CS5487 Programming Assignment 1 Regression

Name: WANG Huanchen

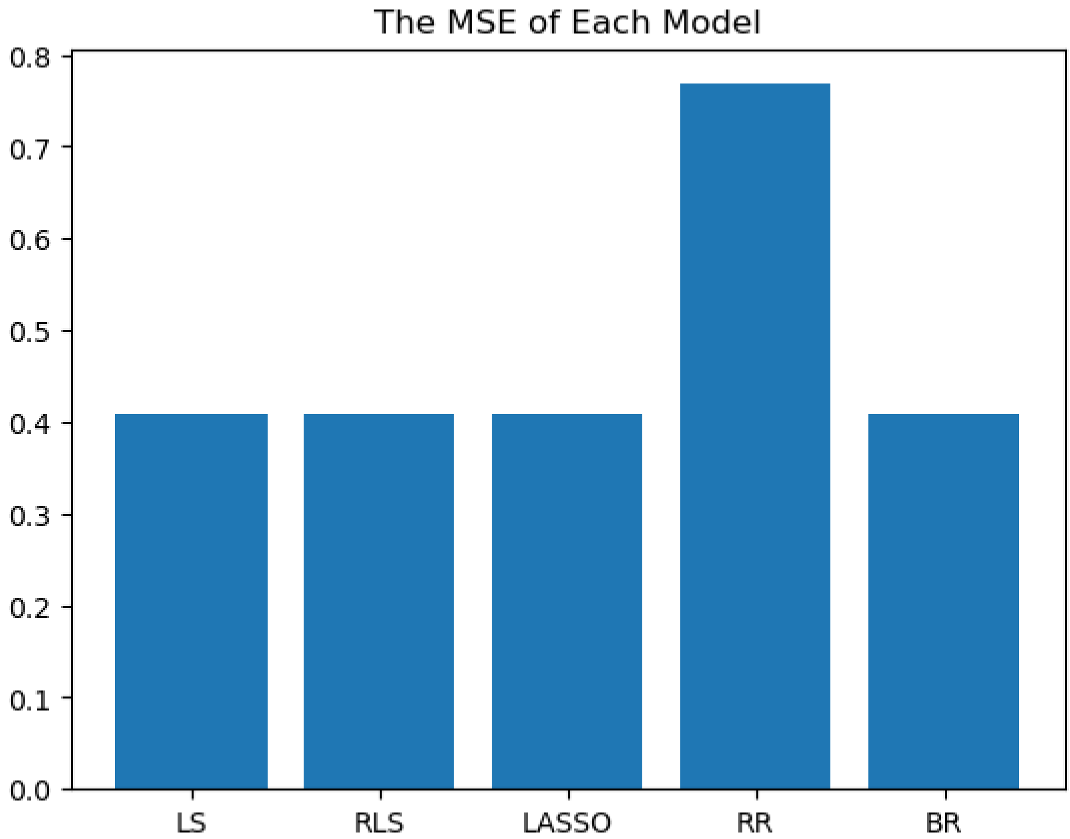
Student ID: 57558749

Part one

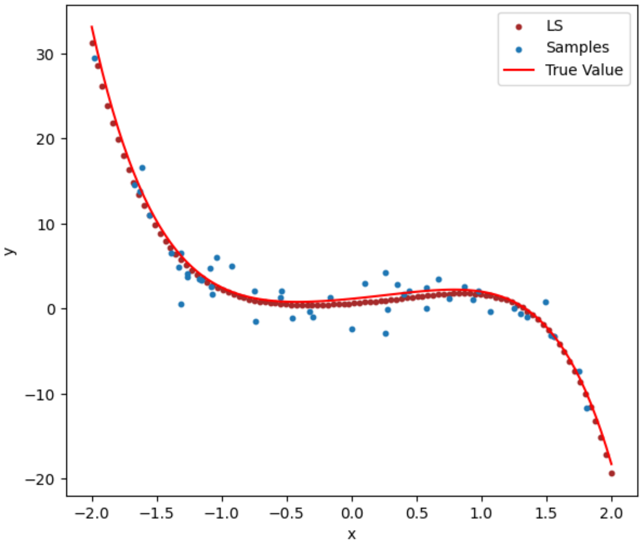
Part 1 Problem a and b

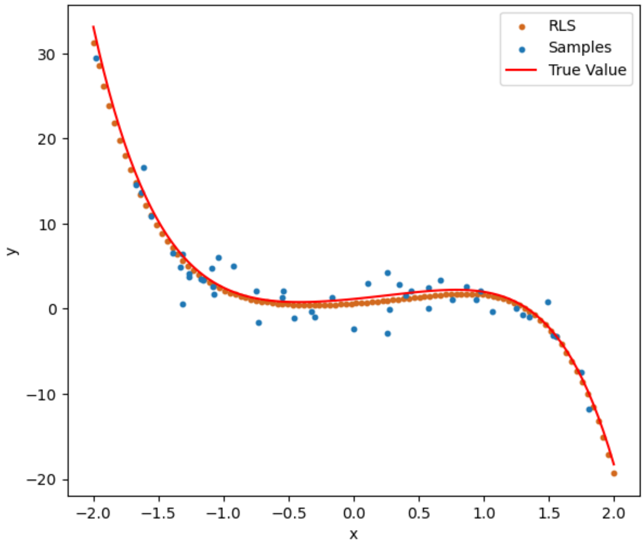
1. To the problem a: those 5 models have been implemented from the above cell codes.
2. To the problem b:
   - Each model's estimated function using polyx as inputs, along with the sample data was ploted as below. And for BR, their standard deviation around mean was also ploted.
   - What is the mean-squared error between the learned function outputs and the true function outputs (polyy), averaged over all input values in polyx?: According to the below output, the MSE:
     `MSE of LS: 0.40864388356990694`
     `MSE of RLS: 0.40856585482590013`
     `MSE of LASSO: 0.4086438835746234`
     `MSE of RR: 0.768046150513354`
     `MSE of BR: 0.4086325708837267`
   - For algorithms with hyperparameters, select some values that tend to work well: I implement a function called **hyperpara** for some models, which need hyperparameters. This function can choose a range of hyperparametes and train the model by each of them. It can select a hyperparameter, which make the estimate's error is minimum. In **anlysis** function, can see that invoking.

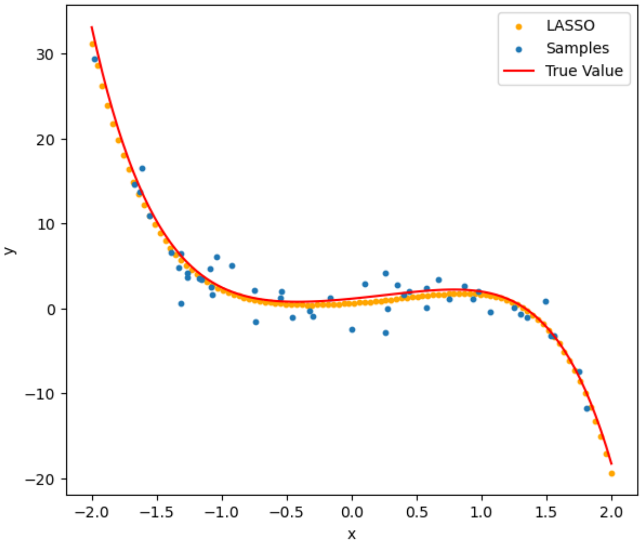
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

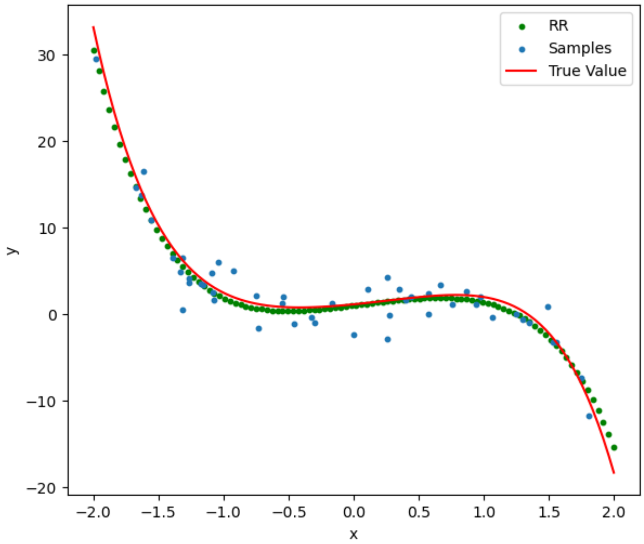


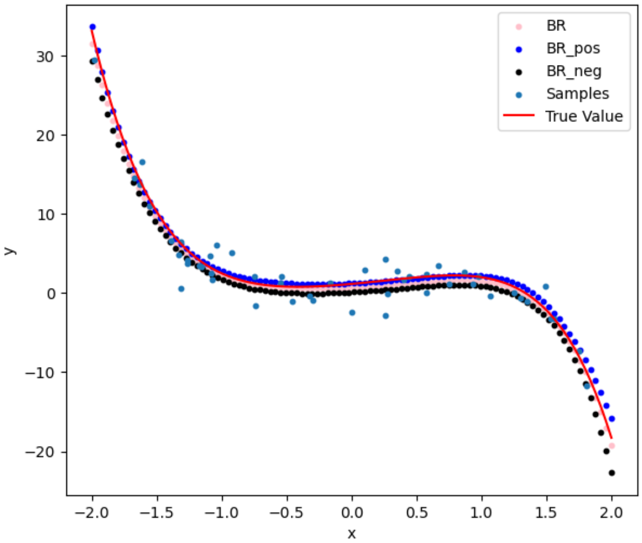

```
MSE of LS: 0.40864388356985565
MSE of RLS: 0.40856585482579666
MSE of LASSO: 0.4086438835745854
MSE of RR: 0.7680461505133516
MSE of BR: 0.4086325708836395
```











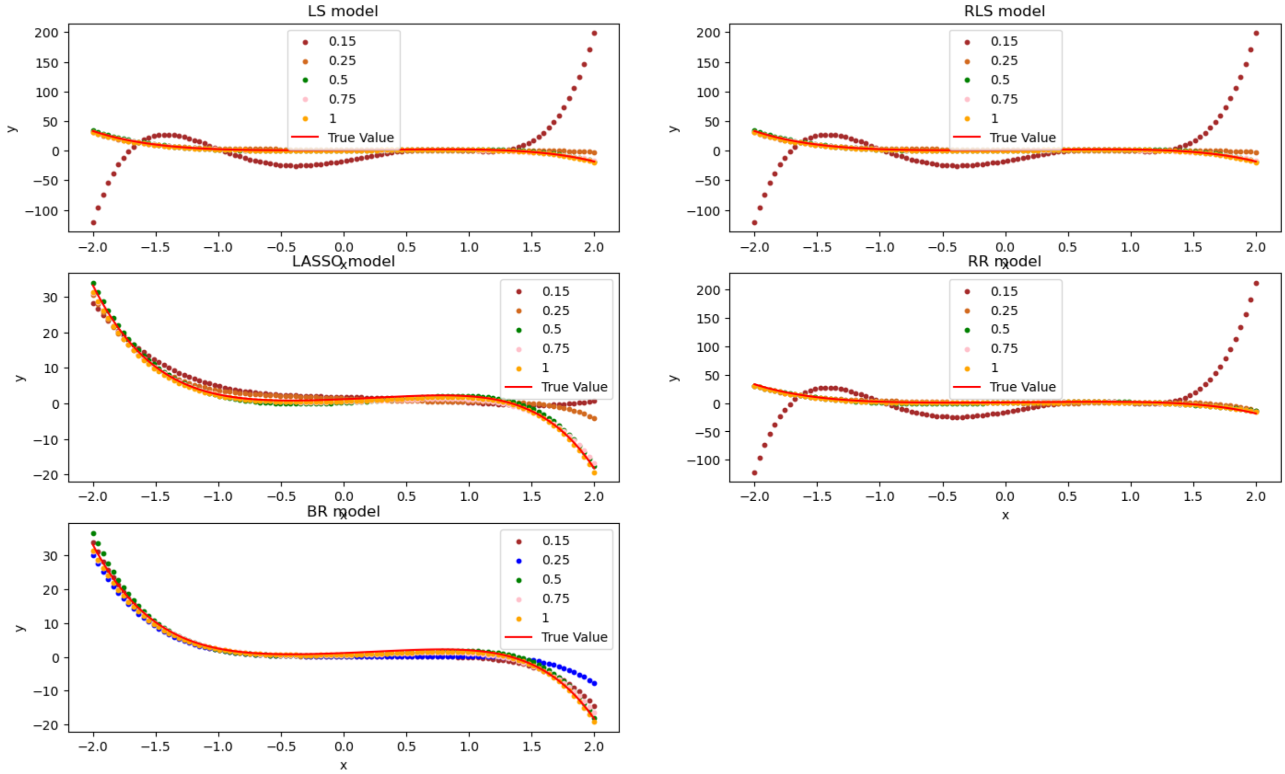
Problem c

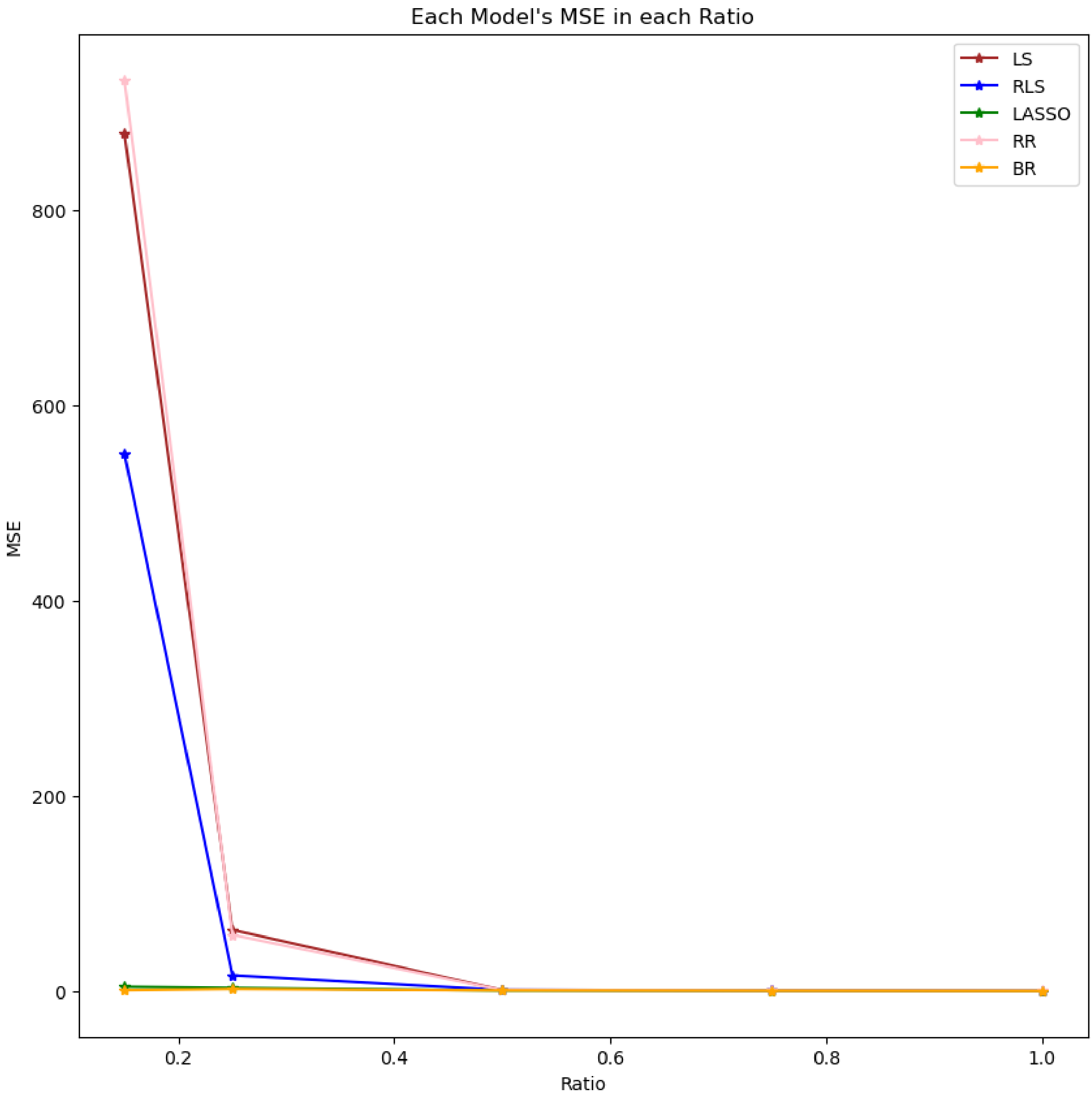
In this problem, I use the ratio set {0.15, 0.25, 0.5, 0.75, 1} to random select a sub-sample dataset. and each model with different ratios was ploted as followed

- Which models are more robust with less data: According to the charts, can get the LASSO and BR are more robust.

- Which tend to overfit: The LS, RLS and RR are likely to overfit, especially LS. Make a plot of error versus training size: I run each model with different ratio in 5 timse and calculate each model's average of MSE and plot them in one chart as following shown.
- Comment on any important trends and findings:
  1. Since the LASSO is use the first normal form (L1) for θ to regulate the regression of least square way. Its performance in small sample is better than LS, RLS and RR. To these three models, LS just do the common least square way for regression without regulation; RLS just give a constant λ to regualte the regression; the way of getting θ from RR is use the L1 for the $||y - \Phi(t)||$, which cannot regular the small sample.
  2. To the BR, since Bayesian Estimation is estimating the distribution of the sample, which can lower the influence of the small sample to overfit.
  3. From the figures bellow, we may observe that for almost all regression methods, their MSEs tend to increase when the training data size decreases.
  4. Another finding is that RLS, BR, LASSO have similar behavior which may be interpreted that all of them have an equivalent Bayesian representation.

```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWarning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWarning: invalid value encountered in multiply
```

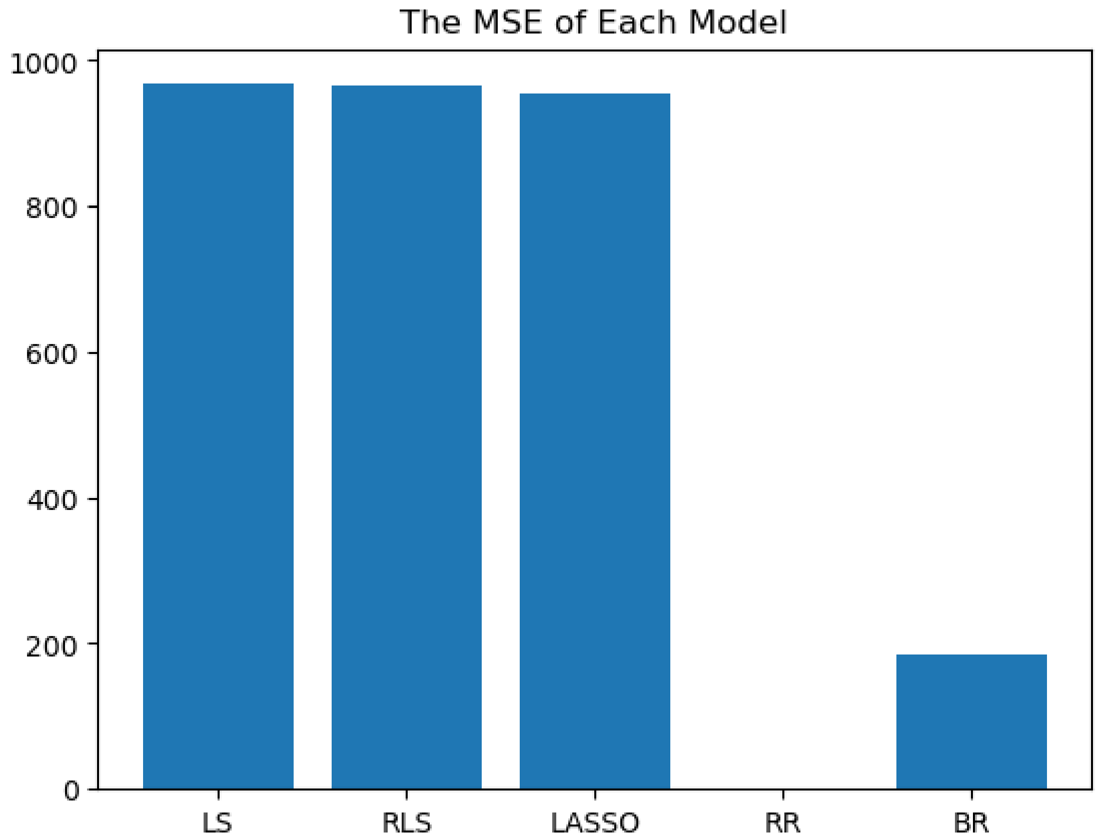




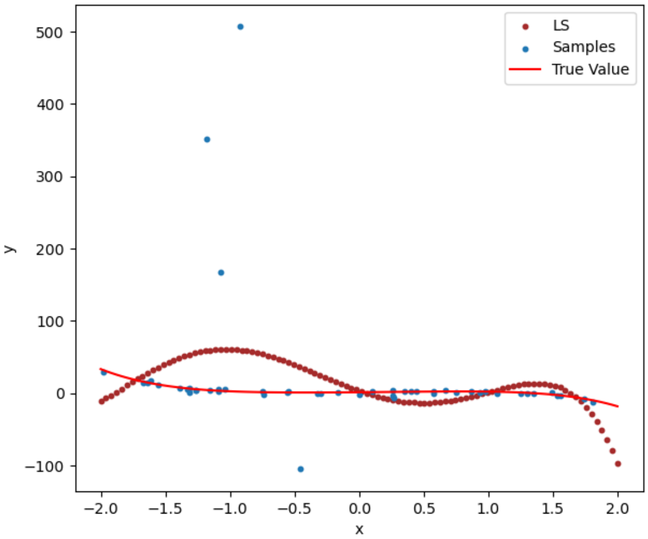
Problem d

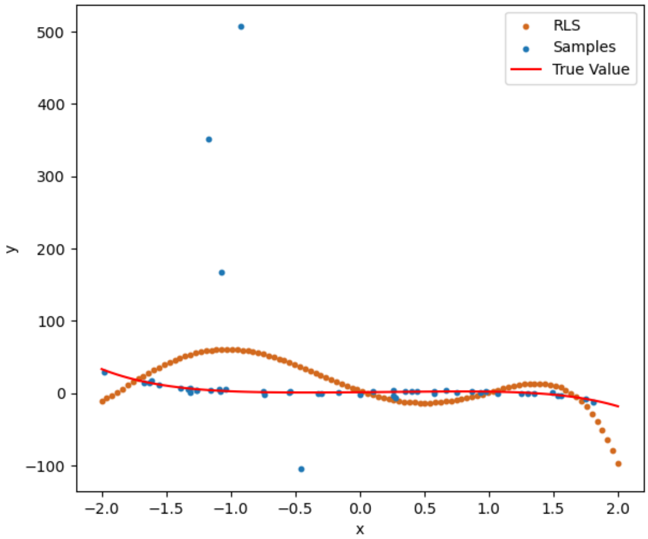
In this problem, I add some outlier is 500 time of orignal sampley. And each model performance with outliers was displayed in below charts. According to the MSE of each model, the RR model performed best, and LS model performed sensitive to those outliers. Because the least square will enpower each sample with sample weight, which lead those models perform non-ideal when the sample has some outliers can these outlier cannot be filtered but need to be trained for regression. Besides, since those least square use the L_2 norm, which is known to be prone to large estimation error if there are outliers in the training sample. But to the RR, it down-weights the influence of outliers, which makes their residuals larger and easier to identify.

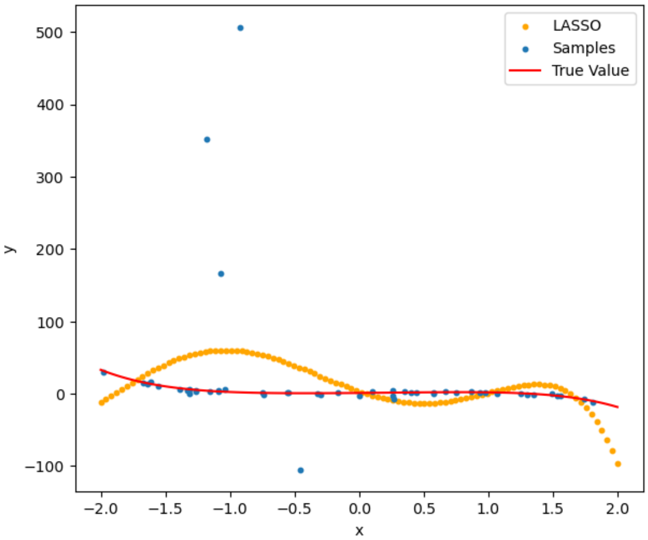
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

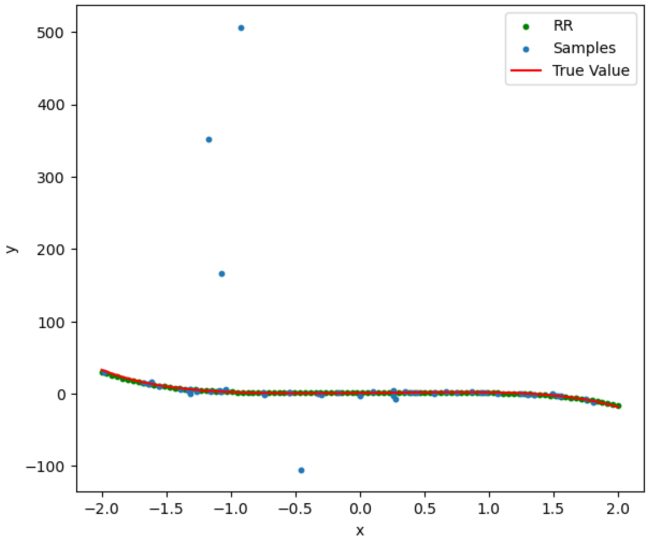


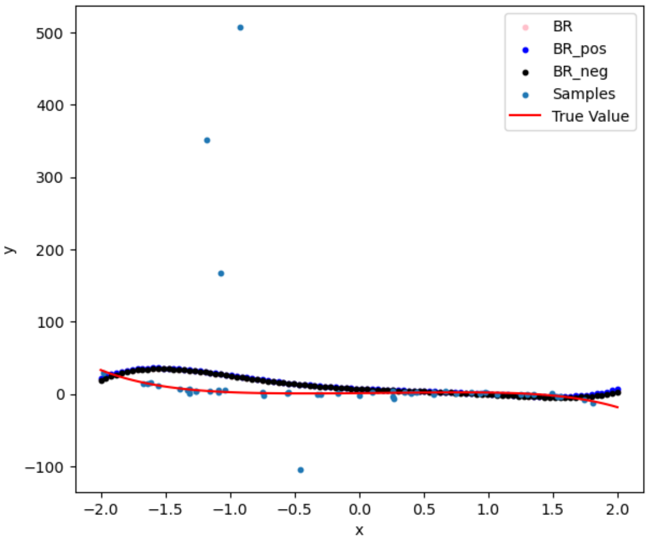

```
MSE of LS: 967.2361100466841
MSE of RLS: 966.4695217913905
MSE of LASSO: 955.3156465633906
MSE of RR: 0.5635669623737695
MSE of BR: 183.63440947515136
```







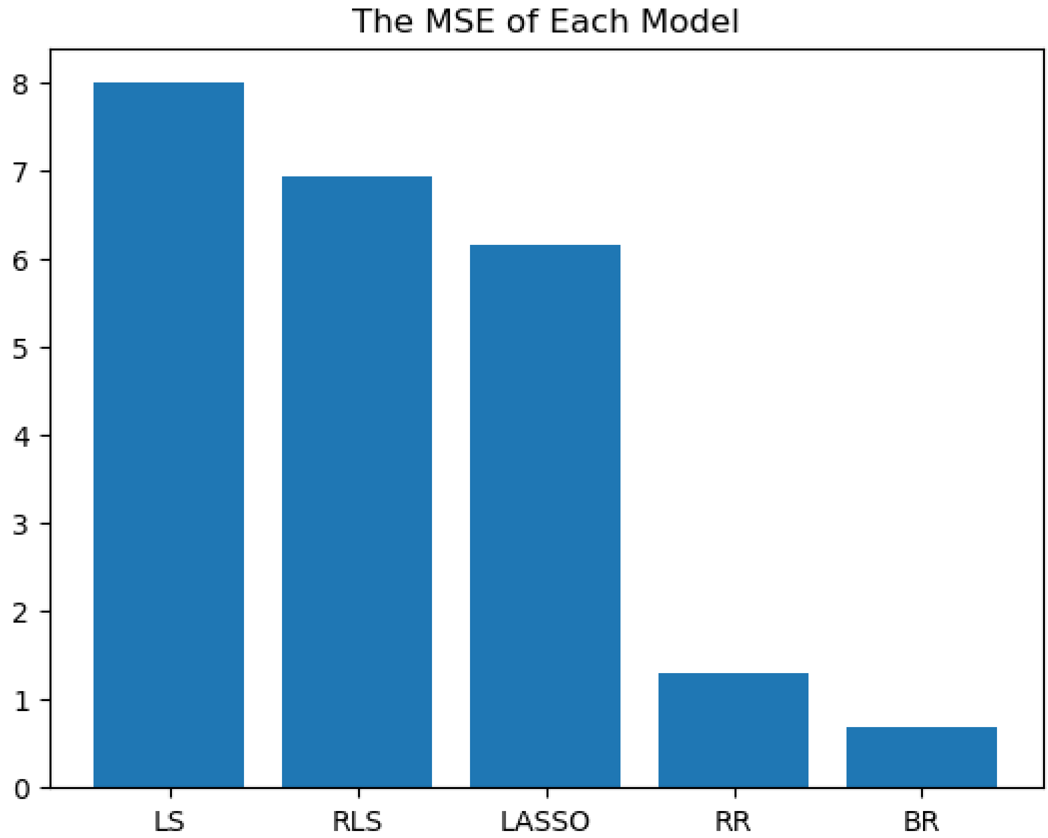




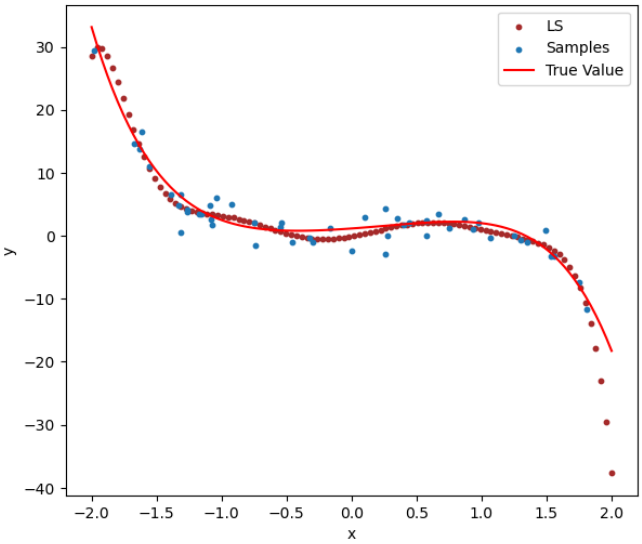
Problem e

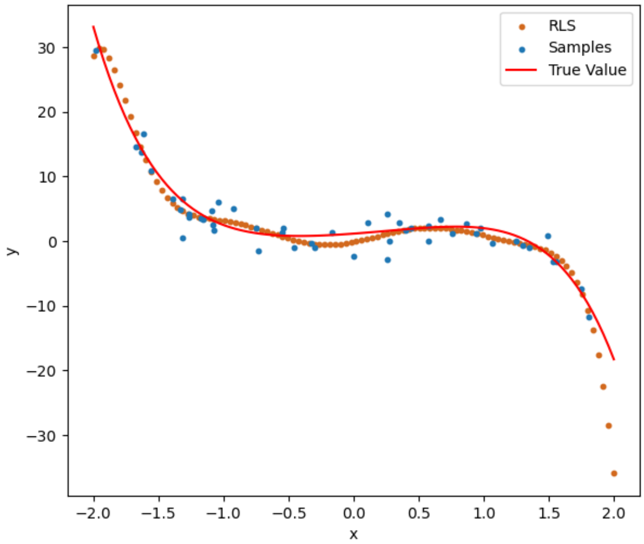
I repeat (b) but estimate a higher-order polynomial and set the K=10. According to the charts below, can get that the LS, RLS and RR tend to overfit the data when learning a more complex model and LASSO, BR 's estimate function were closed to the true values.

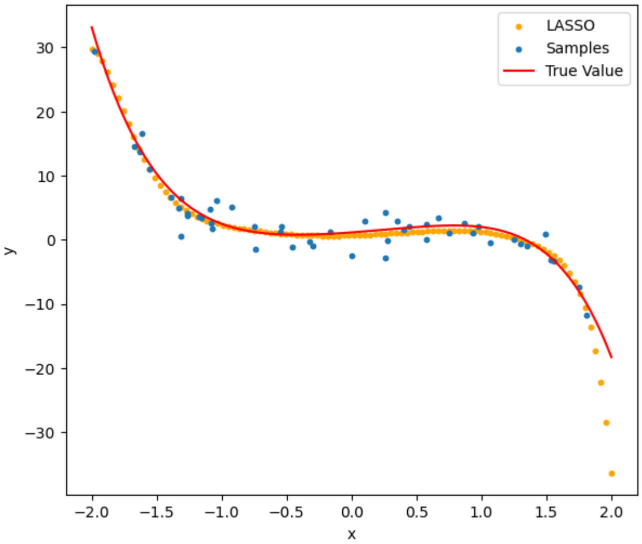
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

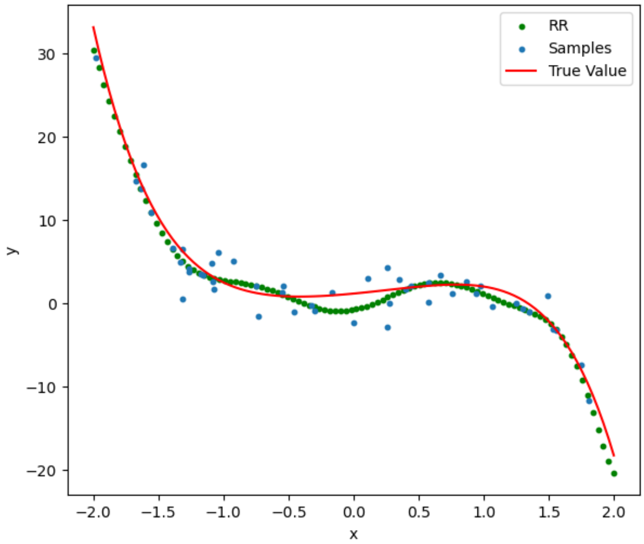


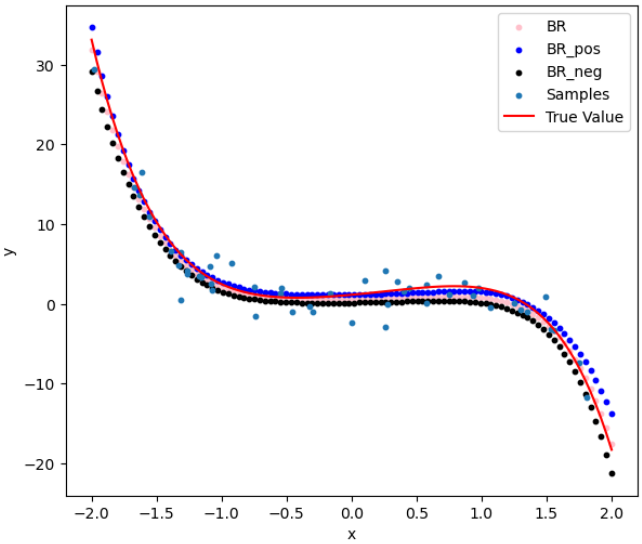

```
MSE of LS: 7.983106590193071
MSE of RLS: 6.929401524586747
MSE of LASSO: 6.1427147417718775
MSE of RR: 1.2898574921007475
MSE of BR: 0.6891047096909112
```









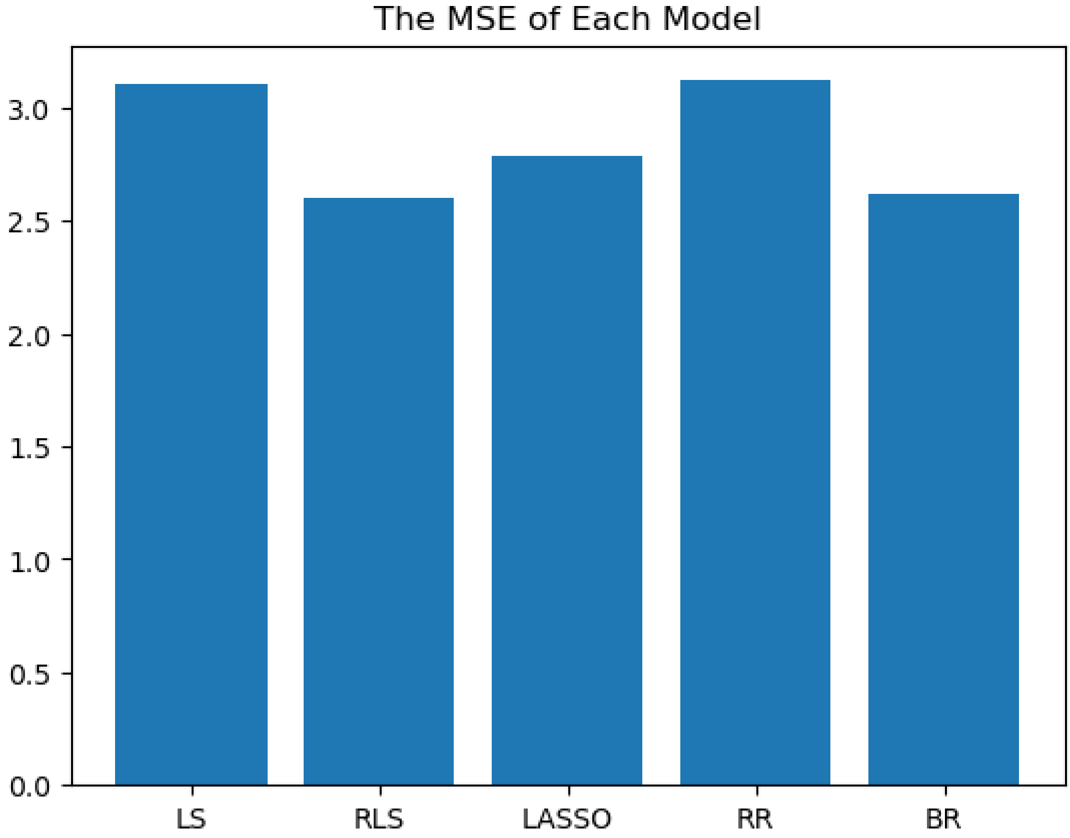


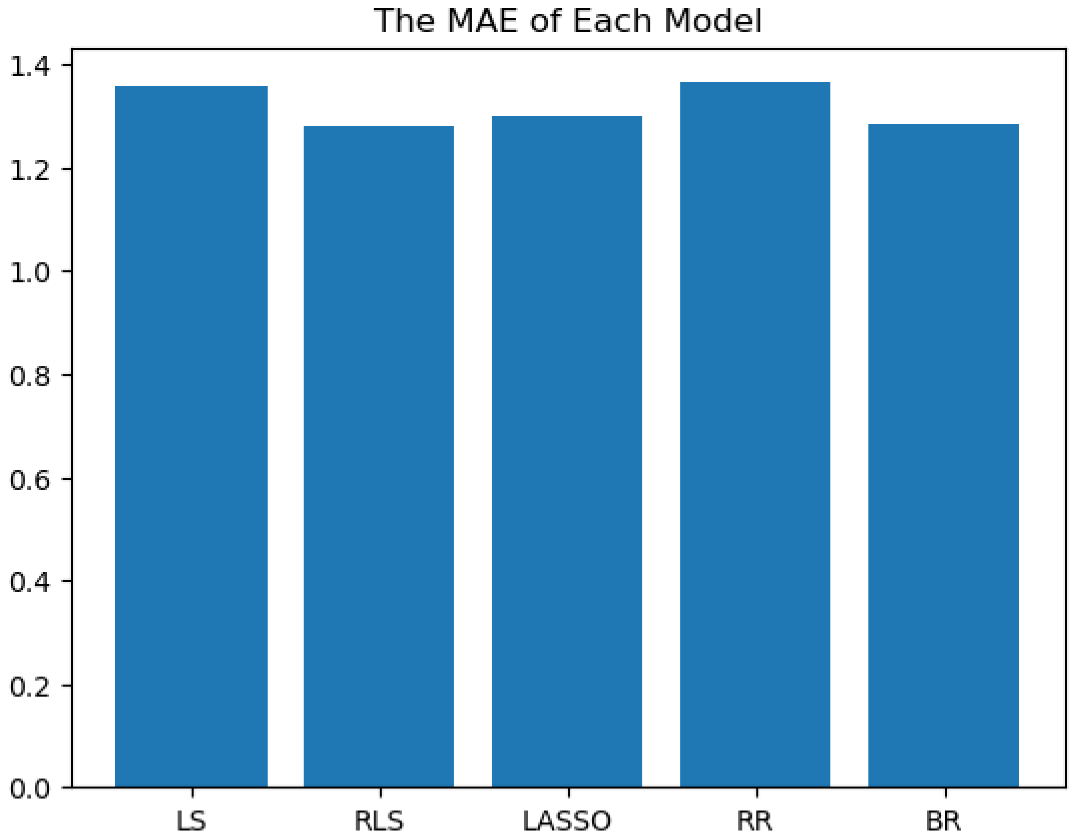
Part 2

Problem a

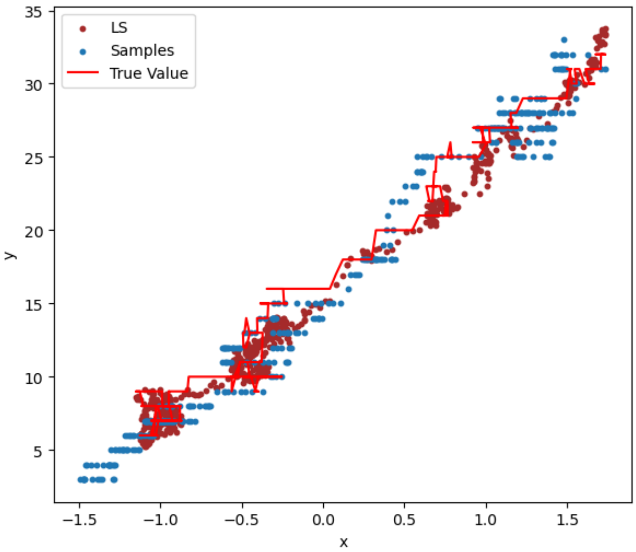
Based on the result of each model's MSE and MAE, can get the RLS works best Plot the test predictions and the true counts as follows To the discussion of any interesting findings: It can be seen from the below figures that all the predictive models behave well when the number of people is huge or small. However, when the number of people is in the middle, which is around 10 to 15 and 20 to 25, all models have predicted values is litte far from the real value.

