PREDICTION OF INSURANCE CHARGES USING REGRESSION ALGORITHMS

A DOCUMENTATION REPORT

Submitted by

MONISHA C

TABLE OF CONTENTS

1. PROBLEM STATEMENT

Insurance is a contract, represented by a policy, in which an individual or entity receives financial protection or reimbursement against losses from an insurance company. Based on the dataset of an individual or a person, insurance amount can be charged for an individual. Dataset: [age of a person, sex, BMI, no. of children, smoker, charges]. Based on this dataset, we cannot predict the insurance charge manually. It is very difficult to calculate the prediction value. To overcome this problem, machine learning can be used.

Machine learning is the process of future prediction based on the past data pattern. To select an algorithm based on the data pattern to solve the problem. Based on age of a person, sex, BMI, no. of children, smoker, we can predict the future value. (ie., insurance charges).

2. REQUIREMENTS

2.1. Basic Information:

- ❖ File format: Excel sheet or .csv file
- ❖ No. of rows: 1338
- ❖ No. of columns: 6

Dataset name(Column name)	**Description**
Age	Age of a person(numerical value)
Sex	Gender of a person(Male or Female)
BMI(Body Mass Index)	BMI of a person(numerical value)
Children	No. of children(numerical value)
Smoker	If the person is smoker or not(Yes or No)
Charges	Insurance charge of the person(numerical value)

2.2. Software requirements:

- ❖ Language: Python(version: 3.7.6)
- ❖ IDE: Jupyter Notebook
- ❖ Software used: Anaconda

2.3. Hardware requirements:

- ❖ Operating system: Windows 10
- ❖ RAM: 4GB

3. METHODOLOGY

Step 1: Data Collection

Age of a person, sex, BMI, no. of children, smoker, charges are the data to be collected and saved as csv file.

	A	B	C	D	E	F
1	age	sex	bmi	children	smoker	charges
2	19	female	27.9	0	yes	16884.92
3	18	male	33.77	1	no	1725.552
4	28	male	33	3	no	4449.462
5	33	male	22.705	0	no	21984.47
6	32	male	28.88	0	no	3866.855
7	31	female	25.74	0	no	3756.622
8	46	female	33.44	1	no	8240.59
9	37	female	27.74	3	no	7281.506
10	37	male	29.83	2	no	6406.411
11	60	female	25.84	0	no	28923.14

Fig: 3.1 Data collection

Step 2: Data Pre-processing

- Collected data can be processed while importing the libraries such as numpy, pandas and matplotlib.
- Reading the csv file using pandas library and then it can be displayed the dataset with total no. of rows and columns.

```
In [1]: #importing the Libraies
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

```
In [2]: # Reading the Dataset
        dataset = pd.read_csv('insurance_pre.csv')
```

```
In [3]: dataset
```

Out[3]:

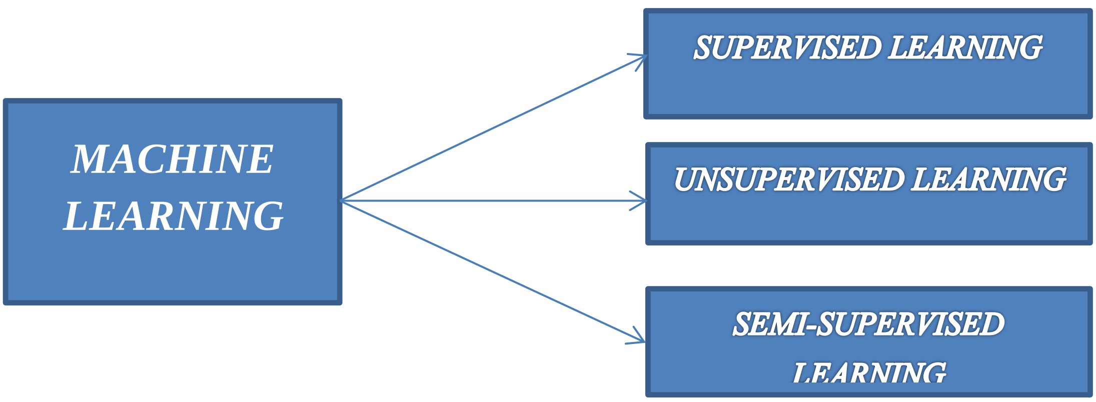
	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230
2	28	male	33.000	3	no	4449.46200
3	33	male	22.705	0	no	21984.47061
4	32	male	28.880	0	no	3866.85520
...	...	...	...	...	...	...
1333	50	male	30.970	3	no	10600.54830
1334	18	female	31.920	0	no	2205.98080
1335	18	female	36.850	0	no	1629.83350
1336	21	female	25.800	0	no	2007.94500
1337	61	female	29.070	0	yes	29141.36030

1338 rows × 6 columns

Fig: 3.2 Data Pre-processing (stage1)

Step 3: Algorithm Identification

In Machine learning, there are three types:



- ❖ In machine learning, the requirement should be clear in the given dataset. Input and output are well defined.

Then it is SUPERVISED LEARNING

- ❖ Under supervised learning, there are 2 types: Classification and Regression.

Classification	It has categorical values.
Regression	It has numerical values.

An output parameter has numerical value, then it is REGRESSION.

- ❖ In regression algorithm, using multiple linear regression, support vector machine, decision tree and random forest on the model

- ➢ The data in the dataset are nominal data. So it is very difficult to calculate the prediction values using the dataset.
- ➢ Because all the data must have numerical value. That is, it can be converted nominal data into ordinal data (numerical value).

Note: using get_dummies function to convert the text data into numerical data.

Sex	Value(0 or 1)
Male	1
Female	0

Smoker	Value(0 or1)
Yes	1
No	0

