LinearRegression

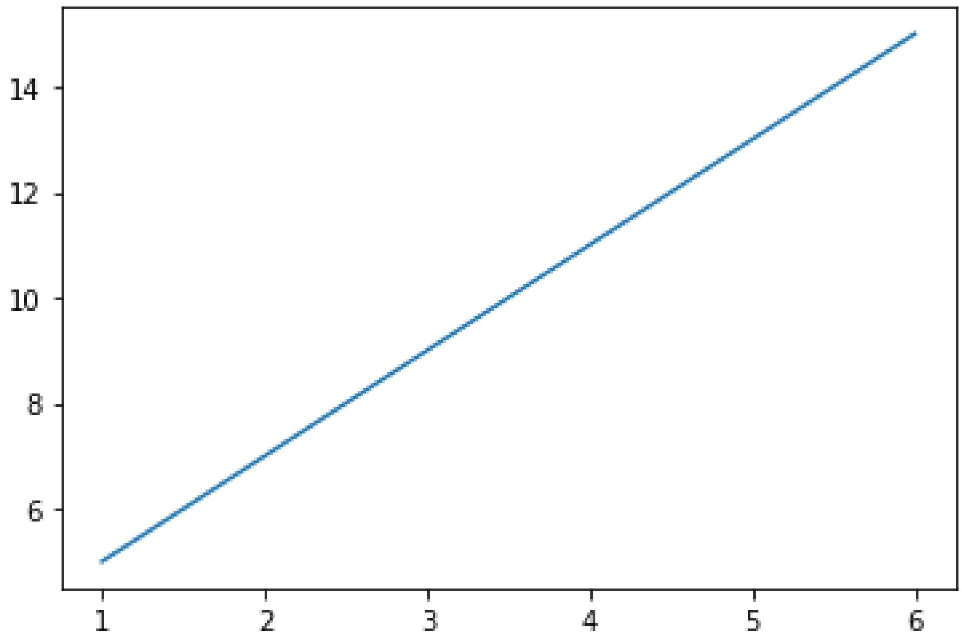
November 17, 2019

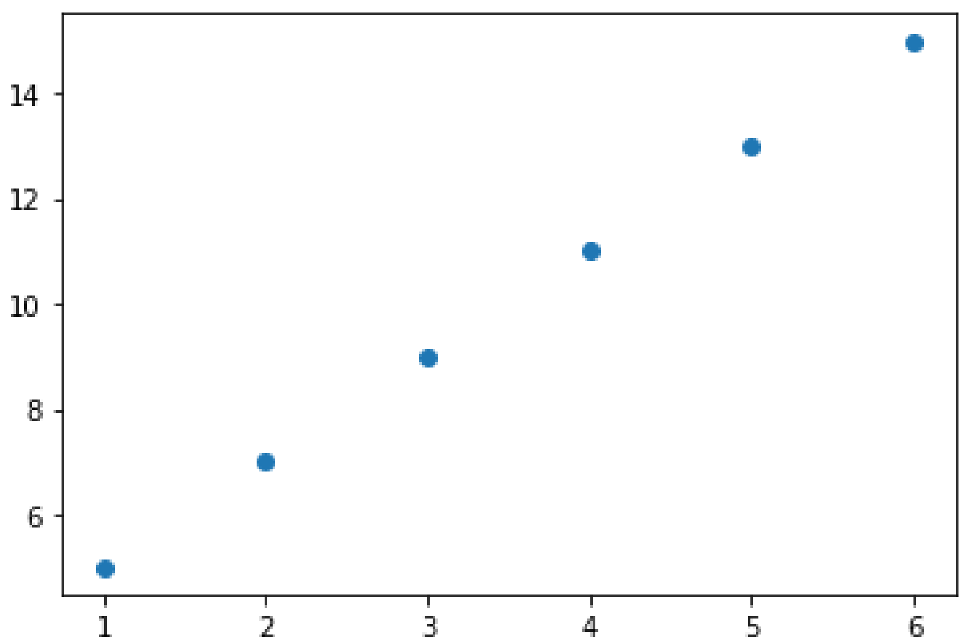
[119]:
```
#Linear Regression
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
colors = cm.rainbow(np.linspace(0.4, 1, 2))
```

[120]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

[121]:
```
#x is the input data
#y_train is the desired output
#In this case, y_train has been generated from x using a linear function


x = np.array([1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0]).reshape(-1,1)
x_train = x
y_train = 2*x_train + 3
print('x: ', x_train)
print('y: ', y_train)
plt.plot(x_train,y_train)
plt.show()
plt.scatter(x_train,y_train)
plt.show()
```

```
x:  [[1.]
 [2.]
 [3.]
 [4.]
 [5.]
 [6.]]
y:  [[ 5.]
 [ 7.]
 [ 9.]
 [11.]
 [13.]
 [15.]]
```



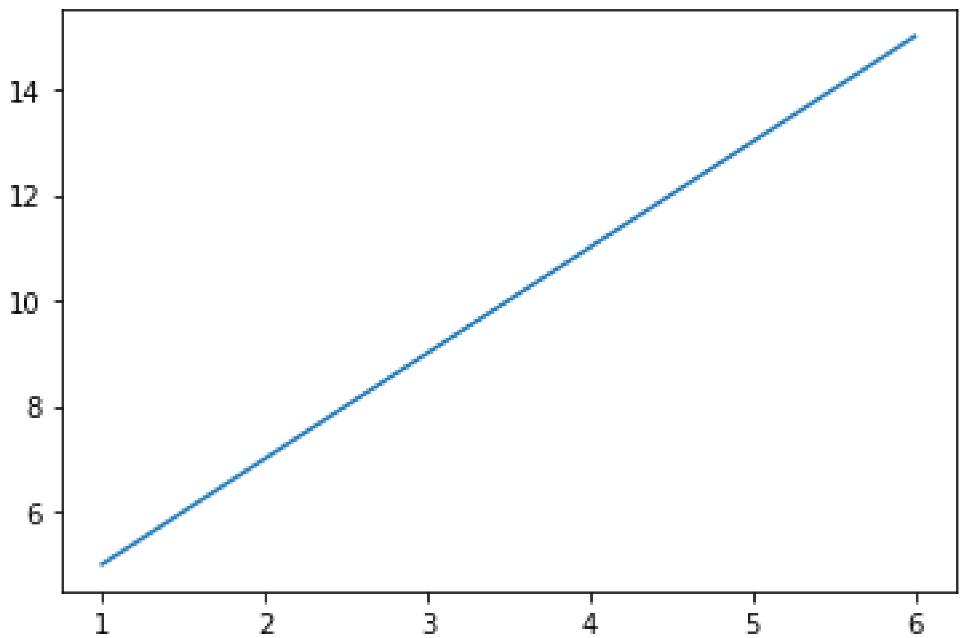


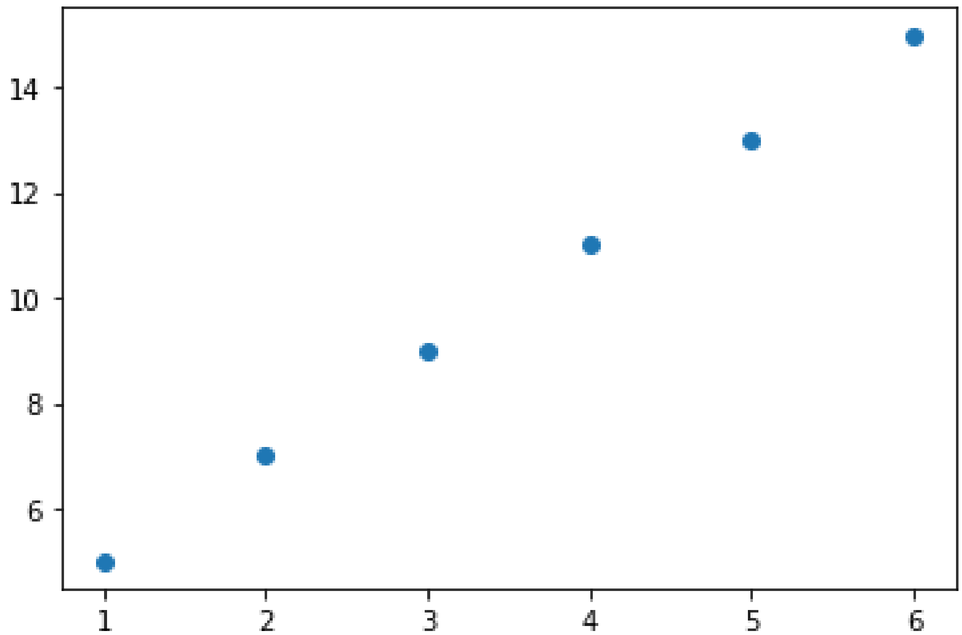
[122]:
```
#Here, we will apply linear regression to x_train and y_train which we know are
 ↪linearly related
regressor.fit(x_train,y_train)
```

[122]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[123]:
```
#Here, we are checking the results of our model, as we can see y_predicted is␣
↪exactly same as y_train
y_predicted = regressor.predict(x_train)
print(y_predicted)
plt.plot(x,y_predicted)
plt.show()
plt.scatter(x,y_predicted)
plt.show()
```

```
[[ 5.]
 [ 7.]
 [ 9.]
 [11.]
 [13.]
 [15.]]
```

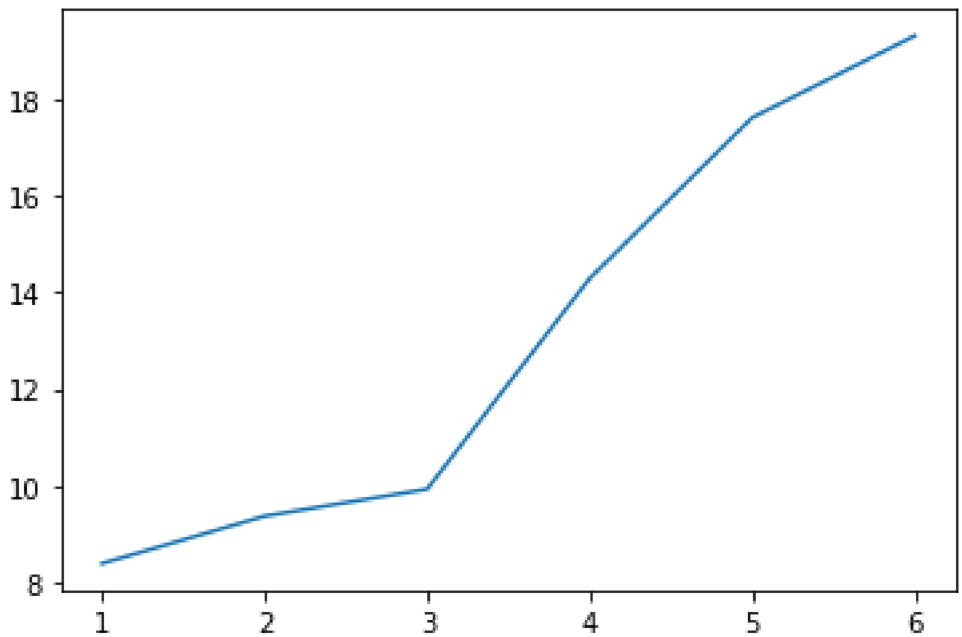


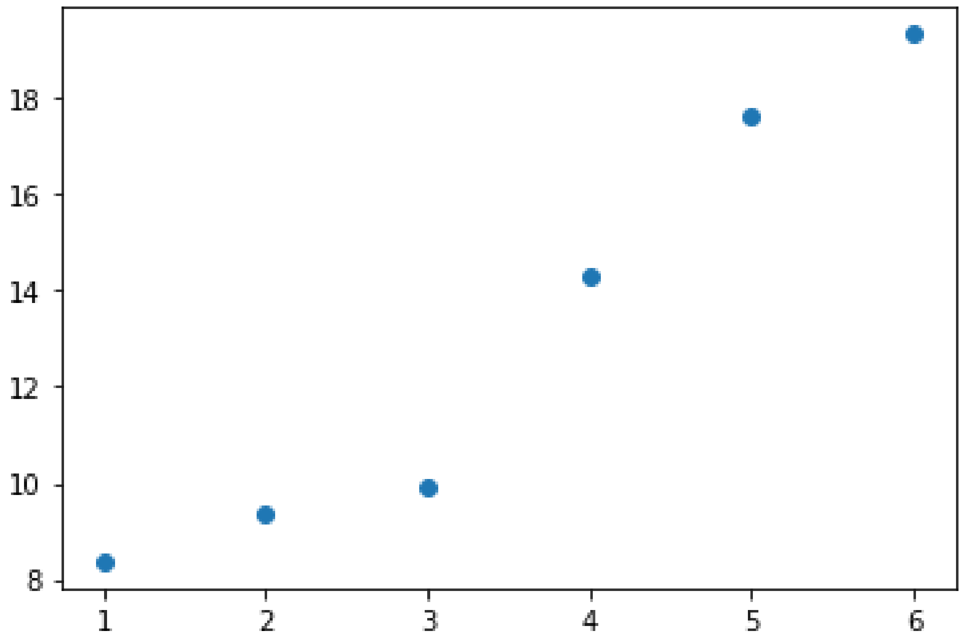


[124]:
```
#Now, we will introduce some noise in our y_train values so that y_train and x␣
↪are not perfectly linearly related
import random
print('y before noise : ', y_train)
for i in range(y_train.shape[0]):
    y_train[i][0] += 5*random.random()
print('y after noise : ', y_train)
print('x: ', x_train)
print('y: ', y_train)
plt.plot(x,y_train)
plt.show()
plt.scatter(x,y_train)
plt.show()
```

```
y before noise :  [[ 5.]
 [ 7.]
 [ 9.]
 [11.]
 [13.]
 [15.]]
y after noise :  [[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
```

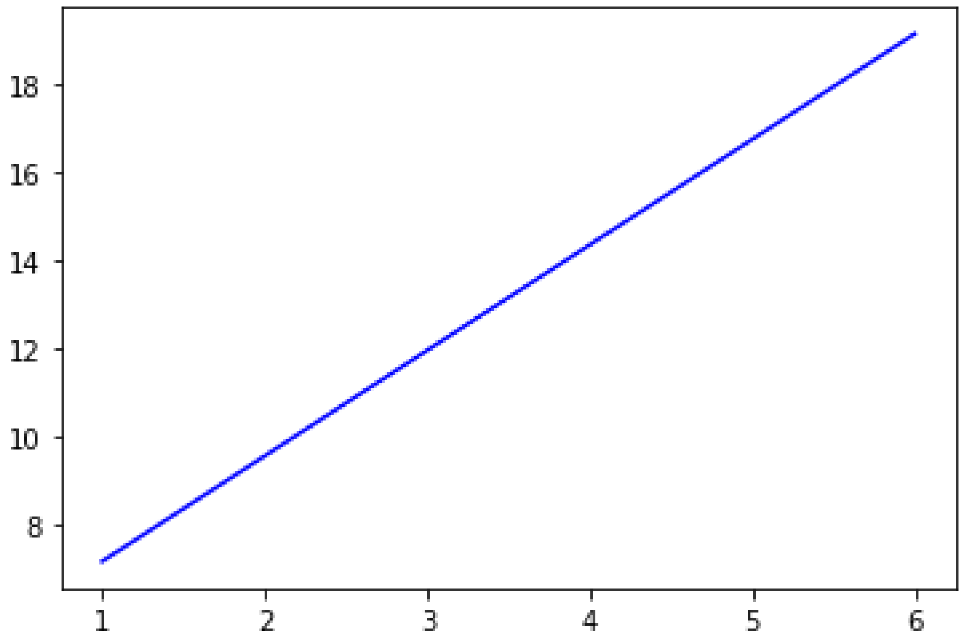
```
x:  [[1.]
 [2.]
 [3.]
 [4.]
 [5.]
 [6.]]
y:  [[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
```

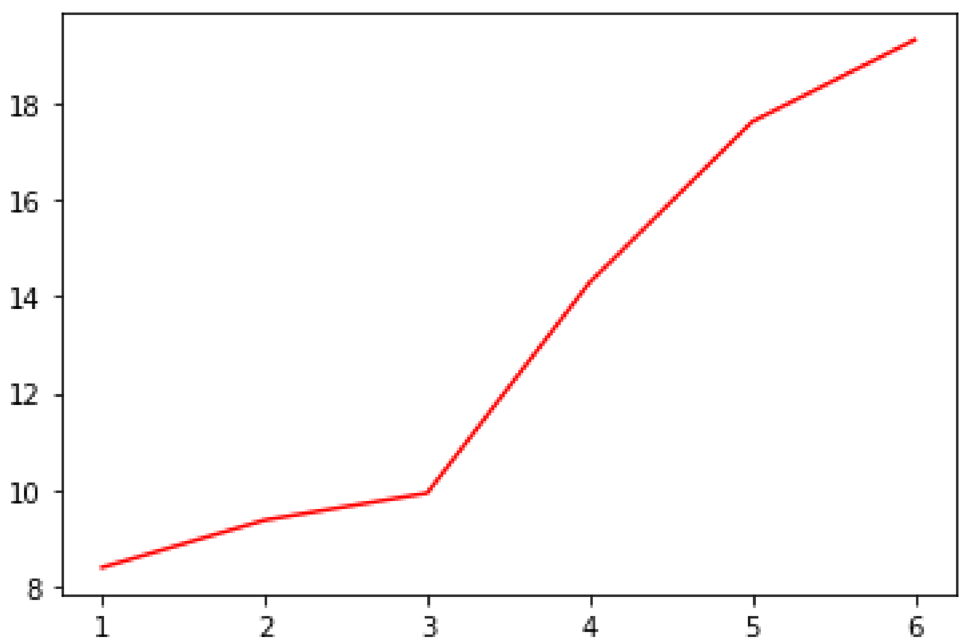


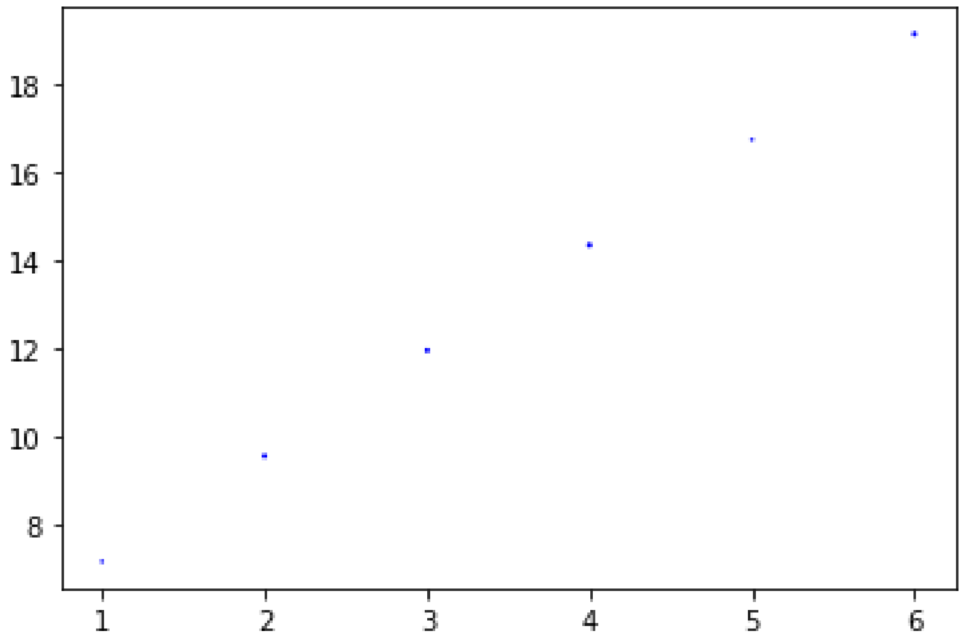


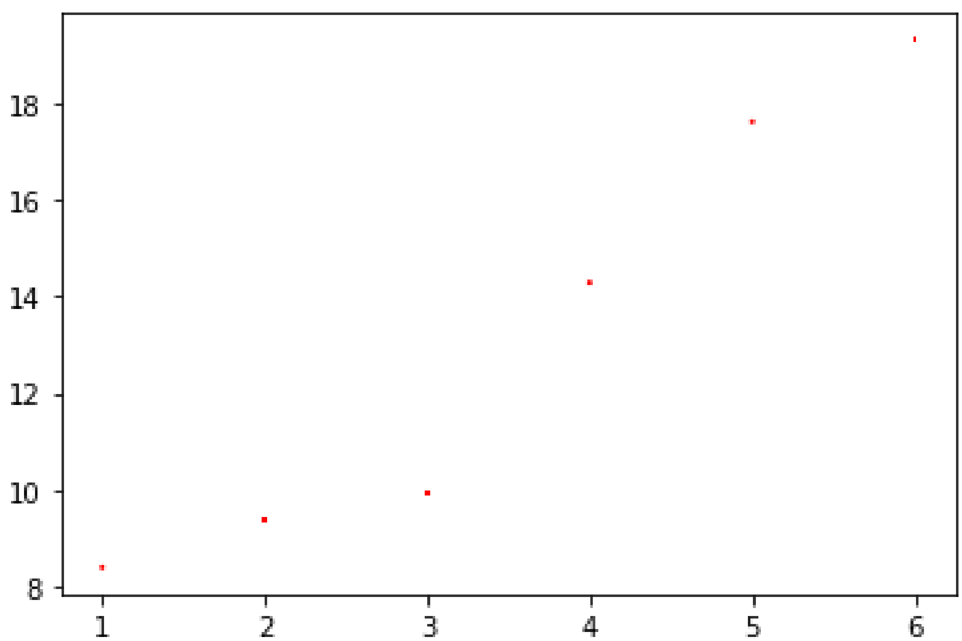
[125]:
```
#We will now apply linear regression to our not perfectly linearly related x␣
↪and y_train
#As we can see, the predicted y from our model is a linear one but it doesnt␣
↪perform well at identifying true y
regressor.fit(x_train,y_train)
y_predicted = regressor.predict(x_train)
print(y_predicted)

plt.plot(x,y_predicted,'b')
plt.show()
plt.plot(x,y_train,'r')
plt.show()

plt.scatter(x,y_predicted,colors[0],'blue','o')
plt.show()
plt.scatter(x,y_train,colors[1],'red','o')
plt.show()
```

```
[[ 7.16393953]
 [ 9.55917986]
 [11.9544202 ]
 [14.34966054]
 [16.74490088]
 [19.14014121]]
```