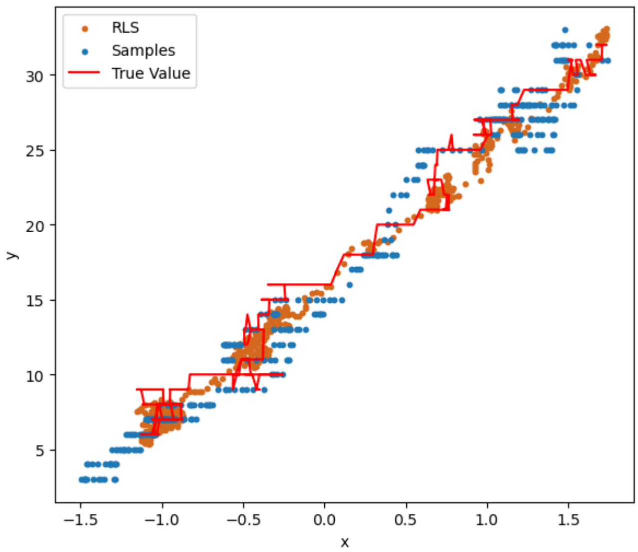
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

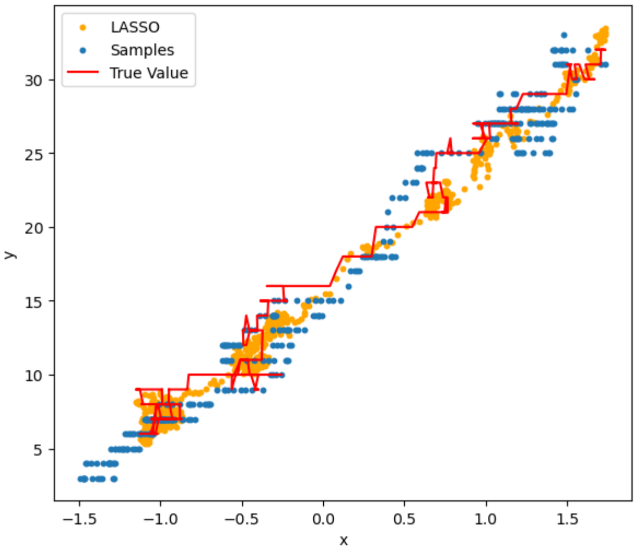


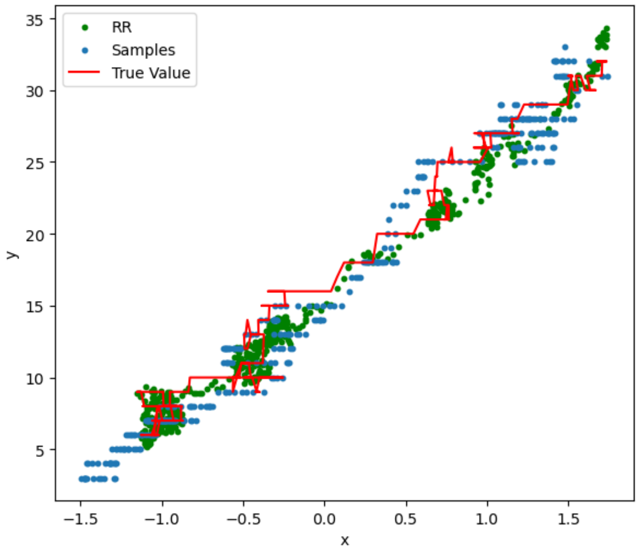


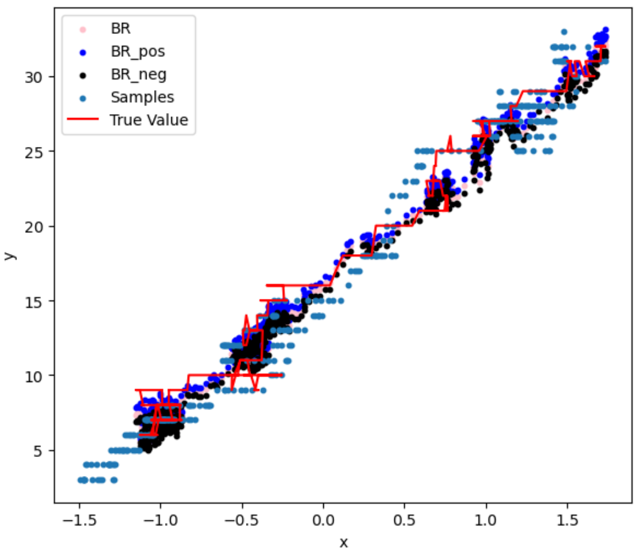
```
MSE of LS: 3.102838014134246
MSE of RLS: 2.5978793141470065
MSE of LASSO: 2.782724076612656
MSE of RR: 3.118997118962825
MSE of BR: 2.618733922241957
==========================
MAE of LS: 1.3584435211465367
MAE of RLS: 1.279905113299301
MAE of LASSO: 1.3008751101061713
MAE of RR: 1.364567082152389
MAE of BR: 1.2824328557329046
```











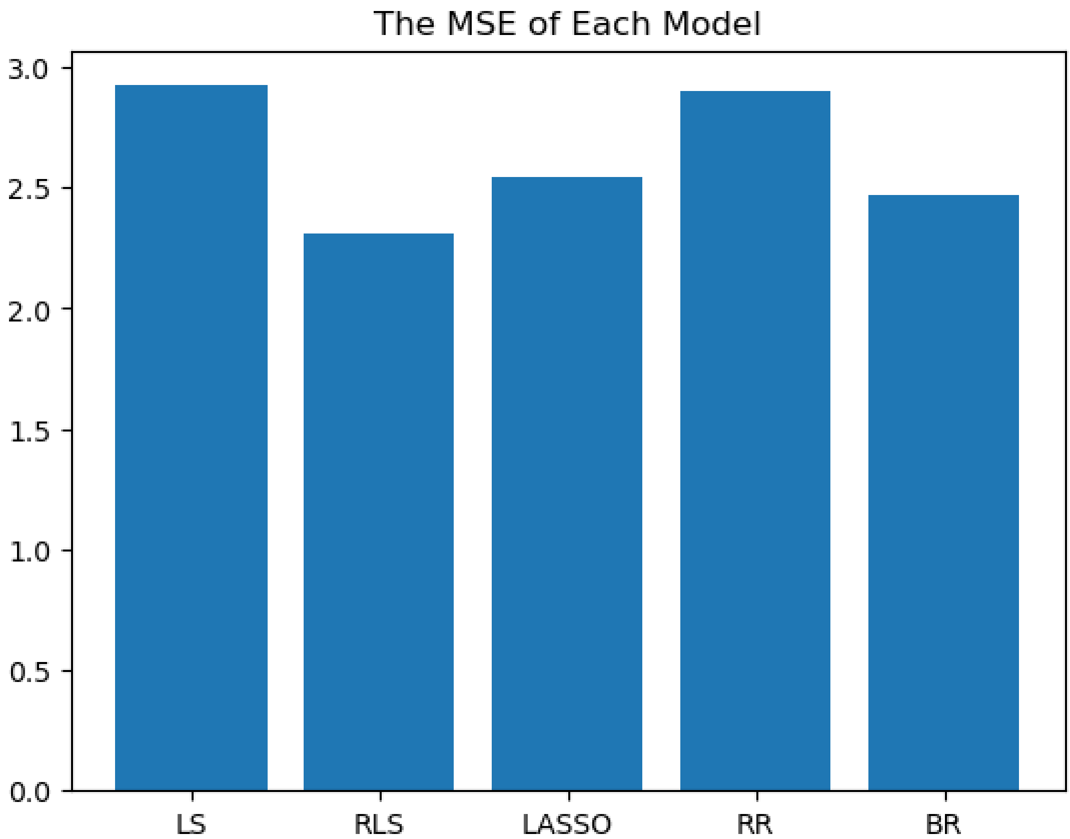
Problem b

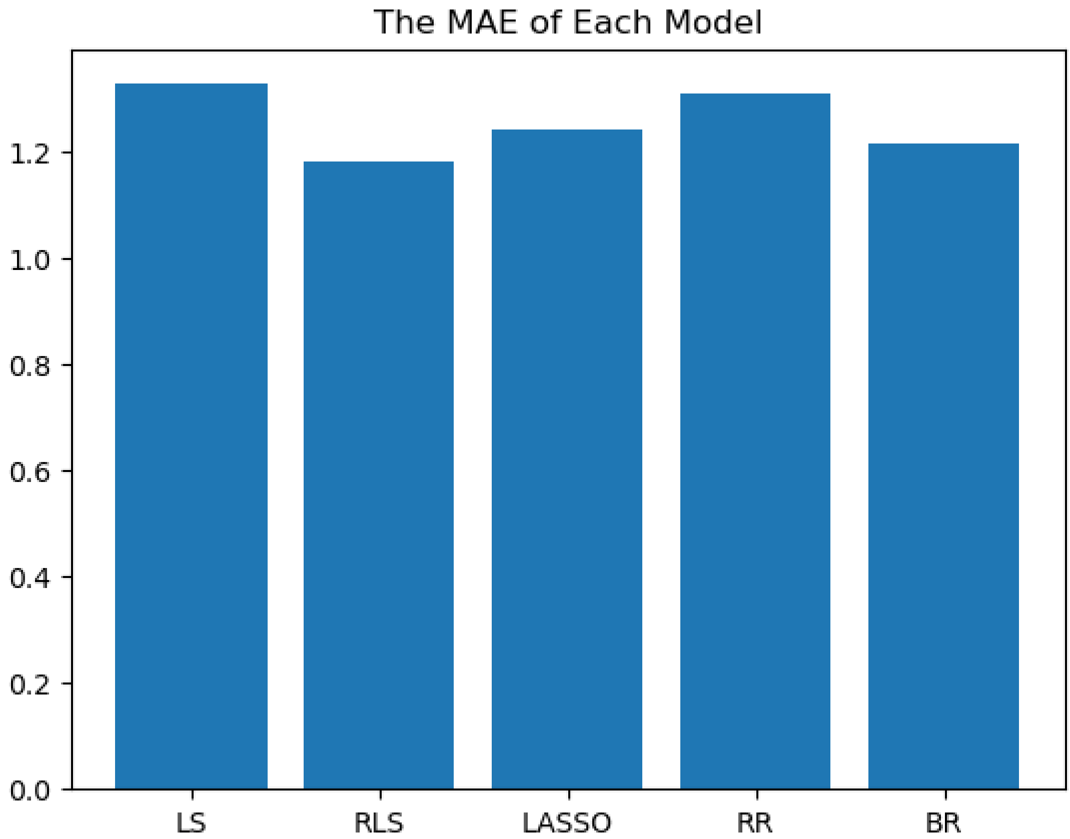
1. First I implement the $\phi(x)$ is $[x_1, \ldots, x_9, (x_1)^2, \ldots (x_9)^2]^T$ and get the following trail results. After improving the $\phi(x)$ into this mode, the MAE and MSE of each model performed better than problem a, which $\phi(x)$ is x. Those results performed better than problem a)

