Intro to Boosting

By Naman Bhayani

Intro to ensemble models

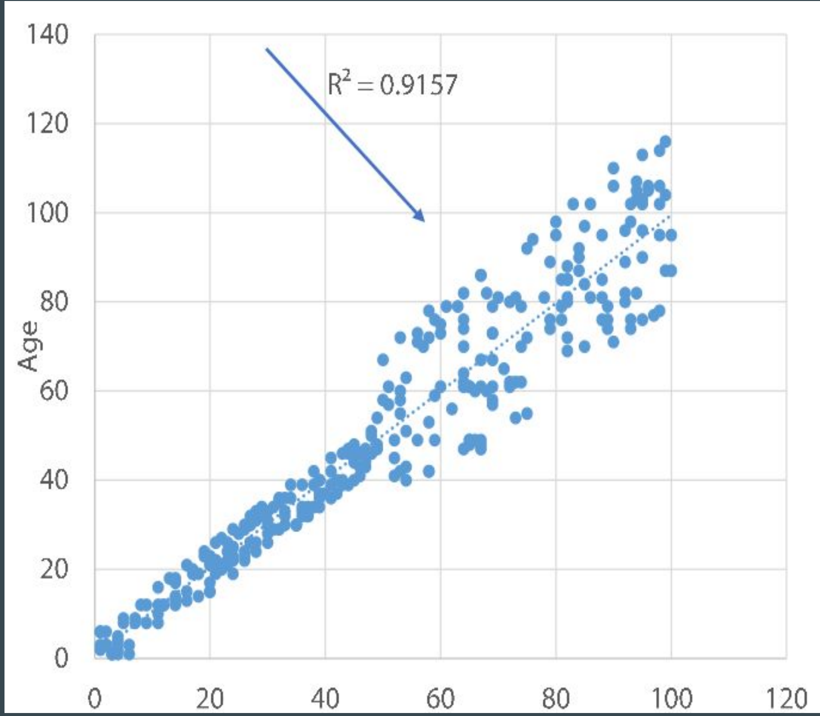
What is ensemble modelling ??

It means combining different ML models to get a better prediction

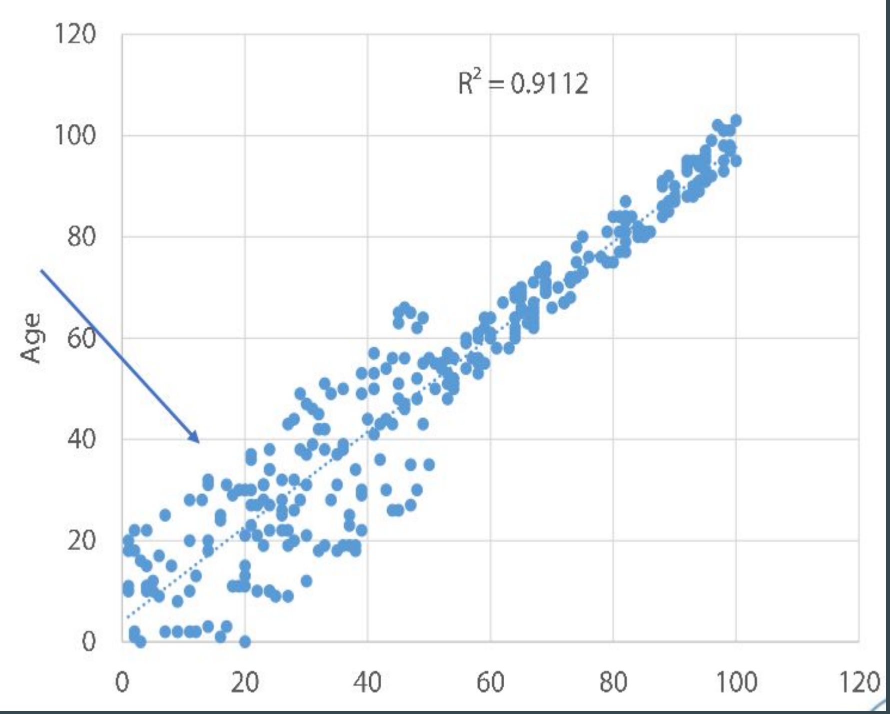
Overview : Ensemble Techniques

- Averaging
- Weighted Averaging
- Conditional Averaging
- Bagging
- **Boosting**
- Stacking

Averaging Ensemble Method

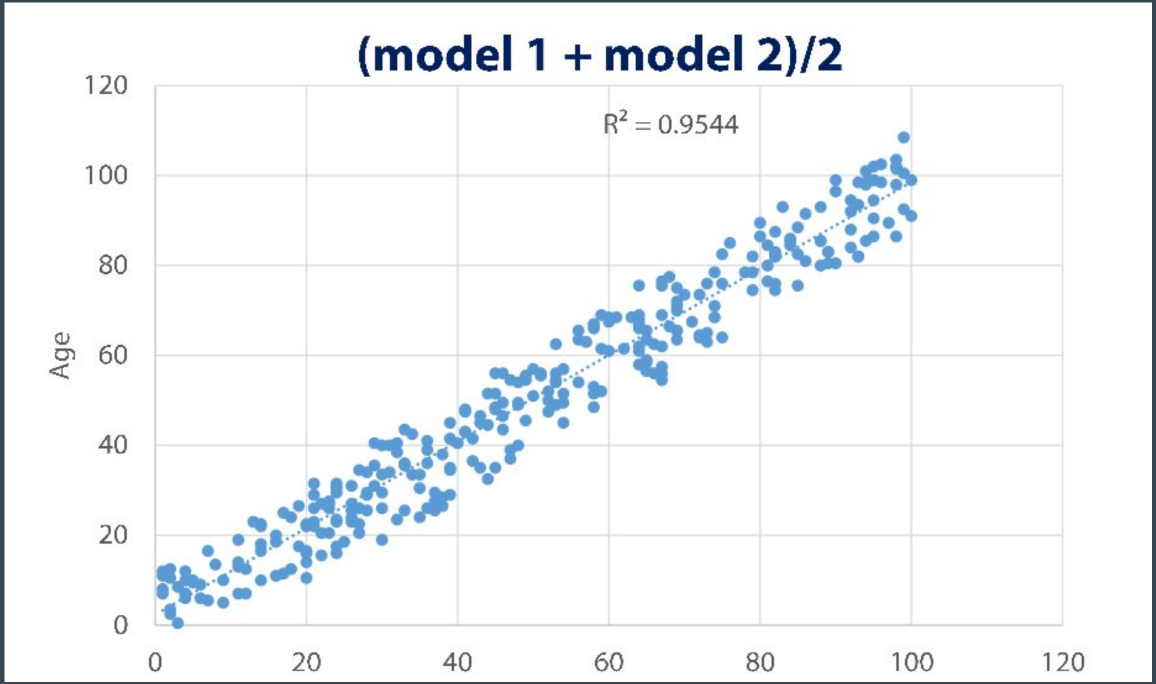


Prediction of age model 1



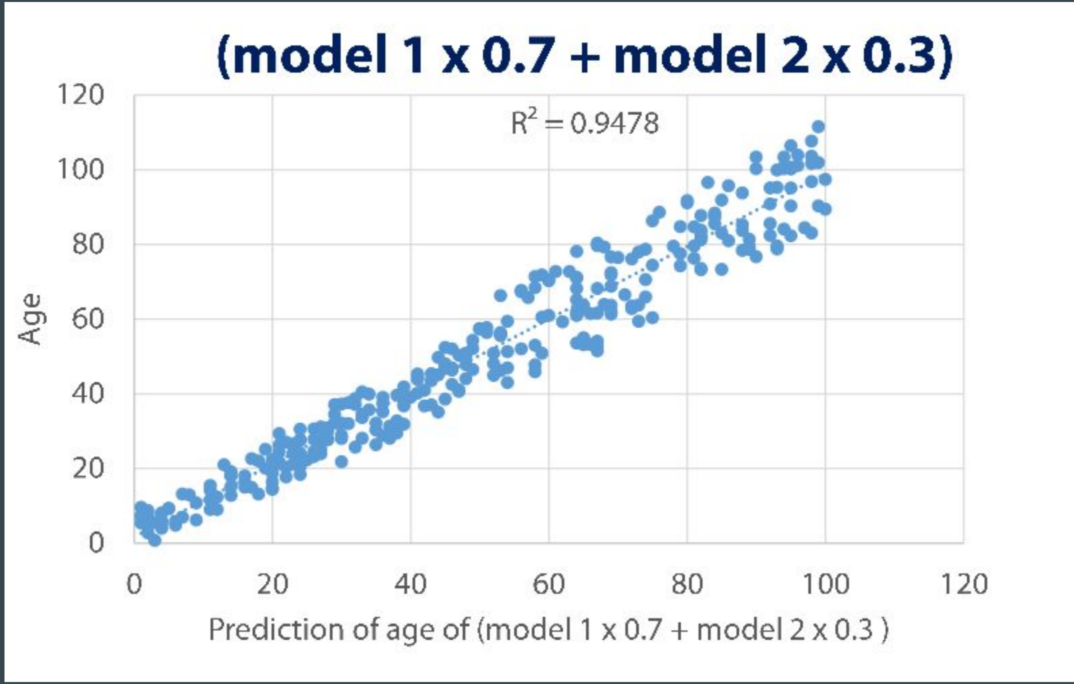
Prediction of age model 2

Averaging Ensemble Method

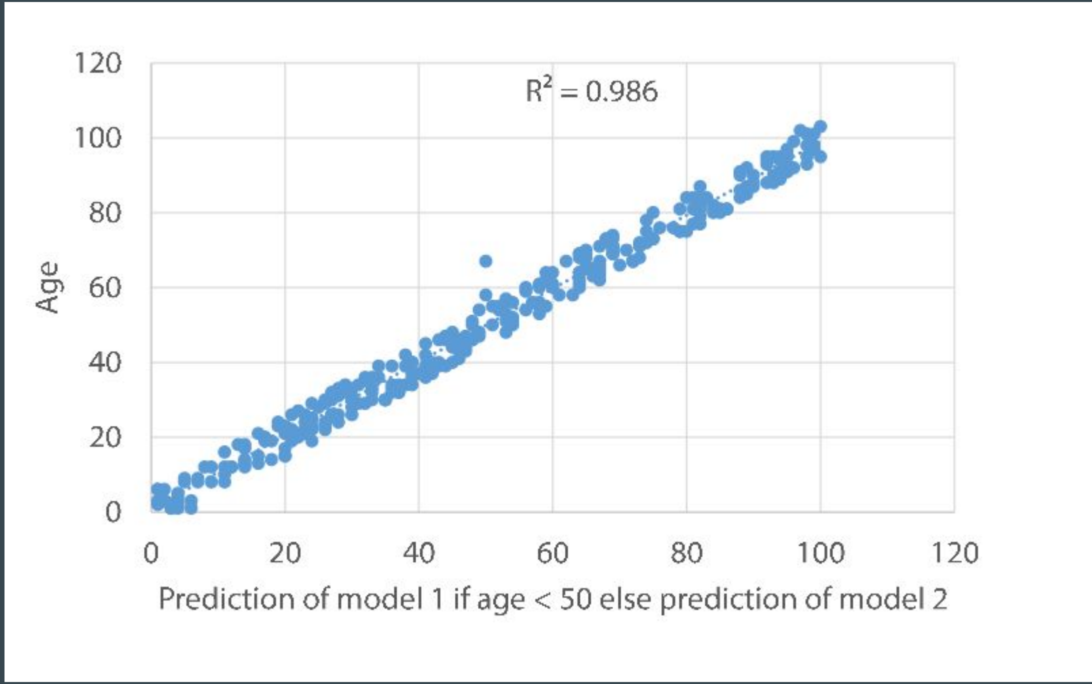


Prediction of age (model 1 + model 2)/2

Averaging Ensemble Method



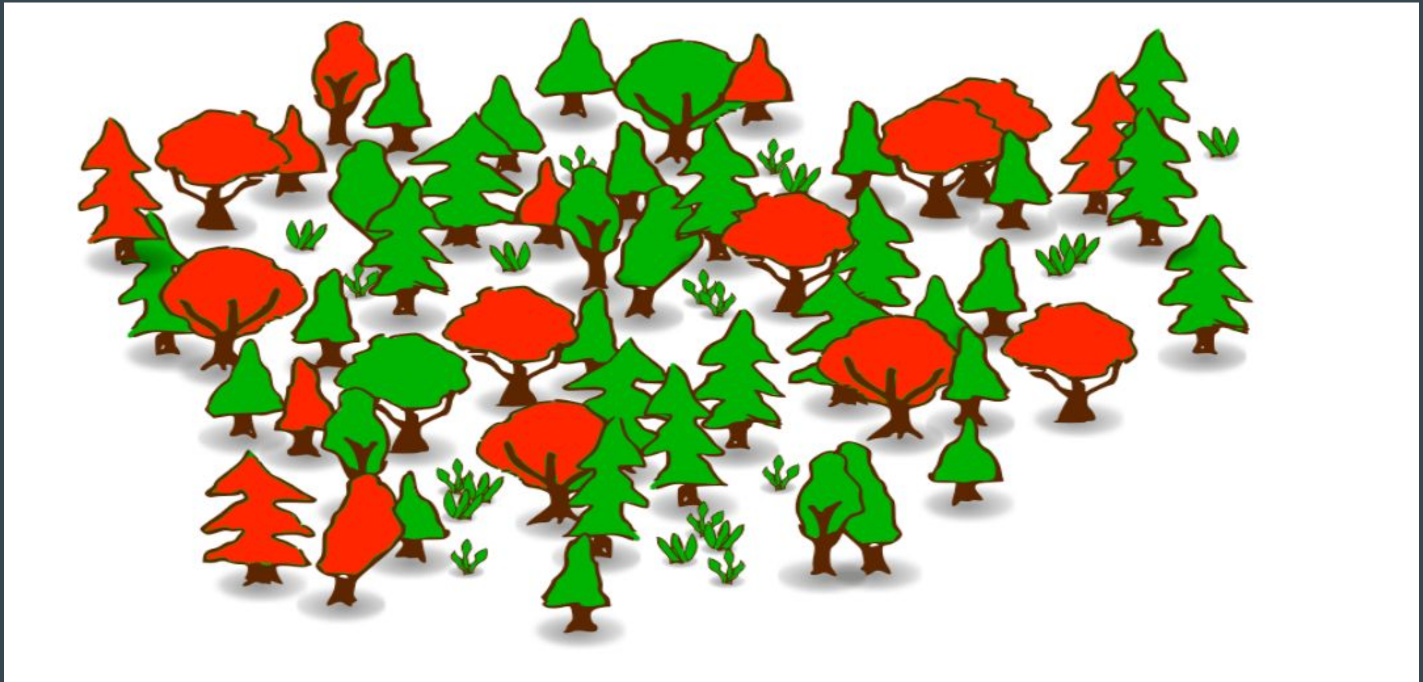
Averaging Ensemble Method



Conditional Averaging

What is Bagging ?

Means averaging slightly different versions of the same model to improve accuracy



Parameters that control bagging?

- Changing the seed
- Row (Sub) sampling or Bootstrapping
- Shuffling
- Column (Sub) sampling
- Model-specific parameters
- Number of models (or bags)
- (Optionally) parallelism

What is Boosting ?

A form of weighted averaging of models where each model is built sequentially via taking into the account the past performance model

