**1.Problem Statement:**

Predict the insurance charges based on the given input&create a best machine learning regression module.

**2.Basic info about the dataset:**

**Coulumns : 6**

**Rows : 1338**

**3.Dataset pre-processiong method:**

AI algorithm doesn’t work for catagorical value .so the given input had a catagorical value,from the input catagorical value is nominal data.String converted into number followed by **one hot encoding method**.

**4.To find the best machine learning regression module based on R2 value:**

**i)Multiple Linear Regression:R2Value-0.7894**

**ii)Suppor Vector Machine:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **s.no** | **kernal** | **C=10**  **R2 value** | **C=100**  **R2 value** | **C=1000**  **R2 value** | **C=2000**  **R2 value** |
| 1 | linear | 0.4624 | 0.6287 | 0.7649 | 0.7440 |
| 2 | rbf | -0.0322 | 0.3200 | 0.8102 | 0.8547 |
| 3 | poly | 0.0387 | 0.6179 | 0.8566 | 0.8605 |
| 4 | sigmoid | 0.0393 | 0.5276 | 0.2874 | -0.5939 |

**iii)Decision tree:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **criterion** | **Splitter** | **Max\_features**  **“sqrt”** | **Max\_features**  **“log2”** | **Max\_features**  **none** | **R2 value** |
| **1** | Squared\_error | Best | **-** | **-** | **none** | **0.6581** |
| **2** | Squared\_error | random | **-** | **-** | **none** | **0.6893** |
| **3** | squared\_error | best | sqrt | **-** | **-** | **0.7209** |
| **4** | squared\_error | best | **-** | **Log2** | **-** | **0.6985** |
| **5** | squared\_error | **random** | sqrt | **-** | **-** | **0.6530** |
| **6** | squared\_error | **random** | **-** | **Log2** | **-** | **0.5824** |
| **7** | **absolute\_error** | best | sqrt | **-** | **-** | **0.6152** |
| **8** | **absolute\_error** | best | **-** | **Log2** | **-** | **0.7359** |
| **9** | **absolute\_error** | **random** | sqrt | **-** | **-** | **0.7154** |
| **10** | **absolute\_error** | **random** | **-** | **Log2** | **-** | **0.6172** |
| **11** | **friedman-mse** | best | sqrt | **-** | **-** | **0.7738** |
| **12** | **friedman-mse** | best | **-** | **Log2** | **-** | **0.6108** |
| **13** | **friedman-mse** | **random** | sqrt | **-** | **-** | **0.6985** |
| **14** | **friedman-mse** | **random** | **-** | **Log2** | **-** | **0.6674** |
| **15** | **poisson** | best | sqrt | **-** | **-** | **0.7368** |
| **16** | **poisson** | best | **-** | **Log2** | **-** | **0.6723** |
| **17** | **poisson** | **random** | sqrt | **-** | **-** | **0.7067** |
| **18** | **poisson** | **random** | **-** | **Log2** | **-** | **0.7138** |

**iv)Random forest:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **s.no** | **n\_estimators**  **50** | **n\_estimators**  **100** | **criterion** | **Max\_features**  **“ Sqrt”** | **Max\_features**  **“log2”** | **R2 value** |
| **1** | **50** | **-** | **absolute\_error** | **sqrt** | **-** | **0.8694** |
| **2** | **50** | **-** | **absolute\_error** | **-** | **Log2** | **0.8714** |
| **3** | **-** | **100** | **absolute\_error** | **sqrt** | **-** | **0.8701** |
| **4** | **-** | **100** | **absolute\_error** | **-** | **Log2** | **0.8705** |
| **5** | **50** | **-** | Squared\_error | **sqrt** | **-** | 0.8681 |
| **6** | **50** | **-** | Squared\_error | **-** | **Log2** | **0.8703** |
| **7** | **-** | **100** | squared\_error | **sqrt** | **-** | **0.8725** |
| **8** | **-** | **100** | squared\_error | **-** | **Log2** | **0.8712** |
| **9** | **50** | **-** | **friedman-mse** | **sqrt** | **-** | **0.8695** |
| **10** | **50** | **-** | **friedman-mse** | **-** | **Log2** | **0.8697** |
| **11** | **-** | **100** | **friedman-mse** | **sqrt** | **-** | **0.8707** |
| **12** | **-** | **100** | **friedman-mse** | **-** | **Log2** | **0.8725** |
| **13** | **50** | **-** | **poisson** | **sqrt** | **-** | **0.8700** |
| **14** | **50** | **-** | **poisson** | **-** | **Log2** | **0.8672** |
| **15** | **-** | **100** | **poisson** | **sqrt** | **-** | **0.8691** |
| **16** | **-** | **100** | **poisson** | **-** | **Log2** | **0.8732** |

**5.Best ML Regression module for given problem statement:**

**Random forest** is the best module because based on the R2value .this R2 value nearest to 1 (i.e-0.8732) compare than others. this the best performance module so we can take this module for deployment phase.