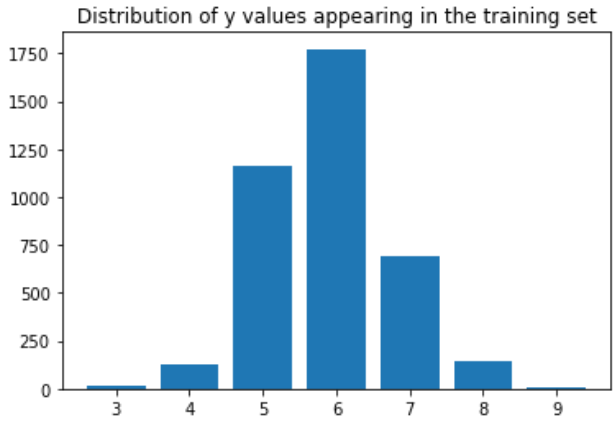
In [1]:
```
import numpy as np
from numpy.linalg import inv
import _pickle as cp
import matplotlib.pyplot as plt
```

In [2]:
```
X, y = cp.load(open('winequality-white.pickle', 'rb'))
```

In [3]:
```
N, D = X.shape
N_train = int(0.8 * N)
N_test = N - N_train
X_train = X[:N_train]
y_train = y[:N_train]
X_test = X[N_train:]
y_test = y[N_train:]
```

Handin 1

In [4]:
```
def categorize(train):
    hist = {}
    for i in train:
        hist[i] = hist.get(i, 0) + 1
    return hist

count = categorize(y_train)
plt.bar(count.keys(), count.values())
plt.title('Distribution of y values appearing in the training set')
plt.show()
```



Handin 2

In [5]:
```
average_y_train = np.mean(y_train)

def mseError(predictor, y):
    y = y - predictor
    y = y * y
    return np.sum(y)/len(y)

print("Eorror on the training dataset", mseError(average_y_train, y_train))
print("Eorror on the testing dataset", mseError(average_y_train, y_test))
```

```
Eorror on the training dataset 0.7767772386501117
Eorror on the testing dataset 0.8138573000045535
```

Handin 3

In [6]:
```
def standarlize(x):
    var = np.var(x)
    mean = np.mean(x)
    return (mean, var)

def transformData(x, mean, var):
    x -= mean
    x /= np.sqrt(var)
    return x

def find_weight(X, y):
    X_trans = X.T
    return np.matmul(np.matmul(inv(np.matmul(X_trans, X)), X_trans), y)
```

In [41]:
```
# Placeholder for the standalized data
X_train_standard = np.zeros(shape = X_train.shape)
X_test_standard = np.zeros(shape = X_test.shape)


# For each feature, calculate for mean and std, transform the data for both tes
t/train
for i in range(D):
    train_data = X_train[:, i]
    test_data = X_test[:, i]

    m,v = standarlize(train_data)

    X_train_standard[:, i] = transformData(train_data, m, v)
    X_test_standard[:, i] = transformData(test_data, m, v)

    print("training: mean =", np.mean(X_train_standard[:, i]), "var =", np.var
(X_train_standard[:, i]))
    print("Testing: mean =", np.mean(X_test_standard[:, i]), "var =", np.var(X_
test_standard[:, i]))
```

```
training: mean = 2.3122562228027764e-17 var = 1.0
Testing: mean = 0.0030527846038981995 var = 1.0559133327466625
training: mean = -1.269474004676034e-17 var = 1.0
Testing: mean = -0.007511859999128221 var = 1.1233168622542755
training: mean = 1.8135342923943342e-17 var = 1.0
Testing: mean = -0.02820964217791591 var = 0.741905848945599
training: mean = 9.067671461971671e-18 var = 1.0000000000000002
Testing: mean = -0.03437040308176073 var = 1.01439428211428
training: mean = 7.254137169577338e-18 var = 1.0000000000000002
Testing: mean = -0.011299108564615596 var = 1.0831531580187885
training: mean = -8.614287888873087e-18 var = 1.0000000000000002
Testing: mean = -0.005327945753180028 var = 0.9445614430786275
training: mean = 1.0881205754366006e-17 var = 1.0
Testing: mean = -0.017991526288517565 var = 1.022899496018027
training: mean = 1.0881205754366006e-17 var = 1.0000000000000002
Testing: mean = -0.02239597338629517 var = 1.0361602306269688
training: mean = 2.1762411508732013e-17 var = 1.0000000000000002
Testing: mean = 0.05970897162986136 var = 1.0664289520416237
training: mean = -1.8135342923943342e-17 var = 1.0000000000000002
Testing: mean = 0.03499859792244784 var = 1.0987830603456172
training: mean = 2.7203014385915015e-17 var = 1.0000000000000002
Testing: mean = 0.010960253895340912 var = 1.0418353212866018
```

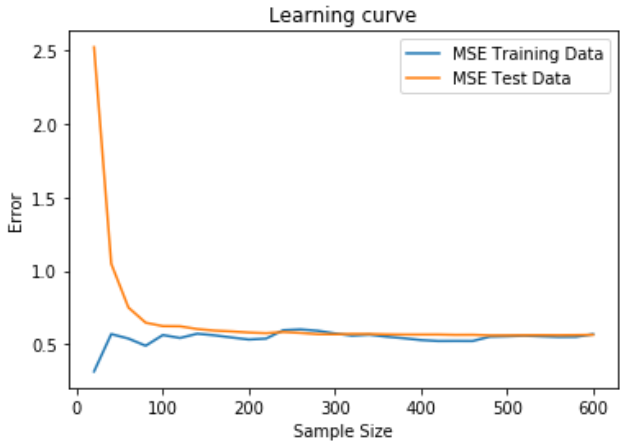
In [42]:
```
# Include a column of ones to data matrices
b_training = np.ones((N_train,1))
b_test = np.ones((N_test,1))
b_tot = np.ones((N,1))

new_X_train_standard = np.hstack((b_training, X_train_standard))
new_X_test_standard = np.hstack((b_test, X_test_standard))

weights = find_weight(new_X_train_standard, y_train)

# Expected y
y_train_expected = np.dot(new_X_train_standard, weights)
y_test_expected = np.dot(new_X_test_standard, weights)

print("Eorror on the training dataset", mseError(y_train_expected, y_train))
print("Eorror on the testing dataset", mseError(y_train_expected, y_train))
```

```
Eorror on the training dataset 0.5639996173941925
Eorror on the testing dataset 0.5639996173941925
```

Handin 4 Learning Curve

In [52]:
```
def FirstNmseError(predictor, y, N):
    t = y[0:N] - predictor[0:N]
    t = t * t
    return np.sum(t)/len(t)
```

In [70]:
```
train_error = []
test_error = []

for i in range(20, 601, 20):
    weights = find_weight(new_X_train_standard[:i], y_train[:i])

    # Expected y
    y_train_expected = np.dot(new_X_train_standard[:i], weights)
    y_test_expected = np.dot(new_X_test_standard, weights)

    train_error.append(FirstNmseError(y_train_expected, y_train, i))
    test_error.append(mseError(y_test_expected, y_test))

mat1 = np.array(train_error)
mat2 = np.array(test_error)

plt.figure()
plt.title('Learning curve')
plt.xlabel("Sample Size")
plt.ylabel('Error')
plt.plot(np.linspace(20,600, num=mat1.shape[0]), mat1, label='MSE Training Data
')
plt.plot(np.linspace(20,600, num=mat2.shape[0]), mat2, label='MSE Test Data')
plt.legend()
plt.show()
```



In [ ]: