Realizado por:

Juan Camilo Restrepo Velez
William Leonardo Andrade Collazos
Wilder Valencia Ocampo

PROYECTO MINERÍA DE DATOS EN R Y PYTHON

Bank Marketing

Los datos están relacionados con campañas de marketing directo de una institución bancaria portuguesa. Las campañas de marketing se basaron en llamadas telefónicas. A menudo, se requería más de un contacto con el mismo cliente, para poder acceder a si el producto (depósito bancario a plazo) sería ('sí') o no ('no') suscrito.

Información de atributos

Información Bancaria de los clientes

Age - Edad

Job - Trabajo: tipo de trabajo

Marital - Estado civil: estado civil

Education - Educación: Nivel educativo

Default - Incumplimiento: ¿tiene el crédito en mora?

Housing - Vivienda: ¿tiene un préstamo de vivienda?

Loan - Préstamo: ¿tiene préstamo personal?

Relacionado con la última llamada de la actual campaña

Contact - Contacto: tipo de comunicación

Month - Mes: último mes de contacto del año

DayofWeek - Día de la semana: último día de contacto de la semana

Duration - Duración: duración del último contacto, en segundos (numérico). Nota importante: este atributo afecta en gran medida al objetivo de salida (por ejemplo, si la duración = 0, entonces y = "no"). Sin embargo, no se conoce la duración antes de una llamada se realiza. Además, después del final de la llamada se conoce obviamente y. Por lo tanto, esta entrada sólo debe incluirse a efectos de referencia y debe descartarse si se pretende tener un modelo predictivo realista.

Otros

Campaign - Campaña: número de contactos realizados durante esta campaña y para este cliente

Pdays - pDías: número de días que pasaron después de que el cliente fue contactado por última vez en una campaña anterior. Nota, 999 significa que el cliente no fue contactado anteriormente

Previous - Anterior: número de contactos realizados antes de esta campaña y para este cliente

Poutcome: resultado de la anterior campaña de marketing

Atributos del contexto social y económico

Emp.var.rate - Tasa de variación del empleo - indicador trimestral

Cons.price.idx: Índice de Precios al Consumidor - Indicador mensual; el Índice de Precios al Consumidor o IPC mide los cambios en los precios pagados por los consumidores por una cesta de bienes y servicios cada mes.

Cons.conf.idx: Índice de confianza del consumidor - Indicador mensual; En Portugal, el índice de confianza del consumidor se basa en entrevistas con los consumidores sobre sus percepciones de la situación económica actual y futura del país y sus tendencias de compra. Se estima utilizando la diferencia entre la proporción de respuestas de evaluación positivas y las respuestas de evaluación negativas, pero no incluye la proporción de respuestas neutras

Euribor3m: euribor 3 meses - Euribor es la abreviatura de Euro Interbank Offered Rate. es un índice de referencia publicado diariamente que indica el tipo de interés promedio al que un gran número de bancos europeos dicen concederse préstamos a corto plazo entre ellos para prestárselo a terceros.

Nr.employed - Número de empleados: Número de empleados - Indicador trimestral; Número de personas empleadas para el trimestre.

y - ¿el cliente ha suscrito un depósito a plazo? (Variable objetivo)

*Tomado de https://www.kaggle.com/henriqueyamahata/bank-marketing

PREPARACIÓN DE DATOS

Ya se realizó debido a que se tomaron los datos utilizados en “Practica Análisis Predictivo”

Aprendizaje supervisado- Árbol de decisión- Modelo R

Objetivo: Realizar un análisis predictivo por medio de la creación de un árbol de decisión que nos ayudará a predecir si 6 nuevos clientes del banco se suscribirán a un depósito a plazo.

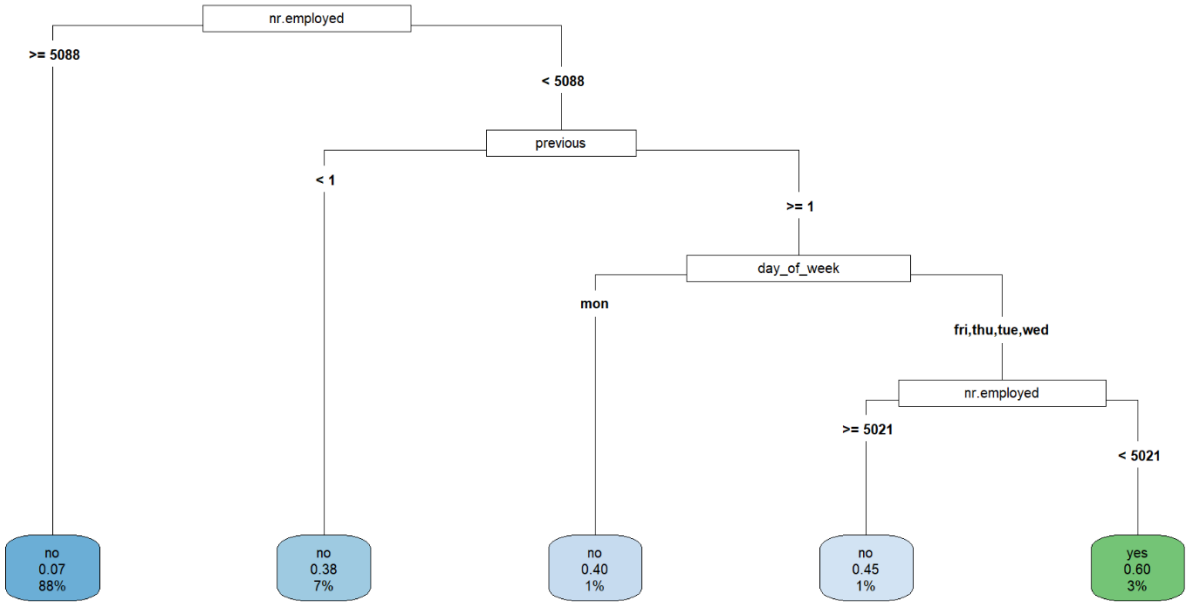
División de datos: Se realiza una división 70-30 al histórico de datos.

```
# DIVISION 70 - 30
##############################################################################################
library(caret)

intrain <- createDataPartition(datos$y, p = 0.7, list = FALSE)
training <- datos[intrain,]
testing <- datos[-intrain,]
##############################################################################################
```

Aprendizaje: Se crea un árbol de decisiones donde la mínima cantidad de instancias por hojas se establece en 5 y la máxima profundidad en 5 para poder realizar un mejor análisis de este.

```
# ARBOL DE DESICION
##############################################################################
library(rpart)
library(rpart.plot)

modelo <- rpart(y ~ ., method = 'class', data = training, control = rpart.control(minbucket=5, maxdepth=7))
windows()
rpart.plot(modelo, type = 5)
##############################################################################
```

La vista gráfica del árbol nos da como resultado que la variable más importante es **nr.employed**



Evaluación: Se realiza la evaluación del modelo sobre el conjunto de entrenamiento.

```
# Evaluacion
###############################################################################
library(e1071)

predicciones <- predict(modelo, testing, type = 'class')
confusionMatrix(data = as.factor(predicciones), reference = as.factor(testing$y), mode = 'prec_recall')

# Guardar el modelo
save(modelo, file="C:/Users/Administrador/Downloads/Arbol.rmodel")
###############################################################################
```

En cuanto a las estadísticas la precisión del 90% y la cobertura del 98%, los cuales son porcentajes aceptables para el modelo predictivo.

```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  10789  1157
       yes   175   235

               Accuracy : 0.8922
                 95% CI : (0.8866, 0.8976)
    No Information Rate : 0.8873
    P-Value [Acc > NIR] : 0.04457

                  Kappa : 0.2209

 Mcnemar's Test P-Value : < 2e-16

              Precision : 0.9031
                 Recall : 0.9840
                     F1 : 0.9419
             Prevalence : 0.8873
         Detection Rate : 0.8732
   Detection Prevalence : 0.9668
      Balanced Accuracy : 0.5764

       'Positive' Class : no
```

Predicción Futura: Se aplica el modelo al conjunto de datos futuros (6) para predecir si los clientes se suscribirán a un depósito a plazo.

```
# Prediccion futura
###############################################################################################
rm(list = ls())

# Cargar el modelo
load(file.choose())

#Cargar datos
datosFuturos <- read.csv(file = file.choose(), header = TRUE, sep = ',')

prediccion <- predict(modelo, datosFuturos, type = 'class')
print(prediccion)
###############################################################################################
```

```
> print(prediccion)
 1  2  3  4  5  6
no no no no no no
Levels: no yes
```

Conocimiento nuevo: Según los resultados del árbol de decisión se observa que la variable más importante es número de empleados y con los 6 datos predictivos se dictaminó que ninguna de las personas aceptaría a la suscripción a plazo.

Aprendizaje Supervisado – Redes Neuronales - Modelo en Python

Objetivo: Realizar un análisis predictivo por medio de la creación de redes neuronales(Backpropagation y Deep Learnig) que nos ayudará a predecir si 6 nuevos clientes del banco se suscribirán a un depósito a plazo.

Nota (Preparación de datos): Debido a que la librería 'sklearn', solo trabaja con variables numéricas se deben realizar los correspondientes encoders y dummies.

```
#Dummies
dummiesClase=pd.get_dummies(data['job'])
data=data.drop('job', axis=1)
data=data.join(dummiesClase)

dummiesClase=pd.get_dummies(data['marital'])
data=data.drop('marital', axis=1)
data=data.join(dummiesClase)

dummiesClase=pd.get_dummies(data['education'])
data=data.drop('education', axis=1)
data=data.join(dummiesClase)

dummiesClase=pd.get_dummies(data['month'])
data=data.drop('month', axis=1)
data=data.join(dummiesClase)

dummiesClase=pd.get_dummies(data['day_of_week'])
data=data.drop('day_of_week', axis=1)
data=data.join(dummiesClase)

data.head()
```

```
#Encoders
data['default']=data['default'].replace({"yes": 1, "no": 0})
data['housing']=data['housing'].replace({"yes": 1, "no": 0})
data['loan']=data['loan'].replace({"yes": 1, "no": 0})
data['y']=data['y'].replace({"yes": 1, "no": 0})
data.head()
```