2. Then, I add the $x_i x_j$ into the $\phi(x)$, depend on use previous $\phi(x)$ in 1. and do the $\phi(x)\phi(x)^T$. Then I select the items of uppertriangular from this matrix, which are all the $x_i x_j$
3. I add transform the $\phi(x)$ with 3rd order like x^3, base on the 1.. and get the better performance than problem a)

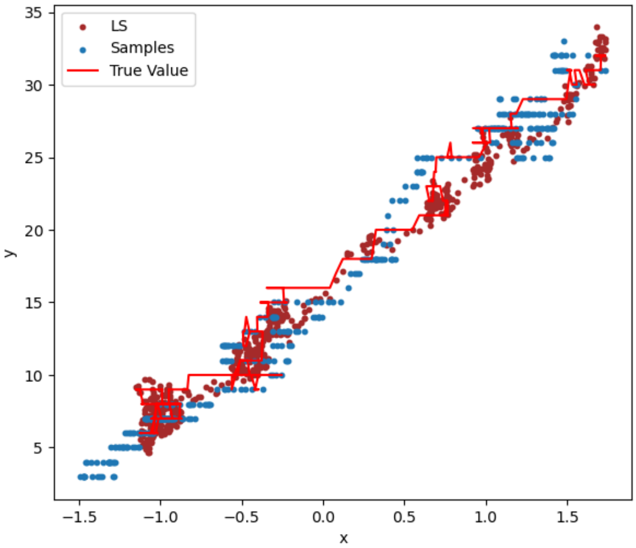
Problem b 1.

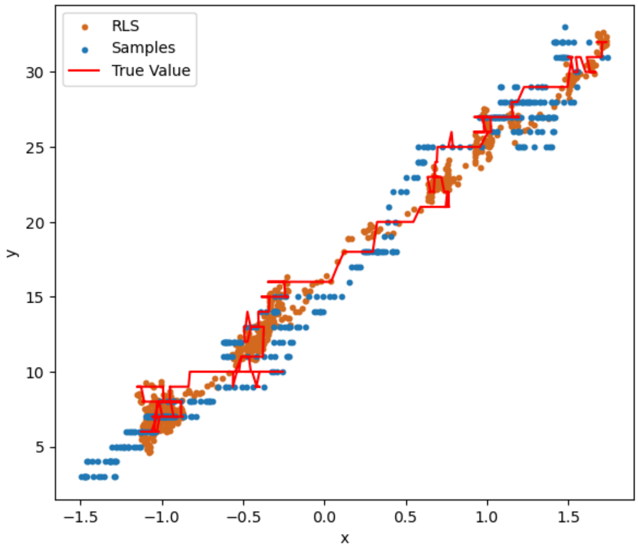
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

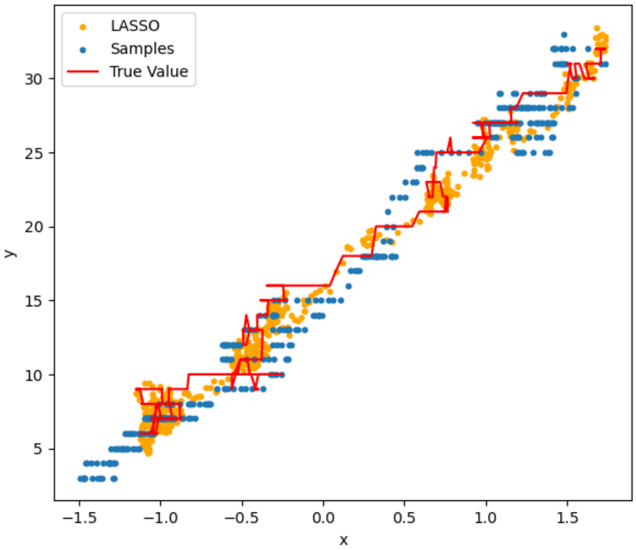


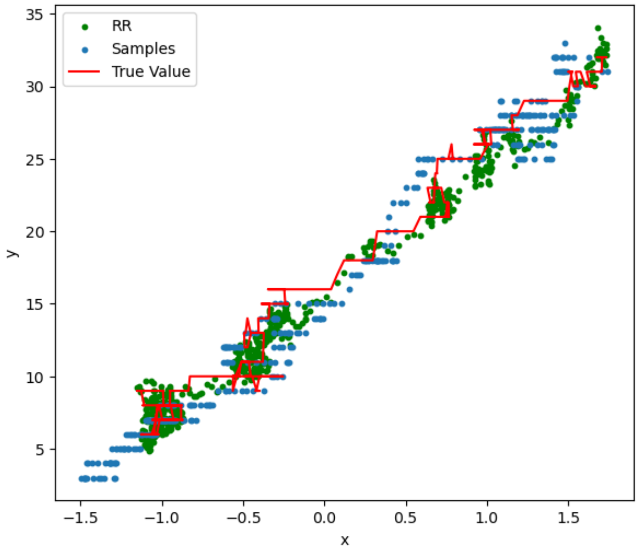


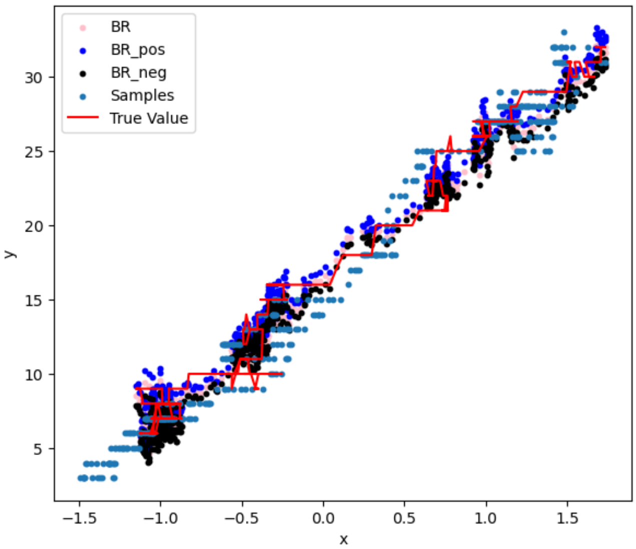

```
MSE of LS: 2.923594026734244
MSE of RLS: 2.314812148513016
MSE of LASSO: 2.5452142083552824
MSE of RR: 2.898128692341148
MSE of BR: 2.4677739025681045
==========================
MAE of LS: 1.3267461241114153
MAE of RLS: 1.1818620393828905
MAE of LASSO: 1.2425531540179677
MAE of RR: 1.3079287591847546
MAE of BR: 1.2136987570470137
```





