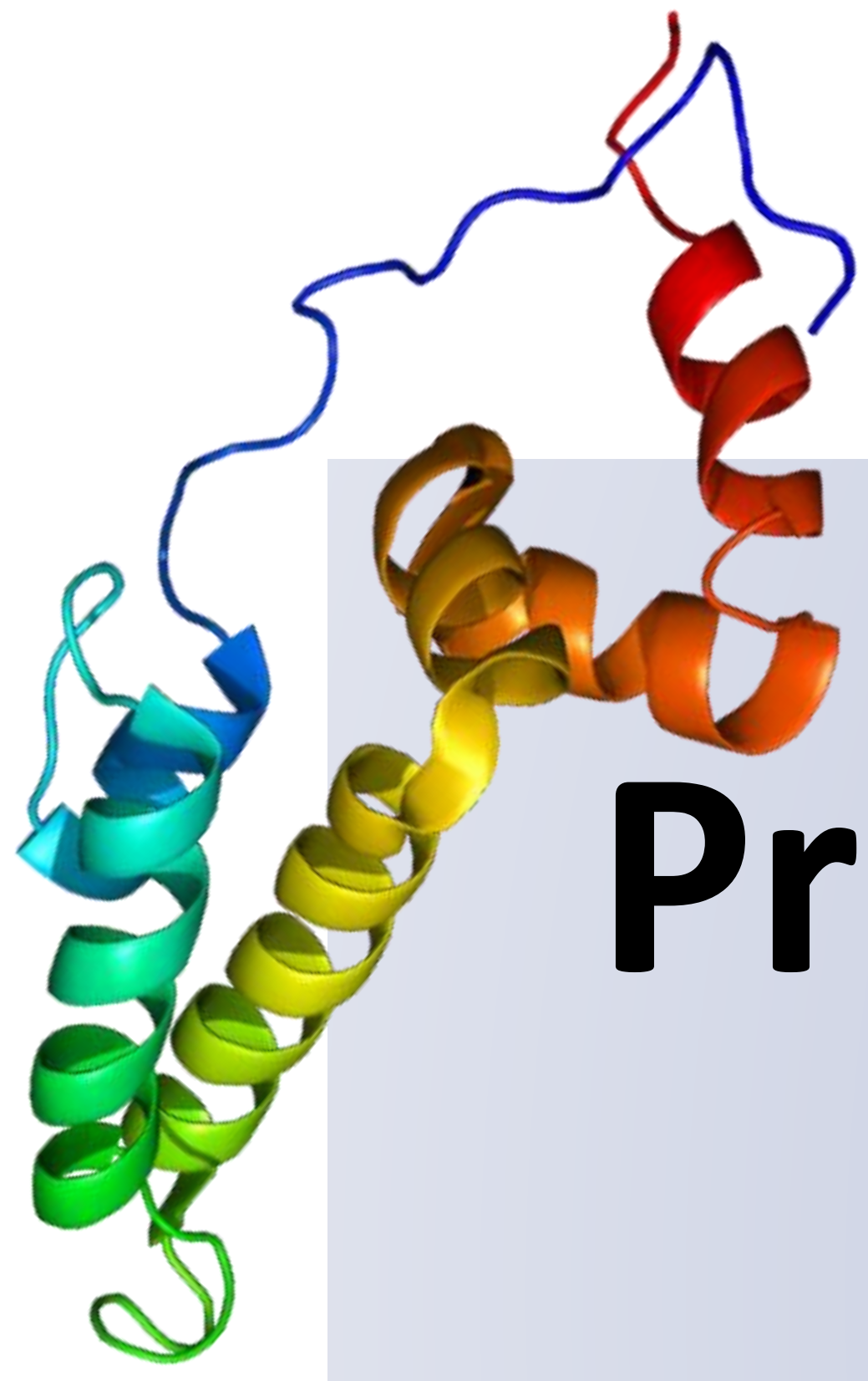
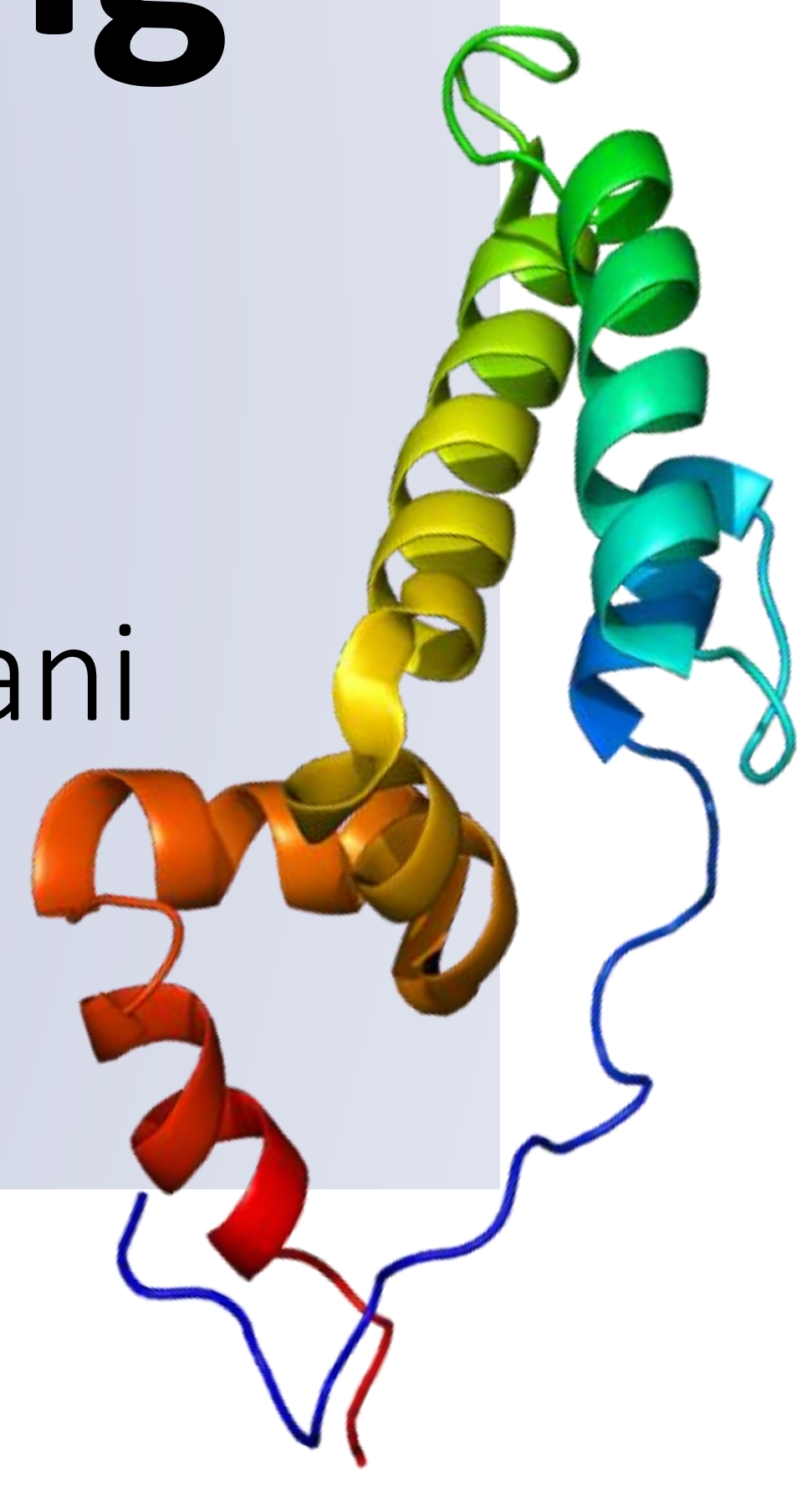
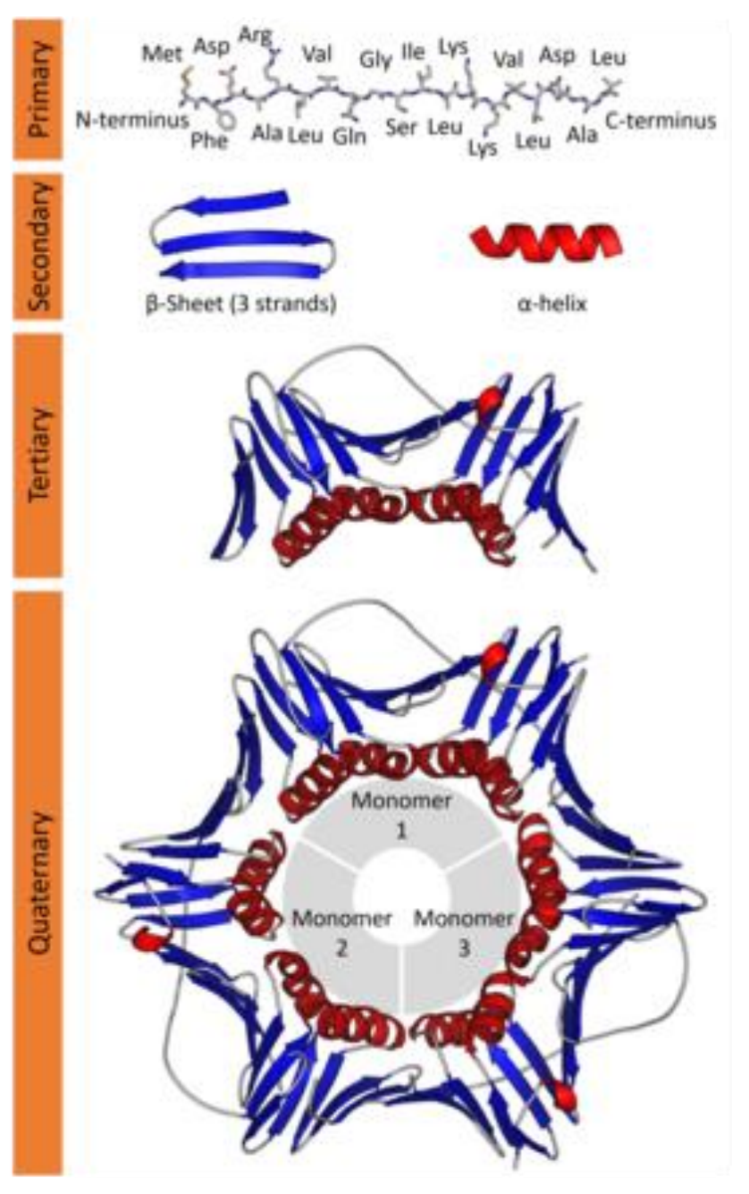


Prediction of Enzyme Classification using Protein Sequence Embeddings

Ambika Sundaresan, Breanne Baldino, Cindy Yu, Matteo Pinto, Tahamtan Dokhani
Advisor: Dr. Peter Rose



Problem Statement


Primary
Met Asp Arg Val Gly Ile Lys Val Asp Leu
N-terminus Phe Ala Leu Gln Ser Leu Lys Leu Ala C-terminus
Secondary
β-Sheet (3 strands)
α-helix
Tertiary
Quaternary
Monomer 1
Monomer 2
Monomer 3

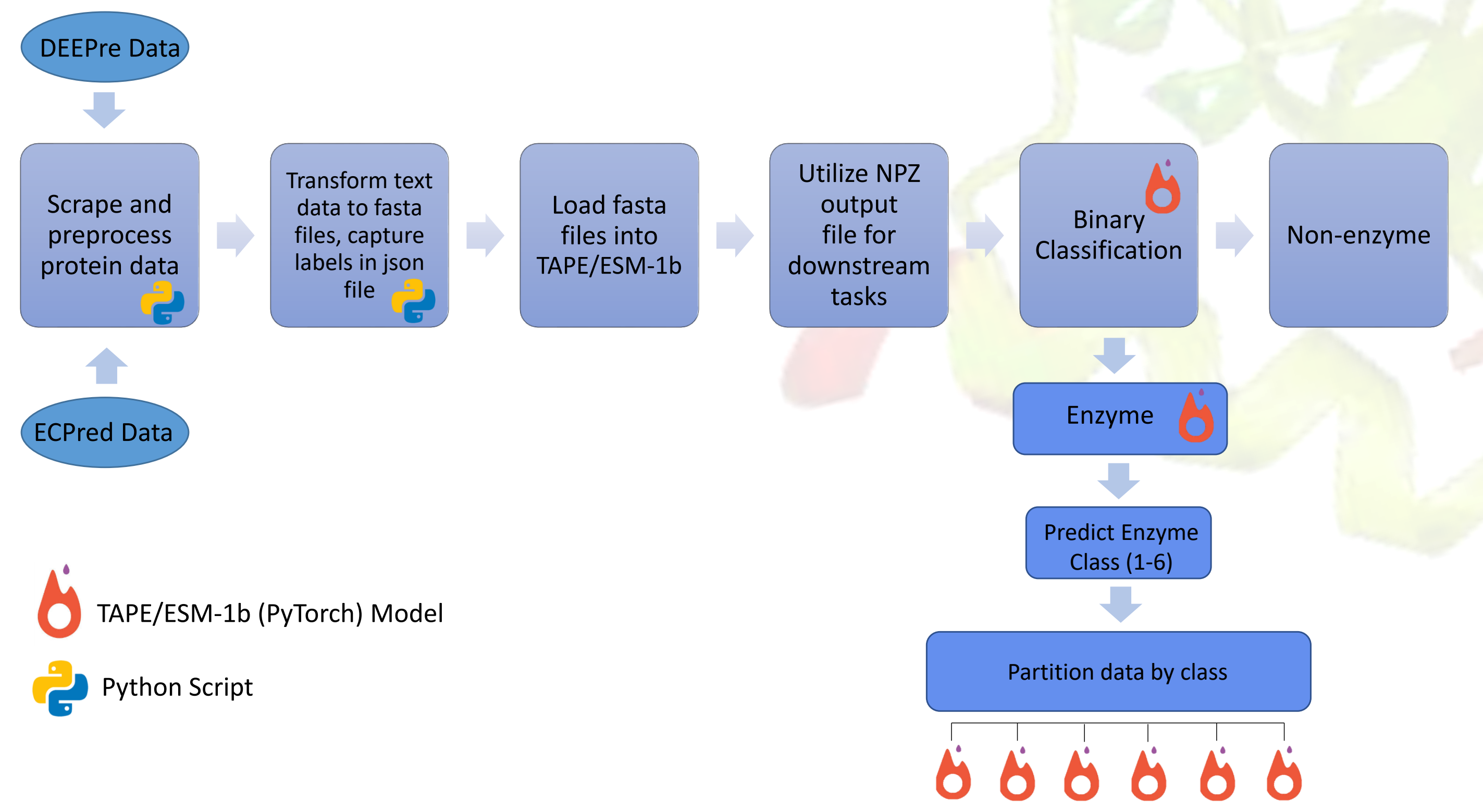
Biologists work with a multitude of protein sequences represented by strings of letters each denoting an amino acid. The amino acid sequence of these proteins allows us to leverage various machine learning Natural Language Processing (NLP) algorithms aimed to predict enzyme classifications, which are indicative of both protein structure and function. We propose a multi-level classification solution that is designed to predict the respective class of a given enzyme. Our approach consists of predicting the classification of an enzyme by applying NLP to a protein sequence. Our method utilizes BERT (Bidirectional Encoder Representations from Transformers) models to create embeddings, or feature vectors, and a variety of machine learning models to predict the respective class of an enzyme.

Currently, protein data discovery is outpacing the rate at which enzymes are classified, thus creating a demand for timely and efficient enzyme classification. The methods available for classifying enzymes can be both time and resource intensive. Our goal is to determine enzyme classes of respective protein strings in a time and cost-efficient manner. As mentioned, there are seven different types of enzymes and several additional subclasses per each enzyme class. Based on the data available, we will be performing predictions of enzyme classes for the first six enzyme classes. Our multi-class classification approach will first perform a binary classification to determine if the sequence is an enzyme first, then if the protein is an enzyme, perform a six-class classification to complete the prediction.

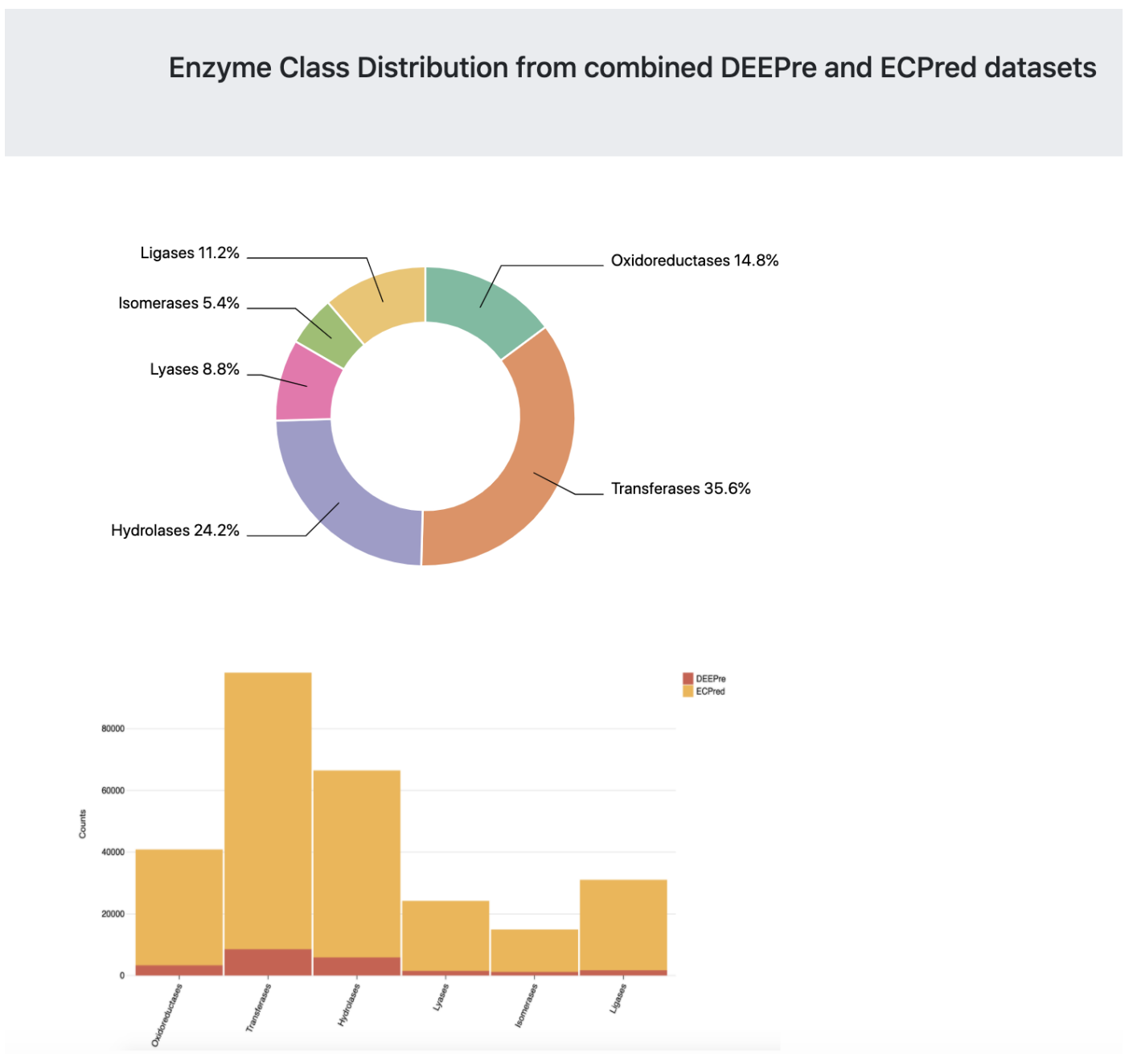
Data Science Pipeline

We utilized data from two different sources to train our models, DEEPre and ECPred datasets. The DEEPre dataset consists of roughly 44,000 protein strings in which the total counts of enzymes and non-enzymes are balanced. The amino acid (AA) lengths in each category range from 50 to 4,900, with a median length of 382 for enzymes and 286 for non-enzymes. Less than 5% of the dataset had sequence lengths of greater than 1,000 AA. Considering this, we acknowledged that in the event we face performance or scaling issues, some data exclusions might be necessary. Our first candidates for any data exclusions to mitigate performance issues would be the data of longer lengths (greater than 1000 AA) which would result in exclusion of approximately 5% of the data in DEEPre. The ECPred dataset (approximately 253,000 AA) included some