = c("Accuracy", "Precision", "Recall", "Default Precision")
rownames(data.table) = c("Logistic Regression", "Random Forest")

pander(data.table)
```

Table 1: Performance for Logistic Regression and Random Forest Models

	Accuracy	Precision	Recall	Default Precision
Logistic Regression	0.7999	0.5049	0.5469	0.8827
Random Forest	0.8029	0.4732	0.5591	0.4732

Overall, it seems the logistic regression model has a higher precision and default precision rate than the random forest model, but does worse than the random forest model in accuracy and recall. This means the logistic regression model has less false positives than the random forest model, but also has more false negatives. Though the accuracy seems fairly high for the random forest prediction model, I am concerned with the false positive and false negative rates and the low default precision percentage.

Though these prediction models are acceptable, there is room for improvement, particularly in accurately predicting client that will default. I believe adding certain variables such as credit score, credit age, and credit card utilization can help improve the prediction models.

Thank you for viewing my project.

END