Question 1

1). We can use a categorical/multinoulli distribution to describe this scenario.

We will have six parameters:

μ_1 μ_2 μ_3 μ_4 μ_5 μ_6.

the probabilities for side 1, 2, 3, 4, 5, 6 comes up.

2). If we have a fair dice then

$\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6 = \frac{1}{6}$

3). If the die always rolls two then

$\mu_2 = 1$, $\mu_1 = \mu_3 = \mu_4 = \mu_5 = \mu_6 = 0$.

4) the domain of parameters is $[0, 1]$

$\mu_1 \in [0, 1]$ $\mu_2 \in [0, 1]$ $\mu_3 \in [0, 1]$

$\mu_4 \in [0, 1]$ $\mu_5 \in [0, 1]$ $\mu_6 \in [0, 1]$

Ïhý uú

Question 2.

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N} \alpha_n \{t_n - w^T\phi(x_n)\}^2$$

$$\frac{\partial}{\partial w} E_D(w) = \frac{1}{2}\sum_{n=1}^{N} 2(t_n - w^T\phi(x_n))(-\phi(x_n)) \cdot \alpha_n$$

$$= \sum_{n=1}^{N} (t_n - w^T\phi(x_n))(-\phi(x_n)) \cdot \alpha_n$$

$$\nabla E_D(w) = \sum_{n=1}^{N} (t_n\alpha_n - w^T\alpha_n\phi(x_n))(-\phi(x_n))^T$$

$$\phi(x_n) = \begin{bmatrix} \phi_0(x_n) \\ \phi_1(x_n) \\ \vdots \\ \phi_{M-1}(x_n) \end{bmatrix} \qquad 0^T = [0, 0, 0 \cdots 0]$$

$$\nabla E_D(w) = \left[\frac{\partial}{\partial w_0}\ln(\cdot), \frac{\partial}{\partial w_1}\ln(\cdot) \cdots \frac{\partial}{\partial w_M}\ln(\cdot)\right]$$

Set the gradient to 0.

$$0^T = \nabla E_D(w) = \sum_{n=1}^{N} (t_n\alpha_n - w^T\alpha_n\phi(x_n))(-\phi(x_n))^T$$

$$0^T = \sum_{n=1}^{N} -t_n\alpha_n\phi(x_n) + w^T \underbrace{\sum_{n=1}^{N} \alpha_n\phi(x_n)\phi(x_n)^T}.$$

$$\boxed{0^T = t^T\Phi^T\Phi - w^T\Phi^T\alpha^T\Phi}$$

Why? ← Using this part as an example:

$$\phi(x_n)\phi(x_n)^T = \begin{bmatrix} \phi_0(x_n) \\ \phi_1(x_n) \\ \vdots \\ \phi_{M-1}(x_n) \end{bmatrix} \begin{bmatrix} \phi_0(x_n) & \phi_1(x_n) & \cdots & \phi_{M-1}(x_n) \end{bmatrix}$$

$M \times 1$ $\quad$ $1 \times M$

$$= \begin{pmatrix} \phi_0(x_n)\phi_0(x_n) & \phi_0(x_n)\phi_1(x_n) & \cdots & \phi_0(x_n)\phi_M(x_n) \\ \phi_1(x_n)\phi_0(x_n) & \phi_1(x_n)\phi_1(x_n) & \cdots & \\ \vdots & & & \end{pmatrix}$$

Ïhý uú

Treat α_n as a scalar then $\alpha_n \phi(x_n) \phi(x_n)^T$ become

$\alpha_n \phi_0(x_n) \phi_0(y_n)$ - - - -

$\alpha_n \phi_1(x_n) \phi_0(x_n)$ - - - -

$\sum_{n=1}^{N} \alpha_n \phi(x_n) \phi(x_n)^T$?

$\alpha_1 \phi_0(x_1) \phi_0(x_1) + \alpha_2 \phi_0(x_2) \phi_0(x_1) + \cdots \alpha_n \phi_0(x_n) \phi_0(x_n)$

$\alpha_1 \phi_1(x_1) \phi_0(x_1) + \alpha_2 \phi_1(x_2) \phi_0(x_2) + \cdots + \alpha_n \phi_1(x_n) \phi_0(x_n)$

$$\underline{\Phi} = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & & & \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix} \quad N \times M$$

$$\underline{\Phi}^T = \begin{pmatrix} \phi_0(x_1) & \phi_0(x_2) & \cdots & \phi_0(x_N) \\ \phi_1(x_1) & \phi_1(x_2) & \cdots & \phi_1(x_N) \\ \vdots & & & \vdots \\ \phi_{M-1}(x_1) & \phi_{M-1}(x_2) & \cdots & \phi_{M-1}(x_N) \end{pmatrix} \quad M \times N.$$

$$\alpha = \begin{pmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & & \alpha_N \end{pmatrix} \quad N \times N$$

Ïhý uú

$\underline{\Phi^T \alpha} =$ $M \times N$

$$\begin{pmatrix} \alpha_1 \phi_0(x_1) & \alpha_2 \phi_0(x_2) & \cdots & \alpha_N \phi_0(x_N) \\ \alpha_1 \phi_1(x_1) & \alpha_2 \phi_1(x_2) & \cdots & \alpha_N \phi_1(x_N) \\ \vdots & \vdots & & \vdots \\ \alpha_1 \phi_{M-1}(x_1) & \alpha_2 \phi_{M-1}(x_2) & \cdots & \alpha_N \phi_{M-1}(x_N) \end{pmatrix}$$

$\underline{\Phi^T \alpha \Phi} =$ $M \times M$

$\rightarrow \underline{\alpha_1 \phi_0(x_1) \phi_0(x_1) + \alpha_2 \phi_0(x_2) \phi_0(x_2) + \cdots}$

$\rightarrow \alpha_1 \phi_1(x_1) \phi_0(x) + \alpha_2 \phi_1(x_2) \phi_0(x_2) + \cdots$

Same as $\begin{pmatrix} \alpha_n \phi_0(x_n) \phi_0(x_n) & \cdots \\ \alpha_n \phi_1(x_n) \phi_0(x_n) & \cdots \\ \vdots & \end{pmatrix}$

$\left.\begin{array}{l} (ABC)^T = C^T B^T A^T \\ (AB)^T = B^T A^T \end{array}\right\}$

Therefore. $O^T = t^T \alpha^T \Phi - w^T \Phi^T \alpha^T \Phi$

$w^T \Phi^T \alpha^T \Phi = t^T \alpha^T \Phi$

$(\Phi^T \alpha t)^T = t^T \alpha^T \Phi$

$(\Phi^T \alpha \Phi) w = \Phi^T \alpha t$

$((\Phi^T \alpha \Phi) w)^T = w^T \Phi^T \alpha^T \Phi$

$w = (\Phi^T \alpha \Phi)^{-1} (\Phi^T \alpha t)$.

Ïhý uú

Question 3

$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2$.

$E(\tilde{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$.

RMS: $E_{RMS} = \sqrt{2E(w^*)/N}$

1) No, The training set and validation set are both randomly distributed data, from the dataset. There is no guarantees on the relationship between training error and validation error.

If the model is overfit the validation error is probably higher than the training error.

Good fit: Validation error low, slightly higher than the training error.

Unknown fit: Validation error low, training error high.

Under-fit: Validation error and training error both high.

Generally speaking, training error will almost always underestimate the validation error. But it is possible for the validation error to be less than the training.

2) Yes. Degree 10 polynomial contains Degree 9 polynomial. The unregularized regression gives us the optimal solution which means Degree 10 polynomial mostly fits the data better. In the worst case, the training error

Ïhý uú

for them two are equal.

But if we change training error to testing error for this question then the answer shuu be "No".

3) No.

In most cases the testing error for regularized regression is lower than unregularized regression since the degree=20 is very high and is highly likely to cause overfitting.

But this is not garuanteed, If we got a weak model then the regularized will underfitting slash the predictie power even more and make the testing error larger compared with the unregularized one.

Question 4.

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 + \frac{1}{2}\sum_{j\in J_1}\lambda_j|w_j| + \frac{1}{2}\sum_{j\in J_2}\lambda_j|w_j|^2.$$

Here, $\lambda_{j\in J_2}$ and $\lambda_{j\in J_1}$ are two subsets/subvector of $\lambda_{n\in N}$,

$$\frac{\partial E(w)}{\partial w} = \frac{1}{2}\sum_{n=1}^{N} 2(t_n - w^T\phi(x_n))(-\phi(x_n)) + \frac{1}{2}\sum_{j\in J_1}\frac{w_j}{|w_j|}\lambda_j + \frac{1}{2}\sum_{j\in J_2}\lambda_j 2w_j$$

$$= \sum_{n=1}^{N}(t_n - w^T\phi(x))(-\phi(x_n)) + \frac{1}{2}\sum_{j\in J_1}\frac{w_j}{|w_j|}\lambda_j + \sum_{j\in J_2}\lambda_j w_j.$$

Define matrices:

$$\varphi = \begin{cases} \frac{w_j}{|w_j|}\lambda_j & \text{for } j\in J_1 \\ 0 & \text{otherwise} \end{cases}$$

$$r = \begin{cases} \alpha_j & \text{for } j\in J_2 \\ 0 & \text{otherwise.} \end{cases}$$

I = Identity matrix = $[1, 1, 1, 1 \ldots]^T$.

$$\nabla E(w) = -t^T\Phi + W^T\Phi^T\Phi + \frac{1}{2}\varphi I^T + r.W^T$$

Here Φ is the same design matrix we declared in das

Ïhý uú

Question 5.

5.1) Niger. 313.7/1000 = 31.37%

Sierra Leone. 185.3/1000 = 18.53%

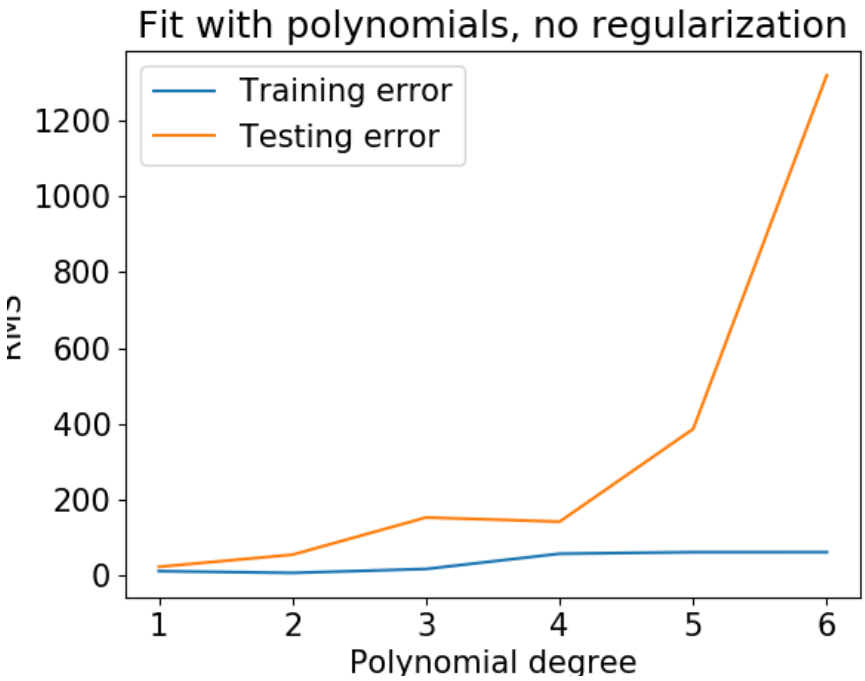
"na. values = "_" will set the missing features to NA values.

