Direct TV Shopping/Livestream

Team 4 - Luke McConnell, Chris Dunn, Price McGinnis

Data Import and Set-Up

Downloaded the data locally to our computers

Took a random sample of 1000 entries

- Any magnitude above this took an extremely long time
- Jupyter Kernals froze up… alot

Defined our “y” and “x” variables

Split the data 80/20 training/tuning

Set up our function to iterate through different tuning parameters

The Gameplan

1. Defined a function where all arguments were our tuning parameters
2. Function would store our results from different tuning combinations
3. Created a for loop to try a number of different tuning parameters
4. From the stored results, we pulled the combination of parameters that provided the lowest loss

Tuning Parameters

Batch Size

Hidden Nodes

Hidden Layers

Activation Function

Optimizer Function

Epochs

360
Combinations

```
try_batch_size = list([1, 10, 50, 100]) # add back in 1000
try_hidden_node = list([2, 5])
try_hidden_layers = list([1, 3, 5]) #can only handle 1 through 5
try_activation_func = ['relu', 'sigmoid']
try_optimizer_func = ['SGD', 'Adam']
try_epoch_size = list([10, 20, 30])
```

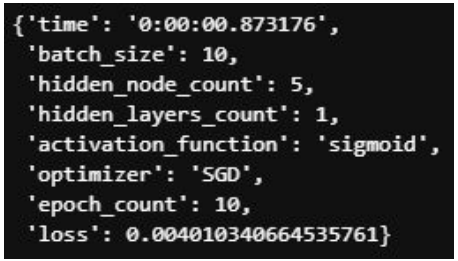
Insights and Findings

Results were somewhat erratic

- Some results came as expected (Adam, Relu, high epochs, etc.)
- Other results went contra to expectations (SGD, sigmoid, low batch size, low epochs)

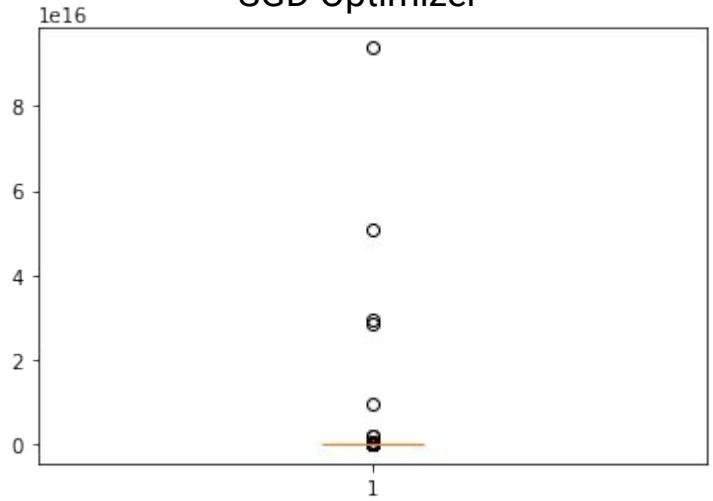
Overall, loss remained relatively low and consistent, however

Issues with JupyterLab Kernals freezing before completion

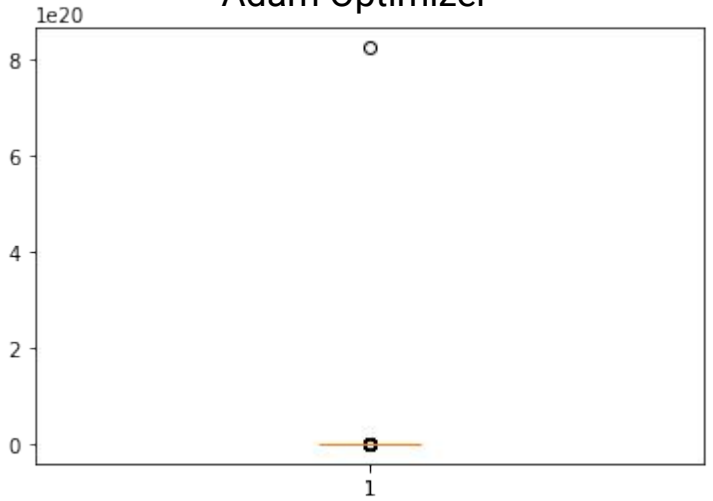


Tuning Observations

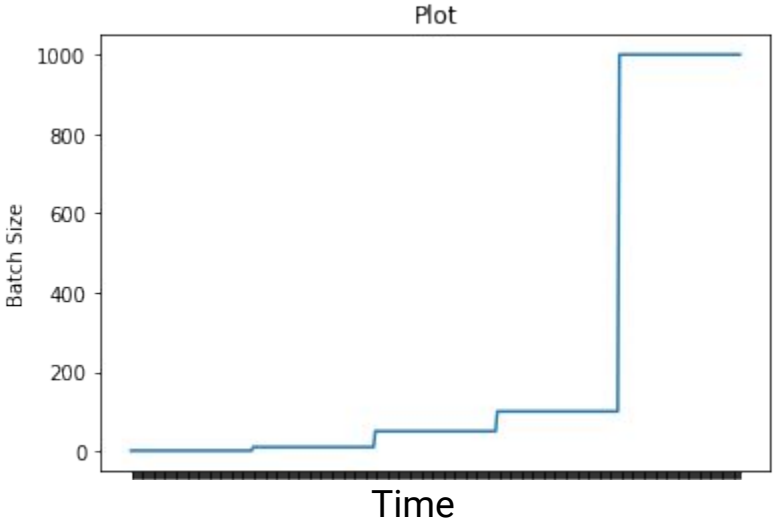
SGD Optimizer



Adam Optimizer



Tuning Observations



Future Recommendations

Sparse the inputs for our categorical variables

Given infinite time, try all combinations on all observations

Increase the number of hidden nodes beyond 5

Try more optimizers and activation functions

Experiment more with learning rate