Adaboosting Hyper Parameter Tuning						
Sno	base_estimator	n_estimators	learning_rate	loss	random_state	R^2 value
1	DecisionTreeRegressor	100	0.5	linear	0	0.860
2	DecisionTreeRegressor	100	1.0	linear	0	0.856
3	DecisionTreeRegressor	1000	1.0	linear	None	0.852
4	DecisionTreeRegressor	1000	1.0	square	None	0.452
5	DecisionTreeRegressor	1000	1.0	exponential	None	0.472
6	DecisionTreeRegressor	10000	1.0	linear	0	0.856
7	DecisionTreeRegressor	10	0.1	linear	0	0.879
8	DecisionTreeRegressor	10	0.01	linear	0	0.880
9	**DecisionTreeRegressor**	**10**	**0.01**	**exponential**	**0**	**0.881**
10	RandomForestRegressor(max_depth=100)	10	0.01	exponential	0	0.865
11	RandomForestRegressor(max_depth=100)	10	0.01	linear	0	0.866

Gradient Boosting Hyper Parameter Tuning						
Sno	n_estimators	loss	criterion	max_features	random_state	R^2 value
1	100	squared_error	friedman_mse	None	None	0.883
2	100	absolute_error	squared_error	sqrt	0	0.868
3	**100**	**huber**	**friedman_mse**	**None**	**None**	**0.892**
4	100	quantile	friedman_mse	None	None	0.667
5	1000	huber	friedman_mse	None	None	0.851
6	10000	huber	friedman_mse	None	0	0.778
7	10000	huber	friedman_mse	log2	0	0.810
8	10	huber	friedman_mse	None	None	0.772
9	100	squared_error	squared_error	None	None	0.883

XGBoosting Hyper Parameter Tuning				
Sno	**eta(learning rate)**	**gamma**	**subsample**	**R^2 value**
1	0.3	0	1	0.832
2	0.6	1	0.05	0.830
3	1	10	0.9	0.768
4	0	0.5	0.5	-1.142
5	0.9	100	0	0.797
6	**0.1**	**1000**	**0.1**	**0.857**

Comparing all 3 boosting alogirthm's hyper parameter tuning, we infer that Gradient Boosting gives better accuracy in terms of R^2. Hence we can consider Gradient Boosting as final model