sequences of longer lengths, up to a maximum of approximately 35,000 AA. Despite these outliers, the median length was 346 AA and the 75th percentile was 472 AA. As in the case of the DEEPre dataset, we acknowledged that the AA sequences of length greater than 1,000 could likely be excluded in our final model.

Our data pipeline began with consolidating and preprocessing the enzyme data. After loading fasta files into the singularity container, we utilized the .npz output file for additional downstream tasks. We then performed a binary classification to segment the data into non-enzymes and enzymes. The enzyme data was then used to predict the first level of respective enzyme classes.


DEEPre Data
Scrape and preprocess protein data
ECPred Data
Transform text data to fasta files, capture labels in json file
Load fasta files into TAPE/ESM-1b
Utilize NPZ output file for downstream tasks
Binary Classification
Non-enzyme
Enzyme
Predict Enzyme Class (1-6)
Partition data by class
TAPE/ESM-1b (PyTorch) Model
Python Script

Also exploratory data analysis led us to anticipate and resolve two identified potential data hindrances. First, our protein data did not have inherent features to start, as such, we utilized an n-gram word embedding to generate a simplified feature and used Principal Component Analysis to estimate preliminary model performances and confirm that our data would form expected clusters. Second, our data was not symmetric, in example, each class of an enzyme did not have an equal amount of protein strings contained within it. Through visualizations and descriptive statistics, we identified the amounts of data in each class and subclass and evaluated the distribution.


Enzyme Class Distribution from combined DEEPre and ECPred datasets

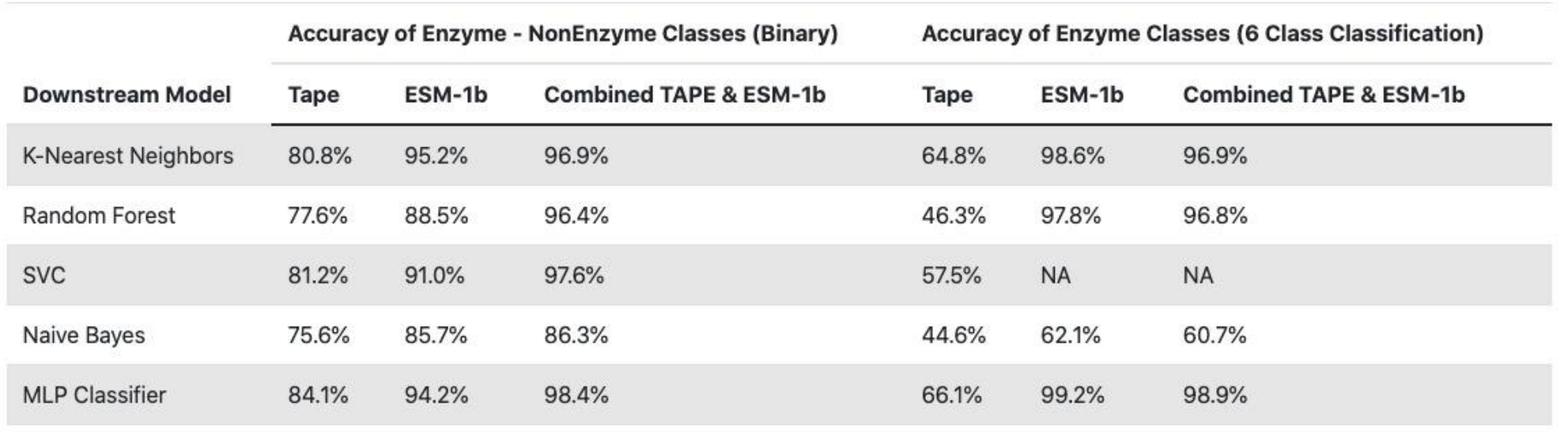
Final Solution

In designing the solution architecture, our multi-class classification solution developed into a two-fold pipeline. The first pipeline (BERT models) produced the features from our text or amino acid chain of letters, while the second pipeline (downstream models) performed the predictions.

Our solution architecture incorporated the use of San Diego Supercomputer Center's (SDSC) Expanse. All of the BERT models and feature engineering scripts were migrated into a singularity container in order to seamlessly run on the Expanse infrastructure. This was key not only in training our model, but also enabling an endpoint for the end user to leverage their own training data by giving them access to this container. The singularity container contained the environment files written in YaML, so that the environment could be recreated, as well as easily duplicated in the event the container is no longer leverageable. The environment files improved the setup time for the environment within the container in addition to keeping every workspace synchronized with the packages and libraries utilized. Two separate singularity containers were created for our use by Martin Kandes at the San Diego Supercomputer Center, one for TAPE and another for ESM-1b to be utilized in Expanse. This allowed for the models to run in their own environments, as each model contained separate and unique requirements, specifically the PyTorch package version differed for the two models.

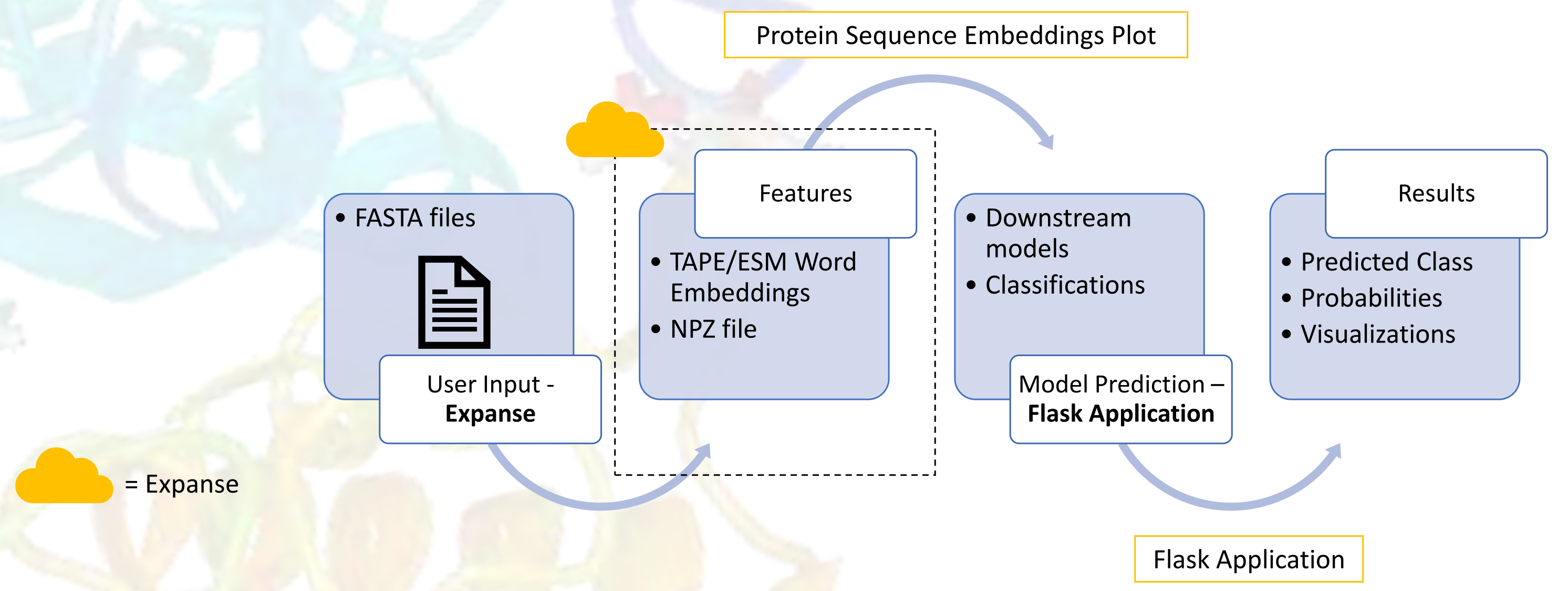
Once these features or protein embeddings are produced, they were fed into our downstream models in the second step of our pipeline, the classification process. These downstream models absorbed the features as input and returned the enzyme classification of the respective amino acid.

The algorithms that we've utilized for our downstream models include SVM, KNN, MLP Classifier, Random Forest, and Naive Bayes. All of these algorithms are used in order to yield the optimal accuracy for our enzyme classifications. With the combination of the feature engineered model and the downstream model, we were able to produce accurate enzyme classifications for our amino acids.



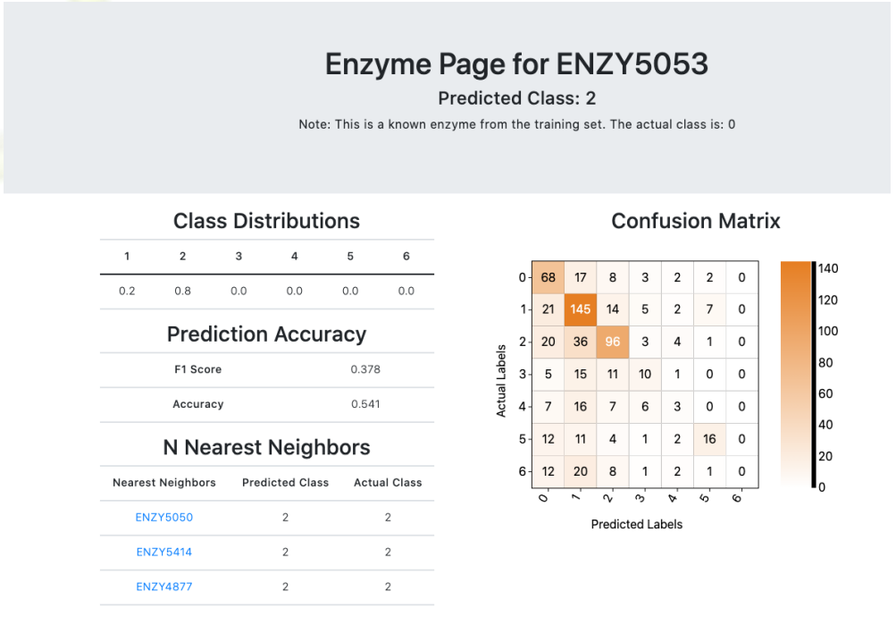
	Accuracy of Enzyme - NonEnzyme Classes (Binary)			Accuracy of Enzyme Classes (6 Class Classification)		
Downstream Model	Tape	ESM-1b	Combined TAPE & ESM-1b	Tape	ESM-1b	Combined TAPE & ESM-1b
K-Nearest Neighbors	80.8%	95.2%	96.9%	64.8%	98.6%	96.9%
Random Forest	77.6%	88.5%	96.4%	46.3%	97.8%	96.8%
SVC	81.2%	91.0%	97.6%	57.5%	NA	NA
Naive Bayes	75.6%	85.7%	86.3%	44.6%	62.1%	60.7%
MLP Classifier	84.1%	94.2%	98.4%	66.1%	99.2%	98.9%

Below is a description of our data product flow that enabled us to achieve our results:



Key Insights and Conclusion : EINSTEIN

In pulling together our end-to-end solution, we discovered findings relating to a range of items consisting of: most accurate downstream models, feature engineering, and training data optimization. In sum, MLP proved to be the most accurate downstream model, followed by KNN; TAPE & ESM-1b combined embeddings yielded the best downstream results. In trialing options to improve our models, we experimented with feature engineering and learned that, other than incorporating the word embeddings derived from BERT models, incorporating additional features did not improve the accuracy of our downstream models. We also discovered that the embeddings carried such significant weight in our model, that incorporating any additional feature engineering had little to no impact on the results. Instead of focusing on a narrow picture of the data, we wanted to broaden the lens and provide the user with as much data findings as possible. We also aimed to present insights about our training data within our application. This included PCA, t-SNE, UMAP and K Nearest Neighbors. These four modalities were leveraged to provide the end user with background as to what data our model was trained with. We also added F-1 scores to reflect accuracy values. Ultimately, we chose to present these results and related data points through discovery interviews with our end user and advisor. Following the direction of a domain knowledge expert allowed us to focus on presenting what results mattered as they pertained to enzyme classification.



