RV Educational Institutions ®
RV College of Engineering ®
Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi
Approved by AICTE, New Delhi, Accredited By NAAC, Bengaluru And NBA, New Delhi

Go, change the world

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

Submitted as part of assignment

MACHINE LEARNING

20MCA251

MASTER OF COMPUTER APPLICATIONS

BY

NAGARATHNA [1RV20MC054]

Under the Guidance Of

Dr. Andhe Dharani
Professor & Director

Dept of MCA, RVCE
2020-2021

RV COLLEGE OF ENGINEERING

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF
MASTER OF COMPUTER APPLICATIONS

Bengaluru-560059

CERTIFICATE

Certified that the Assignment title **"Real time Sign Language Detection using IBM Cloud"** carried out by **Nagarathna [1RV20MC054]** and **Megha M [1RV20MC048]** bonafied students of **RV College of Engineering, Bengaluru** submitted in partial fulfilment for the award **of Master of Computer Applications** of **RV College of Engineering, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi** during the year **2020-2021**.It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the partial academic requirement in respect of the course Machine Learning 20MCA251.

Faculty-In charge

Dr. Andhe Dharani,

Professor & Director,

Department of MCA,RVCE

Bengaluru-560059

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the success of any work would be incomplete unless we mention the name of the people, who made it possible, whose constant guidance and encouragement served a beacon light and served our effort with success.

We express our whole-hearted gratitude to Dr. K.N. Subramanya, Principal, and R.V. College of Engineering or providing me an opportunity and support to complete the project.

We express our special thanks to Dr. Andhe Dharani., Professor and Director, Department of MCA, R.V. College o1 Engineering for her constant support and guidance.

On a moral personal mite, our deepest appreciation and gratitude to my beloved family and friends. who have been a fountain of inspiration and have provided unrelenting encouragement and support.

Date: 04-09-2021 Nagarathna [1RV20MC054]

INDEX

Phase-I

1. PROBLEM STATMENT:

Medical insurance is a type of insurance product that specifically guarantees the health costs or care of the insurance members if they fall ill or have an accident.

The main aim of this project is to predict the cost of the Medical insurance of a person based on the some of the variables with the help of Machine learning algorithm.

Here we analyze the personal data of a person like age, sex, smoker or not etc.. for predict insurance amount for individuals Using regression models naming Multiple Linear Regression Model.

Health insurance is a necessity nowadays, and almost every individual is linked with a government or private health insurance company. Factors determining the amount of insurance vary from company to company. Also people in rural areas are unaware of the fact that the government of India provide free health insurance to those below poverty line. It is very complex method and some rural people either buy some private health insurance or do not invest money in health insurance at all. Apart from this people can be fooled easily about the amount of the insurance and may unnecessarily buy some expensive health insurance.

Our project does not give the exact amount required for any health insurance company but gives enough idea about the amount associated with an individual for his/her own health insurance.

2. Tools and Technologies Used

The Jupyter Notebook:

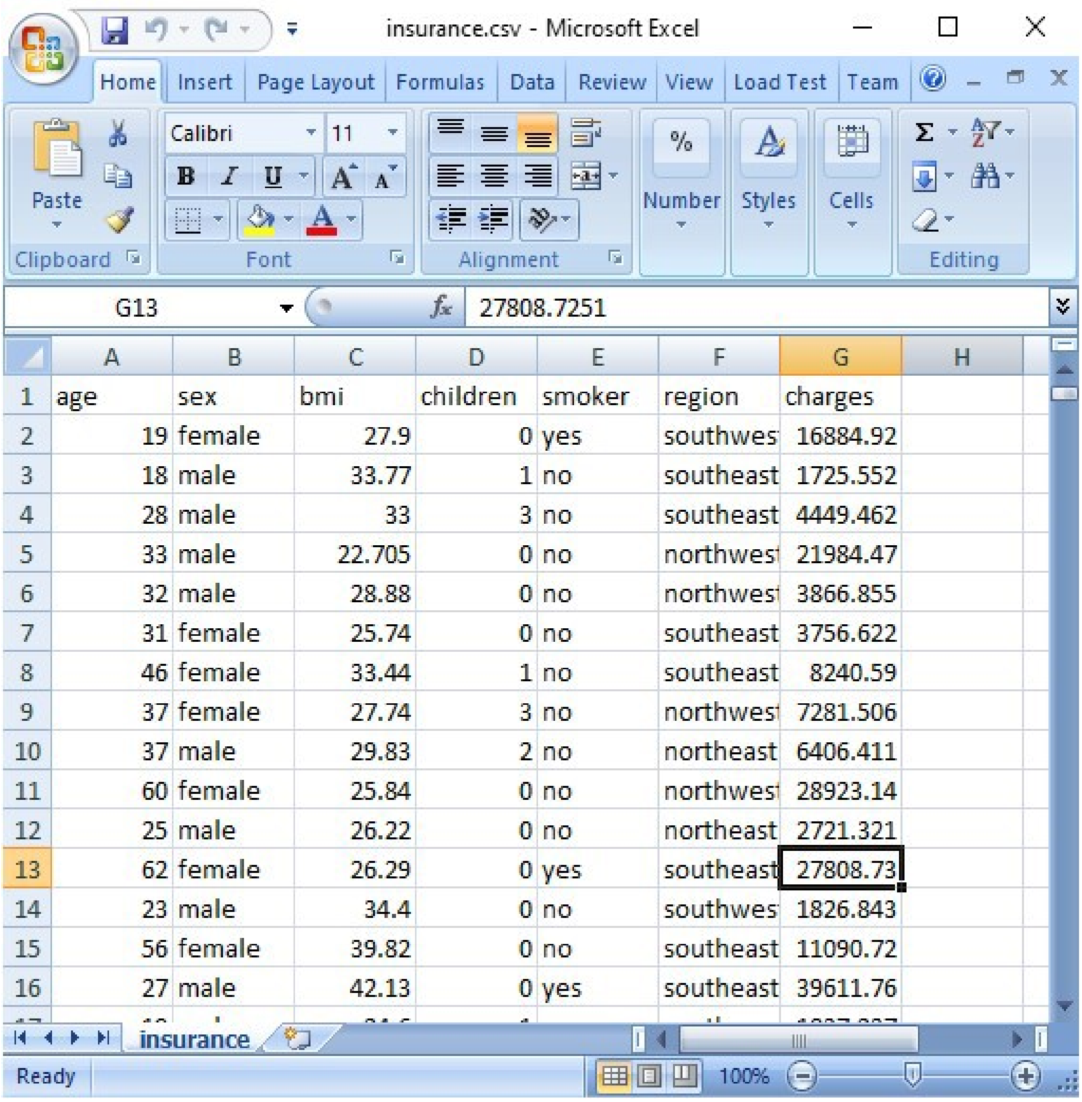
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning,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

Programming language used :

We use the Python3 for coding. Python is a powerful programming language ideal for scripting and rapid application development. It is used in web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces.

3. DATASET AND FEATURES :

The primary source of data for this project was from Kaggle. The dataset is comprised of 1338 records with 7 attributes.Attributes are as follow age, gender, bmi, children, smoker an charges as shown in below. The data was in structured format and was stores in a csv file.



	A	B	C	D	E	F	G
1	age	sex	bmi	children	smoker	region	charges
2	19	female	27.9	0	yes	southwes	16884.92
3	18	male	33.77	1	no	southeast	1725.552
4	28	male	33	3	no	southeast	4449.462
5	33	male	22.705	0	no	northwest	21984.47
6	32	male	28.88	0	no	northwest	3866.855
7	31	female	25.74	0	no	southeast	3756.622
8	46	female	33.44	1	no	southeast	8240.59
9	37	female	27.74	3	no	northwest	7281.506
10	37	male	29.83	2	no	northeast	6406.411
11	60	female	25.84	0	no	northwest	28923.14
12	25	male	26.22	0	no	northeast	2721.321
13	62	female	26.29	0	yes	southeast	27808.73
14	23	male	34.4	0	no	southwes	1826.843
15	56	female	39.82	0	no	southeast	11090.72
16	27	male	42.13	0	yes	southeast	39611.76

Columns Description

- **age:** Age of primary beneficiary.
- **sex:** Primary beneficiary's gender.
- **bmi:** Body mass index (providing an understanding of the body, weights that are relatively high or low relative to height).
- **children:** Number of children covered by health insurance / Number of dependents.
- **smoker:** Smoking (yes, no).
- **region:** Beneficiary's residential area in the US (northeast, southeast, southwest, northwest).
- **charges:** Individual medical costs billed by health insurance

4. Algorithm Analyzed

Linear Regression Algorithm :

Linear Regression is the first machine learning algorithm based on 'Supervised Learning'. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x).

When there is a single input variable (x), the method is referred to as 'Simple Linear Regression'. When there are multiple input variables, the method is referred to as 'Multivariate Linear Regression'.

Multivariate Linear Regression:

Multiple linear regression can be defined as extended simple linear regression. It comes under usage when we want to predict a single output depending upon multiple input or we can say that the predicted value of a variable is based upon the value of two or more different variables. The predicted variable or the variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable) and the variables being used in predict of the value of the dependent variable are called the independent variables.

In this dataset the dependent variable is medical charges and independent variables are age, gender, smoker ,BMI, children,region. Multiple Linear Regression uses ordinary least-squares (OLS) method to find a best fitting line which involves multiple independent variables. The formula for Multiple linear regression is as follows:

$Y=b0+ b1X1+...bkXk + \alpha$

Here, Y is dependent variable, Xi is independent variables, b0 is y-intercept (constant term), bk is slope coefficient for dependent variables, α is model error term.

Support Vector Regression:

Support Vector Regression is variant of all other models. It is used for regression and classification. In Support Vector Regression, a hyperplane is plotted to separate to predict the value of dependent variable. This line is the margin of tolerance. In regression, this hyperplane line is used to predict continuous value.

Applying on Sample data:

Let’s imagine 3 different people and see what charges on health care will be for them.

Bob: 19 years old, BMI 27.9, has no children, smokes, from northwest region.

Lisa: 40 years old, BMI 50, 2 children, doesn’t smoke, from southeast region.

John: 30 years old. BMI 31.2, no children, doesn’t smoke, from northeast region.

Prediction

After I get a regression model, I try to make a prediction using the regression model. The following is the syntax used:

def calc_insurance(age, bmi, smoking): y = ((age*linreg.coef_[0]) + (bmi*linreg.coef_[1]) + (smoking*linreg.coef_[2]) — linreg.intercept_) return y

I try to predict how much insurance costs from someone who is 34 years old, the value of BMI is 24, and not a smoker. Next is writing the script in python.
print(calc_insurance(36, 24, 0))

Phase-II:

1. Libraries/Functions used:

Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more

Pandas:

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like ActiveState's ActivePython.

Seaborn:

Seaborn is an amazing data visualization library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of the matplotlib library and also closely integrated to the data structures from pandas.

Matplotlib:

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

train_test_split:

train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets

linear_model:

linear_model is a class of the sklearn module if contain different functions for performing machine learning with linear models. The term linear model implies that the model is specified as a linear combination of features

2. Model Implementation(code) with Screenshots and Explanations

IMPORTING ALL Modules:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import linear_model

```
In [1]: #!/usr/bin/env python2
        # -*- coding: utf-8 -*-
        """
        Created on Fri Jun 18 21:17:01 2021

        @author: nagarathna
        """

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        from sklearn import linear_model
```

# **loading the data from csv file to a Pandas DataFrame**
insurance_dataset = pd.read_csv('insurance.csv')

# **first 5 rows of the dataframe is displaying**
insurance_dataset.head()

```
# loading the data from csv file to a Pandas DataFrame
insurance_dataset = pd.read_csv('insurance.csv')
```

```
# first 5 rows of the dataframe
insurance_dataset.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

retrieve the total number of rows and columns in dataset
insurance_dataset.shape

```
# number of rows and columns
insurance_dataset.shape
```

```
(1338, 7)
```

# **getting some informations about the dataset**
insurance_dataset.info()

```
# getting some informations about the dataset
insurance_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

# **checking for missing values**
insurance_dataset.isnull().sum()

```
# checking for missing values
insurance_dataset.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

 # statistical Measures of the dataset
insurance_dataset.describe()

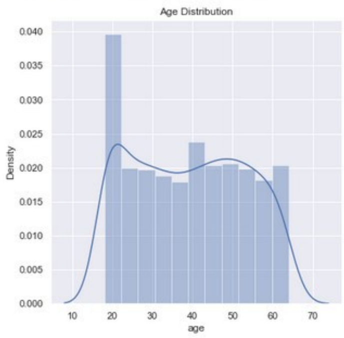
```
# statistical Measures of the dataset
insurance_dataset.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

