PRACTICA 01: “METODO K-NEAREST NEIGHBORS EMPLEANDO DATASET DE CARACOLES ABALONES PARA PREDECIR SUEDAD EMPLEANDO EL NUMERO DE ANILLOS” VERSIÓN EN ESPAÑOL

Elaborado por: Pasante de Ingeniería Bioquímica Vargas Carreño Leonardo Augusto

1. Breve descripción del Abalón.

1.1. Haliotis

Imagen de Haliotis lamellosa. Imagen obtenida de Haliotis lamellosa - Haliotis - Wikipedia, la enciclopedia libre

Los haliótidos (Haliotidae) son una familia de moluscos gasterópodos con un único género, Haliotis. Son conocidos como abalones, u orejas de mar. La palabra abulón deriva de la palabra Ohlone awlun o aluan. Pasó del castellano al inglés de vuelta al castellano.

1.2. Morfología General.

Imagen de Haliotis tuberculata. Imagen obtenida de Haliotis tuberculata tuberculata - Haliotis - Wikipedia, la enciclopedia libre

Tienen una conchilla larga, plana, de forma ovalada, en una espiral de dos o tres vueltas, la más externa de las cuales les da su característica forma auricular. Tiene de cuatro a diez orificios a través de los cuales expulsan en caso de peligro el agua contenida en el interior de la concha, de modo a sujetarse más firmemente a su sustrato. El exterior de la conchilla es rugoso y de color variable; el interior es de nácar irisado y muy vistoso. El cuerpo de los haliótodos sigue el modelo de los moluscos, presenta tres regiones: el pie, la masa visceral y el manto. (Wikipedia, 2023)

2. Objetivos de la Práctica.

2.1. Realizar un algoritmo empleando Python que pueda predecir la edad de los Abalones empleando datos que se han recolectado de sus características morfológicas.

2.2. Comprender el fundamento matemático detrás del Método K-Nearest Neighbor (K-NN).

2.3. Realizar un código de K-NN empleando Python desde cero empleando la librería numpy.

2.4. Implementar Scikit-Learn para ajustar KNN con una cantidad mínima de código.

2.5. Usar GridsearchCV para encontrar los mejores hiperparametros de KNN.

2.6. Llegar al máximo rendimiento de KNN usando el embolsado de datos (bagging method).

NOTA: La práctica está basada en el proyecto de Use KNN to Predict the Age of Sea Slugs para su mejor estudio del Método KNN pueden buscar está misma en la bibliografía consultada. (Korstanje, 2021)

Elaborado por: Pasante de Ingeniería Bioquímica Vargas Carreño Leonardo Augusto

3. Librerías y Descargar DataSet Abulones.

Las librerías y en general todo el código lo podrán consultar en el archivo PRACTICA_01_KNN_DATASET_ABALONES.ipynb anexado en esta carpeta, pero por fines prácticos y explicativos se irán agregando algunos pasos.

3.1. Librerías.

En lo que llevo aprendiendo a programar se me hizo más fácil hacer un archivo en Python con las librerías, el archivo lo llame LIBRERIAS.py en todos los algoritmos que he realizado así tengo almacenadas las librerías y ordenas. Como se puede apreciar en la Imagen 3.1. Librerías para el Método K-NN.

Para poner las librerías en funcionamiento en donde se vayan a usar ya se en un archivo Python (.py) o en un archivo Jupiter Notebook (.ipynb), se emplea el siguiente código:

```
from NOMBRE_ARCHIVO import *
```

LIBRERIAS.py

```
# Librerias Basicas (Basic Libraries)

import pandas as pd

import numpy as np

import math

from math import sqrt

from tabulate import tabulate

import seaborn as sns


# Librerias para Descripción Estadística (Stactic Descripcion Libraries)

import matplotlib.pyplot as plt

# ALgoritmo K-NN  (K-NN ALgorithm)

import scipy

import scipy.stats

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import BaggingRegressor

# Método KNeighborsClassifier

from sklearn.neighbors import KNeighborsClassifier

# Método KNeighborsRegressor

from sklearn.neighbors import KNeighborsRegressor
```

Imagen 3.1. Librerías para el Método K-

3.2. Descargar DataSet Abulones.

Esta parte es más emplear Web Scrapping, la URL a emplear es la siguiente:

```
"https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"
```

El código sería el siguiente:

```
url= ("https://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data")
abalones= pd.read_csv(url, header=None)
```

```
abalones= pd.read_csv(url, header=None)
```

Elaborado por: Pasante de Ingeniería Bioquímica Vargas Carreño Leonardo Augusto

4. Estudio del DataSet

En el tiempo que llevo aprendiendo los fundamentos de Machine Learning para Data Science en todos los algoritmos que he estudiado se realiza un previo estudio del Dataset del cómo se encuentra estructurada la información. Aquí se emplean métodos (comandos) ya sea la librería PANDAS o NumPy.

