Machine Learning Project Report: Abalone Age Prediction using Random Forest Regressor (Individual)

Shuang Ma (G34238734)

JUNE 27, 2020

THE GEORGE WASHINGTON UNIVERSITY
2121 I St NW, Washington, DC 20052

Contents

1. Introduction

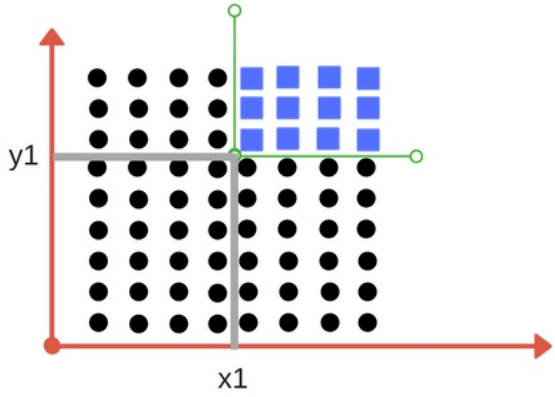
The project is about predicting the age of abalone by predicting its rings based on attributes such as length height weight etc. The motivation for us to do this project is because that economic value of abalone is positively correlated with the age of it but the work for farmers to predicting the age like cutting the shell through the cone, staining it, then counting the number of rings through a microscope is boring and time consuming. If a statistical procedure proves reliable and accurate enough, hours of working hours could be saved. My contribution for this project is to performing preprocessing such as outlier screening, feature engineering for Random Forest model and training the model.

2. Random Forest Model description

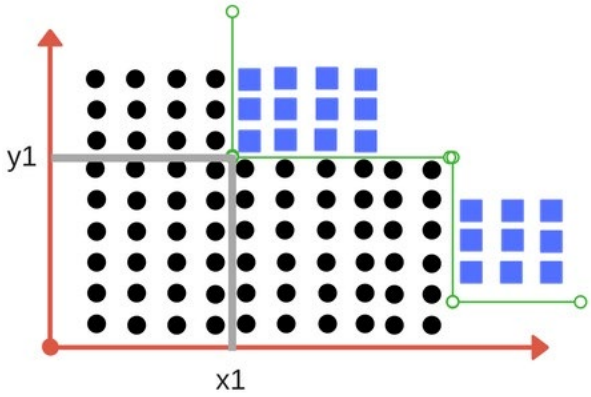
Random Forests randomly chooses observations and features to generate many decision trees to average the result. While a Decision tree generates rules based on the features in the data. The main issue with Random Forests is that they are slow to gather real-time predictions on but are extremely fast to train. However, RF prevents overfitting by generating trees on random subsets. We can see Random Forest as an aggregate of bunch of Decision Trees. To understand what Random Forest, we are having to know what Decision Tree is first.

2.1: Decision Trees

As we can see for the following graph, there are 2 classes one with black dots and the other one with blue squares. It is impossible to divide them into classes with only one line; however, we do need more than one line to separating the classes just like line x1 and line y1.



Then the Decision Tree will repetitively divide the working area into sub part by identifying lines. (There may be two distant regions of the same class divided by other as we have shown in the following graph)



We can think of decision tree as a series of yes/no questions asked about our data eventually leading to a predicted class or continuous value in the case of regression. When Decision Tree meet either it has divided into classes that only containing members of single class (pure) or

some criteria of classifier attributes are met it will stop. The following formula is to calculate Entropy in other words it is measure of impurity.

$$H = -\sum p(x) \log p(x)$$

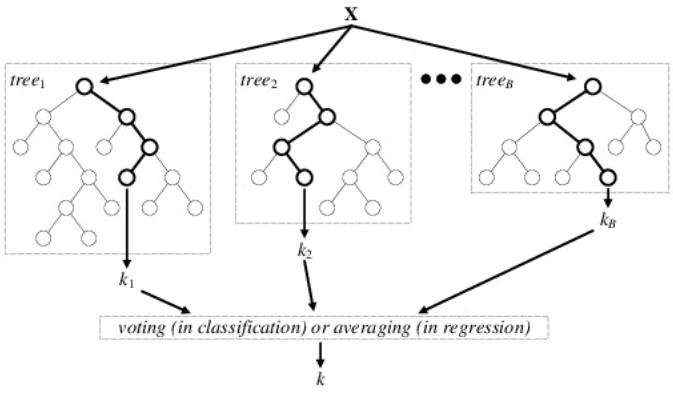
The decision tree will select the feature with less impurity for division. Suppose we divide the classes into multiple branches, the information gain at any node is defined as the following equation.

Information Gain(n) = Entropy(x) – ([weighted average] * entropy (children for feature))

With information gain and entropy, we can build a decision tree.

2.2: Random Forest

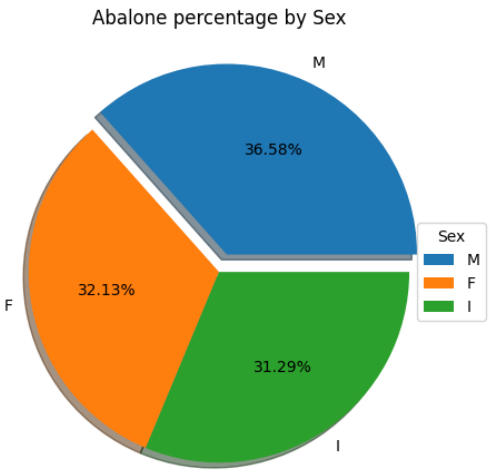
Random Forest creates a set of decision trees by randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object.

