PRACTICE 01: “K-NEAREST NEIGHBORS METHOD USING ABALON SNAIL DATASET TO PREDICT SOTHITY USING THE NUMBER OF RINGS” ENGLISH VERSION

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

1. Brief description of Abalone.

1.1. Haliotis

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

The haliotidae (Haliotidae) are a family of gastropod mollusks with a single genus, Haliotis. They are known as abalone, or abalone. The word abalone derives from the Ohlone word awlun or aluan.

1.2. General Morphology.

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

They have a long, flat, oval-shaped shell, in a spiral with two or three turns, the outermost of which gives them their characteristic auricular shape. It has four to ten holes through which they expel the water contained inside the shell in case of danger, so as to cling more firmly to its substrate. The exterior of the shell is rough and variable in color; The interior is made of iridescent mother-of-pearl and very attractive. The body of haliothodes follows the model of mollusks, it has three regions: the foot, the visceral mass and the mantle.. (Wikipedia, 2023)

2. Objectives of the Practice.

2.1. Create an algorithm using Python that can predict the age of Abalones using data that has been collected from their morphological characteristics.

2.2. Create a K-NN code using Python from scratch using the numpy library.

2.3. Implement Scikit-Learn to tune KNN with a minimal amount of code.

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

2.5. Reach the maximum performance of KNN using the data 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)

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

3. Librerías y Descargar DataSet Abulones.

The libraries and in general all the code can be consulted in the file PRACTICA_01_KNN_DATASET_ABALONES.ipynb attached to this folder, but for practical and explanatory purposes some steps will be added.

3.1. Libraries.

As I have been learning to program, it has become easier for me to make a file in Python with the libraries, call the file LIBRERIAS.py in all the algorithms that I have done so that I have the libraries stored and organized. As can be seen in Image 3.1. Libraries for the K-NN Method. To put the libraries into operation where they are going to be used, whether in a Python file (.py) or in a Jupiter Notebook file (.ipynb), the following code is used:

```
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-NN

3.2. Download Abalone DataSet.

This part is more about using Web Scrapping, the URL to use is the following:

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

The code is the following:

```
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)
```

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

4. DataSet Study.

In the time that I have been learning the fundamentals of Machine Learning for Data Science in all the algorithms that I have studied, a prior study of the Dataset of how the information is structured is carried out. Here methods (commands) are used, either the PANDAS or NumPy library.

4.1. Shape.

The .shape command works to know the number of columns and the amount of data contained in the DataFrame, the code would be as follows:

```
InPut:  abalones.shape
```

OutPut: (4177, 9), outputs a tuple with the DataFrame information, with: 4177 Data and 9 Columns.

4.2. Info().

The .info() command shows the information that makes up the DataFrame, the code is as follows:

```
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
```

We review the Information in the DataFrame:

There are 9 Columns, as we can see they have no attribute, this will be fixed later

There are DataFrames in which we can have the legend Null (empty), in this case there is none.

We can confirm that it contains 4177 data of which we have the type: Object, Decimals and Integers.

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

4.3. Add attributes (names) to columns:

The attributes are added according to the order in which the data is: Sex, Length, Diameter, Height, Whole weight, Shelled weight, Viscera weight, Shell weight and Rings. The code is the following:

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

Before:

```
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

After:

```
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. Remove Data.

Previously review and study the DataFrame in order to filter the information that we are not going to use, in this case we will remove data that is not quantifiable. The Sex column for the K-NN method is not useful for this specific case. The code is the following:

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

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

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. Statiscal Description of the Abalone DataFrame.

The PANDAS library has several commands with which we can operate with our DataFrame.

5.1. Describe.

The .describe() command allows us to make a statistical description, knowing the number of data, the mean, standard deviation, minimums and maximums by default. It also allows us to know the percentiles (0.25, 0.50, 0.75) of the DataFrame. The code is the following:

```
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

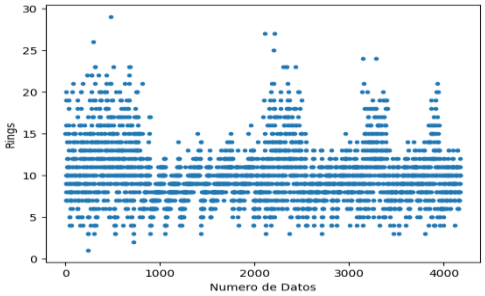
5.2. View Data.

