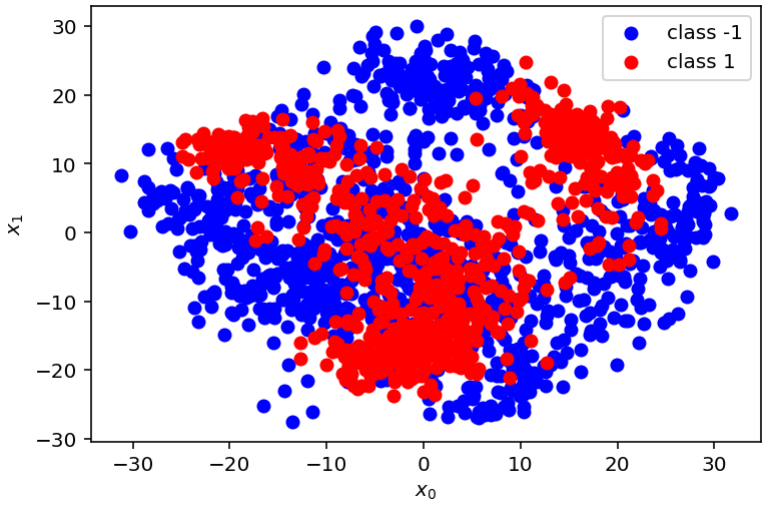
In [2]:
```
%config InlineBackend.figure_format = 'retina'
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree

from statistics import median
```

Load the data, modify, and visualize

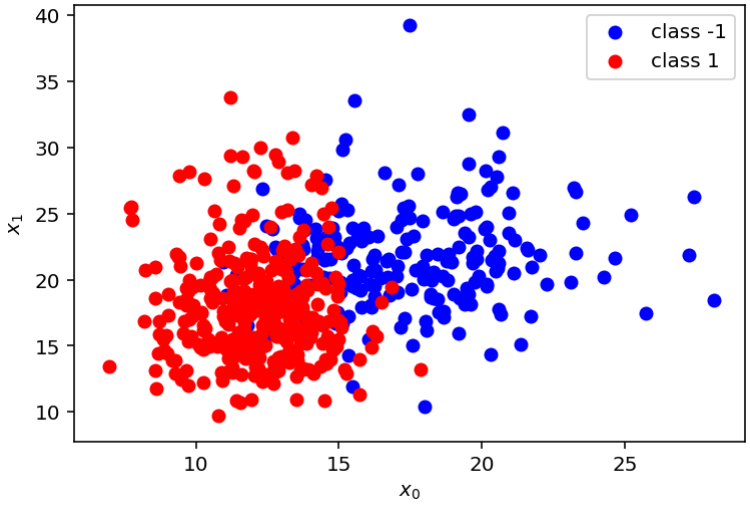
In [3]:
```
def vis(X, Y, W=None, b=None):
    indices_neg1 = (Y == -1).nonzero()[0]
    indices_pos1 = (Y == 1).nonzero()[0]
    plt.scatter(X[:,0][indices_neg1], X[:,1][indices_neg1],
                c='blue', label='class -1')
    plt.scatter(X[:,0][indices_pos1], X[:,1][indices_pos1],
                c='red', label='class 1')
    plt.legend()
    plt.xlabel('$x_0$')
    plt.ylabel('$x_1$')

    if W is not None:
        # w0x0+w1x1+b=0 => x1=-w0x0/w1-b/w1
        w0 = W[0]
        w1 = W[1]
        temp = -w1*np.array([X[:,1].min(), X[:,1].max()])/w0-b/w0
        x0_min = max(temp.min(), X[:,0].min())
        x0_max = min(temp.max(), X[:,1].max())
        x0 = np.linspace(x0_min,x0_max,100)
        x1 = -w0*x0/w1-b/w1
        plt.plot(x0,x1,color='black')

    plt.show()
```

In [4]:
```
digits = datasets.load_digits()
breast_cancer = datasets.load_breast_cancer()
housing = datasets.fetch_california_housing(data_home=None, download_if_mis
#print(housing.target)

#Lower the dimension of digits data
pca = PCA(n_components=32)
X_digits = pca.fit_transform(digits.data)
Y_digits = (digits.target > 5).reshape(-1,1).astype(np.float)
Y_digits[Y_digits==0] = -1

X_breast_cancer = breast_cancer.data
Y_breast_cancer = breast_cancer.target.reshape(-1,1)
Y_breast_cancer[Y_breast_cancer == 0] = -1

mean_housing = np.mean(housing.target)
print("The median housing price is: ", mean_housing)
X_housing = housing.data
Y_housing = (housing.target > mean_housing).reshape(-1,1).astype(np.float)
Y_housing[Y_housing==0] = -1

print(X_digits.shape)
print(Y_digits.shape)
print(X_breast_cancer.shape)
print(Y_breast_cancer.shape)
print(X_housing.shape)
print(Y_housing.shape)

print("Visualizing digits data: ")
vis(X_digits, Y_digits)
print("Visualizing breast_cancer data: ")
vis(X_breast_cancer, Y_breast_cancer)
print("Visualizing housing price data: ")
vis(X_housing, Y_housing)
```

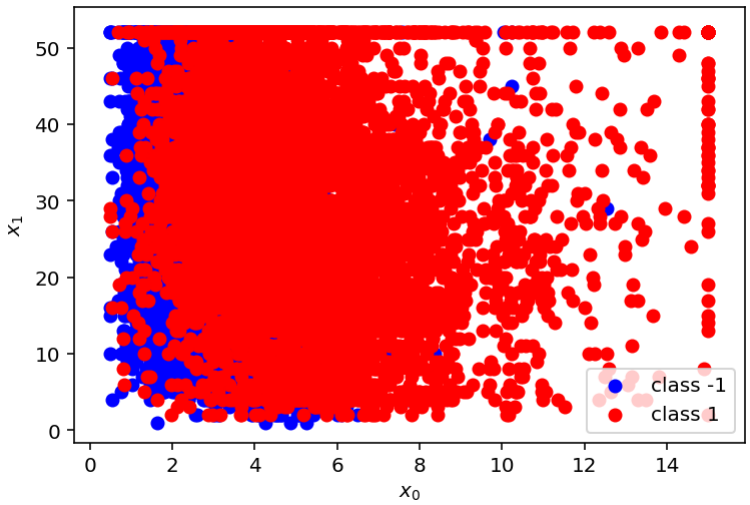
```
The median housing price is:  2.068558169089147
(1797, 32)
(1797, 1)
(569, 30)
(569, 1)
(20640, 8)
(20640, 1)
Visualizing digits data:
```



Visualizing breast_cancer data:



Visualizing housing price data:



Function to split training data and testing data

In [5]:
```
def shuffleAndSplit(X, Y, train, test, seed):
    X_and_Y = np.hstack((X, Y))
    np.random.seed(seed)
    np.random.shuffle(X_and_Y)
    X_shuffled = X_and_Y[:,:X.shape[1]]
    Y_shuffled = X_and_Y[:,X.shape[1]]

    trainIndex = (int)(train*len(X_shuffled))
    testIndex = (int)(test*len(X_shuffled))
    X_train = X_shuffled[0:trainIndex]
    Y_train = Y_shuffled[0:trainIndex]
    X_test = X_shuffled[-testIndex:]
    Y_test = Y_shuffled[-testIndex:]
    #print("Length of total: ", len(X_shuffled))
    #print("Length of X_train: ", len(X_train))
    #print("Length of X_test: ", X_test.shape)
    return X_train, Y_train, X_test, Y_test
```

Functions to calculate the training and testing errors

In [6]:
```
def Cal_error(prediction, X, Y):
    total_error = 0
    for i in range(len(X)):
        if (prediction[i] != Y[i]):
            total_error += 1
    test_error = total_error/len(X)
    return test_error

def getErrorWithCrossValidation(gridValue, train_split, test_split, X, Y):
    train_error = 0;
    test_error = 0;
    for i in range(0,3):
        X_train, Y_train, X_test, Y_test = shuffleAndSplit(X, Y, train_spli
        gridValue.fit(X_train, Y_train)
        prediction_train = gridValue.best_estimator_.predict(X_train)
        prediction_test = gridValue.best_estimator_.predict(X_test)
        train_error += Cal_error(prediction_train, X_train, Y_train)
        test_error += Cal_error(prediction_test, X_test, Y_test)
    train_error = train_error/3.0
    test_error = test_error/3.0
    print("{}/{} Split training error: ".format(train_split*100, test_split
    print("{}/{} Split testing error: ".format(train_split*100, test_split*
```

Decision Tree Classifier ¶

In [7]:
```
DTree = tree.DecisionTreeClassifier()
D_list = [2, 3, 4, 5, 6]
D_param_grid = {'max_depth': D_list, 'criterion':['gini', 'entropy'], 'spli
gridDTree = GridSearchCV(DTree,D_param_grid,cv=10,verbose=1,return_train_sc
```

KNN Classifier

In [8]:
```
knn=KNeighborsClassifier()
k_range=list(range(1,9))
knn_grid_param=dict(n_neighbors=k_range,leaf_size=[10,20])
gridKnn=GridSearchCV(knn,knn_grid_param,cv=10,verbose=1,return_train_score=
```

Random Forest

In [9]:
```
RF=RandomForestClassifier()
rf_parameter=[{'n_estimators':[1,10,50,100,200],'min_samples_leaf':[1,2,4,8
gridRF=GridSearchCV(RF,rf_parameter,cv=10,verbose=1,return_train_score=True
```

Digits dataset on three classifiers

In [151]:
```
#Using Decision Tree
getErrorWithCrossValidation(gridDTree, 0.2, 0.8, X_digits, Y_digits)
getErrorWithCrossValidation(gridDTree, 0.8, 0.2, X_digits, Y_digits)

#Using Knn
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridKnn, 0.2, 0.8, X_digits, Y_digits)
getErrorWithCrossValidation(gridKnn, 0.8, 0.2, X_digits, Y_digits)

#Using Random Forest
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridRF, 0.2, 0.8, X_digits, Y_digits)
getErrorWithCrossValidation(gridRF, 0.8, 0.2, X_digits, Y_digits)
```

```
rs.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:    0.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent worke
rs.

80.0/20.0 Split training error:  0.08049176525168174
80.0/20.0 Split testing error:  0.15784586815227483
############################################################
############################################################
############################################################
Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=-1)]: Done 160 out of 160 | elapsed:    0.5s finished
/Users/liuyouliang/opt/anaconda3/lib/python3.7/site-packages/sklearn/mode
l_selection/_search.py:814: DeprecationWarning: The default of the `iid`
parameter will change from True to False in version 0.22 and will be remo
ved in 0.24. This will change numeric results when test-set sizes are une
qual.
```

Breast Cancer dataset on three classifiers

In [153]:
```
#Using Decision Tree
getErrorWithCrossValidation(gridDTree, 0.2, 0.8, X_breast_cancer, Y_breast_
getErrorWithCrossValidation(gridDTree, 0.8, 0.2, X_breast_cancer, Y_breast_

#Using Knn
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridKnn, 0.2, 0.8, X_breast_cancer, Y_breast_ca
getErrorWithCrossValidation(gridKnn, 0.8, 0.2, X_breast_cancer, Y_breast_ca

#Using Random Forest
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridRF, 0.2, 0.8, X_breast_cancer, Y_breast_can
getErrorWithCrossValidation(gridRF, 0.8, 0.2, X_breast_cancer, Y_breast_can
```

```
  DeprecationWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent worke
rs.
```

```
80.0/20.0 Split training error:  0.018315018315018316
80.0/20.0 Split testing error:  0.05014749262536874
############################################################
############################################################
############################################################
Fitting 10 folds for each of 16 candidates, totalling 160 fits
Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=-1)]: Done 160 out of 160 | elapsed:    0.2s finished
/Users/liuyouliang/opt/anaconda3/lib/python3.7/site-packages/sklearn/mode
l_selection/_search.py:814: DeprecationWarning: The default of the `iid`
parameter will change from True to False in version 0.22 and will be remo
ved in 0.24. This will change numeric results when test-set sizes are une
```

California Housing datasets on three different classifiers

In [10]:
```
#Using Decision Tree
getErrorWithCrossValidation(gridDTree, 0.2, 0.8, X_housing, Y_housing)
getErrorWithCrossValidation(gridDTree, 0.8, 0.2, X_housing, Y_housing)

#Using Knn
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridKnn, 0.2, 0.8, X_housing, Y_housing)
getErrorWithCrossValidation(gridKnn, 0.8, 0.2, X_housing, Y_housing)

#Using Random Forest
print("############################################################")
print("############################################################")
print("############################################################")
getErrorWithCrossValidation(gridRF, 0.2, 0.8, X_housing, Y_housing)
getErrorWithCrossValidation(gridRF, 0.8, 0.2, X_housing, Y_housing)
```

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    1.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    1.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    1.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

20.0/80.0 Split training error:  0.15253552971576226
20.0/80.0 Split testing error:  0.18003068475452197
Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    4.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    4.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

80.0/20.0 Split training error:  0.15423126614987082
80.0/20.0 Split testing error:  0.16343669250645995
```

```
80.0/20.0 Split testing error:  0.1634366925064599
##########################################################
##########################################################
##########################################################
Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   14.7s finished

Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   15.1s finished

Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:   15.4s finished

20.0/80.0 Split training error:  0.24911175710594313
20.0/80.0 Split testing error:  0.39312822997416025
Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:  1.1min finished

Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:  1.1min finished

Fitting 10 folds for each of 16 candidates, totalling 160 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 160 out of 160 | elapsed:  1.1min finished

80.0/20.0 Split training error:  0.21753875968992245
80.0/20.0 Split testing error:  0.35287467700258396
##########################################################
##########################################################
##########################################################
Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  1.1min finished

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  1.1min finished

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.

[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  1.1min finished
```

```
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  1.1min finished

20.0/80.0 Split training error:  0.0
20.0/80.0 Split testing error:  0.12471737726098191
Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  4.4min finished

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  4.3min finished

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent w
orkers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  4.4min finished

80.0/20.0 Split training error:  0.0033309108527131786
80.0/20.0 Split testing error:  0.11191860465116278
```

In [ ]: