In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.graph_objs as go
!pip install statsmodels
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: statsmodels in c:\programdata\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: numpy>=1.17 in c:\programdata\anaconda3\lib\site-packages (from statsmodel
s) (1.21.5)
Requirement already satisfied: scipy>=1.3 in c:\programdata\anaconda3\lib\site-packages (from statsmodel
s) (1.7.3)
Requirement already satisfied: pandas>=0.25 in c:\programdata\anaconda3\lib\site-packages (from statsmode
ls) (1.4.2)
Requirement already satisfied: patsy>=0.5.2 in c:\programdata\anaconda3\lib\site-packages (from statsmode
ls) (0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\programdata\anaconda3\lib\site-packages (from statsm
odels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\programdata\anaconda3\lib\site-packages (fr
om packaging>=21.3->statsmodels) (3.0.4)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=
0.25->statsmodels) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from
pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from patsy>=0.5.2->stat
smodels) (1.16.0)
```

In [2]:

```
df = pd.read_csv("C:\\Users\\HP\\Downloads\\archive (8)\\insurance.csv")
df.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]:

```
df.head
```

Out[3]:

```
<bound method NDFrame.head of       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 x 7 columns]>
```

In [4]:

```
df.shape
```

Out[4]:

```
(1338, 7)
```

In [5]:

```
df.dtypes
```

Out[5]:

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

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

In [7]:

```
df.tail()
```

Out[7]:

	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

data preprocessing

In [8]:

```
df.shape
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

In [9]:

```
df.isnull().sum()
```

Out[9]:

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

In [10]:

```
df.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 [11]:

```
df.age
```

Out[11]:

```
0       19
1       18
2       28
3       33
4       32
        ..
1333    50
1334    18
1335    18
1336    21
1337    61
Name: age, Length: 1338, dtype: int64
```

In [12]:

```
df.sex
```

Out[12]:

```
0       female
1         male
2         male
3         male
4         male
         ...
1333      male
1334    female
1335    female
1336    female
1337    female
Name: sex, Length: 1338, dtype: object
```

In [13]:

```
df.bmi
```

Out[13]:

```
0       27.900
1       33.770
2       33.000
3       22.705
4       28.880
         ...
1333    30.970
1334    31.920
1335    36.850
1336    25.800
1337    29.070
Name: bmi, Length: 1338, dtype: float64
```

In [14]:

```
df.children
```

Out[14]:

```
0       0
1       1
2       3
3       0
4       0
       ..
1333    3
1334    0
1335    0
1336    0
1337    0
Name: children, Length: 1338, dtype: int64
```

In [15]:

```
df.smoker
```

Out[15]:

```
0       yes
1        no
2        no
3        no
4        no
       ...
1333     no
1334     no
1335     no
1336     no
1337    yes
Name: smoker, Length: 1338, dtype: object
```

In [16]:

```
df.region
```

Out[16]:

```
0       southwest
1       southeast
2       southeast
3       northwest
4       northwest
          ...
1333    northwest
1334    northeast
1335    southeast
1336    southwest
1337    northwest
Name: region, Length: 1338, dtype: object
```

In [17]:

```
df.charges
```

Out[17]:

```
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
```

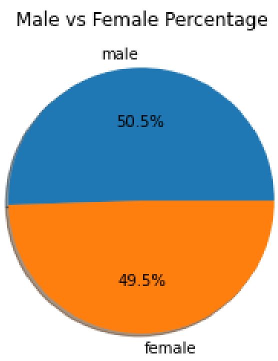
Exploratory data analysis

In [18]:

```
s = df['sex'].value_counts()
plt.pie(s, labels = s.index, autopct='%1.1f%%', shadow=True)
plt.title("Male vs Female Percentage")
```

Out[18]:

```
Text(0.5, 1.0, 'Male vs Female Percentage')
```

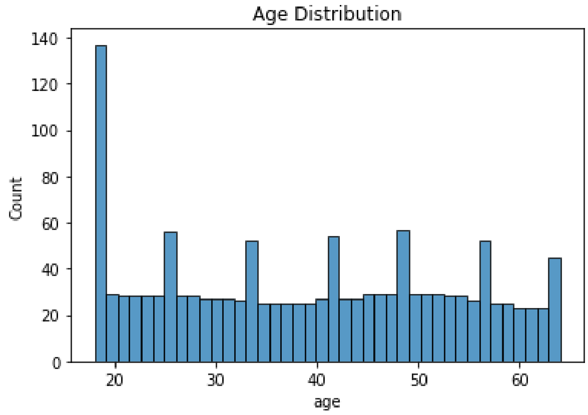


In [19]:

```
sns.histplot(data=df, x='age', bins=40)
plt.title("Age Distribution")
```

Out[19]:

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



In [20]:

```
#Analysis of BMI and Charges
#Let's dive deeper into understanding and visualizing this data. Below, I'll specify weight classes based on BMI to take

#bmi = df["bmi"]
```

In [21]:

```
bmi = df["bmi"]

cond_list = [bmi < 18.5, bmi < 25, bmi < 30, bmi >= 30]
choice_list = ['underweight', 'normal', 'overweight', 'obese']

df["bmi_cat"] = np.select(cond_list, choice_list)
df.head(10)
```

Out[21]:

	age	sex	bmi	children	smoker	region	charges	bmi_cat
0	19	female	27.900	0	yes	southwest	16884.92400	overweight
1	18	male	33.770	1	no	southeast	1725.55230	obese
2	28	male	33.000	3	no	southeast	4449.46200	obese
3	33	male	22.705	0	no	northwest	21984.47061	normal
4	32	male	28.880	0	no	northwest	3866.85520	overweight
5	31	female	25.740	0	no	southeast	3756.62160	overweight
6	46	female	33.440	1	no	southeast	8240.58960	obese
7	37	female	27.740	3	no	northwest	7281.50560	overweight
8	37	male	29.830	2	no	northeast	6406.41070	overweight
9	60	female	25.840	0	no	northwest	28923.13692	overweight

In [22]:

```
print(df.region.value_counts())
```

```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

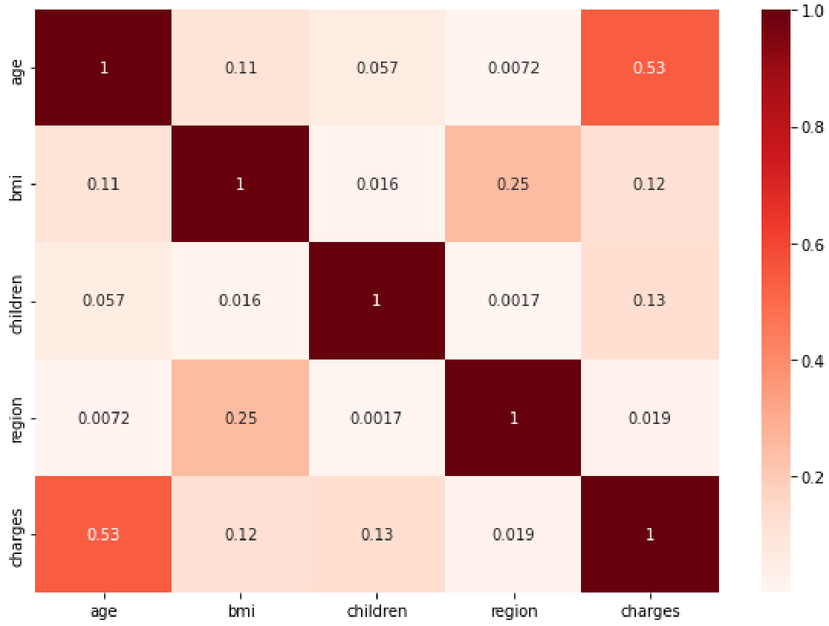
In [23]:

```
df.region = df.region.replace({'southeast': 0,
                               'southwest': 1,
                               'northwest': 2,
                               'northeast': 3})
print(df.region.value_counts())
```

