# Regression Algorithms for Insurance Charges Prediction

1. **Identifying the problem statement**

   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

   **Description:**

   With given parameters such as person's age, sex, number of children, BMI, smoking habits, we need to find the insurance charge/premium that he need to pay. That is, we need to predict the insurance premium amount of a person provided, when the mentioned parameters are available

   **Stage 1:** Domain Selection
   Since the input is in number format, **Machine Learning** can be used to solve this problem
   **Stage 2:** Learning Selection
   In the given dataset, input and output are well defined, so **Supervised Learning** can be used to solve this problem
   **Stage 3:** Type of algorithm: **Regression**

2. **Basic Information about the given dataset**

   Dataset name: insurance_pre.csv
   The given insurance dataset contains 5 input columns & 1 output column
   **Input column names:**
   Age (type: numerical)
   Sex (type: categorical-nominal)
   BMI (type: numerical)
   Children (type: numerical)
   Smoker (type: categorical-nominal)
   **Output column name:**
   Charges (type: numerical)

   Total number of rows present: 1338
   Total number of columns present: 6

3. **Pre-processing method**
    - **One Hot Encoding**
    - **Standardization**

    - There are 2 categorical(nominal) columns namely sex and smoker in dataset. So, in order to apply regression algorithms, we convert those categorical columns into numerical columns using **One Hot Encoding** technique
- Age column values contains 2 digits, charges column values contain 5 digits and above, sex and smoker columns contain single digit value after apply one hot encoding. So, to make all columns into same scale, we apply **Standardization** technique.

4. We develop regression algorithms – Multiple Linear, SVM, Decision tree and Random Forest – Obtain r2_score of various models

5. We documented respective r2_score values for various regression models, including the hyper-parameter tuning of the models

- **Multiple Linear Regression**

| Sno | fit_intercept | copy_X | n_jobs | positive | R^2 value |
|---|---|---|---|---|---|
| 1 | **TRUE** | **TRUE** | **None** | **FALSE** | **0.7865** |
| 2 | FALSE | FALSE | 2 | TRUE | 0.7864 |
| 3 | TRUE | FALSE | 3 | TRUE | 0.7865 |
| 4 | FALSE | TRUE | -1 | TRUE | -0.337 |
| 5 | TRUE | TRUE | -1 | FALSE | 0.7865 |

- **Support Vector Machine**

| Sno | Hyper parameter (C) | R^2 when kernel="linear" | R^2 when kernel="rbf" | R^2 when kernel="poly" | R^2 when kernel="Sigmoid" |
|---|---|---|---|---|---|
| 1 | 0.1 | -0.091 | -0.098 | -0.097 | -0.0975 |
| 2 | 100 | 0.616 | 0.291 | 0.604 | 0.505 |
| 3 | 1000 | 0.759 | 0.791 | 0.851 | 0.184 |
| 4 | 10000 | 0.761 | **0.872** | 0.856 | -26.88 |
| 5 | 100000 | 0.761 | 0.872 | 0.856 | -2738.06 |
| 6 | 1000000 | 0.761 | 0.861 | 0.856 | -336074.73 |

- **Decision Tree**

| SNO | criterion | splitter | max_features | R^2 value without dropping 1st column in get_dummies | R^2 value while 1st column in get_dummies is dropped |
|---|---|---|---|---|---|
| 1 | squared_error | best | None | 0.698 | 0.71 |
| 2 | squared_error | best | auto | 0.695 | 0.694 |
| 3 | squared_error | best | sqrt | 0.682 | 0.688 |
| 4 | squared_error | best | log2 | 0.706 | 0.553 |
| 5 | squared_error | random | auto | 0.708 | 0.708 |
| 6 | squared_error | random | sqrt | 0.673 | 0.683 |
| 7 | **squared_error** | **random** | **log2** | 0.662 | **0.724** |
| 8 | friedman_mse | best | auto | 0.684 | 0.693 |
| 9 | friedman_mse | random | sqrt | 0.722 | 0.718 |
| 10 | **friedman_mse** | **best** | **log2** | **0.750** | 0.692 |
| 11 | friedman_mse | random | auto | 0.737 | 0.699 |
| 12 | absolute_error | best | auto | 0.688 | 0.701 |
| 13 | absolute_error | random | sqrt | 0.674 | 0.657 |
| 14 | absolute_error | best | log2 | 0.708 | 0.657 |
| 15 | absolute_error | random | log2 | 0.701 | 0.628 |
| 16 | poisson | best | auto | 0.680 | 0.699 |
| 17 | poisson | random | sqrt | 0.666 | 0.694 |
| 18 | poisson | best | log2 | 0.641 | 0.654 |
| 19 | poisson | random | auto | 0.690 | 0.587 |

- **Random Forest**

| Sno | n_estimators | criterion | max_features | random_state | max_depth | R^2 value |
|---|---|---|---|---|---|---|
| 1 | 100 | squared_error | 1.0 | None | None | 0.859 |
| 2 | 1000 | squared_error | 1.0 | 0 | None | 0.860 |
| 3 | 10000 | squared_error | sqrt | 0 | 10 | 0.88 |
| 4 | 10000 | squared_error | log2 | 0 | 10 | 0.88 |
| 5 | 10000 | friedman_mse | sqrt | 0 | None | 0.873 |
| 6 | 10000 | friedman_mse | log2 | 0 | None | 0.873 |
| 7 | 10000 | friedman_mse | 1.0 | None | None | 0.860 |
| 8 | 10000 | friedman_mse | sqrt | 0 | 10 | 0.880 |
| 9 | **10000** | **absolute_error** | **sqrt** | **0** | **10** | **0.888** |
| 10 | 10000 | absolute_error | log2 | 0 | None | 0.876 |
| 11 | 10000 | absolute_error | 1.0 | None | 10 | 0.873 |

6. **Final Model:**

The final model is RandomForestRegressor with hyper parameters applied as n_estimators=10000, criterion="absolute_error", max_features="sqrt", random_state=0, max_depth=10

Chosen this model as final model, since it gives better accuracy in terms of R^2 value for mentioned combination of hyper parameters applied.

**Model Comparison - R^2 values:**

| Regression Model | R^2 value |
|---|---|
| Multiple Linear Regression | 0.786 |
| Support Vector Machine Regression | 0.872 |
| Decision Tree Regression | 0.750 |
| **Random Forest Regression** | **0.888** |