Deloitte Data Challenge

Estimate binding affinity

Alexander Muratov

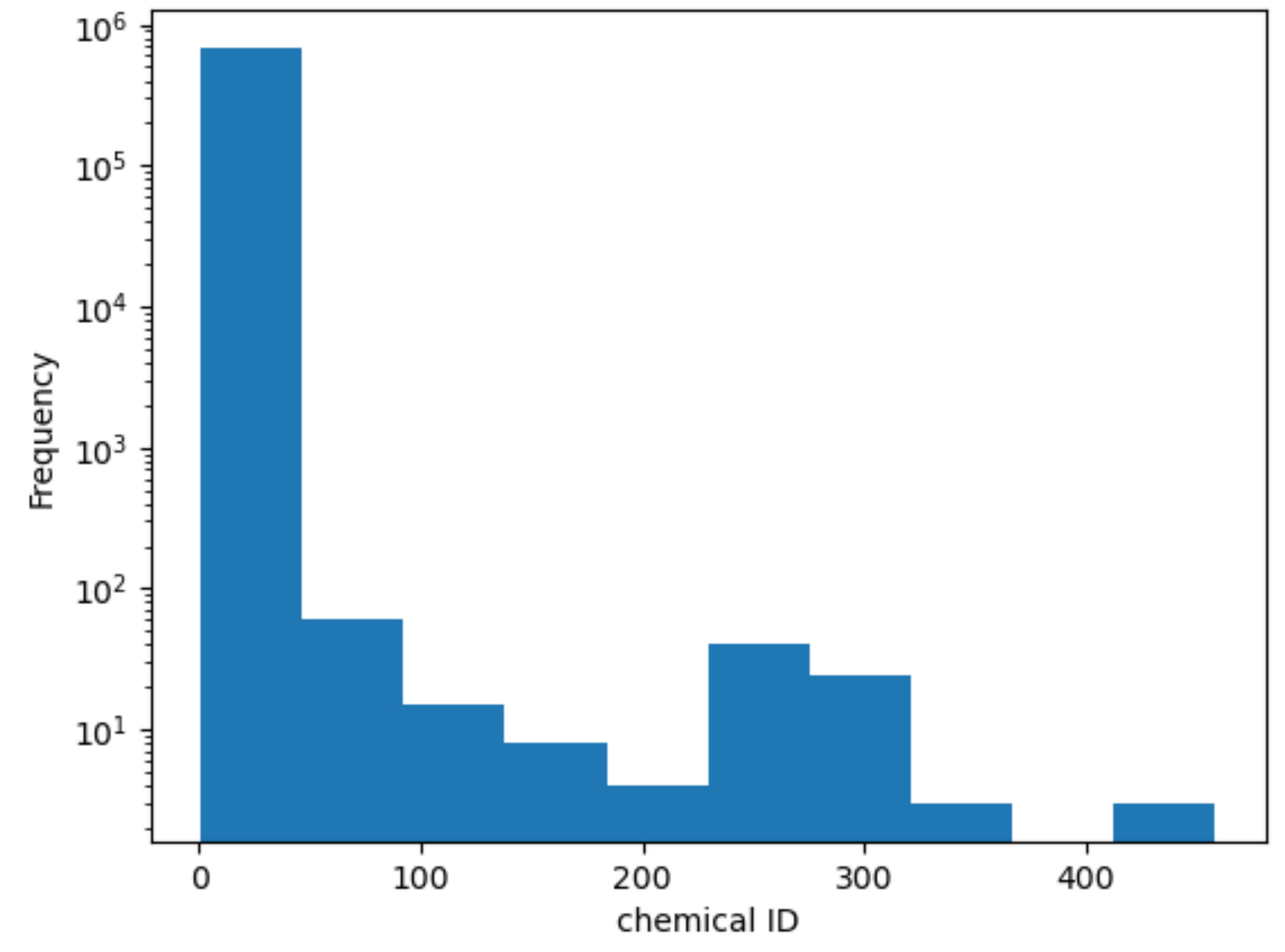
How to run my project

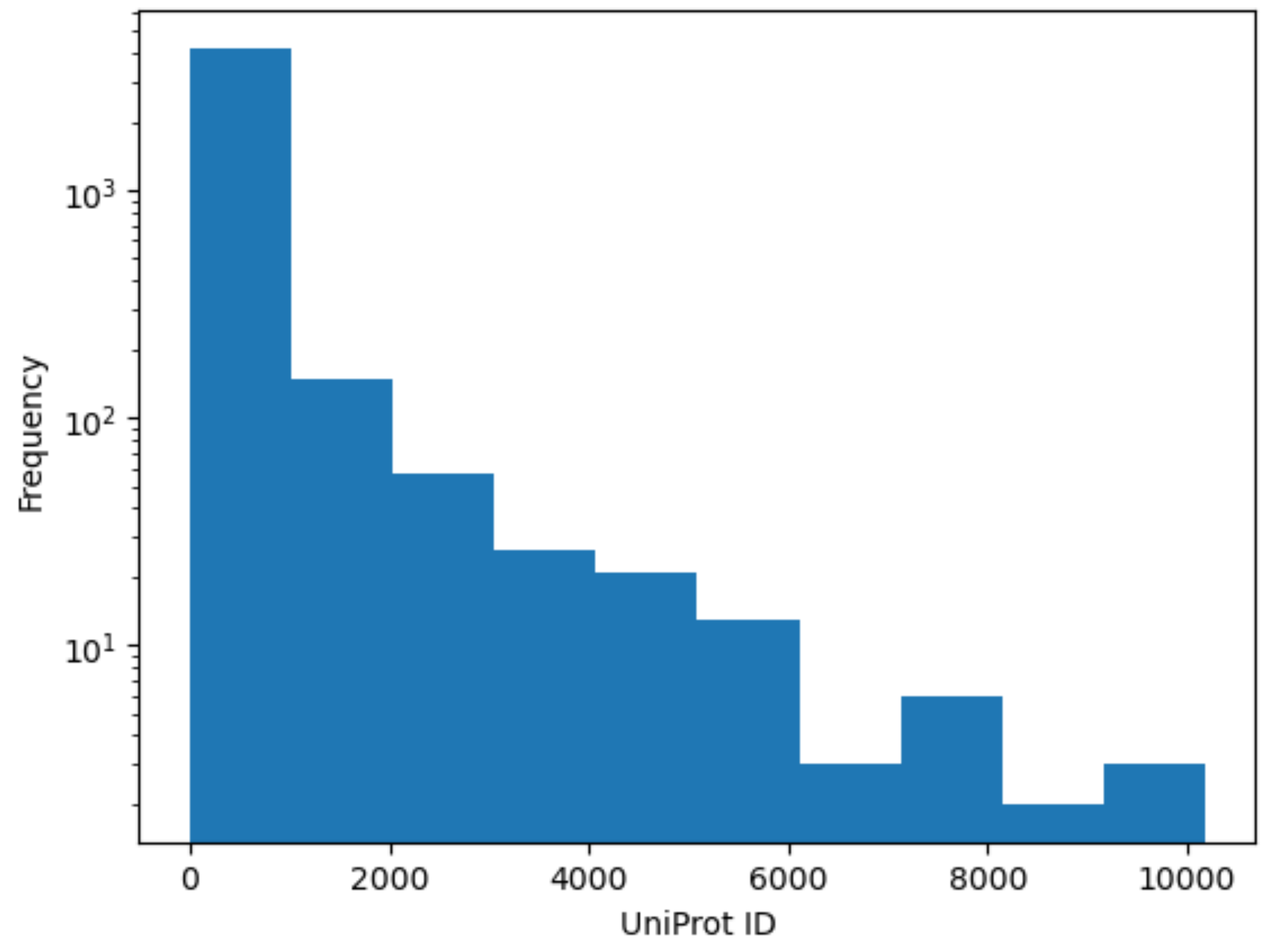
- I include Full Workflow.ipynb where I go through the data and train the model and Evaluation Workflow.ipynb , which is where a different dataset can be brought through the workflow.
- For the purpose of demonstration, I just did a small subsample of the original dataset as a test set
- I will include my conda env file. The main packages I used other than native python were Scikit-learn, pandas, XGBoost, and BioPython. Should be straightforward to install. I have python 3.12 running in my env.

