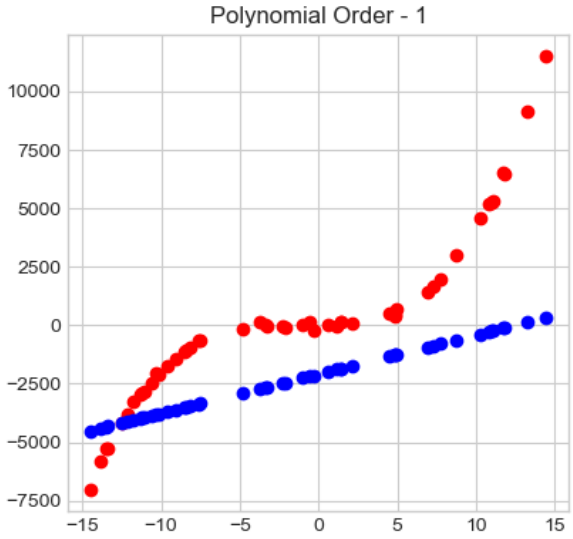
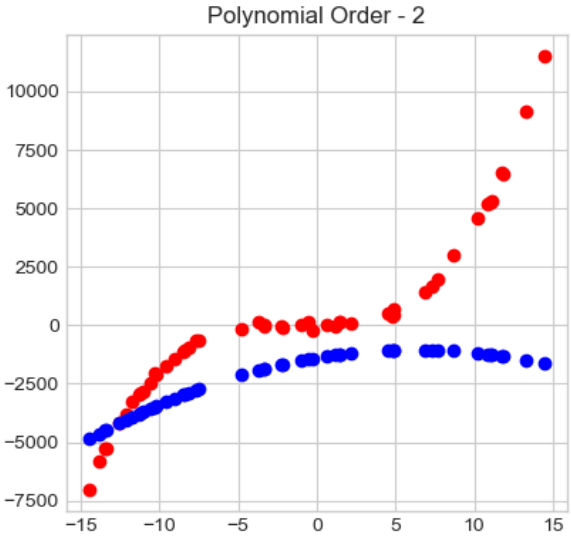
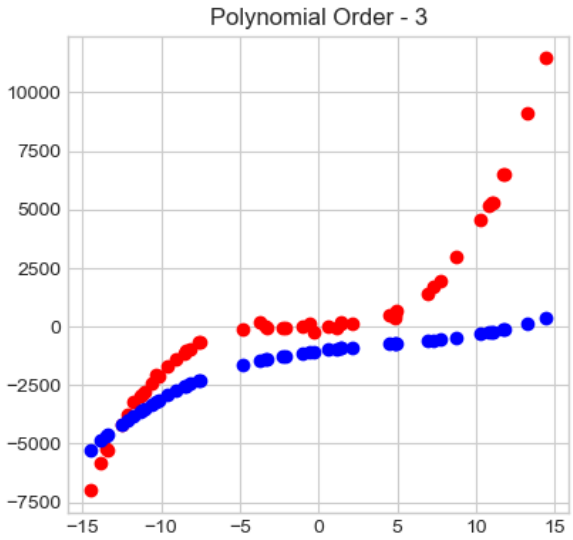
In [228…

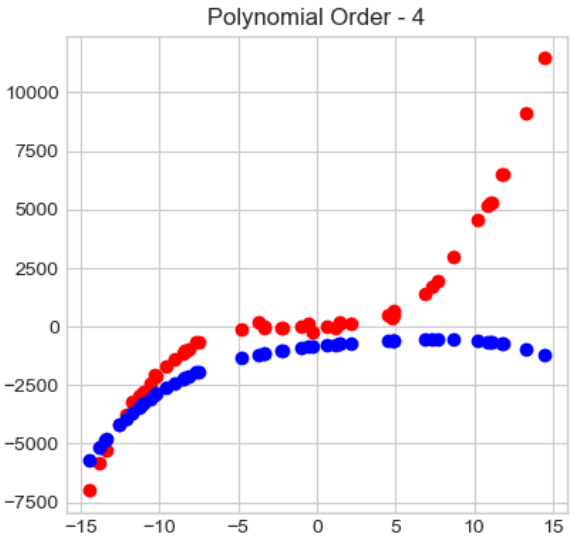
```
#Q1 -  Please plot the noisy data and the polynomial you found (in the same figure,
# polynomial order of m = 1, 2, 3, 4, 5, 6, 7, 8, respectively

#Ignoring warning messages
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
noise_scale = 100
number_of_samples = 50
x = 30*(np.random.rand(number_of_samples, 1) - 0.5)
y = 2 * x + 11 * x**2 + 3 * x**3 + noise_scale*np.random.randn(number_of_samples, 1
# Printing coefficients of the polynomial
print("Coefficients of the polynomial is {} where order is 3.".format(np.polyfit(x
figure, axis = plt.subplots(4,2,figsize=(10, 20))
for i,ax in zip([1,2,3,4,5,6,7,8],axis.flatten()):
    z = np.polyfit(x[0],y[0],i)
    # calculating the ploynimial value using poly1d
    poly = np.poly1d(z)
    polyArr = poly(x)
    # Using subplot to plot muliple graphs in single figure.
    ax.set_title("Polynomial Order - {}".format(i))
    ax.plot(x,y,'ro')
    ax.plot(x,polyArr,'ro',color='b', label='Polynomial')
```

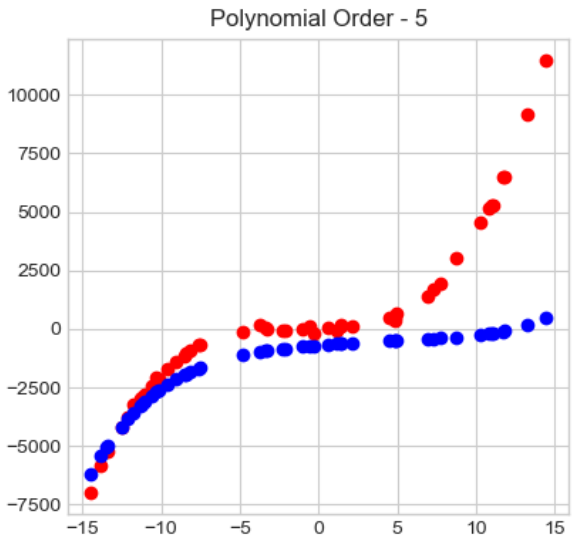
```
Coefficients of the polynomial is [-7.74214577e-07  9.68503670e-06 -1.21154960e-04
1.51558788e-03
 -1.89592455e-02  2.37170669e-01 -2.96688631e+00  3.71142622e+01
 -4.64280837e+02].
```

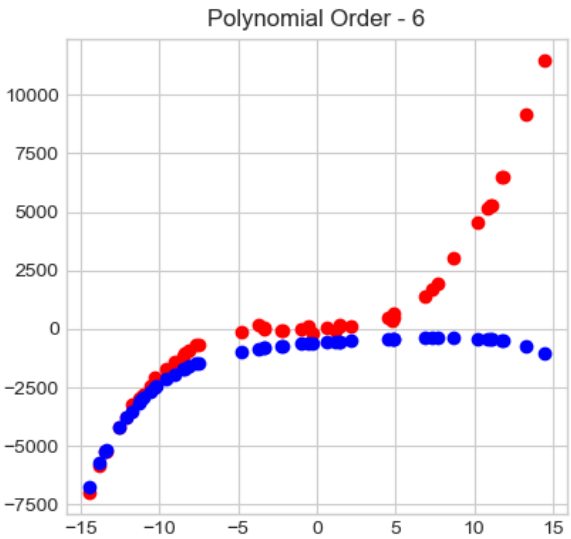

Polynomial Order - 1
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15

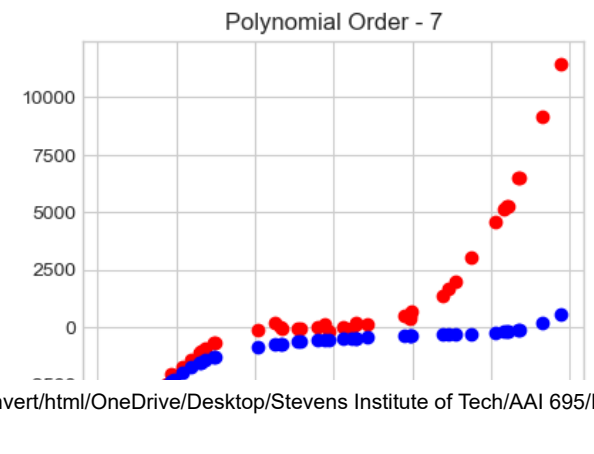

Polynomial Order - 2
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15

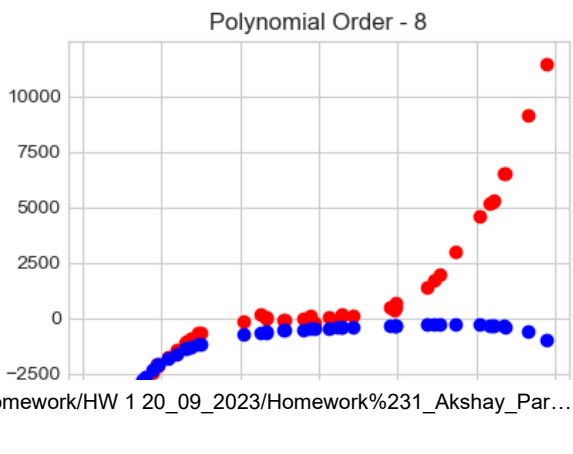

Polynomial Order - 3
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15

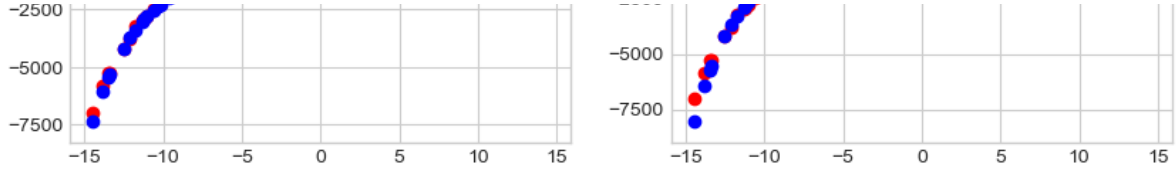

Polynomial Order - 4
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15


Polynomial Order - 5
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15


Polynomial Order - 6
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15

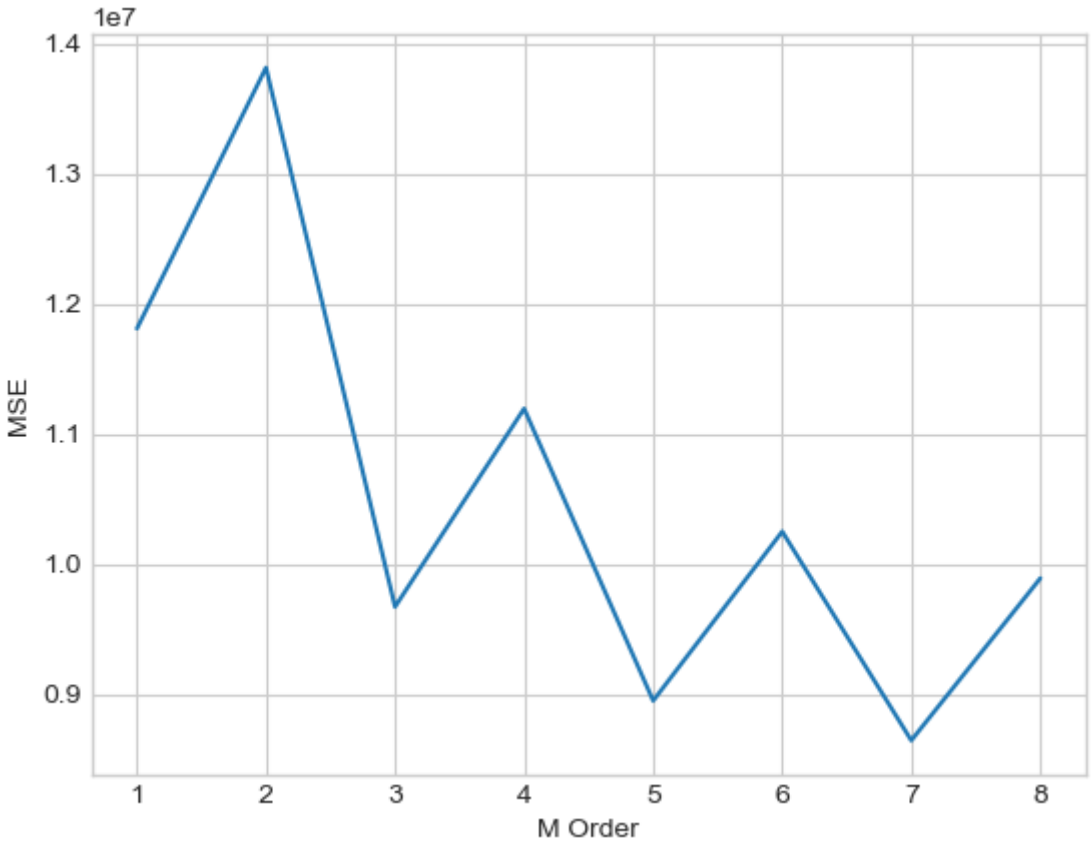

Polynomial Order - 7
10000
7500
5000
2500
0


Polynomial Order - 8
10000
7500
5000
2500
0
−2500



In [229…
```
#Q2 - Plot MSE versus order m, for m = 1, 2, 3, 4, 5, 6, 7, 8 respectively. Identij

mseArr = []
m = [1,2,3,4,5,6,7,8]
for i,ax in zip(m,axis.flatten()):
    z = np.polyfit(x[0],y[0],i)
    poly = np.poly1d(z)
    polyArr = poly(x)
    #Calculating MSE using numpy.
    mse = np.mean((y-polyArr)**2)
    mseArr.append(mse)
print(mseArr)
plt.xlabel("M Order")
plt.ylabel("MSE")
plt.plot(m,mseArr, label='Polynomial')

#7th Order Polynomial is best fit for noisy data as the mean squared error is lowes
#Higher or lower order polynomial will over fit or under fit the data.
```

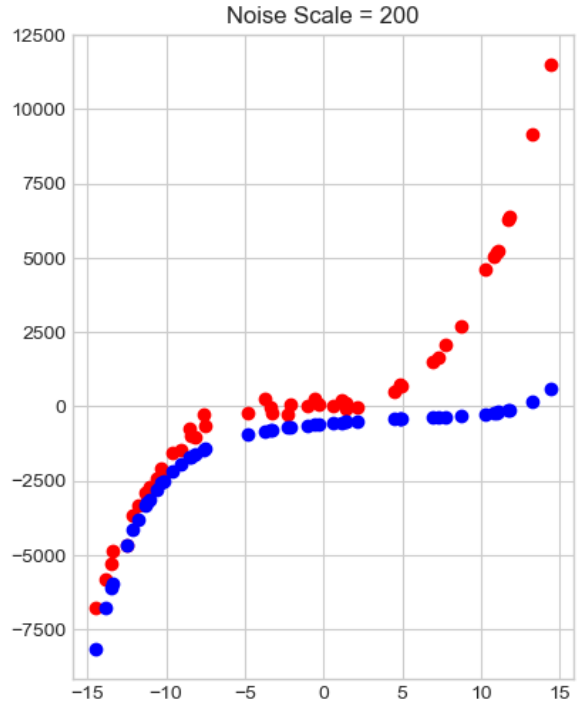
Out[229]:
```
[11807689.801004685, 13812685.900564404, 9669104.552998448, 11192258.03203679, 894
7510.740022516, 10247496.942964828, 8642276.821186109, 9888764.390925959]
[<matplotlib.lines.Line2D at 0x266c6d36ed0>]
```

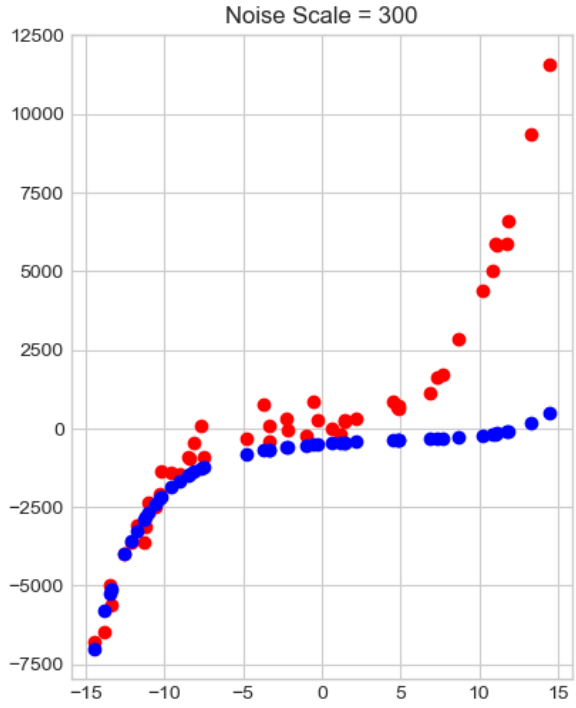


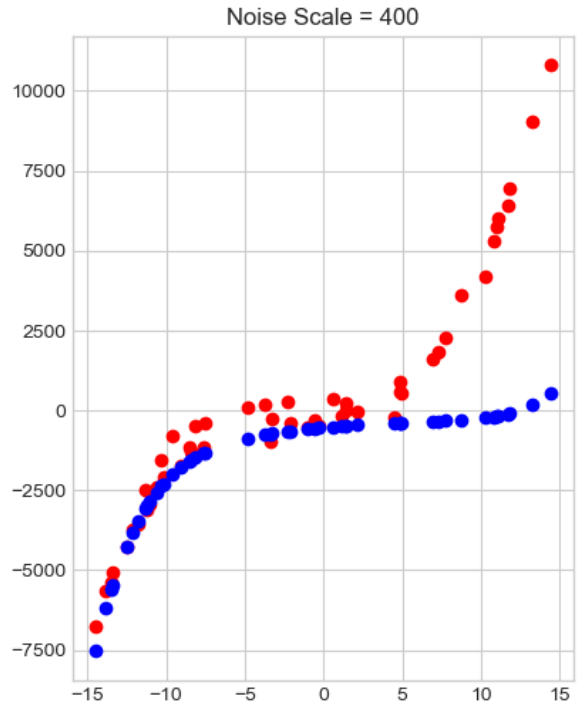
In [230…
```
bestOrderPolynomial = 7
```

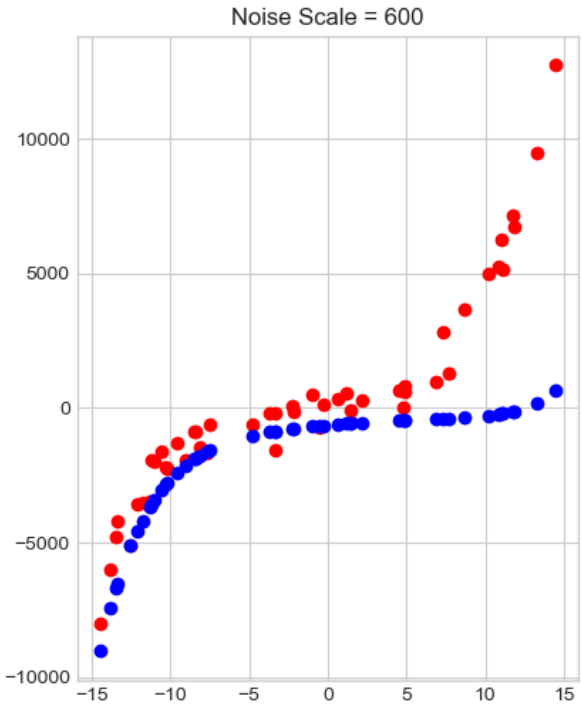
In [231…
```
#Q3 - Change variable noise_scale to 200, 300, 400, 600, 800, 1000 respectively, re
# and plot the polynomials with the best m found in 2). Discuss the impact of noise
# accuracy of the returned parameters. [You need to plot a figure for EACH choice o
```

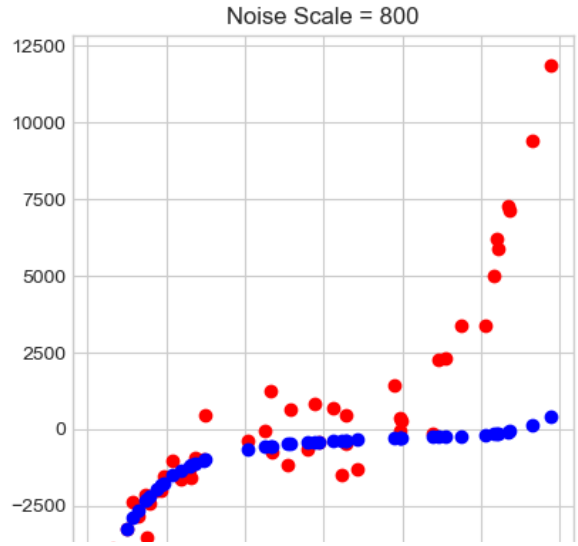
```
noise_scaleArr = [200,300,400,600,800,1000]
figure, axis = plt.subplots(3,2,figsize=(10, 20))
for noise_scale,ax in zip(noise_scaleArr,axis.flatten()):
    number_of_samples = 50
    y = 2 * x + 11 * x**2 + 3 * x**3 + noise_scale*np.random.randn(number_of_sampl
    z = np.polyfit(x[0],y[0],bestOrderPolynomial)
    poly = np.poly1d(z)
    polyArr = poly(x)
    ax.set_title("Noise Scale = {}".format(noise_scale))
    ax.plot(x,y,'ro')
    ax.plot(x,polyArr,'ro',color='b', label='Polynomial')

#As we keep on increasing the noise scale,the data points are getting more and more
#the MSE of 7th Order polynomial increases which leads to overfitting of the data.
```

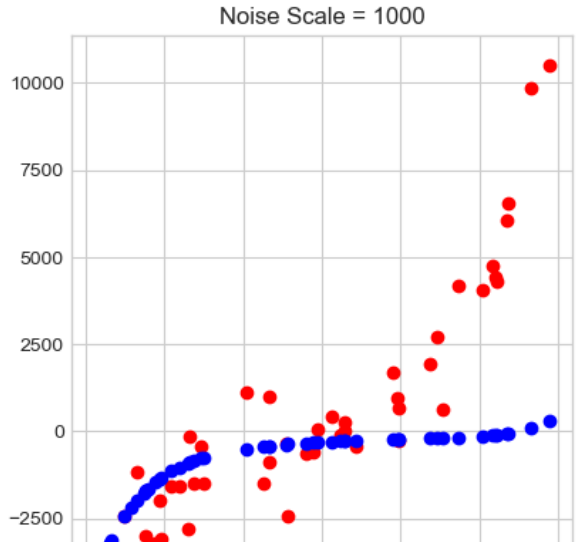

Noise Scale = 200
12500
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15

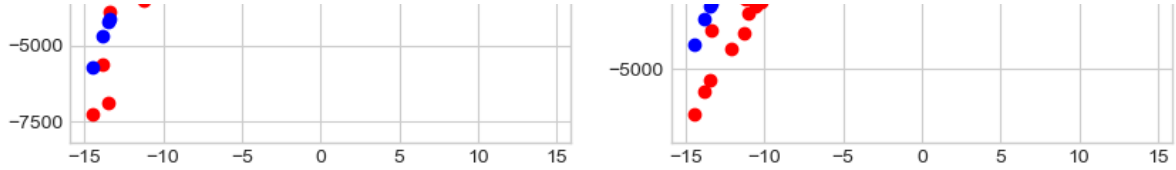

Noise Scale = 300
12500
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15


Noise Scale = 400
10000
7500
5000
2500
0
−2500
−5000
−7500
−15
−10
−5
0
5
10
15


Noise Scale = 600
10000
5000
0
−5000
−10000
−15
−10
−5
0
5
10
15


Noise Scale = 800
12500
10000
7500
5000
2500
0
−2500

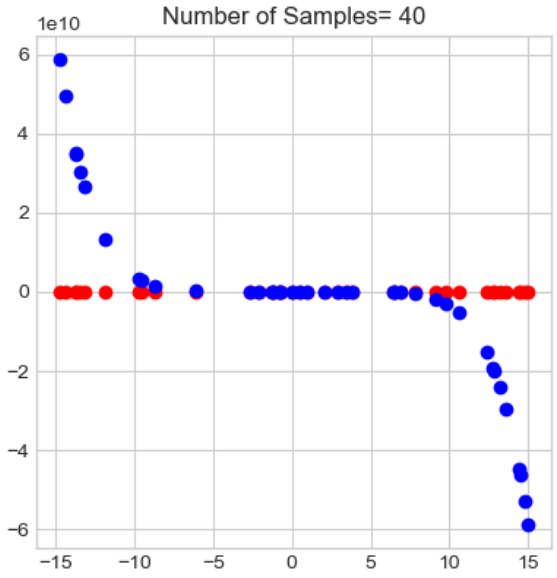

Noise Scale = 1000
10000
7500
5000
2500
0
−2500

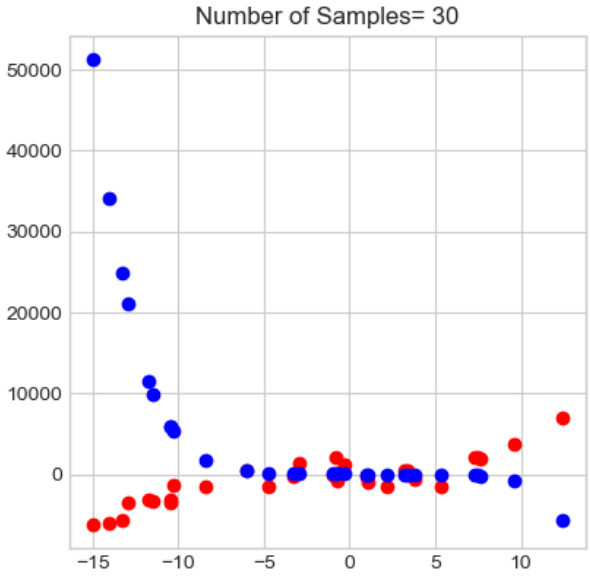


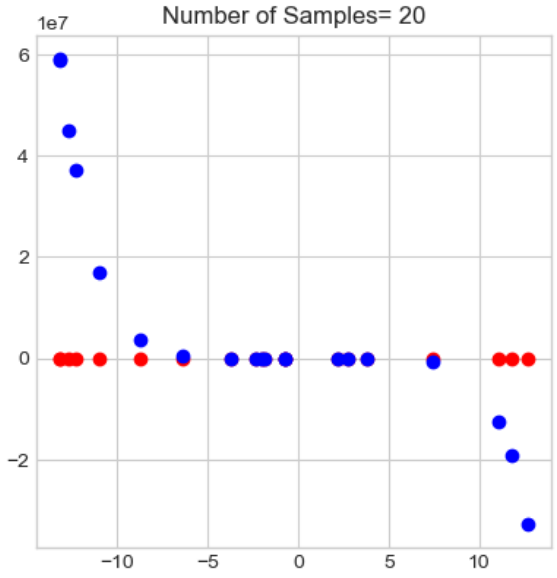
In [232…

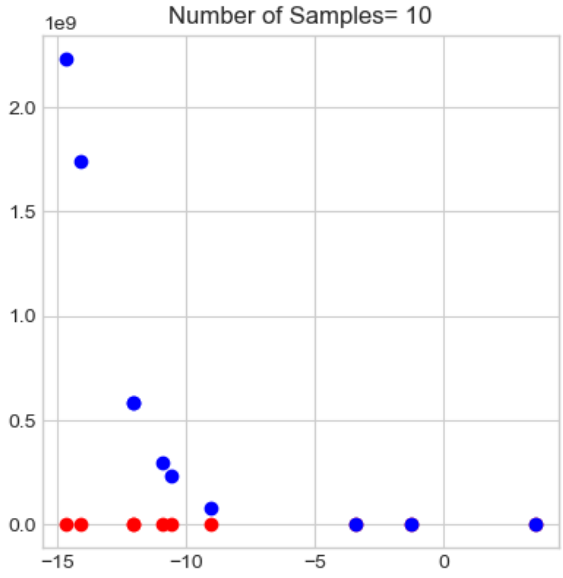
In [233…

```
#Q4 - Change variable number_of_samples to 40, 30, 20, 10 respectively, re-ran the
# the polynomials with the best m found in 2). Discuss the impact of the number of
# accuracy of the returned parameters. [You need to plot a figure for EACH choice o
# number_of_samples.

#This graph may be different, as we are using random function to generate the value
# we are generating the values again, the values of x and y won't be same compared
# are considering in the above examples, may change for this values of x and y.

samplesArr = [40, 30, 20, 10]
figure, axis = plt.subplots(2,2,figsize=(10, 10))
for number_of_samples,ax in zip(samplesArr,axis.flatten()):
    x = 30*(np.random.rand(number_of_samples, 1) - 0.5)
    y = 2 * x + 11 * x**2 + 3 * x**3 + noise_scale*np.random.randn(number_of_sample
    z = np.polyfit(x[0],y[0],bestOrderPolynomial)
    poly = np.poly1d(z)
    polyArr = poly(x)
    ax.set_title("Number of Samples= {}".format(number_of_samples))
    ax.plot(x,y,'ro')
    ax.plot(x,polyArr,'ro',color='b', label='Polynomial')

#If we keep on decreasing the number of samples, we will get very less number of da
```


Number of Samples= 40
1e10
6
4
2
0
−2
−4
−6
−15
−10
−5
0
5
10
15


Number of Samples= 30
50000
40000
30000
20000
10000
0
−15
−10
−5
0
5
10


Number of Samples= 20
1e7
6
4
2
0
−2
−10
−5
0
5
10


Number of Samples= 10
1e9
2.0
1.5
1.0
0.5
0.0
−15
−10
−5
0

In [ ]: