HACKER NEWS UPVOTE PREDICTOR

Gradient Gigglers

WORKFLOW

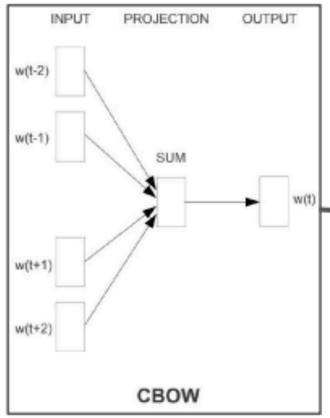
ARCHITECTURE

Overall project directions - Approach 3: Target single score

Step 1:
Input each feat data

Step 0:
Train on
Wikipedia data

Title



User

A difference in time of day users created and a time they posted

Domain

Embeddings

Feat 1

Feat 2

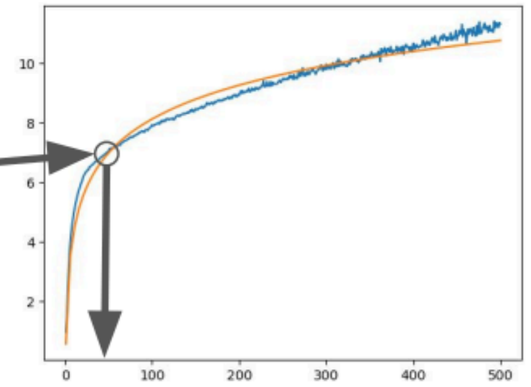
Feat 3

Final
Normalized
values

7

Step 2:
Transform final single val to score

Final model

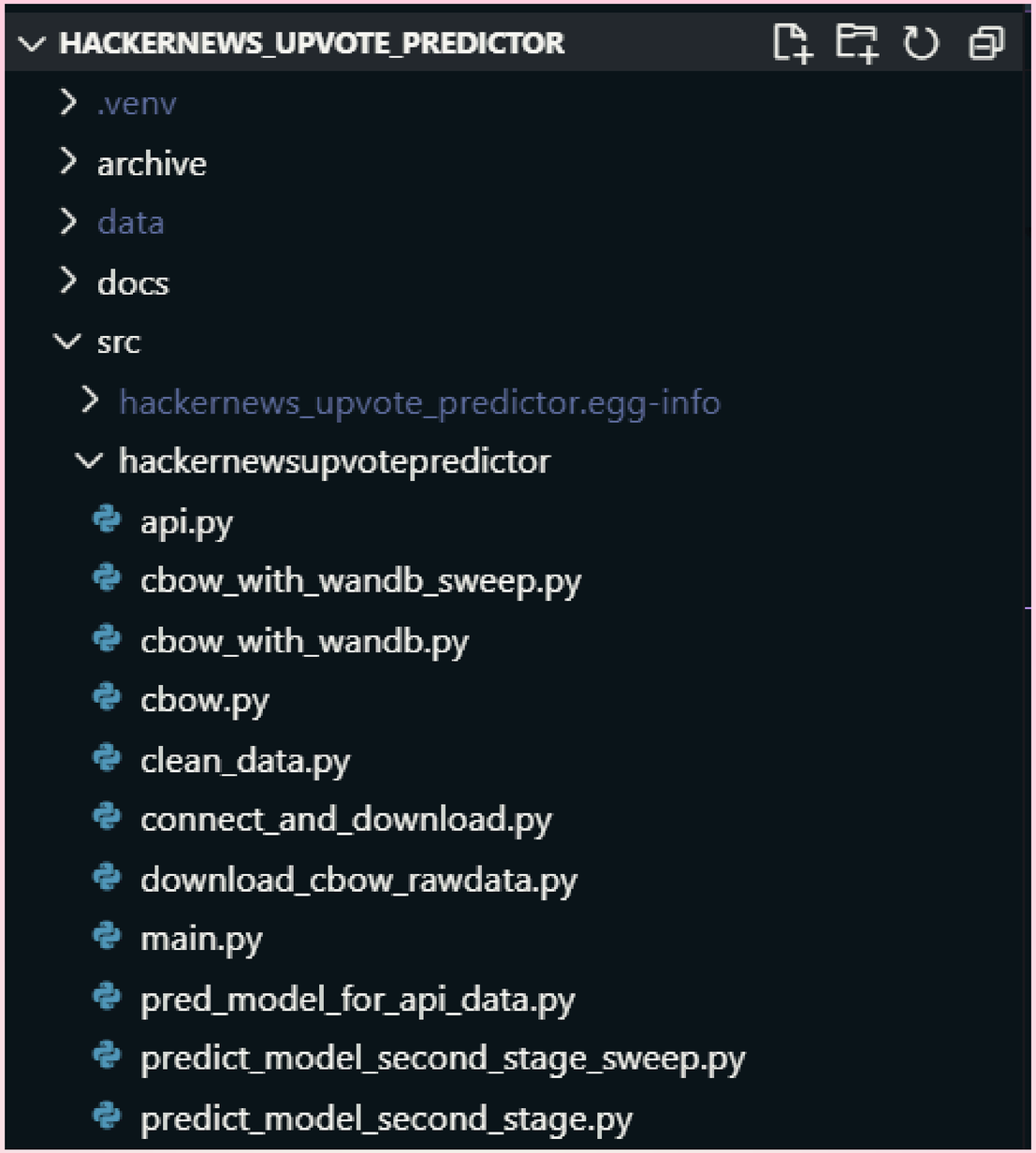


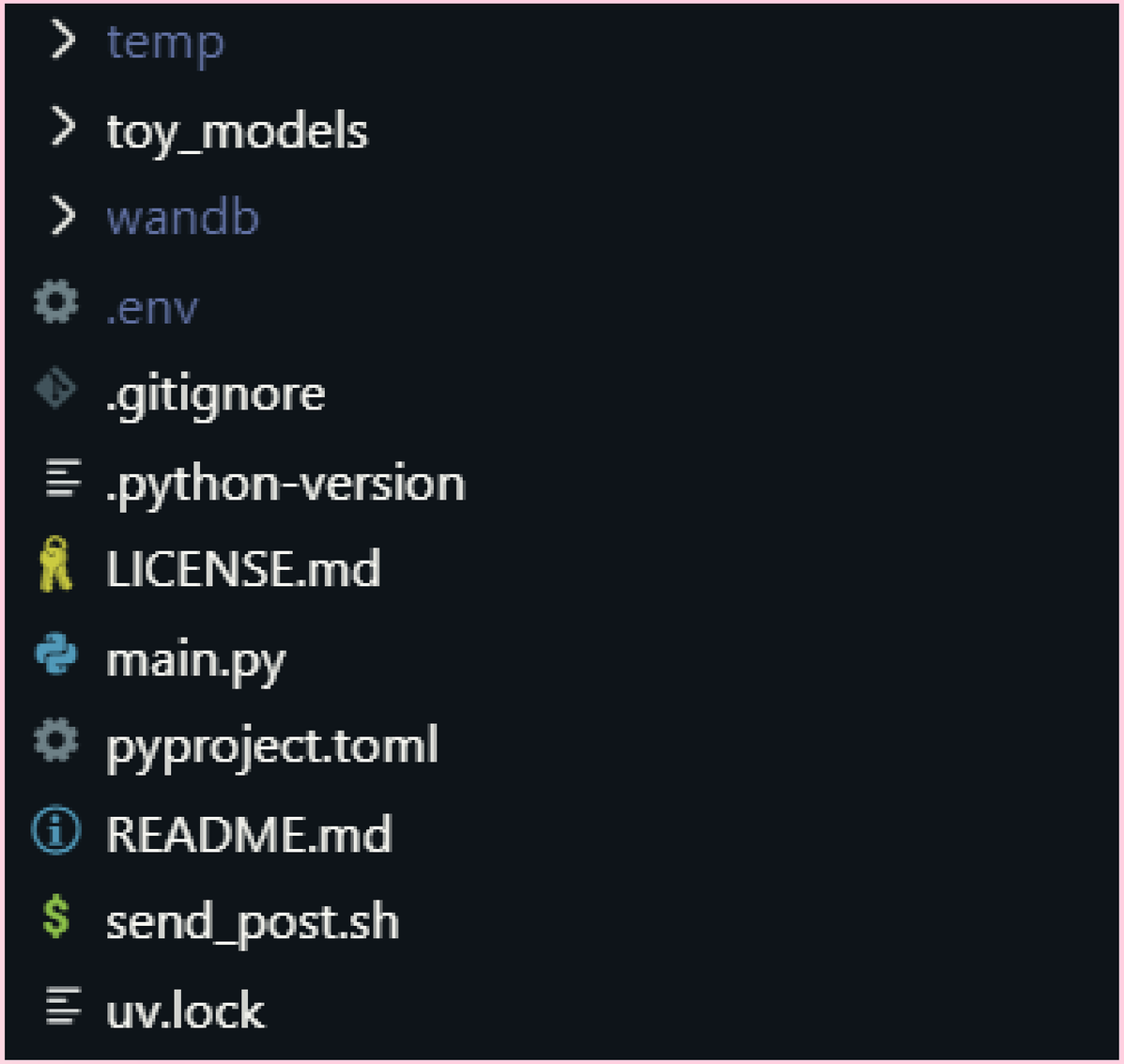
Predict
score

40

*note score > 500, which is 0.28% of data over 4M data is not in this pattern, so a model will have poorer prediction (which is around 11k top scores range 501-6k)

PROJECT STRUCTURE





Sequence of running

Ensure initial python set up has been done

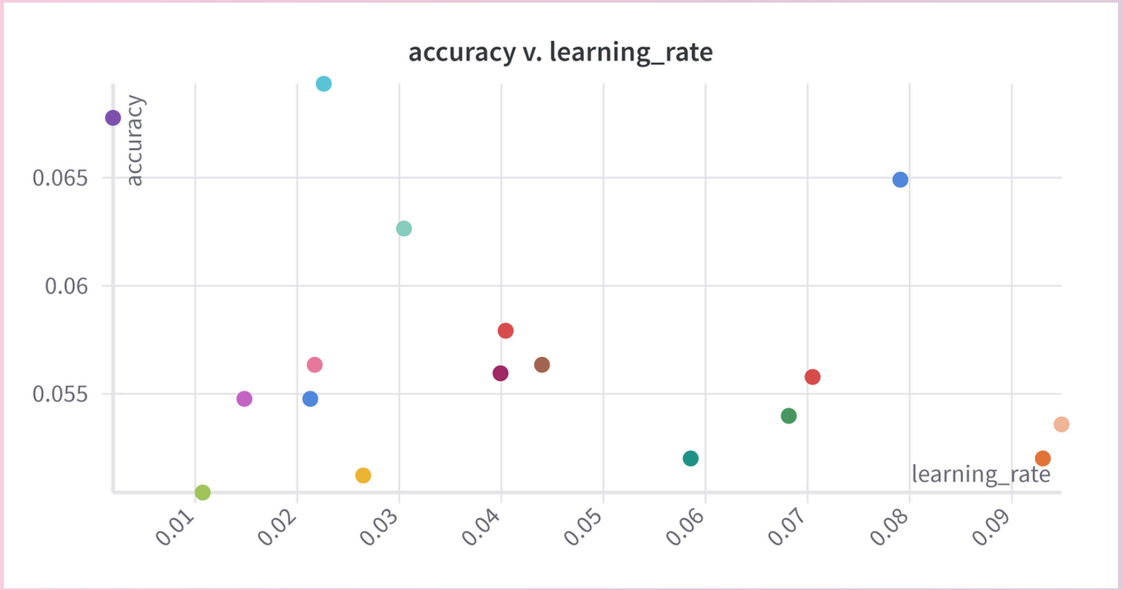
0. Add a data and temp folder to root
1. Run `connect_and_download.py` file - this will connect to the database, and download the hackernews items (joined with user data) into a parquet file
2. Run `download_cbow_rawdata.py` that will download the wikipedia text data to data/text8
3. Run `cbow.py` to run the CBOW code
4. Run `clean_data.py` which will read the above parquet file, and then extract the feature data of how many days the user has existed, title data, as well as the target data (the upvote score)
5. Run `predict_model_second_stage.py` that will train another model on the feature of how many days since the user has been created + title to give final output (score prediction)
- This will output the loss functions for each run

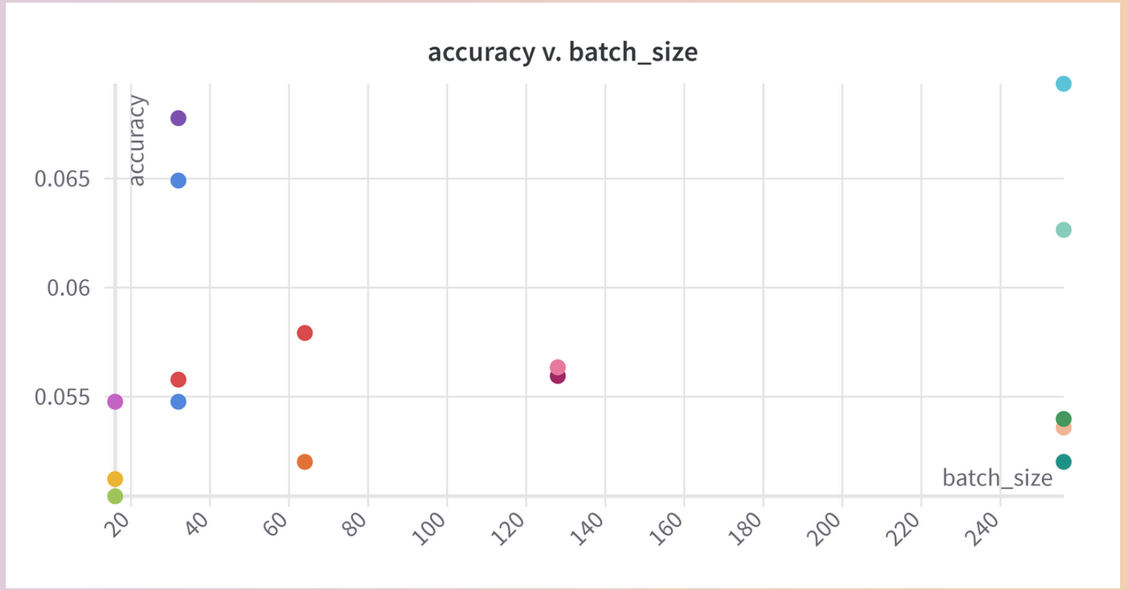
How to run the api

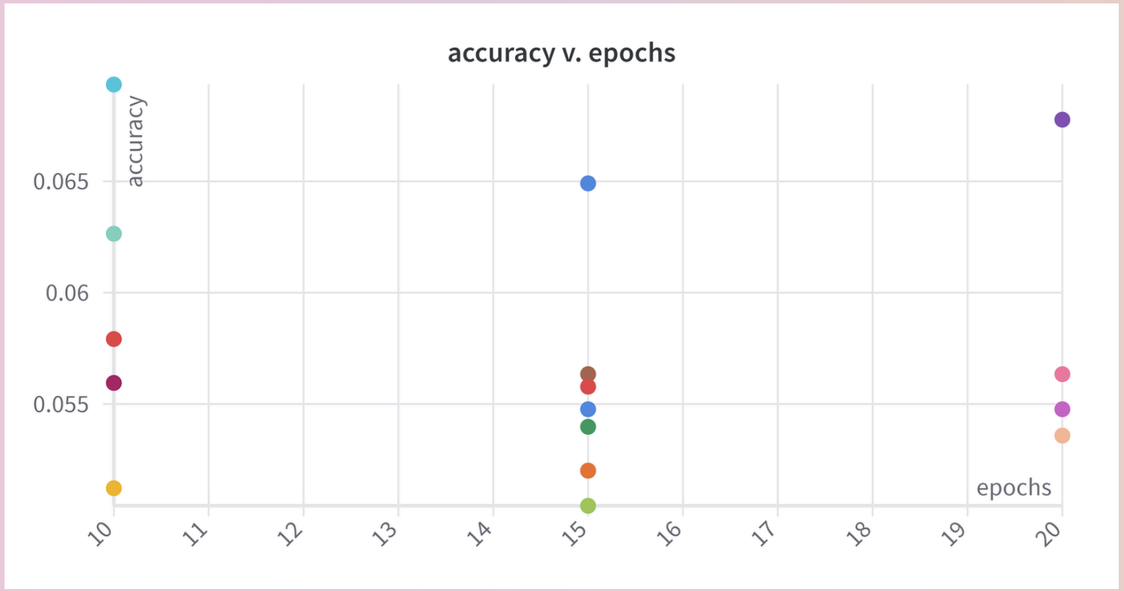
1. Run `uvicorn src.hackernewsupvotepredictor.api:app --reload` to run the API server
- You should be able to see in http://localhost:8000/healthcheck to see it up and running
2. Then edit the send_post.sh bash script with your custom values
3. Run send_post.sh to see the predicted upvote from the model

Time to take action

RESULTS - CBOW

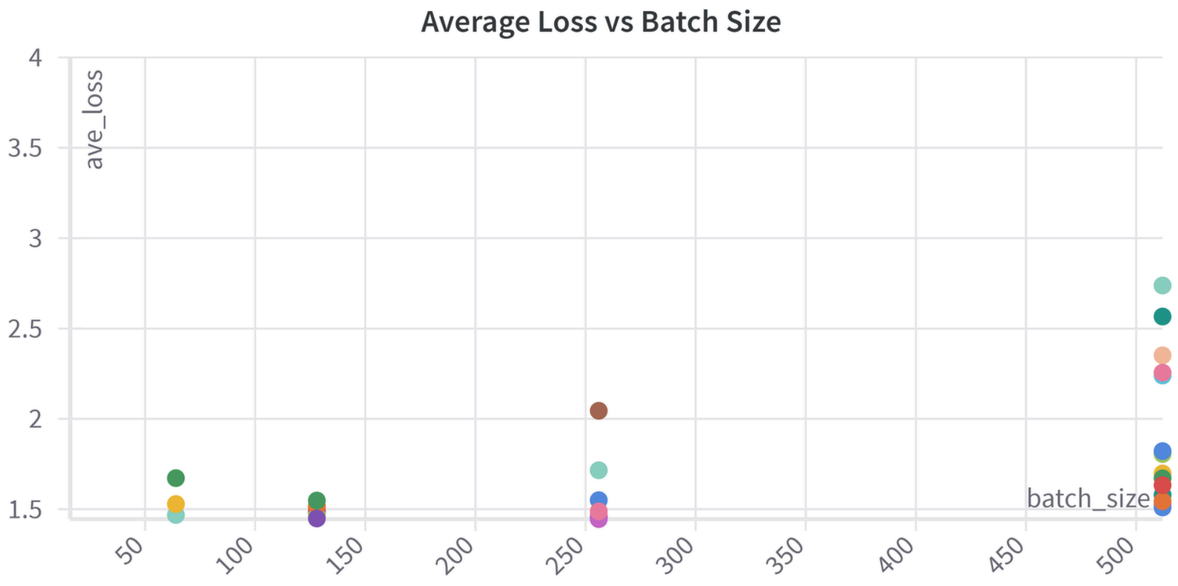


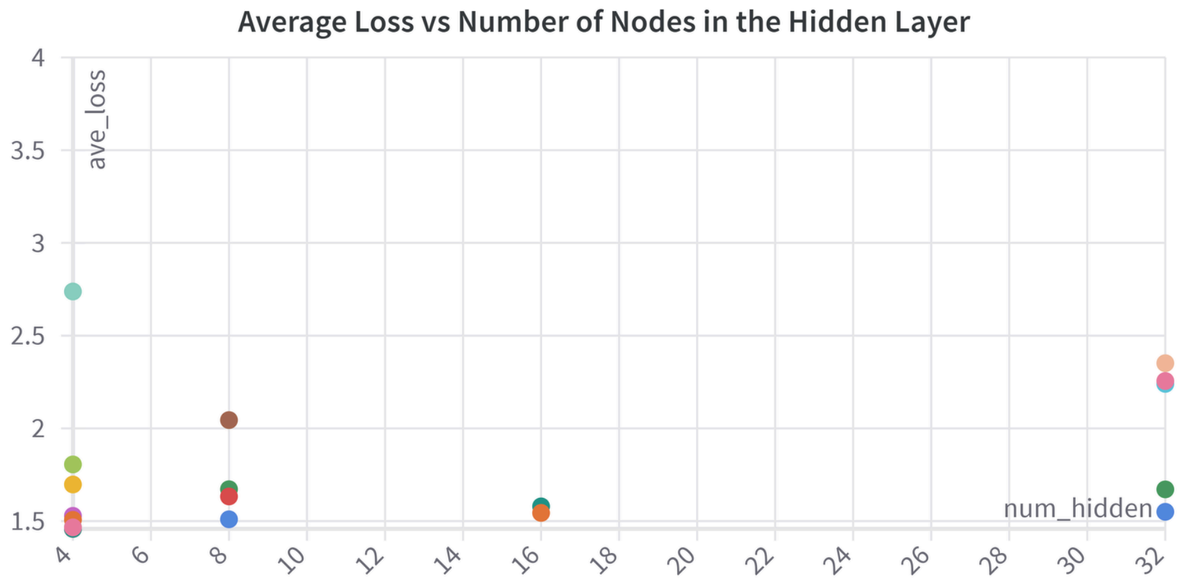


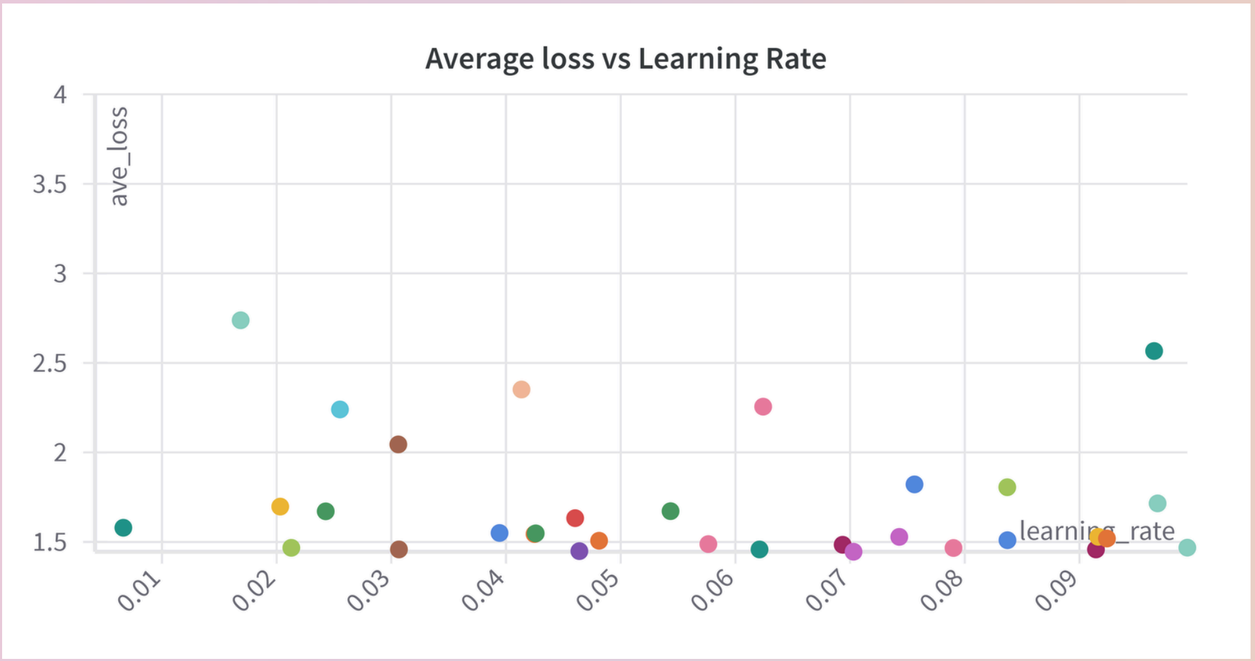


RESULTS - PREDICTION MODEL

```
Epoch 1, Loss: 730.2390
Epoch 51, Loss: 1.5738
Epoch 101, Loss: 2.0703
Epoch 151, Loss: 1.8469
Epoch 201, Loss: 1.7715
```


Average Loss vs Batch Size
ave_loss
4
3.5
3
2.5
2
1.5
batch_size
50
100
150
200
250
300
350
400
450
500


Average Loss vs Number of Nodes in the Hidden Layer
ave_loss
4
3.5
3
2.5
2
1.5
num_hidden
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32


Average loss vs Learning Rate
ave_loss
4
3.5
3
2.5
2
1.5
learning_rate
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09

When you realize that we are closer to 5 PM, than we are to 10 AM