División de datos: Se realiza una división 70-30 al histórico de datos.

```
#División 70-30
from sklearn.model_selection import train_test_split
X = data.drop('y', axis = 1)
Y = data['y']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, stratify=Y)
print(f'70% --> {Y_train.count()} Registros')
print(f'30% --> {Y_test.count()} Registros')
```

```
70% --> 28831 Registros
30% --> 12357 Registros
```

Aprendizaje: Se configuran dos redes neuronales, una Bakpropagation y otra DeepLearning para crear un modelo de cada una.

A. **Backpropagation:** Para la capa oculta se utiliza la función de activación logistic, siguiendo la recomendación de que se utiliza cuando los datos de la variable objetivo son

positivos, además se configuran 23 neuronas (que corresponden a “a”), una tasa de aprendizaje de 0.9 y un momentum de 0.3, con un máximo de 50 iteraciones.

```
#Creación del modelo con el conjunto de entrenamiento
from sklearn.neural_network import MLPClassifier
model_A = MLPClassifier(activation="logistic",hidden_layer_sizes=(23),learning_rate="adaptive",learning_rate_init=0.9,
                        momentum=0.3,max_iter=50,verbose=True,random_state=1)
model_A.fit(X_train, Y_train)
```

```
Iteration 1, loss = 0.92884791
Iteration 2, loss = 0.36692197
Iteration 3, loss = 0.36329122
Iteration 4, loss = 0.36595209
Iteration 5, loss = 0.38168643
Iteration 6, loss = 0.37894622
Iteration 7, loss = 0.39336675
Iteration 8, loss = 0.37325769
Iteration 9, loss = 0.39717390
Iteration 10, loss = 0.38161985
Iteration 11, loss = 0.37606174
Iteration 12, loss = 0.37704632
Iteration 13, loss = 0.37268605
Iteration 14, loss = 0.37671678
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
MLPClassifier(activation='logistic', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=23, learning_rate='adaptive',
              learning_rate_init=0.9, max_fun=15000, max_iter=50, momentum=0.3,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=True, warm_start=False)
```

B. **DeepLearning:** Para el modelo se establecen 2 capas ocultas y la última que es la de salida; la primera capa con 45 neuronas, la segunda capa con 10 neuronas y ambas capas con la función de activación relu y la capa de salida con la función sigmoid, además se establecen 50 iteraciones para evitar el sobre entrenamiento.

```
#Creación del modelo con el conjunto de entrenamiento
from keras.models import Sequential
from keras.layers.core import Dense
model_B=Sequential()
model_B.add(Dense(45,input_dim=45, activation='relu'))
model_B.add(Dense(10, activation='relu'))
model_B.add(Dense(1, activation='sigmoid'))
model_B.compile(loss='mean_squared_error',optimizer='adam',metrics=['accuracy'])
model_B.fit(X_train, Y_train, epochs=50)
```

Evaluación: Se realiza la evaluación del modelo sobre el conjunto de entrenamiento.

A. **Backpropagation:** La exactitud es del 88%

```
#Evaluación sobre el conjunto de prueba
from sklearn import metrics
Y_pred = model_A.predict(X_test)
acc=metrics.accuracy_score(Y_test, Y_pred)
print(f'Exactitud MLP: {acc}')
```

```
Exactitud MLP: 0.8873512988589464
```

B. DeepLearning: Muestra dos resultados, el primero es el error que es del 11% y el segundo que es la exactitud que es del 88%

```
#Evaluación sobre el conjunto de prueba

print(f'Exactitud Aprendizaje Profundo: {model_B.evaluate(X_test, Y_test)}')
```

```
387/387 [==============================] - 0s 696us/step - loss: 0.1126 - accuracy: 0.8874
Exactitud Aprendizaje Profundo: [0.11264870315790176, 0.8873512744903564]
```

Predicción Futura: Se aplica el modelo al conjunto de datos futuros (6) para predecir si los clientes se suscribirán a un depósito a plazo.

A. Backpropagation

```
#Prediccion
print("Red Neuronal Backpropagation")

Y_fut_A = model_A.predict(data_fut)
print(Y_fut_A)
```

```
Red Neuronal Backpropagation
[0 0 0 0 0 0]
```

B. DeepLearning

```
#Prediccion
print("Red Neuronal Profunda")
Y_fut_B = model_B.predict(data_fut)
print(Y_fut_B.round())
```

```
Red Neuronal Profunda
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

Conocimiento nuevo: Según los resultados de las redes neuronales se observa que la exactitud es muy cercana a 1 y con los 6 datos predictivos se dictaminó que ninguna de las personas aceptaría a la suscripción a plazo.