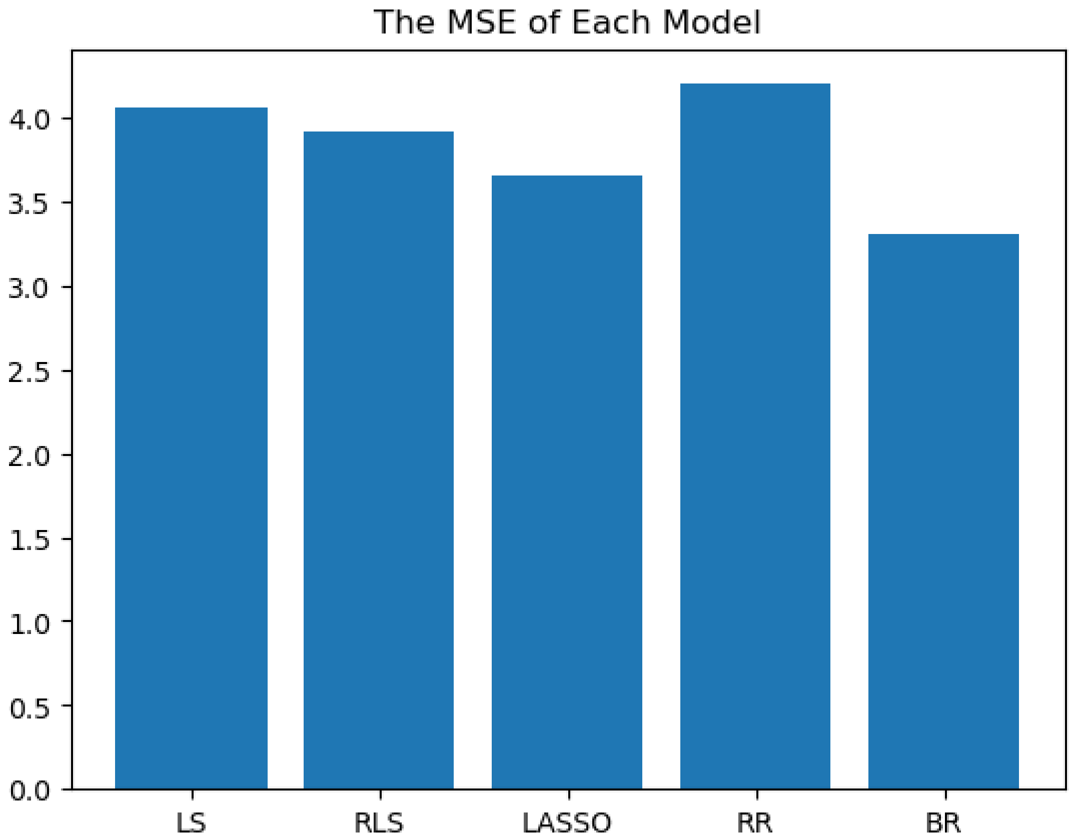


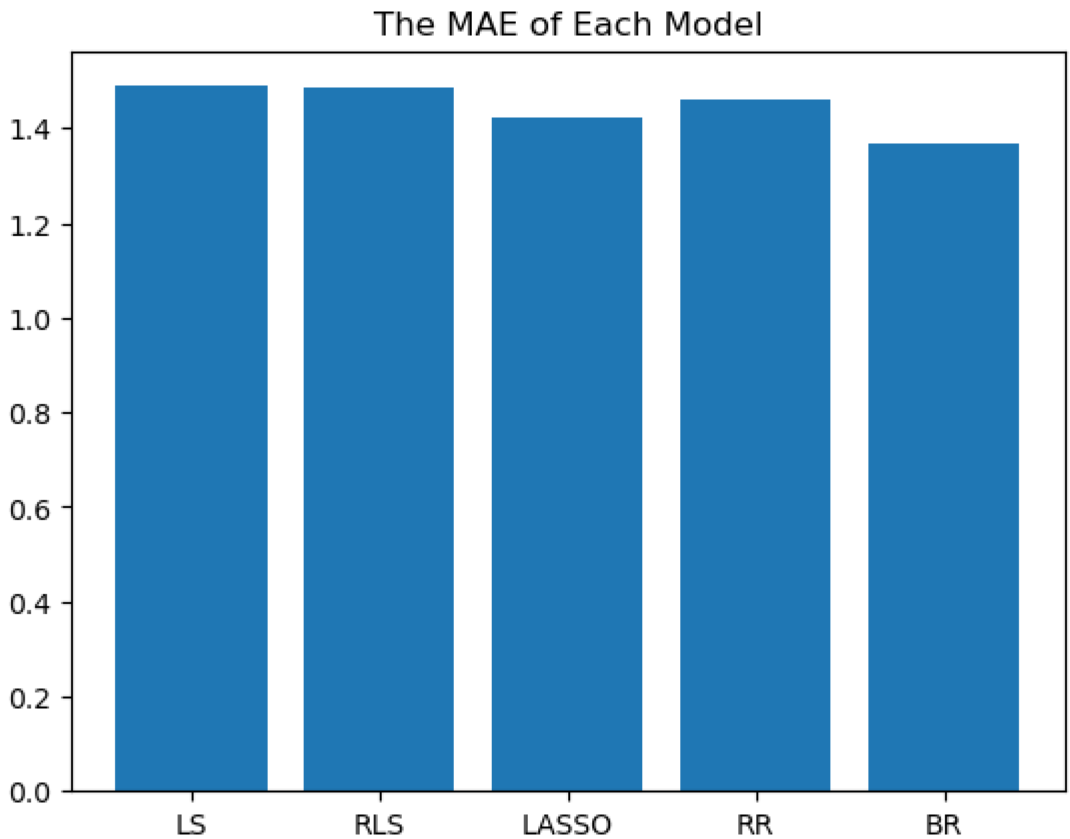




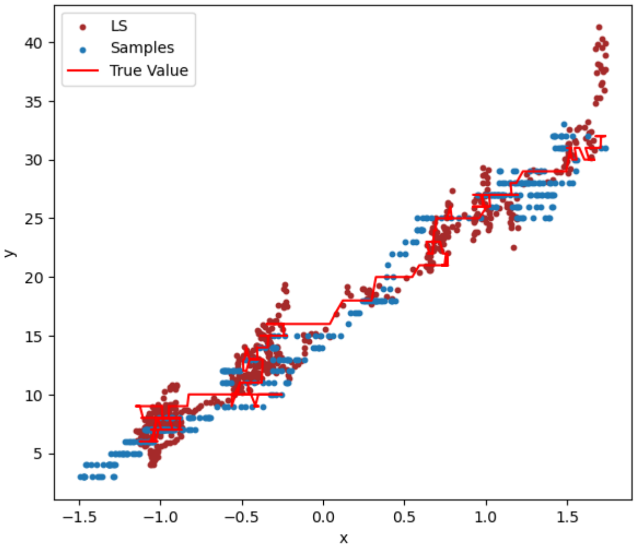
Problem b 2.

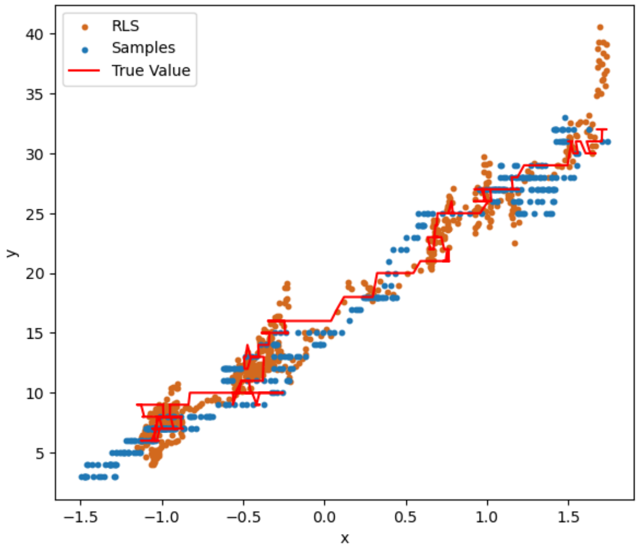
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

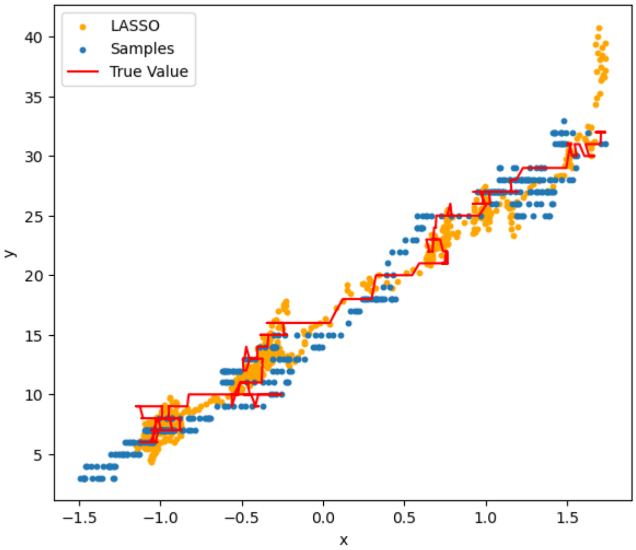

The MSE of Each Model
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
LS
RLS
LASSO
RR
BR

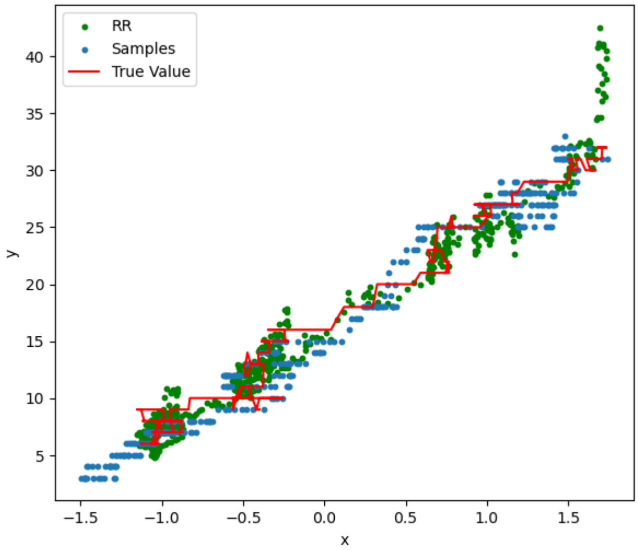

The MAE of Each Model
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
LS
RLS
LASSO
RR
BR

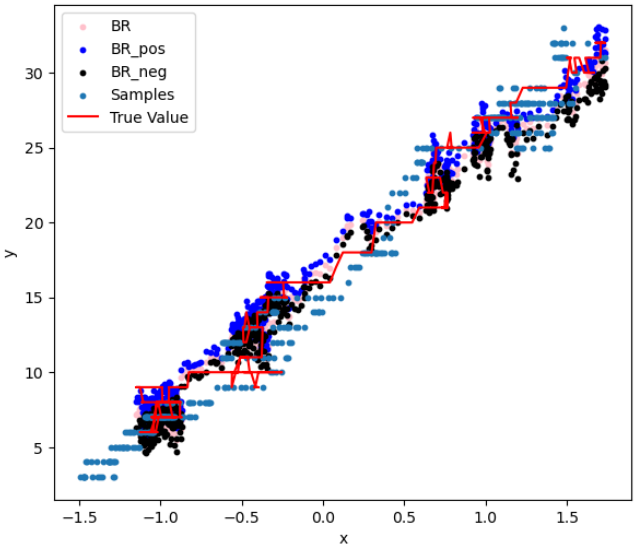
```
MSE of LS: 4.059714635807797
MSE of RLS: 3.9217133308754644
MSE of LASSO: 3.6506680264170193
MSE of RR: 4.199906460834481
MSE of BR: 3.304269465872169
==========================
MAE of LS: 1.4886914191150775
MAE of RLS: 1.4839026857814916
MAE of LASSO: 1.4240020795213721
MAE of RR: 1.4601520059695001
MAE of BR: 1.3656207673876684
```





