1) Probabilistic Modelling

1-1) What is the parameter that explains the behaviour of the die in this case (in analogy to the μ for the coin)?

Since it's a Multinomial variable, the parameter M will be a vector in which each element holds the probability of showing that element when throwing a die. So $\mu=[\mu_1,\mu_2,\mu_3,\mu_4,\mu_5,\mu_6]^T$.

1-2) What is the value of the parameter for a fair die (equal probability of rolling any number)?

If we have a fair die, we will have equal probability for showing up each element.
So $\mu=[\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}]$

1-3) What is the value of the parameter for a die that always rolls a 2?

If a die always shows 2 therefore $\mu=[0,\frac{1}{6},0,0,0,0]$

1-4) Specify the domain of the parameter – which settings of the parameter are valid?

Because it's a 6-sided die and in each throw we will see one side is up, so the domain of this problem is $x\in\{1,2,3,4,5,6,\}$

2) Weighted Square Error

We have : $E_D(w)=\frac{1}{2}\sum_{n=1}^{N}[\alpha_n(t_n-w^T\Phi_{x_n})]^2$

$$\frac{\delta}{\delta_{w_i}}=\frac{1}{2}\sum_{n=1}^{N}[2\alpha_n(t_n-w^T\phi_{x_n})\phi_{x_n}^T] = \sum_{n=1}^{N}[\alpha_n(t_n-w^T\phi_{x_n})\phi_{x_n}^T] = \sum_{n=1}^{N}[\alpha_n t_n\phi_{x_n}^T-\alpha_n W^T\phi_{x_n}\phi_{x_n}^T]$$

We set gradient equal to zero and solve :

$$0^T = \sum_{n=1}^{N}[\alpha_n t_n\phi_{x_n}^T]-W^T\sum_{n=1}^{N}[\alpha_n\phi_{x_n}\phi_{x_n}^T]$$

$$W^T=\frac{\sum_{n=1}^{N}[\alpha_n t_n\phi_{x_n}^T]}{\sum_{n=1}^{N}[\alpha_n\phi_{x_n}\phi_{x_n}^T]} = \sum_{n=1}^{N}[\alpha_n\phi_{x_n}\phi_{x_n}^T]^{-1}\times\sum_{n=1}^{N}[\alpha_n t_n\phi_{x_n}^T]$$

Declare a matrix $\theta_{N\times M}$ to remove summation:

$$\mathbf{\Phi}=\begin{pmatrix}\phi_0(\boldsymbol{x}_1) & \phi_1(\boldsymbol{x}_1) & \cdots & \phi_{M-1}(\boldsymbol{x}_1)\\ \phi_0(\boldsymbol{x}_2) & \phi_1(\boldsymbol{x}_2) & \cdots & \phi_{M-1}(\boldsymbol{x}_2)\\ \vdots & \vdots & \ddots & \vdots\\ \phi_0(\boldsymbol{x}_N) & \phi_1(\boldsymbol{x}_N) & \cdots & \phi_{M-1}(\boldsymbol{x}_N)\end{pmatrix}$$

Taking transpose of both sides we have:
$W=(\theta^T\alpha\theta)^{-1}\theta^T\alpha t$

3) Training vs. Test error

3-1) Suppose we perform unregularized regression on a dataset. Is the validation error always higher than the training error? Explain.

False.
Usually this is the case that validation error is higher than training error. But this is NOT TRUE always. It may happen that we learn a function based on training samples, and later when it comes to test phase, test points lie exactly on the learned curve or very near the curve that leads to minimized validation error lower than training error.

3-2) Suppose we perform unregularized regression on a dataset. Is the training error with a degree 10 polynomial always lower than or equal to that using a degree 9 polynomial? Explain.

True.
A polynomial degree M contains all lower degree polynomials so it is capable of generating results at least as good as a polynomial degree 9.

3-3) Suppose we perform both regularized and unregularized regression on a dataset. Is the testing error with a degree 20 polynomial always lower using regularized regression compared to unregularized regression? Explain.

True.
Unregularized regression results in over-fitting. In contrast, regularized regression gives a much closer representation of training data by putting a penalty on large w coefficient and avoiding wild oscillations. This helps to have a lower testing error in regularized regression compared to unregularized regression.

