Actividad 14: Programando K-Nearest-Neighbor en Python

Gerardo Enrique Torres Flores 2064063

30 de marzo de 2025

1. Introducción

K-Nearest-Neighbor (K-NN) es un algoritmo basado en instancia y de tipo supervisado en Machine Learning. Puede usarse tanto para clasificar nuevas muestras (valores discretos) como para predecir valores continuos (regresión). Al ser un método sencillo, es ideal para introducirse en el mundo del Aprendizaje Automático.

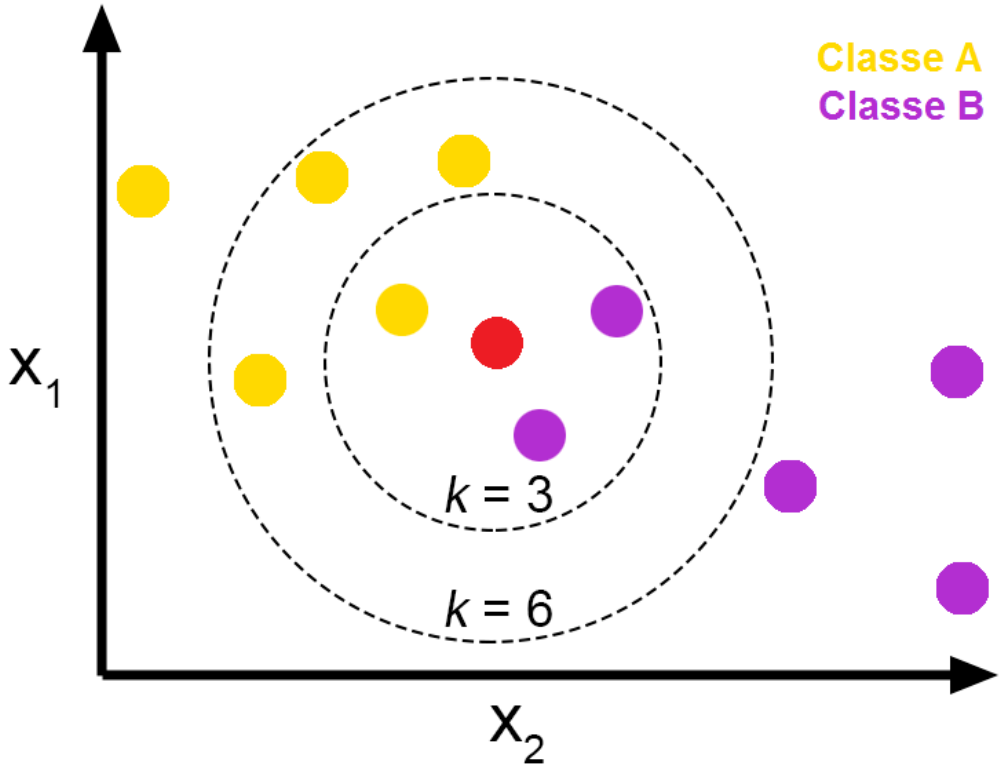
El algoritmo funciona esencialmente buscando los puntos de datos "más similares" (por cercanía) en el conjunto de entrenamiento, y luego clasifica la nueva muestra basándose en la mayoría de las etiquetas de esos vecinos. En otras palabras, k-NN:

- **Supervisado:** Nuestro conjunto de entrenamiento ya viene etiquetado, por lo que cada observación tiene la clase o resultado esperado.
- **Basado en Instancia:** El algoritmo no construye un modelo explícito, sino que memoriza las instancias de entrenamiento que se utilizarán como "base de conocimiento" para predecir la clase de nuevas observaciones.

Aunque es un método simple, k-NN se utiliza en la resolución de numerosos problemas, tales como **sistemas de recomendación, búsqueda semántica y detección de anomalías.**

¿Cómo funciona k-NN?

1. Calcular la distancia entre el item a clasificar y el resto de items del dataset de entrenamiento.
2. Seleccionar los "k" elementos más cercanos (aquellos con menor distancia, según la función utilizada).
3. Realizar una "votación de mayoría" entre los k vecinos: la clase que tenga más votos será la clasificación final asignada al item.



2. Metodología

Para realizar esta actividad se siguieron los siguientes pasos:

1. Importación de librerías y configuración

Se importan las librerías necesarias para la manipulación de datos, visualización y creación del modelo, configurando además el estilo y tamaño de las gráficas:

```
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

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,
    confusion_matrix
```

2. Lectura y exploración de datos

Se carga el dataset `reviews_sentiment.csv` (usando punto y coma como separador) y se exploran sus distribuciones:

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

print(dataframe.groupby('Star Rating').size())
sb.catplot(x='Star Rating', data=dataframe, kind="count",
    aspect=3)
sb.catplot(x='wordcount', data=dataframe, kind="count",
    aspect=3)
```

Visualización de Histograma de los Datos:

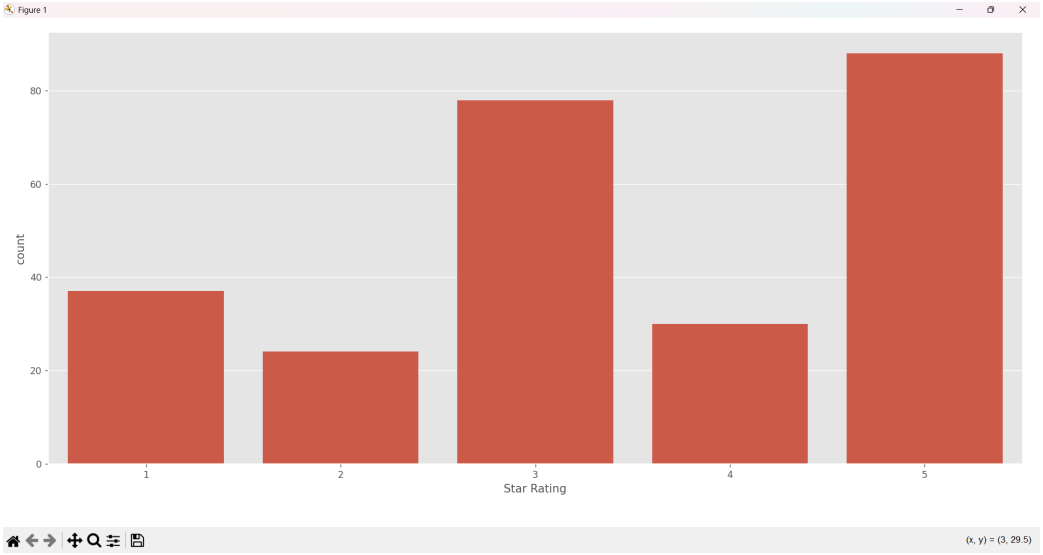
3. Preparación de los datos y entrenamiento del modelo

Se definen las variables predictoras y la variable objetivo, se realiza la partición en datos de entrenamiento y prueba, y se escala la información. Se entrena un clasificador K-NN con $k = 7$:

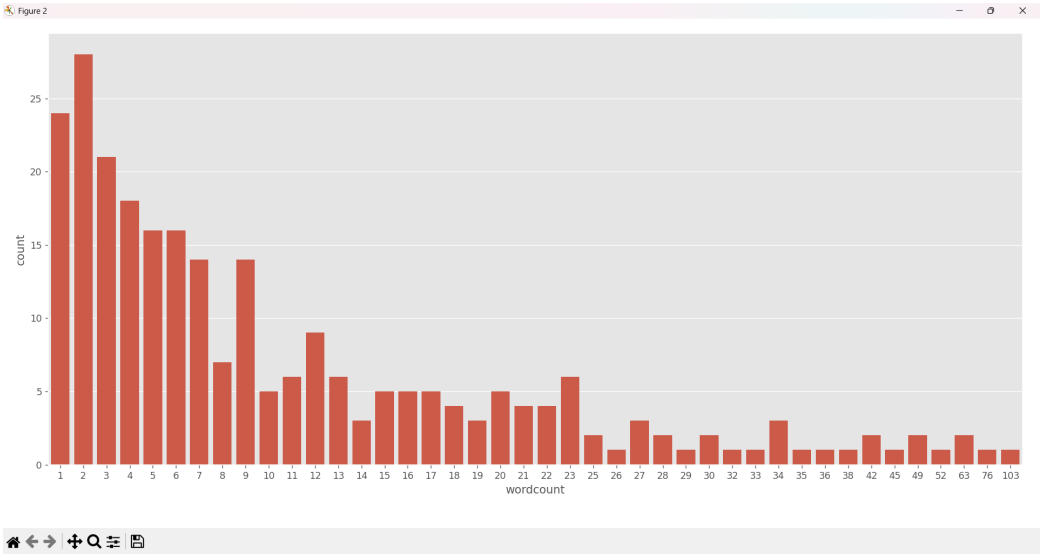
```
X = dataframe[['wordcount','sentimentValue']].values
y = dataframe['Star Rating'].values

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)

