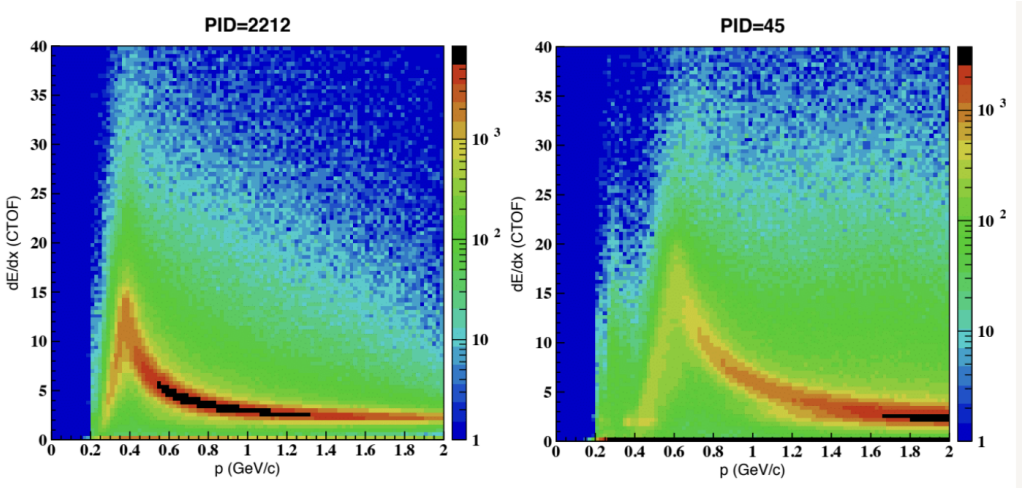
Nick Richardson
Dr. Bandara
Machine Learning
20 December 2021

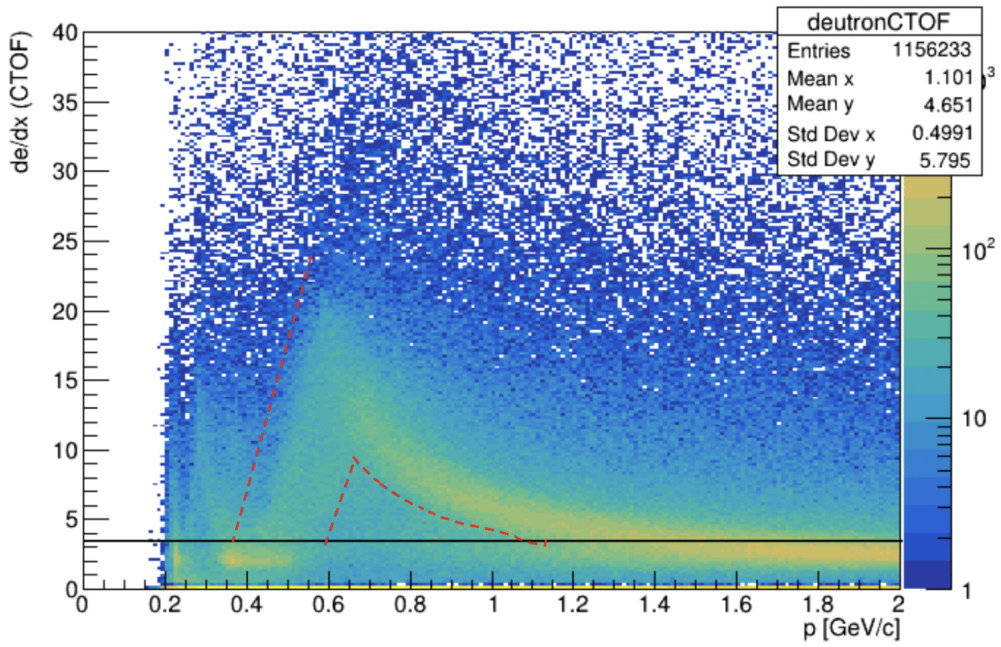
Particle Identification Refinement for CLAS12 Jefferson Lab Data

Introduction:

My senior capstone research project for my physics major entails performing an analysis of Deeply Virtual Compton Scattering off the deuteron to construct a Beam Spin Asymmetry, giving insight into the Generalized Parton Distributions of the deuteron. These GPD's are a powerful theoretical tool which when calculated properly tell how the three quarks are distributed spatially throughout the nucleon. When constructing the BSA, it is extremely important to only include true DVCS events, which is the reaction that is sensitive to the GPDs of the nucleon. To study this, we are using data taken at Jefferson Lab, the particle detector where the experiment is carried out. The Jefferson Lab detector reconstructs the data using algorithms and puts labels on particles such as deuterons, protons, electrons, etc. However, the algorithm is not perfect and we were noticing that occasionally a proton could be misidentified as a deuteron which would be detrimental to the analysis because we could be including events that are not DVCS in our analysis. To get around this, one of our collaborators recommended using a trend he discovered in the de/dx vs momentum to further verify the particles. He said we could simply eyeball a cut on the plot, anything above the line being deuteron, and below it would be the protons.



This is the data our collaborator used to demonstrate the separation of PID 2212 (the proton) and PID 45 (the deuteron) based on de/dx vs momentum plots.



This image is the sample cut that our collaborator proposed for us to make.

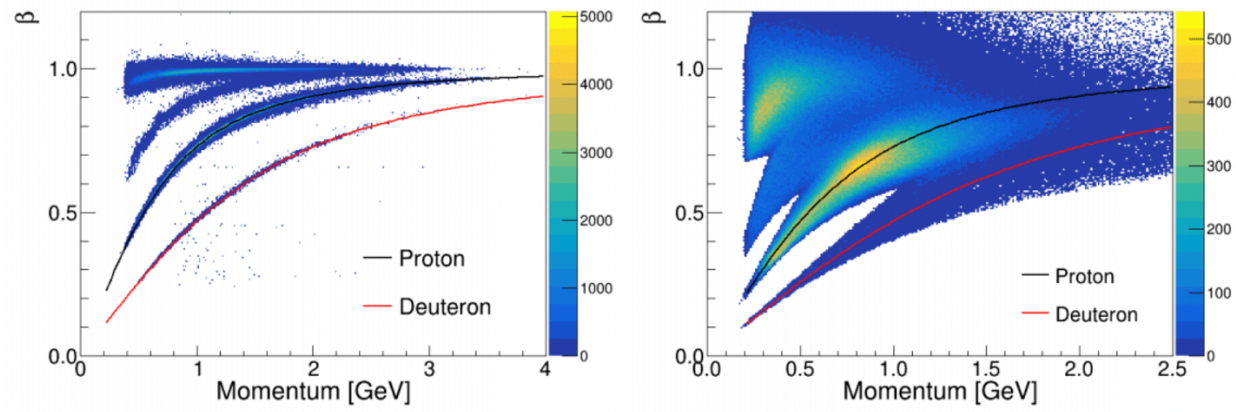
Over the summer, I tried to design cuts on the data to separate the two types of particles by just eye-balling it. However, we quickly learned that there is too much overlap and the data is not as well separated as we originally thought. This is what inspired us to try to make a machine learning model to solve this classification problem.

The input to my models will be different variables of the deuteron and proton, namely the de/dx in the CTOF, de/dx in the CND, and the momentum of the particle. I am using Monte Carlo data that I generated with an event generator that calculates physically allowable kinematics, and then the kinematics are put through a simulation to essentially run the experiment and give us output identical to what the detector would give. The output will be whether the particle is a proton or a deuteron. I will be using K-Nearest Neighbor, Random Forest Classification, and Logistic Regression to create my models.

Related Work:

From a review of literature across both in print and online physics journals, I was unable to find any other research group using machine learning to perform particle identification and verification. However, as I mentioned previously, there is an algorithm that Jefferson Lab uses to put labels on the particles when it creates the data that we physicists perform our analysis on. One of the primary ways that Jefferson Lab identifies particles is through trends in the different properties of each particle. One of the biggest indicators of identity of the particle to discern the

difference between protons and deuterons that Jefferson Lab exercises is in the Beta vs momentum graph.



The above plot[1] shows the trends of different positive particles Beta vs momentum plots. The left plot shows particle hits in the CTOF part of the detector and the right figure shows particle hits in the CND part of the detector.

Based on these trends, Jefferson lab groups particles based on which track or trend line they follow most closely.[2] While at lower momentum, there is more distinction, at higher momentum, the particles behave less cleanly and there is more ambiguity when it comes to classifying the particles.

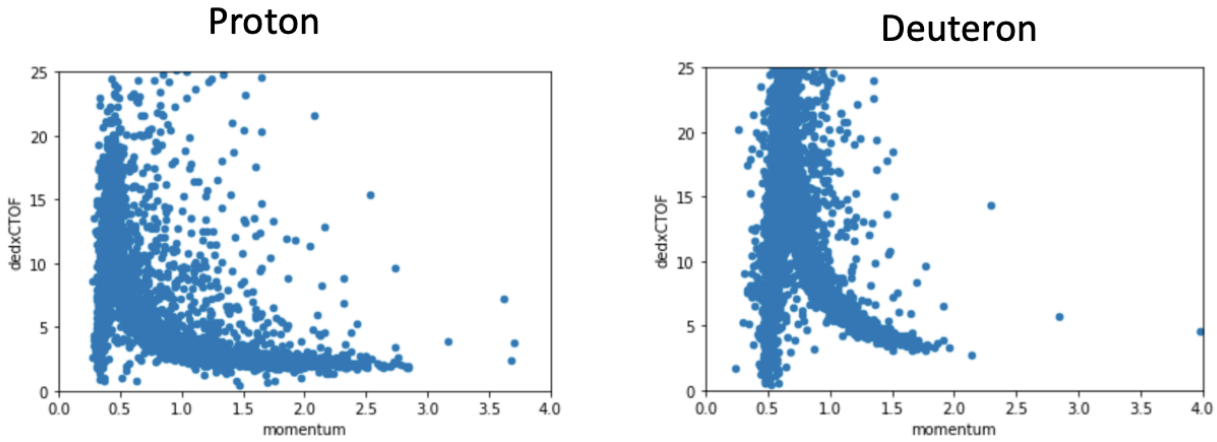
Data:

From the Monte Carlo data that I generated I ended up with 5750 data points for deuteron and a much larger set for the proton (however, I would train the model on an equal number, randomly selecting the same number of proton events). I needed to eliminate any data point that had a de/dx CTOF or de/dx CND equal to 0 because this is an unphysical state and in the analysis we discard those either way so this is not adding any bias to the model. The way the detector works, there must be a value for the de/dx in the CTOF but de/dx in the CND can be a null value, since this part of the detector is not guaranteed a deuteron or proton hit. After I removed events that had de/dx = 0, I ended up with 5705 events for the deuteron that had a non-zero de/dx in the CTOF. Of those, there were 3621 events that had a hit in the CND giving us a value for de/dx CND. I then used an 80:20 split generated randomly between the training set and test set.

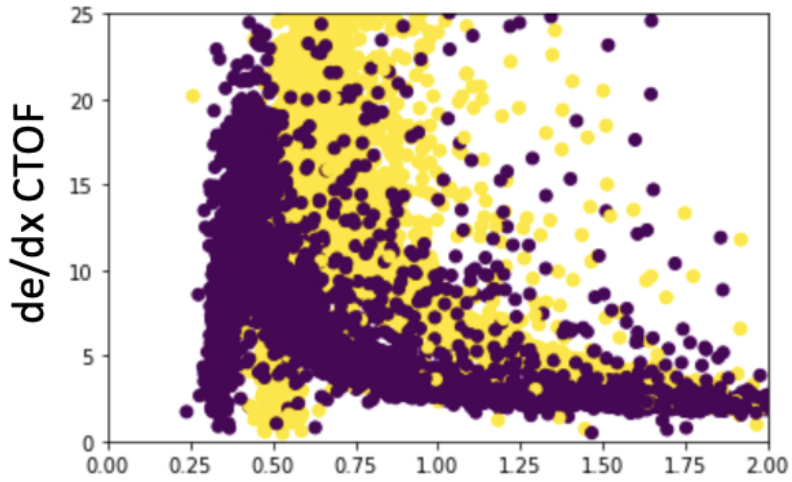
---

[1] Higinbotham et al. (2021). Letter of Intent: Exploring fundamental properties of 3He through the 3He (e, e' d) process in CLAS12. *Applied Physics Letters*.
[2] https://clasweb.jlab.org/wiki/index.php/CLAS12_EventBuilder

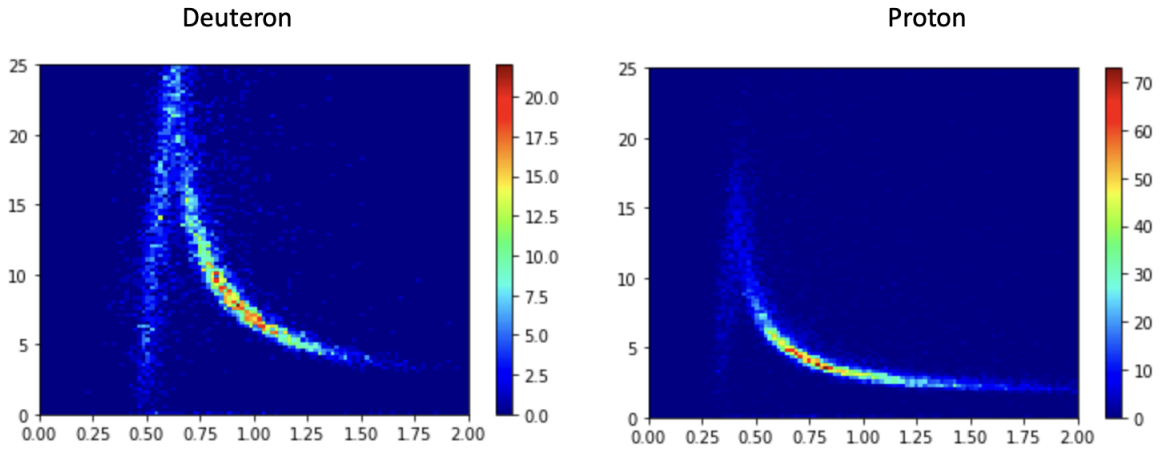
Below is data depicting the de/dx CTOF vs momentum.



MC data of de/dx vs momentum in the CTOF side by side

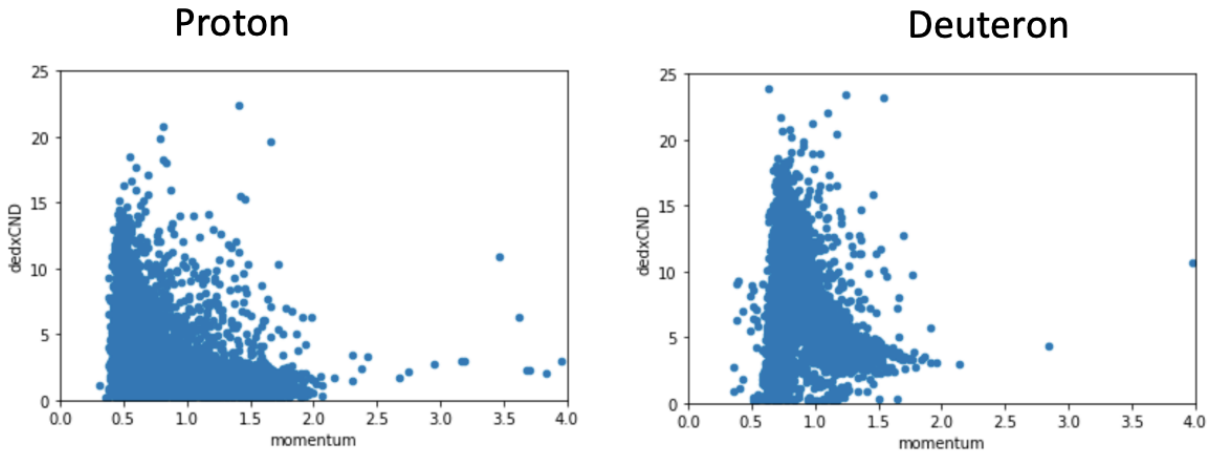


MC data overlapped. Purple is the proton and yellow is the deuteron. The overlap is a bit misleading, because the overlap does not have very many points, whereas we can see the intensity of the points in the figure below.

