Protein Kd Prediction

Cody Schiffer

Presentation Content

1. A review of the exercise objectives and the material accompanying the presentation
2. Data wrangling and first insights
3. Defined hypotheses and proposed experiments
4. Experimental Results and Takeaways
5. Proposed next steps and future improvements
6. Experimental constraints and impact

Objectives and Materials Prepared

Objectives:

1. Create a regressor to predict the Kd value for a pair of proteins. Sequence **a** and Sequence α
    - At least 1 deep learning approach must be used.
    - Demonstrate solid understanding of machine learning and deep learning approaches, explaining why model selections were made
    - Demonstrate satisfactory Machine Learning Pipeline (MLP) development skills and software engineering best practices
2. Provide predictions on the hold out set.
3. Concisely summarize results and prepare codebase for review

Material Prepared:

1. A_alpha_bio_data_exploration.ipynb – The jupyter notebook summarizing data wrangling and model runs
2. Model_architectures.py – A supporting python file providing easier access and review of existing model architectures
3. Modelling_utilities.py – A supporting python file that provides utilities for data prep, metrics calculations, and model instantiation for cleaning modelling work in the jupyter notebook
4. Results_exploration.ipynb – A jupyter notebook responsible for showing best performance per model.
5. Plotting_assistance.py – A supporting python file for plotting.
6. ReadME_CAS.MD – Applicant's prepared README.md
7. Model_results.tar.gz – collected model results stored as .pkl files
8. Requirements_a_alpha_experiments.txt – requirements necessary to run the modelling experiments, plotting for model evaluation, and results processing

Data Wrangling and First Insights

Goal and Purpose:

1. Get familiar with the bio-data used at A Alpha
2. Ask critical questions
3. Identify potential data skews, data errors (like missing values)
4. Develop potential new features for ML work
5. Create plan for developing regressor models

Critical Questions

1. Are both protein sequences useful as variables?
2. Are there missing Kd measurements? If so, how can we fix these?
3. Are there outlier Kd measurements? If so, how should we handle these?
4. Are there length trends amongst the mutated sequences?
5. What is the meaning of the q-value? Would it impact the selection of our data?
6. Are there significant distribution trends amongst the amino acids for the protein sequences of interest?
7. Are all Kd predictions of sufficient quality?
8. Are there correlations within the data?
9. Are all Amino Acids known?
10. What does the ‘window’ mean?

Data Wrangling and First Insights

Q: Are both protein sequences useful as variables?

- *All sequence-alpha sequences are the same. (These describe PD-1).*
- *Accordingly sequence-alpha sequences are **not helpful** as features for predicting Kd.*

Q: Are there missing Kd measurements? If so, how can we fix these?

- *There are **691 missing values** in the Kd column There are **691 missing values** in the Kd_upper_bound column*
- *This is slightly problematic as we trying to predict this. We can't just eliminate these data points as Adrian said they are vali so we have to use another approach*
  - *1) We can exclude them (the easiest option - why would we try to predict a value we don't have data for?). However, Adrian mentioned that all data points are valid so we need to keep them.*
  - *2) We can try to re-create it using the lower bound and the upper bound. If the midpoint is still null we'll just take the lower bound.*
- *We'll go with approach 2. It provides us a method to include data we know should be included. Given the meaning of bounds we can assume that they are entirely possible measurements for the sequence a and sequence alpha interaction.*
- *There are approximately 105 sequence descriptions that have more than 1 sequence. They represent distinct Kd value possibilities so if the same sequence had 2 different Kd values, I took the maximum!*

Q: Are there outlier Kd measurements? If so, how should we handle these?

- *Indeed there are! If we plot the Kd values on a bar plot we can see that we have a clear long low tail. The average is around 3.3, with lower whisker at 1.42 and upper whisker at 4.56. (Fig 1, Fig 2)*
- *These results indicate that there are a significant minority of sequence a – sequence alpha predictions that are significantly stronger. The lack of an upper tail implies tha (relatively) It is very rare the sequences have particularly poor binding affinity.*
- *The long low tail here suggests that our model may be at risk of overfitting and generalizability challenges. We need to use cross-validation and carefully examine epoch level performance to make sure that we don't overfit. Furthermore, we should create the capability to split the data taking the Kd values into account for a representative sample.*

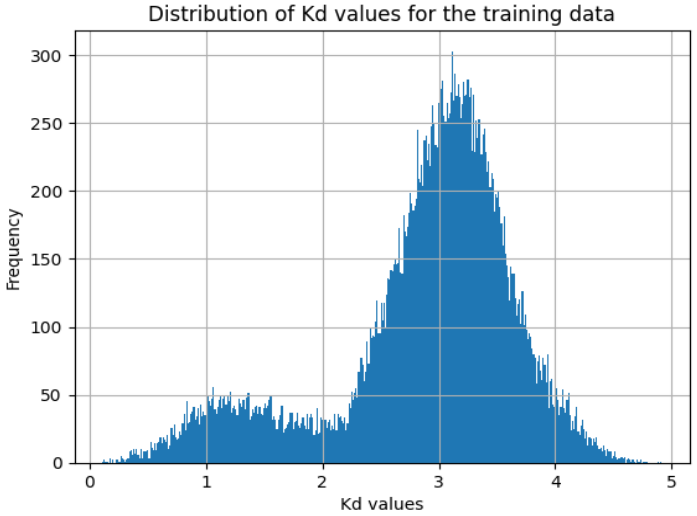


Fig 1. Plotting the distribution of Kd values found within our dataset.

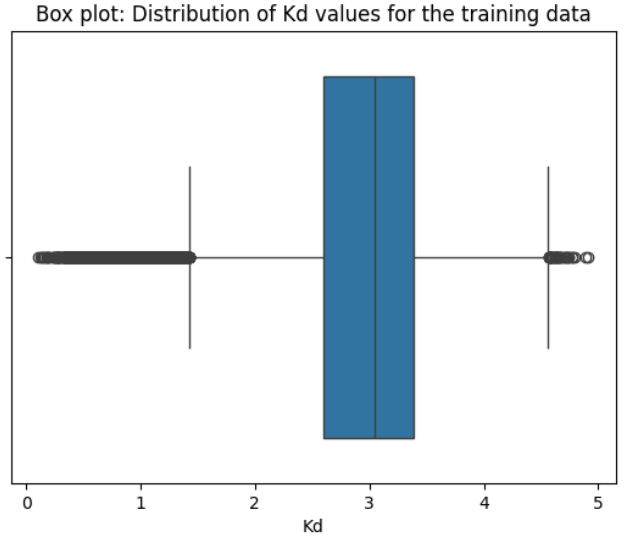


Fig 2. A boxplot of the Kd values. The lower whisker is at: 1.42. The high whisker is 4.56.

Data Wrangling and First Insights

Q: Are there length trends amongst the mutated sequences?

- *Well, that's interesting…And, not what I initially expected! At first, I was concerned that mutations within sequence_a (the scFv variant of Pembrolizumab) could change the overall length of the window. However, that's not the case. This is good news! This means that we don't have to worry about padding our sequences to the same length.*

Q: What is the meaning of the q-value? Would it impact the selection of our data?

- *The q-value is comparable to the p-value. In essence it describes a threshold at which we can refute the null hypothesis for our measurement. Below a certain q-value - the results are significant. Above that threshold, and we can't refute the null hypothesis.*
- *In my opinion it is prudent to focus our initial data selection and training on those results that are statistically significant. We'll choose a relatively common q-value (0.05) to establish a null hypothesis and establish our first results using datapoints that falllower than that. **
- **I'm making an assumption here. Adrian said that all data points are valid. In my opinion, while high q-value results are indeed valid, they may not be suitable for training. The point of A-alpha Bio's research goal here is to design a method to predict realistic Kd values. In my opinion, results that may not pass the null hypothesis for training would be counter productive for this purpose. Furthermore, this speeds up training ☺.*

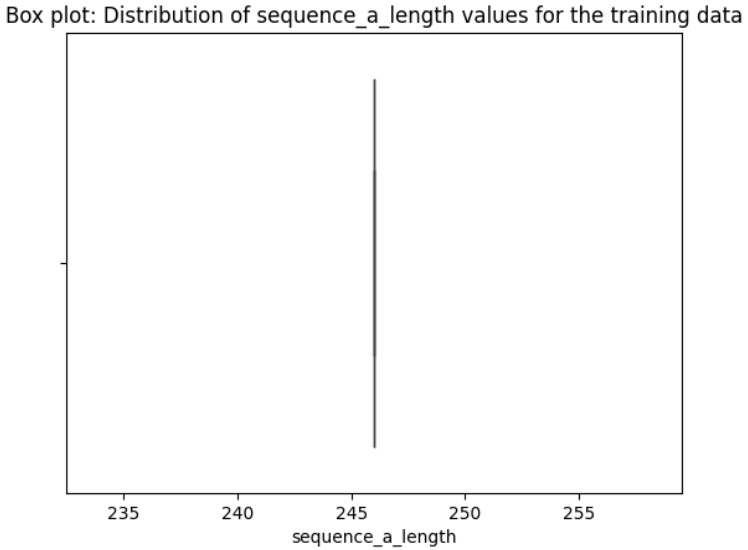


Fig 3. A boxplot of the sequence_a lengths. There is no lower whisker. There is no upper whisker.

Data Wrangling and First Insights

Q: Are there significant distribution trends amongst the amino acids within the protein sequences of interest?

- *Yes, but the frequencies of all amino acids across all sequences isn't that helpful. However, we can create new features out of this. (Fig 4.)*
- *We can calculate how frequently each amino acid (AA) appears in a variant (per row) and add that as a new column. So for each sequence_a variant, we'll get an additional tensor describing the how frequently each AA is found in the variant. (Fig 5)*
- *We can experiment with different scaling methods (MinMax vs StandardScaler) to see if that impacts model performance.*

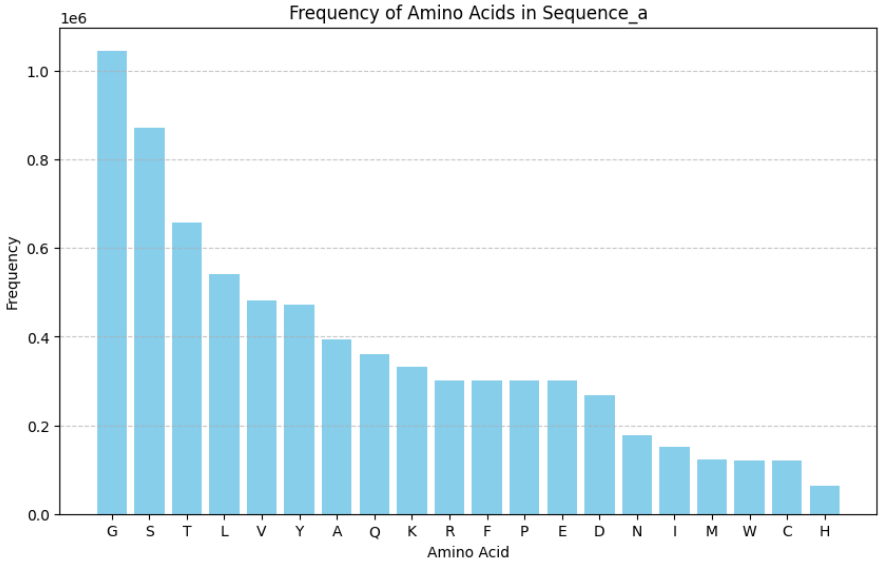


Fig 4. Distribution of amino acids within all variants of sequence_a

Q: Are all Kd predictions of sufficient quality?

- *We can we determine this by taking a look at the bound values. Assuming the upper Kd bound and the lower Kd bound represent the full width of Kd values for a sequence-sequence interaction means that a larger difference represents a less precise measurement*
- *I don't have all the information on Kd measurements but we can check the correlation between q_value and bound diff to determine their relationship. A higher correlation would imply that q_value and bound_diff are a bit redundant for identifying bad data.*

```
Sample AA Features: tensor([0.5000, 0.5000, 0.5000, 0.5000, 0.5000, 0.5000, 0.5000, 0.5000, 0.5000,
        1.0000, 0.5000, 0.6000, 0.5000, 0.5000, 0.5000, 0.5000, 0.5000, 0.3333,
        0.5000, 0.0000])
```

Fig 5. Sample frequencies calculated a single sequence_a variant. These tensors can be added as additional features for our model.

Data Wrangling and First Insights

Q: Are there correlations within the data?

- *The amino acid frequencies do not tend to correlate with Kd values, either Kd bound or the q-value. Therefore, these are safe features to use for our modelling*
- *The q-value and bound_diff are positively correlated. This implies that q-value measurements, at least partially, account for most instances of imprecise Kd measurements.*

Q: Are all Amino Acids known?

- *Yes, I checked that there only 20 unique amino acids. If we had anymore, we would likely be dealing with incorrect sequences that need to be removed.*

Q: What does the 'window' mean?

- Instructions say "TNYYMYWVRQAPGQGLEWMGGINPSNGGTNFNEKFKNRVTLTTDSSTTTAYMELKSLQFDDTAVYYCARRDYRFDMGFD" is where the mutations take place in sequence_a. However, it does not occur in every sequence_a. Given the small number of matches, I believe we should use all sequences in our training.

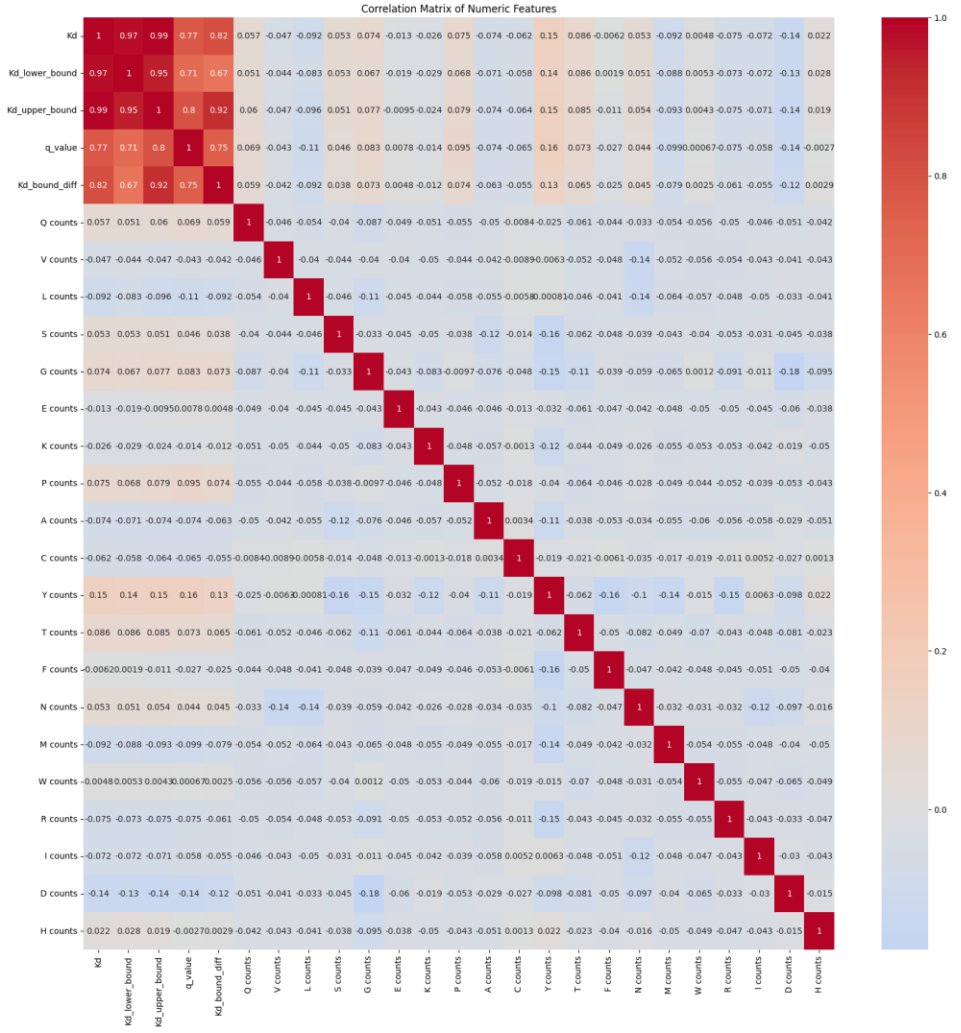


Fig 6. Correlation chart.

Data Wrangling Conclusions and Next Steps

1. There are 691 instances of missing Kd values. We will fill these by finding the midpoint between the lower bound and upper bound. If either bound is missing we will take the other bound as the Kd value. If both bounds are missing we drop the sequence.
2. There is a long low tail of Kd measurements. This suggests that we must be careful with overfitting and data skews within the test,train split.
3. I will focus my analysis on data that is below a q-value threshold of 0.05 because I want to train a model on measurements that do not fail the null hypothesis.
4. I will create new features based on the counts of AAs found within sequence_a mutations. We will experiment with different scaling methodologies.
5. The correlation chart suggests that the AA frequencies can be used and that the q-value and Kd bound diff are positively correlated. As such, we can use a q-value threshold to also, partially address, imprecise Kd measurements.

Model Selection Reasoning

- Proteins are complex structures with properties influenced by:
  - Local structural motifs
  - long range interdependencies (LRDs) between amino acids
  - sequential patterns
- Our sequences are focused on a window of 246 AAs so I believe the influence of LRDs is low.
- Research demonstrates that a simple sequential understanding of the protein sequences in 2d format is often insufficient. Capturing local structural motifs may prove a better approach
- Additionally, resource and time constraints prioritize CPU friendly models. See slide 12.

Recommended models

CNN

- Relatively faster training
- CNNS are particularly effective at feature detection in situations where combinations of local data substructures can be viable data (i.e. different combos of Aas)
- Ease of training – there is limited sequence preprocessing and padding necessary

RNN

- Simple RNNs can provide sequential information that may still be helpful
- LSTMs and GRU layers more helpful for what LRDs exist within the sequence_a string.

Random Forest + GBM

- Minimal tuning and engineering required
- Easy to interpret
- Quick to set up

Model Selection Structure and Reasoning

Contemporary research here largely use CNNs, RNNs, and Transformer architectures. Given my current compute restrictions (andfor ease of reproducibility) I'm going to focus on CNNs and RNNs for deep learning approaches.

- Quick testing suggests that Complex 1-dimensional CNNs **will run 3x faster** than RNN counterparts (especially GRU and LSTM methods) so we'll start with CNNs

Simple RNN

```
Model_Architecture:
ProteinKd_RNN_Prediction(
  (embedding): Embedding(20, 64)
  (rnn): RNN(64, 100, num_layers=2, batch_first=True, dropout=0.2)
  (to_features): Linear(in_features=100, out_features=100, bias=True)
  (fullyconnected): Sequential(
    (0): Linear(in_features=120, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

```
97.89922904968262 - time to  finish one epoch
```

Complex CNN

```
Model_Architecture:
ProteinKd_CNN_Prediction(
  (embedding): Embedding(20, 64)
  (layers): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (4): ReLU()
    (5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Flatten(start_dim=1, end_dim=-1)
  )
  (fullyconnected): Sequential(
    (0): Linear(in_features=15636, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
```

```
31.455698966979 98 - time to  finish one epoch
```

DeepLearning Model Architectures

RNN

A) Simple RNN

```
Model_Architecture:
ProteinKd_RNN_Prediction(
  (embedding): Embedding(20, 64)
  (rnn): RNN(64, 100, num_layers=2, batch_first=True, dropout=0.2)
  (to_features): Linear(in_features=100, out_features=100, bias=True)
  (fullyconnected): Sequential(
    (0): Linear(in_features=120, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

B) GRU RNN

```
Model_Architecture:
ProteinKd_RNN_Prediction(
  (embedding): Embedding(20, 64)
  (rnn): GRU(64, 125, num_layers=3, batch_first=True, dropout=0.2, bidirectional=True)
  (to_features): Linear(in_features=250, out_features=250, bias=True)
  (fullyconnected): Sequential(
    (0): Linear(in_features=270, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

CNN

A) CNN w/ DropOut

```
Model_Architecture:
ProteinKd_CNN_Prediction(
  (embedding): Embedding(20, 64)
  (layers): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (4): ReLU()
    (5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Flatten(start_dim=1, end_dim=-1)
  )
  (fullyconnected): Sequential(
    (0): Linear(in_features=15636, out_features=128, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

B) CNN w/ no DropOut

```
ProteinKd_CNN_Prediction(
  (embedding): Embedding(20, 64)
  (layers): Sequential(
    (0): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
    (1): ReLU()
    (2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv1d(128, 256, kernel_size=(3,), stride=(1,), padding=(1,))
    (4): ReLU()
    (5): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Flatten(start_dim=1, end_dim=-1)
  )
  (fullyconnected): Sequential(
    (0): Linear(in_features=15616, out_features=128, bias=True)
    (1): ReLU()
    (2): Identity()
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

Experiment Organization

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Experiment 6	Experiment 7
CNN	OneHot_No_AA	Embed_No_AA_No_Dropout	Embed_w_AA_No_Dropout	Embed_w_AA_w_DropOut	Embedding_w_AA_Max_Over Fit*	OneHot_w_AA_w_Dropout	
RNN	RNN_OneHot_w_AA_w_DropOut	RNN_Embed_w_AA_w_Dropout	GRU_Embed_w_AA_w_DropOut_Bidirectional				
CNN + RNN							
Classic ML	RandomForest_OneHot_w_AA_hyperparameter tuning	Gradient_OneHot_w_AA_hyperparameter tuning					

*MaxOverfitting = DropOut + BatchNorm

- Key Hypotheses:
  - Learned embedding representations of the sequences will out perform the one hot sequence representations
  - Amino acid frequencies can serve as helpful features and will improve regression performance
  - RNNs can supplement CNN results but CNNs will outperform RNNs
  - More complex RNNs will improve RNN performance
  - DeepLearning methods will out perform classic machine learning efforts
- Computer restraints meant that we could do many more experiments for CNN and Classic Machine Learning Architecture
- Performance to be evaluated by R2, RMSE, MAE, Loss. R2 is the primary metric followed by MAE

Experimental Results and Interpretation

Hypothesis: Embeddings outperform OneHot Encoding within CNNs

- *Likely rejected!*
- *Learned embeddings don't have significantly higher metrics, take longer to train and appear to be more subject to overfitting (qualitative assessment)*
- *However, could this change with the inclusion of AA data?*

	R2	RMSE	MAE
Max	.765	N/A	N/A
Min	N/A	.343	.262
Avg	.534	.467	.377

Table 1: Min, Max, and Avg key metric values for One Hot Encoding. N/A is used for scores not considered in evaluation

	R2	RMSE	MAE
Max	.773	N/A	N/A
Min	N/A	.343	.259
Avg	.542	.539	.382

Table 2: Min, Max, and Avg key metric values for Embedding. N/A is used for scores not considered in evaluation

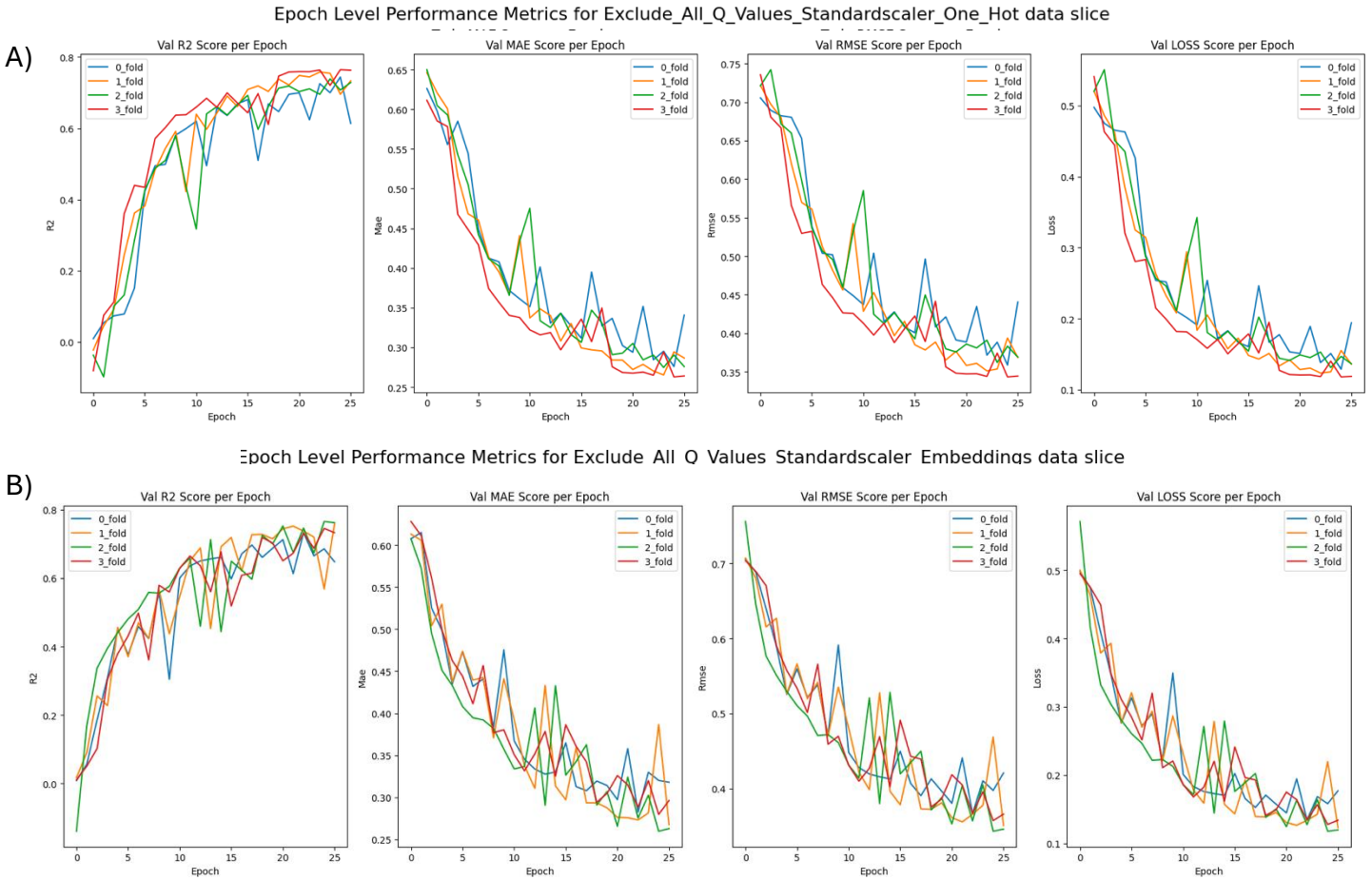


Fig 7. No AA, No Dropout, OneHot (A) vs Embeddings (B) validation performance.

Experimental Results and Interpretation

Hypothesis: AA features improve regression peformance

- *~2% R2 improvement. However, we see that scaling types can cause different levels of overfitting. It seems that MinMax scaling of AA features may be more prone to overfitting. Lets continue to investigate this!*

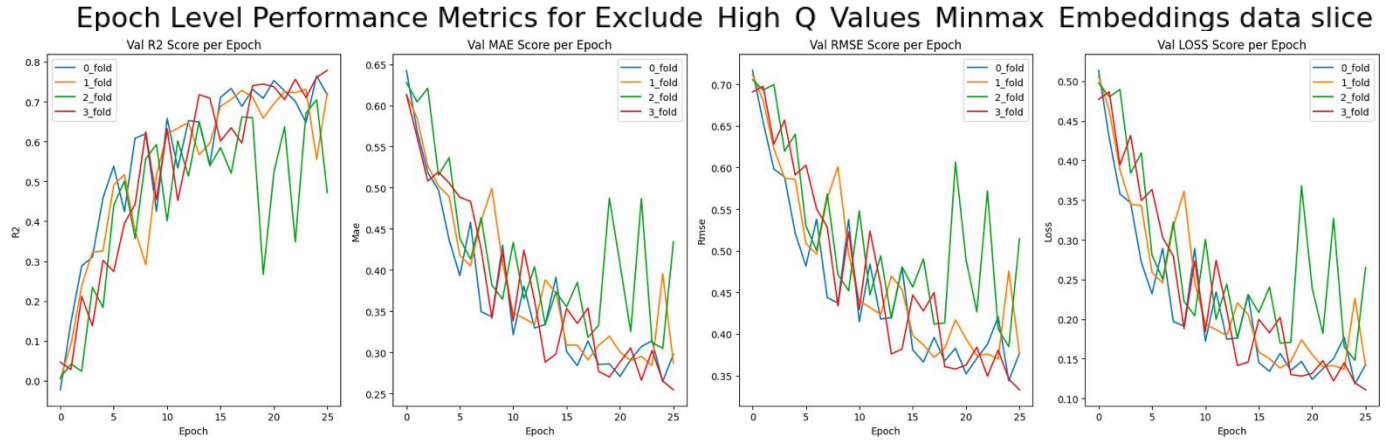
	R2	RMSE	MAE
Max	.778	N/A	N/A
Min	N/A	.333	.255
Avg	.519	.48	.39

Table 3: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

	R2	RMSE	MAE
Max	.752	N/A	N/A
Min	N/A	.352	.28
Avg	.527	.481	.387

Table 4: Min, Max, and Avg key metric values for StandardScaler. N/A is used for scores not considered in evaluation

A)



B)

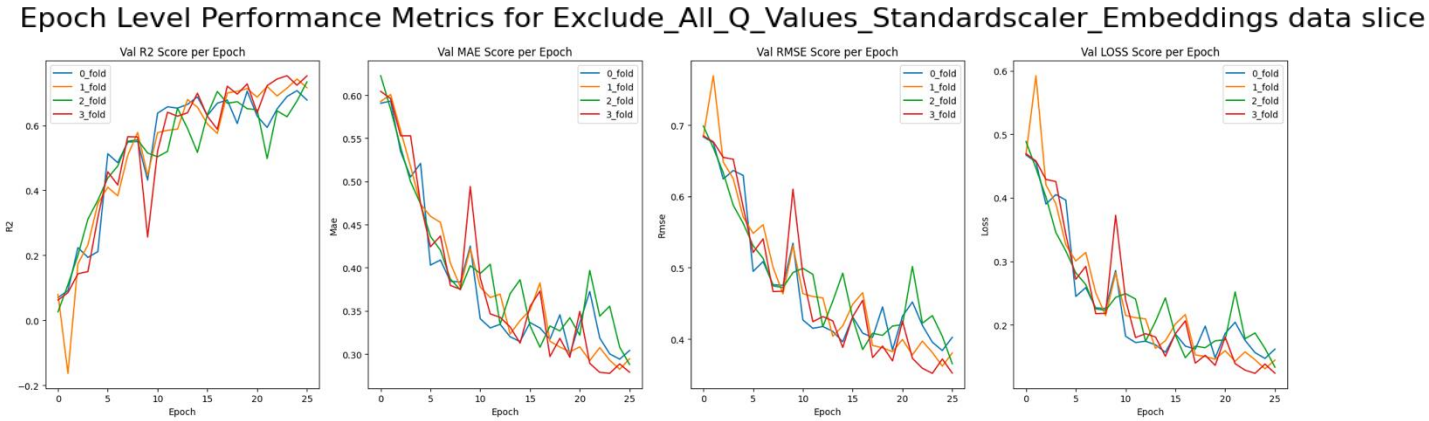


Fig 8. Embeddings, AA features, No Dropout, MinMax Scaling (A) vs StandardScaler Scaling (B) validation performance.

Experimental Results and Interpretation

CNN Experiments 4-6: Can we improve CNN performance?

Previous experiments demonstrated that embeddings and scaling type offer limited improvements. Lets explore how to improve scores using a combination of scaling and overfitting methods.

Experiments 5 and 6 were conducted only with MinMax scaling given time constraints and insignificant performance differences.

CNN Experiment 4:
Embedding_w_AA_w_Dropout

	R2	RMSE	MAE
Max	.717	N/A	N/A
Min	N/A	.377	.293
Avg	.495	.496	.405

Table 5: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

	R2	RMSE	MAE
Max	.715	N/A	N/A
Min	N/A	.377	.29
Avg	.500	.493	.405

Table 6: Min, Max, and Avg key metric values for StandardScaler. N/A is used for scores not considered in evaluation

CNN Experiment 5:
Embedding_w_AA_w_MaxOverfitting

	R2	RMSE	MAE
Max	.664	N/A	N/A
Min	N/A	.412	.325
Avg	.403	.445	.539

Table 7: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

CNN Experiment 6:
OneHot_w_AA_w_Dropout

	R2	RMSE	MAE
Max	.66	N/A	N/A
Min	N/A	.414	.329
Avg	.462	.512	.421

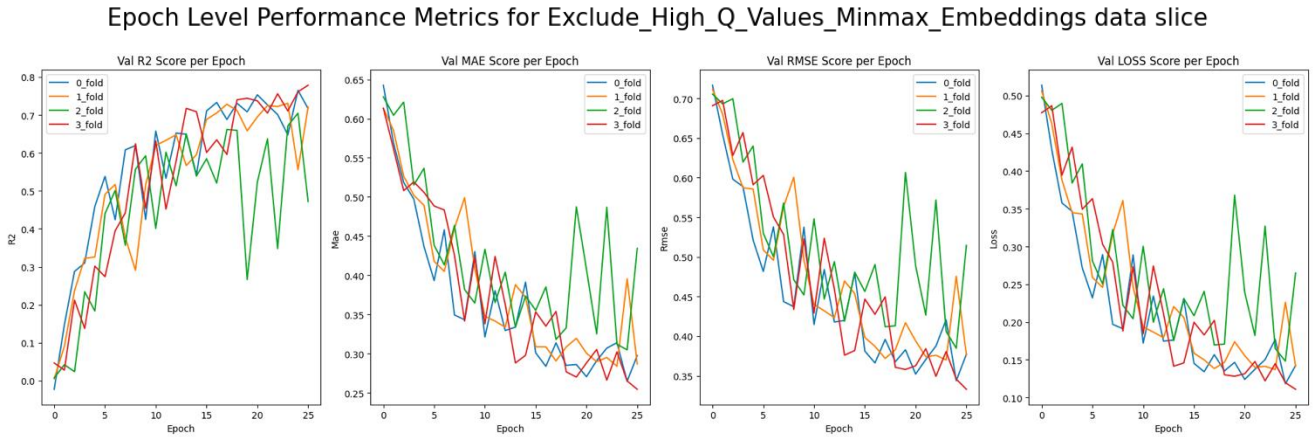
Table 8: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

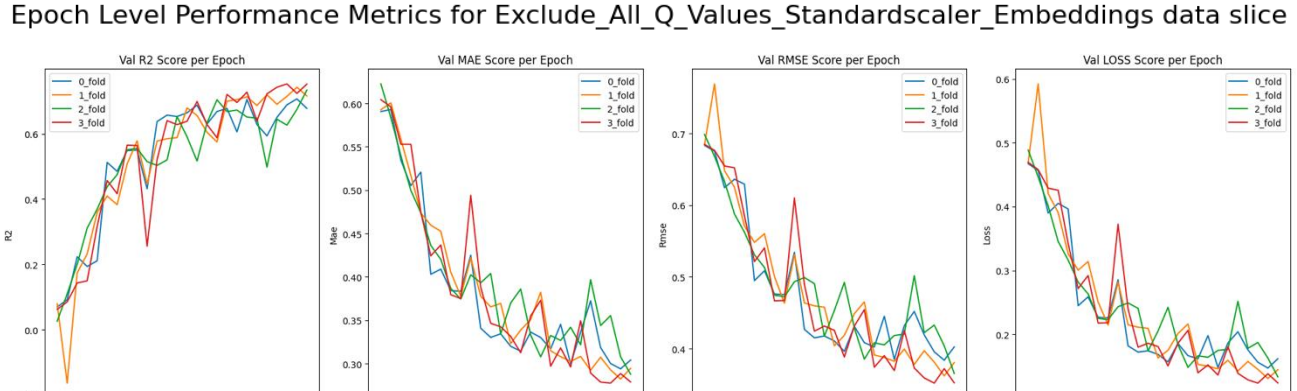
Experimental Results and Interpretation

CNN Experiments 4-6: Can we improve CNN performance?

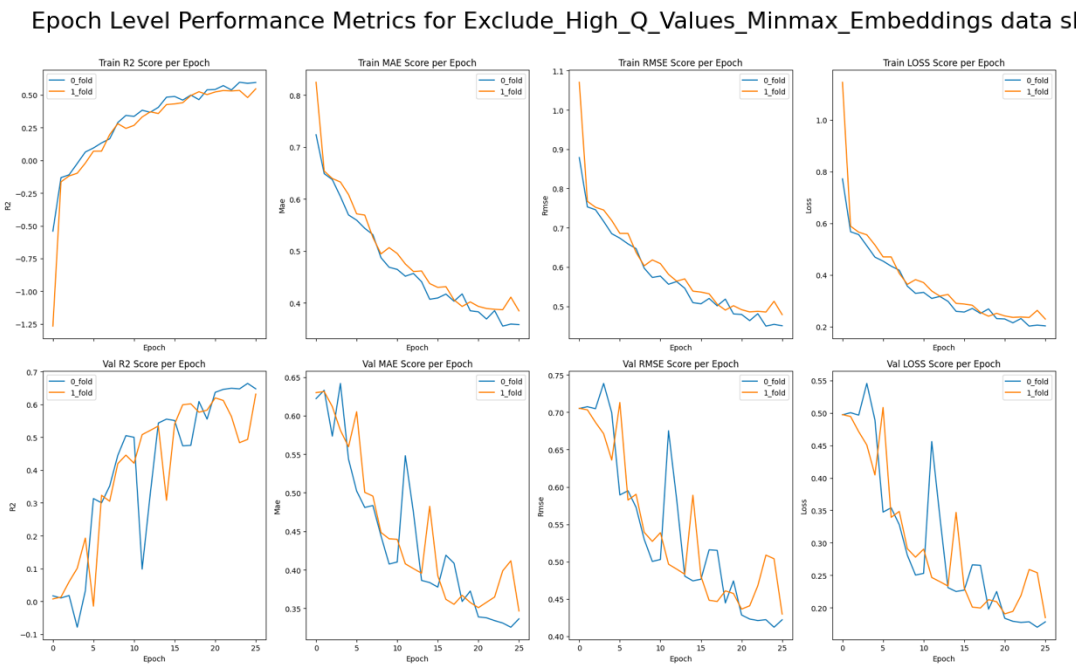
- *Regardless of scaling efforts we see that overfitting remains even with increased dropout rates and batch normalization prior to ReLu().*
- *Our best experiments (based on R2) from AA usage with minor dropout with MinMax scaling. We'll move forward with RNN experiments*
- *Further experiments for CNN should focus on convolution hyperparameter optimization. I'm doubtful greater complexity will assist given overfitting patterns*

Experiment 4: Embedding_w_AA_w_Dropout





Experiment 5: Embedding_w_AA_w_MaxOverfitting



Experimental Results and Interpretation

Hypothesis: RNNs can supplement CNN results but CNNs will outperform RNNs

CNNs outperform RNNs by a long shot. However, simple RNNs cannot supplement CNNs They have low performance and take significantly longer to train (90-120 minutes)! Considering low performance and CNN work, we only evaluated the MinMax scaling method.

I also tested a GRU architecture to see if perhaps simple RNNs were missing long range dependencies. That was also ineffective.

RNN Experiment 1:
Embedding_w_AA_w_Dropout_Bidirectional

	R2	RMSE	MAE
Max	.036	N/A	N/A
Min	N/A	.695	.598
Avg	.001	.709	.636

Table 9: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

RNN Experiment 2:
Embedding_RNN_w_AA_w_Dropout

	R2	RMSE	MAE
Max	.043	N/A	N/A
Min	N/A	.692	.597
Avg	.007	.707	.624

Table 10: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

RNN Experiment 3:
Embedding_GRU_w_AA_w_Dropout

	R2	RMSE	MAE
Max	.037	N/A	N/A
Min	N/A	.691	.602
Avg	.004	.708	.621

Table 11: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

Experimental Results and Interpretation

Hypothesis: *DeepLearning efforts will outperform classic machine learning efforts.*

My hypothesis was correct! I tested a random forest model and a gradient boosting model with 3-fold cross validation and hyperparameter tuning. Even with optimized hyperparameters neither model outperformed CNNs.

One of my subgoals for this set of experiments was to demonstrate the importance of hyperparameter tuning. It was easier andmore effective to show grid search results here.

Classic ML Experiment 1: Random_Forest_w_AA

	R2	RMSE	MAE
Best Score	.439	.532	.425

Table 12: Random Forest best results with parameters: (max_depth=25, min_samples_split=5, n_estimators=200, random_state=15)

Classic ML Experiment 2: Gradient Boosting_w_AA

	R2	RMSE	MAE
Best Score	.48	.412	.44

Table 13: Gradient Boosting best results with parameters: (max_depth=25, min_samples_split=5, n_estimators=200, random_state=15)

Major Conclusions and Next Steps

Conclusions:

- CNN architectures performed the best out of all our experiments.
  - Overfitting remains a challenge even with DropOut and Batch Normalization.
  - Amino Acid frequencies do aid in performance but scaling method seems irrelevant
  - Embeddings and One Hot encoding have a surprisingly similar performance. However, I recommend proceeding with embeddings because of the wider adaptation of protein representation learning in contemporary research
- RNNs have such low performance with such high time complexity and compute requirements that I decided not to further investigate
- Classic models such as Random Forests perform better than RNN but still significantly worse than CNN.

Next steps:

- Hold out predictions!

Recommend improvements:

- Further experiments in deeper CNN models but I suspect that overfitting will worsen.
- CNN hyperparameter tuning focusing on learning rate, conv and pooling kernel size, and stride length experiments
- I recommend additional feature engineering with bioinformatics or chemical data for more complex models. Additionally any protein specific embeddings would likely help.
- GPU adaptation for training speed increase
- Transformer architecture experiments.
- I ran out of time but I'd like evaluate performance of our final model on the data including high q-value.

HoldOut Set Predictions

- Running the hold out predictions came with a surprise! **Not all sequences were the same length as we saw in the training data!**
- To correct for this, I took our best performing CNN architecture (CNN_w_AA_w_DropOut) with the best performing sequence encoding type + AA feature scaling and retrained our model with new padding
  - Padding set to the max length of the hold out set sequences.
  - I double checked to make sure we didn't have any unknown Amino Acids (even though there are only 20.) This was to make sure t here weren't any mistakes in the holdout data
  - Re-training scores show slightly poorer performance (~3% diff)
- After re-training our best architecture, I predicted on the hold out set.
  - Upon evaluating my hold out predictions, I saw there were negative Kd values. Unfortunately, that doesn't really make sense in protein binding so more work is necessary!

Retraining_CNN_w_AA_w_DropOut_Padding

	R2	RMSE	MAE
Max	.734	N/A	N/A
Min	N/A	.368	.293
Avg	.529	.477	.386

Table 14: Min, Max, and Avg key metric values for MinMax. N/A is used for scores not considered in evaluation

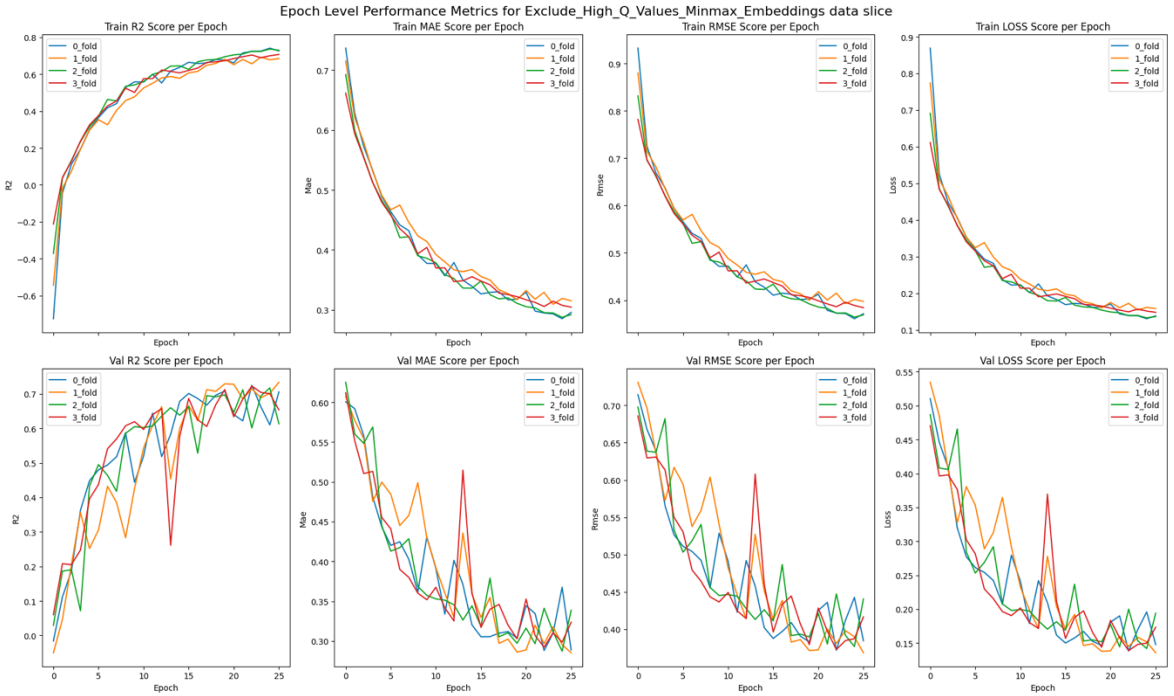


Fig 9. Performance when re-training accounting for padding

Constraints and Impact

- I did not have access to GPU on my personal laptop and I was hesitant to develop on a cloud service given time constraints.
- I focused my analysis on those training samples that had a q_value <= 0.05. While I believe this is appropriate given the q_value's relationship to the null hypothesis this did significantly lower the size of our training set and could limit model generalizability.
    - It's possible that user would want to predict on lower confidence sequence interactions given the relatively small amount of this type of data in the industry.
- I did not re-evaluate all CNN approaches with the padding technique necessary for the HoldOut set. While unlikely, it's possible that the padding would change which approach was most effective.
- I strongly believe that the models would benefit from chemical or bioinformatics information on the protein sequences/amino acids. However, it was not possible to gather this information (especially for the whole protein sequence) given that the challenge was focused on a window of\the whole protein.
- I observed negative Kd values in my predictions for the hold out set, which to my knowledge is not possible. More research is needed to prevent these predictions and establish a floor for my predictions.