 # distribution of age value
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['age'])
plt.title('Age Distribution')
plt.show()

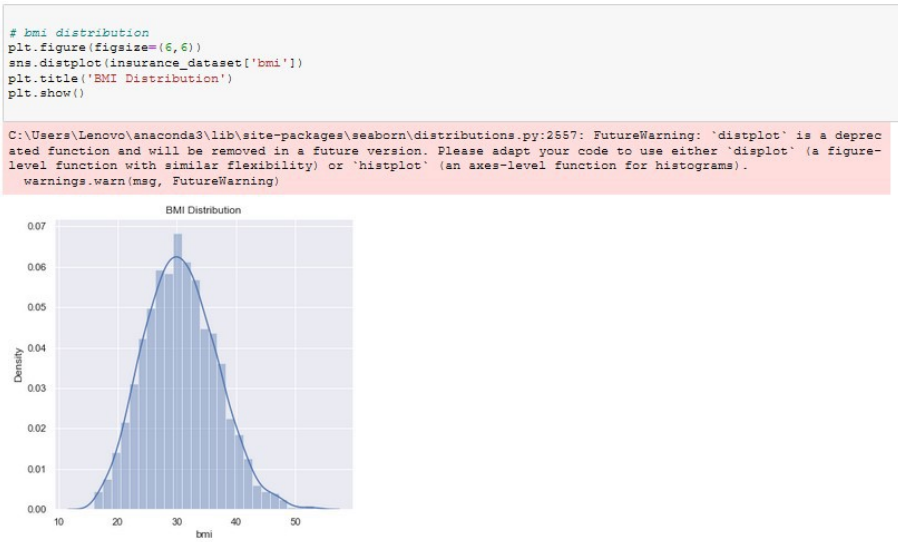
```
# distribution of age value
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['age'])
plt.title('Age Distribution')
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



# Gender column
plt.figure(figsize=(6,6))
sns.countplot(x='sex', data=insurance_dataset)

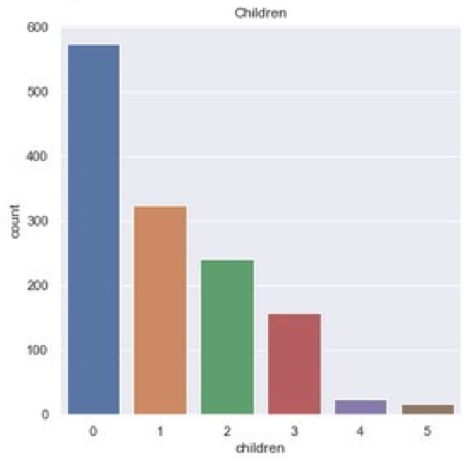
 # bmi distribution
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['bmi'])
plt.title('BMI Distribution')
plt.show()



 # children column
plt.figure(figsize=(6,6))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')

plt.show()
insurance_dataset['children'].value_counts()

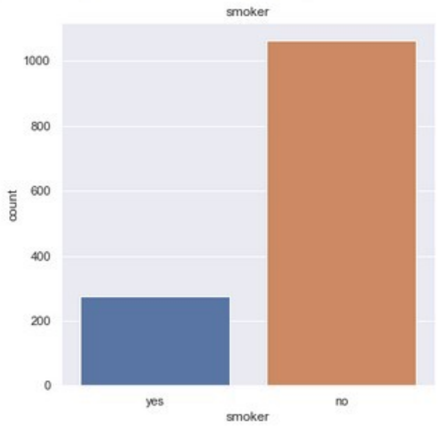
```
# children column
plt.figure(figsize=(6,6))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')
plt.show()
insurance_dataset['children'].value_counts()
```




```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

 # smoker column
plt.figure(figsize=(6,6))
sns.countplot(x='smoker', data=insurance_dataset)
plt.title('smoker')
plt.show()
insurance_dataset['smoker'].value_counts()

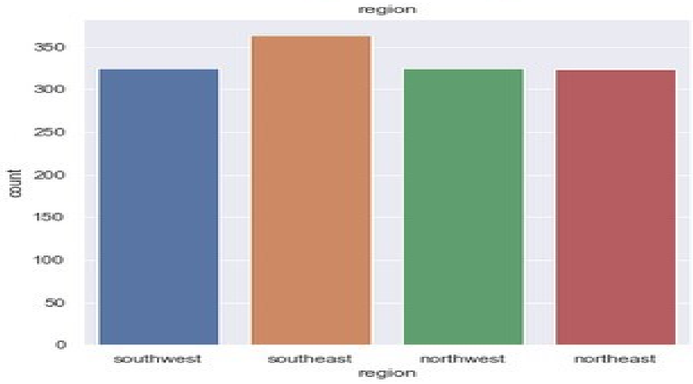
```
# smoker column
plt.figure(figsize=(6,6))
sns.countplot(x='smoker', data=insurance_dataset)
plt.title('smoker')
plt.show()
insurance_dataset['smoker'].value_counts()
```




```
no     1064
yes     274
Name: smoker, dtype: int64
```

 # region column
plt.figure(figsize=(6,6))
sns.countplot(x='region', data=insurance_dataset)
plt.title('region')
plt.show()
insurance_dataset['region'].value_counts()

```
# region column
plt.figure(figsize=(6,6))
sns.countplot(x='region', data=insurance_dataset)
plt.title('region')
plt.show()
insurance_dataset['region'].value_counts()
```



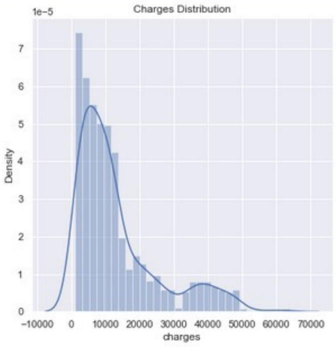

```
southeast    364
northwest    325
southwest    325
northeast    324
Name: region, dtype: int64
```

 # distribution of charges value
plt.figure(figsize=(6,6))

sns.distplot(insurance_dataset['charges'])
plt.title('Charges Distribution')
plt.show()

```
# distribution of charges value
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['charges'])
plt.title('Charges Distribution')
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



 # encoding sex column
#male-0 and female-1
insurance_dataset.replace({'sex':{'male':0,'female':1}}, inplace=True)

 # encoding 'smoker' column
#yes-0 and no-1
insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

 # encoding 'region' column
insurance_dataset.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplace=True)
X = insurance_dataset.drop(columns='charges', axis=1)
Y = insurance_dataset['charges']
print(X)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)

```
# encoding 'region' column
insurance_dataset.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplace=True)
X = insurance_dataset.drop(columns='charges', axis=1)
Y = insurance_dataset['charges']
print(X)
print(Y)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

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

[1338 rows x 6 columns]
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
```

#Loding the dataset to mydata

#mydata=insurance_dataset

mydata=insurance_dataset

print(mydata)

```
#mydata=insurance_dataset.head(5)
mydata=insurance_dataset
print(mydata)
```

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

[1338 rows x 7 columns]
```

#Appling the linear regression to database

reg=linear_model.LinearRegression()
#Start training

reg.fit(mydata[['age','sex','bmi','children','smoker','region']],mydata.charges)

```
reg=linear_model.LinearRegression()
#Start training
reg.fit(mydata[['age','sex','bmi','children','smoker','region']],mydata.charges)

LinearRegression()
```

reg.coef_

```
reg.coef_

array([   256.96796596,    128.95608532,    337.0245187 ,    468.35404686,
       -23867.05868713,    297.83720698])
```

reg.intercept_

```
reg.intercept_

108465.18599200742
```

reg.predict([[21,0,33.770,1,1,0]])

```
reg.predict([[18,0,33.770,1,1,0]])

array([1725.5523])
```

#Manual Calculation value
print((-378.3129839*18+879.31986567*0+ 2995.2920476*33.770+ 2100.33233118*1-879.31986567*1 1702.98703565*0)+108465.18599200742)

```
print((-378.3129839*18+879.31986567*0+-2995.2920476*33.770+2100.33233118*1-879.31986567*1-1702.98703565*0)+10846

1725.5522998654196
```

 # loading the Linear Regression model for calculating the accuracy
regressor = linear_model.LinearRegression()

regressor.fit(X_train, Y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

# prediction on training data
training_data_prediction =regressor.predict(X_train)

# R squared value
r2_train = metrics.r2_score(Y_train, training_data_prediction)
print('R squared vale : ', r2_train)

# prediction on test data
test_data_prediction =regressor.predict(X_test)

# R squared value
r2_test = metrics.r2_score(Y_test, test_data_prediction)
print('R squared vale : ', r2_test)
input_data = (60,1,25.84,0,1,3)
print("Mean squared error: %.2f" % np.mean((regressor.predict(X_test) - Y_test) ** 2))

