

PREDICTION OF CRAB AGE USING MACHINE LEARNING TECHNIQUES

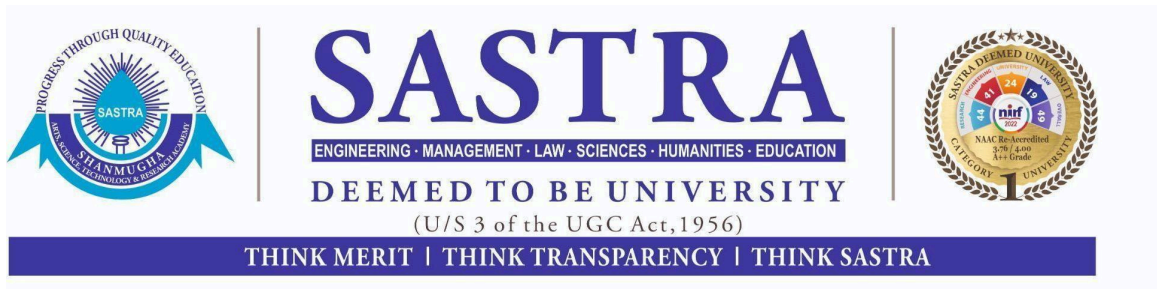
Report submitted to SASTRA Deemed to be University As per the requirement for the course

CSE425 : MACHINE LEARNING ESSENTIALS

Submitted by

Puluru Venkata Gopinadha Reddy
(Reg No: 125018022, B. Tech Computer Science and Business Systems)

OCTOBER- 2024



SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU, INDIA – 613 401

Table of Contents

Abstract	1
Introduction	1
Related Work	2
Background	3
Methodology	13
Results	15
Discussion	19
Learning Outcome	19
Conclusion	20
Code Link	21

ABSTRACT

This project aims to predict the age of crabs using various regression models based on their physical characteristics. The dataset includes features such as length, diameter, height, weight, shucked weight, viscera weight, shell weight, and sex. We employed multiple regression models, including Linear Regression, RandomForestRegressor, XGBRegressor, CatBoostRegressor and LightGBM. The performance of these models was evaluated using metrics such as Mean Absolute Error (MAE) ,Mean Squared Error (MSE) and R2_Score. Final results indicate that ensemble methods like CatBoostRegressor and LightGBM provide superior predictive accuracy compared to linear models.

INTRODUCTION

Importance of Dataset: Accurate age prediction is essential for determining crab quality and value in the fishing industry. It also helps with sustainable fishing practices.

Project Objective: The primary objective of this project is to develop a machine learning model that can accurately predict the age of crabs based on their physical measurements. Accurate age prediction is crucial for the fishing and seafood industry, as it helps in determining the quality and market value of crabs. Additionally, understanding the age distribution of crab populations can aid in sustainable fishing practices and resource management.

Problem Statement: Predicting the age of crabs is a challenging task due to the variability in their physical characteristics. The problem can be formulated as a regression task where the goal is to predict a continuous target variable (age) based on several input features. The key challenges

include handling the multicollinearity among features, dealing with potential outliers, and selecting the appropriate model that can generalize well on unseen data. Furthermore, the dataset may contain missing values and categorical variables that need to be preprocessed effectively.

Approach: Use regression models with a focus on handling multicollinearity, outliers, and appropriate model selection

Related Work

References:

Dataset: <https://www.kaggle.com/datasets/sidhus/crab-age-prediction>

<https://www.learnaboutnature.com/invertebrates/crabs/crab-life-cycle/>

https://www.researchgate.net/publication/317003887_Age_determination_in_crustaceans_a_review

<https://medium.com/geekculture/predicting-age-of-a-crab-in-mud-crab-farming-using-machine-learning-1ae3bf030426>

<https://medium.com/@maddiedunlop/exploratory-data-analysis-on-synthetic-crab-age-dataset-efc884261ae>

<https://rpubs.com/sarvinnah/1055418>

<https://www.kaggle.com/datasets/sidhus/crab-age-prediction>

https://github.com/Aravinth-Megnath/Crab_age

Background

- **Models:**

- *Linear Regression*: Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It's simple, interpretable, and often used for predictive modeling and forecasting.
- *RandomForestRegressor*: The RandomForestRegressor is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of these trees. It's robust against overfitting and works well with non-linear relationships, making it suitable for a variety of regression tasks.
- *XGBRegressor*: XGBRegressor is part of the XGBoost library, which stands for Extreme Gradient Boosting. It is a powerful ensemble method based on gradient boosting that focuses on optimizing performance and speed. XGBoost is particularly effective for structured/tabular data and has built-in regularization to prevent overfitting.
- *CatBoostRegressor*: CatBoostRegressor is developed by Yandex and is designed to handle categorical features automatically without the need for extensive preprocessing. It uses a gradient boosting framework and is known for its efficiency and ability to work well with both numerical and categorical data, often outperforming other models in various scenarios.

- *LightGBM*: LightGBM (Light Gradient Boosting Machine) is another gradient boosting framework that uses a histogram-based approach to improve the training speed and efficiency. It's particularly well-suited for large datasets and high-dimensional data, and it supports parallel and distributed learning, making it scalable and fast.

