(At bottom are AI screenshots and citations)

(Denormalization formula for part 2c used Google Colab autofill AI)

IMPORTANT: This is a generalized version of my steps. A lot of the specifics are in the Colab notebook as text cells (INCLUDING PLOTS AND NUMERICAL VALUES)

# Problem 1 (Exploratory Data Analysis)

## 1a write-up:

* None of the times data is approximately normally distributed and this is likely due to factors within the company that cause some time spans to have more posts than others.
* For the hours, this company likely favors some times of posting much more than others like almost any reasonable marketing strategy would do. The count could peak as early as hour 3 due to it being daytime in another timezone or to make sure everyone sees their product before heading off to work. It also sharply increases from hour 9 to 13 (10:00 A.M. to 2:00 P.M.) since that might be the peak time to shop, especially on weekends in their home timezone. It could be too late for people to shop at later times (hence very few posts at hours 16-23), being either late evening or slightly past midnight depending on time zone
* For weekdays, it seems like the posts by weekday are around the same for Sunday through Thursday before increasing on Friday and Saturday. This is likely because people are more willing to go shopping on Friday and Saturday (such as due to not having work or due to getting their paychecks) and being busy from Sunday through Thursday
* For months, the company currently has fewer posts in some months and more in other months; this could be due to financial factors (such as fiscal quarters) or even the demand of their products varying with seasons or weather (ex: winter makes skin drier on average)

## 1b write-up:

* Here, the distributions would seem to be normal but mostly in the decreasing direction (only the last bit of the left half and mostly the right half of a bell curve). This is probably because the total\_reach and total\_interactions values cannot be negative

## 1c write-up:

* I see that the page\_total\_likes is strongly correlated with the post\_month (in the positive direction). This is probably because the likes only increased with increasing post\_month values since this data is only from one year (being 2014). Aside from that, total\_interactions and total\_reach seem to be somewhat correlated (in the positive direction, and it makes sense that one would more likely increase than decrease if the other increases) but this correlation is not strong enough for me to make any conclusions about the relationships between these total\_interactions and total\_reach just yet
* The rest of these pairs of features don't seem to have any significant correlations (for both the positive and negative correlation coefficients)

# Part 2 (Linear and Multilinear Regression):

## Data Preprocessing

* I saw that there were many high outliers that were heavily skewing the data statistics. I removed them using a loop that iterated over the dataset, removing points with ‘total\_interactions’ more than 3 standard deviations away from the mean each iteration, until the next iteration removed no more data points. Afterwards, I compared, for both the original and outlier-removed dataset, the means and standard deviations for the whole datasets and when grouped by feature values, and the means and standard deviations significantly decreased without changing too much of the feature statistics.
* I then saw that the standard deviations for ‘total\_interactions’ based on post\_hour==16 and 19 through 23 were NaN, indicating that something was wrong. I printed the counts of samples in all categories, finding that the counts for samples with these post\_hour values and for type==3 were far too few to fit in with the rest of the data, so I removed those samples as well.
* After all this, in preparation for model building, I defined a train-test-split function to get training and testing sets, and a z\_score\_normalize function to return z-score-normalized versions of each of these sets. I then created tensors out of them for use with PyTorch.

## Linear Regression:

* For linear regression, I chose the model y\_pred = slope\*(page\_total\_likes) + b for constants ‘slope’ and ‘intercept’. I used the mean squared error between y and y\_pred as a loss function, and I used gradient descent with 300 epochs on the normalized X\_train and y\_train to find optimal values for normalized slope and intercept. I then had another function to denormalize the slope and intercept with equations figured out through AI (screenshots at bottom)
  + I did gradient descent on the normalized training set because otherwise, the scales between features were far too different and I would get nans when doing gradient descent.
  + My function took an index (IE column) of X (IE a single feature) to train the slope and intercept on.
* I tested with various values for learning rate to see the MSE loss from each of the corresponding linear models for each feature, and upon seeing they were around the same, I chose lr = 0.1 because it was the biggest and not too big.
* I then stuck with learning rate = 0.1 and plotted the linear regression lines from each feature with respect to ‘total\_interactions’ along with their corresponding MSE losses. From there, I chose my feature to be page\_total\_likes because it was among the features with the lowest MSE losses and it was the only continuous feature (plus the only one I could discern a clear logical relationship with, IE page\_total\_likes increasing means total\_interactions increases). All the other features were categorical with no clear trend in the categories alone vs chosen metric value from my EDA (and because the variation within each category was so high, I felt I wouldn’t get any real meaning from a category-based model).
* In all, my equation, rounded to 4 decimal places, was ‘total\_predictions’ = -0.0001 \* ‘page\_total\_likes’ + 147.3938

## Multi-linear Regression:

* I saw that the computations weren’t very intensive when choosing all features as my inputs for my multilinear model. As such, I made y\_pred as a sum of coefficients times feature values plus a constant value (an intercept). I had these coefficients (including the intercept) in a tensor and I used gradient descent to train them.
* To confirm that it was converging, I even had a block to print the values of the gradients at the end of the last iteration before resetting the gradients to 0. The gradients at that point were very close to 0.
* I will note that when I tried training another model on the testing data, the coefficients were significantly different from when I trained on the training data.
* Equation rounded to 4 decimal places: y\_pred = -0.0\*page\_total\_likes - 7.835\*type + 0.1467\*category + 6.6479\*post\_month + 3.2115\*post\_weekday - 1.2314\*post\_hour - 6.298\*paid + 98.3161
* The MSE for this is very similar to that for the single linear regression model. This is probably because the only difference in this model is also using the categorical features as inputs, as they were in such a small range of discrete values each with no clear trends (WRT the ‘total\_interactions’ value) by themselves that they clearly did not make much of a difference.

## Using ‘category’ and ‘paid’:

* Noted in my notebook, but I saw that the means and stds were distinct for ‘category’ values and slightly distinct for ‘paid’ values (both with respect to total interactions). These were the only features where there were enough samples from each feature value to constitute a significant portion of data AND where means and stds between feature values were distinct. I then made a product of linear terms relating to these, being (p1 \* category + p2) \* (p3 \* paid + p4) to include interaction and quadratic terms.
* Looking back, this was a mistake as again, they are categorical features in such a small range and with no clear trend by themselves in regards to the ‘total\_interactions’ value. I feel that ‘page\_total\_likes’ would have a clear correspondence with ‘category’ and ‘paid’ as higher page\_total\_likes clearly leads to higher total\_interactions.
  + If I had more time, I’d retrain this model to use ‘page\_total\_likes’ as well.
* Equation rounded to 4 decimal places: y\_pred = ( 0.0513\*category + 18.2641 ) \* ( 0.0297\*paid + 9.2119 )

## Problem 2 boxplots:

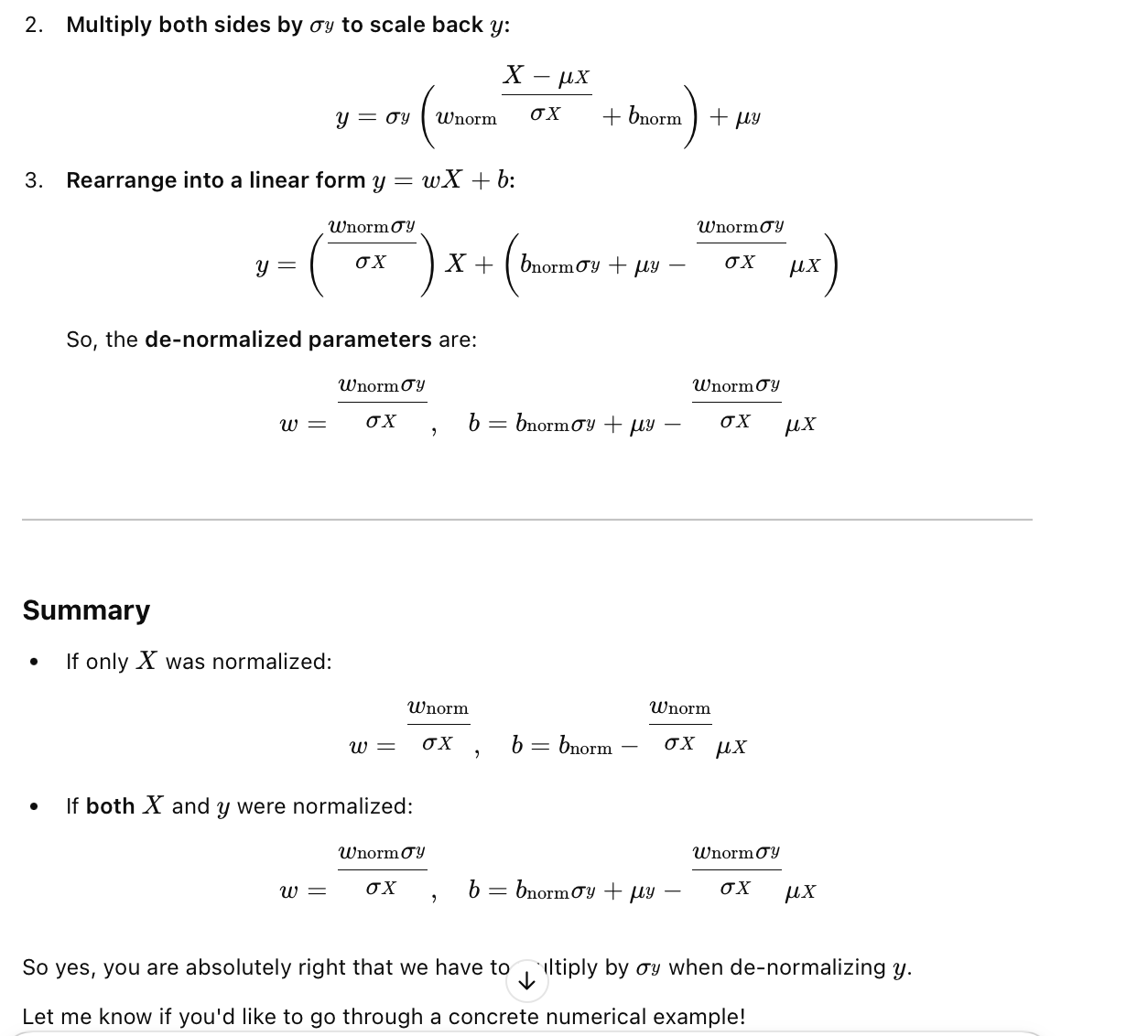
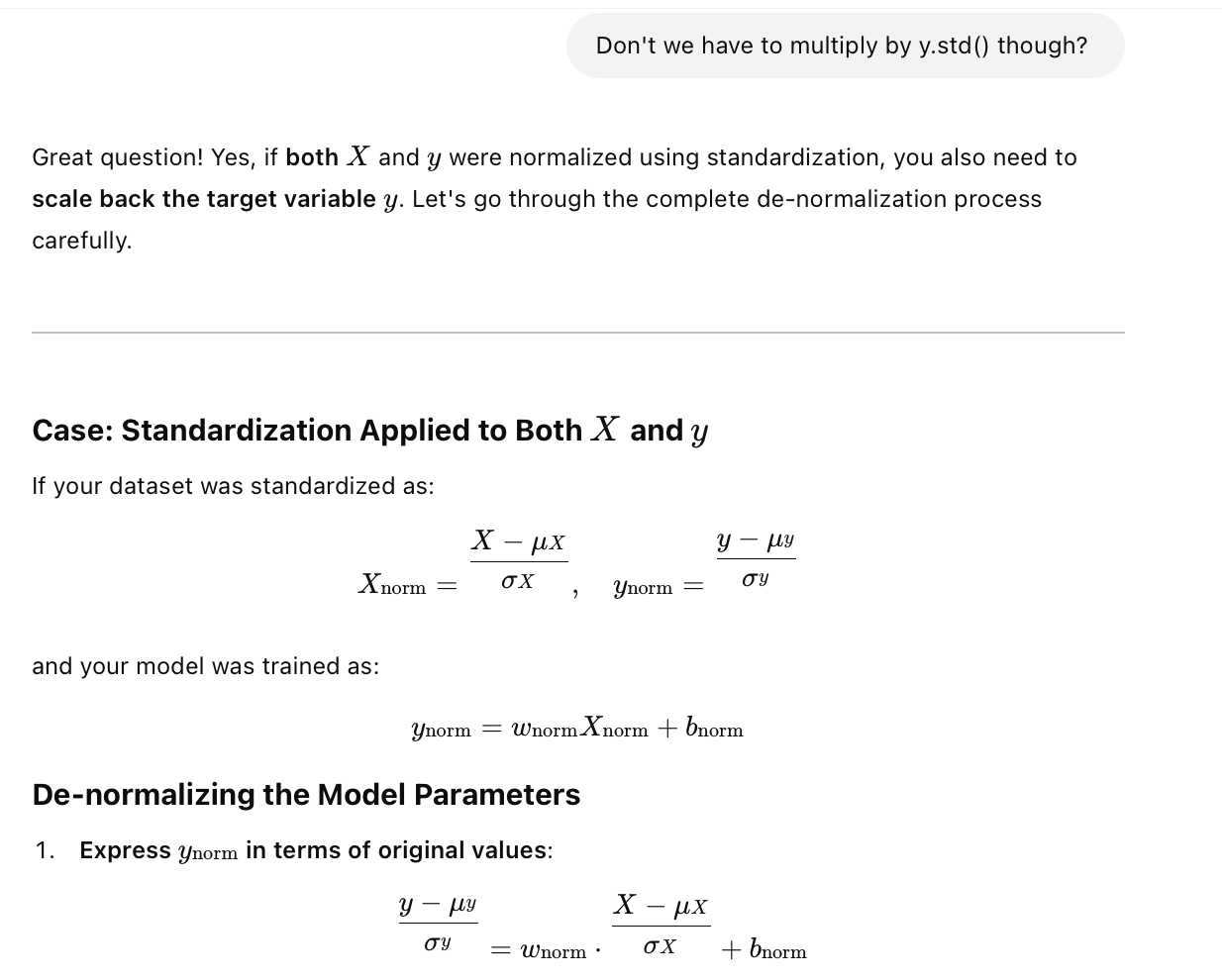
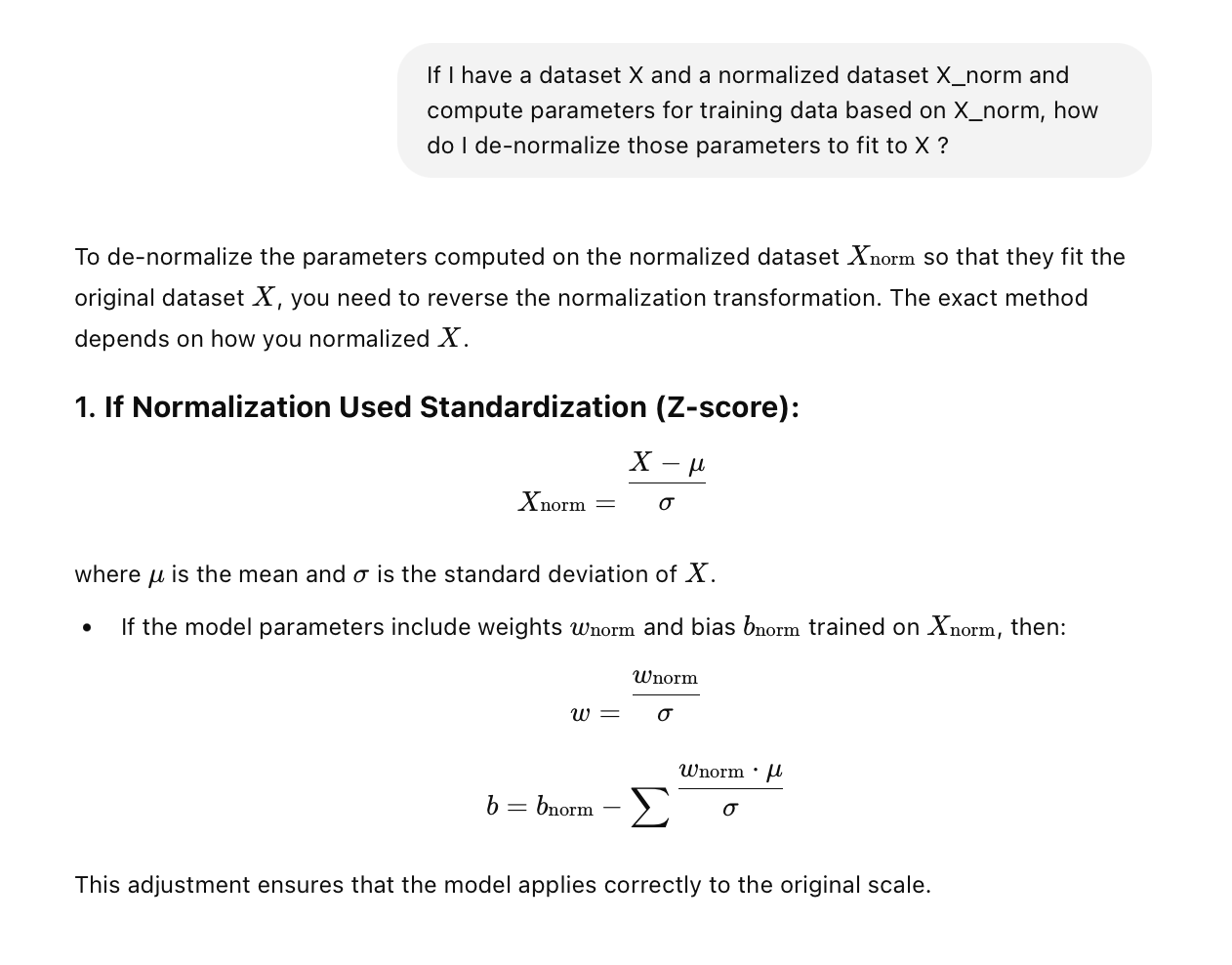
* I chose the groups based on the top fifth, second-top fifth, etc. of the ‘total\_interactions’ value because I felt that was a simple way to compare predictions across different ranges of ‘total\_interactions’ values
* The only boxplot to show the predicted box was the multilinear model. Upon printing out the plotted values, I saw that only the multilinear model had a good variety of predicted values, and that the linear and nonlinear models all had such a small deviation to the point their “boxes” wouldn’t show. Also, the predicted vs actual box was only really good for the 4th quantile.

# Problem 3 write-up (Thresholded Accuracies):

* With thresholded accuracies, the nonlinear model performs the best, slightly outperforming the multilinear model and noticeably outperforming the linear model. This is likely because in the nonlinear model, I take in terms that I deem to be very relevant to calculating 'total\_interactions' (justified at the top of part 2c in a text box) and I am accounting for interaction between them without too much model complexity.

# Problem 4 write-up (Cross Validation):

* I started by making functions to quickly build linear, multilinear, and nonlinear classifiers out of a given train/test split, combining all the steps I used in parts 2a through 2c.
* Over all 3 train/test splits I chose, the linear model strongly outperformed the multilinear and nonlinear models. This is probably due to model complexity and the fact that ‘page\_total\_likes’ is the only feature I can see has a clear relationship (increasing) with ‘total\_interactions’, and I think the other feature coefficients just contributed as ‘noise’ in some sense that lead to slight model overfitting.
  + I note that the accuracy difference is not ultra high with the nonlinear model that didn’t use ‘page\_total\_likes’ since even the linear classifier was only able to predict an ‘average’ value for the most part.

AI screenshots for formula of de-normalizing slope and y-intercept for problem 2a: 

Problem 2b formula screenshots:

