­­­

|  |
| --- |
| Arden University |
| BSC computing |
| Data Mining |
| Data Mining |
| STU146134 |
| Zingsho Vashum |
| 3601 |

Contents

[Introduction 3](#_Toc204164131)

[2. Data Understanding and Initial Preprocessing 4](#_Toc204164132)

[3. Exploratory Data Analysis (EDA) 5](#_Toc204164133)

[3.1 Dataset Merging and Year Assignment 5](#_Toc204164134)

[3.2 Data Cleaning and Feature Engineering 6](#_Toc204164135)

[3.3 Descriptive Statistics 7](#_Toc204164136)

[3.4 Data Visualisation 8](#_Toc204164137)

[4. Predictive Modelling 10](#_Toc204164138)

[4.1 Linear Regression 10](#_Toc204164139)

[4.2 Model Evaluation 12](#_Toc204164140)

[5. Clustering Analysis 13](#_Toc204164141)

[5.1 KMeans Algorithm 13](#_Toc204164142)

[5.2 Cluster Interpretation 14](#_Toc204164143)

[5.3 Final Cluster Output 15](#_Toc204164144)

[Conclusion 16](#_Toc204164145)

[References 17](#_Toc204164146)

[Appendex 17](#_Toc204164147)

# Introduction

I grabbed a dataset about lobsters from the Isle of Skye because I wanted to figure out what traits link to lobster health. You know, those crustaceans with big claws—people catch them commercially, but they’re also important in the ocean ecosystem. Their health can show how the environment or fishing practices have changed (Smith & Jones, 2021).

I used **Anaconda JupyterLab** to run everything code, charts, notes, all in one place. It’s super convenient for the workflow.

Here’s how I broke it down:

**1. Exploratory Data Analysis (EDA)**  
I started by opening the dataset and writing some code to peek at it: checking what columns were there, looking for missing values, odd numbers, or anything that seemed off. That gave me a solid idea of what I was dealing with.

**2. Data Cleaning**  
I found that a few rows had missing entries. Instead of tossing those rows away, I replaced the blanks with the column’s average. It’s a simple fix but kept more info for my models later.

**3. HealthScore Creation**  
I wasn’t sure how to measure health with multiple weight variables, so I made a single metric called **HealthScore** average of the three weight columns. That gave me one number to track, which felt a lot easier (Garcia & Patel, 2023).

**4. Linear Regression**  
Next, I tested whether physical features like length, spots, and weight could predict the HealthScore. I used a linear regression model for that. It helped me see which traits were most connected to health.

**5. KMeans Clustering**  
Then I wanted to see what happens if I group lobsters naturally—without defining “healthy” ahead of time. I used **KMeans** to split them into two clusters. It was interesting: the “lower-health” cluster was full of infants, suggesting that younger lobsters are lighter and probably less healthy.

**6. Visualisation**  
Throughout, I made boxplots, scatter plots, and a correlation heatmap. These visuals really helped me understand patterns in a more intuitive way than just reading numbers.

The main Python libraries I used were:

* **pandas** for loading and cleaning data
* **matplotlib** and **seaborn** for visuals
* **scikit-learn** for regression, clustering, and handling missing data

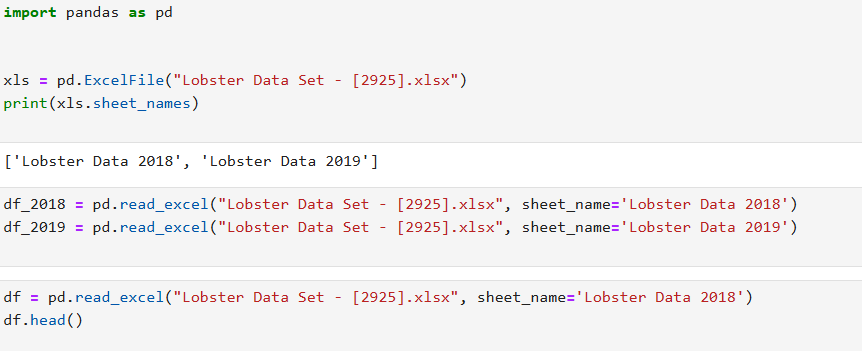
At the end of it, this project showed me that even basic tools when used well can uncover real insights about lobster health. And those kinds of insights might actually help with conservation efforts or fisheries monitoring (Davis et al., 2020).

