Table of Contents

List of Tables

Project Overview

The project 4 extensively focusses on ensemble methods in Machine Learning and aims to expand the prescribed methods by completing the objective necessary. Furthermore, it also expects the comparison from within these methods to establish overall understanding of the best technique through the adjustment of hyperparameters.

Problem Statement 1

Question

The expectation is to shuffle and split the Wisconsin Breast Cancer Dataset into Training/Validation (80%) and Testing (20%). This Training/Validation is further split into Training and Validation separately. A random forest classifier model is to be built to train, validate and test upon the chosen dataset. The list of hyperparameters and its altercations leading the changes in the results is expected to be documented. Along with a list of all examples where prediction has been incorrect.

Code

THE ENTIRE CODE BLOCK FOR THE PROBLEM STATEMENT 1 CAN BE FOUND IN THE APPENDIX WITHIN THE CODE 1 OF THIS DOCUMENT.

Design Process (for both Random Forest Classifier and Gradient Boosting Classifier)

1. Dataset selection and Preprocessing:

In this stage, the dataset is chosen as per the requirement stated (here, Wisconsin Breast Cancer Dataset) and the dataset is shuffled and set ready for it to be split.

2. Data Splitting:

The dataset is split first into Training/Validation and Testing as per the specified requirements and ratios (here, 80% : 20%). The dataset is further split in a certain ratio between the Training and Validation sets.

3. Model Selection:

This stage is straightforward as the problem statement clearly defines which model is to be used to perform operations on the datasets. Here, the Random Forest Classifier is used.

4. Model Training and Evaluation:

The instance of the model is created, and the dataset is initialized and fit for the model from the training subset. This model is then tested on multiple factors such as accuracy score, precision score, recall score and the f1 score upon the validation subset.

5. Model Testing and Error Analysis:

The testing subset is evaluated for the performance of the model and an accuracy score is generated upon the testing subset. This score will define the model's effectiveness from its learning and validation stages. In conclusion, it is essential to also mention the list of failure cases to learn from and improve the model.

Tradeoffs considered (for both Random Forest Classifier and Gradient Boosting Classifier)

1. Model complexity and interpretability:

The ensemble method of Random Forest Classifier used is a very complex model. The interpretability of this model becomes a crucial task considering that medical data is used, and key insights are an expected output from this model.

2. Bias-Variance Tradeoff:

The data used in this problem statement has limited records (in comparison to the real-time data available in the medical industry). This can lead to overfitting or underfitting of the data resulting in deviation from expected results.

3. Computational Efficiency and Model Performance:

The model's capability and prowess can be experimented only with large real-time datasets which in turn demand equivalency in infrastructure and resource requirement which in certain cases is a tradeoff.

Output

```
The Total number of samples in the Wisconsin Breast Cancer Dataset is:

569
The number of samples in the Training Dataset is:

227
The number of samples in the Validation Dataset is:

228
The number of samples in the Testing Dataset is:

114
Accuracy on the Validation set: 0.9517543859649122
Precision on the Validation set: 0.9370629370629371
Recall on the Validation set: 0.9852941176470589
F1 Score on the Validation set: 0.960573476702509

Accuracy on the Test Set: 0.9824561403508771
Examples from the test set where the predictions have been incorrect:

Example: 1
Features: [1.785e+01 1.323e+01 1.146e+02 9.921e+02 7.838e-02 6.217e-02 4.445e-02
 4.178e-02 1.220e-01 5.243e-02 4.834e-01 1.046e+00 3.163e+00 5.095e+01
 4.369e-03 8.274e-03 1.153e-02 7.437e-03 1.302e-02 1.309e-03 1.982e+01
 1.842e+01 1.271e+02 1.210e+03 9.862e-02 9.976e-02 1.048e-01 8.341e-02
 1.783e-01 5.871e-02]
True Label: 1
Predicted Label: 0

Example: 62
Features: [1.234e+01 2.686e+01 8.115e+01 4.774e+02 1.034e-01 1.353e-
01 1.085e-01
 4.562e-02 1.943e-01 6.937e-02 4.053e-01 1.809e+00 2.642e+00
3.444e+01
 9.098e-03 3.845e-02 3.763e-02 1.321e-02 1.878e-02 5.672e-03
1.565e+01
 3.934e+01 1.017e+02 7.689e+02 1.785e-01 4.706e-01 4.425e-01 1.459e-
01
 3.215e-01 1.205e-01]
True Label: 0
Predicted Label: 1
```

The output clearly identifies the total number of samples in the Wisconsin Breast Cancer Dataset which is 569 and the number of samples divided for Training and Validation is 455 i.e. 80% (79.96%). Additionally, the Training and Validation is further split in half with 227 samples shared with Training and 228 samples shared with Validation. The number of samples divided for Testing is 114 i.e. 20% (20.03%).

The above output is obtained from the default set of hyperparameters used to construct the model and test on the dataset. It also has the list of incorrectly predicted examples from the test data.

