Universidad de Costa Rica
Escuela de Matemática

Estadística 2

Proyecto Final

Profesor:

Luis Barboza

Estudiantes:

Amanda Coto Jara B42137

Esteban Molina Calvo B34311

Fecha: 20 de Julio, 2018

Índice

1. Introducción **3**

2. Metodología **4**

3. Resultados **9**
3.1. Regresión Logística 9
3.2. Redes Neuronales 12

4. Conclusiones **15**

5. Bibliografía **18**

6. Anexos **19**

1. Introducción

Se puede definir el riesgo crediticio como la posibilidad de que una persona a la cual se le otorgó un crédito no cumpla con sus obligaciones, es decir, que no pague. Por esto, el análisis de riesgo crediticio es importante en la actualidad para las entidades que operan en el negocio de los créditos, pues es una forma de tomar decisiones que ayuden a reducir el riesgo de que la empresa incurra en pérdidas.

Como parte de esta prática se puede realizar clasificación de riesgo, donde basados en el historial de la persona y otros datos influyentes se puede tomar la decisión de otorgarle o no un crédito. Gracias a los avances computacionales y de manejo de datos, ha sido posible la implementación de modelos de aprendizaje estadístico que pueden resolver la tarea de clasificación, de manera que, las empresas pueden tomar decisiones fundamentadas en métodos respaldados teóricamente.

Hay distintos métodos que pueden ser utilizados para esta labor, pero para efectos del presente proyecto se utilizará el método de redes neuronales. Además se utilizará el método de regresión logística para establecer una comparación de los resultados obtenidos y concluír cuál es mejor.

Según Ciaburro *et. al* (2017) [1] el método de redes neuronales es una técnica de aprendizaje estadístico inspirado en el funcionamiento cerebral. La idea central de comó opera este modelo es introducir información a un conjunto de unidades de procesamiento o neuronas, éstas aplican funciones matemáticas a los datos y devuelven un resultado, más adelante se explicará a fondo su funcionamiento.

El otro modelo que se utilizará es el de regresión logística, este es un modelo más simple; el procedimiento a grandes rasgos consiste en resolver el problema de máxima verosimilitud con respecto a un vector β para una función llamada logit. Con el vector obtenido y la función logit se obtiene la probabilidad de que una observación pertenezca a alguna de las clasificaciones.

2. Metodología

Se utilizarán dos técnicas o modelos para la clasificación de los clientes de un banco en el presente proyecto, la regresión logística y las redes neuronales, como se mencionó anteriormente.

Regresión Logística

El primer modelo a implementar es el de regresión logística, según lo explican James *et.al* (2013) en [2], este es un modelo de clasificación que devuelve la probabilidad de pertenecer a una de las clases que se están clasificando.

La idea de este modelo es tomar una función que dependa de un vector $\beta = (\beta_0, \beta_1, ..., \beta_p)$ y también de los valores que tienen las covariables que se quieren relacionar, para que luego, devuelva valores entre 0 y 1. Esta función produce la probabilidad deseada. En este caso la función a utilizar es la llamada función logística:

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

Donde p es la cantidad de covariables que se toman en cuenta para el modelo. Luego de definir esta función, se obtienen los valores del vector β ajustando el modelo con los datos de entrenamiento elegidos y utilizando el método de Máxima Verosimilitud.

Por último, se valida el modelo con los datos de prueba, insertando los valores de las covariables en la función $p(X)$ que tiene los valores ya encontrados del vector β, así se obtiene la probabilidad de que la observación pertenezca a alguna clase.

Cabe aclarar que este modelo clasifica variables binarias y que

el modelo devuelve probabilidades que no son el resultado final que se espera, es decir, si pertenece a una clase u otra. Para obtener la clasificación como tal, se pone una regla de decisión que coloca cada observación en una clase según su probabilidad asociada, obtenida con el modelo.

Redes Neuronales

El otro método a utilizar es el de redes neuronales, como se explica en [1], éste fue inspirado por el funcionamiento de las neuronas del cerebro humano. Este modelo tiene distintintas formas de utilizarlo, para efectos de este proyecto se utilizará la forma llamada perceptrón multicapa. Para la explicación de este método se usará la construcción de [1].

Las neuronas que se utilizan en el modelo, son unidades de procesamiento central, que mediante una operación matemática transforman los valores iniciales de las covariables en los resultados deseados. Un tipo de neurona es el perceptrón, esta unidad tiene el mismo funcionamiento interno que la neurona pero devuelve un resultado binario. Como el problema a resolver es la clasificación de un cliente en bueno o malo, la utilización de perceptrones es más adecuado, aunque también se puede resolver el problema utilizando las neuronas.

El modelo se organiza en 3 capas. La capa de valores iniciales donde se ubican en neuronas los valores de las covariables que se quieren tomar en cuenta para el modelo. La capa oculta, donde se encuentran una o más capas de neuronas y por ende donde ocurre la transformación. Por último, la capa de resultados donde se encuentra la neurona que produce el valor resultante de las operaciones realizadas en la capa oculta. Así el modelo de

perceptrón multicapa consiste en organizar varios perceptrones en las capas.

En la capa oculta se reciben los resultados de la capa de valores iniciales, acá se calcula el siguiente resultado:

$$resultado = (pesos * valores) + sesgo$$

A este resultado se le aplica una función de activación, es decir, una función matemática que devuelve el resultado deseado, este proceso se realiza en cada neurona y el resultado final es el mejor valor, dado un intervalo de activación.

La función de activación es donde ocurre la magia de las redes neuronales, por lo tanto es importante saber cuál función utilizar. En [1] se mencionan algunos ejemplos, entre ellos la función lineal $f(x) = x$, la función Logística $f(x) = \frac{e^x}{1+e^x}$ y la función Rectified Linear Unit (ReLU):

$$\begin{aligned} f(x) &= 0, x < 0 \\ &= x, x \geq 0 \end{aligned} \tag{1}$$

En [1] se menciona que una función de activación robusta debe ser derivable, simple y rápida de procesar y no centrada en 0.

Los pesos se ajustan con los datos de entrenamiento al ir comparando el resultado real con el obtenido en las neuronas, mediante el método de backpropagation. En éste se calcula la derivada parcial de cada función de activación de las neuronas para encontrar el gradiente, éste sugiere cuánto crece o decrece el error con un cambio en los pesos. El proceso cambia los pesos hasta que el error se reduzca, una cantidad llamada tasa de aprendizaje.

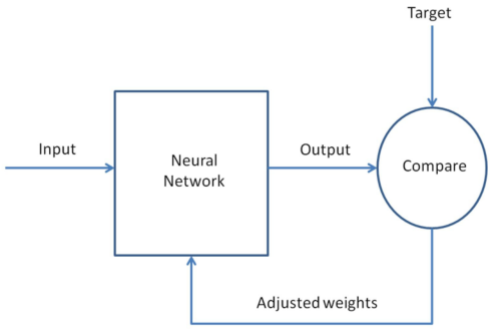

Figura 1: Diagrama del proceso de ajuste de los pesos en una red neuronal. Obtenida de [1]

La tasa de aprendizaje es un parámetro escalar utilizado para fijar la tasa de ajustes a los pesos, para reducir el error rápidamente. En la figura 1 se ve como funcionan, en general, las redes neuronales. El backpropagation ocurre en el ciclo de obtener resultados, compararlos y ajustar los pesos.

Con el fin de entender un poco más la idea detrás de la clasificación con redes neuronales, se pueden estudiar propiedades estadísticas de este modelo, pues éstas están basadas en la teoría del modelo. Con este fin, se elegió la propiedad 1, que explica desde un punto de vista Bayesiano, por qué el modelo de redes neuronales tiene la capacidad de predecir valores. El enunciado y la prueba fueron tomados de Rojas,R. (1996) [5].

Propiedad 1 (Propiedad de Bayes de redes clasificatorias): Un clasificador que implemente redes neuronales entrenado perfectamente y con suficiente flexividad, puede aprender la probabilidad a posteriori de una base de datos empírica.

Demostración. Asuma que el espacio de datos está dividido en una malla con volumen diferencial de tamaño dv, cada uno centrado en el punto n-dimensional v. Si el resultado de salida re-

presenta una clase A, la red da el valor $y(v) \in [0,1]$ para cualquier punto x en el volumen diferencial $V(v)$ centrado en v y si $p(v) = p(A|x \in V(v))$. Entonces el error cuadrático total esperado es:

$$E_A = \sum_V [p(v)(1 - y(v))^2 + (1 - p(v))y(v)^2]dv$$

donde la suma recorre todos los volúmenes diferenciales de la malla. Asuma que los $y(v)$ pueden ser calculados de manera independiente para cada volumen diferencial. O sea, cada término de la suma se puede minimizar de manera independiente, al realizar esto, es decir derivar con respecto a $y(v)$ e igualar a 0 se obtiene:

$$-2p(v)(1 - y(v)) + 2(1 - p(v))y(v) = 0$$

Al despejar $y(v)$ se obtiene que $y(v) = p(v)$, es decir, el resultado que minimiza el error en la región diferencial centrada en v, es la probabilidad a posteriori $p(v)$. Así el error esperado es:

$$p(v)(1 - p(v))^2 + (1 - p(v))p(v)^2 = p(v)(1 - p(v))$$

por lo tanto:

$$E_A = \sum_V [p(v)(1 - p(v))]dv$$

Así E_A se convierte en la varianza esperada del resultado para la clase A. □

Aunque se ha explicado el funcionamiento del modelo y cómo funcionan sus partes, hay fundamentos que parten desde la experimentación y los conocimientos previos que trae la práctica de trabajar con este tipo de modelos.

