Práctica 3
Clasificadores k-NN y regresión logística

Mario Emilio Jiménez Vizcaíno
A01173359@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

Jesus Abraham Haros Madrid
A01252642@itesm.mx
Tecnológico de Monterrey
Ingeniería en Tecnologías Computacionales
Monterrey, N.L., México

ABSTRACT

TODO

1 INTRODUCCIÓN

TODO

2 CONCEPTOS PREVIOS

- Programación básica en los lenguajes R y Python
- Conocimiento de las librerías scikit-learn, pandas y numpy
- Conocimientos de estadística y de regresión logística

3 METODOLOGÍA

3.1 Datasets

Para comparar ambas implementaciones utilizamos dos datasets, expuestos a continuación:

3.1.1 Dataset DEFAULT.

Este dataset está compuesto por 10,000 filas, cada una representa un cliente de un banco que puede o no cumplir con los pagos de su tarjeta de crédito (columna "default"). De cada cliente tenemos la siguiente información:

- Columna "default": Tiene los valores "Yes"/"No", representa si la persona realizó el pago mínimo a su tarjeta de crédito.
- Columna "student": Valores "Yes"/"No", representa si el cliente es un estudiante en ese momento.
- Columna "balance": Número decimal positivo que representa el balance de la tarjeta de crédito del cliente. Promedio de 835.4, números en el rango [0, 2654.3].
- Columna "income": Número decimal positivo que representa los ingresos que tiene el cliente. Promedio de 33517, números en el rango [772, 73554]

De este dataset, nuestro objetivo es predecir la columna "default" a partir de los otros tres parámetros, y como preparación cambiamos los valores de "student" ("Yes"/"No") a valores 1 y 0 respectivamente.

3.1.2 Dataset GENERO.

Este dataset representa las mediciones de peso y altura de 10,000 personas, en conjunto con el género de la persona a la que se realizaron las medidas. Las columnas son:

- "Gender": Valores "Male"/"Female", el género de la persona a la que le corresponde esta fila de mediciones.
- "Height": La altura de la persona en pulgadas, promedio de 66.37, en el rango [54.26 y 79.00].
- "Weight": El peso de la persona en libras, promedio de 161.4, en el rango [64.7, 270.0].

La columna objetivo seleccionada de este dataset fue el género ya que tiene dos clasificaciones.

3.2 Clasificación con k-NN

3.2.1 Dataset DEFAULT.

TODO

El código fuente de este ejemplo se encuentra en el apéndice A.

3.2.2 Dataset GENERO.

TODO

El código fuente puede ser encontrado en el apéndice B..

3.3 Regresión logística

Para la primera parte de la práctica, en la que utilizamos la implementación de *sci-kit learn*, seleccionamos la clase *sklearn.linear_model.LogisticRegression*[1] para nuestros scripts.

3.3.1 Dataset DEFAULT.

Para este dataset primero leemos el archivo CSV a un *Dataframe* de *pandas*, transformamos las columnas "default" y "student" para que contengan valores booleanos y enteros respectivamente, seleccionamos las columnas que nos servirán como variables independientes (columnas "student", "balance" e "income") y variable dependiente (columna "default"). Después partimos las filas del dataset en una porción del 80% que usaremos para entrenar el modelo, y otra porción del 20% para probarlo.

Instanciamos el modelo de regresión de *sklearn*, lo entrenamos con los datos y después predecimos la variable dependiente con el modelo para así compararlo con los datos reales de prueba, usando una medida de tasa de precisión y la matriz de confusión.

El código de este ejemplo puede se encuentra en el apéndice C.

3.3.2 Dataset GENERO.

Para este dataset realizamos un procedimiento similar: leer el dataset para crear un *Dataframe*, seleccionar las columnas de variables independientes ("Height" y "Weight") y la dependiente ("Gender"), dividir el dataset en 80%/20%, entrenar el modelo, y predecir la variable dependiente para los datos de prueba, para así comparar estos con los datos reales.

El código para esta sección se encuentra en el apéndice D.

