Question 1

- Load the Boston housing data from the sklearn datasets module

Ans: See code file.

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

Ans:Shape of X: (506, 13); Shape of y: (506,1); Number of dimensions: 13

- Visualization: present a single grid containing plots for each feature against the target. Choose the appropriate axis for dependent vs. independent variables. **Hint:** *use pyplot.tight_layout function to make your grid readable*

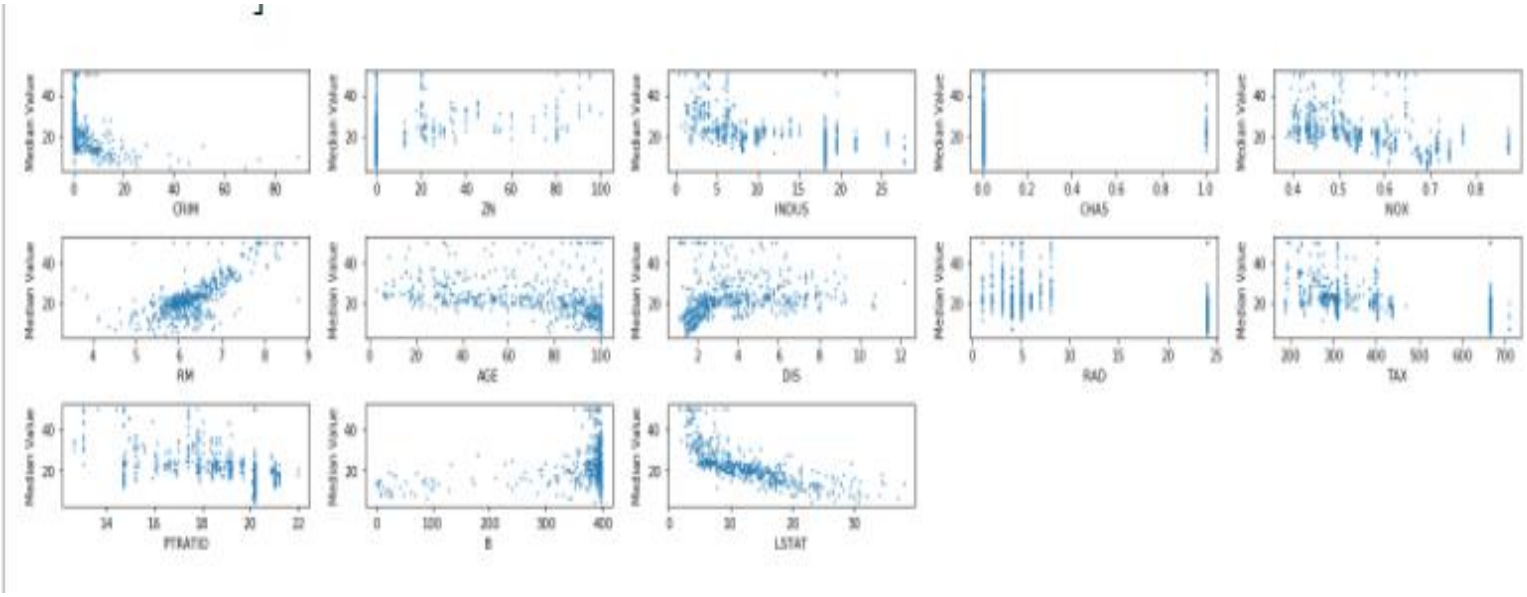


Fig 1: correlation between feature and value

Ans: y: median value of the house

x: value of corresponding variable

- Divide your data into training and test sets, where the training set consists of 80% of the data points (chosen at random). **Hint:** *You may find numpy.random.choice useful*

Ans: See code file.

- Write code to perform linear regression to predict the targets using the training data. Remember to add a bias term to your model.

Ans: See code file.

- 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?

Ans:

For one of the random generated test-train set combination, tabulated **weight vs feature** frame is shown below:

	w
Bias	39.150160
CRIM	-0.109648
ZN	0.044520
INDUS	0.000417
CHAS	3.432187
NOX	-18.381993
RM	3.407676
AGE	0.007932
DIS	-1.384799
RAD	0.293144
TAX	-0.012714
PTRATIO	-0.935280
B	0.007546
LSTAT	-0.511809

- Sign

The sign of w(INDUS) is positive, and the weight is relatively small.

- Meaning

The positive sign of it's weight means that the INDUS feature has a positive correlation with house price. But the small number of weight means the correlation is weak.

- Match of expectation

It matches the expectation. In figure 1, we can see the house value increases lightly with the increase of INDUS, but the increase is slow. Therefore, the expectation sign of INDUS should be positive, and the value of INDUS is expected to be small. So, the outcome matches the expectation.

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

Ans:

Using the data above, MSE is calculated based on the test set.

MSE = `25.0185885985`

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

Ans:

Method 1: **Root mean squared error (RSME)**

```
1 RMSE = np.sqrt(np.mean((X_test_biased.dot(w)-y_test)**2))
2 print("RMSE = {}".format(RMSE))
```

```
RMSE = 5.001858514440699
```

The root mean squared error method is like the MSE method. The only difference is RMSE take the square root of MSE, so the result is more readable and comparable to y.

Method 2: **Mean absolute error (MAE)**

```
1 MAE = np.mean(np.absolute(X_test_biased.dot(w)-y_test))
2 print("MAE = {}".format(MAE))
```

```
MAE = 3.4443104258042094
```

The MAE method takes the absolute difference between test y set and predicted y set. This method doesn't punish larger error as harsh as MSE does.

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

Ans:

RM feature is the best feature to describe the price.

By implementing linear regression, we get the corresponding weight of each feature. From the weight diagram above, it's obvious the RM feature has the largest weight of 4.43, which is almost double the amount of the 2nd largest weight. So, the RM is the most significant feature that best predict the price. Additionally, from the scatter chart, we can see the price increase dramatically with the RM feature.

Q2. Locally reweighted regression (6%)

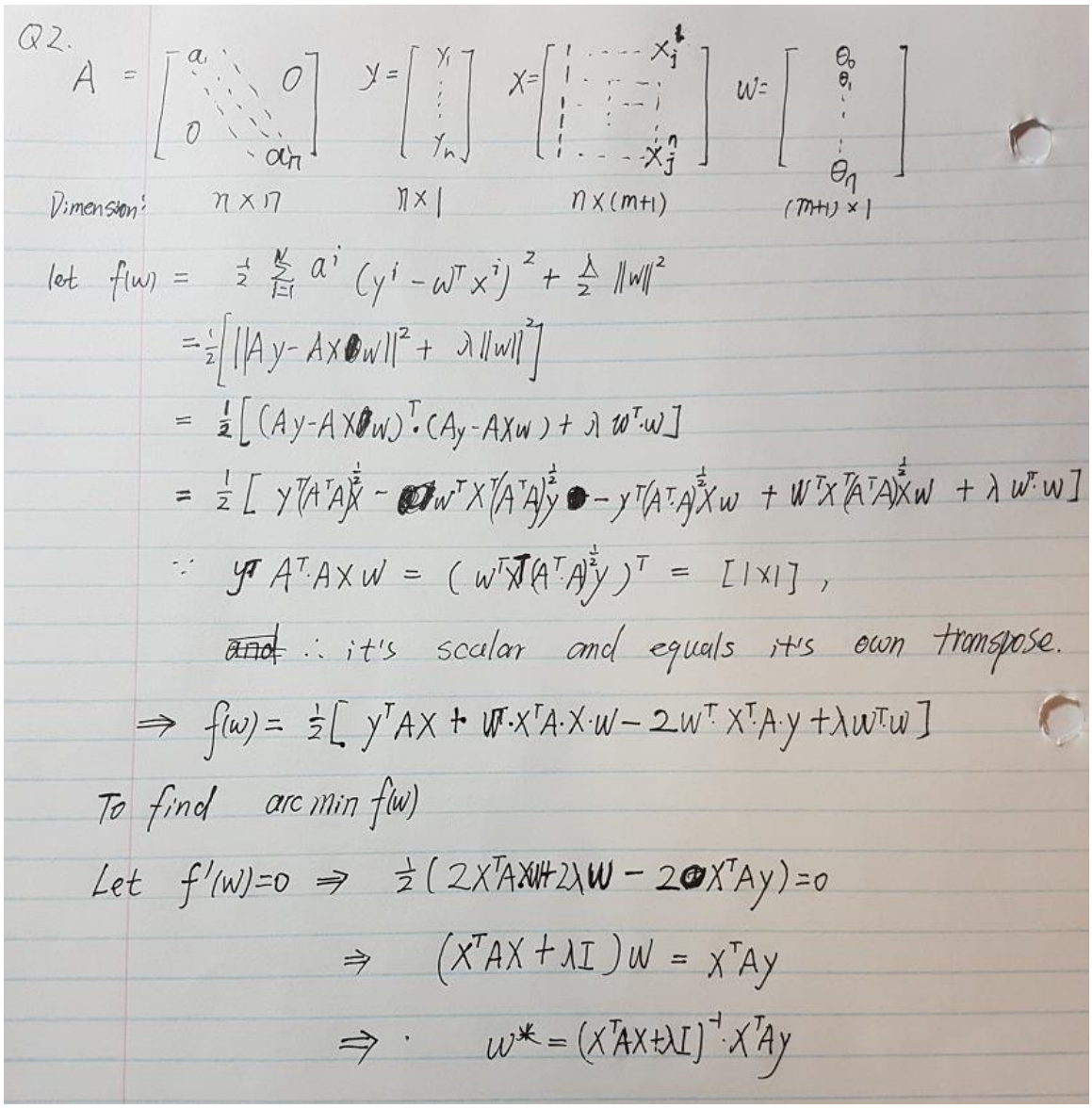
1. Given $\{(\mathbf{x}^{(1)}, y^{(1)}), .., (\mathbf{x}^{(N)}, y^{(N)})\}$ and positive weights $a^{(1)}, ..., a^{(N)}$ show that the solution to the *weighted* least square problem

$$\mathbf{w}^* = \arg\min \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2} ||\mathbf{w}||^2 \tag{1}$$

is given by the formula

$$\mathbf{w}^* = \left(\mathbf{X}^T \mathbf{A} \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^T \mathbf{A} \mathbf{y} \tag{2}$$

where $\mathbf{X}$ is the design matrix (defined in class) and $\mathbf{A}$ is a diagonal matrix where $\mathbf{A}_{ii} = a^{(i)}$



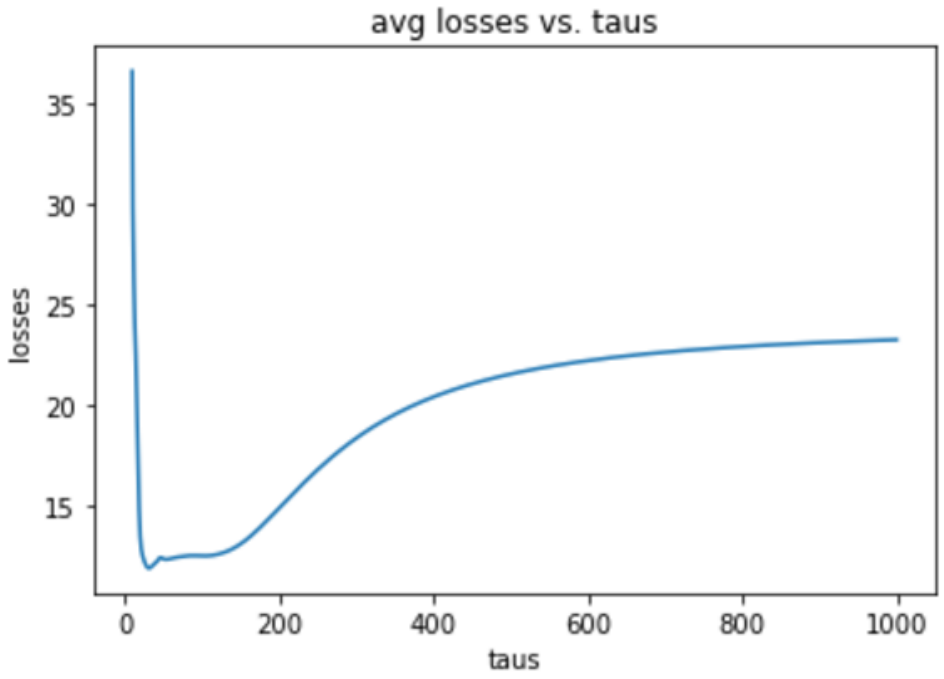
2. Locally reweighted least squares combines ideas from k-NN and linear regression. For each new test example $\mathbf{x}$ we compute distance-based weights for each training example $a^{(i)} = \frac{\exp(-||\mathbf{x}-\mathbf{x}^{(i)}||^2/2\tau^2)}{\sum_j \exp(-||\mathbf{x}-\mathbf{x}^{(j)}||^2/2\tau^2)}$, computes $\mathbf{w}^* = \arg\min \frac{1}{2}\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||^2$ and predicts $\hat{y} = \mathbf{x}^T\mathbf{w}^*$. Complete the implementation of locally reweighted least squares by providing the missing parts for q2.py.

