Problem 1

Part a)

CSC 311) Assignment 2)

Problem 1) a) we are looking to minimize $\frac{1}{n}\sum |Y_i - m|^2 = f(m)$

To do so, first we find critical points by $f'(m) = 0$

$f'(m) = \frac{1}{n}\sum -2(Y_i - m) = \frac{-2}{n}\sum(Y_i - m)$ to have this as zero

we put $m = \frac{1}{n}\sum Y_i$

$\frac{-2}{n}\left[\sum y_i - n\frac{1}{n}\sum y_i\right] = 0$, let us now look at the second derivative

$f''(m) = \frac{-2}{n} \times -1 = \frac{2}{n}$

Since $f'(m) = 0$ and $f''(m) > 0$ we can conclude that

$\frac{1}{n}\sum Y_i$ does minimize $\frac{1}{n}\sum |Y_i - m|^2$ □

Part b)

Problem 1) part b)

$$h(D) = \frac{1}{n}\sum y_i$$

Bias: $|E_D[h(D)] - \mu|^2 = |\mu - \mu|^2 = 0$

$E_D[h(D)] = \mu$ by weak law of large numbers

Variance: $E_D\left[|h(D) - E_D[h(D)]|^2\right]$

$$= E_D\left[\left|\frac{1}{n}\sum y_i - \cancelto{\mu}{E\left[\frac{1}{n}\sum y_i\right]}\right|^2\right]$$

$$= E\left[\left|\frac{1}{n}\sum (y_i - \mu)\right|^2\right] = \frac{1}{n^2}\sum \cancelto{\sigma^2}{E(y_i - m)^2} = \frac{1}{n^2} n\sigma^2 = \frac{\sigma^2}{n}$$

Part c)

Problem 1) Part c)

we need to have $f'(m)=0$ where $f(m) = \frac{1}{n}\sum |Y_i - m|^2 + \lambda m^2$

$f''(m) \geq 0$

$f(m) = \frac{1}{n}\sum y_i^2 + \frac{1}{n}\sum m^2 - \frac{2}{n}\sum y_i m + \lambda m^2$ $\qquad \frac{1}{n}\sum y_i m = m\frac{1}{n}\sum y_i = m\hat{\mu}$

$f'(m) = 2m - 2\hat{\mu} + 2\lambda m$

$f'(m)=0 \Rightarrow 2m - 2\hat{\mu} + 2\lambda m = 0 \Rightarrow m - \hat{\mu} + \lambda m = 0$

$\Rightarrow m + \lambda m = \hat{\mu} \rightarrow \cancel{m = \frac{\hat{\mu}}{\lambda+1}} \quad m = \frac{\hat{\mu}}{\lambda+1}$

$f''(m) = (2m - 2\hat{\mu} + 2\lambda m)\frac{d}{dm} = 2 + 2\lambda$ since $\lambda \geq 0 \Rightarrow 2 + 2\lambda > 0$

$\Rightarrow m = \frac{\hat{\mu}}{\lambda+1}$ minimizes to cost function; this also confirms with the result from part a) where $\lambda = 0$; $\frac{\hat{\mu}}{\lambda+1} = \frac{\hat{\mu}}{0+1} = \hat{\mu}$

Part d)

~~Problem 1)~~ Part d):

$$h_\lambda(D) = \frac{\hat{\mu}}{\lambda+1} = \frac{1}{n(\lambda+1)} \Sigma y_i$$

for Bias: $|E(h(D)) - \mu|^2$

$$E(h(D)) = E\left(\frac{\hat{\mu}}{\lambda+1}\right) = \frac{1}{\lambda+1} E\hat{\mu} = \frac{\mu}{\lambda+1} \Rightarrow \text{Bias}:$$