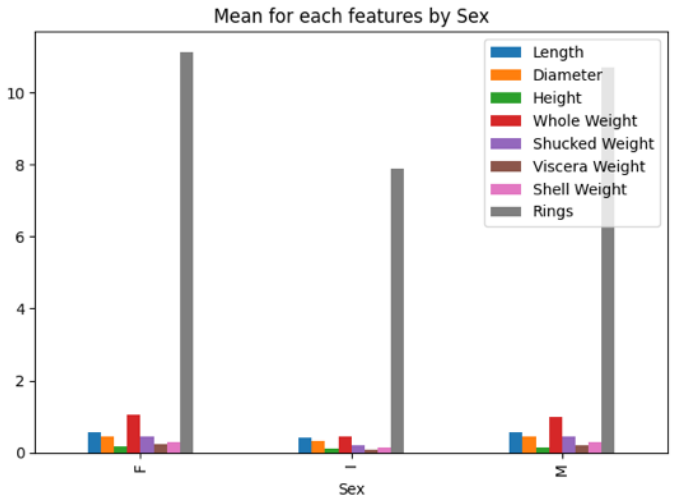


With a single decision three, the result may be prone to a noise but aggregate of many decision trees to build a forest can reduce the effect of noise offering much accurate final results. The basic parameters to Random Forest can be total number of trees to be generated and decision tree related parameters such split criteria, minimum split etc. There are a lot of tuning parameters in Random Forest which included in the package Sklearn in python.

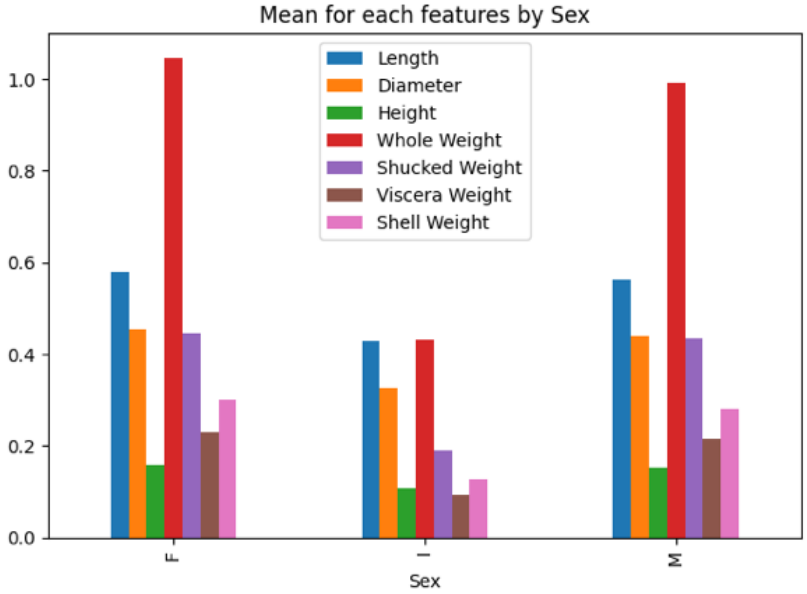
3. Contribution in project (Detailed)

First of all, I want to see some basic information such as the distribution of Sex and the mean for each feature by sex before performing data cleaning. I have generated two plots as following. The left pie chart showing the percentage for each sex and the right chart showing the mean for each feature by sex.





However, the range for Rings is much bigger than other features. So, I decided to generate a new chart. The following bar chart is showing the mean for each feature by sex without Rings in order to see information more clearly.



In the data description, the age of abalone is the number of rings + 1.5. In order to see the prediction directly, I drop the column ‘Rings’ and replace with ‘Age’ even though it will not change much for our prediction.

```
dataset['Age'] = dataset['Rings’] +1.5
dataset.drop('Rings’, axis=1, inplace=True)
```

Next, I want to check if the dataset has any missing values and data type for each feature also some basic information for each feature such as mean, std etc.:

```
print(dataset.isnull().sum())
print(dataset.dtypes)
print(dataset.describe())
```

The result listed on the left below showing that I get no missing value in my dataset. On the right it shows that all features with data in number except for Sex.

```
(4177, 9)
Sex               0
Length            0
Diameter          0
Height            0
Whole Weight      0
Shucked Weight    0
Viscera Weight    0
Shell Weight      0
Age               0
dtype: int64
```

```
Sex               object
Length           float64
Diameter         float64
Height           float64
Whole Weight     float64
Shucked Weight   float64
Viscera Weight   float64
Shell Weight     float64
Age              float64
dtype: object
```

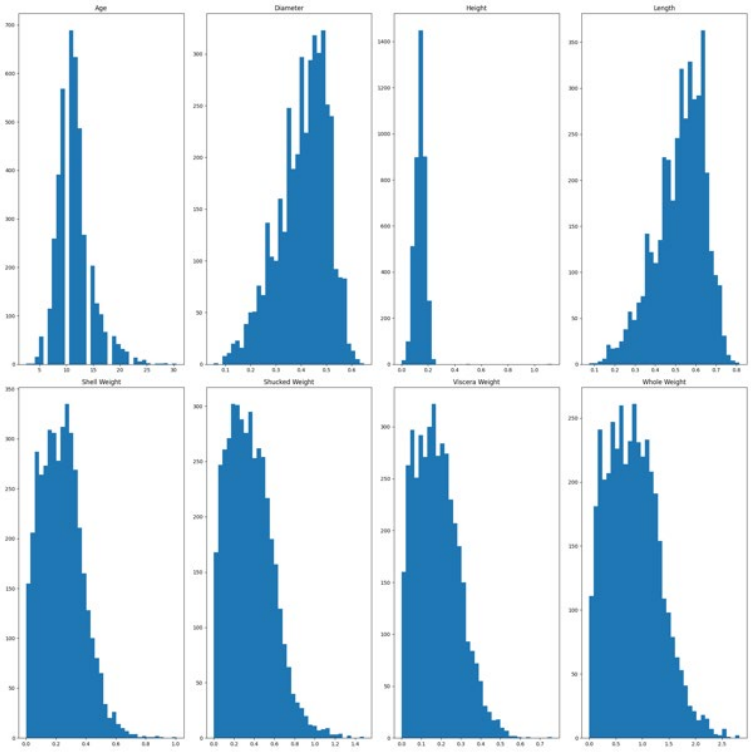
By looking at the following information, we can see that for each feature their mean are not equal to their median but close to their median. So all features are not normally distributed but somehow close to normality.

