# Actividad 14: Programando K-Nearest-Neighbor en Python

*García Herrera Carlos Eduardo Marzo 2025*

## 1 Introducción

K-Nearest Neighbors (KNN) es un algoritmo de aprendizaje supervisado utilizado principalmente para tareas de clasificación y regresión. Se basa en la idea de que las instancias similares tienden a estar cerca unas de otras en el espacio de características. En términos sencillos, KNN clasifica o predice un valor para un nuevo punto de datos, basándose en los k puntos de datos más cercanos en el conjunto de entrenamiento.

## 2 Metodología

### 2.1 Parte 1: Creación del Ambiente Virtual

```
#Automatic creation of an virtual environment to run the script and intall
the libraries
import subprocess
import os
import venv
import sys
script_dir = os.path.dirname(os.path.realpath(__file__))
env_name = os.path.join(script_dir, "VirtualEnv")
if os.path.exists(os.path.join(script_dir, "VirtualEnv")):
    #Checks if the VirtualEnv is activated (This is the path to the Python
installation currently in use. If the virtual environment is active,
sys.prefix will point to the virtual environment directory, while
sys.base_prefix points to the global Python installation.)
    if sys.prefix == sys.base_prefix:
        print("Activating the Virtual Environment...")
        python_exe = os.path.join(env_name, "Scripts", "python")
        subprocess.run([python_exe, __file__])
else:
    print("Installing the Required Libraries on a New Virtual Environment")
    venv.create(env_name, with_pip=True)

    # Step 2: Install the libraries
    libraries = ["scikit-learn", "matplotlib","seaborn","pandas","numpy"]
```

```
    for lib in libraries:
        subprocess.run([os.path.join(env_name, "Scripts", "pip"), "install",
lib], check=True)

    #Re-Run the script with the Virtual Env Activated
    python_exe = os.path.join(env_name, "Scripts", "python")
    subprocess.run([python_exe, __file__])
```

### 2.2 Parte 2: Importación de las librerías Necesarias

```
#Random Forest
#Importacion de Librerias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

from matplotlib import rcParams

from collections import Counter
```

### 2.3 Parte 3: Carga de los datos de Entrada y análisis de los registros

```
dataframe = pd.read_csv(r"reviews_sentiment.csv",sep=';')
pd.set_option('display.max_columns', None) #Muestra todas las columnas del
dataSet

print("Primeros 10 Registros")
print(dataframe.head(10))
print("\nDescripcion del DataFrame:")
print(dataframe.describe())
```

### 2.4 Parte 4: Visualización de los Datos

```
#Visualizaciones Rapida
```

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

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

sb.catplot(data=dataframe,kind="count", x="Star Rating",palette="viridis")
plt.show()

sb.catplot(data=dataframe,kind="count", x="wordcount", palette="viridis")
plt.show()
```

### 2.5 Parte 5: Creacion del Modelo

```
#Creacion del Modelo
n_neighbors = 7

knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print("\n\nPrecision del Modelo")
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)))
```

### 2.6 Parte 6: Resultados

```
#Resultados
pred = knn.predict(X_test)
print("\n\nMatriz de Confusion")
print(confusion_matrix(y_test, pred))

print("\n\nReporte de Clasificacion")
print(classification_report(y_test, pred))
```

### 2.7 Parte 7: Creacion de la Grafica de Clasificacion

```
#Grafica de Clasificacion
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()
```

### 2.8 Parte 8: Obtención del Mejor valor de K

```
#Obtener 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])
plt.show()
```

### 2.9 Parte 9: Predicciones

```
#Predicciones
print("\n\nPrediccion 1 (5 palabras y sentimiento 1):")
print(clf.predict([[5, 1.0]]))

print("\n\nPrediccion 2 con probabilidad en las coordenadas (20, 0.0)")
print(clf.predict_proba([[20, 0.0]]))

