*An Internship Project Report*

**On**

# INSURANCE COST PREDICTION
## Using Machine Learning Models

**~ Submitted By ~**

Karan Dalu

Pranay Nagbanshi

Umang Mehta

Hrithik Talwatkar

**~ Guided by ~**

Mr. Gurvansh Singh

M.Tech.

Knowledge Solutions India.

**(2020-21)**

# Table of content

| | |
|---|---|
| Abstract | 4 |
| 1. Introduction | 5-8 |
| 1.1Introduction | 5 |
| 1.2 Problems and Issues in Supervised learning | 6 |
| 2. Software Libraries | 9-11 |
| 2.1 Libraries Required for MLR | 9 |
| 2.2 Libraries Required for RFR | 9 |
| 2.3 Libraries Required for MLR with PCA | 10 |
| 2.4 Libraries Required for RFR with PCA | 11 |
| 3. Algorithm | 12-15 |
| 3.1Multiple Linear Regressor | 12 |
| 3.2 Assumptions in Multiple Linear Regressor Model | 12 |
| 3.3 Ensemble Learning | 13 |
| 3.4 Random Forest Regressor | 13 |
| 3.5 Principle Component Analysis | 15 |
| 4. Conclusion | 16 |
| 5. Code | 17-33 |
| 5.1 MLR | 17 |
| 5.2 RFR | 21 |
| 5.3 MLR with PCA | 26 |
| 5.4 RFR with PCA | 31 |

## Table of figures/graphs/table

Fig.1: The Structure of RFR 14

Table 1: Comparison of Algorithms 16

Graph 1: MLR 21

Graph 2: RFR 26

Graph 3: MLR with PCA 30

Graph 4: RFR with PCA 33

## Abstract

In this project, three types of machine learning: Supervised Learning, Unsupervised learning and Reinforcement Learning are studied. The aim of the project was to make accurate Prediction of Insurance cost. For this, Supervised Learning Algorithms are used. For algorithm selection, the problems faced are discussed and then the model is created with four algorithms named Multiple Linear Regression (MLR), Random Forest Regression (RFR), Multiple Linear Regression with Principal Component Analysis (MLR with PCA) and Random Forest Algorithm with Principal Component Analysis (RFR with PCA). The Libraries used in all the algorithms are mentioned in the report. MLR, RFR and PCA are explained in brief.

It is observed that RFR with PCA predict the insurance cost more accurately than the other three. So, it is concluded that RFR with PCA is the most accurate algorithm among the four algorithms.

# 1. Introduction

## 1.1 Introduction

Machine learning is a sub-domain of computer science which evolved from the study of pattern recognition in data, and also from the computational learning theory in artificial intelligence.

Machine Learning can be thought of as the study of a list of sub-problems, viz: decision making, clustering, classification, forecasting, deep-learning, inductive logic programming, support vector machines, etc.

There are three types of learning: Supervised Learning, Unsupervised learning and Reinforcement Learning. These are explained in brief below:

### a) Supervised Learning

In Supervised learning, we have a training set, and a test set. The training and test set consists of a set of examples consisting of input and output vectors, and the goal of the supervised learning algorithm is to infer a function that maps the input vector to the output vector with minimal error. In layman's terms, supervised learning can be termed as the process of concept learning, where a brain is exposed to a set of inputs and result vectors and the brain learns the concept that relates said inputs to outputs. A wide arrays of supervised machine learning algorithms are available for Supervised Learning. For example, Decision Trees, Support Vector Machines, Random Forest, etc. Each have their own merits and demerits. There is no single algorithm that works for all cases. In this project, four supervised learning algorithms will be used to check the accuracy of the output and thereby identify the best algorithm for the insurance prediction from them.

### b) Unsupervised Learning

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human-labelled data, unsupervised learning, also known as self-organization allows for modelling of probability densities over inputs.

Two of the main methods used in unsupervised learning are principal component and cluster analysis. Cluster analysis is used in unsupervised learning to group, or segment, datasets with shared attributes in order to extrapolate algorithmic relationships.[2] Cluster analysis is a branch of machine learning that groups the data that has not been labelled, classified or categorized. Instead of responding to feedback, cluster analysis identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data. This approach helps detect anomalous data points that do not fit into either group.