	Length	Diameter	Height	Whole Weight	Shucked Weight
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367
std	0.120093	0.099240	0.041827	0.490389	0.221963
min	0.075000	0.055000	0.000000	0.002000	0.001000
25%	0.450000	0.350000	0.115000	0.441500	0.186000
50%	0.545000	0.425000	0.140000	0.799500	0.336000
75%	0.615000	0.480000	0.165000	1.153000	0.502000
max	0.815000	0.650000	1.130000	2.825500	1.488000

	Viscera Weight	Shell Weight	Age
count	4177.000000	4177.000000	4177.000000
mean	0.180594	0.238831	11.433684
std	0.109614	0.139203	3.224169
min	0.000500	0.001500	2.500000
25%	0.093500	0.130000	9.500000
50%	0.171000	0.234000	10.500000
75%	0.253000	0.329000	12.500000
max	0.760000	1.005000	30.500000

We can also check all features distribution by looking at their histogram:

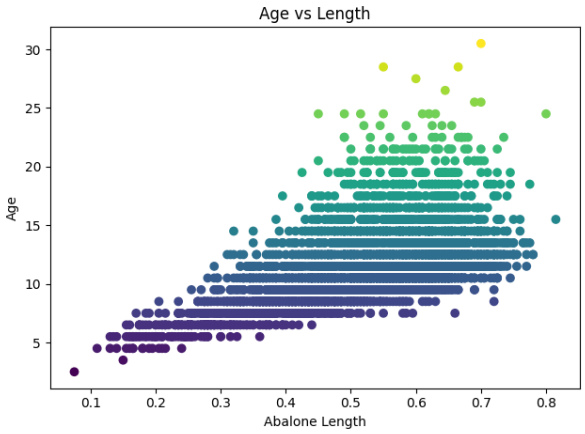
```
dataset.hist(figsize= (20,20), grid=False,layout= (2,4), bins = 35)
plt.show()
```



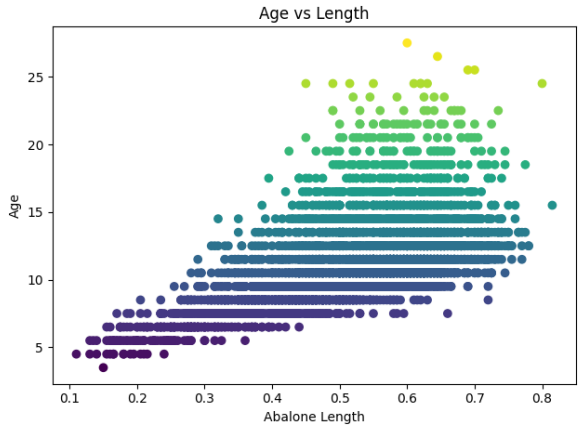
3.1: Data Cleaning

i) Remove outlier by using Scatter Plot

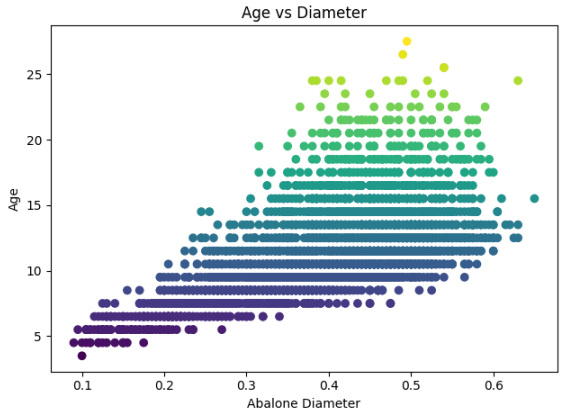
I want to see if there are any outliers by looking at a scatter plot that generated by one feature with Age. We start with Length, and the following graph shows the relationship between the abalone's Length and Age.



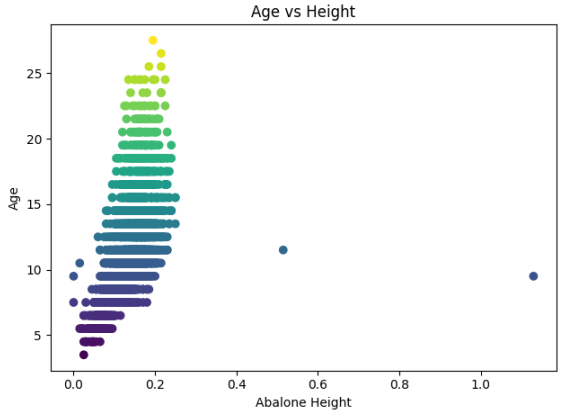
By looking at this graph, I decided to remove three yellow green dots on the top right and the dark blue dot on the left bottom. I get the graph as following.



