Machine Learning in Computational Biology
Assignment 1
Bonus Questions

Konstantinos Konstantinidis
Student number: 7115152400017

April 7, 2025

Repo: The repository for this assignment can be found here:
`https://github.com/KonsKons26/Assignment-1`

Contents

1 First bonus task: Using Optuna for hyperparameter tuning

1.1 Migration to Optuna

Since the `Regressor` object I had created to perform all regression tasks was highly modularized, with specific methods for each task, moving from `sklearn`'s `GridSearchCV` to `Optuna` was quite simple. I created a new class named `RegressorOptuna` which inherited from `Regressor`, to keep it's basic functionalities the same. Then I modified the private methods handling each model's hyperparameter optimization to allow me to pass the appropriate grid spaces –and in the appropriate format– for `Optuna` to tune the models.

The objective function **minimizes** the RMSE of the test set. The sampler I chose is the `TPESampler`, a tree-structured Parzen estimator, which fits one Gaussian Mixture Model (GMM) (lx) to the set of parameters associated with the best objective values and another GMM ($g(x)$) to the remaining parameter values. It chooses the parameter that minimizes the ration $l(x)/g(x)$.

All files are in the appropriate directories, `src/` and `notebooks/` and the models were saved in `models/bonus1_optuna/`. Since I used the features I selected in the previous task, to keep everything organized, the process includes copying the features files in the `models/bonus1_optuna` directory.

1.2 Hyperparameter tuning

I set the number of trials for each model tuning to 1000, which seemed appropriate, as it was not such a demanding number for my machine, and since the hyperparameter space has so many dimension, I believe that a small number of trials might not be enough to search the whole space. The complete hyperparameter space, along with the values chosen by `Optuna` are shown in Table 1.

Model	Parameter	Values	Picked value
ElasticNet	α	(0.01, 1.0)	0.9998
	`l1 ratio`	(0.0, 1.0)	0.00004
	`tolerance`	[1e-3, 1e-4, 1e-5, 1e-6, 1e-7]	0.0001
SVR	`kernel`	['rbf', 'linear', 'poly', 'sigmoid']	'rbf'
	`degree`	(2, 5)	2
	γ	['scale', 'auto']	'auto'
	`coef_0`	(0.0, 1)	0.46631
	`tolerance`	[1e-3, 1e-4, 1e-5, 1e-6, 1e-7]	1e-6
	`C`	(0.1, 10)	1.49583
	ϵ	(0.0, 10.0)	0.10783
BayesianRidge	`tolerance`	[1e-3, 1e-4, 1e-5, 1e-6, 1e-7]	1e-3
	α_1	(1e-9, 1e-3)	0.00001
	α_2	(1e-9, 1e-3)	0.00001
	λ_1	(1e-3, 1e-9)	0.00001
	λ_2	(1e-3, 1e-9)	0.001
	`compute_score`	[True, False]	True

Table 1: Hyperparameter spaces for each model.

1.3 Results

After tuning, the best models were saved (along with their scalers, like in the previous task) and then the validation set was used to measure their performance. For testing, the same approach with resampling was followed, for 1000 repeats. The resulting boxplots are shown in Figure 1.

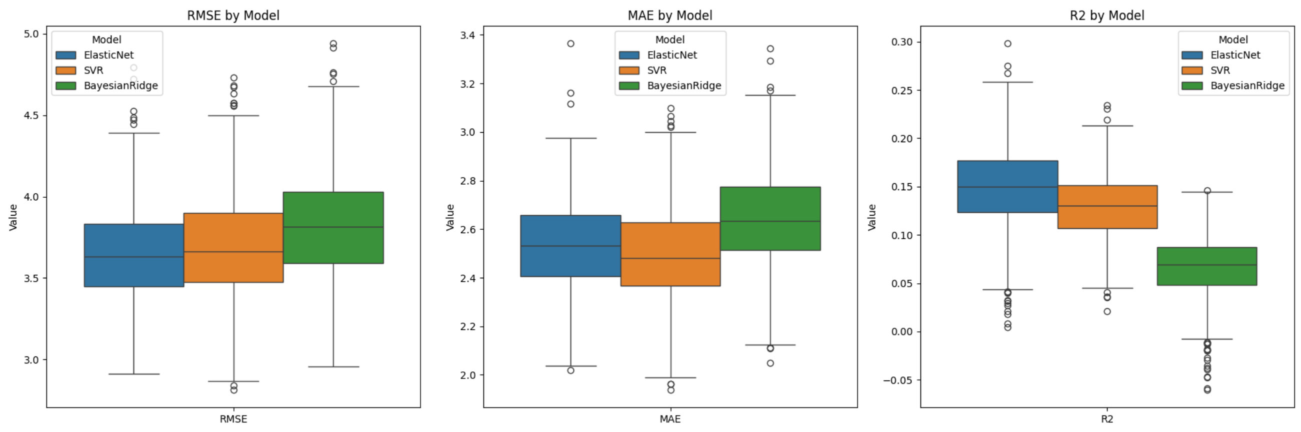


Figure 1: Test metrics after tuning with `Optuna`. ElasticNet in blue, SVR in orange, and Bayesian Ridge in green.

Similar to what was shown in the previous task, Elastic Net is the best performing model, but its RMSE and MAE still fail to drop below 3, while r^2 fails to overcome 0.2. These findings further increase my confidence that this dataset can not be used for regression tasks, at least no with the models that we analyzed.

2 Second bonus task: Converting the problem to a binary classification task

2.1 Overview

Converting the target labels (BMI) to binary labels was quite simple. I used the custom threshold of 25 to separate the labels into two classes, 'overweight' and 'not overweight'. This change happens in-place without the need to store additional files. The structure of the code base is almost identical to that of the first task, with the exception of the models (and their corresponding hyperparameter spaces) used to classify the data. Also, I opted to using `Optuna` for the hyperparameter tuning of the models, instead of `GridSearchCV`.

The way features were selected was the same as in the first task, using the method that performed the best in classifying the data, based on the following scoring function (Equation 1) which in essence averages out the best accuracy, precision, recall, and f1 metric.

$$\text{score} = \frac{(\text{mean}(\mathit{accuracy}) + \text{mean}(\mathit{precision}) + \text{mean}(\mathit{recall}) + \text{mean}(f1))}{4} \tag{1}$$

For testing the models, the validation set was used with resampling for 1000 iterations. Unlike the previous task which involved regression, this task involves classification, so different metrics must be used. I chose the following metrics (as provided by `sklearn`):

- `accuracy`: The percentage of correct classifications.
- `balanced_accuracy`: Same as accuracy but takes into account the prevalence of each class.
- `precision`: The ratio of true positives over all positives.
- `recall`: The ratio of true positives over true positives and false negatives.
- `f1_score`: The F1 score, can be interpreted as the harmonic mean of the precision and recall.
- `roc_auc`: Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
- `average_precision`: summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight.
- `confusion_matrix`: A onfusion matrix.
- `matthews_corrcoef`: It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes.

2.2 Feature Selection

Figure 2 shows the features selected for each model. The `VarianceThreshold` method was used for both of them, with a threshold of 0.5. We can see that both models were optimized for exactly the same features.

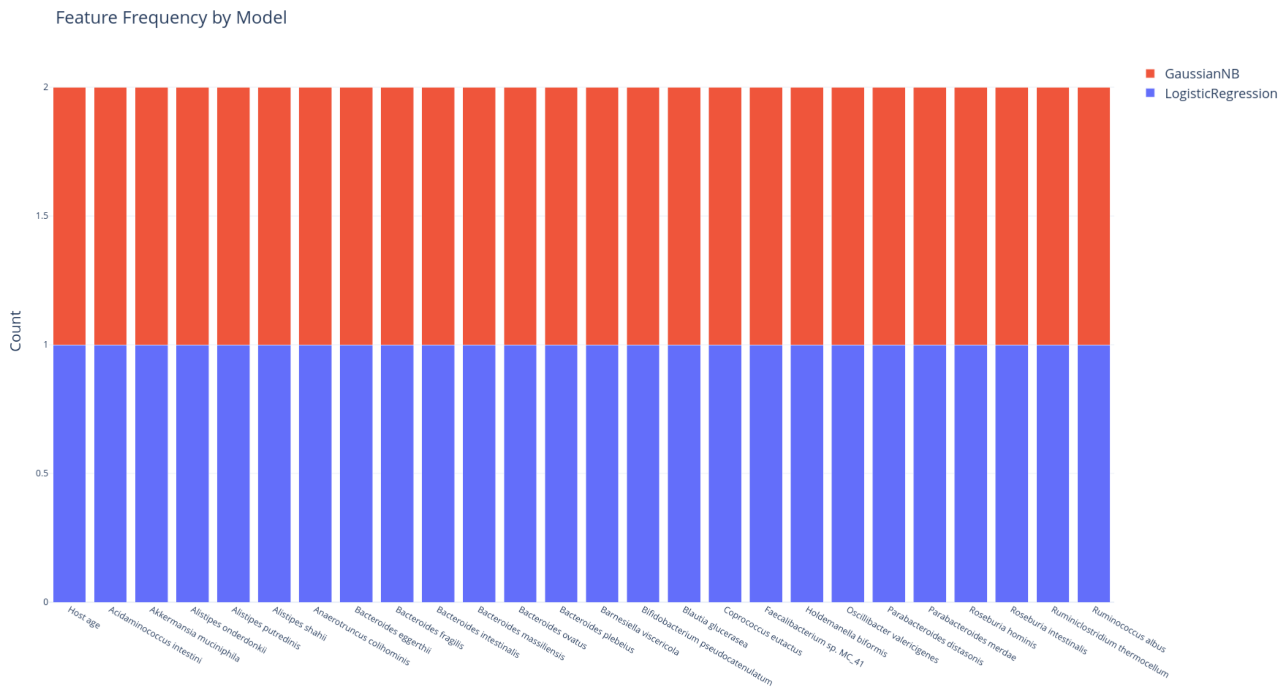


Figure 2: Selected features.

2.3 Tuning

Hyperparameter tuning was done using `Optuna` with cross validation and at 100 trials. Table 2 summarizes the hyperparameter grids passed in and the seelcted values by the model. We can see that for the Logistic Regression model, the penalty is almost exclusievly L2, the tolerance is small, and the intercept is fitted. The variance smoothing for the GNB stays at 1e-6 which is the default of the model.

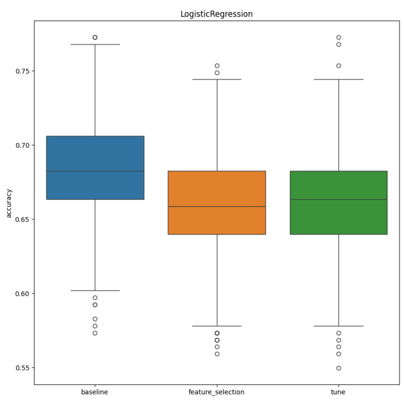
Model	Parameter	Values	Picked value
Logistic Regression	`penalty`	"elasticnet"	"elasticnet"
	`l1_ratio`	(0.0, 1.0)	0.0104
	`tolerance`	(1e-7, 1e-1)	0.02814
	`C`	(0.001, 1000)	0.02330
	`fit_intercept`	[True, False]	True
	`solver`	"saga"	"saga"
	`max_iter`	(1000000, 10000000)	5313982
Gaussian Naive Bayes	`var_smoothing`	(1e-9, 1e-1)	1e-6

Table 2: Hyperparameter spaces for each model.

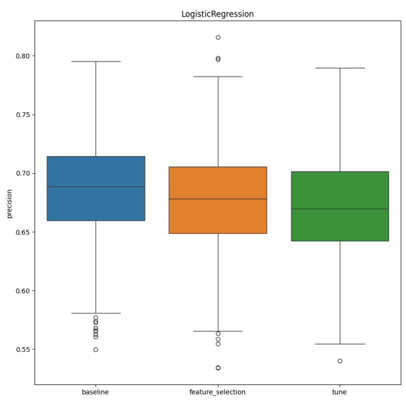
2.4 Results

All models (baseline, feature selection, and tuned) are tested using the validation set. Some of their metrics are presented here (*all plots are presented in the notebooks*).

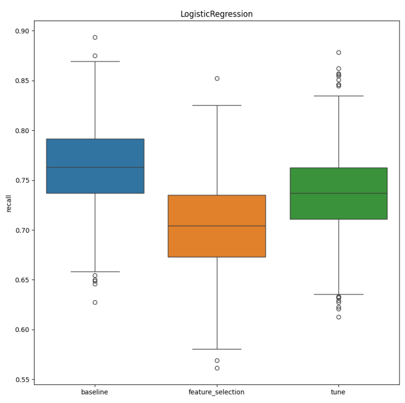
The fact that the baseline models are performing better is a strong indication that something is not working as expected. One issue might be with how I performed hyperparameter tuning. For each trial I let the objective function internally perform cross validation and return the mean of the scores, which might be counterproductive for the `Optuna` optimizer, a possible solution could be to perform corss validation outside the hyperparameter optimization loop; a possible improvement would be to implement that and check if the scores improved.



(a) Accuracy

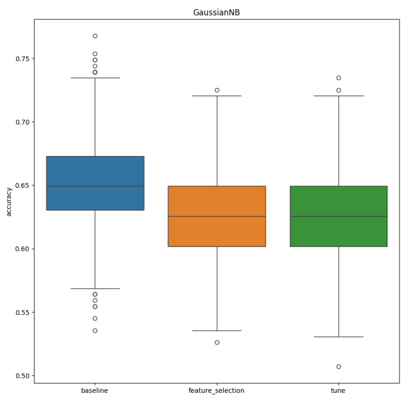


(b) Precision

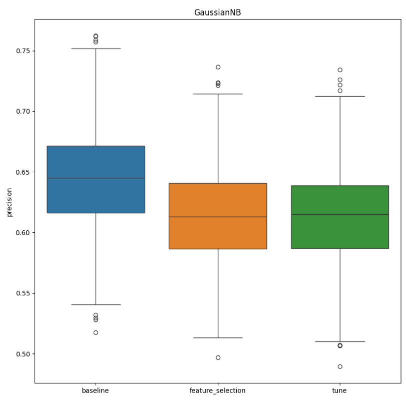


(c) Recall

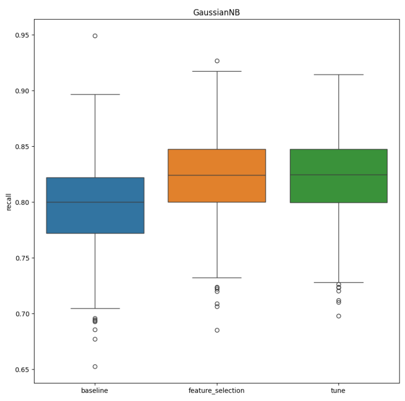
Figure 3: Logistic Regression: Results based on the validation set. Accuracy, precision, and recall for the baseline, feature selection, and tuned models (baseline in blue, feature selection in orange, and tuned in green).



(a) Accuracy



(b) Precision



(c) Recall

Figure 4: Gaussian Naive Bayes: Results based on the validation set. Accuracy, precision, and recall for the baseline, feature selection, and tuned models (baseline in blue, feature selection in orange, and tuned in green).