Nombre: Idalia Alejandra Delgado Moreno
Grupo: 31

Introducción

¿Qué es un K-Nearest-Neighbor?

Es un método de clasificación basado en la similitud entre puntos, como su nombre en inglés nos indica, busca el k vecino más cercano de un nuevo dato y le asigna la etiqueta mayoritaria de entre ellos. Es un algoritmo supervisado (requiere datos etiquetados) y basado en instancia (no construye un modelo, sino que memoriza los datos de entrenamiento).

Aplicaciones: se usa en sistemas de recomendación, búsqueda semántica y detección de anomalías.

Ventajas y desventajas: es fácil de implementar, pero consume mucha memoria y procesamiento, ya que compara cada punto con todo el conjunto de entrenamiento. Funciona mejor en datasets pequeños con pocas características.

Funcionamiento:

Calcula la distancia entre el punto nuevo y los datos de entrenamiento.

Selecciona los k vecinos más cercanos.

Asigna la clase predominante entre ellos.

El valor de k es clave para la precisión del modelo. Valores pequeños pueden causar sobreajuste, mientras que valores grandes pueden hacer que el modelo pierda precisión. Las distancias más usadas son la Euclidiana y la Cosine Similarity.

Este algoritmo es efectivo para problemas con pocas dimensiones, pero su rendimiento depende de una buena elección de k y de las características utilizadas.

Metodología

Para realizar esta actividad, se debe seguir las instrucciones del documento y asegurarse de tener todas las librerías necesarias instaladas.

Importamos las librerías necesarias para el análisis

```
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
```

Cargamos el dataset y visualizamos los primeros registros

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

Generamos estadísticas básicas del dataset

```
print(dataframe.describe())
print(dataframe.groupby('Star Rating').size())
```

Visualizamos la distribución de datos con gráficos

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

sb.factorplot('Star Rating', data=dataframe, kind="count", aspect=3)
sb.factorplot('wordcount', data=dataframe, kind="count", aspect=3)
```

Preparamos los datos para el modelo

```
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)
```

Entrenamos un modelo de k-Nearest Neighbors

```
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
```

Hacemos predicciones y evaluamos el modelo

