Course Project Final Report

Zachary Harrison
Computational Mathematics
Logan, Utah
Zachary.Mark.Harrison@gmail.com

Keldon Boehmer
Computer Science
Logan, Utah
A02324008@aggies.usu.edu

ABSTRACT

In this paper, we will discuss the the problems and models encountered in this competition [2]. In short, the task for this competition is to approximate the melting temperature of proteins based on their amino acid sequence. The three main models (along with their relative efficacy) that will be discussed are:

1. A linear network or decision tree with input as the sequence of letter encoded amino acids
2. A CNN with input as 2-d matrix of physical and chemical properties of each amino acid in the protein sequence
3. A CNN with input as 3-d matrix of voxel features of each unique mutation

For all the above methods, the task was a regression task. So, their objective functions were all some variation of mean square error.

INTRODUCTION

In order to fully explain this competition, we need to review some details. We explained in the above abstract that the task is to predict the melting temperatures of proteins, but the scoring system for the competition itself is more complicated. Unlike other competitions, this one does not use accuracy to determine Kaggle score. Instead, it uses the Spearman's rank correlation, meaning that a model's score is not based on its prediction of the melting temperature of each protein, but rather, the relative order in which the model predicts each protein to melt in. Because of this change, scores range from $[-1, 1]$, where 1 indicates a perfect ordering of the test proteins and -1 is the reverse ordering.

Something important to note about this competition is that all *significant* progress was not made through changes to model architecture. Rather, changes to data were almost always the only method of significant improvement. We are unsure if this is a common theme for challenging data science competitions, but it was absolutely a theme in all Kaggle discussions about this one.

We ended up achieving a public ranking of #172, which was the same score as #47. There were over 1600 teams in the competition at the time, which means we briefly had the same score as someone in the top 1%! The current leader of the competition is Chris Deotte, who has a score of 0.801. The runner-up has a score of 0.666, so everyone is really just aiming for second place because first place is so far out of reach. Overall, we're very happy with how well we did.

Figure 1. Final public leaderboard rank as of November 28th

PROBLEM 1 - THE TRAINING SET(S)

Now, let's get into examining the data for this competition. Every competitor for this competition had to deal with the following unfortunate truth about the training set: it's bad. You see, the Kaggle staff member who created this competition provided a training set with data that has null values. They gave an update, but instead of simply fixing their mistake, they provided an entirely separate .csv file that needs to be manually added to the previous one. What's more is that even after all this hassle, the training set still isn't very good. As a result, many competitors have enlisted the aid of alternative data sets, and some have even found a way to combine many data sets.

PROBLEM 2 - PROTEIN SEQUENCES

The length of each protein sequence is a problem everyone in this competition had to tackle. You see, all the protein sequences are of varying lengths. Because every model has a set of weights (whether it be a matrix or a list) are tuned to produce an accurate output by scaling various portions of the input data. However, if the input data is not of constant size, it becomes hard to scale a particular portion of the input data if said portion does not exist. So,

MODEL TYPE 1 - LETTER ENCODING

Data

The only data this model types uses is the Kaggle training set (and the testing set, obviously)

Data Preprocess steps

1. Convert the protein sequence into a list of letters.
2. Only accept proteins from the training set whose length is between 100 and 224.
3. Lengthen all proteins to 224 by adding "None" for missing values.

If you want more information about how we implemented these models, take a look at our Github[1], which documents all our code.

4. Use LabelEncoding package from sklearn.preprocessing to perform encoding on the amino acids of the protein sequence. This automatically handles the "None" values.

The Model(s)

Each data point in our training set now has a list of numbers for each protein sequence. This will be our input for the following models:

1. XGBRegressor from xgboost
2. DecisionTreeRegressor from sklearn.tree
3. RandomForestRegressor from sklearn.ensemble
4. AdaBoostRegressor from sklearn.ensemble
5. GradientBoostingRegressor from sklearn.ensemble
6. LinearRegressor from sklearn.linear_model
7. SGDRegressor from sklearn.linear_model

Strategy Efficacy

See Figure 2 below for a graph of their relative performance. You'll notice that a couple models performed **worse** than the dummy submission (which simply ordered the proteins by their sequence id). And the models that performed better weren't significantly better than the dummy submission.