**c) Reinforcement Learning**

Reinforcement learning is studied in many disciplines, such as Game Theory, Control Theory, Operations Research, Information Theory, Simulation-based Optimization, Multi-agent systems, Swarm intelligence, and Statistics. In the operations research and control literature, reinforcement learning is called approximate dynamic programming, or neuro-dynamic programming. The problems of interest in reinforcement learning have also been studied in the theory of optimal control, which is concerned mostly with the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

## 1.2 Problems and Issues in Supervised learning

The problems and issues in Supervised learning are explained below

a) Heterogeneity of Data

Many algorithms like neural networks and support vector machines like their feature vectors to be homogeneous numeric and normalized. The algorithms that employ distance metrics are very sensitive to this, and hence if the data is heterogeneous, these methods should be the afterthought.

b) Redundancy of Data

If the data contains redundant information, i.e. contain highly correlated values, then it's useless to use distance-based methods because of numerical instability. In this case, some sort of Regularization can be employed to the data to prevent this situation.

c) Dependent Features

If there is some dependence between the feature vectors, then algorithms that monitor complex interactions like Neural Networks and Decision Trees fare better than other algorithms.

d) Bias-Variance Trade-off

A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x, whereas a learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier can be related to the sum of bias and variance of the learning algorithm, and neither can be high as they will make the prediction error to be high.

e) Curse of Dimensionality

If the problem has an input space that has a large number of dimensions, and the problem only depends on a subspace of the input space with small dimensions, the machine learning algorithm can be confused by the huge number of dimensions and hence the variance of the algorithm can be high.

f) Overfitting

The programmer should know that there is a possibility that the output values may constitute of an inherent noise which is the result of human or sensor errors. In this case, the algorithm must not attempt to infer the function that exactly matches all the data. Being too careful in fitting the data can cause overfitting, after which the model will answer perfectly for all training examples but will have a very high error for unseen samples.

In this project, we have to make accurate Prediction of Insurance cost. In the data set we have seven columns, named: age, sex, bmi, no of children insurance holder has, smoker (Insurance holder smokes or not), Region (In which region the Insurance holder lives in USA) and the charges of insurance.

For prediction of insurance cost, we have to make four models using four algorithms and check which one of them is the most accurate, and also plot their graph.

The four algorithms to be used are:

a) Multiple Linear regressor (MLR)
b) Random Forest regressor (RFR)
c) Multiple Linear Regressor (MLR)with Principle Component Analysis (PCA)
d) Random Forest Regressor (RFR) with Principle Component Analysis (PCA)

# 2. Software-libraries

## 2.1 Libraries Required for MLR

The libraries required for Multiple Linear Regressor are as follows:

a) Numpy
b) Pandas
c) Matplotlib
d) Scipy
e) Scikit Learn

### 2.1.1 Models of Scikit Learn Library

The models used under MLR are as follows:

a) Pre-processing
b) Model Selection
c) Linear Model
d) Metrics

### 2.1.2 Classes of Scikit Learn library

a) Label Encoder

b) Train Test Split

c) Linear Regression

d) Mean Squared Error

## 2.2 Libraries Required for RFR

The libraries required for Random Forest Regressor are as follows:

a) Numpy
b) Pandas
c) Warnings
d) Scikit Learn
e) Seaborn
f) Matplotlib

### 2.2.1 Models of Scikit Learn Library

The models used under RFR are as follows:

a) Pre-Processing
b) Model Selection
c) Metrics
d) Ensemble

### 2.2.2 Classes of Scikit Learn library

a) Label Encoder
b) Train Test Split
c) Polynomial Features
d) R2 Score
e) Mean Squared Error
f) Random Forest Regressor

## 2.3 Libraries Required for MLR with PCA

The libraries required for Multiple Linear Regressor with Principal Component Analysis are as follows:

a) Numpy
b) Pandas
c) Scikit Learn
d) Matplotlib, Matplotlib Toolkits
e) Stats model

