Implementacion de Algoritmo KNN para el set de datos Iris

1st Jose Rojas Calderon
Estudiante Ingenieria en Computacion)
Instituto Tecnologico de Costa Rica)
Cartago, Costa Rica
Josedanielrojas5@estudiantec.cr

***Abstract*—La investigation se realizo haciendo la aplicación del algoritmo K-Nearest Neighbors (KNN) para la clasificación de especies de Iris. Se utilizó conjunto de datos Iris donde el objetivo principal fue evaluar el desempeño del algoritmo KNN en esta tarea de clasificación y determinar el valor óptimo del parámetro K.**

Se implementó el algoritmo KNN en Python utilizando las bibliotecas Pandas, Numpy y Sklearn para el procesamiento y analisis de los datos y las bibliotecas de Matplotlib y Seaborn para la generacion de graficos. Se realizaron experimentos variando el valor de K y se evaluó el rendimiento del modelo utilizando la métrica de "accuracy".Los resultados obtenidos mostraron que el algoritmo KNN es capaz de clasificar las especies de Iris con una precisión aceptable al seleccionar ciertos valores de K.

I. Introduccion

La clasificación de datos es una tarea fundamental en el aprendizaje automático. El algoritmo KNN, un método de clasificación que es ampliamente utilizado debido a su simplicidad y eficacia. En este estudio, se aplicó el algoritmo KNN al conjunto de datos Iris, un conjunto de datos que tiene informacion sobre la especie de plantaas Iris, con el objetivo de evaluar su capacidad para clasificar correctamente las diferentes especies de Iris.

II. Visualizacion de caracteristicas

Dentro del analisis de las caracteristicas de los datos notamos una clara tendencia entre en lardo del sepalo y el ancho del sepalo. Hay una diferencia marcada entre los tipos Iris Setosa y Iris versicolor y Iris virginica. El tipo Setosa, muestra una diferencia muy marcada con respecto a las otras 2. como se observara en la figura 1, la versicolor y la virginica comparten mas las caracteristicas del sepalo qu ela Setosa.

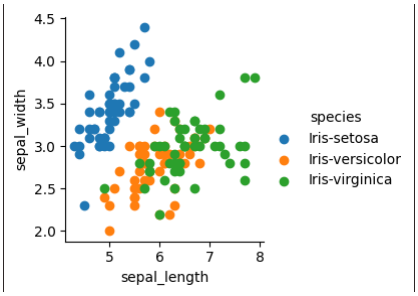

Fig. 1. Ejemplo de relacion entre el largo del sepalo y el ancho por especie.

Tambien se hizo un analisis con respecto a la longitud del petalo y el ancho del petalo. En este grafico se aprecia como la distribucion esta mas separada. Aun cuando Iris Versicolor e Iris Virginica se encuentran mas cerca, hay una diferencia significativa en cada una. En la figura 2 se ejemplifica la distancia entre cada una de las especies.

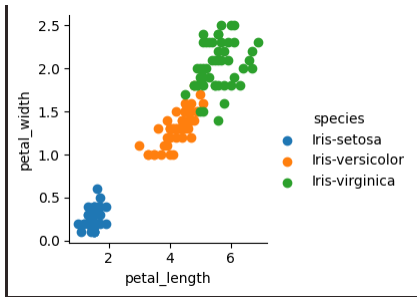

Fig. 2. Ejemplo de relacion entre el largo y el ancho del petalo por especie.

Ademas se hizo un analisis de la longitud del sepalo por especie donde se nota una clara diferencia en la distribucion aun cuando se analiza la misma especie, lo que hace pensar que probablemente seria bueno tener mas datos para hacer el analisis, edad de la plata para asi comparar la longitud con la edad y tener resultados mas asertados. En la figura 3 se puede ejemplificar la diferencia en la longitud del sepalo aun cuando son de la misma especie.

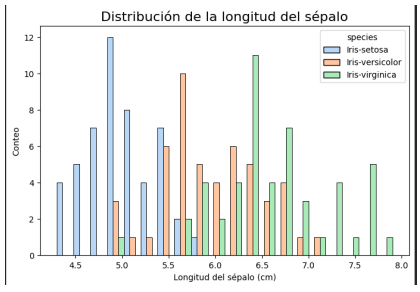

Fig. 3. Ejemplo de cantidad de plantas segun el largo del sepalo.

Ahora al hacer un analisis de la longitud y ancho del sepalo por especie, se puede ver como siguen una distribucion mas uniforme. En la figura 3 se realizo la comparativa pero separando los datos por especie.

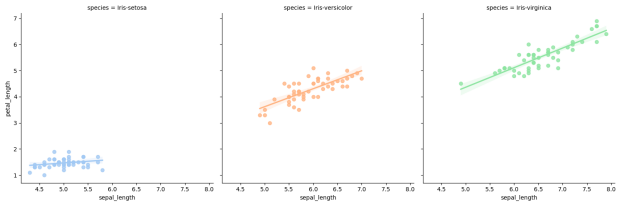

Fig. 4. Ejemplo del largo del sepalo de las plantas separado por cada especie.

III. Implementacion de KNN sobre Iris

Ahora vamos a explicar como se realizo la implementacion del algoritmo KNN sobre el set de datos Iris.Primeramente se hizo una separacion de los datos y de las clases de las plantas (caracteristicas y etiquetas). Primeramente se seleccionan las caracteristicas que se va a utilizar, en el caso de la variable X, se seleccionaron todas las columnas menos la ultima la cual fue seleccionada en la variable Y la cual contiene las etiquetas, en este caso son las especies de Iris. Se hizo una seleccion del 80 por ciento de los datos para el entrenamiento y un 20 porciento para el set de pruebas.

