Act14

Daniel Rojas

March 2025

1 Introducción

K-Nearest-Neighbors es un modelo supervisado del aprendizaje profundo utilizado para hacer predicciones entre clases. Además, es un modelo basado en instancia, lo que significa que memoriza los datos de entrenamiento para realizar predicciones con datos futuros. El modelo destaca por su fácil implementación y por su gran efectividad en conjuntos de datos pequeños.

Para hacer predicciones, el algoritmo calcula la distancia entre todos los puntos del dataset y el de entrada, usualmente se usa la distancia euclidiana para obtener la magnitud de la distancia; luego determina los k puntos más cercanos al dato de entrada, y, por último, la clase de la mayoría de los k puntos más cercanos define la clase predecida.

2 Metodología

Para realizar la actividad se usó como base el capítulo de K-Nearest Neighbors del libro Aprende Machine Learning, adjuntado en las referencias.

Para empezar, hay que preprocesar los datos, de tal forma que se seleccionen solo las columnas útiles para implementar el modelo de KNN. Para ello, hay que cargar el dataset, y simplemente utilizar las columnas de estrellas, valor de sentimiento y rencuento de palabras:

```
#Librerias a utilizar

#Librerias a utilizar

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
```

```
#Cargar el dataset
dataframe = pd.read_csv(r"reviews_sentiment.csv",sep=';')
dataframe.head()
```

```
#Crear los datasets de entrenamiento y testeo

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) #Dividir los datasets en entrenamiento y
    ↪ testeo
scaler = MinMaxScaler() #Rescala los valores de las features entre
    ↪ 0 y 1
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

En la anterior celda, se utilizó MinMaxScaler para reescalar los valores de los features a un rango de valores entre el 0 y el 1.

Ahora, hay que realizar un diseño de experimentos para encontrar el mejor valor de la K en el algoritmo K-NN. En general, se busca que por lo menos el valor sea impar para decidir empates entre clases.

```
#Buscar por el valor de k para los mejores resultados

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))

kmax = np.argmax(scores)+1
print(f'Mejor k = {kmax}, con score de: {round(scores[kmax-1]*100,
    ↪ 2)}%')
```

Con la anterior celda de código se obtiene que el mejor valor a utilizar es k = 7, con una precisión de 86.15%.

Por último, solo queda definir y entrenar un modelo knn con un valor de k de 7.

```
#Definir el knn con mejor valor de k

n_neighbors = kmax

knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print('Precisi n de K-NN en el dataset de entrenamieto: {:.2f}%'
      .format(knn.score(X_train, y_train)*100))
print('Precisi n de K-NN en el dataset de testeo: {:.2f}%'
      .format(knn.score(X_test, y_test)*100))
```

La anterior celda muestra que la precisión del modelo en el dataset de entrenamiento es de 89.58%, y en el dataset de testeo es de 86.15%.

3 Resultados

El diseño del experimento mostró que el mejor valor a utilizar para definir el algoritmo de K-Nearest Neighbors es de K = 7. O sea que se checa el número de estrellas de los 7 puntos más cercanos al comentario de entrada para predecir el número de estrellas con el que sería puntuado. Además, una K-NN con K = 7 mostró buenos resultados para el dataset de validación, a pesar de usar solo 2 features para realizar predicciones, del 86.15%.

4 Conclusión

El algoritmo K-Nearest Neighbors es verdaderamente útil para conjuntos de datos pequeños, ya que no le toma mucho tiempo calcular la distancia con los demás puntos del dataset. Además, y preferentemente, no se debería de poder visualizar graficamente la clasificación de cada punto, tanto en 2D como en 3D, ya que el algoritmo funciona mejor cuando se usan más features.

Por último, es necesario realizar un diseño de experimentos para encontrar el mejor valor de la K, ya que en si no hay una forma exacta de terminar cuál es el mejor número de vecinos para todos los datasets. Sin embargo, es completamente necesario que el valor de k sea impar, ya que de esta forma se evitan los empates.

5 Referencias

Bagnato, J. I. (2020). Aprende Machine Learning.