In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
df=pd.read_csv(r"C:\Users\HP\Downloads\insurance.csv")
df.head()
```

Out[1]:

	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 [2]:
```
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: 57.6+ KB
```

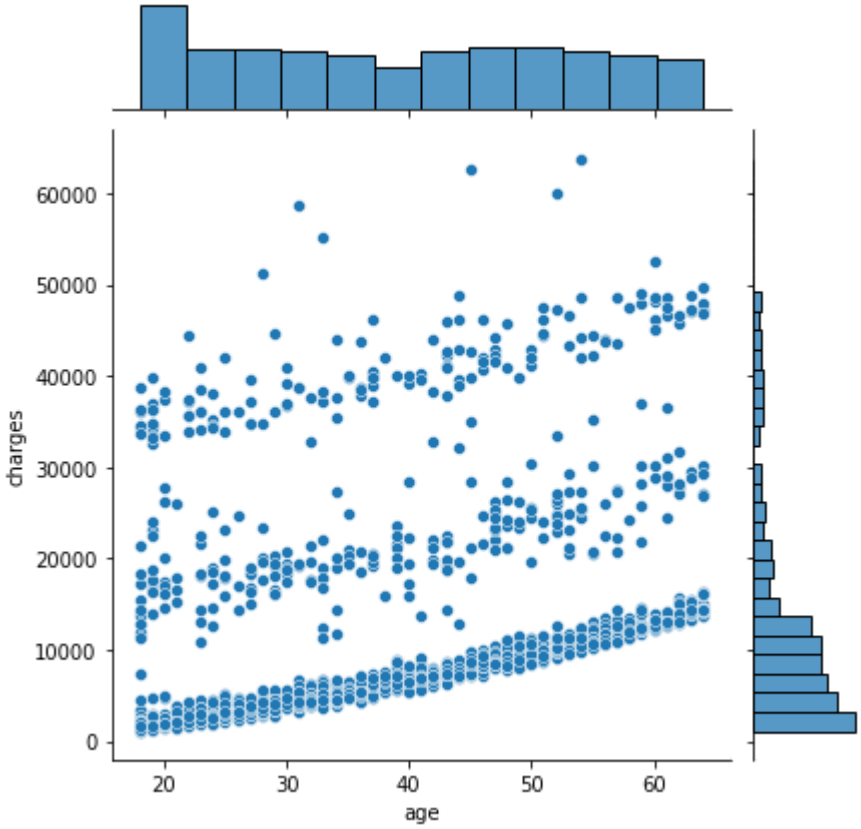
In [3]:
```
df.describe()
```

Out[3]:

	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

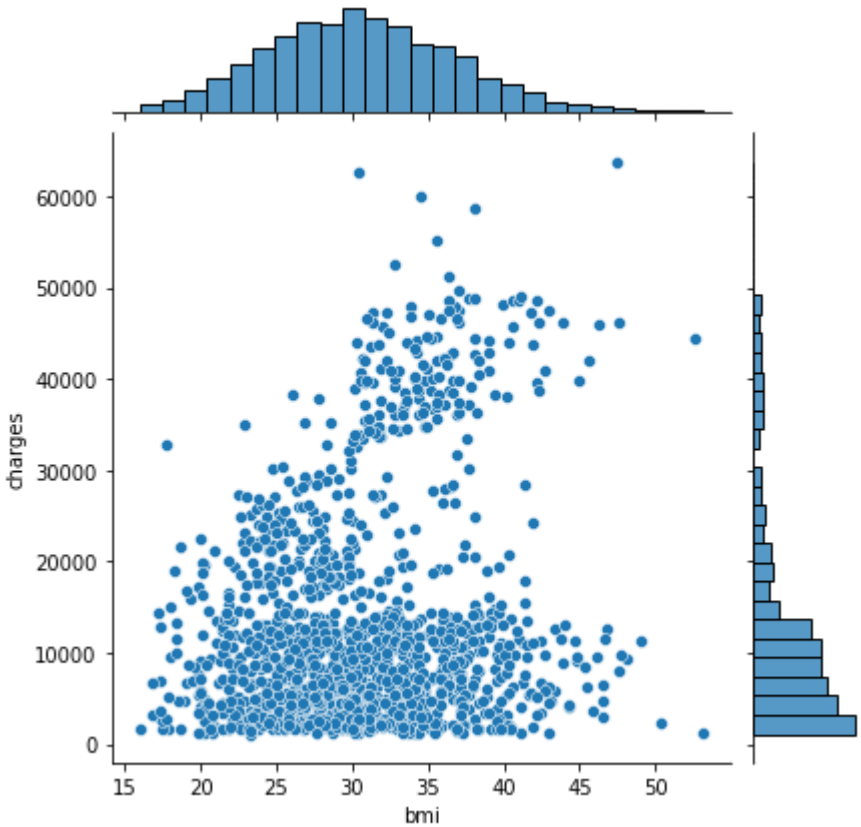
In [4]:
```
sns.jointplot(x='age', y='charges',data=df)
```

Out[4]:
```
<seaborn.axisgrid.JointGrid at 0x1ac541f0>
```



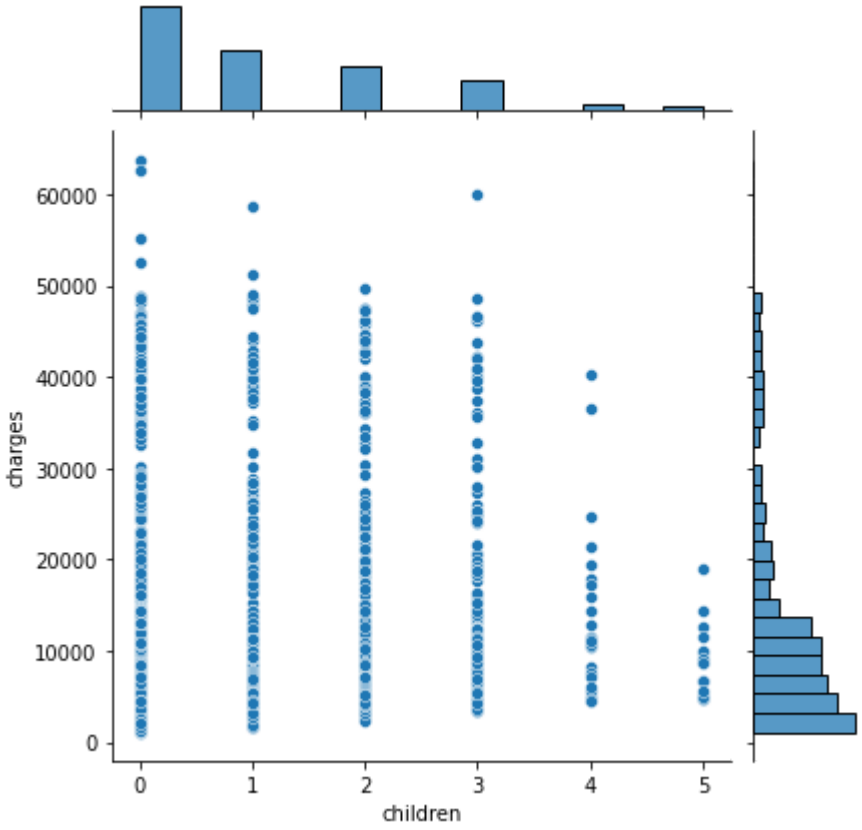
In [5]:
```
sns.jointplot(x='bmi', y='charges',data=df)
```

Out[5]: <seaborn.axisgrid.JointGrid at 0x106df958>



In [6]:
```
sns.jointplot(x='children', y='charges',data=df)
```

Out[6]: <seaborn.axisgrid.JointGrid at 0x1070ea18>



In [7]:
```
df['sex'] = pd.get_dummies(df['sex'])
```

In [8]:
```
df['smoker'] = pd.get_dummies(df['smoker'])
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['region']= label_encoder.fit_transform(df['region'])
```

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

Out[10]:

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

In [17]:
```
x = df.drop('charges', axis=1)
y = df['charges']
```

In [12]:
```
import statsmodels.api as sm
```

In [14]:
```
x_cons = sm.add_constant(x)
```

In [15]:
```
x_cons.head()
