Homework 1 Report:

Task 1:

Theory:

a. Probability of drawing an ace from a standard 52-card deck in one draw:
P(Drawing an Ace) = P(A) = n(A)/n(S) = 4/52 = 1/13 = 0.08

b. Probability that the first success occurs on the 4th draw using the geometric distribution formula.
P(x = k) = (1 – P)^(k – 1) * (P)
P(Drawing an Ace) = 0.08 (from part a)
K = 4;
P(x = 4) = (1 – 0.08)^(4 – 1) * (0.08)
P(x = 4) = (0.92)^(3) * 0.08
P(x = 4) = 0.06

Python Simulation:

c. The average number of draws for the first success for 100,000 trials was ~10.56

d. The probability that the first success is at draw 4:
~0.06

At-Home Experiment:

a. Trial numbers for 30 physical trials:
4, 17, 18, 1, 16, 11, 10, 3, 14, 9, 5, 10, 15, 10, 8, 15, 8, 9, 18, 13, 21, 1, 5, 3, 13, 9, 22, 34, 1, 11
Average number of draws:
11.13

b. Estimate the frequency of the first success occurring at draw 4:
~0.03

Write Up:

The theoretical probability and the Python simulation results will be the most similar because the theoretical probabilities represent the probability as the number of trials approaches infinity and the Python simulations use a much larger number of trials than the physical trials. There is a small number of physical trials so by the Law of Large Numbers, the probability might be slightly or very different from the theoretical probability simply because there has not been enough data gathered for the mean to converge to the expected value.

Task 2:

GitHub Link to Notebook:

cs325-homework/homework-1 at main · efisk15/cs325-homework

1. Provide a justification for your design and experimental decisions.

   My training data is overfitting with custom model I built, but with the epoch equal to 1000 and the learning rate set to 0.01, I was able to get the highest accuracy of all of my previous models. Obviously, overfitting is something I need to work on, but I am not sure what I should change to fix this. I assume it is something I might need to change with my hyper parameters, or possibly with the way I am cleaning the data. I tried using both standard normalization and min-max normalization and using min-max seemed to work best, giving me the lowest cost. Changing the seed did not seem to contribute much of a change in the cost or accuracy.

2. Can you provide any insights on model performance?

   SKLearn Validation Accuracy: 0.7727

   From Scratch Validation Accuracy: 0.7655

   I think that I can definitely get my accuracy higher by spending more time testing different hyperparameters. The SKLearn model probably has a better way of handling hyperparameters that might be a little more fine-tunned than the ones I provided.

3. Submit to the competition your best performing model. What was your final test score and ranking?

   Final test score: 0.7655

   Ranking: 3643

Task 3:

1. I really like the amount of content that we have covered so far in the class. I feel like in the past, I have had classes that would move very slowly, and we would not cover very many new concepts. In this class however, I feel like we are covering a lot of content

but at a rate where you are able to reiterate on more difficult topics that need to be repeated.

2. I would say part 5 of task 2 has definitly been the most difficult simply because of the amount of work for it. It was not too difficult because the write-up was very insightful in what we needed to do for it; it was simply just a lot of code that was somewhat complex.

   I also had some trouble cleaning the data but that is just because I am new to it. I met with a TA and he helped a lot so it was not too much of a problem.