import Dependencies

In [1]:

```
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.metrics import accuracy_score,classification_report,confusion_matrix
import warnings
warnings.filterwarnings("ignore")
```

Data Collection and Analysis

In [2]:

```
## loading the data from csv file to a pandas Dataframe
insurance_dataset = pd.read_csv('insurance dataset.csv')
insurance_dataset.head()
```

Out[2]:

	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

In [3]:

```
insurance_dataset.tail()
```

Out[3]:

	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

In [4]:

```
insurance_dataset.describe()
```

Out[4]:

	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

In [5]:

```
insurance_dataset.isnull().sum()
```

Out[5]:

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

In [6]:

```
insurance_dataset.shape
```

Out[6]:

```
(1338, 7)
```

In [7]:

```
insurance_dataset.columns
```

Out[7]:

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

In [8]:

```
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
```

In [9]:

```
null_values = 100 * (insurance_dataset.isnull().sum()/insurance_dataset.shape[0])
null_values
```

Out[9]:

```
age         0.0
sex         0.0
bmi         0.0
children    0.0
smoker      0.0
region      0.0
charges     0.0
dtype: float64
```

In [10]:

```
null_values[null_values>45].sort_values(ascending=False)
```

Out[10]:

```
Series([], dtype: float64)
```

In [11]:

```
cols = null_values[null_values>45].sort_values(ascending=False).index
insurance_dataset.drop(labels=cols,axis=1,inplace=True)
insurance_dataset.shape
```

Out[11]:

```
(1338, 7)
```

In [12]:

```
null_values = 100 * (insurance_dataset.isnull().sum()/insurance_dataset.shape[0])
null_values[(null_values <13) & (null_values>0)]
```

Out[12]:

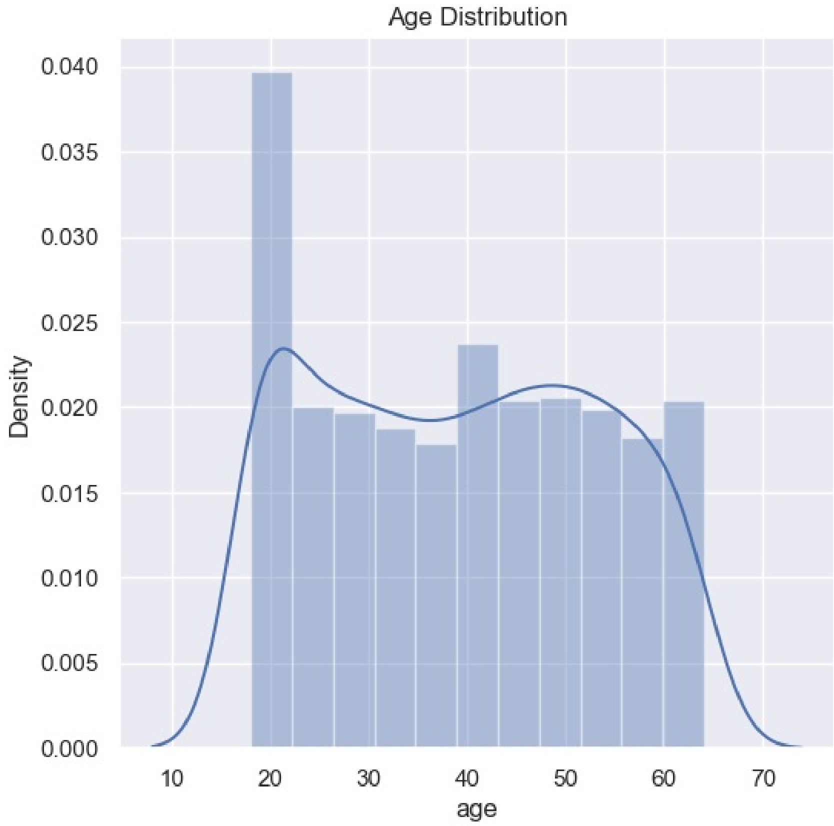
```
Series([], dtype: float64)
```

In [13]:

```
## categorical features: Sex,Smoker,Region
```

In [14]:

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

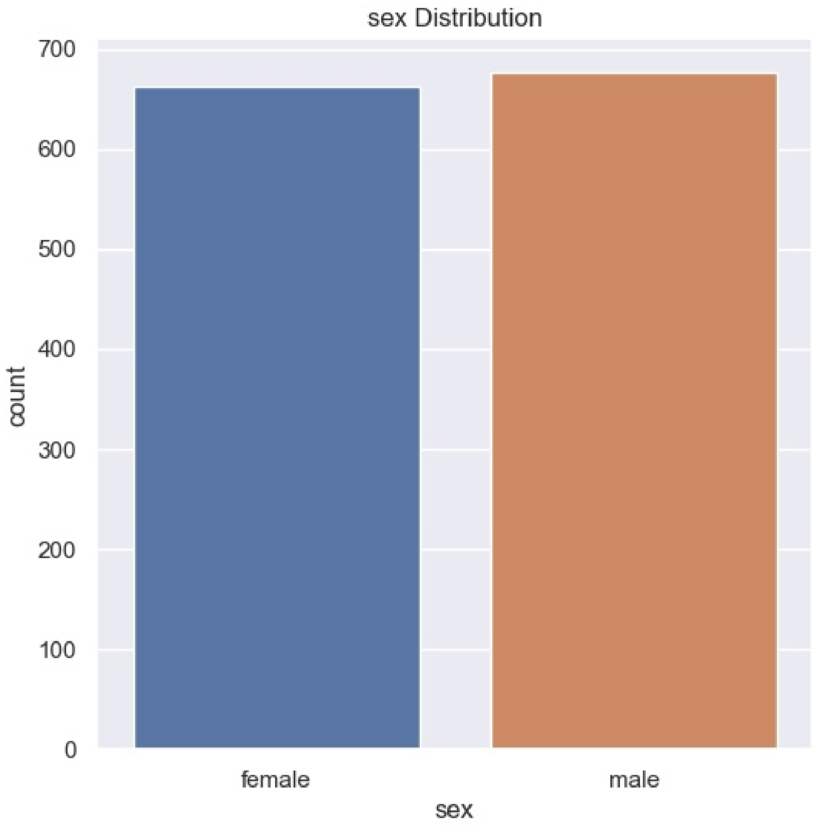


In [15]:

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

Out[15]:

```
Text(0.5, 1.0, 'sex Distribution')
```



In [16]:

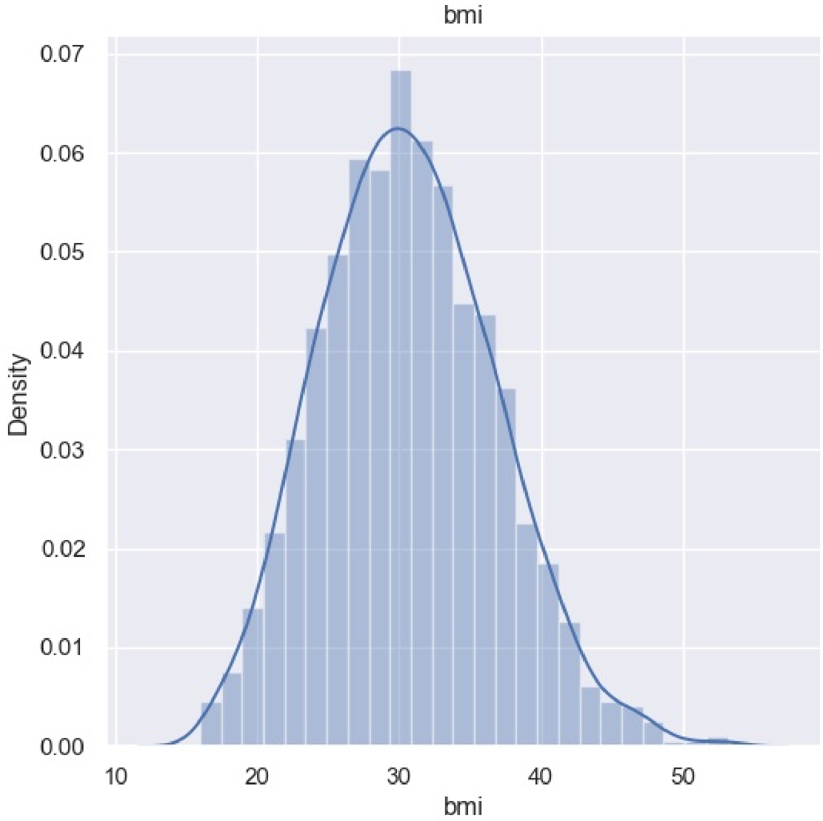
```
insurance_dataset['sex'].value_counts()
```

Out[16]:

```
male      676
female    662
Name: sex, dtype: int64
```

In [17]:

```
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['bmi'])
plt.title('bmi')
plt.show()
```



In [18]:

```
insurance_dataset.replace({'sex':{'male':0,'female':1}},inplace = True)
insurance_dataset.replace({'smoker':{'yes':0,'no':1}},inplace = True)
insurance_dataset.replace({'region':{'northwest':1,'northeast':2, 'southwest':3,'southeast':4}},inplace = True)
```

In [19]:

```
X = insurance_dataset.drop(columns='charges',axis=1)
y = insurance_dataset['charges']
```

In [20]:

```
print(X)
```

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

[1338 rows x 6 columns]
```

In [21]:

```
print(y)
```

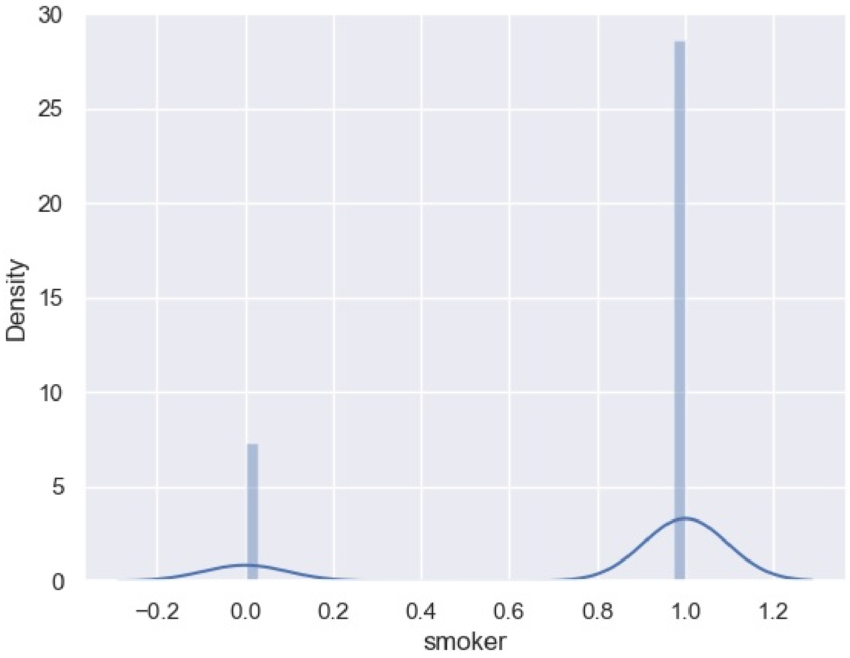
```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

