K-Nearest-Neighbor

Valeria Rodríguez

30 de Marzo de 2025

1 Introducción

K-Nearest-Neighbor es un algoritmo supervisado basado en instancia de Machine Learning. Es comunmente usado para clasificar valores discretos o para predecir con una regresión. Sirve para clasificar valores buscando puntos de datos similares por cercanía aprendidos en la etapa de entrenamiento del modelo así como haciendo conjeturas para nuevos puntos de acuerdo a la clasificación. En este método, la K significa la cantidad de "puntos vecinos" que se tienen en cuenta en las cercanías para clasificar los n gurpos conocidos.

2 Metodología

Para la realización de esta actividad se llevaron a cabo una serie de pasos encaminados a la recreación del problema de árbol de decisión presentado en el libro *Aprende Machine Learning* del autor Juan Ignacio Bagnato, páginas 115-127.

2.1 Creación de carpeta de trabajo

Se creó una carpeta llamada KNN, en la cual se creó un archivo Python llamado knn para la codificación de la actividad. Posteriormente, se descargó el archivo csv necesario para la actividad y se añadió a la carpeta de trabajo.

2.2 Desarrollo del código

Con ayuda del IDE Visual Studio Code se abrió el archivo .py anteriormente creado y se codificó según las especificaciones del libro. Primeramente, se hicieron los imports al archivo

```
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
from sklearn.metrics import confusion_matrix
```

Posteriormente, se hace lectura del archivo csv, se imprimen las primeras 10 líneas de datos y se imprime una descripción de los datos. En este archivo en particular, los datos de cada columna están separados por punto y coma, por lo que es necesario especificar esto en el comando de lectura.

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

Ahora, se crean unas gráficas para verificar la información que proporcionan.

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

Estas gráficas indican la distribución de *estrellas* que tiene la App, los *Valores de sentimientos*, así como el *Recuento de palabras* de la reseña. Se imprime el recuento de estrellas que ha obtenido la app y se crea una gráfica específica de las estrellas recibidas. Así bien, se crea una gráfica del recuento de palabras de las reseñas.

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

Posteriormente, se crean X e Y de entrada así como los sets de entrenamiento y test.

```
X=dataframe[['wordcount','sentimentValue']].values
y=dataframe['StarRating'].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 define el valor de k en 7 y se crea el calsificador.

```
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)))
```

Para confirmar la precisión, se imprime la Matriz de Confusión y el Reporte sobre el conjunto de test.

```
pred = knn.predict(X_test)
print("Matriz de confusión")
print(confusion_matrix(y_test, pred))
print("Reporte")
print(classification_report(y_test, pred))
```

Se crea una gráfica 2D de la clasificación obtenida para ver con facilidad dónde podrían caer las predicciones.

```
h = .02  step size in the mesh
weights = 'distance'
clf = KNeighborsClassifier(n_neighbors, weights=weights)
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()])

    plt.title(f"5-Class classification (k = n_neighbors, weights = 'clf.get_params()['weights']')")
plt.show()
```

Se rectifica el valor k dado en el paso anterior.

```
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()
```

Con esto, se observa que de k=7 a k=14 se obtiene una mayor precisión. Finalmente, se hacen predicciones para una reseña con 5 palabras y sentimiento 1, así como las probabilidades de que se califique la app con 1 a 5 estrellas.

```
print("Predicción de estrellas: ", clf.predict([[5, 1.0]]))
print("Probabilidad de estrellas: ")
print(clf.predict_proba([[20, 0.0]]))
```

3 Resultados

A continuación, se muestran los resultados obtenidos en las diversas fases de codificación mostradas en la sección anterior.

	Review Title	Review Text	...	Star Rating	sentimentValue
0	Sin conexión	Hola desde hace algo más de un mes me pone sin...	...	1	-0.486389
1	faltan cosas	Han mejorado la apariencia pero no	...	1	-0.586187
2	Es muy buena lo recomiendo	Andres e puto amoooo	...	1	-0.602240
3	Version antigua	Me gustana mas la version anterior esta es mas...	...	1	-0.616271
4	Esta bien	Sin ser la biblia.... Esta bien	...	1	-0.651784
5	Buena	Nada del otro mundo pero han mejorado mucho	...	1	-0.720443
6	De gran ayuda	Lo malo q necesita de …,pero la app es muy buena	...	1	-0.726825
7	Muy buena	Estaba más acostumbrado al otro diseño, pero e...	...	1	-0.736769
8	Ta to guapa.	Va de escándalo	...	1	-0.765284
9	Se han corregido	Han corregido muchos fallos pero el diseño es ...	...	1	-0.797961

Figure 1: Resultado obtenido del dataframe.head.

	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: Resultado obtenido del dataframe.describe

Figure 3: Gráficas descriptivas del contenido del archivo csv.

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

Figure 4: Cantidad de estrellas obtenidas por rango.

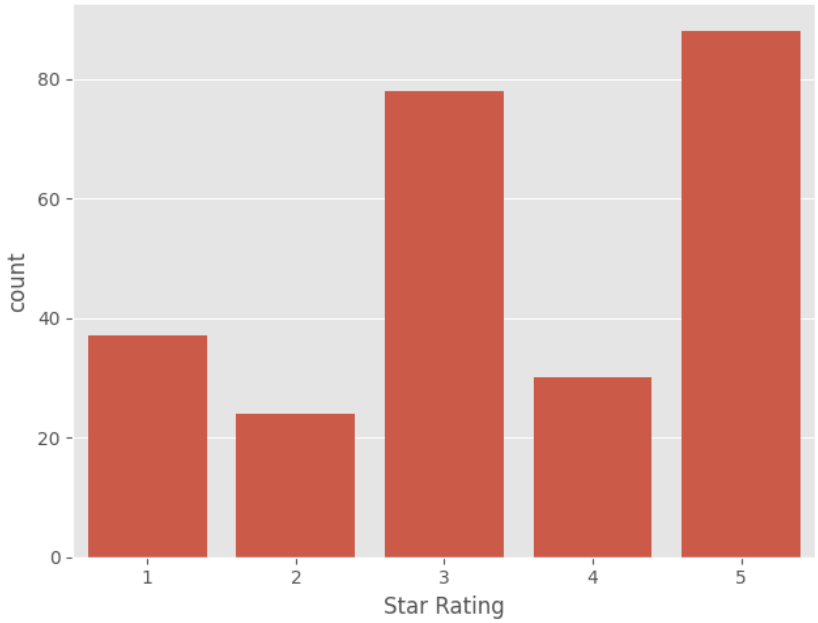

Figure 5: Gráfica de barras de estrellas obtenidas

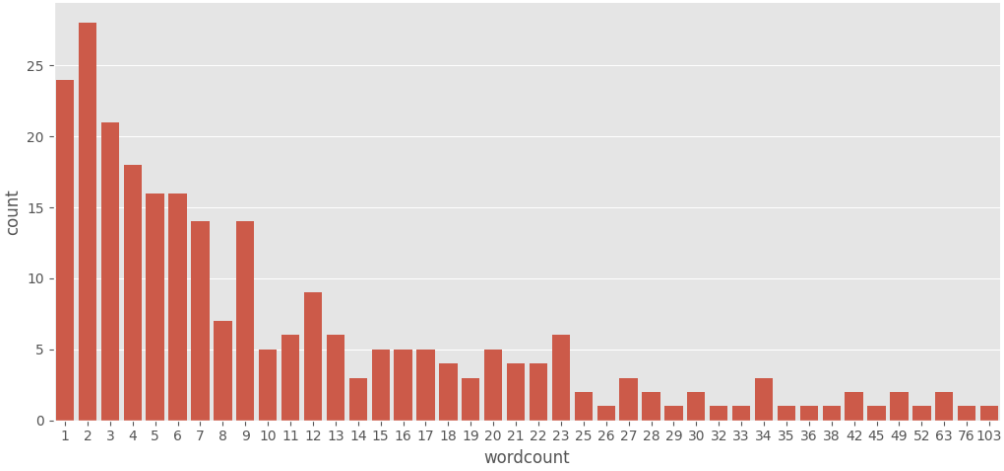

Figure 6: Gráfica de barras de recuento de palabras.

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

Figure 7: Porcentaje de acierto del modelo.

```
Matriz de confusión
[[ 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: Matriz de confusión.

Reporte

	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 del modelo.

