German Credit

Keaton Spiller, Winter 2022, STAT 387

2/24/2022

Report

Conceptual

a. Perform an exploratory analysis of data.

```
[1]  Default : 0 (no) and 1 (yes)                                     (qualitative)

[2]  checkingstatus1: Status of existing checking account             (qualitative)

[3]  duration: Duration in month                                      (numerical)

[4]  history: Credit history                                          (qualitative)

[5]  purpose: 1 of 10 things                                          (qualitative)

[6]  amount: Credit amount                                            (numerical)

[7]  savings:   Savings account/bonds                                 (qualitative)

[8]  employ: Present employment since                                 (qualitative)

[9]  installment: Installment rate in percentage of disposable income  (numerical)

[10] status: Personal status and sex                                  (qualitative)

[11] others: Other debtors / guarantors                               (qualitative)

[12] residence: Present residence since                               (numerical)

[13] property: Type of Property                                       (qualitative)

[14] age: Age in years                                                (numerical)

[15] otherplans: Other installment plans                              (qualitative)

[16] housing: 3 types of housing                                      (qualitative)

[17] cards: Number of existing credits at this bank                   (numerical)

[18] job: type of job                                                 (qualitative)

[19] liable: Number of people being liable to provide maintenance for  (numerical)

[20] tele: none or yes                                                (qualitative)

[21] foreign: yes or no                                               (qualitative)

Qualitative {14 categorical}: Default, checkingstatus1, history, purpose, savings, employ, status, others, proper
ty, otherplans, housing, job, tele, foreign

Numeric {7 Numerical}: duration, amount, installment, residence, age, cards, liable
```

