In [1]:
```
import pandas as pd
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
import numpy as np

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("Multivariate Linear Regression Dataset.csv")
```

In [3]:
```
print (df.head())
```

```
   Squar_Feet  Number_of_Bed_Rooms  Price_of_House
0        2104                    3          399900
1        1600                    3          329900
2        2400                    3          369000
3        1416                    2          232000
4        3000                    4          539900
```

In [4]:
```
Y = df["Price_of_House"]
```

In [5]:
```
X = df[['Squar_Feet','Number_of_Bed_Rooms']]
```

In [6]:
```
scale = StandardScaler()
```

In [7]:
```
X_scaled = scale.fit_transform(X[['Squar_Feet','Number_of_Bed_Rooms']].as_matrix())
```

```
C:\Users\Aditya Bhalsod\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Fut
ureWarning: Method .as_matrix will be removed in a future version. Use .values i
nstead.
  """Entry point for launching an IPython kernel.
C:\Users\Aditya Bhalsod\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
595: DataConversionWarning: Data with input dtype int64 was converted to float64
by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\Aditya Bhalsod\Anaconda3\lib\site-packages\sklearn\utils\validation.py:
595: DataConversionWarning: Data with input dtype int64 was converted to float64
by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [8]:
```
est = sm.OLS(Y, X).fit()
```

In [9]:
```
print (est.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         Price_of_House   R-squared:                       0.966
Model:                            OLS   Adj. R-squared:                  0.964
Method:                 Least Squares   F-statistic:                     631.4
Date:                Wed, 31 Jul 2019   Prob (F-statistic):           1.19e-33
Time:                        11:36:36   Log-Likelihood:                -589.11
No. Observations:                  47   AIC:                             1182.
Df Residuals:                      45   BIC:                             1186.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=======
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Squar_Feet             140.8611     15.355      9.174      0.000     109.935
171.788
Number_of_Bed_Rooms  1.698e+04   1.01e+04      1.676      0.101   -3424.632
3.74e+04
==============================================================================
Omnibus:                        2.046   Durbin-Watson:                   1.923
Prob(Omnibus):                  0.359   Jarque-Bera (JB):                1.215
Skew:                           0.354   Prob(JB):                        0.545
Kurtosis:                       3.346   Cond. No.                     2.17e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.17e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [10]:
```
fig = plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

In [11]:
```
ax = fig.add_subplot(1,2,1, projection='3d')
```

In [12]:
```
ax.scatter(df["Squar_Feet"], df["Number_of_Bed_Rooms"], df["Price_of_House"], c='r
', marker='^')
```

Out[12]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x20dce32ee48>
```

In [13]:
```
ax.set_xlabel('Squar_Feet')
ax.set_ylabel('Number_of_Bed_Rooms')
ax.set_zlabel('Price_of_House')
```

Out[13]:
```
Text(0.5, 0, 'Price_of_House')
```

In [14]:
```
frst_col_surface =  df.iloc[0:len(df),0] #selection de la première colonne de notre
dataset
scnd_col_nb_chambre =  df.iloc[0:len(df),1]
third_col_prix = df.iloc[0:len(df),2]
```

In [15]:
```
def predict_price_of_house(Squar_Feet,Number_of_Bed_Rooms):
    return 140.8611 * Squar_Feet + 1.698e+04 * Number_of_Bed_Rooms # not scaled
    #return 1.094e+05 * taille_maison + (6578.3549 * nb_chambre) # scaled
```

In [16]:
```
def predict_all(lst_sizes, lst_nb_chmbres):
    predicted_prices = []
    for n in range(0, len(Y)):
        predicted_prices.append(predict_price_of_house(lst_sizes[n], lst_nb_chmbres
[n]))
    return predicted_prices
```

In [17]:
```
ax = fig.add_subplot(1, 2, 2, projection='3d')

ax.plot_trisurf(df["Squar_Feet"], df["Number_of_Bed_Rooms"], predict_all(df["Squar_
Feet"], df["Number_of_Bed_Rooms"]))

plt.show()
```

In [18]:
```
print (predict_price_of_house(4500,5))
```

```
718774.95
```