```python
In [1]: import pandas as pd
```

```python
In [2]: df=pd.read_csv('insurance.csv')# ### Prediction on Test Data
```

```python
In [3]: df.head()
```

Out[3]:

	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

```python
In [4]: df.tail()
```

Out[4]:

	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

Finding No. of rows & Columns

```python
In [5]: df.shape
```

```
Out[5]: (1338, 7)
```

Checking null value in Dataset

```python
In [6]: df.isnull()
```

Out[6]:

	age	sex	bmi	children	smoker	region	charges
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1333	False	False	False	False	False	False	False
1334	False	False	False	False	False	False	False
1335	False	False	False	False	False	False	False
1336	False	False	False	False	False	False	False
1337	False	False	False	False	False	False	False

1338 rows × 7 columns

```python
In [7]: df.isnull().sum() # sum of null values in each column
```

```
Out[7]:
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

Getting Overall Statistics of Dataset

```python
In [8]: df.describe()
```

Out[8]:

	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

```python
In [9]: df.describe(include='all') # statistics for caterogical and numerical column both
```

Out[9]:

	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

```python
In [10]: df.head()
```

Out[10]:

	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

Converting Columns from string to Integer

```python
In [11]: df['sex'].unique()
```

```
Out[11]: array(['female', 'male'], dtype=object)
```

```python
In [12]: df['sex']= df['sex'].map({'female':0,'male':1})
```

```python
In [13]: df['sex']
```

```
Out[13]:
0       0
1       1
2       1
3       1
4       1
       ..
1333    1
1334    0
1335    0
1336    0
1337    0
Name: sex, Length: 1338, dtype: int64
```

```python
In [14]: df.head()
```

Out[14]:

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

```python
In [15]: df['smoker'].unique()
```

```
Out[15]: array(['yes', 'no'], dtype=object)
```

```python
In [16]: df['smoker']= df['smoker'].map({'yes':1,'no':0})
```

```python
In [17]: df['smoker']
```

```
Out[17]:
0       1
1       0
2       0
3       0
4       0
       ..
1333    0
1334    0
1335    0
1336    0
1337    1
Name: smoker, Length: 1338, dtype: int64
```

```python
In [18]: df['region'].unique()
```

```
Out[18]: array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

```python
In [19]: df['region']=df['region'].map({'southwest':1,'southeast':2,'northwest':3,'northeast':4})
```

```python
In [20]: df['region']
```

```
Out[20]:
0       1
1       2
2       2
3       3
4       3
       ..
1333    3
1334    4
1335    2
1336    1
1337    3
Name: region, Length: 1338, dtype: int64
```

```python
In [21]: df.head()
```

Out[21]:

	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

```python
In [22]: df.columns
```

```
Out[22]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

Storing Features Matrix in X and response in Y

```python
In [23]: X = df.drop(['charges'],axis=1)
```

```python
In [24]: X
```

Out[24]:

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

1338 rows × 6 columns

```python
In [25]: Y= df['charges']
```

```python
In [26]: Y
```

```
Out[26]:
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

```python
In [27]: from sklearn.model_selection import train_test_split
```

```python
In [28]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2, random_state=42)
```

```python
In [29]: from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

Model Trainning

```python
In [30]: lr=LinearRegression()
lr.fit(X_train,Y_train)
sv=SVR()
sv.fit(X_train,Y_train)
rf= RandomForestRegressor()
rf.fit(X_train,Y_train)
gb=GradientBoostingRegressor()
gb.fit(X_train,Y_train)
```

```
Out[30]: GradientBoostingRegressor()
```

Prediction on Test Data

```python
In [31]: Y_pred1 = lr.predict(X_test)
Y_pred2 = sv.predict(X_test)
Y_pred3 = rf.predict(X_test)
Y_pred4 = gb.predict(X_test)
```

```python
In [32]: df1 = pd.DataFrame({'Actual':Y_test,'lr':Y_pred1,'sv':Y_pred2,'rf':Y_pred3,'gb':Y_pred4 } )
```

```python
In [33]: df1
```

Out[33]:

	Actual	lr	sv	rf	gb
764	9095.06825	8924.407244	9548.261584	11047.132218	11001.128629
887	5272.17580	7116.295018	9492.515425	5286.508781	5840.174656
890	29330.98315	36909.013521	9648.758701	28392.378847	28001.980112
1293	9301.89355	9507.874691	9555.044136	10721.370519	9745.291602
259	33750.29180	27013.350008	9420.421978	34562.763171	33639.100981
...	...	...	...	...	...
109	47055.53210	39116.968669	9648.902852	46681.166100	45431.423211
575	12222.89830	11814.555568	9625.431547	12745.579781	12465.025294
535	6067.12675	7638.107736	9504.168517	6362.580526	6974.336525
543	63770.42801	40959.081722	9605.004594	46890.300122	47862.047791
846	9872.70100	12258.228529	9590.987268	9778.817450	10289.655388

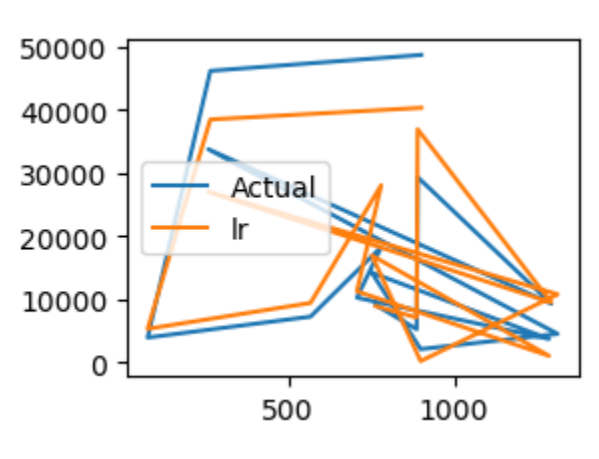
268 rows × 5 columns

Comparing performance Visually

```python
In [34]: import matplotlib.pyplot as plt
```

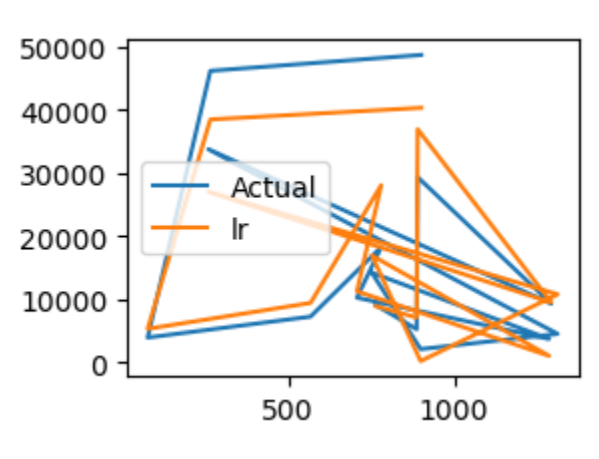
```python
In [35]: plt.subplot(221)
plt.plot(df1['Actual'].iloc[0:15],label='Actual')
plt.plot(df1['lr'].iloc[0:15],label='lr')
plt.legend()
```

```
Out[35]: <matplotlib.legend.Legend at 0x21a85fdcd90>
```



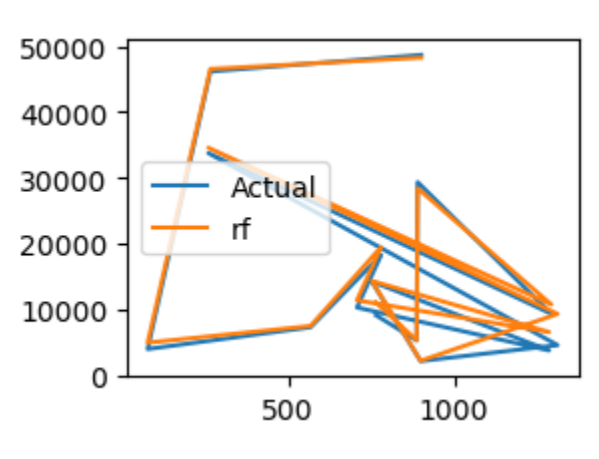

```python
In [36]: plt.subplot(221)
plt.plot(df1['Actual'].iloc[0:15],label='Actual')
plt.plot(df1['lr'].iloc[0:15],label='lr')
plt.legend()
```

```
Out[36]: <matplotlib.legend.Legend at 0x21a85f4caf0>
```



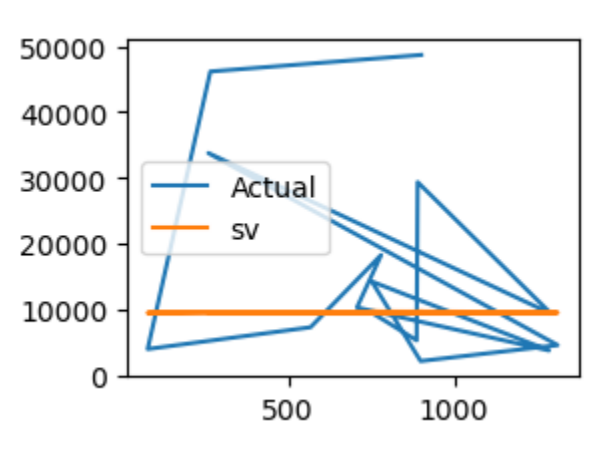

```python
In [37]: plt.subplot(222)
plt.plot(df1['Actual'].iloc[0:15],label='Actual')
plt.plot(df1['rf'].iloc[0:15],label='rf')
plt.legend()
```

```
Out[37]: <matplotlib.legend.Legend at 0x21a8618af40>
```



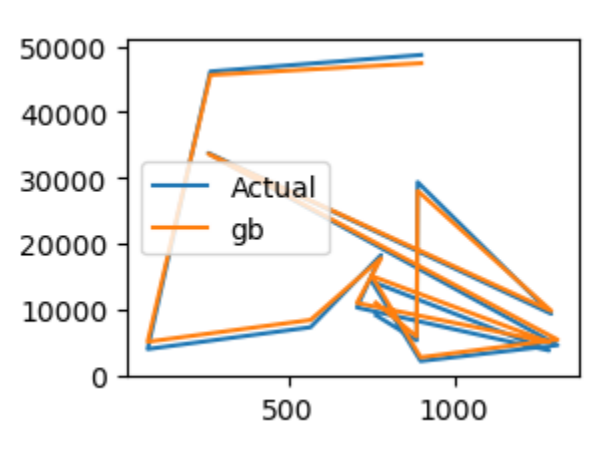

```python
In [38]: plt.subplot(223)
plt.plot(df1['Actual'].iloc[0:15],label='Actual')
plt.plot(df1['sv'].iloc[0:15],label='sv')
plt.legend()
```

```
Out[38]: <matplotlib.legend.Legend at 0x21a860496d0>
```




```python
In [39]: plt.subplot(224)
plt.plot(df1['Actual'].iloc[0:15],label='Actual')
plt.plot(df1['gb'].iloc[0:15],label='gb')
plt.legend()
```

```
Out[39]: <matplotlib.legend.Legend at 0x21a86055e80>
```



Evaluating the Models

```python
In [40]: from sklearn import metrics
```

```python
In [41]: 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)
```

```python
In [42]: print(score1,score2,score3,score4)
```

```
0.7833463107364539 -0.07229762787861826 0.8651213611604314 0.8779726251291786
```

```python
In [43]: 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)
```

```python
In [44]: print(s1,s2,s3,s4)
```

```
4186.508898366435 8592.428727899724 2465.6329064785773 2447.9515580545844
```

Charges prediction for new customer

```python
In [45]: data ={'age':40,
       'sex':1,
       'bmi':50.5,
       'children':3,
       'smoker':1,
       'region':2
        }
df = pd.DataFrame(data, index=[0])
df
```

Out[45]:

	age	sex	bmi	children	smoker	region
0	40	1	50.5	3	1	2

```python
In [46]: new_pred =gb.predict(df)
```

```python
In [47]: print(new_pred)
```

```
[43416.49569132]
```

```python
In [48]: gb= GradientBoostingRegressor()
gb.fit(X,Y)
```

```
Out[48]: GradientBoostingRegressor()
```

Saving the Model

```python
In [49]: import joblib
joblib.dump(gb,'model_joblib_gb')
```

```
Out[49]: ['model_joblib_gb']
```

```python
In [50]: model=joblib.load('model_joblib_gb')
model.predict(df)
```

```
Out[50]: array([46731.44631884])
```

GUI

```python
In [51]: from tkinter import*
import joblib
```

```python
In [53]: 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_gb')
    result=model.predict([[p1,p2,p3,p4,p5,p6]])
    Label(master, text = "Insurance Cost").grid(row=8)
    Label(master,text=result).grid(row=8,column=1)

master =Tk()
master.title("Insurance Cost Prediction")
label = Label(master,text ="Insurance Cost Prediction",bg = "blue",fg ="white").grid(row=0,columnspan=2)
Label(master,text = "Enter Your Age").grid(row=2)
Label(master,text = "Male or Female [ 1/0]").grid(row=3)
Label(master,text = "Enter Your BMI Value").grid(row=4)
Label(master,text = "Enter Number of children").grid(row=5)
Label(master,text = "Smoker Yes/No [1/0]").grid(row=6)
Label(master,text = "Region [1-4]").grid(row=7)

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

e1.grid(row=2,column=1)
e2.grid(row=3,column=1)
e3.grid(row=4,column=1)
e4.grid(row=5,column=1)
e5.grid(row=6,column=1)
e6.grid(row=7,column=1)

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

mainloop()
```

```python
In [ ]:
```