Summer Internship Project Report

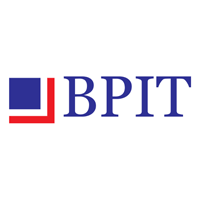
On

**INSURANCE COST PREDICTION**

*Submitted by*

**Tripti Vijayan**

**Shivansh Aggarwal**



**Bhagwan Parshuram Institute of Technology**

**Under the guidance of**

**Mr. Gurvansh Singh**

M.Tech

Knowledge Solutions India



**Knowledge Solutions India**

Dhole Patil Rd, Pune, Maharashtra

[www.ksindia.co.in](http://www.ksindia.co.in)

2020-21

**ACKNOWLEDGEMENT**

Presentation inspiration and motivation have always played a key role in the success of any venture. The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people who made it possible. With deep gratitude, we acknowledge all those guidance and encouragement, which served as beacon of lights and crowned our efforts with success.

We are feeling obliged in taking the opportunity to sincerely thank our project guide **Mr. Gurvansh Singh,** M.Tech, for his guidance and valuable suggestions which helped us to fulfill the experiments prescribed by the management to complete this work successfully.

We are extremely thankful to **Knowledge Solutions India** for giving us so much to work on their team. We can confidently say we wouldn’t have grown and learned as much as we have in the past 1 month without your constant mentorship. We are so grateful for all your advice and specially we all enjoyed learning in your organization.

At last we would also like to extend our gratitude to all the people who have contributed to the successful completion of the project.

**ABSTRACT**

Machine Learning is a type of Artificial Intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine Learning focuses on development of Computer Programs that can change when exposed to new data.

The project is coded in python language. Python is a general purpose and high level programming language. We can use Python for developing desktop GUI applications, websites and web applications. Also, Python as a high level programming language, allows us to focus on core functionality of the application by taking care of common programming tasks.

Given a dataset of information on people regarding Age, BMI, Sex, Region of living, etc, Can we predict the charges of insurance ? This project looks at ML models to predict the ‘Insurance cost’ with minimum MSE and RMSE and maximum R-Square score. The two models that are used are Multiple Linear Regression (MLR) and Random Forest Regression (RFR) models. In order to use the models we first preprocessed the data. We were able to produce required results i.e. Insurance Cost while minimizing MSE and RMSE.

**TABLE OF CONTENTS**

**TITLE PAGE NO.**

**1. INTRODUCTION 3**

**1.1** Introduction to Machine Learning 3

**1.2** Introduction to the Project 5

**2. IMPLEMENTATION 6**

**2.1** Hardware used 6

**2.2** Software used 6

2.2.1 Operating System 6

2.2.2 Languages used 6

**2.3** Libraries 7

**3. MODELS 12**

**3.1** Linear Regression 12

3.1.1 Single Linear Regression 12

3.1.2 Multiple Linear Regression 13

**3.2** Random Forest Regression 13

**3.3** Principal Component Analysis 15

**4. CODE 19**

**4.1** Multiple Linear Regression 19

**4.2** Random Forest Regression 36

**5. CONCLUSION 44**

**6. REFERENCES 44**

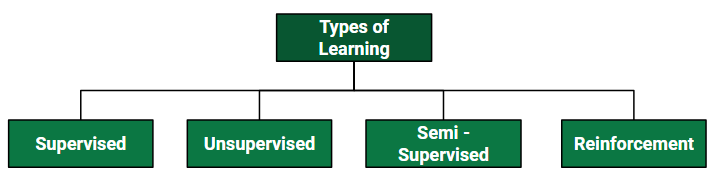
**CHAPTER 1**

**INTRODUCTION**

1. **INTRODUCTION TO MACHINE LEARNING**

**Machine learning** is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal”(training data) or "feedback" available to the learning system:



* **Supervised machine learning** algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* **Unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
* **Semi-supervised machine learning** algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy.
* **Reinforcement machine learning** algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

**INTRODUCTION**

1. **INTRODUCTION TO THE INSURANCE COST PREDICTION PROJECT**

In the Insurance Cost Prediction Project, we will be using the ‘insurance’ dataset in csv format to predict the ‘Insurance costs’ of several clients on the basis of their age, sex, BMI, number of children and other personal data. The implementation of this project will be done on the Jupyter Notebook using Python Programming Language. To predict the output with the highest accuracy, we will be employing a series of steps including Exploratory Data Analysis (EDA), Data Preprocessing, feature scaling, Data Encoding and transformation, feature selection, model selection, training the model including several others, which we will go through in detail in the upcoming sections.

**Information about the dataset (‘insurance.csv’)**

The provided dataset contains the following parameters:

1. Age: age of primary beneficiary
2. Sex: insurance contractor gender, female, male
3. BMI: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height,  
   objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9
4. Children: Number of children covered by health insurance / Number of dependents
5. Smoker: Smoking or not
6. Region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.
7. Charges: Individual medical costs billed by health insurance

Our project mainly focuses on Supervised Machine Learning (MLR and RFR) and Unsupervised Machine Learning (MLR with PCA and RFR with PCA) both of which have been explained above.

**CHAPTER 2**

**IMPLEMENTATION**

**2.1 Hardware used**

Laptop/PC

Processor : Intel® Core™2 Duo

RAM : 4.00 GB

**2.2 Software used**

Operating System : Windows 8.1 Pro

Programming Language : Python

IDE : Jupyter Notebook

**Programming Languages used:**

**Python:**

**Python** is an interpreted, high-level and general-purpose programming language. It is simple, intuitive, provides Platform independence.

Implementing AI and ML algorithms can be tricky and requires a lot of time, but Python comes with an extensive range of libraries and frameworks for Machine learning such as Keras, TensorFlow, and Scikit-learn for machine learning, SciPy for advanced computing, Seaborn for data visualization, Pandas for general-

purpose data analysis, NumPy for scientific computing and data analysis.

**IDE:**

**The Jupyter Notebook:**

The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

* 1. **LIBRARIES USED IN PYTHON**

**We have made use of the following libraries in our project**

* **Pandas**

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install pandas** or **pip install pandas**

* + Import: using the command

**import pandas as pd**

* **Numpy**

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install numpy** or **pip install numpy**

* + Import: using the command

**import numpy as np**

* **Matplotlib**

Matpoltlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc.

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install matplotlib** or **pip install matplotlib**

* + Import: using the command

**import matplotlib.pyplot as plt**

* **Scikit-learn**

Skikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis.

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install scikit-learn** or **pip install scikit-learn**

* + Import: using the command

**import sklearn**

* **Mlxtend**

**Mlxtend (machine learning extensions) is a Python library of useful tools for the day-to-day data science tasks**

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install mlxtend** or **pip install mlxtend**

* + Import: using the command

**import mlxtend**

* **Seaborn**

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots

**Installation and Importing of the library:**

* + Installation: using either of the following commands

**conda install seaborn** or **pip install seaborn**

* + Import: using the command

**import seaborn as sns**

**Some other libraries used widely for Machine Learning**

* **Scipy**

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. The SciPy library is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

* **TensorFlow**

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

* **PyTorch**

PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library which is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that supports on Computer Vision, Natural Language Processing (NLP) and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

* **Keras**

Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

* **Theano**

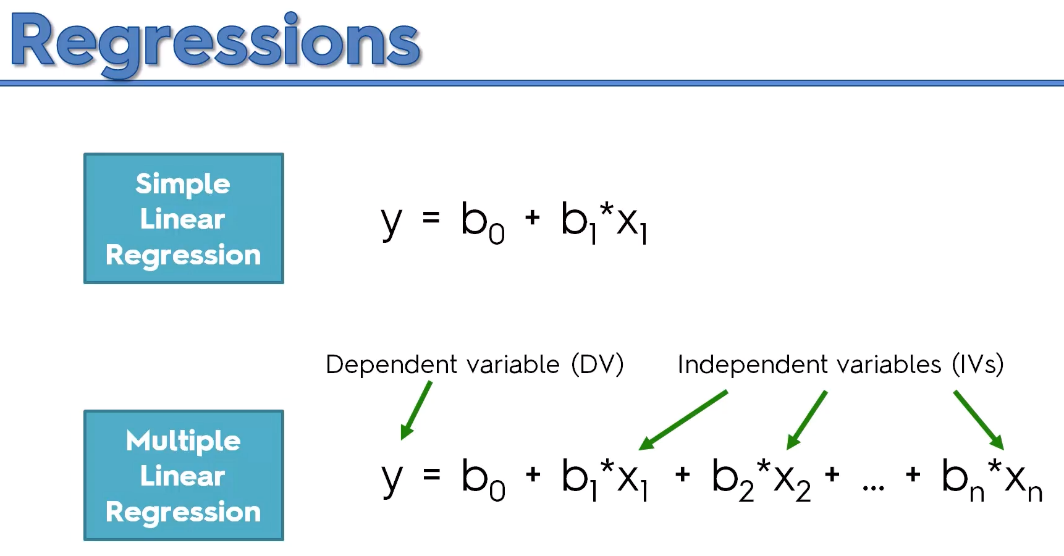
Theano is a popular python library that is used to define, evaluate and optimize mathematical expressions involving multi-dimensional arrays in an efficient manner. It is achieved by optimizing the utilization of CPU and GPU. It is extensively used for unit-testing and self-verification to detect and diagnose different types of errors.

**CHAPTER 3**

**MACHINE LEARNING MODELS**

**3.1 LINEAR REGRESSION MODEL**

**Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering and the number of independent variables being used. Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). As such, both the input values (x) and the output value are numeric.



* **Simple linear regression :**

A single independent variable is used to predict the value of a dependent variable.

{\displaystyle Y\_{i}=\beta \_{0}+\beta \_{1}X\_{i1}+\beta \_{2}X\_{i2}+\ldots +\beta \_{p}X\_{ip}+\epsilon \_{i}}Equation for Single Linear Regression:

  y = b0 + b1x1

Following is the description of the parameters used −

* + **y** is the response variable/predicted value of the dependent variable (y) for any given value of the independent variable (x).
  + **x** is the dependent variable.
  + **b0** is the intercept, predicted value of the dependent variable (y) when x is 0.
  + **b1** is the regression coefficient, change in the value of y with increase in x.
* **Multiple linear regression :**

Two or more independent variables are used to predict the value of a dependent variable. The difference between the two is the number of independent variables.

Multiple linear regression is a generalization of [simple linear regression](https://en.wikipedia.org/wiki/Simple_linear_regression) to the case of more than one independent variable, and a [special case](https://en.wikipedia.org/wiki/Special_case) of general linear models, restricted to one dependent variable. The basic model for multiple linear regression is

{\displaystyle Y\_{i}=\beta \_{0}+\beta \_{1}X\_{i1}+\beta \_{2}X\_{i2}+\ldots +\beta \_{p}X\_{ip}+\epsilon \_{i}}Equation for Multiple Linear Regression:

y = b0 + b1x1 + b2x2 +...bnxn

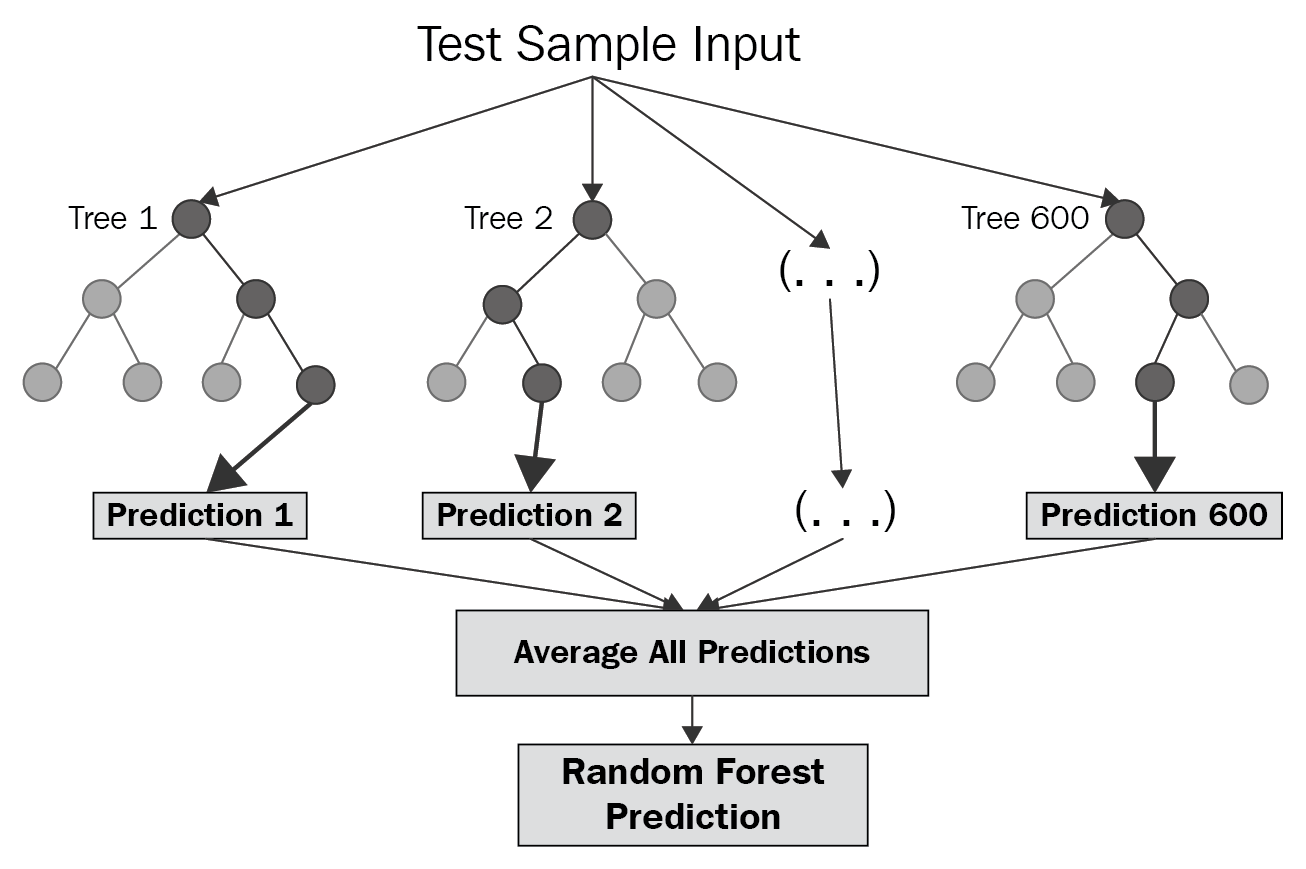
Following is the description of the parameters used −

* **y** is the response variable/predicted value.
* **b0, b1, b2...bn** are the coefficients and b0 is the intercept
* **x1, x2, ...xn** are the predictor variables.

{\displaystyle Y\_{i}=\beta \_{0}+\beta \_{1}X\_{i1}+\beta \_{2}X\_{i2}+\ldots +\beta \_{p}X\_{ip}+\epsilon \_{i}}

**3.2 RANDOM FOREST REGRESSION MODEL**

Random forests or random decision forests are an **ensemble learning** method for classification, regression and other tasks that operate by constructing a multitude of **decision trees** at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.



## **What is 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**.

# **Types of Ensemble Learning:**

1. Boosting.
2. Bootstrap Aggregation (Bagging).

## **1. Boosting**

Boosting refers to a group of algorithms that utilize weighted averages to make weak learners into stronger learners. Boosting is all about “teamwork”. Each model that runs, dictates what features the next model will focus on.

In **boosting** as the name suggests, one is learning from other which in turn **boosts** the learning.

## **2. Bootstrap Aggregation (Bagging)**

Bootstrap refers to **random sampling with replacement**. Bootstrap allows us to better **understand the bias and the variance** with the dataset. Bootstrap involves random sampling of small subset of data from the dataset.

It is a general procedure that can be used to **reduce the variance** for those **algorithm that have high variance, typically decision trees**. Bagging makes each model run independently and then **aggregates the outputs at the end without preference to any model.**

**3.3 PRINCIPAL COMPONENT ANALYSIS**

PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data.

The **principal components** of a collection of points in a real *p*-space are a sequence of **p** {\displaystyle p}direction vectors, where the **i’th** {\displaystyle i^{th}} vector is the direction of a line that best fits the data while being orthogonal to the first **{\displaystyle i-1}i-1** vectors. Here, a best-fitting line is defined as one that minimizes the average squared distance from the points to the line. **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.

The principal components are eigenvectors of the data's **covariance matrix**.

**An algorithm for conducting PCA**

### STEP 1: STANDARDIZATION

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, it is critical to perform standardization prior to PCA, because the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges, which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

Principal Component Analysis Standardization

Once the standardization is done, all the variables will be transformed to the same scale.

### STEP 2: COVARIANCE MATRIX COMPUTATION

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a p × psymmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables.

### STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the **principal components** of the data. Organizing information in principal components this way, will allow us to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as our new variables.

The eigenvectors of the Covariance matrix are actually thedirections of the axes where there is the most variance(most information) and that we call Principal Components. And eigenvalues give the amount of variance carried in each Principal Component.

By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

### STEP 4: FEATURE VECTOR

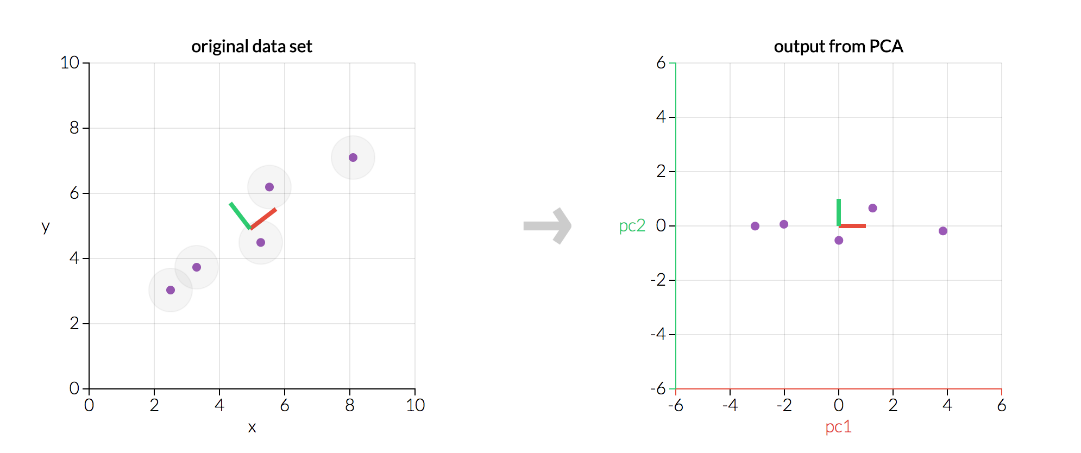
As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allows us to find the principal components in order of significance. In this step, we choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call Feature vector, that is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only **p** eigenvectors (components) out of **n**, the final data set will have only **p** dimensions.

### LAST STEP: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this last step, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal. This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

Principal Component Analysis feature vector



**CHAPTER 4**

**CODE**

**4.1 MULTIPLE LINEAR REGRESSION**

**#importing the libraries**

import numpy as np

import pandas as pd

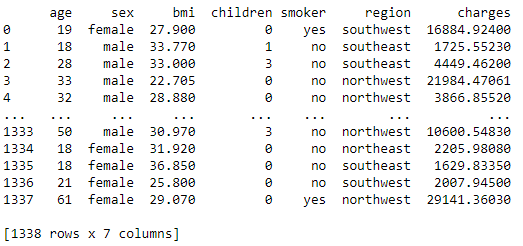
import matplotlib.pyplot as plt

import seaborn as sns

**#importing the dataset**

dataset = pd.read\_csv('insurance.csv')

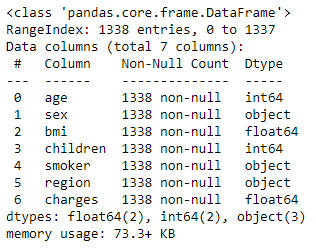
print(dataset)



**#Exploratory Data Analysis (EDA)**

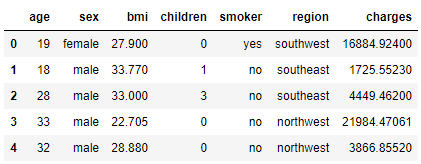
dataset.shape

dataset.info()

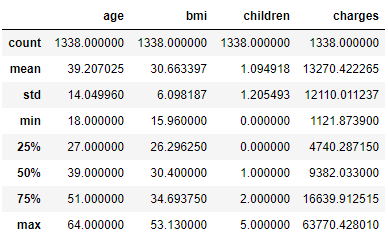


**#to return the first 5 rows of the dataframe**

dataset.head()



dataset.describe()



#### #Charges by age and sex

sns.set\_style("whitegrid", {'grid.linestyle': '--'})

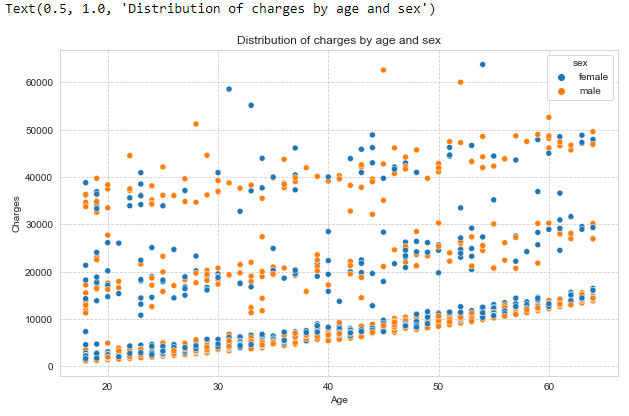
plt.figure(figsize = (10,6))

sns.scatterplot(x = "age", y = "charges", data = dataset, hue = "sex")

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Distribution of charges by age and sex")



#### #Charges by 'Smoker' and 'BMI'

smokers = dataset["smoker"].unique()

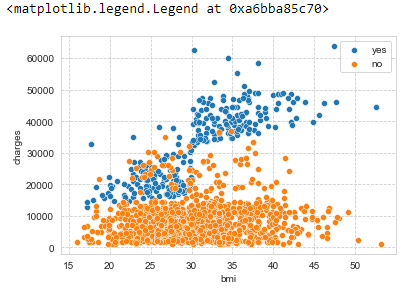
colors = ["Reds", "Greens"]

for i, smoker in enumerate(smokers):

temp = dataset[dataset["smoker"] == smoker]

sns.scatterplot(temp["bmi"], temp["charges"], cmap = colors[i])

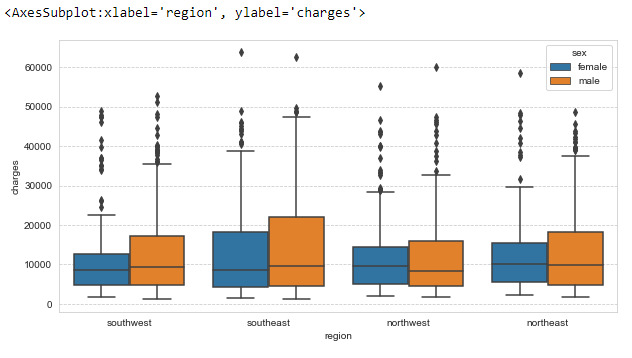
plt.legend(smokers)



#### #Charges by region and sex

plt.figure(figsize = (10,5))

sns.boxplot(x = "region", y = "charges", hue = "sex", data = dataset)



#### #Charges by age and smoker

sns.set\_style("whitegrid", {'grid.linestyle': '--'})

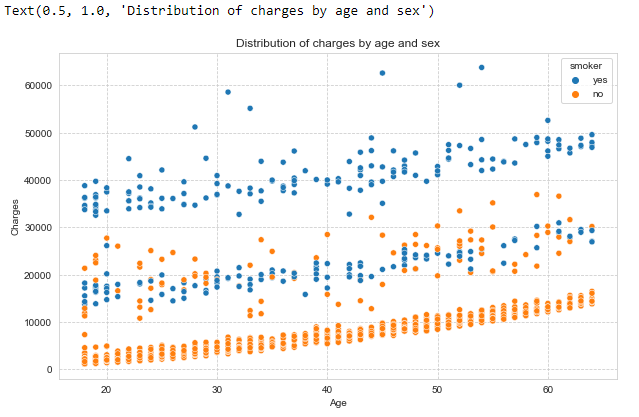
plt.figure(figsize = (10,6))

sns.scatterplot(x = "age", y = "charges", data = dataset, hue = "smoker")