Then we will use nanmean() to find the average value for each column/feature, np.where(np.isnan) will find the coordinate/indices for the NA values and we assign the average value to the missing parts according to their indices we found.

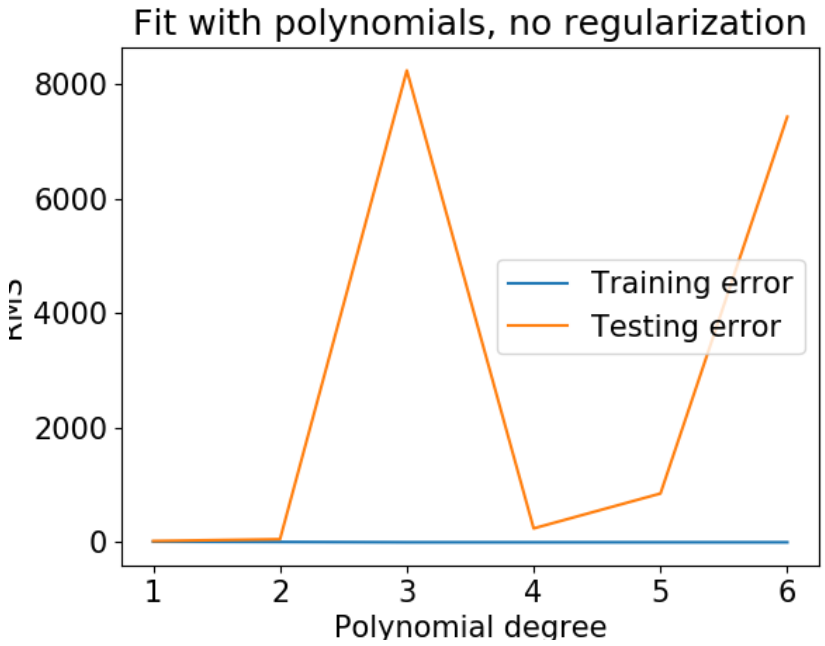
NB: np.where(np.!=nan) will return the row and column indies. But np.take(mean_vals, inds[1]) → we only need the column indies since the average is a 1X40 matrix/vector.
1XN

Ïhý uú

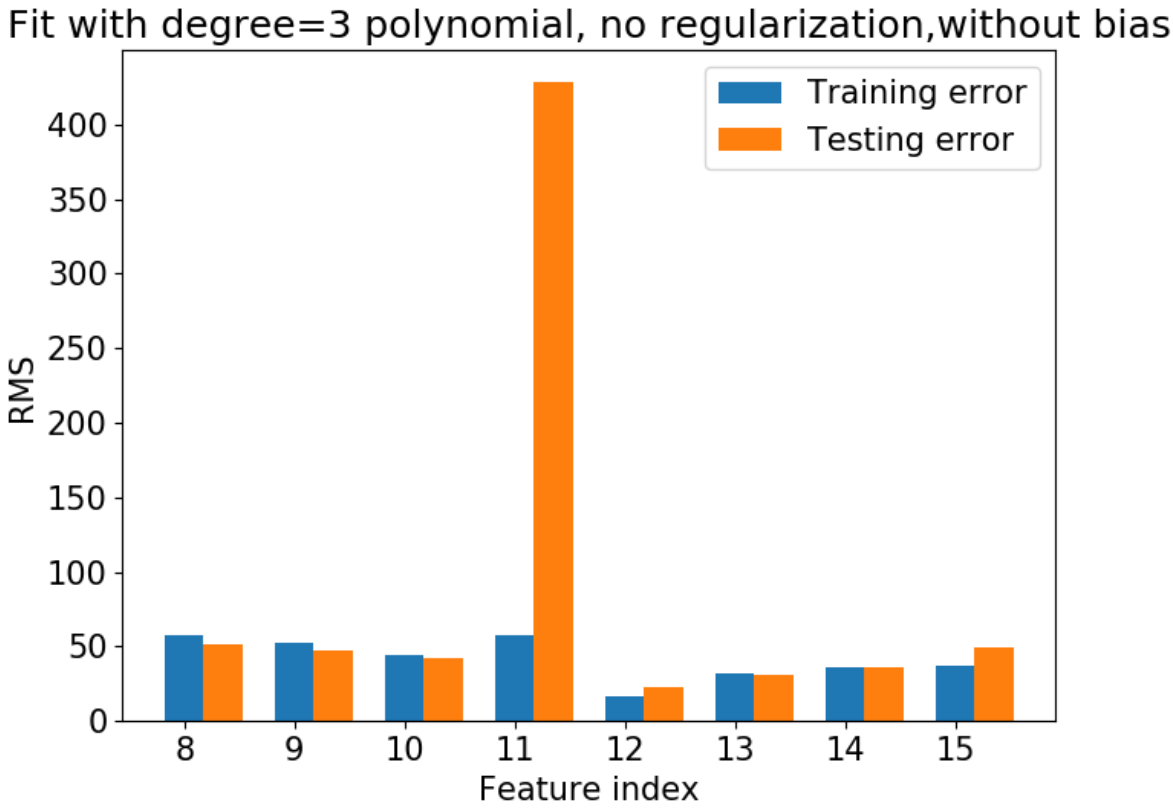
5.2
1)

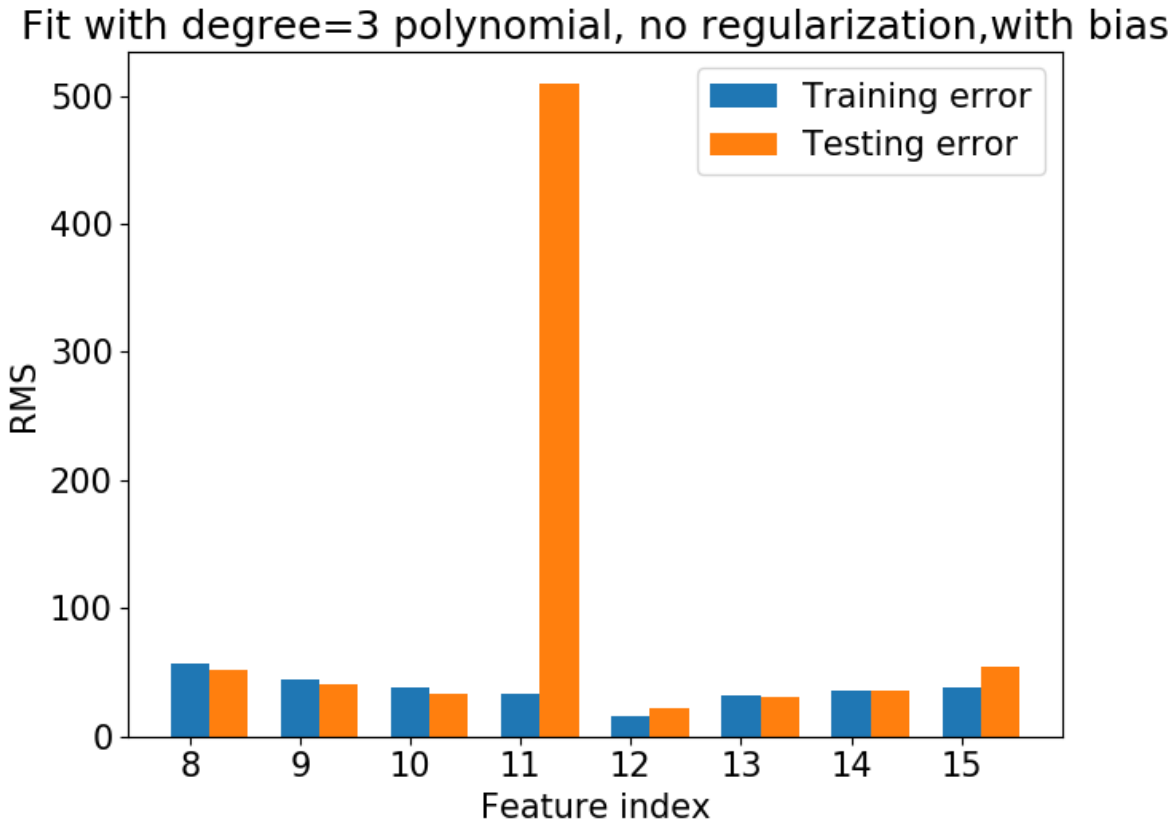


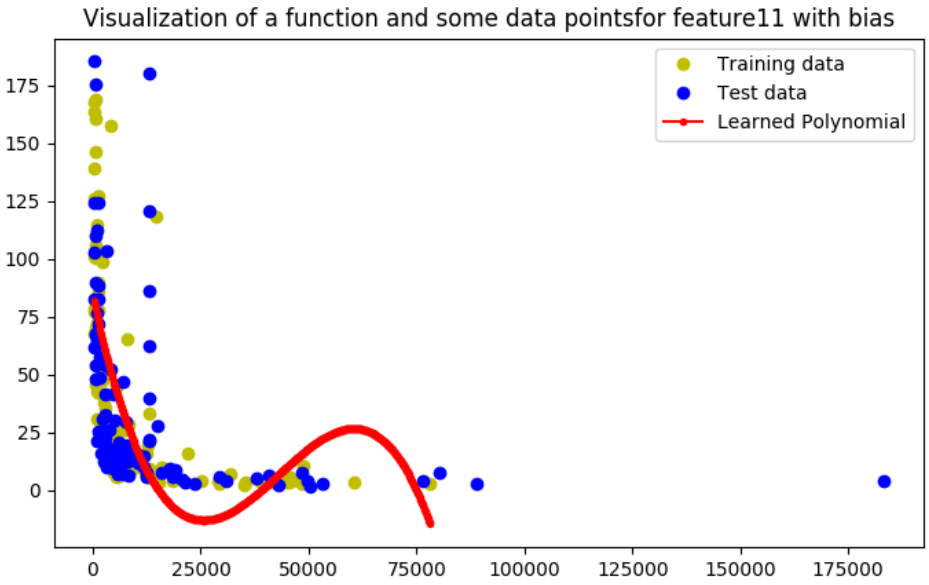
There are two things wrong here. The first one is usually as polynomial degree increases the training error will decrease since higher degree usually fits the data better. After we apply the normalization on input features this problem is solved. The reason is that we may have some features whose orders of magnitudes are larger than others, and they may dominate the objective function and make the estimator unable to learn from other features correctly as expected.
Another problem is as orders of degree increase, the model is being more over-fitted. This problem can be solved with regularized regression which we will do in 5.4.



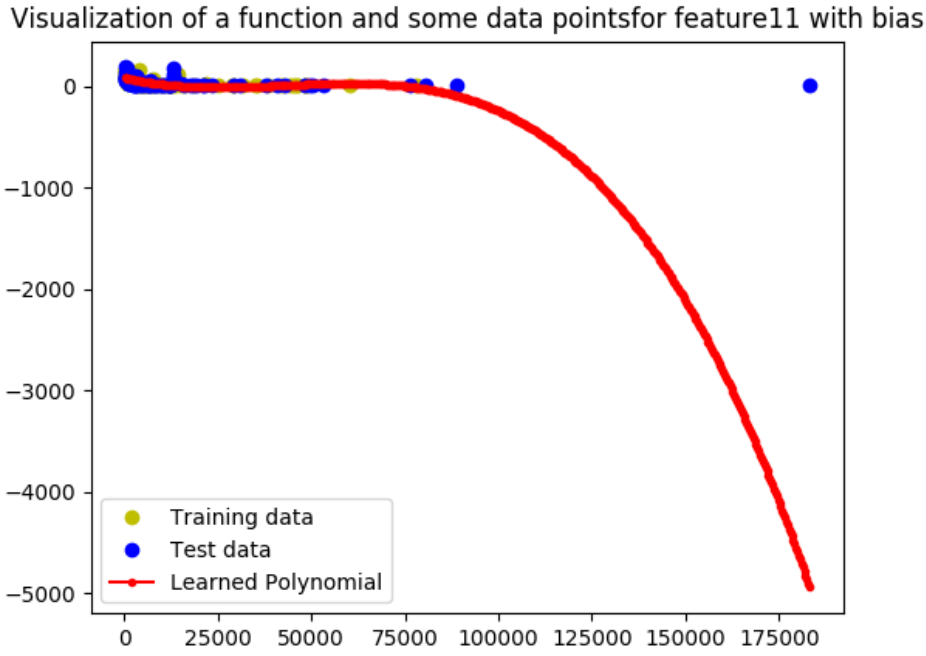
2)

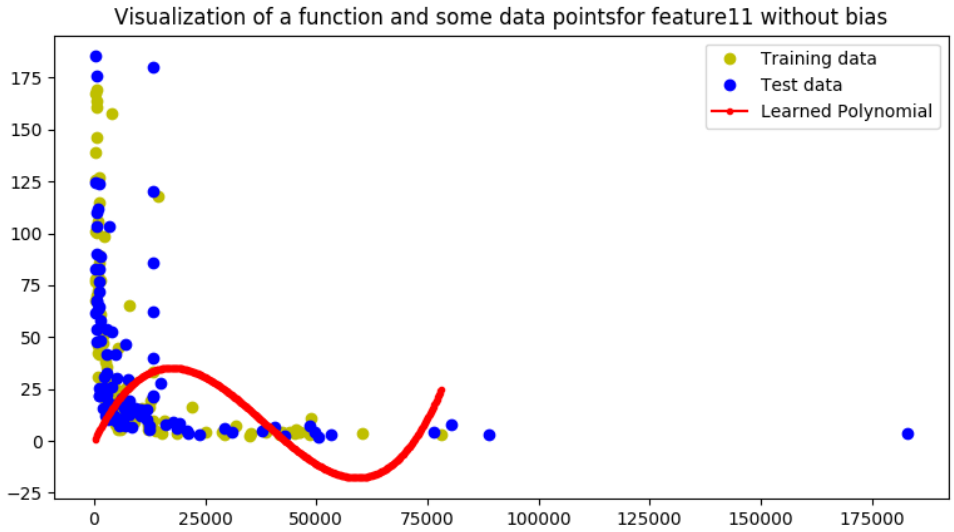

Fit with degree=3 polynomial, no regularization,without bias
Training error
Testing error
RMS
0
50
100
150
200
250
300
350
400
8
9
10
11
12
13
14
15
Feature index

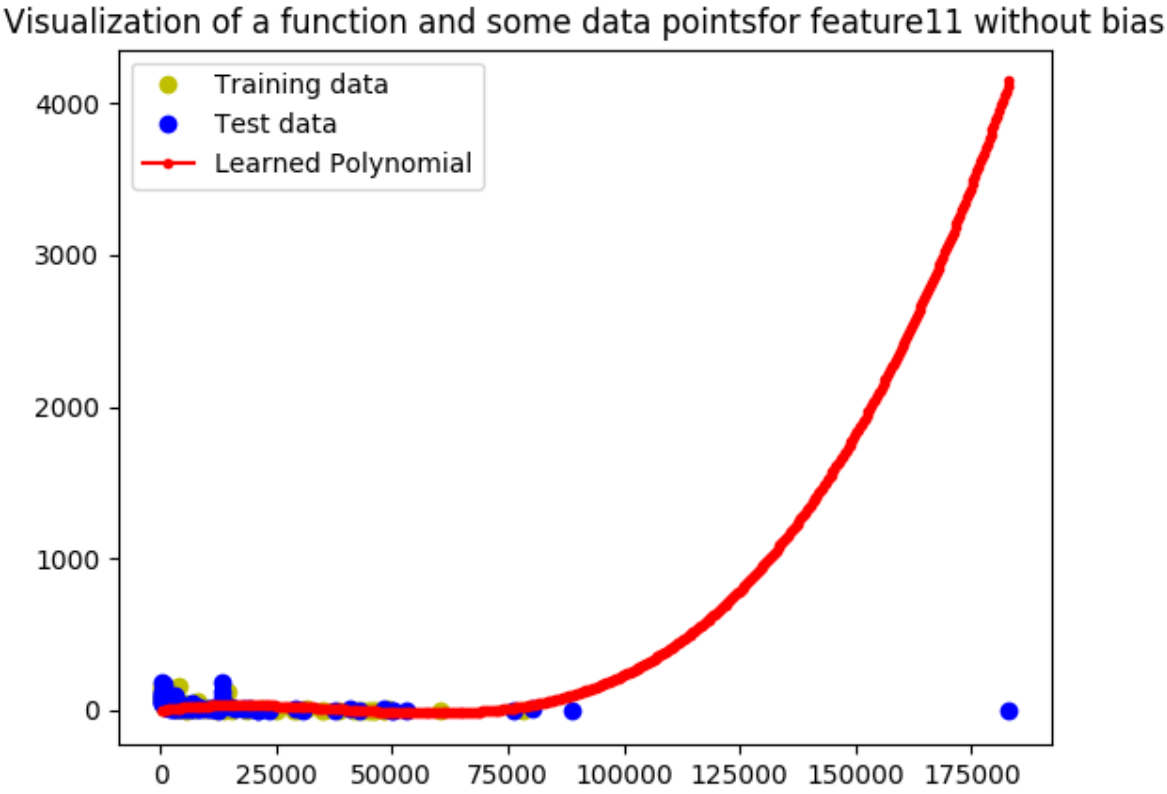

Fit with degree=3 polynomial, no regularization,with bias
Training error
Testing error
RMS
0
100
200
300
400
500
8
9
10
11
12
13
14
15
Feature index



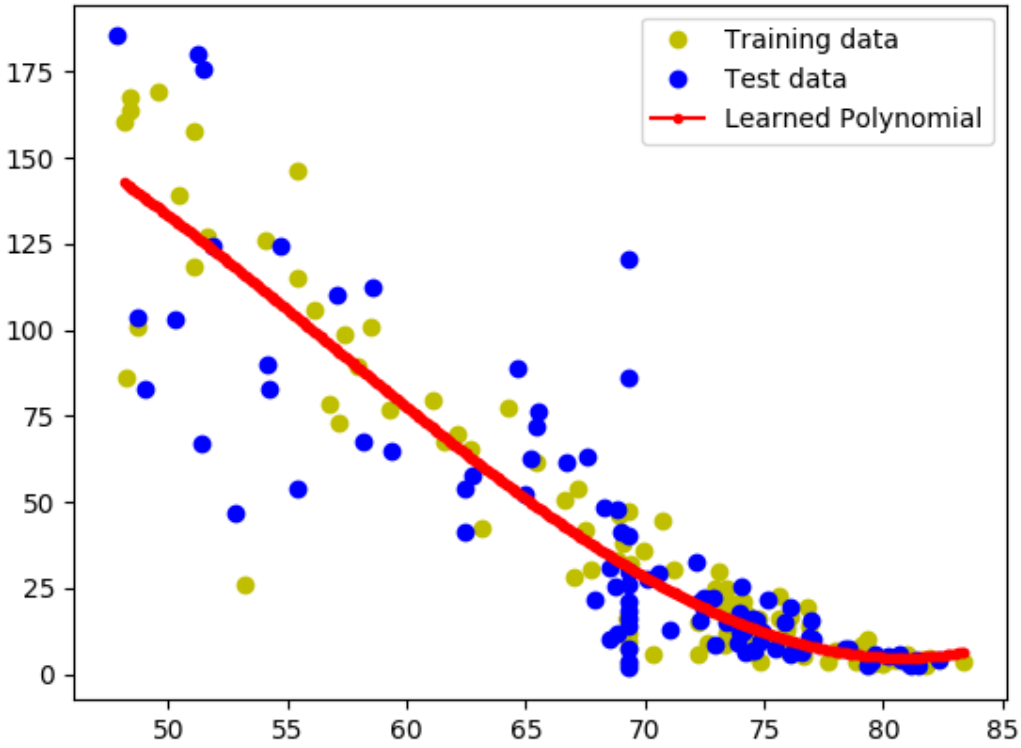
Since there is an outlier so we need to use a larger scale.



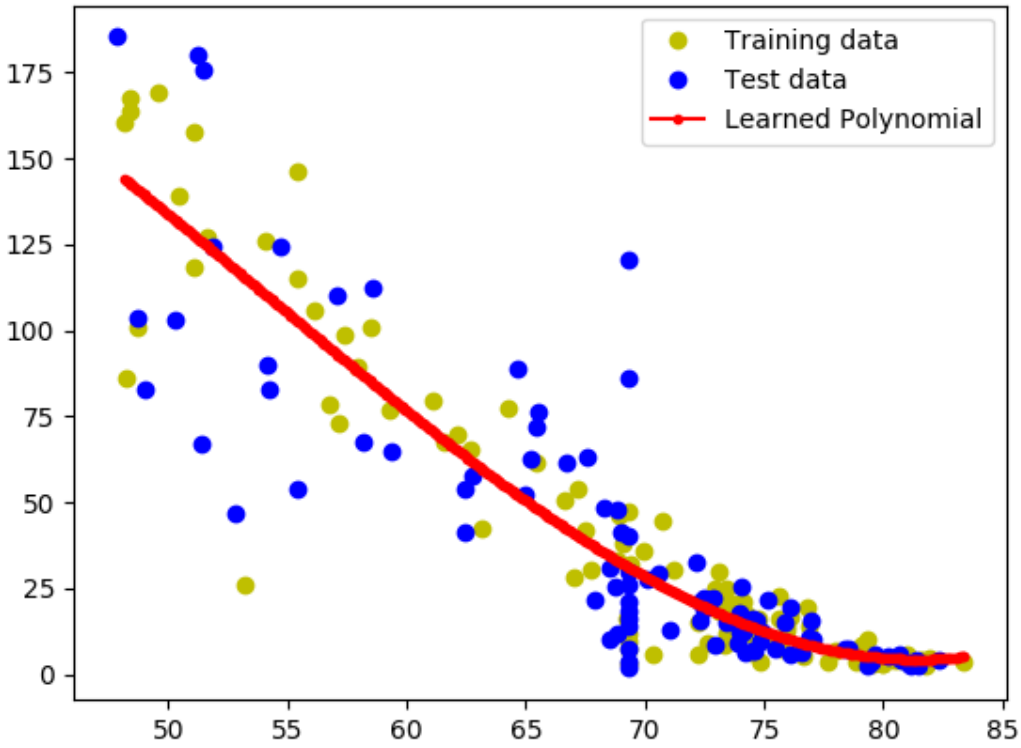

Visualization of a function and some data pointsfor feature11 without bias
Training data
Test data
Learned Polynomial
175
150
125
100
75
50
25
0
−25
0
25000
50000
75000
100000
125000
150000
175000


Visualization of a function and some data pointsfor feature11 without bias
Training data
Test data
Learned Polynomial
4000
3000
2000
1000
0
0
25000
50000
75000
100000
125000
150000
175000

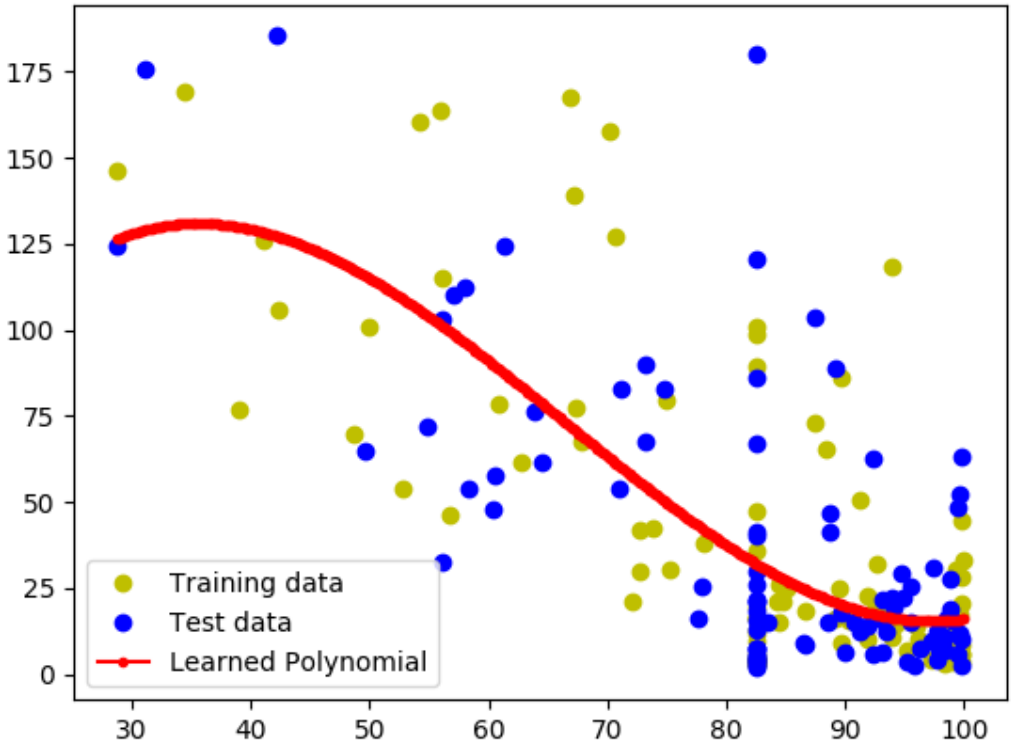
Visualization of a function and some data pointsfor feature12 with bias



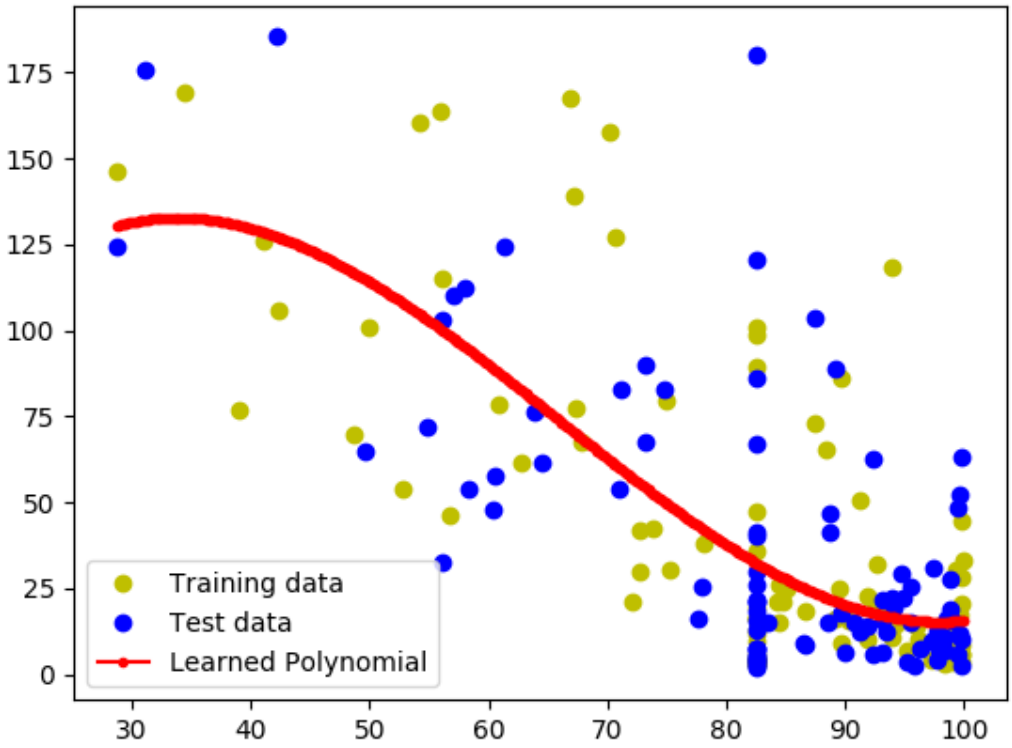
Visualization of a function and some data pointsfor feature12 without bias



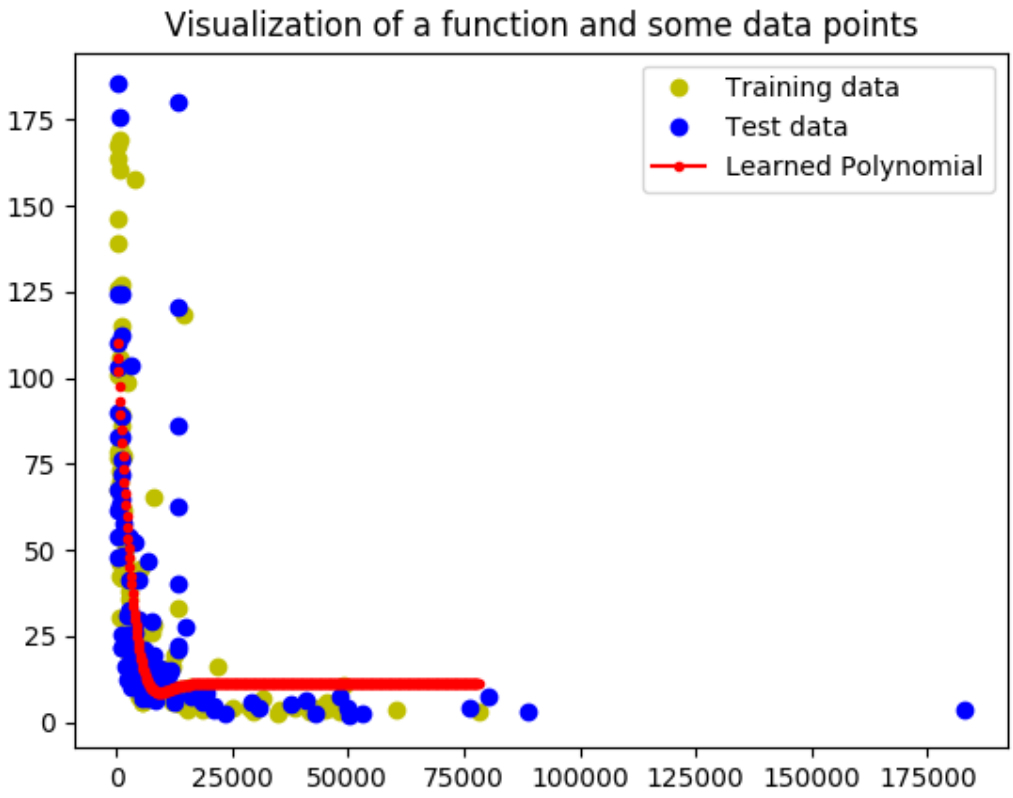
Visualization of a function and some data pointsfor feature13 with bias



Visualization of a function and some data pointsfor feature13 without bias

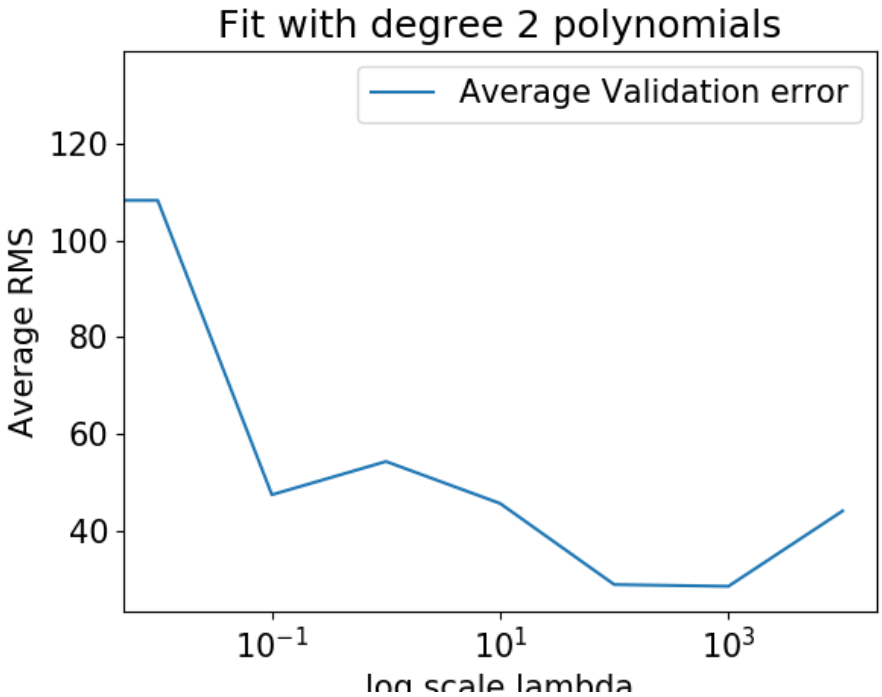


5.3



training error: [[28.45793776]]
testing error: [[33.8067249]]

5.4



the errors for lambda 0 to 1000 are :
lambda=0 134.08724800120225
lambda=0.01 108.33938137863713
lambda=0.1 47.42014676680511
lambda=1 54.30153018685256
lambda=10 45.61747891270658
lambda=100 28.827211650792474
lambda=1000 28.46922327978877
lambda=10000 44.06076704806573

Lambda=1000 yields the lowest average validation error so we will choose lambda=1000