```
In [4]:  dataset=pd.get_dummies(dataset,drop_first=True)
```

```
In [5]:  dataset
```

Out[5]:

	age	bmi	children	charges	sex_male	smoker_yes
0	19	27.900	0	16884.92400	0	1
1	18	33.770	1	1725.55230	1	0
2	28	33.000	3	4449.46200	1	0
3	33	22.705	0	21984.47061	1	0
4	32	28.880	0	3866.85520	1	0
...	...	...	...	...	...	...
1333	50	30.970	3	10600.54830	1	0
1334	18	31.920	0	2205.98080	0	0
1335	18	36.850	0	1629.83350	0	0
1336	21	25.800	0	2007.94500	0	0
1337	61	29.070	0	29141.36030	0	1

1338 rows × 6 columns

Fig: 3.3 Data pre-processing (stage 2)

- Input parameters can be taken as independent variables.
- Output parameter can be taken as dependent variable.

age

bmi

children   independent variables

sex_male

smoker_yes

charges → dependent variable

```
In [6]: indep=dataset[['age', 'bmi', 'children','sex_male', 'smoker_yes']]
        dep=dataset['charges']
```

```
In [7]: indep
```

Out[7]:

	age	bmi	children	sex_male	smoker_yes
0	19	27.900	0	0	1
1	18	33.770	1	1	0
2	28	33.000	3	1	0
3	33	22.705	0	1	0
4	32	28.880	0	1	0
...	...	...	...	...	...
1333	50	30.970	3	1	0
1334	18	31.920	0	0	0
1335	18	36.850	0	0	0
1336	21	25.800	0	0	0
1337	61	29.070	0	0	1

1338 rows × 5 columns

```
In [8]: dep
```

```
Out[8]: 0        16884.92400
        1         1725.55230
        2         4449.46200
        3        21984.47061
        4         3866.85520
                     ...
        1333     10600.54830
        1334      2205.98080
        1335      1629.83350
        1336      2007.94500
        1337     29141.36030
        Name: charges, Length: 1338, dtype: float64
```

Fig: 3.4 Independent and dependent variables

Step 4: Training set and Test set

- ❖ Using sklearn, splitting the dataset into training set and test set. To find the model, training set plays a vital role in machine learning. Training the dataset is used to create a model.

Note: Training set and test set can be split by the parameter test_size in the train_test_split function.

Training set	X_train	Train set and test set splitted by1/3 ratio
	y_train	
Test set	X_test	
	y_test	

In [9]:
```
#split into training set and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(indep, dep, test_size = 1/3, random_state = 0)
```

In [28]:
```
X_train
```

Out[28]:

	age	bmi	children	sex_male	smoker_yes
482	18	31.35	0	0	0
338	50	32.30	1	1	1
356	46	43.89	3	1	0
869	25	24.30	3	0	0
182	22	19.95	3	1	0
...	...	...	...	...	...
763	27	26.03	0	1	0
835	42	35.97	2	1	0
1216	40	25.08	0	1	0
559	19	35.53	0	1	0
684	33	18.50	1	0	0

892 rows × 5 columns

In [29]:
```
X_train.shape
```

Out[29]: (892, 5)

Fig: 3.5 Spitting of training set and test set

4. MULTIPLE LINEAR REGRESSION

Multiple linear regression refers to a statistical technique that uses two or more independent variables to predict the outcome of a dependent variable.

Using LinearRegression() function, it fits a linear model with coefficients using training set.

Note: (.fit) is used for substitution

To calculate the weight and bias, using this formula

$$\boxed{\mathbf{Y=wx+b}}$$

w = weight (or) slope

b =bias (or) intercept

Y = dependent variable

x = independent variable

Based on the independent variables(age, bmi, children, sex_male, smoker_yes), we can predict the dependent variable.

```
In [10]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, y_train)#y=W*x1+b0 for this equation we got value for b1 and bo

Out[10]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [11]: # Viewing the b1 and bo value
         weight=regressor.coef_
         print("Weight of the model={}".format(weight))
         bais=regressor.intercept_
         print("Intercept of the model={}".format(bais))

         Weight of the model=[  260.1423112    315.22441969   545.72248029   -71.76915955
          23252.13608407]
         Intercept of the model=-12013.760127352209
```

Fig: 3.6 Finding weight and bias using model

Finding R^2:

R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

The purpose of find the value of R^2 is to know, how well the model is fitted.

The R^2 value exists between 0 and 1.

$$R^2 = \frac{SSR}{SST}$$

1(better performance)

R^2 nearly to

0(poor performance)

After getting the R squared value with better performance, it indicates how good the future prediction to be estimated.

Using predict function, it gets all the input values from the user to calculate the future predictions.

```
In [16]:  age_input=float(input("Age:"))
          bmi_input=float(input("BMI:"))
          children_input=float(input("Children:"))
          sex_male_input=int(input("Sex Male 0 or 1:"))
          smoker_yes_input=int(input("Smoker Yes 0 or 1:"))

          Age:35
          BMI:30
          Children:2
          Sex Male 0 or 1:0
          Smoker Yes 0 or 1:1

In [19]:  Future_Prediction=regressor.predict([[age_input,bmi_input,children_input,sex_male_input,smoker_yes_input]])# change the param
          print("Future_Prediction={}".format(Future_Prediction))

          Future_Prediction=[30891.53439999]

In [20]:  y_pred=regressor.predict(X_test)

In [21]:  from sklearn.metrics import r2_score
          r_score=r2_score(y_test,y_pred)

In [22]:  r_score

Out[22]:  0.7865108093853883
```

Fig: 3.7 Finding R squared value and future prediction value

5. SVM

Support Vector Machine or SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

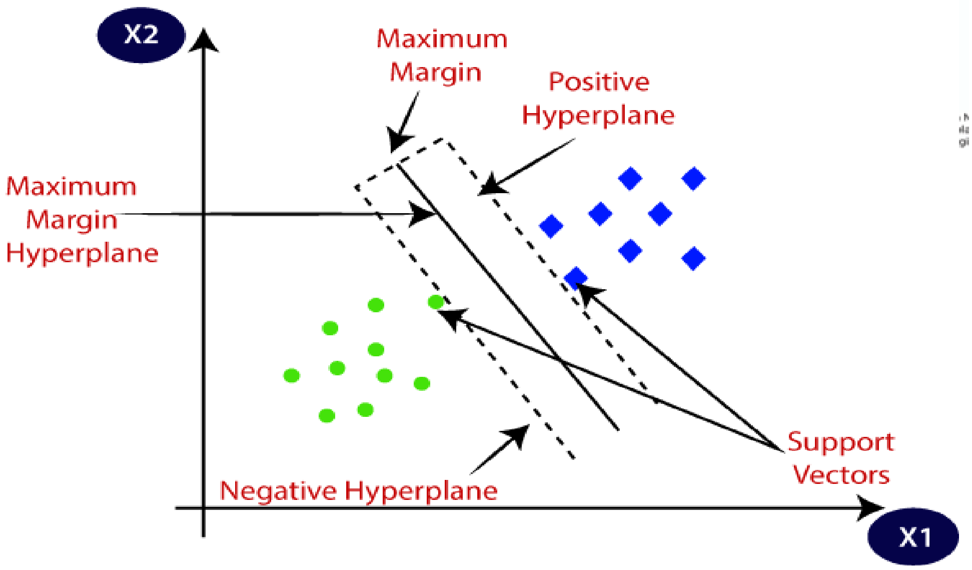


Fig: 3.8 SVM

SVM is of two types:

1. Linear SVM
2. Non-Linear SVM

StandardScaler: It standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.

StandardScaler results in a distribution with a standard deviation equal to 1. The variance is equal to 1 also, because **variance= standard deviation squared**. StandardScaler makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1.

