Assignment-Regression Algorithm

1. Identify your problem statement:

Domain Selection:

Machine Learning

The client's requirement is Very clear

Input and output is present.

So it is Supervised Learning.

The Insurance charge column contains continuous numerical numbers. So it falls under Regression.

Hence it is **Supervised-Regression**

2.

Input data is related to customer's house hold such as age, sex, bmi, no of children and whether the customer is smoker or not.

Output data is related to Insurance charge the customer has to pay

It has totally 1338 rows and 6 columns

3. The input columns sex and smoker contains String values. Sex and smoker columns belongs to Categorical variables (Nominal Data).It is not possible to a create model using String values. So the String values to be converted to numbers (integers).

4. The RandomForest algorithm performs well based on the given data.

Please see the attached screenshot

```
In [135]: from sklearn.ensemble import RandomForestRegressor
          regressor =RandomForestRegressor(n_estimators=100,random_state=0)
          regressor.fit(X_train,y_train)

          C:\Users\hema\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vecto
          r y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for exampl
          e using ravel().
            return fit_method(estimator, *args, **kwargs)

Out[135]: RandomForestRegressor(random_state=0)
          In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
          On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [136]: y_pred=regressor.predict(X_test)

In [137]: from sklearn.metrics import r2_score
          r_score = r2_score(y_test,y_pred)

In [138]: r_score

Out[138]: 0.8552594181703539
```

The same is uploaded to the Repository.

5. Research values are given below

Multiple Linear Regression:

Multiple Linear Regression	
SNO	Linear(r2 value)
1.0000	0.7895

Support Vector Machine:

Support Vector Machine					
Serial number	Hyper parameter	Linear (r2 value)	RBF(Non Linear) r2 value	Poly (r2 value)	Sigmoid (r2 value)
1	C1.0	-0.1001	-0.0885	-0.0725	-0.0899
2	C10	0.1136	-0.0823	-0.0843	-0.0906
3	C100	0.5951	-0.1232	-0.0972	-0.1139
4	C500	0.6341	-0.1195	-0.0766	-0.3455
5	C1000	0.6893	-0.1089	-0.0436	-1.1086
6	C2000	0.7651	-0.0905	0.0219	-3.6653
7	C3000	0.7649	-0.0693	0.0850	-7.5078

Decision Tree Algorithm:

Decision Tree				
Sno	Criterion	Splitter	Max Features	R Value
1	squared_error	best	None	0.6966
1	squared_error	best	log2	0.6963
2	squared_error	best	sqrt	0.7509
3	friedman_mse	best	log2	0.6975
4	friedman_mse	best	sqrt	0.7093
5	friedman_mse	random	sqrt	0.6807
6	friedman_mse	random	log2	0.6687
7	absolute_error	best	log2	0.7716
8	absolute_error	random	log2	0.6807
9	absolute_error	best	sqrt	0.7026
10	absolute_error	random	sqrt	0.7195
11	poisson	best	sqrt	0.7055
12	poisson	best	log2	0.6829
13	poisson	random	log2	0.6485
14	poisson	random	sqrt	0.6504
15	squared_error	random	log2	0.7175
16	squared_error	random	sqrt	0.7062

Random Forest Algorithm:

Random Forest				
Sno	Criterion	n_estimators	random_state	R2 value
1	Squared_error	10	0	0.8392437
2	Squared_error	50	0	0.8515
3	Squared_error	100	0	0.8552594
4	Squared_error	100	5	0.85521277
5	Squared_error	200	0	0.85343869
6	friedman_mse	10	0	0.8395167
7	friedman_mse	50	0	0.851598
8	friedman_mse	100	0	0.8549
9	friedman_mse	150	0	0.8540
10	friedman_mse	200	0	0.853205
11	friedman_mse	200	5	0.8548373
12	friedman_mse	300	5	0.8559179
13	poisson	10	0	0.83258889
14	poisson	50	0	0.84943343
15	poisson	60	0	0.84931362
16	poisson	70	0	0.8516
17	poisson	75	0	0.8517
18	poisson	100	0	0.853596
19	poisson	1000	5	0.8572

6.

I tried with multiple linear regression, Support vector machine, Decision tree and Random Forest algorithms. All the resulted R-squared values are tabulated and presented above in the form of screenshots.

Finally found *Random Forest algorithm* performs well with the following parameters for the given dataset.

criterion=friedman_mse, n_estimators=300 and random_state=5.

Random Forest algorithm produces 0.86 in this case (for the given data set) which makes a model worth Compared to all the other algorithms.

Screenshot:

```
from sklearn.metrics import r2_score
r_score = r2_score(y_test,y_pred)
```

```
r_score
```

```
0.8552594181703539
```

```
import pickle
filename="finalized_model_assignment_RandomForest_default.sav"
```

```
pickle.dump(regressor,open(filename,'wb'))
```

```
loaded_model=pickle.load(open("finalized_model_assignment_RandomForest_default.sav",'rb'))
result=loaded_model.predict([[54,24,2,0,1,1,0]])
```

```
C:\Users\hema\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid fea
ture names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

```
result
```

```
array([11636.0634433])
```