Main boosting types

- Weight based
- Residual based

Weight boosting based

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	1
1	0.84	0.79	0.89	0.05	1
2	0.83	0.11	0.23	0.42	1
3	0.74	0.26	0.03	0.41	0
4	0.08	0.29	0.76	0.37	0
5	0.71	0.76	0.43	0.95	1
6	0.08	0.72	0.97	0.04	0

Weight boosting based

Rownum	x0	x1	x2	x3	y	pred
0	0.94	0.27	0.80	0.34	1	0.80
1	0.84	0.79	0.89	0.05	1	0.75
2	0.83	0.11	0.23	0.42	1	0.65
3	0.74	0.26	0.03	0.41	0	0.40
4	0.08	0.29	0.76	0.37	0	0.55
5	0.71	0.76	0.43	0.95	1	0.34
6	0.08	0.72	0.97	0.04	0	0.02

Weight boosting based

Rownum	x0	x1	x2	x3	y	pred	abs.error
0	0.94	0.27	0.80	0.34	1	0.80	0.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35
3	0.74	0.26	0.03	0.41	0	0.40	0.40
4	0.08	0.29	0.76	0.37	0	0.55	0.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66
6	0.08	0.72	0.97	0.04	0	0.02	0.02

Weight boosting based

Rownum	x0	x1	x2	x3	y	pred	abs.error	weight
0	0.94	0.27	0.80	0.34	1	0.80	0.20	1.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25	1.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35	1.35
3	0.74	0.26	0.03	0.41	0	0.40	0.40	1.40
4	0.08	0.29	0.76	0.37	0	0.55	0.55	1.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66	1.66
6	0.08	0.72	0.97	0.04	0	0.02	0.02	1.02

Weight boosting based

Rownum	x0	x1	x2	x3	y	weight
0	0.94	0.27	0.80	0.34	1	1.20
1	0.84	0.79	0.89	0.05	1	1.25
2	0.83	0.11	0.23	0.42	1	1.35
3	0.74	0.26	0.03	0.41	0	1.40
4	0.08	0.29	0.76	0.37	0	1.55
5	0.71	0.76	0.43	0.95	1	1.66
6	0.08	0.72	0.97	0.04	0	1.02

Weight boosting parameters

- Learning rate (or eta)
- Number of estimators
- Input model
- Sub Boosting type :
  - Adaboost
  - Logitboost

Residual based boosting

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	1
1	0.84	0.79	0.89	0.05	1
2	0.83	0.11	0.23	0.42	1
3	0.74	0.26	0.03	0.41	0
4	0.08	0.29	0.76	0.37	0
5	0.71	0.76	0.43	0.95	1
6	0.08	0.72	0.97	0.04	0

Residual based boosting

Rownum	x0	x1	x2	x3	y	pred
0	0.94	0.27	0.80	0.34	1	0.80
1	0.84	0.79	0.89	0.05	1	0.75
2	0.83	0.11	0.23	0.42	1	0.65
3	0.74	0.26	0.03	0.41	0	0.40
4	0.08	0.29	0.76	0.37	0	0.55
5	0.71	0.76	0.43	0.95	1	0.34
6	0.08	0.72	0.97	0.04	0	0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y	pred	error
0	0.94	0.27	0.80	0.34	1	0.80	0.20
1	0.84	0.79	0.89	0.05	1	0.75	0.25
2	0.83	0.11	0.23	0.42	1	0.65	0.35
3	0.74	0.26	0.03	0.41	0	0.40	-0.40
4	0.08	0.29	0.76	0.37	0	0.55	-0.55
5	0.71	0.76	0.43	0.95	1	0.34	0.66
6	0.08	0.72	0.97	0.04	0	0.02	-0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y
0	0.94	0.27	0.80	0.34	0.2
1	0.84	0.79	0.89	0.05	0.25
2	0.83	0.11	0.23	0.42	0.35
3	0.74	0.26	0.03	0.41	-0.4
4	0.08	0.29	0.76	0.37	-0.55
5	0.71	0.76	0.43	0.95	0.66
6	0.08	0.72	0.97	0.04	-0.02

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred
0	0.94	0.27	0.80	0.34	0.2	0.15
1	0.84	0.79	0.89	0.05	0.25	0.20
2	0.83	0.11	0.23	0.42	0.35	0.40
3	0.74	0.26	0.03	0.41	-0.4	-0.30
4	0.08	0.29	0.76	0.37	-0.55	-0.20
5	0.71	0.76	0.43	0.95	0.66	0.24
6	0.08	0.72	0.97	0.04	-0.02	-0.01

Residual based boosting

Rownum	x0	x1	x2	x3	y	new pred	old pred
0	0.94	0.27	0.80	0.34	0.2	0.15	0.80
1	0.84	0.79	0.89	0.05	0.25	0.20	0.75
2	0.83	0.11	0.23	0.42	0.35	0.40	0.65
3	0.74	0.26	0.03	0.41	-0.4	-0.30	0.40
4	0.08	0.29	0.76	0.37	-0.55	-0.20	0.55
5	0.71	0.76	0.43	0.95	0.66	0.24	0.34
6	0.08	0.72	0.97	0.04	-0.02	-0.01	0.02

To predict row_num 1, we would say : Final prediction = 0.75 + 0.2 = 0.95

Residual based boosting parameters

- Learning rate (or eta)
- Number of estimators
- Row (sub) sampling
- Column sampling
- Input models : Trees
- Sub Boosting type :
  - Fully gradient based
  - DART

Residual based famous Implementations

- XGBOOST
- LIGHTGBM
- H2O's GBM
- CATBOOST
- Sklearn's GBM

A lil bit of Math :)

Objective for Tree Ensemble

- Model: assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- Objective

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss

Complexity of the Trees

- Possible ways to define Ω ?
  - Number of nodes in the tree, depth
  - L2 norm of the leaf weights
  - … detailed later

Types of Loss functions

So far we have learned:

- Using Square loss $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Will results in common gradient boosted machine
- Using Logistic loss $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$
  - Will results in LogitBoost

So How do we Learn?

- Objective: $\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in \mathcal{F}$
- We can not use methods such as SGD, to find f (since they are trees, instead of just numerical vectors)
- Solution: **Additive Training (Boosting)**
  - Start from constant prediction, add a new function each time

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

New function

Model at training round t

Keep functions added in previous round

Additive Training

- How do we decide which f to add?
  - Optimize the objective!!
- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

- Consider square loss

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}
$$

This is usually called residual from previous round

Recap: Boosted Tree Algorithm

- Add a new tree in each iteration
- Beginning of each iteration, calculate

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

- Use the statistics to greedily grow a tree $f_t(x)$

$$Obj = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

- Add $f_t(x)$ to the model $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$
  - Usually, instead we do $y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$
  - ϵ is called step-size or shrinkage, usually set around 0.1
  - This means we do not do full optimization in each step and reserve chance for future rounds, it helps prevent overfitting

Lets Code !!

Thank You