4.1. Shape.

El comando .shape funciona para conocer el número de columnas y la cantidad de datos que contiene el DataFrame, el código sería el siguiente:

```
InPut:  abalones.shape
```

OutPut: (4177, 9), da como salida una tupla con la información del DataFrame, con: 4177 Datos y 9 Columnas.

4.2. Info().

El comando .info() muestra la información por la que se compone el DataFrame, el código es el siguiente:

```
InPut:    abalones.info()
```

OutPut:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       4177 non-null   object
 1   1       4177 non-null   float64
 2   2       4177 non-null   float64
 3   3       4177 non-null   float64
 4   4       4177 non-null   float64
 5   5       4177 non-null   float64
 6   6       4177 non-null   float64
 7   7       4177 non-null   float64
 8   8       4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

Revisamos la Información que hay en el

Hay 9 Columnas, como podemos observar no tienen atributo, más adelante se arregla eso.

Hay DataFrames en los que podremos tener la leyenda Null (vacio) en este caso no hay.

Podemos Corroborar que contiene 4177 datos de los cuales tenemos del tipo: Objeto, Decimales y Enteros.

4.3. Agregar atributos (nombres) a las columnas:

Se agregan los atributos de acuerdo al orden en el que están los datos: Sexo, Largo, Diámetro, Altura, Peso entero, Peso descascarado, Peso de vísceras, Peso de concha y Anillos. El Código es el siguiente:

```
abalones.columns= ["Sex", "Lenght", "Diameter", "Height", "Whole Weight", "Shucked Weight",
"Viscera Weight", "Shell Weight", "Rings"]
```

Antes:

```
InPut: abalones.head()
```

OutPut:

	0	1	2	3	4	5	6	7	8
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Después:

```
InPut: abalones.head()
```

OutPut:

	Lenght	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

4.4. Remover Datos.

Revisar y Estudiar previamente el DataFrame para así filtrar la información que no vamos a emplear, para este caso removeremos datos que no sean cuantificables. La columna Sexo para el método K-NN, no tienen utilidad para este caso en especifico. El código es el siguiente:

```
InPut: abalones= abalones.drop("Sex", axis=1)
```

OutPut:

	Lenght	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

5. Descripción Estadística del DataFrame Abalones.

La librería PANDAS tiene varios comandos con los que podemos operar con nuestro DataFrame.

5.1. Describe

El comando .describe() nos permite hacer una descripción estadística conocer el número de datos, la media, la desviación estándar, mínimos y máximos por default también nos permite conocer los percentiles (0.25, 0.50, 0.75) del DataFrame. El código es el siguiente:

```
InPut: abalones.describe(percentiles=(.25, .5, .75)).transpose()
  # .transpose() permite cambiar la ubicación de las columnas.
```

OutPut:

	count	mean	std	min	25%	50%	75%	max
Lenght	4177.0	0.523992	0.120093	0.0750	0.4500	0.5450	0.615	0.8150
Diameter	4177.0	0.407881	0.099240	0.0550	0.3500	0.4250	0.480	0.6500
Height	4177.0	0.139516	0.041827	0.0000	0.1150	0.1400	0.165	1.1300
Whole Weight	4177.0	0.828742	0.490389	0.0020	0.4415	0.7995	1.153	2.8255
Shucked Weight	4177.0	0.359367	0.221963	0.0010	0.1860	0.3360	0.502	1.4880
Viscera Weight	4177.0	0.180594	0.109614	0.0005	0.0935	0.1710	0.253	0.7600
Shell Weight	4177.0	0.238831	0.139203	0.0015	0.1300	0.2340	0.329	1.0050
Rings	4177.0	9.933684	3.224169	1.0000	8.0000	9.0000	11.000	29.0000

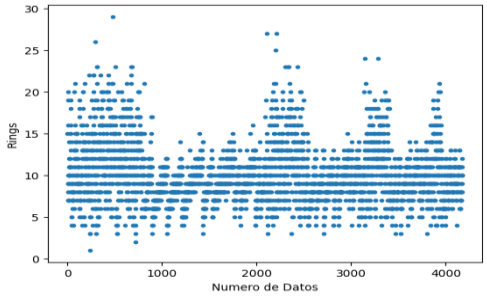
5.2. Visualizar Datos.

Es de interés el comportamiento de los anillos con respecto al número de datos que tenemos (teniendo 4,177 datos por atributo), se empleara una gráfica de puntos (scatter plot), el código es el siguiente:

```
InPut:
plt.plot(abalones["Rings"], marker=".", linestyle="")

plt.xlabel("Numero de Datos")
```

```
plt.ylabel("Rings")

plt.show()
```

OutPut: Gáfica de Puntos Datos de Anillos con Respecto al Número de Datos



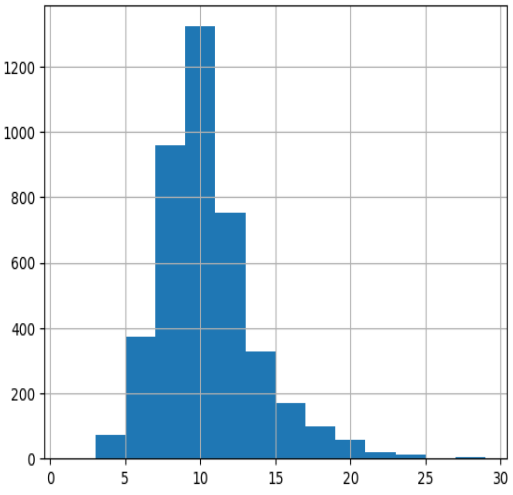
Podemos observar que los datos de número de anillos no tienen tendencia lineal, en la siguiente sección podremos visualizar este mismo comportamiento de los datos de los anillos con respecto con los otros datos con tendencias muy lejanas a la linealidad.

Otro método para visualizar los datos de los anillos con respecto al número de datos es mediante una gráfica de histograma. El código es el siguiente:

```
InPut:
abalones["Rings"].hist(bins=14)
plt.show
```

En el archivo PRACTICA_01_KNN_DATASET_ABALONES.ipynb se explica mejor por qué se usa bins= 14. En el histograma podemos observar en el eje “x” (el ancho de intervalos que ocupan un bin) y en el eje “y” (Cantidad de Valores/ Número de Ocurrencias). Podemos observar que hay abulones que tienen entre 5 a 15 anillos y que también tenemos la posibilidad de tener abulones con más de 20 anillos. Recordando el análisis estadístico previamente realizado tenemos que el máximo de anillos es de 29. También podemos observar que más de 1200 abulones tienen entre 9 a 11 anillos presentes en la concha. Los abulones con mayor edad están subpresentados en este conjunto de datos, las distribuciones de edades generalmente están sesgados de esta manera debido a los procesos naturales.

OutPut: Gráfica Histograma Datos Anillos con respecto al número de datos.



5.3. Matriz de Correlación.

La correlación es una medida estadística que indica el grado de relación entre dos variables.

Tenemos 5 casos de correlaciones:

- r= -1: las dos variables tienen una correlación perfecta negativa (por lo que se puede trazar una recta con pendiente negativa)
- -1<r<0: tenemos dos casos; r mayor a -0.80 los datos tienen tendencia lineal y r menor a -0.79 los datos se separan de la tendencia lineal.
- r=0: la correlación entre las dos variables es muy débil. Esto no significa que las variables sean independientes, pero no tienen tendencia lineal.
- 0<r<1: tenemos dos casos; r mayor a 0.80 los datos tienen tendencia lineal y r menor a 0.79 los datos se separan de la tendencia lineal.
- r= 1: las dos variables tienen una correlación perfecta positiva, es decir, tienen una relación lineal positiva. (Balderix, 2023)

En Python el cálculo de la matriz de correlación es fácil gracias a la librería PANDAS empleando el comando .corr(), el código es el siguiente:

```
InPut: abalones.corr()
```

OutPut:

	Lenght	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight	Rings
Lenght	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole Weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked Weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera Weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell Weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

Se describió previamente que los datos de los anillos con respecto a lo de más datos no tienen una buena tendencia lineal, la r (correlación) de los anillos va desde 0.42 hasta 0.62 para que se pueda considerar que existe una tendencia lineal adecuada r debe de tener valores superiores al 0.80.

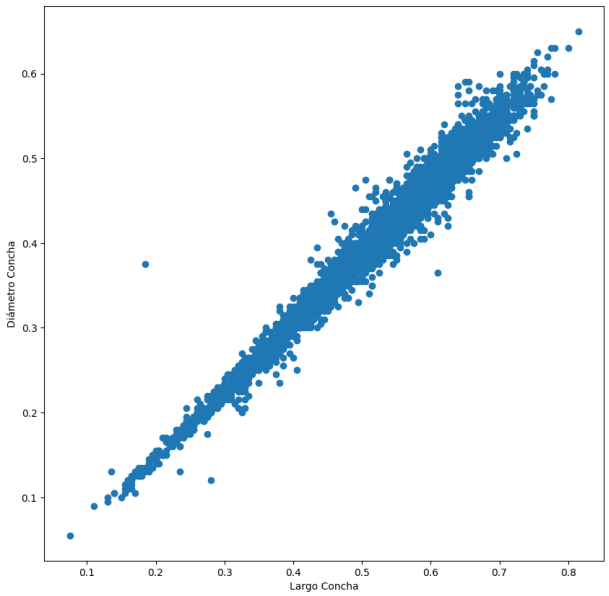
Entonces al no tener una tendencia lineal con los datos de anillos de los abalones, ¿con que datos se pueden trabajar?, se pueden descartar desde un principio los siguientes: Altura, Peso entero, Peso descascarado, Peso de vísceras y Peso de concha, debido a que los anillos se encuentran en la

concha del abalon. Por lo que los anillos son datos dependientes del Diámetro y del Largo de la concha del abalon. Así mismo la correlación $r_{diametro\text{-}largo}$= 0.98, lo que nos indica una tendencia completamente lineal siendo de utilidad para poder generar el algoritmo K-Nearest Neighbors (vecinos cercanos) si deseas leer más a fondo sobre este algoritmo puedes revisar en la bibliografía (Korstanje, 2021). De forma resumida el algoritmo de K-NN, es un modelo supervisado de Machine Learning que emplea el cálculo de distancias Euclidianas entre vectores (puntos), esto significa que predice una variable objetivo utilizando una o varias variables independientes tanto puede emplearse para modelos lineales como para modelos no lineales.

Podemos visualizar de manera rápida la tendencia lineal entre los datos de Largo y Diámetro mediante una gráfica de puntos (scatter plot), el código es el siguiente:

```
InPut:

plt.figure(figsize= (10,10))

x= abalones["Lenght"]
y= abalones["Diameter"]

plt.scatter(x=x, y=y)
plt.show()
```

OutPut: Gráfica de Puntos Diámetro Concha con respecto a la Longitud de la Concha.



Al conocer estos criterios podremos comenzar con el Algoritmo K- Nearest Neighbors.

6. Resultados Algoritmo K-Nearest Neighbors

Primero definimos las variables independientes y dependientes.

```
# Variables Independientes
X= abalones.drop("Rings", axis=1)
x= X.values  # Almacenamos los datos en varios diccionarios
```

```
# Variables Dependientes
Y= abalones["Rings"]
y=Y.values # Almacenamos los datos en varios diccionarios
```

```
# Se definirán el número de vecinos que se van a utilizar
K= 3
```

Datos nuevos a los que se van a predecir su edad:

```
# Datos de un nuevo especimen de Abalon: Length= 0.569552, Diameter= 0.446407, Heigth= 0.154437, Whole Weight= 1.016849, Shucked Weight= 0.439051, Viscera Weight= 0.222526 &  Shell Weight= 0.291208
nuevos_datos={"Length": 0.569552, "Diameter": 0.446407, "Heigth": 0.154437, "Whole_Weight": 1.016849, "Shucked_Weight": 0.439051, "Viscera_Weight": 0.222526 , "Shell_Weight": 0.291208}

datos= [nuevos_datos["Length"], nuevos_datos["Diameter"], nuevos_datos["Heigth"], nuevos_datos["Whole_Weight"], nuevos_datos["Shucked_Weight"], nuevos_datos["Viscera_Weight"], nuevos_datos["Shell_Weight"]]

Input: nuevo_punto_especimen= np.array(datos)
```

