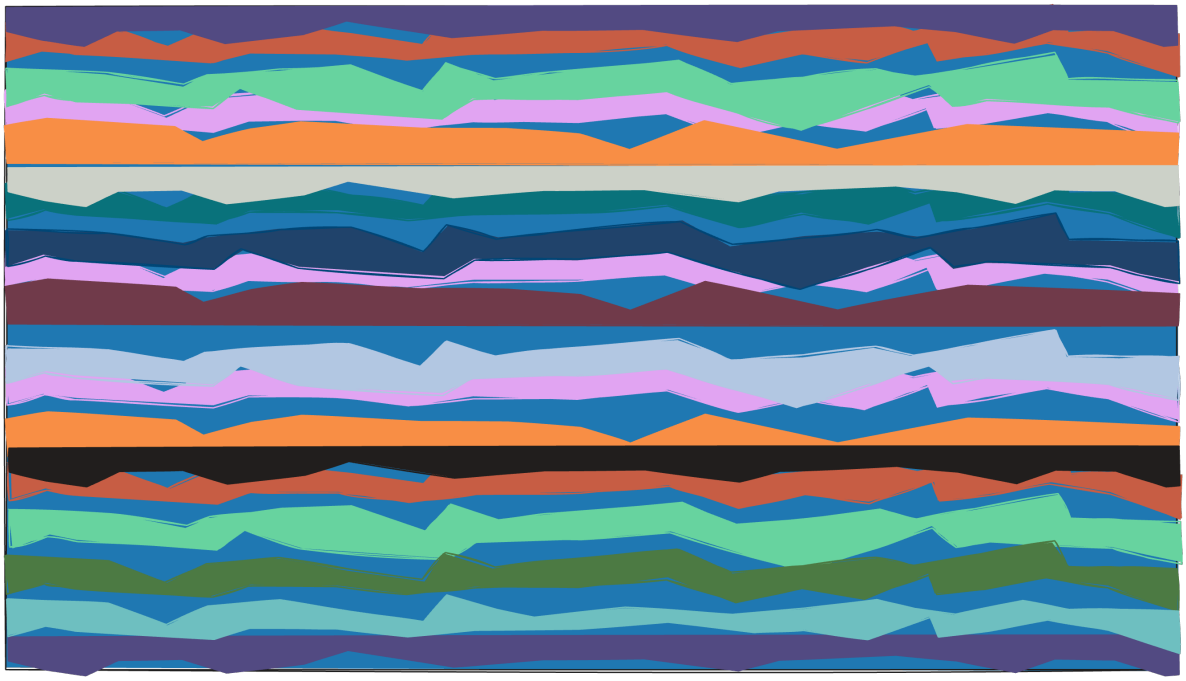
ZZSC5836 Assignment 2

Predicting the Age of abalone from physical measurements

Ty Lange-Smith
z5463091
t.langesmith@student.unsw.edu.au



Hand-drawn Abstract Representation of an Abalone's Shell

Table of Contents

Introduction

This report aims to conduct exploration, analysis and modelling on an abalone dataset. The goal is to develop a predictive model that can estimate the age of abalones from their physical characteristics.

Currently, the method for determining the age of an abalone is both invasive and time-consuming. This method involves a multi-step procedure where the abalone's shell must be cut, stained to highlight the growth rings, and then examined under a microscope to count these rings which corresponds to the abalone's age.

There are other more accessible and non-invasive measurements that can serve as an indicator of an abalone's age. Using these measurements, we aim to construct a predictive model that offers an alternative to the traditional approach.

Data Set Overview

The dataset can be found https://archive.ics.uci.edu/dataset/1/abalone

The dataset contains a variety of information about various abalone physical characteristics. In the dataset we have 4177 rows and 9 columns. These columns are as follows:

Column Name	Type	Description
Sex	Feature	Male, Female or Infant
Length	Feature	Longest shell measurement (mm)
Diameter	Feature	Perpendicular to length (mm)
Height	Feature	with meat in shell (mm)
Whole Weight	Feature	Grams weight of meat (grams)
Shucked Weight	Feature	Grams weight of meat (grams)
Viscera Weight	Feature	Grams gut weight, after bleeding (grams)
Shell Weight	Feature	Grams after being dried (grams)
Rings	Label	The age in years (integer)

A combination of the features will be used to create a model that aims to predict the label which in this case is the Rings column.