### 2.3.1 Models of Scikit Learn Library

The models used under MLR with PCA are as follows:

a) Pre-processing
b) Compose
c) Model selection
d) Linear Model
e) Metrics

### 2.3.2 Classes of Scikit Learn library

The classes used under model of Scikit Learn Library are as follows:

a) Column Transformer
b) One Hot Encoder
c) Train test split
d) Linear Regression

## 2.4 Libraries used in RFR with PCA

The libraries required for Random Forest regressor (RFR) with PCA are as follows:

a) Matplotlib and Matplotlib toolkit
b) Pandas
c) Numpy
d) Scikit Learn

### 2.4.1 Models of Scikit Learn Library

The models used under RFR with PCA are as follows

a) Decomposition
b) Model selection
c) Ensemble
d) Metrics
e) Pre-processing

### 2.4.2 Classes of Scikit Learn library which are required

The classes used under model of Scikit Learn Library are as follows:

a) Train test split
b) PCA
c) Random Forest Regressor
d) Mean Squared Error
e) Label Encoder

### 2.4.3 Classes of mat plot library toolkit

a) mplot3d

# 3. Algorithm

## 3.1 Multiple Linear Regressor

Multiple Linear Regression (MLR), also known simply as multiple regression is a supervised learning algorithm. It is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.

Simple linear regression is a function that allows to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables, an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends it to several explanatory variables, that is: there can me more than one independent variable but the dependent variable is limited to one only

Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable. MLR is used extensively in econometrics and financial inference.

The equation of MLR is:

$$y = a_1*x_1 + a_2*x_2+.......+ b$$

Here,

$a_1$ and $a_2$ are the coefficients

$x_1$, $x_2$ are the independent variable, and

b is the constant

## 3.2 The Assumptions in MLR model

There is a linear relationship between the dependent variables and the independent variables.

a) The independent variables are not too highly correlated with each other.

b) The observations are selected independently and randomly from the population.
c) Residuals should be normally distributed with a mean of 0 and variance σ.

### 3.3 Ensemble learning

An Ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. A model comprised of many models is called an Ensemble model. In Ensemble learning a single algorithm or multiple algorithms are taken multiple times and they are put together to get a more powerful algorithm for prediction.

#### 3.3.1 Types of Ensemble learning

Ensemble learning consist of three types as follows:

**a) Bagging**: Bagging follows parallel method. In bagging one algorithm is used multiple times. Here training data is distributed to number of algorithms formed. Selecting number of algorithms depends on user. Values given as output can be overlapped and also the number of values given to each algorithm is same. For prediction average of all the outputs are taken into consideration. Random Forest Regressor is a subset of bagging.

**b) Boosting:** Boosting follows series method. In boosting too, we use same algorithm multiple times that is the output of one algorithm is fed to the second as reference and so on till the last algorithm is executed.

**c) Stacking**: Stacking follows parallel method. The difference between bagging and stacking is that in stacking we use different algorithms instead of a single algorithm.

### 3.4 Random Forest Regressor

Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees.

It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees, with some helpful modifications:

a) The number of features that can be split on at each node is limited to some percentage of the total (which is known as the hyperparameter). This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features.
b) Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents over fitting.

The above modifications help prevent the trees from being too highly correlated.