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Distribution of charges by age and sex")

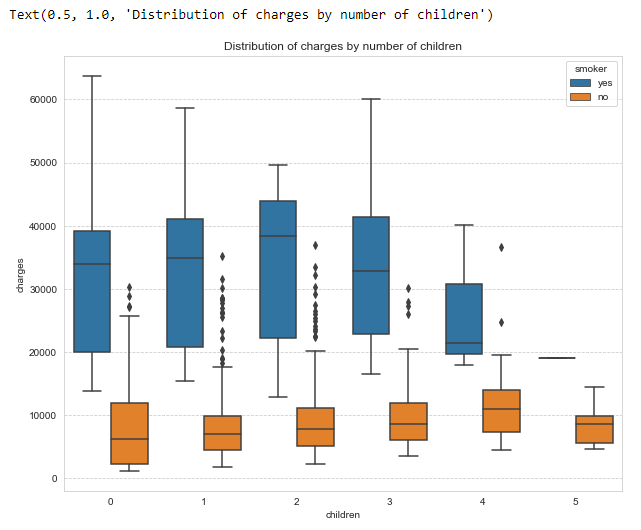


#### #Charges by smoker and number of children

plt.figure(figsize = (10,8))

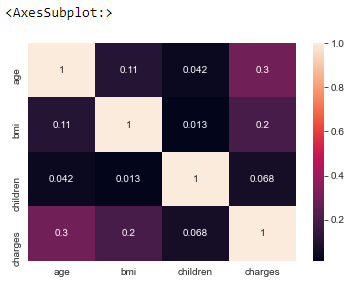
sns.boxplot(x = "children", y = "charges",hue = "smoker", data = dataset)

plt.title("Distribution of charges by number of children")



#### #Correlation between features

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



### #DATA PREPROCESSING

### #extracting x and y columns from the dataset

### x=dataset.iloc[ : , :-1].values

### y=dataset.iloc[ : ,-1].values

### print(x)

### 

### print(y)

### 

### #Encoding the categorical data

### #extracting information about the categorical data first

### dataset["sex"].value\_counts()

### 

### dataset["smoker"].value\_counts()

### 

### dataset["region"].value\_counts()

### 

### #for encoding the categorical column

### 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)

### 

### #Importing and Training the Model

### 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.linear\_model import LinearRegression

### regressor = LinearRegression()

### regressor.fit(x\_tr,y\_tr)

### 

### #to predict the values of output

### y\_pred = regressor.predict(x\_te)

### #calculate and print the errors (MSE and RMSE)

### from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

### mae=mean\_absolute\_error(y\_te, y\_pred)

### mse=mean\_squared\_error(y\_te, y\_pred)

### rmse=np.sqrt(mse)

### print(mae)

### print(mse)

### print(rmse)

### 

### #visualizing the output pair-plot for 'Age' and 'Charges'

### plt.scatter(x\_te[:,8],y\_te,c='red',label = 'Original y\_test')

### plt.scatter(x\_te[:,8],y\_pred,c='blue',label = 'Calculated y\_test')

### plt.xlabel('Age')

### plt.ylabel('Charges')

### plt.title('Age vs Charges plot')

### plt.legend()

### plt.show()

### 

### #visualizing the output pair-plot for 'BMI' and 'Charges'

### plt.scatter(x\_te[:,9],y\_te,c='red',label = 'Original y\_test')

### plt.scatter(x\_te[:,9],y\_pred,c='blue',label = 'Calculated y\_test')

### plt.xlabel('BMI')

### plt.ylabel('Charges')

### plt.title('BMI vs Charges plot')

### plt.legend()

### plt.show()

### 

## **Backward Elimination**

### Performing Backward Elimination Manually

### #preparing our data for Backward Elimination now

### #removing one dummy variable and adding the constant column

### x\_temp=x[ : ,1 : ]

### print(x\_temp)

### 

### #adding a constant column

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

### #np.append takes three values

### 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]], dtype=float)

### import statsmodels.api as sm

### stats = sm.OLS(endog = y, exog = x\_opt).fit()

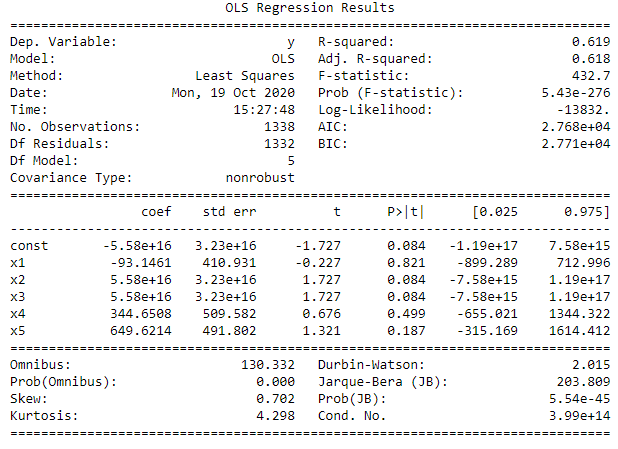
### print(stats.summary())

### 

x\_opt = np.array(x\_temp[ : , [0,1,2,3,4,6]], dtype=float) #removing column with index number 5

stats = sm.OLS(endog = y, exog=x\_opt).fit()

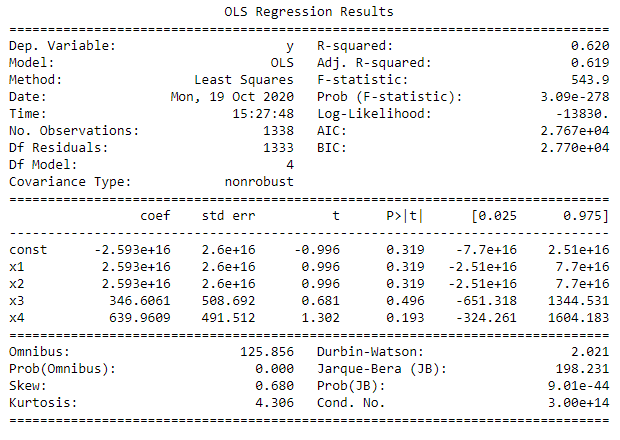
print(stats.summary())



x\_opt = np.array(x\_temp[ : , [0,2,3,4,6]], dtype=float) #removing column with index number 1

stats = sm.OLS(endog = y, exog=x\_opt).fit()

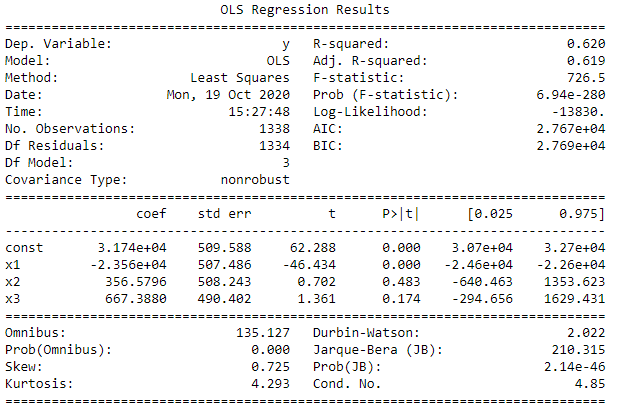
print(stats.summary())



x\_opt = np.array(x\_temp[ : , [0,2,4,6]], dtype=float) #removing column with index number 3

stats = sm.OLS(endog = y, exog=x\_opt).fit()

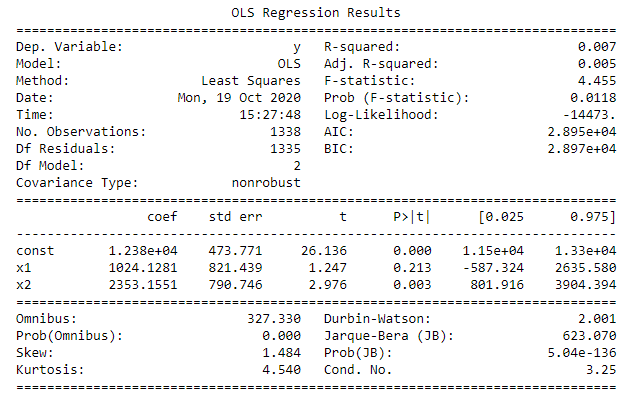
print(stats.summary())



x\_opt = np.array(x\_temp[ : , [0,4,6]], dtype=float) #removing column with index number 2

stats = sm.OLS(endog = y, exog=x\_opt).fit()

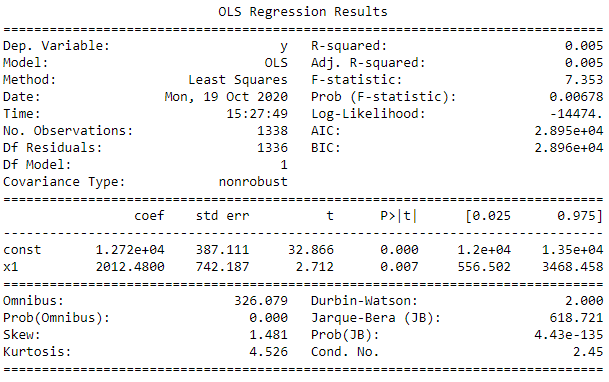
print(stats.summary())



x\_opt = np.array(x\_temp[ : , [0,6]], dtype=float) #removing column with index number 4

stats = sm.OLS(endog = y, exog=x\_opt).fit()

print(stats.summary())



**# Now again splitting and training the model after performing backward elimination**

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x\_opt,y,test\_size = 0.2,random\_state=0)

reg = LinearRegression()

reg.fit(x\_train,y\_train)



**# predicting the output and printing the errors**

y\_pr = reg.predict(x\_test)

print(mean\_absolute\_error(y\_test,y\_pr))

print(mean\_squared\_error(y\_test,y\_pr))



**4.2 RANDOM FOREST REGRESSION**

**#importing libraries**

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from sklearn.ensemble import RandomForestRegressor

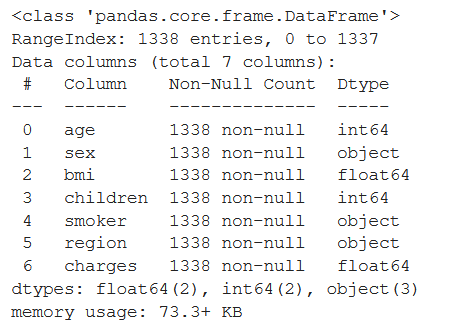
import numpy as np

**#read the csv file**

df = pd.read\_csv('/content/insurance.csv')

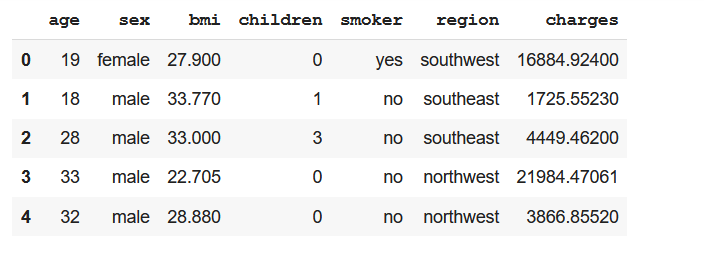
**#Information regarding data**

df.info()



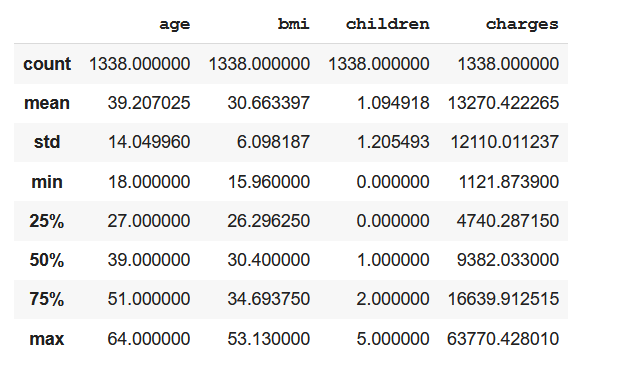
**#fetching the first five rows**

df.head()



