AI-PROJECT 1
ASSIGNMENT-REGRESSION ALGORITHM

Ai | Subramani

AI PREDICTION

1. 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.

2. Dataset

File: insurance_pre.csv

- 5 row Header, 1339 Column dataset

age	sex	bmi	children	smoker	charges
19	female	27.9	0	yes	16884.92
18	male	33.77	1	no	1725.552
28	male	33	3	no	4449.462
33	male	22.705	0	no	21984.47

3. Domain Prediction: Machine Learning

- The predicted value is numeric.
- Data is number

4. Learning Prediction: Supervised Learning

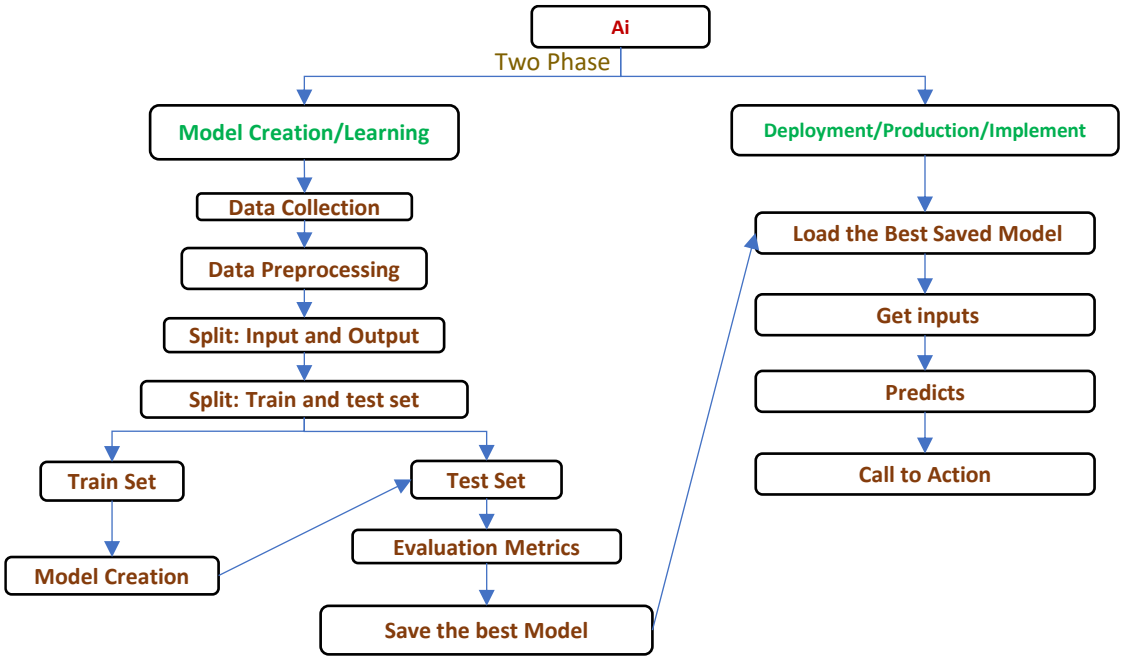
- Requirement is clear
- Both input and output data are available.

5. Algorithm Prediction: Regression

- Prediction is a number, so this is a Regression problem.
- We have 5 inputs and 1 output.
- Since there is more than one input, we can predict using the following algorithms:
    1. Multiple Linear Regression
    2. Support Vector Machine (SVR)
    3. Decision Trees
    4. Random Forests

age	sex	bmi	children	smoker	charges
19	female	27.9	0	yes	16884.92
18	male	33.77	1	no	1725.552
Input 1	Input 2	Input 3	Input 4	Input 5	output 1

6. Ai Work Flow for Prediction



1.Data Collection:

```
#Data collection
import pandas as pd
dataset=pd.read_csv("insurance_pre.csv")
dataset
```

	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230

Dataset: Index(['age','sex','bmi','children','smoker','charges'], dtype='object')

2. Data Preprocessing:

Categorical: Nominal: So label encoding

sex' and 'Smoker" are yes or no columns, converted string into number

```
#data preprocessing
dataset=pd.get_dummies(dataset,drop_first=True).astype(int)
dataset
```

	age	bmi	children	charges	sex_male	smoker_yes
0	19	27	0	16884	0	1
1	18	33	1	1725	1	0

5. Split input & output:

Split X input: `independent=dataset[['age', 'bmi', 'children', 'sex_male','smoker_yes']]`

1338 rows × 5 columns

Split y output: `dependent=dataset[['charges']]`

1338 rows × 1 columns

6. Split Training Set(70%) and Test Set(30%):

70% dataset X training & y training set : *936 rows × 5 columns*

30% dataset X test set & y test set: *402 rows × 5 columns*

```
#split Train and test set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(independent, dependent, test_size=0.30, random_state=0)
X_train
```

	age	bmi	children	sex_male	smoker_yes
1163	18	28	0	0	0
196	39	32	0	0	0

```
#split Train and test set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(independent, dependent, test_size=0.30, random_state=0)
y_test
```

	charges
578	9724
610	8547

7. Model Creation:

Create the Model: Using 70% of the data as the training set (X_train, y_train) with the following algorithms

1. Multiple Linear Regression

```
#Model Creation
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

2. Support Vector Machine (SVM)

```
#Model Creation
from sklearn.svm import SVR
#kernel{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’} or callable, default=’rbf’
#regressor=SVR(kernel="sigmoid",C=100)
regressor=SVR(kernel="poly",C=100000)
regressor.fit(X_train,y_train)
```

3. Decision Trees

```
#Model create: Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(criterion='poisson',splitter="best",max_features="sqrt")
regressor=regressor.fit(X_train,y_train)
```

4. Random Forests

```
#model Creation: RandomForest
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 50, criterion ="absolute_error", random_state = 0)
regressor.fit(X_train, y_train)
```

8. Evaluation Metrics:

- Each model will be tested on the 30% test set (X_test, y_test) using the following metrics:
- R^2 Score (Coefficient of Determination): Measures how well the model explains variance in the data. with the following algorithms

```
#7. Evaluation Metrics
y_pred=regressor.predict(X_test)
from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)
r_score
```

[26]:

0.78913454847886

7. Model Compare Different Regression Methods (R^2 values)

To find the best regression model in Machine Learning, we compare different regression methods using their R^2 values on the given dataset and apply different features across all algorithms.

1. **Multiple Linear Regression**: **Top R^2 value** = **0.7891**

2. **Support Vector Machine: Top R^2 value = 0.7649**

S. No	Hyper Parameter C	Linear R^2 value	RBF R^2 value	POLY R^2 value	SIGMOID R^2 value
1	C=1	-0.1115	-0.0884	-0.0645	-0.0899
2	C=100	0.5432	-0.1245	-0.0992	-0.1185
3	C=1000	0.6338	-0.1176	-0.0546	-1.7112
4	C=10000	0.7440	-0.0165	0.3536	-124.1083
5	C=100000	0.7413	0.5350	0.7649	-11667.4144

3. **Decision Tree: Top R^2 value = 0.8079**

S. No	Criterion	Splitter	max_features	R^2 value
1	squared_error	best	sqrt	0.7270
2	squared_error	best	log2	0.7611
3	squared_error	random	sqrt	0.6610
4	squared_error	random	log2	0.6850
5	friedman_mse	best	sqrt	0.7272
6	friedman_mse	best	log2	0.6252
7	friedman_mse	random	sqrt	0.6967
8	friedman_mse	random	log2	0.6782
9	absolute_error	best	sqrt	0.6755

10	absolute_error	best	log2	0.6257
11	absolute_error	random	sqrt	0.6738
12	absolute_error	random	log2	0.6691
13	poisson	best	sqrt	0.8079
14	poisson	best	log2	0.6643
15	poisson	random	sqrt	0.6410
16	poisson	random	log2	0.7053

4. **Random Forest Top R^2 value = 0.8576**

S. No	n_estimators	criterion	R^2 value
1	40	squared_error	0.8513
2	50	squared_error	0.8519
3	60	squared_error	0.8504
4	40	absolute_error	0.8574
5	50	absolute_error	0.8576
6	60	absolute_error	0.8560
7	40	friedman_mse	0.8513
8	50	friedman_mse	0.8519
9	60	friedman_mse	0.8504
10	40	poisson	0.8494
11	50	poisson	0.8495
12	60	poisson	0.8489

8. Save the Best Model

Compared to all models, **Random Forest Regression** achieved the highest R^2 value of 0.8576, making it the best model to save.

```
#save the Best Model
import pickle
filename="4_Best_Model_Random_Forest_Regression.sav"
pickle.dump(regressor,open(filename,'wb'))
result=regressor.predict([[19,27,0,0,1]])
result
```

9. Deployment / Implement

For the deployment process, load the best **Random Forest Regression** model (which achieved the highest **R^2 value of 0.8576**) and take user input to predict the **insurance charges**.

```
# Deployment
#pickle is library for save model
import pickle
#Load the model
load_model=pickle.load(open("4_Best_Model_Random_Forest_Regression.sav","rb"))

#Get inputs from user for predict the model
result=load_model.predict([[19,27,0,0,1]])
print("insurance charge:",result)
```

```
insurance charge: [16948.09]
```

10. Call to Action

The final model will serve as the call to action.