In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statistics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```python
df = pd.read_csv('Insurance.csv')
df.head()
```

Out[2]:

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

In [3]:
```python
df.head(30)
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
0	19	0	27.900	0	1	3	16884.92400	1
1	18	1	33.770	1	0	2	1725.55230	1
2	28	1	33.000	3	0	2	4449.46200	0
3	33	1	22.705	0	0	1	21984.47061	0
4	32	1	28.880	0	0	1	3866.85520	1
5	31	0	25.740	0	0	2	3756.62160	0
6	46	0	33.440	1	0	2	8240.58960	1
7	37	0	27.740	3	0	1	7281.50560	0
8	37	1	29.830	2	0	0	6406.41070	0
9	60	0	25.840	0	0	1	28923.13692	0
10	25	1	26.220	0	0	0	2721.32080	1
11	62	0	26.290	0	1	2	27808.72510	1
12	23	1	34.400	0	0	3	1826.84300	1
13	56	0	39.820	0	0	2	11090.71780	1
14	27	1	42.130	0	1	2	39611.75770	1
15	19	1	24.600	1	0	3	1837.23700	0
16	52	0	30.780	1	0	0	10797.33620	1
17	23	1	23.845	0	0	0	2395.17155	0
18	56	1	40.300	0	0	3	10602.38500	1
19	30	1	35.300	0	1	3	36837.46700	1
20	60	0	36.005	0	0	0	13228.84695	1
21	30	0	32.400	1	0	3	4149.73600	1
22	18	1	34.100	0	0	2	1137.01100	1
23	34	0	31.920	1	1	0	37701.87680	1
24	37	1	28.025	2	0	1	6203.90175	0
25	59	0	27.720	3	0	2	14001.13380	1
26	63	0	23.085	0	0	0	14451.83515	0
27	55	0	32.775	2	0	1	12268.63225	0
28	23	1	17.385	1	0	1	2775.19215	1
29	31	1	36.300	2	1	3	38711.00000	1

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             1338 non-null   int64  
 1   sex             1338 non-null   int64  
 2   bmi             1338 non-null   float64
 3   children        1338 non-null   int64  
 4   smoker          1338 non-null   int64  
 5   region          1338 non-null   int64  
 6   charges         1338 non-null   float64
 7   insuranceclaim  1338 non-null   int64  
dtypes: float64(2), int64(6)
memory usage: 83.8 KB
```

In [5]:
```python
df.describe()
```

Out[5]:

	age	sex	bmi	children	smoker	region	charges	insuranceclaim
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.204783	1.515695	13270.422265	0.585202
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237	0.492871
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900	0.000000
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150	0.000000
50%	39.000000	1.000000	30.400000	1.000000	0.000000	2.000000	9382.033000	1.000000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515	1.000000
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010	1.000000

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

Out[6]:
```
age               0
sex               0
bmi               0
children          0
smoker            0
region            0
charges           0
insuranceclaim    0
dtype: int64
```

In [7]:
```python
df.shape
```

Out[7]: (1338, 8)

In [8]:
```python
df.size
```

Out[8]: 10704

In [9]:
```python
df.columns
```

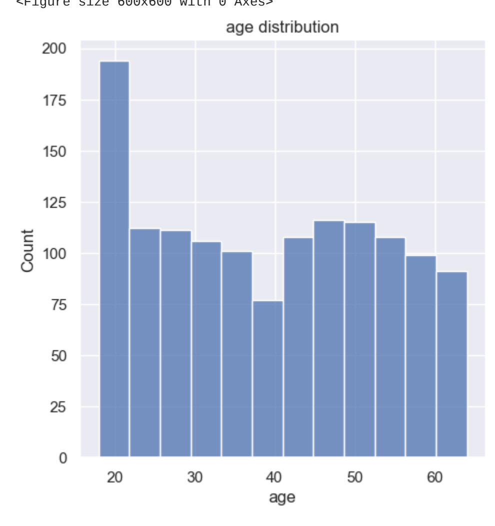
Out[9]:
```
Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges',
       'insuranceclaim'],
      dtype='object')
```

In [10]:
```python
df['sex'].value_counts()
```

Out[10]:
```
1    676
0    662
Name: sex, dtype: int64
```

In [11]:
```python
sns.set()
plt.figure(figsize=(6,6))
sns.displot(df['age'])
plt.title('age distribution')
plt.show()
```

```
<Figure size 600x600 with 0 Axes>
```



In [12]:
```python
dff=np.unique(df)
print(dff)
```

```
[0.00000000e+00 1.00000000e+00 2.00000000e+00 ... 6.00213990e+04
 6.25928731e+04 6.37704280e+04]
```

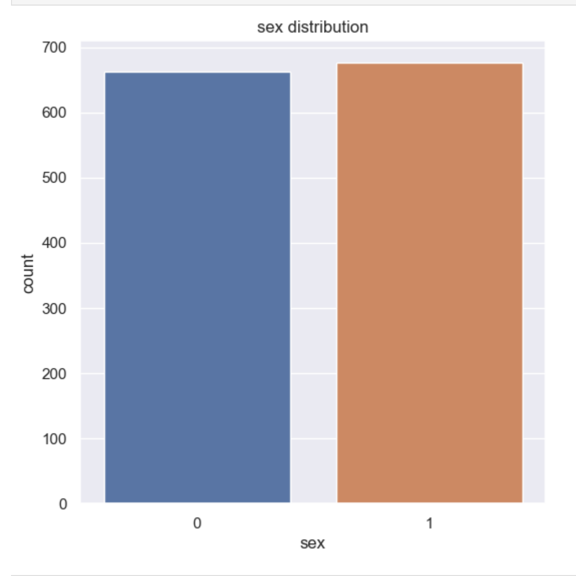
In [13]:
```python
corr = df.corr()
```

In [14]:
```python
corr
```

Out[14]:

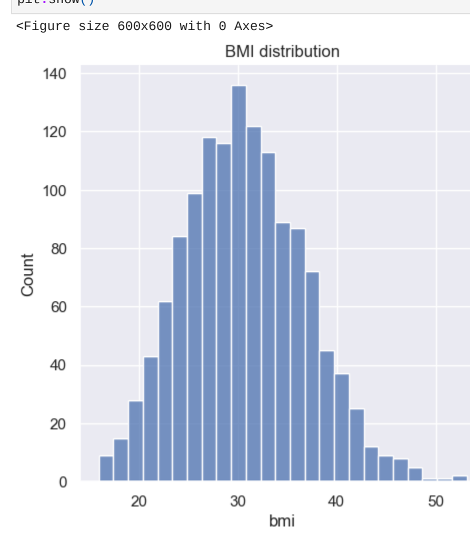
	age	sex	bmi	children	smoker	region	charges	insuranceclaim
age	1.000000	-0.020856	0.109272	0.042469	-0.025019	0.002127	0.299008	0.113723
sex	-0.020856	1.000000	0.046371	0.017163	0.076185	0.004588	0.057292	0.031565
bmi	0.109272	0.046371	1.000000	0.012759	0.003750	0.157566	0.198341	0.384198
children	0.042469	0.017163	0.012759	1.000000	0.007673	0.016569	0.067998	-0.409526
smoker	-0.025019	0.076185	0.003750	0.007673	1.000000	-0.002181	0.787251	0.333261
region	0.002127	0.004588	0.157566	0.016569	-0.002181	1.000000	-0.006208	0.020891
charges	0.299008	0.057292	0.198341	0.067998	0.787251	-0.006208	1.000000	0.309418
insuranceclaim	0.113723	0.031565	0.384198	-0.409526	0.333261	0.020891	0.309418	1.000000

In [15]:
```python
plt.figure(figsize=(6,6))
sns.countplot(x='sex',data=df)
plt.title('sex distribution')
plt.show()
```

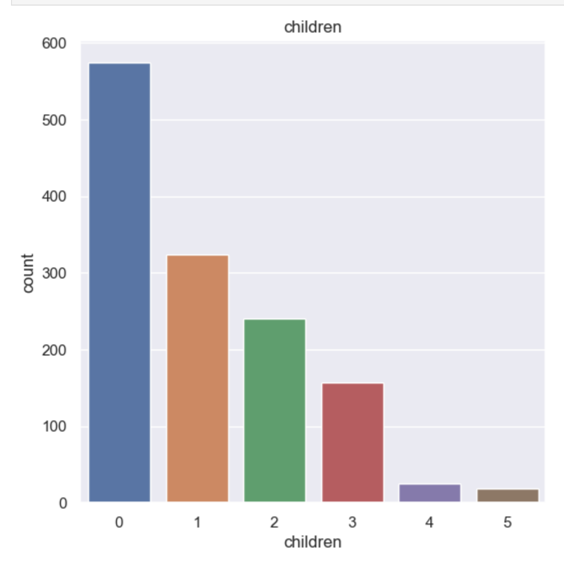


In [16]:
```python
plt.figure(figsize=(6,6))
sns.displot(df['bmi'])
plt.title('BMI distribution')
plt.show()
```

