Assignment For Regression Models

Problem Statement or Requirement:

A client's requirement is, he wants to predict the insurance charges based on 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
2. Tell basic info about the dataset (Total number of rows, columns)
3. Mention the pre-processing method if you're doing any (like converting string to number – nominal data)
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 the final model.
5. All the research values (r2_score of the models) should be documented. (You can make a tabulation or screenshot of the results.)
6. Mention your final model, justify why you have chosen the same.

1. Identification of the Given Problem Statement

Stage 1 - Domain - Machine Learning
Stage 2 - Learning - Supervised Learning
Stage 3 - Sub-Learning - Supervised Regression

2. Basic info of the Given Dataset

No. of Columns	6
No. of Rows	1338
Columns Names	age, sex, bmi, children, smoker, charges

3. Pre-Processing Method

1. We have Nominal data in our dataset.
2. So convert those data into numbers using ***One Hot Encoding***.
3. After converted the dataset, now we have columns as ['age', 'bmi', 'children', 'charges', 'sex_male', 'smoker_yes']

4. R2 Values for Various Models

1. **Multiple Linear Regression**: 0.7894790349867009

2. **Support Vector Machine**: 0.8566487675946572

(d) - default values

S.No	**kernal**	**C**	**R Value**	**After Standarization**	**Error**
1	rbf (d)	1.0 (d)	-0.08842732776913875	-0.08338238593619329	
2	rbf	10	-0.08196910396420853	-0.03227329390671052	
3	rbf	100	-0.12480367775039669	0.3200317832050831	
4	rbf	1000	-0.11749092439183229	0.8102064851758545	
5	linear	1.0	-0.11166128719608448	-0.010102665316081394	
6	linear	10	-0.0016176324886472138	0.4624684142339678	
7	linear	100	0.5432818196692804	0.6288792857320361	
8	linear	1000	0.634036931263208	0.764931173859684	
9	poly	1.0	-0.06429258402105531	-0.07569965570860893	
10	poly	10	-0.09311615532848516	0.038716222760231456	
11	poly	100	-0.09976172333666167	0.6179569624059799	
12	poly	1000	-0.055505937517909665	0.8566487675946572	
13	sigmoid	1.0	-0.0899412170256757	-0.07542924281107188	
14	sigmoid	10	-0.09078319814614	0.03930714378274347	
15	sigmoid	100	-0.11814554828411405	0.5276103546510411	
16	sigmoid	1000	-1.6659081315533064	0.2874706948697682	
17	precomputed	1.0	**Error**	**Error**	‘precomputed’ works for square matrix like 35x35. But in our case its 936x5 matrix

18	precomputed	10	**Error**	**Error**	‘precomputed’ works for square matrix like 35x35. But in our case its 936x5 matrix
19	precomputed	100	**Error**	**Error**	‘precomputed’ works for square matrix like 35x35. But in our case its 936x5 matrix
20	precomputed	1000	**Error**	**Error**	‘precomputed’ works for square matrix like 35x35. But in our case its 936x5 matrix

3. **Decision Tree**: 0.7729957681947621

(d) - default values

S.No	**criterion**	**max_features**	**splitter**	**R Value**
1	squared_error (d)	None (d)	best (d)	0.6931984016916541
2	squared_error	None	random	0.6708058161434312
3	squared_error	sqrt	best	0.6756636303524248
4	squared_error	sqrt	random	0.5682656067400402
5	squared_error	log2	best	0.7254047092795477
6	squared_error	log2	random	0.7250791356922162
7	friedman_mse	None	best	0.6783472755829045
8	friedman_mse	None	random	0.7084434151507868
9	friedman_mse	sqrt	best	0.7421788811393304
10	friedman_mse	sqrt	random	0.6403516471484053
11	friedman_mse	log2	best	0.7683937439284054
12	friedman_mse	log2	random	0.6369399132156228
13	absolute_error	None	best	0.6559618519657617

14	absolute_error	None	random	0.7729957681947621
15	absolute_error	sqrt	best	0.7279364710888775
16	absolute_error	sqrt	random	0.7356251986859715
17	absolute_error	log2	best	0.7048036216728053
18	absolute_error	log2	random	0.6119271376251367
19	poisson	None	best	0.730385695097619
20	poisson	None	random	0.7582915040648521
21	poisson	sqrt	best	0.7048593413813866
22	poisson	sqrt	random	0.6447047537714024
23	poisson	log2	best	0.663066791078027
24	poisson	log2	random	0.6645562284982043

4. **Random Forest**: 0.8540518935149612

(d) - default values,
For all instance => random_state=0

S.No	**criterion**	**n_estimators**	**R Value**
1	squared_error (d)	100 (d)	0.8538307913484513
2	squared_error	10	0.83303041340085
3	friedman_mse	100	0.8540518935149612
4	friedman_mse	10	0.8331662678473348
5	absolute_error	100	0.8520093621081837
6	absolute_error	10	0.835063555313752
7	poisson	100	0.8526334258892607
8	poisson	10	0.8313991040134341

5. Final Model

According to the given dataset we got a Model Using Support Vector Machine (SVM) Algorithm with accuracy of **0.8566487675946572**

Other Models Accuracy:

Multiple Linear Regression	0.7894790349867009
Decision Tree	0.7729957681947621
Random Forest	0.8540518935149612