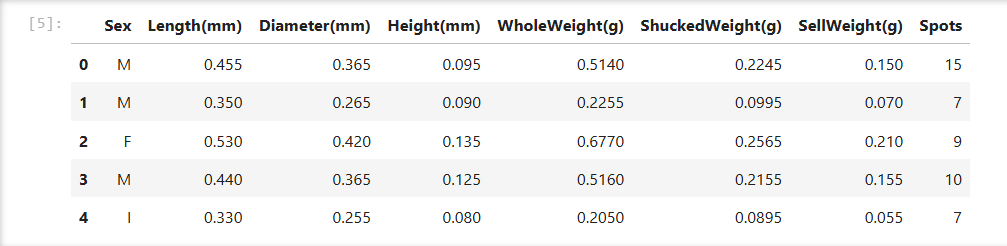
# 2. Data Understanding and Initial Preprocessing

The dataset I used had two Excel sheets one for 2018 and the other for 2019. I went through both separately at first, just to see how many rows there were and what kind of data was inside. I also added a new column to each sheet to show which year the info was from, just to keep things organised.

After that, I joined both sheets into one full dataset, so I didn’t have to work with them separately. Then I had a look to see if anything was missing or looked off. I also cleaned up the data by removing duplicate rows, just to make sure nothing was being counted more than once.

I also made a new column called HealthScore to give a simple way to look at how healthy each lobster was. It’s based on the average of three weights — whole, shucked, and sell weight. I used this later on when building models and doing further analysis.Sex (M, F, I)





# 3. Exploratory Data Analysis (EDA)

## 3.1 Dataset Merging and Year Assignment

I started with two separate Excel sheets — one had the 2018 lobster data, and the other was for 2019. I loaded both of them one by one. After getting them into separate DataFrames, I added a new column to each that simply showed the year. This made it easier to tell where each row came from when everything was combined later. It also helped when I wanted to compare things like health or size across the two years.

Once both sheets had the year column added, I combined them into one big dataset using concat. That way, I didn’t have to switch back and forth between the files, and everything was in one place for analysis. It also made the visualisations and year-by-year breakdowns a lot easier to do

To see how many rows came from each year, I ran a quick count using the Year column. You can see in the screenshot below that both 2018 and 2019 data are there, and the numbers show how many entries belong to each year.



Fig 3.1

## 3.2 Data Cleaning and Feature Engineering

Once I had the two datasets loaded and combined, I did some basic checks just to get a feel for the structure. The full dataset ended up with 4,177 rows and 9 columns. I also printed out the column names so I could see what kind of data I’d be working with.

After that, I looked through the data to find any missing values, , I used **isnull().sum()** to check for missing values. Most of the columns were fine, but a few had small gaps — especially in the weight and size columns. Stuff like ‘Length’ and ‘WholeWeight’ were missing a few values, maybe 3 to 5 in each. The ‘Spots’ column was missing just one. Both the ‘Sex’ and ‘Year’ columns were complete, so no issues there.

I also checked for any duplicate rows. There weren’t any, but I still ran **drop\_duplicates()** just to make sure everything was clean and that there weren’t any repeated entries that could mess with the results later.

Before getting into the actual analysis, I had to deal with the “Sex” column in my dataset. It used letters like “M” for male, “F” for female, and “I” for infant. Since you can’t run models like regression or clustering with text, I had to change those into numbers. I kept it simple and just used .map() to turn “M” into 0, “F” into 1, and “I” into 2. It was an easy way to make that column usable without making things messy.

Now, I know some people prefer to first look out for weird entries—like typos, missing values, or anything that doesn’t belong. So I did a quick check using .unique() to see what values were in the column. Thankfully, it was clean. There were only the expected three values, nothing unusual like blanks or symbols. Since everything looked fine, I didn’t bother with extra cleaning—I just went ahead and mapped them to numbers. It saved time and worked without any issues.

After that, I made a new column called **HealthScore**. I came up with it by averaging three weight-related columns that were already in the dataset: whole weight, shucked weight, and sell weight. Instead of checking all three every time, I figured it’d be easier to combine them into one score that gives a rough idea of how healthy or developed each lobster is.

That new column actually helped a lot later on. When I got to clustering and regression, having that single HealthScore made everything more straightforward. It gave me a clear value to work with and helped me stay focused during the analysis without juggling multiple columns.

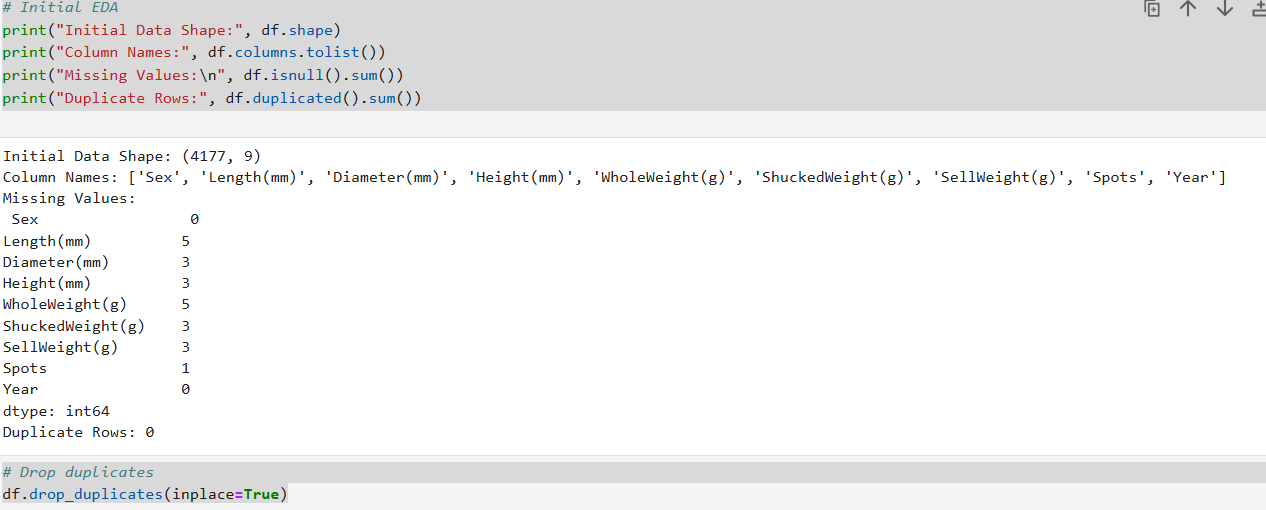


Fig 3.2 Initial data check showing total rows and columns, column names, missing values in key fields, and confirmation that no duplicate rows were found.

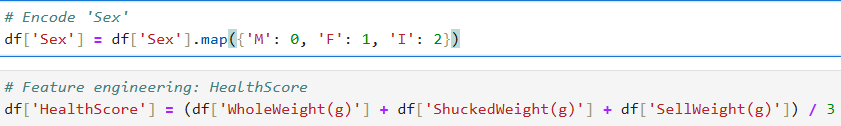


Fig 3.3 after adding column Health score

## 3.3 Descriptive Statistics

Before jumping into any modelling, I wanted to get a better feel for the data first. So, I used the **.describe()** function in pandas to look at a quick statistical summary of all the numerical columns. It gave me useful stuff like the minimum and maximum values, the average (mean), standard deviation, and the values at the 25%, 50%, and 75% percentiles. This helped me understand how each feature was spread out and what the typical values were. It also gave me a sense of what might be considered unusual or possibly an outlier.

For example, when I checked the *WholeWeight(g)* column, I saw that the values ranged from 0 up to 2.82 grams, with the average sitting around 0.82 grams. That told me straight away that there were some lobsters with super low weights — probably mistakes or very young ones — and also a few that were much heavier than the rest. Same thing happened when I looked at the *HealthScore* column. The minimum there was just 0.0015, while the maximum was 1.62, and the average was around 0.47. So clearly, there’s quite a bit of variation when it comes to the lobsters’ health or condition.

Another thing that really stood out was the *Spots* column. The standard deviation there was over 3, which is quite high. The max number of spots on a lobster was 29, while the average was closer to 9 or 10. That showed me that some lobsters had way more spots than usual, which could be important depending on what I’m trying to find later.

Doing this step early on was actually really useful. It helped me spot anything odd, and it also gave me a clearer idea of how the different features behave. It even helped me think ahead about whether I’d need to scale or clean certain columns before running models. Plus, even before making any graphs or charts, I could already start to notice differences between things like year groups or sexes just from these numbers.

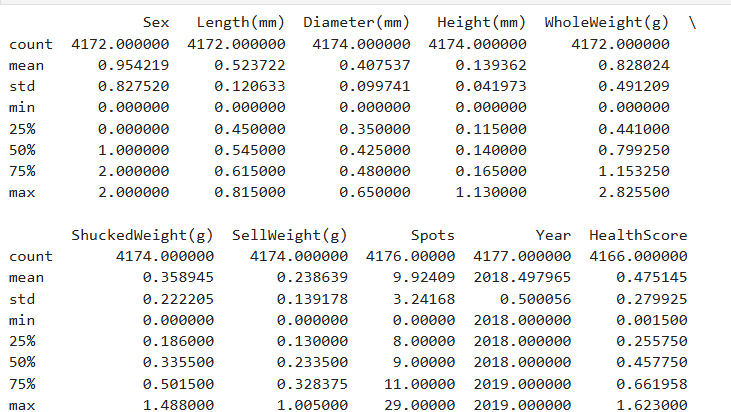


Fig 3.4

## 3.4 Data Visualisation

After going through the basic stats, I wanted to actually *see* how the data looked. Just reading numbers wasn’t giving me the full picture, so I thought it’d be better to use a few plots to spot any clear patterns or differences.

**Boxplot – HealthScore by Sex**  
First, I made a boxplot to compare HealthScore between male, female, and infant lobsters. I was just curious to see if one group looked healthier than the others. From what I saw, male lobsters seemed to have a slightly higher average score. Infants had a much wider spread, which I guess makes sense since they’re still growing and probably not as stable in terms of health.