Category	Percentage split	Number of rows (samples)
Training and Validation	80%	455 (227+228)
Training	50% of 80%	227
Validation	50% of 80%	228
Testing	20%	114

Table 1: Split of Data into Training, Validation and Testing with values.

The hyperparameters once altered deliver the following results:

Slno	Hyperparameters						Model Performance				
	No. of dataset features	Shuffling random state	Split #1 random state	Validation size	Split #2 random state	Model random state	Validation Accuracy	Validation Precision	Validation Recall	Validation f1 score	Test Accuracy
1	30	48	50	0.5	64	39	0.951	0.937	0.985	0.96	0.982
2	10	18	20	0.3	24	49	0.941	0.916	1	0.956	0.894
3	29	48	40	0.1	54	39	0.978	1	0.965	0.982	0.947
4	9	8	1	0.6	4	99	0.915	0.931	0.936	0.933	0.973
5	30	99	150	0.9	200	199	0.934	0.966	0.924	0.945	0.894

Table 2: Model's Performance affected by varying hyperparameters.

Error! Reference source not found. compares the model's performance to the varying hyperparameters. The row colored in grey is the default state of the model and in this case, it is the best performing scenario for the model with all the hyperparameter(s) altercations. The scenarios 2 and 5 marked in red symbolize the worst performing scenario(s) of the model.

Upon observation, it can be noticed that the model functions at an ideal and best state when the hyperparameters are spread across a reasonably middle spectrum and the model perform very poorly when the hyperparameters are either too low or too high.

However, this cannot be the only deciding factor after considering scenario 4 when the hyperparameters were all drafted to be even lower, and the model's performance wasn't the poorest. So, it can be estimated that the maximum weightage of hyperparameter altercation can be assigned to the Validation Size of the subset.

Problem Statement 2

Question

The expectation is to shuffle and split the Wisconsin Breast Cancer Dataset into Training/Validation (80%) and Testing (20%). This Training/Validation is further split into Training and Validation separately. A Gradient Boosting Learning classifier model is to be built to train, validate and test upon the chosen dataset. The list of hyperparameters and its altercations leading the changes in the results is expected to be documented. Along with a list of all examples where prediction has been incorrect.

Code

THE ENTIRE CODE BLOCK FOR THE PROBLEM STATEMENT 2 CAN BE FOUND IN THE APPENDIX WITHIN THE CODE 2 OF THIS DOCUMENT.

Output

```
The Total number of samples in the Wisconsin Breast Cancer Dataset is:

569
The number of samples in the Training Dataset is:

227
The number of samples in the Validation Dataset is:

228
The number of samples in the Testing Dataset is:

114
Accuracy on the Validation set: 0.9342105263157895
Precision on the Validation set: 0.9548872180451128
Recall on the Validation set: 0.9382352941176471
F1 Score on the Validation set: 0.9442379182156134

Accuracy on the Test Set: 0.9298245614035088
Examples from the test set where the predictions have been incorrect:

Example: 1
Features: [1.785e+01 1.323e+01 1.146e+02 9.921e+02 7.838e-02 6.217e-02 4.445e-02
 4.178e-02 1.220e-01 5.243e-02 4.834e-01 1.046e+00 3.163e+00 5.095e+01
 4.369e-03 8.274e-03 1.153e-02 7.437e-03 1.302e-02 1.309e-03 1.982e+01
 1.842e+01 1.271e+02 1.210e+03 9.862e-02 9.976e-02 1.048e-01 8.341e-02
 1.783e-01 5.871e-02]
True Label: 1
Predicted Label: 0

Example: 10
Features: [1.361e+01 2.498e+01 8.805e+01 5.827e+02 9.488e-02 8.511e-02 8.625e-02
 4.489e-02 1.609e-01 5.871e-02 4.565e-01 1.290e+00 2.861e+00 4.314e+01
 5.872e-03 1.488e-02 2.647e-02 9.921e-03 1.465e-02 2.355e-03 1.699e+01
 3.527e+01 1.086e+02 9.065e+02 1.265e-01 1.943e-01 3.169e-01 1.184e-01
 2.651e-01 7.397e-02]
True Label: 0
Predicted Label: 1

Example: 46
Features: [1.334e+01 1.586e+01 8.649e+01 5.200e+02 1.078e-01 1.535e-01 1.169e-01
 6.987e-02 1.942e-01 6.902e-02 2.860e-01 1.016e+00 1.535e+00 1.296e+01
 6.794e-03 3.575e-02 3.980e-02 1.383e-02 2.134e-02 4.603e-03 1.553e+01
 2.319e+01 9.666e+01 6.149e+02 1.536e-01 4.791e-01 4.858e-01 1.708e-01
 3.527e-01 1.016e-01]
True Label: 1
Predicted Label: 0

Example: 54
Features: [1.426e+01 1.965e+01 9.783e+01 6.299e+02 7.837e-02 2.233e-01 3.003e-01
 7.798e-02 1.704e-01 7.769e-02 3.628e-01 1.490e+00 3.399e+00 2.925e+01
 5.298e-03 7.446e-02 1.435e-01 2.292e-02 2.566e-02 1.298e-02 1.530e+01
 2.373e+01 1.070e+02 7.090e+02 8.949e-02 4.193e-01 6.783e-01 1.505e-01
 2.398e-01 1.082e-01]
True Label: 1
Predicted Label: 0

Example: 60
Features: [1.311e+01 2.254e+01 8.702e+01 5.294e+02 1.002e-01 1.483e-01 8.705e-02
 5.102e-02 1.850e-01 7.310e-02 1.931e-01 9.223e-01 1.491e+00 1.509e+01
 5.251e-03 3.041e-02 2.526e-02 8.304e-03 2.514e-02 4.198e-03 1.455e+01
 2.916e+01 9.948e+01 6.393e+02 1.349e-01 4.402e-01 3.162e-01 1.126e-01
 4.128e-01 1.076e-01]
True Label: 1
Predicted Label: 0

Example: 62
Features: [1.234e+01 2.686e+01 8.115e+01 4.774e+02 1.034e-01 1.353e-01 1.085e-01
 4.562e-02 1.943e-01 6.937e-02 4.053e-01 1.809e+00 2.642e+00 3.444e+01
 9.098e-03 3.845e-02 3.763e-02 1.321e-02 1.878e-02 5.672e-03 1.565e+01
 3.934e+01 1.017e+02 7.689e+02 1.785e-01 4.706e-01 4.425e-01 1.459e-01
 3.215e-01 1.205e-01]
True Label: 0
Predicted Label: 1

Example: 68
Features: [1.265e+01 1.817e+01 8.269e+01 4.856e+02 1.076e-01 1.334e-01 8.017e-02
 5.074e-02 1.641e-01 6.854e-02 2.324e-01 6.332e-01 1.696e+00 1.840e+01
 5.704e-03 2.502e-02 2.636e-02 1.032e-02 1.759e-02 3.563e-03 1.438e+01
 2.215e+01 9.529e+01 6.337e+02 1.533e-01 3.842e-01 3.582e-01 1.407e-01
 3.230e-01 1.033e-01]
True Label: 1
Predicted Label: 0

Example: 109
Features: [1.585e+01 2.395e+01 1.037e+02 7.827e+02 8.401e-02 1.002e-01 9.938e-02
 5.364e-02 1.847e-01 5.338e-02 4.033e-01 1.078e+00 2.903e+00 3.658e+01
 9.769e-03 3.126e-02 5.051e-02 1.992e-02 2.981e-02 3.002e-03 1.684e+01
 2.766e+01 1.120e+02 8.765e+02 1.131e-01 1.924e-01 2.322e-01 1.119e-01
 2.809e-01 6.287e-02]
True Label: 0
Predicted Label: 1
```

The above output has the list of incorrectly predicted examples from the test data.

The hyperparameters once altered deliver the following results:

Slno	Hyperparameters						Model Performance				
	No. of dataset features	Shuffling random state	Split #1 random state	Validation size	Split #2 random state	Model random state	Validation Accuracy	Validation Precision	Validation Recall	Validation f1 score	Test Accuracy
1	30	48	50	0.5	64	29	0.934	0.954	0.933	0.944	0.929
2	5	8	10	0.2	4	9	0.934	0.961	0.925	0.943	0.885
3	10	28	30	0.9	40	59	0.892	0.865	0.976	0.917	0.868
4	20	88	130	0.1	140	159	0.934	0.931	0.964	0.947	0.929
5	29	188	230	0.3	250	559	0.941	0.933	0.976	0.954	0.956

Table 3: Model's Performance based on varying Hyperparameters.

Table 3 compares the model's performance to the varying hyperparameters. The row colored in grey is the default state of the model. The scenario 5 in this case is the best performing scenario for the model with all the hyperparameter(s) altercations. The scenario 3 marked in red symbolize the worst performing scenario of the model.

Upon observation, it can be noticed that the model functions at an ideal and best state when the hyperparameters are spread across the higher end spectrum and the model perform very poorly when the hyperparameters are either too low with lesser number of features from the dataset and the lesser training set.

However, it can be estimated that the maximum weightage of hyperparameter altercation can be assigned to the Validation Size of the subset and the number of features that a model draws from a dataset.

Problem Statement 3

Question

The expectation with this question is to compare the techniques from Problem Statement 1 and Problem Statement 2 in terms of their achieved accuracy and weight of faulty behavior. It is also expected to determine the performance of one over the other with a list of advantages and disadvantages for both.

Solution

Random Forest Classifier model	**Gradient Boosting Learning Classifier Algorithm**
Relies on validation set size to provide best efficiency	Relies on validation set size and Number of features from the dataset to provide best efficiency
Best Efficiency when hyperparameters are near the mean of the altercation distribution	Best Efficiency when the number of features from dataset is high and the validation subset has reasonable quantity of data
Less prone to overfitting	More prone to overfitting if hyperparameters are not tuned effectively

Table 4: Comparison between the two algorithms.

The Random Forest Classifier model generally is prone from overfitting so the mistakes on the test data. However, from the performed experiment it can witnessed that the mistakes have been distinct and there is lesser chance of repetition thus improving learning with each iteration. In the case of Gradient Boosting Learning Classifier Algorithm, however, the number of True Negatives is higher in the default case scenario. This explains that the model is fragile over the Random Forest Classifier model.

Random Forest Classifier model	**Gradient Boosting Learning Classifier Algorithm**
Advantages	
Higher Accuracy	Higher Prediction Accuracy
Less overfitting	Relevance of features
Scalable	Handles unbalanced data
Disadvantages	
Harder to interpret	Prone to overfitting
Slow Training process	Slow Training process
With Noisy data, prone to overfitting	Lesser patterns to interpret

Table 5: Advantages and Disadvantages between the two algorithms.

Problem Statement 4

Question

The expectation is to shuffle and split the Original Diabetes Dataset into Training/Validation (80%) and Testing (20%). This Training/Validation is further split into Training and Validation separately. A random forest regressor model is to be built to train, validate and test upon the chosen dataset. The list of hyperparameters and its altercations leading the changes in the results is expected to be documented. Along with a list of all examples where prediction has been incorrect.

Code

THE ENTIRE CODE BLOCK FOR THE PROBLEM STATEMENT 4 CAN BE FOUND IN THE APPENDIX WITHIN THE CODE 3 OF THIS DOCUMENT.

Design Process

1. Ensemble of Decision Trees:

It constructs multiple decision trees during the training phase and finds the average prediction of individual trees in the output.

2. Feature Randomness:

In each decision iteration, a random set of features is chosen which adds to the feature's overall randomness.

3. Parallel Training:

The individual trees can be trained in parallel leading to efficiency in training and prediction times.

Tradeoffs Considered

1. Higher performance but lesser interpretability.
2. Lesser risk of overfitting but can largely generalize the model.
3. Training time is less but prediction time is higher due to the need for aggregate prediction from multiple trees.

Output

```
The Total number of samples in the Diabetes Dataset is:

442
The number of samples in the Training Dataset is:

176
The number of samples in the Validation Dataset is:

177
The number of samples in the Testing Dataset is:

89
Mean Squared Error on the Validation dataset: 3146.703933898305
Mean Squared Error on the Test Set: 3718.239983146067
Top 10 Examples from the test set where the predictions have been incorrect:

Example: 63
Features: [ 0.01628068 -0.04464164  0.02612841
0.05860761 -0.06073493 -0.04421522
 -0.01394774 -0.03395821 -0.05140387 -0.02593034]
True Label: 52.0
Predicted Label: 223.74
Prediction Error: 171.74

Example: 7
Features: [-0.04910502 -0.04464164  0.16085492 -
0.04698463 -0.02908802 -0.01978964
 -0.04708248  0.03430886  0.02802037  0.01134862]
True Label: 346.0
Predicted Label: 186.57
Prediction Error: 159.43

Example: 73
Features: [-0.00914709 -0.04464164  0.01103904 -
0.05731319 -0.02496016 -0.04296262
  0.03023191 -0.03949338  0.01703607 -0.0052198 ]
True Label: 276.0
Predicted Label: 128.32
Prediction Error: 147.68

Example: 56
Features: [ 0.01628068 -0.04464164  0.02397278 -
0.02288468 -0.02496016 -0.02605261
 -0.03235593 -0.00259226  0.03723625  0.03205916]
True Label: 265.0
Predicted Label: 143.57
Prediction Error: 121.43

Example: 11
Features: [ 0.02717829 -0.04464164  0.00672779
0.03564379  0.07961226  0.07071027
  0.01550536  0.03430886  0.04067283  0.01134862]
True Label: 67.0
Predicted Label: 184.69
Prediction Error: 117.69

Example: 87
Features: [-0.03457486  0.05068012 -0.00081689
0.0700723   0.03970963  0.06695249
 -0.06549067  0.1081111   0.02671684  0.07348023]
True Label: 292.0
Predicted Label: 177.73
Prediction Error: 114.27000000000001

Example: 79
Features: [ 0.01628068 -0.04464164  0.02073935
0.02187239 -0.01395254 -0.01321352
 -0.00658447 -0.00259226  0.01331691  0.04034337]
True Label: 281.0
Predicted Label: 173.34
Prediction Error: 107.66

Example: 17
Features: [-0.04910502 -0.04464164  0.00457217
0.01154383 -0.03734373 -0.01853704
 -0.01762938 -0.00259226 -0.03980883 -0.02178823]
True Label: 200.0
Predicted Label: 92.91
Prediction Error: 107.09

Example: 54
Features: [ 0.03444337  0.05068012  0.12528712
0.02875809 -0.05385517 -0.01290037
 -0.10230705  0.1081111   0.00027248  0.02791705]
True Label: 341.0
Predicted Label: 239.95
Prediction Error: 101.05000000000001

Example: 18
Features: [-0.05637009 -0.04464164  0.09295276 -
0.01944183  0.01494247  0.02342485
 -0.02867429  0.02545259  0.02606052  0.04034337]
True Label: 128.0
Predicted Label: 223.0
Prediction Error: 95.0
```

The output clearly identifies the total number of samples in the Original Diabetes Dataset which is 442 and the number of samples divided for Training and Validation is 353 i.e. 80% (79.86%). Additionally, the Training and Validation is further split in half with 176 samples shared with Training and 177 samples shared with Validation. The number of samples divided for Testing is 89 i.e. 20% (20.13%).

The above output is obtained from the default set of hyperparameters used to construct the model and test on the dataset. It also has the list of top 10 incorrectly predicted examples from the test data.

Category	**Percentage split**	**Number of rows (samples)**
Training and Validation	80%	353 (176+177)
Training	50% of 80%	176
Validation	50% of 80%	177
Testing	20%	89

Table 6: Split of Data into Training, Validation and Testing with values.

Slno	Hyperparameters						Model Performance	
	No. of dataset features	Shuffling random state	Split #1 random state	Validation size	Split #2 random state	Model random state	Validation Mean Squared Error	Test Mean Squared Error
1	10	88	70	0.5	94	19	3146.7	3718.23
2	1	8	7	0.1	9	1	9106.3	8377.56
3	5	99	247	0.3	900	165	5288.59	3397.01
4	7	999	798	0.9	89	16	4440.44	4616.73
5	9	42	79	0.2	13	25	3423.72	3469.33

Table 7: Model's Performance based on varying Hyperparameters.

Table 7 compares the model's performance to the varying hyperparameters. The row colored in grey is the default state of the model. The scenario 3 in this case is the best performing scenario for the model with all the hyperparameter(s) altercations. The scenario 2 marked in red symbolize the worst performing scenario(s) of the model.

Upon observation, it can be noticed that the model functions with very less heed towards the hyperparameters in consideration to the random state and the validation size of the dataset as well.

Nevertheless, the number of features from the dataset is the highest determining factor to alter the mean squared error performed on the test data. It is also important to note that the application this algorithm upon the dataset isn't providing the desired necessary efficiency in results.

Considering, that the best-case scenario is closer to the default case scenario and from the output of the default case scenario, it is visible that the prediction Error ratio is in the average range of 100s between the predicted and the true label. Thus, it can be said that the learning curve of the model is quite minimal.

Problem Statement 5

Question

The expectation is to shuffle and split the Original Diabetes Dataset into Training/Validation (80%) and Testing (20%). This Training/Validation is further split into Training and Validation separately. A Gradient Boosting Learning regressor model is to be built to train, validate and test upon the chosen dataset. The list of hyperparameters and its altercations leading the changes in the results is expected to be documented. Along with a list of all examples where prediction has been incorrect.

Code

THE ENTIRE CODE BLOCK FOR THE PROBLEM STATEMENT 4 CAN BE FOUND IN THE APPENDIX WITHIN THE CODE 4 OF THIS DOCUMENT.

Output

The hyperparameters once altered deliver the following results:

Slno	Hyperparameters						Model Performance	
	No. of dataset features	Shuffling random state	Split #1 random state	Validation size	Split #2 random state	Model random state	Validation Mean Squared Error	Test Mean Squared Error
1	10	88	70	0.5	94	19	3602.77	3509.26
2	1	2	7	0.2	4	1	6131.88	5947.65
3	3	200	78	0.3	42	19	3973.58	4648.25
4	6	20	83	0.7	55	47	4414.73	4449.44
5	9	53	97	0.6	92	23	3527.7	3521.2

Table 8: Model's Performance based on varying Hyperparameters.

Table 8 compares the model's performance to the varying hyperparameters. The row colored in grey is the default state of the model. The scenario 1 which is the default, in this case is the best performing scenario for the model with all the hyperparameter(s) altercations. The scenario 2 marked in red symbolize the worst performing scenario of the model.

Upon observation, it can be noticed that the model functions at an ideal state with less deviation from the mean irrespective of the hyperparameter altercations. It can be also observed that with maximum increase to the validation subset, it does not directly influence the mean squared error values, but the number of features opted from the dataset does play a larger role in comparison if not to the entire model but against validation size.

However, it can be estimated that the maximum weightage of hyperparameter altercation can be assigned to the number of features from the subset and the validation size of the subset being assigned from the training/validation subset.

Considering, the best-case scenario from the example is the default case scenario whose output is below, it can be observed that the prediction Error ratio is in the average range of >100s between the predicted and the true label. Thus, it can be said that the learning curve of the model is quite minimal.

The Total number of samples in the Diabetes Dataset is:

442
The number of samples in the Training Dataset is:

176
The number of samples in the Validation Dataset is:

177
The number of samples in the Testing Dataset is:

89
Mean Squared Error on the Validation dataset: 3602.7711266144415
Mean Squared Error on the Test Set: 3509.2680548440494
Top 10 Examples from the test set where the predictions have been incorrect:

Example: 63
Features: [0.01628068 -0.04464164 0.02612841
0.05860761 -0.06073493 -0.04421522
-0.01394774 -0.03395821 -0.05140387 -0.02593034]
True Label: 52.0
Predicted Label: 238.92981951886725
Prediction Error: 186.92981951886725

Example: 73
Features: [-0.00914709 -0.04464164 0.01103904 -
0.05731319 -0.02496016 -0.04296262
0.03023191 -0.03949338 0.01703607 -0.0052198]
True Label: 276.0
Predicted Label: 127.52772750392793
Prediction Error: 148.47227249607207

Example: 71
Features: [-0.09996055 -0.04464164 -0.06764124 -
0.10895595 -0.07449446 -0.07271173
0.01550536 -0.03949338 -0.04987245 -0.009361911]
True Label: 55.0
Predicted Label: 186.21262818285655
Prediction Error: 131.21262818285655

Example: 11
Features: [0.02717829 -0.04464164 0.00672779
0.03564379 0.07961226 0.07071027
0.01550536 0.03430886 0.04067283 0.01134862]
True Label: 67.0
Predicted Label: 192.8494190495152
Prediction Error: 125.8494190495152

Example: 56
Features: [0.01628068 -0.04464164 0.02397278 -
0.02288468 -0.02496016 -0.02605261
-0.03235593 -0.00259226 0.03723625 0.03205916]
True Label: 265.0
Predicted Label: 145.09894582466094
Prediction Error: 119.90105417533906

Example: 7
Features: [-0.04910502 -0.04464164 0.16085492 -
0.04698463 -0.02908802 -0.01978964
-0.04708248 0.03430886 0.02802037 0.01134862]
True Label: 346.0
Predicted Label: 228.8565114644619
Prediction Error: 117.1434885355381

Example: 87
Features: [-0.03457486 0.05068012 -0.00081689
0.0700723 0.03970963 0.06695249
-0.06549067 0.1081111 0.02671684 0.07348023]
True Label: 292.0
Predicted Label: 177.0351850021873
Prediction Error: 114.96481499781271

Example: 18
Features: [-0.05637009 -0.04464164 0.09295276 -
0.01944183 0.01494247 0.02342485
-0.02867429 0.02545259 0.02606052 0.04034337]
True Label: 128.0
Predicted Label: 241.2126613984844
Prediction Error: 113.21266139848441

Example: 17
Features: [-0.04910502 -0.04464164 0.00457217
0.01154383 -0.03734373 -0.01853704
-0.01762938 -0.00259226 -0.03980883 -0.02178823]
True Label: 200.0
Predicted Label: 87.74850282122945
Prediction Error: 112.25149717877055

Example: 79
Features: [0.01628068 -0.04464164 0.02073935
0.02187239 -0.01395254 -0.01321352
-0.00658447 -0.00259226 0.01331691 0.04034337]
True Label: 281.0
Predicted Label: 184.47341036485378
Prediction Error: 96.52658963514622

Design Process

1. Sequential Training of Decision Trees.
2. Learning Rate of the model is controllable and is a more gradual process allowing for better generalization.
3. Relevance of features is important for this model as it provides insights based on the selected features.

Tradeoffs Considered

1. Potential Overfitting.
2. Sequential Training method requires longer processing time.
3. Interpretation of complex results is harder, but the prediction accuracy tends to be higher.

Problem Statement 6

Question

The expectation with this question is to compare the techniques from Problem Statement 4 and Problem Statement 5 in terms of their achieved accuracy and weight of faulty behavior. It is also expected to determine the performance of one over the other with a list of advantages and disadvantages for both.

Solution

Random Forest Regressor model	**Gradient Boosting Learning Regressor Algorithm**
Parallel Computing	Sequential Computing
High Accuracy, less prone to overfitting	Higher accuracy if hyperparameters are tuned appropriately
High performance when statistical noise present in data	High performance when unbalanced data available
Easier to Interpret	Harder to Interpret

Table 9: Comparison between Random Forest Regressor mode and Gradient Boosting Learning Regressor Algorithm.

The Random Forest Regressor model generally is known for its stability and robustness while the Gradient Boosting Regressor Algorithm can perform better if the parameters are tuned appropriately.
So, the expected outcome, dataset, and the tuning of hyperparameters completely determine the effectiveness of each algorithm. From the performed experiment it can witnessed that both the models don't provide an ideal fit for this type of dataset in real world scenarios. On those lines, the gradient boosting algorithm in a very remote case scenario performs better than the Random Forest Regressor due to the consistency being slightly better in obtaining values with lesser variance.

Random Forest Regressor model	**Gradient Boosting Learning Regressor Algorithm**
Advantages	
Stability and Robustness	Higher Accuracy
Efficiency in handling high-dimensional data	Relevance of features
Ensemble Approach	Handles unbalanced data

Disadvantages	
Reduced Interpretability	Prone to overfitting
Low performance on time series data	Sequential Training process
Slower training process	Reduced Interpretability

Table 10: Advantages and Disadvantages between the two algorithms.

Project 4 Conclusion

Project 4 explores the ensemble methods available in machine learning with predominant focus upon,

1. Random Forest Classifier Algorithm
2. Gradient Boosting Classifier Learning Algorithm
3. Random Forest Regressor Algorithm
4. Gradient Boosting Regressor Learning Algorithm

These methods being implemented on the Breast Cancer and Diabetes dataset helped understand and analyze the pitfalls and enabled the straightforward visibility upon the effectiveness of a model upon a particular type of dataset irrespective the model's prowess in general.

Appendix

Code 1

```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

data = load_breast_cancer()

#Selecting the no. of hyperparameters
ftr = data.data[: , :30]
trgt = data.target

#Shuffle the dataset before the split
ftr, trgt = shuffle(ftr, trgt, random_state = 48)

# Split data into training/validation (80%) and test (20%) sets
ftr_train_val, ftr_test, trgt_train_val, trgt_test = train_test_split(ftr, trgt,
test_size = 0.2, random_state = 50)

# Further split training/validation set into training and validation sets
ftr_train, ftr_val, trgt_train, trgt_val = train_test_split(ftr_train_val,
trgt_train_val, test_size = 0.5, random_state = 64)

# Printing the number of samples in the original dataset and split subsets
print("The Total number of samples in the Wisconsin Breast Cancer Dataset is: \n")
print(len(ftr))

print("The number of samples in the Training Dataset is: \n")
print(len(ftr_train))

print("The number of samples in the Validation Dataset is: \n")
print(len(ftr_val))

print("The number of samples in the Testing Dataset is: \n")
print(len(ftr_test))

#Creating the Random Forest Classifier Instance and training the model on the
Training Data
rfc = RandomForestClassifier(random_state = 39)
rfc.fit(ftr_train, trgt_train)

#Validating and printing the model on the validation dataset
val_predictions = rfc.predict(ftr_val)
print("Accuracy on the Validation set:", accuracy_score(trgt_val, val_predictions))
print("Precision on the Validation set:", precision_score(trgt_val,
val_predictions))
print("Recall on the Validation set:", recall_score(trgt_val, val_predictions))
print("F1 Score on the Validation set:", f1_score(trgt_val, val_predictions),"\n")

#Testing the Model on the test set
test_predictions = rfc.predict(ftr_test)
print("Accuracy on the Test Set:", accuracy_score(trgt_test, test_predictions))

#Listing the incorrect predications from the test set
incorrect = [i for i in range(len(ftr_test)) if test_predictions[i] !=
trgt_test[i]]
print("Examples from the test set where the predictions have been incorrect:\n")
for i in incorrect:
    print("Example:", i+1)
    print("Features:", ftr_test[i])
    print("True Label:", trgt_test[i])
    print("Predicted Label:", test_predictions[i])
    print()
```

Code 2

```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

data = load_breast_cancer()

#Selecting the no. of hyperparameters
ftr = data.data[: , :30]
trgt = data.target

#Shuffle the dataset before the split
ftr, trgt = shuffle(ftr, trgt, random_state = 48)

# Split data into training/validation (80%) and test (20%) sets
ftr_train_val, ftr_test, trgt_train_val, trgt_test = train_test_split(ftr, trgt,
test_size = 0.2, random_state = 50)

# Further split training/validation set into training and validation sets
ftr_train, ftr_val, trgt_train, trgt_val = train_test_split(ftr_train_val,
trgt_train_val, test_size = 0.5, random_state = 64)

# Printing the number of samples in the original dataset and split subsets
print("The Total number of samples in the Wisconsin Breast Cancer Dataset is: \n")
print(len(ftr))

print("The number of samples in the Training Dataset is: \n")
print(len(ftr_train))

print("The number of samples in the Validation Dataset is: \n")
print(len(ftr_val))

print("The number of samples in the Testing Dataset is: \n")
print(len(ftr_test))

#Creating the Random Forest Classifier Instance and training the model on the
Training Data
gbc = GradientBoostingClassifier(random_state = 29)
gbc.fit(ftr_train, trgt_train)

#Validating and printing the model on the validation dataset
val_predictions = gbc.predict(ftr_val)
print("Accuracy on the Validation set:", accuracy_score(trgt_val, val_predictions))
print("Precision on the Validation set:", precision_score(trgt_val,
val_predictions))
print("Recall on the Validation set:", recall_score(trgt_val, val_predictions))
print("F1 Score on the Validation set:", f1_score(trgt_val, val_predictions),"\n")

#Testing the Model on the test set
test_predictions = gbc.predict(ftr_test)
print("Accuracy on the Test Set:", accuracy_score(trgt_test, test_predictions))

#Listing the incorrect predications from the test set
incorrect = [i for i in range(len(ftr_test)) if test_predictions[i] !=
trgt_test[i]]
print("Examples from the test set where the predictions have been incorrect:\n")
for i in incorrect:
    print("Example:", i+1)
    print("Features:", ftr_test[i])
    print("True Label:", trgt_test[i])
    print("Predicted Label:", test_predictions[i])
    print()
```

Code 3

```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.datasets import load_diabetes
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

data = load_diabetes()

#Selecting the no. of hyperparameters
ftr = data.data[: , :10]
trgt = data.target

#Shuffle the dataset before the split
ftr, trgt = shuffle(ftr, trgt, random_state = 88)

# Split data into training/validation (80%) and test (20%) sets
ftr_train_val, ftr_test, trgt_train_val, trgt_test = train_test_split(ftr, trgt,
test_size = 0.2, random_state = 70)

# Further split training/validation set into training and validation sets
ftr_train, ftr_val, trgt_train, trgt_val = train_test_split(ftr_train_val,
trgt_train_val, test_size = 0.5, random_state = 94)

# Printing the number of samples in the original dataset and split subsets
print("The Total number of samples in the Diabetes Dataset is: \n")
print(len(ftr))

print("The number of samples in the Training Dataset is: \n")
print(len(ftr_train))

print("The number of samples in the Validation Dataset is: \n")
print(len(ftr_val))

print("The number of samples in the Testing Dataset is: \n")
print(len(ftr_test))

#Creating the Random Forest Classifier Instance and training the model on the
Training Data
rfr = RandomForestRegressor(random_state = 19)
rfr.fit(ftr_train, trgt_train)

#Validating and printing the model on the validation dataset
val_predictions = rfr.predict(ftr_val)
mse = mean_squared_error(trgt_val, val_predictions)
print("Mean Squared Error on the Validation dataset:", mse)

#Testing the Model on the test set
test_predictions = rfr.predict(ftr_test)
mse_test = mean_squared_error(trgt_test, test_predictions)
print("Mean Squared Error on the Test Set:", mse_test)

#Listing the top 10 incorrect predications from the test set
incorrect = abs(test_predictions - trgt_test)
incorrect_idx = incorrect.argsort()[::-1][:10]
print("Top 10 Examples from the test set where the predictions have been
incorrect:\n")
for i in incorrect_idx:
    print("Example:", i+1)
    print("Features:", ftr_test[i])
    print("True Label:", trgt_test[i])
    print("Predicted Label:", test_predictions[i])
    print("Prediction Error:", incorrect[i])
    print()
```

Code 4

```
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.datasets import load_diabetes
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error

data = load_diabetes()

#Selecting the no. of hyperparameters
ftr = data.data[: , :10]
trgt = data.target

#Shuffle the dataset before the split
ftr, trgt = shuffle(ftr, trgt, random_state = 88)

# Split data into training/validation (80%) and test (20%) sets
ftr_train_val, ftr_test, trgt_train_val, trgt_test = train_test_split(ftr, trgt,
test_size = 0.2, random_state = 70)

# Further split training/validation set into training and validation sets
ftr_train, ftr_val, trgt_train, trgt_val = train_test_split(ftr_train_val,
trgt_train_val, test_size = 0.5, random_state = 94)

# Printing the number of samples in the original dataset and split subsets
print("The Total number of samples in the Diabetes Dataset is: \n")
print(len(ftr))

print("The number of samples in the Training Dataset is: \n")
print(len(ftr_train))

print("The number of samples in the Validation Dataset is: \n")
print(len(ftr_val))

print("The number of samples in the Testing Dataset is: \n")
print(len(ftr_test))

#Creating the Random Forest Classifier Instance and training the model on the
Training Data
gbr = GradientBoostingRegressor(random_state = 19)
gbr.fit(ftr_train, trgt_train)

#Validating and printing the model on the validation dataset
val_predictions = gbr.predict(ftr_val)
mse = mean_squared_error(trgt_val, val_predictions)
print("Mean Squared Error on the Validation dataset:", mse)

#Testing the Model on the test set
test_predictions = gbr.predict(ftr_test)
mse_test = mean_squared_error(trgt_test, test_predictions)
print("Mean Squared Error on the Test Set:", mse_test)

#Listing the top 10 incorrect predications from the test set
incorrect = abs(test_predictions - trgt_test)
incorrect_idx = incorrect.argsort()[::-1][:10]
print("Top 10 Examples from the test set where the predictions have been
incorrect:\n")
for i in incorrect_idx:
    print("Example:", i+1)
    print("Features:", ftr_test[i])
    print("True Label:", trgt_test[i])
    print("Predicted Label:", test_predictions[i])
    print("Prediction Error:", incorrect[i])
    print()
```