1. **Problem Definition**

The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Measurement Unit** |
| Sex | Gender of the abalone –  M, F, and I (infant) | Nominal | -- |
| Length | Longest shell measurement | Continuous | mm |
| Diameter | Perpendicular to length | Continuous | mm |
| Height | With meat in shell | Continuous | mm |
| Whole weight | Total Weight of the abalone | Continuous | grams |
| Shucked weight | Weight of meat | Continuous | grams |
| Viscera weight | Gut weight (after bleeding) | Continuous | grams |
| Shell weight | Weight after being dried | Continuous | grams |
| Rings | No. of Rings found in the abalone -> +1.5 gives the age in years. | Integer | -- |

We have to predict the rings of each abalone which will lead us to the age of that abalone.

We can find the dataset on below location :

<https://github.com/dsrscientist/dataset1/blob/master/abalone.csv>

1. **Data Analysis**

There is total 9 columns including target variable, all of them are self-explanatory.

And we have 4177 records across the dataset.

Out of 9 columns, 8 are numerical & rest 1 is categorical in nature.

As per data, no null value is present in the dataset in any of the 9 columns, 1 - Object Type, 7 - Float Type and 1 - Int Type.

Following are the observations:

1. No feature has Minimum Value = 0, except Height
2. All features are not Normally Distributed, (theoretically if feature is normally distributed, Mean = Median = Mode).
3. But features are close to Normality
4. All columns are numerical, except Sex
5. Each feature has different Scale

**Skewness**:

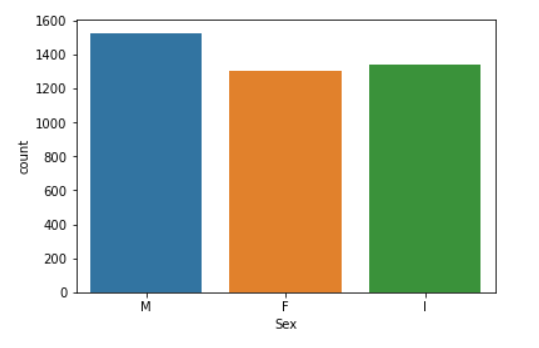
|  |  |  |
| --- | --- | --- |
|  | **Features** | **Skewness degree** |
| 1 | Height | 3.127694 |
| 2 | Age | 1.113702 |
| 3 | Shucked weight | 0.71884 |
| 4 | Shell weight | 0.620704 |
| 5 | Viscera weight | 0.59164 |
| 6 | Whole weight | 0.530768 |
| 7 | Diameter | -0.60898 |
| 8 | Length | -0.63964 |

For normally distributed data, the skewness should be about 0. For unimodal continuous distributions, a skewness value > 0 means that there is more weight in the right tail of the distribution. The function skew test can be used to determine if the skewness value is close enough to 0, statistically speaking.

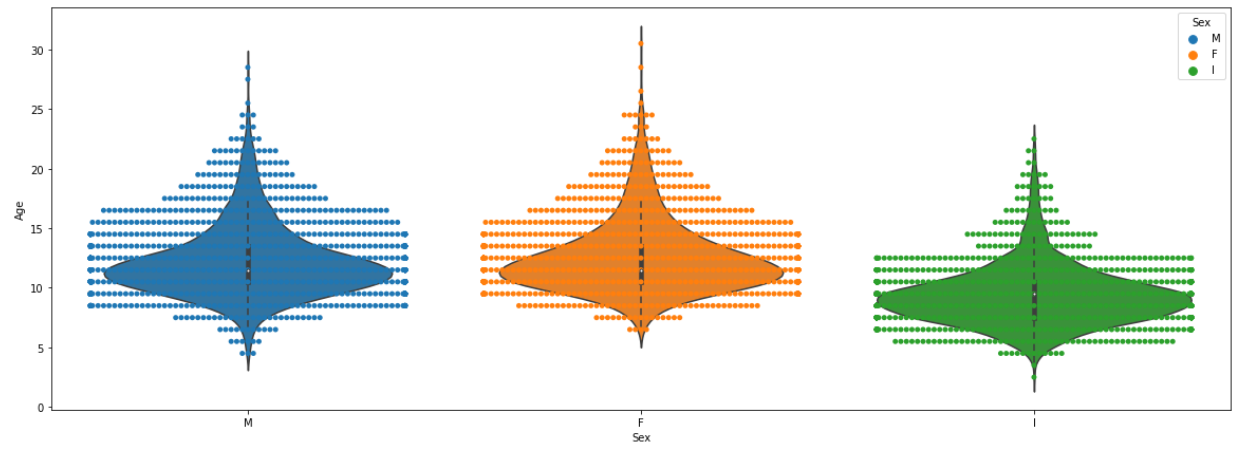
Height has highest skewedness followed by age, Shucked weight (can be cross verified through histogram plot)

1. **EDA Concluding Remarks**

Here is the count plot of the sex column:



From the count plot, we can deduce that almost same distribution gender wise in the dataset.



Combination of Swarm plot & Violin plot with Gender on X - axis and Age on the Y – axis.

By looking at the plot, we can say that:

Male - Age majority lies in between 7.5 years to 19 years

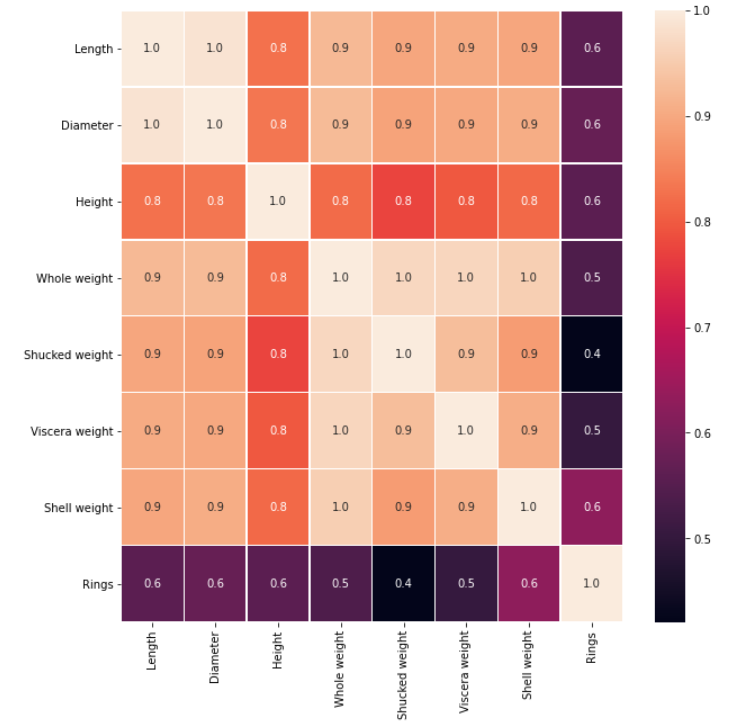
Female - Age majority lies in between 8 years to 19 years

Immature - Age majority lies in between 6 years to < 10 years

**Bivariate Analysis:**

Bivariate analysis is vital part of data analysis process for, it gives clear picture on how each feature are affected in presence of other features.  
It also helps us understand and identify significance features, overcome multi-collinearity effect, inter-dependency and thus, provides insights on hidden data noise pattern.

We plotted the pair plot and found that key insights length is linearly correlated with diameter while, non-linear relation with height, whole weight, shucked weight, viscera weight and shell weight.



Through heatmap, we can say that:

1. Whole Weight is almost linearly varying with all other features except age.
2. Height has least linearity with remaining features.
3. Age is most linearly proportional with Shell Weight followed by Diameter and length.
4. Age is least correlated with Shucked Weight.

Such high correlation coefficients among features can result into multi-collinearity. We need to check for that too, however, we have not done it here.

1. **Pre-processing Pipeline**

We have used **LabelEncoder** as the main pre-processor to convert the only categorical column to Numerical which is Gender.

In machine learning, we usually deal with datasets that contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labelled in words.

**Label Encoding** refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

For Example:

Suppose we have a column Height in some dataset.

|  |
| --- |
| **Height** |
| Tall |
| Medium |
| Short |

After applying label encoding, the Height column is converted into:

|  |
| --- |
| **Height** |
| 0 |
| 1 |
| 2 |

Where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

Also, as mentioned earlier, each feature has different Scale. Hence, used **StandardScaler** to transform the data into same scale.

Standardize features by removing the mean and scaling to unit variance.

The standard score of a sample x is calculated as:

z = (x - u) / s

where u is the mean of the training samples or zero if **with\_mean** = False, and s is the standard deviation of the training samples or one if **with\_std** = False.

Centring and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g., Gaussian with 0 mean and unit variance).