Data Exploration

Basic facts

- 1,134,638 entries
- 683,413 unique pubchem IDs
- 4,480 unique UniProt IDs
- 39,611 rows with NA values - I choose to drop these. Not too many and have no idea what to do with them.
- Some repeat measurements for same UniProt ID and pubchem ID - I choose to average together these measurements to avoid train/test contamination

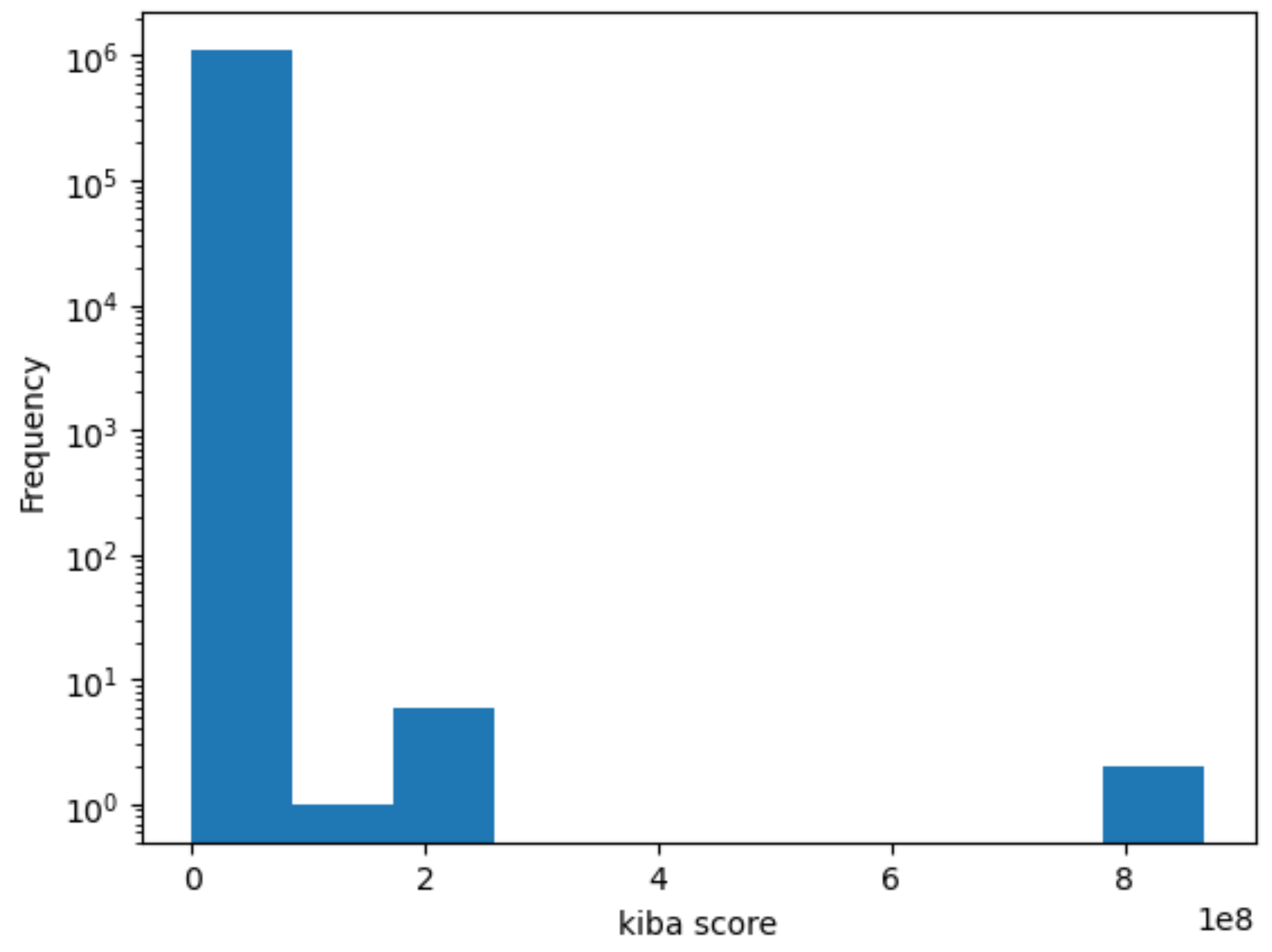


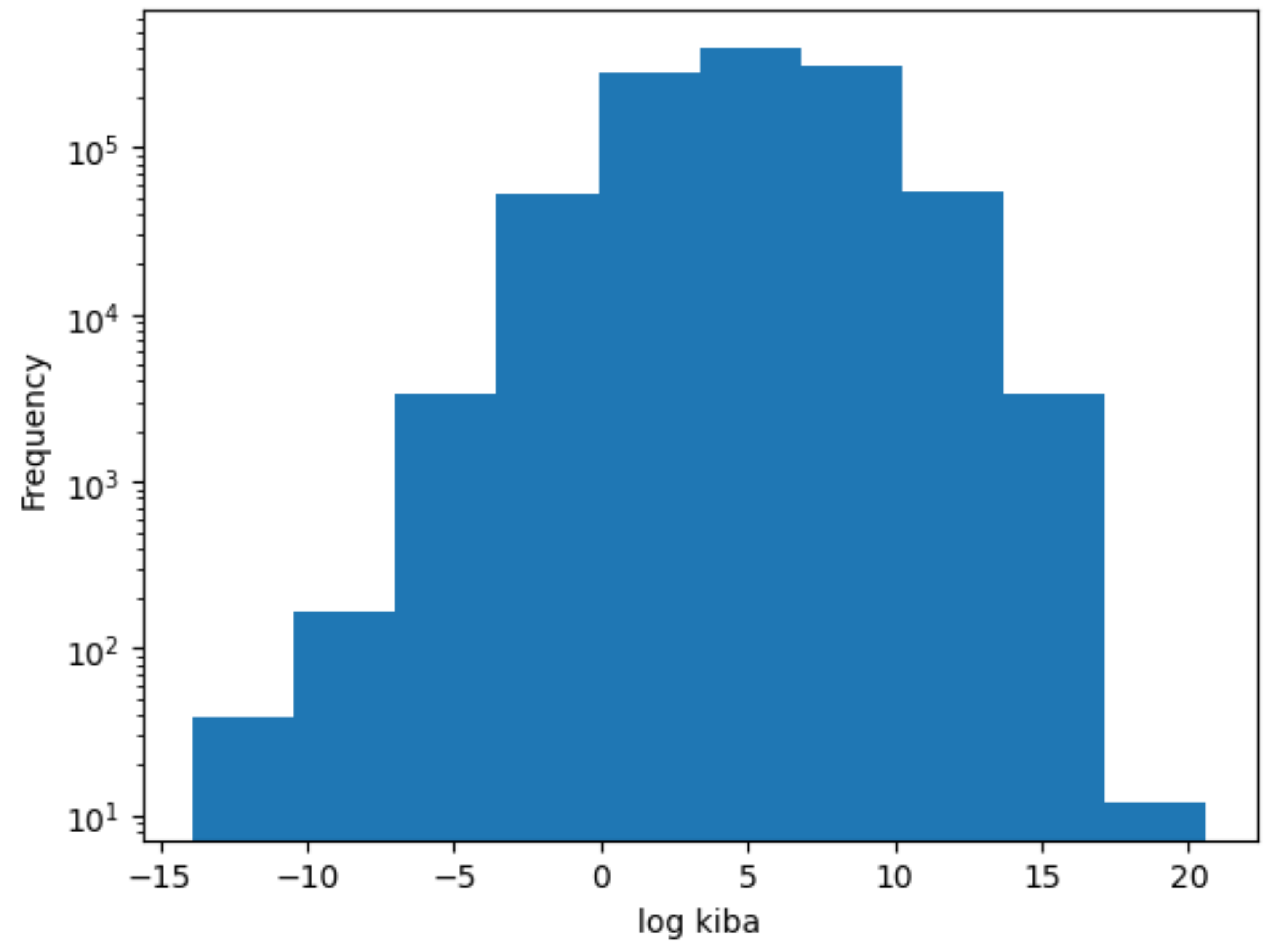


Data Exploration 2

Kiba score

- KIBA score itself seems to be fairly unevenly distributed.
- Therefore, I am choosing to take (natural) log of KIBA score for this exercise - a much more beautiful distribution





External Data Acquisition

From PubChem

- I devised a query system to retrieve all quantitative “properties” for batches of 400 pubchem IDs at a time. The properties were taken from the API documentation and included things like molecular weight, number of heavy atoms, etc.
    - Also includes how many patents and papers the molecule was in - this will be important later :)
    - Total of 35 features per chemical id
- Querying in batches was necessary because it crashed if I tried to query all 600,000+ chem IDs at once. I apologize if this portion of the code takes a while to run. Could not come up with a faster way to query in my limited time.
- I queried for the Canonical SMILE as well - but ultimately I just ran out of time to find a good way to use this. I tried the RDKit package, but it was too slow on my 2019 MacBook Pro.

External Data Acquisition

From UniProt

- Similarly had to query in batches of 100, but at least there were only 4400 or so unique protein IDs.
- Again apologize if this takes a while - since proteins are more widely represented, even in my test set, there are plenty of them.
- I query for the sequence, annotation count, and a list of "extra attributes" with counts. I wind up with about 40 features that come from these "extra attributes"

Additional Features from Protein Seq

Using Biopython

- I use the ProteinAnalysis module from BioPython to calculate additional properties about the protein from the sequence
- This feels fairly old-fashioned in a world where sequences and structures can be fed directly into neural networks, but I have limited time and an old laptop.
- Also calculate fraction of each type of amino acid.
- 28 features derived this way

Final Dataset

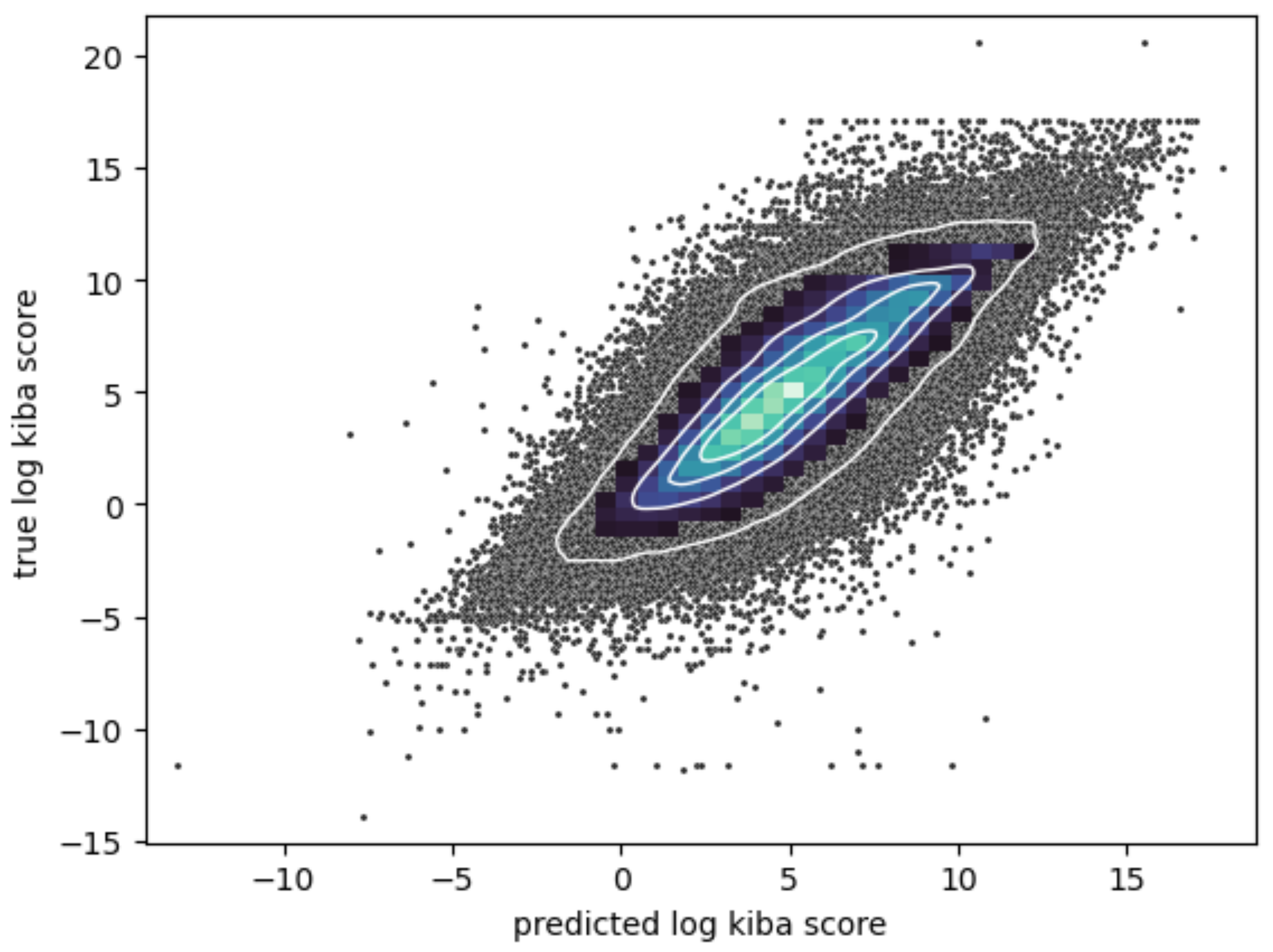
- 103 features
- 1071302 samples
- In the interest of time, I will not be doing k-fold cross validation and instead just use a simple 67/33 train/test split.
- Maybe before submitting the assignment I should re-train with all of the training data, but I trust that it should be robust enough :)

Model Selection

- I tried:
  - Linear Regression (with and without feature normalization).
    - Maxed out at R2 = 0.1 or so
  - NLP regression (dense neural nets)
    - Maxed out at R2 = 0.3 for both training and test
  - Random Forest Regression
    - With unlimited depth can get to R2 = 0.9 for train set, R2=0.7 for test set
    - But with reasonable depth limits, maxes out closer to R2=0.5
  - **XGBoost Regression**
    - **With Depth=15 can get similar results to Random Forest Regressor, but is much faster to train and more reliable.**

Best fit model

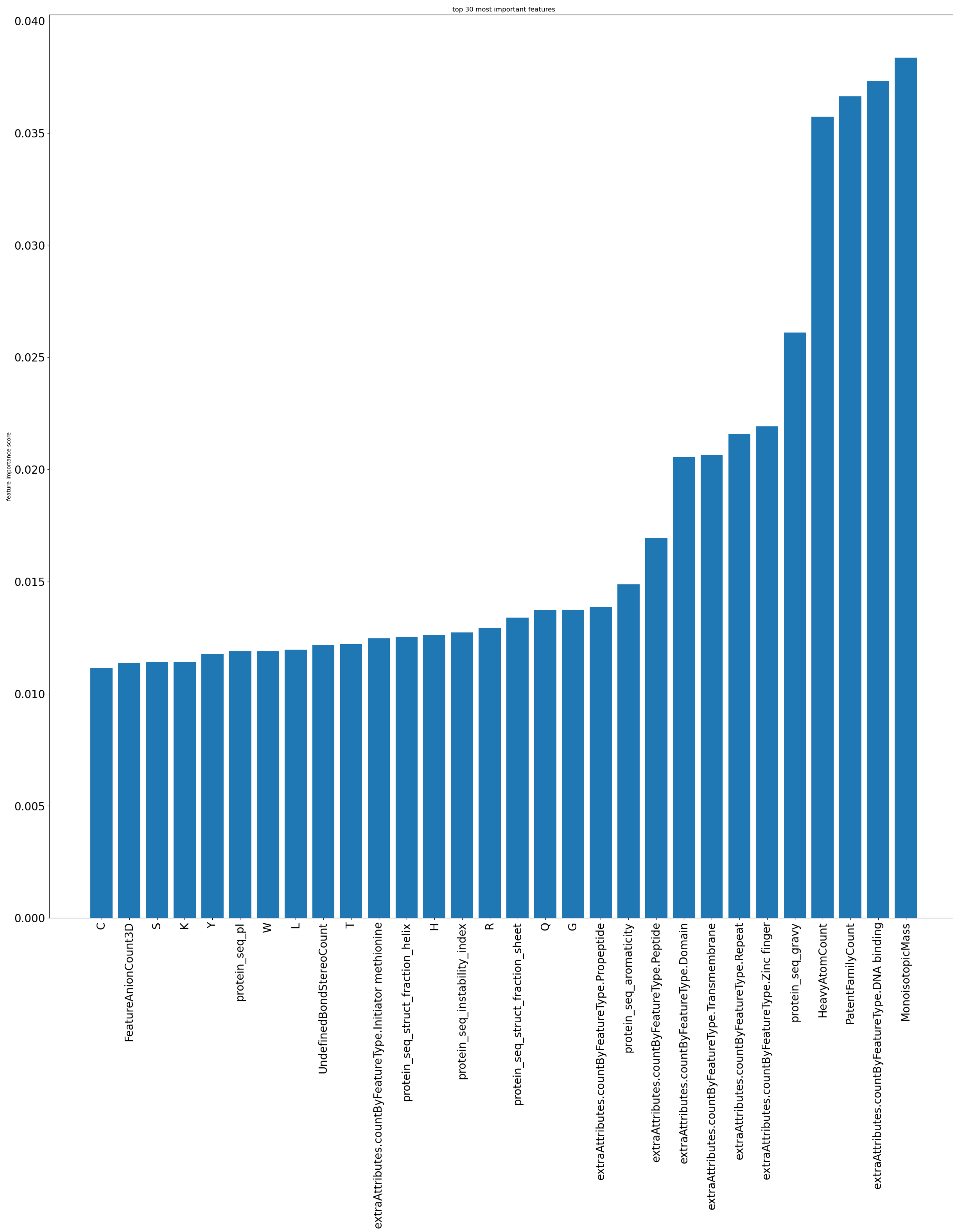
- XGBoost Regression model with max_depth = 15
- R2 = 0.678 (on validation set)
- MSE = 3.54
- It's a bit messy, but it looks pretty smooth, and especially looks so much better than other types of models I tried to train!



Feature importance

Top 30 most important features plotted

- Top 5 most important features:
    - MonoisotopicMass
    - extraAttributes.countByFeatureType.DNA binding
    - PatentFamilyCount
    - HeavyAtomCount
    - protein_seq_gravy



Thoughts about Feature Importance

And other ponderances.

- Different iterations of (tree-based) models I've fit generally have differences in the feature importance rankings. This suggests some covariance in the data, maybe some instability, or maybe I am not unlocking the full potential of the data by using more estimators.
- Max_depth = 15 is quite large — the model is definitely overfitting. However, performance on the validation set is still improving with max_depth levels. If I had more time I'd explore other ways to prevent overfitting while perhaps still leveraging as many things as I could
- There is an interesting mix of features that derive from chemical properties of the small molecule, annotated attributes of the protein, and some sequence-derived properties of the protein.
- One cheeky thing - the PatentFamilyCount feature importance is a bit of a cheat - it seems that by knowing how many patents the small molecule has been involved in allows you to predict whether it will bind or not :). Other times, I've seen the Literature Count be the #1 most predictive feature. Obviously does not seem conducive to de-novo discovery of novel interactions, but it does make sense in a data science way

Conclusion

- I think my model did a decent job, at least in log space, of predicting binding affinity values
- I was a bit surprised at how much better tree models were in this regime, but I guess it is possible that there are many non-linear effects that are difficult to model without some specialization
- I did not use much deep learning for this project, but recognize that there are lots of places where I could have - supplementary ideas follow!

Things I did not get to try: Protein Seq

I had better ideas for protein sequence data,
But I just didn’t quite have the resources to execute on it

- Kmer embedding feature for protein sequence
    - For k=3, I get feature vectors of length 8000. Just too much for my laptop to handle
- CNN approach with one-hot embedding for protein sequence
- ProtBERT embeddings from protein data
    - I was getting size 1024 per protein sequence. This seemed promising, but it was just taking too long to calculate them all on my laptop.

Things I did not get to try: Chemicals

Similarly, just kinda ran out of time to work on this

- Using RDkit, I can extract Molecular Descriptors or Morgan Fingerprint data
- I got all this set up and was getting ready to try it out, but again it was just taking too long to run on my old laptop.

Other Ideas: Recommender system!

- There was a way to solve the problem without even querying any data - if I just made a matrix of all possible chem IDs vs protein IDs and their KIBA score, and used a matrix factorization approach to fill in missing values
- This could have been fun, but then I would have to find a custom workflow to add in “new users” and that’s a little straightforward