```
0    364
1    325
2    325
3    324
Name: region, dtype: int64
```

In [24]:

```
corr_matrix = df.corr(method='spearman').abs()

plt.figure(figsize=(10, 7))
sns.heatmap(corr_matrix, annot=True, cmap="Reds")
plt.show()
```

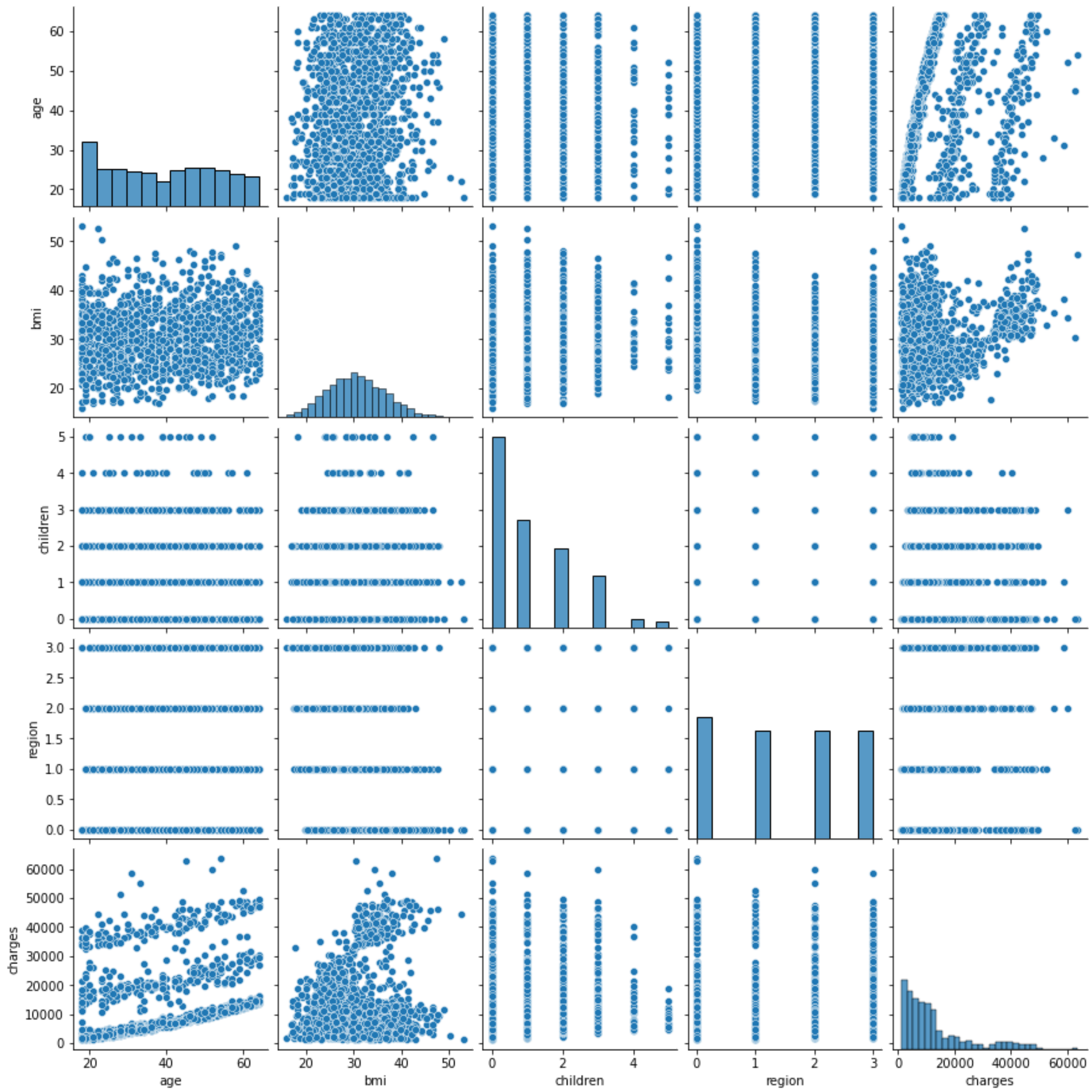


In [25]:

```
sns.pairplot(df)
```

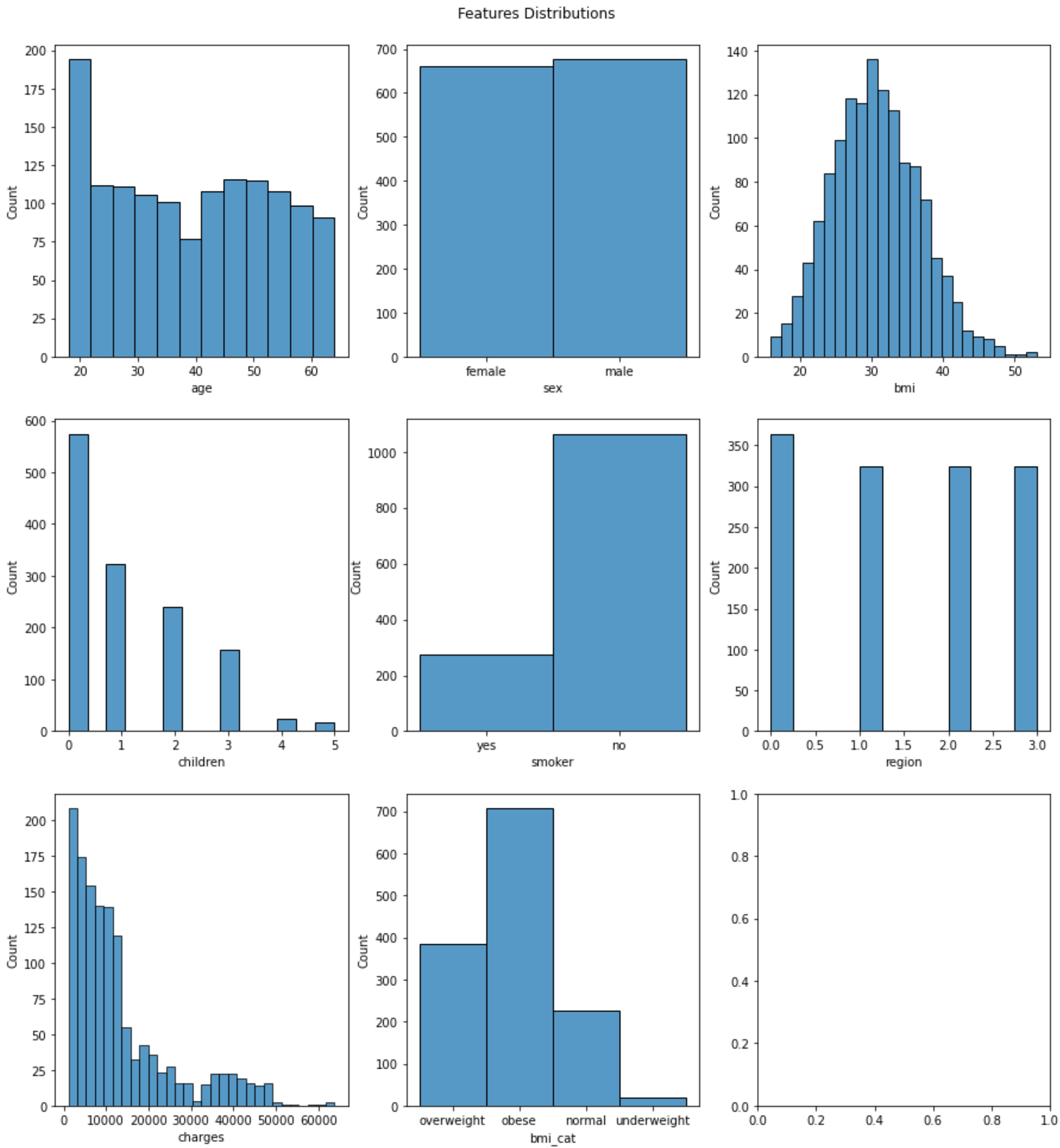
Out[25]:

<seaborn.axisgrid.PairGrid at 0x157bc376d30>



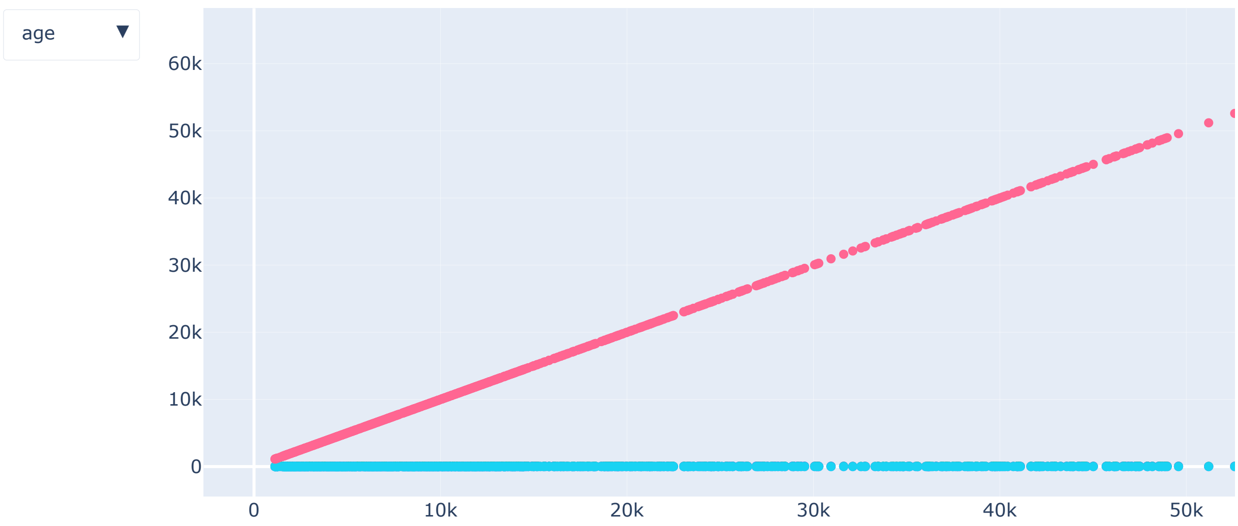
In [26]:

```
#Distribution for each feature
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15))

for i in range(len(df.columns)):
    sns.histplot(data=df, x = df.columns[i], ax = axes[i // 3][i % 3])

fig.suptitle("Features Distributions")

plt.subplots_adjust(top=0.95)

plt.show()
```



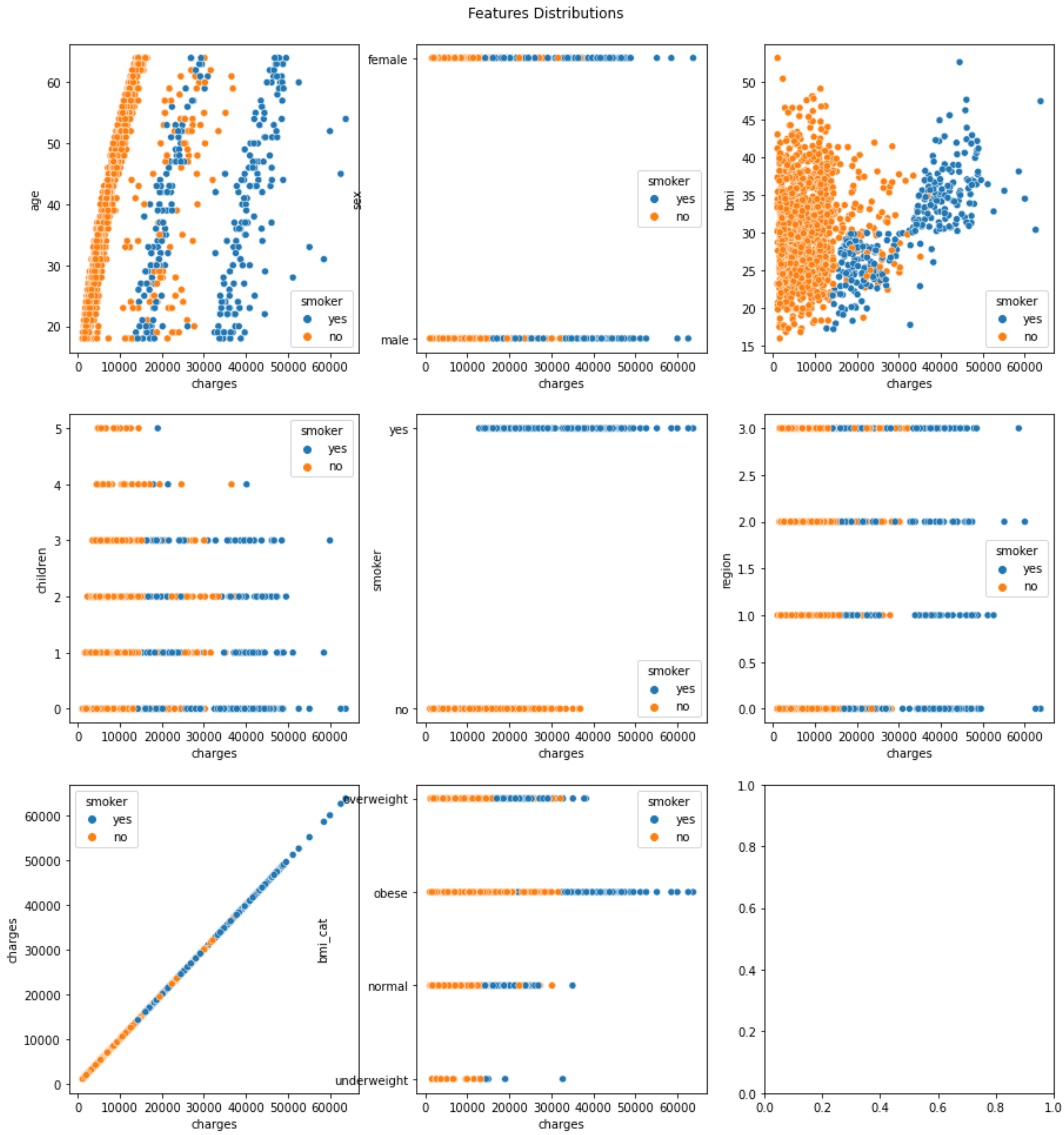
In [27]:

```
# Some plots to see the relation between charges and each feature
fig = go.Figure()

for i in range(len(df.columns)):
    fig.add_trace(go.Scatter(x=df.charges, y=df[df.columns[i]], mode='markers', name='Charges vs. ' + df.columns[i]))

buttons = []

for i in range(len(df.columns)):
    buttons.append(dict(method='update', label=df.columns[i], args=[{'visible': [True if j == i else False for j in rang
                                                                     {'title': 'Charges vs ' + df.columns[i]}]))

updatemenu = []
menu = {}
updatemenu.append(menu)

updatemenu[0]['buttons'] = buttons

fig.update_layout(updatemenus=updatemenu)
fig.update_layout(width=1200, height=500)