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regressor.score(X_test, Y_test))

```
R squared vale :  0.751505643411174
R squared vale :  0.7447273869684077
```

```
print("Mean squared error: %.2f" % np.mean((regressor.predict(X_test) - Y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regressor.score(X_test, Y_test))
```

```
Mean squared error: 38337035.49
Variance score: 0.74
```

#ACCURACE CALCULATED USING Support Vector Machine (Regression)

X_c = mydata.drop('charges',axis=1).values
y_c = mydata['charges'].values.reshape(-1,1)
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_c,y_c,test_size=0.2, random_state=42)

X_train_scaled = StandardScaler().fit_transform(X_train_c)
y_train_scaled = StandardScaler().fit_transform(y_train_c)
X_test_scaled = StandardScaler().fit_transform(X_test_c)

```
y_test_scaled = StandardScaler().fit_transform(y_test_c)
svr = SVR()


#svr.fit(X_train_scaled, y_train_scaled.ravel())
parameters =  { 'kernel' : ['rbf', 'sigmoid'],'gamma' : [0.001, 0.01, 0.1, 1, 'scale'],
                 'tol' : [0.0001],'C': [0.001, 0.01, 0.1, 1, 10, 100] }
svr_grid = GridSearchCV(estimator=svr, param_grid=parameters, cv=10,
            verbose=4, n_jobs=-1)
svr_grid.fit(X_train_scaled, y_train_scaled.ravel())
svr = SVR(C=10, gamma=0.1, tol=0.0001)
svr.fit(X_train_scaled, y_train_scaled.ravel())

print(svr_grid.best_estimator_)
print(svr_grid.best_score_)
cv_svr = svr_grid.best_score_

y_pred_svr_train = svr.predict(X_train_scaled)
r2_score_svr_train = r2_score(y_train_scaled, y_pred_svr_train)

y_pred_svr_test = svr.predict(X_test_scaled)
r2_score_svr_test = r2_score(y_test_scaled, y_pred_svr_test)

print('CV : {0:.3f}'.format(cv_svr.mean()))
print('R2_score (train) : {0:.3f}'.format(r2_score_svr_train))
print('R2 score (test) : {0:.3f}'.format(r2_score_svr_test))
print('RMSE : {0:.3f}'.format(rmse_svr))
```

```
CV : 0.832
R2_score (train) : 0.858
R2 score (test) : 0.872
RMSE : 0.358
```

#Displaying the accuracy of two model

```
models = [('Linear Regression',  r2_train, r2_test, regressor.score(X_test,
Y_test)),
 ('Support Vector Regression', r2_score_svr_train, r2_score_svr_test,
cv_svr.mean()),]
predict = pd.DataFrame(data = models, columns=['Model', 'R2_Score(training)',
'R2_Score(test)', 'Cross-Validation'])
predict
```

```
models = [('Linear Regression',  r2_train, r2_test, regressor.score(X_test, Y_test)),

          ('Support Vector Regression', r2_score_svr_train, r2_score_svr_test, cv_svr.mean()),

         ]
predict = pd.DataFrame(data = models, columns=['Model', 'R2_Score(training)', 'R2_Score(test)', 'Cross-Validatio
predict
```

	Model	R2_Score(training)	R2_Score(test)	Cross-Validation
0	Linear Regression	0.751506	0.744727	0.744727
1	Support Vector Regression	0.858192	0.871719	0.832362

3. CONCLUSION:

Various factors were used and their effect on predicted amount was examined. It was observed that a persons age and smoking status affects the prediction most in every algorithm applied. Attributes which had no effect on the prediction were removed from the features.

Using linear regression we can get the nearly 75% accuracy and using support vecotor regression we can get 85% accuracy.

4. References

- https://www.kaggle.com/mirichoi0218/insurance
- Machine Learning, SaikatDutt, Subramanian Chandramouli, Amit Kumar Das, Pearson, 4 thimpression, 2019, Pearson Publications, ISBN 978-93-530-6669-7
- https://www.linkedin.com/pulse/health-insurance-cost-prediction-using-machine-learning-santani