In [22]:

```
sns.distplot(insurance_dataset.smoker)
```

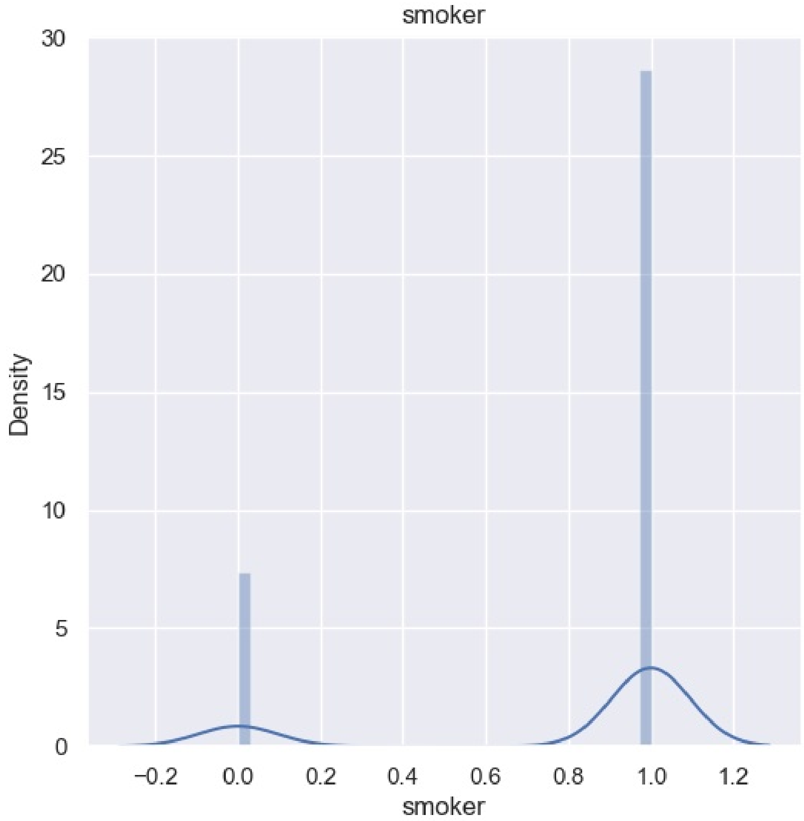
Out[22]:

```
<AxesSubplot:xlabel='smoker', ylabel='Density'>
```



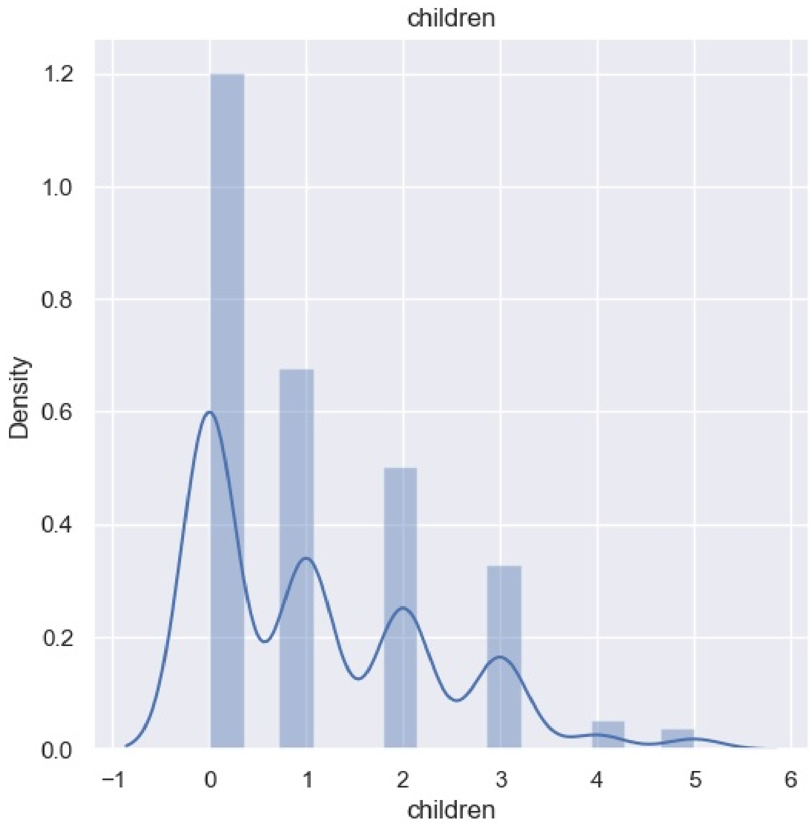
In [23]:

```
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['smoker'])
plt.title('smoker')
plt.show()
```



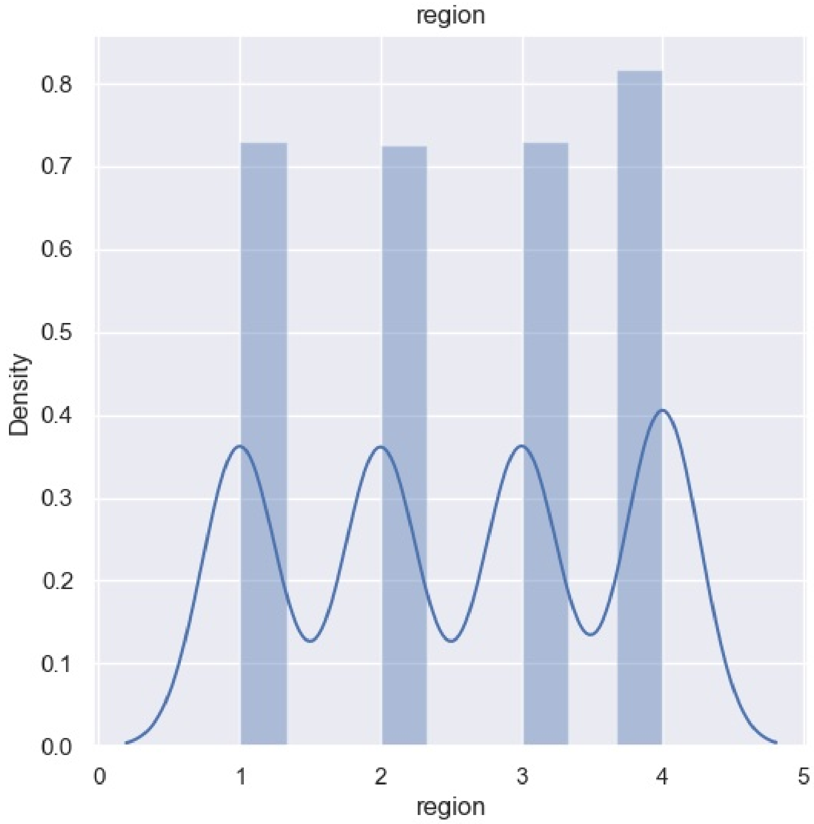
In [24]:

```
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['children'])
plt.title('children')
plt.show()
```