4 RESULTADOS

4.1 Clasificación con k-NN

4.1.1 Dataset DEFAULT.

TODO

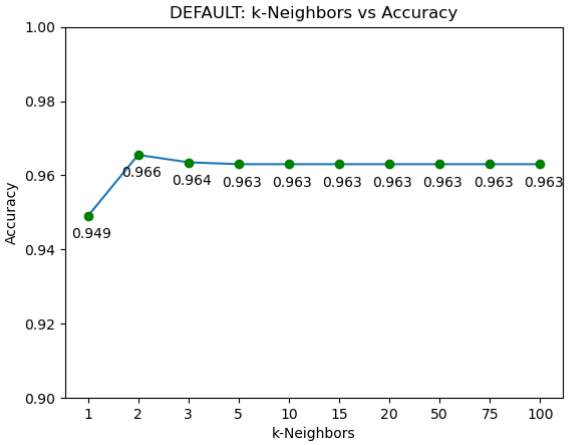

Figure 1: Variación del número de vecinos elegidos contra la precisión en el dataset DEFAULT

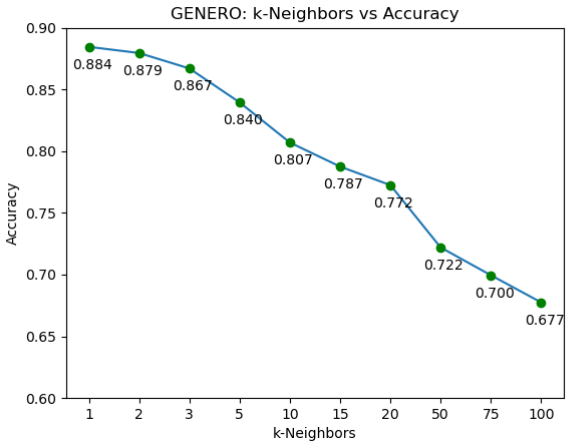

Figure 3: Variación del número de vecinos elegidos contra la precisión en el dataset GENERO