$\left(\frac{\mu}{\lambda+1} - \mu\right)^2$; which again for $\lambda = 0$, $\left(\frac{\mu}{\lambda+1} - \mu\right)^2 = 0$

for Variance: $E\left[|h(D) - E[h(D)]|^2\right]$

$$= E\left[\left|\frac{1}{n(\lambda+1)} \Sigma y_i - \frac{\mu}{\lambda+1}\right|^2\right] = \frac{1}{(\lambda+1)^2} E\left[\left|\frac{1}{n} \Sigma y_i - \mu\right|^2\right]$$

$$= \frac{1}{(\lambda+1)^2} E\left[\left|\frac{1}{n} \Sigma (y_i - \mu)\right|^2\right]$$

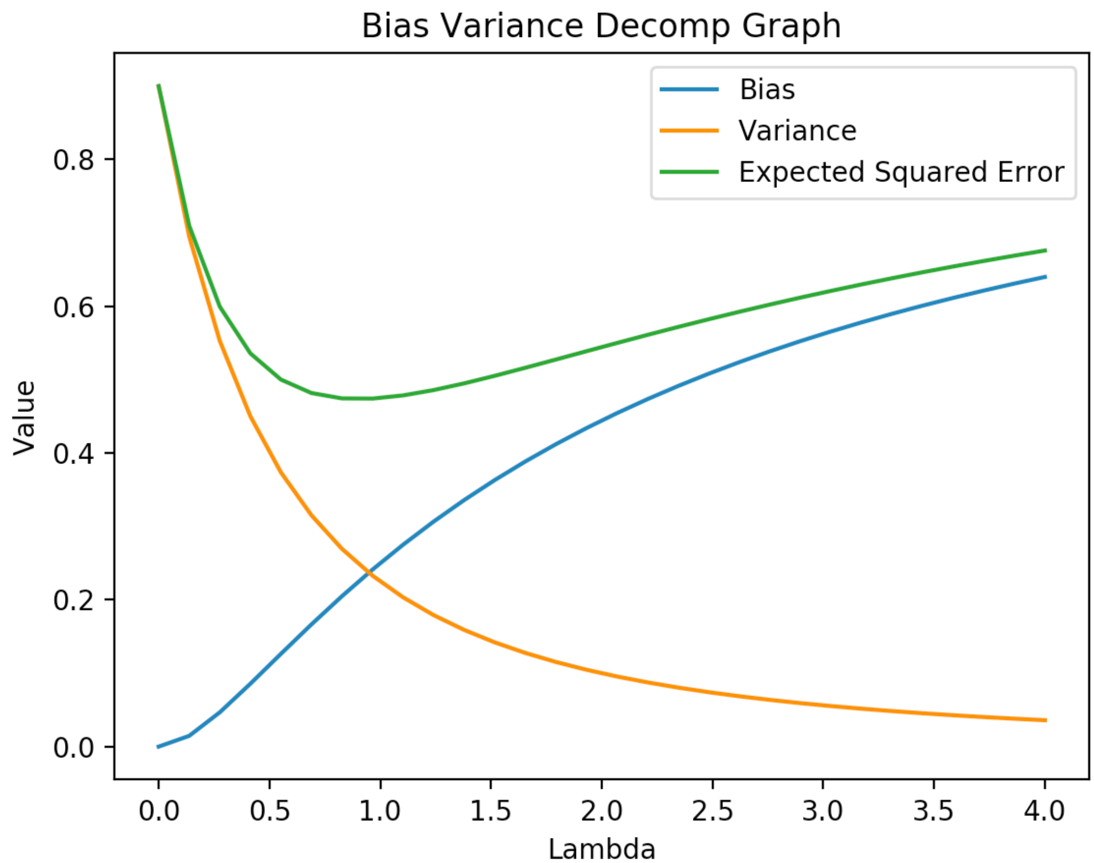
$$= \frac{1}{(\lambda+1)^2 n^2} \Sigma E(y_i - \mu)^2 = \frac{1}{(\lambda+1)^2 n^2} n\sigma^2 = \frac{1}{n(\lambda+1)^2} \sigma^2$$

which again confirms for $\lambda = 0$

$: \frac{1}{n} \sigma^2$

Part e)

See the graphs here and please review *q1.py* to review the implementation



Part f)

Problem 1) Part f)

By looking at the graph we see the phenomenon regarded as "bias - Variance Tradeoff"

higher values of λ result in a simpler model where variance is low, but bias is high

the lower values of λ result in a more complicated model where bias is low, but variance is high

The graph manifests the conflict of trying to minimize both bias and variance for minimizing the squared error.

We also see argmin (ESE) happens where bias and variance intercept.

Problem 2

Part a)

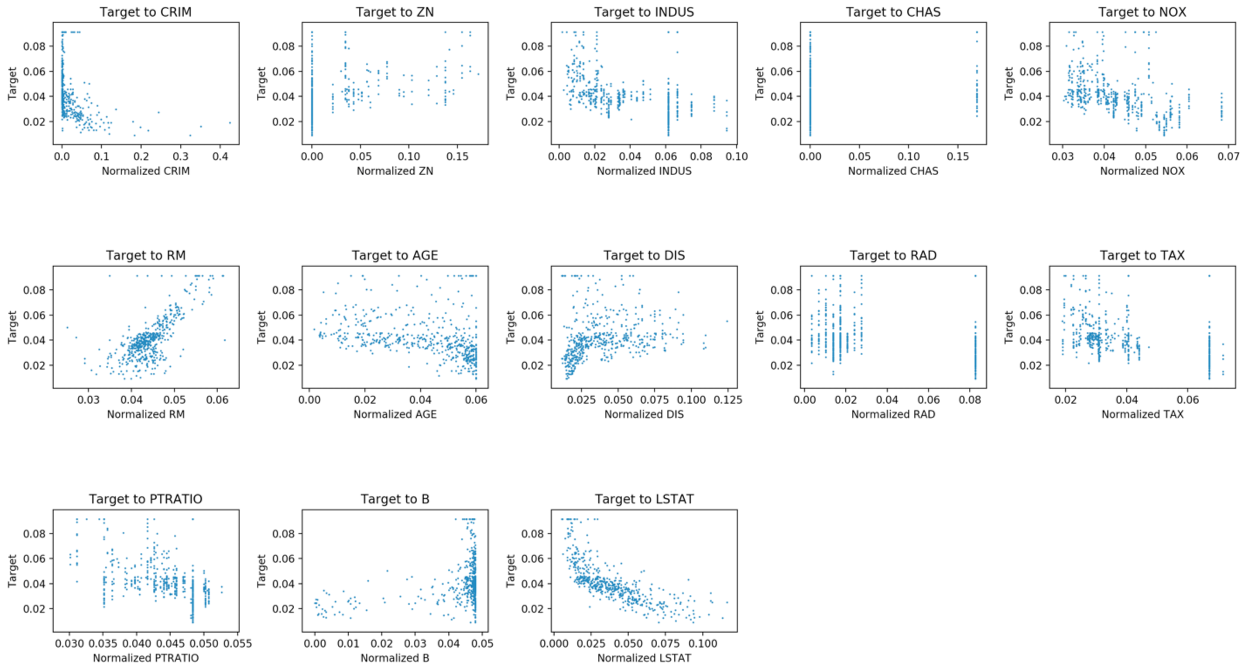
Review q2.py for the code.