input("Presiona cualquier tecla para Cerrar...")
```

# 3 Resultados

Al ejecutar el script de Python la información obtenida es la siguiente, cabe recalcar que la información es obtenida de forma secuencial a como se muestran a continuación:

## 3.1 Vista de los Datos

Se hace un análisis de los datos al ver distintas características de ellos, por medio de distintas graficas de barras para poder reconocer ciertas relaciones entre ellos.

Primeros 10 Registros

| | Review Title | Review Text | wordcount | titleSentiment | textSentiment | Star Rating | sentimentValue |
|---|---|---|---|---|---|---|---|
| 0 | Sin conexión | Hola desde hace algo más de un mes me pone sin... | 23 | negative | negative | 1 | -0.486389 |
| 1 | faltan cosas | Han mejorado la apariencia pero no | 20 | negative | negative | 1 | -0.586187 |
| 2 | Es muy buena lo recomiendo | Andres e puto amoooo | 4 | NaN | negative | 1 | -0.602240 |
| 3 | Version antigua | Me gustana mas la version anterior esta es mas... | 17 | NaN | negative | 1 | -0.616271 |
| 4 | Esta bien | Sin ser la biblia.... Esta bien | 6 | negative | negative | 1 | -0.651784 |
| 5 | Buena | Nada del otro mundo pero han mejorado mucho | 8 | positive | negative | 1 | -0.720443 |
| 6 | De gran ayuda | Lo malo q necesita de …,pero la app es muy buena | 23 | positive | negative | 1 | -0.726825 |
| 7 | Muy buena | Estaba más acostumbrado al otro diseño, pero e... | 16 | positive | negative | 1 | -0.736769 |
| 8 | Ta to guapa. | Va de escándalo | 21 | positive | negative | 1 | -0.765284 |
| 9 | Se han corregido | Han corregido muchos fallos pero el diseño es ... | 13 | negative | negative | 1 | -0.797961 |

*Ilustración 1: Primeros 10 registros*

Descripcion del DataFrame:

| | 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 |

*Ilustración 2: Descripción estadística de los datos*


*Ilustración 3: Grafica de barras de las distintas características de los datos*

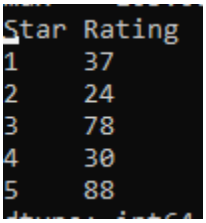


*Ilustración 4: Cantidad de Datos por # de Estrellas*

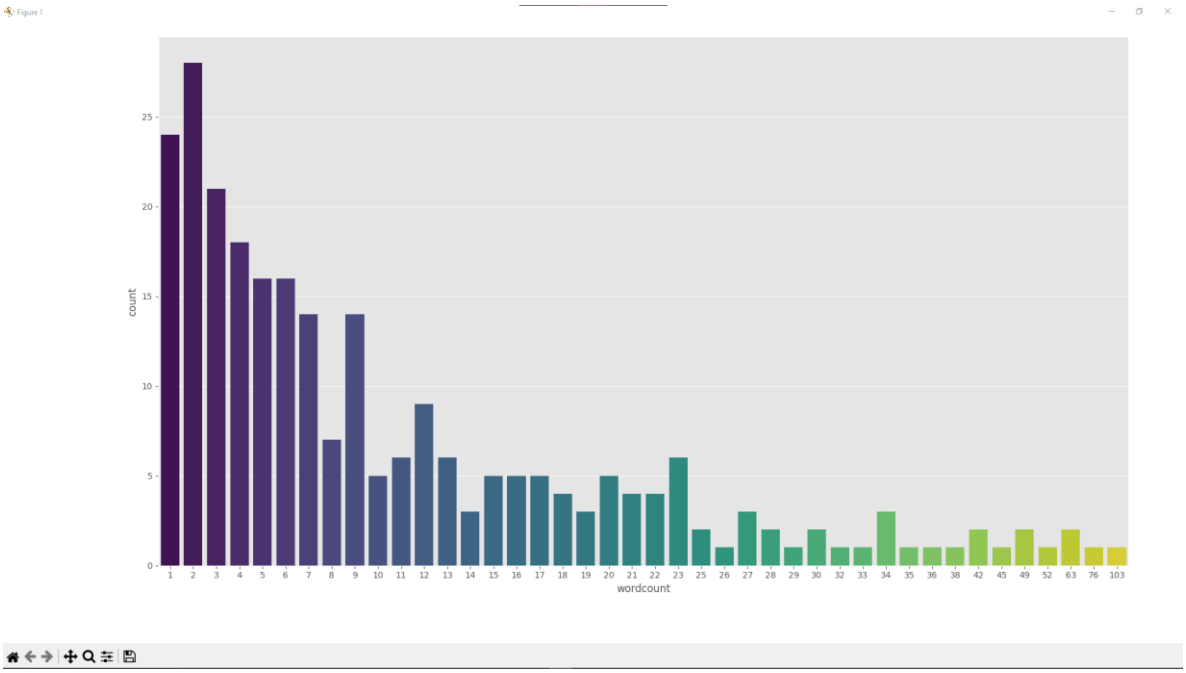


*Ilustración 5: Cantidad de Palabras por review*

## 3.2 Calculo de KNN

Una vez que se calculo el modelo con los datos de entrenamiento, se obtuvieron los siguientes valores de precisión:

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

*Ilustración 6: Precisión del Modelo Entrenado*

