`Name: Mariappan

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.

A) How will you achieve this in AI?

On understanding the problem statement First, I have to determine the domain selection, second find the learning selection for the given problem and finally decide whether regression or classification if the first stage falls under supervised learning.

First, the data is to be collected from the company and visualized to identify the nature of the data collected.

B) Lets find out the 3 - Stage of Problem Identification

Stage 1: The client's requirement is very clear to predict the insurance charges. The data provided by the client is not in pure numbers. But, It is a mixture of categorical column and numeric value. So, the categorical column can be converted into numbers in data preprocessing and hence it will be subjected to machine language algorithm processing. The problem for the client can be solved using machine learning algorithm.

Stage 2: Identify the learning:-The solution for the insurance company requirement comes under supervised learning as the output and to predict the insurance charges and the input from the client data are very clear.

Stage 3: Dataset falls under multi linear regression because the client wanted to estimate insurance charges with the several input parameters such as 'age', 'sex', 'bmi', 'children', 'smoker'.

C) Name the project

insurance charges Prediction.

To provide the best algorithm to the client, let me subject client's data to different set of Machine Learning Algorithm namely.

1. SVM
2. Decision Tree
3. Random Forest

Finally, conclude on providing best fit algorithm,

To do :

1. Analyse the data
2. Preprocess data

3. Segregate the data input and output.
4. Split dataset to train set and test set
5. Find std deviation if data value range is not within standard range
6. Generate the model
7. Evaluate error assessment with r2 square.

1. SVM

The evaluation metrics value is observed using Support Vector Machine Algorithm.

S.No	kernel	citem	r_sqr
1	poly	1	-0.0756997
2	poly	100	0.617957
3	poly	300	0.792641
4	poly	400	0.812933
5	poly	500	0.826368
6	poly	600	0.838446
7	poly	700	0.845331
8	poly	800	0.851926
9	poly	900	0.854766
10	poly	1000	0.856649
11	poly	1100	0.858369
12	poly	1200	0.858164
13	poly	1300	0.857776
14	poly	1400	0.857559
15	poly	1500	0.858089
16	poly	1600	0.858731
17	poly	1700	0.859291
18	poly	1800	0.860144
19	poly	1900	0.860443
20	poly	2000	0.860558
21	poly	2100	0.860359
22	poly	2200	0.859992
23	poly	2300	0.859835
24	poly	2400	0.859703
25	poly	2500	0.859532
26	poly	2600	0.8595
27	poly	2700	0.859483
28	poly	2800	0.859634
29	poly	2900	0.859943
30	poly	3000	0.859893
31	sigmoid	1	-0.0754292

32	sigmoid	100	0.52761
33	sigmoid	300	0.500073
34	sigmoid	400	0.512508
35	sigmoid	500	0.444606
36	sigmoid	600	0.408051
37	sigmoid	700	0.349038
38	sigmoid	800	0.29702
39	sigmoid	900	0.239545
40	sigmoid	1000	0.287471
41	sigmoid	1100	0.145949
42	sigmoid	1200	0.156555
43	sigmoid	1300	-0.0531944
44	sigmoid	1400	-0.138466
45	sigmoid	1500	-0.0674411
46	sigmoid	1600	-0.159746
47	sigmoid	1700	-0.263201
48	sigmoid	1800	-0.36851
49	sigmoid	1900	-0.474463
50	sigmoid	2000	-0.593951
51	sigmoid	2100	-0.733739
52	sigmoid	2200	-0.853878
53	sigmoid	2300	-0.973903
54	sigmoid	2400	-1.12089
55	sigmoid	2500	-1.28241
56	sigmoid	2600	-1.92053
57	sigmoid	2700	-1.58299
58	sigmoid	2800	-1.75781
59	sigmoid	2900	-1.9374
60	sigmoid	3000	-2.12442
61	rbf	1	-0.0833824
62	rbf	100	0.320032
63	rbf	300	0.558423
64	rbf	400	0.617553
65	rbf	500	0.664298
66	rbf	600	0.706767
67	rbf	700	0.739811
68	rbf	800	0.766578
69	rbf	900	0.794597
70	rbf	1000	0.810206
71	rbf	1100	0.818902
72	rbf	1200	0.829067
73	rbf	1300	0.836874
74	rbf	1400	0.840014

75	rbf	1500	0.842749
76	rbf	1600	0.845196
77	rbf	1700	0.848773
78	rbf	1800	0.851411
79	rbf	1900	0.8535
80	rbf	2000	0.854777
81	rbf	2100	0.856062
82	rbf	2200	0.857036
83	rbf	2300	0.857957
84	rbf	2400	0.858882
85	rbf	2500	0.860509
86	rbf	2600	0.86214
87	rbf	2700	0.863644
88	rbf	2800	0.864696
89	rbf	2900	0.865625
90	rbf	3000	0.866339

Best fit at [0. 866339] in the run- model combination is criterion:['rbf'], splitter:[3000].

2. Decision Tree

The same run is done for Decision Tree Algorithm.

splitter	max_features		r_sqr
squared_error	sqrt	random	0.718599
squared_error	log2	random	0.647903
squared_error	sqrt	best	0.757424
squared_error	log2	best	0.70505
friedman_mse	sqrt	random	0.7623
friedman_mse	log2	random	0.706207
friedman_mse	sqrt	best	0.767127
friedman_mse	log2	best	0.720833
poisson	sqrt	random	0.646175
poisson	log2	random	0.697868
poisson	sqrt	best	0.601276
poisson	log2	best	0.762722
absolute_error	sqrt	random	0.617812
absolute_error	log2	random	0.689508
absolute_error	sqrt	best	0.736536
absolute_error	log2	best	0.761641

Best fit at [0.76589155] in the run- model combination is criterion:[friedman_mse '], splitter:['best'],max_features:['sqrt']

3. Random Forest

The same run is done for Random Forest Algorithm.

S.No	criterion	max_features	r_sqr	estimators
0	squared_error	None	0.851359	50
1	squared_error	None	0.853264	60
2	squared_error	None	0.858279	70
3	squared_error	None	0.85346	80
4	squared_error	None	0.857558	90
5	squared_error	None	0.852955	100
6	squared_error	None	0.856526	110
7	squared_error	None	0.854473	120
8	squared_error	None	0.854951	130
9	squared_error	None	0.858771	140
10	squared_error	sqrt	0.872467	50
11	squared_error	sqrt	0.86849	60
12	squared_error	sqrt	0.868254	70
13	squared_error	sqrt	0.867346	80
14	squared_error	sqrt	0.872021	90
15	squared_error	sqrt	0.868102	100
16	squared_error	sqrt	0.871396	110
17	squared_error	sqrt	0.87067	120
18	squared_error	sqrt	0.870972	130
19	squared_error	sqrt	0.870757	140
20	squared_error	log2	0.868974	50
21	squared_error	log2	0.864864	60
22	squared_error	log2	0.868614	70
23	squared_error	log2	0.867219	80
24	squared_error	log2	0.870766	90
25	squared_error	log2	0.87499	100
26	squared_error	log2	0.870378	110
27	squared_error	log2	0.8701	120
28	squared_error	log2	0.869333	130
29	squared_error	log2	0.87201	140
30	friedman_mse	None	0.845878	50
31	friedman_mse	None	0.856008	60
32	friedman_mse	None	0.857622	70
33	friedman_mse	None	0.85485	80
34	friedman_mse	None	0.849373	90
35	friedman_mse	None	0.85107	100
36	friedman_mse	None	0.854495	110
37	friedman_mse	None	0.854327	120
38	friedman_mse	None	0.855816	130

39	friedman_mse	None	0.85149	140
40	friedman_mse	sqrt	0.866633	50
41	friedman_mse	sqrt	0.867697	60
42	friedman_mse	sqrt	0.871135	70
43	friedman_mse	sqrt	0.869833	80
44	friedman_mse	sqrt	0.871498	90
45	friedman_mse	sqrt	0.874995	100
46	friedman_mse	sqrt	0.868814	110
47	friedman_mse	sqrt	0.869751	120
48	friedman_mse	sqrt	0.871339	130
49	friedman_mse	sqrt	0.86953	140
50	friedman_mse	log2	0.867785	50
51	friedman_mse	log2	0.869957	60
52	friedman_mse	log2	0.870282	70
53	friedman_mse	log2	0.868461	80
54	friedman_mse	log2	0.872071	90
55	friedman_mse	log2	0.869482	100
56	friedman_mse	log2	0.872084	110
57	friedman_mse	log2	0.870138	120
58	friedman_mse	log2	0.870307	130
59	friedman_mse	log2	0.869929	140
60	poisson	None	0.851891	50
61	poisson	None	0.853231	60
62	poisson	None	0.857851	70
63	poisson	None	0.848579	80
64	poisson	None	0.857322	90
65	poisson	None	0.853926	100
66	poisson	None	0.853918	110
67	poisson	None	0.85537	120
68	poisson	None	0.856105	130
69	poisson	None	0.856574	140
70	poisson	sqrt	0.868009	50
71	poisson	sqrt	0.869477	60
72	poisson	sqrt	0.871563	70
73	poisson	sqrt	0.869119	80
74	poisson	sqrt	0.871501	90
75	poisson	sqrt	0.870982	100
76	poisson	sqrt	0.872983	110
77	poisson	sqrt	0.869688	120
78	poisson	sqrt	0.870896	130
79	poisson	sqrt	0.871778	140
80	poisson	log2	0.868462	50
81	poisson	log2	0.870311	60

82	poisson	log2	0.874466	70
83	poisson	log2	0.867044	80
84	poisson	log2	0.871084	90
85	poisson	log2	0.868574	100
86	poisson	log2	0.870431	110
87	poisson	log2	0.873073	120
88	poisson	log2	0.870597	130
89	poisson	log2	0.871306	140
90	absolute_error	None	0.858283	50
91	absolute_error	None	0.855132	60
92	absolute_error	None	0.854554	70
93	absolute_error	None	0.857253	80
94	absolute_error	None	0.858475	90
95	absolute_error	None	0.855005	100
96	absolute_error	None	0.854253	110
97	absolute_error	None	0.855244	120
98	absolute_error	None	0.856954	130
99	absolute_error	None	0.858468	140
100	absolute_error	sqrt	0.8691	50
101	absolute_error	sqrt	0.866835	60
102	absolute_error	sqrt	0.874655	70
103	absolute_error	sqrt	0.872464	80
104	absolute_error	sqrt	0.872449	90
105	absolute_error	sqrt	0.871549	100
106	absolute_error	sqrt	0.870702	110
107	absolute_error	sqrt	0.871385	120
108	absolute_error	sqrt	0.87201	130
109	absolute_error	sqrt	0.874504	140
110	absolute_error	log2	0.870912	50
111	absolute_error	log2	0.87234	60
112	absolute_error	log2	0.872317	70
113	absolute_error	log2	0.875277	80
114	absolute_error	log2	0.873868	90
115	absolute_error	log2	0.873613	100
116	absolute_error	log2	0.875175	110
117	absolute_error	log2	0.87408	120
118	absolute_error	log2	0.873815	130
119	absolute_error	log2	0.873533	140

Best fit at [0.875277] in the run- model combination is criterion:[' absolute_error'], max_features:['log2'], estimators:[80]

Best Algorithm is with [0.87632566] in the model ['RandomForestRegressor']

Conclusion: The final model will be the model created with Random Forest algorithm since the evaluation metric result shows a higher value at Best fit at [0.875277] in the run- model combination is criterion:[' absolute_error'], max_features:['log2'], estimators:[80].

The best model saved and will be deployed in the client place to fulfill the insurance company requirement.