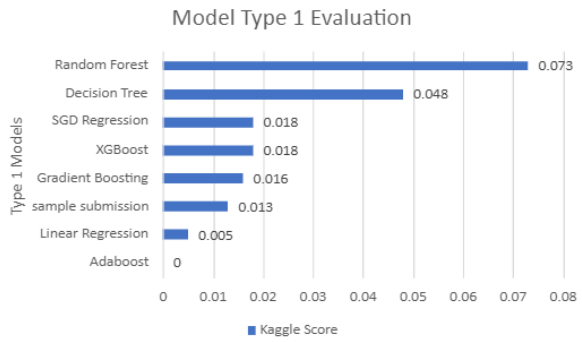

Figure 2. The Kaggles Scores for each model of type 1 and the dummy submission

Clearly, this model type is not very effective. So, if the protein sequence can't accurately predict its melting temperature, then more information about the protein is required. The next model type explores the idea of using the physical and chemical properties of each amino acid.

MODEL TYPE 2 - HOLTRYNET2D

This model works under the assumption that features of one amino acid can, if given sufficient context for the physical and chemical properties of nearby amino acids, accurately predict the melting temperature of a protein sequence. For instance, it is known that the more polar the amino acids that comprise a protein are, the higher its melting temperature. So, an example of giving context to the polarity of an amino acid might mean weighting its polarity based on its molecular weight. That is, a very polar amino acid with a large molecular weight will more greatly contribute to the high melting temperature of the protein. Now that we've explained the idea behind this model, let's get into the specifics of how it works.

Data

This model also uses the training and testing set provided by Kaggle. Additionally, it uses output from Moth's Notebook: Aminoacids: Physical and Chemical Properties [7].

Data Preprocess steps

1. Convert the protein sequence into a list of letters.
2. Only accept proteins from the training set whose length is between 100 and 224.
3. Lengthen all proteins to 224 by adding "None" for missing values.
4. Make a transformation dictionary that converts amino acids into a list of physical and chemical properties, such as molecular mass and polarity.
5. Standardize the physical and chemical properties for each animo acid.
6. Add an amino acid that converts amino acids with label "None" into padding for CNN.
7. Convert each protein sequence into a matrix of physical and chemical properties, where each column is a different property and each row is an amino acid.

Feature Engineering

The feature engineering for this model type was limited to selecting the physical and chemical properties that were believed to be useful. We kept all physical and chemical properties that were numbers. Some of these included: Molecular Weight, pI4 (the pH at the isoelectric point), polarity, and polarizability. There were 16 total features that did not get ignored.

The Model(s)

This model type only used one model. When our first attempt performed this well, we brainstormed different ideas to improve it by increasing the feature space (either via polynomial feature scaling or attempting to provide more context for the model). While we think these ideas would have improved performance, we abandoned them in favor of the next model type due to time constraints. we also briefly attempted to use the larger training set described in Model Type 3, but we were unable to get it working well and discarded the idea.

Strategy Efficacy

See Figure 3 for a graph of this model's performance in comparison to the previous attempt. You'll notice that, unlike model type 1, it did perform orders of magnitude better than the dummy submission (and those that did worse than it). However, it did not significantly outperform the RandomForestTree model.

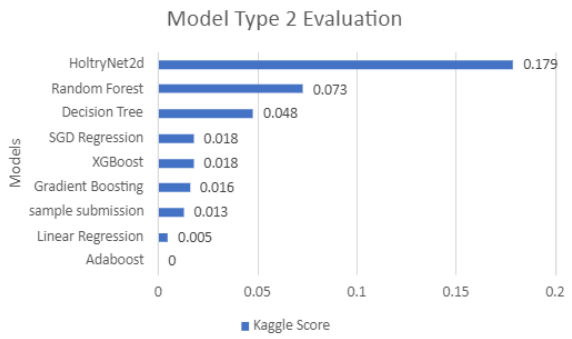

Figure 3. The Kaggles Scores for the type 1 models, dummy submission, and HoltryNet2d