```

Out[15]:

	const	age	sex	bmi	children	smoker	region
0	1.0	19	1	27.900	0	0	3
1	1.0	18	0	33.770	1	1	2
2	1.0	28	0	33.000	3	1	2
3	1.0	33	0	22.705	0	1	1
4	1.0	32	0	28.880	0	1	1

In [18]:
```
x_sm = sm.OLS(y,x_cons).fit()
```

In [19]:
```
x_sm.summary()
```

Out[19]:

OLS Regression Results

Dep. Variable:	charges	**R-squared:**	0.751
Model:	OLS	**Adj. R-squared:**	0.750
Method:	Least Squares	**F-statistic:**	668.1
Date:	Thu, 19 Nov 2020	**Prob (F-statistic):**	0.00
Time:	21:04:01	**Log-Likelihood:**	-13548.
No. Observations:	1338	**AIC:**	2.711e+04
Df Residuals:	1331	**BIC:**	2.715e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.187e+04	1009.703	11.760	0.000	9893.088	1.39e+04
age	257.2881	11.886	21.647	0.000	233.971	280.605
sex	131.1106	332.811	0.394	0.694	-521.780	784.001
bmi	332.5701	27.722	11.997	0.000	278.186	386.954
children	479.3694	137.644	3.483	0.001	209.346	749.393
smoker	-2.382e+04	411.843	-57.839	0.000	-2.46e+04	-2.3e+04
region	-353.6400	151.927	-2.328	0.020	-651.682	-55.598

Omnibus:	299.003	**Durbin-Watson:**	2.088
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	713.975
Skew:	1.207	**Prob(JB):**	9.17e-156
Kurtosis:	5.642	**Cond. No.**	316.

Notes:

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

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [21]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0
```

In [22]:
```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(1070, 6) (268, 6) (1070,) (268,)
```

In [23]:
```
from sklearn.linear_model import LinearRegression
```

In [24]:
```
lr = LinearRegression().fit(x,y)
```

In [25]:
```
y_test_pred = lr.predict(x_test)
```

In [26]:
```
y_train_pred = lr.predict(x_train)
```

In [27]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

In [28]:
```
r2_score(y_test, y_test_pred)
```

Out[28]:
```
0.8012066600423946
```

In [29]:
```
r2_score(y_train, y_train_pred)
```

Out[29]:
```
0.7366647054302722
```

In [30]:
```
from sklearn.linear_model import Lasso
```

In [31]:
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(x_train)
```

In [32]:
```
x_train_s = scaler.transform(x_train)
x_test_s = scaler.transform(x_test)
```

In [33]:
```
lr_las = Lasso(alpha = 0.4)
```

In [34]:
```
lr_las.fit(x_train_s, y_train)
```

Out[34]:
```
Lasso(alpha=0.4)
```

In [35]:
```
r2_score(y_test, lr_las.predict(x_test_s))
```

Out[35]:
```
0.7998662953803359
```

In [36]:
```
from sklearn.linear_model import SGDRegressor
```

In [37]:
```
lm_sgd = SGDRegressor()
lm_sgd
```

Out[37]:
```
SGDRegressor()
```

In [38]:
```
lm_sgd.fit(x_train_s, y_train)
```

Out[38]:
```
SGDRegressor()
```

In [39]:
```
lm_sgd.coef_
```

Out[39]:
```
array([ 3626.75329112,    44.46570878,  2039.67251463,   533.51336301,
       -9521.98921599,  -277.5422736 ])
```

In [40]:
```
r2_score(y_test, lm_sgd.predict(x_test_s))
```

Out[40]:
```
0.8000146853583883
```

In [41]:
```
from sklearn import tree
```

In [42]:
```
regtree = tree.DecisionTreeRegressor()
```

In [43]:
```
regtree.fit(x_train, y_train)
```

Out[43]:
```
DecisionTreeRegressor()
```

In [44]:
```
y_train_pred = regtree.predict(x_train)
```

In [45]:
```
y_test_pred = regtree.predict(x_test)
```

In [46]:
```
mean_squared_error(y_test, y_test_pred)
```

Out[46]:
```
51553290.25653334
```

In [47]:
```
r2_score(y_test, y_test_pred)
```

Out[47]:
```
0.6760302742221833
```

In [48]:
```
r2_score(y_train, y_train_pred)
```

Out[48]:
```
0.9982963931606104
```

In [49]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [50]:
```
regtree = tree.DecisionTreeRegressor()
```

In [51]:
```
rf_reg = RandomForestRegressor(n_estimators=50,n_jobs=-1 ,random_state=42)
```

In [52]:
```
rf_reg.fit(x_train, y_train)
```

Out[52]:
```
RandomForestRegressor(n_estimators=50, n_jobs=-1, random_state=42)
```

In [53]:
```
mean_squared_error(y_test, rf_reg.predict(x_test))
```

Out[53]:
```
20008345.623353824
```

In [54]:
```
r2_score(y_train, rf_reg.predict(x_train))
```

Out[54]: 0.9730862438086345

In [55]:
```
r2_score(y_test, rf_reg.predict(x_test))
```

Out[55]: 0.8742641214050496

In [56]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [57]:
```
gbc_reg = GradientBoostingRegressor()
```

In [58]:
```
gbc_reg.fit(x_train, y_train)
```

Out[58]: GradientBoostingRegressor()

In [59]:
```
r2_score(y_test, gbc_reg.predict(x_test))
```

Out[59]: 0.89815294289773

In [60]:
```
r2_score(y_train, gbc_reg.predict(x_train))
```

Out[60]: 0.897352642499688

In [61]:
```
from sklearn.ensemble import AdaBoostRegressor
```

In [62]:
```
ada_reg = AdaBoostRegressor(learning_rate = 0.02, n_estimators = 5000)
```

In [63]:
```
ada_reg.fit(x_train, y_train)
```

Out[63]: AdaBoostRegressor(learning_rate=0.02, n_estimators=5000)

In [64]:
```
r2_score(y_train, ada_reg.predict(x_train))
```

Out[64]: 0.8333118910804627

In [65]:
```
r2_score(y_test, ada_reg.predict(x_test))
```

Out[65]: 0.869906245310401

In [ ]:
```

```