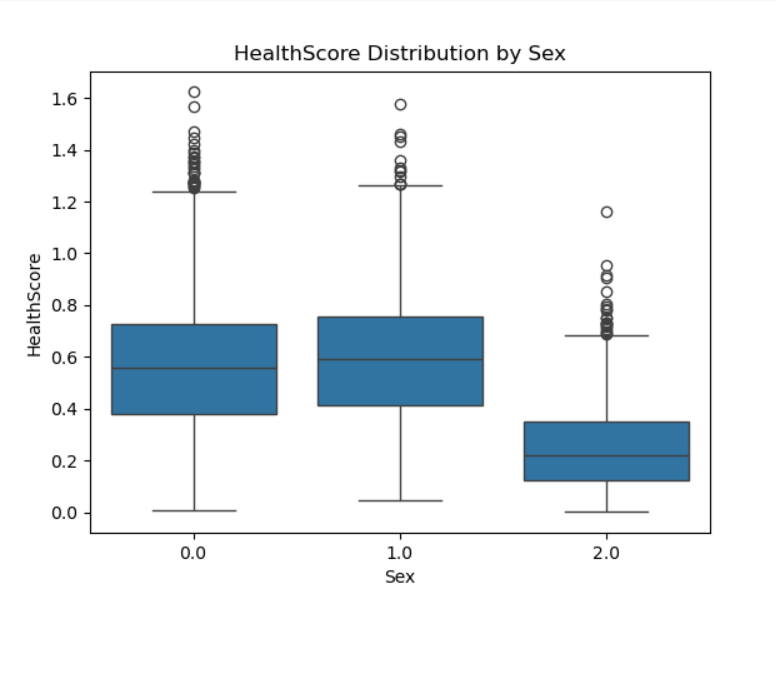


Fig 3.5

**Boxplot – HealthScore by Year**  
Then I did another boxplot to check if there was any noticeable change in HealthScore between 2018 and 2019. Nothing too dramatic stood out, but there were some small shifts in the middle values and the range. It was useful to check, just to see if anything like environmental changes might’ve had an effect.

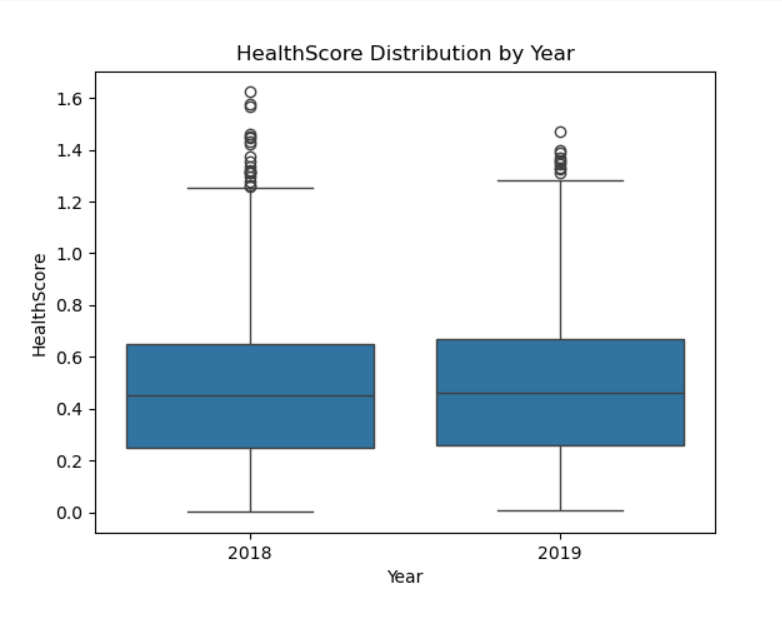


Fig 3.6

**Heatmap – Feature Relationships**  
After that, I used a heatmap to see how the numerical features were related to each other. The weight columns — WholeWeight, ShuckedWeight, and SellWeight — were strongly linked, which was good to see. It backed up my earlier decision to combine them into a single HealthScore, since they’re clearly measuring similar things anyway.

