Assignment-Regression Algorithm

1.) Identify your problem statement

Problem Statement

A client requires a predictive model to estimate **insurance charges** based on several input parameters. The client has provided a dataset containing historical records of customers and their corresponding insurance charges.

As a data scientist, my task is to develop a machine learning model that accurately predicts the insurance charges using the given features.

Prediction Goal

Insurance Charge Prediction

1. Stage 1 – domain selection: Machine Learning
2. Stage 2 – learning type: Supervised learning
3. Stage 3 : Regression

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

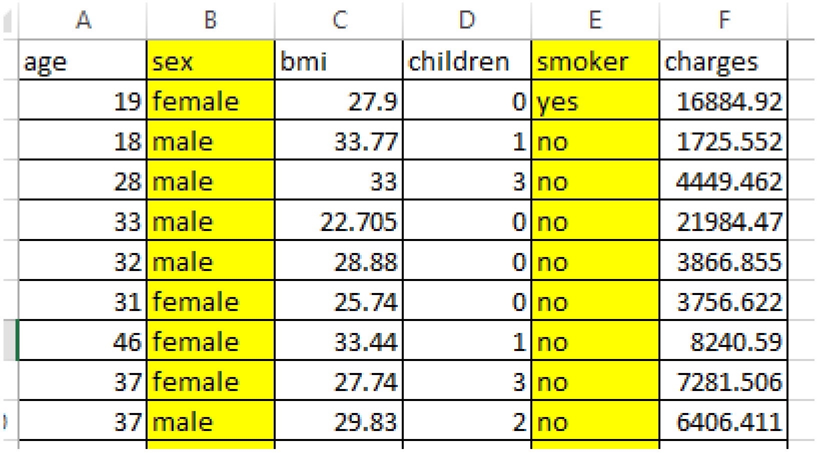
Dataset Details

- **Number of rows:** 1338
- **Number of columns:** 6

Column Details

Column Name	Data Type
Age	Integer
Sex	String
BMI	Integer
Children	Integer
Smoker	String
Charges	Integer

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



A	B	C	D	E	F
age	sex	bmi	children	smoker	charges
19	female	27.9	0	yes	16884.92
18	male	33.77	1	no	1725.552
28	male	33	3	no	4449.462
33	male	22.705	0	no	21984.47
32	male	28.88	0	no	3866.855
31	female	25.74	0	no	3756.622
46	female	33.44	1	no	8240.59
37	female	27.74	3	no	7281.506
37	male	29.83	2	no	6406.411

Encoding Categorical Data:

- I need to convert 2 columns, sex and smoker, as they are in string format, into numbers.
- As both the data in the columns sex and smoker are ordered, we can say that they are Categorical - ordinal data.
- The ordinal data can be converted into numerical data using the ‘mapping – label encoder’ method.
- During this process, there is no column expansion, and the data can be compared or ordered using this method.
- In Python code, this can be achieved using the pandas.get_dummies () function.

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.

- *My final model is a Random Forest Regressor.*
- *The model accuracy is 0.875772611487943.*

5.) All the research values (r2_score of the models):-

SLR – Simple Linear Regression Algorithm

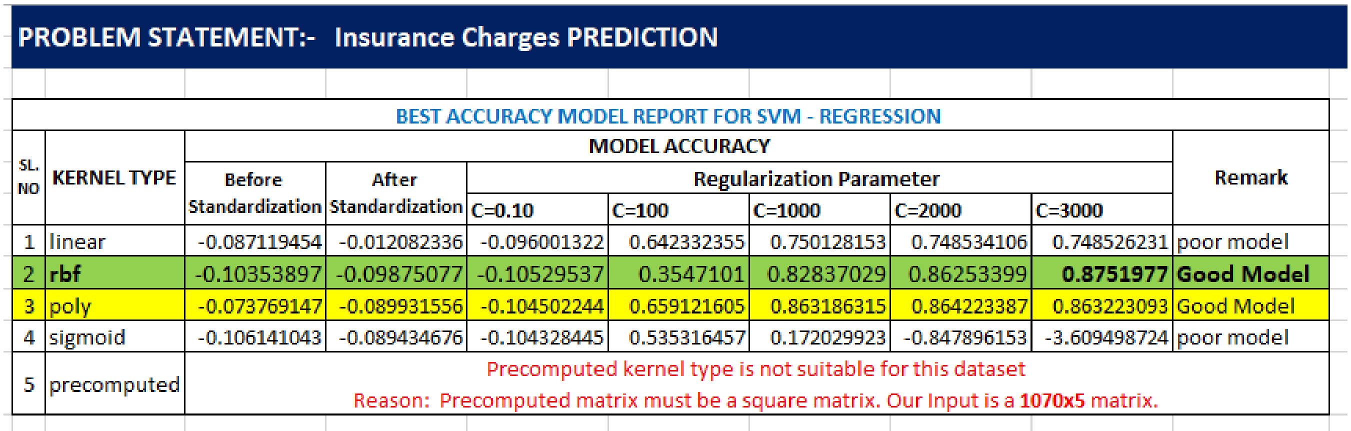
I am not trying SLR because it requires one input and one output, and this dataset does not satisfy that condition.

MLR – Multiple Linear Regression Algorithm

- Next, I tried MLR – Multiple Linear Regression Algorithm. Since the dataset has multiple inputs and one output.
- MLR does not have hyper parameters to tune.
- The accuracy of the model Insurance charge prediction using MLR is **0.7978644236809905**.

SVM

- Since the MLR accuracy is only 79%, I tried using the SVM algorithm to improve the model's performance.
- With the initial SVM model training, I did not get much improvement, so I applied standardization to enhance the model. However, even after standardization, there was still no significant improvement in accuracy.
- Therefore, I increased the **C (regularization) parameter** for each kernel type and fine-tuned the model to achieve better performance.
- Finally, I improved the accuracy to **87%** using the SVM with the **RBF kernel** and **C = 3000**, after applying standardization.



PROBLEM STATEMENT:- Insurance Charges PREDICTION

BEST ACCURACY MODEL REPORT FOR SVM - REGRESSION

SL. NO	KERNEL TYPE	MODEL ACCURACY							Remark
		Before Standardization	After Standardization	Regularization Parameter					
				C=0.10	C=100	C=1000	C=2000	C=3000	
1	linear	-0.087119454	-0.012082336	-0.096001322	0.642332355	0.750128153	0.748534106	0.748526231	poor model
2	**rbf**	-0.10353897	-0.09875077	-0.10529537	0.3547101	0.82837029	0.86253399	**0.8751977**	**Good Model**
3	poly	-0.073769147	-0.089931556	-0.104502244	0.659121605	0.863186315	0.864223387	0.863223093	Good Model
4	sigmoid	-0.106141043	-0.089434676	-0.104328445	0.535316457	0.172029923	-0.847896153	-3.609498724	poor model
5	precomputed	Precomputed kernel type is not suitable for this dataset Reason: Precomputed matrix must be a square matrix. Our Input is a **1070x5** matrix.							

Decision Tree

- I tried to improve the model for the problem statement of **insurance charges prediction** using a Decision Tree Regressor.

- I applied hyperparameter tuning to find the best-performing model. The parameters I tuned include:

  - **criterion:** {"squared_error", "friedman_mse", "absolute_error", "poisson"}
  - **splitter:** {"best", "random"}
  - **max_features:** int, float, or {"sqrt", "log2"}

- I achieved the best accuracy with the following parameter combinations:

  - criterion = "absolute_error", splitter = "best", max_features = "sqrt" → **78% accuracy**
  - criterion = "absolute_error", splitter = "best", max_features = "log2"→ **78% accuracy**

- However, the accuracy obtained from the Decision Tree model is lower (**78% accuracy**) compared to the Support Vector Regression (SVR) model, which achieved **87% accuracy**.
- Therefore, we can conclude that, as of now, the **SVR model (with 87% accuracy)** performs best for predicting insurance charges.

PROBLEM STATEMENT:- Insurance Charges PREDICTION

BEST ACCURACY MODEL REPORT FOR "DECISION TREE" - REGRESSION

SL.NO	criterion	splitter	max features	model accuracy	model accuracy (random_state=0)	Remark
1	squared_error	best	None	0.732653243	0.724268339	poor model
2	friedman_mse	best	None	0.746369699	0.72384961	poor model
3	absolute_error	best	None	0.701381117	0.709543497	poor model
4	poisson	best	None	0.749663107	0.74389867	poor model
5	squared_error	best	sqrt	0.737260837	0.716312344	poor model
6	friedman_mse	best	sqrt	0.749952938	0.71221531	poor model
7	absolute_error	best	sqrt	0.760277846	**0.788552428**	Good Model
8	poisson	best	sqrt	0.766723614	0.719931236	poor model
9	squared_error	best	log2	0.682091851	0.716312344	poor model
10	friedman_mse	best	log2	0.746791651	0.71221531	poor model
11	absolute_error	best	log2	0.743965353	**0.788552428**	Good Model

12	poisson	best	log2	0.715378919	0.719931236	poor model
13	squared_error	random	None	0.659774387	0.732395784	poor model
14	friedman_mse	random	None	0.694201328	0.725476019	poor model
15	absolute_error	random	None	0.795645357	**0.771653116**	Good Model
16	poisson	random	None	0.745256989	0.726637592	poor model
17	squared_error	random	sqrt	0.765237207	0.744620401	poor model
18	friedman_mse	random	sqrt	0.756965818	0.736780068	poor model
19	absolute_error	random	sqrt	0.698426288	0.666400496	poor model
20	poisson	random	sqrt	0.749841512	0.732279439	poor model
21	squared_error	random	log2	0.743742731	0.744620401	poor model
22	friedman_mse	random	log2	0.721297251	0.736780068	poor model
23	absolute_error	random	log2	0.708555001	0.666400496	poor model
24	poisson	random	log2	0.619731456	0.732279439	poor model

Random forest

- I tried to improve the model using a Random Forest Regressor by tuning the hyperparameters

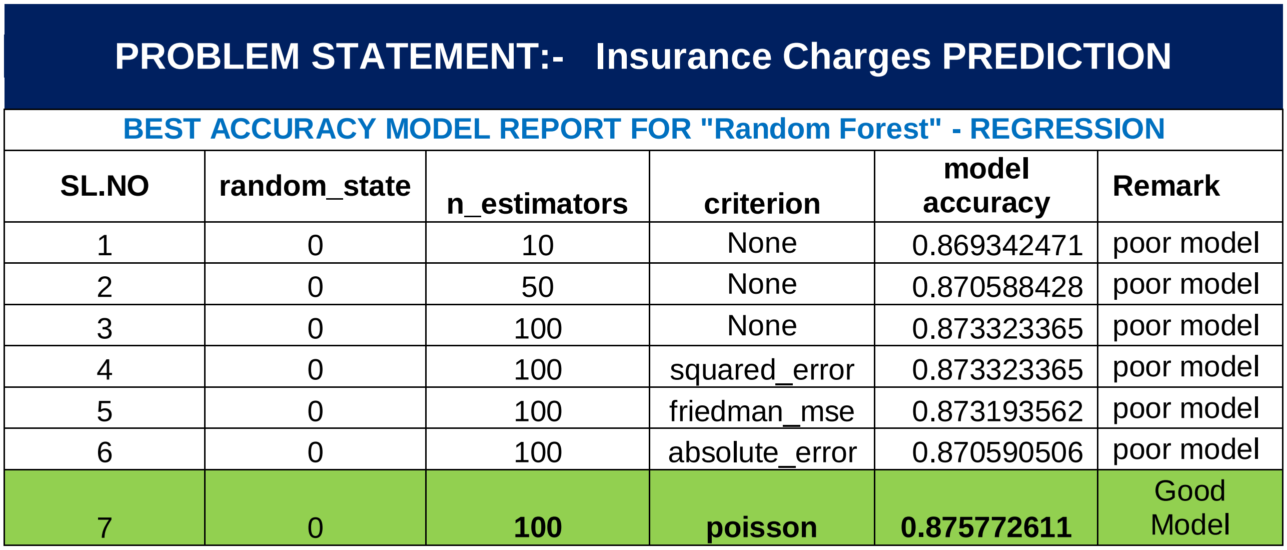
  n_estimators,
  criterion {"squared_error", "friedman_mse", "absolute_error", "poisson"}, and
  random_state = 0.

- I achieved better accuracy using the following parameters:

  - n_estimators = 100
  - criterion = "poisson"
  - random_state = 0

- With this configuration, the model achieved an accuracy of **87.577%**.

- I conclude that this is the best-performing model so far, as it is slightly better than the SVR model accuracy of **87.519%**.



PROBLEM STATEMENT:- Insurance Charges PREDICTION

BEST ACCURACY MODEL REPORT FOR "Random Forest" - REGRESSION

SL.NO	random_state	n_estimators	criterion	model accuracy	Remark
1	0	10	None	0.869342471	poor model
2	0	50	None	0.870588428	poor model
3	0	100	None	0.873323365	poor model
4	0	100	squared_error	0.873323365	poor model
5	0	100	friedman_mse	0.873193562	poor model
6	0	100	absolute_error	0.870590506	poor model
7	0	**100**	**poisson**	**0.875772611**	Good Model

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

The final model is the Random Forest Regressor, as it achieved higher accuracy compared to the other algorithms. Therefore, I conclude that this model performs well for the given problem statement—insurance charges prediction.

I am finalizing the Random Forest model because, after evaluating multiple algorithms (MLR, SVM - SVR, Decision Tree, and Random Forest Regressor), the Random Forest Regressor produced the best accuracy of ***87.577%***. It performed slightly better than the SVR model and showed improved stability after hyper parameter tuning.