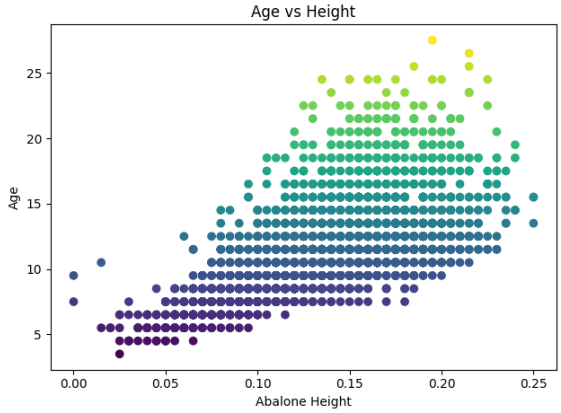
Next, I check Age vs Diameter:



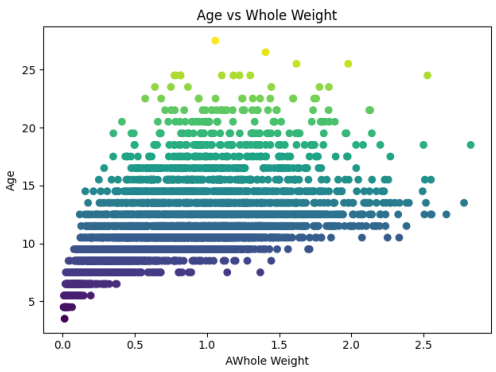
There is no point needs to be removed. So, I move on to check Age vs Height:

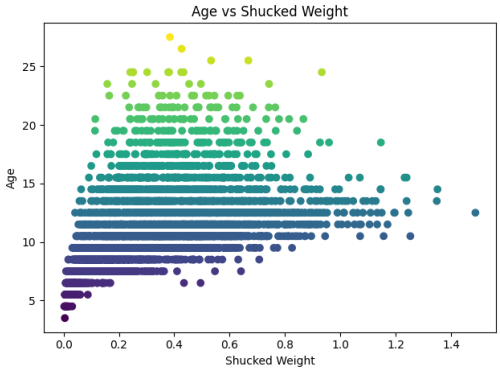


There are two points need to be removed on the right. After I have done that, I get the following graph:

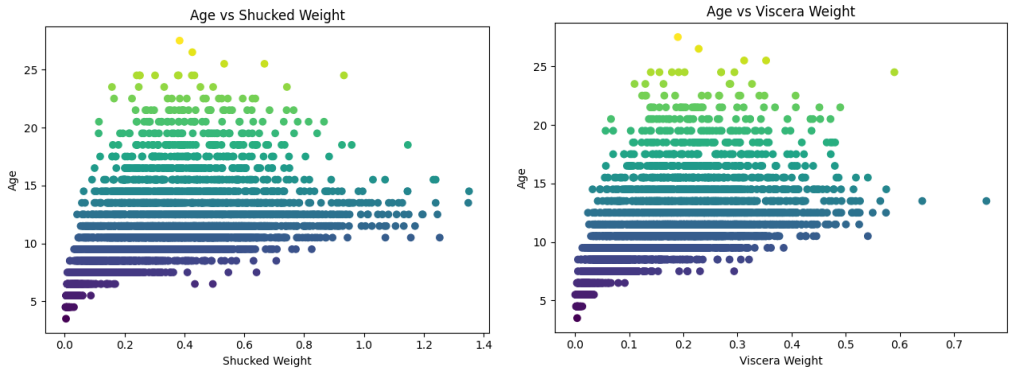


Next I check Age vs Whole weight, Shucked weight, Viscera weight and Shell weight one by one, and performed the same steps as before for previous features.

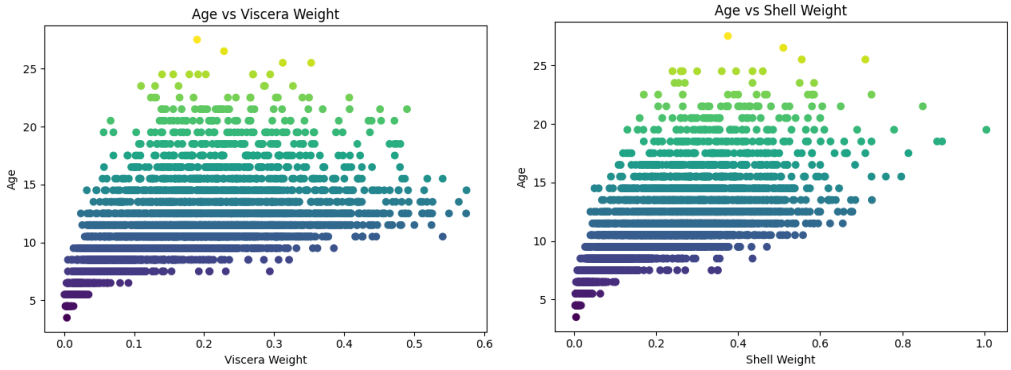




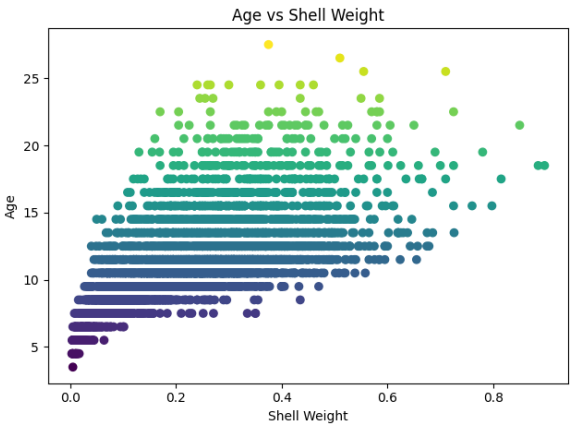
Remove one point on the right in the graph Age vs Shucked Weight



Remove 3 points on the right in Age vs Viscera Weight

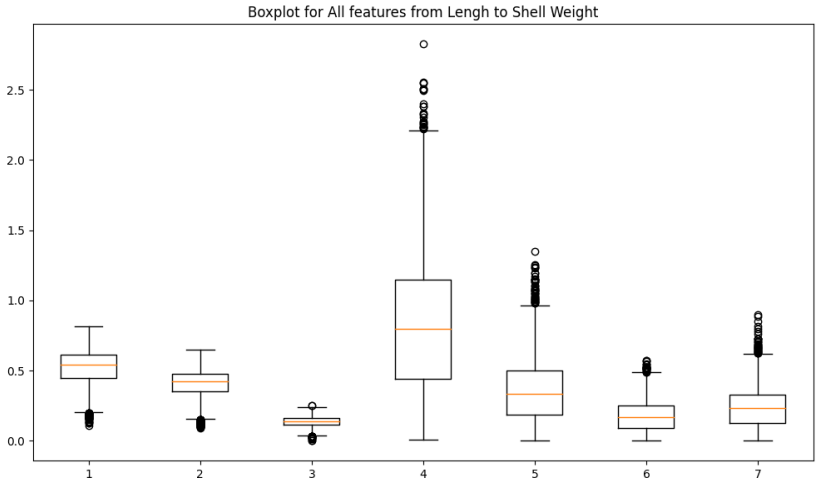


Remove one point on the right in Age vs Shell Weight graph. The we get the following graph.

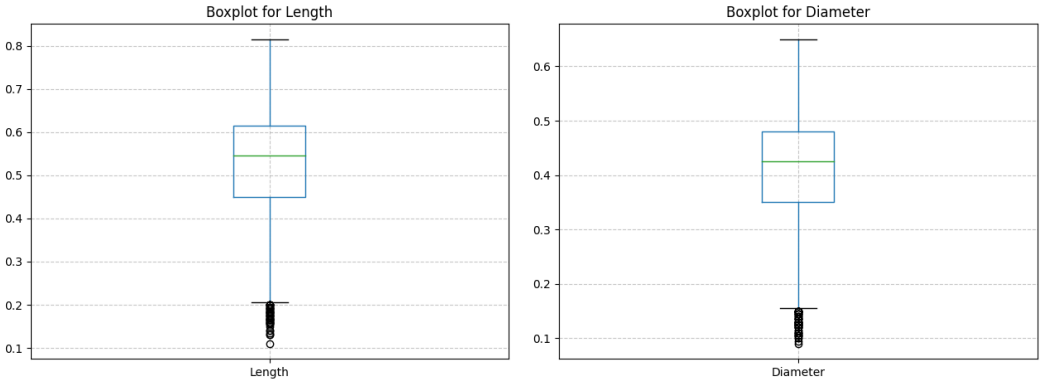


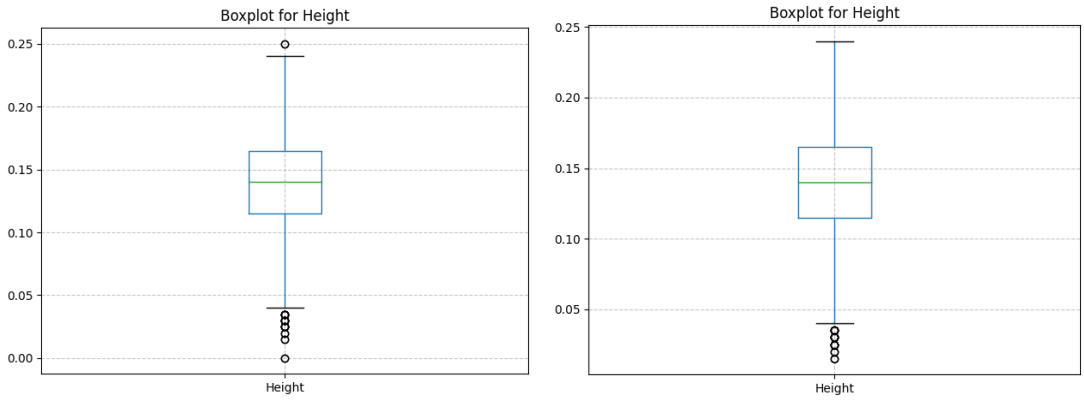
ii) Remove outlier by using Boxplot

Next, I want to see if there are any other outliers that I can remove by check the boxplot for each feature.

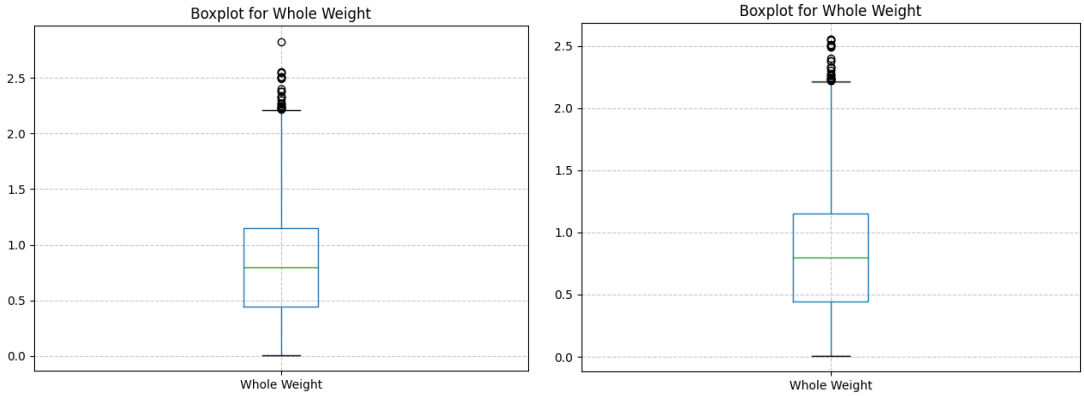


By looking at this combined graph we can see that there definitely exist some outliers but hard to decide which point should be removed. So, we check boxplot for each feature one by one.