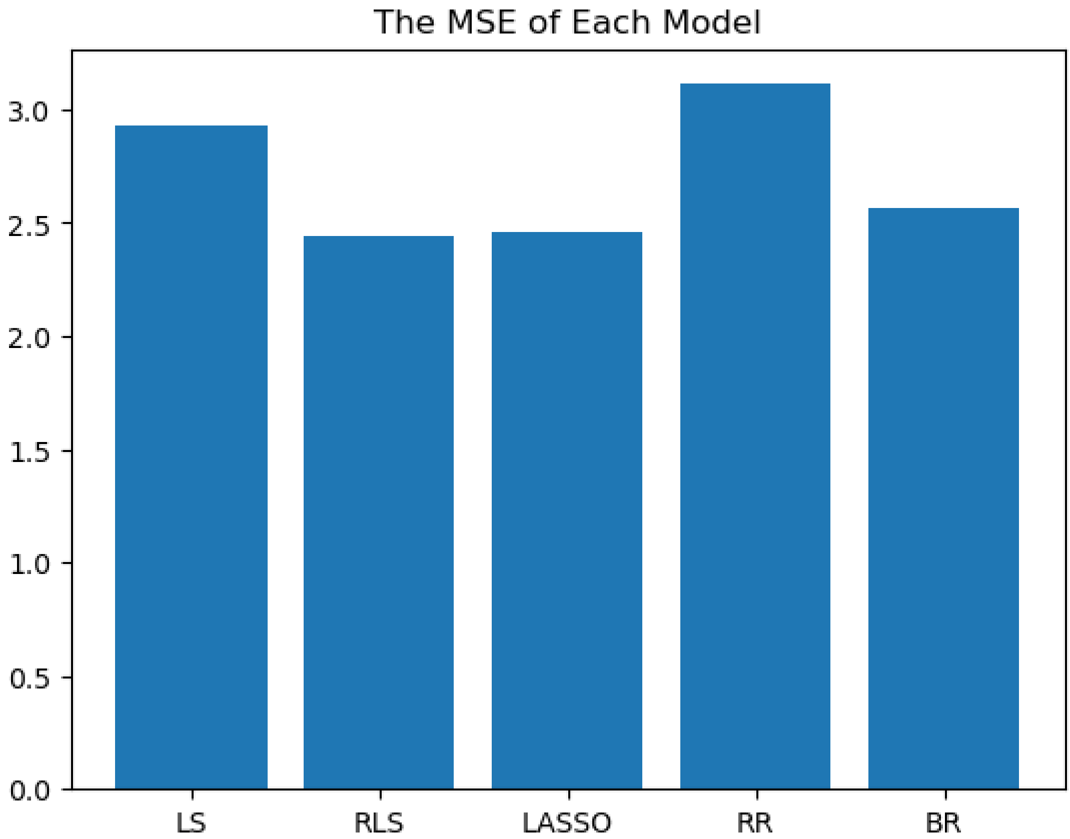


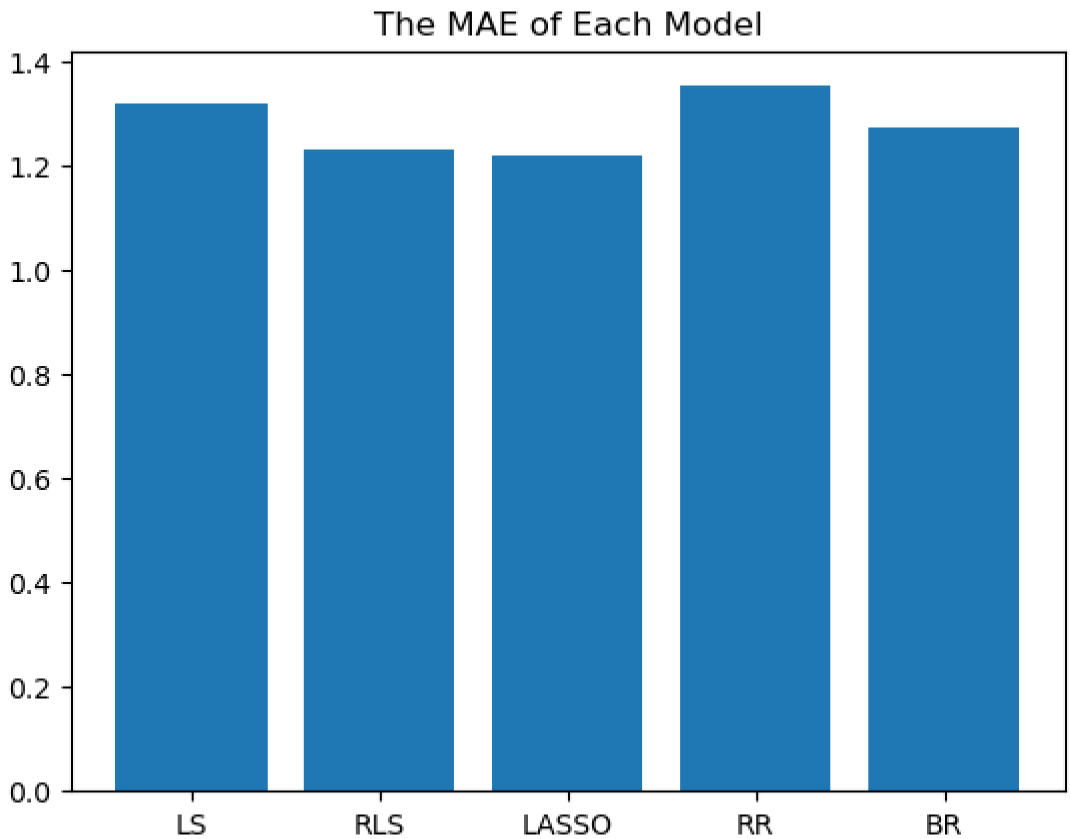




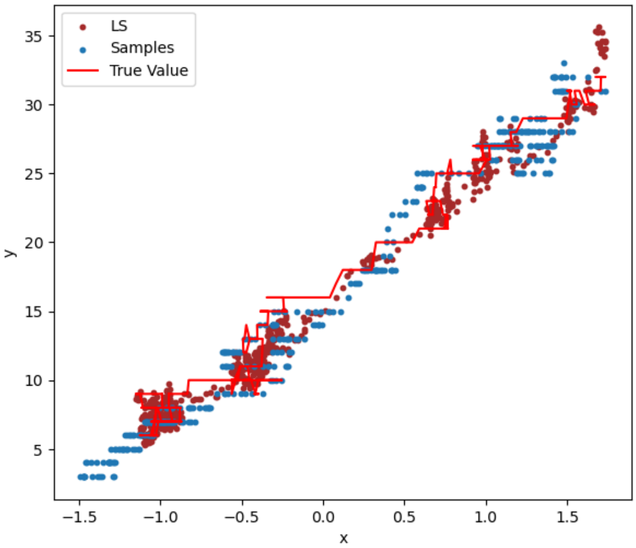
Problem b 3.

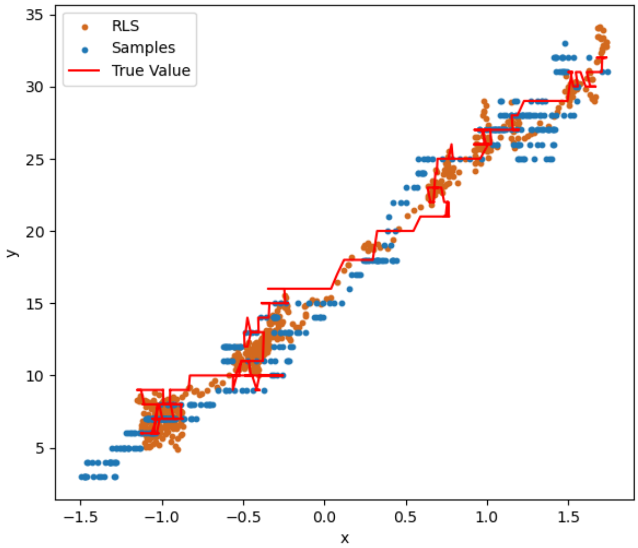
```
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: divide by zero encountered in double_scalars
c:\Users\huancwang2\.conda\envs\regular\lib\site-packages\ipykernel_launcher.py:48: RuntimeWa
rning: invalid value encountered in multiply
```

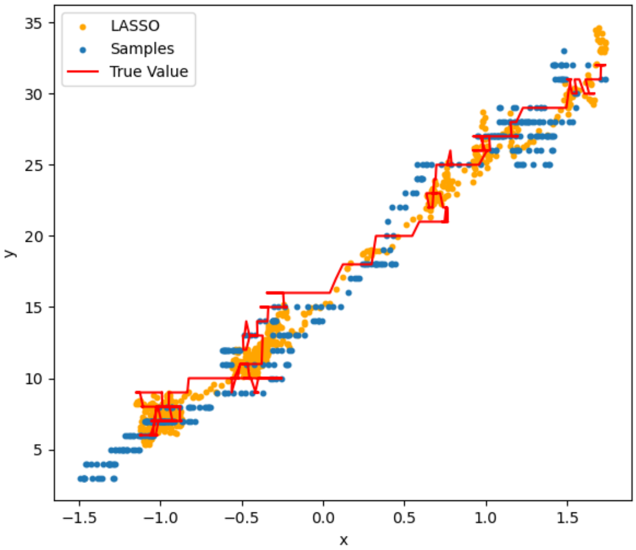

The MSE of Each Model
3.0
2.5
2.0
1.5
1.0
0.5
0.0
LS
RLS
LASSO
RR
BR

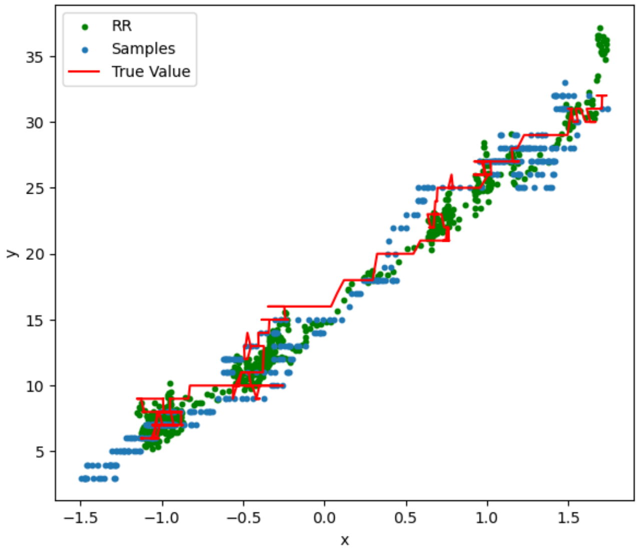

The MAE of Each Model
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
LS
RLS
LASSO
RR
BR

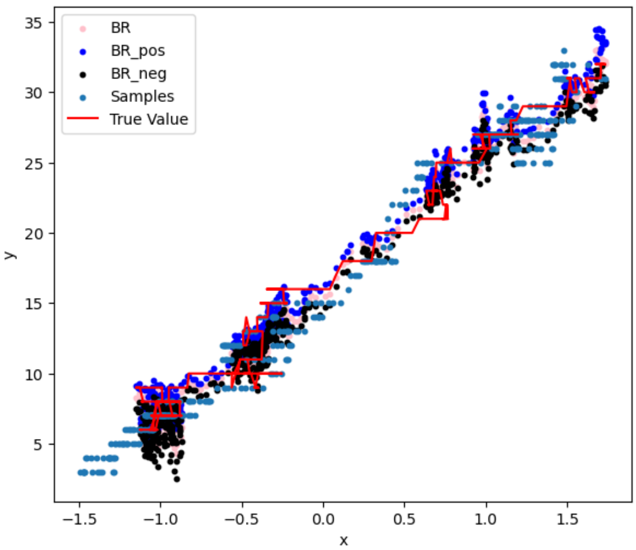
```
MSE of LS: 2.928288067211331
MSE of RLS: 2.4393685900701576
MSE of LASSO: 2.4597543046897057
MSE of RR: 3.111646272716377
MSE of BR: 2.5622208274667106
==========================
MAE of LS: 1.3191990563261018
MAE of RLS: 1.2316043360224
MAE of LASSO: 1.2185566820616565
MAE of RR: 1.3517843231497306
MAE of BR: 1.2712200828207196
```











Reference of library in my code

1. Numpy for process and load the data for regression, and do the matrix calculation. Numpy
2. CVXOPT for do the optimizing to solve the QP and LP problrm. CVXOPT
3. matplot.pyplot for plot each chart in this assignment. matplot.pyplot