Homework 2

Problem 1: Linear Regression (60 points)

1.1 Print the shapes of these four objects. (5 points)

In [250]:

```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml

data = np.genfromtxt ( "curve80.txt", delimiter = None)
#The first column (data[:,0]) is the scalar feature value x;
X = data[:,0]

# code expects shape (M,N) so make sure it's 2-dimensional
X = np.atleast_2d(X).T

#The second column data[:,1] is the target value y for each example.
Y = data[:,1]
# split data set 75/25

Xtr,Xte,Ytr,Yte = ml.splitData(X,Y,0.75)

print ("Xtr = ", Xtr.shape)
print ("Xte = ", Xte.shape)
print ("Ytr = ", Ytr.shape)
print ("Yte = ", Yte.shape)
```

```
Xtr =  (60, 1)
Xte =  (20, 1)
Ytr =  (60,)
Yte =  (20,)
```

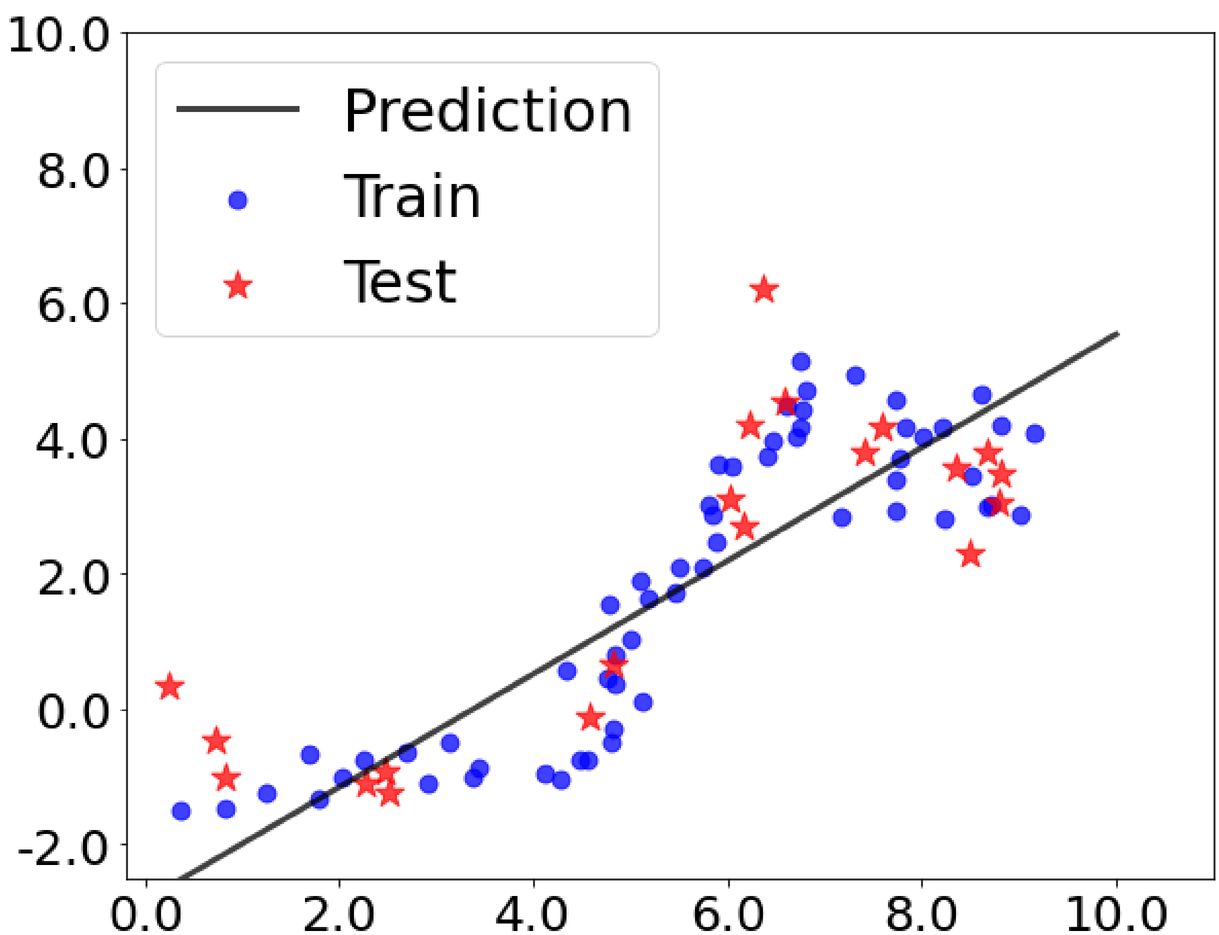
1.2 Use the provided linearRegress class to create a linear regression predictor of y given x. You can plot the resulting function by simply evaluating the model at a large number of x values xs

In [251]:
```
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
ys = lr.predict( xs ) # make predictions at xs
```

1.2.a : Plot the training data points along with your prediction function in a single plot. (10 points)

In [252]:

```
# Plotting the data
f, ax = plt.subplots(1, 1, figsize=(10, 8))

ax.scatter(Xtr, Ytr, s=80, color='blue', alpha=0.75, label='Train')
ax.scatter(Xte, Yte, s=240, marker='*', color='red', alpha=0.75, label=

# Also plotting the regression line
ax.plot(xs, ys, lw=3, color='black', alpha=0.75, label='Prediction')

ax.set_xlim(-0.2, 11)
ax.set_ylim(-2.5, 10)
ax.set_xticklabels(ax.get_xticks(), fontsize=25)
ax.set_yticklabels(ax.get_yticks(), fontsize=25)

# Controlling the size of the legend and the location.
ax.legend(fontsize=30, loc=0)

plt.show()
```



In [5]:

```
print ( " Linear regression coefficients", lr.theta )
```

```
Linear regression coefficients [[-2.82765049  0.83606916]]
```

y = 0.836 + (-)2.8277x

1.2.c What is the mean squared error of the predictions on the training and test data? (10 points)

In [253]:

```
def MSE(y_true, y_hat):
    y_true = y_true.reshape (-1, 1)
    mse = np.sum ( (y_true - y_hat)**2 ) / len (y_true)

    return mse
YtrHat = lr.theta[0][1] * Xtr + lr.theta[0][0]
YteHat = lr.theta[0][1] * Xte + lr.theta[0][0]

print('Mean Squared Erro (traning data) = ', MSE(Ytr, YtrHat))
print('Mean Squared Erro (test data    ) = ', MSE(Yte, YteHat))
```

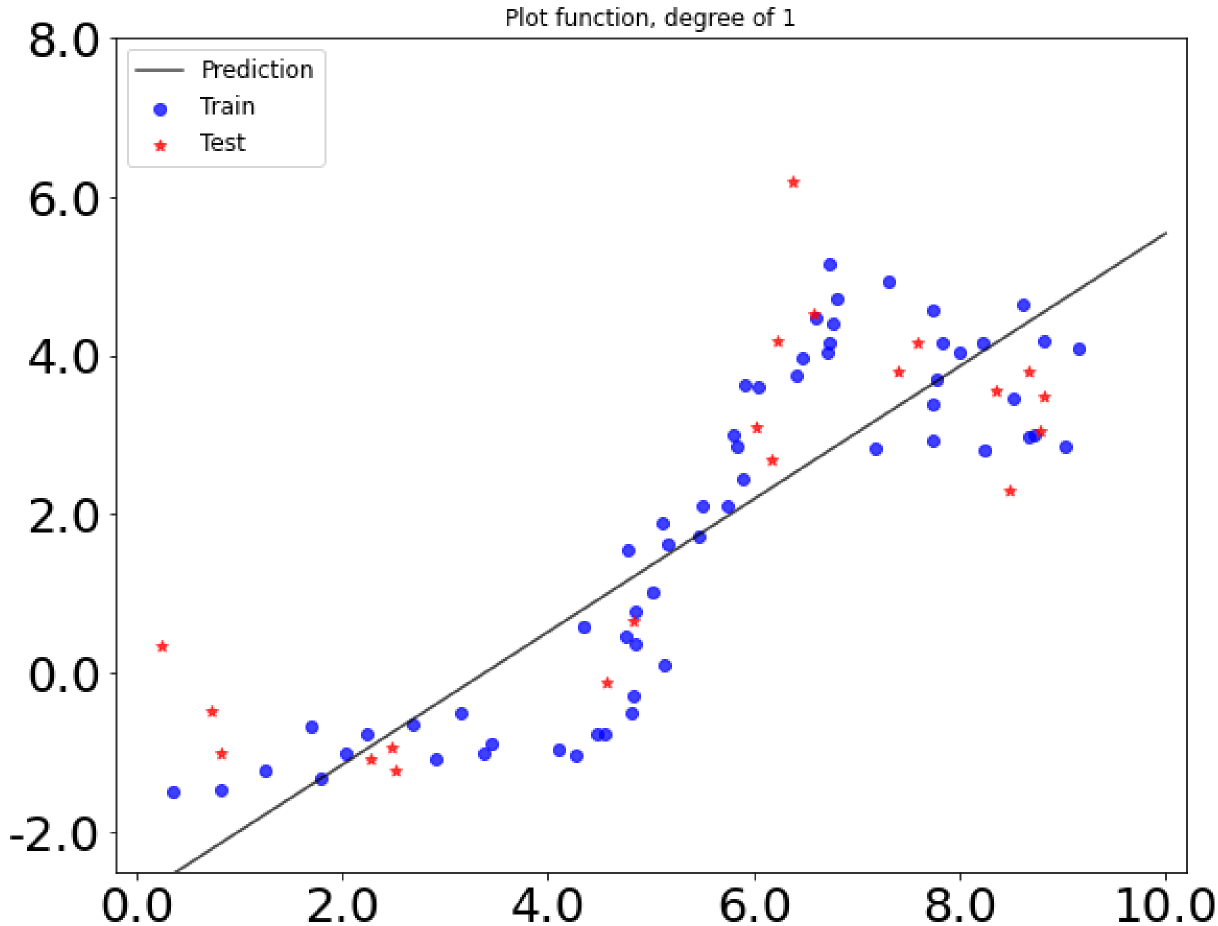
```
Mean Squared Erro (traning data) =  1.127711955609391
Mean Squared Erro (test data    ) =  2.2423492030101246
```

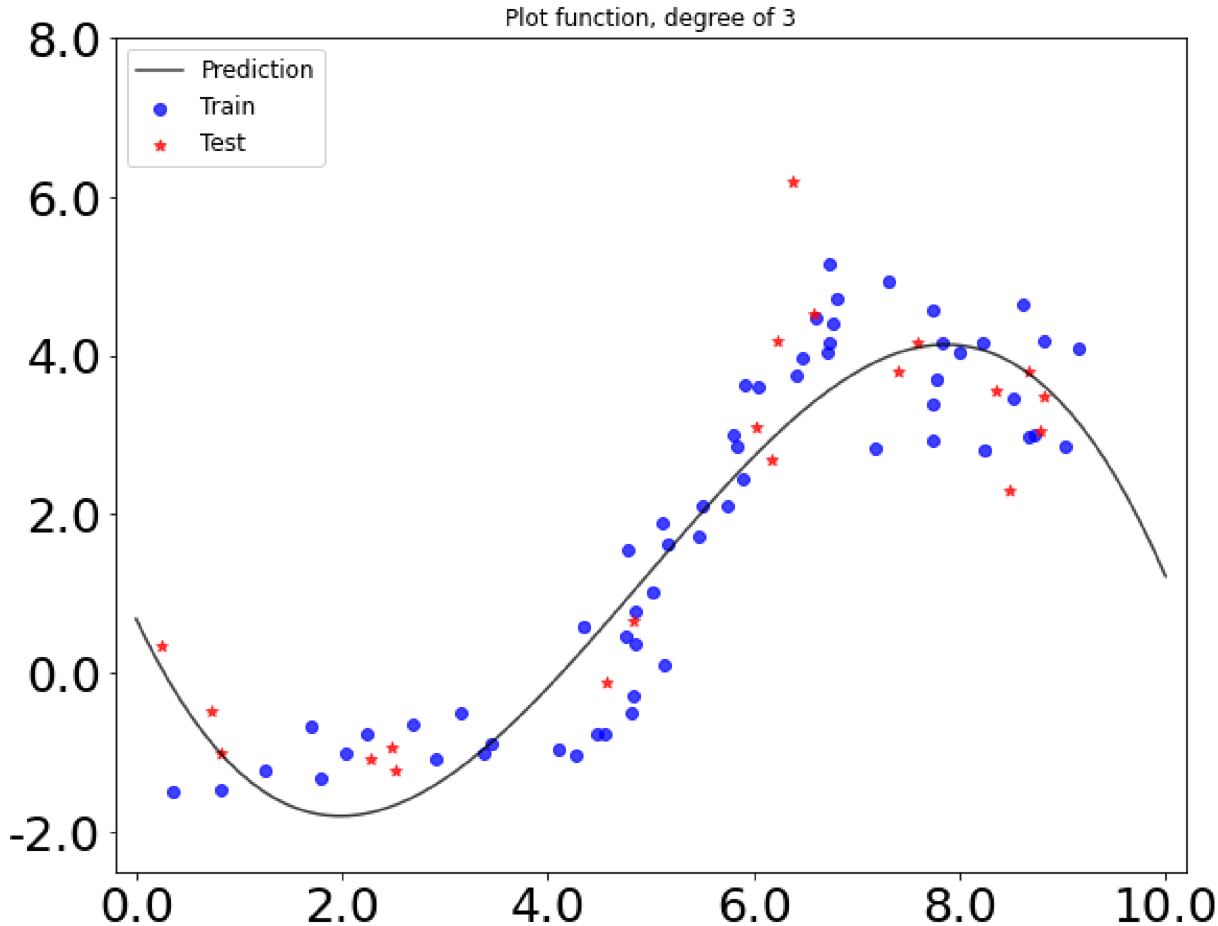
Promblem1.3

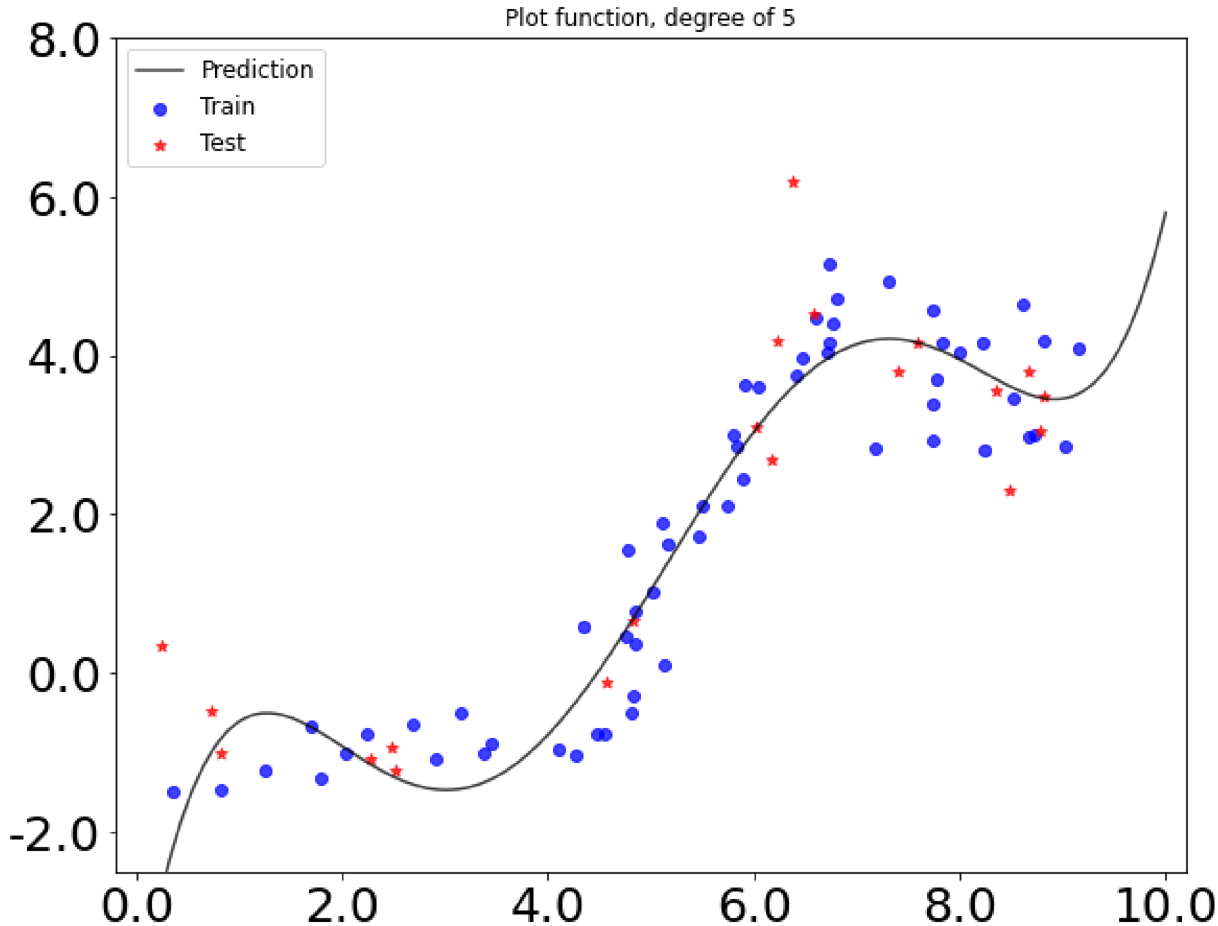
1.3.a : For each model, plot the learned prediction function f (x). (15 points)

In [254]:

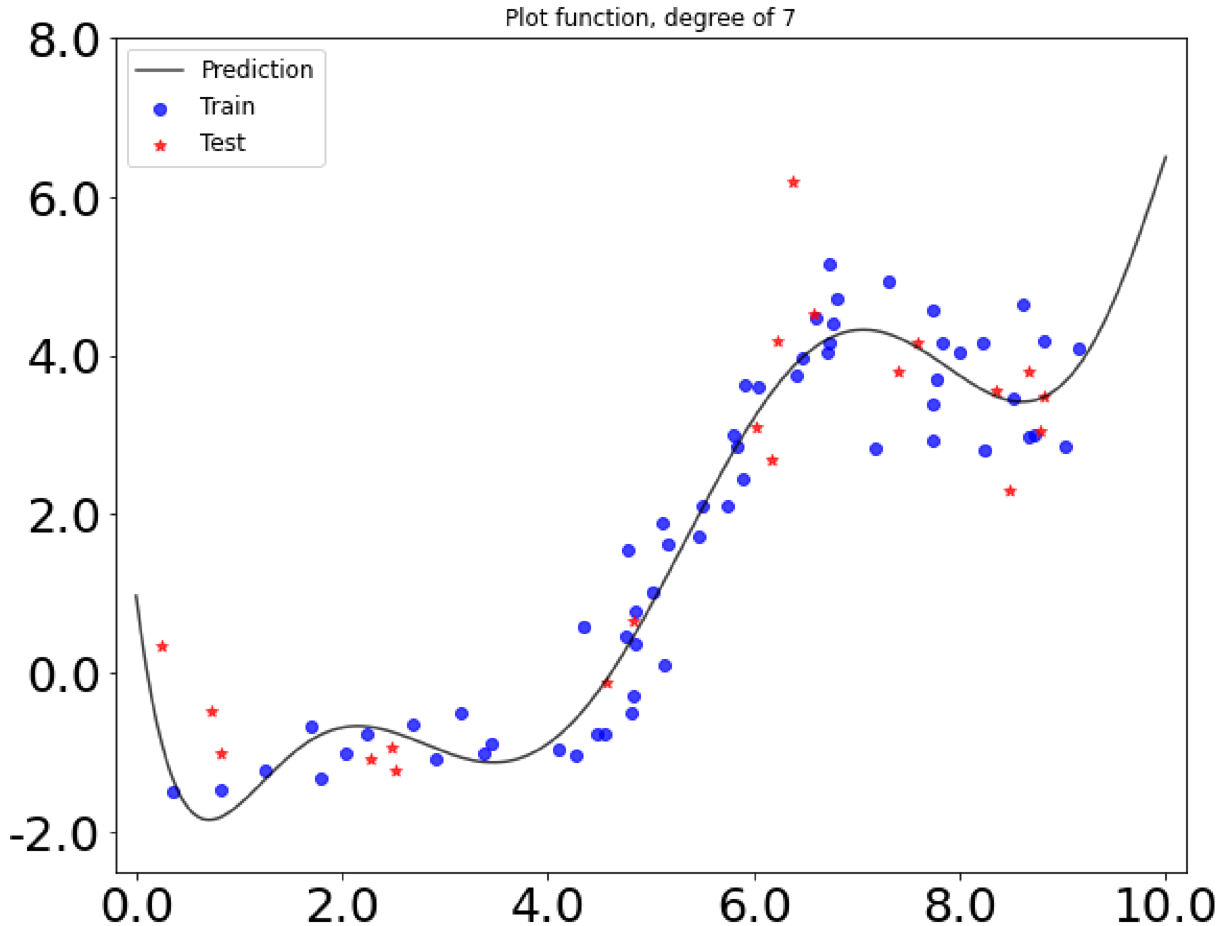
```python


degree = np.array([1,3,5,7,10,15,18])
for d in(degree):

    XtrP = ml.transforms.fpoly(Xtr, d, bias=False)
    XtrP,params = ml.transforms.rescale(XtrP)

    lr = ml.linear.linearRegress( XtrP, Ytr ) # create and train model


    # Make sure you use the currect space.
    xs = np.linspace(0, 10,200)
    xs = np.atleast_2d(xs).T

    xsP,_ = ml.transforms.rescale(ml.transforms.fpoly(xs, d, bias=False

    ys = lr.predict(xsP)

    # draw graph
    # Plotting the data
    f, ax = plt.subplots(1, 1, figsize=(10, 8))    # size of graph
    ax.scatter(Xtr, Ytr, color='blue', alpha=0.75, label='Train')  #
    ax.scatter(Xte, Yte, marker='*', color='red', alpha=0.75, label='Te
    # Also plotting the regression line
    ax.plot(xs, ys, color='black', alpha=0.75, label='Prediction')

    plt.title( "Plot function, degree of {}".format(d))

    ax.set_xlim(-0.2, 10.2)
    ax.set_ylim(-2.5, 8)
    ax.set_xticklabels(ax.get_xticks(), fontsize=25)
    ax.set_yticklabels(ax.get_yticks(), fontsize=25)


    ax.legend(fontsize=12, loc='upper left')

    plt.show()
```

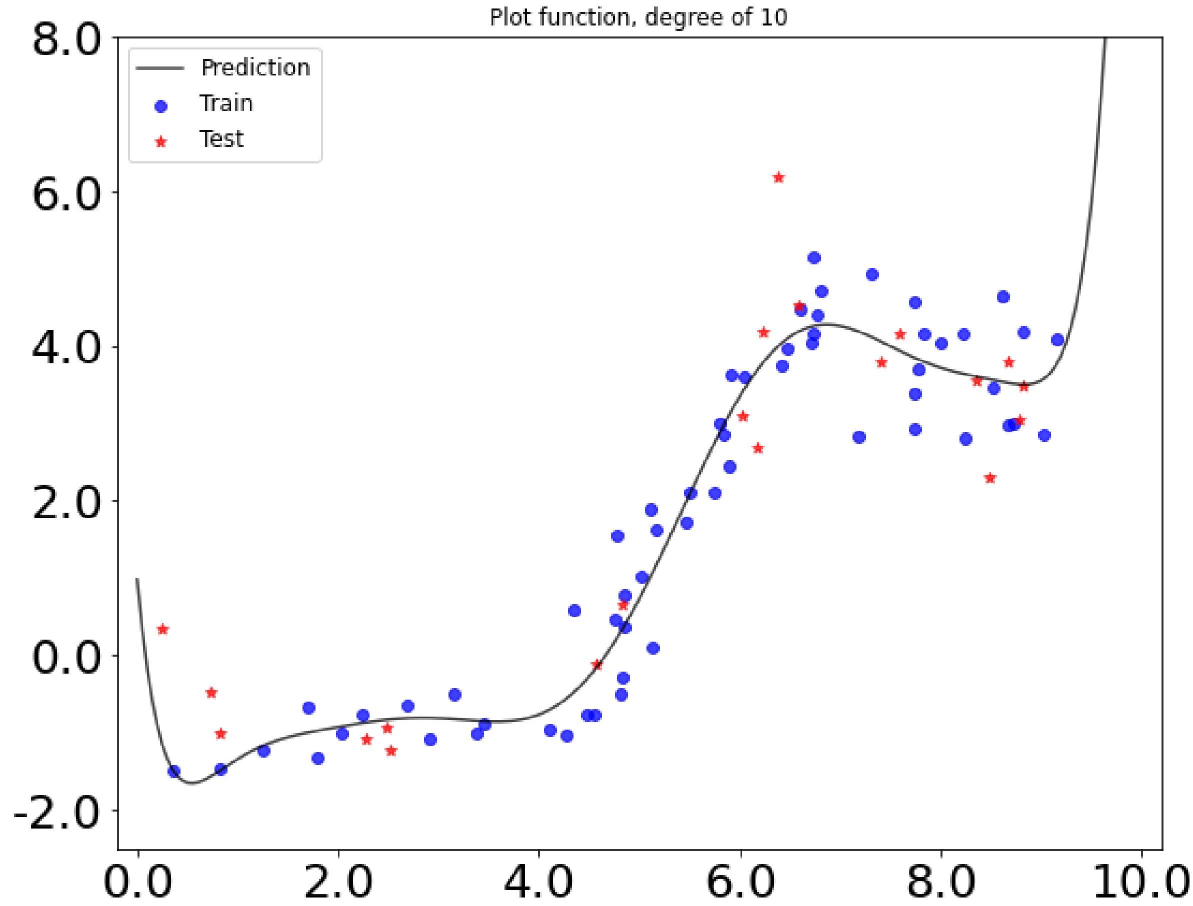

Plot function, degree of 1
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0

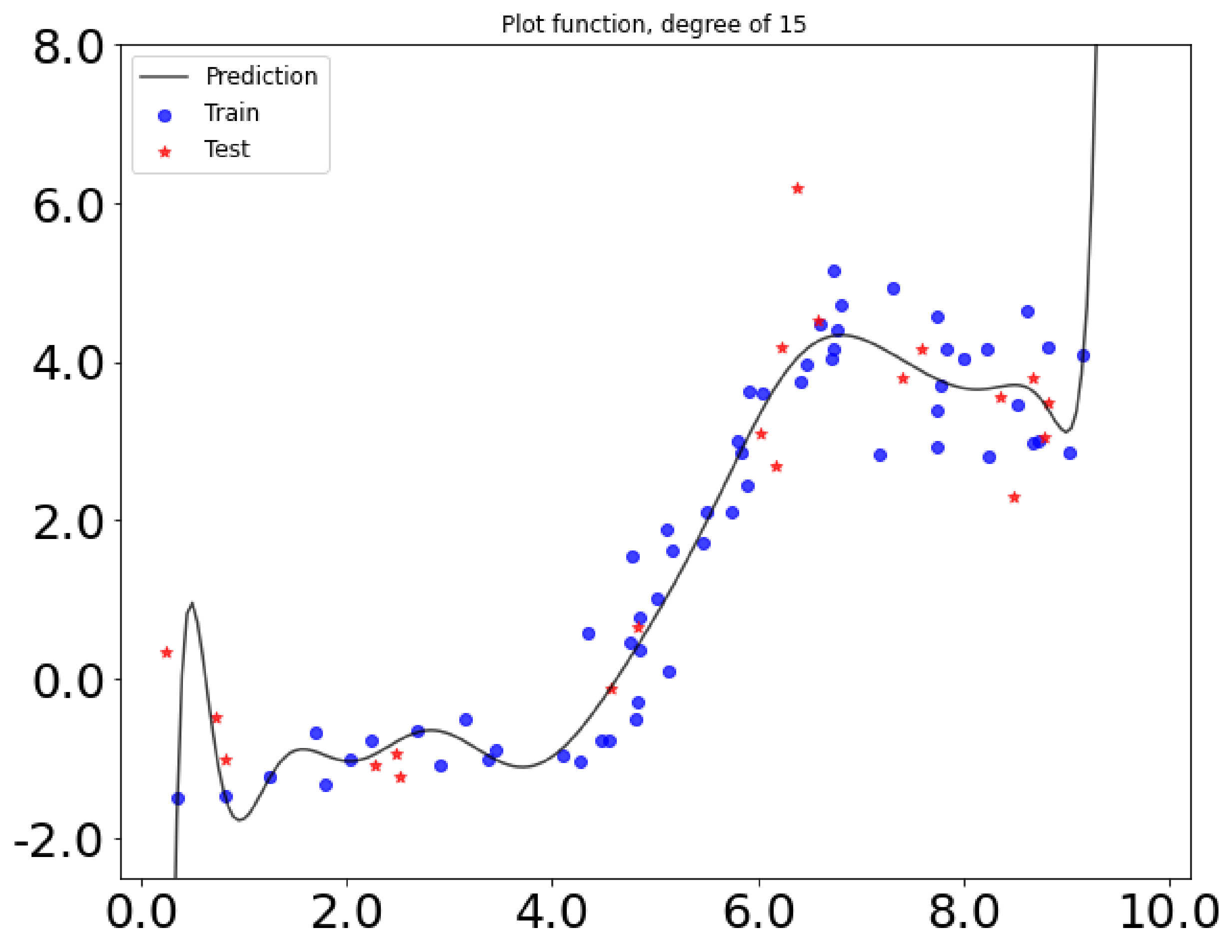

Plot function, degree of 3
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0

