Tarea 7: Aplicación perceptrón multicapa

En la tabla adjunta se muestran los datos de 1000 clientes que solicitaron créditos a un banco dado. La última columna muestra la información de los clientes que cayeron en mora en algún momento del período del crédito. El monto máximo de crédito que puede asignarse son $300,000 y la antigüedad laboral máxima que se toma en cuenta para asignar el crédito es de 15 años (es decir, antigüedades mayores ya no generan más posibilidad de ser aprobado).

Se busca una relación entre la información presentada (que se obtiene al contratar el crédito) y la posibilidad de que el cliente caiga en mora en algún momento del plazo.

Entrene un perceptrón multicapa para encontrar una relación tomando como entradas el monto solicitado (normalizado), la carga que implica al salario el pago de la mensualidad y la antigüedad laboral al contratar (normalizada).

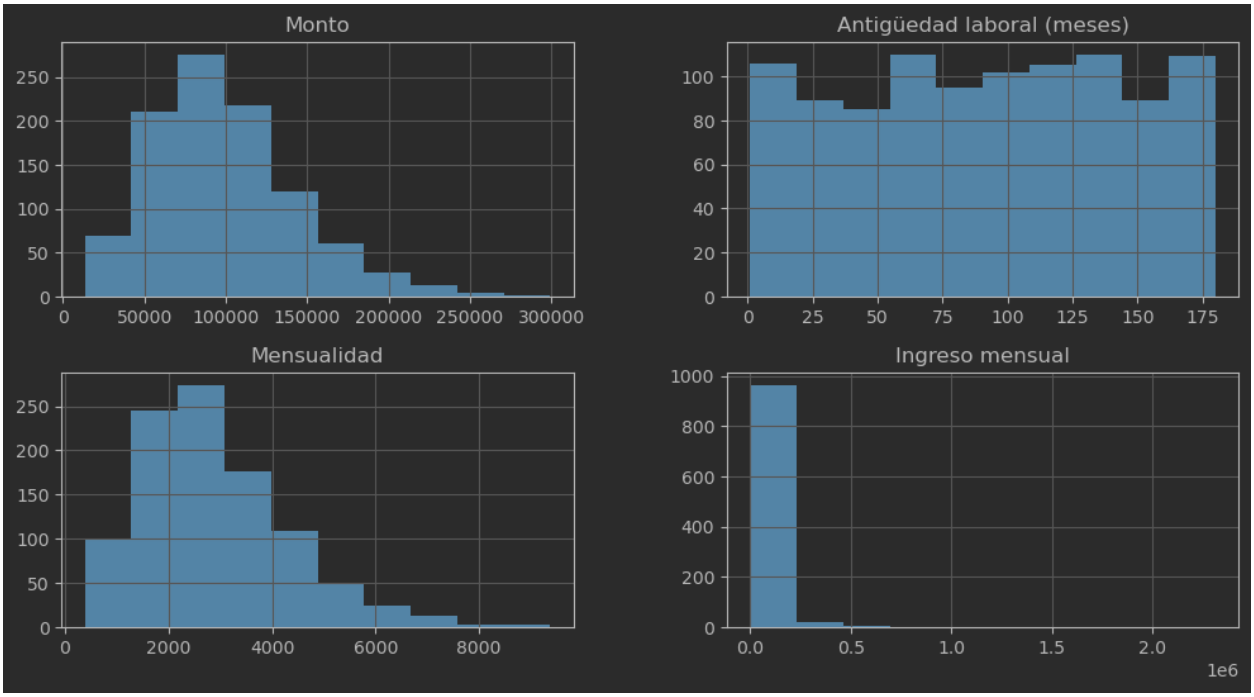
Utilice 70-30 de relación entrenamiento-prueba y calcule el accuracy.

Desarrollo de actividad

El número de neuronas ocultas es a su criterio.

Para el proyecto se utilizarán las variables de monto, antigüedad, mensualidad e ingreso mensual para predecir si un cliente caerá en mora o no.