MATH BEHIND THE MODELS:

1. Linear Regression Equation

In its basic form, linear regression assumes the relationship between the dependent variable 'y' and the independent variable(s) 'x' is linear. For one variable, the model is:

$$y = \beta_0 + \beta_1 x + \epsilon$$

For multiple variables (features), it becomes:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

Where:

- 'y' is the predicted output (dependent variable),
- x_1, x_2, \dots, x_n are the input features (independent variables),
- β_0 is the intercept (the value of 'y' when all $x_i=0$),
- $\beta_1, \beta_2, \dots, \beta_n$ are the coefficients of the respective features,
- ϵ is the error term, representing the deviation of the predicted value from the actual value (assumed to be normally distributed with mean 0).

2. Random forest Regression,Xgboost regressor,Catboost regressor,LGBM regressor all these models works by using Decision trees.

Here is a short explanation of how regression works in a decision tree with an example

1. Understanding Decision Trees

A decision tree is a flowchart-like structure where:

- **Internal nodes** represent decisions based on feature values.
- **Branches** represent the outcome of those decisions.
- **Leaf nodes** represent final predictions (in regression, these are typically the average values of the target variable).

2. Splitting Criteria for Regression

For regression tasks, the objective is to predict a continuous output. Decision trees use a splitting criterion to decide where to split the data:

- **Mean Squared Error (MSE)** is a common metric used:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2$$

where y_i are the actual values, \bar{y} is the average of those values, and N is the number of observations.

The goal of splitting is to minimize the MSE after a split.

3. Recursive Partitioning

Once a split is made, the decision tree continues to split each subset of data recursively until:

- A maximum depth is reached.
- A minimum number of samples per leaf is achieved.
- The MSE reduction becomes negligible.

4. Building the Tree

Let's illustrate this with a detailed example.

Example: Predicting House Prices

Step 1: Prepare the Data

Assume we have the following dataset containing information about houses:

Square Footage	Bedrooms	Price
1500	3	300,000
1800	4	350,000
2400	4	500,000
3000	5	600,000
1200	2	200,000

Step 2: Initial MSE Calculation

1. Calculate the overall MSE for the dataset:

- Average price:
$$\bar{y} = \frac{300,000 + 350,000 + 500,000 + 600,000 + 200,000}{5} = 390,000$$
- Calculate the MSE:
$$\begin{aligned} \text{MSE} &= \frac{1}{5} \left((300,000 - 390,000)^2 + (350,000 - 390,000)^2 + (500,000 - 390,000)^2 + (600,000 - 390,000)^2 + (200,000 - 390,000)^2 \right) \\ &= \frac{1}{5} \left((90,000)^2 + (40,000)^2 + (110,000)^2 + (210,000)^2 + (190,000)^2 \right) \\ &= \frac{1}{5} (8,100,000,000 + 1,600,000,000 + 12,100,000,000 + 44,100,000,000 + 36,100,000,000) = 206,000,000 \end{aligned}$$

Step 3: Finding the Best Split

Next, we evaluate potential splits. For this example, let's test splitting on **Square Footage** at **2000**.

1. Left Subset (≤ 2000):

- Data points:
 - (1500, 300,000)
 - (1800, 350,000)
 - (1200, 200,000)
- Average price:

$$\bar{y}_{left} = \frac{300,000 + 350,000 + 200,000}{3} = 283,333$$

- Calculate the MSE for the left subset:

$$\begin{aligned} \text{MSE}_{left} &= \frac{1}{3} ((300,000 - 283,333)^2 + (350,000 - 283,333)^2 + (200,000 - 283,333)^2) \\ &= \frac{1}{3} ((16,667)^2 + (66,667)^2 + (83,333)^2) \approx 1,482,638 \end{aligned}$$

2. Right Subset (> 2000):

- Data points:
 - (2400, 500,000)
 - (3000, 600,000)
- Average price:

$$\bar{y}_{right} = \frac{500,000 + 600,000}{2} = 550,000$$

- Calculate the MSE for the right subset:

$$\text{MSE}_{right} = \frac{1}{2} ((500,000 - 550,000)^2 + (600,000 - 550,000)^2) = \frac{1}{2} ((50,000)^2 + (50,000)^2) = 2,500,000$$

Step 4: Calculate Overall MSE After the Split

Now we can calculate the overall MSE after the split:

- Number of samples: Left subset = 3, Right subset = 2.

- Total samples = 5.

Weighted average of MSE:

$$\text{Overall MSE} = \frac{3}{5} \cdot 1,482,638 + \frac{2}{5} \cdot 2,500,000 \approx 1,958,678$$

Step 5: Evaluate Splits and Choose Best One

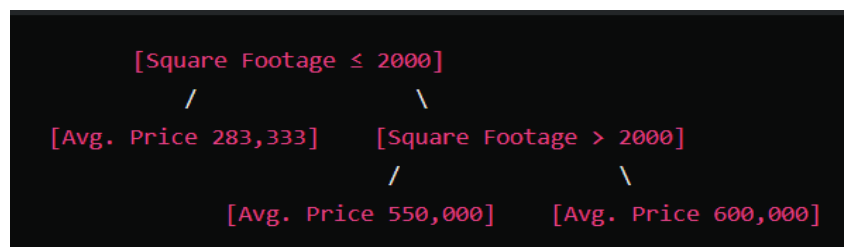
We compare this overall MSE with the initial MSE (206,000,000). Since this is a significant reduction, we make the split.

Step 6: Recursive Splitting

Next, we recursively apply the same process to the left and right subsets. For instance, we can further split the left subset based on the number of bedrooms or additional splits based on square footage.

Step 7: Final Tree Structure

After multiple rounds of splitting, you might end up with a tree like this:



Step 8: Making Predictions

To predict the price of a new house, say with 1700 square feet and 3 bedrooms:

1. Start at the root: $1700 \leq 2000$.
2. Go to the left leaf, which contains houses ≤ 2000 square feet.
3. The predicted price for this house is approximately **\$283,333**.

All the 4 models use decision trees to work but the difference occurs in how they build the decision tree

1. *Random forest* - Trees are grown **depth-wise**, meaning each tree is grown to its full depth without pruning.
2. *Xgboost* - Builds trees **sequentially** using **level-wise** growth (all nodes at a certain level are split before moving to the next level).

3. *LightGBM* - Builds trees **leaf-wise**, meaning it splits the leaf with the highest potential to reduce the loss.
4. *CatBoost* - Uses **symmetric trees** (all nodes at the same depth are split in the same way), which ensures balanced trees.

DATASET DESCRIPTION

The dataset used in this project consists of 123418 instances. This includes a range of features that are likely useful for age prediction, including physical measurements and weight-related metrics.

Dataset Features:

1. **id**: Unique identifier for each crab.
2. **Sex**: Sex of the crab (M = Male, F = Female, I = Immature).
3. **Length**: Length of the crab (in cm).
4. **Diameter**: Diameter of the crab (in cm).
5. **Height**: Height of the crab (in cm).
6. **Weight**: Total weight of the crab (in grams).
7. **Shucked Weight**: Weight of the crab meat (in grams).
8. **Viscera Weight**: Weight of the crab's internal organs (in grams).
9. **Shell Weight**: Weight of the crab's shell (in grams).
10. **Age**: Age of the crab (in years) - the target variable for prediction.

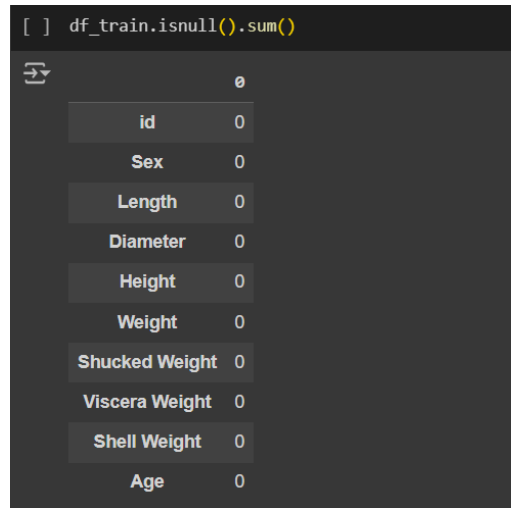
Data Sample

id	Sex	Length	Diameter	Height	Weight	Shucked W	Viscera W	Shell Weig	Age
0	I	1.525	1.175	0.375	28.97319	12.72893	6.647958	8.348928	9
1	I	1.1	0.825	0.275	10.41844	4.521745	2.324659	3.40194	8
2	M	1.3875	1.1125	0.375	24.77746	11.3398	5.556502	6.662133	9
3	F	1.7	1.4125	0.5	50.66056	20.35494	10.99184	14.99689	11
4	I	1.25	1.0125	0.3375	23.28911	11.97766	4.507571	5.953395	8
5	M	1.5	1.175	0.4125	28.84562	13.40931	6.789705	7.93786	10
6	M	1.575	1.1375	0.35	30.02212	11.93514	7.342521	8.646598	11
7	I	1.3125	1.025	0.35	18.2996	8.249705	3.898056	5.6699	11

- **Preprocessing:**

- *Handling missing values:* No missing values were found in the dataset.

```
[ ] df_train.isnull().sum()
```



	0
id	0
Sex	0
Length	0
Diameter	0
Height	0
Weight	0
Shucked Weight	0
Viscera Weight	0
Shell Weight	0
Age	0

- *encoding categorical data:* Sex variable was transformed using label encoder.
- *outlier detection:* Using Z-Score method to detect and remove outliers.

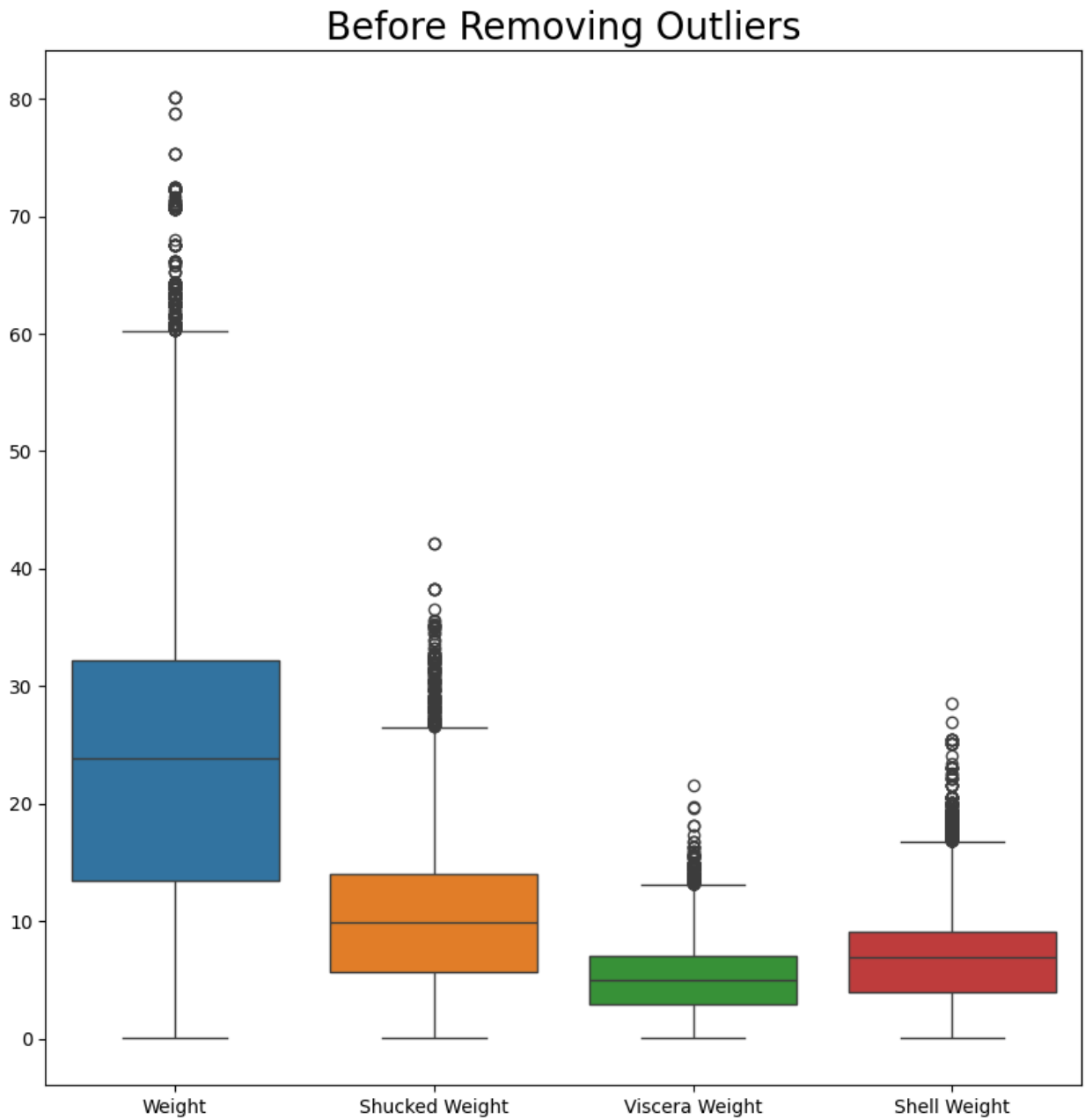
The formula for calculating the Z-score of a data point x is:

$$Z = \frac{(x - \mu)}{\sigma}$$

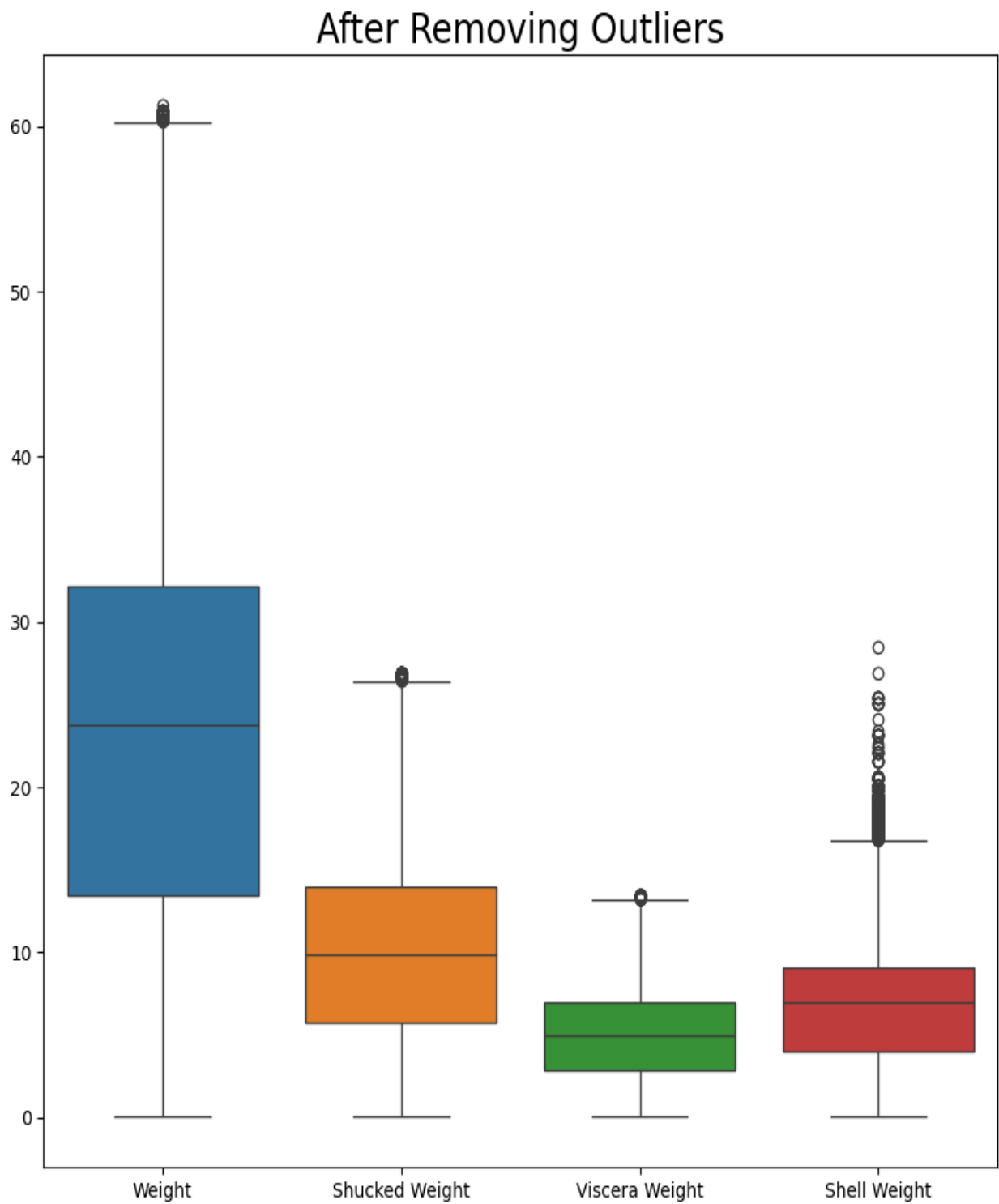
Typically, a Z-score greater than 3 or less than -3 indicates that the data point is an outlier, as it falls far from the average. This method works well for normally distributed data but may not be as effective for skewed distributions.

OUTLIER DETECTION

Before removing outliers



After Removing Outliers



Methodology

Experimental Design:

- *Preprocessing*: Data cleaning, encoding, and scaling.
- *Model Training*: Split dataset into training and testing sets and train models using default parameters.
- *Model Evaluation*: Use MAE, MSE, and R^2 Score to assess model performance
- *Optimization*: Optimizing the used models to get the best parameters.

Environment and Tools: Python, Scikit-learn, XGBoost, CatBoost, LightGBM and Google Colab.

OPTIMIZATION ALGORITHM USED:

Bayesian Optimization: It is a technique used to optimize hyperparameters of machine learning models efficiently, especially when the function being optimized is expensive to evaluate (e.g., deep learning models). It builds a probabilistic model of the objective function and uses this model to select the most promising hyperparameters to try next.

Working:

The working of Bayesian Optimization can be summarized in these steps:

Initialize: Evaluate the objective function with a few initial random hyperparameters.

Model: Fit the surrogate model to approximate the objective function.

Optimize: Use the acquisition function to choose new hyperparameters.

Iterate: Evaluate the objective function at the new hyperparameters, update the surrogate model, and repeat.

Best parameters of each model after optimization:

MODEL	BEST PARAMETERS
Linear regression	<code>{'fit_intercept': True}</code>
Random forest Regression	<code>{'bootstrap': 1.0, 'max_depth': 11.02334829439182, 'min_samples_leaf': 4.0, 'min_samples_split': 9.193158680091239, 'n_estimators': 125.84379466965744}</code>
Xgboost Regressor	<code>{'colsample_bytree': 0.6833149296885029, 'learning_rate': 0.03880498039109571, 'max_depth': 7.748953326138044, 'n_estimators': 244.53928595287186, 'subsample': 0.8879515790869158}</code>
LGBM Regressor	<code>{'learning_rate': 0.04240341035887028, 'max_depth': 6.346345559112615, 'n_estimators': 272.5998034488772, 'num_leaves': 50.983786351441836}</code>
Cat boost Regressor	<code>{'depth': 6.2472407130841745, 'iterations': 480.2857225639665, 'l2_leaf_reg': 3.9279757672456204, 'learning_rate': 0.12374511199743694}</code>
Ridge regressor	<code>{'alpha': 1}</code>
Lasso Regressor	<code>{'alpha': 0.01}</code>

Results

Evaluation: Summary of model performance:

- *MSE*: Mean Squared Error, lower is better.
- *MAE*: Mean Absolute Error, indicates prediction accuracy.
- *R² Score*: Represents model fit, values closer to 1 indicate better performance.