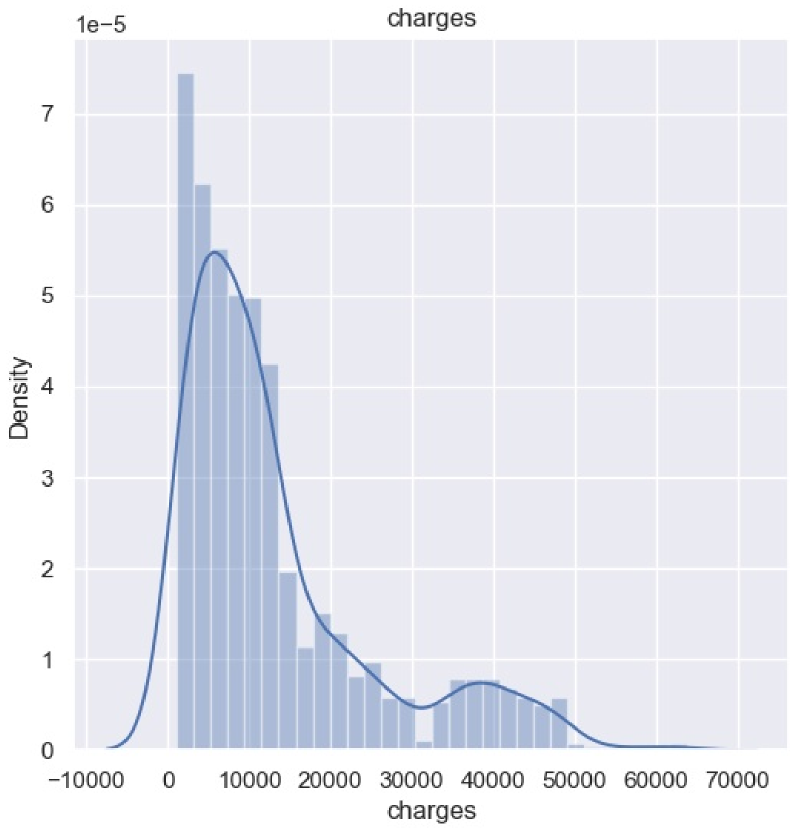
In [25]:

```
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['region'])
plt.title('region')
plt.show()
```



In [26]:

```
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['charges'])
plt.title('charges')
plt.show()
```



In [27]:

```
charges = insurance_dataset["charges"].value_counts(normalize=True)
```

In [28]:

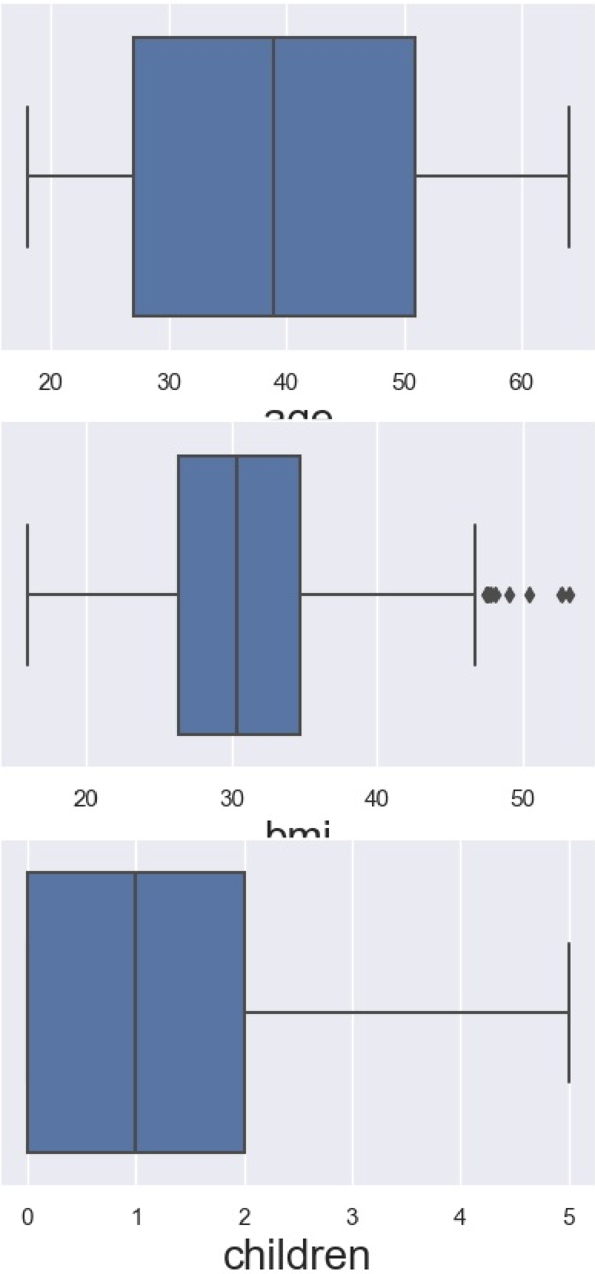
```
insurance_dataset.columns
```

Out[28]:

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

In [29]:

```
box=insurance_dataset[['age', 'bmi','children']]
plt.figure(figsize=(5,10), facecolor='white')
plotnumber = 1

for column in box:
    if plotnumber<=6 :     # as there are 9 columns in the data
        ax = plt.subplot(3,1,plotnumber)
        sns.boxplot(x=box[column])
        plt.xlabel(column,fontsize=20)
        #plt.ylabel('Salary',fontsize=20)
    plotnumber+=1
plt.show()
```



In [30]:

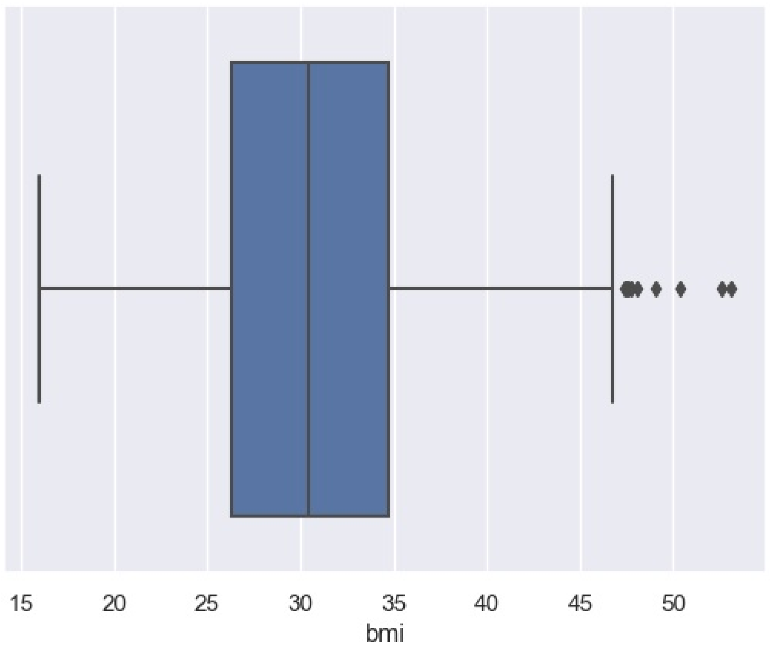
```
## Analysis of continous variables
def findoutliers(column):
    outliers=[]
    Q1=column.quantile(.25)
    Q3=column.quantile(.75)
    IQR=Q3-Q1
    lower_limit=Q1-(1.5*IQR)
    upper_limit=Q3+(1.5*IQR)
    for out1 in column:
        if out1>upper_limit or out1 <lower_limit:
            outliers.append(out1)

    return np.array(outliers)
```

In [31]:

```
sns.boxplot(insurance_dataset.bmi)
```

Out[31]:

```
<AxesSubplot:xlabel='bmi'>
```



In [32]:

```
from scipy import stats
for column in box:
    print(stats.skew(box[column]),column)
```

```
0.055610083072599126 age
0.28372857291709386 bmi
0.9373281163874423 children
```

In [33]:

```
for column in box:
    print(stats.kurtosis(box[column]),column)
```

```
-1.2449206804584227 age
-0.05502310583700032 bmi
0.1972174268623732 children
```

In [34]:

```
insurance_dataset.head()
```

Out[34]:

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

In [35]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler() ## object creation
insurance_dataset[['age', 'sex','bmi']]= sc.fit_transform(insurance_dataset[['age', 'sex','bmi']])
```

In [36]:

```
insurance_dataset.head()
```

Out[36]:

	age	sex	bmi	children	smoker	region	charges
0	-1.438764	1.010519	-0.453320	0	0	3	16884.92400
1	-1.509965	-0.989591	0.509621	1	1	4	1725.55230
2	-0.797954	-0.989591	0.383307	3	1	4	4449.46200
3	-0.441948	-0.989591	-1.305531	0	1	1	21984.47061
4	-0.513149	-0.989591	-0.292556	0	1	1	3866.85520

In [37]:

```
insurance_dataset.corr()
```

Out[37]:

	age	sex	bmi	children	smoker	region	charges
age	1.000000	0.020856	0.109272	0.042469	0.025019	-0.005212	0.299008
sex	0.020856	1.000000	-0.046371	-0.017163	0.076185	-0.016121	-0.057292
bmi	0.109272	-0.046371	1.000000	0.012759	-0.003750	0.261829	0.198341
children	0.042469	-0.017163	0.012759	1.000000	-0.007673	-0.019257	0.067998
smoker	0.025019	0.076185	-0.003750	-0.007673	1.000000	-0.053930	-0.787251
region	-0.005212	-0.016121	0.261829	-0.019257	-0.053930	1.000000	0.056993
charges	0.299008	-0.057292	0.198341	0.067998	-0.787251	0.056993	1.000000

In [38]:

```
sns.heatmap(insurance_dataset.corr(),annot=True)
```

Out[38]:

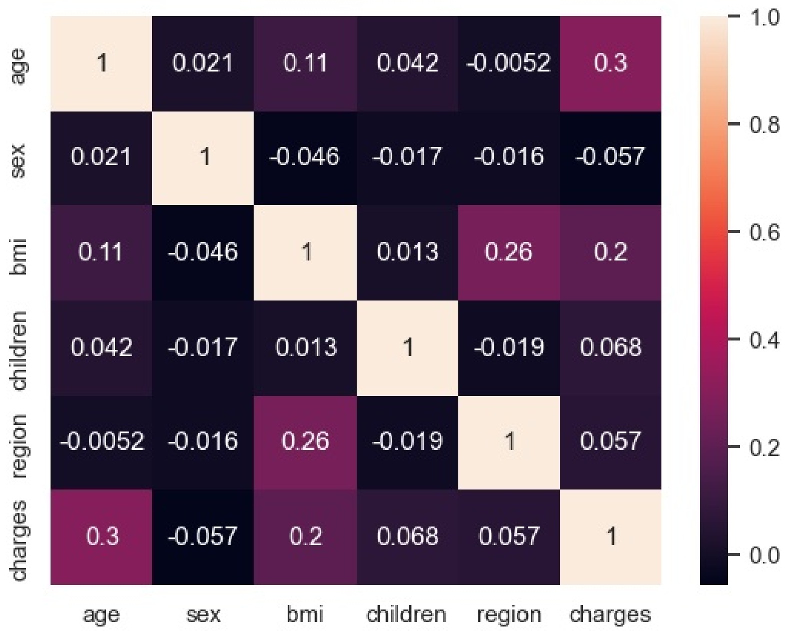
```
<AxesSubplot:>
```

In [39]:

```
sns.heatmap(insurance_dataset.drop('smoker',axis=1).corr(),annot=True)
```

Out[39]:

```
<AxesSubplot:>
```



In [40]:

```
X=insurance_dataset.iloc[:,0:-1]
y=insurance_dataset.charges
```

In [41]:

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [42]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

In [43]:

```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_train,y_train)
y_hat=LR.predict(X_test)
```

In [44]:

```
y_train_predict=LR.predict(X_train)
from sklearn.metrics import r2_score
train_score=r2_score(y_train,y_train_predict)
train_score
```

Out[44]:

```
0.7445275825163911
```

In [45]:

```
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[45]:

```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [46]:

```
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 [47]:

```
df1
```

Out[47]:

	Actual	LR	svm	rf	gr
764	9095.06825	8569.027921	9437.071321	10426.627536	10439.228445
887	5272.17580	7226.242261	9417.213949	5398.134939	5606.916108
890	29330.98315	37082.125966	9492.950021	28268.059099	28566.928173
1293	9301.89355	9704.404710	9440.574065	12008.725036	9902.029326
259	33750.29180	27154.894451	9403.306521	34516.069995	34032.647133
...	...	...	...	...	...
342	13217.09450	12407.173239	9482.058145	12908.002340	15095.379369
308	11944.59435	14400.524856	9484.637128	12210.065148	13087.304572
1128	14358.36437	7695.376058	9405.368238	5217.074133	5348.881694
503	32548.34050	25958.982415	9399.437172	33825.497241	35494.122600
1197	5699.83750	9126.299697	9429.945653	7417.260636	6097.325465

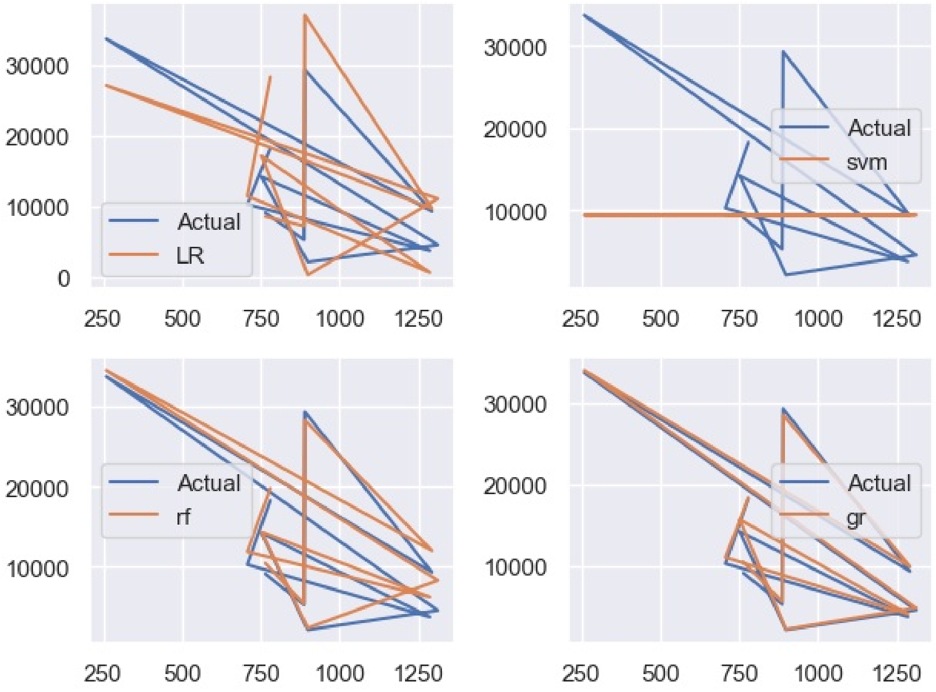
335 rows × 5 columns

In [48]:

```
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="svm")
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[48]:

```
<matplotlib.legend.Legend at 0x1d859f09220>
```



In [49]:

```
from sklearn import metrics
```

In [50]:

```
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 [51]:

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

```
0.7667469908213234 -0.09548003110575842 0.8509953382461486 0.8623623097145213
```

In [52]:

```
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 [53]:

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

```
4245.940270673539 8487.96113724879 2609.6138757475373 2491.428032536737
```

In [54]:

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

df = pd.DataFrame(df1,index=[0])
df
```

Out[54]:

	Actual	LR	svm	rf	gr
0	NaN	NaN	NaN	NaN	NaN

In [56]:

```
model = LinearRegression()
```

In [57]:

```
model.fit(X_train,y_train)
```

Out[57]:

```
▾ LinearRegression
LinearRegression()
```

In [58]:

```
y_predict = model.predict(X_test)
y_predict
```

Out[58]:

```
array([ 8569.02792102,  7226.24226101, 37082.12596551,  9704.40470976,
       27154.89445082, 11099.50396214,   287.75903995, 17192.26142479,
         649.86159841, 11475.04981963, 28330.08174632,  9614.31257617,
        5059.10380215, 38318.60361415, 40188.64744507, 37002.33742225,
       15084.32748246, 35750.87578953,  8949.03881572, 31635.73858748,
        4066.74697862, 10388.4371046 ,  2569.89417776,  6682.25980692,
       11495.6971861 , 12580.08990064, 14740.82601338,  6299.0628445 ,
        9568.01685349,  2011.8942882 ,  9304.15869233, 13280.99111107,
        4337.56975838,  3555.02545142,  4676.84319149, 12693.94100914,
        2120.14150719,  9005.52894884, 33467.65638726, 32462.62589045,
        4048.14515865,  4516.4774556 , 14365.65205159, 11701.01027073,
        8645.58566659, 12392.84601145,  5405.29550866,  3302.36665877,
       35390.2179744 ,  9016.67395506, 15712.31509893,  2178.67516687,
       12532.23586296,  1080.70337419, 13313.1736781 , 12249.76589852,
        3968.79155916, 32378.52095891, 13548.03772002, 12512.8522698 ,
       14406.64154274, 10304.94230562, 16662.72092772,  7937.95162592,
       11453.0248688 ,  4200.91345457, 26860.37430507, 10848.22685655,
        2270.17322048,  6415.03505514, 10376.23033043, 11229.80656421,
       10821.02257102,  9052.38187532, 11850.71961171,  6559.48674144,
        6871.07358599, 10940.40271632,  6375.36999404,  8961.26157038,
        3605.76299802, 36334.95752609,  6545.25863428, 30388.25245614,
       34689.49730632, 34951.76350878,  6884.99673484, 13129.06806407,
       10118.44489644, 14756.42787186, 17332.59555817, 35136.29466403,
       32632.27991515,  5817.18791265, 31869.94385246,  9734.91659808,
       29311.96313115,  3447.94875261, 28454.42910729,  5399.16767478,
        5552.11166032,  2017.74036322, 11387.85397781, 15365.96202819,
       11400.33526422,  4143.63216517,  9793.02086967, 31625.59680288,
        -488.22884315, 32672.65392581,  3479.90856806, 10037.13641999,
       13947.74344683, 31213.86729183, 11069.25400533,  4131.85961238,
       12805.97277618, 32003.95856322,  7968.04280629,  3431.03019208,
        8011.47195267, 10831.03552022, 14835.13483364,  5495.88730559,
```

```
        3922.7984091 , 10400.21695772, 10729.13503983, 10795.65402128,
       14304.16137586,  7275.64184467,  5188.40767871,  9108.622143  ,
        9154.77362236, 12135.04634727,  8519.24119177, 15628.4527325 ,
        8044.80645777, 32050.82985586, 35961.50652253, 31181.39083233,
        6169.61444117, 12218.28188389,  5866.29222402, 14757.98070434,
        2344.37033373, 33235.91290358,  6482.87051707,  5197.37019545,
       14540.7041274 ,  7179.53039403, 38870.1644984 ,  3214.28862535,
        5719.36032273, 31318.09644359, 11725.36107956,  8112.04408862,
       14509.38025521,  9653.28651977, 26876.33332002, 33104.73945924,
       14181.93279886,  1802.70938573, 13475.5279152 ,  1833.13885138,
        5661.40924425, 11461.72227023, 40045.85319368, 36707.51962413,
       33661.86120621,  4045.71035614,  7663.075274  ,  8844.43198283,
       12098.40107502,  5025.55905867,  2189.11944274, 32015.85069869,
       25310.54147444, 17427.09423634, 26561.6496257 , 10367.25985029,
       36899.9283149 ,  -899.39248406,  7029.91287264,  7952.12392343,
        3995.76051237,  5234.68677166,  5544.58037346,  4466.15910206,
       15029.11226984, 11400.04072251,  7144.39565806,  2058.72190481,
        1078.93045928, 32173.93418977, 16352.27077135, 11858.60053315,
        1386.46206276, 12151.93590908,  1098.1511595 ,  8945.40572995,
        2007.16442476, 34176.58999523, 10722.27917345,  2787.9075487 ,
       25448.73296133, 26133.47337302,  9687.03450001,  1928.47755998,
       13166.24177681,  1270.11137287, 10688.63941275, 10430.14596192,
       16544.96468825, 26699.27972475,  6827.73141209,  4754.69594071,
        5651.61499763, 13099.069636  , 10967.27110227,  8172.35021609,
        5257.10705303, 12549.549249  , 13742.61300114, 35668.33301987,
        4346.32615326, 28775.11901906,  -779.71128102,  3055.96641604,
       10959.73186585, 15529.11322085,  5051.50390898,  7082.1218293 ,
        4014.58298733, 31568.68842886,  7437.53068374, 12313.35374635,
        5820.0929521 ,  9782.04187267, 36140.95884283,  4272.76519922,
        9301.32782869, 31385.59673653,  5878.26684969,  4392.27244347,
        1230.88199995,  5031.52399961,  4732.65407107,  6738.51742109,
       18513.05575823, -1362.65901238,  2513.02179987, 10946.69965627,
        3307.07504537,  9836.05250756,  3851.92697572,  4911.48853081,
       12697.75360375,  6418.39824601,  7903.88560722,  7157.47189468,
        9094.2935361 , 10700.24842336, 27663.05005121, 39001.74578648,
       11971.41067363,  7334.42297028, 40773.35146162, 12529.62536522,
        7278.35042883,  7946.95325506,  9003.68084405, 10952.04600043,
       10291.77256667, 17501.41439954,  1245.40081577, 23194.45186866,
       12303.77098336, 32806.96949218,  4834.19688597, 13073.15834804,
       10565.79637494, 17414.93943876, 10153.51485353, 11640.19501548,
       32493.27022565,  2928.18485837, 13508.36782564, 39263.03749076,
        5131.53263165,  6309.08541219,  2987.27362527, 11650.06437061,
       25117.91168156, 13768.37326011,  9180.95285283,  9712.94890425,
       13388.40871061,  1192.420786  ,  2298.29259688, 30592.46074602,
       30245.64951003, 13464.26645641,  3861.98315385, 25596.01841161,
       13349.62562516, 30591.43569096,  3057.71083727, 38853.27630259,
       11055.26774652,  4708.82888937,  7041.82850487,  2401.12312894,
       26085.44509308, 14619.14907632,   784.85806779, 13014.34366009,
       12571.8513137 , 15109.27510831, 34909.10160246, 14546.64585506,
       31515.2270344 , 10249.4467815 , 18428.4812446 ,  6186.1096246 ,
        8837.81483469,  9592.67543399, 15184.59537686,  9333.18469118,
        7972.61954414, 15266.69980863, 12407.17323852, 14400.52485605,
        7695.37605771, 25958.98241542,  9126.29969674])
```

In [59]:

```
test_score=r2_score(y_test,y_hat)
test_score
```

Out[59]:

```
0.7667469908213234
```

In [60]:

```
X_train.shape,X_test.shape
```

Out[60]:

```
((1003, 6), (335, 6))
```

In [61]:

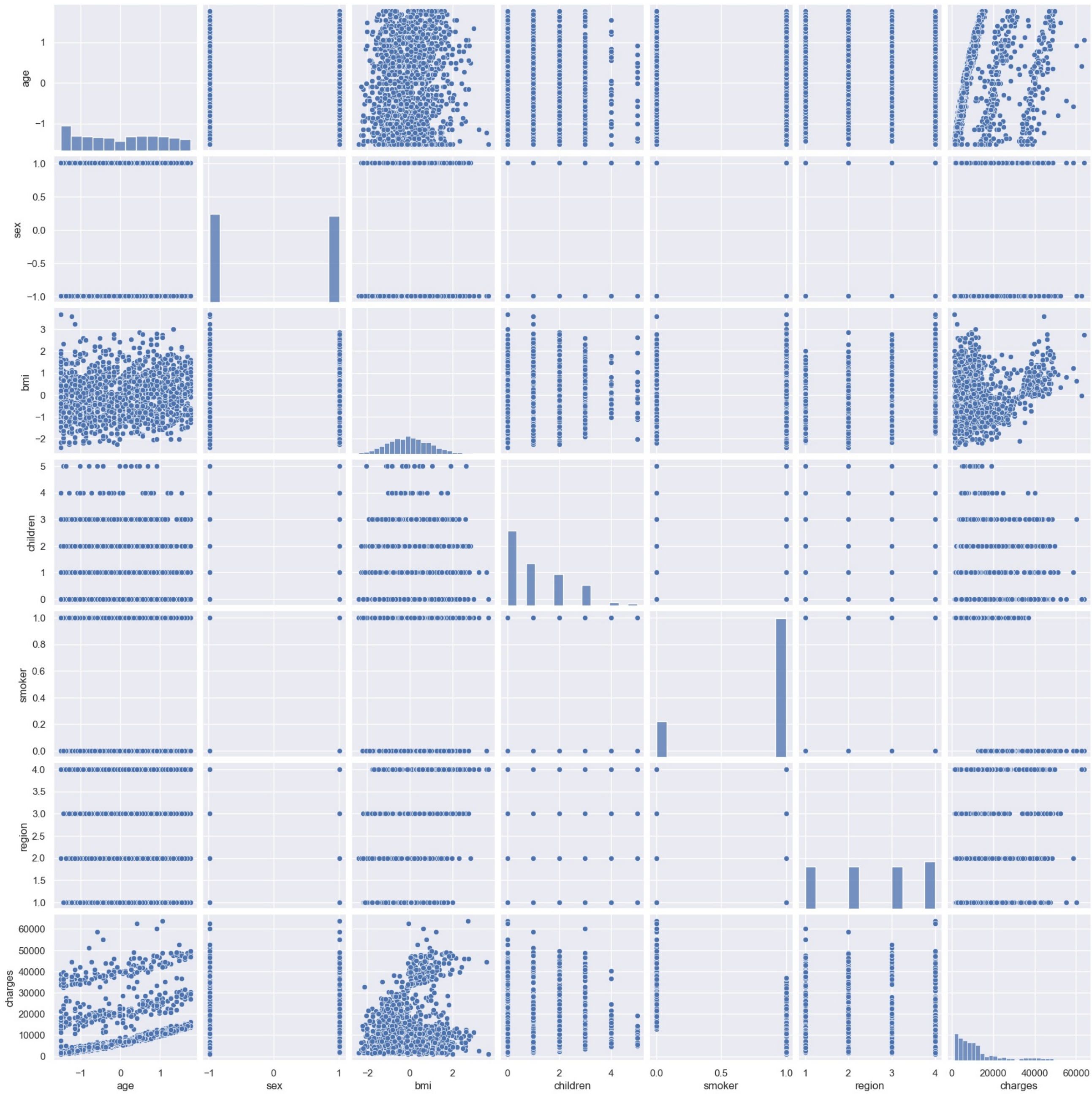
```
import seaborn as sb
```

In [62]:

```
sb.pairplot(insurance_dataset)
```

Out[62]:

```
<seaborn.axisgrid.PairGrid at 0x1d8586cfca0>
```

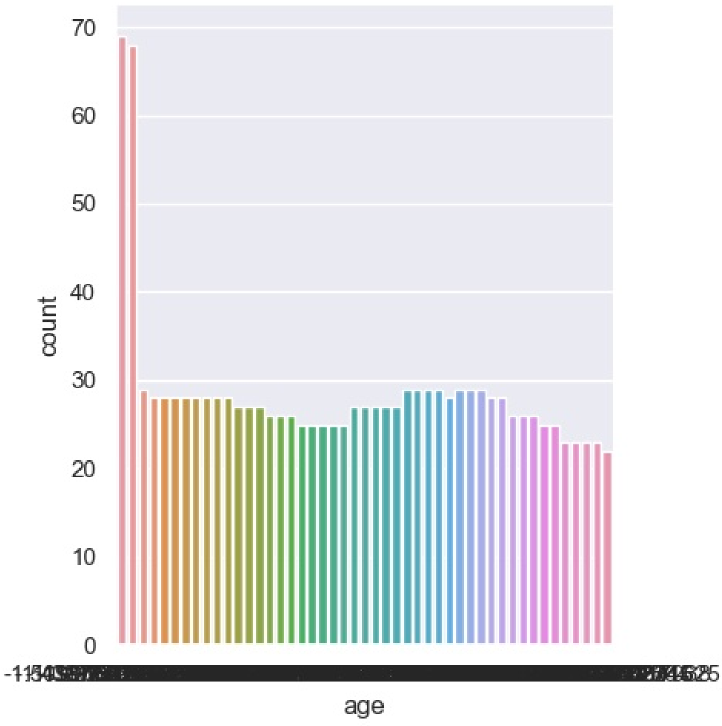


In [63]:

```
sb.factorplot('age',data=insurance_dataset,kind='count')
```

Out[63]:

```
<seaborn.axisgrid.FacetGrid at 0x1d85a2e85b0>
```



In [64]:

```
# model training

# linear regression

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

Out[64]:

```
▾ LinearRegression
LinearRegression()
```

In [65]:

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

Out[65]:

```
▾ LinearRegression
LinearRegression(normalize=False)
```

In [66]:

```
training_data_prediction = regressor.predict(X_train)
```

In [67]:

```
r2_train = metrics.r2_score(y_train,training_data_prediction)
print('R squared value :',r2_train)
```

```
R squared value : 0.7445275825163911
```

In [68]:

```
test_data_prediction = regressor.predict(X_test)
```

In [69]:

```
r2_test= metrics.r2_score(y_test,test_data_prediction)
print('R squared value :',r2_test)
```

```
R squared value : 0.7667469908213234
```

Building a predictive system

In [70]:

```
input_data = (31,1,25.74,0,1,0)

input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = regressor.predict(input_data_reshaped)
print(prediction)
```

```
[174289.10008086]
```

In [71]:

```
print('The insurance cost is USD',prediction[0])
```

```
The insurance cost is USD 174289.1000808612
```

In [72]:

```
if (prediction[0])==0:
    print('The person has no  insursnce')
else:
    print('The person has insurance')
```

```
The person has insurance
```

saving the train model

In [73]:

```
import pickle
```

In [74]:

```
filename = 'trained_model.sav'
pickle.dump(regressor,open(filename,'wb'))
```

loading the saved model

In [75]:

```
loaded_model = pickle.load(open('trained_model.sav','rb'))
```

In [76]:

```
input_data = (31,1,25.74,0,1,0)

input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = regressor.predict(input_data_reshaped)
print(prediction)


if (prediction[0])==0:
    print('The person has no  insursnce')
else:
    print('The person has insurance')
```

```
[174289.10008086]
The person has insurance
```

In [ ]:

```

```