To find the following Machine Learning Regression using r2 value

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.

Sample dataset: "insurance_pre.csv"

As a data scientist, you must develop a model which will predict the insurance charges.

1.) **Identify your problem statement**
   - ➢ **Machine Learning – (predict the insurance charge – numbers)**
   - ➢ **Supervised Learning – (we have input and output)**
   - ➢ **Regression Type (Numerical value)**

2.) **Tell basic info about the dataset (Total number of rows, columns)**
   - ➢ **6 columns (5 columns is input and 1 column is output)**
   - ➢ **No.of rows is 1338 (exclude column header)**
   - ➢ **Input column is (age, sex, bmi, children, smoker)**
   - ➢ **Output column is (charges)**

3.) **Mention the pre-processing method if you're doing any (like converting string to number – nominal data)**
   - ➢ **We are using Standardization(Pre-processing, StandardScaler) method**

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'm going with random forest comparing with other model. Execution time is less and more productivity to other model.**
   - ➢ **Accuracy of this model is 0.8 not equal to 1.0 but compare to others. It's ok for probability decision.**

5.) **R2_score of the models:**

1. **Multiple Linear Regression: {r2 value = 0.78947}**

2. **SVM – Support Vector Machine Regression:**

   Kernal = {'linear', 'rbf', 'poly', 'sigmoid'}

S.NO	HYPER PARAMETER	LINEAR (r value)	RBF (NON-LINEAR) (r value)	POLY (r value)	SIGMOID (r value)

1	C=10	0.4624	-0.03227	0.0387	0.0393
2	C=100	0.62887	0.3200	0.61795	0.52761
3	C=500	0.76310	0.66429	0.8263	0.4446
4	C=1000	0.76493	0.81020	0.8566	0.2874
5	C=2000	0.7440	0.8547	0.8605	-0.5939
6	C=3000	0.74142	0.8663	0.8598	-2.1244

SVM Regression - *R^2 value* (nonlinear "rbf" & hyper parameter C=3000) = 0.8663

3. Decision Tree Regressor:

Criterion List = *{"squared_error", "friedman_mse", "absolute_error", "poisson"}*

Mse = Mean squared error

Mae = Mean absolute error

S.NO	**CRITERION LIST**	**MAX FEATURES**	**SPLITTER**	**R VALUE**
1	squared_error	None	*best*	0.6880
2	squared_error	None	*random*	0.6775
3	squared_error	*sqrt*	*best*	0.7421
4	squared_error	*sqrt*	*random*	0.7011
5	squared_error	*log2*	*best*	0.7282
6	squared_error	*log2*	*random*	0.6959
7	Mae	None	*best*	0.6744
8	Mae	None	*random*	0.6869
9	Mae	*sqrt*	*best*	0.7625
10	Mae	*sqrt*	*random*	0.6742
11	Mae	*log2*	*best*	0.75607
12	Mae	*log2*	*random*	0.72709
13	friedman_mse	None	*best*	0.6923
14	friedman_mse	None	*random*	0.6815
15	friedman_mse	*sqrt*	*best*	0.6582
16	friedman_mse	*sqrt*	*random*	0.6977
17	friedman_mse	*log2*	*best*	0.67605
18	friedman_mse	*log2*	*random*	0.7107
19	poisson	None	*best*	0.7319
20	poisson	None	*random*	0.7541
21	poisson	*sqrt*	*best*	0.7231
22	poisson	*sqrt*	*random*	0.6900

23	poisson	*log2*	*best*	0.6037
24	poisson	*log2*	*random*	0.6712

Decision Tree Regressor - *R^2 value* (Criterion list = "absolute_error", Max feature = "sqrt", Splitter=best) = 0.7625

4. **Random Forest:**

Criterion List = *{"squared_error", "friedman_mse", "absolute_error", "poisson"}*

S.NO	**N_ESTIMATORS**	**CRITERION LIST**	**MAX FEATURES**	**R VALUE**
1	*50*	squared_error	None	0.85091
2	*50*	squared_error	*sqrt*	0.86961
3	*50*	squared_error	*log2*	0.8696
4	*50*	Mae	None	0.85412
5	*50*	Mae	*sqrt*	0.87168
6	*50*	Mae	*log2*	0.87168
7	*50*	friedman_mse	None	0.85111
8	*50*	friedman_mse	*sqrt*	0.87023
9	*50*	friedman_mse	*log2*	0.87023
10	*50*	poisson	None	0.85032
11	*50*	poisson	*sqrt*	0.86320
12	*50*	poisson	*log2*	0.86320

Random Forest Regressor - *R^2 value* (Criterion list = "Mae", Max feature = "sqrt or log2") = 0.87168

6.) Random Forest gives the best result compare to other model.

- **Execution time is less and compare to other model accuracy is considerable**