

Tarea 1

Estudiante

**John Daniel hoyos Arias
Ivan Santiago Rojas Martinez
Genaro Alfonso Aristizabal Echeverri**

Docente

Juan Carlos Salazar Uribe

Asignatura

Analitica de datos



Sede Medellín
17 de septiembre del 2022

Índice

1. Ejercicio1	4
1.1. Análisis descriptivo de la base de datos bank	4
1.2. a) cree un conjunto de datos de entrenamiento del 75 % y el restante 25 % tratetelo como datos de test o prueba.	6
1.3. b)Con los datos de entrenamiento implemente naive bayes usando loan como el supervisor y las demás como predictores.	6
1.4. c) Con los datos de entrenamiento, Implemente un modelo Knn con loan como supervisor y las demás como predictoras. utilizar varios valores de k, pero reportar solo uno.	7
1.5. d)Con los datos de entrenamiento, implemente un modelo Logístico con loan como supervisor y las demás como predictoras.	9
1.5.1. Planteamiento del Modelo.	9
1.6. e) Con los datos de entrenamiento, implemente un modelo LDA con loan como supervisor y las demás como predictoras. En LDA se modela la distribución de los predictores X de manera separada en cada una de las categorías de respuesta (es decir, condicionando en Y) y luego se usa el teorema de Bayes para obtener estimaciones de $Pr(Y = k X = x)$ Cuando estas distribuciones se asumen normales, el modelo es muy similar en forma al de regresión logística. esta probabilidad de clasificación esta dada por:	10
1.7. f) Con los datos de entrenamiento calcular training MSE, matriz confusión y curva roc para cada uno de los modelos.	10
1.8. g) Con los datos de test calcular training MSE, matriz confusión y curva roc para cada uno de los modelos.	12
1.9. h) Con cual modelo observo mejor desempeño y por qué?	14
2. Ejercicio2	14
2.1. a) Datos de entrenamiento y de prueba	14
2.2. b)Implementación de KNN	15
2.3. Con los datos de entrenamiento, implemente regresión lineal simple usando income como el supervisor y debts como predictor. Grafique e interprete. . .	15
3. Ejercicio3	17
4. Ejercicio4	17

Índice de figuras

1. Ejercicio1

```
## The following object is masked from package:MASS:
##
##     housing
```

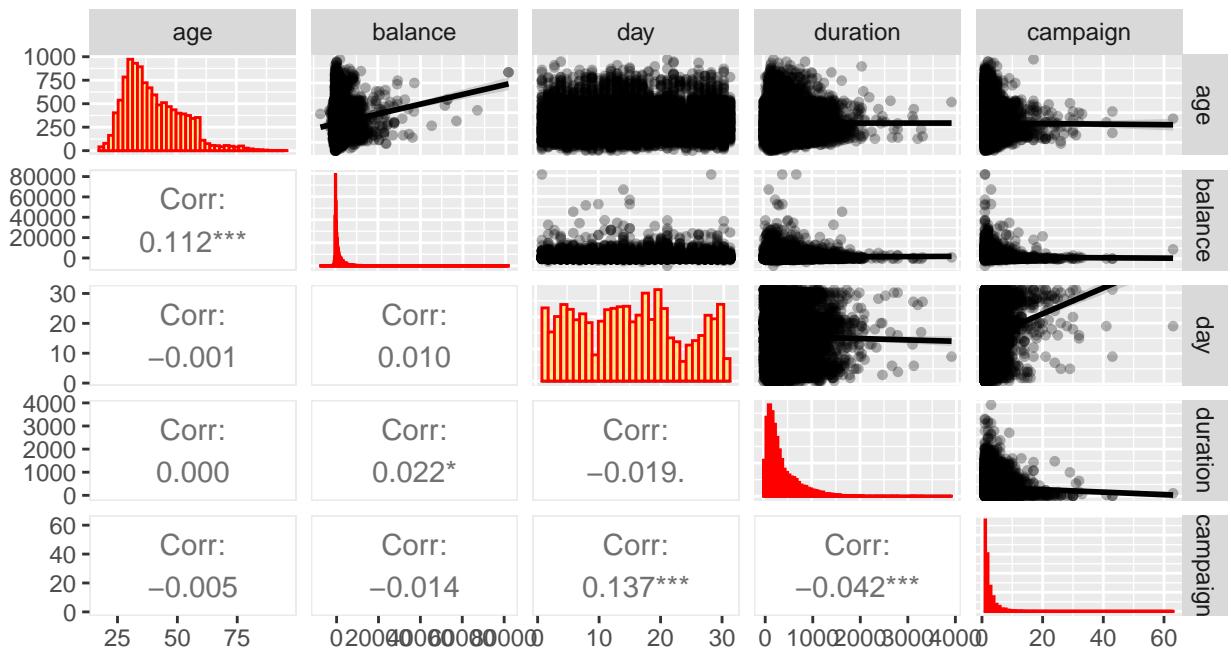
1.1. Análisis descriptivo de la base de datos bank

La base de datos bank, contiene un total de 17 variables y 11.162 observaciones, de las cuales 7 son de tipo numéricas y 10 son de tipo categórica, esta base de datos resume algunas características acerca de clientes de un banco en particular tales como la edad(age), el tipo de trabajo que desempeña(job), el estado marital(marital), nivel educativo(education), si ha cometido o no alguna falta pagos(default), el estado de sus fondos económicos(balance), si el cliente tiene o no algún préstamo de vivienda(housing), si el cliente tiene o no algún préstamo (loan), medio de contacto con el cliente(contact), fecha de afiliación(day, month), tiempo de vencimiento (El tiempo de vencimiento). entre otras, a continuación veremos un pequeño resumen de las variables.

```
##      age          job        marital       education
##  Min.   :18.00  Length:11162    Length:11162    Length:11162
##  1st Qu.:32.00  Class  :character  Class  :character  Class  :character
##  Median :39.00  Mode   :character  Mode   :character  Mode   :character
##  Mean   :41.23
##  3rd Qu.:49.00
##  Max.   :95.00
##      default        balance        housing        loan
##  Length:11162    Min.   :-6847    Length:11162    Length:11162
##  Class  :character  1st Qu.: 122    Class  :character  Class  :character
##  Mode   :character  Median : 550    Mode   :character  Mode   :character
##                      Mean   : 1529
##                      3rd Qu.: 1708
##                      Max.   :81204
##      contact         day        month       duration
##  Length:11162    Min.   : 1.00    Length:11162    Min.   : 2
##  Class  :character  1st Qu.: 8.00    Class  :character  1st Qu.: 138
##  Mode   :character  Median :15.00    Mode   :character  Median : 255
##                      Mean   :15.66
##                      3rd Qu.:22.00
##                      Max.   :31.00
##                      Max.   :3881
##      campaign        pdays       previous      poutcome
##  Min.   : 1.000  Min.   :-1.00    Min.   :0.0000  Length:11162
##  1st Qu.: 1.000  1st Qu.:-1.00  1st Qu.:0.0000  Class  :character
##  Median : 2.000  Median :-1.00  Median :0.0000  Mode   :character
##  Mean   : 2.508  Mean   :51.33   Mean   : 0.8326
```

```

##   3rd Qu.: 3.000   3rd Qu.: 20.75   3rd Qu.: 1.0000
##   Max.    :63.000   Max.    :854.00   Max.    :58.0000
##   deposit
##   Length:11162
##   Class :character
##   Mode   :character
##
## 
## 
## 
```



El gráfico anterior es importante para identificar el posible comportamiento de nuestras variables numéricas, en este caso vemos que existe poca relación lineal entre las mismas, situación que nos da una idea de pensar que es poco probable que existan problemas de multicolinealidad.



La variable **loan**, es una de las variables de mayor interés en el estudio, es una variable dicótoma que representa si un cliente tiene o no algún préstamo, del gráfico anterior, claramente no hay diferencia en mediana para la variable **loan** con respecto a la edad, día, duración y balance, por otra parte, vemos que la mayoría de personas no tienen ningún tipo de préstamo.

1.2. a) cree un conjunto de datos de entrenamiento del 75 % y el restante 25 % tratetelo como datos de test o prueba.

```
df=data.frame(bank)
smp_sz <- floor(0.75 * nrow(df))
train_ind <- sample(seq_len(nrow(df)), size = smp_sz)

train <- df[train_ind, ] #datos de Entrenamiento 75%
test <- df[-train_ind, ] #datos de Test 25%

y_train=df[train_ind,8]
y_test=df[-train_ind,8]
```

1.3. b) Con los datos de entrenamiento implemente naive bayes usando **loan** como el supervisor y las demás como predictores.

El clasificador Naive de Bayes asume que todas las features (componentes del vector x) son independientes y que son igualmente importantes. Con este supuesto, la probabilidad

(Likelihood-verosimilitud) de observar el vector de features $x = (x_1, x_2, \dots, x_p)$ con $p = 1 \dots 16$ dado que se esta en la clase j es:

$$Pr(x|Y=j) = Pr(x_1|Y=j) * \dots * Pr(x_p|Y=j)$$

Por el teorema de bayes tenemos:

$$Pr(y=j|x) = \frac{Pr(x|Y=j)(Pr(Y=j))}{Pr(x)}$$

lo anterior representa un modelo bayesiando donde, la distribucion posterior, esta representada por el producto de la verosimilitud y la probabilidad a priori.

ahora, vamos a implementar un modelo de naivebayes para calcular la probabilidad de que los proximos clientes tengan o no un credito.

```
#modelo naive bayes
naiveB.fit <- naiveBayes(loan~., data=train, laplace=0.128)

#predict train and testing
predict_test<-predict(naiveB.fit,test,type="class")
predict_train<-predict(naiveB.fit,train,type="class")

#prediccion probabilidades
predict_train2<-predict(naiveB.fit,train,type="raw")
predict_test2<-predict(naiveB.fit,test,type="raw")
```

1.4. c) Con los datos de entrenamiento, Implemente un modelo Knn con loan como supervisor y las demás como predictoras. utilizar varios valores de k, pero reportar solo uno.

El modelo knn también conocido como k vecinos más cercanos, es una metodología muy eficiente que permite a partir de un valor de k, tomar los k valores más cercanos de una estimación para ajustarse a su comportamiento, a medida que incrementamos el valor de k, se tiende a perder la señal y comenzamos a guiarlos por el ruido del modelo, por lo tanto, se debe tener cuidado a la hora de utilizar esta metodología. por otra parte, en este modelo, se deben crear vectores de variables dummy para las variables categóricas, y las variables continuas se deben normalizar. cómo se verá a continuación.

```
#modelo KNN
normalize <- function(x) {
  norm <- ((x - min(x))/(max(x) - min(x)))
  return (norm)
}
#variables categoricas, a vectores de dummy.
suppressWarnings(dummyJob<-dummy(df$job, sep="_"))
suppressWarnings(dummyMarital<-dummy(df$marital, sep="_"))
```

```

suppressWarnings(dummyEducation<-dummy(df$education, sep="_"))
suppressWarnings(dummydefault<-dummy(df$default ,sep="_"))
suppressWarnings(dummyhousing<-dummy(df$housing, sep="_"))
suppressWarnings(dummycontact<-dummy(df$contact ,sep="_"))
suppressWarnings(dummymonth<-dummy(df$month, sep="_"))
suppressWarnings(dummypoutcome<-dummy(df$poutcome, sep="_"))
suppressWarnings(dummydeposit<-dummy(df$deposit, sep="_"))

#normalización de las variables numericas.
age<-normalize(df$age)
balance<-normalize(df$balance)
day<-normalize(df$day)
duration<-normalize(df$duration)
campaign<-normalize(df$campaign)
pdays<-normalize(df$pdays)
previous<-normalize(df$previous)

#nueva base de datos, con las variables transformadas.
Newdata<-cbind(df,dummyjob,dummyMarital,dummyEducation,dummydefault,dummyhousing,dummyco
               dummpoutcome, dummydeposit,age,balance,day,duration,campaign,pdays,previo

#nueva selección de datos de entrenamiento y test
train1 <- Newdata[ train_ind,18:50 ]
test1 <- Newdata[-train_ind,18:50 ]
y_train1=df[train_ind,8]
y_test1=df[-train_ind,8]

#Modelo 1 k=1
#fit.knn_train<-knn(train=train1, test=train1,cl=y_train1, k=1, prob=TRUE)
#fit.knn_Test<-knn(train=train1, test=test1,cl=y_train1, k=1, prob=TRUE)

#Modelo 2 k=2 modelo seleccionado
fit.knn_train<-knn(train=train1, test=train1,cl=y_train1, k=2, prob=TRUE,use.all=TRUE)

fit.knn_Test<-knn(train=train1, test=test1,cl=y_train1, k=2, prob=TRUE)

#modelo 3 k= 5
#fit.knn_train<-knn(train=train1, test=train1,cl=y_train1, k=5, prob=TRUE,use.all=TRUE)
#fit.knn_Test<-knn(train=train1, test=test1,cl=y_train1, k=5, prob=TRUE)

#predict para verificar el ajuste de nuestros datos de Test.
Predicted_train<-factor(fit.knn_train)
Predicted_test<-factor(fit.knn_Test)

```

```
#probabilidades
prob_train <- attr(fit.knn_train, "prob")
prob_test <- attr(fit.knn_Test, "prob")
```

1.5. d) Con los datos de entrenamiento, implemente un modelo Logístico con loan como supervisor y las demás como predictoras.

1.5.1. Planteamiento del Modelo.

$$\text{Logit}(\pi_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_{1xi1} + \beta_{2xi2} + \dots + \beta_{kxik}$$

El modelo logístico, es una variación del modelo de regresión lineal en el que la variable respuesta es una variable dicótoma, es decir solo toma 2 valores 0 ó 1, es por esto que el logit, o el logaritmo de la razón de dos se utiliza como su predictor lineal. este modelo es naturalmente un modelo de clasificación, por lo anterior, el resultado que se va obtener es la probabilidad asociada a si una persona con diversas características tiene o no un crédito. para mayor comprensión, el modelo ajustado entregara el resultado de evaluar la siguiente expresión:

$$\pi_i = \frac{e^{\beta_0 + \beta_{1xi1} + \beta_{2xi2} + \dots + \beta_{kxik}}}{1 + e^{\beta_0 + \beta_{1xi1} + \beta_{2xi2} + \dots + \beta_{kxik}}}$$

Si el valor de la probabilidad es mayor a 0.5 entonces esta persona tiene un crédito(loan=si), en caso contrario, no tiene crédito(loan=no).

```
#Modelo logístico.
lr.fit=glm(as.factor(loan)~., data = train, family=binomial)

#predict para verificar ajuste.
lrPred_train<-predict(lr.fit,train, type = c("response"))
lrPred_test<-predict(lr.fit,test, type = c("response"))

#clasificación
predict_lr_train<-ifelse(lrPred_train<=0.5,0,1)
predict_lr_test<-ifelse(lrPred_test<=0.5,0,1)
```

1.6. e) Con los datos de entrenamiento, implemente un modelo LDA con loan como supervisor y las demás como predictoras. En LDA se modela la distribución de los predictores X de manera separada en cada una de las categorías de respuesta (es decir, condicionando en Y) y luego se usa el teorema de Bayes para obtener estimaciones de $Pr(Y = k|X = x)$ Cuando estas distribuciones se asumen normales, el modelo es muy similar en forma al de regresión logística. esta probabilidad de clasificación esta dada por:

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^k \pi_l f_l(x)}$$

```
#modelo LDA
lda.fit <- lda(as.factor(loan) ~., data=train)

#prediccion
predict_train_lda<-predict(lda.fit,train,type=c("class"))
predict_test_lda<-predict(lda.fit,test,type=c("class"))

#clasificación
train_lda<-ifelse(as.factor(train$loan)==predict_train_lda$class,0,1)
test_lda<-ifelse(as.factor(test$loan)==predict_test_lda$class,0,1)
```

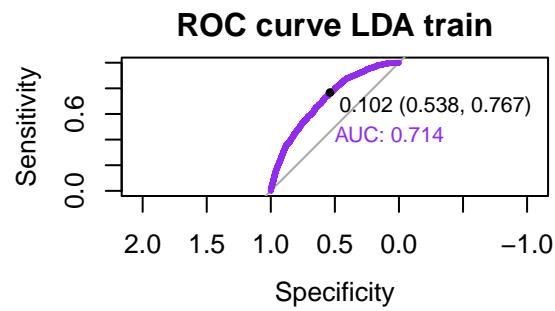
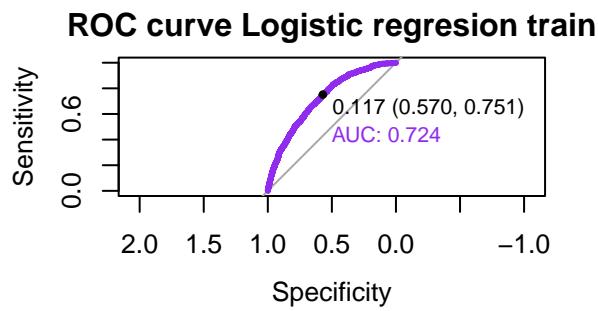
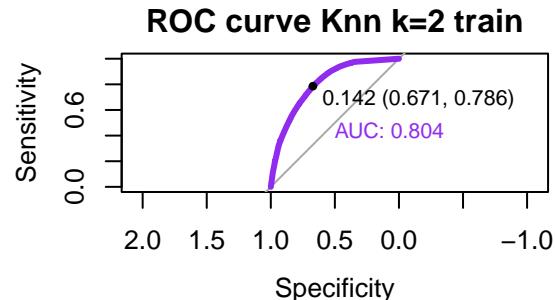
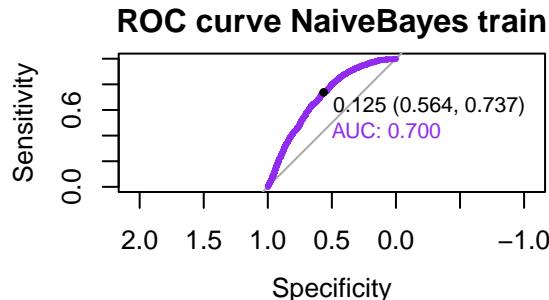
1.7. f) Con los datos de entrenamiento calcular training MSE, matriz confusión y curva roc para cada uno de los modelos.

```
## $NaiveBayes
##           y_train
## predict_train  no  yes
##           no  6864  930
##           yes   429   148
##
## $Train_NaiveBayes_err
## [1] 0.1623462
##
## $Knn
##           y_train
## Predicted_train  no  yes
##           no  7207  943
##           yes   86   135
##
```

```
## $Train_error_Knn
## [1] 0.1229244
##
## $logistic_reg
## predict_lr_train
##      0     1
## no 7286    7
## yes 1068   10
##
## $lr_train_err
## [1] 0.1284195
##
## $Lda
## train_lda
##      0     1
## no 7255   38
## yes 28 1050
##
## $LDA_train_err
## [1] 0.007884363

## $sensitividad_NaiveBayes
## [1] 0.1372913
##
## $Especificidad_NaiveBayes
## [1] 0.9411765
##
## $sensitividad_Knn
## [1] 0.1252319
##
## $Especificidad_Knn
## [1] 0.9882079
##
## $sensitividad_logistic_reg
## [1] 0.5882353
##
## $Especificidad_lr
## [1] 0.8721571
##
## $sensitividad_Lda
## [1] 0.9650735
##
## $Especificidad_LDA
## [1] 0.9961554
```

```
## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes
```



1.8. g) Con los datos de test calcular training MSE, matriz confusión y curva roc para cada uno de los modelos.

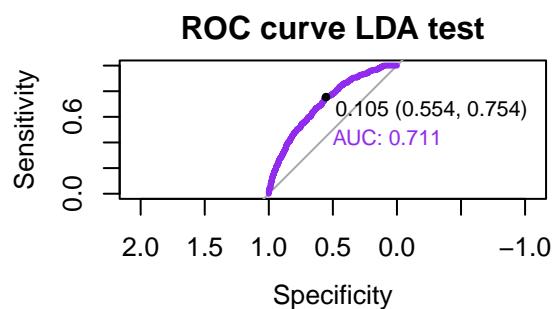
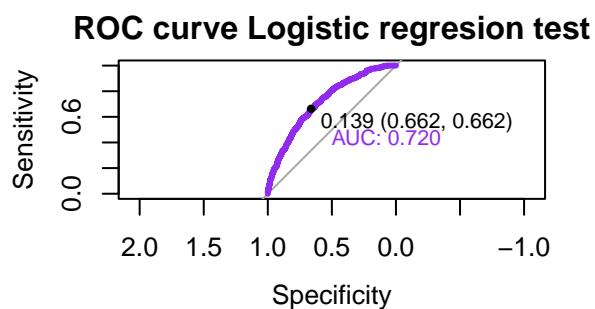
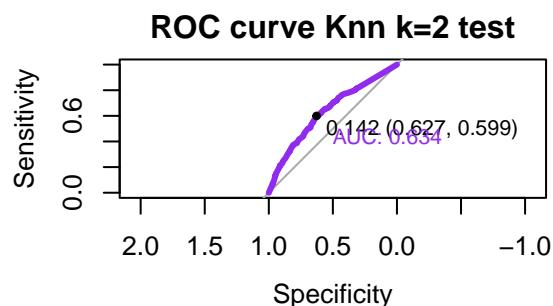
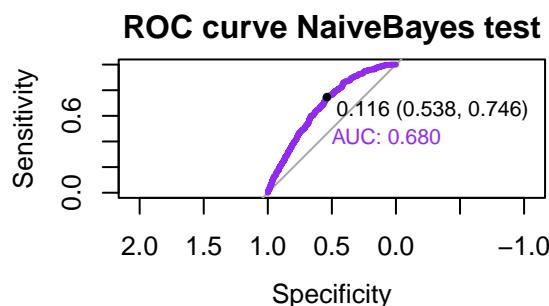
```
## $NaiveBayes
##           y_test
## predict_test  no  yes
##       no  2297   336
##       yes   112    46
##
## $Test_error_NB
## [1] 0.1605159
##
## $Knn
##           y_test1
## Predicted_test  no  yes
##       no  2346   353
##       yes   63    29
##
## $Test_error_Knn
## [1] 0.1490505
```

```

## 
## $logistic_reg
##   predict_lr_test
##     0    1
##   no  2409    0
##   yes 377    5
##
## $lr_test_err
## [1] 0.135077
##
## $Lda
##   test_lda
##     0    1
##   no  2404    5
##   yes 12  370
##
## $LDA_test_err
## [1] 0.006091007

## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes
## Setting levels: control = no, case = yes

```



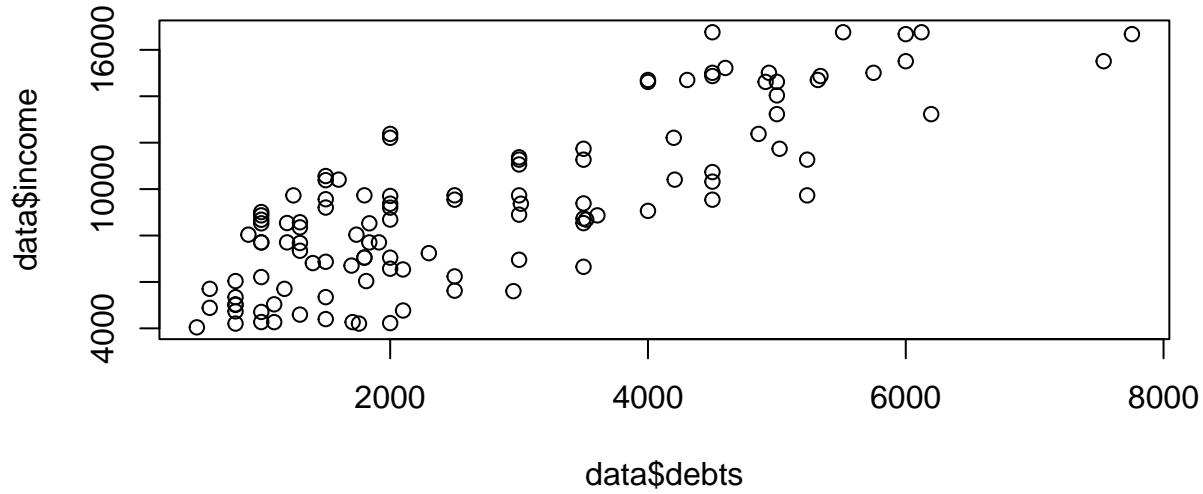
1.9. h) Con cual modelo observo mejor desempeño y por qué?

Para seleccionar el mejor modelo, es muy importante tener claro que es lo que se quiere responder. La variable loan como se dijo anteriormente representa si una persona tiene o no un crédito, esto es importante porque cometer un error de darle un crédito a una persona ya tenía uno o no darle un crédito a una persona que no lo tenía, es una decisión que genera un gran impacto negativo en las ganancias del banco, por lo cual, el modelo seleccionado debe cumplir con unos altos índices de sensitividad y de especificidad. de lo anterior, sin duda alguna el modelo LDA, obtuvo unos índices de sensibilidad y de especificidad superiores al 96 %, lo cual es una excelente tasa de clasificación, por esta razón se selecciona como el mejor modelo, a pesar de que el AUC de las curvas Roc sean similares para todos los demás modelos.

2. Ejercicio2

2.1. a) Datos de entrenamiento y de prueba

```
data <- read.csv("Data/costumer_loan_details.csv",
                  stringsAsFactors=TRUE, header = TRUE, sep = ",")  
  
set.seed(123)  
  
normalize <- function(x) {  
  norm <- ((x - min(x))/(max(x) - min(x)))  
  return (norm)  
}  
  
smp_sz <- floor(0.75 * nrow(data))  
train_indx <- sample(seq_len(nrow(data)), size = smp_sz)  
  
train <- data[train_indx, 10] %>% as.data.frame()  
test <- data[-train_indx, 10] %>% as.data.frame()  
  
y_train <- data[train_indx, 9]  
y_test <- data[-train_indx, 9]  
  
plot(data$debts, data$income)
```



