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

We need to predict the insurance charges based on the given parameters in the dataset

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

Total number of columns:6

Total number of rows: 1338

Independent/Input: 5 columns (age, bmi, children, sex, smoker)

Dependent/Output: 1 column (charges)

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

I have used pd.get_dummies which converted strings into int (columns are sex and smoker, the yes, no values converted to 1, 0 respectively)

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, Multiple Linear Regression, Support vector machine, Decision tree, Random Forest algorithm with various parameter. Finally best model is Random Forest.

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

Please refer below.

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

At last, The best final Machine learning regression algorithm is Random forest with the parameter (n_estimators=100, criterion='squared_error', max_features='sqrt') = 0.8734 which is the highest accuracy among other models.

1.MULTIPLE LINEAR REGRESSION (R2 value) = 0.7865

2.SUPPORT VECTOR MACHINE (SVM)

Sl no	Regularization Parameter (C)	Linear (Kernal)	RBF – Non Linear (Kernal)	POLY (Kernal)	SIGMOID (Kernal)
1	Default 1.0	-0.1116	-0.0884	-0.0642	-0.0899
2	10	-0.0016	-0.0819	-0.0931	-0.090
3	100	0.5432	-0.1248	-0.0997	-0.1181
4	500	0.6270	-0.1246	-0.0820	-0.4562
5	1000	0.6340	-0.1174	-0.0555	-1.6659
6	2000	0.6893	-0.1077	-0.0027	-5.6164
7	5000	0.7648	-0.0731	0.1462	-31.5682
8	10000	0.7444	-0.0172	0.3529	-119.5185

3.DECISION TREE

Sl no	Criterion	Splitter	Max features	R Value
1	squared_error	Best	None	0.6954
2	squared_error	Random	None	0.6888
3	squared_error	Best	Sqrt	0.7127
4	squared_error	Random	Sqrt	0.6156
5	squared_error	Best	Log2	0.5587
6	squared_error	Random	Log2	0.7351
7	friedman_mse	Best	None	0.7351
8	friedman_mse	Random	None	0.7587
9	friedman_mse	Best	Sqrt	0.5721
10	friedman_mse	Random	Sqrt	0.6852
11	friedman_mse	Best	Log2	0.6729
12	friedman_mse	Random	Log2	0.7005
13	absolute_error	Best	None	0.6691
14	absolute_error	Random	None	0.7289
15	absolute_error	Best	Sqrt	0.6980
16	absolute_error	Random	Sqrt	0.6980
17	absolute_error	Best	Log2	0.6845
18	absolute_error	Random	Log2	0.7463
19	Poisson	Best	None	0.7251
20	Poisson	Random	None	0.6714
21	Poisson	Best	Sqrt	0.7060
22	Poisson	Random	Sqrt	0.6475
23	Poisson	Best	Log2	0.7273

