Insurance Charges AI Prediction

Problem Statement:

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.

1.) Identify your problem statement

- ***Machine Learning*** – Here data are number format.
- ***Supervised Learning*** - Here the requirement is "predict the insurance charges" clear. Then Input and Output data are available (insurance_pre.csv file).
- ***Regression*** - It's continuous value (i.e Numerical)

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

In dataset, it contains "age", "sex", "bmi", "children", "smoker" and "charges" attributes.

Input parameters are "age", "sex", "bmi", "children", "smoker and output is "charges"

```
[2]: dataset = pd.read_csv("insurance_pre.csv")
```

```
[5]: dataset
```

[5]:

	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230
2	28	male	33.000	3	no	4449.46200
3	33	male	22.705	0	no	21984.47061
4	32	male	28.880	0	no	3866.85520
...	...	...	...	...	...	...
1333	50	male	30.970	3	no	10600.54830
1334	18	female	31.920	0	no	2205.98080
1335	18	female	36.850	0	no	1629.83350
1336	21	female	25.800	0	no	2007.94500
1337	61	female	29.070	0	yes	29141.36030

1338 rows × 6 columns

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

"sex" and "smoker" attributes are categorial data. So, it has pre-processed into nominal data by using ***"one hot encoding"***

```
#data preprocessing - because categorial data can't be handle so we modify the data into numerical data using one hot encoding(Nominal)

dataset = pd.get_dummies(dataset, drop_first=True)
```

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 used regression algorithms like ***Multiple Linear Regression, Support vector machine, Decision tree and Random Forest algorithm*** from Machine Learning to predict the insurance charges.

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

Dataset – insurance_pre

1.**Multiple Linear Regression** – R^2 value = 0.789

2.**Support Vector Machine**

Parameters:

- *Cfloat, default=1.0*
- kernel*{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*

Without Standardization:

S.No	***Hyper Parameter C***	***Linear R2 value***	***RBF R2 value***	***Poly R2 value***	***Sigmoid R2 value***
1	10	-0.001	-0.081	-0.093	-0.090
2	100	0.543	-0.124	-0.099	-0.118
3	1000	0.634	-0.117	-0.055	-1.665
4	2000	0.689	-0.107	-0.002	-5.616
5	3000	0.759	-0.096	0.048	-12.019

With Standardization:

S.No	***Hyper Parameter C***	***Linear R2 value***	***RBF R2 value***	***Poly R2 value***	***Sigmoid R2 value***
1	10	0.462	-0.032	0.038	0.039
2	100	0.628	0.320	0.617	0.527
3	1000	0.764	0.810	0.856	0.287
4	2000	0.744	0.854	0.860	-0.593
5	3000	0.741	0.866	0.859	-2.124

R^2 value from SVR is 0.866 (kernel='rbf', c=3000)

3.Decision Tree

Parameters:

- 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***	***Splitter***	***max_features***	***R^2 Value***
1. 1				0.706
2.	*squared_error*	*Best*	*None*	0.718
3.	*squared_error*	*Best*	*Sqrt*	0.577
4.	*squared_error*	*Best*	*Log2*	0.711
5.	*squared_error*	*Random*		0.660
6.	*squared_error*	*Random*	*None*	0.746
7.	*squared_error*	*Random*	*Sqrt*	0.651
8.	*squared_error*	*Random*	*Log2*	0.670
9.	*friedman_mse*	*Best*		0.695
10.	*friedman_mse*	*Best*	*None*	0.703
11.	*friedman_mse*	*Best*	*Sqrt*	0.700
12.	*friedman_mse*	*Best*	*Log2*	0.717
13.	*friedman_mse*	*Random*		0.708
14.	*friedman_mse*	*Random*	*None*	0.730
15.	*friedman_mse*	*Random*	*Sqrt*	0.696
16.	*friedman_mse*	*Random*	*Log2*	0.630
17.	*absolute_error*	*Best*		0.661
18.	*absolute_error*	*Best*	*None*	0.690
19.	*absolute_error*	*Best*	*Sqrt*	0.726
20.	*absolute_error*	*Best*	*Log2*	0.713
21.	*absolute_error*	*Random*		0.723
22.	*absolute_error*	*Random*	*None*	0.715
23.	*absolute_error*	*Random*	*Sqrt*	0.719
24.	*absolute_error*	*Random*	*Log2*	0.754
25.	*poisson*	*Best*		0.727
26.	*poisson*	*Best*	*None*	0.720
27.	*poisson*	*Best*	*Sqrt*	0.733
28.	*poisson*	*Best*	*Log2*	0.691
29.	*poisson*	*Random*		0.716
30.	*poisson*	*Random*	*None*	0.668
31.	*poisson*	*Random*	*Sqrt*	0.707
32.	*poisson*	*Random*	*Log2*	0.755

In decision tree, criterion='poisson', splitter='best' gives best R^2 value as 0.727

4. **Random Forest**

Parameters:

- n_estimators*int, default=100*
- 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***	***Criterion***	***max_features***	***R^2 Value***
1	*100*	*squared_error*	*None*	0.854
2	*100*	*squared_error*	*Sqrt*	0.870
3	*100*	*squared_error*	*Log2*	0.870
4	*50*	*squared_error*	*None*	0.850
5	*50*	*squared_error*	*Sqrt*	0.869
6	*50*	*squared_error*	*Log2*	0.869
7	*10*	*squared_error*	*None*	0.833
8	*10*	*squared_error*	*Sqrt*	0.852
9	*10*	*squared_error*	*Log2*	0.852
10	*100*	*friedman_mse*	*None*	0.855
11	*100*	*friedman_mse*	*Sqrt*	0.870
12	*100*	*friedman_mse*	*Log2*	0.870
13	*50*	*friedman_mse*	*None*	0.851
14	*50*	*friedman_mse*	*Sqrt*	0.870
15	*50*	*friedman_mse*	*Log2*	0.870
16	*10*	*friedman_mse*	*None*	0.833
17	*10*	*friedman_mse*	*Sqrt*	0.851
18	*10*	*friedman_mse*	*Log2*	0.851
19	*100*	*absolute_error*	*None*	0.853
20	*100*	*absolute_error*	*Sqrt*	0.871
21	*100*	*absolute_error*	*Log2*	0.871
22	*50*	*absolute_error*	*None*	0.854
23	*50*	*absolute_error*	*Sqrt*	0.871
24	*50*	*absolute_error*	*Log2*	0.871
25	*10*	*absolute_error*	*None*	0.836
26	*10*	*absolute_error*	*Sqrt*	0.858
27	*10*	*absolute_error*	*Log2*	0.858
28	*100*	*poisson*	*None*	0.853
29	*100*	*poisson*	*Sqrt*	0.867
30	*100*	*poisson*	*Log2*	0.867
31	*50*	*poisson*	*None*	0.850
32	*50*	*poisson*	*Sqrt*	0.863
33	*50*	*poisson*	*Log2*	0.863
34	*10*	*poisson*	*None*	0.832
35	*10*	*poisson*	*Sqrt*	0.854
36	*10*	*poisson*	*Log2*	0.854

In Random Forest, below model gives best R^2 value = 0.871

- (n_estimators=100, criterion='absolute_error', max_features='sqrt')
- (n_estimators=100, criterion='absolute_error', max_features='log2')
- (n_estimators=50, criterion='absolute_error', max_features='sqrt')
- (n_estimators=50, criterion='absolute_error', max_features='log2')

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

I have chosen ***"Random Forest"*** algorithm as a final model, because "R score" value is good when compare to other algorithms.

S.No	***Algorithm***	***Parameter***	***R Score***
1	Multiple Linear Regression		0.789
2	Support Vector Machine	(kernel='rbf', c=3000)	0.866
3	Decision Tree	(criterion='poisson', splitter='best')	0.727
4	Random Forest	(n_estimators=100, criterion='absolute_error', max_features='sqrt')	0.871