Computing Solutions:

Linear Regression Solutions:

1)

The method for building the model is in ***navieRegression.py***

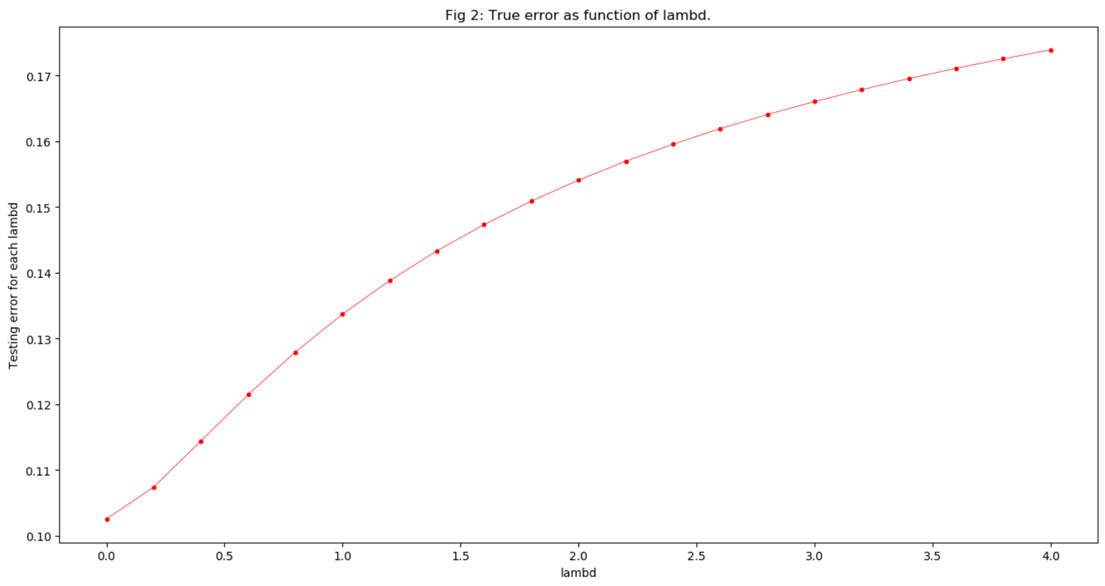
	My estimate	True value
b	10.1564499	10
w1	0.363606539	0.6
w2	0.310887166	0.36
w3	0.143799563	0.216
w4	0.130382430	0.1296
w5	0.0838776763	0.07776
w6	0.0242482722	0.046656
w7	-0.00603130016	0.0279936
w8	0.0260364626	0.01679616
w9	0.0107085265	0.010077696
w10	-0.0324024139	0.0060466176
w11	-0.00440056885	0
w12	-0.00715925599	0
w13	-0.0414272769	0
w14	0.0539065284	0
w15	-0.0450492447	0
w16	0.00293632300	0
w17	0.00124798852	0
w18	-0.00732592891	0
w19	0.00255733950	0
w20	-0.00377808108	0

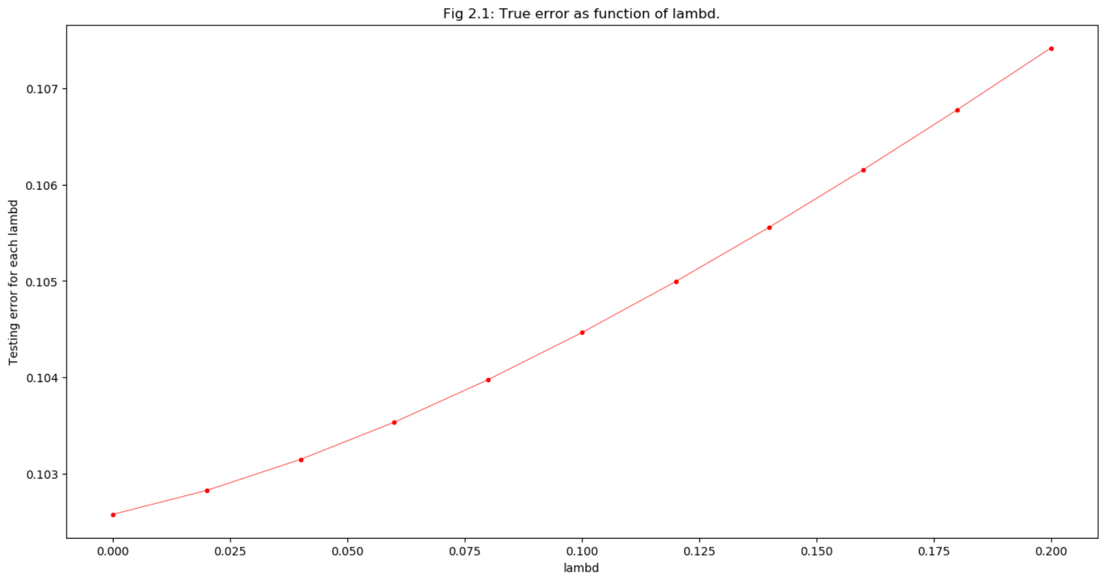
train error: 0.09454763320553954
test error: 0.10257722520979141 (under 1×10^6 test data)

We can see from the above form that the most significant feature is X1 and the least is X17. And it is not able to prune anything because all Xi will have impact on Y as we can see.

2)

The code is in ***ridgeRegression.py*** and ***question2.py***. I use the same data as in the question 1. And below is the results.





So we can see that when $\boldsymbol{\lambda = 0.0}$ the estimated true error is optimal. Because the ridge is focusing on penalizing models with large norms or large coefficients in the weight vector, and we have a 10 as the intercept, so if we want to punish the model, it will definitely loss accuracy.

	My estimate at $\boldsymbol{\lambda = 0.0}$	True value
b	10.1564499255683	10
w1	0.363606539194954	0.6
w2	0.310887165877882	0.36
w3	0.143799562673655	0.216

w4	0.130382429777793	0.1296
w5	0.083877676349746	0.07776
w6	0.0242482721648634	0.046656
w7	-0.00603130016490349	0.0279936
w8	0.0260364625916005	0.01679616
w9	0.0107085264603378	0.010077696
w10	-0.0324024138751003	0.0060466176
w11	-0.004400568854863	0
w12	-0.0071592559916459	0
w13	-0.0414272769070035	0
w14	0.0539065283826011	0
w15	-0.0450492447328048	0
w16	0.002936323002557	0
w17	0.00124798851510468	0
w18	-0.00732592890790617	0
w19	0.00255733949762045	0
w20	-0.00377808107975324	0

train error: 0.09454763320553954
test error: 0.10257722520979141 (under 1×10^6 test data)

We can see from the above form that the most significant feature is X1 and the least is X17. And it is not able to prune anything because all Xi will have impact on Y as we can see. Because when $\boldsymbol{\lambda = 0.0}$ we have the same situation as the question 1, it is the same as naïve regression model.

3)

The code is in ***lassoRegression.py*** and ***question3.py***. I use the same data as in the question 1. And below is the results. And the weight data is in ***result3.*.csv***

Fig 3: Number of features been eliminated as function of lambd.