```
In [9]:  from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()
          X_train = sc.fit_transform(X_train)
          X_test = sc.transform(X_test)

In [10]:  X_train

Out[10]: array([[-1.53963418,  0.11036616, -0.90788827, -0.98885138, -0.49929923],
                [ 0.74809711,  0.26412451, -0.0755796 ,  1.01127431,  2.00280702],
                [ 0.4621307 ,  2.13997636,  1.58903774,  1.01127431, -0.49929923],
                ...,
                [ 0.03318108, -0.90443894, -0.90788827,  1.01127431, -0.49929923],
                [-1.46814257,  0.7869029 , -0.90788827,  1.01127431, -0.49929923],
                [-0.46726014, -1.96941782, -0.0755796 , -0.98885138, -0.49929923]])
```

Fig: 3.9 Standard scalar preprocessing

Linear SVM:

Linear SVM is used for linearly separable data, which means if a dataset can be separated into two classes by using a single straight line.

Note: 'C' is called as hyperplane parameter

In linear support vector regression, when the parameters **kernel='linear' and C=3000 in SVR**, the r squared value is **0.76**.

```
In [24]: from sklearn.svm import SVR
         regressor = SVR(kernel ='linear',C=3000)
         regressor.fit(X_train, y_train)

Out[24]: SVR(C=3000, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
             gamma='auto_deprecated', kernel='linear', max_iter=-1, shrinking=True,
             tol=0.001, verbose=False)

In [25]: y_pred=regressor.predict(X_test)
         from sklearn.metrics import r2_score
         r_score=r2_score(y_test,y_pred)
         r_score

Out[25]: 0.7612136779519126
```

Fig: 3.10 Fitting a Linear SVM algorithm on a training set

Non-Linear SVM:

Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be separated by using a straight line.

Note: rbf(radial basis function)

In non-linear support vector regression, when the parameters **kernel='rbf' and C=3000 in SVR**, the r squared value is **0.86**.

```
In [18]:  from sklearn.svm import SVR
          regressor = SVR(kernel ='rbf',C=3000)
          regressor.fit(X_train, y_train)

Out[18]:  SVR(C=3000, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
              gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
              tol=0.001, verbose=False)
```

```
In [20]:  y_pred=regressor.predict(X_test)
          from sklearn.metrics import r2_score
          r_score=r2_score(y_test,y_pred)
          r_score

Out[20]:  0.8609984980441916
```

Fig: 3.11 Fitting a Non-Linear SVM algorithm on a training set

```
In [21]:  age_input=float(input("Age:"))
          bmi_input=float(input("BMI:"))
          children_input=float(input("Children:"))
          sex_male_input=int(input("Sex Male 0 or 1:"))
          smoker_yes_input=int(input("Smoker Yes 0 or 1:"))

          Age:35
          BMI:30
          Children:2
          Sex Male 0 or 1:0
          Smoker Yes 0 or 1:1
```

```
In [22]:  Future_Prediction=regressor.predict([[age_input,bmi_input,children_input,sex_male_input,smoker_yes_input]])#
          print("Future_Prediction={}".format(Future_Prediction))