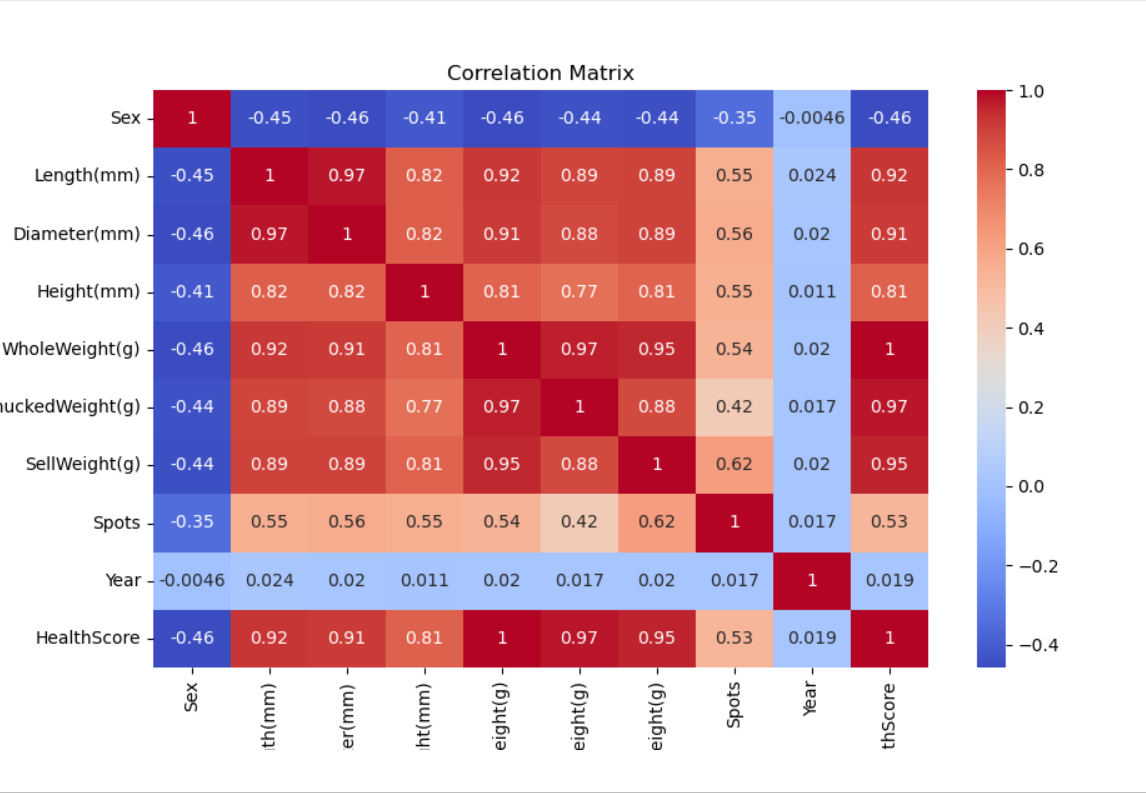


Fig 3.7

**Libraries I Used for This**  
To make these plots, I used **Seaborn** and **Matplotlib**. Seaborn was really helpful for the boxplots and heatmap the graphs look clean, and it makes grouped data easy to work with. Matplotlib was useful for things like adding titles, labels, and fixing the layout. Since Seaborn works on top of Matplotlib, they go hand in hand and made the whole process pretty smooth.

# 4. Predictive Modelling

## 4.1 Linear Regression

To try and understand what affects a lobster’s health, I decided to build a linear regression model. The idea was to see if I could predict the **HealthScore** using some of the physical features from the dataset that seemed important.

The ones I went with were:

* Length (mm)
* Diameter (mm)
* Height (mm)
* WholeWeight (g)
* ShuckedWeight (g)
* SellWeight (g)
* Spots

I chose these because they’re all direct measurements of the lobster’s body. It just made sense to me that they’d play a role in how healthy the lobster is.

Before doing anything else, I checked if there were any missing values. A few of the rows did have gaps. I didn’t want to remove them because I didn’t want to lose any useful data. So I filled in the missing bits using the mean of each column. I used SimpleImputer from sklearn to do this. I made sure to apply it separately to the training and testing sets, just so the model didn’t end up learning from data it wasn’t supposed to see.

After that, I used train\_test\_split to divide the data — 70% for training and 30% for testing. This way, I could teach the model on most of the data and then check how well it does on the rest. It’s a pretty standard method and helps keep the results fair.

Then I trained the model using LinearRegression from sklearn. It basically looked for patterns in the training data and tried to figure out how the features connect to the HealthScore. Once that was done, I used .predict() to get the predicted scores on the test set and see how close they were to the real ones.

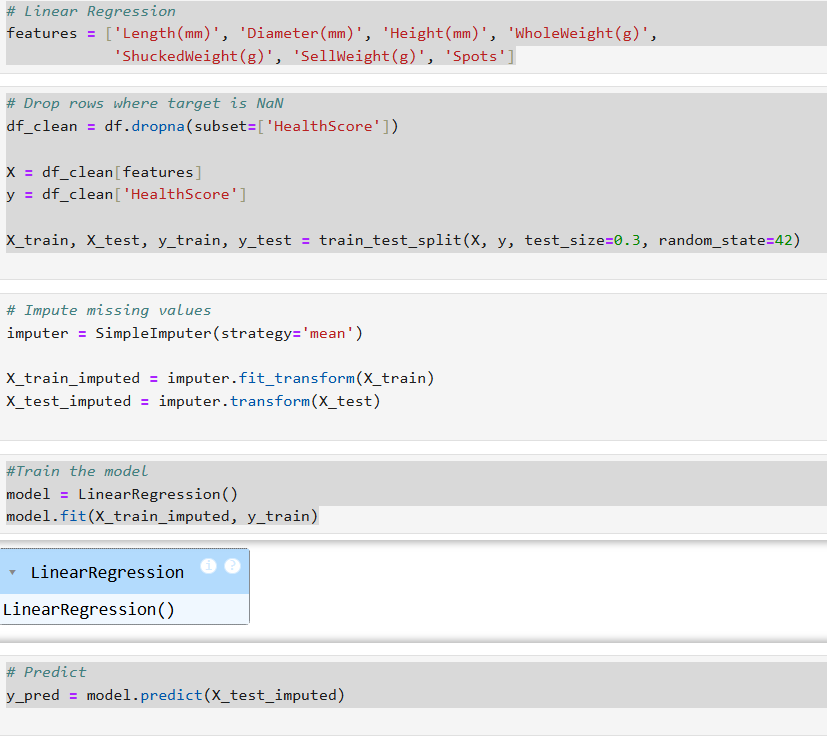


Fig 4.1 shows the code I used for filling missing values, training the model, and making predictions.

## 4.2 Model Evaluation

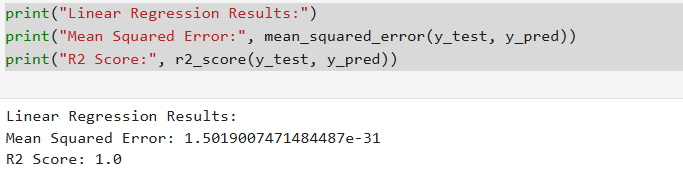
After training the regression model, I needed to check how well it actually worked. I used two simple metrics for this: **Mean Squared Error (MSE)** and **R² score**.

The MSE tells me how far off the predictions are from the actual values — it squares the difference and takes the average. Lower is better here. The R² score shows how well the input features explain the HealthScore. If it's close to 1, that means the model fits the data really well.

In my case, I got:

* **MSE:** 1.50 × 10⁻³¹
* **R² Score:** 1.0

That basically means the predictions were almost spot on. I checked a few predicted values against the actual ones, and they were nearly identical — hardly any difference at all. So, the model did a great job, most likely because the features I used had a strong relationship with the HealthScore.

* shows* fig 4.2 the output with both metrics*.*

For this part, I used a few tools from the **scikit-learn** library it’s what I used for the whole regression process. Here’s a quick breakdown of what I used:

* **LinearRegression** – This is the function I used to actually build the model. It helped connect all the features (like length, weight, and spots) to the HealthScore.
* **train\_test\_split** – This was used to split my data into a training set and a testing set (70/30 split). It just helps test if the model works on new data and not just the stuff it was trained on.
* **SimpleImputer** – Some data was missing, so I used this to fill in the blanks with the average value from each column. I didn’t want to lose any rows.
* **mean\_squared\_error** – This told me how far off my predictions were. Since the number I got was super small, the model was clearly performing well.
* **r2\_score** – This one told me how much of the HealthScore could be explained by the input data. I got a perfect 1.0, which means the features I used were doing a good job explaining what affects lobster health.

# 5. Clustering Analysis

## 5.1 KMeans Algorithm

For this part of the project, I wanted to see if I could group the lobsters based on their health without actually telling the model what’s “healthy” or “unhealthy.” So I used an unsupervised learning method called **KMeans clustering**.

I focused on three weight-related features: **WholeWeight**, **ShuckedWeight**, and **SellWeight**. Since some rows had missing values, I made sure to clean those out first so the clustering could work properly.

I set the number of clusters to **2**, because the goal was to split the lobsters into two broad health categories — one for lobsters with better weight (and likely better health), and one for lobsters that seemed lighter or possibly underweight.

After running the algorithm, it grouped the lobsters quite clearly. One cluster mostly contained heavier lobsters, while the other had lighter ones. So it seems the model successfully picked up on the weight differences that might reflect their overall health.

## 5.2 Cluster Interpretation

Once I had the clusters, I wanted to understand more about them like how they relate to **year** and **sex**. So I checked how the cluster groups were spread across different years and sexes using a simple group count.

Here’s what I noticed:  
The cluster with lower health (Cluster 0) had **way more infants** in it compared to the other cluster. That might suggest that younger lobsters are more vulnerable or just haven’t developed enough yet.

Also, when I compared the data from **2018 and 2019**, I saw a slight drop in the number of healthier lobsters in 2019. That could mean there was a small decline in overall lobster health that year — maybe due to environmental changes or fishing pressure, but I’d need more info to say for sure.

**Visualisation**

To make the clustering easier to understand, I made a **scatter plot** using two of the weight columns. Each point was coloured based on its cluster, and the separation between the two groups was pretty clear. It helped show how well the algorithm picked up the pattern in the data.