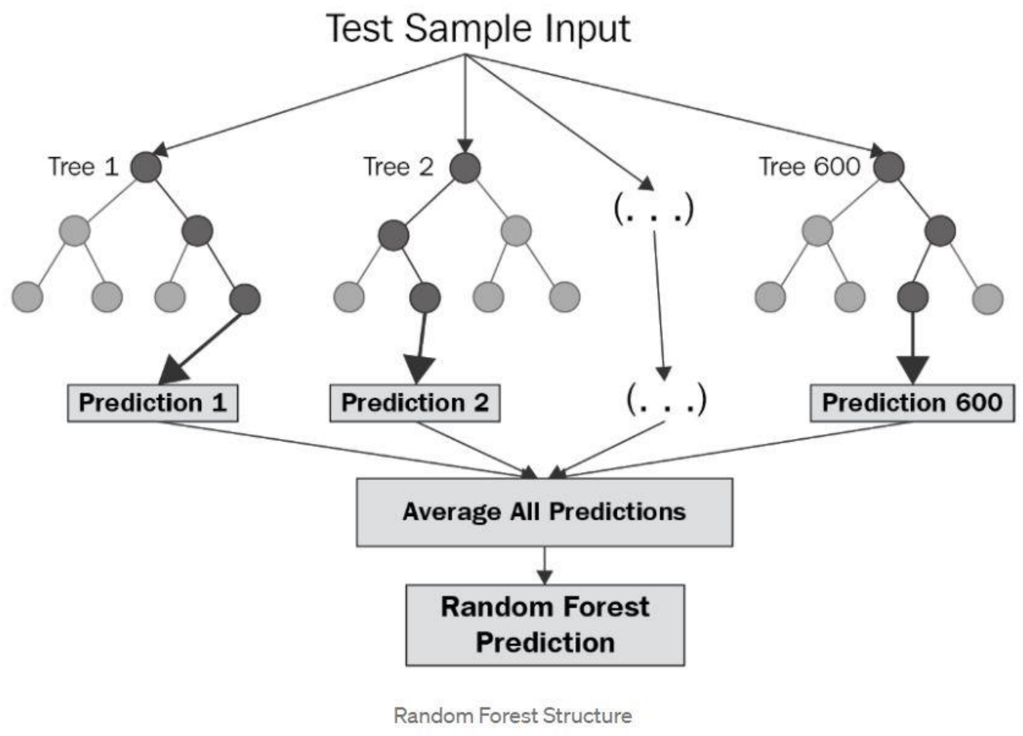


Fig. 3.1 The Structure of RFR

## 3.5 Principal Component Analysis

Principal Component Analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest, that is it is commonly used for dimensionality reduction. PCA is basically an unsupervised learning algorithm but we used it for supervised learning for getting better accuracy.

PCA refers on variance of data. We reduce the dimension of data by dropping features with low spread, preserving the data with higher spread. it can be shown that the principal components are eigenvectors of the data's covariance matrix. Thus, the principal components are often computed by eigen decomposition of the data covariance matrix or singular value decomposition of the data matrix. PCA is the simplest of the true eigenvector-based multivariate analyses and is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

## 4. Conclusion

To compare the Algorithms, codes are generated for all four algorithms. The programs are executed. The Mean Squared Error and Root Mean Squared Error of the four algorithms are obtained and shown in the table 4.1.

**Table 4.1: Comparison of Algorithms**

| **Algorithm** | **MSE value** | **RMSE value** |
|---|---|---|
| MLR | 33635210.431178406 | 5799.587091438356 |
| RFR | 19933823.142 | 4464.7310268 |
| MLR with PCA | 32171708.572455764 | 5672.010981341253 |
| RFR with PCA | 18782614.012534544 | 4333.8913244951755 |

The Algorithm with lowest MSE and RMSE values is the most accurate for prediction of the insurance cost. The table shows that the Random Forest Regressor with Principal Component Analysis has MSE value as 18782614.012534544 and RMSE value as 4333.8913244951755. As these both values are lowest than the other Algorithms, we may conclude that RFR with PCA is the most accurate algorithm for Insurance Cost Prediction.