```
[[ 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]]
```

*Ilustración 7: Matriz de Confusión del modelo*

| | 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 |

*Ilustración 8: Reporte de clasificación*

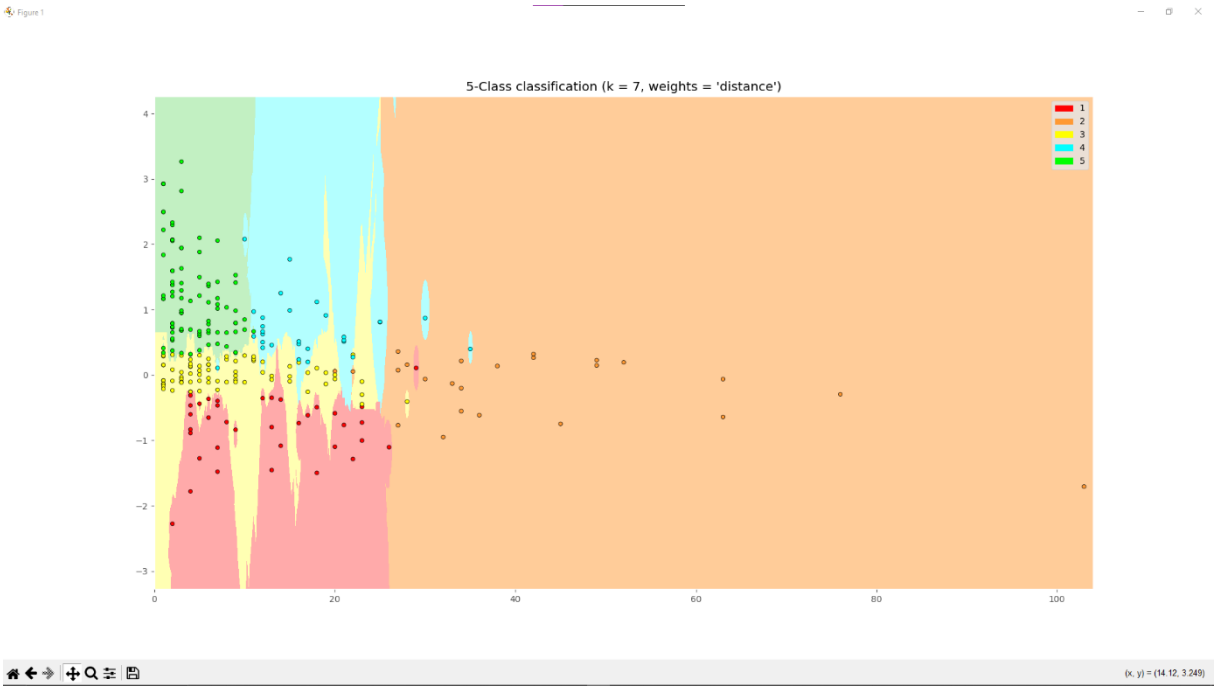


*Ilustración 9: Grafica de clasificación*

## 3.3 Obtención del mejor valor de K

Para la obtención del mejor valor de K, se ejecuto la parte del código 2.8, para obtener la grafica siguiente:

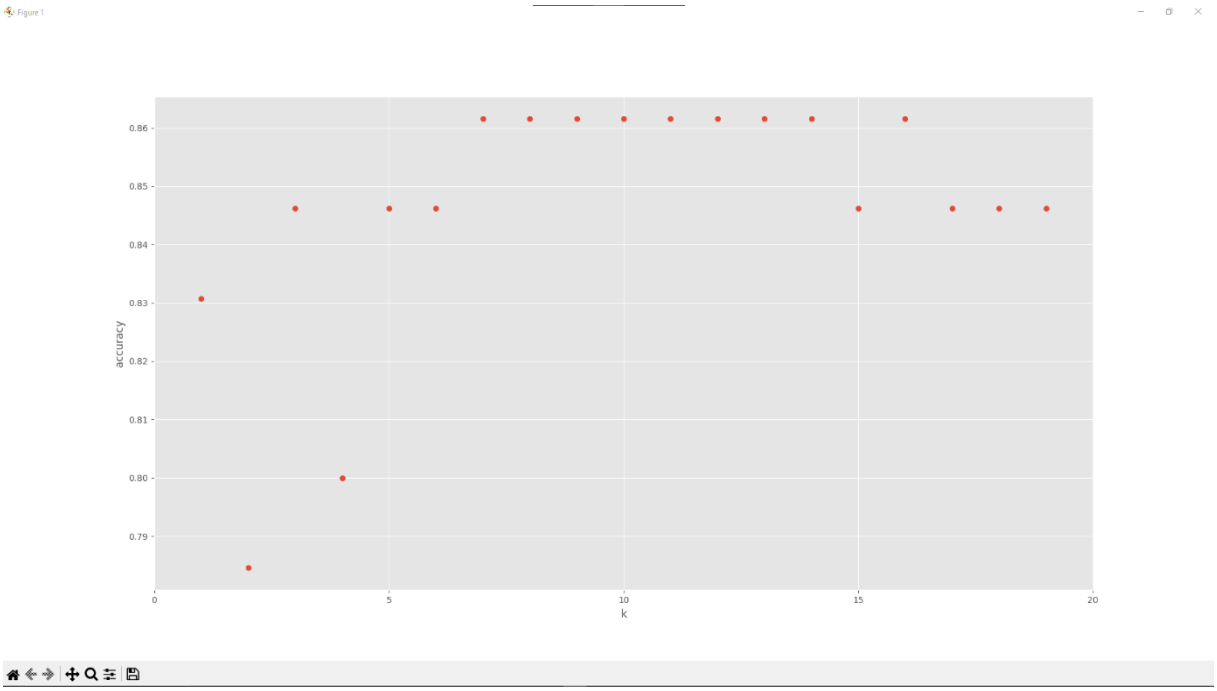


*Ilustración 10: Grafica en donde se registra los mejores valores de K*

## 3.4 Predicciones

A continuación, se muestran 2 predicciones hechas por medio del modelo KNN

```
Prediccion 1 (5 palabras y sentimiento 1):
[5]
```

*Ilustración 11: Predicción 1, en donde se predice que tendrá 5 estrellas*

```
Prediccion 2 con probabilidad en las coordenadas (20, 0.0)
[[0.00381998 0.02520212 0.97097789 0.         0.        ]]
```

*Ilustración 12: Predicción 2, en donde hay un 97% de posibilidad que se califique con 3 estrellas (los demás elementos del arreglo, cuentan con probabilidades muy bajas)*

# 4 Conclusión

En conclusión, K-Nearest Neighbors (KNN) es un algoritmo de clasificación y regresión simple pero poderoso, basado en la idea de que los puntos de datos cercanos en el espacio de características tienen más probabilidades de compartir la misma clase o valor. Su principal ventaja es su facilidad de implementación y flexibilidad, ya que no requiere un modelo explícito de entrenamiento, y se adapta bien a diferentes tipos de datos.

Sin embargo, KNN también presenta algunas limitaciones, como la alta complejidad computacional durante la predicción, especialmente cuando se trabaja con grandes volúmenes de datos, ya que necesita calcular la distancia a cada punto de entrenamiento. Además, su rendimiento depende significativamente de la elección del parámetro k y de la escala de las características.

A pesar de estos desafíos, KNN sigue siendo útil en muchas aplicaciones prácticas debido a su capacidad para manejar datos complejos sin suposiciones fuertes sobre la distribución de los mismos. Es una buena opción cuando la interpretabilidad es clave y cuando los datos no presentan una estructura lineal compleja.