					ddG	dT			
256	ADQLTEEC	K	75	74	A	0.6	1.5			['ThermoN 1cfdA
257	ADQLTEEC	K	75	74	I	1.7	7.8			['ThermoN 1cfdA
258	ADQLTEEC	L	39	38	F	-1.1	-5.5			['ThermoN 1cfdA
259	ADQLTEEC	Q	41	40	A	-0.1	-3.3			['ThermoN 1cfdA
260	ADQLTEEC	Q	41	40	L	1.3	2.8			['ThermoN 1cfdA
261	AEAGGPGI	D	189	188	A		-4	69	7.5	['pucci-prc 1ezmA
262	AEAGGPGI	D	201	200	A		1	69	7.5	['pucci-prc 1ezmA
263	AEAGGPGI	R	179	178	A		-7	69	7.5	['pucci-prc 1ezmA
264	AEAGGPGI	R	205	204	A		1	69	7.5	['pucci-prc 1ezmA
265	AECSVDIQ	A	82	81	G	-3.11			7	['dataset_'1e65A
266	AECSVDIQ	F	110	109	S	-3.95			7.3	['dataset_'1e65A
267	AECSVDIQ	H	46	45	G	-2.5			7	['dataset_'1e65A
268	AECSVDIQ	H	117	116	G	-2.18			7	['dataset_'1e65A
269	AECSVDIQ	I	7	6	A	-3.11			7	['dataset_'1e65A
270	AECSVDIQ	I	7	6	S	-3.44			7.3	['dataset_'1e65A
271	AECSVDIQ	I	20	19	A	-1.56			7	['dataset_'1e65A
272	AECSVDIQ	I	20	19	T	-2.39			7	['dataset_'1e65A
273	AECSVDIQ	I	20	19	V	-0.38			7	['dataset_'1e65A
274	AECSVDIQ	I	81	80	A	-2.39			7	['dataset_'1e65A
275	AECSVDIQ	I	81	80	T	-2.51			7	['dataset_'1e65A
276	AECSVDIQ	I	81	80	V	-0.07			7	['dataset_'1e65A
277	AECSVDIQ	K	24	23	R		0.8	78.7	7	['pucci-prc 1e65A
278	AECSVDIQ	L	50	49	T	-2.34			7	['dataset_'1e65A
279	AECSVDIQ	L	50	49	V	-0.36			7	['dataset_'1e65A

Figure 4. The Kaggles Scores for the type 1 models, dummy submission, and HoltryNet2d

So, even though this model type was more effective than previous attempts, it is still inadequate and needs revision. Even though we had more ideas to improve this attempt, we abandoned the idea when we read some Kaggle discussions and realized the high volume of information stored in PDB (Protein Data Bank) files.

The reason we abandoned this model type is because we suspected that convolutions were unable to obtain sufficient context for the properties of each amino acid. Consider the example we provided to illustrate the idea of context of amino acid properties, where the molecular weight needs to be a factor of polarity (which is known to be positively correlated with high melting temperatures). Because molecular weight was so far away from polarity in the list of properties, it was impossible for these two features to interact in the convolutions. And if we were to move columns around to allow these two features to interact, we were depriving other features from interacting with them. So, sufficient context was impossible to be obtained through this method, we instead abandoned the idea for the next model type.

MODEL TYPE 3 - HOLTRYNET3D

Data

This model uses a dataset which is a combination of multiple external datasets. See NESP: 14656 Unique Mutations + Voxel Features [9] for more details.

Additionally, it was difficult to figure out how to use this dataset. As a result, NESP: ThermoNet v2 [10] was extraordinarily helpful in understanding how best to use it because it demonstrated how to use it.

Understanding the Dataset

This dataset is extremely abnormal because it has one or both of two different target values: ddG (the change in the change in Gibb's Free Energy) and dT (the difference in melting temperature). See Figure 4.

You'll notice that each data point has either one or both target values (ddG and dT). This means that our model needs to have an objective function that is able to handle two target values. However, it may be the case that one of these target values is more/less important than the other and should be scaled up/down. Thus, the objective function is:

$$\mathscr{L} = (y_{ddG} - \hat{y}_{ddG})^2 + C \cdot (y_{dT} - \hat{y}_{dT})^2,$$

where $C = 0.01$ is a hyperparameter that determines how much dT contributes to the loss function.