For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centred around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

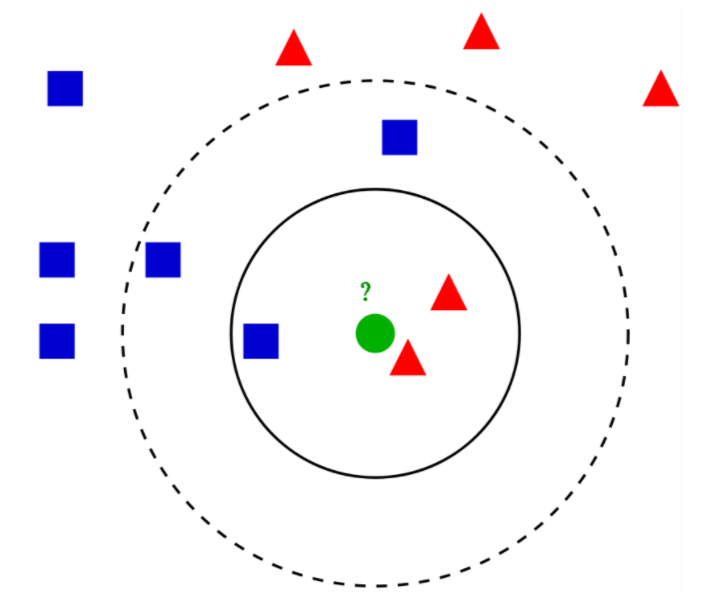
This scaler can also be applied to sparse CSR or CSC matrices by passing **with\_mean** = False to avoid breaking the sparsity structure of the data.

1. **Building Machine Learning Models**

Here to predict the outcome of the Abalone age we have constructed below 6 regression models, as age will be continuous in nature not categorical:

1. K Nearest Neighbors Regressor
2. Support Vector Regressor
3. Decision Tree Regressor
4. Random Forest Regressor
5. XGB Regressor
6. Linear Regression
7. **K- Nearest Neighbors Regressor:**

Given a new, unknown sample, how do you tell which group it belongs to? Well, naturally, you would look at the surrounding points. But the result would be really dependent on how far you look. If you just look at the closest 3, (solid circle) the green dot would belong to red triangles. If you look further, (the dashed circle) dot would be classified as a blue square.



kNN works the same way. Depending on the value of k, the algorithm classifies new samples by the majority vote of the nearest k neighbors in classification. For regression which predicts the actual numerical value of a new sample, the algorithm just takes the mean of the nearest k neighbors. That’s it. As easy as that.

In this model, with the help of GridSearchCV we have calculated n\_neighbors = 4 as the best value, “neg\_mean\_squared\_error” is used for scoring.

KNN Root Mean Squared Error value: 0.42

Accuracy Score with KNN Regression is **98.2** %

1. **Support Vector Regressor:**

Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value. The threshold value is the distance between the hyperplane and boundary line. The fit time complexity of SVR is more than quadratic with the number of samples which makes it hard to scale to datasets with more than a couple of 10000 samples.

In this model, with the help of RandomizedSearchCV, we have calculated the best value for kernel as 'linear', gamma as ‘1’ and C as ‘1’.

We have kept CV = 5, n\_iter = 20 and scoring = 'neg\_mean\_squared\_error', thus Fitting 5 folds for each of 20 candidates, totalling 100 fits.

SVR Root Mean Squared Error value: 0.17

Accuracy Score with SVR is **99.69** %

1. **Decision Tree Regressor**

The decision tree is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

We can see that if the maximum depth of the tree (controlled by the max\_depth parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e., they overfit.

In this model, with the help of GridSearchCV, we have calculated max\_depth = 14 as the best value and kept cv = 10.

Decision Tree Regressor Root Mean Squared Error value: 0.16

Accuracy Score with Decision Tree Regressor is **99.75** %

1. **Random Forest Regressor**

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

In this model, with the help of RandomizedSearchCV, we have calculated the best value for n\_estimators as ‘180’, criterion as 'mse' and bootstrap as False, have kept cv = 5, n\_iter = 10, thus fitting 5 folds for each of 10 candidates, totalling 50 fits.

Random Forest Regressor Root Mean Squared Error value: 0.18

Accuracy Score with Random Forest Regressor is **99.67** %

1. **XGB Regressor**

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modelling problems.

Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.

Models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, “gradient boosting,” as the loss gradient is minimized as the model is fit, much like a neural network.

In this model, with the help of RandomizedSearchCV, we have calculated the best value for reg\_lambda as ‘0.1’, n\_estimators as ‘500’, max\_depth as 2, and learning\_rate as ‘0.1’, have kept cv = 5, n\_iter = 20.

XGB Regressor Root Mean Squared Error value: 0.15

Accuracy Score with XGB Regressor is **99.74** %

1. **Linear Regression**

LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Linear Regression Root Mean Squared Error value: 0.19

Accuracy Score with Linear Regression is **99.64** %

1. **Concluding Remarks**

We have constructed and tried six models and found that accuracy varies between 98 to 99.75 %, but Root Mean Squared Error varies between 0.15 to 0.42.

Hence, we have selected the model which is having least Root Mean Squared Error and high accuracy, which is XGB Regressor.

XGB Regressor Root Mean Squared Error value: 0.15

Accuracy Score with XGB Regressor is **99.74** %