```
predictions = knn.predict(X_test)
print("Classification Report:\n", classification_report(y_test, predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, predictions))
```

Resultados

A continuación, se presentan los resultados obtenidos del modelo k-Nearest Neighbors. Se incluye la selección del mejor valor de k y la clasificación obtenida.

Clasificación obtenida

La siguiente gráfica nos ayuda a ver fácilmente en donde caerán las predicciones.

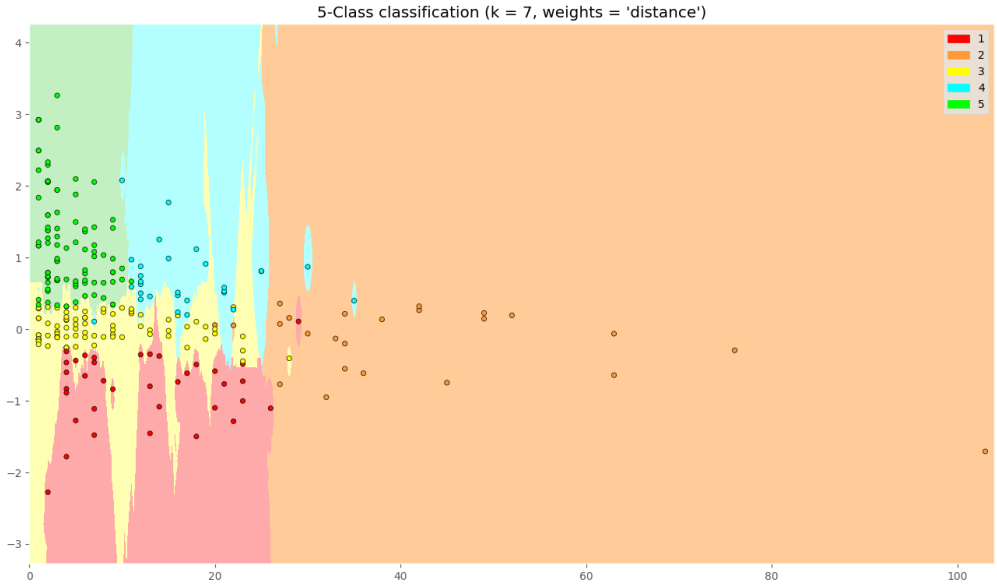

Figure 1: Clasificación de los datos con el modelo k-NN. Se observa una correcta separación de clases.

Interpretación de las zonas de clasificación

Con con la gráfica anterior, podemos asumir las siguientes cosas:

- Los usuarios que ponen **1 estrella** tienen sentimiento negativo y hasta 25 palabras.
- Los usuarios que ponen **2 estrellas** dan muchas explicaciones (hasta 100 palabras) y su sentimiento puede variar entre negativo y algo positivo.
- Los usuarios que ponen **3 estrellas** son bastante neutrales en sentimientos, puesto que están en torno al cero y hasta unas 25 palabras.
- Los usuarios que dan **5 estrellas** son bastante positivos (de 0.5 en adelante, aproximadamente) y ponen pocas palabras (hasta 10).

Selección del mejor valor de k

Para determinar el mejor valor de k, se realizó un análisis de error en función de distintos valores de k.

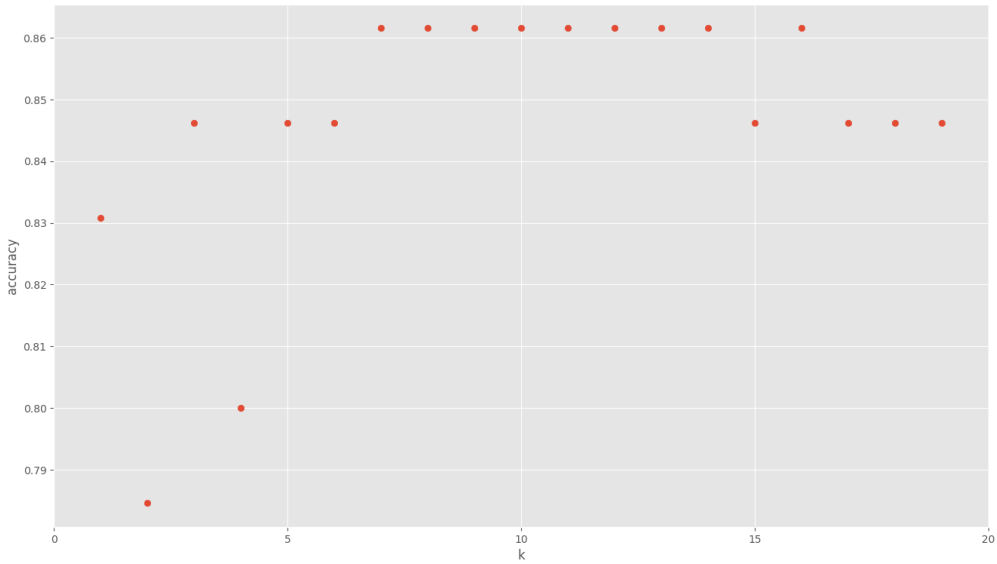

Figure 2: Error en función de diferentes valores de k. Se observa que el valor óptimo minimiza el error.

En la gráfica vemos que con valores k=7 a k=14 es donde mayor precisión se logra.

Conclusión:
En este ejercicio aplicamos el algoritmo k-Nearest Neigbors para clasificar las reseñas de una apliación tomando en cuenta dos variables: la cantidad de palabras usadas y el valor de sentimiento. Además, se observó que el parámetro k es suma importancia para el modelo ya que ya que valores demasiado pequeños pueden llevar a sobreajuste, mientras que valores grandes pueden generar predicciones menos precisas. Los resultados muestran que el modelo es capaz de diferenciar entre distintos niveles de satisfacción de los usuarios, aunque con algunas limitaciones debido a la simplicidad de las características seleccionadas. En una situación real, sería recomendable incluir más atributos para mejorar la precisión del modelo. En conclusión, k-NN es una técnica efectiva para clasificación en problemas con pocas dimensiones, pero su rendimiento depende de una adecuada selección de características, del valor de k y del tamñano de los datos.