Data Preprocess steps

1. Use the wildtype *.npy files generated by NESP: 14656 Unique Mutations + Voxel Features [9] to generate mutant PDB files.
2. Use the mutant PDB files to obtain the 3D structure for each protein via the numpy function numpy.load()

Feature Engineering

In order to use two target values, we need to ensure that they are strongly correlated. Look at the graphs in figures 5-7 and observe that there is absolutely a strong positive correlation between ddG and dT.

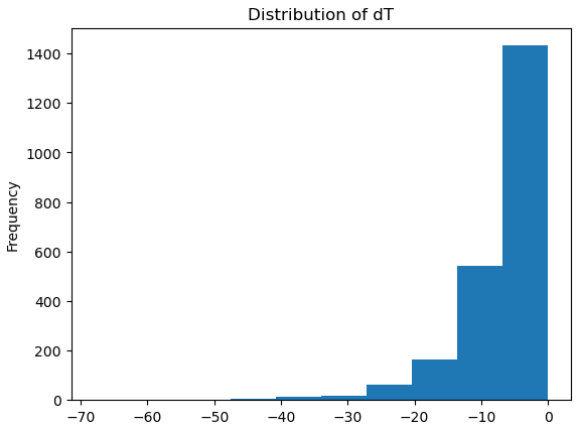

Figure 5. Graph of the frequency of dT target values in certain interval ranges

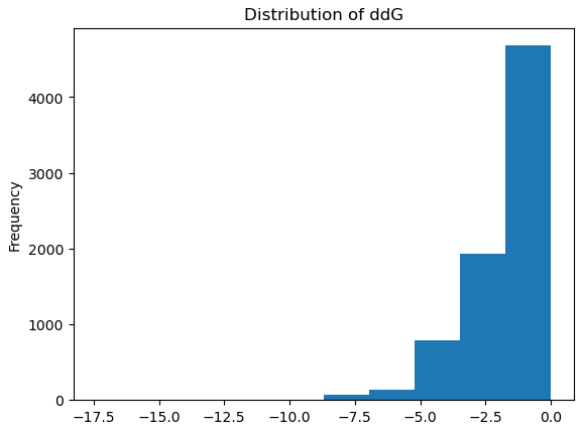

Figure 6. Graph of the frequency of dT target values in certain interval ranges

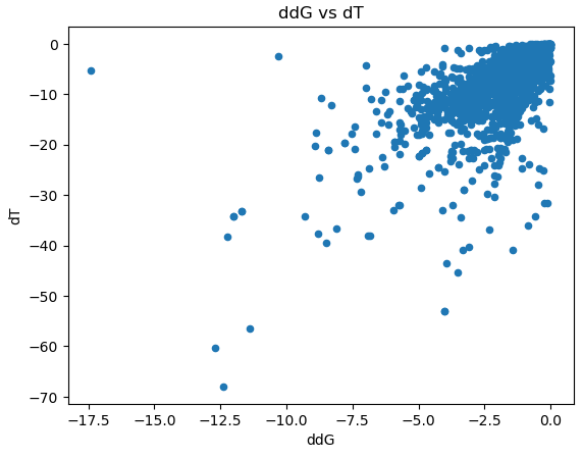

Figure 7. Comparison of ddG and dT distribution

Strategy Efficacy

There were three attempts of this model type: HoltryNet3d without batch norm (0.297), HoltryNet3d with batch norm (0.446), and HoltryNet3d ensemble (0.592). The best performing model was Deletion Specific Enzyme (0.603), which was a public model from the Kaggle Discussion boards that does some data preprocessing that deletes specific enzymes responsible for the model tuning poorly.

Look to Figure 8 below for a visual comparison of these models and the ones that came before it.

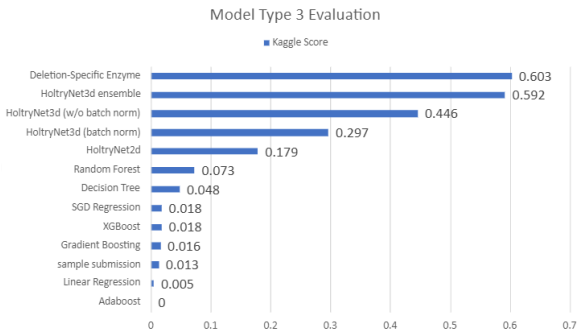

Figure 8. The Kaggles Scores for all previous models as well as Model Type 3 and Deletion-Specific Enzyme

Each hidden layer in HoltryNet3d with batch norms had a batch norm. For HoltryNet3d without batch norms, we removed *all* of them.

In order to explain how HoltryNet3d ensemble was created, we need to dive deeper into how the score for this competition is calculated. As you recall from the Introduction, a perfect score of "1" is a perfect ordering of the proteins and the reverse ordering results in a score of "-1". We can take advantage of this by combining our model with other top public models using a function called rankdata from scipy.stats. This function essentially ranks your data, ordering it for you. Proteins that a model predicts to melt first receive a low number, and proteins that melt last will receive a high number. Now, using the ordering of our data and other public predictions, we can *ensemble* them by multiplying each model's prediction by an assigned weight and adding them all together. Here is a list of the public models (with hyperlinks) used and their assigned weights

Public Model	Ensemble Weight
RSMD from Molecular Dynamics [3]	0.7
pLLDT, DDG, demask, sasa [6]	0.3
NOVO ESP - Performant Approaches [8]	0.3
NESP: AlphaFold+GetArea [5]	0.3
HoltryNet3d (without batch norms [1]	0.3
NESP: Relaxed Rosetta Scores [4]	1.0

Note that each ensemble weight is a hyperparameter.

FUTURE WORK

We would be interested in doing some research in the following techniques discovered throughout this competition:

1. How different Schedulers work and potentially improve performance
2. More models that use Ensemble Learning as its focus
3. Weights And Biases (W&B). From what we understand, it has some method of producing optimal hyperparameters. You still need to define a model architecture, but it can find the optimal dropout rate, learning rate, and even weight decay! We weren't able to understand how to use it, but it seems like an extremely powerful resource.

REFERENCES

[1] Keldon Boehmer and Zachary Harrison. 2022. Truth. (2022). https://github.com/Zachary-Harrison/cs5665-Truth.

[2] Dennis Pultz, Esben Friis, HCL-kanishkaa, Jesper Salomon, Maggie, Peter Fischer Hallin, Sarah Baagøe Jørgensen, Walter Reade. 2022. Novozymes Enzyme Stability Prediction. (2022). https://kaggle.com/competitions/novozymes-enzyme-stability-prediction

[3] Oscar Villarreal Escamilla. 2022. RMSD from Molecular Dynamics. (2022). https://www.kaggle.com/code/oxzplvifi/rmsd-from-molecular-dynamics.

[4] greySnow. 2022. NESP: relaxed rosetta scores. (2022). https://www.kaggle.com/code/shlomoron/nesp-relaxed-rosetta-scores.

[5] Robert Hatch. 2022. NESP: AlphaFold+GetArea exploration. (2022). https://www.kaggle.com/code/roberthatch/nesp-alphafold-getarea-exploration.

[6] kvigly. 2022. pLLDT, DDG, demask, sasa. (2022). https://www.kaggle.com/code/kvigly55/plldt-ddg-demask-sasa.

[7] moth. 2022. Aminoacids: Physical and Chemical Properties. (2022). https://www.kaggle.com/datasets/alejopaullier/aminoacids-physical-and-chemical-properties.

[8] Darien Schettler. 2022. NOVO ESP – ELI5 - Performant Approaches [LB=0.451]. (2022). https://www.kaggle.com/code/dschettler8845/novo-esp-eli5-performant-approaches-lb-0-451.

[9] Vladimir Slaykovskiy. 2022a. 14656 Unique Mutations+Voxel Features+PDBs. (2022). https://www.kaggle.com/code/vslaykovsky/14656-unique-mutations-voxel-features-pdbs.

[10] Vladimir Slaykovskiy. 2022b. NESP: ThermoNetv2. (2022). https://www.kaggle.com/code/vslaykovsky/nesp-thermonet-v2.