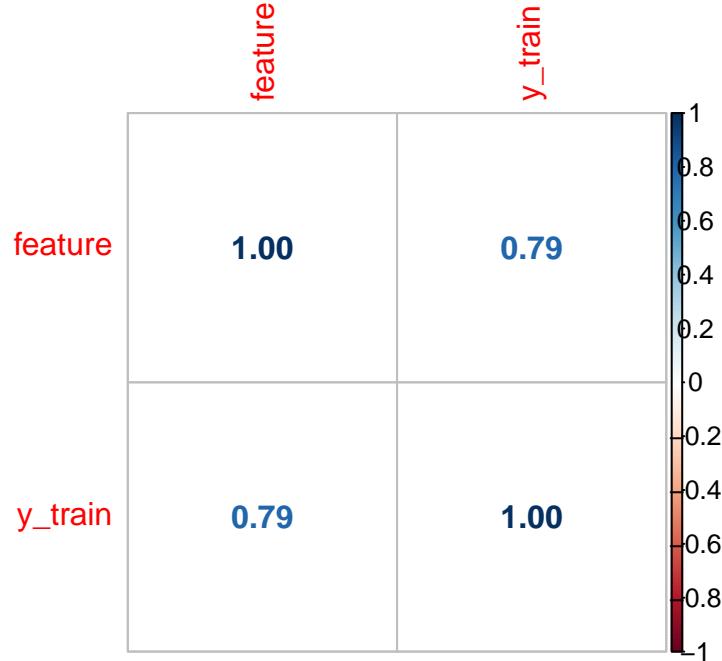
2.2. b) Implementación de KNN

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

2.3. Con los datos de entrenamiento, implemente regresión lineal simple usando income como el supervisor y debts como predictor. Grafique e interprete.

```
train_df <- cbind(train, y_train) %>% as.data.frame()
colnames(train_df) <- c("feature", "y_train")

corrplot(cor(train_df), method="numb")
```



```
model <- lm(y_train ~ feature, data=train_df)
```

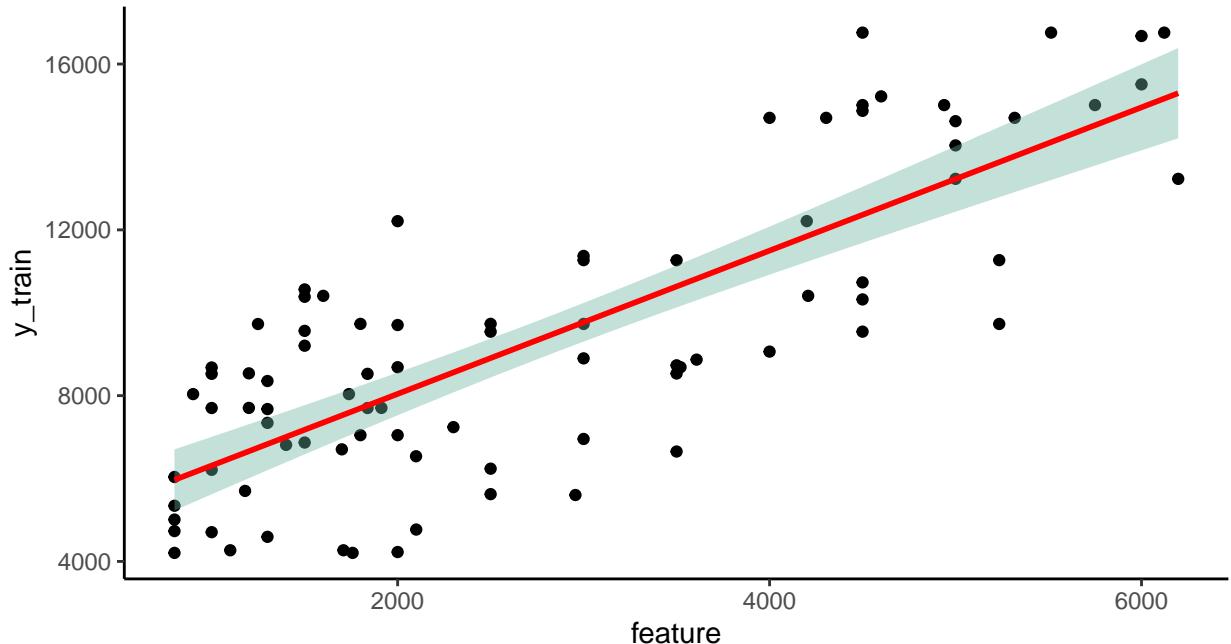
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	4584.580082	466.8251791	9.820764	0
feature	1.728516	0.1453318	11.893582	0

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

## Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

## if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow

## `geom_smooth()` using formula 'y ~ x'
```



3. Ejercicio3

4. Ejercicio4