In [95]:
```
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_fact
or
```

Load data ¶

In [68]:
```
df = pd.read_csv("insurance.csv")
df.columns=['age','sex','bmi','children','smoker','region','spending']
X=df[['bmi','children','age']]
Y_true=df['spending']
```

Already split up only the number data into X values and Y_true values. df.describe only takes the numeric values.

In [69]:
```
df.describe()
```

Out[69]:

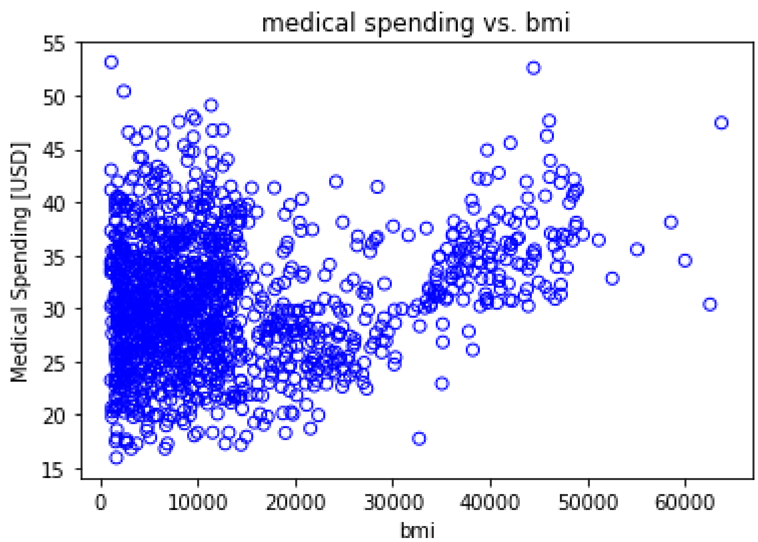
	age	bmi	children	spending
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

You will need to create dummy variables in your homework by yourself.

Plotting - simple

In [70]:
```
plt.scatter(df['spending'],df['bmi'],marker="o",facecolors='none', edgec
olors='b')
plt.title('medical spending vs. bmi')
plt.ylabel('Medical Spending [USD]')
plt.xlabel('bmi')
```

Out[70]: Text(0.5,0,'bmi')



Check for Multicollinearity

In [73]:
```
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in ran
ge(X.shape[1])]
vif["features"] = X.columns
vif.round(1)
```

Out[73]:

	VIF Factor	features
0	7.8	bmi
1	1.8	children
2	7.5	age

Learn more about Variance Inflation Factors here. [https://en.wikipedia.org/wiki/Variance_inflation_factor (https://en.wikipedia.org/wiki/Variance_inflation_factor)](https://en.wikipedia.org/wiki/Variance_inflation_factor). A cutoff from 10 is ok.

using Statistics model

In [94]:
```
X2 = sm.add_constant(X) # to force the linear model
model = sm.OLS(Y_true, X2)
model2 = model.fit()
print(model2.summary())
```

```
                            OLS Regression Results
========================================================================
=======
Dep. Variable:               spending   R-squared:
0.120
Model:                            OLS   Adj. R-squared:
0.118
Method:                 Least Squares   F-statistic:
60.69
Date:                Tue, 25 Aug 2020   Prob (F-statistic):
8.80e-37
Time:                        13:56:22   Log-Likelihood:
-14392.
No. Observations:                1338   AIC:                          2.
879e+04
Df Residuals:                    1334   BIC:                          2.
881e+04
Df Model:                           3
Covariance Type:            nonrobust
========================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------
-------
const      -6916.2433   1757.480     -3.935      0.000   -1.04e+04    -3
468.518
bmi          332.0834     51.310      6.472      0.000     231.425
432.741
children     542.8647    258.241      2.102      0.036      36.261     1
049.468
age          239.9945     22.289     10.767      0.000     196.269
283.720
========================================================================
=======
Omnibus:                      325.395   Durbin-Watson:
2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
603.372
Skew:                           1.520   Prob(JB):                     9.
54e-132
Kurtosis:                       4.255   Cond. No.
290.
========================================================================
=======

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

Rebuilding it by 'hand'

OLS is only one of many ways... here is another. Using the standard linear regression model.

In [99]:
```
regr = linear_model.LinearRegression()
regr.fit(X,Y_true)
```

Out[99]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=F
alse)
```

The coefficient of the model are:

In [100]:
```
regr.coef_
#note the way we created the matrix: X=df[['bmi','children','age']]
```

Out[100]:
```
array([332.0833645 , 542.86465225, 239.99447429])
```

To do the actual calculation we have now to create the predictions Y_pred and compare them with the true data Y_true.

In [101]:
```
Y_pred=regr.predict(X)
```

In [86]:
```
explained_variance=metrics.explained_variance_score(Y_true, Y_pred)
mean_absolute_error=metrics.mean_absolute_error(Y_true, Y_pred)
mse=metrics.mean_squared_error(Y_true, Y_pred)
mean_squared_log_error=metrics.mean_squared_log_error(Y_true, Y_pred)
median_absolute_error=metrics.median_absolute_error(Y_true, Y_pred)
r2=metrics.r2_score(Y_true, Y_pred)
print('explained_variance: ', round(explained_variance,4))
print('mean_squared_log_error: ', round(mean_squared_log_error,4))
print('r2: ', round(r2,4))
print('MAE: ', round(mean_absolute_error,4))
print('MSE: ', round(mse,4))
print('RMSE: ', round(np.sqrt(mse),4))
```

```
explained_variance:  0.1201
mean_squared_log_error:  0.747
r2:  0.1201
MAE:  9015.4422
MSE:  128943244.6356
RMSE:  11355.3179
```