In [2]:
```
import pandas as pd
```

In [3]:
```
data=pd.read_csv(r"C:\Users\bikas\Downloads\insurance.csv")
```

In [4]:
```
data
```

Out[4]:

	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
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

Displaing Top 5 Rows of The Dataset

In [5]:
```
data.head()
```

Out[5]:

	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

Displaing Last 5 Rows of The Dataset¶

In [6]:
```
data.tail()
```

Out[6]:

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

Find Shape of Our Dataset (Number of Rows And Number of Columns)

In [7]:
```
data.shape
```

Out[7]:
```
(1338, 7)
```

In [8]:
```
print("Number of Rows",data.shape[0])
print("Number of Columns",data.shape[1])
```

```
Number of Rows 1338
Number of Columns 7
```

Getting Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

In [9]:
```
data.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 Null Values In The Dataset

In [10]:

```
data.isnull().sum()
```

Out[10]:

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

Get Overall Statistics About The Dataset

In [11]:

```
data.describe(include='all')
```

Out[11]:

	age	sex	bmi	children	smoker	region	charges
count	1338.000000	1338	1338.000000	1338.000000	1338	1338	1338.000000
unique	NaN	2	NaN	NaN	2	4	NaN
top	NaN	male	NaN	NaN	no	southeast	NaN
freq	NaN	676	NaN	NaN	1064	364	NaN
mean	39.207025	NaN	30.663397	1.094918	NaN	NaN	13270.422265
std	14.049960	NaN	6.098187	1.205493	NaN	NaN	12110.011237
min	18.000000	NaN	15.960000	0.000000	NaN	NaN	1121.873900
25%	27.000000	NaN	26.296250	0.000000	NaN	NaN	4740.287150
50%	39.000000	NaN	30.400000	1.000000	NaN	NaN	9382.033000
75%	51.000000	NaN	34.693750	2.000000	NaN	NaN	16639.912515
max	64.000000	NaN	53.130000	5.000000	NaN	NaN	63770.428010

Covert Columns From String ['sex' ,'smoker','region'] To Numerical Values

In [12]:

```
data['sex'].unique()
data['sex']=data['sex'].map({'female':0,'male':1})
data['smoker']=data['smoker'].map({'yes':1,'no':0})
data['region']=data['region'].map({'southwest':1,'southeast':2,
                   'northwest':3,'northeast':4})
```

In [13]:
```
data.head()
```

Out[13]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	1	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	3	21984.47061
4	32	1	28.880	0	0	3	3866.85520

Store Feature Matrix In X and Response(Target) In Vector y

In [14]:
```
X = data.drop(['charges'],axis=1)
y = data['charges']
```

Train/Test split

Split data into two part : a training set and a testing set

Train the model(s) on training set

Test the Model(s) on Testing set

In [15]:
```
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=
```

Import the models¶

In [16]:
```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

Model Training

In [17]:

```
lr = LinearRegression()
lr.fit(X_train,y_train)
svm = SVR()
svm.fit(X_train,y_train)
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
gr = GradientBoostingRegressor()
gr.fit(X_train,y_train)
```

Out[17]:

```
GradientBoostingRegressor()
```

Prediction on Test Data

In [18]:

```
y_pred1 = lr.predict(X_test)
y_pred2 = svm.predict(X_test)
y_pred3 = rf.predict(X_test)
y_pred4 = gr.predict(X_test)

df1 = pd.DataFrame({'Actual':y_test,'Lr':y_pred1,
                  'svm':y_pred2,'rf':y_pred3,'gr':y_pred4})
```

In [19]:

```
df1
```

Out[19]:

	Actual	Lr	svm	rf	gr
764	9095.06825	8924.407244	9548.261584	10283.193858	11001.128629
887	5272.17580	7116.295018	9492.515425	5166.881968	5840.174656
890	29330.98315	36909.013521	9648.758701	28203.891089	28001.980112
1293	9301.89355	9507.874691	9555.044136	10167.529963	9745.291602
259	33750.29180	27013.350008	9420.421978	34602.016263	33639.100981
...	...	...	...	...	...
109	47055.53210	39116.968669	9648.902852	47282.000434	45431.423211
575	12222.89830	11814.555568	9625.431547	12386.130390	12465.025294
535	6067.12675	7638.107736	9504.168517	6408.961349	6974.336525
543	63770.42801	40959.081722	9605.004594	46569.913222	47862.047791
846	9872.70100	12258.228529	9590.987268	9833.422604	10289.655388

268 rows × 5 columns

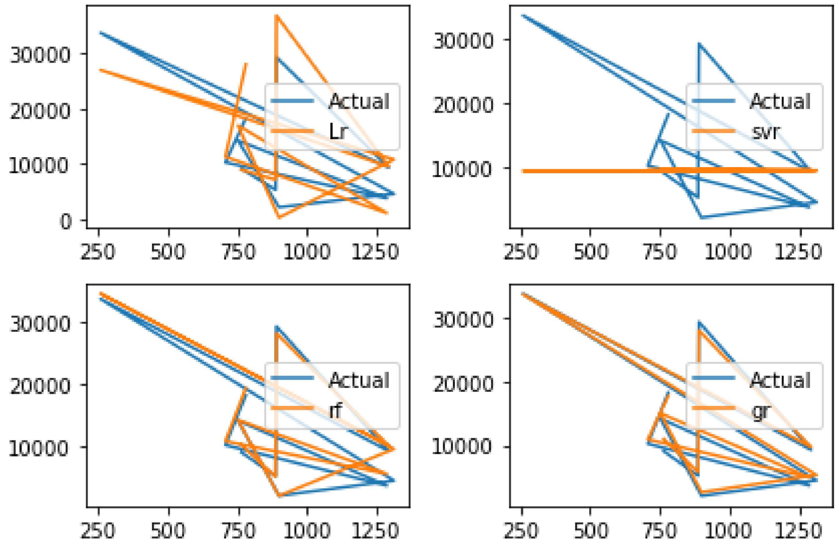
Compare Performance Visually

In [20]:

```
import matplotlib.pyplot as plt
```

In [21]:
```
plt.subplot(221)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['Lr'].iloc[0:11],label="Lr")
plt.legend()

plt.subplot(222)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['svm'].iloc[0:11],label="svr")
plt.legend()

plt.subplot(223)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['rf'].iloc[0:11],label="rf")
plt.legend()

plt.subplot(224)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['gr'].iloc[0:11],label="gr")

plt.tight_layout()

plt.legend()
```

Out[21]:
```
<matplotlib.legend.Legend at 0x2071999e850>
```



Evaluating the Algorithm

In [22]:
```
from sklearn import metrics
```

In [23]:
```
score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
score3 = metrics.r2_score(y_test,y_pred3)
score4 = metrics.r2_score(y_test,y_pred4)
```

In [24]:

```
print(score1,score2,score3,score4)
```

```
0.7833463107364539 -0.07229762787861826 0.8623111782646651 0.8779726251291786
```

In [25]:

```
s1 = metrics.mean_absolute_error(y_test,y_pred1)
s2 = metrics.mean_absolute_error(y_test,y_pred2)
s3 = metrics.mean_absolute_error(y_test,y_pred3)
s4 = metrics.mean_absolute_error(y_test,y_pred4)
```

In [26]:

```
print(s1,s2,s3,s4)
```

```
4186.508898366432 8592.428727899724 2545.121415903821 2447.9515580545844
```

Predict Charges For New Customer

In [27]:

```
data = {'age' : 40,
        'sex' : 1,
        'bmi' : 40.30,
        'children' : 4,
        'smoker' : 1,
        'region' : 2}
```

In [28]:

```
df = pd.DataFrame(data,index=[0])
df
```

Out[28]:

	age	sex	bmi	children	smoker	region
0	40	1	40.3	4	1	2

In [29]:

```
new_pred = gr.predict(df)
print("Medical Insurance cost for New Customer is : ",new_pred[0])
```

```
Medical Insurance cost for New Customer is :  43839.34586293143
```

From above we found that gredianBoostingRegressor is the best model for this dataset. Before production, it is good practice to train our model on the entire dataset

Save model using joblib

In [30]:

```
gr = GradientBoostingRegressor()
gr.fit(X,y)
```

Out[30]:

```
GradientBoostingRegressor()
```

```
In [31]: import joblib
```

```
In [32]: joblib.dump(gr,'model_joblib_gr')
```

```
Out[32]: ['model_joblib_gr']
```

```
In [33]: model1=joblib.load('model_joblib_gr')
```

```
In [34]: model1.predict(df)
```

```
Out[34]: array([42148.361888])
```

GUI

```
In [35]: from tkinter import *
```

```
In [36]: import joblib
```

In [38]:

```
def show_entry():

    p1=float(e1.get())
    p2=float(e2.get())
    p3=float(e3.get())
    p4=float(e4.get())
    p5=float(e5.get())
    p6=float(e6.get())

    model=joblib.load('model_joblib_gr')
    result=model.predict([[p1,p2,p3,p4,p5,p6]])


    Label(master,text="Insurance Cost").grid(row=7)
    Label(master,text=result).grid(row=8)

master=Tk()
master.title("Insurance Cost Predection")
label=Label(master,text="Insurance Cost Predection",bg="black",
           fg="white").grid(row=0,columnspan=2)
Label(master,text="Enter Your Age").grid(row=1)
Label(master,text="Male or Female[1/0]").grid(row=2)
Label(master,text="Enter Your BMI Value").grid(row=3)
Label(master,text="Enter Number Of Children").grid(row=4)
Label(master,text="Smoker Yes/No [1/0]").grid(row=5)
Label(master,text="Region [1-4]").grid(row=6)

e1=Entry(master)
e2=Entry(master)
e3=Entry(master)
e4=Entry(master)
e5=Entry(master)
e6=Entry(master)

e1.grid(row=1,column=1)
e2.grid(row=2,column=1)
e3.grid(row=3,column=1)
e4.grid(row=4,column=1)
e5.grid(row=5,column=1)
e6.grid(row=6,column=1)

Button(master,text="Predict",command=show_entry).grid()


mainloop()
```

```
C:\Users\bikas\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but GradientBoostingRegressor was fitted
with feature names
  warnings.warn(
```

In [ ]:

```
```