Of interest is the behavior of the rings with respect to the number of data we have (having 4,177 data per attribute), a scatter plot will be used, the code is as follows:

```
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

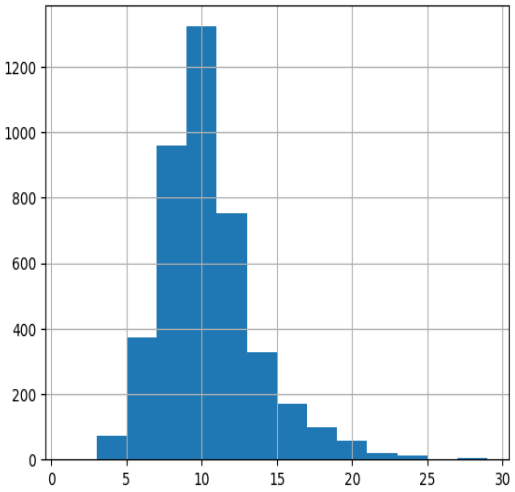


We can observe that the data on the number of rings does not have a linear trend. In the next section we can visualize this same behavior of the ring data with respect to the other data with trends very far from linearity. Another method of visualizing ring data with respect to the number of data is using a histogram plot. The code is the following:

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

The file PRACTICA_01_KNN_DATASET_ABALONES.ipynb explains better why bins= 14 is used. In the histogram we can see on the “x” axis (the width of intervals that occupy a bin) and on the “y” axis (Number of Values/ Number of Occurrences). We can see that there are abalone that have between 5 to 15 rings and that we also have the possibility of having abalone with more than 20 rings. Remembering the statistical analysis previously carried out, we have that the maximum number of rings is 29. We can also observe that more than 1200 abalone have between 9 to 11 rings present in the shell. Older abalones are underrepresented in this data set; age distributions are generally skewed in this way due to natural processes.

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



5.3. Correlation Matrix.

Correlation is a statistical measure that indicates the degree of relationship between two variables. We have 5 cases of correlations:

- r= -1: the two variables have a perfect negative correlation (so a line with a negative slope can be drawn)
- -1<r<0: we have two cases; r greater than -0.80 the data have a linear trend and r less than -0.79 the data separate from the linear trend.
- r=0: the correlation between the two variables is very weak. This does not mean that the variables are independent, but they do not have a linear trend.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: the two variables have a perfect positive correlation, that is, they have a positive linear relationship.(Balderix, 2023)

In Python, calculating the correlation matrix is easy thanks to the PANDAS library using the .corr() command, the code is as follows:

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

OutPut:

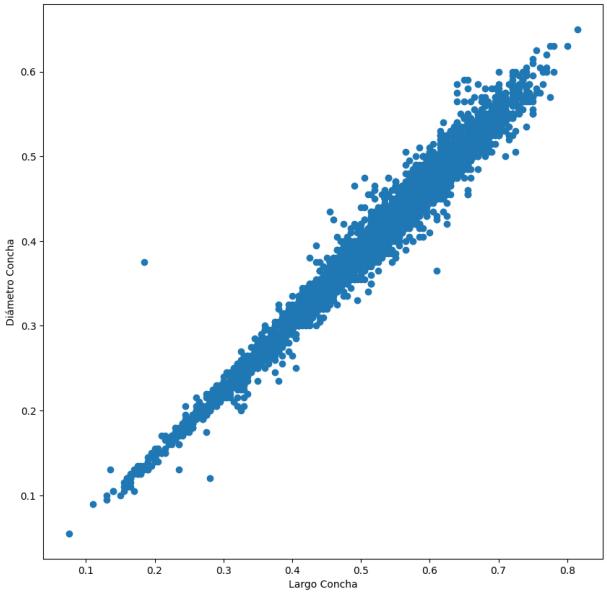
	Lenght	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight	Rings
Lenght	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	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

It was previously described that the data of the rings with respect to the most data do not have a good linear trend, the r (correlation) of the rings goes from 0.42 to 0.62 so that it can be considered that there is an adequate linear trend r must have values greater than 0.80. So not having a linear trend with the data of rings of the abalones, with what data can be worked?, the following can be discarded from the beginning: Height, Whole weight, Shelled weight, Weight of viscera and Weight of shell, because the rings are in the shell of the abalon. So the rings are data dependent on the Diameter and Length of the shell of the abalon. Likewise, the rdiametro-largo correlation = 0.98, which indicates a completely linear

trend being useful to generate the K-Nearest Neighbors algorithm if you want to read more about this algorithm you can review in the bibliography (Korstanje, 2021). In summary, the K-NN algorithm is a supervised Machine Learning model that uses the calculation of Euclidean distances between vectors (points), this means that it predicts a target variable using one or more independent variables both for linear models and for nonlinear models. We can quickly visualize the linear trend between the Length and Diameter data using a point plot (scatter plot), the code is as follows:

```
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.



By knowing these criteria, We can start with the K- Nearest Neighbors Algorithm.

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

6. K-Nearest Neighbors Algorithm Results

First we define the independent and dependent variables.

```
# 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
```

New data that will be predicted for their age:

```
# 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)
```

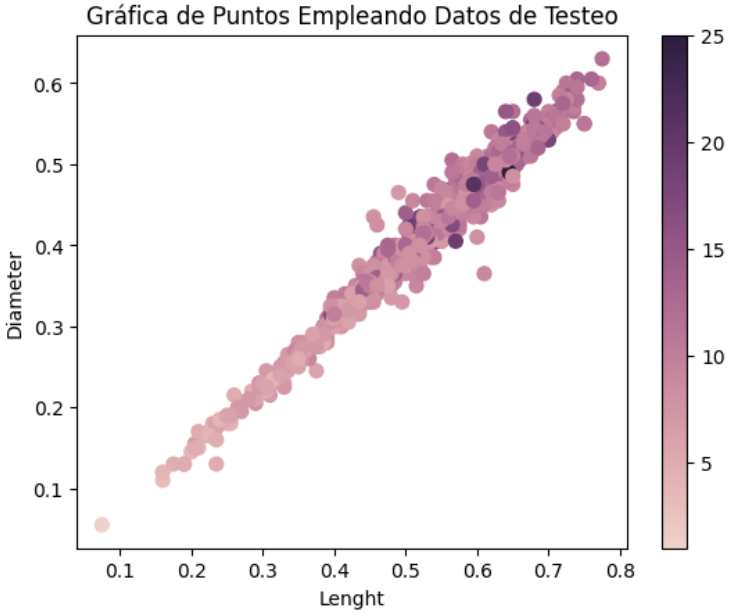
```
OutPut: nuevo_punto_especimen= array([0.569552, 0.446407, 0.154437, 1.016849, 0.439051, 0.222526,0.291208])
```

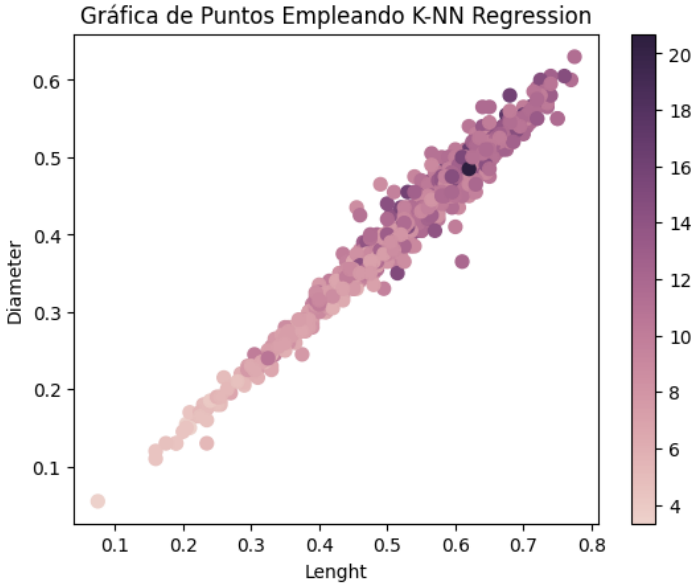
Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

Table 6.1 Results K-Nearest Neighbors Algorithm

Result / Method	Age Prediction	Precisión	Half-Square Error	Mean square error
Euclidean Distance	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

Point Plots Shell Diameter with respect to Shell Length with Age Prediction Bar.



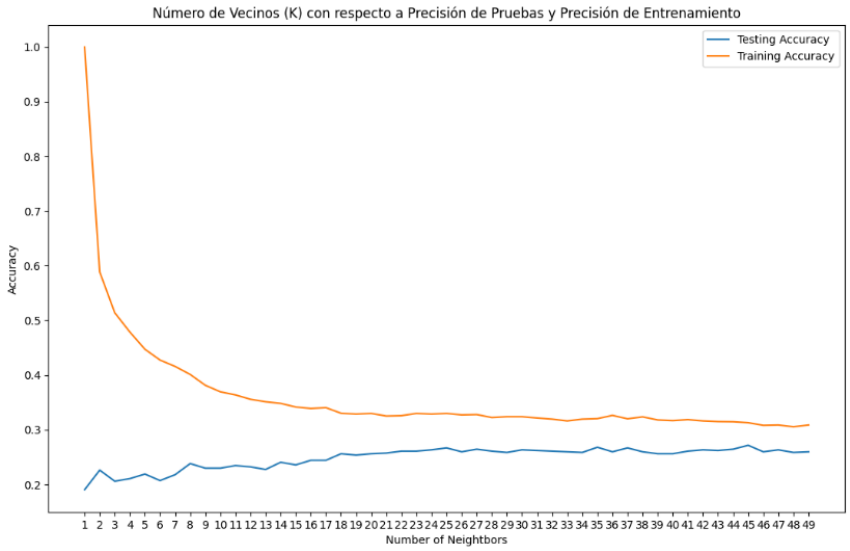


Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

We can improve the accuracy of the Model by calculating the most suitable K, the code is as follows:

```
#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))
```

In the file PRACTICA_01_KNN_DATASET_ABALONES.ipynb comes the code of how the graph was made.




```
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**

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

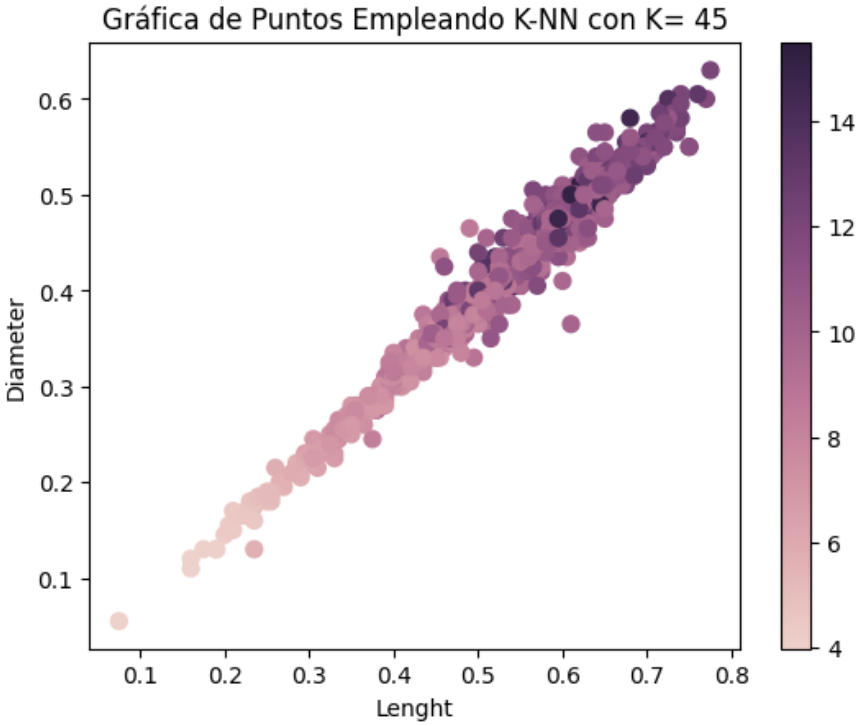
We can see that the Algortimo K-Nearest Neighbors can be further optimized not only by using the Bagging method, we can also use K = 45. In the following Table 6.2. Results K-Nearest-Neighbors algorithm, the results will be shown in comparison with the results presented in Table 6.1. Results K-Nearest Neighbors algorithm.

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

Prepared by: Biochemical Engineering Intern Vargas Carreño Leonardo Augusto

As we can see in Table 6.2. Results K-Nearest-Neighbors algorithm if there is an improvement in the prediction of the age of the abalon data studied, resulting in an age of 13 years. In all algorithms with K = 45 gives the same result because the same value is used for K and in the precision of the method as such there is no difference in variation that can indicate otherwise.



The complete code of how these results were arrived at can be found in the file PARTEII_PRACTICA_01.ipynb.

In the graphs presented we can interpolate with the data and make an approximate of the predictions of the ages from the data of Length and Diameter of Shell.

7. Conclusions.

The KNN algorithm can be optimized to improve age predictions, likewise knowing how the data behaves when executing the algorithm is important to be able to select the optimal number of neighbors value for the DataSet. It was learned:

1. Understand the mathematical underpinnings behind the kNN algorithm.
2. Code the kNN algorithm from scratch in NumPy.
3. Use the scikit-learn implementation to tune a kNN with a minimal amount of code.
4. Use GridSearchCV to find the best kNN hyperparameters.
5. Bring kNN to its maximum performance by bagging.

I recommend studying the codes of how they work and how they are performing the calculations to reach the predictions works a lot. This helps a lot to then make your own codes in a simple way and gradually implement or learn more codes to automate the algorithm.

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