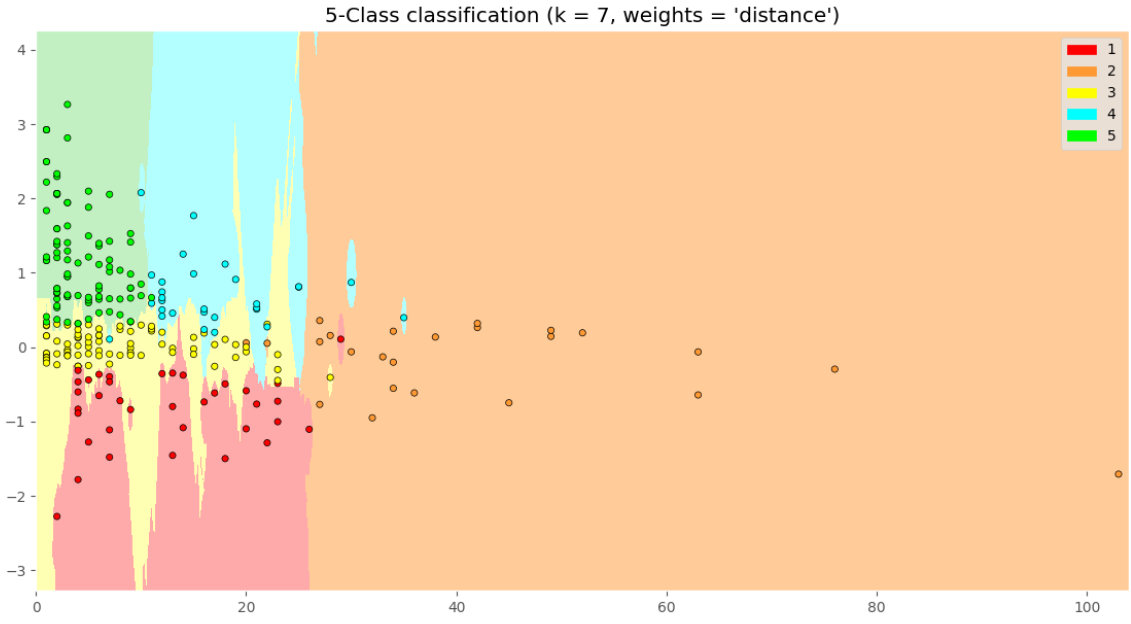

Figure 10: Gráfica descriptiva de reseñas.

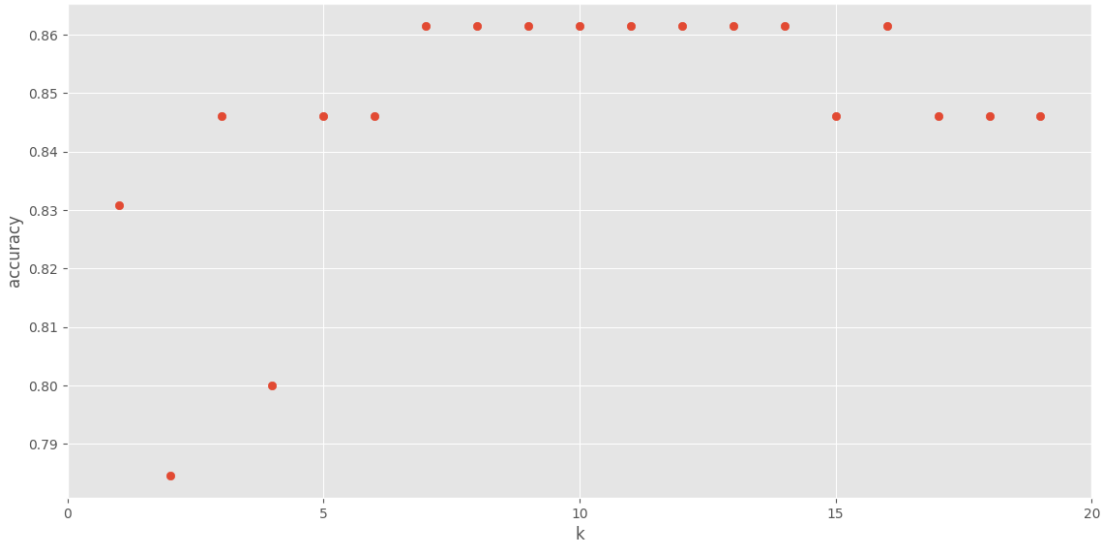

Figure 11: Gráfica de precisión por k valores.

```
Predicción de estrellas:  [5]
```

Figure 12: Predicción de estrellas para una reseña de 5 palabras y sentimiento 1.

```
Probabilidad de estrellas:
[[0.00381998 0.02520212 0.97097789 0.         0.        ]]
```

Figure 13: Probabilidad de obtener estrellas.

4 Conclusiones

Considero que la realización de esta práctica fue bastante útil para comprender cómo procesar y clasificar los diversos puntos de entrada con un algoritmo k-Nearest Neighbor. La realización del código no representó grandes retos más allá de algunas correcciones menores en ciertos comandos para asegurar el funcionamiento debido a distintas versiones del lenguaje. Los resultados fueron los esperados y estos, a su vez, fueron altamente favorecedores para el modelo y su análisis, en comparación con otros modelos estudiados anteriormente.

5 Referencias

Bagnato, J. (2019). Aprende Machine Learning. Leanpub.
Materiales de clase (2025). UANL.