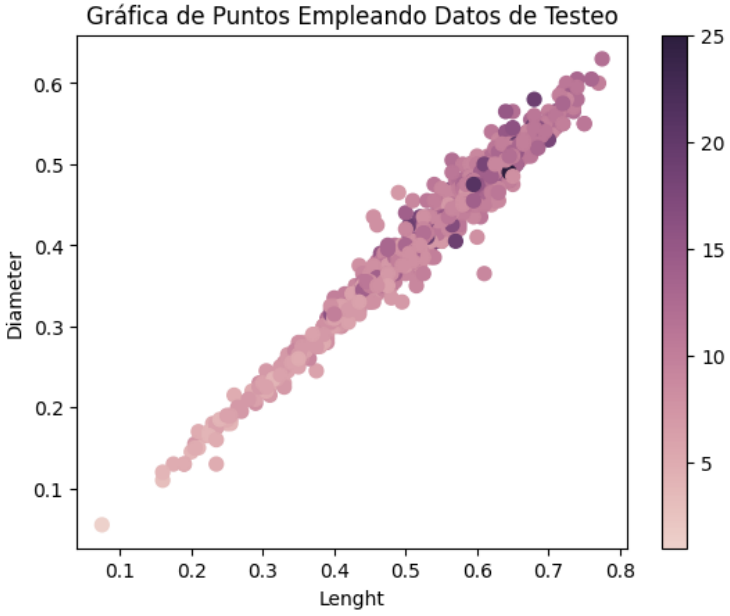
```
OutPut: nuevo_punto_especimen= array([0.569552, 0.446407, 0.154437, 1.016849, 0.439051, 0.222526,0.291208])
```

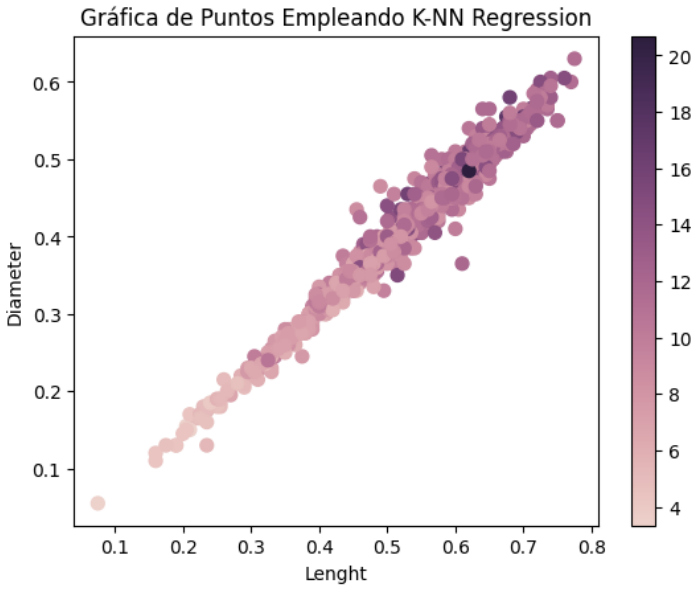
Elaborado por: Pasante de Ingeniería Bioquímica Vargas Carreño Leonardo Augusto

Tabla 6.1 Resultados Algoritmo K-Nearest Neighbors

Resultado / Método	Predicción Edad	Precisión	Error Medio Cuadrado	Error Cuadrático Medio
Distancia Euclidiana	10.0	N/A	N/A	N/A
K-Nearest Neighbors Regression Test	11.0	0.48281193591174143	5.642610313662946	2.375417924000521
K-Nearest Neighbors GridSearchCV Test	14.333333333333334	0.5710044879268722	4.680414474000431	2.1700197339962175
K-Nearest Neighbors GridSeachCV afinado	14.333333333333334	0.5710044879268722	4.680414474000431	2.1634265584947485
K-Nearest_Neighbors BaggingRegressor	14.333333333333334	0.5710596163547625	4.680414474000431	2.1634265584947485

Gráficas de puntos Diámetro de Concha con respecto a Largo de Concha con Barra de Predicción de Edades.