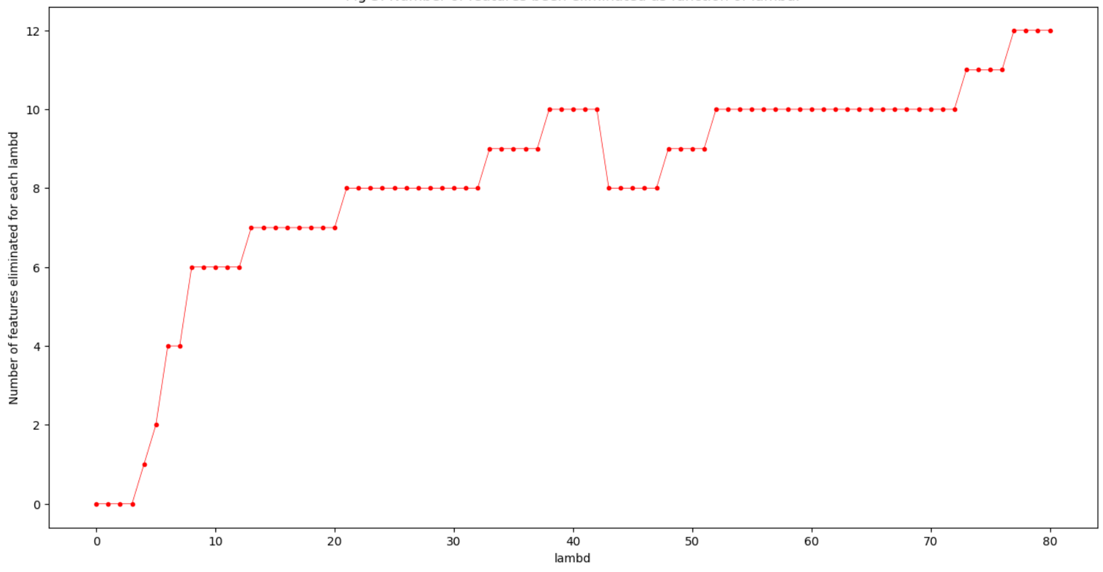


Fig 3: Number of features been eliminated as function of lambd.

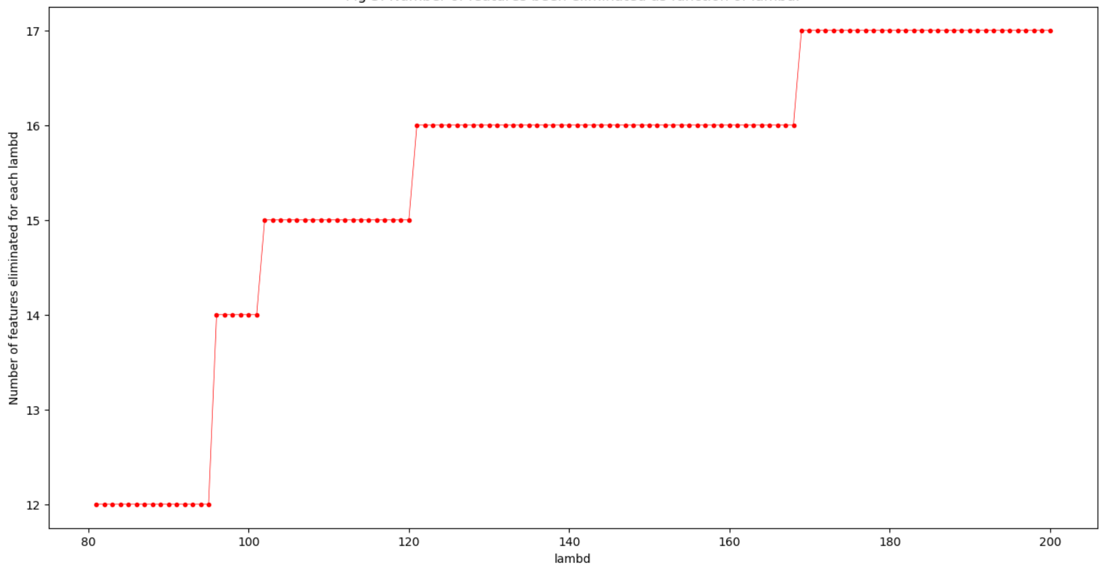
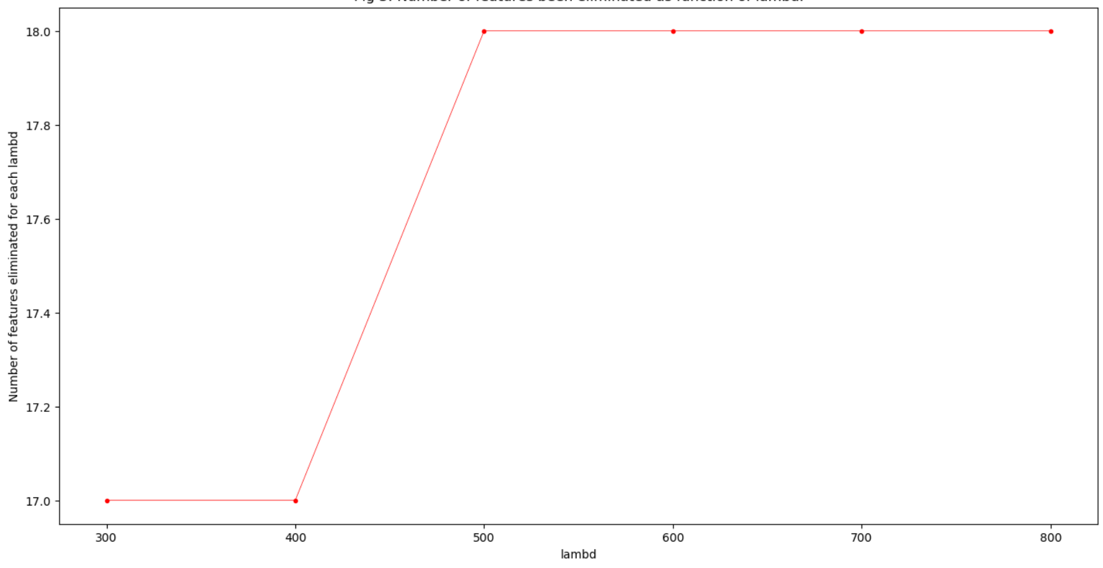


