k-Nearest Neighbor en Python

31 de Marzo de 2025

1 Introducción

k-Nearest Neighbors (k-NN) es un algoritmo de aprendizaje supervisado no paramétrico basado en instancias que clasifica nuevas observaciones según la mayoría de votos de sus k vecinos más cercanos en el espacio de características. Este modelo es más utilizado en problemas de clasificación cuando se tiene una variable dependiente categórica. Al igual que con el modelo Random Forest, k-NN también se puede emplear para la predicción de valores numéricos promediando los valores de sus vecinos más cercanos.

2 Metodología

Para realizar el ejercicio de k-NN, se siguieron los siguientes pasos:

```
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
```

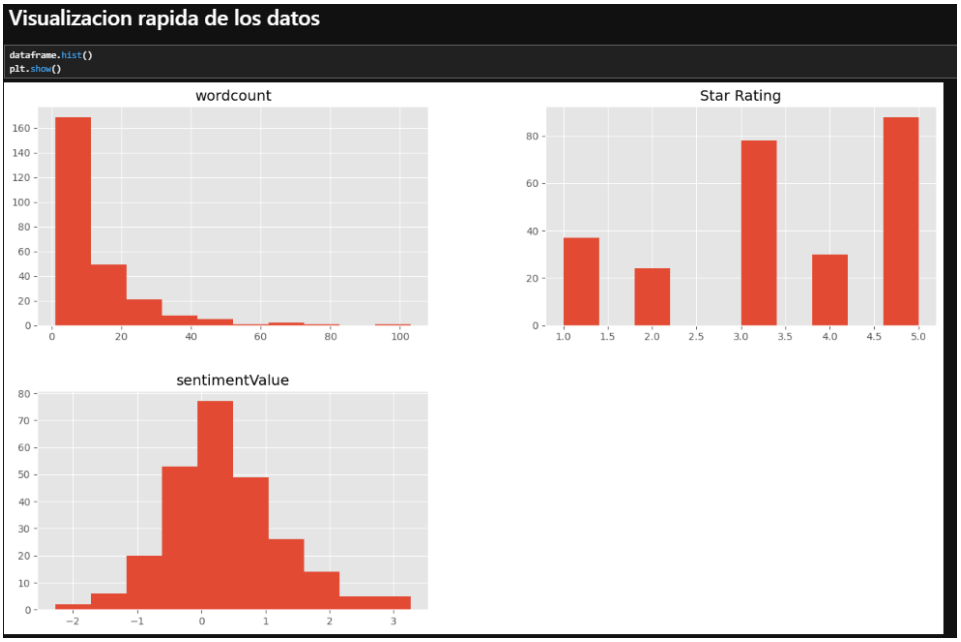
```
dataframe = pd.read_csv("reviews_sentiment.csv",sep=';')
dataframe.head(10)
```

[4]:

	Review Title	Review Text	wordcount	titleSentiment	textSentiment	S Rati
0	Sin conexión	Hola desde hace algo más de un mes me pone sin...	23	negative	negative	
1	faltan cosas	Han mejorado la apariencia pero no	20	negative	negative	
2	Es muy buena lo recomiendo	Andres e puto amoooo	4	NaN	negative	
3	Version antigua	Me gustana mas la version anterior esta es mas...	17	NaN	negative	

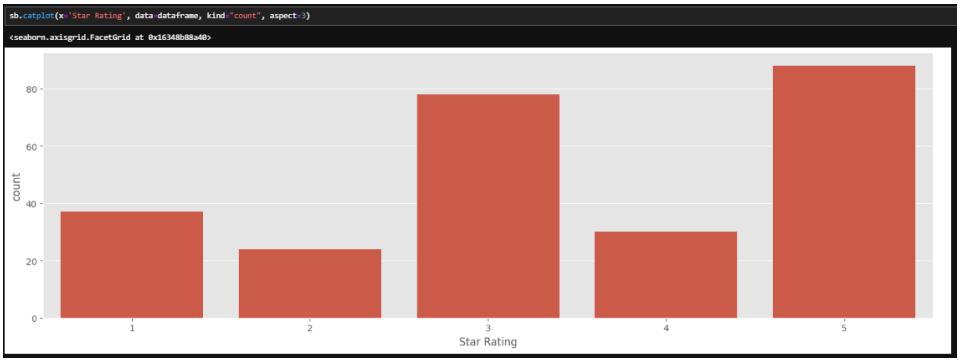
2.1 Visualización de los Datos de Entrada

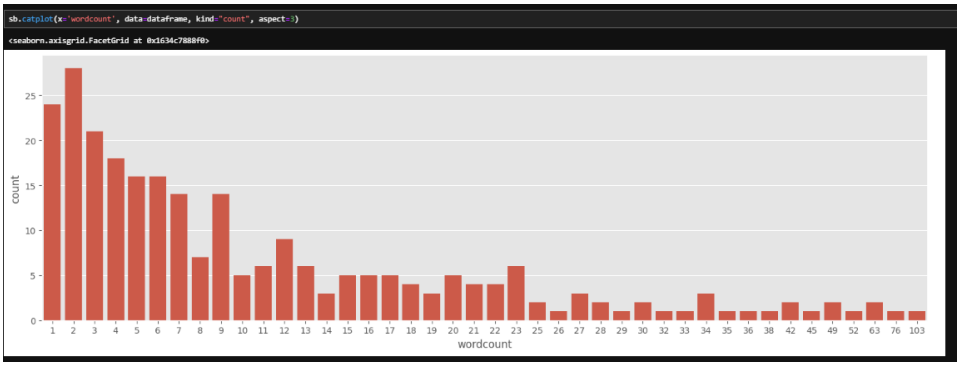
En esta sección se visualizan las distribuciones de las columnas del dataset. Se emplean las columnas de sentimiento del usuario y la cantidad de palabras en la reseña para predecir el número de estrellas.




```
print(dataframe.groupby("Star Rating").size())
```

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





2.2 Creación y Ajuste del Modelo

En esta sección se separan los datos en conjuntos de entrenamiento y prueba para ajustar el modelo k-NN.

Preparacion del 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)
```

Creacion del modelo

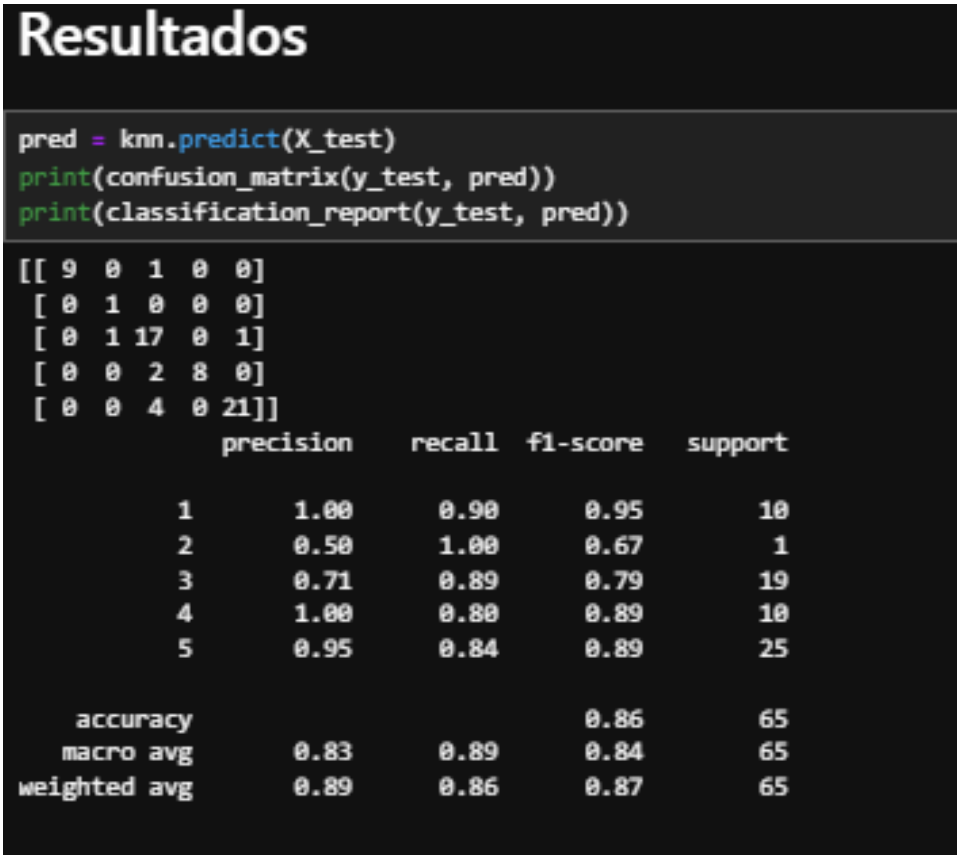
```
n_neighbors = 7