A. Separacion de variables

La variable dataTrain contiene contiene 120 datos los cuales se van a utilizar para el entrenamiento, estos fueron seleccionados en orden, es decir que se seleccionaron del 0 al 119. Por otro lado labelTrain contiene las etiquetas de esos primeros 120 datos. Para el set de pruebas, simplemente se seleccionaron los datos restantes y se asignaron a la variable dataTest, cabe aclarar que labelTest es usado para comparar los datos al final del algoritomo KNN.

```
#Separacion de datos y de las clases de las plantas.
X = df.iloc[:, :-1].values
#X = df.iloc[:, [0, 2, 3]].values
#X = df.iloc[:, [0, 2]].values  # Usando las características 0 y 2
y = df.iloc[:, -1].values

# Separar el conjunto de entrenamiento
trainSize = int(0.8 * len(X))
dataTrain= X[:trainSize]
labelTrain= y[:trainSize]

#conjunto de entrenamiento
dataTest= X[trainSize:]
labelTest= y[trainSize:]
```

Fig. 5. Separacion de los datos de entrenamiento y de los datos de prueba.

B. Calculo de distancia Euclidiana

Para la ejecucion del KNN se necesita calcular la distancia euclidiana entre dos puntos, primero se toman 2 vectores, en este caso x1 y x2, se resta cada valor de los 2 vectores y se eleva al cuadrado la diferencia, posterior a esto suma todos los cuadrados para calcular la raiz cuadrada de la suma y esta seria la distancia euclidiana.

```
#calcular la distancia euclidiana
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2)**2))
```

Fig. 6. Ejemplo de la funcion para el calculo de la distancia Euclidiana.

C. Implementacion de K-Nearest Neighbor

Para la implementacion del algoritmo knn vamos a seguir los siguientes pasos: Iteración sobre los puntos de prueba y se calcula la distancia de todos los puntos del conjunto de entrenamiento. Luego procedemos a encontrar los k vecinos mas cercanos, esto ordenanandno las distancias con la ayuda de la funcion argsort. Posterior a esto debemos hacer el proceso de votacion donde se obtiene las etuiquetas de clase de los k vecinos mas cercanos. La etiqueta de clase que aparece con mas frecuencia entre estos, se selecciona como la prediccion de el punto y se procede a almacenarla en la el array pred. Finalmente retornamos el array pred con todas las predicciones para cada punto.

```
#Calculo de KNN
pred=[]
def knn_predict(dataTrain, pLabelTrain, dataTest, k):
    for x in dataTest:
        # Calcular distancias para los puntos
        distances = [euclidean_distance(x, data) for data in dataTrain]
        # Encontrar los k vecinos más cercanos
        k_indices = np.argsort(distances)[:k]
        # Votar
        kNearestLabels = pLabelTrain[k_indices]
        commonLabel = Counter(kNearestLabels).most_common(1)[0][0]
        pred.append(commonLabel)
    return np.array(pred)
```

Fig. 7. Ejemplo del codigo de KNN.

D. Resultados de pruebas

A continuacion vamos a explorar los resultados de las pruebas de KNN con diferentes valores de K.

```
Resultados del algoritmo KNN con k=3:
------------------------------
Predicciones:
['Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor']
Valores de prueba:
['Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica']
------------------------------
Accuracy: 76.66666666666667 %
```

Fig. 8. Resultados de prueba con k=3.

Fig. 9. Resultados de prueba con k=8

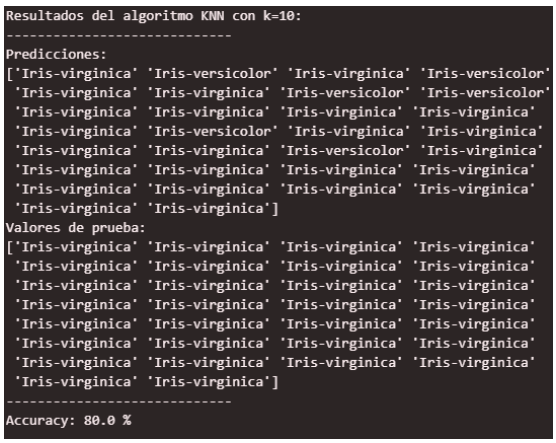

Fig. 10. Resultados de prueba con k=10.

Fig. 11. Resultados de prueba con k=20.

IV. Conclusion

El algoritmo KNN es una herramienta util para la clasificación, pero su rendimiento puede variar dependiendo del problema y los datos. Es importante experimentar con diferentes valores de k y realizar una adecuada preparacion de los datos para obtener los mejores resultados. La calidad de los datos de entrenamiento y prueba afectara directamente la precision del modelo. Algunas de los detalles que se notaron durante esta investigacion es que agregar un dato como por ejemplo edad de las plantas seria de ayuda para lograr clasificar mejor las plantas ya que el tamaño del petalo o del sepalo, puede variar dependiendo de la edad de la planta. Aunque las variaciones en el valor de k no parecen afectar drásticamente los resultados en este caso particular, es importante recordar que la elección de k puede tener un impacto significativo en el rendimiento del algoritmo en otros conjuntos de datos.

Podemos observar una clara dominancia de dos de las clases en particular, Iris versicolor o Iris virginica. Esto parece indicar un desbalance en los datos de entrenamiento o que la clase mas repetida puede ser mas facil de clasificar. Parece que la seleccion de los datos para entrenamiento no fue el mas adecuado y que ese desbalance fue producto de la forma en la que se seleccionaron los porcentajes.

Rúbrica:

Criterios	Puntuación máxima	Puntuación obtenida
Descarga y lectura del dataset	10	
Análisis de Características	20	
Implementación de KNN y experimentación de parámetros	30	
Evaluación del modelo con las métricas recomendadas	15	
Presentación de los resultados	15	
Estructura y claridad del informe	10	