Data Processing

All code can be found on https://github.com/tylangesmith/ZZSC5836-assignment-2

Data Processing code can be found in the data_processing.ipynb notebook.

Initial Look and Data Cleaning

1. Clean the data (e.g., convert M, F and I to 0, 1 and 2). You can do this with code or simple find and replace (2 Marks).

Let's take an initial look at our data:

```
# Lets take a quick look at our data
df.head()
```

✓ 0.0s

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
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

We can see that we have all the above feature and label columns, we can also see that the sex column is nominal and for modelling we require this to be in a numeric form - let's process this.

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

✓ 0.0s

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Correlation Map

2. Develop a correlation map using a heatmap and discuss major observations (2 Marks).

Ok, now all our data is in a numerical form let's look at the correlations between the various columns.

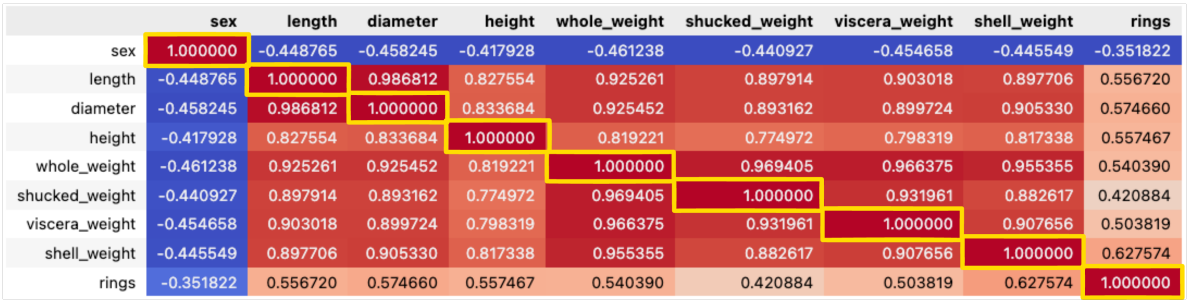
```
df.corr().style.background_gradient(cmap='coolwarm')
```

✓ 0.2s

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
viscera_weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

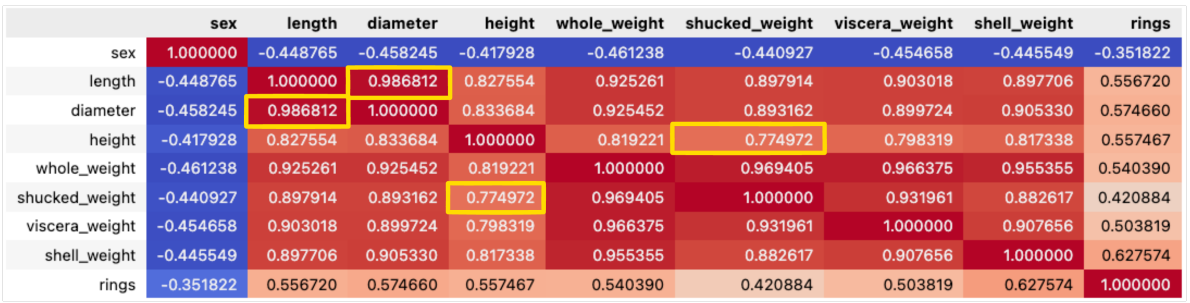
For a correlation matrix a value of 1 indicates a perfect positive correlation, 0 indicates no correlation and -1 indicates a perfect negative correlation.

The first observation we can note is that we have a diagonal line of 1 values from the top left to the bottom right. This is expected as each variable is perfectly positively correlated with itself.



	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
viscera_weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

The next observation we can see is that all the size and weight features are strongly positively correlated with each other - with the strongest positive correlation between length and diameter with 0.986 and the weakest positive correlation between height and shucked weight being 0.775 overall these correlations are strongly positive.