```
<Figure size 600x600 with 0 Axes>
```



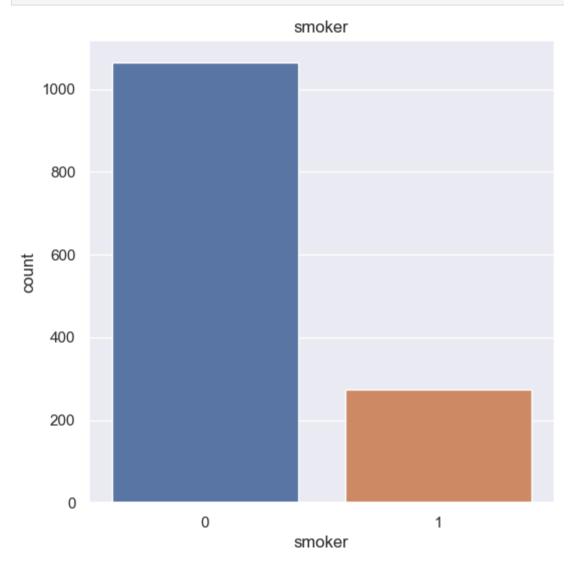
In [17]:
```python
plt.figure(figsize=(6,6))
sns.countplot(x='children',data=df)
plt.title('children')
plt.show()
```



In [18]:
```python
df['children'].value_counts()
```

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

In [19]:
```python
plt.figure(figsize=(6,6))
sns.countplot(x='smoker',data=df)
plt.title('smoker')
plt.show()
```

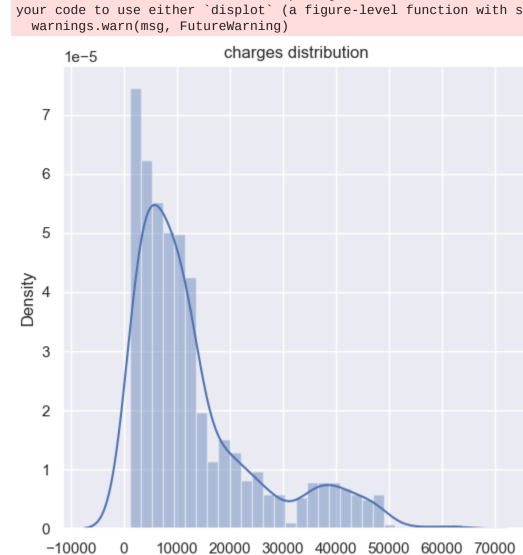


In [20]:
```python
df['smoker'].value_counts()
```

Out[20]:
```
0    1064
1     274
Name: smoker, dtype: int64
```

In [21]:
```python
plt.figure(figsize=(6,6))
sns.distplot(df['charges'])
plt.title('charges distribution')
plt.show()
```

```
C:\Users\Sonali\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated 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)
```



In [22]:
```python
X = df.drop(columns='charges', axis=1)
Y = df['charges']
print(X)
```

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

[1338 rows x 7 columns]
```

In [23]:
```python
print(Y)
```

```
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
```

In [31]:
```python
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
```

In [32]:
```python
X_train
```

Out[32]:

	age	sex	bmi	children	smoker	region	insuranceclaim
560	46	0	19.950	2	0	1	0
1285	47	0	24.320	0	0	0	0
1142	52	0	24.860	0	0	2	0
969	39	0	34.320	5	0	2	0
486	54	0	21.470	3	0	1	0
...	...	...	...	...	...	...	...
1095	18	0	31.350	4	0	0	0
1130	39	0	23.870	5	0	2	0
1294	58	1	25.175	0	0	0	0
860	37	0	47.600	2	1	3	1
1126	55	1	29.900	0	0	3	1

1070 rows × 7 columns

In [35]:
```python
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [36]:
```python
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[36]: GradientBoostingRegressor()

In [41]:
```python
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 [42]:
```python
df1
```

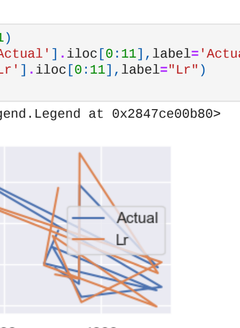
Out[42]:

	Actual	Lr	svm	rf	gr
764	9095.06825	7585.365409	9549.847772	14211.911225	11067.182447
887	5272.17580	6532.953193	9497.254380	5268.904884	5829.271736
890	29330.98315	37069.034525	9645.823319	28363.092036	27532.354796
1293	9301.89355	7903.623482	9556.241924	10923.492179	9743.834159
259	33750.29180	27160.887292	9427.256466	34591.833691	33626.881814
...	...	...	...	...	...
109	47055.53210	39693.372730	9645.836943	46888.208819	45727.511007
575	12222.89830	11224.198269	9623.037214	12145.653105	12365.233685
535	6067.12675	8410.955106	9508.142238	6505.093226	6798.054543
543	63770.42801	42111.654361	9604.068774	46682.190209	47981.343043
846	9872.70100	11651.236659	9590.166857	9653.774523	10374.446527

268 rows × 5 columns

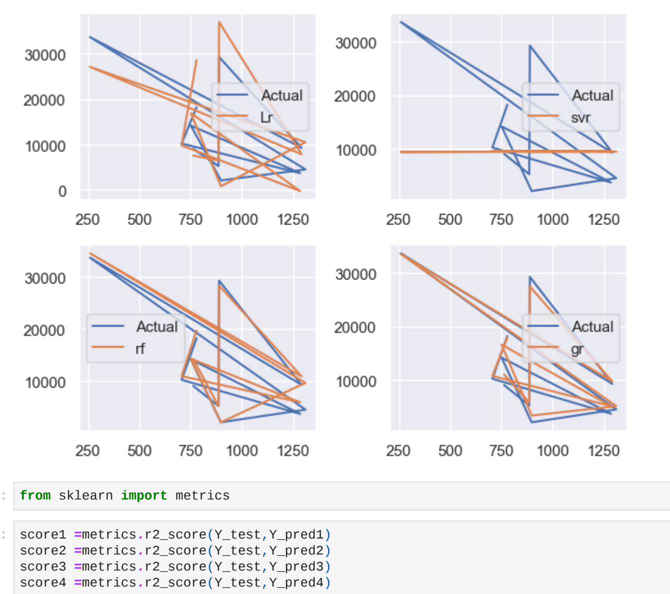
In [44]:
```python
plt.subplot(221)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['Lr'].iloc[0:11],label="Lr")
plt.legend()
```

Out[44]: <matplotlib.legend.Legend at 0x2847ce00b80>



In [46]:
```python
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[46]: <matplotlib.legend.Legend at 0x2847b573760>



In [47]:
```python
from sklearn import metrics
```

In [48]:
```python
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 [49]:
```python
print(score1,score2,score3,score4)
```

```
0.7824434217148323 -0.07236455414008303 0.8625838436393364 0.8785486163733124
```

In [52]:
```python
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_pred3)
```

In [53]:
```python
print(s1,s2,s3,s4)
```

```
4202.073483424077 8594.784126770652 2547.505711386106 2547.505711386106
```

In [ ]:
```python
# from this diiferent evaluation we conclude  that the 4th model is the best method for this dataset. For prediction and absolute error
# Gradient Boosting Regressor is best for this model.
```