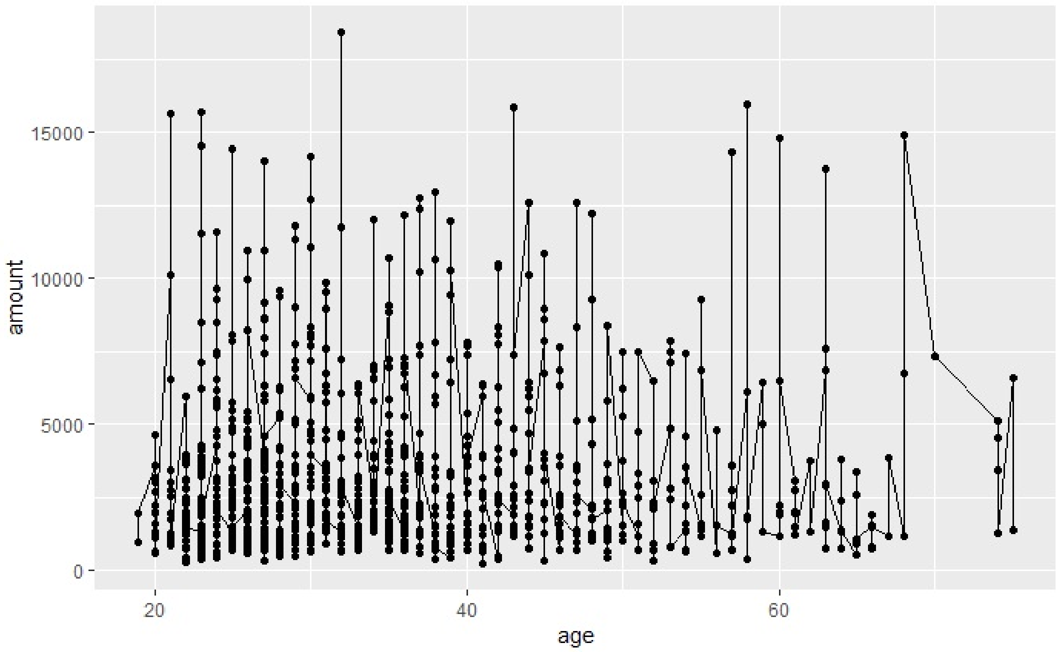
```{r}
str(german_raw)
```

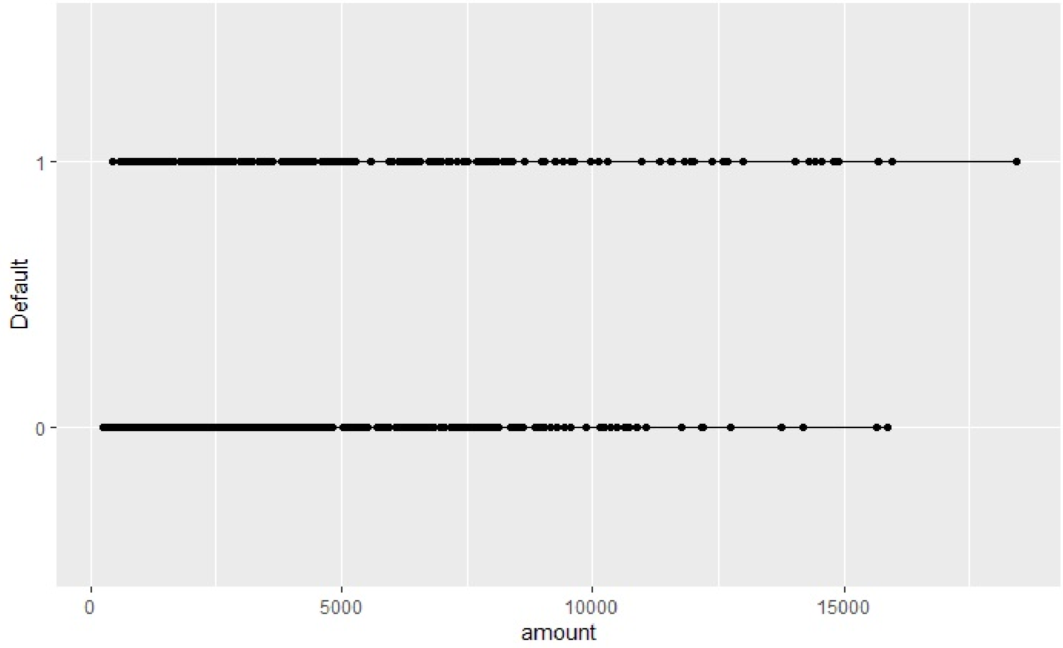
```
'data.frame':	1000 obs. of  21 variables:
 $ Default        : int  0 1 0 0 1 0 0 0 0 1 ...
 $ checkingstatus1: chr  "A11" "A12" "A14" "A11" ...
 $ duration       : int  6 48 12 42 24 36 24 36 12 30 ...
 $ history        : chr  "A34" "A32" "A34" "A32" ...
 $ purpose        : chr  "A43" "A43" "A46" "A42" ...
 $ amount         : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ savings        : chr  "A65" "A61" "A61" "A61" ...
 $ employ         : chr  "A75" "A73" "A74" "A74" ...
 $ installment    : int  4 2 2 2 3 2 3 2 2 4 ...
 $ status         : chr  "A93" "A92" "A93" "A93" ...
 $ others         : chr  "A101" "A101" "A101" "A103" ...
 $ residence      : int  4 2 3 4 4 4 4 2 4 2 ...
 $ property       : chr  "A121" "A121" "A121" "A122" ...
 $ age            : int  67 22 49 45 53 35 53 35 61 28 ...
 $ otherplans     : chr  "A143" "A143" "A143" "A143" ...
 $ housing        : chr  "A152" "A152" "A152" "A153" ...
 $ cards          : int  2 1 1 1 2 1 1 1 1 2 ...
 $ job            : chr  "A173" "A173" "A172" "A173" ...
 $ liable         : int  1 1 2 2 2 2 1 1 1 1 ...
 $ tele           : chr  "A192" "A191" "A191" "A191" ...
 $ foreign        : chr  "A201" "A201" "A201" "A201" ...
```

Placing the qualitative variables into factor form with as.factor()

```
'data.frame':	1000 obs. of  21 variables:
 $ Default        : Factor w/ 2 levels "0","1": 1 2 1 1 2 1 1 1 1 2 ...
 $ checkingstatus1: Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...
 $ duration       : int  6 48 12 42 24 36 24 36 12 30 ...
 $ history        : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...
 $ purpose        : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...
 $ amount         : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ savings        : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...
 $ employ         : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...
 $ installment    : int  4 2 2 2 3 2 3 2 2 4 ...
 $ status         : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...
 $ others         : Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...
 $ residence      : int  4 2 3 4 4 4 4 2 4 2 ...
 $ property       : Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...
 $ age            : int  67 22 49 45 53 35 53 35 61 28 ...
 $ otherplans     : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ housing        : Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...
 $ cards          : int  2 1 1 1 2 1 1 1 1 2 ...
 $ job            : Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...
 $ liable         : int  1 1 2 2 2 2 1 1 1 1 ...
 $ tele           : Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...
 $ foreign        : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
```

Interesting plots of variable exploration





b. Build a reasonably “good” logistic regression model for these data. There is no need to explore interactions. Carefully justify all the choices you make in building the model.

```
full model without training data or variable reduction

AIC = 993.82


Train = amount < 5000

Test = amount >= 5000

Train = 812 rows 21 variables

Test = 188 rows 21 variables


full logistic Model with training data without variable reduction tested on test data

AIC = 776.1


Variable reduction using Step AIC and BIC methods (forwards and backwards)

Both methods resulting in the same variables and AIC/BIC values

Reduced logistic Model with training data tested on test data

AIC = 758.3341

BIC = 694.3341
```

```
Default ~ checkingstatus1 + duration + history + purpose + savings +
    installment + others + otherplans + housing + tele + foreign

                  Df Deviance    AIC
<none>                 700.44 794.82
- otherplans       2   707.33 795.62
- others           2   708.01 796.30
- housing          2   708.02 796.31
+ amount           1   699.14 796.57
+ residence        1   699.48 796.90
+ age              1   700.08 797.50
+ status           3   694.15 797.67
+ liable           1   700.37 797.79
+ cards            1   700.43 797.86
- savings          4   715.91 798.11
- installment      1   706.89 798.23
- tele             1   707.63 798.96
+ employ           4   693.08 799.64
- foreign          1   708.84 800.17
- history          4   719.50 801.70
+ property         3   699.34 802.85
+ job              3   700.03 803.54
- purpose          9   739.45 806.43
- duration         1   721.39 812.72
- checkingstatus1  3   758.83 844.07
```

```
AIC went from 776.1 to 758.3341, which is slighly smaller, but acceptable for the simplified model with fewer var
iables


final variables : checkingstatus1 + duration + history + purpose + savings + installment + others + age + otherpl
ans + housing + tele + foreign

The accuracy was also slightly smaller

0.6489362 to 0.6382979 accuracy

0.3510638 to 0.3617021 error

However Since the reduced model is 12 variables, and the full model is 20 variables (not including Default),

the smaller model will be significantly better for explainability, and cut away unnecesary/noisy variables.
```

c. Write the final model in equation form. Provide a summary of estimates of the regression coefficients, the standard errors of the estimates, and 95% confidence intervals of the coefficients. Interpret the estimated coefficients of at least two predictors. Provide training error rate for the model.

```
Page 135 section {4.3}
```

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \tag{4.6}$$

where $X = (X_1, \ldots, X_p)$ are p predictors. Equation 4.6 can be rewritten as

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}. \tag{4.7}$$

```
X is a matrix with 12 variables checkingstatus1 + duration + history + purpose + savings + installment + others +
age + otherplans + housing + tele + foreign

logistic model Summary of regression coefficients and standard errors of the estimates
```

```
Call:
glm(formula = Default ~ checkingstatus1 + duration + history + 
    purpose + savings + installment + others + age + otherplans + 
    housing + tele + foreign, family = binomial(link = "logit"), 
    data = train.x)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.0153  -0.6666  -0.3589   0.6118   2.9207  

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)    
(Intercept)         2.223174   0.754540   2.946 0.003215 ** 
checkingstatus1A12 -0.342305   0.245912  -1.392 0.163928    
checkingstatus1A13 -0.800394   0.375664  -2.131 0.033121 *  
checkingstatus1A14 -1.798596   0.265598  -6.772 1.27e-11 ***
duration            0.046140   0.010674   4.323 1.54e-05 ***
historyA31         -0.904105   0.653271  -1.384 0.166369    
historyA32         -1.212759   0.521555  -2.325 0.020057 *  
historyA33         -0.866613   0.594350  -1.458 0.144817    
historyA34         -1.830004   0.547500  -3.342 0.000830 ***
purposeA41         -2.687781   0.730715  -3.678 0.000235 ***
purposeA410        -1.145843   1.238546  -0.925 0.354887    
purposeA42         -0.851999   0.281209  -3.030 0.002447 ** 
purposeA43         -1.019851   0.268370  -3.800 0.000145 ***
purposeA44         -0.480523   0.737343  -0.652 0.514598    
purposeA45         -0.146351   0.615545  -0.238 0.812069    
purposeA46          0.537021   0.453834   1.183 0.236691    
purposeA48         -2.046419   1.185192  -1.727 0.084229 .  
purposeA49         -0.824451   0.380196  -2.168 0.030121 *  
savingsA62         -0.286769   0.318660  -0.900 0.368161    
savingsA63         -0.160004   0.409342  -0.391 0.695885    
savingsA64         -1.214894   0.553162  -2.196 0.028072 *  
savingsA65         -0.971837   0.326588  -2.976 0.002923 ** 
installment         0.252804   0.095521   2.647 0.008131 ** 
othersA102          0.578557   0.456521   1.267 0.205041    
othersA103         -1.025950   0.462817  -2.217 0.026640 *  
age                -0.023993   0.009897  -2.424 0.015344 *  
otherplansA142     -0.050144   0.473732  -0.106 0.915702    
otherplansA143     -0.682298   0.268874  -2.538 0.011161 *  
housingA152        -0.561036   0.244794  -2.292 0.021913 *  
housingA153        -0.107641   0.430129  -0.250 0.802393    
teleA192           -0.497699   0.215140  -2.313 0.020702 *  
foreignA202        -1.911985   0.794674  -2.406 0.016128 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 952.66  on 811  degrees of freedom
Residual deviance: 694.33  on 780  degrees of freedom
AIC: 758.33

Number of Fisher Scoring iterations: 6
```

```
95% confidence intervals
```

```
Waiting for profiling to be done...
                          2.5 %       97.5 %
(Intercept)          0.76690967  3.732323214
checkingstatus1A12 -0.82620164  0.139126537
checkingstatus1A13 -1.56235430 -0.081829871
checkingstatus1A14 -2.33104920 -1.287681430
duration             0.02541300  0.067338198
historyA31          -2.20648051  0.366182780
historyA32          -2.27224867 -0.213437850
historyA33          -2.06287955  0.278511771
historyA34          -2.93763929 -0.778995290
purposeA41          -4.34852555 -1.417443200
purposeA410         -4.24552868  1.096289795
purposeA42          -1.40960737 -0.305608664
purposeA43          -1.55173293 -0.498023755
purposeA44          -2.00248162  0.943613683
purposeA45          -1.39161911  1.045493028
purposeA46          -0.35685333  1.428749925
purposeA48          -5.09347862 -0.005435417
purposeA49          -1.58634096 -0.091440263
savingsA62          -0.92536350  0.327716269
savingsA63          -1.00809447  0.610565908
savingsA64          -2.41567636 -0.210571862
savingsA65          -1.63569513 -0.351096078
installment          0.06766494  0.442676120
othersA102          -0.32899681  1.474183156
othersA103          -1.99376260 -0.162295934
age                 -0.04376757 -0.004892125
otherplansA142      -0.98756897  0.876783918
otherplansA143      -1.20845784 -0.152165711
housingA152         -1.04106804 -0.079885695
housingA153         -0.95503424  0.735158331
teleA192            -0.92454142 -0.079845981
foreignA202         -3.81662615 -0.554535730
```

```
glm.pred3   0   1
        0 416 153
        1 174  69
```

```
training error rate for the final logistic model = 0.3617021
```

d. Fit a KNN with K chosen optimally using test error rate. Report error rate, sensitivity, specificity, and AUC for the optimal KNN based on the training data. Also, report its estimated test error rate.

```
K optimal test error rate = 11

train error rate = 0.2413793

train sensitivity = 0.9525424

train Specificity = 0.2432432

train AUC = 0.7828218

test error rate = 0.3829787
```

e. Repeat (d) using LDA.

```
train error rate = 0.4248768

train sensitivity = 0.9135593

train Specificity = 0.5135135

train AUC = 0.8352497

test error rate = 0.3244681
```

f. Repeat (d) using QDA.

```
train error rate = 0.4889163

train sensitivity = 0.7661017

train Specificity = 0.8063063

train AUC = 0.8477248

test error rate = 0.393617
```

g. Compare the results in (b), (d)-(f). Which classifier would you recommend? Justify your answer.

Results training on train data and testing on train data

	Model	Accuracy	Error	Sensitivity	specificity	AUC
1	Logistic_full	0.6200000	0.3800000	0.7985714	0.2033333	0.5009524
2	Logistic_reduced	0.5972906	0.4027094	0.7050847	0.3108108	0.5079478
3	KNN	0.7586207	0.2413793	0.9525424	0.2432432	0.7828218
4	LDA	0.5751232	0.4248768	0.9135593	0.5135135	0.8352497
5	QDA	0.5110837	0.4889163	0.7661017	0.8063063	0.8477248

Results training on train data and testing on test data

	Model	Accuracy	Error	Sensitivity	specificity	AUC
1	Logistic_full	0.6489362	0.3510638	0.8818182	0.3205128	0.6977855
2	Logistic_reduced	0.6382979	0.3617021	0.7909091	0.4230769	0.7057110
3	KNN	0.6170213	0.3829787	0.7909091	0.3717949	0.4050117
4	LDA	0.6755319	0.3244681	0.8000000	0.5000000	0.7168998
5	QDA	0.6063830	0.3936170	0.6181818	0.5897436	0.6618881

```
Based on the results I would recommend LDA, with the highest accuracy ~ 0.67 and lowest error rate ~ 0.32 on the
test data.
Although KNN is the highest tested on train data, it doesn't do a good job accounting for unknown data (test data
) and it's not the best model to choose.

Therefore LDA is the best model for predicting Default given the reduced variable selection
```

Additional notes

Cross Validation for model selection?

```
LOOCV:
more Bias,less Variance

K-Fold:
less Bias, > Variance
```

Ridge or Lasso For variable selection?

```
Ridge:

L2 Selection

Small coefficients

Lasso:

L1 Selection

Large coefficients (Sparcity)
```

chi square test of association is more accurate for logistic regression variable selection (Not Taught in this class)

```
anova(chi square test)
```

R Code

```
rm(list=ls())
```

```r
set.seed(1)
library(bookdown)# load the libraries
library(dplyr)
library(broom)
library(faraway)
library(ellipse)
library(rstudioapi)
library(lmtest)
library(simex)
library(ggplot2)
library(lars)
library(MASS)
library(pls)
library(olsrr)
library(leaps)
library(matlib)
library(olsrr)
library(ggplot2)
library(lattice)
library(class)
library(MASS)
library(ISLR2)
library(boot)
library(ISLR2)
library(class)
library(e1071)
library(gam)
library(glmnet)
library(Amelia)
library(caret)
library(pROC)
library(ROCR)
```

a. exploratory analysis

```r
german_raw <-  read.csv(file = "germancredit.csv", header = TRUE)
german <- german_raw
head(german_raw,5)
```

```r
missmap(german, main = "Missing values vs observed")
```

```r
any(is.na(german_raw))
```

```r
str(german_raw)
```

```r
german$Default <- as.factor(german_raw$Default)
german$checkingstatus1 <- as.factor(german_raw$checkingstatus1)
german$history <- as.factor(german_raw$history)
german$purpose <- as.factor(german_raw$purpose)
german$savings <- as.factor(german_raw$savings)
german$employ <- as.factor(german_raw$employ)
german$status <- as.factor(german_raw$status)
german$others <- as.factor(german_raw$others)
german$property <- as.factor(german_raw$property)
german$otherplans <- as.factor(german_raw$otherplans)
german$housing <- as.factor(german_raw$housing)
german$job <- as.factor(german_raw$job)
german$tele <- as.factor(german_raw$tele)
german$foreign <- as.factor(german_raw$foreign)
str(german)
```

```r
plot(german[1:10])
```

```r
plot(german[11:21])
```

```r
plot(german[1:5])
```

```r
plot(german[6:10])
```

```r
plot(german[11:15])
```

```
plot(german[16:21])
```

```
colnames <-  colnames(german)
colnames
```

```
ggplot(data=german, aes(x=Default, y=amount), x=Default, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=checkingstatus1, y=amount), x=checkingstatus1, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=duration, y=amount), x=duration, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=history, y=amount), x=history, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=purpose, y=amount), x=purpose, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=savings, y=amount), x=savings, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=employ, y=amount), x=employ, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=installment, y=amount), x=installment, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=status, y=amount), x=status, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=others, y=amount), x=others, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=residence, y=amount), x=residence, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=property, y=amount), x=property, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=age, y=amount), x=age, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=otherplans, y=amount), x=otherplans, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=housing, y=amount), x=housing, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=cards, y=amount), x=cards, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=job, y=amount), x=job, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=liable, y=amount), x=liable, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=tele, y=amount), x=tele, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=foreign, y=amount), x=foreign, y=amount)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=Default, y=age), x=Default, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=checkingstatus1, y=age), x=checkingstatus1, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=duration, y=age), x=duration, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=history, y=age), x=history, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=purpose, y=age), x=purpose, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=savings, y=age), x=savings, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=employ, y=age), x=employ, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=installment, y=age), x=installment, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=status, y=age), x=status, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=others, y=age), x=others, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=residence, y=age), x=residence, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=property, y=age), x=property, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=amount, y=age), x=age, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=otherplans, y=age), x=otherplans, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=housing, y=age), x=housing, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=cards, y=age), x=cards, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=job, y=age), x=job, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=liable, y=age), x=liable, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=tele, y=age), x=tele, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=foreign, y=age), x=foreign, y=age)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=Default, y=Default), x=Default, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=checkingstatus1, y=Default), x=checkingstatus1, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=duration, y=Default), x=duration, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=history, y=Default), x=history, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=purpose, y=Default), x=purpose, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=savings, y=Default), x=savings, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=employ, y=Default), x=employ, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=installment, y=Default), x=installment, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=status, y=Default), x=status, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=others, y=Default), x=others, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=residence, y=Default), x=residence, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=property, y=Default), x=property, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=amount, y=Default), x=Default, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=otherplans, y=Default), x=otherplans, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=housing, y=Default), x=housing, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=cards, y=Default), x=cards, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=job, y=Default), x=job, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=liable, y=Default), x=liable, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=tele, y=Default), x=tele, y=Default)+
  geom_line()+
  geom_point()
```

```
ggplot(data=german, aes(x=foreign, y=Default), x=foreign, y=Default)+
  geom_line()+
  geom_point()
```

b.

```
german_full <- german # In Case of doing modifications on the german data

train <- ( german_full$amount < 5000)

test.y <-  german_full$Default[!train ]
train.x <-  german_full[train,]
test.x <-  german_full[!train,]

print(dim(train.x))
print(dim(test.x))
german_full
```

```
# logistic Model on full data without variable reduction
glm.full <- glm(Default ~ ., data=german_full, family = "binomial"(link = "logit"))
summary(glm.full)
```

```
# logistic Model on the training data without variable reduction
glm.fits <- glm(Default ~ ., data=train.x, family = "binomial"(link = "logit"))
glm.fits
(summary(glm.fits))
par(mfrow = c(2,2))
plot(glm.fits)
```

```
glm.probs <- predict( glm.fits, newdata=test.x,
type = "response")
```

```
glm.pred <- rep (0, length(test.y))
glm.pred [ glm.probs > .5] <- 1
(confusion_matrix <- table ( glm.pred, test.y))
(Accuracy_glm <- mean ( glm.pred == test.y))
(error_glm <- mean ( glm.pred != test.y))

(specificity_glm <- specificity(confusion_matrix))
(sensitivity_glm <- sensitivity(confusion_matrix))

(glm_auc <- auc(test.y, glm.probs))

plot(x = test.x$amount, y = glm.pred)
```

```
plot(x = test.x$amount, y = glm.probs)
```

```
glm.pred_train_full <- rep (0, length(german_full$Default))
glm.pred_train_full [ glm.probs > .5] <- 1


(confusion_matrix_train_full <- table ( glm.pred_train_full, german_full$Default))
(accuracy_glm_train_full <- mean ( glm.pred_train_full == german_full$Default))
(error_glm_train_full <- mean ( glm.pred_train_full != german_full$Default))
(specificity_glm_train_full <- specificity(confusion_matrix_train_full))
(sensitivity_glm_train_full <- sensitivity(confusion_matrix_train_full))

(glm_auc_train_full <- auc(german_full$Default, glm.pred_train_full))
```

```
glmAIC <- step(object=glm.fits, direction="both",criterion="AIC", trace=FALSE)
glmAIC$deviance
AIC(glmAIC)
BIC(glmAIC)
```

```
n <- length(train.x)
glmBIC <- step(object=glm.fits, direction="both",criterion="BIC",k=log(n), trace=FALSE)
glmBIC$deviance
AIC(glmBIC)
BIC(glmBIC)
```

```
glm.fits2 <- glm(Default ~ checkingstatus1 + duration + history + purpose + savings +
                   installment + others + age + otherplans + housing + tele + foreign,
                 data=train.x, family = "binomial"(link = "logit"))
glm.fits2
glms2 <- summary(glm.fits2)
par(mfrow = c(2,2))
plot(glm.fits2)
```

```
glm.probs2 <- predict( glm.fits2, newdata=test.x,
type = "response")
```

```
glm.pred2 <- rep (0, length(test.y))
glm.pred2 [ glm.probs2 > .5] <- 1

(confusion_matrix2 <- table ( glm.pred2, test.y))
(accuracy_glm2 <- mean ( glm.pred2 == test.y))
(error_glm2 <- mean ( glm.pred2 != test.y))
(specificity_glm2 <- specificity(confusion_matrix2))
(sensitivity_glm2 <- sensitivity(confusion_matrix2))

plot(x = test.x$amount, y = glm.pred2)
```

```
plot(x = test.x$amount, y = glm.probs2)
```

```
(glm_auc2 <- auc(test.y, glm.probs2))
```

c.

```
summary(glm.fits2)
```

```
confint(glm.fits2)
```

```
glm.pred_train <- rep (0, length(train.x$Default))
glm.pred_train [ glm.probs2 > .5] <- 1

(confusion_matrix_train <- table ( glm.pred_train, train.x$Default))
(accuracy_glm_train <- mean ( glm.pred_train == train.x$Default))
(error_glm_train <- mean ( glm.pred_train != train.x$Default))
(specificity_glm_train <- specificity(confusion_matrix_train))
(sensitivity_glm_train <- sensitivity(confusion_matrix_train))

(glm_auc_train <- auc(train.x$Default, glm.pred_train))
```

d.

```
# KNN training and test on test data
set.seed(1)
train <- ( german_full$amount < 5000)
train.x <- cbind (german_full$checkingstatus1, german_full$duration, german_full$history, german_full$purpose, ge
rman_full$savings,
                    german_full$installment, german_full$others, german_full$age, german_full$otherplans,
                  german_full$housing, german_full$tele, german_full$foreign)[train , ]

test.x <- cbind (german_full$checkingstatus1, german_full$duration, german_full$history, german_full$purpose, ger
man_full$savings,
                    german_full$installment, german_full$others, german_full$age, german_full$otherplans,
                  german_full$housing, german_full$tele, german_full$foreign)[! train , ]

train.Default <- german_full$Default[ train ]

test.Default <- german_full$Default[!train ]

knn.pred <- knn(train = train.x,
                test = test.x,
                cl = train.Default,
                k = 11)
knn.prob <- knn(train = train.x,
                test = test.x,
                cl = train.Default,
                k = 11,
                prob=TRUE)

knn_confusion_matrix <-  table (knn.pred, test.Default)
(knn_accuracy <- mean ( knn.pred == test.Default))
(knn_error <- mean ( knn.pred != test.Default))
(knn_Specificity <- specificity(knn_confusion_matrix))
# true negative rate.
(knn_sensitivity <- sensitivity(knn_confusion_matrix))
# true positive rate.

scores.knn <- attr(knn.prob,"prob")
knn_roc <- roc(test.Default, scores.knn)
knn_AUC <- knn_roc$auc

print(knn_AUC)
pred_knn <- prediction(scores.knn, test.Default)
pred_knn <- performance(pred_knn, "tpr", "fpr")
plot(pred_knn, avg= "threshold", colorize=T, lwd=3, main="ROC")
abline(a = 0, b = 1)
```

```
# KNN training and test on train data
knn.pred_train <- knn(train = train.x,
                test = train.x,
                cl = train.Default,
                k = 11)
knn.prob_train <- knn(train = train.x,
                test = train.x,
                cl = train.Default,
                k = 11,
                prob=TRUE)
knn_confusion_matrix_train <-  table (knn.pred_train, train.Default)
(knn_accuracy_train <- mean ( knn.pred_train == train.Default))
(knn_error_train <- mean ( knn.pred_train != train.Default))
(knn_Specificity_train <- specificity(knn_confusion_matrix_train))
# true negative rate.
(knn_sensitivity_train <- sensitivity(knn_confusion_matrix_train))
# true positive rate.

scores.knn_train <- attr(knn.prob_train,"prob")
knn_roc_train <- roc(train.Default, scores.knn_train)
knn_AUC_train <- knn_roc_train$auc

print(knn_AUC_train)
pred_knn_train <- prediction(scores.knn_train, train.Default)
pred_knn_train <- performance(pred_knn_train, "tpr", "fpr")
plot(pred_knn_train, avg= "threshold", colorize=T, lwd=3, main="ROC")
abline(a = 0, b = 1)
```

e.

```
# LDA training
lda.fit <- lda(
Default ~ checkingstatus1 + duration + history + purpose + savings +
  installment + others + age + otherplans + housing + tele + foreign,
data=german_full,
subset = train
)

plot( lda.fit )
```

```
lda.fit
```

```
# LDA test prediction
lda.pred <- predict(lda.fit , german_full[!train, ])
lda.prob <- predict(lda.fit , german_full[!train, ], type="response")
names ( lda.pred )
lda.class <- lda.pred$class

lda_confusion_matrix <- table ( lda.class, test.Default)

(lda_accuracy <- mean ( lda.class == test.Default))
(lda_error <- mean ( lda.class != test.Default))
(lda_specificity <- specificity(lda_confusion_matrix))
(lda_sensitivity <- sensitivity(lda_confusion_matrix))

pred <- prediction(lda.pred$posterior[,2], test.Default)
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE)
abline(a = 0, b = 1)
```

```
auc_lda <- performance(pred, measure = "auc")
auc_lda <- auc_lda@y.values[[1]]
print(auc_lda)
```

```
# LDA train prediction
lda.pred_train <- predict(lda.fit , german_full[train, ])
lda.prob_train <- predict(lda.fit , german_full[train, ], type="response")

lda.class_train <- lda.pred_train$class

lda_confusion_matrix_train <- table ( lda.class_train, train.Default)

(lda_accuracy_train <- mean ( lda.class == train.Default))
(lda_error_train <- mean ( lda.class != train.Default))
(lda_specificity_train <- specificity(lda_confusion_matrix_train))
(lda_sensitivity_train <- sensitivity(lda_confusion_matrix_train))

lda_pred_train <- prediction(lda.pred_train$posterior[,2], train.Default)
lda_perf_train <- performance(lda_pred_train,"tpr","fpr")
plot(lda_perf_train,colorize=TRUE)
abline(a = 0, b = 1)
```

```
auc_lda_train <- performance(lda_pred_train, measure = "auc")
auc_lda_train <- auc_lda_train@y.values[[1]]
print(auc_lda_train)
```