Como explica Hastie *et.al* (2009) [3] cuando se realiza la elección tanto del número de capas ocultas como la cantidad de neuronas no hay una forma óptima descrita por la teoría. Si se agregan

demasiadas se puede caer en un problema de sobreajuste , y por otro lado, si son muy pocas se carece de la flexibilidad que requiere la red para su correcto funcionamiento.

3. Resultados

Tanto en la implementación de la regresión logística como en la de las redes neuronales, se utilizó la base de datos German Credit Dataset (1994) [4], cuenta con 20 variables como historial crediticio, propósito del crédito, información personal, entre otras. Y la variable dependiente a clasificar llamada Good que tiene los niveles bueno y malo.

3.1. Regresión Logística

Para la regresión logística se trabajó con dos modelos. El primero fue obtenido según el criterio BIC y el otro por el método de Lasso. El modelo que se obtuvo con el criterio BIC es de vital importancia pues según [3] elegir un modelo bajo el criterio BIC es equivalente a elegir el modelo con la probabilidad posterior aproximada más grande. Este acercamiento bayesiano, a la hora de la elección del modelo, es de suma relevancia pues la propiedad de las redes neuronales, antes descrita, nos da un punto de conexión entre ambos enfoques, regresión logística y redes neuronales.

También se elije el modelo Lasso para tener una comparación directa entre modelos logísticos y hacer gala de la variedad de alternativas que existe, para la regresión logística a la hora de elección de un modelo. Para los tres modelos se determinó separar la base de datos total en 750 datos de entrenamiento y 250

datos de prueba.

En la implementación del BIC para encontrar el mejor modelo, se utilizó el método *Forward Stepwise Selection* y luego se comparó el valor BIC de cada uno de los modelos obtenidos. Se determinó que el mejor modelo será el que contenga las siguientes siete covariables: Status of existing checking account, Duration, Credit history, Savings account/bonds, Other debtors/guarantors, Other installment plans y Personal status and sex.

Se realizó una prueba de Hosmer-Lemeshow para verificar si el modelo tiene un buen ajuste, se encontró que el valor p de la prueba es 0.4292, lo que nos hace aceptar la hipótesis de que el modelo tiene un buen ajuste. Por otro lado, se realizó validación cruzada por el método de K-folds, utilizando k igual a 4 y se estima que el error de prueba será de 0.1665.

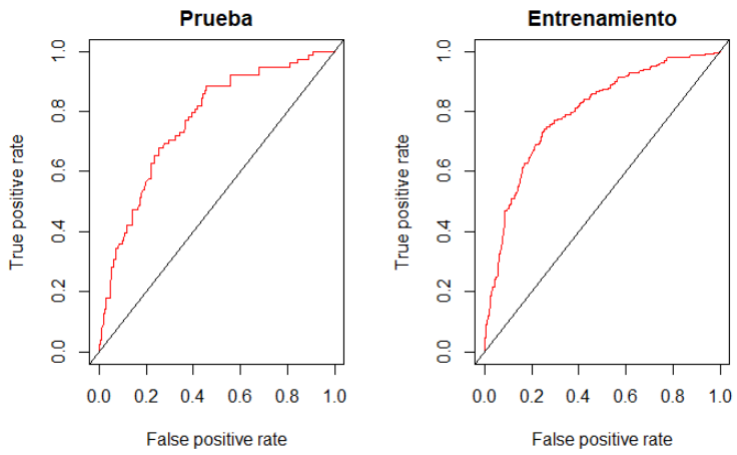

Figura 2: Curvas ROC obtenidas con el modelo elegido por el criterio BIC. Fuente: Elaboración propia

También se construyó una curva ROC para estudiar el error de entrenamiento y de prueba. Esto dejó como resultado que la

curva ROC con los datos de entrenamiento tiene un área bajo la curva de 0,7958, que nos deja ver que hay un ajuste bastante bueno, pues el valor es bastante alto sin caer tampoco en el sobreajuste, o sea no se acerca mucho a 1. Esto llega a validar el resultado de la prueba Hosmer-Lemeshow. Con los datos de prueba se obtuvo 0.7679 de poder de predicción, por lo que el modelo es bastante bueno prediciendo.

Por otro lado, con el modelo que se eligió mediante el método Lasso, se determinó que las variables que no se utilizarían son: Purpose, Present residence since, Job, Number of people being liable to provide maintenance for. Todas las demás variables de la base fueron escogidas. Las pruebas que se hicieron fueron las mismas que con el modelo anterior. Se obtuvo que el valor p de la prueba Hosmer-Lemeshow fue de 0,3575 lo que permite ver que el modelo también tiene una buena bondad de ajuste. Además, la validación cruzada estima que el error de prueba será de 0.1657.

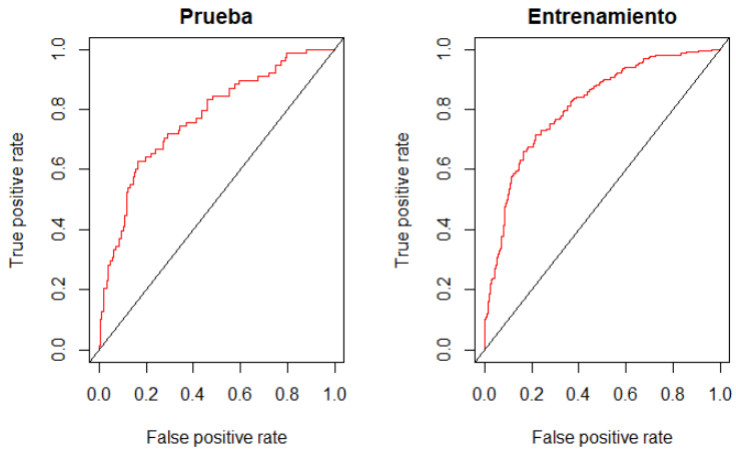

Figura 3: Curvas ROC obtenidas con el modelo elegido por el método Lasso. Fuente: Elaboración propia

Las curvas ROC de entrenamiento y prueba arrojaron los si-

guientes resultados. Para la curva de entrenamiento dio un valor de 0,8127, lo que sigue siendo bastante satisfactorio pues permite flexibilidad para no tener sobreajuste y predice bastante bien. La curva de los datos de prueba fue de 0,7723, que es una mejora significativa con respecto al anterior y se determina que es un buen modelo.

Para terminar el análisis, se nota que las covariables del modelo elegido por medio del criterio BIC son un subconjunto de las covariables del modelo elegido por Lasso. Esto nos permite realizar una prueba anova para comparar los modelos. Esta prueba nos dio un valor p de 0.000547, que deja ver que las variables que tienen de más el segundo modelo, sí son determinantes a la hora de mejorar el rendimiento del modelo. Lo que valida los resultados anteriores donde las curvas ROC eran mejores.

3.2. Redes Neuronales

El tipo de red neuronal utilizada fue el perceptrón multicapa con tres capas: una de entrada, una oculta y una de salida. La capa de entrada tienen veinte perceptrones, uno por cada covariable de la base. Para la capa oculta, gracias a la experimentación, se decidió utilizar diez perceptrones y un solo perceptrón en la capa de salida que da el resultado final. La función de activación utilizada es la función logística y el ajuste de los pesos se hizo por medio del algoritmo backpropagation y un máximo de cincuenta repeticiones de entrenamiento. Además se normalizan los datos para un mejoramiento del modelo en términos de rapidez y predicción.

Al entrenar la red se obtiene la representación gráfica de la figura 4, que permite ver la organización de las capas del modelo.

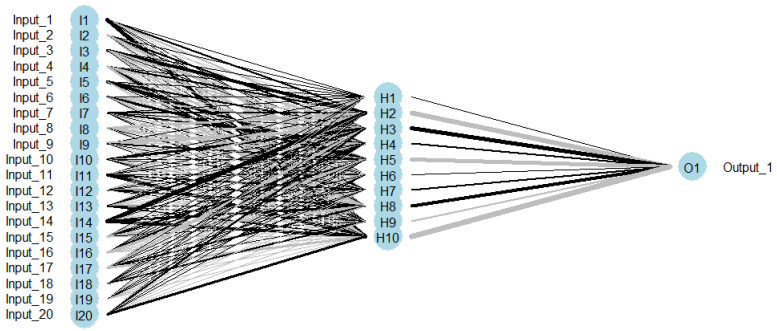

Figura 4: Descripción gráfica de la red neuronal. Fuente: Elaboración propia

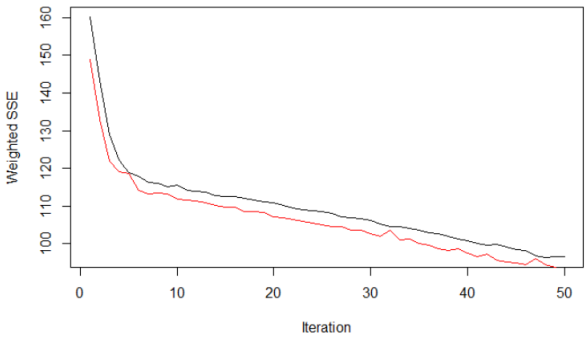

Figura 5: Error de iteración de la red neuronal con 10 perceptrones y una capa oculta. Fuente: Elaboración propia

Por otro lado, el gráfico de la figura 5 representa el error en cada iteración donde la línea negra es el error de ajuste por iteración y la línea roja es el error de prueba por iteración. Se puede notar que ambas curvas van decreciendo lo que deja ver que el algoritmo converge, o sea el error va hacia cero.

