UNIVERSIDAD
NACIONAL
DE COLOMBIA

Proyecto Final Sistemas Inteligentes Computacionales
Selección de variables para redes neuronales mlp

John Eder Posada Zapata - Jorge Luis Ordoñez Ospina - Hans Rivera Londoño

Universidad Nacional de Colombia
Manizales - Caldas

2020

Índice

Problemática

De acuerdo a diferentes análisis realizados en redes neuronales se ha determinado que en ocasiones es difícil seleccionar las variables más importantes que deben ingresar a la red neuronal para obtener un mejor resultado en su proceso de entrenamiento para la clasificación de clases.

Solución propuesta

Se ha creado este proyecto el cual permite aplicar algoritmos genéticos para la clasificación de variables que serán utilizadas para un mejor entrenamiento de una red neuronal y además esta red permite ser aplicada a diferentes problemas donde la solución es una clasificación binaria; adicionalmente este proyecto está desarrollado de tal manera que permite la parametrización de las diferentes características tanto del algoritmo genético como de la red neuronal.

Diseño y funcionamiento

A través de un esquema de varios archivos se mostrará el funcionamiento de este proyecto explicando cada uno por separado de la siguiente manera:

Preselec.py

En este archivo lo que hace es generar un valor booleano para cada variable que será almacenado dentro de una lista que posteriormente se usará para generar una probabilidad cuando sea ingresado al algoritmo genético, teniendo en cuenta el nivel de variación de cada una de las columnas del data frame. Esto se realiza tomando la resta entre sus cuartiles 1 y 3 (percentil 25 y 75) y comparándolo con el rango multiplicado por 10^ (- dígitos de la mediana de la variable), y con base a esto si los datos tienen una gran variación se les retornará un valor de False, puesto que, se le consideran datos dispersos y poco significativos. Para las columnas que esto no sea así se retornará True ya que se consideran variables importantes para el siguiente paso en el algoritmo genético.

```
if df3[f].quantile(0.75) -  df3[f].quantile(0.25) > rango * (1/(10**conteo)) :
  entra.append(False)
else:
  entra.append(True)
```

AgProyecto.py

En este archivo lo primero que se realiza es utilizar el vector de valores booleanos generado en el archivo preselec.py y generar una probabilidad entre un rango 0 y 1, utilizando la función sigmoidea, para cada columna dependiendo del valor booleano generado, con base a esto se generará una probabilidad entre [0, 0.5) si su valor es False y (0.5, 1] si es True.

```
def probaDeVar(entra): #determina la probabilidad para entregarle a la funcion evaluacion de cada variable
  if entra:
    proba= random.uniform(0,10)
  else:
    proba= random.uniform(-10, 0)

  return 1/(1+(math.e)**(-(proba)))
```

Luego de ser creada esta lista de porcentaje será enviado a la función evalúa que la utilizará para calcular el fitness que se utilizará para determinar cuál será nuestro mejor individuo que en este caso no dirá cuáles son las mejores columnas de nuestro data frame para utilizarse después en la red neuronal.

```
def evalua(n,x,poblIt,varDec,listaProb):
  fitness= np.empty((n))
  suma=0
  total=0
  # -> 1 * 0,32 + 0* 0,2 + .........
  for p in range(len(poblIt)):
    for j in range(len(listaProb)):
      suma += (poblIt[p][j] * listaProb[j])
    fitness[p]=suma
    total+=suma
    suma=0
  return fitness,total
```

Ejemplo de individuo.

```
[0,1,1,1,0,0,0,1,0,1,1,1,1,0,1,0,0,0,0,0,1]
```

nn.py

En este archivo se utiliza la lista entregada por el algoritmo genético que determinó cuáles serán las columnas que se usarán en el entrenamiento. A través del framework pytorch se realizará la creación de la red neuronal para una clasificación binaria que utilizara las columnas anteriormente seleccionadas.

```
for i in range(len(columDF)-1):
    if listVar[i] == 0:
        df3 = df3.drop([columDF[i]], axis=1)
```

```
D_out = 1
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
    torch.nn.Sigmoid()
)
```

```
for e in range(1, epochs+1):
    iteraVec.append(e)
    # forward
    y_pred = model(X_t)

    # loss
    loss = criterion(y_pred, Y_t)
    l.append(loss.item())

    vecerror.append(np.mean(l))

    # ponemos a cero los gradientes
    optimizer.zero_grad()

    # Backprop (calculamos todos los gradientes automáticamente)
    loss.backward()

    # update de los pesos
    optimizer.step()
```

main.py

En este archivo se realiza la unión de los archivos anteriormente mencionados, y poder obtener una serie de resultados que serán mostrados por consola y a través de las gráficas que poseen el No. de iteración (época) y el error cuadrático medio. Las gráficas serán guardadas en un archivo pdf que se localizará dentro de la carpeta raíz con el nombre de graf.pdf.

```
def pruebas(cantidadPruebas):···

def mejor(mPerdidas,mColumnas):···

def table(mPerdidas,mColumnas):···

def saveFig(mfig):···

#datos que se pasan entre algoritmos
entra = preselec.prese(nomArch,ColY)
mperd, mcol,mfig = pruebas(cantidadPruebas)

#muestra los datos y guarda los gráficos
table(mperd,mcol)
mejor(mperd,mcol)
saveFig(mfig)
```

Pruebas

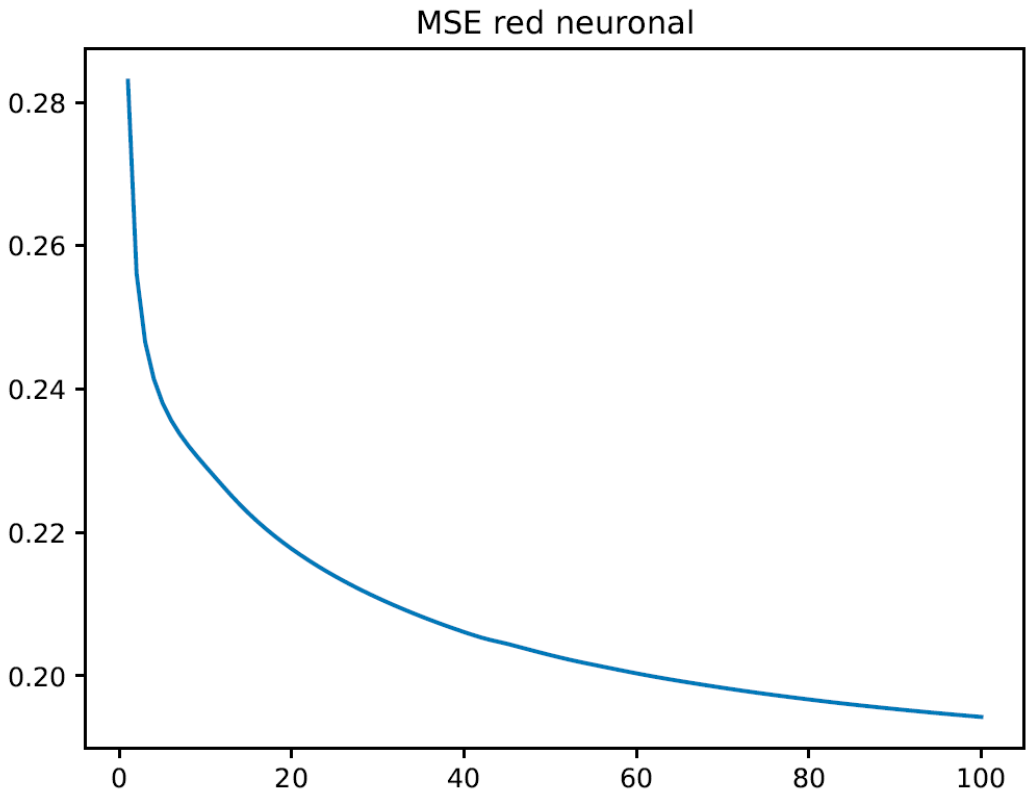
Utilizando el data frame Software Defect Prediction (https://www.kaggle.com/semustafacevik/software-defect-prediction?select=about+JM1+Dataset.txt). Se busca realizar la clasificación entre un software que tiene defectos y otro que no lo tiene.
En la siguiente imagen se muestra las columnas del data frame usado, con el nombre de todas sus columnas.

```
% 7. Attribute Information:
%
%      1. loc             : numeric % McCabe's line count of code
%      2. v(g)            : numeric % McCabe "cyclomatic complexity"
%      3. ev(g)           : numeric % McCabe "essential complexity"
%      4. iv(g)           : numeric % McCabe "design complexity"
%      5. n               : numeric % Halstead total operators + operands
%      6. v               : numeric % Halstead "volume"
%      7. l               : numeric % Halstead "program length"
%      8. d               : numeric % Halstead "difficulty"
%      9. i               : numeric % Halstead "intelligence"
%     10. e               : numeric % Halstead "effort"
%     11. b               : numeric % Halstead
%     12. t               : numeric % Halstead's time estimator
%     13. lOCode          : numeric % Halstead's line count
%     14. lOComment       : numeric % Halstead's count of lines of comments
%     15. lOBlank         : numeric % Halstead's count of blank lines
%     16. lOCodeAndComment: numeric
%     17. uniq_Op         : numeric % unique operators
%     18. uniq_Opnd       : numeric % unique operands
%     19. total_Op        : numeric % total operators
%     20. total_Opnd      : numeric % total operands
%     21: branchCount     : numeric % of the flow graph
%     22. defects         : {false,true} % module has/has not one or more
%                                        % reported defects
```

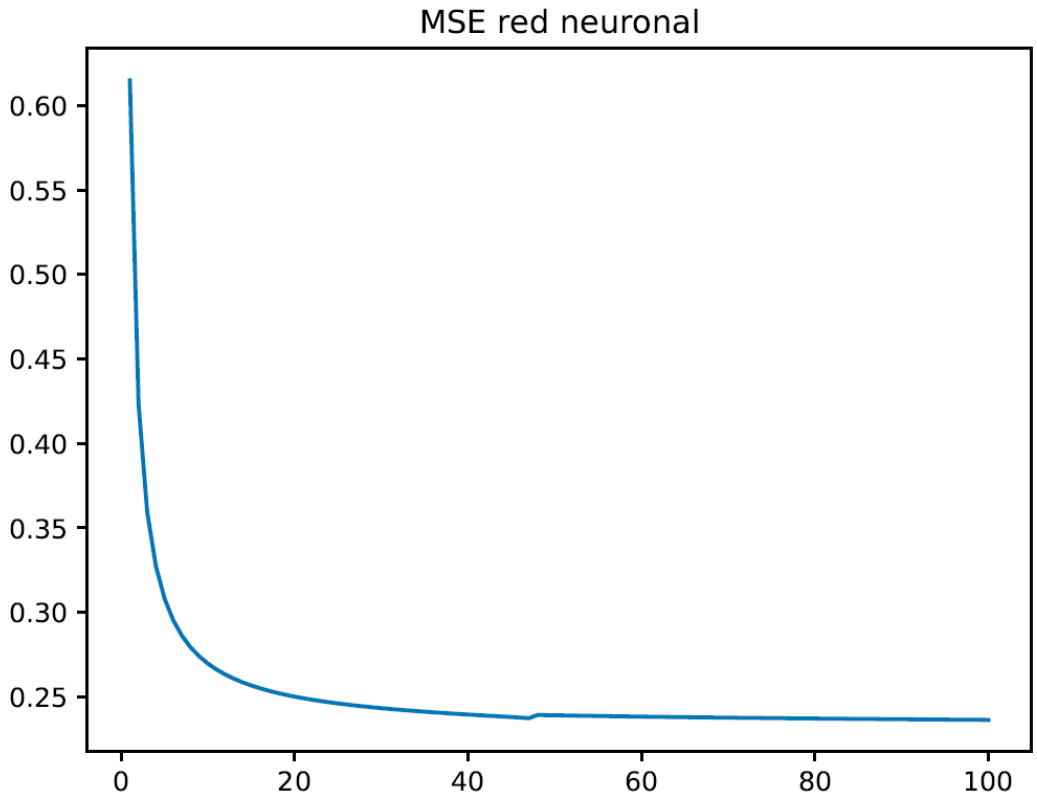
Luego de usar nuestras soluciones se han encontrado los siguientes resultados

```
Tabla:

Prueba    Error                        Columnas
   [1]        0.194             Index(['v(g)', 'iv(g)', 'l', 'd', 'b', 'lOCode', 'lOComment', 'lOBlank',
       'locCodeAndComment', 'branchCount'],
      dtype='object')


   [2]        0.236             Index(['ev(g)', 'l', 'd', 'b', 't', 'lOCode', 'lOComment', 'lOBlank',
       'locCodeAndComment', 'branchCount'],
      dtype='object')


   [3]        0.240             Index(['ev(g)', 'iv(g)', 'l', 'd', 'b', 't', 'lOCode', 'lOComment', 'lOBlank',
       'branchCount'],
      dtype='object')


   [4]        0.229             Index(['v(g)', 'ev(g)', 'l', 'd', 'b', 't', 'lOCode', 'lOComment', 'lOBlank',
       'branchCount'],
      dtype='object')


   [5]        0.223             Index(['v(g)', 'ev(g)', 'l', 'b', 't', 'lOCode', 'lOComment', 'lOBlank',
       'locCodeAndComment', 'branchCount'],
      dtype='object')



el mejor error es 0.194  con las columnas Index(['v(g)', 'iv(g)', 'l', 'd', 'b', 'lOCode', 'lOComment', 'lOBlank',
       'locCodeAndComment', 'branchCount'],
      dtype='object')
```

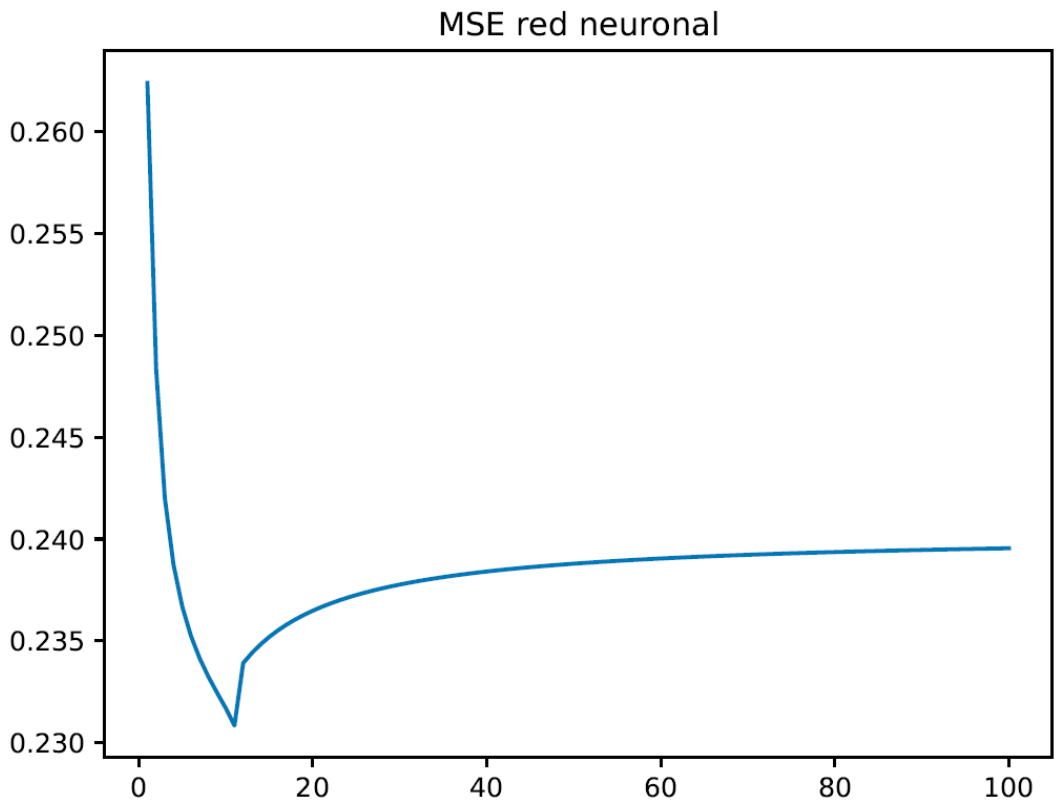
Iteración 1



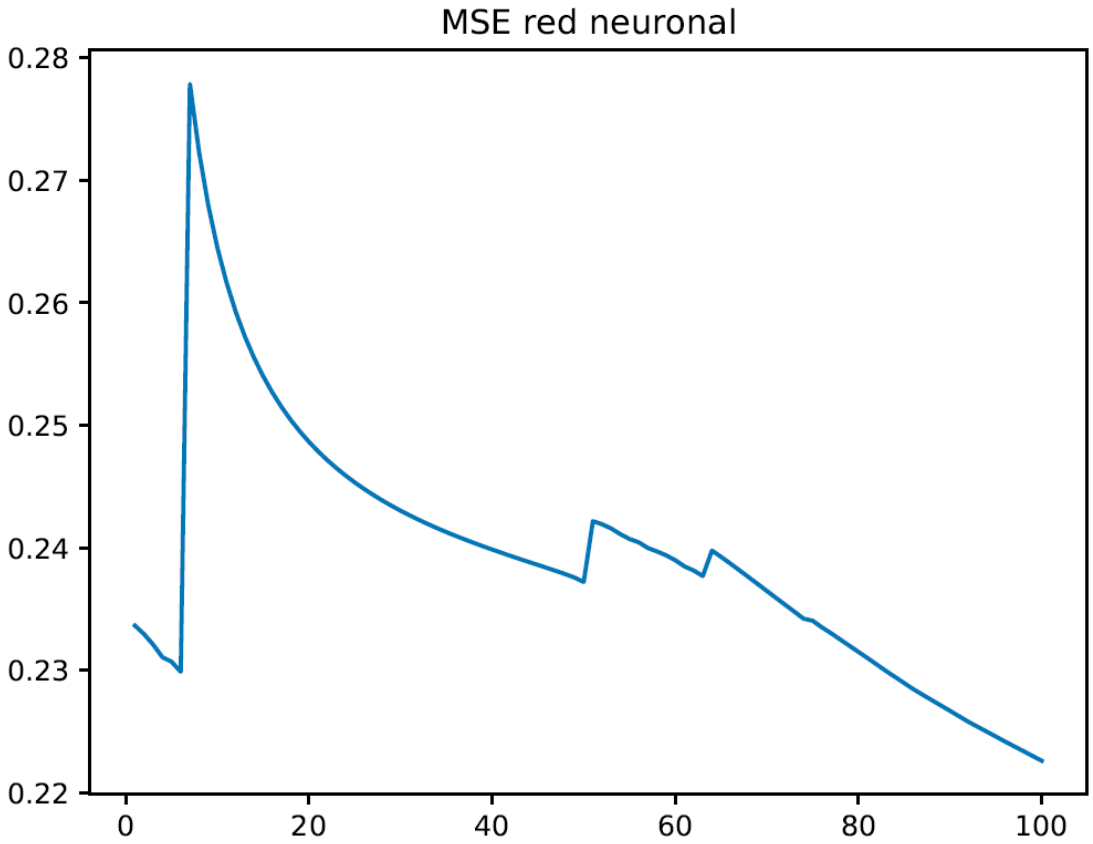
Iteración 2


MSE red neuronal
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
20
40
60
80
100

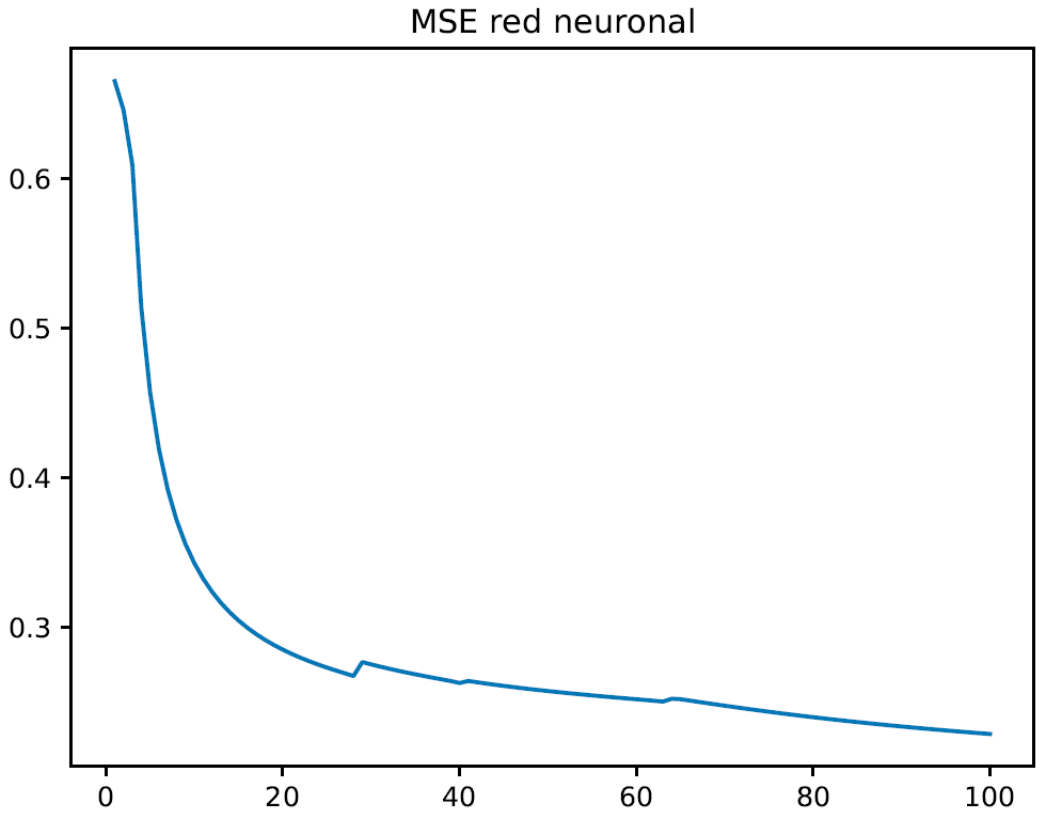
Iteración 3


MSE red neuronal
0.260
0.255
0.250
0.245
0.240
0.235
0.230
0
20
40
60
80
100

Iteración 4



Iteración 5



Requerimientos

1. Poseer Python 3.7 o superior.

2. Se necesitan las siguientes librerías para el funcionamiento correcto del proyecto:
   - pytorch=1.7.0
   - scikit-learn=0.23.2
   - scipy=1.5.2
   - pandas=1.1.1
   - numpy=1.19.1
   - numpy-base=1.19.1
   - matplotlib=3.2.2
   - matplotlib-base=3.2.2

   Para instalar dichas librerías, se requiere ejecutar el siguiente comando:
   `py -m pip install -r requirements.txt` , adicionalmente ejecutar
   `pip install torch==1.7.0+cpu torchvision==0.8.1+cpu torchaudio===0.7.0 -f https://download.pytorch.org/whl/torch_stable.html`

Uso

Luego de cumplir con los requerimientos, se deje ejecutar el archivo main.py a través de la línea de comandos con la instrucción siguiente: Python main.py

Si desea cambiar algún parámetro, esto lo encontrará en el archivo main.py, allí encontrará una descripción de cada parámetro que es posible cambiar.

Restricciones

Algunas restricciones que presenta son:

- Todas sus columnas deben de ser numéricas.

- El data frame no puede ser clustering, ya que nuestra red neuronal hace uso de la columna de Y real que son los valores que tienen en la vida real el sistema que se está modelando.

Bibliografía

- PEDRO, J. (2020). El Perceptrón. Retrieved 29 November 2020, from https://sensioai.com/blog/012_perceptron1

- PEDRO, J. (2020). El Perceptrón Multicapa - Introducción. Retrieved 29 November 2020, from https://sensioai.com/blog/023_mlp_backprop

- Automatiza decisiones con Inteligencia Artificial en Excel (instructivo para novatos. Parte 1). (2020). Retrieved 29 November 2020, from https://www.rankia.co/blog/comstar/3618510-automatiza-decisiones-inteligencia-artificial-excel-instructivo-para-novatos-parte-1

- ¿Cómo seleccionar las variables adecuadas para tu modelo? | Máxima Formación. (2020). Retrieved 29 November 2020, from https://www.maximaformacion.es/blog-dat/como-seleccionar-las-variables-adecuadas-para-tu-modelo/

- Best subset selection — STATS 202. (2020). Retrieved 29 November 2020, from https://web.stanford.edu/class/stats202/notes/Model-selection/Best-subset.html

- Ana González, Vidal. (2020). Retrieved 29 November 2020, from http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_1263.pdf

- (2020). Retrieved 29 November 2020, from https://www.datos.gov.co/browse?category=Ciencia%2C+Tecnolog%C3%ADa+e+Innovaci%C3%B3n&sortBy=newest

- DIFERENCIA ENTRE CLASIFICACIÓN Y REGRESIÓN EN MACHINE LEARNING. (2020). Retrieved 29 November 2020, from https://ingenierobeta.com/clasificacion-vs-regresion-machine-learning/#:~:text=La%20regresi%C3%B3n%20tiene%20el%20objetivo,clasificaci%C3%B3n%20los%20valores%20son%20discretos

- Heras, J. (2020). Regularización Lasso L1, Ridge L2 y ElasticNet - IArtificial.net. Retrieved 29 November 2020, from https://www.iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/#Regularizacion_Lasso_(L1)

- Corral Sastre, Antonio, (2020). Retrieved 29 November 2020, from https://riunet.upv.es/bitstream/handle/10251/111512/Corral%20-%20Desarrollo%20y%20Evaluaci%C3%B3n%20de%20Algoritmos%20Gen%C3%A9ticos%20Multiobjetivo.%20Aplicaci%C3%B3n%20al%20problema%20del%20viajante..pdf?sequence=1&isAllowed=y

- Damián Jorge, Matich, (2020). Retrieved 29 November 2020, from https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

- Tutorial básico de redes neurales con Python - VIDEO — gidahatari. (2020). Retrieved 29 November 2020, from https://gidahatari.com/ih-es/tutorial-basico-de-redes-neurales-con-python-video
- python-engineer/pytorchTutorial. (2020). Retrieved 29 November 2020, from https://github.com/python-engineer/pytorchTutorial/blob/master/07_linear_regression.py

- Sequential — PyTorch 1.7.0 documentation. (2020). Retrieved 29 November 2020, from https://pytorch.org/docs/stable/generated/torch.nn.Sequential.html

- Deep Learning with PyTorch. (2020). Retrieved 29 November 2020, from https://books.google.com.co/books?hl=es&lr=&id=DOIODwAAQBAJ&oi=fnd&pg=PP1&dq=using+sigmoid+for+neural+networks+in+pytorch&ots=koY_e2aDMj&sig=dBuY-tr2i0eqt9zRx8TPKbTaXv4#v=onepage&q=using%20sigmoid%20for%20neural%20networks%20in%20pytorch&f=false

- Studies!), G. (2020). Pytorch Introduction | How to Build A Neural Network. Retrieved 29 November 2020, from https://www.analyticsvidhya.com/blog/2019/01/guide-pytorch-neural-networks-case-studies/

- Análisis de clasificación binaria | Interactive Chaos. (2020). Retrieved 29 November 2020, from https://www.interactivechaos.com/manual/tutorial-de-deep-learning/analisis-de-clasificacion-binaria

- (2020). Retrieved 29 November 2020, from https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/#:~:text=La%20funci%C3%B3n%20de%20activaci%C3%B3n%20se,con%20ello%20el%20coste%20computacional

- torch. argmax — PyTorch 1.7.0 documentation. (2020). Retrieved 29 November 2020, from https://pytorch.org/docs/stable/generated/torch.argmax.html

- Multipage PDF — Matplotlib 3.1.0 documentation. (2020). Retrieved 29 November 2020, from https://matplotlib.org/3.1.0/gallery/misc/multipage_pdf.html

- Joshi, S. (2020). Cómo guardar parcelas como archivo PDF en Matplotlib. Retrieved 29 November 2020, from https://www.delftstack.com/es/howto/matplotlib/how-to-save-plots-as-pdf-file-in-matplotlib/#:~:text=Los%20gr%C3%A1ficos%20generados%20a%20partir,PDF%2C%20utilizamos%20la%20clase%20PdfPages%20

- Software Defect Prediction. (2020). Retrieved 29 November 2020, from https://www.kaggle.com/semustafacevik/software-defect-prediction?select=about+JM1+Dataset.txt