The performance of each model was evaluated using MAE, MSE and R2_Score. The results are summarized in the table below:

BEFORE OPTIMIZATION:

MODEL	MSE	MAE	R2_Score
Linear Regressor	4.876540	1.532873	0.521512
Random Forest Regressor	4.400940	1.465116	0.568178
XGB Regressor	4.258938	1.420414	0.58211
LGBM Regressor	4.175620	1.411501	0.590287
CatBoost Regressor	4.135949	1.406607	0.594179
Ridge Regressor	4.910088	1.531527	0.5182204
Lasso Regressor	5.00919	1.546218	0.508496

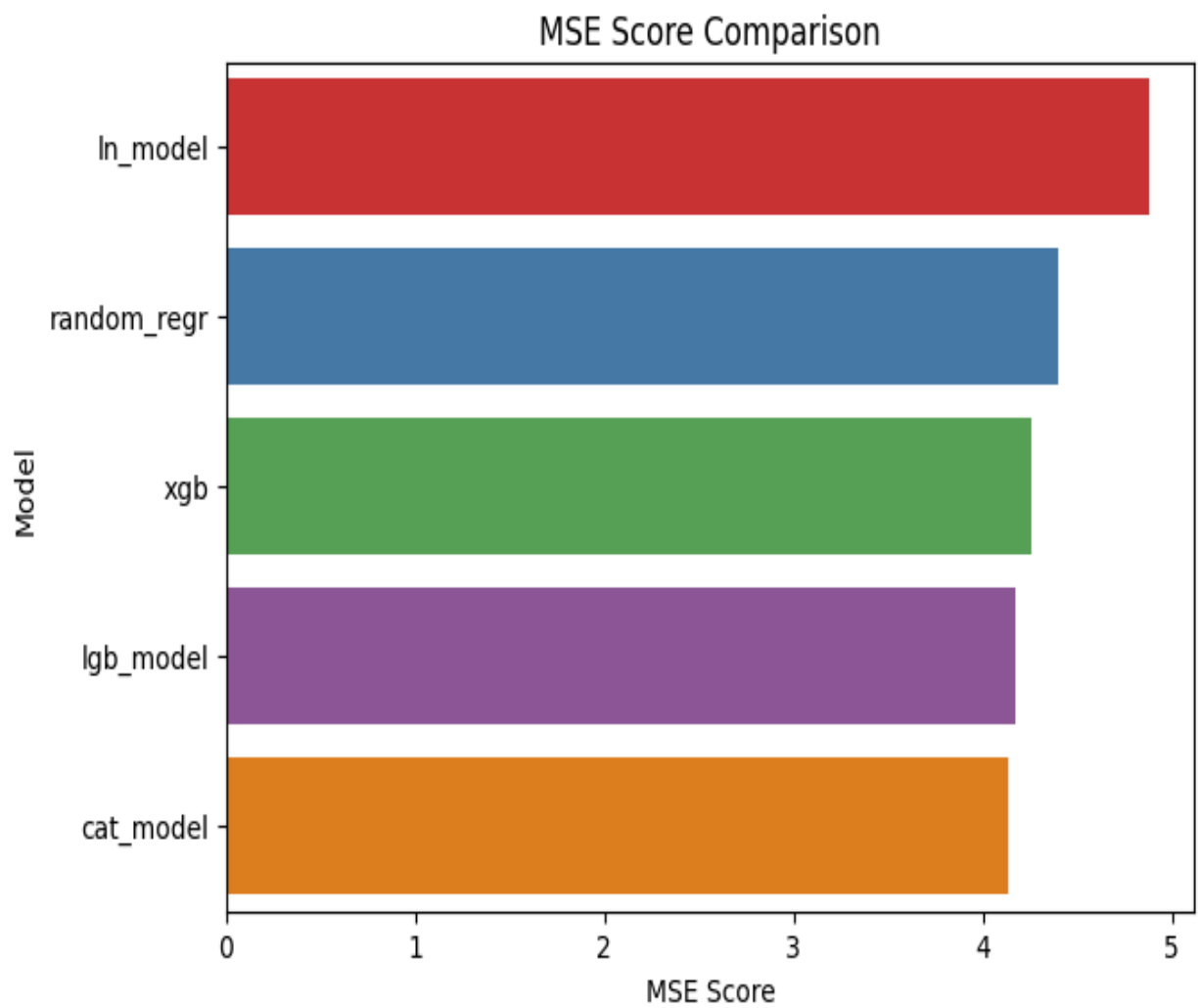
AFTER OPTIMIZATION:

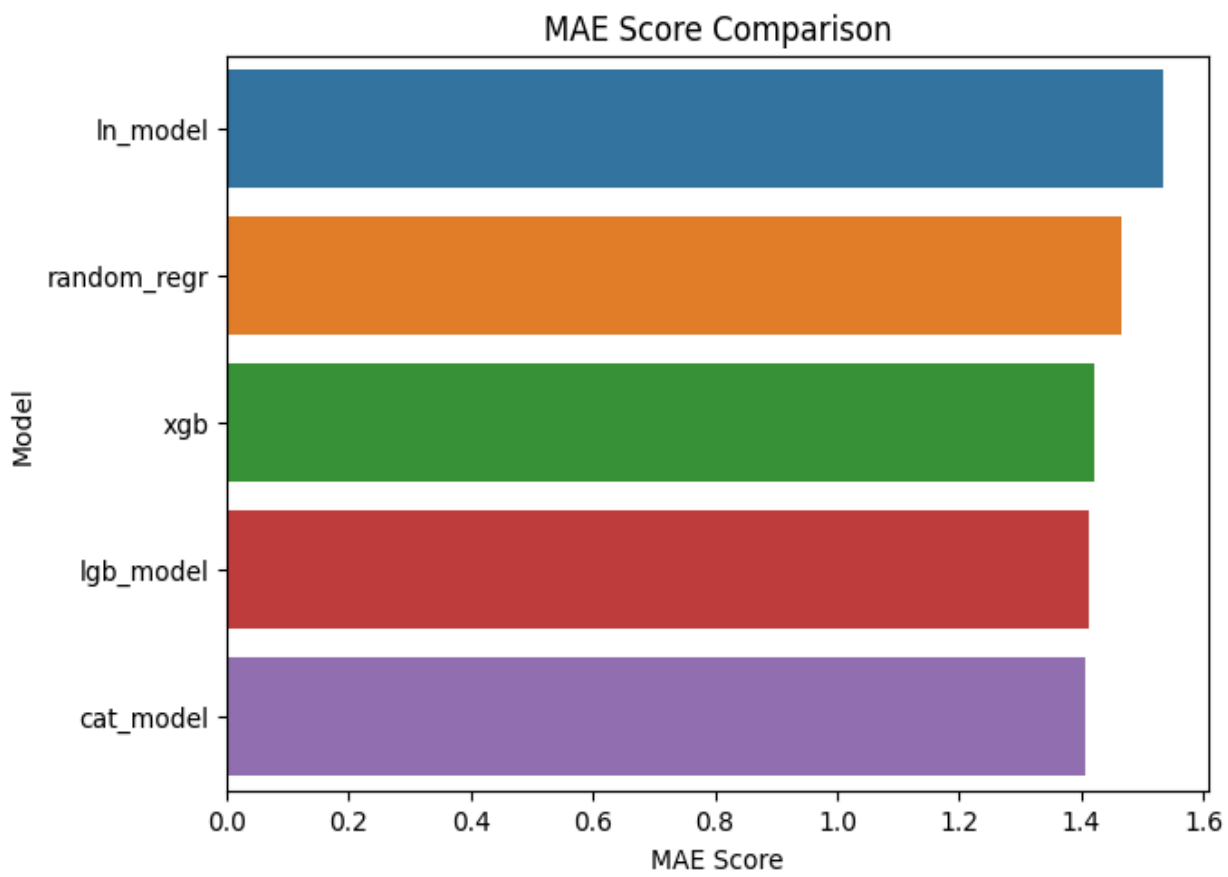
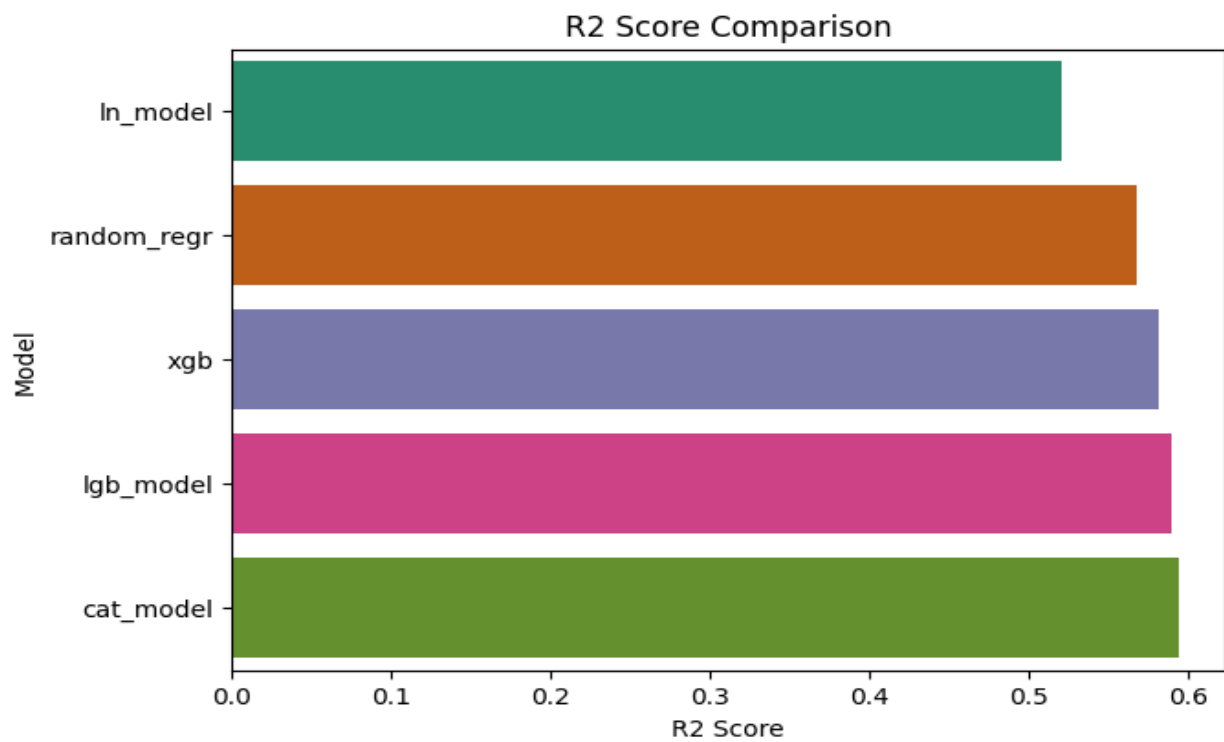
MODEL	MSE	MAE	R2_Score
Linear Regressor	4.876540	1.532873	0.521512
Random Forest Regressor	4.209490	1.414101	0.568963
XGB Regressor	4.139684	1.404533	0.593813
LGBM Regressor	4.137876	1.404239	0.593990
CatBoost Regressor	4.143671	1.406943	0.593422
Ridge Regressor	4.910088	1.531527	0.5182204
Lasso Regressor	5.00919	1.546218	0.508496

