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

```
m = pd.read_csv('med_insurance.csv')
```

```
m.head(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
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

```
m.isnull()
```

	age	sex	bmi	children	smoker	region	charges
0	False	False	False	False	False	False	False

```
X.head()
```

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

```
y.head()
```

```
0    16884.92400
1     1725.55230
2     4449.46200
3    21984.47061
4     3866.85520
Name: charges, dtype: float64
```

```
m.describe
```

```
<bound method NDFrame.describe of       age     sex     bmi  children smoker     reg
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 x 7 columns]>
```

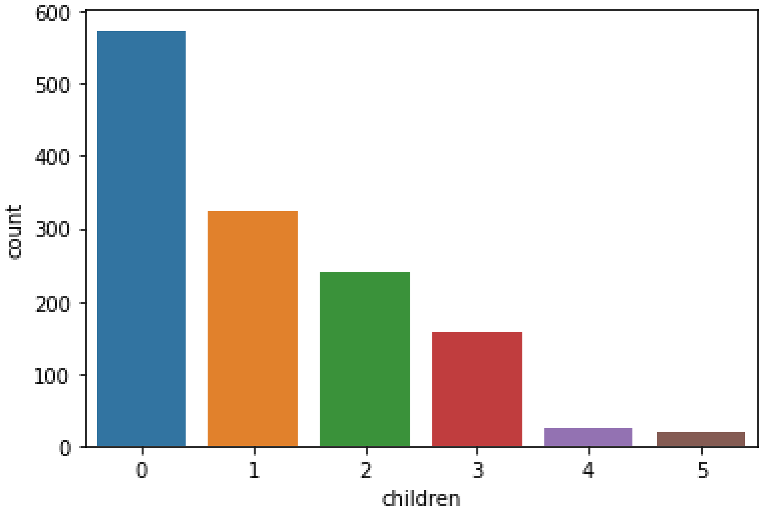
```
sns.heatmap(m.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1b6591f50>
```

0
64
128
192
256
320
384
448
512
576
640
704
0.100
0.075
0.050
0.025
0.000

```
sns.countplot(x='children',data=m)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff1ae83ead0>
```



```
g=pd.get_dummies(m['sex'])
```

```
g
```

female male

```
r=pd.get_dummies(m['region'])
r
```

	northeast	northwest	southeast	southwest
0	0	0	0	1
1	0	0	1	0
2	0	0	1	0
3	0	1	0	0
4	0	1	0	0
...	...	...	...	...
1333	0	1	0	0
1334	1	0	0	0
1335	0	0	1	0
1336	0	0	0	1
1337	0	1	0	0

1338 rows × 4 columns

```
s=pd.get_dummies(m['smoker'])
s
```

	no	yes
0	0	1
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
1333	1	0
1334	1	0
1335	1	0
1336	1	0
1337	0	1

1338 rows × 2 columns

```
m=m.drop(['sex','region','smoker'],axis=1)
```

```
m
```

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

1338 rows × 4 columns

```
med = pd.concat([g,r,s],axis=1)
med
```

	female	male	northeast	northwest	southeast	southwest	no	yes
0	1	0	0	0	0	1	0	1
1	0	1	0	0	1	0	1	0
2	0	1	0	0	1	0	1	0
3	0	1	0	1	0	0	1	0
4	0	1	0	1	0	0	1	0
...	...	...	...	...	...	...	...	...
1333	0	1	0	1	0	0	1	0
1334	1	0	1	0	0	0	1	0
1335	1	0	0	0	1	0	1	0
1336	1	0	0	0	0	1	1	0
1337	1	0	0	1	0	0	0	1

1338 rows × 8 columns

```
m
```

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

1338 rows × 4 columns

```
med =pd.concat([m,g,r,s],axis=1)
```

```
med
```

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

1338 rows × 12 columns

```
med.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
```

```
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   age        1338 non-null   int64  
 1   bmi        1338 non-null   float64
 2   children   1338 non-null   int64  
 3   charges    1338 non-null   float64
 4   female     1338 non-null   uint8  
 5   male       1338 non-null   uint8  
 6   northeast  1338 non-null   uint8  
 7   northwest  1338 non-null   uint8  
 8   southeast  1338 non-null   uint8  
 9   southwest  1338 non-null   uint8  
 10  no         1338 non-null   uint8  
 11  yes        1338 non-null   uint8  
dtypes: float64(2), int64(2), uint8(8)
memory usage: 52.4 KB
```

```
y = med.charges
```

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

```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=0)
```

```
regr = LinearRegression()
regr.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred = regr.predict(X_test)
```

