Kn Vecinos

Código

ROANO NAVARRO ANGEL

Facultad de estadistica e informatica  21 de mayo de 2023

# Introducción

Como parte del curso se presento el tema de minería de datos. Para esta práctica se presentará el algoritmo de K vecinos cercanos o conocido K-NN. Este ayudara a fortalecer los temas de aprendizaje supervisado.

El algoritmo ayuda a clasificar datos en base a datos establecidos. En esta práctica manejaremos el código en Python, y ocuparemos los datos presentados por el profesor para poder sacar métricas de resultados de este.

# Propuesta

En principio importamos las siguientes librerías:

import numpy as np

from collections import Counter

from sklearn.metrics import confusion\_matrix

Calculamos la distancia

def distancia\_euclidiana(var1, var2):

    return np.sqrt(np.sum((var1 - var2) \*\* 2))

ocuparemos la siguiente clase con su constructor y funciones:

class KNN:

    #Inicializacion de clase, k representa al numero de vecinos cercanos

    def \_\_init\_\_(self, k=3):

        self.k = k

    #Datos de entrenamiento

    def fit(self, X, y):

        self.X\_train = X

        self.y\_train = y

    #Prediccion de clase

    def predict(self, X):

        y\_pred = [self.\_predict(x) for x in X]

        return np.array(y\_pred)

    def \_predict(self, x):

        # Calcular las distancias entre el punto de prueba y todos los puntos de entrenamiento

        distancia = [distancia\_euclidiana(x, x\_train) for x\_train in self.X\_train]

        # Obtener las clases de los k vecinos más cercanos

        k\_indices = np.argsort(distancia)[:self.k]

        k\_nearest\_labels = [self.y\_train[i] for i in k\_indices]

        # Realizar la predicción basada en la mayoría

        most\_common = Counter(k\_nearest\_labels).most\_common(1)

        return most\_common[0][0]

implementaremos los datos de entrenamiento:

X\_train = np.array([[1.869423182, 2.791976048], [2.27318592, 2.477111136], [2.82734086, 1.893176098], [1.539694984, 2.50469424], [2.875420219, 2.221298127],

                    [2.092386085, 1.813369122], [3.693069637, 1.427792679], [1.576019048, 1.163440381], [2.009507082, 1.173333733], [0.8756594522, 0.6441210923],

                    [7.409944278, -0.3059050957], [8.435077921, -0.3707635751], [3.989400599, -0.949987061], [4.589165213, -0.03945974499], [4.766679469, -2.232500282],

                    [5.953879833, -0.976158823], [5.426023982, -1.37631105], [1.984108012, -0.5616472507], [6.159202003, 1.423240065], [3.373001483, -0.9532170467]])

y\_train = np.array(["A", "A","A","A","A","A","A","A","A","A","B","B","B","B","B","B","B","B","B","B"])

basándonos en una distancia de 3 para poder predecir qué resultado será:

knn = KNN(k=3)

knn.fit(X\_train, y\_train)

Manejaremos los siguientes datos de prueba:

X\_test = np.array([[2.49856666, 2.295987279], [0.5060700405, 2.870605328], [2.821145579, 3.044439904], [1.103348767, 3.259222168], [2.178202675, 1.471900676],

                    [1.077220263, 2.603175495], [1.832200641, 2.745357961], [4.441079885, 2.683848879], [2.002112344, 1.410244628], [1.470189159, 2.31022513],

                    [2.430317441, -2.233903936], [4.173766546, 1.257892673], [7.21439806, -0.3865905836], [5.21280248, -3.246487248], [5.365220161, -3.660845623],

                    [4.867157086, 0.6122608129], [5.186137477, -3.325490691], [4.002715986, -0.1184238242], [4.133560572, -2.589857889], [3.482855376, 0.5401176995]

])

predictions = knn.predict(X\_test)

Generamos dos variables para poder manejar los arreglos:

y\_true = y\_train

y\_pred = predictions

Generamos la matriz de confusión:

matriz\_confusion = confusion\_matrix(y\_true, y\_pred)

print("Matriz de confusión:\n",matriz\_confusion)

Separaremos la matriz en verdadero negativo, falso positivo, falso negativo y verdadero positivo:

vn, fp, fn, vp = matriz\_confusion.ravel()

Ya para finalizar generamos métricas para validar modelos:

accuracy = (vp + vn) / (vp + fp + vn + fn)

print("Exactitud(Acuracy):",accuracy)

precision = vp/(vp + fp)

print("Precision:", precision)

sensitivity = vp / (vp + fn)

print("Sensibilidad:", sensitivity)

specificity = vn / (vn + fp)

print("Especificidad:", specificity)

f1 = (2 \* precision \* sensitivity) / (precision  + sensitivity)

print("Métrica F:", f1)

# Experimento y resultados

Datos de entrenamiento:

[[ 1.86942318 2.79197605]

[ 2.27318592 2.47711114]

[ 2.82734086 1.8931761 ]

[ 1.53969498 2.50469424]

[ 2.87542022 2.22129813]

[ 2.09238608 1.81336912]

[ 3.69306964 1.42779268]

[ 1.57601905 1.16344038]

[ 2.00950708 1.17333373]

[ 0.87565945 0.64412109]

[ 7.40994428 -0.3059051 ]

[ 8.43507792 -0.37076358]

[ 3.9894006 -0.94998706]

[ 4.58916521 -0.03945974]

[ 4.76667947 -2.23250028]

[ 5.95387983 -0.97615882]

[ 5.42602398 -1.37631105]

[ 1.98410801 -0.56164725]

[ 6.159202 1.42324007]

[ 3.37300148 -0.95321705]]

['A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'B' 'B' 'B' 'B' 'B' 'B' 'B' 'B'

'B' 'B']

Datos de predeicción:

[[ 2.49856666 2.29598728]

[ 0.50607004 2.87060533]

[ 2.82114558 3.0444399 ]

[ 1.10334877 3.25922217]

[ 2.17820268 1.47190068]

[ 1.07722026 2.60317549]

[ 1.83220064 2.74535796]

[ 4.44107988 2.68384888]

[ 2.00211234 1.41024463]

[ 1.47018916 2.31022513]

[ 2.43031744 -2.23390394]

[ 4.17376655 1.25789267]

[ 7.21439806 -0.38659058]

[ 5.21280248 -3.24648725]

[ 5.36522016 -3.66084562]

[ 4.86715709 0.61226081]

[ 5.18613748 -3.32549069]

[ 4.00271599 -0.11842382]

[ 4.13356057 -2.58985789]

[ 3.48285538 0.5401177 ]]

['A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'B' 'A' 'B' 'B' 'B' 'B' 'B' 'B'

'B' 'B']

Matriz de confusión:

[[10 0]

[ 1 9]]

Exactitud(Acuracy): 0.95

Precision: 1.0

Sensibilidad: 0.9

Especificidad: 1.0

Métrica F: 0.9473684210526316

# Conclusiones

Cuenta con un 95% de exactitud, esto debido a que solo uno no concordó con los resultados esperados. Precisión de un 100% ya que en su mayoría concuerda el resultado con lo esperado. Sensibilidad de un 90% ya que se identifico una gran cantidad de positivos reales. Especificidad de 100% por la cantidad de negativos reales encontrados. Una medida F del 94% por su precisión para el modelo.