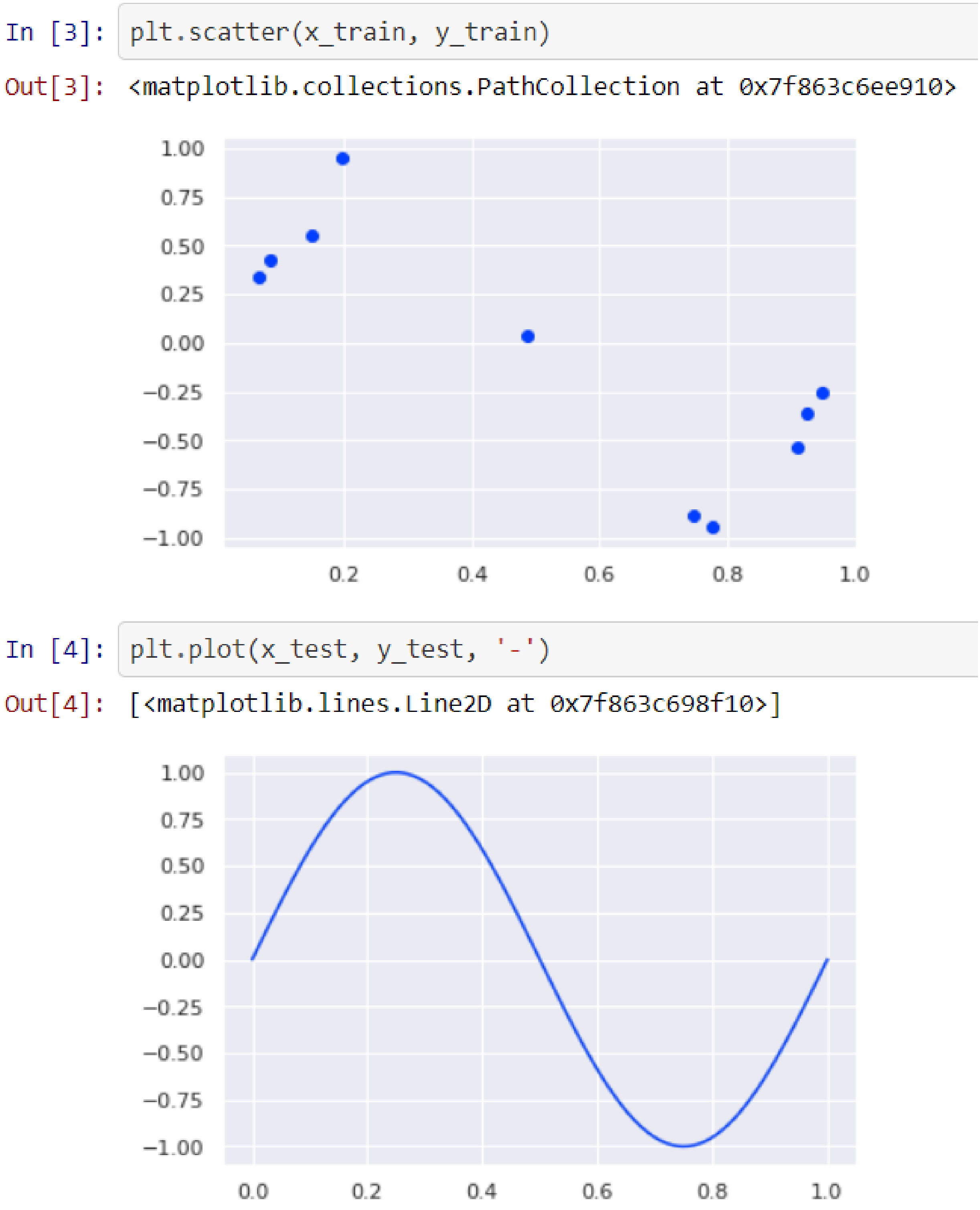
Week 2



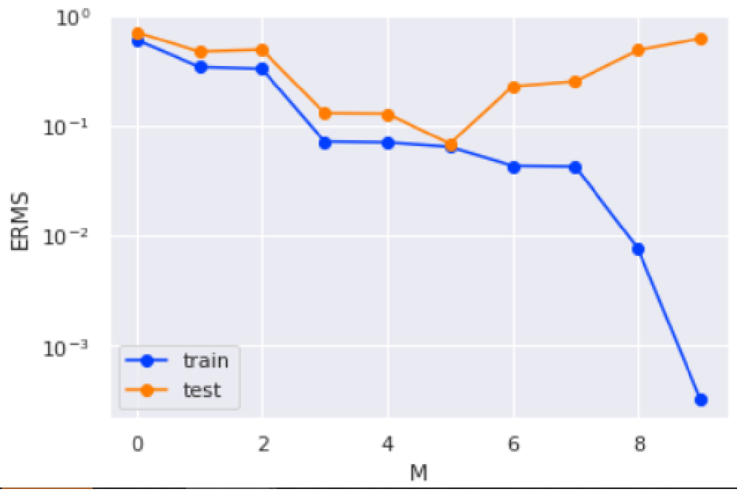
```
In [3]: plt.scatter(x_train, y_train)
Out[3]: <matplotlib.collections.PathCollection at 0x7f863c6ee910>
```

```
In [4]: plt.plot(x_test, y_test, '-')
Out[4]: [<matplotlib.lines.Line2D at 0x7f863c698f10>]
```

1.5) Test the model

In [9]:

```
def test_all(start_M, end_M, x_train, y_train, x_test, y_test):

    results_train = []
    results_test = []
    all_weights = []

    for M in range(start_M, end_M + 1):
        weights = optimial_weights(x_train, y_train, M)
        all_weights.append(weights)
        error_train = erms(weights, x_train, y_train)
        error_test = erms(weights, x_test, y_test)
        results_train.append(error_train)
        results_test.append(error_test)
    return results_train, results_test, all_weights

r_tr, r_tt, all_weights = test_all(0, 9, x_train, y_train, x_test, y_test)

plt.plot(list(range(0, 10)), r_tr, '-o', label='train')
plt.plot(list(range(0, 10)), r_tt, '-o', label='test')
plt.xlabel('M')
plt.ylabel('ERMS')
plt.legend()
plt.yscale('log')
```



Weights table for different `M`

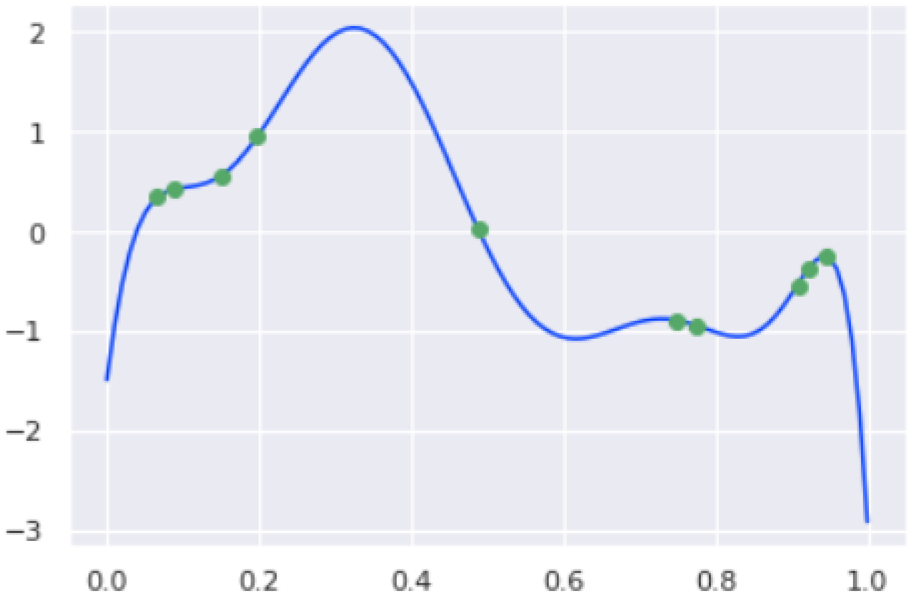
```
In [10]: print(pd.DataFrame(all_weights))
```

```
          0          1            2            3             4  \
0 -0.067444        NaN          NaN          NaN           NaN   
1  0.674906  -1.403250          NaN          NaN           NaN   
2  0.870601  -2.937208     1.521622          NaN           NaN   
3 -0.348533  11.654727   -33.094539    22.088529           NaN   
4 -0.274931  10.447965   -27.877273    14.161034      3.897826   
5  0.052313   2.957062    24.077616  -126.364847    164.358999   
6  1.343704 -29.263446   282.780632 -1019.109579   1661.058635   
7  1.660035 -39.467094   401.667411 -1667.834803   3460.305311   
8 -2.482754 104.700486 -1467.099640 10140.618109 -36828.570753   
9 -1.490888  61.009355  -711.028213  3455.126278  -3628.441682   

              5             6              7             8            9  
0           NaN           NaN            NaN           NaN          NaN  
1           NaN           NaN            NaN           NaN          NaN  
2           NaN           NaN            NaN           NaN          NaN  
3           NaN           NaN            NaN           NaN          NaN  
4           NaN           NaN            NaN           NaN          NaN  
5    -65.103637           NaN            NaN           NaN          NaN  
6  -1272.405596    376.039116            NaN           NaN          NaN  
7  -3885.775599   2271.534475    -541.872761           NaN          NaN  
8  74281.956331 -84068.960375   50041.296017 -12202.871583          NaN  
9 -22214.099542  82834.690416 -118767.661098  79903.514042 -20934.53309  
```

