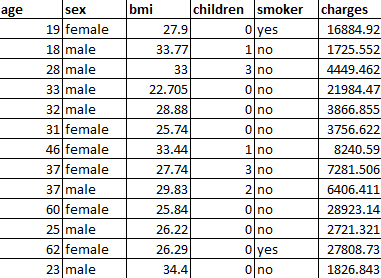
**Assignment – Regression Algorithm**

**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.

**Solution :**

**1.) Identify your problem statement**

* Stage 1: **Machine Learning** because input has majority of excel numerical data
* Stage 2: **Supervision** because we got clear input and output datas
* Stage 3: **Regression** because we have to find out numerical data output of insurance charges not classification.

**2.) Tell basic info about the dataset**

We have totally 1338 inputs (Rows) with different of 4 numerical and 2 categorical ( totally 6 columns) input data.

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

**string to number – nominal data)**

* Yes, we have to preprocess the input data that categorical input of Sex and Smoker column to be convert into meaningful number.
* Because Python cannot handle categorical input like Sex and Smoker as input.
* So here, Age, Sex, BMI, Childern, Smoker are (Independent variable) inputs and

Charges is (Dependent variable) output

**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.)**

1. Multiple Linear Regression: **R² value is 0.7894**
2. Support vector Machine:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Hyper Parameter** | **Linear (R value)** | **RBF (Non Linear R value)** | **Poly**  **(R value)** | **SIGMOID (R Value)** |
| 1 | C10 | 0.46 | -0.03 | 0.03 | 0.03 |
| 2 | C100 | 0.62 | 0.32 | 0.61 | 0.52 |
| 3 | C500 | 0.76 | 0.66 | 0.82 | 0.44 |
| 4 | C1000 | 0.76 | 0.81 | 0.85 | 0.28 |
| 5 | C2000 | 0.74 | 0.85 | 0.86 | -0.59 |
| **6** | **C5000** | **0.74** | **0.87** | **0.86** | **-7.53** |
| 7 | C10000 | 0.74 | 0.87 | 0.86 | -34.15 |

* **Nonlinear (rbf)** with Hyper tuning parameter of **C=5000** is best model from SVM algorithm as highlighted above **R² value is 0.87**

3. Decision Tree:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CRITERION** | **MAX FEATURE** | **SPLITTER** | **R Value** |
| **1** | **squared\_error** | **sqrt** | **best** | **0.78** |
| 2 | squared\_error | sqrt | random | 0.60 |
| 3 | squared\_error | Log2 | best | 0.74 |
| 4 | squared\_error | Log2 | random | 0.72 |
| 5 | Friedman\_mse | sqrt | best | 0.76 |
| 6 | Friedman\_mse | sqrt | random | 0.67 |
| 7 | Friedman\_mse | sqrt | best | 0.69 |
| 8 | Friedman\_mse | Log2 | best | 0.70 |
| 9 | Friedman\_mse | Log2 | random | 0.67 |

* Decision tree regression use **R² value (squared\_error, sqrt , best) = is 0.78**

4. Random Forest:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **CRITERION** | **MAX FEATURE** | **N\_ESTIMATORS** | **R Value** |
| 1 | squared\_error | sqrt | 10 | 85 |
| 2 | squared\_error | log2 | 10 | 85 |
| 3 | squared\_error | sqrt | 50 | 87 |
| 4 | squared\_error | log2 | 50 | 86 |
| 5 | absolute\_error | sqrt | 10 | 86 |
| 6 | absolute\_error | log2 | 10 | 86 |
| **7** | **absolute\_error** | **sqrt** | **50** | **88** |
| 8 | absolute\_error | Log2 | 50 | 87 |
| 9 | friedman\_mse | sqrt | 10 | 85 |
| 10 | friedman\_mse | log2 | 10 | 85 |
| 11 | friedman\_mse | sqrt | 50 | 87 |
| 12 | friedman\_mse | Log2 | 50 | 87 |

* Random forest regression use **R² value (absolute error, sqrt , N-estimator=50) = is 0.88**

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

* **Support vector machine and Random forest are giving best reslt**
* **Nonlinear (rbf)** with Hyper tuning parameter of **C=5000** is best model from SVM algorithm as highlighted above **R² value is 0.87**
* **R** Random forest regression use **R² value (absolute error, sqrt , N-estimator=50) is 0.88**