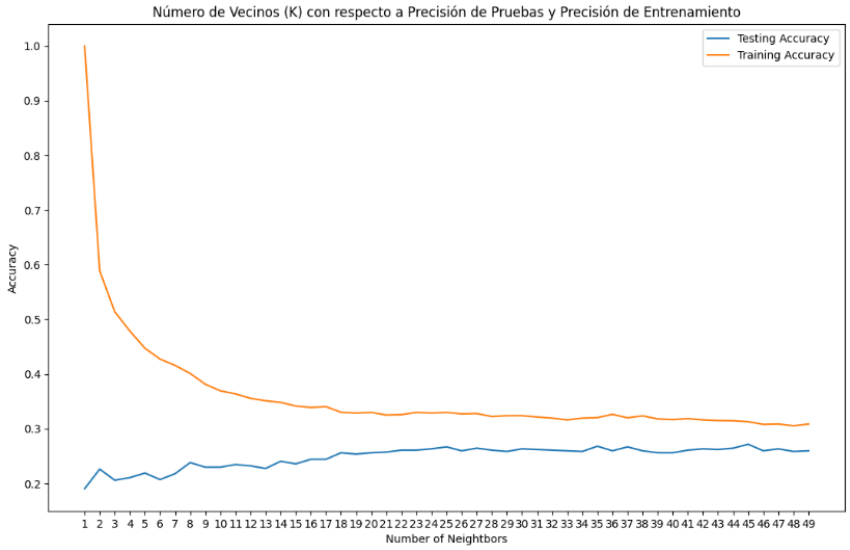


Elaborado por: Pasante de Ingeniería Bioquímica Vargas Carreño Leonardo Augusto

Podemos mejorar la precisión del Modelo calculando la K más adecuada, el código es el siguiente:

```
#En los modelos previamente realizados se empleó n_neighbors(K) con rangos de 1 a 50

neig= np.arange(1,50)
train_accurary=[]
test_accuracy=[]

#Bucle para diferentes valores de K

for i, K in enumerate(neig):

    #K from 1 a 50
    knn= KNeighborsClassifier (n_neighbors=K)

    #Ajustar con Knn (método fit)
    knn.fit(X_train, Y_train)

    #Train Accuracy
    train_accurary.append(knn.score(X_train,Y_train))

    #test_accuracy
    test_accuracy.append(knn.score(X_test,Y_test))
```

En el Archivo PRACTICA_01_KNN_DATASET_ABALONES.ipynb viene el código de como se realizo la gráfica.




```
InPut:
print ("Best Accuracy is {} with
K={}".format(np.max(test_accuracy),1+test_accuracy.index(np.max(test_accuracy))))
```

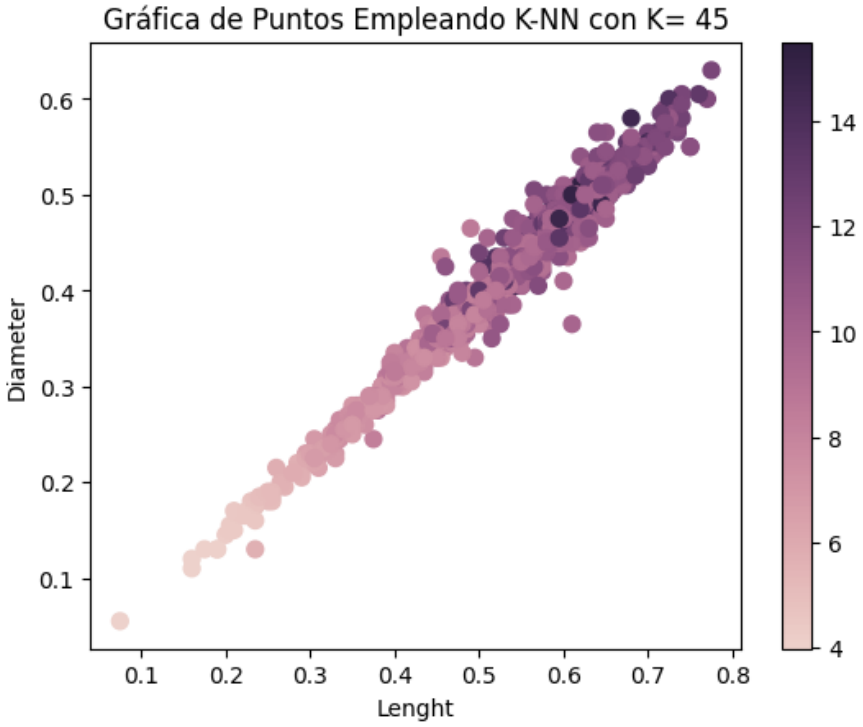
OutPut: **Best Accuracy is 0.2715311004784689 with K=45**

Podemos observar que el Algortimo K-Nearest Neighbors se puede optimizar aún más no con solo emplear el método Bagging, también podemos emplear K= 45. En la siguiente Tabla 6.2. Resultados Algoritmo K-Nearest-Neighbors, se mostrarán los resultados en comparación con los resultados expuestos en la Tabla 6.1. Resultados Algoritmo K-Nearest Neighbors.

Tabla 6.2 Resultados Algoritmo K-Nearest Neighbors

Resultado / Método	Predicción Edad	Precisión	Error Medio Cuadrado	Error Cuadrático Medio
Distancia Euclidiana	10.0			
K-Nearest Neighbors Regression Test	11.0	0.48281193591174143	5.642610313662946	2.375417924000521
K-Nearest Neighbors GridSearchCV Test	14.333333333333334	0.5710044879268722	4.680414474000431	2.1700197339962175
K-Nearest Neighbors GridSeachCV afinado	14.333333333333334	0.5710044879268722	4.680414474000431	2.1634265584947485
K-Nearest_Neighbors BaggingRegressor	14.333333333333334	0.5710596163547625	4.680414474000431	2.1634265584947485
Resultados Algoritmo K-Nearest Neighbors empleando K= 45				
Distancia Euclidiana	13.333333333333334			
K-Nearest Neighbors GridSearchCV Test	13.333333333333334	0.5571474922954613	4.831596668438774	2.1980893222157225
K-Nearest Neighbors GridSeachCV afinado	13.333333333333334	0.5625767373997524	4.772362674949609	2.18457379709398
K-Nearest_Neighbors BaggingRegressor	13.333333333333334	0.5618390142241652	4.772362674949609	2.18457379709398

Como podemos observar en la Tabla 6.2. Resultados Algoritmo K-Nearest-Neighbors si existe una mejoría en la predicción de la edad de los datos de abalon estudiado, dando como resultado una edad de 13 años. En todos algotimos con K=45 da el mismo resultado debido a que se emplea el mismo valor para K y en las precisión del método como tal no hay una diferencia de variación que pueda indicar lo contrario.



El código completo de cómo se llegaron a esos resultados lo podrán consultar en el archivo PARTEII_PRACTICA_01.ipynb.

En las gráficas presentadas nosotros podemos interpolar con los datos y realizar un aproximado de las predicciones de las edades a partir de los datos de Largo y Diámetro de Concha.

7. Conclusiones.

El algoritmo KNN se puede optimizar para mejorar las predicciones de edad, así mismo ir conociendo cómo se comportan los datos al ir ejecutando el algoritmo es importante para poder seleccionar el valor de número de vecinos óptimo para el DataSet. Se aprendió:

- Comprender los fundamentos matemáticos detrás del algoritmo kNN.
- Codificar el algoritmo kNN desde cero en NumPy.
- Utilice la implementación scikit-learn para ajustar un kNN con una cantidad mínima de código.
- Utilice GridSearchCV para encontrar los mejores hiperparámetros kNN.
- Llevar kNN a su máximo rendimiento mediante ensacado.

Recomiendo estudiar los códigos del como funcionan y cómo van realizando los calculo para llegar a las predicciones funciona mucho. Esto ayuda mucho para después hacer códigos propios de forma sencilla e ir poco a poco implementando o aprendiendo mas códigos para ir automatizando el algoritmo.

Bibliografía

Balderix, A. (2023). *Probabilidad y Estadistica de Academia Balderix*. Obtenido de Correlación: https://www.probabilidadyestadistica.net/correlacion/

Korstanje, J. (07 de Abril de 2021). *Real Python*. Obtenido de The K-Nearest Neightbors (KNN) Algoritm in Python: https://realpython.com/knn-python/

Wikipedia. (27 de Julio de 2023). *Wikipedia La enciclopedia libre*. Obtenido de Haliotis: https://es.wikipedia.org/wiki/Haliotis