          Future_Prediction=[16390.90477086]
```

Fig: 3.12 Future prediction using Non-Linear SVM

6. DECISION TREE ALGORITHM

Decision Tree is a supervised learning technique, it is a graphical representation for getting all the possible solutions to a problem (or) decision based on given conditions.

In a decision tree, there are two nodes, which are the Decision node and Leaf node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

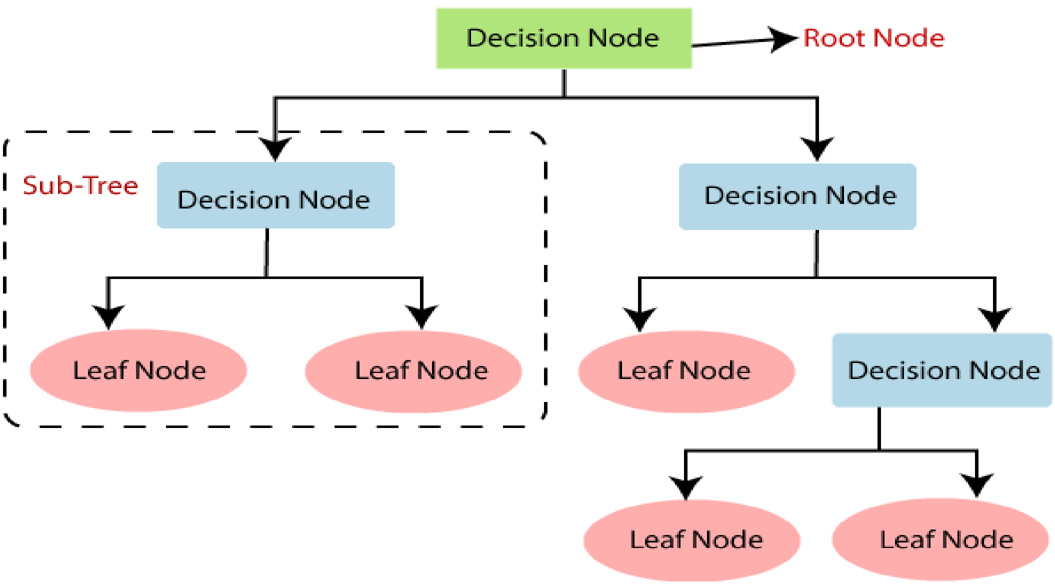


Fig: 3.13 Decision Tree

Note: Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

Two popular techniques for Attribute Selection Measures(ASM), which are:

1. Information Gain:

    Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

2. Gini index:

    It is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

To check all the possible ways while changing the parameters to know the better performance of r squared value.

criterion	**max_features**	**r squared value**
mae	sqrt	**0.75**
	auto	0.68
	log2	0.75
mse	sqrt	0.73
	auto	0.71
	log2	0.73
friedman_mse	sqrt	0.72
	auto	0.71
	log2	0.72

Note: When the parameters **criterion=”mae” and max_features=”sqrt” in DecisionTreeRegressor**, the r squared value is **0.75**.

```
In [11]:  from sklearn.tree import DecisionTreeRegressor
          regressor = DecisionTreeRegressor(criterion="mae",max_features="sqrt",random_state = 0)
          regressor.fit(X_train, y_train)

Out[11]:  DecisionTreeRegressor(criterion='mae', max_depth=None, max_features='sqrt',
                                max_leaf_nodes=None, min_impurity_decrease=0.0,
                                min_impurity_split=None, min_samples_leaf=1,
                                min_samples_split=2, min_weight_fraction_leaf=0.0,
                                presort=False, random_state=0, splitter='best')

In [12]:  y_pred=regressor.predict(X_test)
          from sklearn.metrics import r2_score
          r_score=r2_score(y_test,y_pred)
          r_score

Out[12]:  0.7581517082451512
```

Fig: 3.14 Fitting a Decision tree algorithm on a training set

```
In [13]:  age_input=float(input("Age:"))
          bmi_input=float(input("BMI:"))
          children_input=float(input("Children:"))
          sex_male_input=int(input("Sex Male 0 or 1:"))
          smoker_yes_input=int(input("Smoker Yes 0 or 1:"))

          Age:35
          BMI:30
          Children:2
          Sex Male 0 or 1:0
          Smoker Yes 0 or 1:1

In [14]:  Future_Prediction=regressor.predict([[age_input,bmi_input,children_input,sex_male_input,smoker_yes_input]])#
          print("Future_Prediction={}".format(Future_Prediction))