4) Basis Function Dependent Regularization
Error function for L1 (Lasso) is :

$$\nabla E(w)=\frac{1}{2}\sum_{n=1}^{N}[y(x_n,w)-t_n]^2+\frac{\lambda}{2}\sum_{j=1}^{p}|w_j|$$

Error function for L2 (Ridge) is :

$$\nabla E(w)=\frac{1}{2}\sum_{n=1}^{N}[y(x_n,w)-t_n]^2+\frac{\lambda}{2}\sum_{j=1}^{p}|w_j^2|$$

Since we would like to have different lambda for different weights, the formulas change to :

$$\nabla E(w)=\frac{1}{2}\sum_{n=1}^{N}[y(x_n,w)-t_n]^2+\frac{1}{2}\sum_{j=1}^{p}\lambda_j|w_j|$$

$$\nabla E(w)=\frac{1}{2}\sum_{n=1}^{N}[y(x_n,w)-t_n]^2+\frac{1}{2}\sum_{j=1}^{p}\lambda_j|w_j^2|$$

So we can derive that the loss function formula for this scenario:

$$\nabla E(w)=\frac{1}{2}\sum_{n=1}^{N}[y(x_n,w)-t_n]^2+\frac{1}{2}\sum_{j=1}^{M}\lambda_j|w_j|+\frac{1}{2}\sum_{j=1}^{M}\lambda_j|w_j^2|$$

5) Regression

5-1) Getting Started

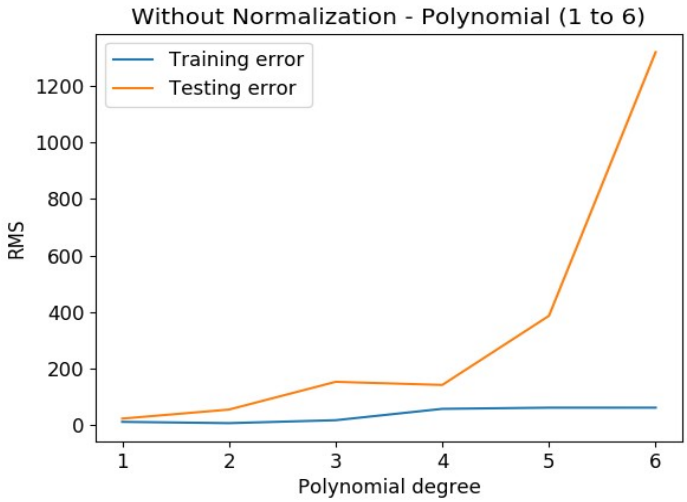
1. Which country had the highest child mortality rate in 1990? What was the rate?
Niger , 313.7

2. Which country had the highest child mortality rate in 2011? What was the rate?
Sierra Leone , 185.3

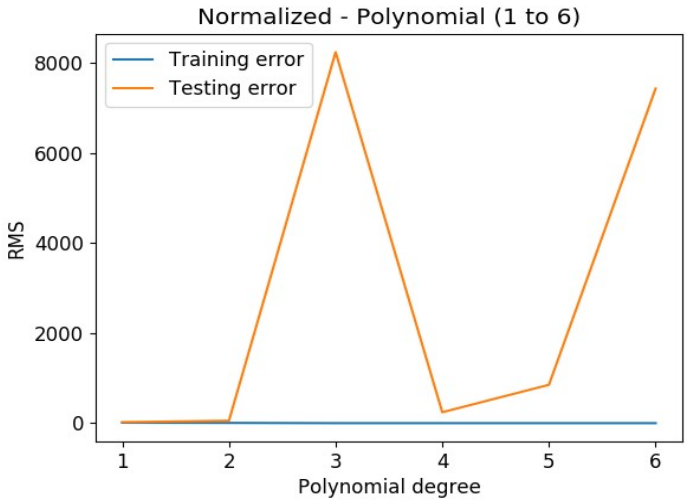
3. Some countries are missing some features (see original .xlsx/.csv spreadsheet). How is this handled in the function assignment1.load_unicef_data()?
For each feature, it calculates the average of non-N/A values in other countries and use that for N/A values.

5-2) Polynomial Regression

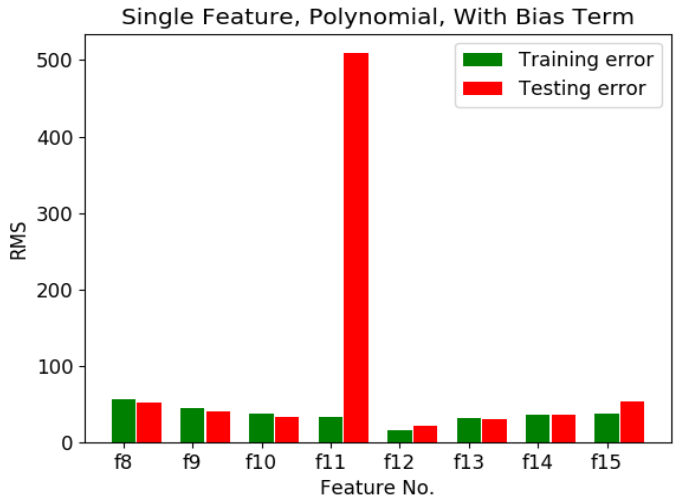
5-2-1) Following is the plot for polynomials degree 1 to 6 without normalization. The considerable point is that error rate has increased for training data as the polynomial degree increases.



