Labwork 2

Le Linh Long - 22BI13262

I. Introduction

Ultrasound is a non-invasive imaging test that shows structures inside your body using high-intensity sound waves. Healthcare providers use ultrasound exams for several purposes, including during pregnancy, to diagnose conditions, and to guide images during certain procedures.

II. Exploratory Data Analysis

This data contains two features: pixel size (mm) and head circumference (mm), along with 1,000 rows of data. In addition, there are two image folders: Image and Ground Truth, but in this task, I will focus only on pixel size and head circumference.

Head circumference represents the measurement of the perimeter of the fetal head in millimeters. Meanwhile, pixel size refers to the physical dimension of each pixel in millimeters.

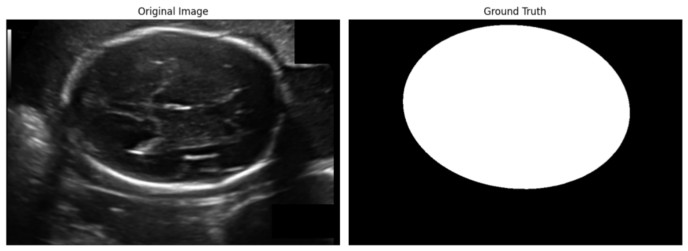

Fig. 1: Sample Ultrasound Image

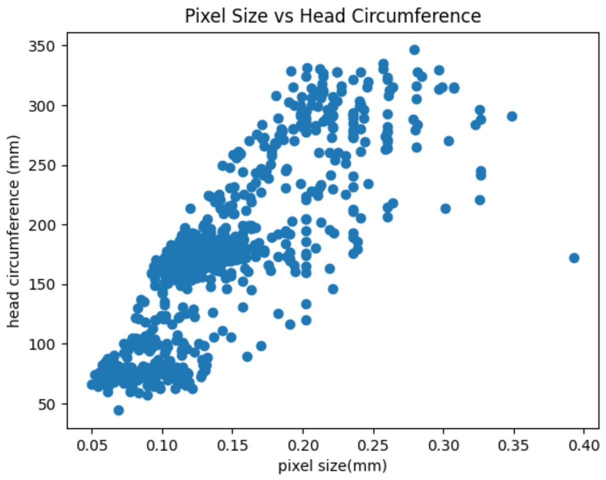

Fig. 2: Data Distribution

III. Model Selection

In this task, I used LazyRegressor, a Python library that compares multiple regression models to identify the top two models for further hyperparameter tuning.

The table below presents the results of 41 different models, all evaluated using their default parameters.

TABLE I: Regression Model Performance Comparison

Model	**Adjusted R-Squared**	**R-Squared**	**RMSE**	**Time Taken (s)**	**Mean Absolute Error**
HistGradientBoostingRegressor	0.69	0.70	35.05	0.16	24.33
GradientBoostingRegressor	0.68	0.68	35.78	0.07	24.67
LGBMRegressor	0.67	0.67	36.56	0.09	25.64
KNeighborsRegressor	0.65	0.65	37.50	0.01	25.56
XGBRegressor	0.65	0.65	37.52	1.87	26.19
AdaBoostRegressor	0.64	0.64	37.90	0.04	30.88
NuSVR	0.60	0.60	39.97	0.03	30.84
SVR	0.60	0.60	40.09	0.03	28.66
PassiveAggressiveRegressor	0.60	0.60	40.16	0.01	29.16
ElasticNetCV	0.59	0.59	40.50	0.06	29.45
Lasso	0.59	0.59	40.80	0.02	29.58
LassoLars	0.59	0.59	40.80	0.00	29.58
SGDRegressor	0.58	0.58	40.98	0.02	29.68
RidgeCV	0.58	0.58	40.99	0.00	29.66
Ridge	0.58	0.58	40.99	0.00	29.66
LassoCV	0.58	0.58	41.00	0.05	29.66
BayesianRidge	0.58	0.58	41.00	0.01	29.67
LinearRegression	0.58	0.58	41.01	0.00	29.67
TransformedTargetRegressor	0.58	0.58	41.01	0.02	29.67
Lars	0.58	0.58	41.01	0.00	29.67
LarsCV	0.58	0.58	41.01	0.00	29.67
LassoLarsCV	0.58	0.58	41.01	0.01	29.67
LassoLarsIC	0.58	0.58	41.01	0.01	29.67
OrthogonalMatchingPursuit	0.58	0.58	41.01	0.01	29.67
LinearSVR	0.57	0.57	41.63	0.00	30.62
ElasticNet	0.57	0.57	41.67	0.00	30.64
RandomForestRegressor	0.56	0.57	41.86	0.15	29.04
HuberRegressor	0.55	0.55	42.40	0.00	29.81
BaggingRegressor	0.54	0.54	43.15	0.03	30.25
ExtraTreesRegressor	0.52	0.52	44.06	0.12	29.63
TweedieRegressor	0.49	0.50	45.07	0.01	32.88
GammaRegressor	0.46	0.46	46.48	0.12	34.83
ExtraTreeRegressor	0.44	0.44	47.62	0.01	31.94
DecisionTreeRegressor	0.42	0.42	48.39	0.00	32.41
PoissonRegressor	0.34	0.34	51.59	0.01	34.53
RANSACRegressor	0.20	0.21	56.51	0.04	39.79
QuantileRegressor	-0.01	-0.00	63.63	0.04	44.26
DummyRegressor	-0.01	-0.01	63.68	0.01	44.24
MLPRegressor	-1.30	-1.29	96.16	0.48	84.83
KernelRidge	-6.92	-6.88	178.30	0.10	174.43
GaussianProcessRegressor	-238.90	-237.69	981.28	0.32	93.52