24	poisson	Random	Log2	0.7063

3.RANDOM FOREST

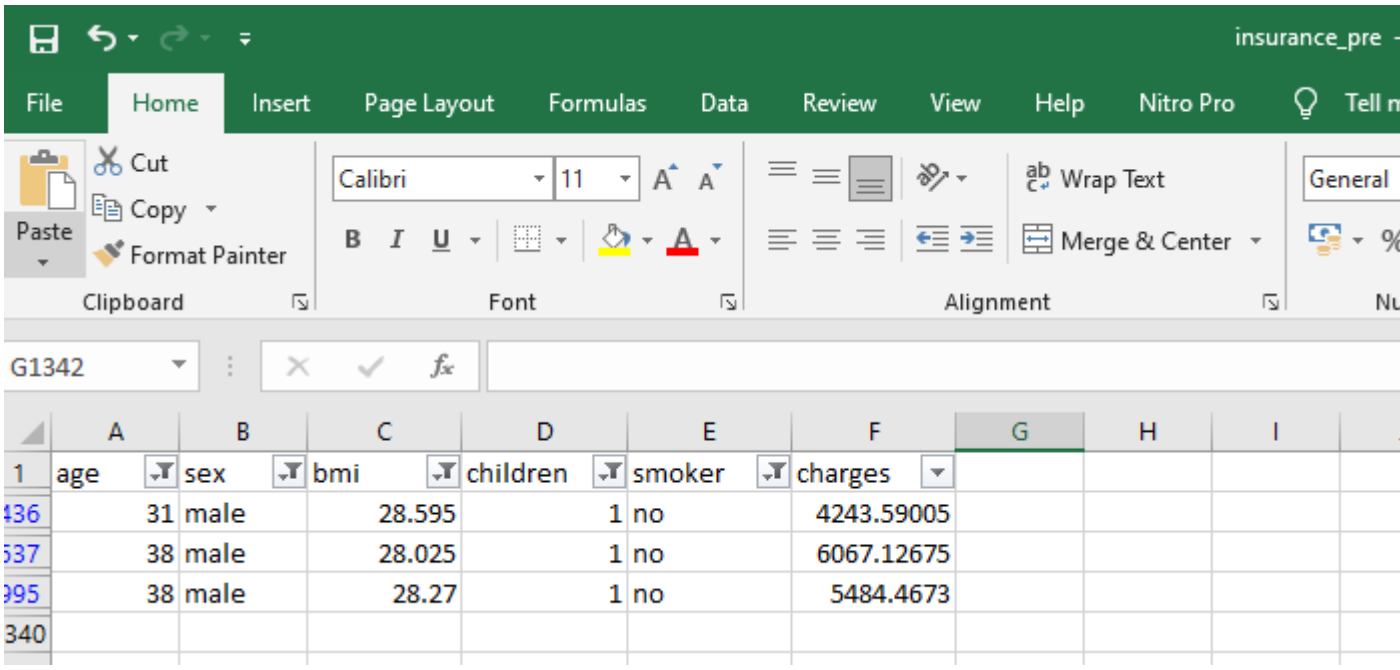
Sl no	N_Estimators	Criterion	Max features	R Value
1	50	squared_error	None	0.8509
2	100	squared_error	None	0.8556
3	50	squared_error	Sqrt	0.8689
4	100	squared_error	Sqrt	**0.8734**
5	50	squared_error	Log2	0.8642
6	100	squared_error	Log2	0.8698
7	50	friedman_mse	None	0.8525
8	100	friedman_mse	None	0.8519
9	50	friedman_mse	Sqrt	0.8695
10	100	friedman_mse	Sqrt	0.8697
11	50	friedman_mse	Log2	0.8697
12	100	friedman_mse	Log2	0.8704
13	50	absolute_error	None	0.8522
14	100	absolute_error	None	0.8542
15	50	absolute_error	Sqrt	0.8692
16	100	absolute_error	Sqrt	0.8731
17	50	absolute_error	Log2	0.8690
18	100	absolute_error	Log2	0.8706
19	50	Poisson	None	0.8459
20	100	Poisson	None	0.8492
21	50	Poisson	Sqrt	0.8652
22	100	Poisson	Sqrt	0.8698
23	50	Poisson	Log2	0.8706
24	100	poisson	Log2	0.8711

The best final Machine learning regression algorithm is Random forest with the parameter (n_estimators=100, criterion='squared_error', max_features='sqrt') = 0.8734

Deployment:

```
[1]: import pickle

[2]: loaded_model=pickle.load(open("finalized_model_insurance_forest.sav",'rb'))
     result=loaded_model.predict([[35, 1, 28.800, 1, 0]])  #['age', 'sex_male', 'bmi',

     C:\Anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have
     es
       warnings.warn(

[3]: result

[3]: array([6033.6332203])
```

Manual comparison with the dataset for the nearest given input, it is close match!



	A	B	C	D	E	F
1	age	sex	bmi	children	smoker	charges
436	31	male	28.595	1	no	4243.59005
537	38	male	28.025	1	no	6067.12675
995	38	male	28.27	1	no	5484.4673
340						