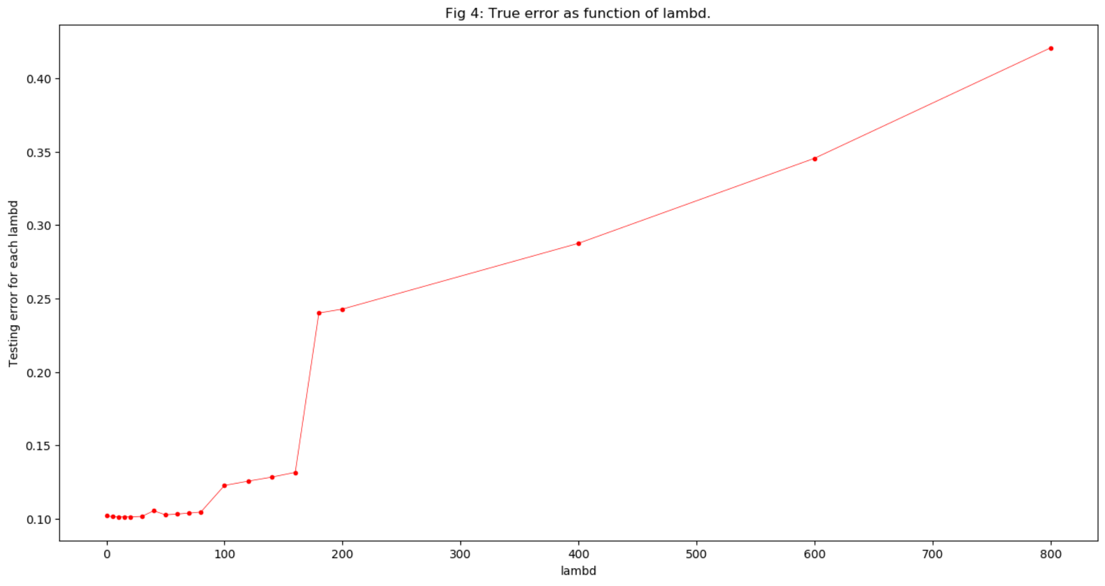
Fig 3: Number of features been eliminated as function of lambd.