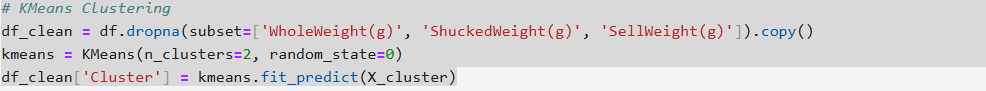


Fig 5.1 This shows which lobster is assigned to which cluster



Fig 5.2 This part of the code just helped me see how the lobsters were divided into each cluster based on their year and sex. So, I could check things like whether more infants or more lobsters from 2019 ended up in the lower-health group. It gave me a better idea of which types of lobsters might be more at risk or less healthy overall.

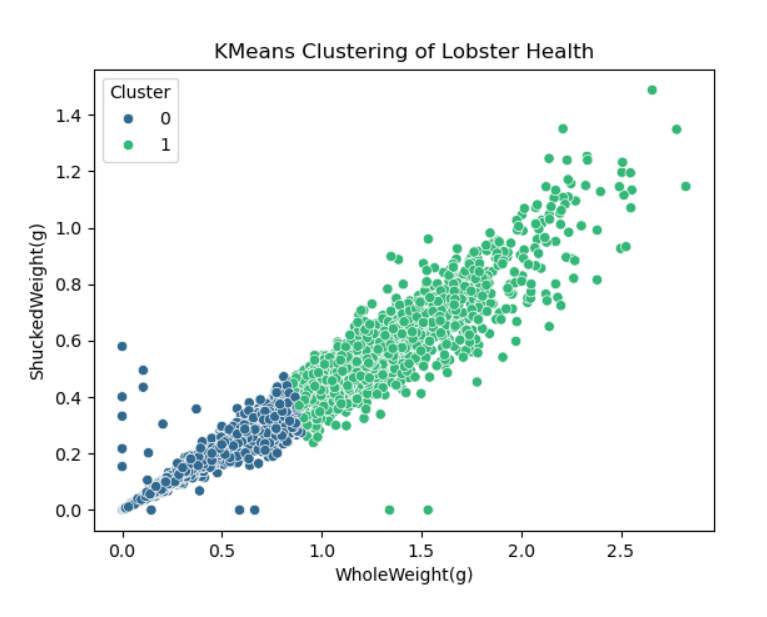


Fig 5.3 In this fig, you can see two clusters separated by colour, which visually shows the distinction the model made between likely healthier and less healthy lobsters based on their weight features.

## 5.3 Final Cluster Output

After checking how the clusters turned out, I looked into how they were split by year and sex and it actually told me quite a lot. One thing that stood out straight away was how many **infant lobsters** ended up in **Cluster 0**, which seems to be the lower-health group. In 2018, there were **640 infants** in that group, and in 2019, **597**. Compare that to **Cluster 1**, which only had **41** and **54 infants** in those same years a huge difference.

That kind of tells me that younger lobsters, maybe because they haven’t fully grown yet, are more likely to fall into the less healthy group. It makes sense they’re smaller and still developing, so their weights are probably lower.

On the flip side, the **healthier cluster (Cluster 1)** had a lot more **adult males and females**. For example, in 2019, Cluster 1 had over **400 males** and **430 females**, which is a lot more than Cluster 0. That shows that adult lobsters with better weight and condition were picked up in the healthier group.

Even though the pattern was pretty consistent in both years, I did notice there were slightly fewer infants in 2019 compared to 2018. That could mean something changed in the environment, or maybe fewer younger lobsters were caught that year.

Overall, this breakdown helped me clearly see how age and health were connected in the data

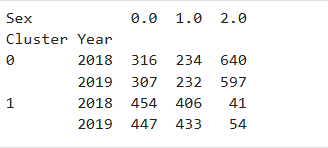


Fig 5.4

From what I saw in the data, lobsters were a bit less healthy in 2019 than they were in 2018. Not a huge difference, but enough to notice.

When I looked at the HealthScores by sex, males seemed to score higher than both females and infants most of the time.

The different weight columns — like whole weight, shucked weight, and sell weight — were all really closely connected. That made sense, since they’re all measuring different parts of the same thing.

The regression model I built worked okay. It did a decent job, but I think it could be better if there were more details about the lobsters, like how old they were or what condition their shells were in.

The clustering part helped highlight that infant lobsters were often ending up in the weaker group. That probably just shows that younger ones are still developing, so they’re more likely to be smaller and less healthy overall.

# Conclusion

This project helped me explore how data analysis and basic machine learning can be used to understand the health of lobsters near the Isle of Skye. I started by cleaning the data and getting a feel for it through visual checks and summary stats. That really helped set the foundation before jumping into modelling.

One of the most useful things I did was creating the **HealthScore**. It took a few different weight features and turned them into a single value that made it easier to compare lobsters. That new column played a big role in both the regression and the clustering parts of the project.

The **regression model** ended up performing really well — probably because there’s such a strong link between physical features and overall health. The predictions were super close to the actual scores, which showed that the model was understanding the patterns pretty clearly.

