1.I would like to predict the price of gold by examining the annual prices of gold in the past. Logistic regression is the best choice for this because I anticipate that the price will generally follow a linear pattern, which gradient descent algorithms should be able to estimate.

2. I chose a database that shows the annual prices of gold:
https://www.kaggle.com/tunguz/gold-prices
The data was fairly simple so I did not do a lot of EDA to it. One thing I did do was simplify the dates in the csv, I just represented them as the numbers from 0-69 in my code instead of the full dates because this was easier to work with. There were only two columns of data, so all of the data is used in my program with no other changes.

3. The cost algorithm I used is known as the “Mean squared error which is represented by the equation:

$$1/2M * {}^{M}\sum_{(i=1)} ((\theta_0 + \theta_1 * x(i)) - y(i))^2$$

There are two derivatives that are computed in my batch and stochastic gradient descent function using these equations:

$$\partial / \partial \theta_0 = 1 / M * {}^{M}\sum_{(i=1)} ((\theta_0 + \theta_1 * x(i)) - y(i))$$

$$\partial / \partial \theta_1 = 1 / M * {}^{M}\sum_{(i=1)} ((\theta_0 + \theta_1 * x(i)) - y(i)) * x(i)$$

4.I used tensorflow to implement two more optimization algorithms, RMSprop and Adam. The RMSprop optimization performed better than Adam, but they both performed worse than my basic batch gradient descent.

RMSprop usually had a loss somewhere in the one hundred thousands
Adam usually had a loss somewhere in the two hundred thousands
Vanilla gradient descent was around seventy thousand.

Judging from the data we should definitely not use optimization on this problem because they performed significantly worse than the default. My guess as to why would be because my data set, while generally having a linear rise over time, is fairly inconsistent and randomly makes large jumps messing with the algorithms.