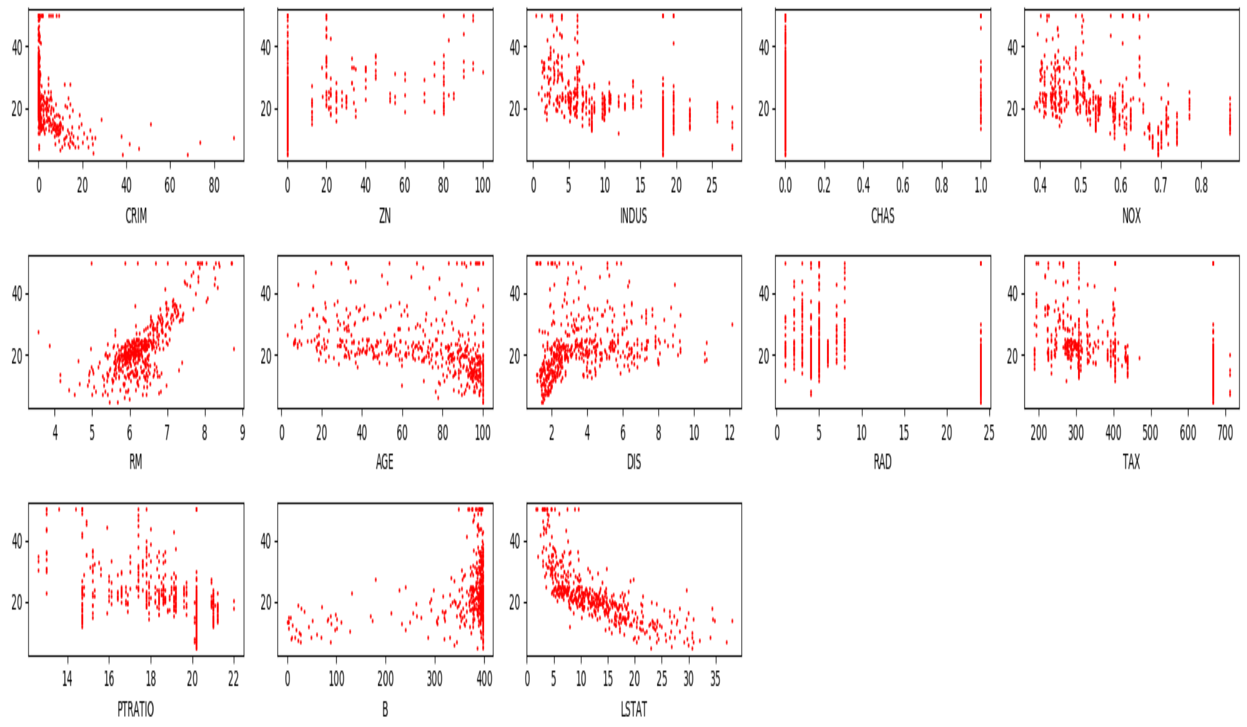
1 Learning basics of regression in Python (3%)

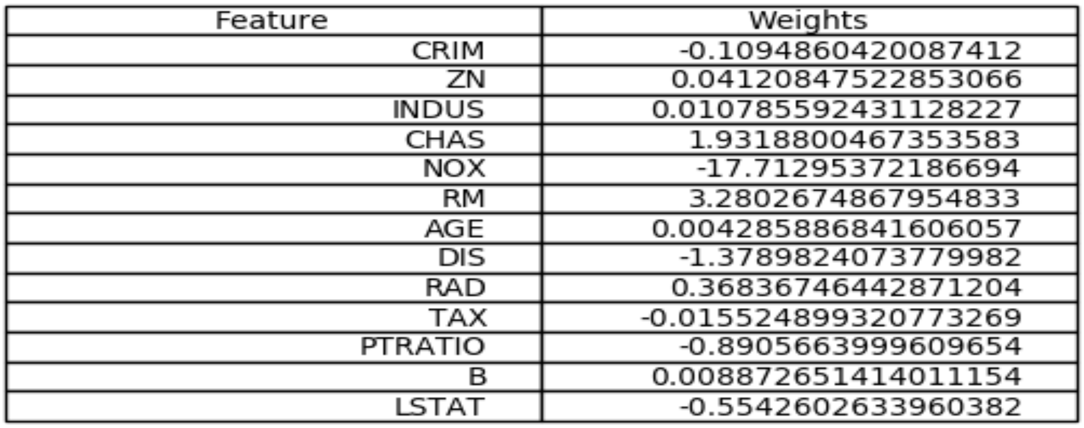
- Describe and summarize the data in terms of number of data points, dimensions, target, etc.

  We have 506 data points, each point has the dimension of 13, which mean 13 features we want to measure. Thus, x is of dimension (506,13). All these points have corresponding target value, which is y of dimension (506,). And it is the house price.

- Visualization:



- Tabulate each feature along with its associated weight and present them in a table. Explain what the sign of the weight means in the third column ('INDUS') of this table. Does the sign match what you expected? Why?



Feature	Weights
CRIM	-0.1094860420087412
ZN	0.04120847522853066
INDUS	0.010785592431128227
CHAS	1.9318800467353583
NOX	-17.71295372186694
RM	3.2802674867954833
AGE	0.004285886841606057
DIS	-1.3789824073779982
RAD	0.36836746442871204
TAX	-0.015524899320773269
PTRATIO	-0.8905663999609654
B	0.008872651414011154
LSTAT	-0.5542602633960382

The sign of weight of “INDUS” is positive which means the house price is positively related to “INDUS”. To be more precise, when “INDUS” goes up, the house price goes up. But it does not indicate the magnitude of the relationship. Yes. It matches what I expected. Because it seems quite random when we draw out all the points in the figure. But a very weak positive relationship can still be observed from the figure.

- Test the fitted model on your test set and calculate the Mean Square Error of the result.

MSE = 16.4860293731

- Suggest and calculate two more error measurement metrics; justify your choice.

Mean Absolute Error Metrics: $\frac{1}{n}\sum_{i=1}^{n}|y - w^t x|$

MAE = 3.02310960593

Mean Absolute Percentage Error: $\mathrm{M} = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$

MAPE = 14.5370741766

My choice would go to MAPE because it does not depend on scale.

- Feature Selection: Based on your results, what are the most significant features that best predict the price? Justify your answer.

I would choose LSTAT as the best feature to predict the price. Because of all the weights, LSTAT has the largest normalized weight compared to the rest of the features. This implies that LSTAT has the strongest relationship with house price.

2 Locally reweighted regression (6%)

1. $L(w) = (y - Xw)^T A (y - Xw) + \frac{\lambda}{2} w^T w$

$= y^T A y - y^T A X w - w^T X^T A y + w^T X^T A X w + \frac{\lambda}{2} w^T w$

$= y^T A y - 2 w^T X^T A y + w^T X^T A X w + \frac{\lambda}{2} w^T w$

$\nabla L(w) = -2 X^T A y + X^T A X w + \lambda w = 0$

$\Rightarrow X^T A X w + \lambda w = 2 X^T A y$

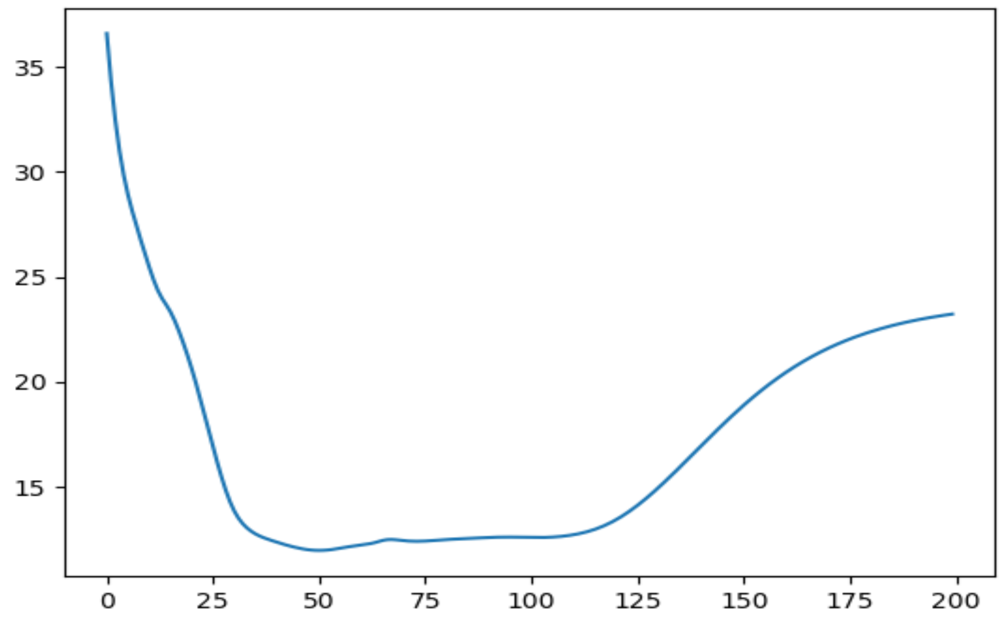
$\Rightarrow (X^T A X + \lambda I) w = 2 X^T A y$

~~$\Rightarrow X^T A X w^*$~~ $\Rightarrow w = 2(X^T A X + \lambda I)^{-1} X^T A y$

Then we have $w^* = (X^T A X + \lambda I)^{-1} X^T A y$

2. See q2.py

3. Use k-fold cross-validation to compute the average loss for different values of tau in the range [10,1000] when performing regression on the Boston Houses dataset. Plot these loss values for each choice of tau.

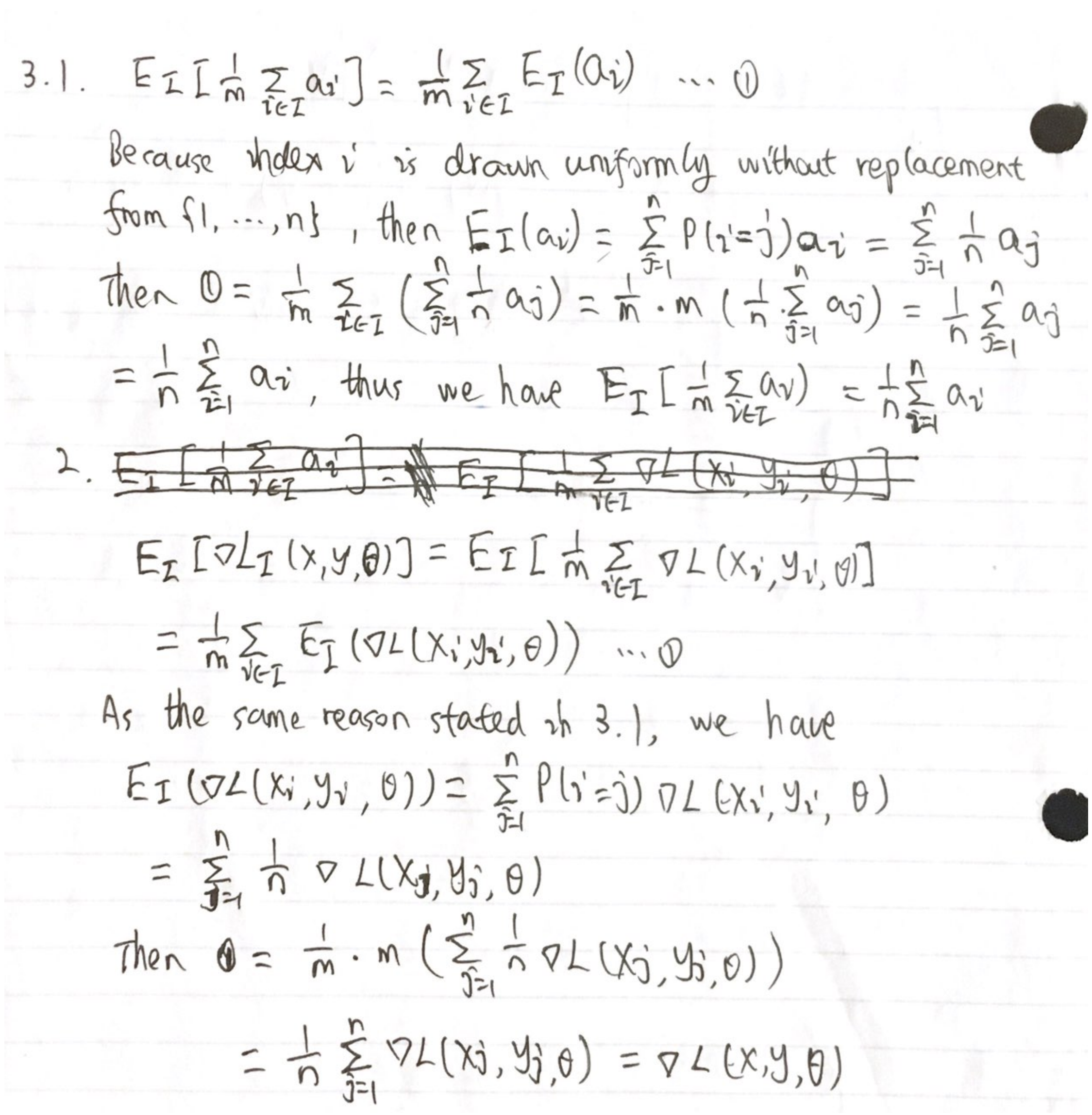


4. How does this algorithm behave when tau -> $+\infty$? When tau -> 0?

When tau -> $+\infty$, $a^{(i)}$ will not have any effect on deciding the average loss. $a^{(i)}$ is going to be uniformly distributed because $\frac{\exp(-||\mathbf{x}-\mathbf{x}^{(i)}||^2/2\tau^2)}{\sum_j \exp(-||\mathbf{x}-\mathbf{x}^{(j)}||^2/2\tau^2)}, = \frac{1}{n}$. The average loss will converge to a point. It behaves closer to a linear regression as tau increases.

When tau -> 0, the loss will increase to $+\infty$. It behaves closer to K-NN as tau decreases.

3 Mini-batch SGD Gradient Estimator (6%)



3.1. $E_I[\frac{1}{m}\sum_{i\in I} a_i] = \frac{1}{m}\sum_{i\in I} E_I(a_i) \quad \cdots ①$

Because index i is drawn uniformly without replacement from $\{1,\cdots,n\}$, then $E_I(a_i) = \sum_{j=1}^{n} P(i=j)\, a_i = \sum_{j=1}^{n} \frac{1}{n} a_j$

Then $① = \frac{1}{m}\sum_{i\in I}\left(\sum_{j=1}^{n}\frac{1}{n} a_j\right) = \frac{1}{m}\cdot m\left(\frac{1}{n}\sum_{j=1}^{n} a_j\right) = \frac{1}{n}\sum_{j=1}^{n} a_j$

$= \frac{1}{n}\sum_{i=1}^{n} a_i$, thus we have $E_I[\frac{1}{m}\sum_{i\in I} a_i) = \frac{1}{n}\sum_{i=1}^{n} a_i$

2. ~~$E_I[\frac{1}{m}\sum_{i\in I} a_i] = E_I[\frac{1}{m}\sum_{i\in I}\nabla L(x_i, y_i, \theta)]$~~

$E_I[\nabla L_I(x,y,\theta)] = E_I[\frac{1}{m}\sum_{i\in I}\nabla L(x_i, y_i, \theta)]$

$= \frac{1}{m}\sum_{i\in I} E_I(\nabla L(x_i, y_i, \theta)) \quad \cdots ①$

As the same reason stated in 3.1, we have

$E_I(\nabla L(x_i, y_i, \theta)) = \sum_{j=1}^{n} P(i=j)\,\nabla L(x_i, y_i, \theta)$

$= \sum_{j=1}^{n} \frac{1}{n}\nabla L(x_j, y_j, \theta)$

Then $① = \frac{1}{m}\cdot m\left(\sum_{j=1}^{n}\frac{1}{n}\nabla L(x_j, y_j, \theta)\right)$

$= \frac{1}{n}\sum_{j=1}^{n}\nabla L(x_j, y_j, \theta) = \nabla L(x, y, \theta)$

3. The importance of this result is that we can use subset of the data set points to

calculate the gradients such that we can further do the gradient descent to get the optimal weights, and this result would get pretty close as did with whole data set points which consume more computing resources. It takes less time but gains a descent result.

4.

4. (a) $L_I(x, y, \theta) = \frac{1}{m} \sum_{i \in I} l(x^{(i)}, y^{(i)}, \theta)$

$$= \frac{1}{m} \sum_{i \in I} (y^{(i)} - w^T x^{(i)})^2$$

$$\nabla L_I = \frac{1}{m} \sum_{i \in I} (-2) \cdot (x^{(i)}) \cdot (y^{(i)} - w^T x^{(i)})$$

(b) See q3.py

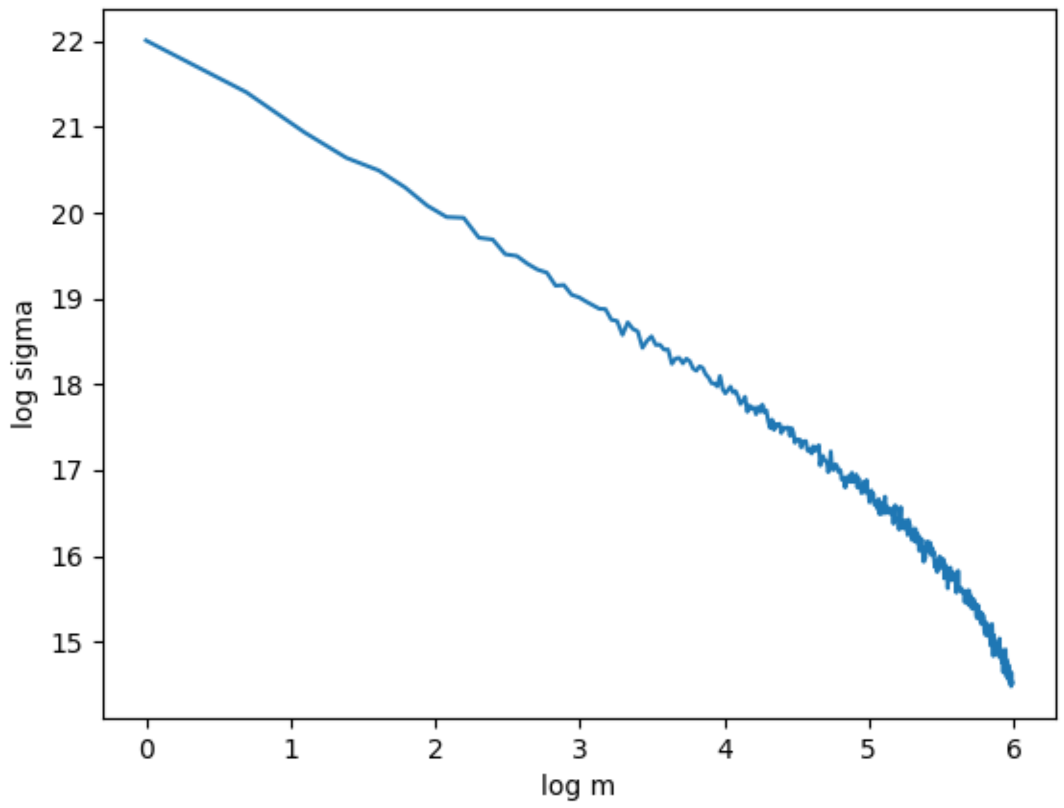
5. The following result is run from my laptop:

```
Square Distance Metric is: 6389521842.24

Cosine Similarity is: 0.998980609454
```

Cosine similarity is more meaningful in this case. Because the gradient itself has large values, even though a small difference between two large values would still be large. This is not accurate to measure how close two gradients are to numerical large base, making only measure comparative difference between two gradients and thus Cosine similarity is more meaningful here.

6.


log sigma
log m