Se presentan los histogramas a continuación

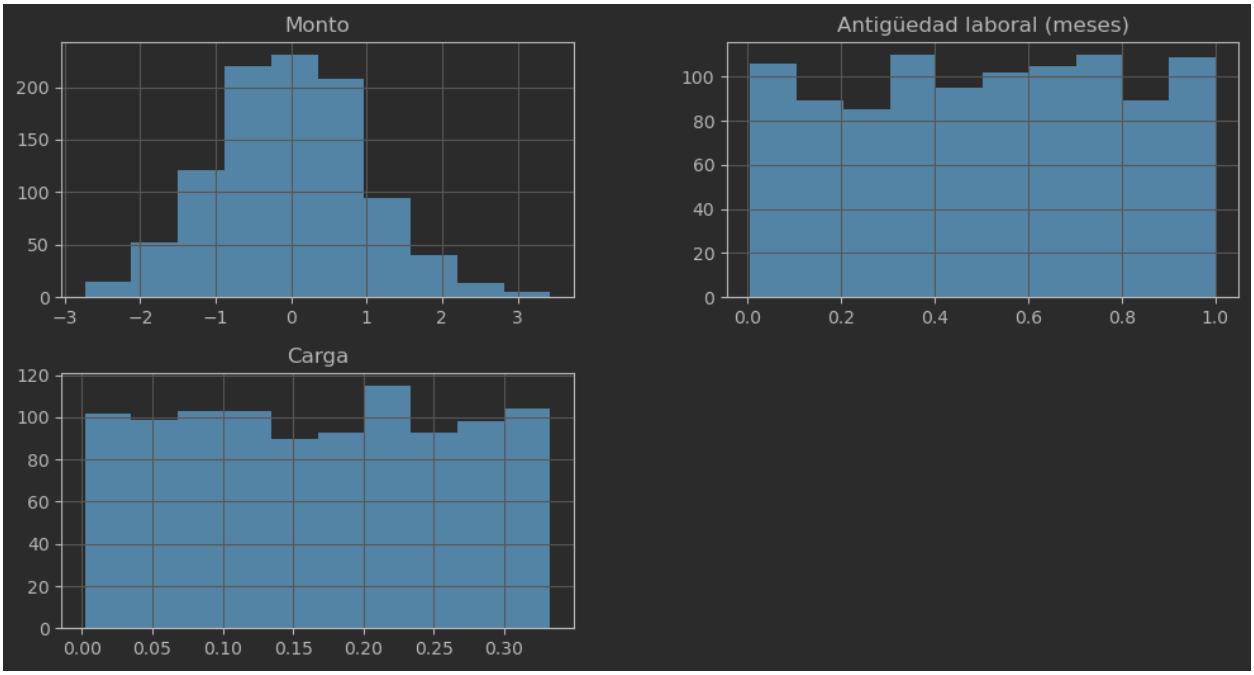


Estas variables serán normalizadas de la siguiente manera:

- Al monto se le aplicará la raíz cuadrada y a esos valores se le restará la media y dividirá la desviación estándar

- La variable de antigüedad se dividirá entre el valor máximo de la antigüedad laboral
- Finalmente se creara una variable de carga que considerara la mensualidad de pago y el ingreso mensual

Tras el tratamiento de los datos se presentan las variables transformadas donde monto presenta una distribución cercana a la normal. Antigüedad y carga tienen un comportamiento similar a una distribución uniforme



Una vez transformados los datos se inicializan las variables de la siguiente manera para la red neuronal:

Variables	
Proporción train/test	70/30
Alpha	0.01
Neuronas	5
Épocas	1500

Se hace la corrida con los datos de entrenamiento obteniendo la siguiente información

Corrida de entrenamiento	
Tiempo de entrenamiento	35.21 segundos
Accuracy Train	98.85%

Con esa información se hace la validación para el los datos de test obteniendo la siguiente información

Corrida de prueba	
Accuracy Test	98%

Anexo 1 Codigo empleado

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
from pytictoc import TicToc
t = TicToc()
df = pd.read_excel("PercMultAplicado.xlsx")
#%%
df.head()
#%%
df[["Monto","Antigüedad laboral (meses)","Mensualidad","Ingreso mensual"]].hist(figsize=(12,6))

#%%
df["Carga"] = df["Mensualidad"]/df["Ingreso mensual"]
df["Monto"] = df["Monto"]**(1/2)
df["Monto"] = (df["Monto"]-df["Monto"].mean())/df["Monto"].std()
df["Antigüedad laboral (meses)"] = df["Antigüedad laboral (meses)"]/max(df["Antigüedad laboral
(meses)"])
df["Mora"] = df["Mora"]=="SI"


#%%
dffin = df[["Monto","Antigüedad laboral (meses)","Carga","Mora"]]
dffin.head()
dffin.hist(figsize=(12,6))


#%%
Xt,xt,Yt,yt=train_test_split(dffin[["Monto","Antigüedad laboral
(meses)","Carga"]],dffin['Mora'],train_size=0.7)

#%%
#Inicializacion de variables
a= 1
alpha = 0.01
N = Xt.shape[1]#inputs
M = 1 #Salidas conocidas
Q = len(Xt)#Patrones de aprendizaje
L = 4 #Neuronas
epocas = 1500
wh = np.random.uniform(-1, 1, (L, N))#Vector de pesos
wo = np.random.uniform(-1, 1, (M, L))
```

```
x = np.float64(Xt.to_numpy())
d = np.float64(Yt.to_numpy())
y = np.zeros([Q,M])

#%%
c = 0
t.tic()
for i in range(epocas):
   #Forward
   for i in range(Q):
      net_h = wh @ x[i].transpose()
      y_h = np.reshape(1/(1+np.exp(-a*net_h)),(L,1))
      net_o = wo @ y_h
      y = 1 / (1 + np.exp(-a*net_o))
      #Backward
      d_o= ( np.reshape(d[i],(M,1)) - y)*y* (1 - y)
      d_h = y_h * (1 - y_h) * (np.transpose(wo) @ d_o)

      wo += alpha * d_o @ y_h.transpose()
      wh += alpha * d_h @ np.reshape(x[i], (1, N))
   c +=1
   if  np.linalg.norm(d_o) < 0.0001:
      break

t.toc()
#%%
#Originales
res_o = []
for i in range(Q):
   net_h = wh @ x[i].transpose()
   y_h = np.reshape(1/(1+np.exp(-a*net_h)),(L,1))
   net_o = wo @ y_h
   y = 1 / (1 + np.exp(-a*net_o))
   res_o = np.append(res_o,np.round(y))

res_o = np.reshape(res_o,(Q,M))


#%%
accuracy = accuracy_score(Yt,res_o)
accuracy

#%%
testlen = len(xt)
```

```
xtest = np.float64(xt.to_numpy())
ytest = np.float64(yt.to_numpy())
restest = []
for i in range(testlen):
    net_h = wh @ xtest[i].transpose()
    y_h = np.reshape(1/(1+np.exp(-a*net_h)),(L,1))
    net_o = wo @ y_h
    y = 1 / (1 + np.exp(-a*net_o))
    restest = np.append(restest,np.round(y))

restest = np.reshape(restest,(testlen,M))
accuracy_score(ytest,restest)
```