|  |  |
| --- | --- |
| Título  Artificial Neural Network | |
| Código  # Artificial Neural Network  # Importing the libraries  import numpy as np  import pandas as pd  import tensorflow as tf  tf.\_\_version\_\_  # Part 1 - Data Preprocessing  # Importing the dataset  dataset = pd.read\_csv('Churn\_Modelling.csv')  X = dataset.iloc[:, 3:-1].values  y = dataset.iloc[:, -1].values  print(X)  print(y)  # Encoding categorical data  # Label Encoding the "Gender" column  from sklearn.preprocessing import LabelEncoder  le = LabelEncoder( )  X[:, 2] = le.fit\_transform(X[:, 2])  print (X)  # One Hot Encoding the "Geography" column  from sklearn.compose import ColumnTransformer  from sklearn.preprocessing import OneHotEncoder  ct = ColumnTransformer (transformers=[('encoder', OneHotEncoder(), [1])],  remainder= 'passthrough')  X = np.array(ct.fit\_transform(X))  print(X)  # Splitting the dataset into the Training set and Test set  from sklearn.model\_selection import train\_test\_split  X\_train,X\_test,y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  # Feature Scaling  from sklearn.preprocessing import StandardScaler  sc = StandardScaler()  X\_train = sc.fit\_transform(X\_train)  X\_test = sc.transform(X\_test)  # Part 2 - Building the ANN  # Initializing the ANN  ann = tf.keras.models.Sequential()  # Adding the input layer and the first hidden layer  ann.add(tf.keras.layers.Dense(units=6, activation='relu'))  # Adding the second hidden layer  ann.add(tf.keras.layers.Dense(units=6, activation='relu'))  # Adding the output layer  ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))  # Part 3 - Training the ANN  # Compiling the ANN  ann.compile(optimizer = 'adam', loss = 'binary \_crossentropy', metrics =  ['accuracy'])  # Training the AN on the Training set  ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 0)  # Part 4 - Making the predictions and evaluating the model  # Predicting the result of a single observation  print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000,2, 1, 1,  50000]])) > 0.5)  # Predicting the Test set results  y\_pred = ann.predict(X\_test)  y\_pred = (y\_pred > 0.5)  print(np.concatenate((y\_pred.reshape(len(y\_pred),1),  y\_test.reshape(len(y\_test),1)),1))  # Making the Confusion Matrix  from sklearn.metrics import confusion\_matrix, accuracy\_score  cm = confusion\_matrix(y\_test, y\_pred)  print(cm)  accuracy\_score(y\_test, y\_pred) | |
| Ejecución | Código QR del repositorio en GitHub |