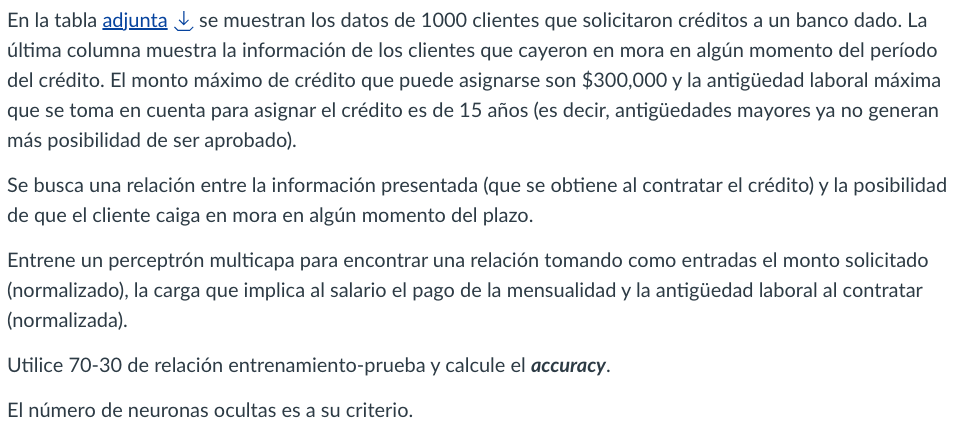
Alumno: Ricardo De León



## **Desarrollo:**

Identificamos las columnas requeridas en el problema y analizamos su distribución

**Para Monto:**

# **Observamos la gráfica de densidad de la columna monto**

plt.hist(df['Monto'],density=True, bins=30)

Chart, histogram

Description automatically generated

**# Tratamos de hacer la curva de más parecida a una desviacion normal**

monto = np.sqrt(np.asarray(df['Monto']))

**# Normalizamos lo datos con min y max**

def normalizar(data):

return (data - data.min())/ (data.max() - data.min())

monto\_norm = normalizar(monto)

plt.hist(monto\_norm,density=True, bins=30)

Chart, histogram

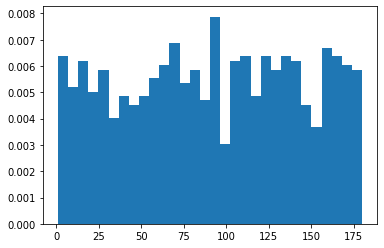
Description automatically generated

**Para Edad Laboral**

# **Observamos la gráfica de densidad de la columna Edad Laboral**

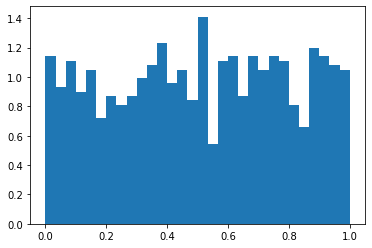
edad\_laboral = np.asarray(df.iloc[:,6])

plt.hist(edad\_laboral,density=True, bins=30)



# **Normalizamos lo datos con min y max**

edad\_laboral\_norm = normalizar(edad\_laboral)



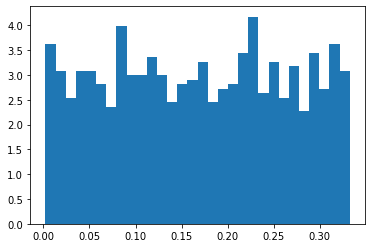
Observamos que la gráfica es casi idéntica, pero en una escala diferente ya que la distribución es más uniforme.

**Para Salario**

# **Observamos la gráfica de densidad de la columna Edad Laboral**

carga\_salario = np.asarray(df['Mensualidad']/df.iloc[:,5])

plt.hist(carga\_salario,density=True, bins=30)



**# Normalizamos lo datos con min y max**

carga\_salario\_norm = normalizar(carga\_salario)

plt.hist(carga\_salario\_norm,density=True, bins=30)

Chart, histogram

Description automatically generated

Observamos que la gráfica es casi idéntica, pero en una escala diferente ya que la distribución es más uniforme.

* Una vez homogeneizados nuestros datos comenzamos a unir nuestro dataset final

mora = np.asarray(df['Mora'].map({'SI': 1, 'NO': 0}))

df\_norm = pd.DataFrame({'Monto' : monto\_norm,

'Salario' : carga\_salario\_norm,

'eLaboral' : edad\_laboral\_norm,

'Mora': mora})

* Hacemos la estratificación de la muestra de un 70-30 definiendo dataset de sus proporciones train (70) y test (30).

**# Estratificando muestreo**

df\_train = df\_norm.groupby("Mora").sample(frac=0.7, random\_state=1) # usar andom\_state=1 para tomar las mismas muestras siempre

df\_test = df\_norm[~df\_norm.index.isin(df\_train.index)]

* Por último ejecutamos nuestro train y test y observamos nuestro acurracy

X\_70,D\_70 = get\_X\_D(df\_train)

errors\_70, wh\_70, wo\_70 = perceptron\_m(X\_70, D\_70)

print('Comprobación de datos entrenados para el 70 de muestreo')

result\_70 = []

for i in range(X\_70.shape[0]):

\_, y = forward(X\_70, wo\_70, wh\_70, i)

result\_70.append(round(y[0]))

accuracy\_70 = np.mean(D\_70 == result\_70)

print(f'accuracy 70% sample {round(accuracy\_70 \* 100, 2)}%')

plt.plot(errors\_70)

plt.xlabel("Iteraciones 70")

plt.ylabel("Errores 70")

plt.title("Training Errores 70")

plt.show()

X\_30,D\_30 = get\_X\_D(df\_test)

errors\_30, wh\_30, wo\_30 = perceptron\_m(X\_70, D\_70)

print('Comprobación de datos entrenados para el 30 de muestreo')

result\_30 = []

for i in range(X\_30.shape[0]):

\_, y = forward(X\_30, wo\_30, wh\_30, i)

result\_30.append(round(y[0]))

accuracy\_30 = np.mean(D\_30 == result\_30)

print(f'accuracy 30% sample {round(accuracy\_30 \* 100, 2)}%')

plt.plot(errors\_30)

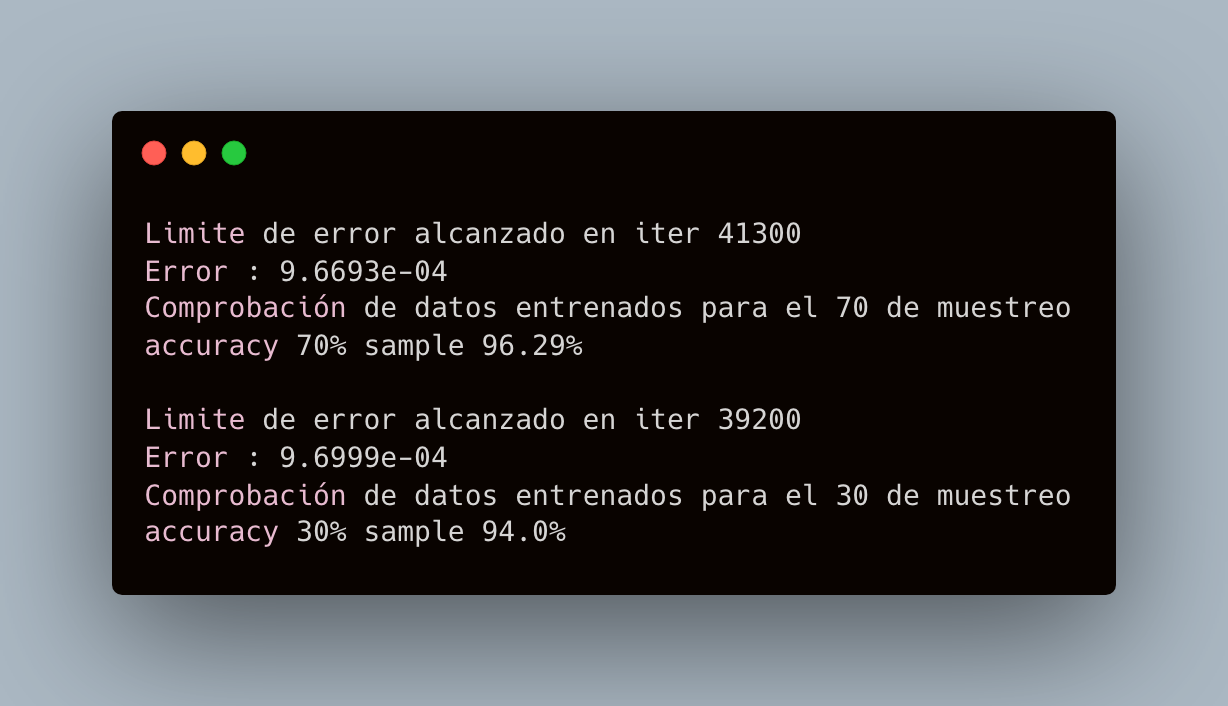
plt.xlabel("Iteraciones 30")

plt.ylabel("Errores 30")

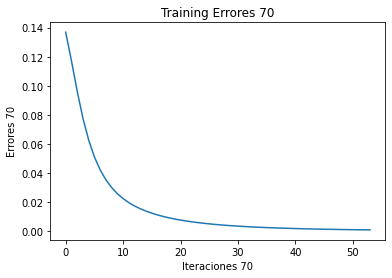
plt.title("Training Errores 30")

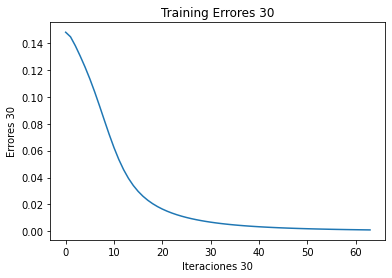
plt.show()

* Resultados



**Gráficas de errores**





* Código



