Name: Joyce Le
ID#: 82549113
UCI NetID: lejy

CS 178 Homework 2

Problem 1

1.

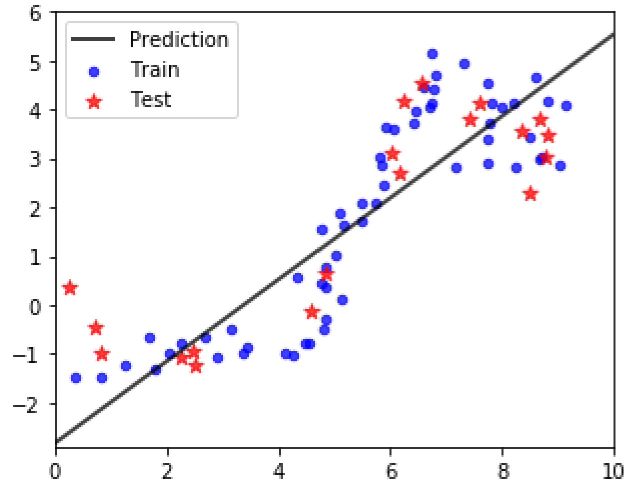
In [237]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml

# load data
data = np.genfromtxt("data/curve80.txt", delimiter=None)

#split data
X = data[:,0]
X = np.atleast_2d(X).T # code expects shape (M,N) so make sure it's 2-dimen
sional
Y = data[:,1]
Xtr,Xte,Ytr,Yte = ml.splitData(X,Y,0.75) # split data set 75/25

# print the shapes of these four objects
print(Xtr.shape)
print(Xte.shape)
print(Ytr.shape)
print(Yte.shape)
```

```
(60, 1)
(20, 1)
(60,)
(20,)
```

2.

In [244]:
```
print("a)")
lr = ml.linear.linearRegress( Xtr, Ytr ) # create and train model
xs = np.linspace(0,10,200) # densely sample possible x-values
xs = xs[:,np.newaxis] # force "xs" to be an Mx1 matrix (expected by our cod
e)
ys = lr.predict( xs ) # make predictions at xs

f, ax = plt.subplots(1, 1, figsize=(5, 4))
ax.scatter(Xtr, Ytr, s=20, color='blue', alpha=0.75, label='Train')
ax.scatter(Xte, Yte, s=60, marker='*', color='red', alpha=0.75, label='Tes
t')
ax.plot(xs, ys, lw=2, color='black', alpha=0.75, label='Prediction')
ax.set_xlim(0, 10)
ax.set_ylim(-2.9, 6)
ax.legend(fontsize=10, loc=0)
plt.show()

print("b)")
print(lr.theta)
print('''The linear regression coefficients do match the plot
because the y-intercept does appear to be somewhere close to
-3 and taking the points (2,-1) and (8,4) on the graph, which
are close approximates that fall on the line of regression,
we can can calculate that the slope is 5/6 = 0.833, which is
a value close to the coefficient 0.8361.''')

print("\nc)")

def MSE(lr, X, Y):
    Yhat = lr.predict(X)
    return np.mean((Y - Yhat.reshape(Y.shape))**2 , axis=0)

print("Training data MSE:", MSE(lr, Xtr, Ytr))
print("Test data MSE:", MSE(lr, Xte, Yte))
```

a)



b)

```
[[-2.82765049  0.83606916]]
The linear regression coefficients do match the plot
because the y-intercept does appear to be somewhere close to
-3 and taking the points (2,-1) and (8,4) on the graph, which
are close approximates that fall on the line of regression,
we can can calculate that the slope is 5/6 = 0.833, which is
a value close to the coefficient 0.8361.
```

c)

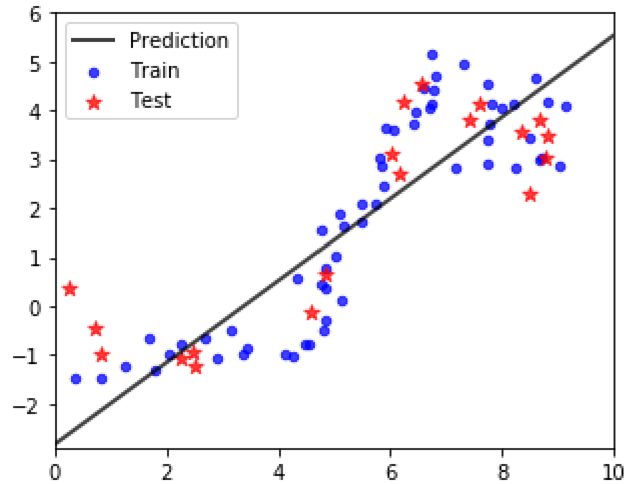
```
Training data MSE: 1.127711955609391
Test data MSE: 2.2423492030101246
```

3.

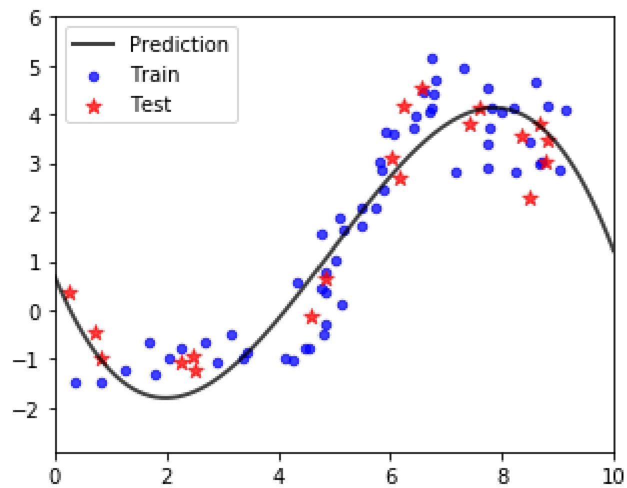
In [239]:
```
print("a)\n")
degrees = np.array([1,3,5,7,10,18])
train_errors = np.zeros(degrees.shape[0])
test_errors = np.zeros(degrees.shape[0])
for i in range(6):
    degree = degrees[i]

    XtrP = ml.transforms.fpoly(Xtr, degree, False)
    XtrP,params = ml.transforms.rescale(XtrP)
    XteP,_ = ml.transforms.rescale(ml.transforms.fpoly(Xte, degree, False),
 params)
    lr = ml.linear.linearRegress(XtrP, Ytr)
    xs = np.linspace(0, 10, 200)
    xs = np.atleast_2d(xs).T

    # Transform the predicting xs
    xsP,_ = ml.transforms.rescale(ml.transforms.fpoly(xs, degree, False), p
arams)
    ys = lr.predict(xsP)
    train_errors[i] = MSE(lr, XtrP, Ytr)
    test_errors[i] = MSE(lr, XteP, Yte)

    # Plot the data
    f, ax = plt.subplots(1, 1, figsize=(5, 4))
    ax.scatter(Xtr, Ytr, s=20, color='blue', alpha=0.75, label='Train')
    ax.scatter(Xte, Yte, s=60, marker='*', color='red', alpha=0.75, label=
'Test')

    # Plot the regression line
    ax.plot(xs, ys, lw=2, color='black', alpha=0.75, label='Prediction')
    ax.set_xlim(0, 10)
    ax.set_ylim(-2.9, 6)
    # Control the size of the legend and the location.
    ax.legend(fontsize=10, loc=0)

    print("degree", degrees[i])
    plt.show()
```

a)

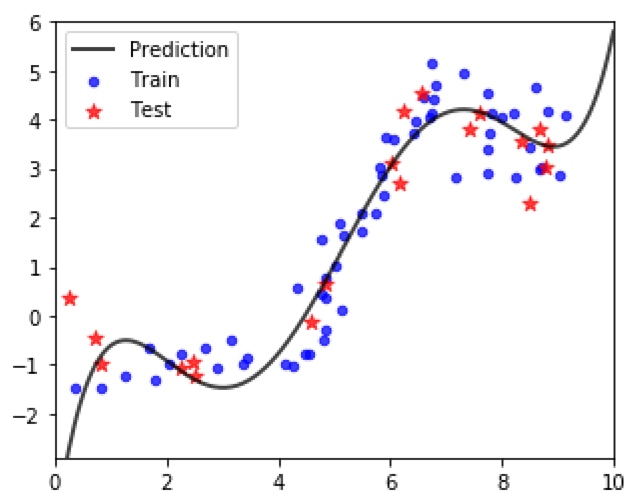
degree 1



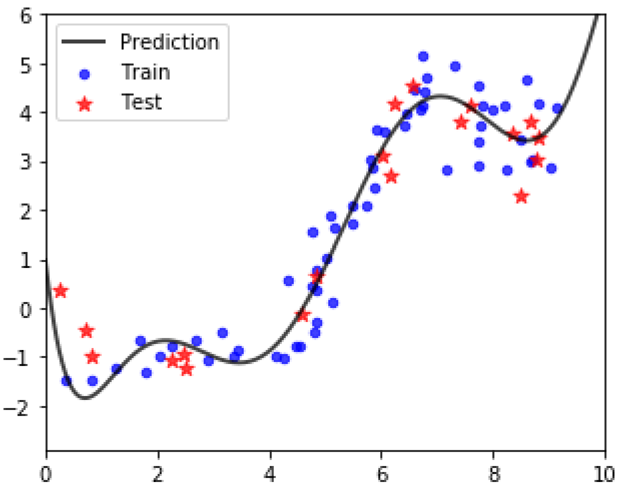
degree 3



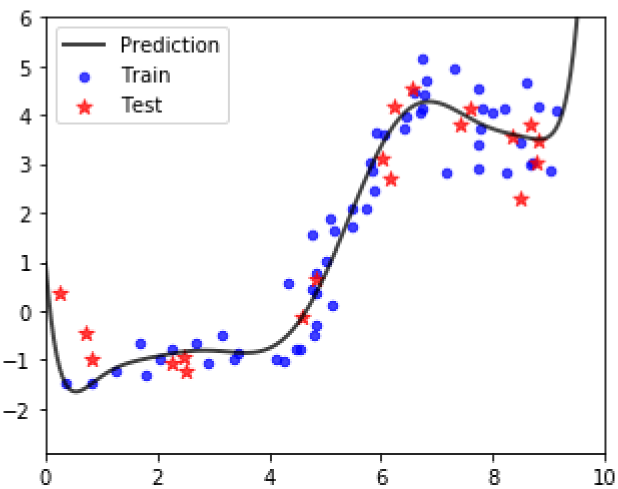
degree 5



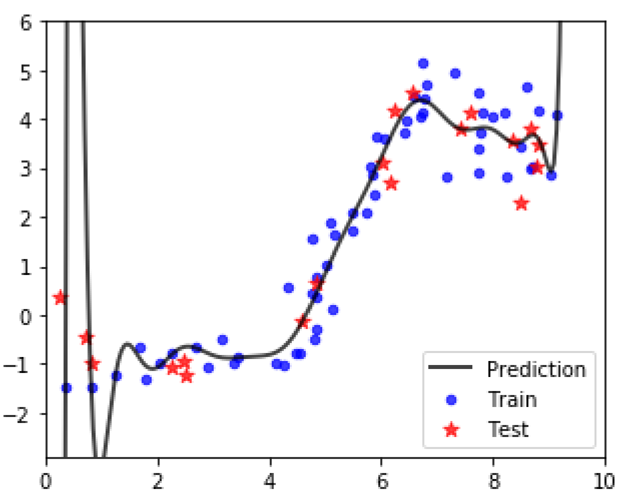
degree 7




```
degree 10
```

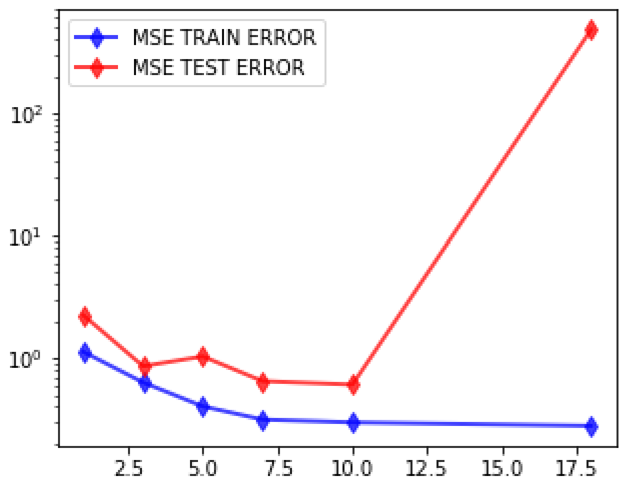



```
degree 18
```



In [240]:
```
print("b)")
f, ax = plt.subplots(1, 1, figsize=(5, 4))
ax.semilogy(degrees, train_errors, lw=2, marker='d', color = "blue", marker
size=7, alpha=0.75, label='MSE TRAIN ERROR')
ax.semilogy(degrees, test_errors, lw=2, marker='d', color = "red", markersi
ze=7, alpha=0.75, label='MSE TEST ERROR')
ax.legend(fontsize=10, loc=0)
plt.show()

print("\nc) I would recommend polynomial degree 5.")
```

b)

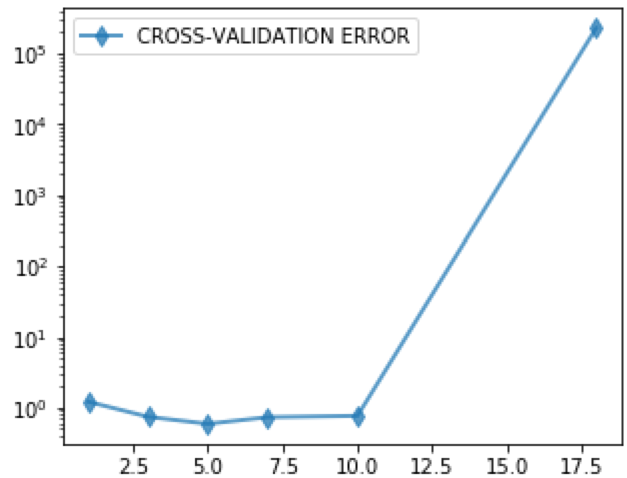


c) I would recommend polynomial degree 5.

Problem 2

1.

In [241]:
```
def find_cross_validation_error(degree: int, folds: int):
    J = np.zeros(folds)
    for iFold in range(folds):
        Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,folds,iFold) # use ith b
lock as validation
        XtiP = ml.transforms.fpoly(Xti, degree, False)
        XtiP,params = ml.transforms.rescale(XtiP)
        XviP,_ = ml.transforms.rescale(ml.transforms.fpoly(Xvi, degree, Fal
se), params)
        lr = ml.linear.linearRegress(XtiP, Yti)
        J[iFold] = MSE(lr, XviP, Yvi)
    return np.mean(J)

errors = np.zeros(degrees.shape)
for i,degree in enumerate(degrees):
    errors[i] = find_cross_validation_error(degree, 5)

f, ax = plt.subplots(1, 1, figsize=(5, 4))
ax.semilogy(degrees, errors, lw=2, marker='d', markersize=7, alpha=0.75, la
bel='CROSS-VALIDATION ERROR')
ax.legend(fontsize=10, loc=0)
plt.show()
```



2.

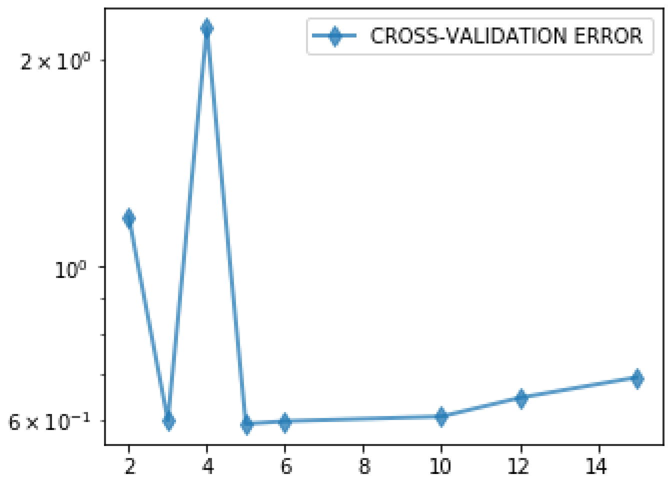
The MSE estimates from five-fold cross-validation are pretty similar compared to the MSE's evaluated on Problem 1, though the higher degrees are notably worse and start overfitting sooner.

3.

I would recommend polynomial degree 5 based on five-fold cross-validation error.

4.

In [242]:
```
nFolds = np.array([2,3,4,5,6,10,12,15])
errors = np.zeros(nFolds.shape)

for i,fold in enumerate(nFolds):
    errors[i] = find_cross_validation_error(5, fold)

f, ax = plt.subplots(1, 1, figsize=(5, 4))
ax.semilogy(nFolds, errors, lw=2, marker='d', markersize=7, alpha=0.75, lab
el='CROSS-VALIDATION ERROR')
ax.legend(fontsize=10, loc=0)
plt.show()

print('''There's a large amount of fluctuation of cross-validation
errors at the beginning because we don't have enough test samples
to average out our errors.  However, past a certain number of folds,
the errors start to increase a lot more steadily as the numbres of
folds increases.  This can be explained by the fact that we have
more test results to average as we increaes the number of folds,
but the validation errors will steadily increaseover time because
we are reducing the amount of training data each time we create a
new fold, which will make way for more errors.''')
```




```
There's a large amount of fluctuation of cross-validation
errors at the beginning because we don't have enough test samples
to average out our errors.  However, past a certain number of folds,
the errors start to increase a lot more steadily as the numbres of
folds increases.  This can be explained by the fact that we have
more test results to average as we increaes the number of folds,
but the validation errors will steadily increaseover time because
we are reducing the amount of training data each time we create a
new fold, which will make way for more errors.
```

Problem 3: Statement of Collaboration

I collaborated with Jordan Rayfield on Problem 1.2 to discuss how to define an MSE function.