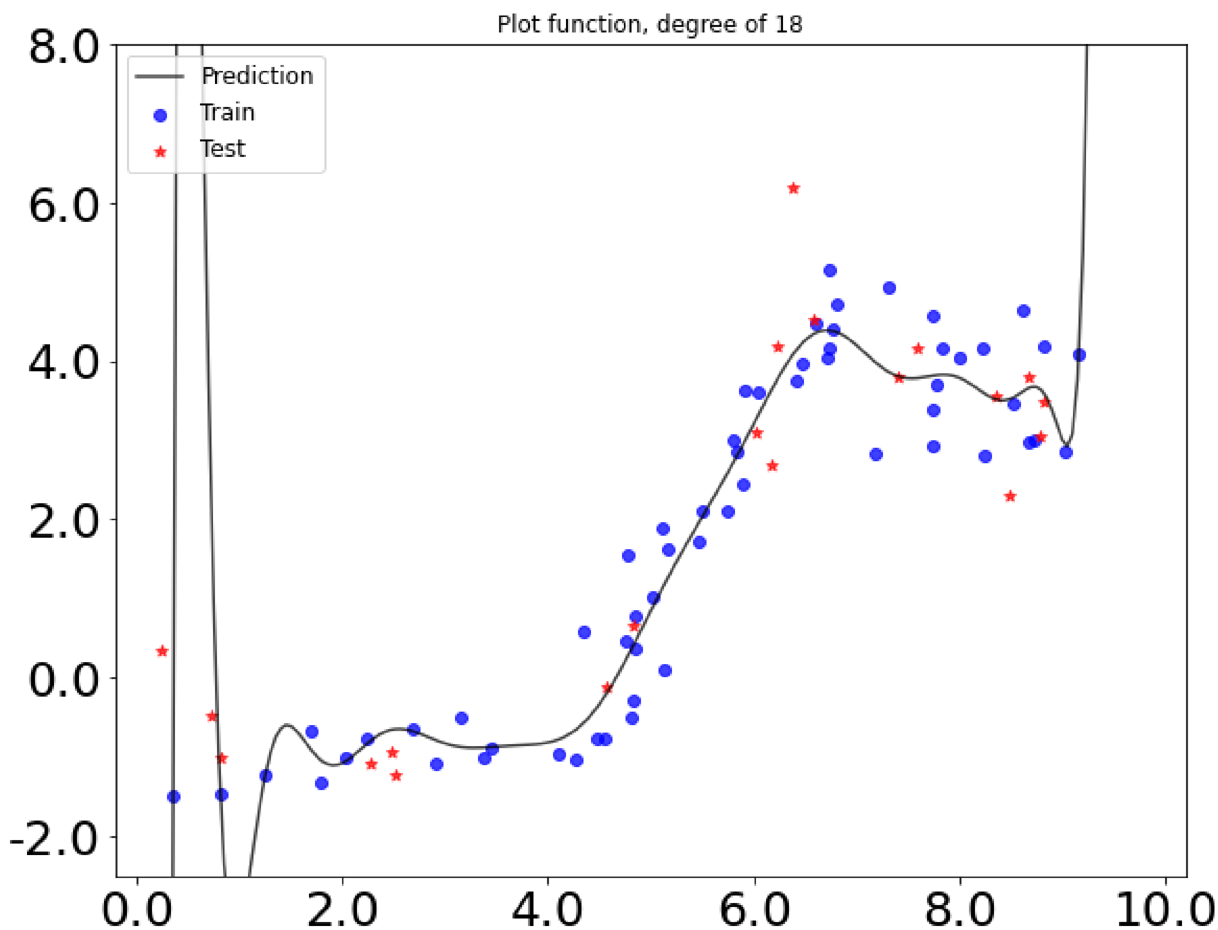


Plot function, degree of 5
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0


Plot function, degree of 7
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0


Plot function, degree of 10
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0


Plot function, degree of 15
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0

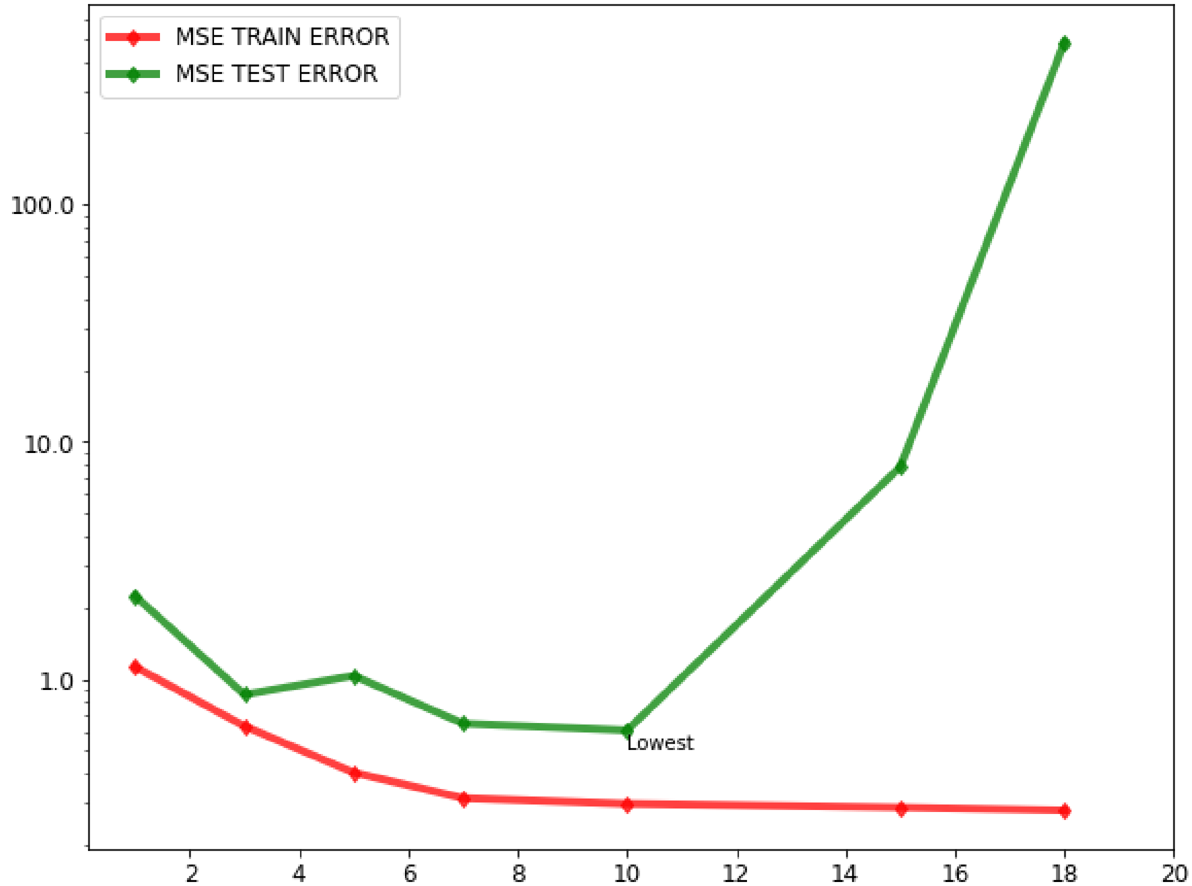

Plot function, degree of 18
Prediction
Train
Test
8.0
6.0
4.0
2.0
0.0
-2.0
0.0
2.0
4.0
6.0
8.0
10.0

1.2.b : Plot the training and test errors on a log scale (semilogy) as a function of the model degree. (10 points)

In [256]:

```
degrees = np.array([1,3,5,7,10,15,18])

mse_train_error = np.zeros(degrees.shape[0])
mse_test_error = np.zeros(degrees.shape[0])
for i,degree in enumerate(degrees):

    XtrP,params = ml.transforms.rescale(ml.transforms.fpoly(Xtr, degree
    lr = ml.linear.linearRegress(XtrP, Ytr)
    YtrHat = lr.predict(XtrP)

    XteP,_ = ml.transforms.rescale(ml.transforms.fpoly(Xte, degree, bia
    YteHat = lr.predict(XteP)

    mse_train_error[i] = MSE(Ytr, YtrHat)
    mse_test_error[i] = MSE(Yte, YteHat)


fig, ax = plt.subplots(1, 1, figsize=(10, 8)) # Create axes for single
# Plotting a line with markers where there's an actual x value.
ax.semilogy(degrees, mse_train_error, lw=4, color = "red",marker='d', a
ax.semilogy(degrees, mse_test_error, lw=4, color = "green", marker='d',

a = degrees
b = mse_train_error
c = mse_test_error
t = Table([a, b, c], names=('degree', 'mse_train_error', 'mse_test_erro
print ( t)

ax.text(10, 0.5090600748904027, 'Lowest' )
ax.set_xticks(np.arange(2, 21, 2))
ax.set_xticklabels(ax.get_xticks(), fontsize=12)
ax.set_yticklabels(ax.get_yticks(), fontsize=12)
ax.legend(fontsize=12, loc=0)

plt.show()
```

```
degree   mse_train_error      mse_test_error
------ ------------------- ------------------
     1  1.127711955609390 9  2.242349203010125
     3  0.6339652063119635 0.8616114815449999
     5  0.4042489464459056 1.0344190205632156
     7  0.3156346739892996 0.6502246079670317
    10  0.2989479796813433 0.6090600748904027
    15 0.28817930796536423  7.863359085837317
    18 0.28048505409585217  482.2803273735196
```



1.2.c : What is the mean squared error of the predictions on the training and test data? (10 points)

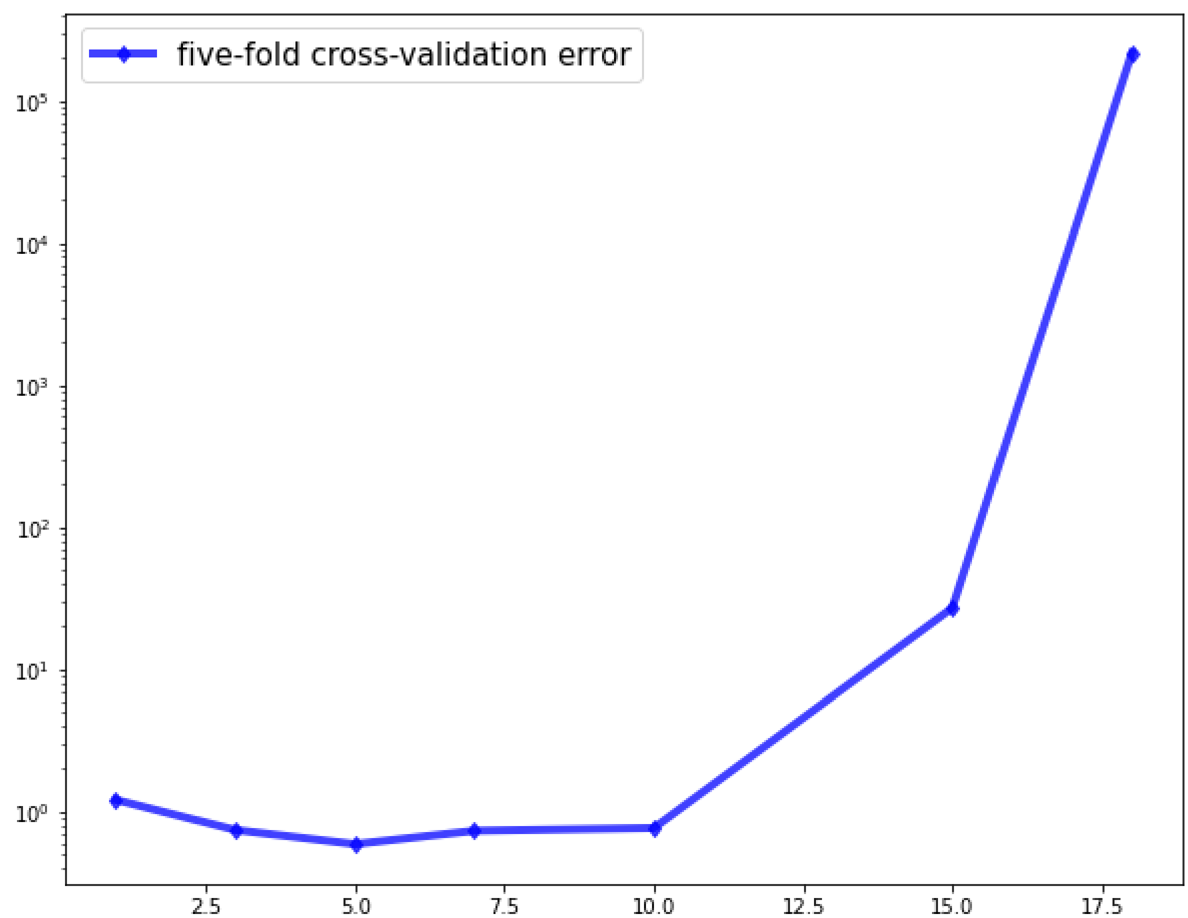
before degree of 10, our test data's performance improve. However, after the lowest point, degree of 15 and 18 are increasing which mean our test data is going worest.

Problem 3: Statement of Collaboration (5 points)

1.3.1

In [259]:

```
newDegree = [1, 3, 5, 7, 10, 15, 18]
nFolds = 5; #given

J = np.zeros(nFolds)
crsVal = []
for i, degree in enumerate(newDegree):
    for iFold in range(nFolds):#given

        Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,nFolds,iFold)

        XtiP,params = ml.transforms.rescale(ml.transforms.fpoly(Xti, de
        learner = ml.linear.linearRegress(XtiP,Yti)


        XviP, _ = ml.transforms.rescale(ml.transforms.fpoly(Xvi, degree


        J[iFold] = learner.mse(XviP, Yvi)

    crsVal.append(np.mean(J))

f, ax = plt.subplots(1, 1, figsize=(10, 8))     # size of graph
ax.semilogy(newDegree,crsVal, lw=4, color = "blue",marker='d', alpha=0.
#plt.semilogy(newDegree,mse_test_error, color = 'red')
ax.legend(fontsize=15, loc=0)
plt.show()
```



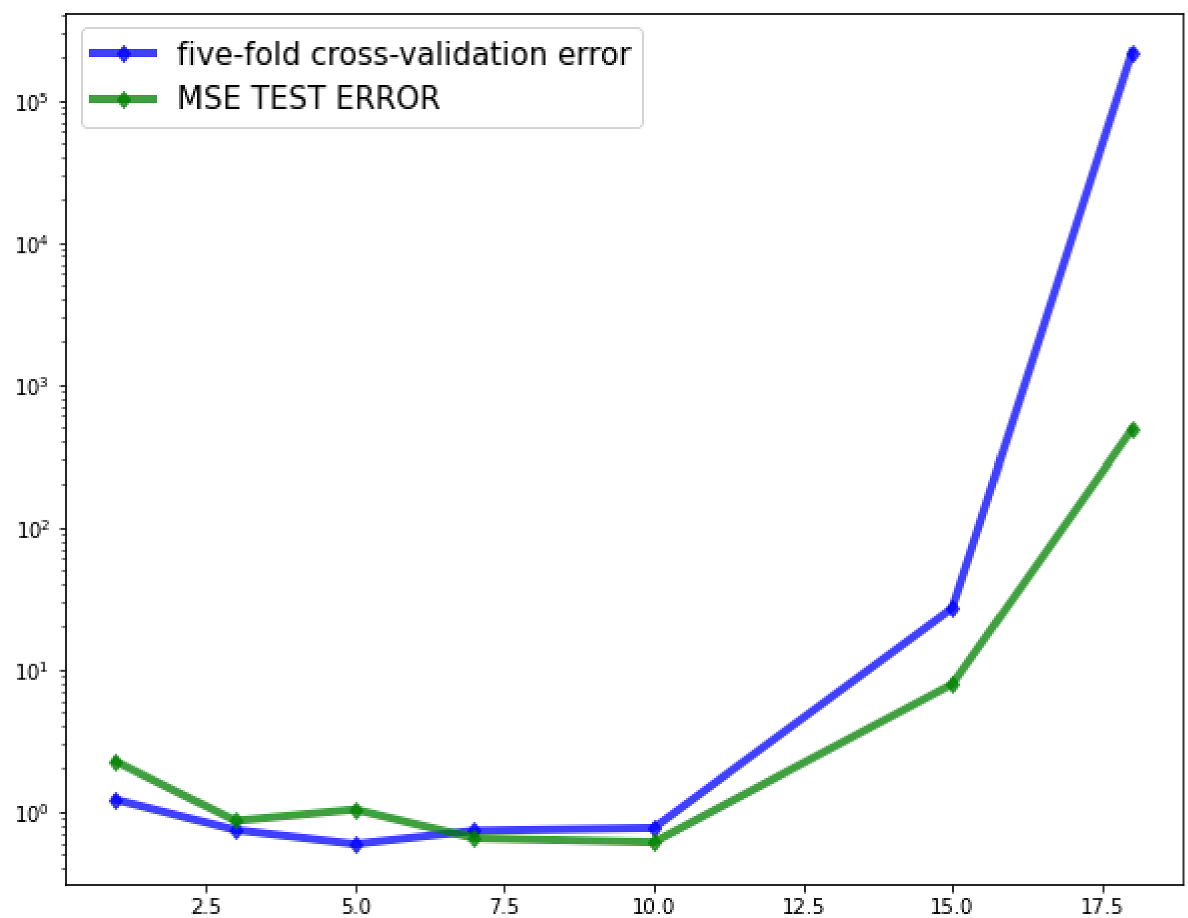
```
In [ ]: 1
```

1.3.2 : How do the MSE estimates from five-fold cross-validation compare to the MSEs evaluated on the actual test data (Problem 1)? (5 points)

In [187]:

```
a = degrees
b = mse_test_error
c = crsVal
t = Table([a, b, c], names=('degree', 'mse_train_error', 'error'))
print ( t)

f, ax = plt.subplots(1, 1, figsize=(10, 8))     # size of graph
ax.semilogy(newDegree,crsVal, lw=4, color = "blue",marker='d', alpha=0.
ax.semilogy(degrees, mse_test_error, lw=4, color = "green", marker='d',
#plt.semilogy(newDegree,mse_test_error, color = 'red')
ax.legend(fontsize=15, loc=0)
plt.show()
```

```
degree  mse_train_error          error
------ ------------------ ------------------
     1  2.242349203010125 1.2118626629641984
     3 0.8616114815449999 0.7429005752051661
     5 1.0344190205632156 0.5910703726406558
     7 0.6502246079670317 0.7335637831345124
    10 0.6090600748904027 0.7677056859101964
    15  7.863359085837317 26.989609532127144
    18  482.2803273735196 216818.07410494355
```



1.3.3 : Which polynomial degree do you recommend based on five-fold cross-validation

error? (5 points)

Based on five-fold cross-validation error, I recommand degree of 5. At this degree, we have the smallest error rate(0.5918).

1.3.4. For the degree that you picked in step 3, plot (with semilogy) the cross-validation error as the number of folds is varied from nFolds = 2, 3, 4, 5, 6, 10, 12, 15. What pattern do you observe, and how do you explain why it occurs? (15 points)

In [ ]:

```
nFolds = [2, 3, 4, 5, 6, 10, 12, 15]
myDegree = 5; #given

J = np.zeros(myDegree)
crsVal = []
for i, degree in enumerate(myDegree):
    for iFold in range(nFolds):#given

        Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,nFolds,iFold)

        XtiP,params = ml.transforms.rescale(ml.transforms.fpoly(Xti, de
        learner = ml.linear.linearRegress(XtiP,Yti)


        XviP, _ = ml.transforms.rescale(ml.transforms.fpoly(Xvi, degree


        J[iFold] = learner.mse(XviP, Yvi)

    crsVal.append(np.mean(J))

f, ax = plt.subplots(1, 1, figsize=(10, 8))     # size of graph
ax.semilogy(newDegree,crsVal, lw=4, color = "blue",marker='d', alpha=0.
#plt.semilogy(newDegree,mse_test_error, color = 'red')
ax.legend(fontsize=15, loc=0)
plt.show()
```

Problem 3: Statement of Collaboration (5 points)

- Piazza Question : question@222
- peter Park : Discuss about defination about MSE