Estimated curve for `M=9` (same as the amount of data points)

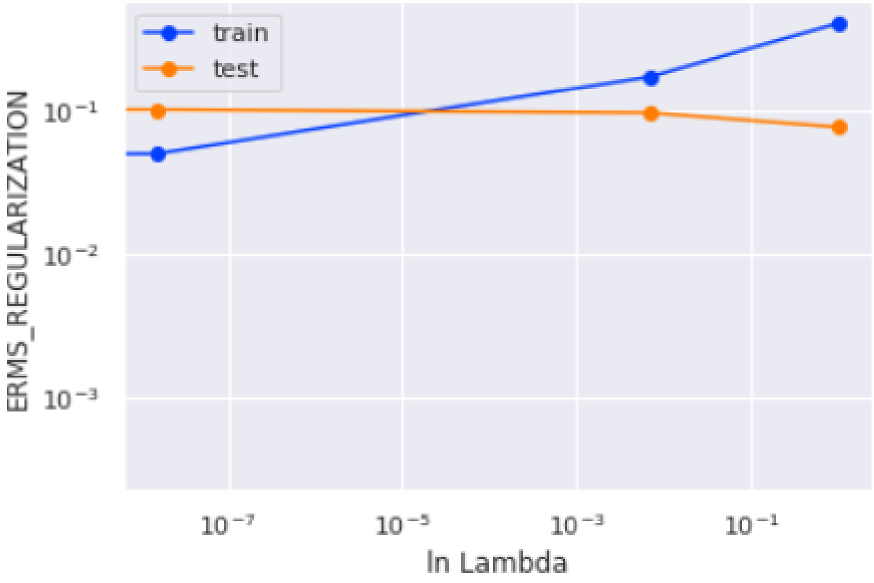
In [11]:
```
plt.plot(x_test, list(map(lambda x: linear(x, optimia
plt.plot(x_train, y_train, 'og')
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x7f863c45e310>]
```



2.3) Test with regularization

In [15]:
```
def test_all_regularization(ls, M, x_train, y_train, x_

    results_train = []
    results_test = []
    all_weights = []

    for l in ls:
        weights = optimial_weights_regularization(x_tra
        all_weights.append(weights)
        error_train = erms_regularization(weights, x_tr
        error_test = erms_regularization(weights, x_tes
        results_train.append(error_train)
        results_test.append(error_test)
    return results_train, results_test, all_weights

ls = [0, exp(-18), exp(-5), exp(0)]

r_tr_r, r_tt_r, all_weights_r = test_all_regularizatio

plt.plot(ls, r_tr_r, '-o', label='train')
plt.plot(ls, r_tt_r, '-o', label='test')
plt.xlabel('ln Lambda')
plt.ylabel('ERMS_REGULARIZATION')
plt.legend()
plt.yscale('log')
plt.xscale('log')
```

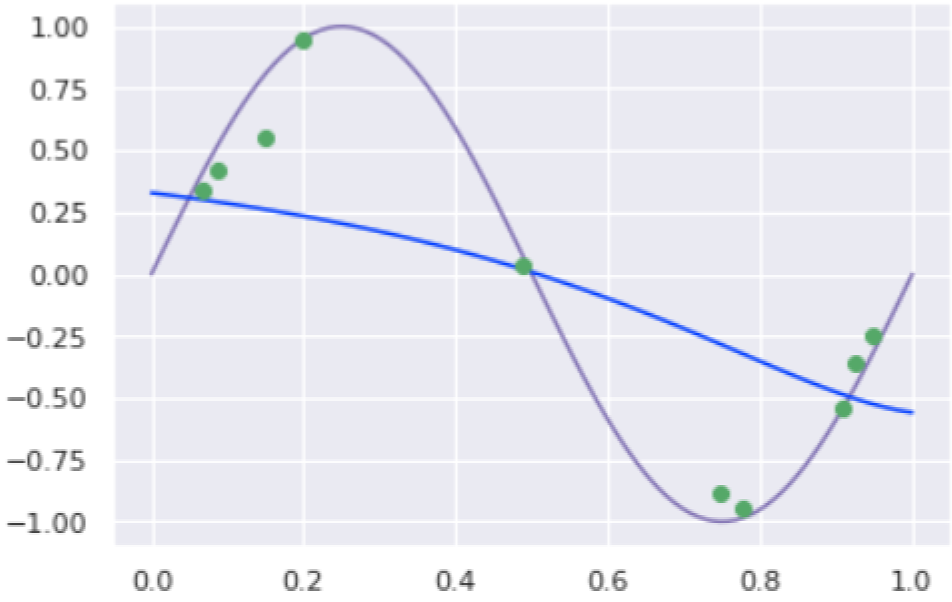


Weights for `M=9` with regularization terms `0`, `exp(-18)`, `exp(-5)`, `exp(0)`

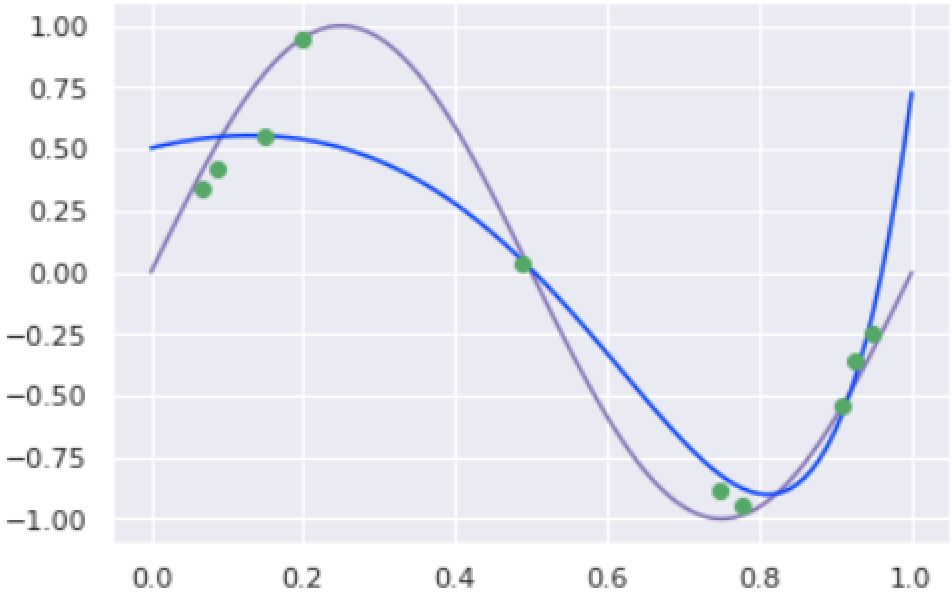
```
In [16]: print(pd.DataFrame(np.transpose(all_weights_r)))
```

```
               0           1         2         3
0      -1.490888    0.699011  0.503543  0.328543
1      61.009355  -11.795681  0.743539 -0.389393
2    -711.028213  121.043844 -2.425527 -0.353466
3    3455.126278 -335.150175 -1.908122 -0.232935
4   -3628.441682  217.987058 -0.795824 -0.127092
5  -22214.099542  178.622535  0.106829 -0.046211
6   82834.690416  -87.722822  0.719266  0.013218
7 -118767.661098 -148.839188  1.096707  0.055948
8   79903.514042  -12.387886  1.302133  0.085993
9  -20934.533090   78.207422  1.385740  0.106489
```

```
In [18]: plot_by_lambda(exp(0))
```

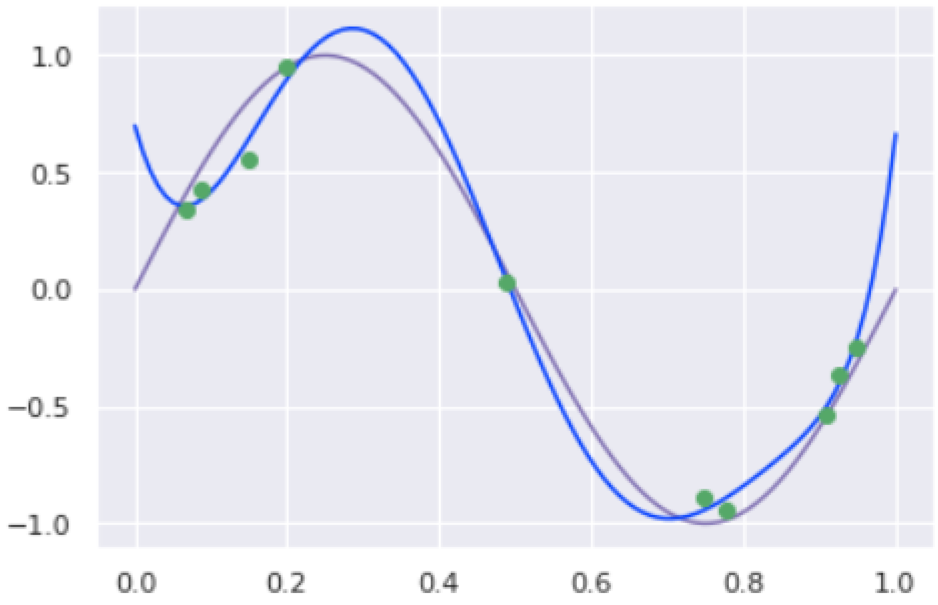


```
In [19]: plot_by_lambda(exp(-5))
```



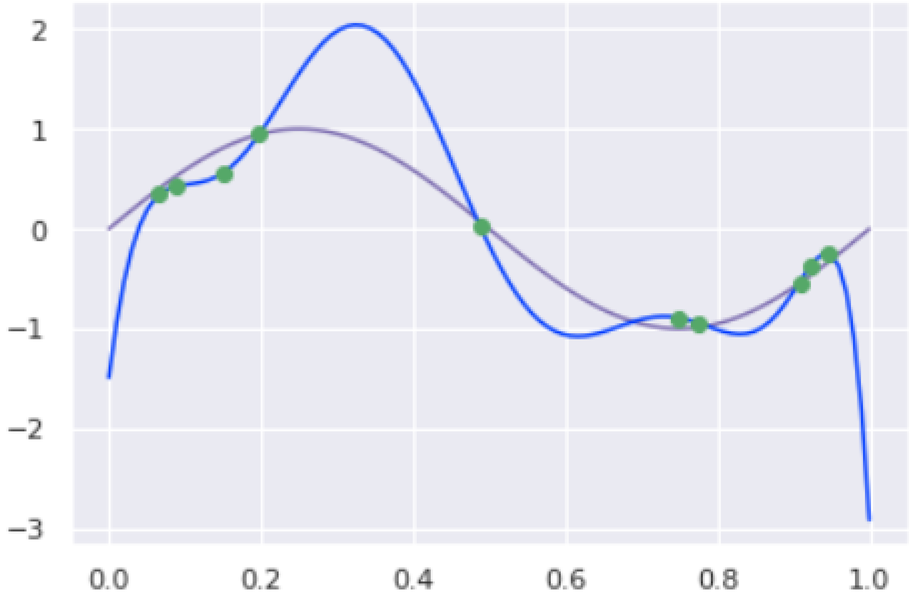
In [20]:

```
plot_by_lambda(exp(-18))
```



In [21]:

```
plot_by_lambda(0)
```

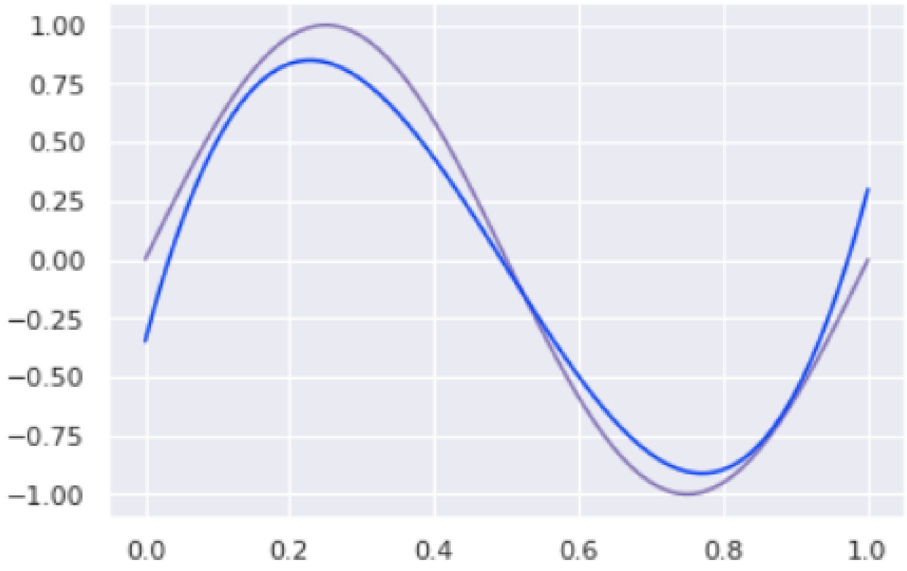


In [23]:

```
def best_model(start_M, end_M, ls, sets):
```

```
M = 3 lambda = 0 erms = 0.13287880898345142
```

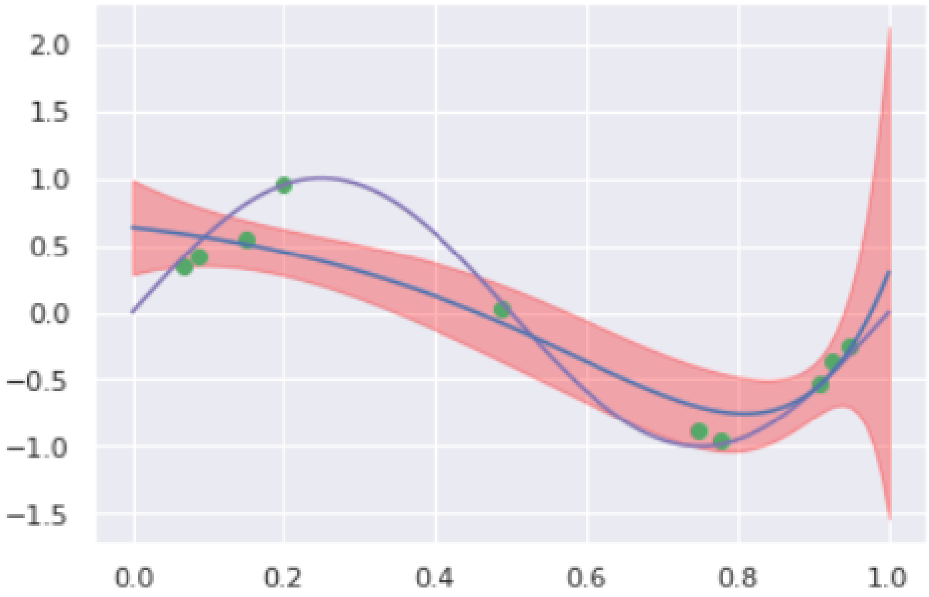
```
Out[23]: [<matplotlib.lines.Line2D at 0x7f863c239ed0>]
```



4.1) Display results

```
In [28]: alpha = 0.05
beta = 1.1
M = 9

means = np.array(list(map(lambda x: mean(alpha, beta,
variances = np.array(list(map(lambda x: variance(alpha

plt.plot(x_train, y_train, 'og')
plt.plot(x_test, y_test, '-m')
plt.plot(x_test, means, '-b')
plt.fill_between(x_test, means + variances, means - va
```

```
Out[28]: <matplotlib.collections.PolyCollection at 0x7f863c16d7
```


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0