**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
2. Tell basic info about the dataset (Total number of rows, columns)
3. Mention the pre-processing method if you’re doing any (like converting string to number – nominal data)
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. Mention your final model, justify why u have chosen the same.

**1.) 3 Stages of Problem Statement**

**Stage 1** - Domain - ***Machine Learning*** (inputs - number)

**Stage 2** - Learning ***– Supervised Learning*** (clear requirements and both input and output present)

**Stage 3** – numerical value - ***Supervised Regression Learning*** – output numerical value

**Project Name: *“InsureCast”: Forecasting Insurance Charges with ML***

**2.) Dataset basic information**

Total number of rows – ***1338***

Total number of columns – ***6***

**3.) Data Preprocessing: One-Hot Encoding for Categorical Values**

When working with machine learning models, especially regression models, it's important to **convert categorical variables into a numerical format** because most algorithms require numerical input. For categorical data, we use techniques like One-Hot Encoding.

**Categorical Variables:**

Column like “Sex” and “Smoker” have categorical values.

* **Sex:** Male, Female
* **Smoker:** Yes, No

These categorical values are **nominal** - represent distinct categories without any inherent order.

**One-Hot Encoding:**

Nominal data 🡪 One Hot Encoding 🡪 convert string to number

One-Hot Encoding is a method to convert categorical variables into a numerical format.

1. **Identify the Categorical Values:**

* "Sex": Male, Female
* "Smoker": Yes, No

1. **Create New Binary Columns:**

For each unique value in the categorical column, create a new binary (0 or 1) column:

* For "Sex": Create two new columns, "Sex\_Male" and "Sex\_Female".
* For "Smoker": Create two new columns, "Smoker\_Yes" and "Smoker\_No".

1. **Assign Binary Values:**

* In the "Sex\_Male" column, assign 1 if the original value is "Male" and 0 if it's "Female".
* In the "Sex\_Female" column, assign 1 if the original value is "Female" and 0 if it's "Male".
* Similarly, for "Smoker\_Yes" and "Smoker\_No".

**4.) Develop Good Model**

***Source code for Developed Model:***

[**https://github.com/Marudhanayagam4/Assignment\_Regression**](https://github.com/Marudhanayagam4/Assignment_Regression)

**Model Creation / Learning Phase**

* Data collection
* Data preprocessing
* Input / Output split
* Split train set and test set
* Train set 🡪 model creation
* Test set 🡪 evaluation metrics 🡪 save the best model

**Deployment Phase / End User**

* Load the saved model
* Get inputs
* Predicts
* Call to action

**5.) R² score for many models**

**Algorithm** – Multiple Linear Regression - **R² score – 0.78**

**Algorithm** – Support Vector Machine Regression – **R² score – 0.86**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ***SVMR R2 score*** |  |
| **S No** | **kernel** | **Regularization parameter C** | **R2 score** |
| 1 | rbf | 1 | -0.08 |
| 2 | rbf | 10 | -0.03 |
| 3 | rbf | 100 | 0.32 |
| 4 | rbf | 1000 | 0.81 |
| 5 | rbf | 2000 | 0.85 |
| 6 | rbf | 3000 | 0.86 |
| 7 | linear | 1 | -0.01 |
| 8 | linear | 10 | 0.46 |
| 9 | linear | 100 | 0.62 |
| 10 | linear | 1000 | 0.76 |
| 11 | linear | 2000 | 0.74 |
| 12 | linear | 3000 | 0.74 |
| 13 | poly | 1 | -0.07 |
| 14 | poly | 10 | 0.03 |
| 15 | poly | 100 | 0.61 |
| 16 | poly | 1000 | 0.85 |
| 17 | poly | 2000 | 0.86 |
| 18 | poly | 3000 | 0.85 |
| 19 | sigmoid | 1 | -0.07 |
| 20 | sigmoid | 10 | 0.03 |
| 21 | sigmoid | 100 | 0.52 |
| 22 | sigmoid | 1000 | 0.28 |
| 23 | sigmoid | 2000 | -0.59 |
| 24 | sigmoid | 3000 | -2 |

**Algorithm** – Decision Tree Regression – **R² score – 0.75**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Decision Tree*** | ***R2 score*** |  |  |
| **S No** | **criterion** | **splitter** | **max\_features** | **R2 score** |
| 1 | squared\_error | *best* | *None* | 0.7 |
| 2 | squared\_error | random | None | 0.71 |
| 3 | squared\_error | best | sqrt | 0.67 |
| 4 | squared\_error | random | sqrt | 0.69 |
| 5 | squared\_error | best | log2 | 0.67 |
| 6 | squared\_error | random | log2 | 0.69 |
| 7 | friedman\_mse | best | None | 0.71 |
| 8 | friedman\_mse | random | None | 0.71 |
| 9 | friedman\_mse | best | sqrt | 0.67 |
| 10 | friedman\_mse | random | sqrt | 0.69 |
| 11 | friedman\_mse | best | log2 | 0.67 |
| 12 | friedman\_mse | random | log2 | 0.69 |
| 13 | absolute\_error | best | None | 0.67 |
| 14 | absolute\_error | random | None | 0.7506 |
| 15 | absolute\_error | best | sqrt | 0.69 |
| 16 | absolute\_error | random | sqrt | 0.64 |
| 17 | absolute\_error | best | log2 | 0.69 |
| 18 | absolute\_error | random | log2 | 0.64 |
| 19 | poisson | best | None | 0.7288 |
| 20 | poisson | random | None | 0.7 |
| 21 | poisson | best | sqrt | 0.7591 |
| 22 | poisson | random | sqrt | 0.62 |
| 23 | poisson | best | log2 | 0.7591 |
| 24 | poisson | random | log2 | 0.62 |

**Algorithm** –Random Forest Regression – **R² score – 0.75**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Random Forest R2 score*** |  |  |  |
| **S No** | **criterion** | **n\_estimators** | **max\_features** | **R2 score** |
| 1 | squared\_error | 100 | sqrt | 0.871 |
| 2 | squared\_error | 50 | sqrt | 0.86 |
| 3 | squared\_error | 100 | log2 | 0.871 |
| 4 | squared\_error | 50 | log2 | 0.86 |
| 5 | squared\_error | 100 | None | 0.85 |
| 6 | squared\_error | 50 | None | 0.84 |
| 7 | absolut\_error | 100 | None | 0.85 |
| 8 | absolute\_error | 50 | None | 0.85 |
| 9 | absolute\_error | 100 | sqrt | 0.8711 |
| 10 | absolute\_error | 50 | sqrt | 0.8708 |
| 11 | absolute\_error | 100 | log2 | 0.8711 |
| 12 | absolute\_error | 50 | log2 | 0.8708 |
| 13 | friedman\_mse | 100 | sqrt | 0.871 |
| 14 | friedman\_mse | 50 | sqrt | 0.87 |
| 15 | friedman\_mse | 100 | log2 | 0.8702 |
| 16 | friedman\_mse | 50 | log2 | 0.8702 |
| 17 | friedman\_mse | 100 | None | 0.85 |
| 18 | friedman\_mse | 50 | None | 0.85 |
| 19 | poisson | 100 | sqrt | 0.86 |
| 20 | poisson | 50 | sqrt | 0.86 |
| 21 | poisson | 100 | log2 | 0.86 |
| 22 | poisson | 50 | log2 | 0.86 |
| 23 | poisson | 100 | None | 0.85 |
| 24 | poisson | 50 | None | 0.84 |

**6.) Final Model – Random Forest Regression**

**R2 Score:**

* shows how good the model's predictions are compared to the actual data.
* **Value Range:** The R2 score ranges from 0 to 1.

|  |  |
| --- | --- |
| ***Algorithm*** | ***R2 score*** |
| Multiple Linear Regression (MLR) | 0.78 |
| Support Vector Machine Regression (SVMR) | 0.86 |
| Decision Tree Regression | 0.75 |
| Random Forest Regression | 0.87 |

**Justification:**

**Highest R2 Score:**

* Random Forest Regression: 0.87 (closest to 1)

**Model Stability:**

* Ensemble learning method reduces overfitting

**Handling Non-linearity and Interactions:**

* Captures complex relationships better than linear models

**Reduced Variance:**

* Averaging multiple decision trees leads to more reliable predictions

**Conclusion:**

* Random Forest Regression is the most accurate, reliable, and robust model for predicting insurance charges based on the R2 score and its numerous advantages over other models
* R2 score is 0.87, it means the model explains 87% of the variability in the target variable. This indicates a very good fit, with predictions that are quite accurate.