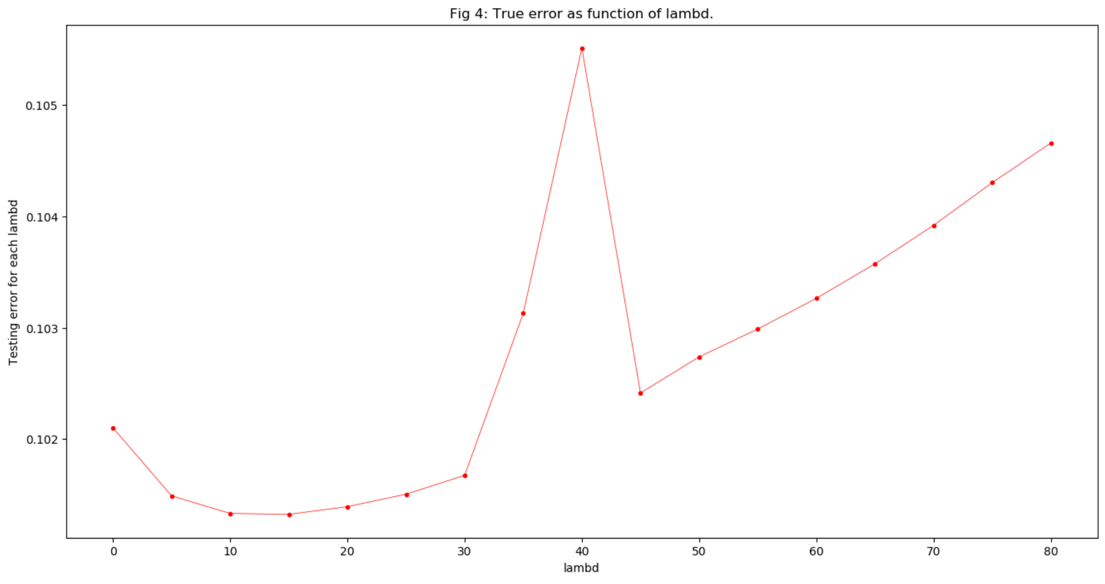


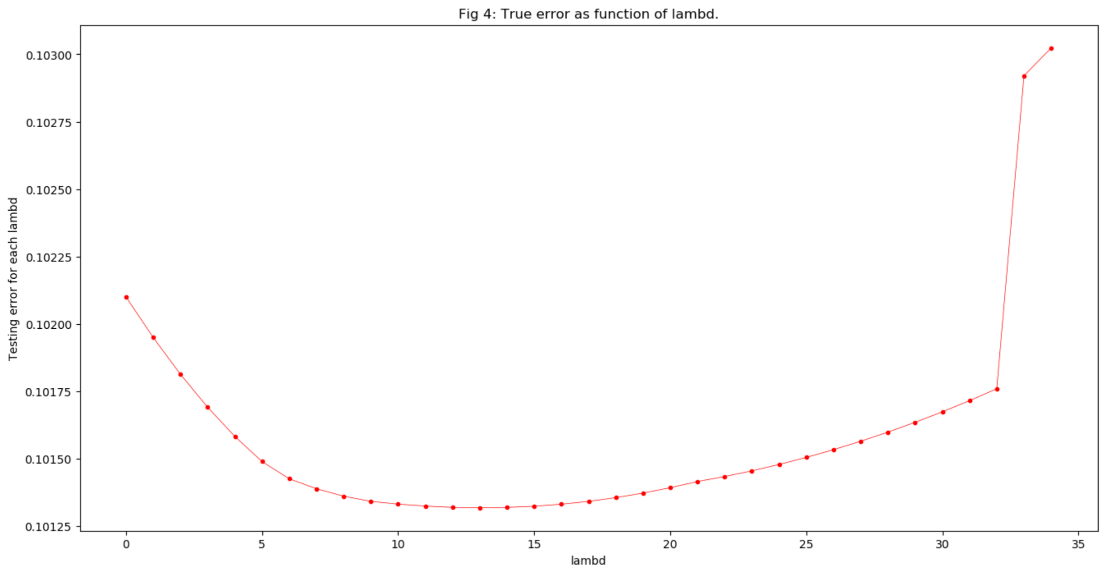
So we can see that when **λ** is increasing the number of features been eliminated is increasing.

