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 -

To develop a machine learning regression model that can accurately predict **insurance charges** based on input features such as age, sex, BMI, number of children, smoking status, and region.

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

The dataset has 1338 rows and , 7 columns.

3.) Mention the pre-processing method if you’re doing any (like converting

string to number – nominal data)

The data standardization has been done by transforming x\_train & x\_test using StandardScaler class.

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.

The best model created is Random Forest Regressor with hyper tuning parameter (n\_estimators=50,criterion='absolute\_error',max\_features='sqrt'). The R2\_score of prediction accuracy is .875

5.) All the research values (r2\_score of the models) should be documented.

Research values of all 5 algorithms are documented and uploaded in the github.

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

The model mentioned in point 5 is the best model to choose as this has .875 accuracy compare to all other algorithms & parameters.

Kindly created the Regression Repository and added all dependent files.

Assignment.

Upload all the ipynb and final document in the pdf

Communication is important (How you are representing the

document.)

Output Values

1. Simple Linear Regression

Predicted R2 score - .110

2. Multiple Linear Regression

Predicted R2 Score - .789

3. Support Vector Machine

To identify models accuracy while using different hyper parameters for Random Forest Regressor

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | kernel | gamma | R Value |
| 1 | linear | scale | -0.010 |
| 2 | linear | auto | -0.010 |
| 3 | poly | scale | -0.075 |
| 4 | poly | auto | -0.075 |
| 5 | rbf | scale | -0.083 |
| 6 | rbf | auto | -0.083 |
| 7 | sigmoid | scale | -0.075 |
| 8 | sigmoid | auto | -0.075 |
| 9 | precomputed | scale | NA(getting error) |
| 10 | precomputed | auto | NA(getting error) |

4. Decision Tree

Decision Tree:

To identify models accuracy while using different hyper parameters for Decision Tree Regressor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | criterion | splitter | max\_features | R Value |
| 1 | squared\_error | best | sqrt | .556 |
| 2 | squared\_error | best | Log2 | .706 |
| 3 | squared\_error | random | sqrt | .689 |
| 4 | squared\_error | random | Log2 | .659 |
| 5 | absolute\_error | best | sqrt | .743 |
| 6 | absolute\_error | best | Log2 | .787 |
| 7 | absolute\_error | random | sqrt | .655 |
| 8 | absolute\_error | random | Log2 | .639 |
| 9 | poisson | best | sqrt | .733 |
| 10 | poisson | best | Log2 | .725 |
| 11 | poisson | random | sqrt | .603 |
| 12 | poisson | random | Log2 | .720 |
| 13 | friedman\_mse | best | sqrt | .656 |
| 14 | friedman\_mse | best | Log2 | .606 |
| 15 | friedman\_mse | random | sqrt | .654 |
| 16 | friedman\_mse | random | Log2 | .661 |
| 17 | squared\_error | best | None | .728 |
| 18 | squared\_error | random | None | .748 |
| 19 | absolute\_error | best | None | .698 |
| 20 | absolute\_error | random | None | .687 |
| 21 | poisson | best | None | .714 |
| 22 | poisson | random | None | .636 |
| 23 | friedman\_mse | best | None | .704 |
| 24 | friedman\_mse | random | None | .710 |

5. Random Forest

To identify models accuracy while using different hyper parameters for Random Forest Regressor

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | criterion | Max\_features | R Value |
| 1 | squared\_error | sqrt | .870 |
| 2 | squared\_error | Log2 | .868 |
| 3 | friedman\_mse | sqrt | .873 |
| 4 | friedman\_mse | Log2 | .871 |
| 5 | absolute\_error | sqrt | .875 |
| 6 | absolute\_error | Log2 | .874 |
| 7 | poisson | sqrt | .871 |
| 8 | poisson | Log2 | .871 |
| 9 | squared\_error | None | .851 |
| 10 | friedman\_mse | None | .846 |
| 11 | absolute\_error | None | .854 |
| 12 | poisson | None | .849 |