f.

```
# QDA Training
qda.fit <- qda(
Default ~ checkingstatus1 + duration + history + purpose + savings +
  installment + others + age + otherplans + housing + tele + foreign,
data=german_full,
subset = train
)
qda.fit
```

```r
# QDA test prediction
qda.pred <- predict(qda.fit , german_full[!train, ])
names ( qda.pred )
qda.class <- qda.pred$class

qda_confusion_matrix <- table ( qda.class, test.Default)

(qda_accuracy <- mean ( qda.class == test.Default))
(qda_error <- mean ( qda.class != test.Default))
(qda_specificity <- specificity(qda_confusion_matrix))
(qda_sensitivity <- sensitivity(qda_confusion_matrix))

pred <- prediction(qda.pred$posterior[,2], test.Default)
perf <- performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE)
abline(a = 0, b = 1)
```

```r
auc_qda <- performance(pred, measure = "auc")
auc_qda <- auc_qda@y.values[[1]]
print(auc_qda)
```

```r
# QDA train prediction
qda.pred_train <- predict(qda.fit , german_full[train, ])
qda.prob_train <- predict(qda.fit , german_full[train, ], type="response")

qda.class_train <- qda.pred_train$class

qda_confusion_matrix_train <- table ( qda.class_train, train.Default)

(qda_accuracy_train <- mean ( qda.class == train.Default))
(qda_error_train <- mean ( qda.class != train.Default))
(qda_specificity_train <- specificity(qda_confusion_matrix_train))
(qda_sensitivity_train <- sensitivity(qda_confusion_matrix_train))

qda_pred_train <- prediction(qda.pred_train$posterior[,2], train.Default)
qda_perf_train <- performance(qda_pred_train,"tpr","fpr")
plot(qda_perf_train,colorize=TRUE)
abline(a = 0, b = 1)
```

```r
auc_qda_train <- performance(qda_pred_train, measure = "auc")
auc_qda_train <- auc_qda_train@y.values[[1]]
print(auc_qda_train)
```

g.

```r
train_results <- data.frame(
  Model = c("Logistic_full", "Logistic_reduced", "KNN", "LDA", "QDA"),
   Accuracy = c(accuracy_glm_train_full,accuracy_glm_train,knn_accuracy_train, lda_accuracy_train, qda_accuracy_t
rain) ,
   Error = c(error_glm_train_full,error_glm_train,knn_error_train,lda_error_train,qda_error_train),
   Sensitivity = c(sensitivity_glm_train_full,sensitivity_glm_train,knn_sensitivity_train,lda_sensitivity_train,q
da_sensitivity_train),
     specificity =c(specificity_glm_train_full,specificity_glm_train,knn_Specificity_train,lda_specificity_train,
qda_specificity_train),
     AUC = c(glm_auc_train_full,glm_auc_train,knn_AUC_train,auc_lda_train,auc_qda_train)
)
View(train_results)
```

```r
test_results <- data.frame(
  Model = c("Logistic_full", "Logistic_reduced", "KNN", "LDA", "QDA"),
   Accuracy = c(Accuracy_glm,accuracy_glm2,knn_accuracy, lda_accuracy, qda_accuracy) ,
   Error = c(error_glm,error_glm2,knn_error,lda_error,qda_error),
   Sensitivity = c(sensitivity_glm,sensitivity_glm2,knn_sensitivity,lda_sensitivity,qda_sensitivity),
     specificity =c(specificity_glm,specificity_glm2,knn_Specificity,lda_specificity,qda_specificity),
     AUC = c(glm_auc,glm_auc2,knn_AUC,auc_lda,auc_qda)
)
View(test_results)
```