Actividad 14

Jose Manuel Enriquez

29 de marzo del 2025

1 Introduction

1.1 ¿Qué es k-Nearest Neighbor?

El algoritmo K-Nearest Neighbors (KNN) es un método de aprendizaje supervisado utilizado para tareas de clasificación y regresión. Se basa en la idea de que una muestra será clasificada según la mayoría de sus vecinos más cercanos en el espacio de características. Para hacer una predicción, KNN calcula la distancia entre la muestra de prueba y todas las muestras del conjunto de entrenamiento, generalmente utilizando la distancia euclidiana, aunque pueden emplearse otras métricas como la distancia de Manhattan o Minkowski. Luego, selecciona los K vecinos más cercanos y asigna la clase más frecuente (para clasificación) o el promedio de sus valores (para regresión). KNN es un algoritmo no paramétrico, lo que significa que no hace suposiciones sobre la distribución de los datos, y es perezoso (lazy learning), ya que no genera un modelo explícito durante el entrenamiento, sino que almacena los datos y los compara en el momento de la predicción. Sus principales aplicaciones incluyen el reconocimiento de patrones, la detección de anomalías, los sistemas de recomendación y la clasificación de imágenes. Aunque es fácil de entender e implementar, KNN puede volverse computacionalmente costoso para grandes volúmenes de datos, ya que requiere calcular distancias con todas las muestras almacenadas.

2 Metodología

Este ejercicio fue realizado en una Jupyter Notebook y Python 3.12.4 Lo primero que se hizo fue importar todas las librerías necesarias.

Primero se importan todas las librerias necesarias

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
import seaborn as sb

```

```
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

Luego importamos los datos

```
dataframe = pd.read_csv(r"reviews_sentiment.csv",sep=';')
```

Se prepara el dataset

```
X = dataframe[['wordcount','sentimentValue']].values
y = dataframe['Star Rating'].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
    random_state=0)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Se crea el modelo

```
n_neighbors = 7

knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

Los resultados obtenidos fueron los siguientes

```
[[ 9  0  1  0  0]
 [ 0  1  0  0  0]
 [ 0  1 17  0  1]
 [ 0  0  2  8  0]
 [ 0  0  4  0 21]]
              precision    recall  f1-score   support

           1       1.00      0.90      0.95        10
           2       0.50      1.00      0.67         1
           3       0.71      0.89      0.79        19
           4       1.00      0.80      0.89        10
           5       0.95      0.84      0.89        25

    accuracy                           0.86        65
   macro avg       0.83      0.89      0.84        65
weighted avg       0.89      0.86      0.87        65
```

Figure 1: Resultados

La gráfica de clasificación que se obtuvo fue la siguiente:

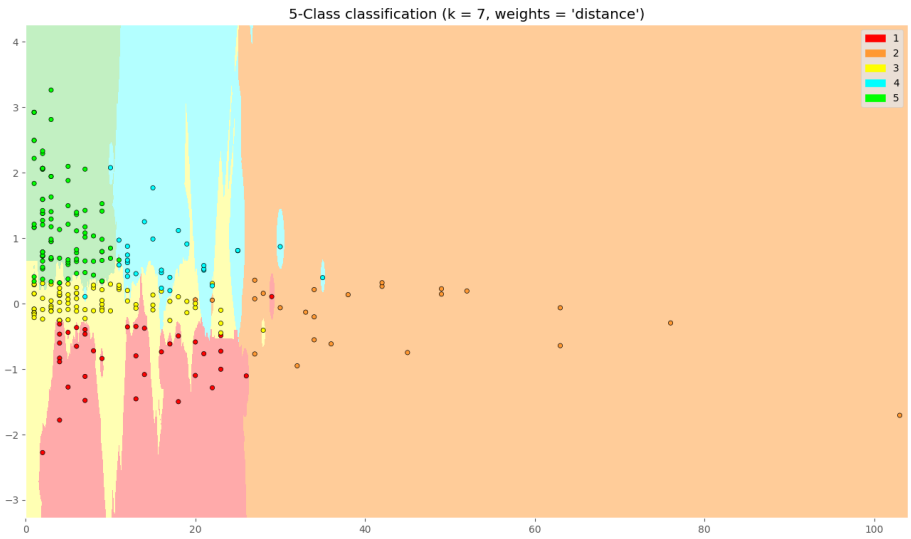

Figure 2: Clasificación

3 Conlusiones

Me pareció muy interesante realizar predicciones con KNN con Python ya que es de las primeras veces en las cuales implemento esta herramienta.