18
16
14
12
10
8
1
2
3
4
5
6


18
16
14
12
10
8
1
2
3
4
5
6

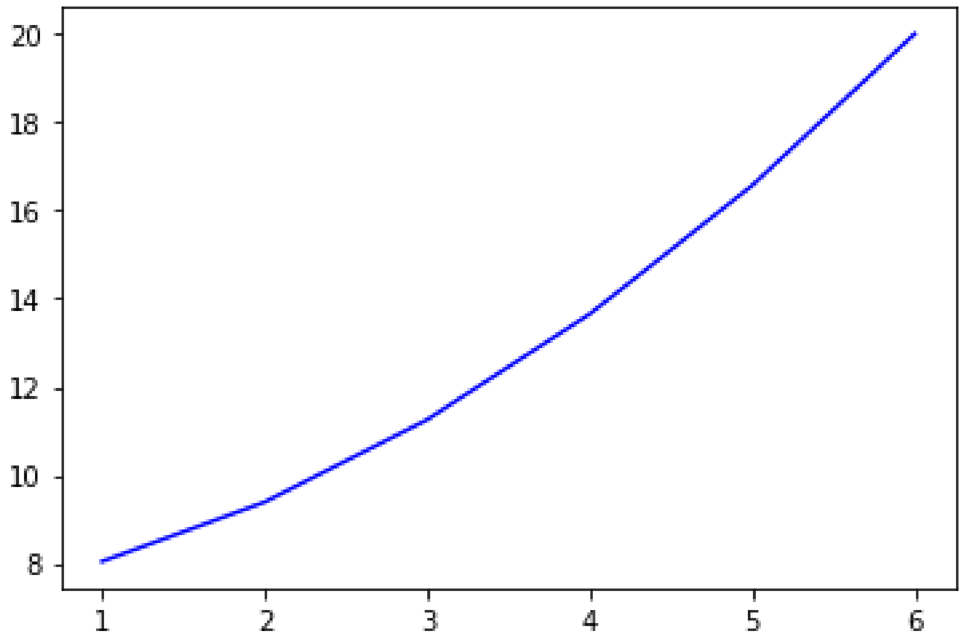


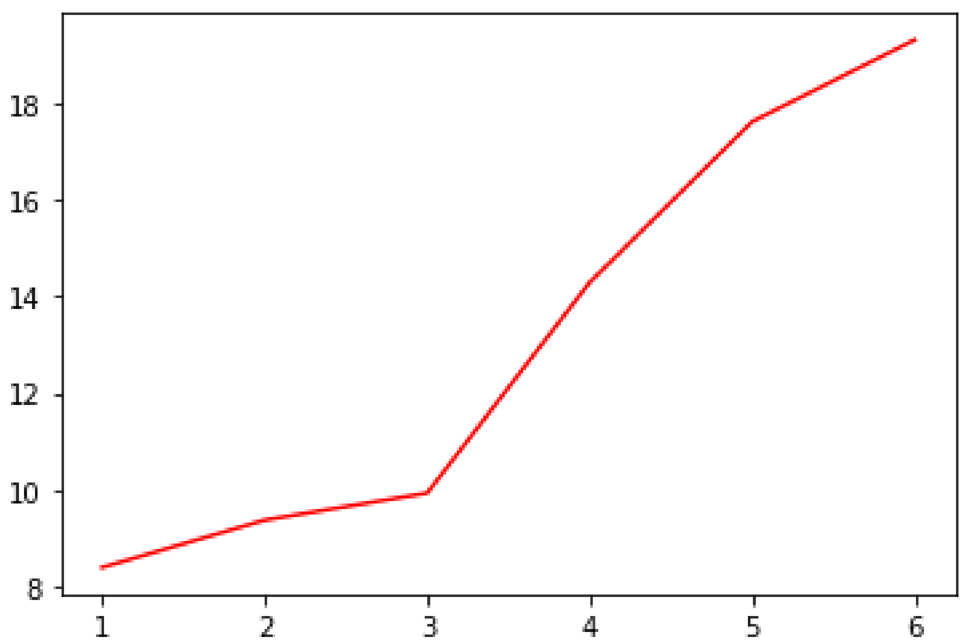


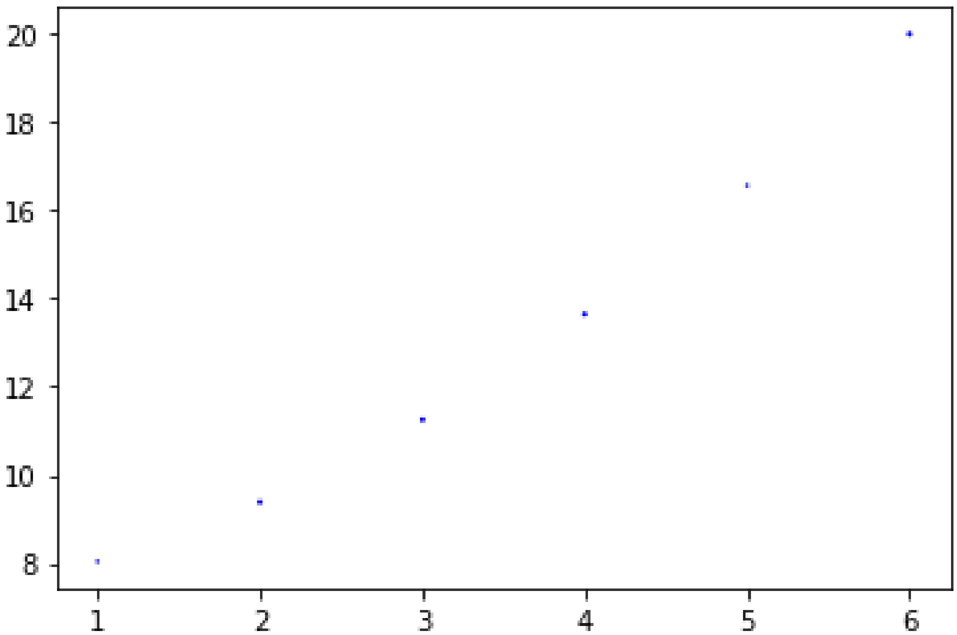
[126]:
```
#Here, we will model Y = a + bx + cx^2
x_train = np.hstack((x,x*x))
print(x_train)
print(y_train)
```

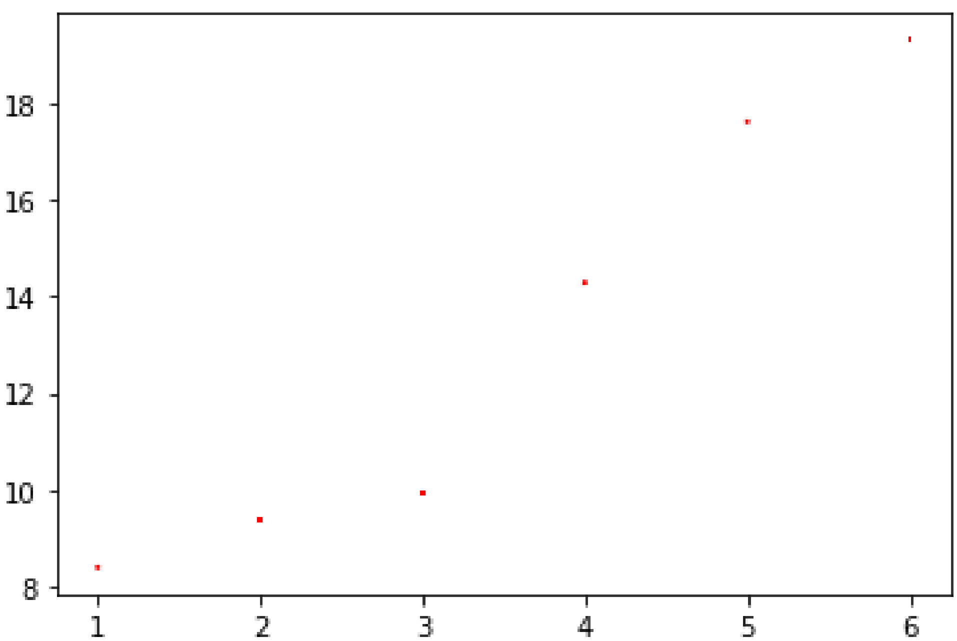
```python
regressor.fit(x_train,y_train)
y_predicted = regressor.predict(x_train)
print(y_predicted)

plt.plot(x,y_predicted,'b')
plt.show()
plt.plot(x,y_train,'r')
plt.show()

plt.scatter(x,y_predicted,colors[0],'blue','o')
plt.show()
plt.scatter(x,y_train,colors[1],'red','o')
plt.show()
```

```
[[ 1.  1.]
 [ 2.  4.]
 [ 3.  9.]
 [ 4. 16.]
 [ 5. 25.]
 [ 6. 36.]]
[[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
[[ 8.03672006]
 [ 9.38462376]
 [11.25619578]
 [13.65143611]
 [16.57034477]
 [20.01292174]]
```


20
18
16
14
12
10
8
1
2
3
4
5
6


18
16
14
12
10
8
1
2
3
4
5
6

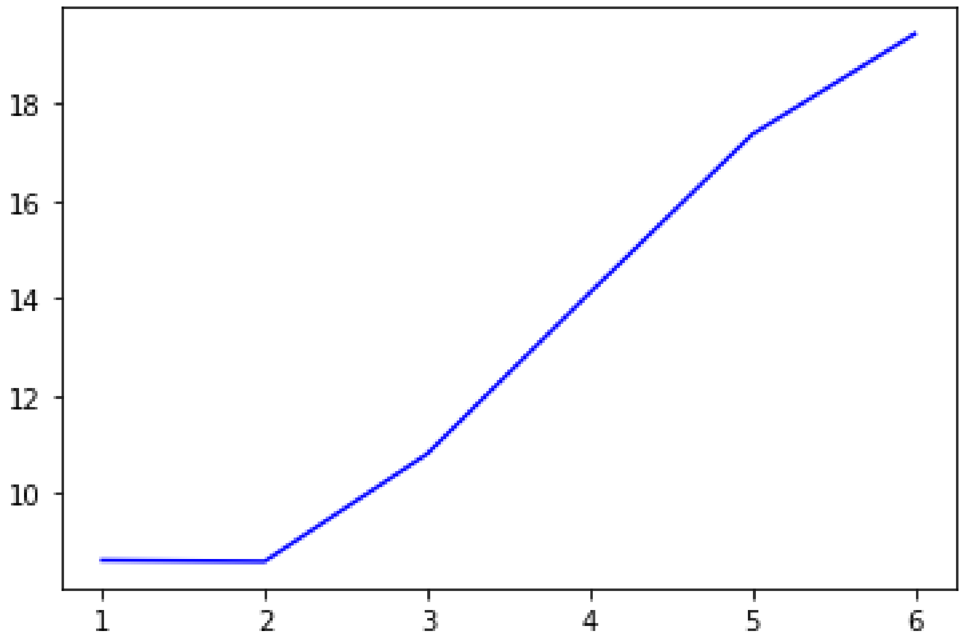


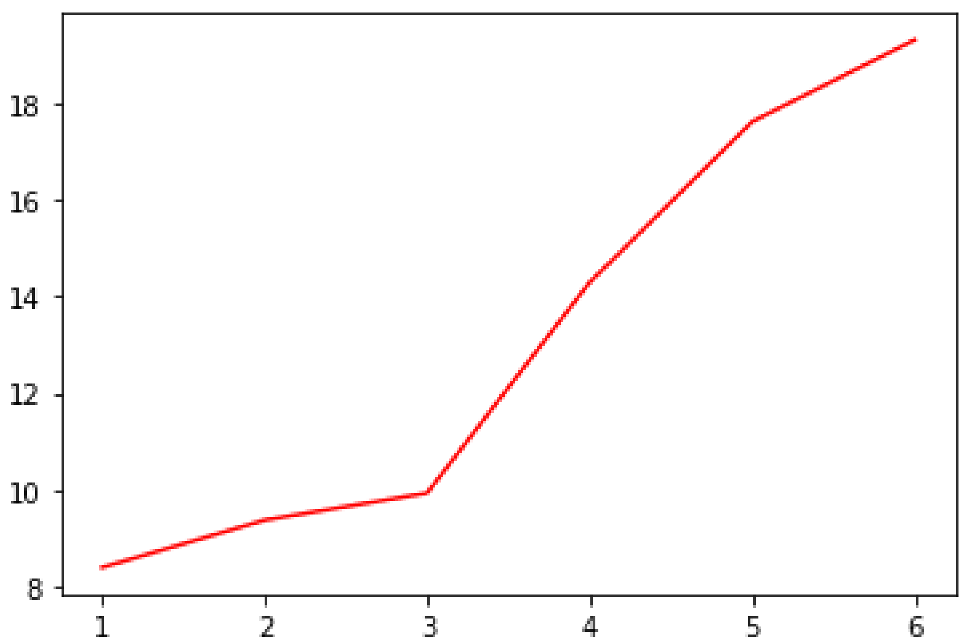


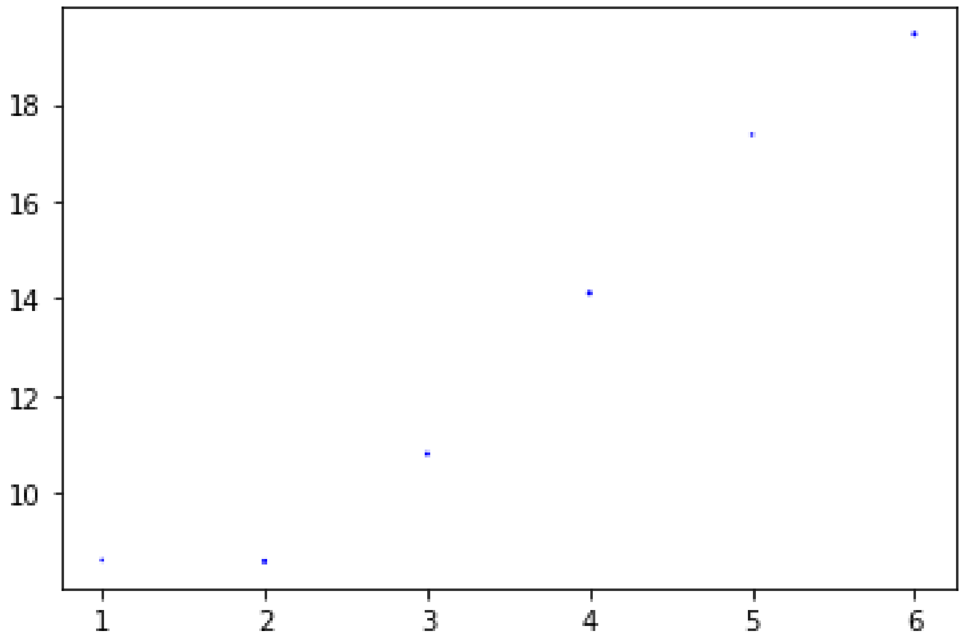
[127]:
```
#Here, we will model Y = a + bx + cx^2 + dx^3
x_train = np.hstack((x,x*x,x*x*x))
print(x_train)
print(y_train)
```

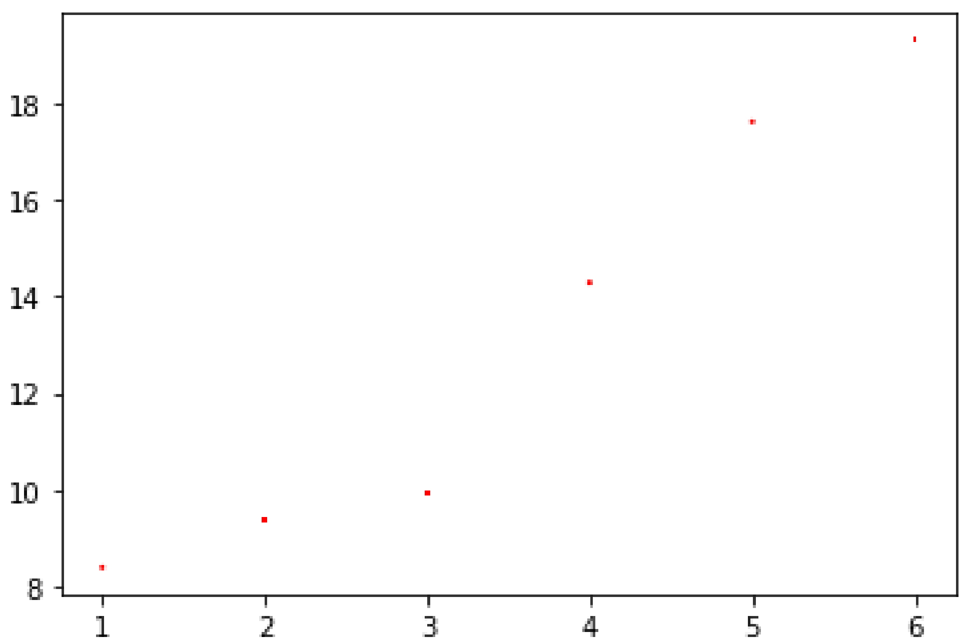
```python
regressor.fit(x_train,y_train)
y_predicted = regressor.predict(x_train)
print(y_predicted)

plt.plot(x,y_predicted,'b')
plt.show()
plt.plot(x,y_train,'r')
plt.show()

plt.scatter(x,y_predicted,colors[0],'blue','o')
plt.show()
plt.scatter(x,y_train,colors[1],'red','o')
plt.show()
```

```
[[  1.   1.   1.]
 [  2.   4.   8.]
 [  3.   9.  27.]
 [  4.  16.  64.]
 [  5.  25. 125.]
 [  6.  36. 216.]]
[[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
[[ 8.60890265]
 [ 8.58356813]
 [10.7984497 ]
 [14.10918219]
 [17.3714004 ]
 [19.44073915]]
```


18
16
14
12
10
1
2
3
4
5
6


18
16
14
12
10
8
1
2
3
4
5
6

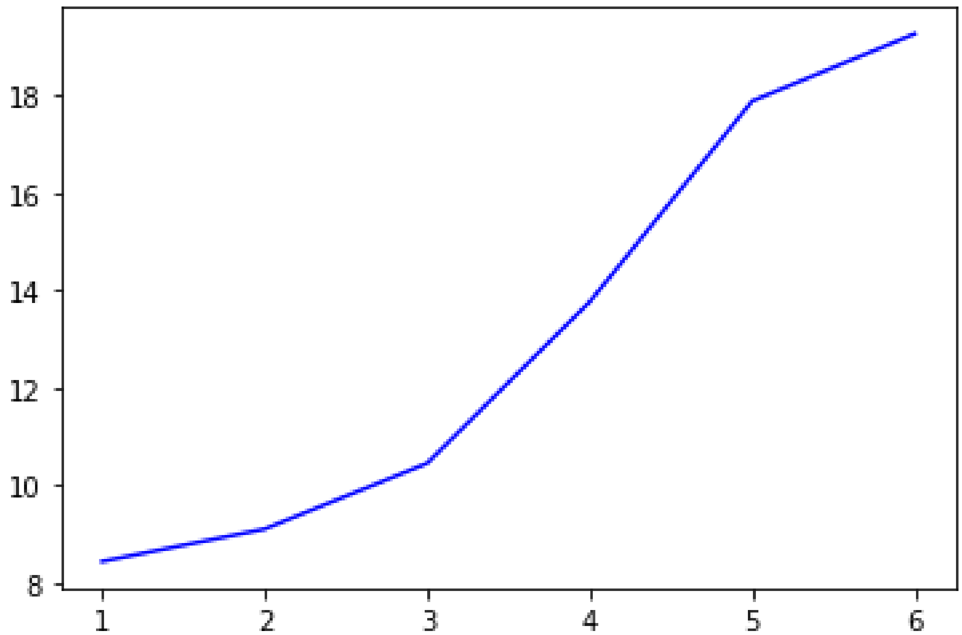




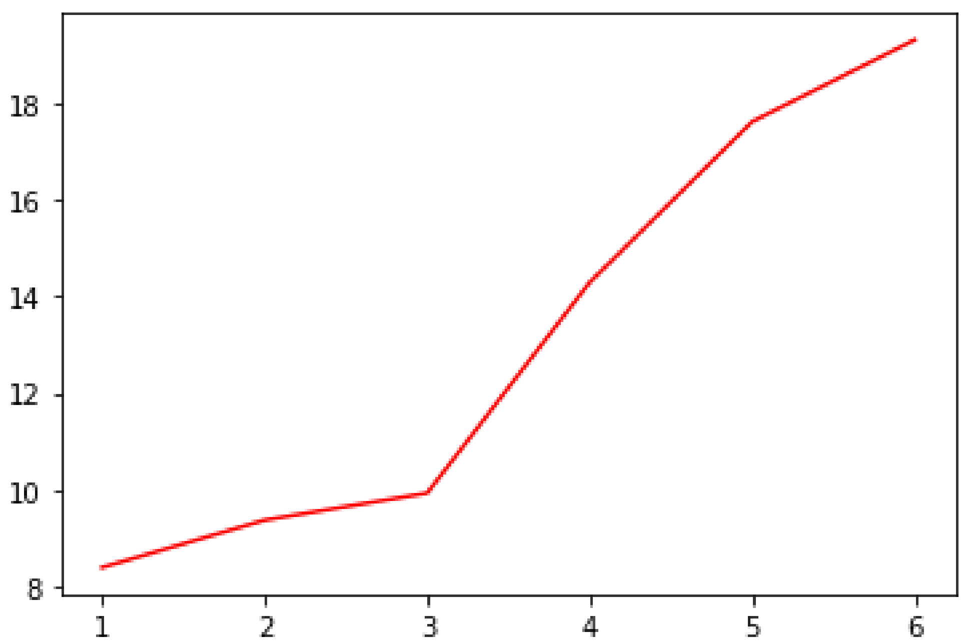
[128]:
```
#Here, we will model Y = a + bx + cx^2 + dx^3 + ex^4
x_train = np.hstack((x,x*x,x*x*x,x*x*x*x))
print(x_train)
print(y_train)
```

```python
regressor.fit(x_train,y_train)
y_predicted = regressor.predict(x_train)
print(y_predicted)

plt.plot(x,y_predicted,'b')
plt.show()
plt.plot(x,y_train,'r')
plt.show()

plt.scatter(x,y_predicted,colors[0],'blue','o')
plt.show()
plt.scatter(x,y_train,colors[1],'red','o')
plt.show()
```

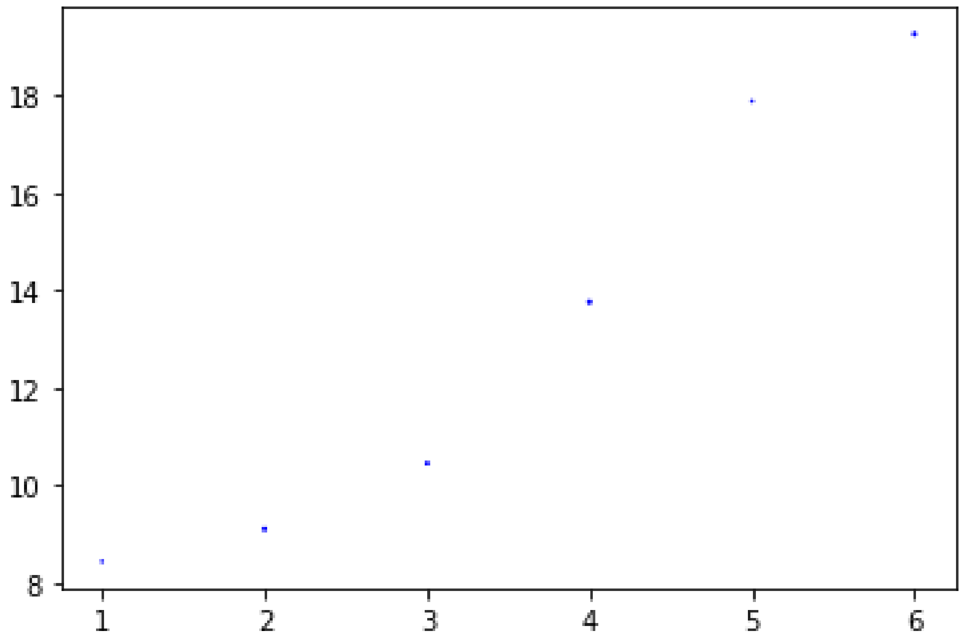
```
[[1.000e+00 1.000e+00 1.000e+00 1.000e+00]
 [2.000e+00 4.000e+00 8.000e+00 1.600e+01]
 [3.000e+00 9.000e+00 2.700e+01 8.100e+01]
 [4.000e+00 1.600e+01 6.400e+01 2.560e+02]
 [5.000e+00 2.500e+01 1.250e+02 6.250e+02]
 [6.000e+00 3.600e+01 2.160e+02 1.296e+03]]
[[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
[[ 8.43623561]
 [ 9.10156925]
 [10.45311562]
 [13.76384811]
 [17.88940152]
 [19.26807211]]
```

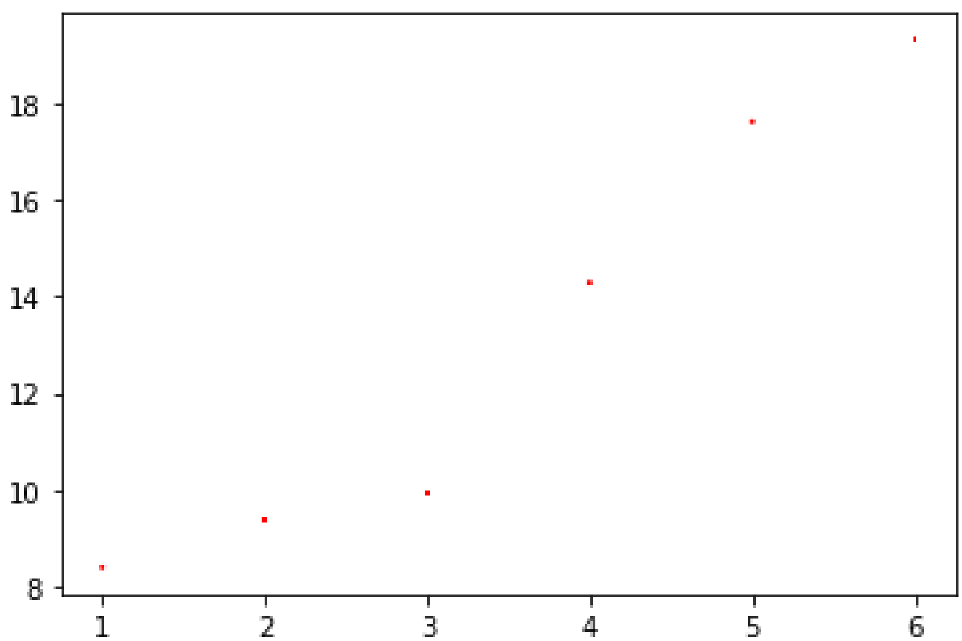


18
16
14
12
10
8
1
2
3
4
5
6



18
16
14
12
10
8
1
2
3
4
5
6

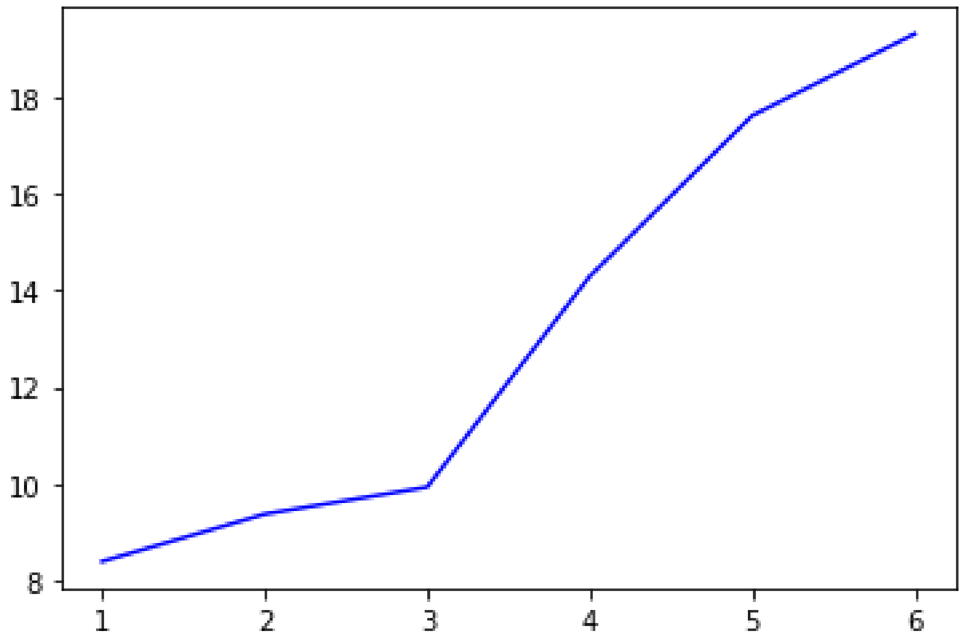


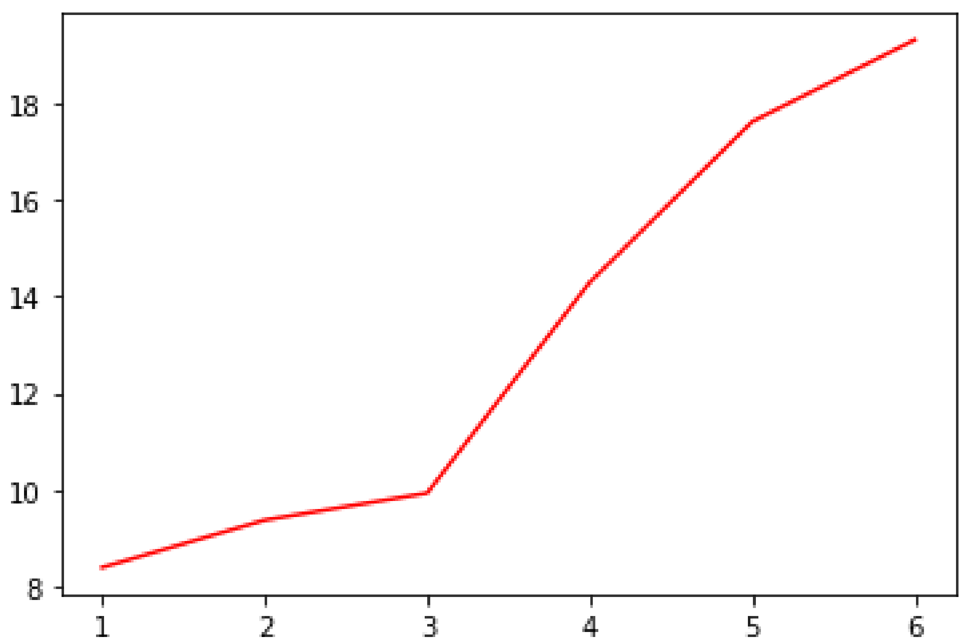


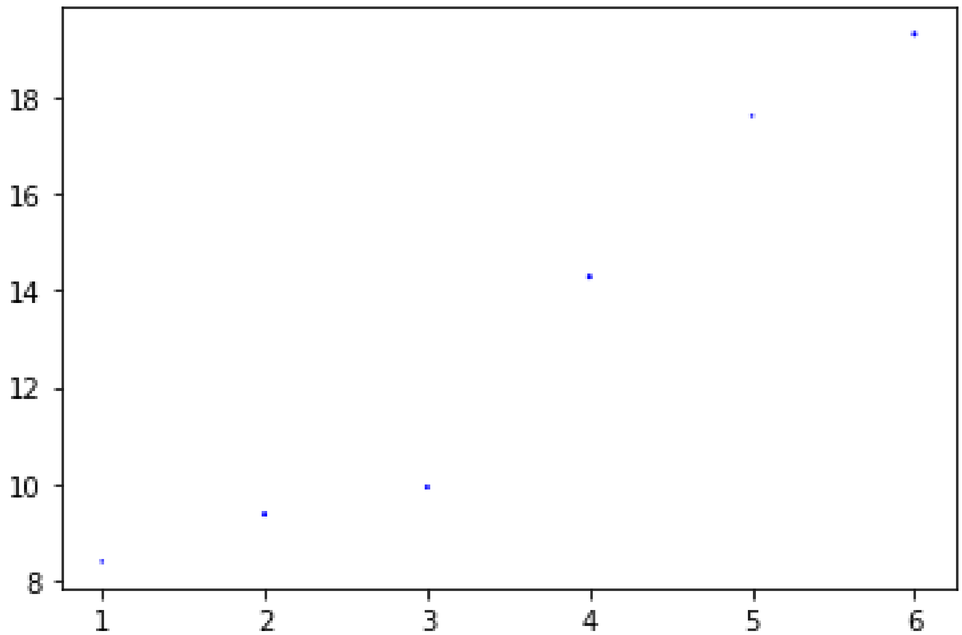
[129]:
```
#Here, we will model Y = a + bx + cx^2 + dx^3 + ex^4 + fx^5

x_train = np.hstack((x,x*x,x*x*x,x*x*x*x,x*x*x*x*x))
print(x_train)
```

```
print(y_train)
regressor.fit(x_train,y_train)
y_predicted = regressor.predict(x_train)
print(y_predicted)

plt.plot(x,y_predicted,'b')
plt.show()
plt.plot(x,y_train,'r')
plt.show()

plt.scatter(x,y_predicted,colors[0],'blue','o')
plt.show()
plt.scatter(x,y_train,colors[1],'red','o')
plt.show()
```

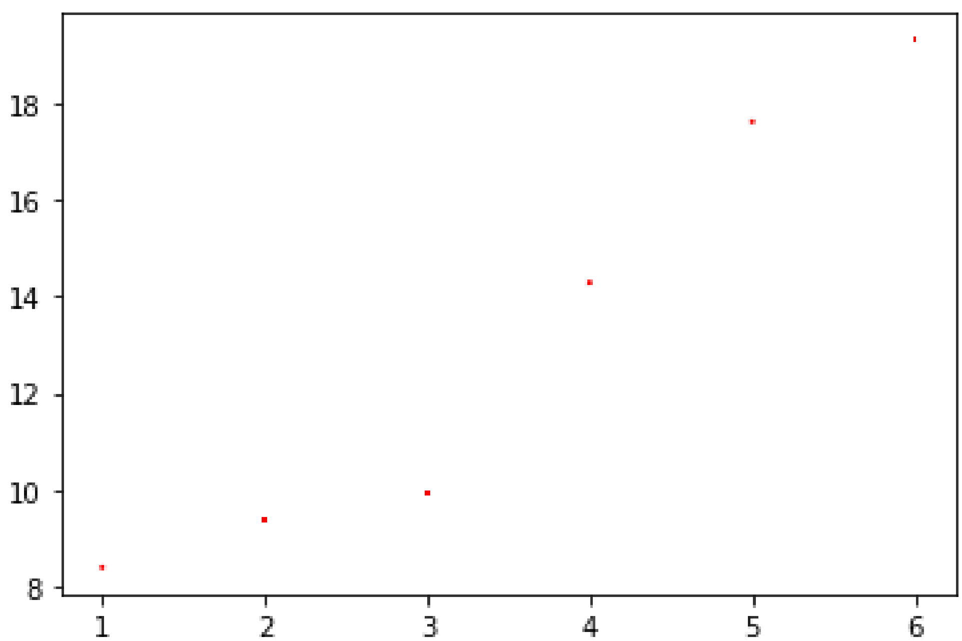
```
[[1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]
 [2.000e+00 4.000e+00 8.000e+00 1.600e+01 3.200e+01]
 [3.000e+00 9.000e+00 2.700e+01 8.100e+01 2.430e+02]
 [4.000e+00 1.600e+01 6.400e+01 2.560e+02 1.024e+03]
 [5.000e+00 2.500e+01 1.250e+02 6.250e+02 3.125e+03]
 [6.000e+00 3.600e+01 2.160e+02 1.296e+03 7.776e+03]]
[[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
[[ 8.38347641]
 [ 9.36536528]
 [ 9.92552356]
 [14.29144017]
 [17.62560549]
 [19.32083131]]
```


18
16
14
12
10
8
1
2
3
4
5
6


18
16
14
12
10
8
1
2
3
4
5
6



18
16
14
12
10
8
1
2
3
4
5
6



18
16
14
12
10
8
1
2
3
4
5
6


```
[ ]:
```

```
[ ]:
```