Assignment-Regression Algorithm

**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

Stage 1: Machine Learning

Stage 2: Supervised Learning

Stage 3: Regression

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

(1338, 6)

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

Used pd.get\_dummies for converting categorical column (Nominal) to numerical column(One Hot Encoding)

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.

5.) All the research values (r2\_score of the models) should be documented. (You can make tabulation or screenshot of the results.)

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

**Assignment 🡪 insurance\_pre**

1. Multiple Linear Regression - R2\_Score = 0.7894790349867009
2. Support Vector Machine –

Kernal = “rbf” & R2\_Score = -0.11166128719608448 (without Standardization)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Hyper tuning | rbf(R2\_Score) | linear(R2\_Score) | poly(R2\_Score) | sigmoid(R2\_Score) |
| 1 | C=100 | 0.320031 | 0.628879 | 0.617956 | 0.527610 |
| 2 | C=1000 | 0.810206 | 0.764931 | 0.856648 | 0.287470 |
| 3 | C=2000 | 0.854776 | 0.744041 | 0.860557 | -0.593950 |
| 4 | C=3000 | 0.866339 | 0.741423 | 0.859893 | -2.124419 |
| 5 | C=4000 | 0.871740 | 0.741419 | 0.860004 | -5.510333 |

Good performance:

Parameter = rbf

Hyper Tuning = 4000

R2\_Score = 0.871740

1. Decision Tree:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **criterion** | **splitter** | **max\_features** | **R2\_Score** |
| 1 | squared\_error | best | sqrt | 0.760596 |
| 2 | squared\_error | best | log2 | 0.736013 |
| 3 | squared\_error | random | sqrt | 0.702131 |
| 4 | squared\_error | random | log2 | 0.733725 |
| 5 | friedman\_mse | best | sqrt | 0.725680 |
| 6 | friedman\_mse | best | log2 | 0.742910 |
| 7 | friedman\_mse | random | sqrt | 0.692897 |
| 8 | friedman\_mse | random | log2 | 0.685652 |
| 9 | absolute\_error | best | sqrt | 0.744082 |
| 10 | absolute\_error | best | log2 | 0.681287 |
| 11 | absolute\_error | random | sqrt | 0.759051 |
| 12 | absolute\_error | random | log2 | 0.564800 |
| 13 | poisson | best | sqrt | 0.699569 |
| 14 | poisson | best | log2 | 0.731877 |
| 15 | poisson | random | sqrt | 0.627572 |
| 16 | poisson | random | log2 | 0.666446 |
| 17 | squared\_error | best | - | 0.696948 |
| 18 | squared\_error | random | - | 0.642064 |
| 19 | friedman\_mse | best | - | 0.696290 |
| 20 | friedman\_mse | random | - | 0.674220 |
| 21 | absolute\_error | best | - | 0.672453 |
| 22 | absolute\_error | random | - | 0.711783 |
| 23 | poisson | best | - | 0.721151 |
| 24 | poisson | random | - | 0.785460 |

Good performance:

criterion = poisson

splitter = random

R2\_Score = 0.785460

1. Random Forest:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **n\_estimators** | **criterion** | **max\_features** | **R2\_Score** |
| 1 | 50 | - | - | 0.849832 |
| 2 | 100 | - | - | 0.853830 |
| 3 | 50 | squared\_error | - | 0.849832 |
| 4 | 50 | friedman\_mse | - | 0.850071 |
| 5 | 50 | absolute\_error | - | 0.852665 |
| 6 | 50 | poisson | - | 0.849107 |
| 7 | 100 | squared\_error |  | 0.853830 |
| 8 | 100 | friedman\_mse |  | 0.854051 |
| 9 | 100 | absolute\_error |  | 0.852009 |
| 10 | 100 | poisson |  | 0.852633 |
| 11 | 50 | - | sqrt | 0.869583 |
| 12 | 100 | - | sqrt | 0.871027 |
| 13 | 50 | - | log2 | 0.869583 |
| 14 | 100 | - | log2 | 0.871027 |
| 15 | 50 | squared\_error | sqrt | 0.869583 |
| 16 | 100 | squared\_error | sqrt | 0.871027 |
| 17 | 50 | squared\_error | log2 | 0.869583 |
| 18 | 100 | squared\_error | log2 | 0.871027 |
| 19 | 50 | friedman\_mse | sqrt | 0.870241 |
| 20 | 100 | friedman\_mse | sqrt | 0.871054 |
| 21 | 50 | friedman\_mse | log2 | 0.870241 |
| 22 | 100 | friedman\_mse | log2 | 0.871054 |
| 23 | 50 | absolute\_error | sqrt | 0.870814 |
| 24 | 100 | absolute\_error | sqrt | 0.871068 |
| 25 | 50 | absolute\_error | log2 | 0.870814 |
| 26 | 100 | absolute\_error | log2 | 0.871068 |
| 27 | 50 | poisson | sqrt | 0.863239 |
| 28 | 100 | poisson | sqrt | 0.868015 |
| 29 | 50 | poisson | log2 | 0.863239 |
| 30 | 100 | poisson | log2 | 0.868015 |

Good performance:

n\_estimators: 100

criterion: absolute\_error

max\_features: sqrt

R2\_Score: 0.871068

**Final:**

1. Multiple Linear Regression - R2\_Score = 0.789479
2. Support Vector Machine - R2\_Score = 0.871740
3. Decision Tree - R2\_Score = 0.785460
4. Random Forest - R2\_Score = 0.871068

The best Model is Support Vector Machine which gave R2\_score = 0.871740 which is performing good comparing to other model.