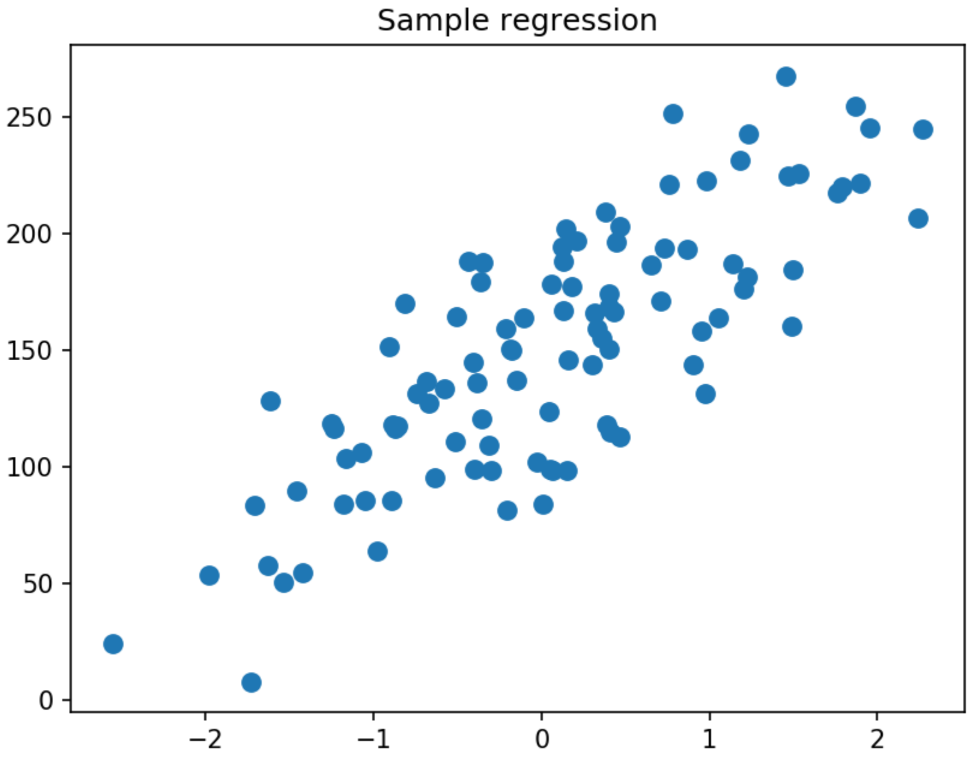
In [2]:

```
%matplotlib notebook
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

np.set_printoptions(precision=2)

from sklearn.datasets import make_regression
plt.figure()
plt.title('Sample regression')
X_R1, y_R1=make_regression(n_samples=100,n_features=1,
                           n_informative=1,bias=150.0,
                           noise=30,random_state=0)
plt.scatter(X_R1,y_R1,marker='o',s=50)
plt.show()
```



In [4]:

```
#for classification
fruits=pd.read_table('fruit_data_with_colors.txt')

X_fruits=fruits[['height','width','mass','color_score']]
y_fruit=fruits['fruit_label']

X_fruits_2d=fruits[['height','width']]
y_fruits_2d=fruits['fruit_label']
#for evaluation
from sklearn.datasets import load_breast_cancer
cancer=load_breast_cancer()
(X_cancer,y_cancer)=load_breast_cancer(return_X_y=True)
```

In [5]:

```
X_fruits.head()
```

Out[5]:

	height	width	mass	color_score
0	7.3	8.4	192	0.55
1	6.8	8.0	180	0.59
2	7.2	7.4	176	0.60
3	4.7	6.2	86	0.80
4	4.6	6.0	84	0.79

In [6]:

```
X_cancer
```

Out[6]:

```
array([[1.80e+01, 1.04e+01, 1.23e+02, ..., 2.65e-01, 4.60e-01, 1.19e-01],
       [2.06e+01, 1.78e+01, 1.33e+02, ..., 1.86e-01, 2.75e-01, 8.90e-02],
       [1.97e+01, 2.12e+01, 1.30e+02, ..., 2.43e-01, 3.61e-01, 8.76e-02],
       ...,
       [1.66e+01, 2.81e+01, 1.08e+02, ..., 1.42e-01, 2.22e-01, 7.82e-02],
       [2.06e+01, 2.93e+01, 1.40e+02, ..., 2.65e-01, 4.09e-01, 1.24e-01],
       [7.76e+00, 2.45e+01, 4.79e+01, ..., 0.00e+00, 2.87e-01, 7.04e-02]])
```

In [7]:

```
y_cancer
```

Out[7]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [8]:

```
##Linear Regression
```

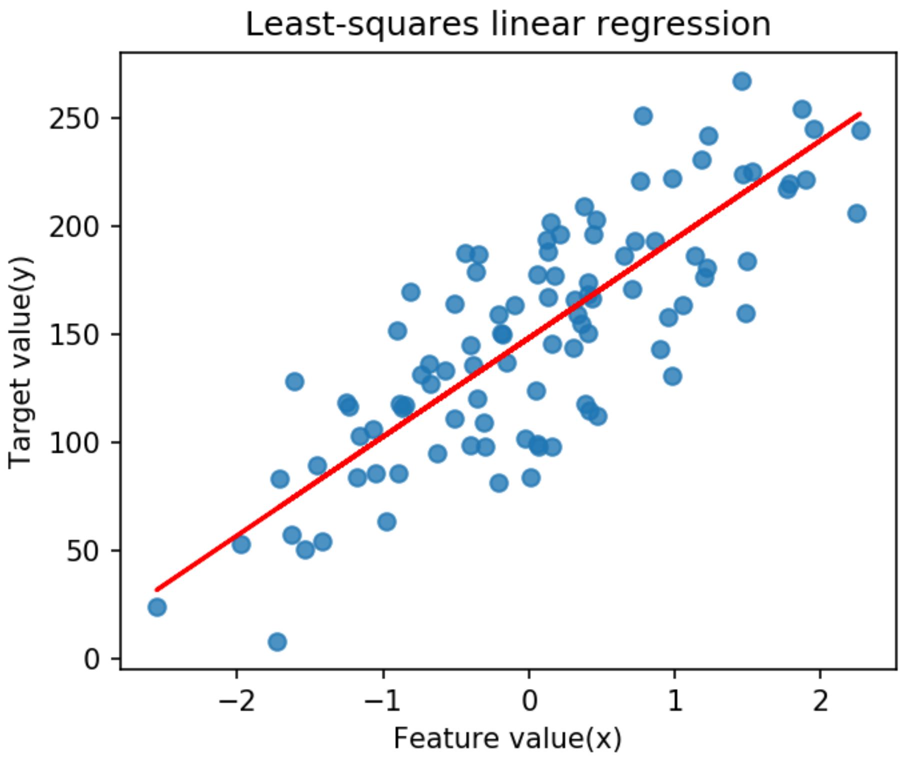
In [9]:

```
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train,  y_test=train_test_split(X_R1,
                                                   y_R1,
                                                   random_state=0)
linreg=LinearRegression().fit(X_train,y_train)

print('linear model coeff(w):{}'.format(linreg.coef_))
print('linear model intercept(b):{:.2f}'.format(linreg.intercept_))
print('R-squared score(training):{:.3f}'.format(linreg.score(X_train,y_train)))
print('R-squared score(training):{:.3f}'.format(linreg.score(X_test,y_test)))
```

```
linear model coeff(w):[45.71]
linear model intercept(b):148.45
R-squared score(training):0.679
R-squared score(training):0.492
```

In [10]:

```
plt.figure(figsize=(5,4))
plt.scatter(X_R1,y_R1,marker='o',alpha=0.8)
plt.plot(X_R1, linreg.coef_* X_R1+linreg.intercept_,'r-')
plt.title('Least-squares linear regression')
plt.xlabel('Feature value(x)')
plt.ylabel('Target value(y)')
plt.show()
```



In [ ]:

```
#Logistic Regression
```

In [14]:

```
from sklearn.linear_model import LogisticRegression

y_fruits_apple=y_fruits_2d==1
X_train,X_test,y_train,y_test=train_test_split(X_fruits_2d,
                                               y_fruits_apple,
                                               random_state=0)
clf=LogisticRegression().fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:0.75
Accuracy of Logistic classifier on test set:0.67
```

In [15]:

```
clf.predict([[6,8]])
```

Out[15]:

```
array([ True])
```

In [16]:

```
X_train,X_test,y_train,y_test=train_test_split(X_cancer,
                                               y_cancer,
                                               random_state=0)
clf=LogisticRegression().fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:0.95
Accuracy of Logistic classifier on test set:0.94

C:₩Users₩donghyunkim₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.p
y:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [17]:

```
##Support Vector Machine
```

In [18]:

```
from sklearn.svm import SVC
X_train,X_test,y_train,y_test=train_test_split(X_fruits_2d,
                                               y_fruits_apple,
                                               random_state=0)
clf=SVC(kernel='linear').fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:0.84
Accuracy of Logistic classifier on test set:0.67
```

In [19]:

```
X_train,X_test,y_train,y_test=train_test_split(X_cancer,
                                               y_cancer,
                                               random_state=0)
clf=SVC(kernel='linear').fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:0.97
Accuracy of Logistic classifier on test set:0.96
```

In [20]:

```
## Decision Tree
```

In [22]:

```
from sklearn.tree import DecisionTreeClassifier

X_train,X_test,y_train,y_test=train_test_split(X_fruits_2d,
                                               y_fruits_apple,
                                               random_state=0)
clf=DecisionTreeClassifier().fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:1.00
Accuracy of Logistic classifier on test set:0.67
```

In [23]:

```
X_train,X_test,y_train,y_test=train_test_split(X_cancer,
                                               y_cancer,
                                               random_state=0)
clf=DecisionTreeClassifier().fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:1.00
Accuracy of Logistic classifier on test set:0.89
```

In [24]:

```
##Random Forest
```

In [25]:

```
from sklearn.ensemble import RandomForestClassifier

X_train,X_test,y_train,y_test=train_test_split(X_fruits_2d,
                                               y_fruits_apple,
                                               random_state=0)
clf=RandomForestClassifier(n_estimators=10,random_state=0).fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:1.00
Accuracy of Logistic classifier on test set:0.67
```

In [27]:

```
X_train,X_test,y_train,y_test=train_test_split(X_cancer,
                                               y_cancer,
                                               random_state=0)
clf=RandomForestClassifier(max_features=8,n_estimators=10,random_state=0).fit(X_train,y_train)

print('Accuracy of Logistic classifier on training set:{:.2f}'
     .format(clf.score(X_train,y_train)))
print('Accuracy of Logistic classifier on test set:{:.2f}'
     .format(clf.score(X_test,y_test)))
```

```
Accuracy of Logistic classifier on training set:1.00
Accuracy of Logistic classifier on test set:0.99
```

In [ ]: