Hope Artificial Intelligence

Assignment-Regression Algorithm

Problem Statement or Requirement:

A client's requirement is, he wants to predict the insurance charges based on the several parameters. The Client has provided the dataset of the same.
As a data scientist, you must develop a model which will predict the insurance charges.

1.) Identify your problem statement.

- Given Input and output in Numerical value, So, it comes under the **Machine Learning.**
- Given input and output data is very clear. So, It comes under **the Supervised learning.**
- Given output is numerical data. So, It is comes under the **Regression.**

The client is asking us to create a model for Insurance charges prediction by using the dataset provided. This AI Model can be named as "**Premium Predict**"

2.) Tell basic info about the dataset (Total number of rows, columns)

- Rows : 1338 & Column : 6

3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)

- Ordinal – Mapping-Label Encoder method is used for converting string to number.

4.) Develop a good model with r2_score. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.

- I have created the AI models by using the machine learning algorithms like Multiple Linear Regression, Support Vector Regression, Decision Tree Regression and Random Forest Regression.

- I have used with and without **standardisation** models and also the **hyper parameters** for each and every model to find a higher performance model.

- From all among this algorithms, the **Random Forest Regression** gives the maximum value of **R2_score= 0.87285773**. And it gives the better results.

5.) All the research values (r2_score of the models) should be documented. (You can make tabulation or screenshot of the results.)

1. **Multiple Linear Regression : R2_Score = 0.7894790349867009 (STD)**

2. SVM_Regression: R2_Score = 0.75908903677411					
Sl.No.	**HYPER PARAMETER**	**LINEAR**	**RBF (NON-LINEAR)**	**POLY (NON-LINEAR)**	**SIGMOID (NON-LINEAR)**
1	C=1.0	-0.111661287	-0.088427328	-0.064292584	-0.089941217
2	C=10	-0.001617632	-0.081969104	-0.093116155	-0.090783198
3	C=100	0.54328182	-0.124803678	-0.099761723	-0.118145548
4	C=1000	0.634036931	-0.117490924	-0.055505938	-1.665908132
5	C=2000	0.689326311	-0.10778764	-0.002702451	-5.616431542
6	C=3000	0.759089037	-0.096212851	0.048928964	-12.01904811

2. SVM_Regression: R2_Score = 0.866339395091675 (STD)					
Sl.No.	**HYPER PARAMETER**	**LINEAR**	**RBF (NON-LINEAR)**	**POLY (NON-LINEAR)**	**SIGMOID (NON-LINEAR)**
1	C=1.0	-0.010102665	-0.083382386	-0.075699656	-0.075429243
2	C=10	0.462468414	-0.032273294	0.038716223	0.039307144
3	C=100	0.628879286	0.320031783	0.617956962	0.527610355
4	C=1000	0.764931174	0.810206485	0.856648768	0.287470695
5	C=2000	0.744041831	0.854776643	0.860557929	-0.593950973
6	C=3000	0.74142366	0.866339395	0.859893008	-2.124419479

3. Decision Tree Regression: R2_Score = 0.775952702332582 (STD)					
Sl.No.	**CRITERION**	**MAX FEATURES**	**SPLITTER**	**R_SCORE**	**R_SCORE (STD)**
1	squared_error	auto	best	0.697449415	0.721457029
2	squared_error	auto	random	0.676150437	0.671478331
3	squared_error	sqrt	best	0.775645403	0.775952702
4	squared_error	sqrt	random	0.602559735	0.658163795
5	squared_error	log2	best	0.643086699	0.656257605
6	squared_error	log2	random	0.624212612	0.54025081
7	squared_error	none	best	0.701801277	0.685262754
8	squared_error	none	random	0.76541829	0.676026522
9	friedman_mse	auto	best	0.688322302	0.700040133
10	friedman_mse	auto	random	0.734005392	0.689688451
11	friedman_mse	sqrt	best	0.690146634	0.614484065
12	friedman_mse	sqrt	random	0.676168862	0.714304717
13	friedman_mse	log2	best	0.617136987	0.735773085
14	friedman_mse	log2	random	0.731399505	0.621620027
15	friedman_mse	none	best	0.691928116	0.703995129
16	friedman_mse	none	random	0.713582495	0.723449985
17	absolute_error	auto	best	0.66764376	0.704032354
18	absolute_error	auto	random	0.74021839	0.71255446
19	absolute_error	sqrt	best	0.733197689	0.734647411
20	absolute_error	sqrt	random	0.744316099	0.661932571
21	absolute_error	log2	best	0.732033052	0.665659619
22	absolute_error	log2	random	0.774666203	0.66346427
23	absolute_error	none	best	0.670290227	0.688690896
24	absolute_error	none	random	0.736012851	0.715081198
25	poisson	auto	best	0.674913973	0.675324019
26	poisson	auto	random	0.709351977	0.65317657
27	poisson	sqrt	best	0.660861342	0.593302833
28	poisson	sqrt	random	0.606392102	0.697469666
29	poisson	log2	best	0.631001412	0.615847441
30	poisson	log2	random	0.648829107	0.706710025
31	poisson	none	best	0.686916328	0.678954732
32	poisson	none	random	0.624396255	0.66806455

4. Random Forest_Regression: R2_Score = 0.872857729963818 (STD)						
Sl.No.	**N_ESTIMATORS**	**CRITERION**	**MAX_FEATURES**	**Random_state**	**R_SCORE**	**R_SCORE (STD)**
1	50	squared_error	sqrt	0	0.86949674	0.86952805
2	100	squared_error	sqrt	0	0.87099539	0.87078037
3	50	squared_error	log2	0	0.86949674	0.86952805
4	100	squared_error	log2	0	0.87099539	0.87078037
5	50	squared_error	None	0	0.84988238	0.85096117
6	100	squared_error	None	0	0.85392358	0.85504088
7	50	friedman_mse	sqrt	0	0.87004442	0.87003276
8	100	friedman_mse	sqrt	0	0.87094577	0.87075058
9	50	friedman_mse	log2	0	0.87004442	0.87003276
10	100	friedman_mse	log2	0	0.87094577	0.87075058
11	50	friedman_mse	None	0	0.849998	0.85105196
12	100	friedman_mse	None	0	0.85400513	0.85509863
13	50	absolute_error	sqrt	0	0.87215896	0.87285773
14	100	absolute_error	sqrt	0	0.87172155	0.87257453
15	50	absolute_error	log2	0	0.87215896	0.87285773
16	100	absolute_error	log2	0	0.87172155	0.87257453
17	50	absolute_error	None	0	0.85290278	0.8544254
18	100	absolute_error	None	0	0.85214689	0.85335346
19	50	poisson	sqrt	0	0.82878664	0.83071885
20	100	poisson	sqrt	0	0.82932334	0.83110727
21	50	poisson	log2	0	0.82878664	0.83071885
22	100	poisson	log2	0	0.82932334	0.83110727
23	50	poisson	None	0	0.82795454	0.82923336
24	100	poisson	None	0	0.83321012	0.83404452

6.) Mention your final model, justify why u have chosen the same.

- **Random Forest Regression** algorithm gives the maximum value of **R2_score= 0.87285773**. And it gives the better performance than other algorithms.