Una práctica de interés a la hora de interpretar los modelos de redes neuronales es ver qué tan importante es cada covariable para la clasificación, esto se puede analizar observando los pesos asociados a cada covariable. Para obtener los pesos, existen varios algoritmos, pero como explica Olden *et. al* (2004) en [6] el

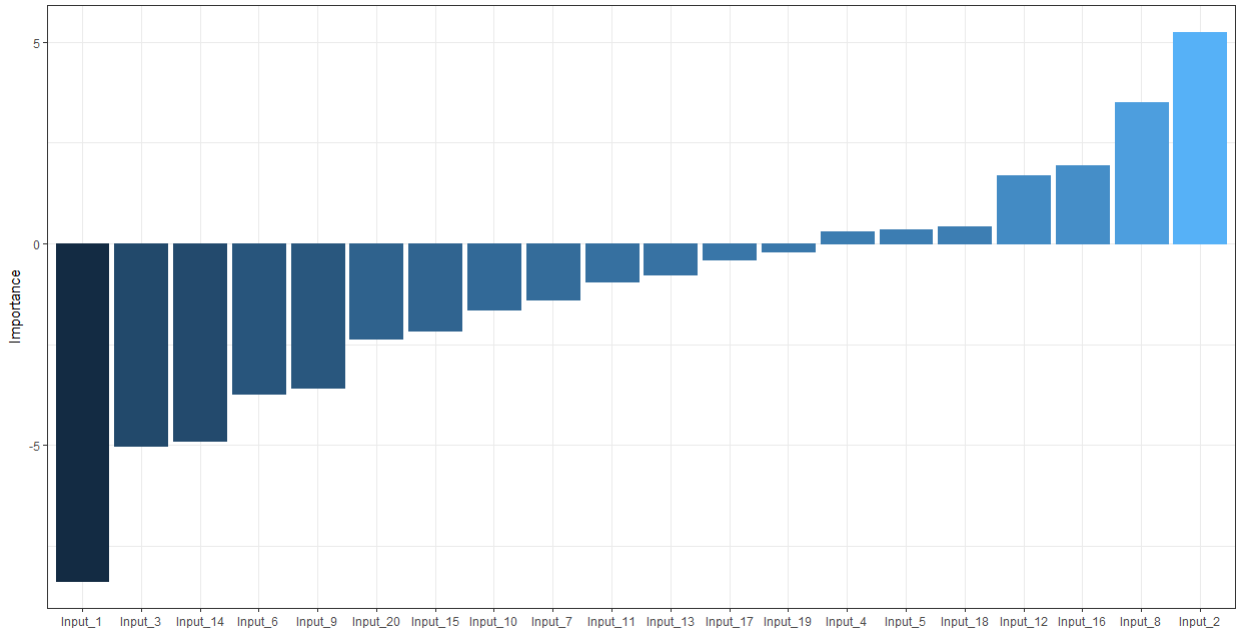

Figura 6: Comparación relativa de los pesos del modelo mediante el algoritmo Connection Weights. Fuente: Elaboración propia

que mejor desempeño tiene es *Connection Weights*. El algoritmo calcula el producto de las ponderaciones de los pesos en la capa oculta y la capa de salida entre cada neurona de entrada y neurona de salida. Luego suma los productos en todas las neuronas ocultas.

En el gráfico de la figura 6 se observa la importancia de cada variable, lo que quiere decir que entre más cercano a cero sea el valor del peso, menos importante es la variable en cuestión a la hora de clasificar en comparación a las demás. Es importante aclarar que el método es solamente para estudiar la importancia relativa de las variables a la hora de predecir, no es una selección de variables. Esto pues, lo que muestra el gráfico, no indica si hay un mejoramiento o pérdida si la variable no se tomara en cuenta en el modelo para la predicción.

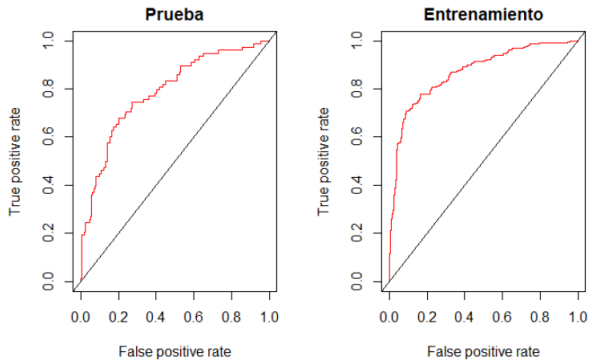

Figura 7: Curvas ROC obtenidas con las redes neuronales. Fuente: Elaboración propia

Se realizó una curva ROC, en la figura 7, igual que en el modelo logístico para determinar el poder de clasificación y el ajuste. Para los datos de entrenamiento se obtuvo 0,8691 de área bajo la curva que esta dentro de los normal aunque se acerca más a un posible sobreajuste. De la curva para los datos de prueba se obtiene un poder de clasificación de 0.7875, que es mejor con respecto a los modelos logísticos, pues el área bajo la curva ROC de este modelo es mayor, por lo que la red neuronal clasifica mejor.