```
y_pred
```

```
       5.55574972e+03,  5.14017616e+03,  6.54358067e+03,  5.29101323e+03,
       9.86573296e+03,  5.30318056e+03,  5.82126396e+03,  6.71264320e+03,
       3.79359032e+03,  5.35674887e+03,  3.81050149e+04,  1.49449863e+03,
       1.25241127e+04,  8.81600344e+03,  1.35779650e+04,  5.33467494e+03,
       5.30034294e+03,  3.64048098e+04,  4.39998407e+03,  2.07683907e+03,
       1.50700993e+04,  1.28583285e+04,  3.50480278e+04,  4.81692549e+03,
       5.75568624e+03,  3.10570003e+04,  5.91365180e+03,  2.12466743e+03,
       8.55473411e+03,  1.02157331e+04,  8.01035514e+03,  5.56732969e+03,
       1.33173353e+04,  3.83325588e+04,  1.36438797e+04,  2.88073730e+04,
       6.57855063e+03,  3.54156777e+04,  3.85264559e+03,  1.18931235e+04,
       9.11693680e+03,  6.16245674e+03,  1.11692747e+04,  1.47049487e+04,
       5.02634820e+03,  4.46458369e+03,  7.96288703e+03,  1.31978165e+03,
       8.00636697e+03,  4.52316428e+03,  1.30510837e+04,  4.39675270e+03,
       1.02462007e+04,  7.38951728e+03,  9.36630903e+03,  2.44501178e+03,
       1.33392136e+04,  1.66792757e+04,  1.50554402e+04,  1.06897979e+04,
       5.36806605e+03,  2.16568099e+03,  1.84625678e+03,  1.36078096e+04,
       1.41046244e+04,  5.18518795e+03,  3.78006430e+03,  9.57585167e+03,
       1.01219994e+04,  2.81258257e+04,  7.73311859e+03,  1.06662041e+04,
       6.36597477e+03,  2.94855951e+04,  1.12667069e+04,  7.60247088e+03,
       1.00997356e+04,  1.20691312e+04,  3.08178430e+03,  1.06992857e+04,
       1.58012525e+03,  7.13244227e+03,  2.82894803e+04,  3.87009694e+04,
       6.44760486e+03,  8.18263663e+03,  2.56156737e+03,  4.43116587e+02,
       1.06319373e+04,  4.14413001e+03,  4.80964514e+03,  2.35032467e+03,
       6.90487999e+03,  3.33519011e+04,  3.81889733e+04,  1.46742863e+04,
       8.06796404e+03,  1.62426303e+04,  3.28132879e+04,  9.64677964e+03,
       3.35687219e+04,  3.28499010e+03,  3.07421002e+04,  8.27403986e+03,
       1.43749371e+04,  3.94363238e+03,  3.20677633e+04,  8.50722697e+03,
       1.12272940e+04,  9.17836109e+03,  4.02094225e+03,  1.28491704e+04,
       1.15880228e+04,  8.28529095e+03,  1.34436097e+04,  2.88016799e+03,
       1.07598245e+04,  5.60040795e+03,  1.10443396e+04,  3.15748920e+04,
       9.86714924e+03,  1.27147820e+03,  2.86568919e+02,  3.99183572e+04,
       9.82112709e+03,  6.98485911e+03,  1.38987602e+04,  1.36056308e+04,
       2.72298367e+04,  7.28448164e+03,  6.99440177e+03,  1.22798274e+04,
       2.68645584e+03,  3.63656263e+03,  2.49800939e+04,  2.58300744e+04,
       1.31031820e+04,  3.32140682e+03,  4.86248281e+03,  9.49806860e+03,
       1.26028085e+04,  2.34806136e+04,  3.05185854e+04,  1.02915974e+04,
       2.36864639e+04,  2.70214254e+03,  1.13613494e+04,  7.31417605e+03,
       8.37333217e+03, -1.20090918e+01,  7.87352474e+03,  3.53845654e+04,
       6.34875226e+03,  6.42543828e+03,  2.58973345e+01,  1.06566943e+04,
       6.52223275e+03,  9.67955134e+03,  3.90212856e+04,  2.73398516e+04,
       1.16690107e+04,  3.52946768e+04,  1.52284127e+04,  6.70599088e+03,

       1.07444457e+04,  7.07603187e+03,  3.64305215e+04,  5.77215659e+03,
       1.11987869e+04,  9.12757093e+02,  2.39243545e+04,  1.72536528e+03,
       3.44370930e+04,  1.11893513e+04,  1.64284856e+03,  3.23084344e+04,
       6.83933850e+03,  5.38358717e+03,  3.77008671e+04,  2.38518793e+03,
       9.73769497e+03,  2.51677325e+03,  1.29924006e+04,  1.14558043e+03,
       1.09157187e+04,  6.82118928e+03,  3.65819756e+04,  7.32055769e+03,
       3.03617950e+04,  2.92892602e+04,  6.82053534e+03,  1.09348851e+04,
       1.76251405e+03,  2.37139310e+03,  3.67942218e+03,  1.26348376e+04,
       3.68242957e+04,  9.88276212e+03,  5.21130967e+02,  1.15304510e+04,
       4.96731683e+03,  1.00055528e+04,  5.74380019e+03,  7.07572800e+03,
       4.03843476e+03,  2.82570712e+04,  4.44634108e+03, -1.24189730e+03,
       3.29513389e+04,  1.26555619e+04,  3.59905000e+04,  1.00050499e+04,
       7.57606854e+03, -2.70310110e+02,  2.44881699e+03,  1.17300960e+04,
       5.85414692e+03,  3.48550916e+03,  1.22947034e+04,  7.91825176e+03,
       7.10941145e+03,  5.62219064e+03,  2.93138277e+03,  3.20691197e+04,
       3.42234717e+03,  8.77246198e+03,  4.58681719e+03,  1.32668158e+04,
       1.49586949e+04,  7.41278952e+03,  2.66486545e+04,  1.40642726e+04,
       1.71400262e+04,  1.16484659e+04])
```

```
y_test
```

```
578      9724.53000
610      8547.69130
569     45702.02235
1034    12950.07120
198      9644.25250
           ...
1261     3277.16100
494     17942.10600
97      10226.28420
418     14418.28040
920     13451.12200
Name: charges, Length: 402, dtype: float64
```

```
weights=regr.coef_
intercept=regr.intercept_
print(weights,intercept)
```

```
[   256.43544682    335.36907276    472.70978916     23.77337759
    -23.77337759    589.02469054     27.12354518   -405.723989
```

```
 -210.42424672 -11717.99558474  11717.99558474] -730.1121662702444
```

```
regr.score(X_test,y_test)
```

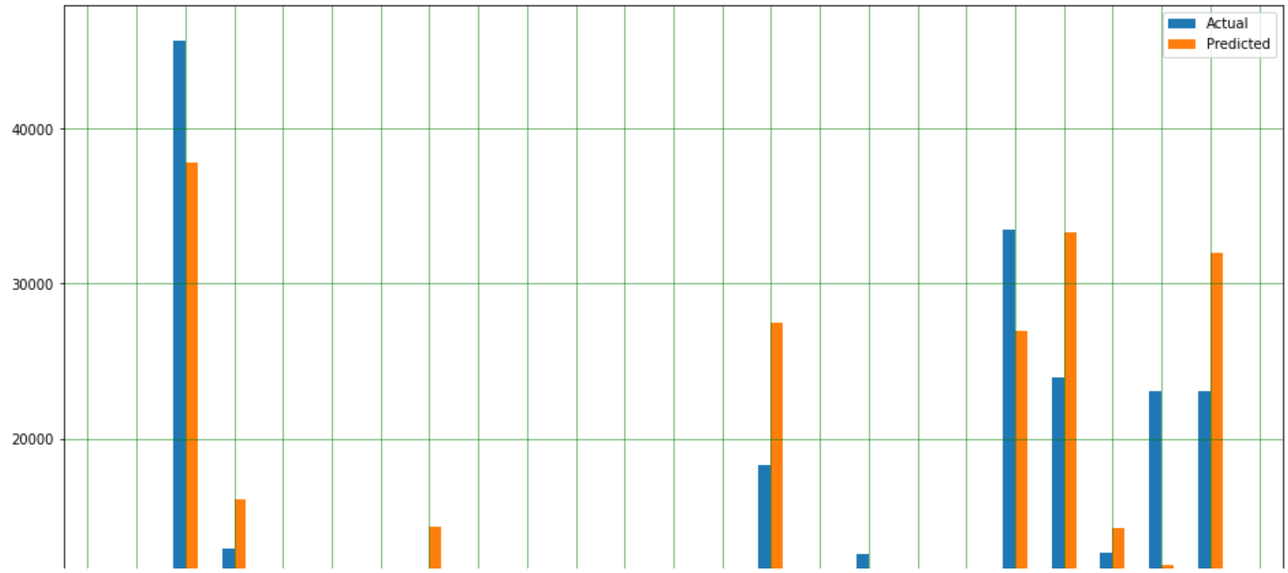
```
0.7909160991789905
```

```
df=pd.DataFrame({'Actual' : y_test,'Predicted' : y_pred})
df
```

```
           Actual     Predicted
578    9724.53000  11253.193646
610    8547.69130   9544.907094
569   45702.02235  37849.801048
1034  12950.07120  16069.269685
198    9644.25250   6734.408723
...           ...           ...
1261   3277.16100   7412.789524
494   17942.10600  26648.654504
97    10226.28420  14064.272563
418   14418.28040  17140.026157
920   13451.12200  11648.465902

402 rows × 2 columns
```

```
df1=df.head(25)
df1.plot(kind="bar",figsize=(16,10))
plt.grid(which="major",linestyle='-',linewidth="0.5",color="green")
plt.grid(which="minor",linestyle=':',linewidth="0.5",color="black")
plt.show()
```



```
print('Mean Absolute Error: ',mean_absolute_error(y_test,y_pred))
print('Mean Square Error: ',mean_squared_error(y_test,y_pred))
print('Root Mean Absolute Error: ',np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
    Mean Absolute Error:  4011.4496793279864
    Mean Square Error:  33342497.826954577
    Root Mean Absolute Error:  5774.296305780867
```

✓ 0s completed at 10:13 AM ● ×