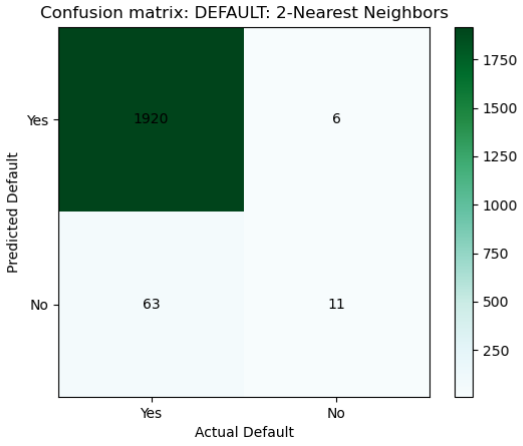

Figure 2: Matriz de confusión del dataset DEFAULT con 2-Nearest Neighbors

4.1.2 Dataset GENERO.

TODO

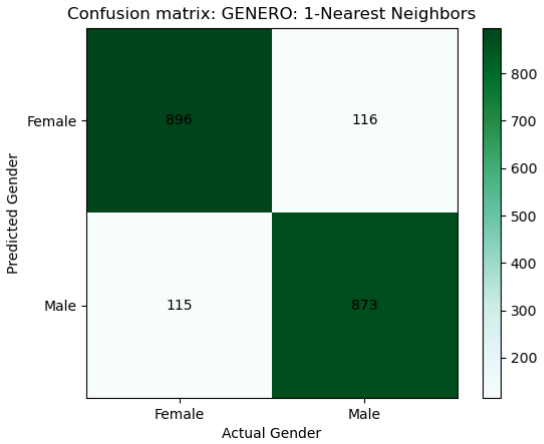

Figure 4: Matriz de confusión del dataset GENERO con 1-Nearest Neighbors

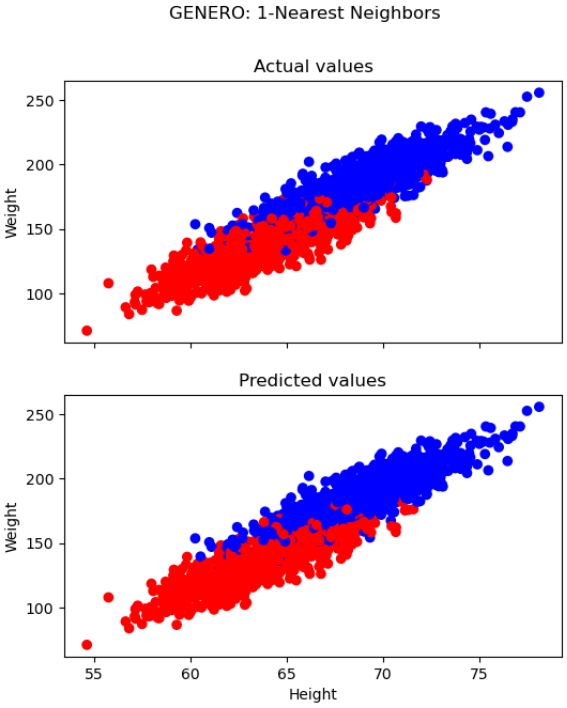

Figure 5: Gráfica de dispersión del dataset GENERO con 1-Nearest Neighbors

4.2 Regresión logística

4.2.1 Dataset DEFAULT.

TODO

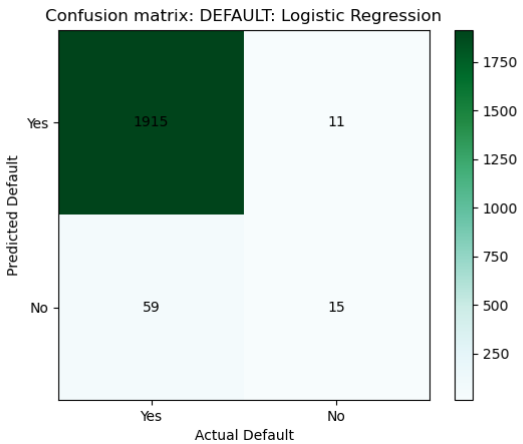

Figure 6: Matriz de confusión del dataset DEFAULT con regresión logística

4.2.2 Dataset GENERO.

TODO

Figure 7: Matriz de confusión del dataset GENERO con regresión logística

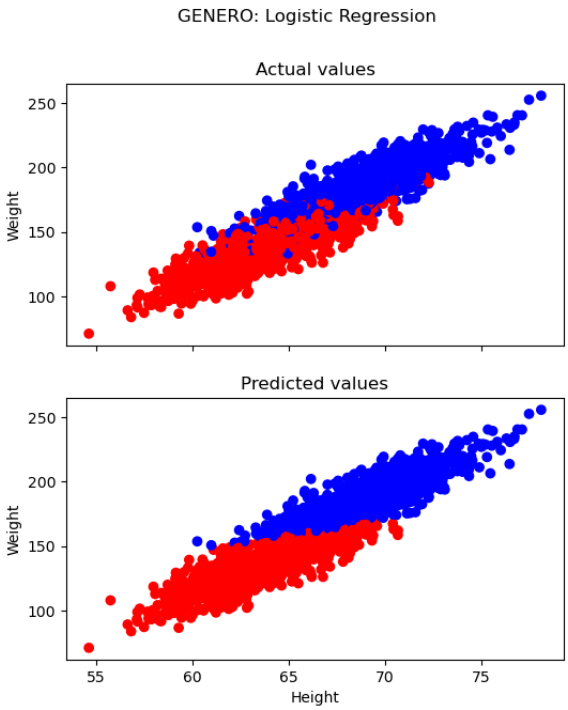

Figure 8: Gráfica de dispersión del dataset GENERO con regresión logística

5 CONCLUSIONES Y REFLEXIONES

TODO

5.1 Refrexión de Abraham

TODO

5.2 Reflexión de Mario

TODO

REFERENCES

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

A CÓDIGO DE REGRESIÓN LOGÍSTICA DEL DATASET DEFAULT

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

from graphs import graphConfusionMatrix, graphNeighborsAccuracy

print(" ~ Reading default.txt and generating train and test sets")
data = pd.read_csv('default.txt', sep=" ")
# Transform 'default' and 'student' columns from Yes/No to integers (1/0)
data["default"] = (data["default"] == "Yes").astype(int)
data["student"] = (data["student"] == "Yes").astype(int)
x = data.iloc[:, 1:4].values.reshape(-1, 3)
y = data.iloc[:, 0].values.reshape(-1, 1)
xTrain, xTest, yTrain, yTest = train_test_split(
    x, y, test_size=0.2, random_state=0)

print(" ~ Creating and testing the k-NN models for different values")
print(" →   neighbors   accuracy       confusion matrix")
k_neighbors_vals = [1, 2, 3, 5, 10, 15, 20, 50, 75, 100]
accuracy_vals = []
best_k_val = (-1, 0, [])  # k_neighbors, accuracy, conf_matrix
for val in k_neighbors_vals:
    nn = KNeighborsRegressor(n_neighbors=val, algorithm="brute")
    nn.fit(xTrain, yTrain)
    yPredicted = nn.predict(xTest).astype(int)
    accuracy = accuracy_score(yTest, yPredicted)
    accuracy_vals.append(accuracy)
    cm = confusion_matrix(yTest, yPredicted)
    print(" →    {:^9}    {:^8}    {}".format(val, accuracy, cm.tolist()))
    if best_k_val[1] < accuracy:
        best_k_val = (val, accuracy, cm)

graphNeighborsAccuracy(k_neighbors_vals, accuracy_vals,
                       "DEFAULT", "default_neighbors_acc.png")
graphConfusionMatrix(best_k_val[2], ["Yes", "No"], "Default",
                     "DEFAULT: {}-Nearest Neighbors".format(best_k_val[0]), "default_neighbors_cm.png")
```

B CÓDIGO DE CLASIFICACIÓN K-NN DEL DATASET GENERO

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

from graphs import graphConfusionMatrix, graphNeighborsAccuracy, graphGENERO

print(" ~ Reading genero.txt and generating train and test sets")
data = pd.read_csv('genero.txt')
x = data.iloc[:, 1:3].values.reshape(-1, 2)
y = data.iloc[:, 0].values.reshape(-1, 1)
xTrain, xTest, yTrain, yTest = train_test_split(
    x, y, test_size=0.2, random_state=0)


yIntToStr = np.vectorize(lambda x: ["Female", "Male"][int(x)])

print(" ~ Creating and testing the k-NN models for different values")
print(" →   neighbors   accuracy       confusion matrix")
k_neighbors_vals = [1, 2, 3, 5, 10, 15, 20, 50, 75, 100]
accuracy_vals = []
best_k_val = (-1, 0, [], [])  # k_neighbors, accuracy, conf_matrix, yPredicted
for val in k_neighbors_vals:
    nn = KNeighborsRegressor(n_neighbors=val, algorithm="brute")
    # Convert String Male/Female to integers 1/0
    yTrainInt = (yTrain.ravel() == "Male").astype(int)
    nn.fit(xTrain, yTrainInt)
    yPredicted = nn.predict(xTest)
    yPredicted = yIntToStr(yPredicted)
    accuracy = accuracy_score(yTest, yPredicted)
    accuracy_vals.append(accuracy)
    cm = confusion_matrix(yTest, yPredicted)
    print(" →   {:^9}   {:^8}   {}".format(val, accuracy, cm.tolist()))
    if best_k_val[1] < accuracy:
        best_k_val = (val, accuracy, cm, yPredicted)

graphNeighborsAccuracy(k_neighbors_vals, accuracy_vals,
                       "GENERO", "genero_neighbors_acc.png")
graphGENERO(xTest, yTest, best_k_val[3], "{}-Nearest Neighbors".format(best_k_val[0]),
            "genero_neighbors_pred.png")
graphConfusionMatrix(best_k_val[2], ["Female", "Male"], "Gender",
                     "GENERO: {}-Nearest Neighbors".format(best_k_val[0]),
                     "genero_neighbors_cm.png")
```

C CÓDIGO DE REGRESIÓN LOGÍSTICA DEL DATASET DEFAULT

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

from graphs import graphConfusionMatrix

print(" ~ Reading default.txt and generating train and test sets")
data = pd.read_csv('default.txt', sep=" ")
# Transform 'default' column from Yes/No to a boolean
data["default"] = (data["default"] == "Yes").astype(bool)
# Transform 'student' column from Yes/No to an integer
data["student"] = (data["student"] == "Yes").astype(int)
x = data.iloc[:, 1:4].values.reshape(-1, 3)
y = data.iloc[:, 0].values.reshape(-1, 1)
xTrain, xTest, yTrain, yTest = train_test_split(
    x, y, test_size=0.2, random_state=0)

print(" ~ Creating the logistic regression model")
regressor = LogisticRegression()
regressor.fit(xTrain, yTrain.ravel())

print(" ~ Testing the logistic regression model")
yPredicted = regressor.predict(xTest)
accuracy = accuracy_score(yTest, yPredicted)
print(" →   Accuracy:", accuracy)
cm = confusion_matrix(yTest, yPredicted)
print(" →   Confusion matrix:", cm.tolist())

graphConfusionMatrix(cm, ["Yes", "No"], "Default",
                     "DEFAULT: Logistic Regression", "default_regression_cm.png")
```

D CÓDIGO DE REGRESIÓN LOGÍSTICA DEL DATASET GENERO

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

from graphs import graphConfusionMatrix, graphGENERO

print(" ~ Reading genero.txt and generating train and test sets")
data = pd.read_csv('genero.txt')
x = data.iloc[:, 1:3].values.reshape(-1, 2)
y = data.iloc[:, 0].values.reshape(-1, 1)
xTrain, xTest, yTrain, yTest = train_test_split(
    x, y, test_size=0.2, random_state=0)

print(" ~ Creating the logistic regression model")
regressor = LogisticRegression()
regressor.fit(xTrain, yTrain.ravel())

print(" ~ Testing the logistic regression model")
yPredicted = regressor.predict(xTest)
accuracy = accuracy_score(yTest, yPredicted)
print(" →   Accuracy:", accuracy)
cm = confusion_matrix(yTest, yPredicted)
print(" →   Confusion matrix:", cm.tolist())

graphGENERO(xTest, yTest, yPredicted, "Logistic Regression",
            "genero_regression_pred.png")
graphConfusionMatrix(cm, ["Male", "Female"], "Gender",
                     "GENERO: Logistic Regression", "genero_regression_cm.png")
```

E CÓDIGO DE GENERACIÓN DE GRÁFICAS

```
import math
import sys
import matplotlib.pyplot as plt
import numpy as np

shouldDisplay = "--display-graphs" in sys.argv
shouldSave = "--save-graphs" in sys.argv

mapColor = np.vectorize(lambda x: "b" if x == "Male" else "r")


def graphGENERO(xys, actualVals, predictedVals, title, filename):
    fig, (ax1, ax2) = plt.subplots(2, sharex=True)
    fig.set_size_inches(6, 7)
    fig.suptitle("GENERO: " + title)

    ax1.set_title("Actual values")
    ax1.set_ylabel("Weight")
    actualColors = mapColor(actualVals).flatten()
    ax1.scatter(xys[:, 0], xys[:, 1], c=actualColors)

    ax2.set_title("Predicted values")
    ax2.set_xlabel("Height")
    ax2.set_ylabel("Weight")
    predictedColors = mapColor(predictedVals).flatten()
    ax2.scatter(xys[:, 0], xys[:, 1], c=predictedColors)

    if shouldSave:
        fig.savefig(filename)
    if shouldDisplay:
        plt.show()


def graphConfusionMatrix(cm, labels, variableName, title, filename):
    plt.figure()
    plt.imshow(cm, interpolation="nearest", cmap="BuGn")
    plt.title("Confusion matrix: " + title)
    plt.colorbar()
    plt.xticks(np.arange(2), labels, size=10)
    plt.yticks(np.arange(2), labels, size=10)
    plt.xlabel("Actual " + variableName)
    plt.ylabel("Predicted " + variableName)

    for x in range(2):
        for y in range(2):
            plt.annotate(cm[x][y], xy=(y, x),
                         horizontalalignment="center",
                         verticalalignment="center")

    if shouldSave:
        plt.savefig(filename)
    if shouldDisplay:
        plt.show()


def graphNeighborsAccuracy(k_neighbors, accuracy, title, filename):
    x = list(map(str, k_neighbors))

    plt.figure()
    plt.title(title + ": k-Neighbors vs Accuracy")
    plt.plot(x, accuracy)
    plt.plot(x, accuracy, 'og')
```