	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
viscera_weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

Next, we can observe an interesting negative correlation between sex and the rest of the columns. Recalling that the male, female and infant values were encoded to 0, 1 and 2 respectively – this indicates that the size and weight values tend to decrease between the sex of the abalone with males having larger size and weight and infants having a smaller size and weight.

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
viscera_weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

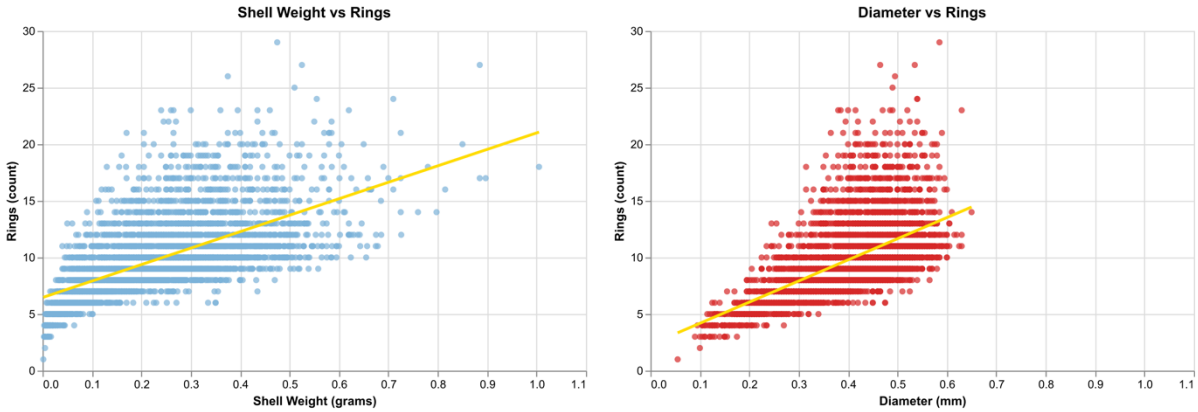
Finally, looking at our rings column or label column we can observe that most features have a positive correlation with the number of rings except again the sex column – with both shell weight and diameter having the strongest positive correlation with the number of rings.

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
sex	1.000000	-0.448765	-0.458245	-0.417928	-0.461238	-0.440927	-0.454658	-0.445549	-0.351822
length	-0.448765	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
diameter	-0.458245	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
height	-0.417928	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
whole_weight	-0.461238	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
shucked_weight	-0.440927	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
viscera_weight	-0.454658	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
shell_weight	-0.445549	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
rings	-0.351822	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

Scatter Plot of Most Correlated Features

3. Pick two of the most correlated features (negative or positive) and create a scatter plot with ring-age. Discuss major observations (2 Marks).

Let's visualise this relationship between shell weight, diameter and the number of rings.



From these scatter plot visualisations, we can again observe the positive correlation for both shell weight and diameter with the number of rings.

Both scatter plots are denser towards the lower range of both shell weight and diameter, potentially indicating there are more abalone in the data set with smaller shell weight and diameter.

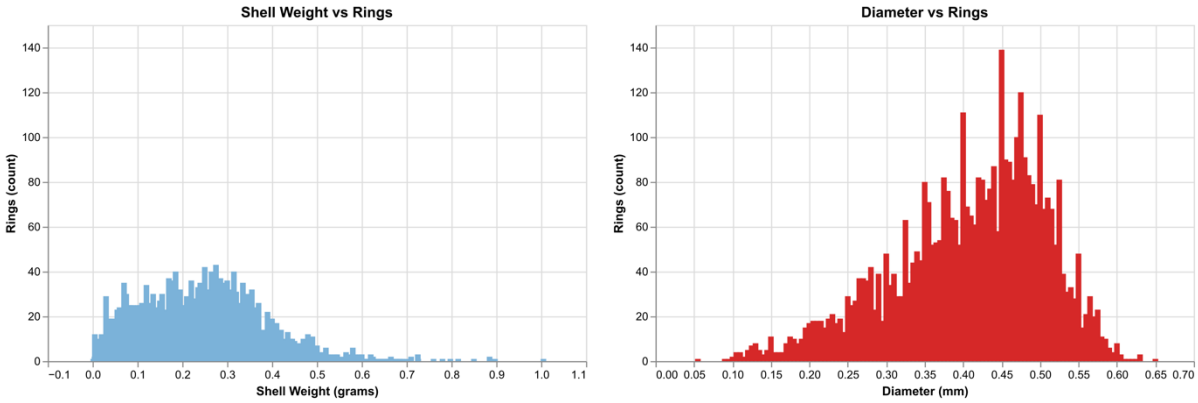
We can also observe that both plots contain outliers especially in the shell weight vs rings plot, showing that there are some abalone that have a high shell weight but don't follow the expected given trend for number of rings.

Finally, we can also observe that the diameter vs rings plot is tighter than the shell weight vs rings plot indicating that diameter might be a slightly better predictor of rings.

Histogram of Most Correlated Features

4. Create histograms of the two most correlated features, and the ring-age. What are the major observations? (2 Marks)

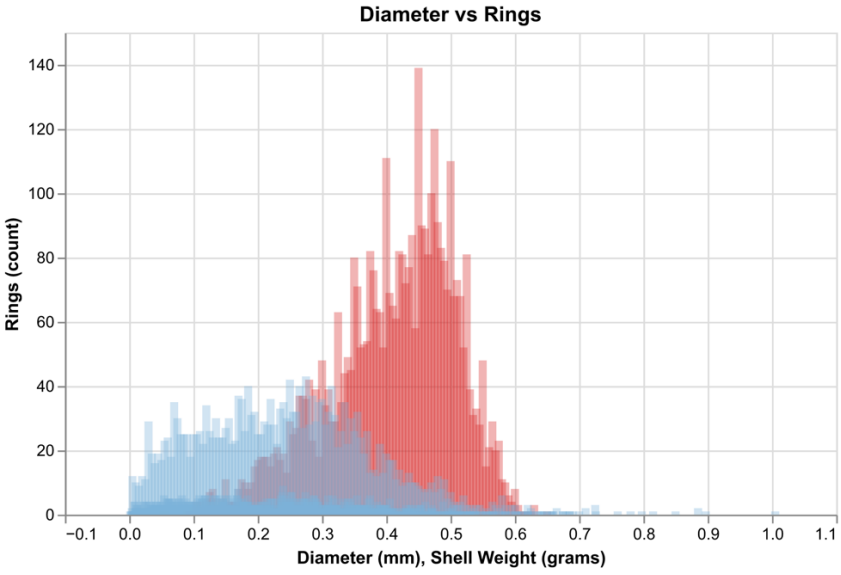
Now, let's look at a histogram visualisation of these features.



From this we can see that both histograms have different distribution properties.

Firstly, we can see that the distribution of the shell weight vs rings histogram appears to be relatively broad, indicating a variability in shell weight among the population. Despite this breadth, the distribution is not purely uniform with a skew to the right. This implies while there is variability there's also a common range where most shell weight fall.

Next, we can look at the diameter vs rings histogram. Although this distribution appears to still be broad it is narrower than the previous histogram's distribution indicating that while there is still some variability most abalone diameters fall around the main central peak. We can also observe that the distribution has steeper tendencies indicating that extreme values are less common. Finally, this distribution is not purely uniform and has a skew to the left.



Superimposing these two histograms we can visualise the difference between the distribution properties. This again reinforces the difference in spread, range and skewness.

Create Training and Testing Split

5. Create a 60/40 train/test split - which takes a random seed based on the experiment number to create a new dataset for every experiment (2 Marks).

Now let's create a reusable function that splits our data into a 60/40 train/test split and can take a random seed so we can specify various splits for experimentation.

```python
# Lets create a function that splits our data into a 60/40 train/test split
# We also need to make sure the function can accept a random seed
def get_train_test_split(df_X: pd.DataFrame, df_y: pd.DataFrame, random_seed: int) -> tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    """
    A function that splits our data into a 60/40 train/test split and accepts a random seed

    Args:
        df_X (pd.DataFrame): The dataframe containing our feature (X) variables
        df_y (pd.DataFrame): The dataframe containing our label (y) variables
        random_seed (int): The random seed to use for reproducibility and experimentation

    Returns:
        tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame]: A tuple containing X_train, X_test, y_train, y_test
    """
    # Call the sklearn train_test_split function
    return train_test_split(df_X, df_y, test_size=0.4, random_state=random_seed)
```

✓ 0.0s

This function leverages the train_test_split function from the sklearn library. More information can be found https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Modelling

All code can be found on https://github.com/tylangesmith/ZZSC5836-assignment-2

Data Processing code can be found in the modelling.ipynb notebook.

Develop a Linear Regression Model

1. Develop a linear regression model using all features for ring-age using 60 percent of data picked randomly for training and remaining for testing. Visualise your model prediction using appropriate plots. Report the RMSE and R-squared score. (4 Marks)

Now let's go ahead and create a linear regression model using all features to predict our rings label. This will be the first experiment of many so we will benefit from making this linear regression modelling experiment code reusable.

Checkout the LinearRegressionExperiment dataclass in helpers.py

After fitting this first experiment, we get the following training and testing metrics:

```
# Instantiate the experiment
experiment = LinearRegressionExperiment(
    df_X=df_X,
    df_y=df_y,
    experiment_number=42
)

# Fit, predict and evaluate
experiment.fit_predict_evaluate()

# Report our RMSE and R-squared
print(experiment.evaluation)
```

✓ 0.1s

```
Training Metrics: [RMSE: 4.893183983114856, R-squared: 0.5429807974750387]
Test Metrics: [RMSE: 4.767145362532978, R-squared: 0.5182048268987614]
```

We can interpret these results by first recalling that a root mean squared (RMSE) error of 0 means that the predicted values from the model perfectly match the actual values. In terms of R-squared, a value of 0 means that the model does not explain any variance label and a value of 1 means the model explains all the variance.

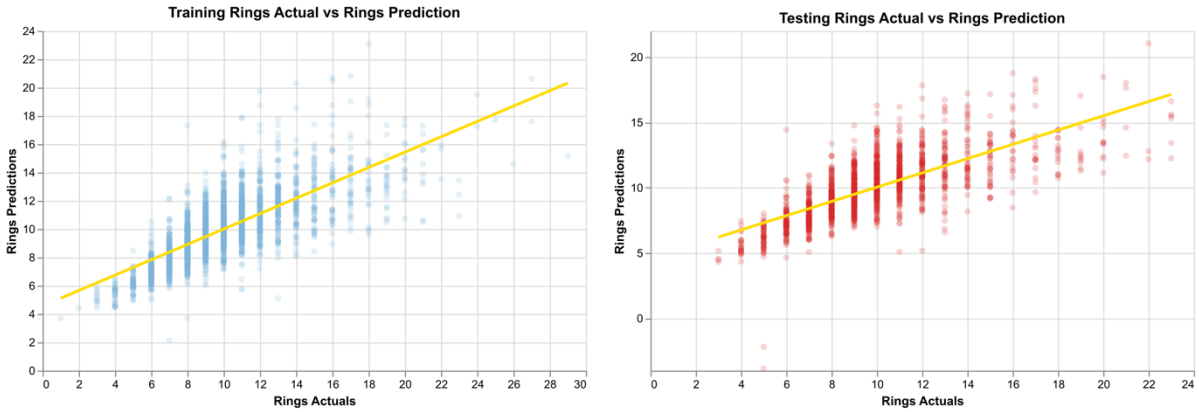
Therefore, for our linear regression model a training RMSE score of 4.89 indicates that our model's predictions were 4.89 units (in this case rings) away from the actual values and

likewise with our testing RMSE score of 4.77. This means for every prediction we make we are on average within around +-4.9 rings away from the actual value.

Furthermore, we have a training R-squared value of 0.543 which means approximately 54.3% of variance can be explained by the model when using the training data, and a testing R-squared value of 0.518 again means we can approximately 51.8% of the variance can be explained by the middle of wind using the testing data. This R-squared value indicates that the model can explain more than half of the variability in the target variable which can be seen as moderately predictive.

Finally, evaluating these metrics together, the model appears to be reasonably consistent across both the training and testing sets. This suggest that the model is not overfitting to the train data.

We can further visualise this model's performance by plotting actual values by the predicted values.



The positive slope of the line in both plots, align with the positive R-squared values, confirming that the model has learned a general trend in the data. One interesting takeaway from the visualisation is that the distance of the points from the trendline increases at higher ring values – this non-uniformity likely contributes to the RMSE value not being lower.

Develop a Linear Regression Model with Data Normalisation

1. Develop a linear regression model with all input features
   i) ~~without normalising input data~~ (this was done in the above)
   ii) with normalising input data. (2 Marks)

Now let's look at how normalising the input data effects the model performance. As mentioned we'll leverage our reusable code we wrote above which allows for a parameter that'll normalise our input features using the inbuilt sklearn standard scalar which can be found here https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

```
# Lets now run an experiment with data normalised using a standard scaler
experiment = LinearRegressionExperiment(
    df_X=df_X,
    df_y=df_y,
    experiment_number=42
)

# Fit, predict and evaluate
experiment.fit_predict_evaluate(normalise_X=True)

# Report our RMSE and R-squared
print(experiment.evaluation)
```

✓ 0.0s

```
Training Metrics: [RMSE: 4.893183983114856, R-squared: 0.5429807974750387]
Test Metrics: [RMSE: 4.767145362532981, R-squared: 0.5182048268987611]
```

For this experiment, we use the same experiment number to ensure our train/test split was the same as the non-normalised experiment above – this allows us to compare the two results fairly.

Comparing these normalised model scores with the above non-normalised model scores, we can conclude that normalising the input data does not make a significant difference to model performance.

Develop a Linear Regression Model with Most Correlated Features

Develop a linear regression model with two selected input features from the data processing step. (2 Marks)

Now, let's look at how well a model that uses our two most correlated features performs. Again, we can reuse our linear regression model experiment class to perform this experiment.

Recalling from above the two features that are most correlated with our label column is shell weight and diameter. We first need to prepare the input data set with only these two feature columns.

```
# Lets create a linear regression model with our 2 most correlated features
# Recalling from before these where `shell_weight` and `diameter`

# Lets arrange a dataframe with only thes 2 features
df_X_2_features = df_X[['shell_weight', 'diameter']]

experiment = LinearRegressionExperiment(
    df_X=df_X_2_features,
    df_y=df_y,
    experiment_number=42
)

# Fit, predict and evaluate
experiment.fit_predict_evaluate()

# Report our RMSE and R-squared
print(experiment.evaluation)
```

✓ 0.0s

```
Training Metrics: [RMSE: 6.511299050343968, R-squared: 0.39185023296519517]
Test Metrics: [RMSE: 5.987007789295018, R-squared: 0.394918502617424]
```

Again, in this case, we keep the experiment number, the same, so we can fairly compare this result to the previous results. Using only these two features we can see a significant decrease in in model performance as indicated by the RMSE and R-squared values.

Experimentation

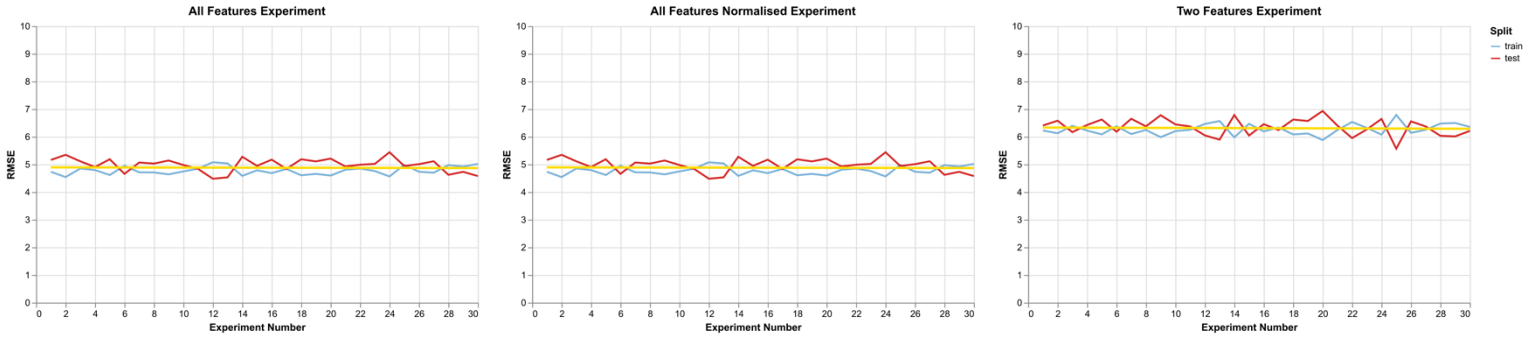
In each of the above investigations, run 30 experiments each and report the mean and std of the RMSE and R-squared score of the train and test datasets. Write a paragraph to compare your results of the different approaches taken. Note that if your code can't work for 30 experiments, only 1 experiment run is fine. You won't be penalised if you just do 1 experiment run. (2 Marks)

Now let's repeat the above investigations 30 times so we can average the model performance for each. The results are as follows:

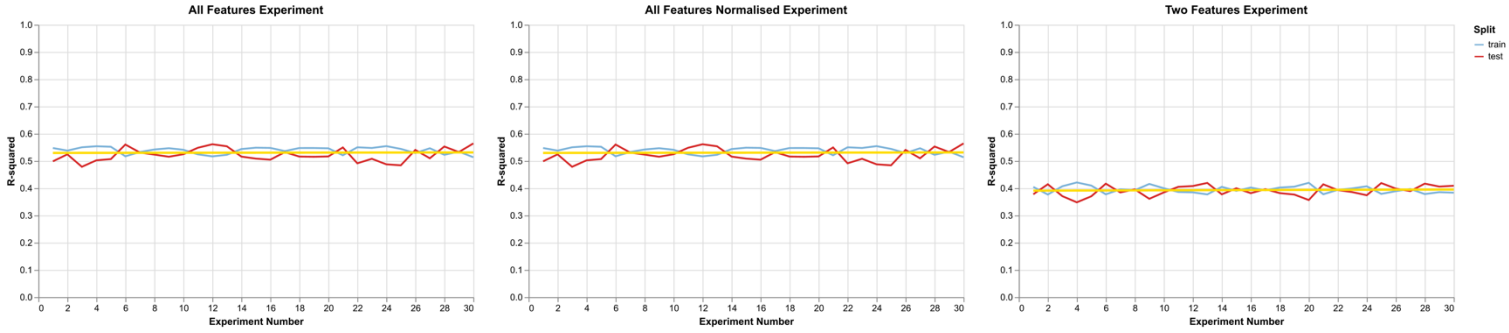
	experiment	split	rmse_mean	rmse_std	rsquared_mean	rsquared_std
1	all_features	train	4.777105	0.150145	0.538882	0.012411
0	all_features	test	4.984308	0.241033	0.521976	0.023865
3	all_features_normalised	train	4.777105	0.150145	0.538882	0.012411
2	all_features_normalised	test	4.984308	0.241033	0.521976	0.023865
5	two_features	train	6.266308	0.201212	0.395219	0.013043
4	two_features	test	6.353889	0.304369	0.391050	0.019540

These results are like what we saw in our initial experiments. We see that the model trained on all features without normalisation again performed closely to the model with normalisation. We also saw the model train on the two most correlated features perform significantly worse.

The following visualisations reinforce these observations.



This visualisation contrasts the three different investigations with respect to their training, testing and mean RMSE scores – again highlighting the worse performance of the two features investigation.



A similar set of conclusions can be drawn from the visualisations contrasting the three different investigations with respect to their training, testing and mean R-squared scores.