Results Difference:

MODEL	MSE	MAE	R2_Score
Linear Regressor	0	0	0
Random Forest Regressor	0.191450	0.051015	- 0.018785
XGB Regressor	0.119254	0.015881	-0.011702
LGBM Regressor	0.037744	0.007262	-0.003703
CatBoost Regressor	-0.007722	-0.000336	0.000757

Result Visualization:





Discussion

- *Overall Results*: Ensemble methods outperformed simpler models like linear regression.
- *Overfitting/Underfitting*: No significant overfitting was observed with the ensemble models.
- *Hyperparameter Tuning*: After Hyperparameter tuning there was a slight increase in performance of some models.
- *Model Comparison*: CatBoost and LightGBM were superior in handling complex patterns in the data

Learning Outcome

1. Skills and Techniques:

- *Regression Analysis*: Gained experience in applying various regression techniques like Linear Regression, RandomForestRegressor, XGBRegressor, CatBoostRegressor, and LightGBM.
- *Data Preprocessing*: Enhanced skills in handling missing data, feature scaling, and encoding categorical variables to prepare the dataset for machine learning models.
- *Outlier Detection*: Applied outlier detection methods such as Z-score analysis to clean and optimize the dataset.
- *Model Evaluation*: Learned to evaluate model performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 Score.

2. Tools and Libraries:

- *Python Libraries*: Improved proficiency in Python using popular libraries like Scikit-learn, XGBoost, CatBoost, and LightGBM.

- *Proficiency in colab*: improve understanding of how to work using Google colab.

3.Ensemble Learning:

- Gained insights into the power of ensemble methods like Random Forest and gradient boosting algorithms (CatBoost, LightGBM), which provided better accuracy and generalization than simpler models like Linear Regression.

4.Model Optimization:

- Understood the importance of tuning hyperparameters in ensemble methods to further improve model performance, and how default parameters can already yield robust results.

5.Challenges Encountered:

- Worked on handling multicollinearity, outliers, and feature engineering, learning how these factors impact model accuracy and generalization.

Conclusion

Summary: The project successfully predicted the age of crabs using various machine learning models, including Linear Regression, RandomForestRegressor, XGBRegressor, CatBoostRegressor, and LightGBM. Among these, **CatBoostRegressor** delivered the best performance, achieving the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE), along with the highest R^2 Score, making it the most accurate model for this task. The ensemble methods demonstrated superior accuracy compared to simpler models, effectively handling complex relationships between the crab's physical characteristics and their age.

Accomplishments:

- The primary goal of the project—predicting crab age with a high degree of accuracy—was achieved using multiple machine learning models.
- CatBoostRegressor emerged as the top-performing model, followed closely by LightGBM, both of which utilized advanced ensemble techniques to produce highly accurate predictions.
- The project successfully demonstrated the capability of machine learning to solve real-world problems, such as predicting biological variables, with practical applications in industries like fishing and seafood quality control.
- Various preprocessing techniques, model evaluation metrics, and performance comparisons were applied to ensure a thorough analysis of the models used.

Advantages/Limitations:

- **Advantages:** Ensemble models like CatBoostRegressor and LightGBM provided highly accurate predictions, handling complex data relationships efficiently, especially in cases with multiple interrelated features. These models excelled in generalization and reduced the risk of overfitting.
- **Limitations:** The ensemble methods, while powerful, were computationally more intensive and required more memory and processing power compared to simpler models like Linear Regression. Additionally, hyperparameter tuning and model optimization were not fully explored, leaving potential for further improvement.

LINK TO CODE :

<https://colab.research.google.com/drive/10a2fb3RWALR5uENnLG7RXIbcGpX47KiE?usp=sharing>