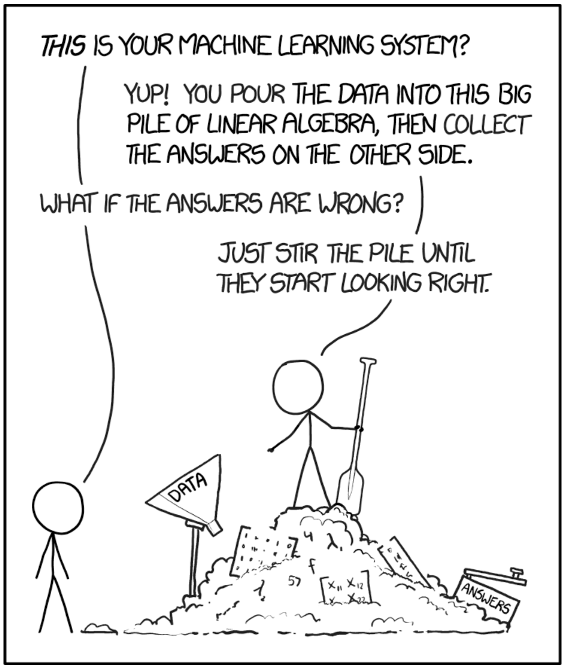
Nolan Pestano

Professor Culmer

Machine Learning

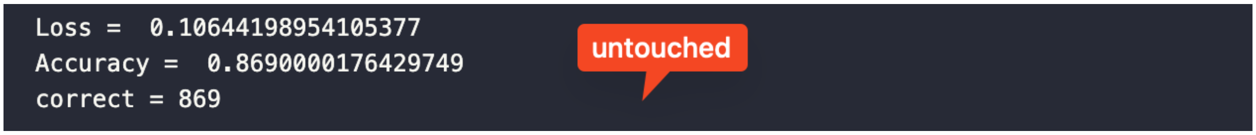
2024-10-11


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

I present my hacky machine learning model primarily using Keras to predict if given a collection of various user inputs, if they will make a purchase as denoted by the revenue column.

Initially, I was trying to work with the previous implementations of our past regression code, but I was struggling quite badly. Ultimately I turned to Keras and realized that those libraries exist for altruistic reasons, and that there is no need to sweat over the small things (although ironically I am more confident in understanding ML even with the “failures” I had originally).

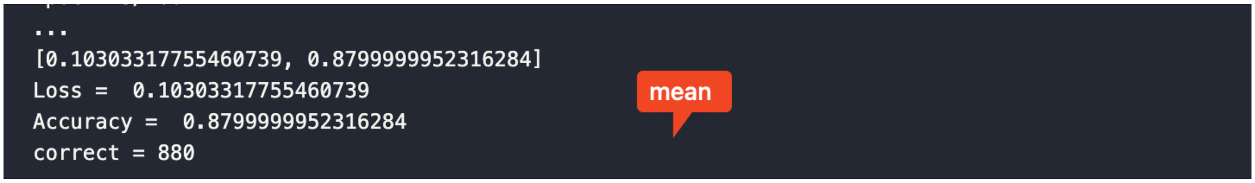
We can start by looking at the model without any scaling.



```
Loss =  0.10644198954105377
Accuracy =  0.8690000176429749
correct = 869
```

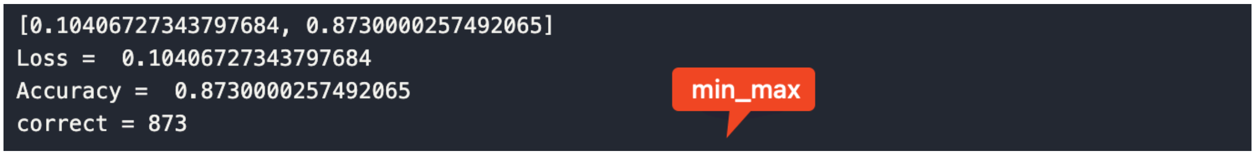
This is still fairly high! But let’s compare with other scaling algorithms to get a better gauge on how accurate our model can be.

Mean Normalization:



```
...
[0.10303317755460739, 0.8799999952316284]
Loss =  0.10303317755460739
Accuracy =  0.8799999952316284
correct = 880
```

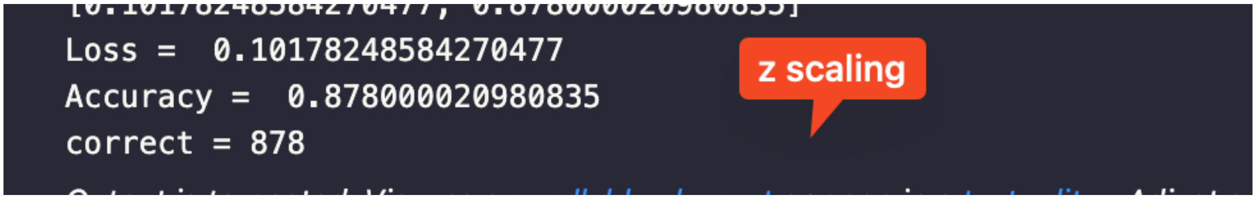
Min Max :



```
[0.10406727343797684, 0.8730000257492065]
Loss =  0.10406727343797684
Accuracy =  0.8730000257492065
correct = 873
```

Scaling *should* have had much more of a positive impact on our data set. Alas, there was a fairly negligible difference in the scaled accuracies, and the unscaled. My theory is that Keras does some scaling automatically; maybe behind the scenes Keras recognizes the gap and does it automatically.

Here is what happens when we apply z score scaling to the entire dataframe:



```
[0.10178248584270477, 0.878000020980835]
Loss =  0.10178248584270477
Accuracy =  0.878000020980835
correct = 878
```

Z Scaling had a positive effect on our model, but again it feels pretty irrelevant considering how well the untouched data set did. However, Mean Normalization still had the largest positive impact, and thus was the most effective for this data set.

Some things I would do to improve this model is to understand the regression part more. I think one of the largest issues I had was with how long my model took to compile, taking well over 6 minutes to go through 1000 iterations. It was quite annoying to make small tweaks and have to wait so long to see changes, so that is eventually why I did move over to Keras. However, with that accuracy, that is a pretty acceptable number. 88% is a great number and I would be confident working from there. One thing I would improve if I had the time is definitely something in cross-validation. It not only could help the data set, but it would also get a better understanding of machine learning.

Overall though, I'm satisfied with this project. I recognize some of my shortcomings, namely in how I tried to do the actual regression part myself, but machine learning is becoming less and less of a nebulous idea.