4. Conclusiones

	Entrenamiento	Prueba	MSE
BIC	0.7958171	0.7679264	0.2600000
Lasso	0.8127389	0.7723614	0.2600000
Red	0.8691987	0.7874180	0.1696653

Figura 8: Resultados de los modelos. Fuente: Elaboración propia

Con los resultados de la implementación se llega a las siguientes conclusiones:

- Según los datos utilizados se puede ver que con los mismos datos de entrenamiento y de prueba, el desempeño de la

red neuronal fue significativamente mejor, en esta instancia la red neuronal es mejor que la regresión logística. La flexibilidad que tienen las redes le dan un beneficio importante sobre las regresiones. No obstante, los tres modelos predijieron bastante bien y dan resultados bastante buenos.

- A la hora de ver la importancia de las covariables en la predicción, tanto en la selección del Forward Stepwise selection y el algoritmo Connection Weigths coinciden en las covariables más importantes, tales como Status of existing checking account, credit history, duration, entre otras. Por lo que se puede ver, la importancia de las covariables coincide en los modelos y permite inferir que para el proceso de clasificación de riesgo crediticio son covariables de bastante peso, que sí determinan en qué categoría se clasifica cada observación.

- A la hora de implementar las redes neuronales hay bastantes desventajas. Es muy costosa computacionalmente, lo que hace más lento el periodo de pruebas, y con bases más grandes que la utilizada, que es relativamente pequeña, eso se convierte en un problema serio. No hay una selección de variables computacionalmente viable y los procesos más sofisticados actualmente están lejos de ser accesibles teóricamente, por lo que se tuvo que descartar la opción de realizar ese proceso. Por una parte, para la validación cruzada sucede lo mismo, computacionalmente llevar a cabo un leave one out o un K-folds para una red neuronal lleva muchísimo tiempo y es de muy alto costo. Por otra parte, todas estas técnicas para mejorar el modelo y sus predicciones en la regresión logística es muy sencillo, hay una amplia gama para elegir y también hay una alta variedad de formas para validar modelos, como las utilizadas anteriormente y computacionalmente es de bajo costo, por

lo que en este rubro la regresión logística es mejor.

- La interpretabilidad con las redes neuronales queda limitada a ver la importancia que tiene cada covariable con respecto a las demás, según el algoritmo connection weigths, lo que reduce mucho la inferencia que se pueda hacer con las redes. Además se tienen que normalizar las variables para el correcto funcionamiento del modelo, por lo que se termina acabando con la interpretabilidad del modelo y se limita su utilización meramente a la clasificación, lo que reduce las redes a una caja negra y pierde interés en muchos ámbitos. En su implementación para riesgo crediticio, al tener un mejor desempeño, puede tener gran importancia pues ese rendimiento extra significa réditos para la entidad financiera.

- La validación cruzada, la prueba Hosmer-Lemeshow y la prueba anova son herramientas de importancia significativa que permitió hacer inferencia sobre los modelos y su bondad de ajuste. La nula existencia de estas herramientas cobran factura a las redes neuronales pues se está a ciegas en este ámbito.

Se determina entonces, que más allá del poder de clasificación de las redes neuronales, es más seguro utilizar la regresión logística y su amplia gama de herramientas, por lo que llevar acabo ambas es una buena práctica, pues se puede ver qué tanta importancia tienen las covariables y comparar la clasificación. En la necesidad de utilizar solamente una, es más recomendable la reresión logística.

5. Bibliografía

Referencias

[1] Ciaburro, G. Venkateswaran, B. (2017). *Neural Networks with R*, Packt Publishing.

[2] Hastie, T. James, G. Tibshirani, R. Witten, D. (2013). *An Introduction to Statistical Learning*. Springer Science+Business Media, New York.

[3] Hastie, T. Tibshirani, R. Friedman, J. (2009) *The Elements of Statistical Learning Data Mining,Inference,and Prediction*, Springer Series in Statistics. Springer, 2nd edition.

[4] Hofmann, H. (1994-11-17). *German Credit Data*, Institut für Statistik und .ºkonometrie, Universit.ªt Hamburg. Obtenido de UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[5] Rojas,R. (1996). *Neural Networks*, Springer-Verlag, Berlin.

[6] Olden, J. Joy, M. Death, R. (2004) *An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data*, Ecological Modelling

6. Anexos

```
```{r, warning=FALSE}
setwd("C:/Users/esteb/Desktop/Actuariales/Estadística/Proyecto X")
library(readr)
library("readxl", lib.loc="~/R/win-library/3.4")
library("leaps", lib.loc="~/R/win-library/3.4")
library("ResourceSelection", lib.loc="~/R/win-library/3.4")
library("ROCR", lib.loc="~/R/win-library/3.4")
library("ggplot2", lib.loc="~/R/win-library/3.4")
library("glmnet", lib.loc="~/R/win-library/3.4")
library("ISLR", lib.loc="~/R/win-library/3.4")
library("boot", lib.loc="C:/Program Files/R/R-3.4.1/library")
library("neuralnet", lib.loc="~/R/win-library/3.4")
library("nnet", lib.loc="C:/Program Files/R/R-3.4.1/library")
library("NeuralNetTools", lib.loc="~/R/win-library/3.4")
library("deepnet", lib.loc="~/R/win-library/3.4")
library("RSNNS", lib.loc="~/R/win-library/3.4")
library("devtools", lib.loc="~/R/win-library/3.4")
library("clusterGeneration", lib.loc="~/R/win-library/3.4")
creditoG <- read_excel("C:/Users/esteb/Desktop/Actuariales/
    Estadística/Proyecto X/german credit dataset.xlsx",
    col_names = TRUE,skip = 1)

```


#Carga de datos y transformación de variables
```{r}
auxiliar = creditoG[,1:21]
auxiliar[,21] = auxiliar[,21]-1

set.seed(1)
indices = sample(1:1000, 1000, replace = FALSE)
```
```

```
auxiliar = auxiliar[indices,]
entrenamiento = auxiliar[1:750,]
prueba = auxiliar[751:1000,]


auxiliar2 = splitForTrainingAndTest(auxiliar[,1:20],
    auxiliar$Good, ratio=0.25)
auxiliar2 = normTrainingAndTestSet(auxiliar2)

auxiliar3 = splitForTrainingAndTest(auxiliar[,c(1,2,3,6,8,12,14,
    16)], auxiliar$Good, ratio=0.25)
auxiliar3 = normTrainingAndTestSet(auxiliar3)
```
```
Selección de variables
```{r}
#Lasso
x=model.matrix(Good~.,auxiliar)[,-1]
y=auxiliar$Good
grid=10^seq(10,-2, length =100)


lasso.mod=glmnet(x[1:750,] , y[1:750], alpha=1,lambda=grid)
plot(lasso.mod)

cv.out=cv.glmnet(x[1:750,],y[1:750],alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam ,newx=x[-(1:750),])
mean((lasso.pred-y[-(1:750)])^2)

out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]
lasso.coef
```
```

```
```

```{r}
auxiliar$`Credit history` = as.factor(auxiliar$`Credit history`)
auxiliar$Purpose = as.factor(auxiliar$Purpose)
auxiliar$`Savings account/bonds` =
    as.factor(auxiliar$`Savings account/bonds`)
auxiliar$`Present employment since` =
    as.factor(auxiliar$`Present employment since`)
auxiliar$`Installment rate in percentage of disposable income` =
    as.factor(auxiliar$`Installment rate in percentage of
    disposable income`)
auxiliar$`Personal status and sex` =
    as.factor(auxiliar$`Personal status and sex`)
auxiliar$`Other debtors / guarantors` =
    as.factor(auxiliar$`Other debtors / guarantors`)
auxiliar$`Present residence since` =
    as.factor(auxiliar$`Present residence since`)
auxiliar$Property = as.factor(auxiliar$Property)
auxiliar$`Other installment plans` =
    as.factor(auxiliar$`Other installment plans`)
auxiliar$Housing = as.factor(auxiliar$Housing)
auxiliar$Job = as.factor(auxiliar$Job)
auxiliar$`Number of people being liable to provide maintenance
    for` = as.factor(auxiliar$`Number of people being liable to
    provide maintenance for`)
auxiliar$Telephone = as.factor(auxiliar$Telephone)
auxiliar$`foreign worker` = as.factor(auxiliar$`foreign worker`)

conjunto <- regsubsets(Good~., data = entrenamiento, nvmax = 20,
    method = "forward")
resumen <- summary(conjunto)
resumen$bic
```

```
```
Validacion del modelo
```{r}
modeloBIC <- glm(Good~`Status of existing checking account`+
    Duration+`Credit history`+`Savings account/bonds`+
    `Other debtors / guarantors`+`Other installment plans`+
    `Personal status and sex`,family = binomial,
    data = entrenamiento)

modeloLasso <- glm(Good~.-Purpose-`Present residence since`-Job-
    `Number of people being liable to provide maintenance for`,
    family = binomial, data = entrenamiento)

#K-fold
cv.errorBIC = cv.glm(entrenamiento,modeloBIC,K=10)$delta[1]
cv.errorLasso= cv.glm(entrenamiento,modeloLasso,K=10)$delta[1]

anova(modeloBIC,modeloLasso,test='Chi')

observaciones <- as.numeric(entrenamiento$Good)
hoslemBIC <- hoslem.test(x = observaciones,
    y = modeloBIC$fitted.values)
hoslemBIC

hoslemLasso <- hoslem.test(x = observaciones,
    y = modeloLasso$fitted.values)
hoslemLasso

