hw2

April 26, 2018

1 CS 273P Homework 2

In [126]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
np.random.seed(0)
```

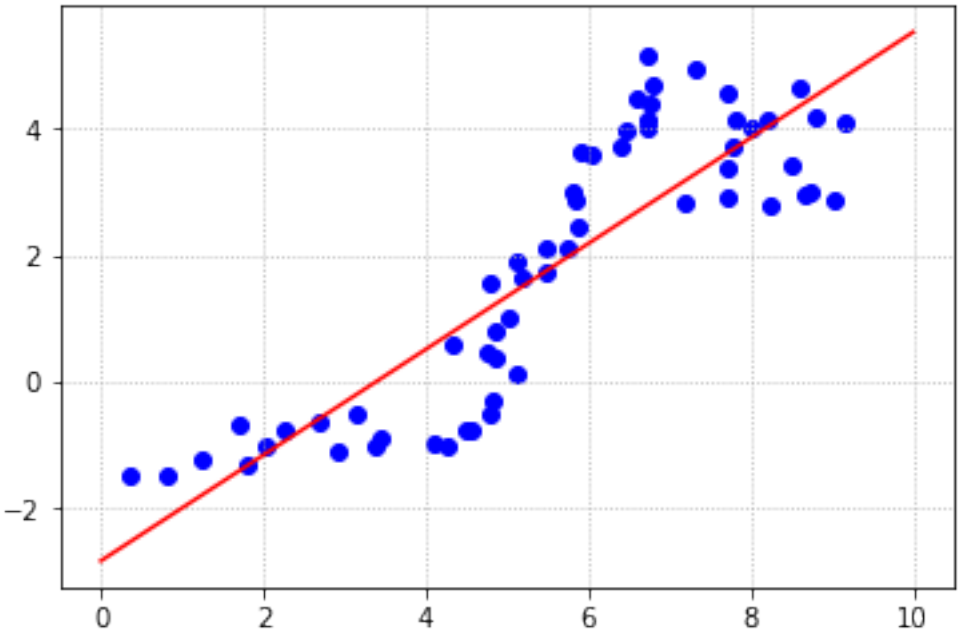
Problem 1 (a)

In [127]:
```
data=np.genfromtxt("data/curve80.txt",delimiter=None)
X=data[:,0]
X = X[:,np.newaxis]
Y = data[:,1]
Xtr,Xte,Ytr,Yte = ml.splitData(X,Y,0.75)
```

Problem 1 (b)

In [128]:
```
lr = ml.linear.linearRegress( Xtr, Ytr );
xs = np.linspace(0,10,200);
xs = xs[:,np.newaxis]
ys = lr.predict( xs );
plt.scatter(Xtr,Ytr,c='b')
plt.plot(xs,ys,'r')
plt.grid(linestyle='dotted')
ax=plt.axis()
plt.show()

print("Theta: ",lr.theta)
YhatTrain=lr.predict(Xtr)
YhatTest=lr.predict(Xte)
def mse(Yhat,Y):
    mse=0
    for i in range(0,len(Y)):
        mse=mse+((Y[i]-Yhat[i])*(Y[i]-Yhat[i]))
    return mse/len(Y)
print("Mean squared error on Training data: ",mse(YhatTrain,Ytr))
print("Mean squared error on Test data: ",mse(YhatTest,Yte))
```



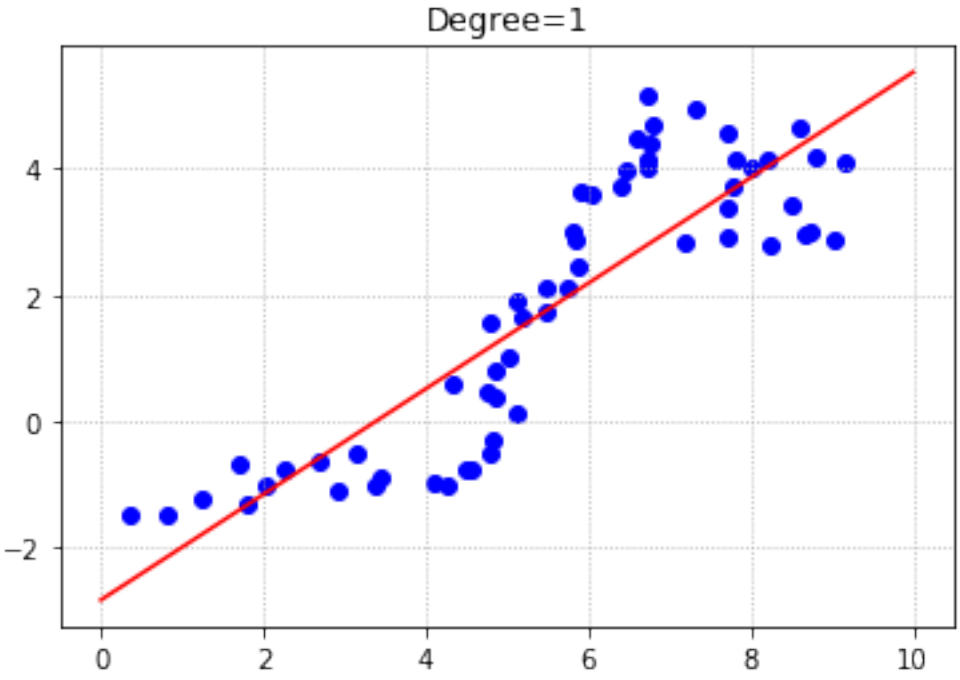

```
Theta:  [[-2.82765049  0.83606916]]
Mean squared error on Training data:  [1.12771196]
Mean squared error on Test data:  [2.2423492]
```

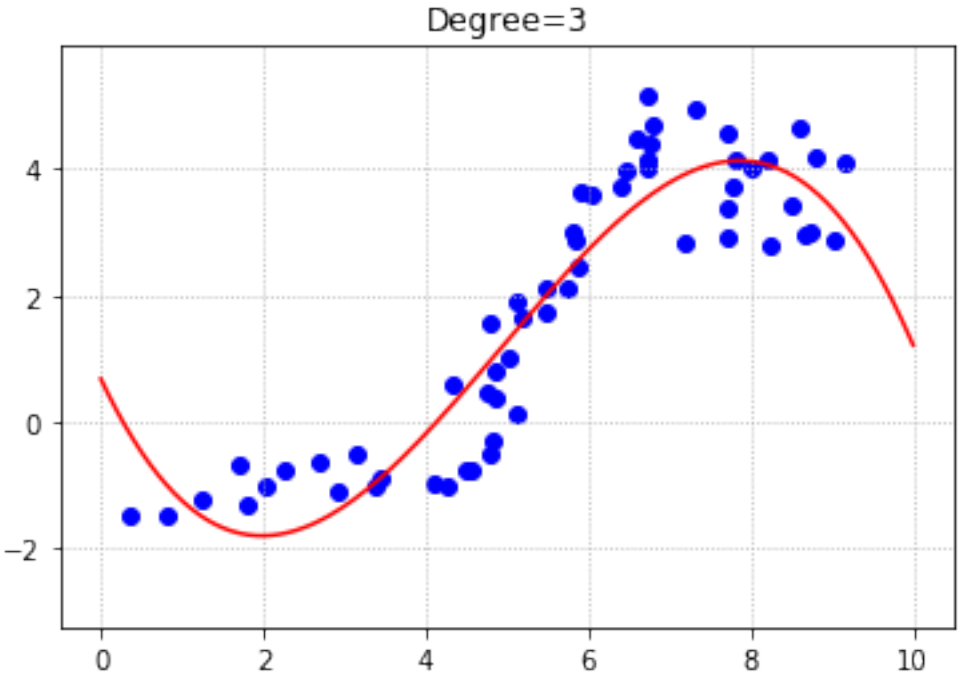
Problem 1(c)

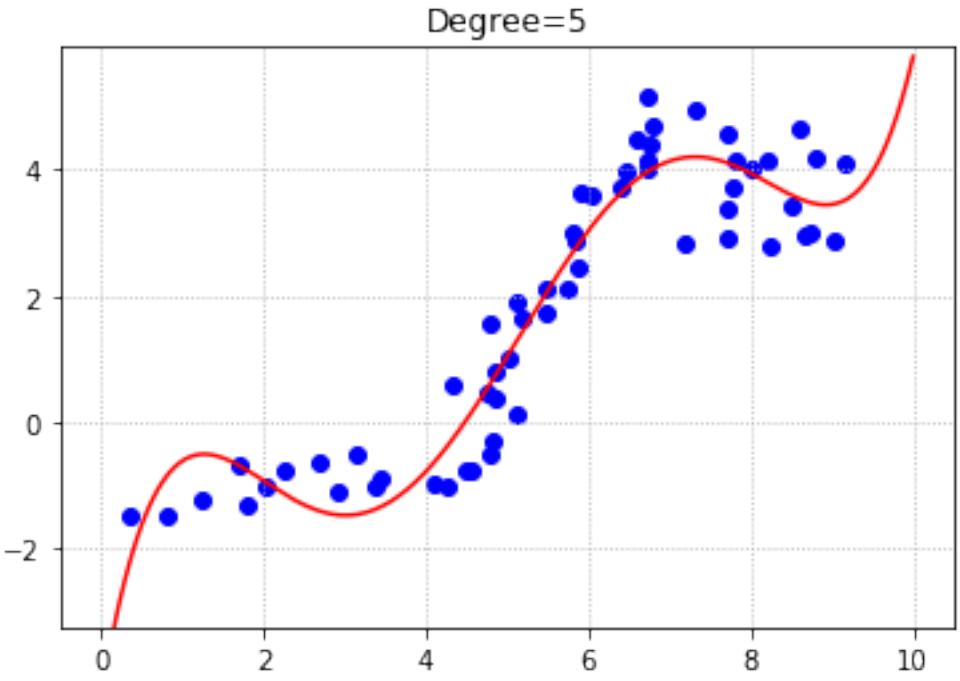
1) Plots of Learned prediction functions

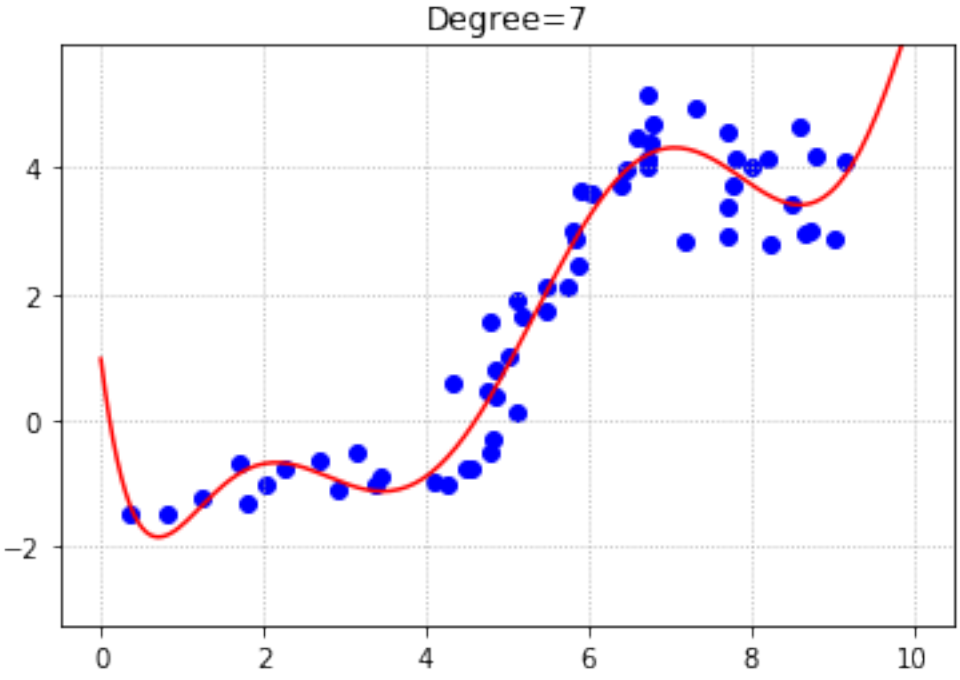
```
In [129]: mseTrain=[]
          mseTest=[]
          degrees=[1,3,5,7,10,18]
          for degree in degrees:
              XtrPoly=ml.transforms.fpoly(Xtr,degree,bias=False);
              XtrPoly,params=ml.transforms.rescale(XtrPoly)
              lr=ml.linear.linearRegress(XtrPoly,Ytr)
              XteP,_=ml.transforms.rescale(ml.transforms.fpoly(Xte,degree,False))
              Phi = lambda X: ml.transforms.rescale( ml.transforms.fpoly(X, degree,False), para
              YhatTrain = lr.predict( Phi(Xtr) ); # predict on training data
              YhatTest = lr.predict( Phi(Xte) ); # predict on test data
              ys=lr.predict(Phi(xs))
              mseTrain.append(mse(YhatTrain,Ytr))
              mseTest.append(mse(YhatTest,Yte))
              plt.axis(ax)
              plt.title('Degree='+str(degree))
              plt.scatter(Xtr,Ytr,c='b')
```

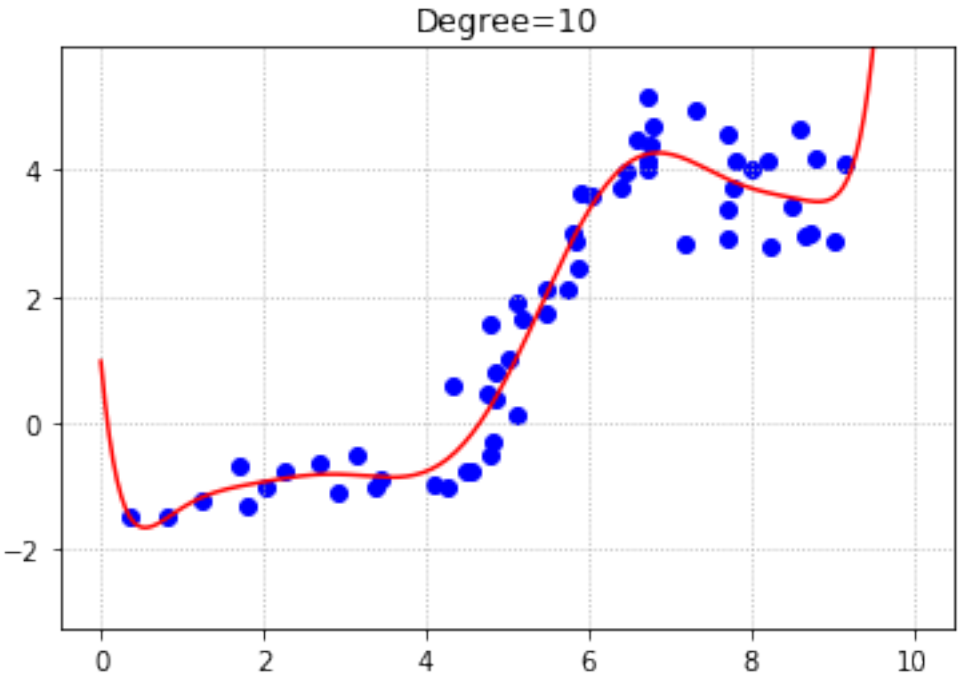
```
plt.plot(xs,ys,'r')
plt.grid(linestyle='dotted')
plt.show()
```

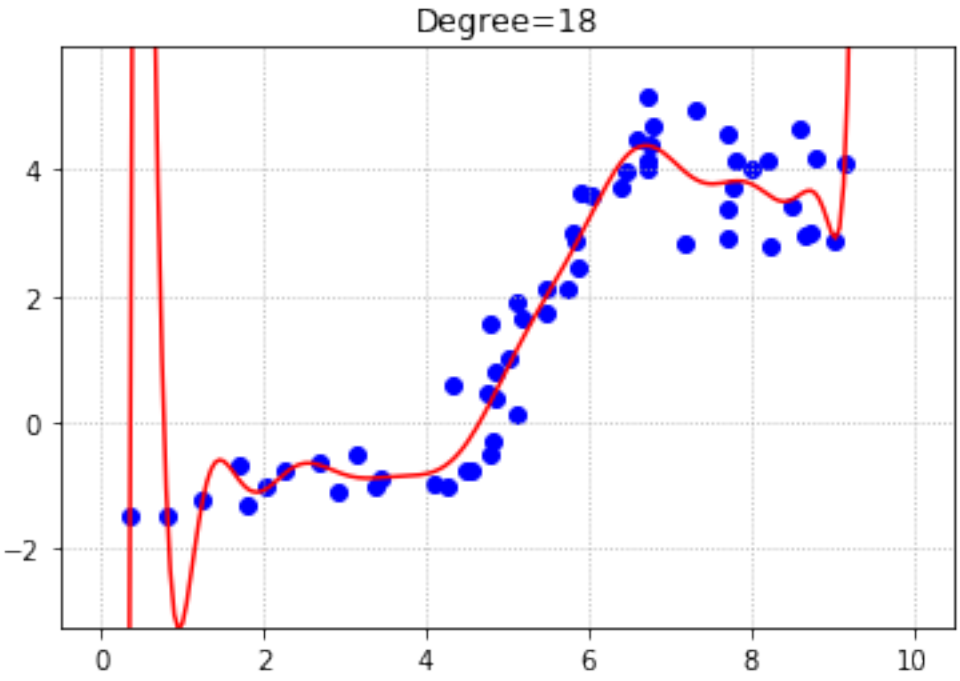





Degree=5
4
2
0
−2
0
2
4
6
8
10

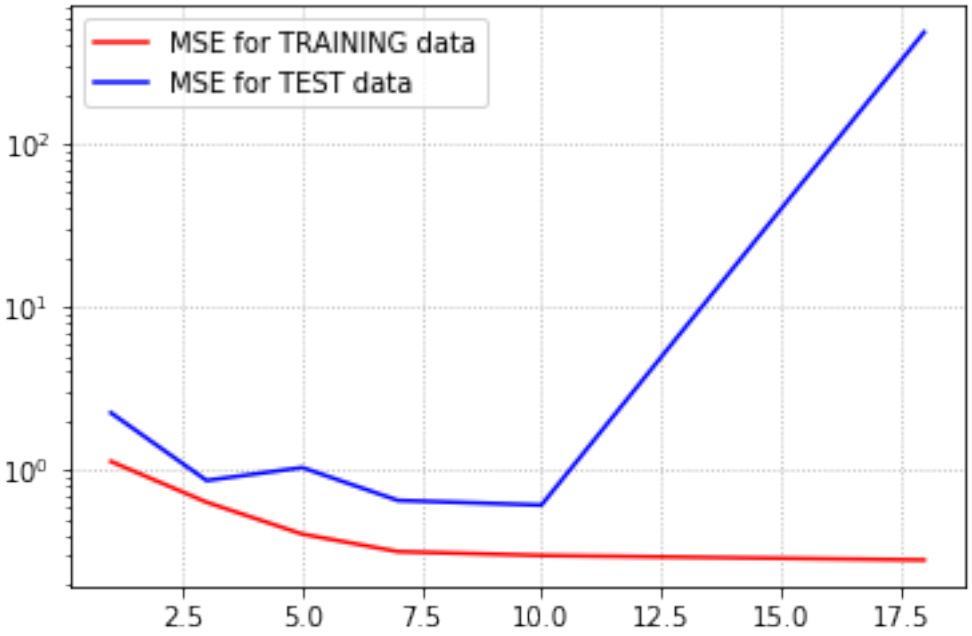

Degree=7
4
2
0
−2
0
2
4
6
8
10





2) Plots of training and test errors

```
In [130]: plt.semilogy(degrees,mseTrain,'r',label='MSE for TRAINING data')
          plt.semilogy(degrees,mseTest,'b',label='MSE for TEST data')
          plt.legend(loc='upper left')
          plt.grid(linestyle='dotted')
          plt.show()
```



Problem 2) Cross-Validation

A) MSE values for each degree.

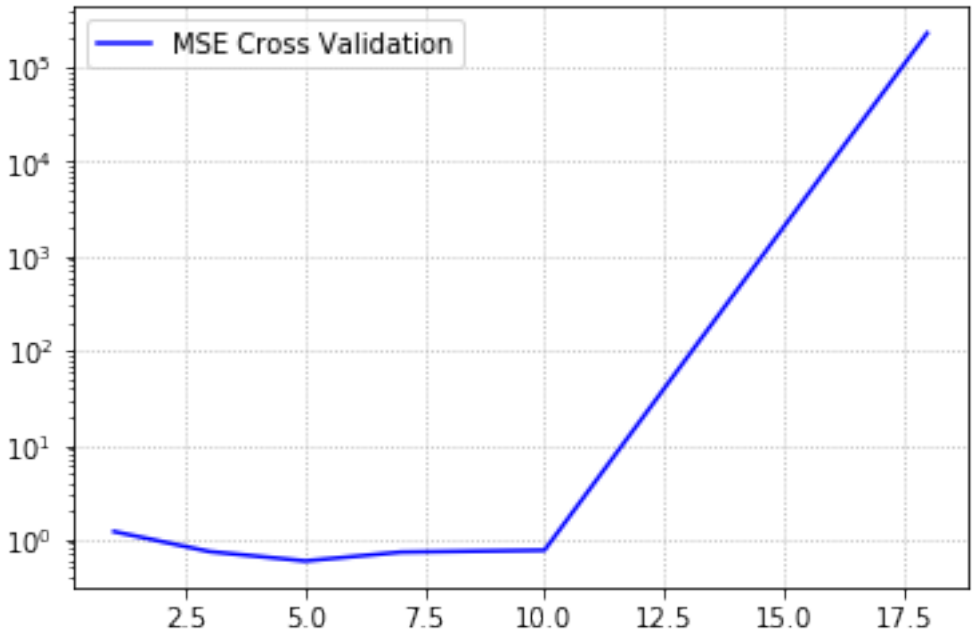
```
In [131]: degrees=[1,3,5,7,10,18]
          mseCrossValidation=[]
          for degree in degrees:
              #Phi = lambda X: ml.transforms.rescale( ml.transforms.fpoly(X, degree,False), pa
              J=[]
              nFolds = 5;
              for iFold in range(nFolds):
                  Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,nFolds,iFold); # take ith data bl
                  XtiP=ml.transforms.fpoly(Xti,degree,False);
                  XviP=ml.transforms.fpoly(Xvi,degree,False);
                  XtiP,params=ml.transforms.rescale(XtiP)
                  XviP,_=ml.transforms.rescale(XviP,params)
                  learner = ml.linear.linearRegress(XtiP,Yti)

                  #YviHat=learner.predict(Phi(Xvi))
                  J.append(learner.mse(XviP,Yvi))
              mseCrossValidation.append(np.mean(J))
          print(mseCrossValidation)
```

```
[1.2118626629641986, 0.7429005752051656, 0.5910703726407606, 0.7335637831345172, 0.767705687542
```

B) Plot of degree (x-axis) vs cross validation error (y-axis)

```
In [132]: plt.semilogy(degrees,mseCrossValidation,'b',label='MSE Cross Validation')
          plt.legend(loc='upper left')
          plt.grid(linestyle='dotted')
          plt.show()
```



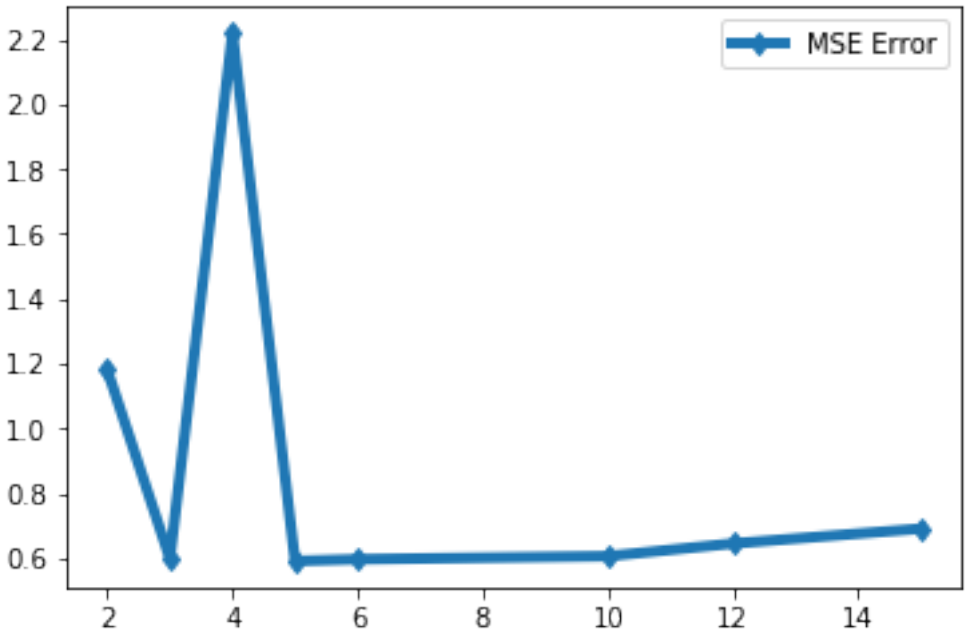
C) Degree with minimum cross-validation error is 5.

Plot for MSE using cross validation, for degree 5 :

```
In [133]: J=[]
          mse_error=[]
          nFoldsArray=np.array([2,3,4,5,6,10,12,15])
          for i,Folds in enumerate(nFoldsArray):
              J.clear()
              for iFold in range(nFoldsArray[i]):
                  Xti,Xvi,Yti,Yvi = ml.crossValidate(Xtr,Ytr,nFoldsArray[i],iFold) # take ith
                  XtiP=ml.transforms.fpoly(Xti,5,False);
                  XviP=ml.transforms.fpoly(Xvi,5,False);
                  XtiP,params=ml.transforms.rescale(XtiP)
                  XviP,_=ml.transforms.rescale(XviP,params)
                  learner = ml.linear.linearRegress(XtiP,Yti)
```

```
            YviHat=learner.predict(XviP)
            J.append(learner.mse(XviP,Yvi))
        mse_error.append(np.mean(J))
    print("MSE Error: ")
    print(mse_error)
    f,ax=plt.subplots(1,1)
    ax.plot(nFoldsArray,mse_error,lw=4,marker='d',label='MSE Error')
    ax.legend()
    plt.show()
```

```
MSE Error:
[1.1795458641313061, 0.5984555010978058, 2.219526156063488, 0.5910703726407606, 0.596338005001
```



D) Plot of MSE estimated from Cross Validation vs MSE evaluated for Actual Test data

```
In [134]: # Mean squared Error with degree 5 without folds:
          print("Mean squared Error with degree 5 without folds(on Actual test data:")
          print(mseTest[2])
          print("Mean squared Error with degree 5 with 5-folds:")
          print(mseCrossValidation[2])
```

```
Mean squared Error with degree 5 without folds(on Actual test data:
[1.03441902]
Mean squared Error with degree 5 with 5-folds:
0.5910703726407606
```

In [135]:
```
plt.semilogy(degrees,mseTest,'r',label='MSE Test Data')
plt.semilogy(degrees,mseCrossValidation,'g',label='MSE Cross Validation')
plt.legend(loc='upper left')
plt.grid(linestyle='dotted')
plt.show()
```