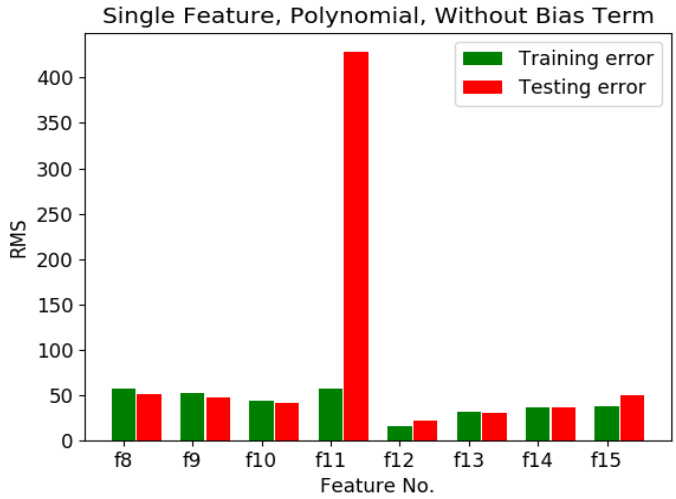
Plot after applying normalize_data() method:



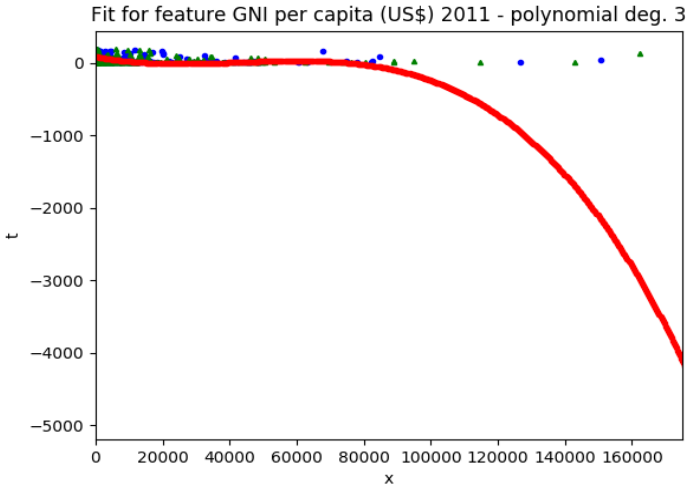
5-2-2) Here is the plot for fitting polynomial of degree 3 to each feature (features 8 to 15) including bias term:

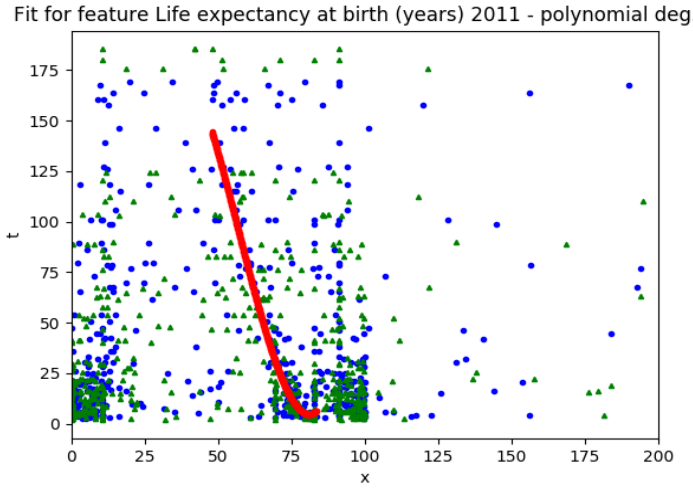


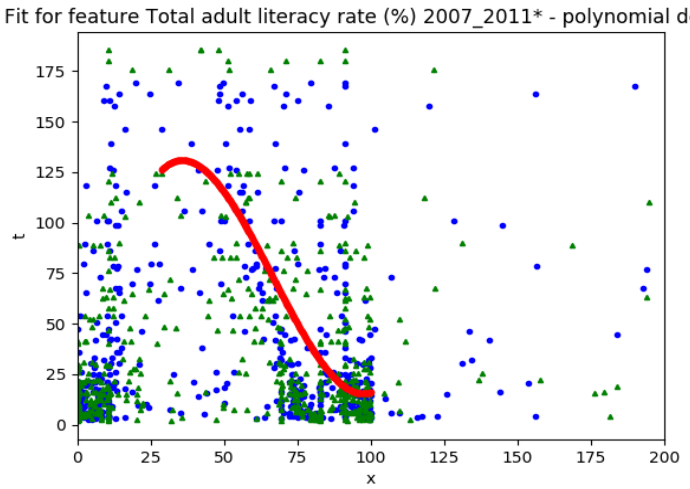
and this is the plot for fitting polynomial of degree 3 to each future (features 8 to 15) without bias term:



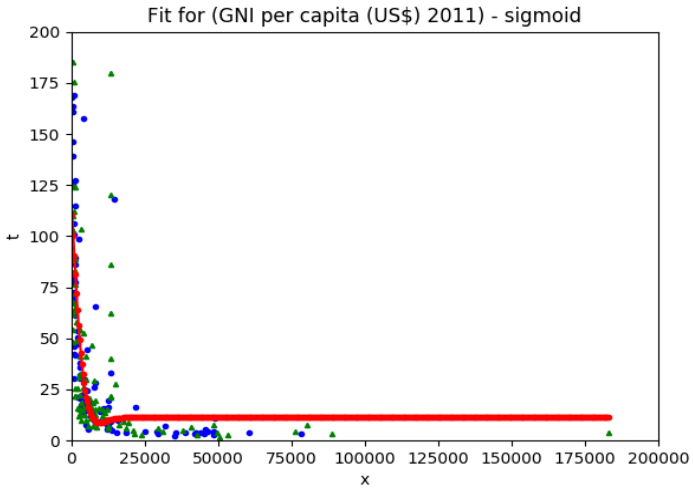
These are plots for fitting to polynomial degree 3 for features (11 , 12 , 13) :

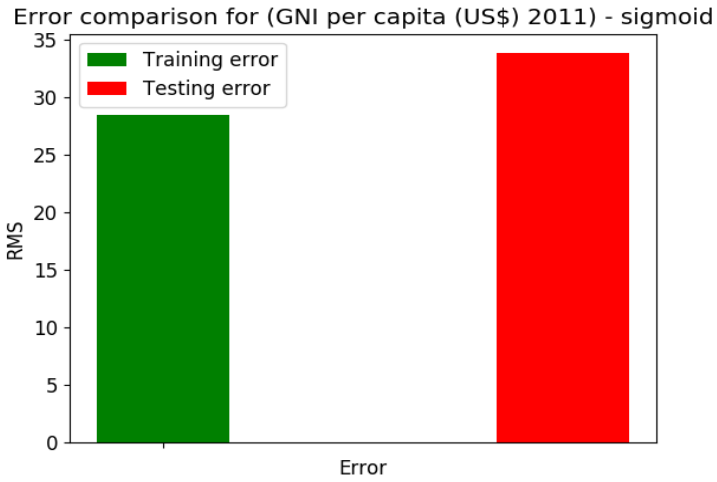



Fit for feature Life expectancy at birth (years) 2011 - polynomial deg. 3
t
x
0
25
50
75
100
125
150
175
200


Fit for feature Total adult literacy rate (%) 2007_2011* - polynomial deg. 3
t
x
0
25
50
75
100
125
150
175
200

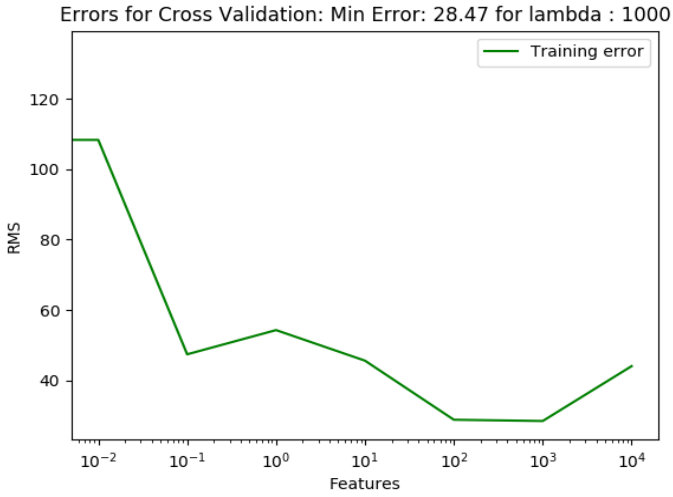
5-3) Sigmoid


Fit for (GNI per capita (US$) 2011) - sigmoid
t
x
0
25
50
75
100
125
150
175
200
0
25000
50000
75000
100000
125000
150000
175000
200000



Based on above bar chart: training error: [28.45793776273188] , testing error: [33.806724900990964]

5-4) Regularized Polynomial Regression



The minimum error was 28.47 for lambda : 1000