Final Val Method 18/03/2024

1. The Client Requirement Input is provides the Data Set of Insurance and Output is he wants to predict the Insurance charges based on the several parameters.

2. The Basic Information is,

Input →Dataset, Output →Insurance charges

Total No of Rows= 1338

Total No of Columns= 6

3. Here the Preprocessing Method is, to handle Categorical column using, Converting String to Number (Nominal Data→ One Hot Encoder)

1. Multiple Linear Regressions

MLR of r2 value = **0.789**

2. Support Vector Machine

SNO	Hyper Parameter	Linear(rvalue)	Rbf(Non linear value)	Poly(r value)	Sigmoid(r value)
1.	C10	0.462	-0.032	0.038	0.039
2.	C100	0.628	0.32	0.617	0.527
3.	C500	0.763	0.664	0.826	0.444
4.	C1000	0.764	0.82	0.856	0.287
5.	C2000	0.7440	0.85	0.860	-0.593
6.	C3000	0.741	0.866	0.859	-2.124

The SVM Regression use r2 value, rbf and Hyper Parameter(c=3000)=**0.866**

3. Decision Tree

criterion*{"squared_error", "friedman_mse", "absolute_error", "poisson"}, default="squared_error"*
splitter*{"best", "random"}, default="best"*
max_features*int, float or {"sqrt", "log2"}, default=None*

S.No	CRITERION	MAXFEATURES	SPLITTER	RVALUE r2
1.	Squared_error	sqrt	best	0.77

2.	Squared_error	log2	Best	0.719
3.	Squared_error	sqrt	random	0.58
4.	Squared_error	log2	random	0.68
5.	Friedman_mse	sqrt	best	0.759
6.	Friedman_mse	log2	Best	0.677
7.	Friedman_mse	sqrt	random	0.716
8.	Friedman_mse	log2	random	0.721
9.	absolute_error	sqrt	best	0.7211
10.	absolute_error	log2	Best	0.714
11.	absolute_error	sqrt	random	0.757
12.	absolute_error	log2	random	0.678
13.	poisson	sqrt	best	0.724
14.	poisson	log2	Best	0.683
15.	poisson	sqrt	random	0.662
16.	poisson	log2	random	0.692
17.	absolute_error		best	0.68
18	absolute_error		random	0.745
19.	Friedman_mse		best	0.697
20.	Friedman_mse		random	0.693

The Decision Tree use r2 value, criterion=squared_error ,Maxfeatures = sqrt, splitter = best = **0.77**

4. Random Forest

criterion*{"squared_error", "absolute_error", "friedman_mse", "poisson"}, default="squared_error"*
max_features*{"sqrt", "log2", None}, int or float, default=1.0*

s.no	n_estimators	random state	criterion	max_features	r2
1.	100	0	squared_error	sqrt	0.871
2.	100	0	squared_error	log2	0.8710
3.	100	0	absolute_error	sqrt	0.87106
4.	100	0	absolute_error	log2	0.87108
5.	100	0	friedman_mse	sqrt	0.8710
6.	100	0	friedman_mse	log2	0.871054
7.	100	0	poisson	sqrt	0.868
8.	100	0	poisson	log2	0.868
9.	100	0			0.849

The Random Forest use r2 value,n_estimators=100,randomstate=0,criterion=squared_error,max_features = **0.87108**

- **The Result of Random Forest is better accuracy value 0.871 compared to all Algorithm.**

5.Ada Boost Algorithm

regr = AdaBoostRegressor(random_state=0, n_estimators=100)

The AdaAlgorithm r2value is = **0.8618**

6.XGBoost Algorithm

regr = XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7, colsample_bytree=0.8,use_rmm="true",booster="gbtree",device="cpu",verbosity=1,validate_parameters="false",disable_default_eval_metric="false",gamma=0,min_child_weight=1,max_delta_step=0,sampling_method="uniform",colsample_bylevel=1,colsample_bynode=1,Lambda=1,alpha=1,tree_method="auto",scale_pos_weight=1,refresh_leaf=1,process_type="default",grow_policy="depthwise",max_leaves=0,max_bin=256,num_parallel_tree=1,save_period=0,task="train",model_in="NULL",model_out="NULL",model_dir="models/",dump_format="text",name_dump="dump.txt",name_pred="pred.txt",pred_margin=0)

The XG Boost Algorithm r2 value is= **0.8128**

7.LG Boost Algorithm

from lightgbm import LGBMRegressor regr = LGBMRegressor(boosting_type='gbdt', num_leaves=31, max_depth=-1, learning_rate=0.1, n_estimators=100, subsample_for_bin=200000, objective=None, class_weight=None, min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=0,force_row_wise="true", colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None, n_jobs=None, importance_type='split') regr.fit(X_train, y_train)

The LG Boost Algorithm r2 value is= **0.8699**