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.

**1.Multiple Linear Regression:**

* R2\_score=0.7894790349867009

**2.Support Vector Machine:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.no | Hyper Tuning | Linear(R2\_score) | rbf(R2\_score) | poly(R2\_score) | Sigmoid(R2\_score) |
| 1 | c=100 | 0.628879 | 0.320031 | 0.617956 | 0.5276103 |
| 2 | c=500 | 0.763105 | 0.664298 | 0.826368 | 0.4446061 |
| 3 | c=1000 | 0.764931 | 0.810206 | 0.856648 | 0.2874706 |
| 5 | c=2000 | 0.744041 | 0.854776 | 0.860557 | -0.593950 |
| 7 | c=3000 | 0.741423 | 0.866339 | 0.859893 | -2.124419 |

* Best performance in SVM: c=3000,’rbf’=0.866339

**3.Decision Tree:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s.no | criterion | splitter | Max\_Feautures | R2\_score |
| 1 | Squared\_error | best | None | 0.7093221346551533 |
| 2 | Squared\_error | best | sqrt | 0.48338064793824176 |
| 3 | Squared\_error | best | Log2 | 0.6262428443958453 |
| 4 | Squared\_error | random | None | 0.6982581215239171 |
| 5 | Squared\_error | random | Sqrt | 0.687303572460628 |
| 6 | Squared\_error | random | Log2 | 0.6877535989689072 |
| 7 | friedman\_mse | Best | None | 0.7044361465763099 |
| 8 | friedman\_mse | Best | sqrt | 0.7296226489011768 |
| 9 | friedman\_mse | Best | Log2 | 0.7505324447518632 |
| 10 | friedman\_mse | Random | None | 0.6602039364578351 |
| 11 | friedman\_mse | Random | sqrt | 0.647008146103017 |
| 12 | friedman\_mse | Random | Log2 | 0.6372276198929773 |
| 13 | absolute\_error | Best | None | 0.6428214847446903 |
| 14 | absolute\_error | Best | sqrt | 0.5910422412166882 |
| 15 | absolute\_error | Best | Log2 | 0.6491772298605141 |
| 16 | absolute\_error | Random | None | 0.6857673154832085 |
| 17 | absolute\_error | Random | sqrt | 0.6579496771955842 |
| 18 | absolute\_error | Random | Log2 | 0.7256034329505419 |
| 19 | poisson | Best | None | 0.7277291714101468 |
| 20 | poisson | Best | sqrt | 0.7376011212328613 |
| 21 | poisson | Best | Log2 | 0.7378093307747176 |
| 22 | poisson | Random | None | 0.6853361230906097 |
| 23 | poisson | Random | sqrt | 0.6799813589572441 |
| 24 | poisson | Random | Log2 | 0.6487326317906484 |

Best Performance: friedman\_mse, best,log2= 0.7505324447518632

4. Random Forest:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s.no | n\_estimators | criterion | Max\_Feautures | R2\_score |
| 1 | 50 | Squared\_error | None | 0.8498329315421834 |
| 2 | 100 | Squared\_error | None | 0.8538307913484513 |
| 3 | 50 | Squared\_error | Sqrt | 0.8695836787761578 |
| 4 | 100 | Squared\_error | Sqrt | 0.8710271903471005 |
| 5 | 50 | Squared\_error | Log2 | 0.8695836787761578 |
| 6 | 100 | Squared\_error | Log2 | 0.8710271903471005 |
| 7 | 50 | friedman\_mse | None | 0.8500716139332296 |
| 8 | 100 | friedman\_mse | None | 0.8540518935149612 |
| 9 | 50 | friedman\_mse | Sqrt | 0.8702417511198071 |
| 10 | 100 | friedman\_mse | Sqrt | 0.8710544015500664 |
| 11 | 50 | friedman\_mse | Log2 | 0.8702417511198071 |
| 12 | 100 | friedman\_mse | Log2 | 0.8710544015500664 |
| 13 | 50 | absolute\_error | None | 0.8526655993519747 |
| 14 | 100 | absolute\_error | None | 0.8520093621081837 |
| 15 | 50 | absolute\_error | Sqrt | 0.8708144250343052 |
| 16 | 100 | absolute\_error | Sqrt | 0.8710685856341518 |
| 17 | 50 | absolute\_error | Log2 | 0.8708144250343052 |
| 18 | 100 | absolute\_error | Log2 | 0.8710685856341518 |
| 19 | 50 | poisson | None | 0.8491075958392151 |
| 20 | 100 | poisson | None | 0.8526334258892607 |
| 21 | 50 | poisson | Sqrt | 0.8632391369285537 |
| 22 | 100 | poisson | Sqrt | 0.8680156984764337 |
| 23 | 50 | poisson | Log2 | 0.8632391369285537 |
| 24 | 100 | poisson | Log2 | 0.8680156984764337 |

Best performance: (100,absolute\_error,sqrt&log2)= 0.8710685856341518

6. Comparing all 4 R2\_score:

|  |  |  |
| --- | --- | --- |
| S.No | Models | R2\_score |
| 1 | Multiple\_linear | 0.7894790349867009 |
| 2 | SVR | 0.866339 |
| 3 | Decision\_Tree | 0.7505324447518632 |
| 4 | Random\_forest | 0.8710685856341518 |

The best Model is Random\_forest which gave R2\_score=0.8710685856341518 which is performing good comparing to other model.