4)

The code is in ***question4.py***. I use the same data as in the question 1. And below is the results.







So we can see that when $\lambda = 13$ the estimated true error is optimal, it is 0.101318163550322.

	My estimate at $\lambda = 13$	True value
b	10.5860049007994	10
w1	0.341023594531225	0.6
w2	0.20056284166834	0.36
w3	0.119806223587807	0.216
w4	0.0689195138320536	0.1296
w5	0.0374595406863312	0.07776
w6	0.0177189919610972	0.046656
w7	0	0.0279936
w8	0.0188373388911338	0.01679616
w9	0.00431004160233874	0.010077696
w10	-0.0246132559401568	0.0060466176
w11	0.0127537093143632	0
w12	0.0108652147669304	0
w13	0.00059541067441157	0
w14	0	0
w15	-0.0018951267354529	0
w16	0	0
w17	0	0
w18	0	0
w19	0	0
w20	0	0

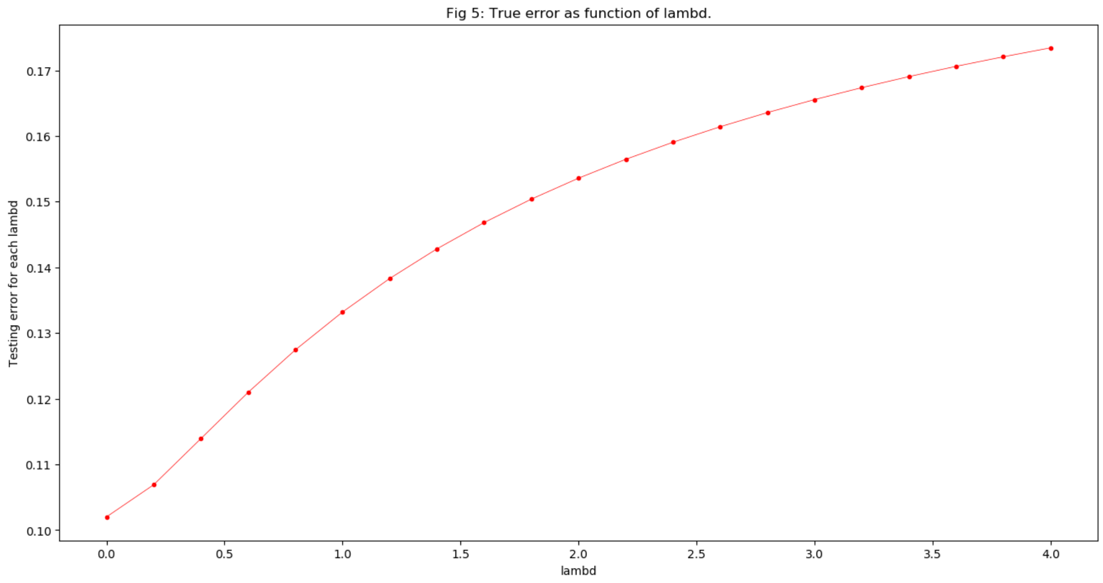
train error: 0.0170658605075715

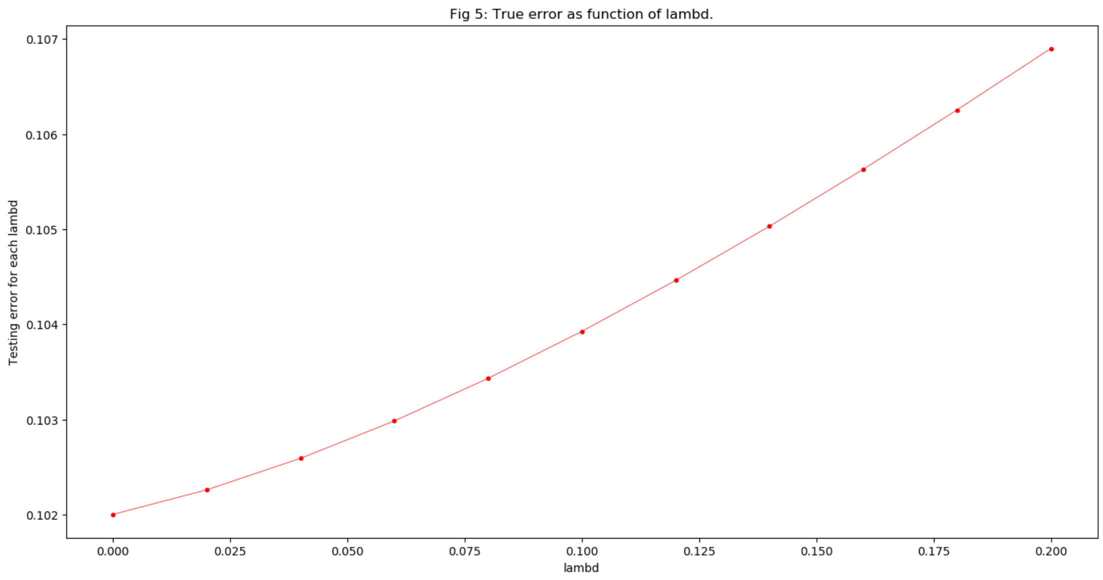
test error: 0. 101318163550322 (under 1×10^6 test data)

We can see from the above form that the most significant feature is X1 and the least is X13. And it is not able to prune anything because all X11 to X13 and X15 will have impact on Y as we can see. But it is still better than the naïve one.

5)

The code is in ***question5.py***. I use the same data as in the question 1. And below is the results.





So we can see that when $\boldsymbol{\lambda = 0.0}$ the estimated true error is optimal. Because the ridge is focusing on penalizing models with large norms or large coefficients in the weight vector, and we have a 10 as the intercept, so if we want to punish the model, it will definitely lose accuracy.

	My estimate at $\boldsymbol{\lambda = 0.0}$	Naïve model	True value
b	10.148369948708	10.1564499	10
w1	0.363777570948682	0.363606539	0.6
w2	0.312022342534499	0.310887166	0.36
w3	0.139816194847501	0.143799563	0.216
w4	0.123519128784532	0.130382430	0.1296
w5	0.0806695872168671	0.0838776763	0.07776
w6	0.0244416840312745	0.0242482722	0.046656
w7	0	-0.00603130016	0.0279936
w8	0.0261276404742855	0.0260364626	0.01679616
w9	0.0114056600848439	0.0107085265	0.010077696
w10	-0.0303327956506536	-0.0324024139	0.0060466176
w11	-0.00514762118700504	-0.00440056885	0
w12	-0.00317474334394735	-0.00715925599	0
w13	-0.0369996274438786	-0.0414272769	0
w14	0	0.0539065284	0
w15	-0.045801806194572	-0.0450492447	0
w16	0	0.00293632300	0
w17	0	0.00124798852	0
w18	0	-0.00732592891	0
w19	0	0.00255733950	0
w20	0	-0.00377808108	0
Train error	0.0949274282140526	0.09454763320553954	
Test error	0.102004006129843	0.10257722520979141	

train error: 0.0949274282140526
test error: 0.102004006129843 (under 1×10^6 test data)

Comparing to the naïve one, the train error is worse but the test error is better, so we can think this model is better than the naïve least squares model.
We can see from the above form that the most significant feature is X1 and the least is X12. And X11 and X12 are relatively insignificant as we can see.

SVMs Solutions:

1)

The solver is in the file ***svmSolver.py***
In update step, I use a max threshold error to control that we always inside the constraint region.

2)

We have got $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.125$ in the former homework.
And the result of our solver is: [0.125139850106026, 0.125094843551614, 0.125116862432028, 0.125071855877616]
They are approximately 0.125
So, my work is right.

3)