   Important things to notice while implementing: First, do not invert any matrix, use a linear solver (numpy.linalg.solve is one example). Second, notice that $\frac{\exp(A_i)}{\sum_j \exp(A_j)} = \frac{\exp(A_i - B)}{\sum_j \exp(A_j - B)}$ but if we use $B = \max_j A_j$ it is much more numerically stable as $\frac{\exp(A_i)}{\sum_j \exp(A_j)}$ overflows/underflows easily. *This is handled automatically in the scipy package with the scipy.misc.logsumexp function.*

Ans:

Code in the separate file.

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




```
min loss = 11.9415251855364
```

4. How does this algorithm behave when $\tau \to \infty$? When $\tau \to 0$?

Ans.

The algorithm yield smaller loss values as the increase of taus.

As tau → 0, the algorithm yield big loss.

As the tau increases toward infinity, loss values start to reduce and reach a steady-state level as tau → infiniy

3 Mini-batch SGD Gradient Estimator (6%)

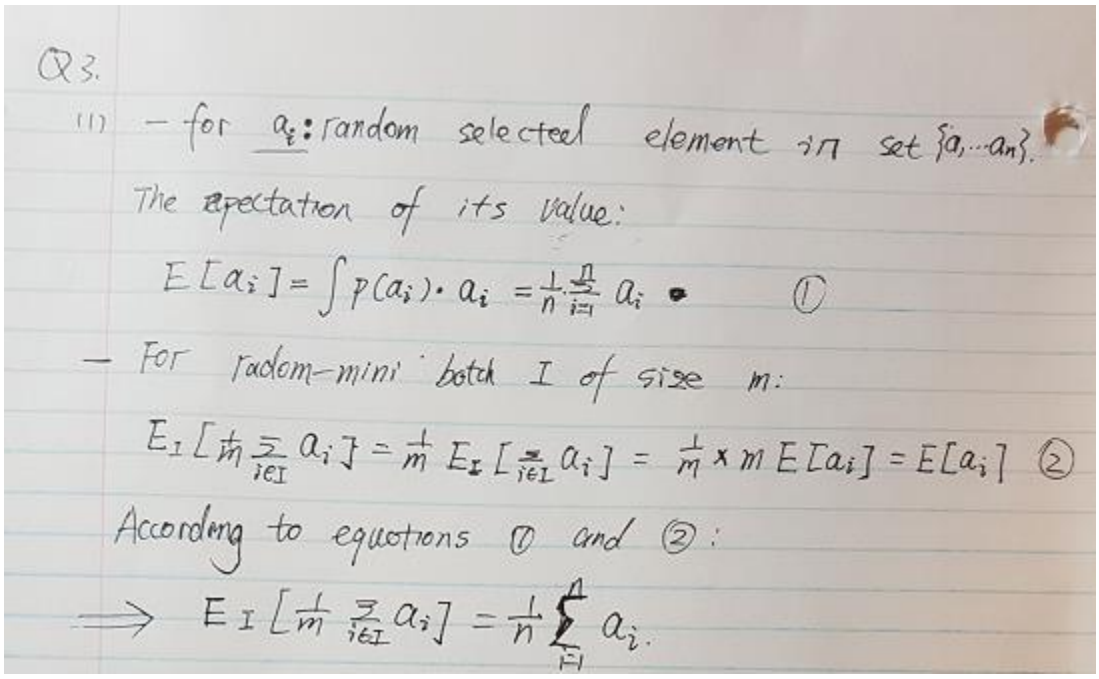
Consider a dataset $\mathcal{D}$ of size n consisting of $(\mathbf{x}, y)$ pairs. Consider also a model $\mathcal{M}$ with parameters θ to be optimized with respect to a loss function $L(\mathbf{x}, y, \theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta)$.

We will aim to optimize L using mini-batches drawn randomly from $\mathcal{D}$ of size m. The indices of these points are contained in the set $\mathcal{I} = \{i_1, \ldots, i_m\}$, where each index is distinct and drawn uniformly without replacement from $\{1, \ldots, n\}$. We define the loss function for a single mini-batch as,

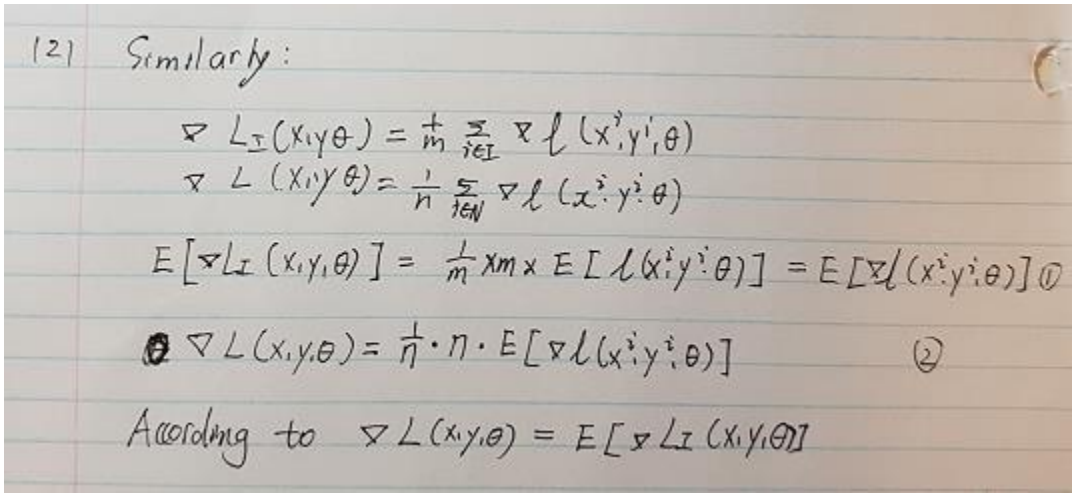
$$L_{\mathcal{I}}(\mathbf{x}, y, \theta) = \frac{1}{m}\sum_{i \in \mathcal{I}} \ell(\mathbf{x}^{(i)}, y^{(i)}, \theta) \tag{3}$$

1. Given a set $\{a_1, \ldots, a_n\}$ and random mini-batches $\mathcal{I}$ of size m, show that

$$\mathbb{E}_{\mathcal{I}}\left[\frac{1}{m}\sum_{i \in \mathcal{I}} a_i\right] = \frac{1}{n}\sum_{i=1}^{n} a_i$$



2. Show that $\mathbb{E}_I \left[\nabla L_I(\mathbf{x}, y, \theta)\right] = \nabla L(\mathbf{x}, y, \theta)$



3. Write, in a sentence, the importance of this result.

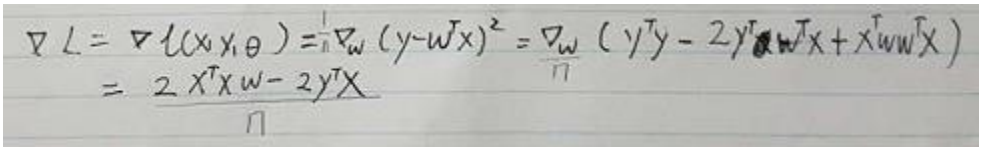
Ans:

The random selected mini-batch has the identical mean and loss function gradient with the entire set, which allows mini-batch gradient descent to yield similar accuracy in shorter runtime compares with big batch.

4. (a) Write down the gradient, ∇L above, for a linear regression model with cost function $\ell(\mathbf{x}, y, \theta) = (y - w^T\mathbf{x})^2$.
   (b) Write code to compute this gradient.

Ans:

a.



b.

Code is in the separate python file.

5. Using your code from the previous section, for $m = 50$ and $K = 500$ compute

$$\frac{1}{K}\sum_{k=1}^{K}\nabla L_{\mathcal{I}_k}(\mathbf{x}, y, \theta)$$

, where $\mathcal{I}_k$ is the mini-batch sampled for the kth time.

Randomly initialize the weight parameters for your model from a $\mathcal{N}(0, I)$ distribution. Compare the value you have computed to the true gradient, ∇L, using both the squared distance metric and cosine similarity. Which is a more meaningful measure in this case and why?

[Note: Cosine similarity between two vectors $\mathbf{a}$ and $\mathbf{b}$ is given by $\cos(\theta) = \frac{\mathbf{a}\cdot\mathbf{b}}{||\mathbf{a}||_2||\mathbf{b}||_2}$.]

Ans:

For the purpose of illustration, a pseudo-random is used here by setting np.random.seed(1).

The test run result is show below.

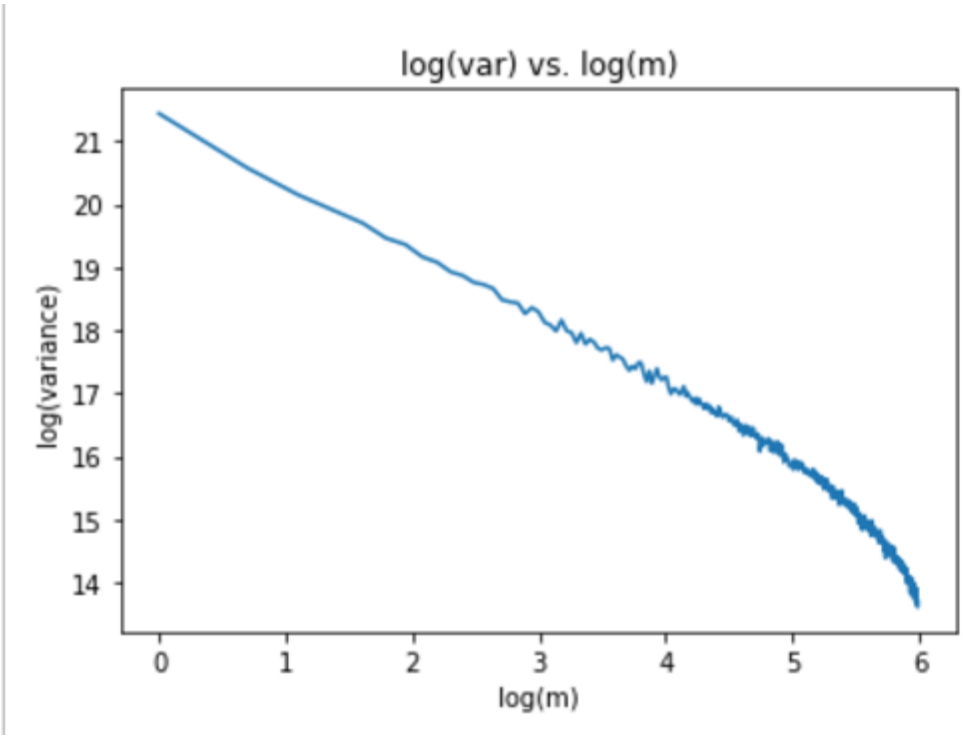
```
------ Loading Boston Houses Dataset ------
Loaded...
Total data points: 506
Feature count: 13
Random parameters, w: [ 1.62434536 -0.61175641 -0.52817175 -1.07296862
 0.86540763 -2.3015387
  1.74481176 -0.7612069   0.3190391  -0.24937038  1.46210794 -2.06014071
 -0.3224172 ]
-----------------------------------------



True_grad :
 [ -3.83595524e+03  -1.99712251e+04  -1.52498029e+04  -1.05047599e+02
  -7.95845338e+02  -9.32340185e+03  -9.57634361e+04  -5.96123591e+03
  -1.25808975e+04  -5.73750695e+05  -2.70294773e+04  -5.61461009e+05
  -1.73159509e+04]
Mini-batch grad:
 [ -3.89932526e+03  -1.99126233e+04  -1.52246155e+04  -1.01006954e+02
  -7.94539357e+02  -9.31495016e+03  -9.56336042e+04  -5.96357795e+03
  -1.25167459e+04  -5.72921584e+05  -2.70086232e+04  -5.60956682e+05
  -1.73001798e+04]
squared distance metric: 971605.5939824795
cosine similarity: 0.9999999564132128
```

The cosine similarity is a more meaningful measurement here. Cosine similarity doesn't consider the magnitude of 2 vectors. Cosine is a monotonically decreasing function for the interval [0, 180] degree. The 0.9999 cosine means the nearly maximum similarity between mini-batch gradient and true batch gradient. While Euclidean distance between two vectors with different length can be quite large here. It's hard to tell the similarity by visualizing the squared distance metric.

6. For a single parameter, w_j, compare the sample variance, $\tilde{\sigma}_j$, of the mini-batch gradient estimate for values of m in the range [1,400] (using $K = 500$ again). Plot $\log \tilde{\sigma}_j$ against $\log m$.

Ans:

For the purpose of illustration, a pseudo-random is used here by setting np.random.seed(1).



The elements with index=1 in the w gradient vector are chosen here. The variance represents the level of noise among the K times mini-batch gradient calculation. According to the graph, the variance decreases gradually with the increasing sample size m.