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)))
```

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

Dado que el dataset tiene relativamente pocos datos y que los valores en el número de estrellas no están muy desbalanceados, es mejor utilizar una cantidad de k más pequeña. En este caso, el modelo muestra una alta precisión, incluso con los datos de prueba que no ha visto previamente.

2.3 Evaluación del Modelo

En esta sección se realizan las predicciones del modelo y se imprime la matriz de confusión. Esta matriz muestra la diferencia entre la clase predicha y la clase real de cada observación del conjunto de prueba.



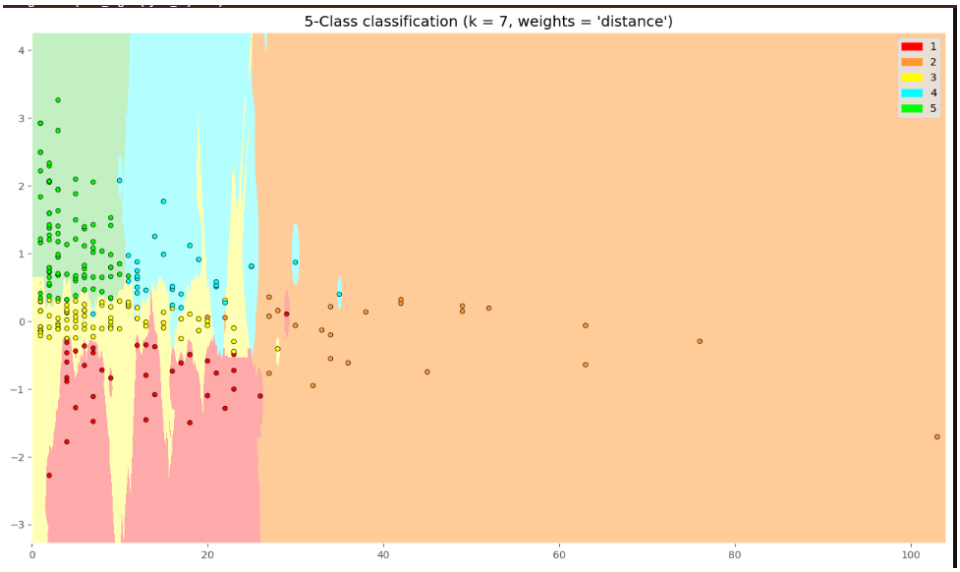
Podemos ver que el modelo casi siempre predice correctamente reseñas malas (1 estrella) y reseñas buenas (4 a 5 estrellas), pero tiene más problemas con las reseñas intermedias (2 a 3 estrellas). Sin embargo, el modelo clasifica correctamente más de la mitad de estos casos.

3 Resultados

En esta sección se grafican los resultados obtenidos en la sección anterior. En la gráfica de colores siguiente, los dos ejes representan las dos columnas de entrada y los espacios de colores representan cada una de las clases de la variable dependiente (el número de estrellas en la reseña).

Grafica de la clasificaion obtenida

```
h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#ffcc99', '#ffffb3','#b3ffff','#c2f0c2'])
cmap_bold = ListedColormap(['#FF0000', '#ff9933','#FFFF00','#00ffff','#00FF00'])

# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors, weights='distance')
clf.fit(X, y)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
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()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
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()
```

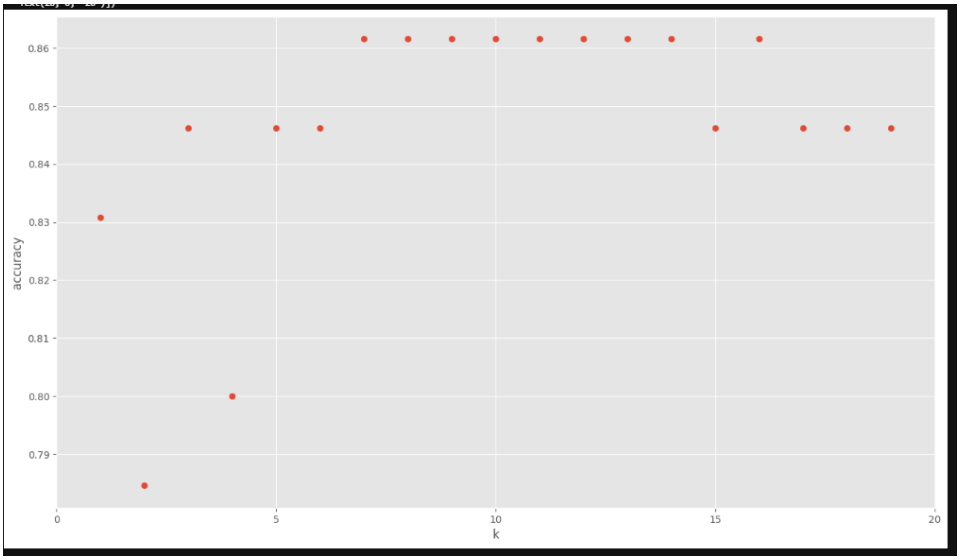


Como muestra la gráfica y el histograma de cantidad de palabras, la mayoría de las reseñas no son muy largas. Sin embargo, un patrón interesante es que las reseñas con muchas palabras suelen tener 2 estrellas. Por otro lado, las reseñas de 1 estrella y de 3 a 5 estrellas están concentradas en la parte izquierda de la gráfica, indicando que suelen tener pocas palabras.

Obtenemos el 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])
```

```
([<matplotlib.axis.XTick at 0x163699d4fb0>,
  <matplotlib.axis.XTick at 0x16369a05850>,
  <matplotlib.axis.XTick at 0x163699f9520>,
  <matplotlib.axis.XTick at 0x16369a0dd30>,
  <matplotlib.axis.XTick at 0x16369a0cef0>],
 [Text(0, 0, '0'),
  Text(5, 0, '5'),
  Text(10, 0, '10'),
  Text(15, 0, '15'),
  Text(20, 0, '20')])
```



Ahora se evalúa la cantidad óptima de vecinos k para mejorar la precisión del modelo. Como se observa en la gráfica, los mejores resultados se obtienen cuando k está entre 7 y 14 estimadores.

Predicciones

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

```
[5]
```

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

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

4 Conclusión

Los modelos de k-Nearest Neighbor (k-NN) son herramientas de aprendizaje supervisado que realizan predicciones basadas en la mayoría de votos de sus k vecinos más cercanos. A diferencia de otros modelos de clasificación por consenso, k-NN permite asignar pesos a las distancias entre los vecinos, favoreciendo valores más cercanos.

La cantidad de k en un modelo k-NN depende en gran medida de las características del dataset. Un conjunto de datos más pequeño o con clases balanceadas necesita menos vecinos k, mientras que un conjunto de datos más grande o con clases desbalanceadas requiere más k. Sin embargo, un número excesivo de vecinos puede causar sobreajuste y reducir la precisión del modelo con datos no vistos.

En este estudio, se descubrió que la relación entre la variable dependiente (número de estrellas en la reseña) y la cantidad de palabras es peculiar: las reseñas con más palabras suelen recibir 2 estrellas, mientras que las reseñas cortas pueden pertenecer a cualquier otra categoría. También se utilizó la columna de sentimiento del usuario, que resultó ser un buen predictor de la clasificación de reseñas. Finalmente, se encontró que la cantidad óptima de vecinos k para este modelo está entre 7 y 14.