Universidad del Valle de Guatemala
Minería de Datos - Sección 10
María Fernanda Estrada 14198
Christopher Kevin Sandoval 13660
27/04/2020

# Hoja de Trabajo 7

## Redes neuronales

### Transformaciones

Para aplicar los modelos de redes neuronales se utilizaron las mismas variables que el modelo de Naive Bayes, las cuales eran: MSSubClasss, OverallCond, YearBuilt, BsmtFinSF1, X2ndFlrSF, BsmtFullBath, BedroomAbvGr y SceenPorch. Al leer los datos, se pretendió dejar estas variables únicamente para clasificar una casa como Económica, Intermedia o Cara. Además, se dejaron los mismos límites de precio para crear la variable respuesta “Class”; si el precio es mayor a $270,000 es Cara, si el precio es menor a $195,000 es Económica, y si el precio está entre esos límites es Intermedia.

### Modelo nnet (paquete caret)

Este modelo de red neuronal sirve para clasificación (función logística) o para regresión (función lineal). Como parámetros específicos recibe el tamaño de la red neuronal y el decaimiento, si se desea especificar. Como en este caso se deseaba clasificar el tipo de casa, se utilizó la función de activación default que es la logística.

```
> modeloCaret
Neural Network

1460 samples
   8 predictor
   3 classes: 'Cara', 'Economica', 'Intermedia'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1460, 1460, 1460, 1460, 1460, 1460, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0e+00  0.6964943  0.05217303
  1     1e-04  0.6941298  0.03964442
  1     1e-01  0.7069948  0.25567536
  3     0e+00  0.6983512  0.10453143
  3     1e-04  0.6996024  0.09670911
  3     1e-01  0.7318342  0.35710290
  5     0e+00  0.7148717  0.18213173
  5     1e-04  0.7072960  0.17219862
  5     1e-01  0.7343054  0.34301647
```

Ahora para determinar si el modelo de nnet fue exitoso, se obtuvo la matriz de confusión respectiva. A simple vista el algoritmo tuvo un buen desempeño, siendo el accuracy de **78.9%**. Sin embargo, se observa que predijo la mayoría como Económica e Intermedia y ninguna Cara. Tuvo más aciertos en clasificar casas Económicas y menos aciertos en las casas Intermedias. El error que más puede pesar es el de clasificar una casa como Económica cuando realmente es Cara; en este caso tuvo 2 errores de este tipo.

```
Confusion Matrix and Statistics

            Reference
Prediction   Cara Economica Intermedia
  Cara          0        71          17
  Economica     2      1087         108
  Intermedia    0       109          63

Overall Statistics

               Accuracy : 0.7893
                 95% CI : (0.7674, 0.81)
    No Information Rate : 0.8696
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2204

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

**Modelo pcaNNet (paquete caret)**

Al igual que el modelo anterior, este modelo sirve para clasificación (función logística) o para regresión (función lineal). Como parámetros específicos recibe el tamaño de la red neuronal y el decaimiento, si se desea especificar. Como en este caso se deseaba clasificar el tipo de casa, se utilizó la función de activación default que es la logística.

```
> modeloCaret2
Neural Networks with Feature Extraction

1460 samples
   8 predictor
   3 classes: 'Cara', 'Economica', 'Intermedia'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1460, 1460, 1460, 1460, 1460, 1460, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0e+00  0.6917227  0.03325928
  1     1e-04  0.7199904  0.22283601
  1     1e-01  0.7658873  0.44314733
  3     0e+00  0.7016152  0.08172899
  3     1e-04  0.7125684  0.16773502
  3     1e-01  0.7782329  0.49747836
  5     0e+00  0.6992129  0.07628140
  5     1e-04  0.7390064  0.33701730
  5     1e-01  0.7865837  0.52039565
```

Ahora para determinar si el modelo de pcaNNet fue exitoso, se obtuvo la matriz de confusión respectiva. A simple vista el algoritmo tuvo un buen desempeño, siendo el accuracy de **72.1%**. El algoritmo tuvo más aciertos en clasificar casas Económicas y menos aciertos en las casas Intermedias. El error que más puede pesar es el de clasificar una casa como Económica cuando realmente es Cara; en este caso tuvo 1 errores de este tipo. Este modelo sí logró clasificar una casa como Cara.

```
Confusion Matrix and Statistics

            Reference
Prediction   Cara Economica Intermedia
  Cara          1       112         26
  Economica     1       994        107
  Intermedia    0       161         55

Overall Statistics

               Accuracy : 0.7207
                 95% CI : (0.6968, 0.7436)
    No Information Rate : 0.8696
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1352

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

**Comparación de resultados entre modelos**

En cuanto a efectividad, el mejor algoritmo de RNA fue el de nnet que obtuvo un 6.8% mayor accuracy que el de pcaNNet. Igualmente el tiempo de ejecución de nnet fue mucho mejor (tomó menos tiempo) que ejecutar pcaNNet. Al observar la clasificación como tal, ambos algoritmos estuvieron muy similares; clasificaron correctamente a la mayoría de Económicas y se confundieron con Intermedias y Caras. Sin embargo, es importante notar que nnet no logró clasificar ninguna Cara y pcaNNet sí.

**Comparación RNA nnet con anteriores algoritmos**

Si se compara únicamente el accuracy de los algoritmos, el modelo de red neuronal nnet es superior. Sin embargo, se debe considerar que los otros algoritmos sí lograron clasificar con más facilidad las casas Caras. Además, cometieron menos el error de clasificar una casa Cara como Económica, que es el peor tipo de error en este caso y se tardó más en procesar el modelo de red neuronal. Como sugerencia, se debe aumentar los niveles de capas de las redes neuronales, su complejidad o cambiar los parámetros en los que se basa la clasificación.

## SVM

### Modelo 1

El primer modelo de SVM utilizó los siguientes parámetros:

- kernel: lineal
- c: $2^5$

Su propósito (o tipo) fue el de clasificación.

```
> modeloSVM_1

Call:
svm(formula = Class ~ ., data = data_training_filtered, cost = 2^5, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  32

Number of Support Vectors:  696
```

Al aplicar la matriz de confusión, se obtuvo un accuracy de **77.5%**, siendo este algoritmo muy bueno en la clasificación de casas. Tuvo mayor aciertos en clasificar casas Económicas e Intermedias. En este algoritmo, sí hubieron mayor cantidad de casas clasificadas correctamente como Caras.

```
Confusion Matrix and Statistics

             Reference
Prediction    Cara Economica Intermedia
  Cara          83        42         45
  Economica      6       931         66
  Intermedia    26       144        117

Overall Statistics

               Accuracy : 0.7747
                 95% CI : (0.7523, 0.7959)
    No Information Rate : 0.7651
    P-Value [Acc > NIR] : 0.2028

                  Kappa : 0.4814

 Mcnemar's Test P-Value : 3.496e-13
```

### Modelo 2

El segundo modelo de SVM utilizó los siguientes parámetros:

- kernel: lineal
- c: $2^{-5}$

Su propósito (o tipo) fue el de clasificación.

```
> modeloSVM_2

Call:
svm(formula = Class ~ ., data = data_training_filtered, cost = 2^-5,
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.03125

Number of Support Vectors:  736
```

Al aplicar la matriz de confusión, se obtuvo un accuracy de **78.2%**, siendo este algoritmo muy bueno en la clasificación de casas. Tuvo mayor aciertos en clasificar casas Económicas e Intermedias. En este algoritmo, sí hubieron mayor cantidad de casas clasificadas correctamente como Caras.

```
Confusion Matrix and Statistics

            Reference
Prediction   Cara Economica Intermedia
  Cara         82        43         45
  Economica     8       955         40
  Intermedia   29       154        104

Overall Statistics

               Accuracy : 0.7815
                 95% CI : (0.7594, 0.8025)
    No Information Rate : 0.789
    P-Value [Acc > NIR] : 0.7705

                  Kappa : 0.4835

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

**Modelo 3**

El tercer modelo de SVM utilizó los siguientes parámetros:

- kernel: radial
- gamma: $2^{-5}$

Su propósito (o tipo) fue el de clasificación.

```
> modeloSVM_3

Call:
svm(formula = Class ~ ., data = data_training_filtered, gamma = 2^-5,
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  716
```

Al aplicar la matriz de confusión, se obtuvo un accuracy de **79.3%**, siendo este algoritmo muy bueno en la clasificación de casas. Tuvo mayor aciertos en clasificar casas Económicas e Intermedias. En este algoritmo, sí hubieron mayor cantidad de casas clasificadas correctamente como Caras.

```
Confusion Matrix and Statistics

            Reference
Prediction   Cara Economica Intermedia
  Cara         80        38         52
  Economica     5       958         40
  Intermedia   18       150        119

Overall Statistics

               Accuracy : 0.7925
                 95% CI : (0.7707, 0.813)
    No Information Rate : 0.7849
    P-Value [Acc > NIR] : 0.2529

                  Kappa : 0.5107

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

**Comparación de resultados entre modelos**

Los tres modelos fueron similares en la cantidad de clasificaciones correctas e incorrectas de los tres tipos de casas. También el tiempo de procesamiento de los tres fue rápido y de similar duración. La diferencia está en el accuracy de cada uno, siendo el mejor el del modelo radial con un accuracy del 79.3%.

**Comparación SVM con anteriores algoritmos**

El algoritmo de SVM, específicamente el modelo radial, superó a todos los algoritmos de las hojas anteriores y al de RNA. Su accuracy fue mejor, su clasificación fue mejor y el tiempo de procesamiento fue el menor de todos. En general, si se desea tener un algoritmo de clasificación preciso y rápido, se recomienda usar SVM.