As observed in the table I, the two best-performing models are HistGradientBoostingRegressor and GradientBoostingRegressor.

IV. Hyperparameters Tuning

A. *HistGradientBoostingRegressor*

To find the optimal hyperparameters for HistGradientBoostingRegressor, I performed a grid search over the following parameters:

- **Learning Rate:** {0.01, 0.1, 0.2}
- **Max Iterations:** {100, 150, 200}
- **Min Samples per Leaf:** {10, 15, 20}
- **L2 Regularization:** {0, 0.1, 0.001}
- **Max Leaf Nodes:** {11, 21, 31}
- **Warm Start:** {True, False}
- **Loss Function:** absolute_error

B. *GradientBoostingRegressor*

Similarly, for GradientBoostingRegressor, I explored the following parameters:

- **Learning Rate:** {0.01, 0.1}
- **Number of Estimators:** {100, 150}
- **Min Samples per Leaf:** {1, 10}
- **Loss Function:** absolute_error
- **Warm Start:** {True, False}
- **Max Leaf Nodes:** {None, 10}
- **Early Stopping (n_iter_no_change):** 10

V. Results

For HistGradientBoostingRegressor (HGBR), the best parameters are a learning rate of 0.1, 100 boosting iterations with an L2 regularization value of 0.1, a maximum of 21 leaf nodes, and a minimum of 20 samples per leaf. Furthermore, enabling warm start improved performance.

For GradientBoostingRegressor (GBR), the best parameters are a learning rate of 0.1, 100 estimators, a maximum of 10 leaf nodes, a minimum of 10 samples per leaf, and a set of early stoppings at 10 iterations. Similar to HGBR, enabling warm start proved beneficial.

TABLE II: Metrics Comparison of HGBR and GBR

Metric	**HGBR**	**GBR**
Mean Absolute Error (MAE)	**19.192**	22.186
Mean Squared Error (MSE)	**1174.87**	1233.42
Root Mean Squared Error (RMSE)	**34.28**	35.12
R^2 Score	0.693	**0.694**
Mean Absolute Percentage Error (MAPE)	**15.88%**	16.15%

Table II summarizes the performance metrics for both models. While GradientBoostingRegressor achieved a slightly higher R^2 score, HistGradientBoostingRegressor showed lower MAE, MSE, RMSE and MAPE, proving a better overall predictive accuracy.

VI. Conclusion

In this task, I used LazyRegressor and GridSearchCV to find and optimize two best performance regression models: HistGradientBoostingRegressor and GradientBoostingRegressor, with a decent Mean Absolute Error and R^2 score achieved.

Future work includes exploring deep learning models to further reduce error and improve overall predictive accuracy.