# Calories Predicator GBR\_Grid (Gradient Boosted Regression)

# Introduction

Gradient Boosted Regression (GBR) is a flexible non-parametric supervised machine learning technique for classification and regression problems. It is an ensemble of boosted trees which are essentially a collection of decision stumps. For our data set we are building a regression model. We used GBR implementations available in Scikit-learn and XGBoost.

The goal of the project is to predict the burnt calories given the 11 input features. The features included

1. TotalSteps
2. TrackerDistance
3. LoggedActivitiesDistance
4. VeryActiveDistance
5. ModeratelyActiveDistance
6. LightActiveDistance
7. SedentaryActiveDistance
8. VeryActiveMinutes
9. FairlyActiveMinutes
10. LightlyActiveMinutes

We preprocess the data, then scale the features to range within , finally train and test a GBR model.

# Model Implementation

## Data post-processing:

1. Load the data
2. Split the data into 80% train and 20% test sets
3. Scale the features

## Training:

1. Set up grid search
   1. Choose values for hyper-parameters. Models will be trained and test for the cross product of all hyper-parameter options.
   2. A 5-fold cross-validation was done for each set of hyper-parameters
   3. Choose the scoring function whose value is used to choose the best set of hyper-parameters
2. Initialize a GBR model and perform grid search. An example is shown below.

Fitting 5 folds for each of 3456 candidates, totalling 17280 fits

[Parallel(n\_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.

[Parallel(n\_jobs=-1)]: Done 2 tasks | elapsed: 1.1s

[Parallel(n\_jobs=-1)]: Done 9 tasks | elapsed: 1.2s

[Parallel(n\_jobs=-1)]: Done 16 tasks | elapsed: 1.2s

[Parallel(n\_jobs=-1)]: Done 25 tasks | elapsed: 1.3s

[Parallel(n\_jobs=-1)]: Batch computation too fast (0.1922s.) Setting batch\_size=2.

[Parallel(n\_jobs=-1)]: Done 34 tasks | elapsed: 1.6s

[Parallel(n\_jobs=-1)]: Done 53 tasks | elapsed: 1.9s

[Parallel(n\_jobs=-1)]: Done 72 tasks | elapsed: 2.2s

[Parallel(n\_jobs=-1)]: Done 98 tasks | elapsed: 2.6s

…

[Parallel(n\_jobs=-1)]: Done 15096 tasks | elapsed: 19.6min

[Parallel(n\_jobs=-1)]: Done 15421 tasks | elapsed: 19.7min

[Parallel(n\_jobs=-1)]: Done 15743 tasks | elapsed: 19.9min

[Parallel(n\_jobs=-1)]: Batch computation too slow (2.3116s.) Setting batch\_size=1.

[Parallel(n\_jobs=-1)]: Done 15996 tasks | elapsed: 20.1min

[Parallel(n\_jobs=-1)]: Done 16159 tasks | elapsed: 20.4min

[Parallel(n\_jobs=-1)]: Done 16324 tasks | elapsed: 20.7min

[Parallel(n\_jobs=-1)]: Done 16489 tasks | elapsed: 20.8min

[Parallel(n\_jobs=-1)]: Done 16656 tasks | elapsed: 20.9min

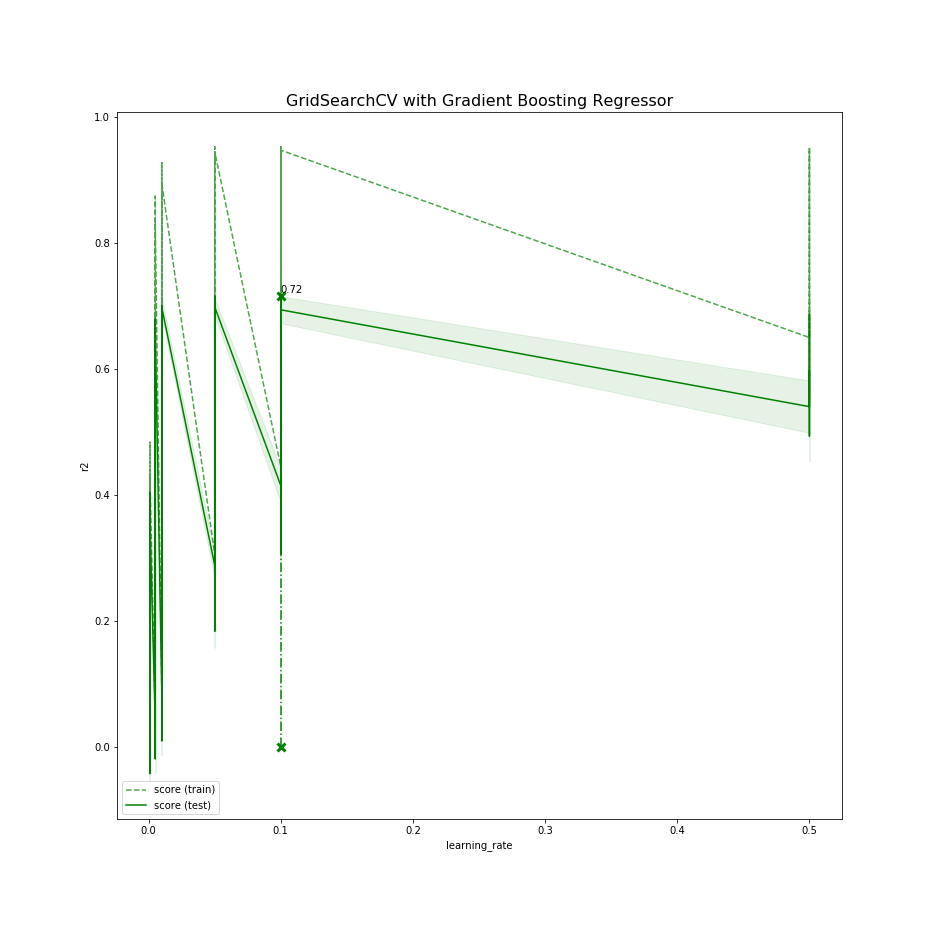
[Parallel(n\_jobs=-1)]: Done 16823 tasks | elapsed: 21.1min

[Parallel(n\_jobs=-1)]: Done 16992 tasks | elapsed: 21.4min

[Parallel(n\_jobs=-1)]: Done 17161 tasks | elapsed: 21.8min

[Parallel(n\_jobs=-1)]: Done 17280 out of 17280 | elapsed: 22.2min finished

The figure below plots r2 score vs different learning rates used to train the GBR model. The solid green line indicates the mean r2 score of the GBR model trained on training set with 5-fold cross validation at each learning rate. The dotted green line shows the same for models tested on testing set. As annotated on the figure, the best model is trained with a learning rate of 0.1 and achieves best r2 score of 0.72.

A

An example of a tree that was trained is given below. The tree nodes indicate the feature and threshold used to split. The left sub-tree (in blue) indicates samples that satisfied the decision stump criteria. The right sub-tree (in red) shows the results of non-complying with the decision stump.A picture containing text

Description generated with high confidence

1. Select the set of hyper-parameters that were used to train the model that gave the best performance on the chosen scoring function. An example is given below:

Best parameters found by grid search on the training set:

{'learning\_rate': 0.1, 'loss': 'ls', 'max\_depth': 3, 'max\_features': 'auto', 'min\_samples\_split': 7, 'n\_estimators': 500}

1. Train a GBR model with this set of hyper-parameters using the entire training set
2. Test the trained GBR model on the testing set. The Root Mean Square Error (RMSE) and r2 score for the best GBR model is given below.

Training Root Mean Square Error (RMSE): 211.66188137758098

Testing Root Mean Square Error (RMSE): 370.5609089746723

Training R2 score: 0.9238298370772501

Testing R2 score: 0.7189915684488977

## Testing:

1. Load the test data or user given data
2. Scale the data using the params found during the training process
3. Load the model
4. Run the prediction

## Conclusion:

Model train performance is great but when train and test performance is compared results is not good. Looking at difference in train and test root mean square we can conclude it is a case of overfitting.