Act 14: Programando K-Nearest-Neighbor en Python

Ana Isabel Loera Gil

30 de mayo del 2025

1 Introducción

K-Nearest-Neighbor es un algoritmo basado en instancia de tipo supervisado de Machine Learning. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Busca en las observaciones más cercanas a la que se está tratando de predecir y clasifica el punto de interés basado en la mayoría de datos que le rodean.

2 Metodología

2.1 Importación de librerias

```
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
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
```

2.2 Leer el archivo csv y mostrar las primeros 10 registros

```
dataframe= pd.read_csv(r"reviews_sentiment.csv", sep=';')
print(dataframe.head(10))
```

2.3 Resumen estadístico de los datos

```
print(dataframe.describe())
```

2.4 Visualización de la información

```
dataframe.hist()
plt.show()

print(dataframe.groupby('Star Rating').size())
sb.catplot(x='Star Rating', data=dataframe, kind="count", aspect=3,palette="Set2")
plt.show()

sb.catplot(x='wordcount',data=dataframe, kind="count", aspect=3, palette="Set3")
plt.show()
```

2.5 Crear X, y de entrada y los sets de entrenamiento

```
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)
```

2.6 Usar k-Nearest Neighbor con Scikit Learn

```
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)))
```

2.7 Confirmación de la precisión del modelo

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

2.8 Gráfica con la clasificación obtenida

```
h=.02
weights = 'distance'

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')
patch1 = mpatches.Patch(color='#ff9933')
patch2 = mpatches.Patch(color='#FFFF00')
patch3 = mpatches.Patch(color='#00ffff')
patch4 = mpatches.Patch(color='#00FF00')

plt.legend(handles=[patch0, patch1, patch2, patch3, patch4])
plt.title("5-Class classification (k = %i, weights = '%s')" % (n_neighbors, weights))
plt.show()
```

2.9 Elección del mejor valor de k

```
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])
plt.show()
```

2.10 Clasificación o predicción de nuevas muestras

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

3 Resultados

	Review Title	...	sentimentValue
0	Sin conexión	...	-0.486389
1	faltan cosas	...	-0.586187
2	Es muy buena lo recomiendo	...	-0.602240
3	Version antigua	...	-0.616271
4	Esta bien	...	-0.651784
5	Buena	...	-0.720443
6	De gran ayuda	...	-0.726825
7	Muy buena	...	-0.736769
8	Ta to guapa.	...	-0.765284
9	Se han corregido	...	-0.797961

[10 rows x 7 columns]

Figure 1: Primeros registros del csv

	wordcount	Star Rating	sentimentValue
count	257.000000	257.000000	257.000000
mean	11.501946	3.420233	0.383849
std	13.159812	1.409531	0.897987
min	1.000000	1.000000	-2.276469
25%	3.000000	3.000000	-0.108144
50%	7.000000	3.000000	0.264091
75%	16.000000	5.000000	0.808384
max	103.000000	5.000000	3.264579

Figure 2: Información estadística

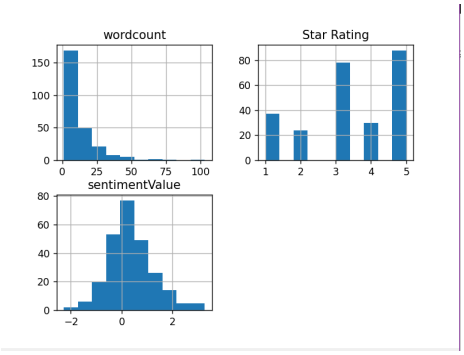

Figure 3: Gráfico de barras de la información

```
Star Rating
1    37
2    24
3    78
4    30
5    88
dtype: int64
```

Figure 4: Valoraciones de estrellas

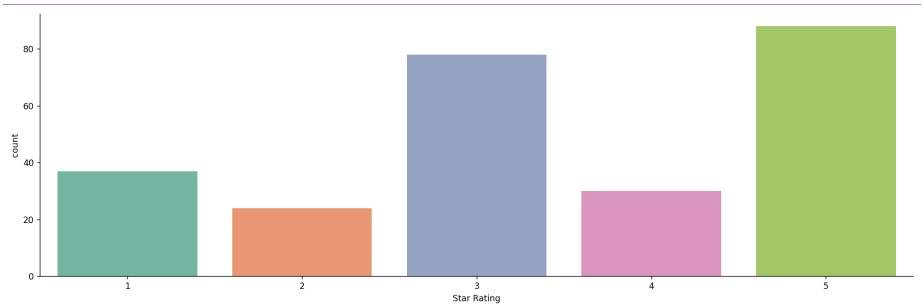

Figure 5: Gráfica de rating de estrellas

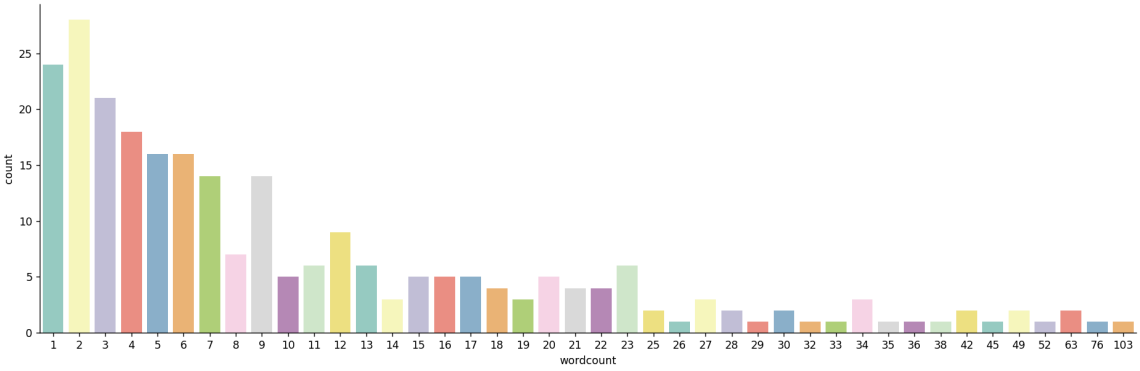

Figure 6: Gráfica cantidad de palabras

```
Accuracy of K-NN classifier on training set 0.90
Accuracy of K-NN classifier on test set 0.86
```

Figure 7: Precisión del entrenamiento y test

```
[[ 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]]
```

Figure 8: Mátriz de confusión

	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 9: Reporte de clasificación

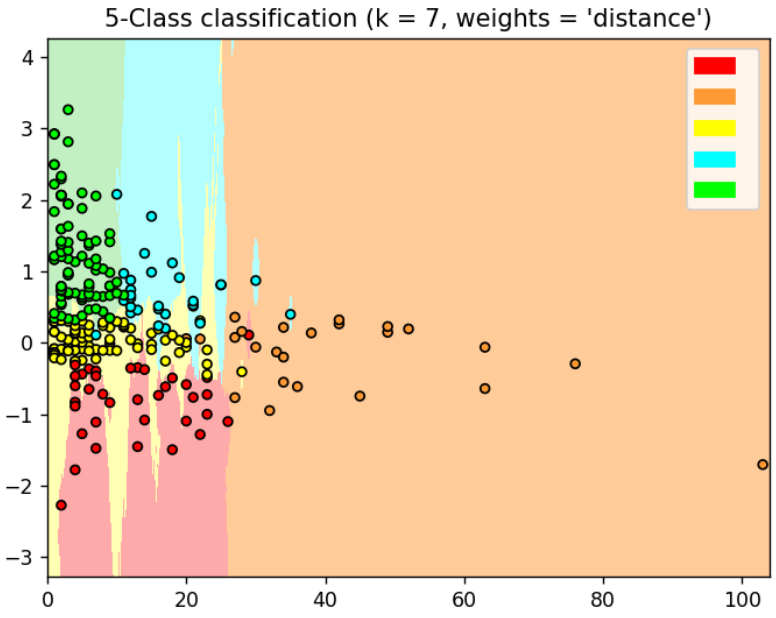

Figure 10: Gráfica de la clasificación obtenida

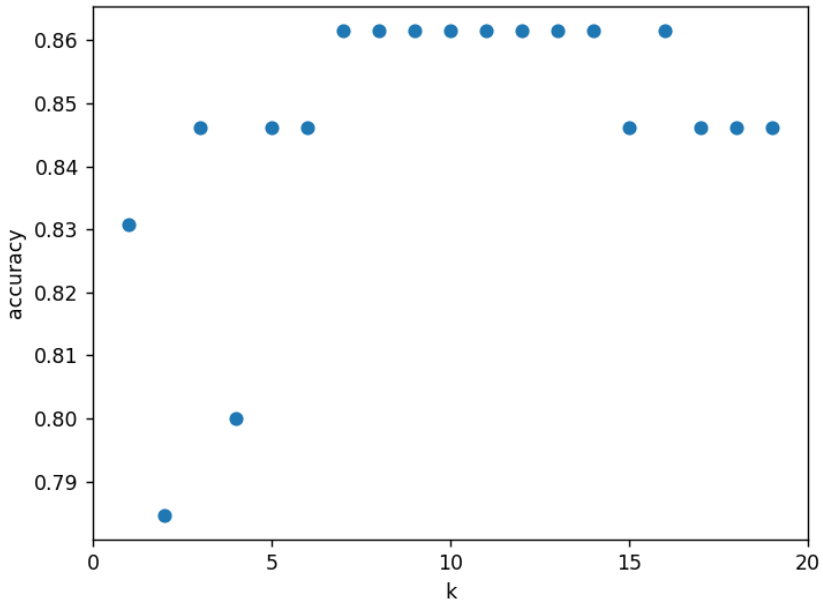

Figure 11: Gráfico de la mejor elección de k

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

Figure 12: Predicciones

4 Conclusión

Se creó un modelo con Python utilizando el algoritmo k-Nearest Neighbor para clasificar puntos de un conjunto de entrada. Dado que es un algoritmo supervisado, fue necesario contar con suficientes datos etiquetados para lograr un entrenamiento efectivo. Aunque es un método simple, requiere una gran cantidad de memoria y poder de cómputo, lo que limita su uso en conjuntos de datos muy grandes. En este caso, se trabajó con dos dimensiones para visualizar gráficamente la clasificación y comprender mejor la formación de los grupos. Finalmente, el modelo permitió realizar nuevas predicciones, lo que ayudó a profundizar en la problemática analizada.

5 Referencias bibliograficas

Ignacio Bagnato, J. (2020). Aprende machine learning. Leanpub.