**Hope Artificial Intelligence**

**Assignment-Regression Algorithms**

**Project - Predict the Insurance Charges**

**Problem Statement**

A client’s requirement is, he wants to predict the insurance charges based on several parameters. The Client has provided the dataset of the same in the CSV file format which is given below.



**1.Identify the Problem Statement**

**Stage 1**: Domain selection

The input is number so it is "**Machine Learning**"

**Stage 2**: Learning selection

The requirement is clear and both input & output are available so it is "**Supervised**" learning

**Stage 3**: Supervised (Classification or Regression)

The output has numerical data so it is "**Regression**"

**2.Dataset Basic Information** (Total number of rows, columns)

Let us see little information about client given dataset.

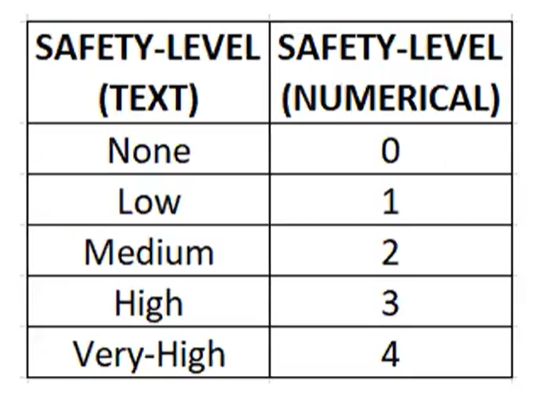
* Rows Detail - Total number of rows is 1338
* Columns Detail - Age, Sex, BMI, Children, Smoker, Charges
* We need to predict the insurance charges so Charges column is the output
* Age, Sex, BMI, Children, and Smoker is our Input Parameter

**3.Pre-processing method for dataset**

* Here observed that Sex and Smoker column looks the categorical data
* Categorical Data isn’t handle/process in AI using Python so we need to convert into meaning number
* We can defined Categorical Column into Nominal and Ordinal data.
* Nominal Data
  + - Data can’t compare because it isn’t in order wise.
    - For Example
      * Place Name(Erode, Chennai, Bangalore)
      * Things Name(Toys, Computer, Clothes, Door)
    - Use One hot encoding to convert into meaning number
    - Single Column will be expand based on column value once converted into numerical number like below



* Ordinal Data
* Ordinal Data can compare and it is in order wise
* For Example
  + Dress Size(S,M,L,XL,XXL,XXXL)
  + Priority(High,Medium,Low)
* Use Mapping/Label Encoder to convert into meaning number
* Column will not expand once converted into numerical number. See below table pic.



* Here I’m go with Nominal Data conversion.

**4.Develop a good model with r2\_score**

To Find the R2 Score using different Algorithms

1. **Multiple Linear Regression:** *R2 Score* = 0.78947
2. **Support Vector Machine:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.No | Hyper Parameter (C Value) | Linear  (r score) | RBF  (r score) | Poly  (r score) | Sigmoid  (r score) | Precomputed  (r score) |
| 1 | 10 | 0.4624 | -0.0322 | 0.0387 | 0.0393 | Not able to apply this. Reason below |
| 2 | 100 | 0.6288 | 0.3200 | 0.6179 | 0.5276 |
| 3 | 1000 | 0.7649 | 0.8102 | 0.8566 | 0.2874 |
| 4 | 2000 | 0.7440 | 0.8547 | 0.8605 | -0.5939 |
| 5 | 3000 | 0.7414 | 0.8663 | 0.8598 | -2.1244 |
| 6 | 4000 | 0.7414 | 0.8717 | 0.8600 | -5.5103 |

Get R2 score near to 1 when kernel is RBF and C is 4000 = 0.8717

1. **Decision Tree:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Criterion | Max Features | Splitter | R Score |
| 1 | squared\_error | auto | best | 0.6997 |
| 2 | squared\_error | auto | random | 0.7461 |
| 3 | squared\_error | sqrt | best | 0.7794 |
| 4 | squared\_error | sqrt | random | 0.7338 |
| 5 | squared\_error | log2 | best | 0.7520 |
| 6 | squared\_error | log2 | random | 0.7323 |
| 7 | friedman\_mse | auto | best | 0.6815 |
| 8 | friedman\_mse | auto | random | 0.6954 |
| 9 | friedman\_mse | sqrt | best | 0.6914 |
| 10 | friedman\_mse | sqrt | random | 0.6972 |
| 11 | friedman\_mse | log2 | best | 0.7771 |
| 12 | friedman\_mse | log2 | random | 0.7477 |
| 13 | absolute\_error | auto | best | 0.6797 |
| 14 | absolute\_error | auto | random | 0.7445 |
| 15 | absolute\_error | sqrt | best | 0.5990 |
| 16 | absolute\_error | sqrt | random | 0.6521 |
| 17 | absolute\_error | log2 | best | 0.7034 |
| 18 | absolute\_error | log2 | random | 0.7182 |
| 19 | poisson | auto | best | 0.6697 |
| 20 | poisson | auto | random | 0.6995 |
| 21 | poisson | sqrt | best | 0.6372 |
| 22 | poisson | sqrt | random | 0.6494 |
| 23 | poisson | log2 | best | 0.6942 |
| 24 | poisson | log2 | random | 0.6258 |

Get R2 score near to 1 (squared\_error, sqrt, best) = 0.7794

1. **Random Forest:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Criterion | Max Features | N\_Estimators | R Score |
| 1 | squared\_error | auto | 10 | 0.8331 |
| 2 | squared\_error | auto | 100 | 0.8539 |
| 3 | squared\_error | sqrt | 10 | 0.8520 |
| 4 | squared\_error | sqrt | 100 | 0.8709 |
| 5 | squared\_error | log2 | 10 | 0.8520 |
| 6 | squared\_error | log2 | 100 | 0.8709 |
| 7 | friedman\_mse | auto | 10 | 0.8332 |
| 8 | friedman\_mse | auto | 100 | 0.8540 |
| 9 | friedman\_mse | sqrt | 10 | 0.8503 |
| 10 | friedman\_mse | sqrt | 100 | 0.8709 |
| 11 | friedman\_mse | log2 | 10 | 0.8503 |
| 12 | friedman\_mse | log2 | 100 | 0.8709 |
| 13 | absolute\_error | auto | 10 | 0.8355 |
| 14 | absolute\_error | auto | 100 | 0.8521 |
| 15 | absolute\_error | sqrt | 10 | 0.8574 |
| 16 | absolute\_error | sqrt | 100 | 0.8717 |
| 17 | absolute\_error | log2 | 10 | 0.8574 |
| 18 | absolute\_error | log2 | 100 | 0.8717 |
| 19 | poisson | auto | 10 | 0.8178 |
| 20 | poisson | auto | 100 | 0.8332 |
| 21 | poisson | sqrt | 10 | 0.7967 |
| 22 | poisson | sqrt | 100 | 0.8293 |
| 23 | poisson | log2 | 10 | 0.7967 |
| 24 | poisson | log2 | 100 | 0.8293 |

Get R2 score near to 1 (squared\_error, sqrt/log2, 100) = 0.8717

1. **The Final Machine Learning best method of regression:**
2. Random Forest R2 score(squared\_error, sqrt/log2, 100) =0.8717

(or)

1. Support Vector machine R2 score(kernel is RBF and C is 4000) = 0.8717
2. **Justification:**

Why I chosen the Random Forest and SVM because these only given the R2 value near to 1 higher than other algorithms.