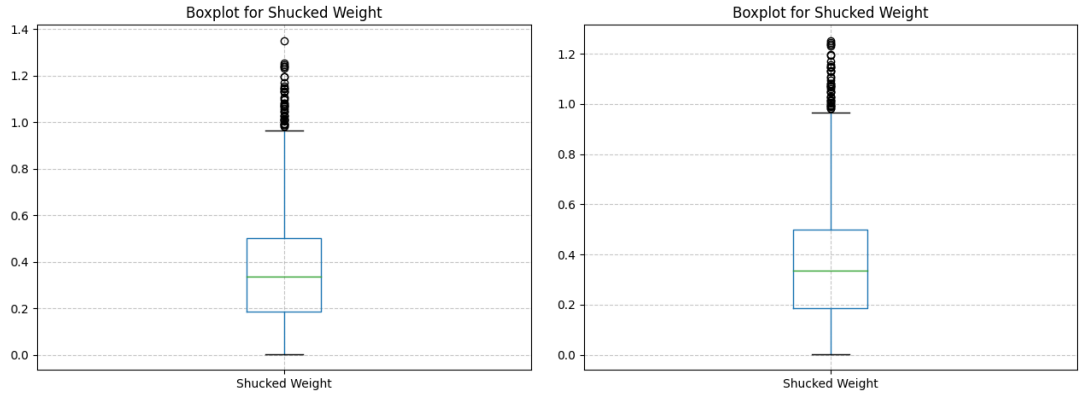




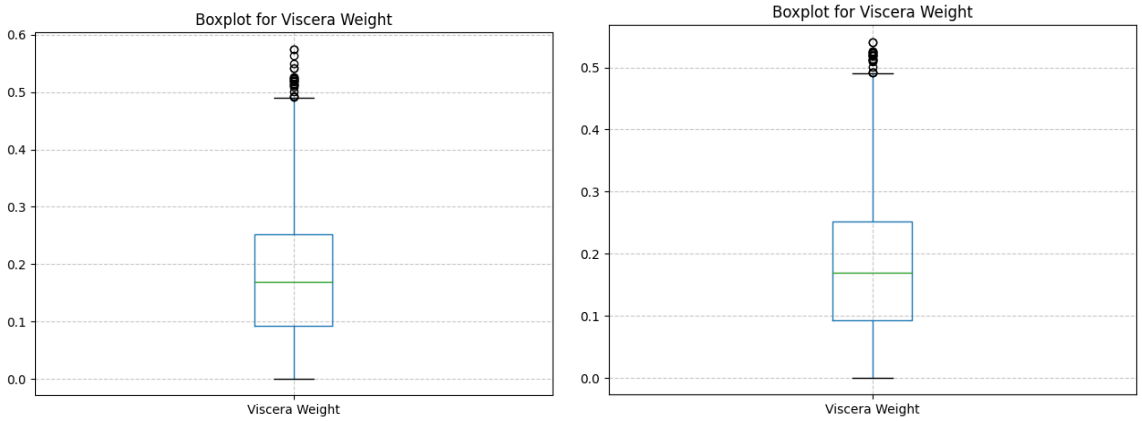
We remove 2 points, one on top, and one on bottom in boxplot graph for Height.



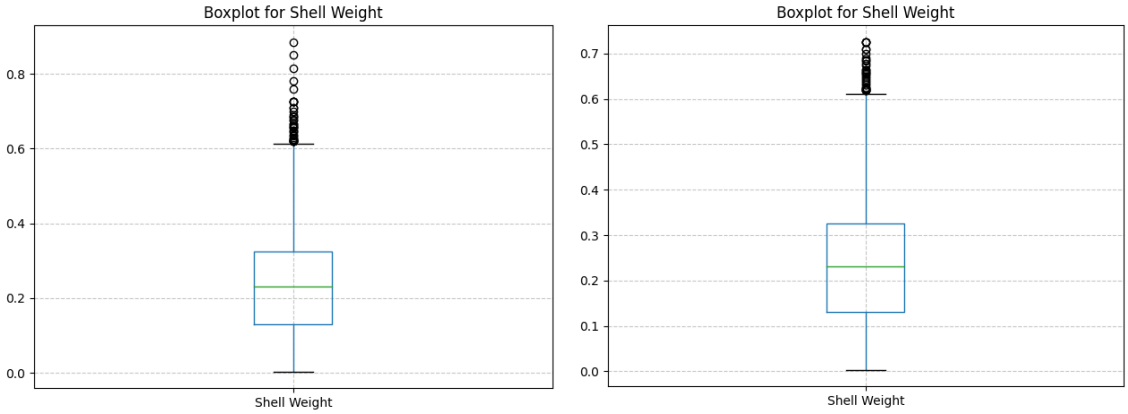
We remove 1 point on top in boxplot graph for Whole Weight.



Remove 1 point on top in boxplot graph for Shucked Weight.