```

Modelo logistico
```{r}
```

```
Tabla = matrix(0, nrow = 3, ncol = 3, dimnames =
    list(c("BIC", "Lasso", "Red"),
    c("Entrenamiento", "Prueba", "MSE")))
prediccionesBIC <- predict(modeloBIC,prueba)
glmBIC.pred <- rep(0 ,250)
glmBIC.pred[prediccionesBIC>0.5]<-1
MSE.glmBIC <- sum((glmBIC.pred- prueba$Good)^2)/nrow(prueba)
Tabla[1,3] = MSE.glmBIC

#Curva ROC prueba
prediccionesBIC2 <- predict(modeloBIC , prueba ,type = "response")
obj.predBIC <- prediction(prediccionesBIC2,prueba$Good)
obj.perBIC <- performance(obj.predBIC,"tpr","fpr")
plot(obj.perBIC,col=2)
abline(a = 0,b = 1)
areaROCBIC <- performance(obj.predBIC,"auc")
Tabla[1,2] = as.numeric(areaROCBIC@y.values)

#Curva ROC entrenamiento
prediccionesBIC3 <- predict(modeloBIC , entrenamiento,
    type = "response")
obj.predBIC3 <- prediction(prediccionesBIC3,entrenamiento$Good)
obj.perBIC3 <- performance(obj.predBIC3,"tpr","fpr")
plot(obj.perBIC3,col=2)
abline(a = 0,b = 1)
areaROCBIC3 <- performance(obj.predBIC3,"auc")
Tabla[1,1] = as.numeric(areaROCBIC3@y.values)

par(mfrow = c(1,2))
plot(obj.perBIC,col=2, main = "Prueba")
abline(a = 0,b = 1)
plot(obj.perBIC3,col=2, main = "Entrenamiento")
abline(a = 0,b = 1)
```

```
prediccionesLasso <- predict(modeloLasso,prueba)
glmLasso.pred <- rep(0 ,250)
glmLasso.pred[prediccionesLasso>0.5]<-1
MSE.glmLasso <- sum((glmLasso.pred- prueba$Good)^2)/nrow(prueba)
Tabla[2,3] = MSE.glmLasso

#Curva ROC prueba
prediccionesLasso2 <- predict(modeloLasso , prueba,
    type = "response")
obj.predLasso <- prediction(prediccionesLasso2,prueba$Good)
obj.perLasso <- performance(obj.predLasso,"tpr","fpr")
plot(obj.perLasso,col=2)
abline(a = 0,b = 1)
areaROCLasso <- performance(obj.predLasso,"auc")
Tabla[2,2] = as.numeric(areaROCLasso@y.values)

#Curva ROC entrenamiento
prediccionesLasso3 <- predict(modeloLasso , entrenamiento,
    type = "response")
obj.predLasso3 <- prediction(prediccionesLasso3,
    entrenamiento$Good)
obj.perLasso3 <- performance(obj.predLasso3,"tpr","fpr")
plot(obj.perLasso3,col=2)
abline(a = 0,b = 1)
areaROCLasso3 <- performance(obj.predLasso3,"auc")
Tabla[2,1] = as.numeric(areaROCLasso3@y.values)

par(mfrow = c(1,2))
plot(obj.perLasso,col=2, main = "Prueba")
abline(a = 0,b = 1)
plot(obj.perLasso3,col=2, main = "Entrenamiento")
abline(a = 0,b = 1)
```

```
```
#Redes Neuronales
```{r}
modelo <- mlp(auxiliar2$inputsTrain, auxiliar2$targetsTrain,
    size= 10,learnFuncParams=c(0.1),maxit=50,
    inputsTest=auxiliar2$inputsTrain,
    targetsTest=auxiliar2$targetsTrain)
```
```{r}
olden(modelo)

plotnet(modelo)
plotIterativeError(modelo)
predicciones.perceptron = predict(modelo,auxiliar2$inputsTest)
plotRegressionError(predicciones.perceptron,
    auxiliar2$targetsTest)

MSE.net <- sum(( predicciones.perceptron - prueba$Good)^2)/250
Tabla[3,3] = MSE.net

#Curva ROC prueba

plotROC(predicciones.perceptron, auxiliar2$targetsTest, col = 2)
abline(a = 0,b = 1)
obj.predPerceptron <- prediction(predicciones.perceptron,
    auxiliar2$targetsTest)
obj.perPerceptron <- performance(obj.predPerceptron,"tpr","fpr")
areaROCPerceptron <- performance(obj.predPerceptron,"auc")
Tabla[3,2] = as.numeric(areaROCPerceptron@y.values)

#Curva ROC entrenamiento

predicciones.perceptron2 = predict(modelo,auxiliar2$inputsTrain)
```

```
plotROC(fitted.values(modelo), auxiliar2$targetsTrain, col = 2)
abline(a = 0,b = 1)
obj.predPerceptron2 <- prediction(predicciones.perceptron2,
    auxiliar2$targetsTrain)
obj.perPerceptron2 <- performance(obj.predPerceptron2,"tpr","fpr")
areaROCPerceptron2 <- performance(obj.predPerceptron2,"auc")
Tabla[3,1] = as.numeric(areaROCPerceptron2@y.values)

par(mfrow = c(1,2))
plot(obj.perPerceptron,col=2, main = "Prueba")
abline(a = 0,b = 1)
plot(obj.perPerceptron2,col=2, main = "Entrenamiento")
abline(a = 0,b = 1)
```
‘‘‘