**#describing the dataset**

df.describe()



**#Chanrges by Age and Sex**

sns.set\_style("whitegrid", {'grid.linestyle': '--'})

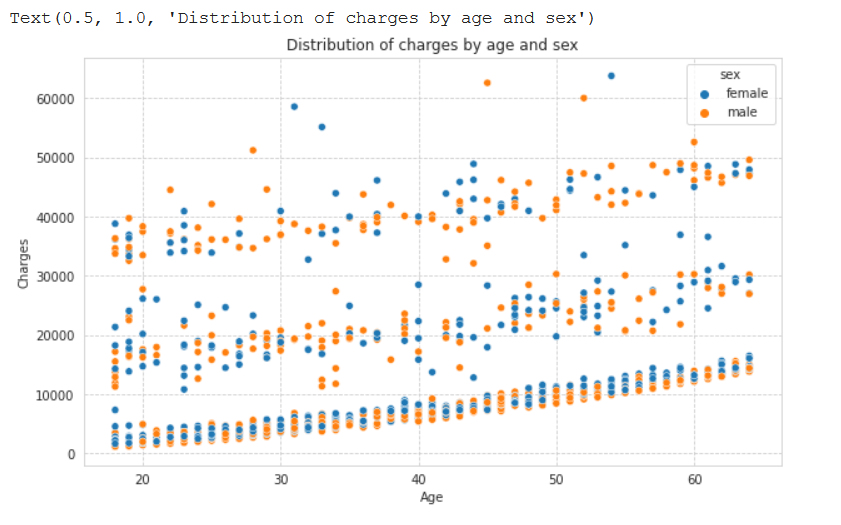
plt.figure(figsize = (10,6))

sns.scatterplot(x = "age", y = "charges", data = df, hue = "sex")

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Distribution of charges by age and sex")



#Charges by Smoker and BMI

smokers = df["smoker"].unique()

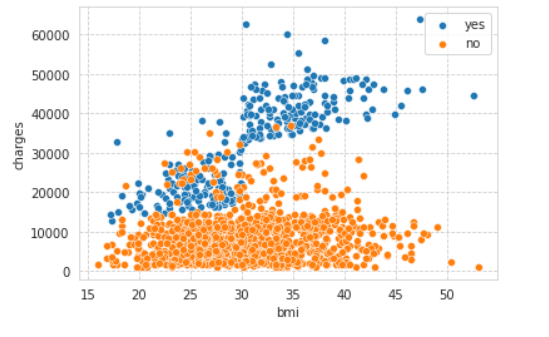
colors = ["Reds", "Greens"]

for i, smoker in enumerate(smokers):

temp = df[df["smoker"] == smoker]

sns.scatterplot(temp["bmi"], temp["charges"], cmap = colors[i])

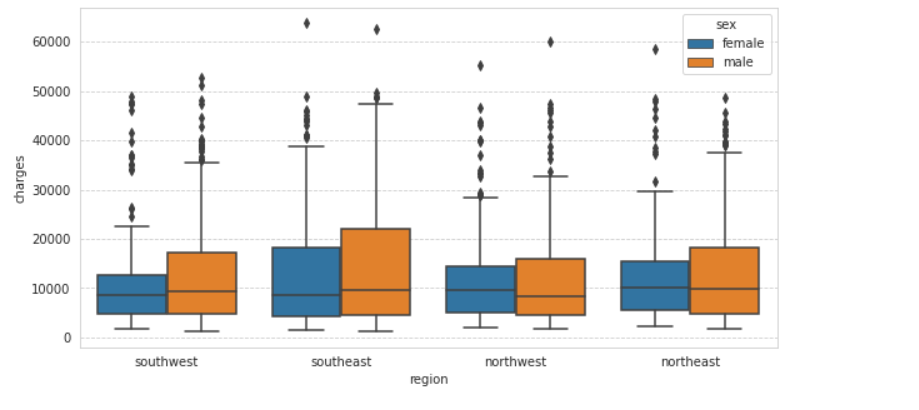
plt.legend(smokers)



**#Charges by Region and Sex**

plt.figure(figsize = (10,5))

sns.boxplot(x = "region", y = "charges", hue = "sex", data = df)



**#Charges by age and smoker**

sns.set\_style("whitegrid", {'grid.linestyle': '--'})

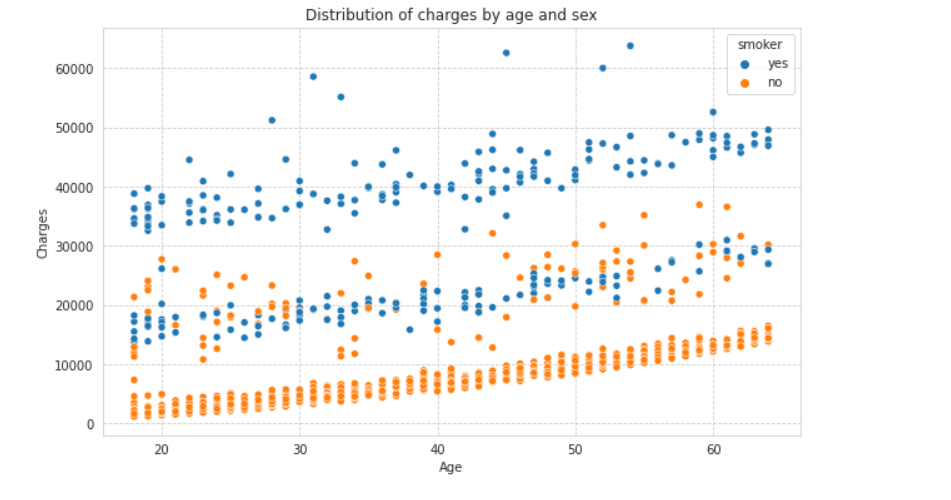
plt.figure(figsize = (10,6))

sns.scatterplot(x = "age", y = "charges", data = df, hue = "smoker")

plt.xlabel("Age")

plt.ylabel("Charges")

plt.title("Distribution of charges by age and sex")