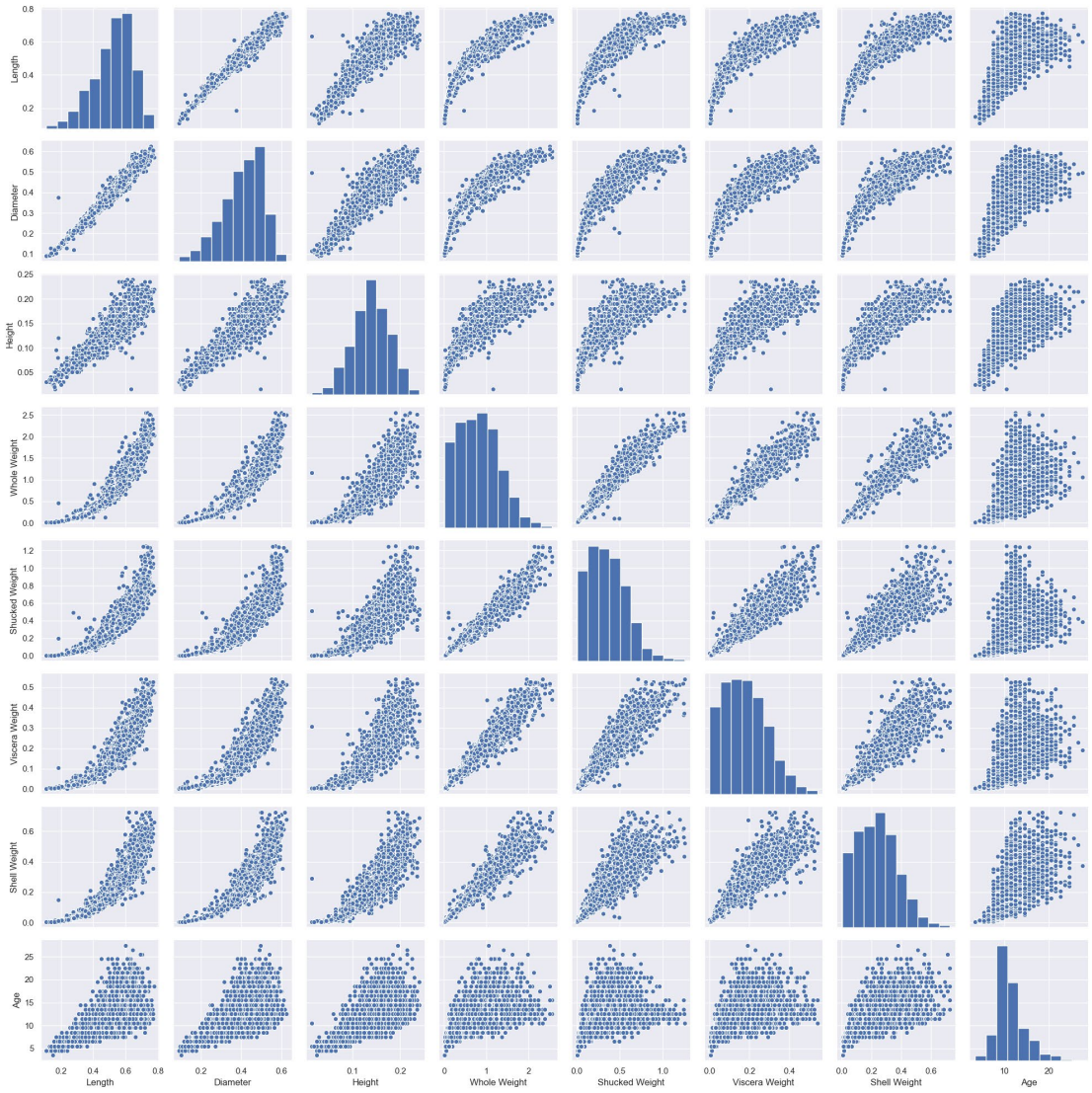


Remove 3 points on top in boxplot graph for Viscera Weight.



Remove 5 points on top in the boxplot graph for Shell Weight.

Then I check the scatter plots again by using pairplot function in Seaborn package. We only check the last row or last column to see if there are any missing outliers.



I split the dataset to x and y. Y as the target which is the Age, and x contains the rest of the features. I scaled x using MinMax to make all data in (0,1). Then I check the data information using “describe” again. Following is the formula for each feature to scale its data.

$$\frac{x_i - min(x)}{max(x) - min(x)}$$

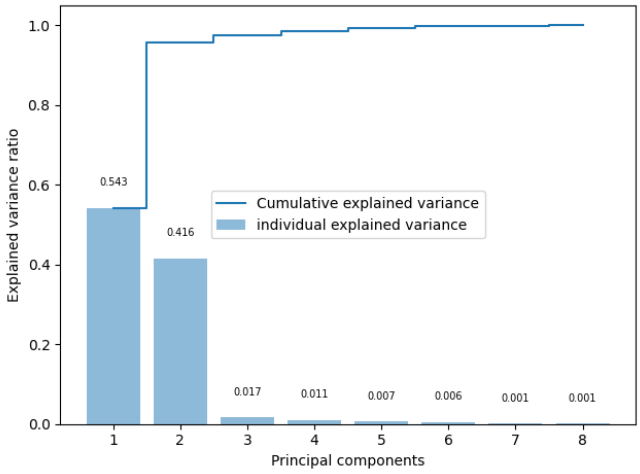
It essentially shrinks the range such that the range is now between 0 and 1. If the distribution is not Gaussian or the standard deviation is very small, the min-max scaler works better. However, it is sensitive to outliers, so if there are outliers in the data, Robust Scaler is a better option.

3.2: Feature Engineering

I did not do much feature engineering for my model but it's doable to separate the Sex column into three columns with Female, Male, and Infant. However, I do encode "Sex" because categorical data is not suitable for regression algorithms. So, I change all rows with "M" (Male) to 2, "F" (Female) to 0 and "I" (Infant) to 1 in the dataset. Also, we can separate some groups for each feature such as for Length, we can set the data into groups with 0.1 increase since the min is 0.075 and the maximum is 0.815. After setup groups for all features we can use Random Forest Classifier.

3.3: Feature Selection

From the scatter plot that I generated before, we can see that all feature has some relationship with the target Age. I also check the feature importance by using PCA function with all components included.



The graph above is the explained variance ratio for each component. The first 4 explained variances are:

```
[0.54270354 0.41555799 0.01672544 0.01071558]
```

With the first 4 principal components, we can explain 98% of the variation of the feature variables as we see the cumulative explained variance.

```
[[0.126 0.369 0.379 0.335 0.395 0.352 0.409 0.383]
 [0.992 0.044 0.044 0.038 0.054 0.056 0.05  0.047]
 [0.01  0.366 0.412 0.481 0.311 0.438 0.42  0.007]
 [0.004 0.451 0.394 0.501 0.075 0.301 0.045 0.54 ]]
```

By using pca.components_ to generate the above matrix. We can see that PC1 is the first row with the most important feature as the 7th feature 0.409 which is Viscera Weight. By looking at this matrix, we can see that all features are needed.