# 5.Code

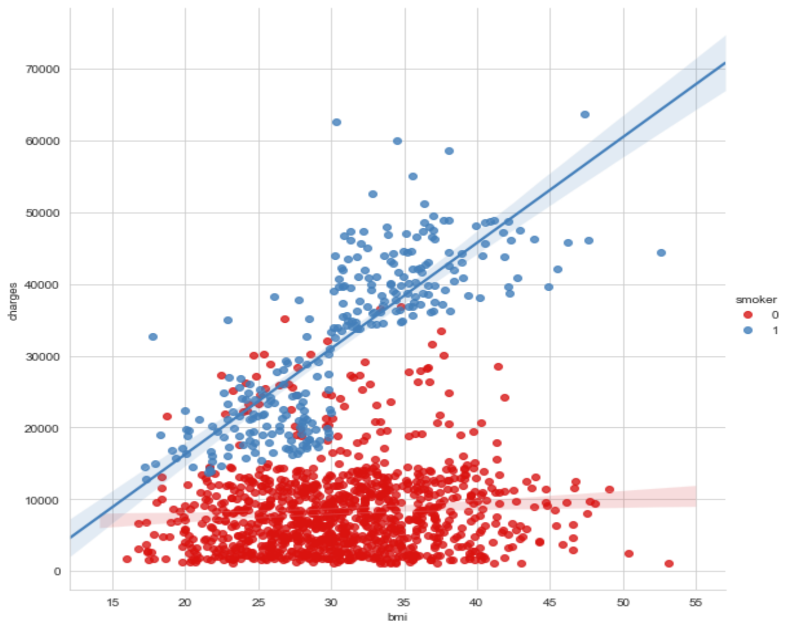
## 5.1 MLR

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv('insurance.csv')

print(df)

print(df.shape)

df.info()

df.isna()

df.isna().sum()

print(df['age'].mean())

print(df['age'].median())

print(df['bmi'].mean())

print(df['bmi'].median())

print(df['children'].mean())

print(df['children'].median())

print(df['sex'].value_counts())

print(df['smoker'].value_counts())

print(df['region'].value_counts())

print(df.select_dtypes(include=['object']).columns.values)

df.describe()

from scipy.stats import norm

from scipy import stats
```

```
plt.figure(figsize=(12,6))

sns.distplot(df['charges'], fit=norm)

fig = plt.figure(figsize=(12,6))

res = stats.probplot(df['charges'], plot=plt)

plt.figure(figsize=(12,6))

sns.distplot(df['age'], fit=norm)

fig = plt.figure(figsize=(12,6))

res = stats.probplot(df['age'], plot=plt)

plt.figure(figsize=(12,6))

sns.distplot(df['bmi'], fit=norm)

fig = plt.figure(figsize=(12,6))

res = stats.probplot(df['bmi'], plot=plt)

df.corr()['charges'].sort_values()

fig = plt.figure(figsize=(12,6))

sns.heatmap(df.corr(), annot=True)

sns.jointplot(x='bmi', y='charges', data=df, kind='kde')

sns.jointplot(x='age', y='charges', data=df, kind='hex')

x = df.iloc[:,:-1].values

y = df.iloc[:,-1].values

print(x) # It extracts Salary,Age, Country as X

print(y)

# catColumns = ['sex', 'smoker', 'region']

# df_dum = pd.get_dummies(df, columns = catColumns,
drop_first=False)

# df_dum.head()
# Import label encoder

from sklearn import preprocessing

# label_encoder object knows how to understand word labels.

label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Country'.

df['sex']= label_encoder.fit_transform(df['sex'])

df['smoker']= label_encoder.fit_transform(df['smoker'])

df['region']= label_encoder.fit_transform(df['region'])

print(df)

x = df.drop(['charges'], axis = 1)

 y = df.charges

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)

y_log_train = np.log1p(y_train)

from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(x_train,y_train)

coeff_df =
pd.DataFrame(lm1.coef_,x.columns,columns=['Coefficient'])

coeff_df

print(x_train)

print(x_test)

y_pred = lr.predict(X_test)

plt.scatter(y_test, y_pred)
```

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error

mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

print(mse)

print(mae)

print(rmse)

sns.distplot((y_test-y_pred), hist_kws=dict(edgecolor="k",
linewidth=2))

sns.lmplot(x="bmi", y="charges", hue="smoker", data=df, size = 8,
palette="Set1")

plt.figure(figsize=(12,6))

sns.boxplot(x='region', y = 'charges', data = df)

# Set theme

plt.figure(figsize=(12,6))

sns.set_style('whitegrid')

sns.boxenplot(x='sex', y='charges', data=df, palette ="plasma")

# sns.boxplot(x='sex', y = 'charges', data = df)

sns.catplot(x="smoker", kind="count",hue = 'sex', data=df)

sns.catplot(x="sex", y="charges", hue="smoker",
kind="swarm", data=df)

sns.jointplot(x='children', y='charges', data=df, kind='hex')
```

**Graph 1: MLR**



## 5.2 RFR

```
import numpy as np

import pandas as pd

import warnings

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import r2_score,mean_squared_error

from sklearn.ensemble import RandomForestRegressor
```

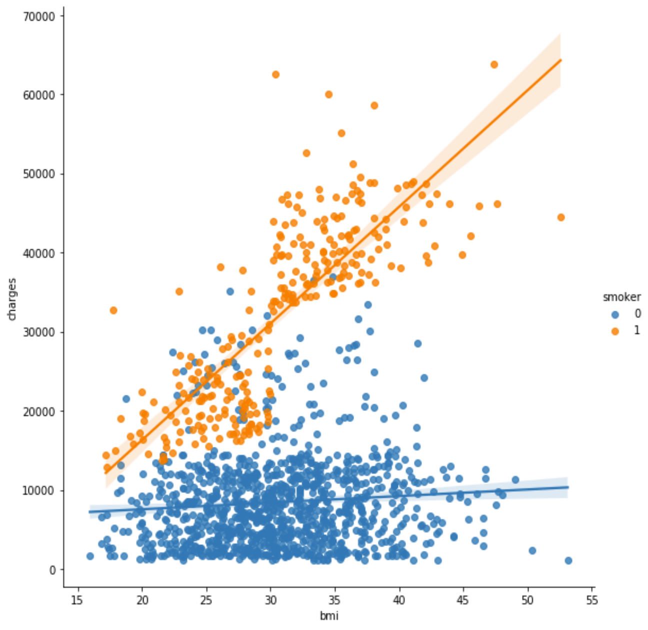
```
warnings.filterwarnings('ignore')

data = pd.read_csv('insurance.csv')

data.head(5)

data.tail(6)

data.shape

data.isnull().sum()

# Label Encoding

# Sex

lencode = LabelEncoder()

lencode.fit(data.sex.drop_duplicates())

data.sex = lencode.transform(data.sex)

data.head()

# Smoker Or Non smoker

lencode.fit(data.smoker.drop_duplicates())

data.smoker = lencode.transform(data.smoker)

#region

lencode.fit(data.region.drop_duplicates())

data.region = lencode.transform(data.region)

data.head()

data.corr()['charges'].sort_values()

f, ax = plt.subplots(figsize=(10, 8))

corr = data.corr()

sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool))

f= plt.figure(figsize=(12,5))
```

```
ax=f.add_subplot(121)

sns.distplot(data[(data.smoker == 1)]["charges"],color='c',ax=ax)

ax.set_title('Distribution of charges for smokers')

ax=f.add_subplot(122)

sns.distplot(data[(data.smoker == 0)]['charges'],ax=ax)

sns.catplot(x="smoker", kind="count",hue = 'sex', data=data)

sns.catplot(x="sex", y="charges", hue="smoker",

            kind="violin", data=data)

# The number of smokers and non-smokers (18 years old)

sns.catplot(x="smoker", kind="count",hue = 'sex',  data=data[(data.age
== 18)])

plt.figure(figsize=(12,5))

plt.title("Box plot for charges 18 years old smokers")

sns.boxplot(y="smoker", x="charges", data = data[(data.age == 18)] ,
orient="h", palette = 'pink')

# Distribution of charges and age for non-smokers

g = sns.jointplot(x="age", y="charges", data = data[(data.smoker ==
0)],kind="kde")

g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")

g.ax_joint.collections[0].set_alpha(0)

g.set_axis_labels("$X$", "$Y$")

# Distribution of charges and age for smokers

g = sns.jointplot(x="age", y="charges", data = data[(data.smoker ==
1)],kind="kde", color="c")

g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")

g.ax_joint.collections[0].set_alpha(0)
```

```
g.set_axis_labels("$X$", "$Y$")

plt.figure(figsize=(12,5))

plt.title("Distribution of bmi")

ax = sns.distplot(data["bmi"], color = 'm')

plt.figure(figsize=(12,5))

plt.title("Distribution of charges for patients with BMI greater than 30")

ax = sns.distplot(data[(data.bmi >= 30)]['charges'], color = 'm')

plt.figure(figsize=(12,5))

plt.title("Distribution of charges for patients with BMI less than 30")

ax = sns.distplot(data[(data.bmi < 30)]['charges'], color = 'm')

g = sns.jointplot(x="bmi", y="charges", data = data,kind="kde",
color="r")

g.plot_joint(plt.scatter, c="w", s=30, linewidth=1, marker="+")

g.ax_joint.collections[0].set_alpha(0)

g.set_axis_labels("$X$", "$Y$")

ax.set_title('Distribution of bmi and charges')

plt.figure(figsize=(10,6))

ax = sns.scatterplot(x='bmi',y='charges',data=data,hue='smoker')

ax.set_title('Scatter plot of charges and bmi')

sns.lmplot(x="bmi", y="charges", hue="smoker", data=data, size = 8)

sns.catplot(x="children", kind="count", data=data, size = 6)

sns.catplot(x="smoker", kind="count",hue = "sex",

            data=data[(data.children > 0)], size = 6

X = data.drop(['charges'], axis = 1)

y = data.charges
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, random_state = 0)
forest = RandomForestRegressor(n_estimators = 100,
                              criterion = 'mse',
                              random_state = 1,
                              n_jobs = -1)
forest.fit(X_train,y_train)
forest_train_pred = forest.predict(X_train)
forest_test_pred = forest.predict(X_test)

print('MSE train data: %.3f, MSE test data: %.3f' % (
mean_squared_error(y_train,forest_train_pred),
mean_squared_error(y_test,forest_test_pred)))
print('R2 train data: %.3f, R2 test data: %.3f' % (
r2_score(y_train,forest_train_pred),
r2_score(y_test,forest_test_pred)))
```

## Graph 2: RFR



## 5.3 MLR with PCA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
df = pd.read_csv('insurance.csv')
print(df)
df = pd.read_csv('insurance.csv')
print(df)
df.info()
df.isna().sum()
x = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

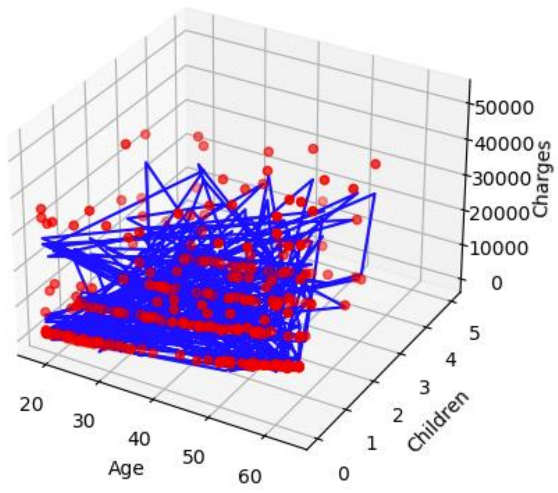
```
print(x)

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer
(transformers=[('encoder',OneHotEncoder(),[1,4,5])],remainder =
'passthrough')

x = ct.fit_transform(x)

print(x)

print(x[0,:])

from sklearn.model_selection import train_test_split

x_tr,x_te,y_tr,y_te = train_test_split(x,y,test_size =0.2,random_state =
0)

print(x_te)

x_temp = x[:,1:]

print(x_temp)

x_temp.shape

const = np.ones((1338,1))

x_temp = np.append(arr = const, values=x_temp,axis=1)

print(x_temp)

x_opt = np.array(x_temp[:,[0,1,2,3,4,5,6,7,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

x_opt = np.array(x_temp[:,[0,2,3,4,5,6,7,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())
```

```
x_opt = np.array(x_temp[:,[0,2,3,4,6,7,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

x_opt = np.array(x_temp[:,[2,3,4,6,7,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

x_opt = np.array(x_temp[:,[2,3,6,7,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

x_opt = np.array(x_temp[:,[2,3,6,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

x_opt = np.array(x_temp[:,[2,3,8,9,10]], dtype=float)

stats = sm.OLS(endog = y, exog=x_opt).fit()

print(stats.summary())

from sklearn.linear_model import LinearRegression

x_train,x_test,y_train,y_test = train_test_split(x_opt,y,test_size
=0.2,random_state = 0)

regressor1 = LinearRegression()

regressor1.fit(x_train,y_train)

print(x_test)

y_pred1 = regressor1.predict(x_test)

from sklearn.metrics import mean_absolute_error,mean_squared_error

mae1 = mean_absolute_error(y_test,y_pred1)

mse1 = mean_squared_error(y_test,y_pred1)
```

```
rmse1 = np.sqrt(mse1)
print(mae1)
print(mse1)
print(rmse1)
%matplotlib notebook
from mpl_toolkits import mplot3d
ax = plt.figure().add_subplot(projection = '3d')
ax.scatter3D(x_test[:,2],x_test[:,4],y_test,c='red')
ax.plot3D(x_test[:,2],x_test[:,4],y_pred1,c='blue')
ax.set_xlabel('Age')
ax.set_ylabel('Children')
ax.set_zlabel('Charges')
plt.show()
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
x_opt = pca.fit_transform(x_opt)
PVE = pca.explained_variance_ratio_
print(PVE)
print(x_opt)
from sklearn.preprocessing import LabelEncoder
rt = LabelEncoder()
y_train = rt.fit_transform(y_train)
y_test = rt.fit_transform(y_test)
from sklearn.svm import SVC
```

```
classifier = SVC(kernel='linear',random_state=0)

classifier.fit(x_train,y_train)

y_pred2 = classifier.predict(x_test)

from sklearn.metrics import accuracy_score,confusion_matrix

acc = accuracy_score(y_test,y_pred2)

cm = confusion_matrix(y_test,y_pred2)

print(acc)

print(cm)
```