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)))

```

```
16 pred = knn.predict(X_test)
17 print(confusion_matrix(y_test, pred))
18 print(classification_report(y_test, pred))
```

Visualización de Star Rating:



Visualización de Conteo de Palabras:

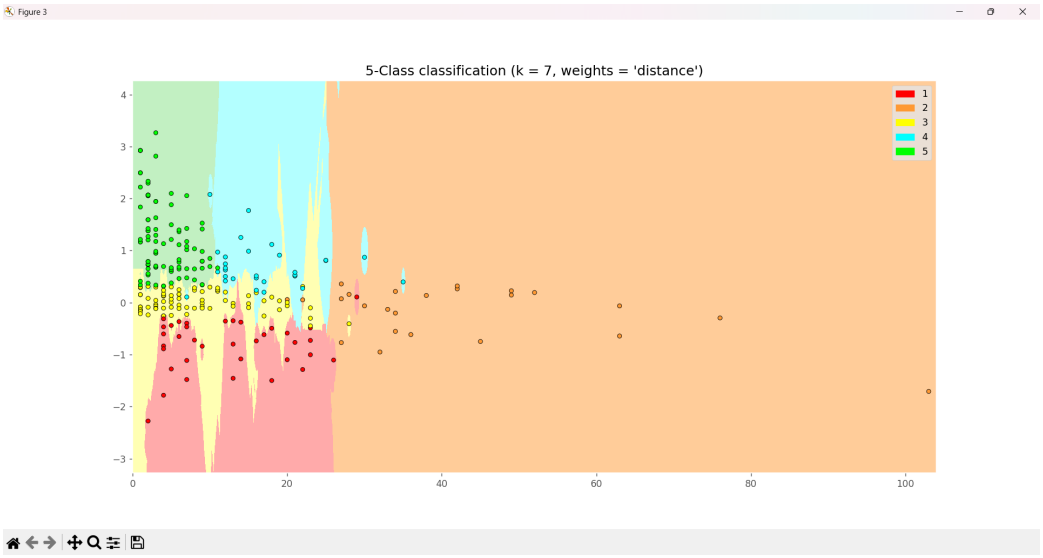


4. Visualización de la frontera de decisión

Se crea una gráfica que ilustra la frontera de decisión del modelo utilizando un clasificador con ponderación por distancia y se asigna un color a cada clase:

```
h = .02  # Tama o del paso en la malla

# Mapas de color para la gr fica
cmap_light = ListedColormap(['#FFAAAA', '#ffcc99', '#ffffb3',
    '#b3ffff', '#c2f0c2'])
cmap_bold = ListedColormap(['#FF0000', '#ff9933', '#FFFF00',
   '#00ffff', '#00FF00'])

clf = KNeighborsClassifier(n_neighbors, weights='distance')
clf.fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(
   y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor=
   'k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())

patch0 = mpatches.Patch(color='#FF0000', label='1')
patch1 = mpatches.Patch(color='#ff9933', label='2')
patch2 = mpatches.Patch(color='#FFFF00', label='3')
patch3 = mpatches.Patch(color='#00ffff', label='4')
patch4 = mpatches.Patch(color='#00FF00', label='5')
plt.legend(handles=[patch0, patch1, patch2, patch3, patch4],
   loc='upper right')

plt.title("5-Class classification (k = %i, weights = '%s')" %
    (n_neighbors, 'distance'))
plt.show()
```

Visualización de Gráfico KNN:



5. Selección del parámetro k

Se evalúa el desempeño del clasificador para distintos valores de k mediante una gráfica de precisión:

```
k_range = range(1, 20)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```

6. Predicción de nuevos casos

Se muestran ejemplos de predicción y la probabilidad asociada:

```
print(clf.predict([[5, 1.0]]))
print(clf.predict_proba([[20, 0.0]]))
```

3. Resultados

Los resultados obtenidos en esta actividad incluyen:

- **Exactitud del clasificador:** Se muestran las precisiones en los conjuntos de entrenamiento y prueba.
- **Matriz de confusión y reporte de clasificación:** Permiten evaluar el desempeño del modelo para cada clase.
- **Frontera de decisión:** La visualización muestra cómo se separan las 5 clases en el espacio de características.
- **Selección del parámetro k:** Se determina el valor óptimo de k mediante la gráfica de precisión.

Resultados de Predecir Nuevas Muestras:

```
[5]
# Este resultado nos indica que para 5 palabras y sentimiento
    1, nos valoraran la app con 5 estrellas.

[[0.00381998 0.02520212 0.97097789 0.         0.        ]]
# Para las coordenadas 20, 0.0 hay 97% probabilidades que nos
    den 3 estrellas.
```

4. Conclusión

El algoritmo **K-Nearest-Neighbor** nos sirvió para clasificar reseñas en 5 niveles de calificación utilizando las variables `wordcount` y `sentimentValue`. Hubo un preprocesamiento de los datos mediante escalado, se entrenó el modelo y se evaluó su desempeño a través de la exactitud, la matriz de confusión y el reporte de clasificación. Además, se pudo visualizar la frontera de decisión para entender mejor el comportamiento del clasificador, y se exploró la influencia del parámetro k en la precisión del modelo. En conclusión, se muestra la capacidad de K-NN para abordar problemas de clasificación en conjuntos de datos reales.

```
Star Rating
1    37
2    24
3    78
4    30
5    88
dtype: int64
Accuracy of K-NN classifier on training set: 0.90
Accuracy of K-NN classifier on test set: 0.86
[[ 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

[5]
[[0.00381998 0.02520212 0.97097789 0.         0.        ]]
```