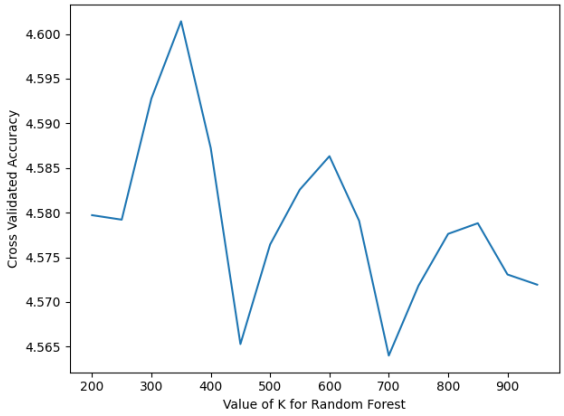
We can also find out the feature importance after we run the Random Forest Regressor by using feature_importances function.

1) Shell Weight	0.503441
2) Shucked Weight	0.169467
3) Whole Weight	0.078438
4) Viscera Weight	0.072763
5) Height	0.053996
6) Diameter	0.053086
7) Length	0.049624
8) Sex	0.019184

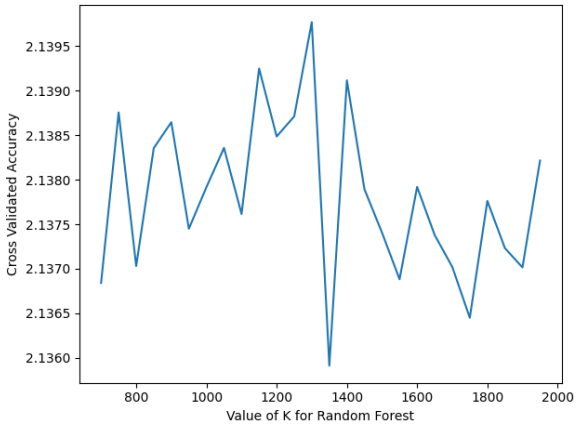
We can see that the Shell Weight has the most importance for the model and Sex has the least importance for the model.

3.4: Training model and parameter optimization

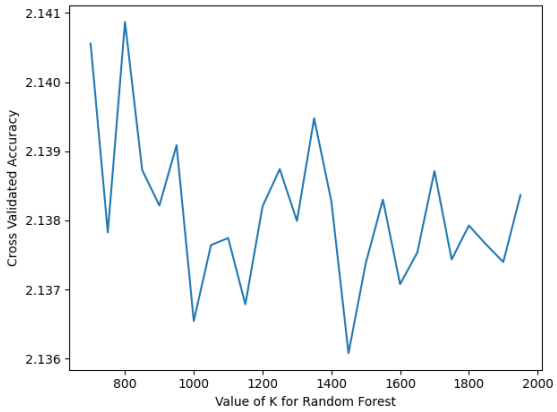
I first to set the tree range from 200 to 900 with step size 50, and I get the cross-validation score as Root Mean Square Error. The graph we can see as the following:



I find out that I can still make some improvement by increasing the tree numbers. So, I set the tree range to (500,2000) with 50 step size, and I get the following graph:

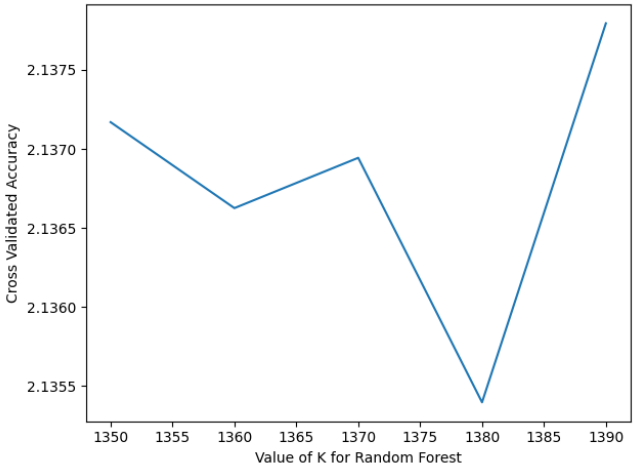


We can see that the best condition is at the three number around 1350. Then I add the PCA to choose 4 main components to train our model with the same range, and I get the following graph:



With less main component information, we need more trees to get to the optimized result.

I set the range for trees to (1300,1400) with step size 10, and I get the following graph.



3.5: Result

Finally, we find the best tree numbers for our model which is 1380 with RMSE 2.1049581 and R2 score 0.535383635270687

4. Conclusion

After using the Random Forest Regressor, I have got the lowest RMSE 2.1029581which is acceptable compare to those scores on some leaderboards. In this summer, I have learned different types of machine learning such as what is supervised leaning, what is unsupervised learning and what is reinforcement learning. After getting some basic ideas about machine learning, I have gained a lot of knowledge about Neuron Model, Network Architectures,

Perceptron Learning Rule, Supervised Hebbian Learning, Performance Optimization, Backpropagation etc. During the process for the final project, I have learned the frameworks for the machine learning process. For example, steps including data collection, data preparation, choose model, train the model, evaluate the model, parameter tuning etc. In the future, I hope that we can have more exercise based on Python so we can have more hands-on experience with Python which is much useful for us to implementing the machine learning.

5. Code percentage

Number of lines copied from internet: Around 60

Number of lines that I modified: Around 30

Percentage = (60-30)/ (60+200) = 11.5 %

References

[1] Wikipedia: Random forest. (2020, June 22). Retrieved June 30, 2020, from https://en.wikipedia.org/wiki/Random_forest

[2] Patel, Savan. “Chapter 3 : Decision Tree Classifier - Theory.” *Medium*, Machine Learning 101, 14 May 2017, medium.com/machine-learning-101/chapter-3-decision-trees-theory-e7398adac567.

[3] “3.2.4.3.2. Sklearn.Ensemble.RandomForestRegressor — Scikit-Learn 0.20.3 Documentation.” *Scikit-Learn.Org*, 2018, scikitlearn.org /stable /modules/generated /sklearn.ensemble.RandomForestRegressor.html.