**Graph 3: MLR with PCA**



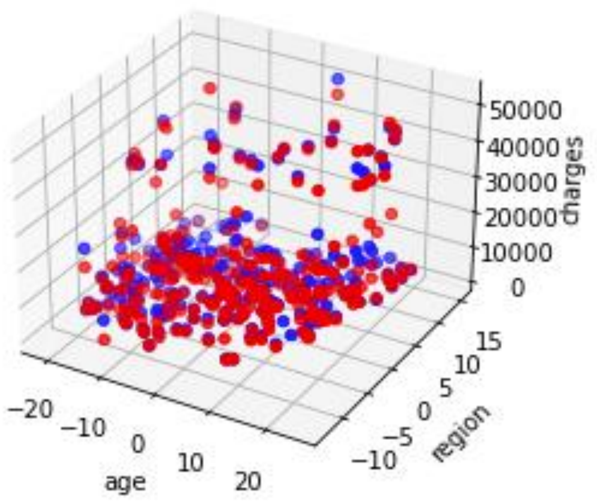
## 5.4 RFR with PCA

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df = pd.read_csv('insurance.csv')
print(df)
df.info()
print(df['region'].value_counts())
print(df['sex'].value_counts())
print(df['smoker'].value_counts())
from sklearn.preprocessing import LabelEncoder
lencode = LabelEncoder()
lencode.fit(df.sex.drop_duplicates())
df.sex = lencode.transform(df.sex)
lencode.fit(df.smoker.drop_duplicates())
df.smoker = lencode.transform(df.smoker)
lencode.fit(df.region.drop_duplicates())
df.region = lencode.transform(df.region)
df.head
x = df.iloc[:,[0,1,2,3,4,5]]
print(x)
y = df.iloc[:,-1]
print(y)
from sklearn.decomposition import PCA
```

```
pca = PCA()

x = pca.fit_transform(x)

PVE = pca.explained_variance_ratio_

print(PVE)

pca = PCA(n_components = 2)

from sklearn.model_selection import train_test_split

x_tr,x_te,y_tr,y_te = train_test_split(x,y,test_size = 0.2, random_state =
0)

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 1000, random_state
= 0)

regressor.fit(x_tr, y_tr)

y_pred = regressor.predict(x_te)

from mpl_toolkits import mplot3d

ax = plt.figure().add_subplot(projection = "3d")

ax.scatter3D(x_te[:,0],x_te[:,1],y_te,c = 'red')

ax.scatter3D(x_te[:,0],x_te[:,1],y_pred,c = 'blue')

ax.set_xlabel('age')

ax.set_ylabel('region')

ax.set_zlabel('charges')

plt.show()

from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_te,y_pred)

rmse = np.sqrt(mse)

r2 = r2_score(y_te,y_pred)
```

```
print(mse)

print(rmse)

print(r2)
```

**Graph 4: RFR with PCA**


charges
region
age