          Future_Prediction=[63770.42801]
```

Fig: 3.15 Future prediction using decision tree algorithm

7. RANDOM FOREST ALGORITHM

A Random Forest is an ensemble technique capable of performing regression task with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

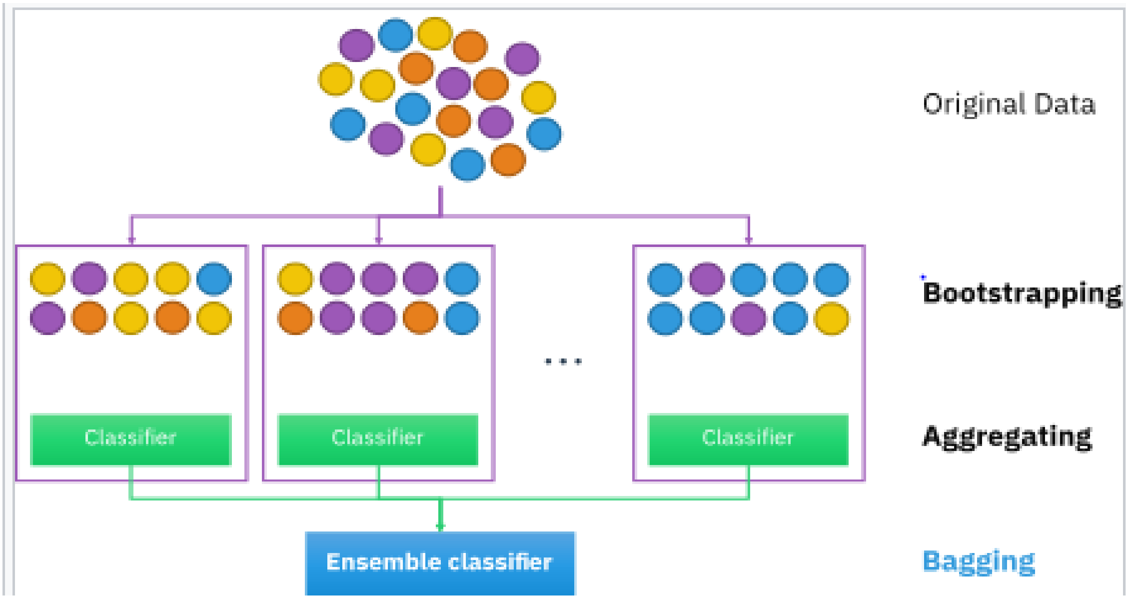


Fig: 3.16 Random Forest

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees.

In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.

Hyper-parameters of Random Forest:

1. n_estimators: Number of trees

   To set the no. of trees as we need in the random forest. This is done using a hyperparameter “n_estimators”.

2. criteria:

   Another important hyper-parameter is “criteria”. While deciding a split in decision trees, there are several criteria such as Gini impurity, information gain, entropy, etc.

3. max_features:

   “max_features” is one of the parameters that it can tune to randomly select the number of features at each node.

n_estimators	criterion	max_features	R squared value
10	mse	sqrt	0.84
		auto	0.85
		log2	0.84
	mae	sqrt	0.86
		auto	0.84
		log2	0.86
100	mse	sqrt	0.87
		auto	0.86
		log2	0.87
	mae	sqrt	**0.87**
		auto	0.85
		log2	0.87

```
In [40]: from sklearn.ensemble import RandomForestRegressor
         regressor = RandomForestRegressor(n_estimators = 100, random_state = 0, criterion = "mae", max_features = "sqrt")
         regressor.fit(X_train, y_train)

Out[40]: RandomForestRegressor(bootstrap=True, criterion='mae', max_depth=None,
                               max_features='sqrt', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=100,
                               n_jobs=None, oob_score=False, random_state=0, verbose=0,
                               warm_start=False)

In [41]: y_pred=regressor.predict(X_test)
         from sklearn.metrics import r2_score
         r_score=r2_score(y_test,y_pred)
         r_score

Out[41]: 0.8756481464068124

In [42]: age_input=float(input("Age:"))
         bmi_input=float(input("BMI:"))
         children_input=float(input("Children:"))
         sex_male_input=int(input("Sex Male 0 or 1:"))
         smoker_yes_input=int(input("Smoker Yes 0 or 1:"))

         Age:35
         BMI:30
         Children:2
         Sex Male 0 or 1:0
         Smoker Yes 0 or 1:1

In [43]: Future_Prediction=regressor.predict([[age_input,bmi_input,children_input,sex_male_input,smoker_yes_input]])# change
         print("Future_Prediction={}".format(Future_Prediction))

         Future_Prediction=[47892.1426878]
```

Fig: 3.16 Fitting a random forest algorithm in training set to get future prediction value

8. FINAL MODEL

Using regression algorithms, to calculate the r squared value to know the best performance of a model.

Tabulation of R^2 value using Regression Algorithms

ALGORITHM	R^2 VALUE
Multiple Linear Regression	0.78
SVM(Linear)	0.76
SVM(Non-Linear)	0.86
Decision Tree	0.75
Random Forest	**0.87**

From the tabulation, RANDOM FOREST algorithm is the final model based on the R^2 value with better performance.

9. CONCLUSION

The purpose of this work is to establish the methodological procedure for the prediction of insurance charges using regression algorithms like Multiple linear regression, Support vector machine, Decision tree and Random forest. Random forest algorithm is the best performance model for future prediction based on r squared value.