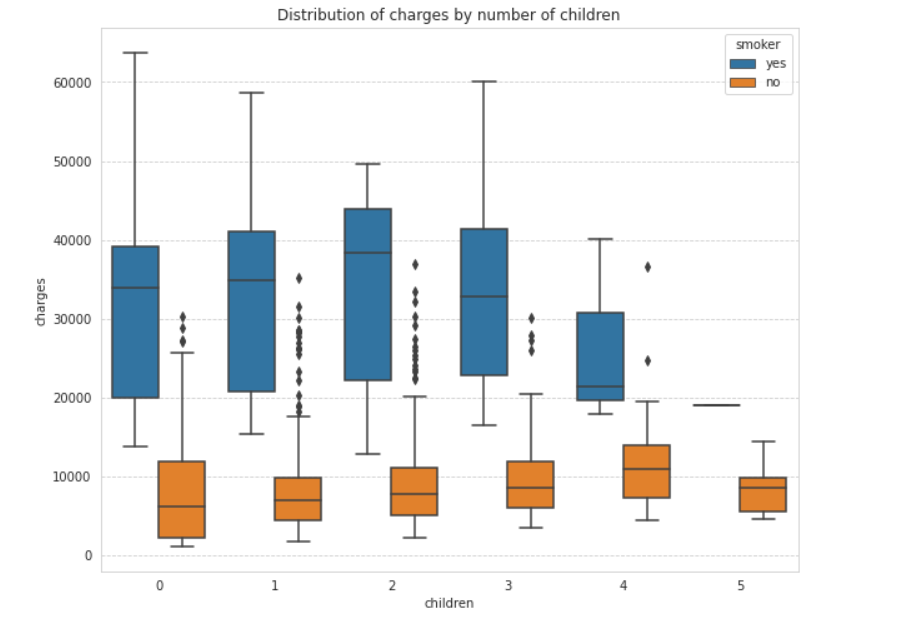


**#Charges by smoker and number of children**

plt.figure(figsize = (10,8))

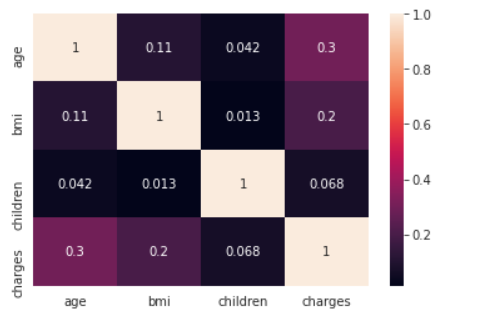
sns.boxplot(x = "children", y = "charges",hue = "smoker", data = df)

plt.title("Distribution of charges by number of children")



**#Correlation between features**

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



#convering categorical data to continuous

df['sex'] = df['sex'].map({'male' : 1, 'female' : 0})

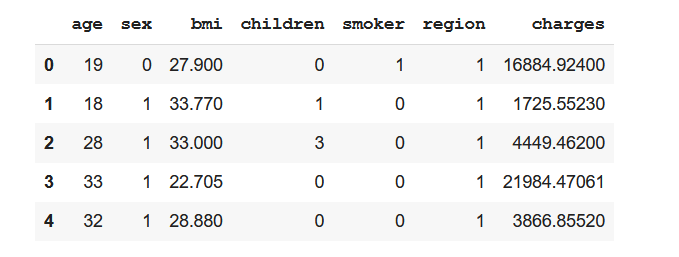
df['smoker'] = df['smoker'].map({'yes' : 1, 'no' : 0})

#Converting four region to two

df['region'] = df['region'].apply(lambda x: 1 if x == 'southeast' or 'southwest' else 0)

**#fetching first 5 rows of changed dataset**

df.head()



**#defining input and target**

Y = df['charges']

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

**#scaling the data**

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(X)

scaled\_data.shape



**#dividing data to train test set**

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(scaled\_data, Y, test\_size = 0.3, random\_state = 0)

**#defining Random Forest Regressor model**

regr = RandomForestRegressor(n\_estimators=100, max\_depth=2, random\_state=0)

#fit the model

regr.fit(X\_train, Y\_train)

#checking the accuracy

regr.score(X\_test, Y\_test)



Y\_pred = regr.predict(X\_test)

**#printing value of mean absolute error, mean squared error and rmse**

mae=mean\_absolute\_error(Y\_test, Y\_pred)

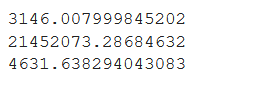
mse=mean\_squared\_error(Y\_test, Y\_pred)

rmse=np.sqrt(mse)

print(mae)

print(mse)

print(rmse)



**CHAPTER 5**

**CONCLUSION**

According to the above observations, we conclude that out of Multiple Linear Regression and Random Forest Regression Models, **RFR Model** is comparatively the best with greater accuracy and lower MSE and RMSE.

Under this project we made use of Multiple Linear Regression and Random Forest Regression Models to visualize the importance of each feature for the prediction of Insurance Costs.

This could prove to be of great help and use to Insurance Companies and beneficiaries.

**CHAPTER 6**

**REFERENCES**

"Machine Learning textbook". *www.cs.cmu.edu*. Retrieved 2020-05-28.

The definition "without being explicitly programmed" is often attributed to Arthur Samuel, who coined the term "machine learning" in 1959, but the phrase is not found verbatim in this publication, and may be a paraphrase that appeared later. Confer "Paraphrasing Arthur Samuel (1959), the question is: How can computers learn to solve problems without being explicitly programmed?" in Koza, John R.; Bennett, Forrest H.; Andre, David; Keane, Martin A. (1996).

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer

https://expertsystem.com/

https://en.wikipedia.org/wiki/Machine\_learning

https://hackernoon.com/why-python-used-for-machine-learning-u13f922ug

https://jupyter.org/

<https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/>

https://stackoverflow.com/

https://builtin.com/data-science/step-step-explanation-principal-component-analysis

https://seaborn.pydata.org/

https://www.technologyreview.com/