fig.show()
```



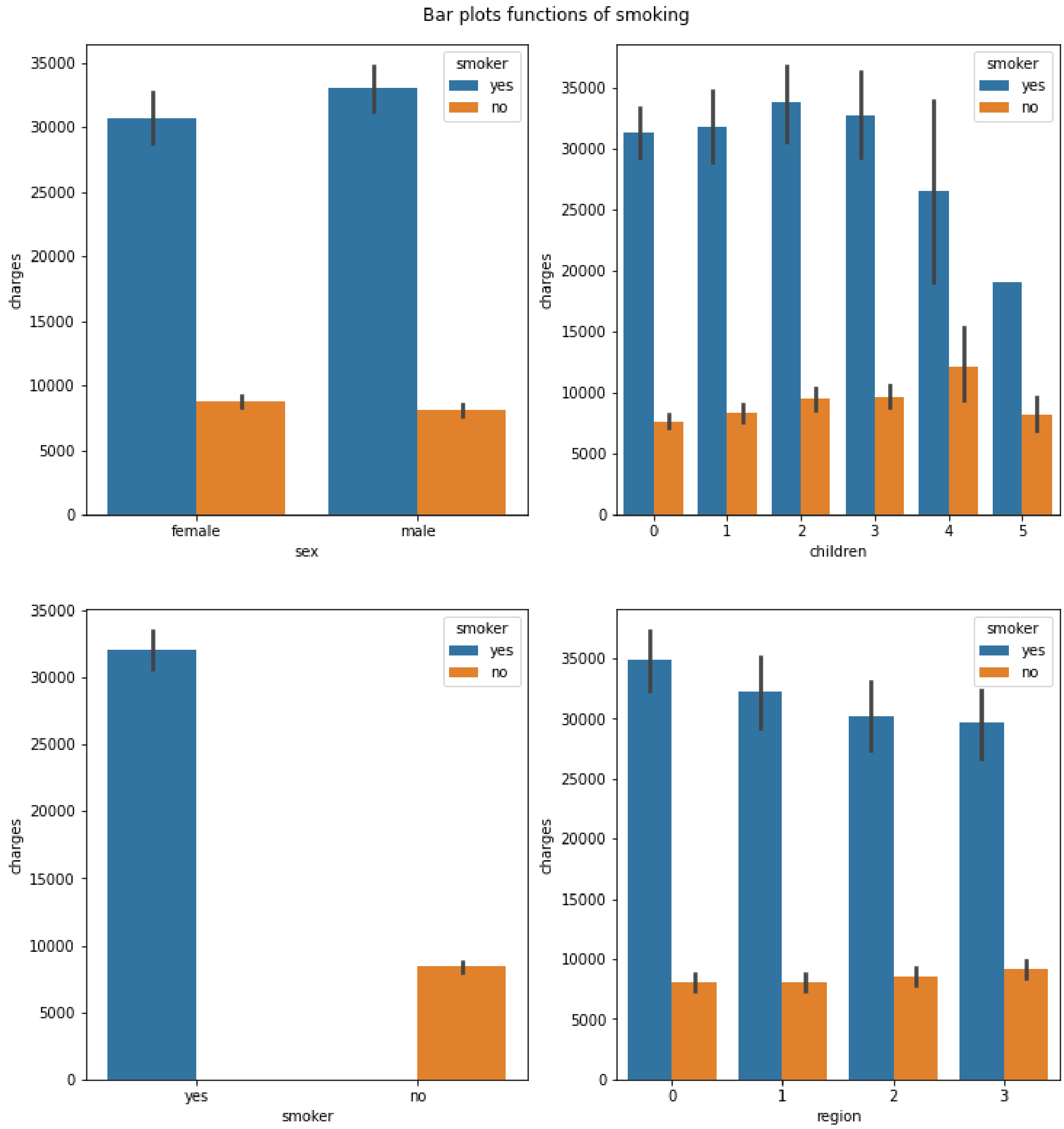
In [28]:

```
# Here we can see the importance of smoking feature
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15))

for i in range(len(df.columns)):
    sns.scatterplot(data=df, x = df.charges, y=df.columns[i],ax = axes[i // 3][i % 3], hue='smoker')

fig.suptitle("Features Distributions")

plt.subplots_adjust(top=0.95)

plt.show()
```



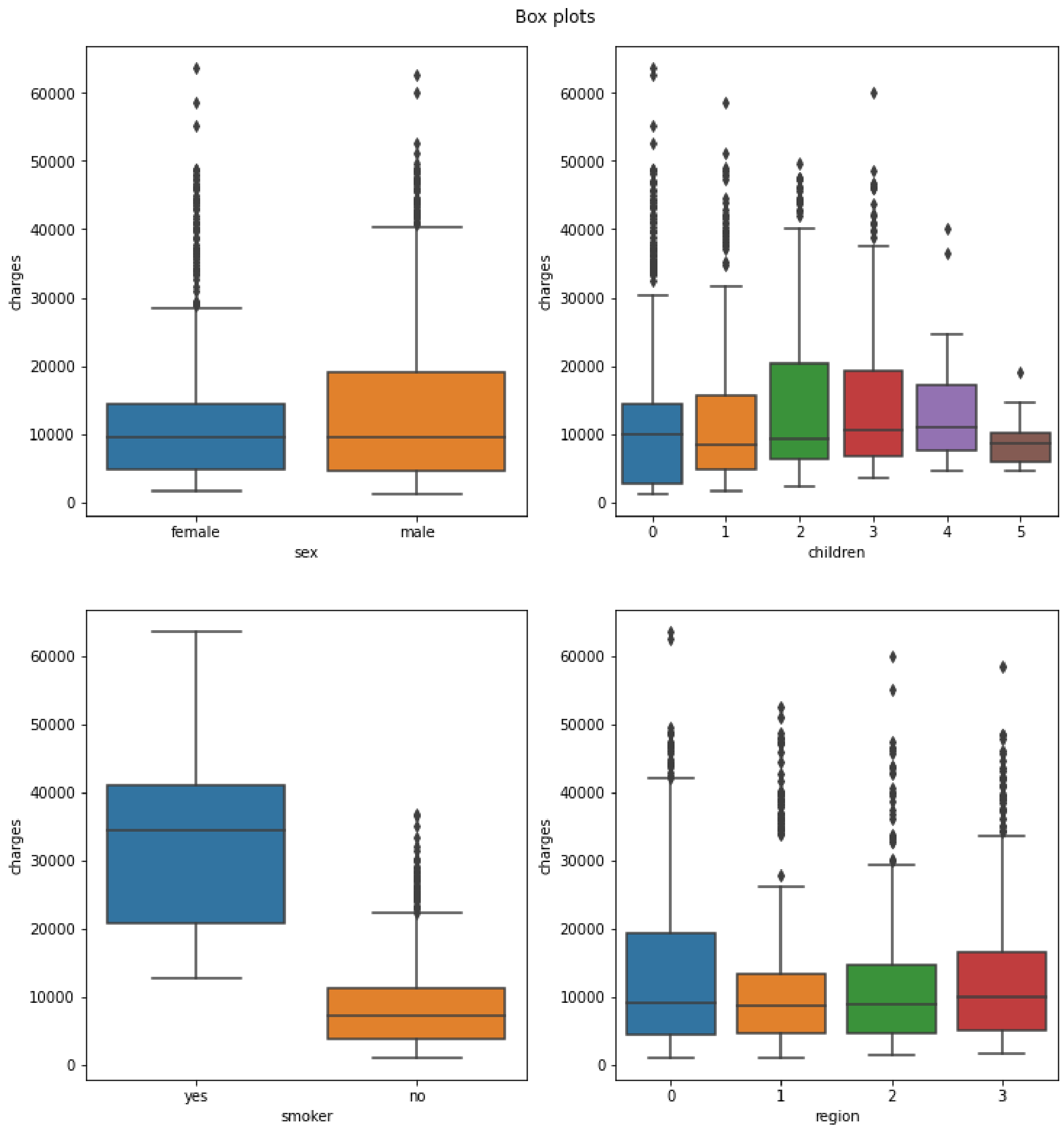
In [29]:

```
cols=['sex', 'children', 'smoker', 'region']
fig, axes = plt.subplots(nrows=2, ncols=2, figsize= (12, 12))

for i in range(len(cols)):
    sns.barplot(data=df, x = cols[i], y = 'charges', ax = axes[i // 2][i % 2], hue='smoker')

fig.suptitle("Bar plots functions of smoking")

plt.subplots_adjust(top=0.95)

plt.show()
```



In [30]:

```
cols=['sex', 'children', 'smoker', 'region']
fig, axes = plt.subplots(nrows=2, ncols=2, figsize= (12, 12))

for i in range(len(cols)):
    sns.boxplot(data=df, x = cols[i], y = 'charges', ax = axes[i // 2][i % 2])

fig.suptitle("Box plots")

plt.subplots_adjust(top=0.95)

plt.show()
```



In [31]:

```
df.corr
```

Out[31]:

```
<bound method DataFrame.corr of       age     sex     bmi  children smoker  region      charges     bmi_c
at
0      19  female  27.900         0    yes       1  16884.92400  overweight
1      18    male  33.770         1     no       0   1725.55230       obese
2      28    male  33.000         3     no       0   4449.46200       obese
3      33    male  22.705         0     no       2  21984.47061      normal
4      32    male  28.880         0     no       2   3866.85520  overweight
...   ...     ...     ...       ...    ...     ...          ...         ...
1333   50    male  30.970         3     no       2  10600.54830       obese
1334   18  female  31.920         0     no       3   2205.98080       obese
1335   18  female  36.850         0     no       0   1629.83350       obese
1336   21  female  25.800         0     no       1   2007.94500  overweight
1337   61  female  29.070         0    yes       2  29141.36030  overweight

[1338 rows x 8 columns]>
```

In [32]:

```
from sklearn.preprocessing import LabelEncoder
def labelling(x):
    df[x] = LabelEncoder().fit_transform(df[x])
    return df
cat = ['sex','smoker','bmi_cat']
for i in cat:
    labelling(i)
df
```

Out[32]:

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

1338 rows × 8 columns

Train-Test split

In [33]:

```
x = df.drop("bmi",axis=1)
y = df["bmi"]
```

In [34]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=23)
```

In [35]:

```
from sklearn.linear_model import LinearRegression
```

In [36]:

```
model = LinearRegression()
```

In [37]:

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

Out[37]:

```
LinearRegression()
```

In [38]:

```
model.intercept_
```

Out[38]:

```
30.972842843720212
```

In [39]:

```
model.coef_
```

Out[39]:

```
array([-3.17132357e-02,  3.02799086e-01, -9.35575339e-02, -7.38178067e+00,
       -1.41745867e+00,  3.07895827e-04,  5.73600193e-01])
```

In [40]:

```
train_predictions = model.predict(X_train)
```

In [41]:

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

In [42]:

```
from sklearn.metrics import mean_absolute_error
print("MAE for test data: ",mean_absolute_error(y_test,test_predictions))
print("MAE for train data: ",mean_absolute_error(y_train,train_predictions))
```

```
MAE for test data:  4.296922287205947
MAE for train data:  4.446706134683215
```

In [43]:

```
from sklearn.metrics import mean_absolute_error
print("MSE for test data: ",mean_absolute_error(y_test,test_predictions))
print("MSE for train data: ",mean_absolute_error(y_train,train_predictions))
```

```
MSE for test data:  4.296922287205947
MSE for train data:  4.446706134683215
```

In [44]:

```
from sklearn.metrics import mean_squared_error
print("RMSE for test data: ",np.sqrt(mean_squared_error(y_test,test_predictions)))
print("RMSE for train data: ",np.sqrt(mean_squared_error(y_train,train_predictions)))
```

```
RMSE for test data:  5.324991306012566
RMSE for train data:  5.647891089758658
```

In [45]:

```
from sklearn.metrics import r2_score
print("R2 for test data: ",r2_score(y_test,test_predictions))
print("R2 for train data: ",r2_score(y_train,train_predictions))
```

```
R2 for test data:  0.11300011110165908
R2 for train data:  0.18369254028695137
```

In [46]:

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

Out[46]:

```
0.11300011110165908
```

In [47]:

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

Out[47]:

```
0.18369254028695137
```

In [48]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,x,y,cv=5)
print(scores)
cv_score = scores.mean()
print("Cross Validation Score:",cv_score)
```

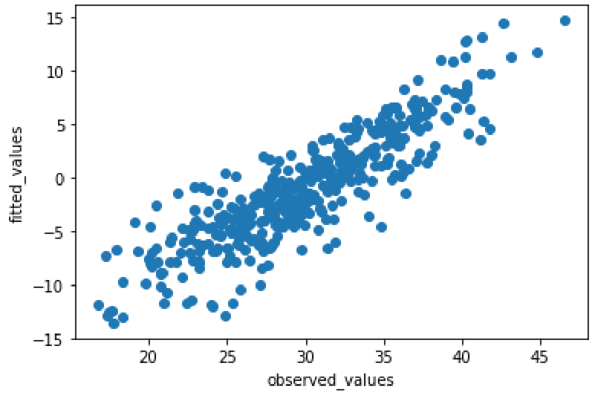
```
[0.1210946  0.11595634 0.14870749 0.20824497 0.16474452]
Cross Validation Score: 0.1517495857091608
```

In [49]:

```
test_res = y_test - test_predictions
```

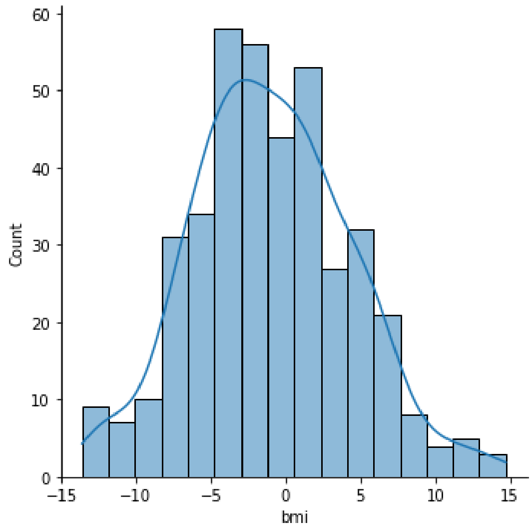
In [50]:

```
plt.scatter(y_test,test_res)
plt.xlabel("observed_values")
plt.ylabel("fitted_values")
plt.show()
```



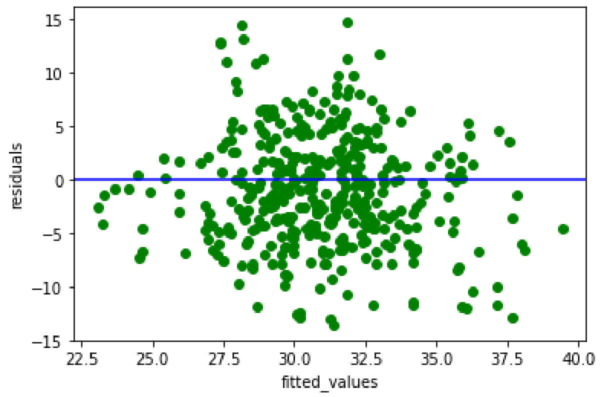
In [51]:

```
sns.displot(test_res,kde=True)
plt.show()
```



In [52]:

```
plt.scatter(test_predictions,test_res,c="g")
plt.axhline(y=0,color='blue')
plt.xlabel("fitted_values")
plt.ylabel("residuals")
plt.show()
```



In [53]:

```
import statsmodels.formula.api as smf
model2=smf.ols("y~x",data=df).fit()
model2.summary()
```

Out[53]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.174
Model:	OLS	**Adj. R-squared:**	0.170
Method:	Least Squares	**F-statistic:**	40.11
Date:	Wed, 19 Apr 2023	**Prob (F-statistic):**	2.19e-51
Time:	11:15:01	**Log-Likelihood:**	-4189.0
No. Observations:	1338	**AIC:**	8394.
Df Residuals:	1330	**BIC:**	8436.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	30.9613	0.593	52.247	0.000	29.799	32.124
x[0]	-0.0306	0.013	-2.420	0.016	-0.055	-0.006
x[1]	0.5459	0.305	1.791	0.074	-0.052	1.144
x[2]	-0.0978	0.127	-0.772	0.441	-0.346	0.151
x[3]	-6.9682	0.683	-10.199	0.000	-8.309	-5.628
x[4]	-1.3901	0.135	-10.328	0.000	-1.654	-1.126
x[5]	0.0003	2.39e-05	12.060	0.000	0.000	0.000
x[6]	0.3112	0.217	1.437	0.151	-0.114	0.736

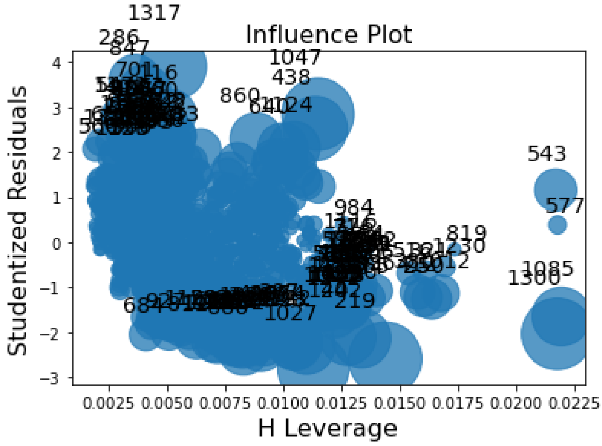
Omnibus:	18.091	**Durbin-Watson:**	2.081
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18.544
Skew:	0.270	**Prob(JB):**	9.40e-05
Kurtosis:	3.203	**Cond. No.**	8.33e+04

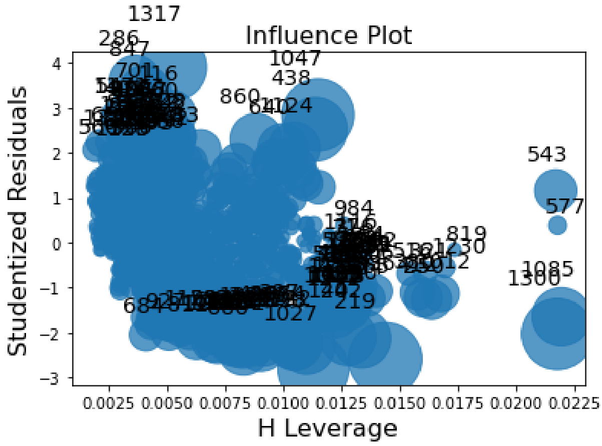
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.33e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [54]:

```
import statsmodels.api as sm
sm.graphics.influence_plot(model2)
```

Out[54]:





In [55]:

```
from joblib import dump
```

In [56]:

```
dump(model, 'bmi_model.joblib')
```

Out[56]:

```
['bmi_model.joblib']
```

In [57]:

```
from joblib import load
```

In [58]:

```
loaded_model = load('bmi_model.joblib')
```

In [59]:

```
from sklearn.linear_model import LinearRegression
```

In [60]:

```
model = LinearRegression()
```

In [61]:

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

Out[61]:

```
LinearRegression()
```

In [62]:

```
model.intercept_
```

Out[62]:

```
30.972842843720212
```

In [63]:

```
model.coef_
```

Out[63]:

```
array([-3.17132357e-02,  3.02799086e-01, -9.35575339e-02, -7.38178067e+00,
       -1.41745867e+00,  3.07895827e-04,  5.73600193e-01])
```

In [64]:

```
train_predictions = model.predict(X_train)
```

In [65]:

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

In [66]:

```
from sklearn.metrics import mean_squared_error
test_RMSE = np.sqrt(mean_squared_error(y_test,test_predictions))
train_RMSE = np.sqrt(mean_squared_error(y_train,train_predictions))
print(train_RMSE,test_RMSE)
```

```
5.647891089758658 5.324991306012566
```

In [67]:

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

Out[67]:

```
0.11300011110165908
```

In [68]:

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

Out[68]:

```
0.18369254028695137
```

In [69]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,x,y,cv=5)
print(scores)
cv_score = scores.mean()
print("Cross Validation Score:",cv_score)
```

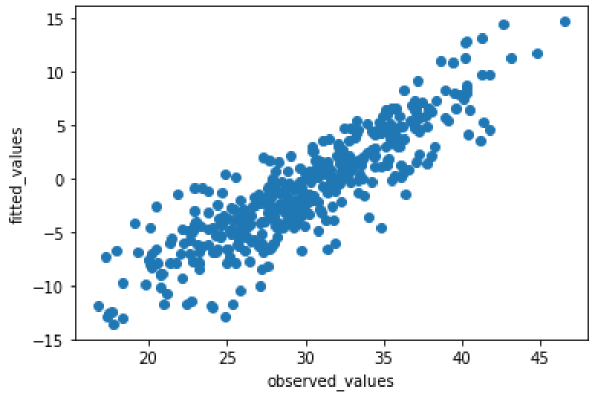
```
[0.1210946  0.11595634 0.14870749 0.20824497 0.16474452]
Cross Validation Score: 0.1517495857091608
```

In [70]:

```
test_res = y_test - test_predictions
```

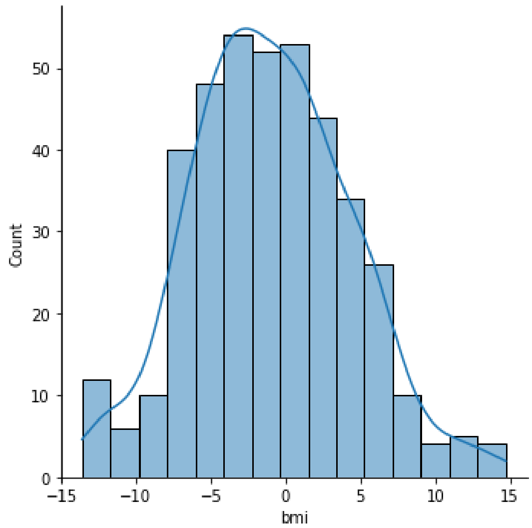
In [71]:

```
plt.scatter(y_test,test_res)
plt.xlabel("observed_values")
plt.ylabel("fitted_values")
plt.show()
```



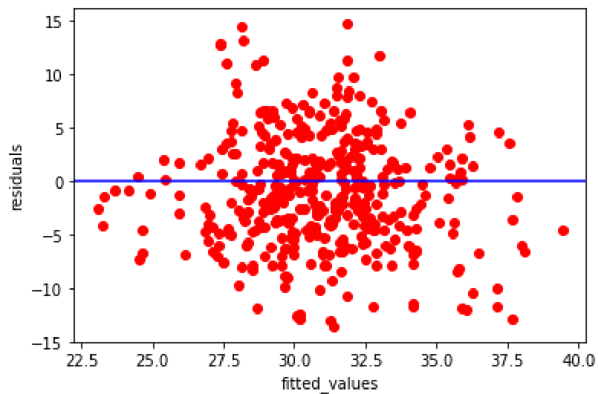
In [72]:

```
sns.displot(test_res,bins=15,kde=True)
plt.show()
```



In [73]:

```
plt.scatter(test_predictions,test_res,c="r")
plt.axhline(y=0,color='blue')
plt.xlabel("fitted_values")
plt.ylabel("residuals")
plt.show()
```



In [74]:

```
import statsmodels.formula.api as smf
model1=smf.ols("y~x",data=df).fit()
model1.summary()
```

Out[74]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.174
Model:	OLS	**Adj. R-squared:**	0.170
Method:	Least Squares	**F-statistic:**	40.11
Date:	Wed, 19 Apr 2023	**Prob (F-statistic):**	2.19e-51
Time:	11:15:04	**Log-Likelihood:**	-4189.0
No. Observations:	1338	**AIC:**	8394.
Df Residuals:	1330	**BIC:**	8436.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	30.9613	0.593	52.247	0.000	29.799	32.124
x[0]	-0.0306	0.013	-2.420	0.016	-0.055	-0.006
x[1]	0.5459	0.305	1.791	0.074	-0.052	1.144
x[2]	-0.0978	0.127	-0.772	0.441	-0.346	0.151
x[3]	-6.9682	0.683	-10.199	0.000	-8.309	-5.628
x[4]	-1.3901	0.135	-10.328	0.000	-1.654	-1.126
x[5]	0.0003	2.39e-05	12.060	0.000	0.000	0.000
x[6]	0.3112	0.217	1.437	0.151	-0.114	0.736

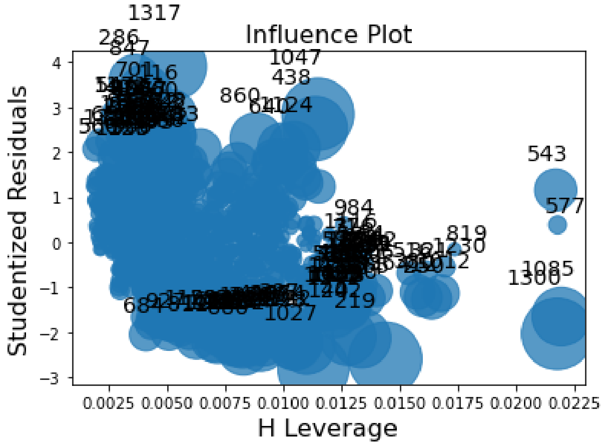
Omnibus:	18.091	**Durbin-Watson:**	2.081
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	18.544
Skew:	0.270	**Prob(JB):**	9.40e-05
Kurtosis:	3.203	**Cond. No.**	8.33e+04

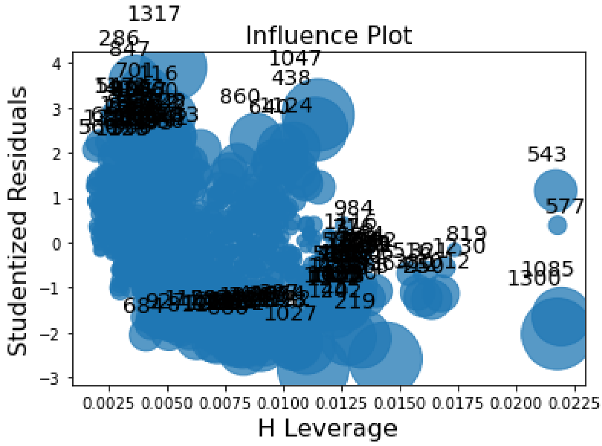
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.33e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [75]:

```
import statsmodels.api as sm
sm.graphics.influence_plot(model1)
```

Out[75]:





In [76]:

```
df.iloc[130]
```

Out[76]:

```
age            59.00000
sex             0.00000
bmi            26.50500
children        0.00000
smoker          0.00000
region          3.00000
charges     12815.44495
bmi_cat         2.00000
Name: 130, dtype: float64
```

In [77]:

```
df_new=df.drop(df.index[[130]],axis=0)
df_new
```

Out[77]:

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

1337 rows × 8 columns

In [78]:

```
lm = smf.ols(formula='age~sex + bmi + children + smoker + region + charges',data=df_new).fit()
lm.summary()
```

Out[78]:

OLS Regression Results

Dep. Variable:	age	**R-squared:**	0.271
Model:	OLS	**Adj. R-squared:**	0.268
Method:	Least Squares	**F-statistic:**	82.42
Date:	Wed, 19 Apr 2023	**Prob (F-statistic):**	8.11e-88
Time:	11:15:06	**Log-Likelihood:**	-5218.0
No. Observations:	1337	**AIC:**	1.045e+04
Df Residuals:	1330	**BIC:**	1.049e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	35.4968	1.965	18.062	0.000	31.641	39.352
sex	-0.3664	0.660	-0.555	0.579	-1.662	0.929
bmi	-0.1391	0.059	-2.352	0.019	-0.255	-0.023
children	-0.1113	0.274	-0.406	0.685	-0.649	0.427
smoker	-24.6507	1.377	-17.908	0.000	-27.351	-21.950
region	-0.0492	0.303	-0.163	0.871	-0.643	0.544
charges	0.0010	4.68e-05	21.562	0.000	0.001	0.001

Omnibus:	34.910	**Durbin-Watson:**	1.986
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	24.309
Skew:	-0.216	**Prob(JB):**	5.26e-06
Kurtosis:	2.500	**Cond. No.**	1.09e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.09e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [79]:

```
rsq_sex = smf.ols('sex~bmi+children+smoker+region+charges',data=df).fit()
rsq_sex.summary()
```

Out[79]:

OLS Regression Results

Dep. Variable:	sex	**R-squared:**	0.009
Model:	OLS	**Adj. R-squared:**	0.005
Method:	Least Squares	**F-statistic:**	2.330
Date:	Wed, 19 Apr 2023	**Prob (F-statistic):**	0.0405
Time:	11:15:06	**Log-Likelihood:**	-965.21
No. Observations:	1338	**AIC:**	1942.
Df Residuals:	1332	**BIC:**	1974.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.3517	0.081	4.341	0.000	0.193	0.511
bmi	0.0045	0.002	1.817	0.070	-0.000	0.009
children	0.0076	0.011	0.665	0.506	-0.015	0.030
smoker	0.1311	0.057	2.299	0.022	0.019	0.243
region	0.0018	0.013	0.143	0.886	-0.023	0.026
charges	-1.563e-06	1.94e-06	-0.805	0.421	-5.37e-06	2.25e-06

Omnibus:	5015.796	**Durbin-Watson:**	2.010
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	215.406
Skew:	-0.019	**Prob(JB):**	1.68e-47
Kurtosis:	1.035	**Cond. No.**	1.08e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.08e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [80]:

```
rsq_bmi = smf.ols('bmi~children+smoker+region+charges',data=df).fit().rsquared
vif_bmi = 1/(1-rsq_bmi)
```

In [81]:

```
rsq_children = smf.ols('children~bmi+smoker+region+charges',data=df).fit().rsquared
vif_children = 1/(1-rsq_children)
```

In [82]:

```
rsq_smoker = smf.ols('smoker~bmi+children+region+charges',data=df).fit().rsquared
vif_smoker = 1/(1-rsq_smoker)
```

In [83]:

```
rsq_region = smf.ols('region~smoker+bmi+children+charges',data=df).fit().rsquared
vif_region = 1/(1-rsq_region)
```

In [84]:

```
rsq_charges = smf.ols('charges~region+smoker+bmi+children',data=df).fit().rsquared
vif_charges = 1/(1-rsq_charges)
```

In [85]:

```
d1 = {'Variables':['bmi','children','smoker','region','charges'],'VIF':[vif_bmi,vif_children,vif_smoker,vif_region,vif_c
vif_frame = pd.DataFrame(d1)
vif_frame
```

Out[85]:

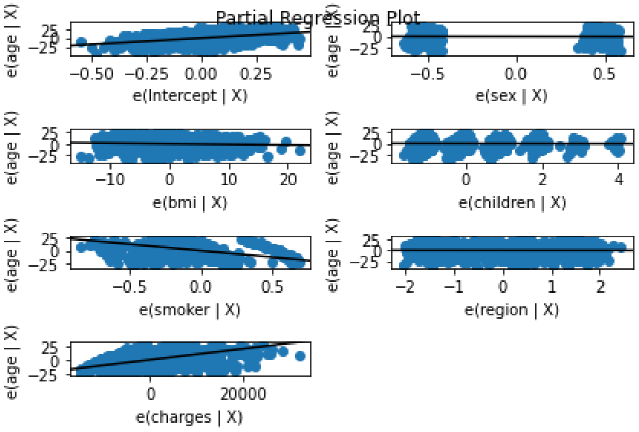
	Variables	VIF
0	bmi	1.201145
1	children	1.010680
2	smoker	2.845317
3	region	1.081956
4	charges	2.970807

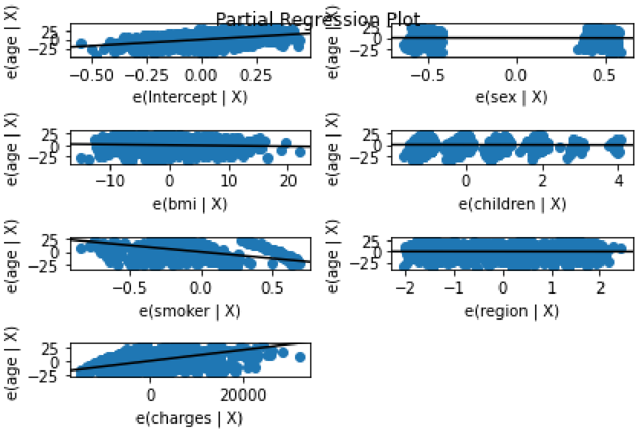
In [86]:

```
sm.graphics.plot_partregress_grid(lm)
```

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```

Out[86]:





In [87]:

```
final_model = smf.ols(formula='charges ~ children + smoker',data=df).fit()
final_model.summary()
```

Out[87]:

OLS Regression Results

Dep. Variable:	charges	**R-squared:**	0.624
Model:	OLS	**Adj. R-squared:**	0.623
Method:	Least Squares	**F-statistic:**	1106.
Date:	Wed, 19 Apr 2023	**Prob (F-statistic):**	5.54e-284
Time:	11:15:08	**Log-Likelihood:**	-13824.
No. Observations:	1338	**AIC:**	2.765e+04
Df Residuals:	1335	**BIC:**	2.767e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7755.6647	292.878	26.481	0.000	7181.114	8330.216
children	622.4433	168.684	3.690	0.000	291.528	953.358
smoker	2.36e+04	503.717	46.855	0.000	2.26e+04	2.46e+04

Omnibus:	139.785	**Durbin-Watson:**	2.029
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	219.354
Skew:	0.741	**Prob(JB):**	2.33e-48
Kurtosis:	4.317	**Cond. No.**	4.58

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [88]:

```
from sklearn.linear_model import LinearRegression
```

In [89]:

```
model = LinearRegression()
```

In [90]:

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

Out[90]:

```
LinearRegression()
```

In [91]:

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

In [92]:

```
model.intercept_
```

Out[92]:

```
30.972842843720212
```

In [93]:

```
model.coef_
```

Out[93]:

```
array([-3.17132357e-02,  3.02799086e-01, -9.35575339e-02, -7.38178067e+00,
       -1.41745867e+00,  3.07895827e-04,  5.73600193e-01])
```

In [94]:

```
test_predictions = model.predict(X_test)
train_predictions = model.predict(X_train)
```

In [95]:

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

Out[95]:

```
0.11300011110165908
```

In [96]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model,x,y,cv=5)
print(scores)
cv_score = scores.mean()
print("Cross Validation Score:",cv_score)
```

```
[0.1210946  0.11595634 0.14870749 0.20824497 0.16474452]
Cross Validation Score: 0.1517495857091608
```

In [97]:

```
from sklearn.metrics import mean_squared_error
test_RMSE = np.sqrt(mean_squared_error(y_test,test_predictions))
train_RMSE = np.sqrt(mean_squared_error(y_train,train_predictions))
print(train_RMSE,test_RMSE)
```

```
5.647891089758658 5.324991306012566
```

In [98]:

```
from sklearn.preprocessing import PolynomialFeatures
polynomial_converter=PolynomialFeatures(degree=2,include_bias=False)
X_poly=polynomial_converter.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.3,random_state=23)
model=LinearRegression()
model.fit(X_train,y_train)
train_pred=model.predict(X_train)
test_pred=model.predict(X_test)
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.6978876884460314
0.7093223892522964
```

In [99]:

```
train_rmse_errors=[]
test_rmse_errors=[]
for d in range(1,10):
    polynomial_converter=PolynomialFeatures(degree=d,include_bias=False)
    X_poly=polynomial_converter.fit_transform(x)
    X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.3,random_state=23)
    model=LinearRegression()
    model.fit(X_train,y_train)
    train_pred=model.predict(X_train)
    test_pred=model.predict(X_test)
    train_rmse=np.sqrt(mean_squared_error(y_train,train_pred))
    train_rmse_errors.append(train_rmse)
    test_rmse=np.sqrt(mean_squared_error(y_test,test_pred))
    test_rmse_errors.append(test_rmse)
```

In [100]:

```
train_rmse_errors
```

Out[100]:

```
[5.647891089758658,
 3.43592474170093,
 5.4449752867434835,
 3.826720382415379,
 4.717947507708819,
 5.735857780002097,
 5.424114727941829,
 5.678029943107806,
 7.498024402975816]
```

In [101]:

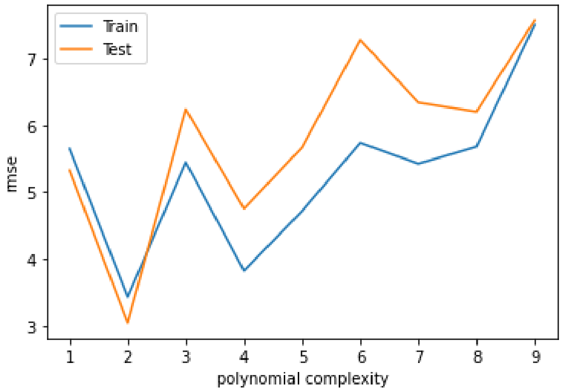
```
test_rmse_errors
```

Out[101]:

```
[5.324991306012565,
 3.048336535036782,
 6.236691028913041,
 4.752871721997629,
 5.668354657635049,
 7.272698960660427,
 6.337946622269486,
 6.200308644187235,
 7.563679066395718]
```

In [102]:

```
plt.plot(range(1,10),train_rmse_errors,label='Train')
plt.plot(range(1,10),test_rmse_errors,label='Test')
plt.xlabel("polynomial complexity")
plt.ylabel("rmse")
plt.legend()
plt.show()
```



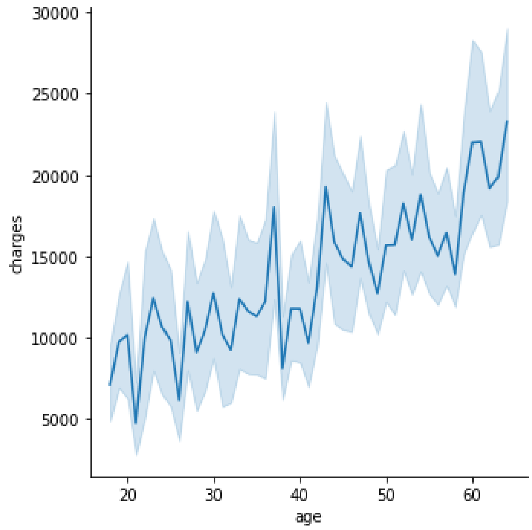
In [103]:

```
final_poly_converter=PolynomialFeatures(degree=2,include_bias=False)
X_poly=final_poly_converter.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.3,random_state=23)
final_model=LinearRegression()
final_model.fit(X_train,y_train)
train_pred=final_model.predict(X_train)
test_pred=final_model.predict(X_test)
print("train r2:",final_model.score(X_train,y_train))
print("test r2:",final_model.score(X_test,y_test))
```

```
train r2: 0.6978876884460314
test r2: 0.7093223892522964
```

In [104]:

```
sns.relplot(x = 'age', y ='charges', kind = 'line', data = df)
plt.show()
```



In [106]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

In [108]:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(X_train,y_train)
```

Out[108]:

```
RandomForestRegressor()
```