Part b)

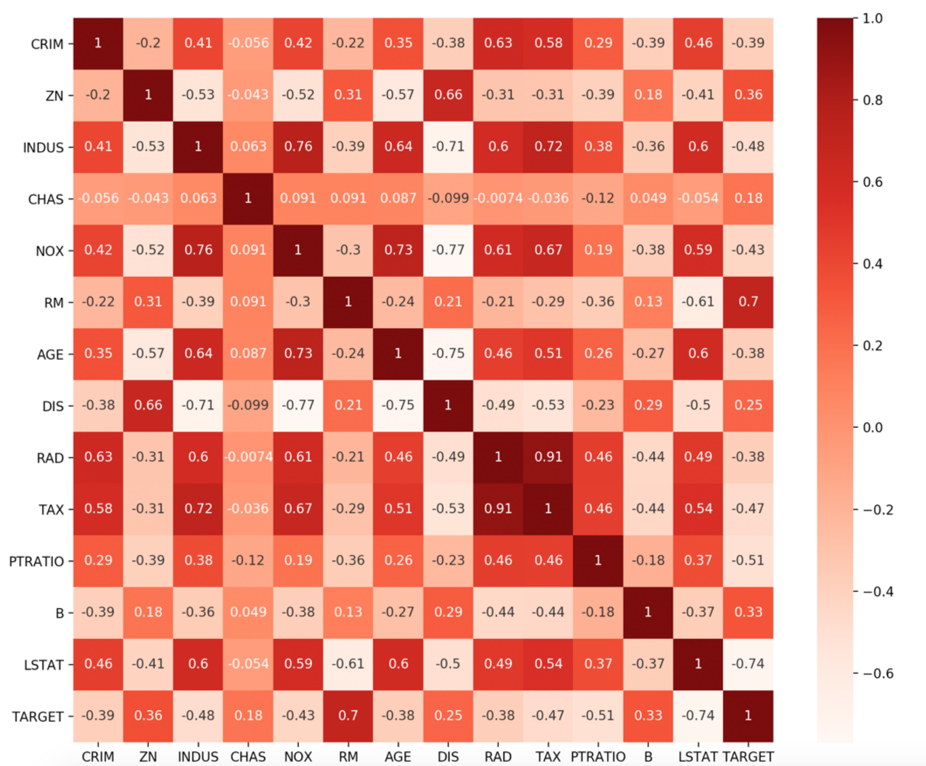
See summary statistics here: (I first transfer the data set into a pandas DataFrame and then extract the statistics with function *summarize_and_describe()*

```
             CRIM          ZN       INDUS        CHAS         NOX          RM         AGE         DIS         RAD         TAX     PTRATIO           B       LSTAT      TARGET
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     0.017233    0.019488    0.037861    0.011692    0.043518    0.044181    0.041131    0.038882    0.032865    0.041097    0.044153    0.043070    0.038724    0.041165
std      0.041020    0.039996    0.023322    0.042933    0.009091    0.004939    0.016884    0.021574    0.029966    0.016967    0.005179    0.011024    0.021855    0.016802
min      0.000030    0.000000    0.001564    0.000000    0.030205    0.025034    0.001739    0.011573    0.003442    0.018825    0.030144    0.000039    0.005295    0.009134
25%      0.000391    0.000000    0.017644    0.000000    0.035226    0.041375    0.027006    0.021517    0.013766    0.028087    0.041628    0.045328    0.021270    0.031103
50%      0.001223    0.000000    0.032942    0.000000    0.042208    0.043645    0.046485    0.032861    0.017208    0.033221    0.045575    0.047268    0.034767    0.038730
75%      0.017536    0.021436    0.061533    0.000000    0.048955    0.046563    0.056426    0.053157    0.082597    0.067046    0.048327    0.047846    0.051890    0.045672
max      0.424320    0.171491    0.094305    0.169031    0.068333    0.061723    0.059980    0.124240    0.082597    0.071576    0.052633    0.047927    0.116206    0.091344
---------
Features' Names: Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
---------
Head of the DataFrame:
       CRIM        ZN     INDUS  CHAS       NOX        RM       AGE       DIS       RAD       TAX   PTRATIO         B     LSTAT    TARGET
0  0.000030  0.030868  0.007853   0.0  0.042208  0.046222  0.039107  0.041904  0.003442  0.029798  0.036604  0.047927  0.015241  0.043845
1  0.000130  0.000000  0.024035   0.0  0.036795  0.045139  0.047324  0.050890  0.006883  0.024362  0.042585  0.047927  0.027973  0.039461
2  0.000130  0.000000  0.024035   0.0  0.036795  0.050510  0.036648  0.050890  0.006883  0.024362  0.042585  0.047436  0.012334  0.063393
3  0.000154  0.000000  0.007411   0.0  0.035932  0.049196  0.027471  0.062109  0.010325  0.022349  0.044738  0.047653  0.008998  0.061018
4  0.000329  0.000000  0.007411   0.0  0.035932  0.050243  0.032509  0.062109  0.010325  0.022349  0.044738  0.047927  0.016312  0.066133
---------
Dimensions of the Data:
        (506, 14)
---------
Number of Data Points:
        506
---------
```

Part c)



I have also included a heat map



Part d)

Please review *fit_regression()*

Part e)

Negative weight for a feature means that the feature has a negative linear correlation with the target and positive weight means the vice versa.
We see that INDUS has the lowest absolute weight of all. This is because (and we can also see that in the previous visualization) INDUS is not really associated with Target in a linear manner.

Feature	Weight
Intercept	1.2261171813
CRIM	-0.0491578877
ZN	0.0314812203
INDUS	0.0262559535
CHAS	0.0299759329
NOX	-0.3781393003
RM	1.1353113119
AGE	-0.0281678664
DIS	-0.2497925974
RAD	0.1368283270
TAX	-0.1806906661
PTRATIO	-0.7049094249
B	0.1994823106
LSTAT	-0.3089923515

Part f)

Please review function *predict()* and *messuare_errors()*
My result:

```
Mean squared error: 0.000073754
```

Part g)

I have chosen Mean Absolute Error and Max Error. I believe the first one is a good measure of efficiency because it can encourage the error to be close to zero. The other one is good because it can tell us what is the worse that our model might do.

```
Mean absolute  error: 0.005591969
Max error: 0.046246425
```

Part h)

If we look into the table of weights, we see that Max(abs(w)) belongs to RM, PTRATIO, and NOX.

These are the most significant factors contribution to the prediction of Target.

Problem 3

Part a)

Problem 3) Part a)

Show $\vec{w}^* = \text{argmin} \frac{1}{2} \sum \alpha^{(i)} (y^{(i)} - \vec{w}^T \vec{x}^{(i)})^2$

is given by: $\vec{w}^* = (X^T A X)^{-1} X^T A \vec{y}$

To show this, we need to look into derivatives [first, second]

$$P(\vec{w}) = \frac{1}{2} \sum \alpha^{(i)} (y^{(i)} - \vec{w}^T \vec{x}^{(i)})$$

it is notable that $\vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$ and $X = \begin{bmatrix} x^{(1)T} \\ \vdots \\ x^{(n)T} \end{bmatrix}$

and that $A = \begin{bmatrix} \alpha^{(1)} & & 0 \\ & \ddots & \\ 0 & & \alpha^{(n)} \end{bmatrix}$

Cont'd) if we look closely at $\sum a^{(i)}(y^{(i)} - \vec{w}^T\vec{x}^{(i)})^2$

we see that it is the same as:

$$\sum (a^{(i)\frac{1}{2}} y^{(i)} - a^{(i)\frac{1}{2}} \vec{w}^T x^{(i)})^2$$

$$= \|A^{\frac{1}{2}} \vec{y} - A^{\frac{1}{2}} \cancel{X} \vec{w}\|_2^2$$

$$\Rightarrow f(x) = \frac{1}{2}\|A^{\frac{1}{2}}\vec{y} - A^{\frac{1}{2}} X\vec{w}\|_2^2 = \frac{1}{2}\|\vec{a}\vec{y}\|_2^2 + \frac{1}{2}\vec{w}^T X^T A X\vec{w} - t^T A X\vec{w}$$

$$\Rightarrow \nabla f(x) = X^T A X\vec{w} - X^T A\vec{y} \qquad \underline{\text{we want } \nabla f(x) = 0}$$

$$\Rightarrow X^T A X\vec{w} - X^T A\vec{y} = 0 \Rightarrow \boxed{\vec{w} = (X^T A X)^{-1} X A\vec{y}}$$

we also know since $a_1 - a_n > 0 \Rightarrow \nabla^2 f(x) > 0 \Rightarrow$ The above does indeed minimice the cost function.

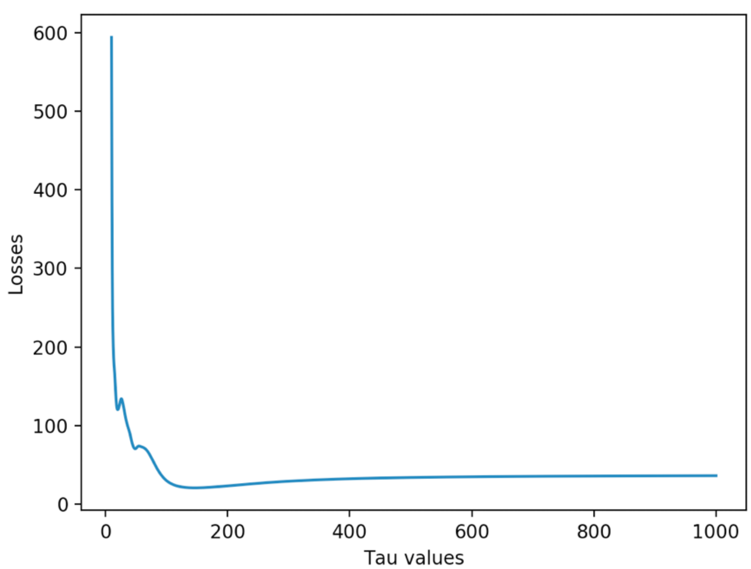
Part b)

Please see code and *LRLS()* . Here you can find some of the predictions on a test_set of size 10 Note that I have not normalized the data for this question (unlike q2) so the *tau* values in [10,1000] do create meaningful curve. The data can be normalized where Tau value is limited to [0.1,10]

```
Peredict:  25.148883114282924
Target: 24.0
<---->
Peredict:  22.887565561665127
Target: 21.6
<---->
Peredict:  31.431622140260867
Target: 34.7
<---->
Peredict:  31.743479134358502
Target: 33.4
<---->
Peredict:  32.8914950958465
Target: 36.2
<---->
Peredict:  26.32763304817411
Target: 28.7
<---->
Peredict:  20.74948909910244
Target: 22.9
<---->
Peredict:  18.141160822621128
Target: 27.1
<---->
Peredict:  13.363534077709602
Target: 16.5
<---->
Peredict:  17.80514555416795
Target: 18.9
<---->
```

Part c)

Please run *run_k_fold*() to try. Graph is here to review:



Part d)

As we can see, lower values of tau (values close to 0) result in very high losses. This predictable because lower values of tau result in bigger results for a(i) weights
While we increase the tau parameter, there is a point around 150 where losses are minimized and after that, as tau goes to infinity, we see that the loss function is mostly unchanged.

Part e)

Advantages of Locally Weighted Regression:

1. It is very flexible and can model population that have no known theoretical and deterministic model/structure, since the model predication is computed each time for each data entry separately
2. Unlike simple linear regression, it does not require a specification of a function to fit a model of all data into a single model.

Disadvantages:

1. The big thing is that unlike linear regression where we only need to have the model to predict targets, here we need to always have the whole data set to make predictions. Portability issues, memory, etc.
2. Locally Weighted Regression is much more computationally complex and expensive than simple linear regression