Enzyme Page for ENZY5053

Predicted Class: 2

Note: This is a known enzyme from the training set. The actual class is: 0

Class Distributions

1	2	3	4	5	6
0.2	0.8	0.0	0.0	0.0	0.0

Prediction Accuracy

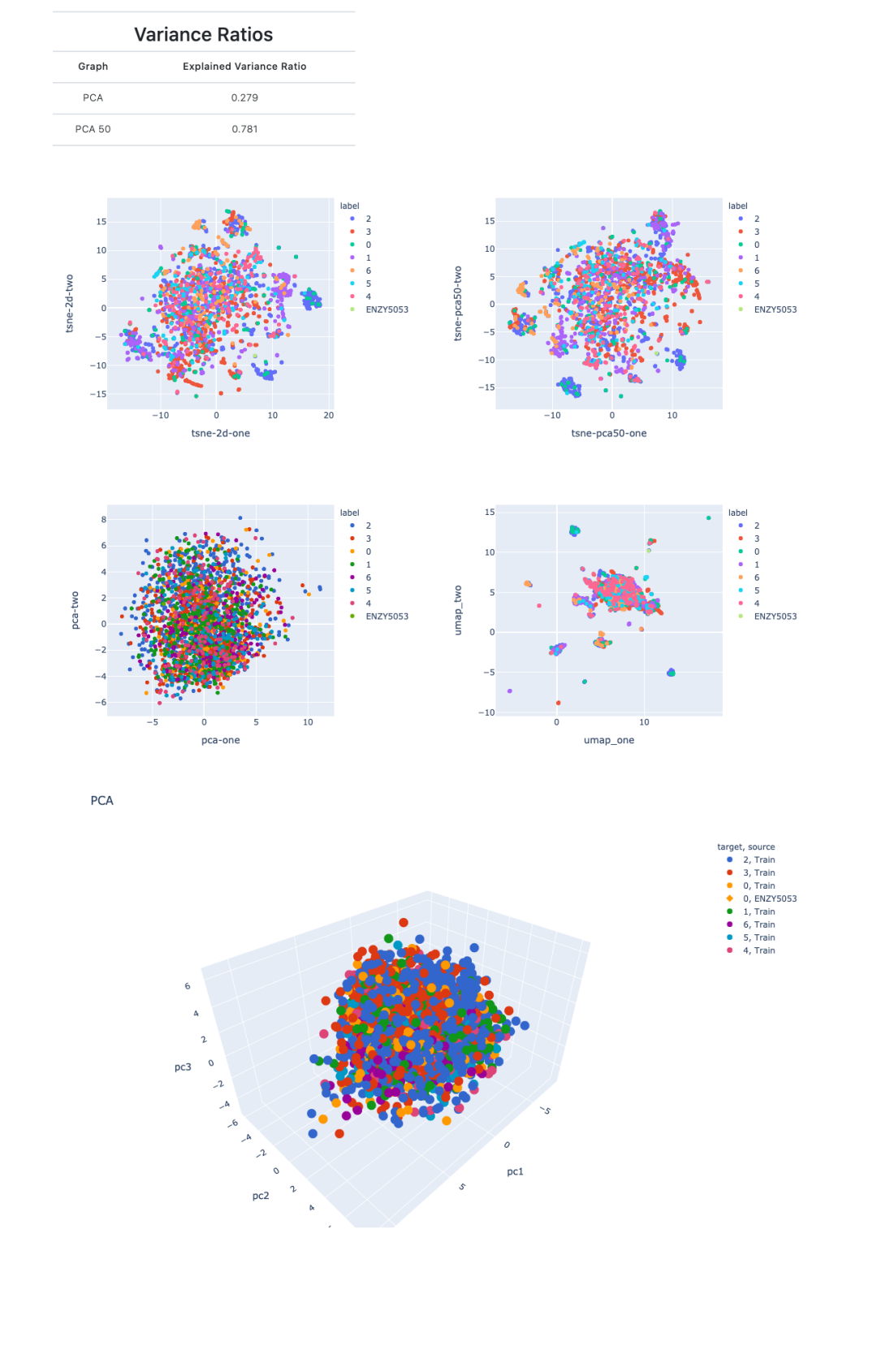
F1 Score	0.378
Accuracy	0.541

N Nearest Neighbors

Nearest Neighbors	Predicted Class	Actual Class
ENZY5050	2	2
ENZY5414	2	2
ENZY4877	2	2

Confusion Matrix

	0	1	2	3	4	5	6
0	68	17	8	3	2	2	0
1	21	145	14	5	2	7	0
2	20	36	96	3	4	1	0
3	5	15	11	10	1	0	0
4	7	16	7	6	3	0	0
5	12	11	4	1	2	16	0
6	12	20	8	1	2	1	0



We relied on visualizations to capture a complete picture of the data for the end user. Visualizations allowed us to communicate details about the training data we worked with to develop this model and also communicate details about a user's predictions. To inform the user about the data the model was trained on, we developed a dashboard that demonstrates the distribution of the enzyme classes. We also incorporated PCA, t-SNE and UMAP visuals to capture an estimate of the model performance and confirm that the data forms accurate clusters.