Análisis del Algoritmo K-NN

Alain Lobato

30 de Marzo del 2025

1 Introducción

El algoritmo K-NN es un algoritmo utilizado en Machine Learning basado en instancias. Se puede utilizar para clasificar nuevas muestras o realizar predicciones. Al ser un algoritmo demasiado simple, lo convierte en uno de los algoritmos más usados por principiantes. Este algoritmo consiste en clasificar valores al identificar los puntos de datos más cercanos o similares los cuales fueron aprendidos durante la fase de entrenamiento, infiriendo nuevos puntos en base a este conocimiento. Es más eficiente cuando se trabaja con pocos datos o pocas características.

2 Metodología

Primero clonamos nuestro repositorio de la materia en nuestra máquina y luego creamos una carpeta llamada "K-Nearest-Neighbor". Descargamos el archivo CSV para extraer los datos y creamos un archivo llamado `KNN.py` donde desarrollamos nuestro código usando Visual Studio Code.

Iniciamos importando las librerías necesarias, instalando las dependencias en caso de ser necesario.

```python
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

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

Figure 1: Importación de librerías necesarias.

Ahora, vamos a leer el archivo CSV, así como validar que se carguen bien los datos, visualizando los primeros 10 registros y usando un `describe()` para ver ciertos valores estadísticos de nuestra tabla.

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

Figure 2: Lectura del archivo CSV.

```
              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

[10 rows x 7 columns]
```

Figure 3: Primeros 10 registros.

	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
count	257.000000	257.000000	257.000000
mean	11.501946	3.420233	0.383849
std	13.159812	1.409531	0.897987
count	257.000000	257.000000	257.000000
mean	11.501946	3.420233	0.383849
count	257.000000	257.000000	257.000000
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 4: Descripción estadística de los datos.

Generamos un histograma con estos datos:

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

Figure 5: Generación de histogramas.

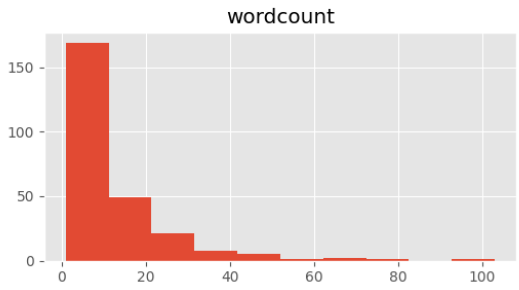

Figure 6: Histograma de Wordcount.

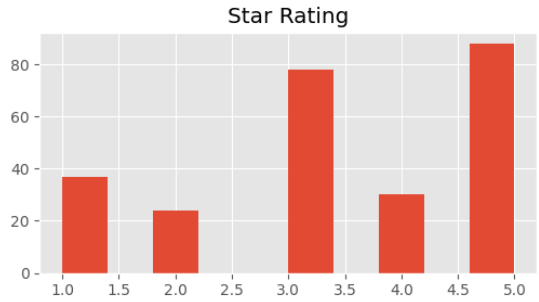

Figure 7: Histograma de Star Rating.

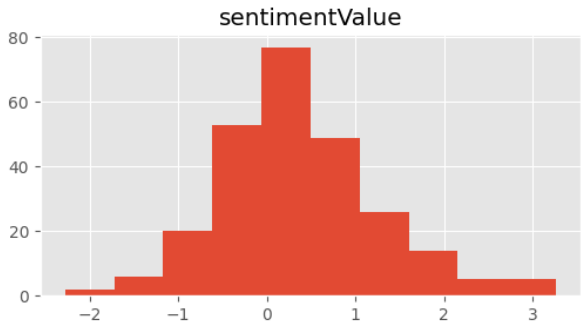

Figure 8: Histograma de Sentiment Value.

Ahora dividimos los datos agrupándolos según su `star rating` y lo graficamos:

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

Figure 9: Agrupación de datos por Star Rating.

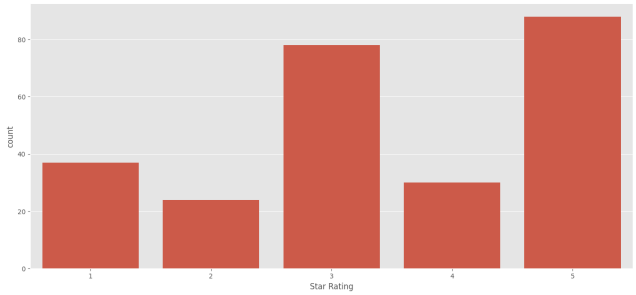

Figure 10: Grafica usando Star Rating.

Ahora lo hacemos graficando dependiendo del `word count`:

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

Figure 11: Histograma usando Word Count.

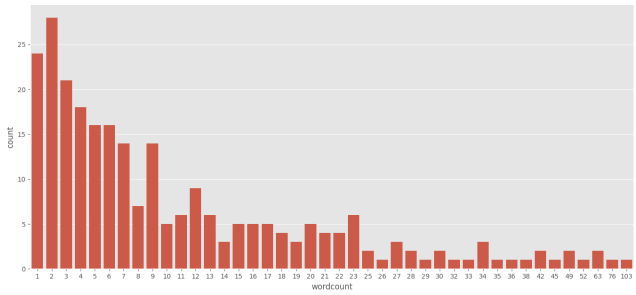

Figure 12: Grafica usando Word Count.

Ahora usaremos estas dos características para utilizarlas como variable independiente y el target, creando así los conjuntos de entrenamiento y prueba:

```
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)
```

Figure 13: Creación de los conjuntos de entrenamiento y prueba.

Ahora empezamos a crear nuestro modelo KNN fijando el número `k` como 7, e imprimimos la precisión de los conjuntos de prueba y entrenamiento:

```python
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)))
```

Figure 14: Codificacion de KNN.

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

Figure 15: Precisión del conjunto de prueba.

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

Figure 16: Mostrar tabla de precisio .

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

Figure 17: Buen porcentaje de F1.

Ahora vamos a graficar la clasificación obtenida para visualizar las predicciones:

```
h = .02
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])


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

  plt.show()
```

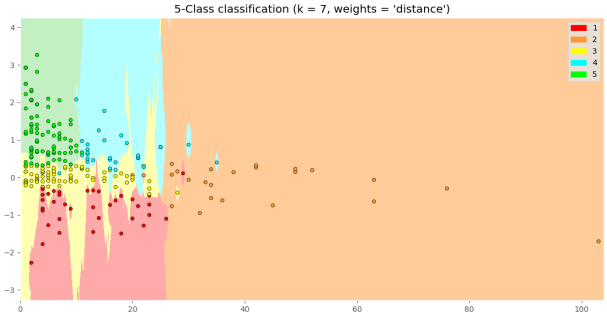

Figure 18: Gráfica de clasificación.

Gracias a esta grafica podemos ver, por ejemplo que los usuarios con 1 estrella tienen hasta 25 palabras y un sentimiento negativo, o que los de 5 estrellas tienen hasta 10 palabras y un sentimiento positivo. Ahora vamos a realizar un análisis para diferentes valores de k del 1 al 20 y ver cuál es el mejor graficándolo:

```
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])
```

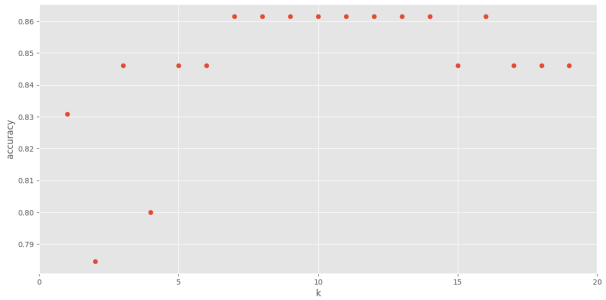

Figure 19: Análisis del valor óptimo de k.

Podemos ver que al tener `k` entre 7-14 y 16, nuestra precisión es más exitosa.

3 Resultados

Después de nuestra primera predicción nos encontramos que para 5 palabras y un nivel de sentimiento de 1 se nos valorará con 5 estrellas.

```
Prediccion 1:
[5]
```

Figure 20: Predicción para 5 palabras y sentimiento 1.

Si predecimos por coordenadas, para nuestras coordenadas (20, 0), existe un 97% de que nos den 3 estrellas.

```
Prediccion 2:
[[0.00381998 0.02520212 0.97097789 0.         0.        ]]
```

Figure 21: Predicción para coordenadas (20,0).

4 Conclusión

Es interesante y de hecho demasiado fácil usar el algoritmo KNN, es demasiado intuitivo y fácil de configurar. Sin embargo, la desventaja de que trabaje con conjuntos de datos pequeños nos encapsula en una burbuja donde probablemente nuestras predicciones serán muy generalizadas. Aun así, realizar predicciones me permitió entender mejor cómo funciona este algoritmo.

5 Referencias

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