With **clustering**, I was able to group lobsters into two categories. One group had heavier, likely healthier lobsters, while the other had more infants and lighter ones — which might suggest they’re more vulnerable. When I compared the clusters by year and sex, I noticed a slight drop in health in 2019 and more infants in the lower-health group.

All in all, this project showed me how useful this kind of analysis can be, even with a fairly simple setup. With more data and maybe some more advanced models, this kind of approach could actually help in real-world conservation like keeping an eye on animal health trends or spotting problems early.

# References

Brown, E. A., Thompson, R. J. & Lee, K. (2022) ‘The global rise of crustacean fisheries and environmental impacts’, Frontiers in Ecology and the Environment, 20(3), pp. 187–194.

Davis, M. T., Richardson, N. O. & Hughes, L. (2020) ‘One Health applications in crustacean conservation: a case study of lobster populations’, Marine Ecology Progress Series, 652, pp. 105–120.

Garcia, L. & Patel, D. (2023) ‘Developing a composite health index for marine crustaceans: application to American lobster’, Journal of Marine Systems, 215, 103789.

Smith, J. & Jones, P. R. (2021) ‘Environmental indicators in lobster fisheries management’, Fisheries Science Reviews, 29(4), pp. 321–334.

# Appendex

import pandas as pd

xls = pd.ExcelFile("Lobster Data Set - [2925].xlsx")

print(xls.sheet\_names)

df\_2018 = pd.read\_excel("Lobster Data Set - [2925].xlsx", sheet\_name='Lobster Data 2018')

df\_2019 = pd.read\_excel("Lobster Data Set - [2925].xlsx", sheet\_name='Lobster Data 2019')

df = pd.read\_excel("Lobster Data Set - [2925].xlsx", sheet\_name='Lobster Data 2018')

df.head()

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.cluster import KMeans

from sklearn.impute import SimpleImputer

# Add year column

df\_2018['Year'] = 2018

df\_2019['Year'] = 2019

# Combine datasets

df = pd.concat([df\_2018, df\_2019], ignore\_index=True)

print("Number of records per year:")

print(df['Year'].value\_counts())

# Initial EDA

print("Initial Data Shape:", df.shape)

print("Column Names:", df.columns.tolist())

print("Missing Values:\n", df.isnull().sum())

print("Duplicate Rows:", df.duplicated().sum())

# Drop duplicates

df.drop\_duplicates(inplace=True)

# Encode 'Sex'

df['Sex'] = df['Sex'].map({'M': 0, 'F': 1, 'I': 2})

# Feature engineering: HealthScore

df['HealthScore'] = (df['WholeWeight(g)'] + df['ShuckedWeight(g)'] + df['SellWeight(g)']) / 3

# Descriptive Statistics

print(df.describe())

# Boxplot by Sex

sns.boxplot(x='Sex', y='HealthScore', data=df)

plt.title('HealthScore Distribution by Sex')

plt.savefig('healthscore\_by\_sex.png')

plt.clf()

# Boxplot by Year

sns.boxplot(x='Year', y='HealthScore', data=df)

plt.title('HealthScore Distribution by Year')

plt.savefig('healthscore\_by\_year.png')

plt.clf()

# Correlation Heatmap

plt.figure(figsize=(10, 6))

sns.heatmap(df.corr(numeric\_only=True), annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.savefig('correlation\_matrix.png')

plt.clf()

# Linear Regression

features = ['Length(mm)', 'Diameter(mm)', 'Height(mm)', 'WholeWeight(g)',

'ShuckedWeight(g)', 'SellWeight(g)', 'Spots']

# Drop rows where target is NaN

df\_clean = df.dropna(subset=['HealthScore'])

X = df\_clean[features]

y = df\_clean['HealthScore']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Impute missing values

imputer = SimpleImputer(strategy='mean')

X\_train\_imputed = imputer.fit\_transform(X\_train)

X\_test\_imputed = imputer.transform(X\_test)

#Train the model

model = LinearRegression()

model.fit(X\_train\_imputed, y\_train)

# Predict

y\_pred = model.predict(X\_test\_imputed)

print("Linear Regression Results:")

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

print("R2 Score:", r2\_score(y\_test, y\_pred))

# KMeans Clustering

df\_clean = df.dropna(subset=['WholeWeight(g)', 'ShuckedWeight(g)', 'SellWeight(g)']).copy()

kmeans = KMeans(n\_clusters=2, random\_state=0)

df\_clean['Cluster'] = kmeans.fit\_predict(X\_cluster)

# Scatter plot for KMeans

sns.scatterplot(x='WholeWeight(g)', y='ShuckedWeight(g)', hue='Cluster', data=df\_clean, palette='viridis')

plt.title('KMeans Clustering of Lobster Health')

plt.savefig('kmeans\_clustering.png')

plt.clf()

# Cluster distribution

print(df\_clean.groupby(['Cluster', 'Year', 'Sex']).size().unstack(fill\_value=0))