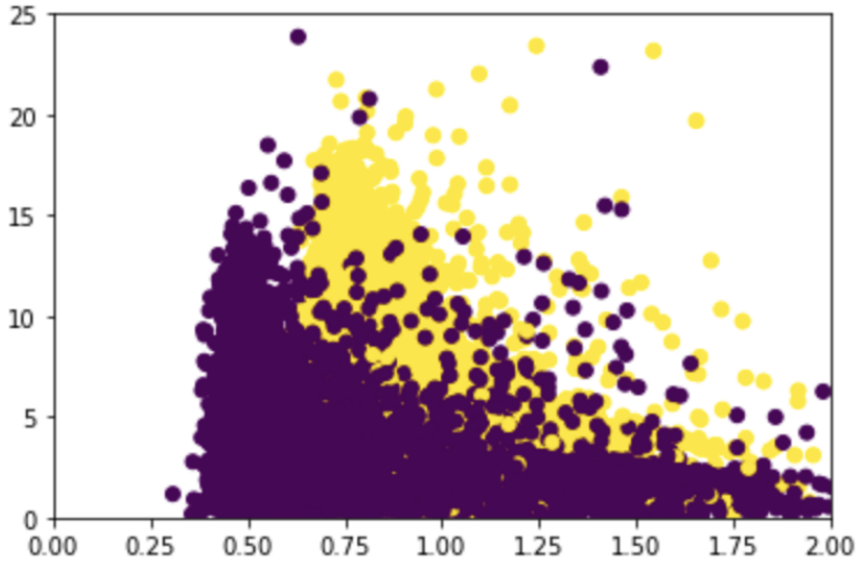


This is histogram data of dedx (CTOF) vs momentum. You can see that there is more separation and less overlap between the deuteron and proton in the region where there are the most particles between 0.5 and 1.25 momentum.

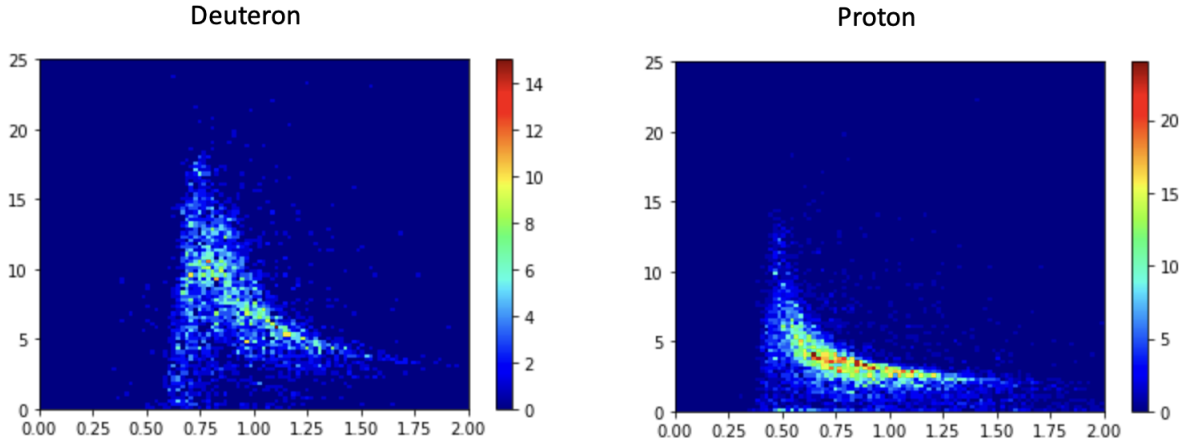
Next, I will depict the data of de/dx CND vs momentum.



MC data of de/dx vs momentum in the CND side by side



MC data overlapped. Purple is the proton and yellow is the deuteron. Once again, the overlap is misleading because in the regions where there is the highest intensity of particles, there is minimal overlap, so this does not hurt the model.



This is histogram data of dedx (CND) vs momentum. As you can see, there is less separation and more overlap in these variables between the deuteron and proton.

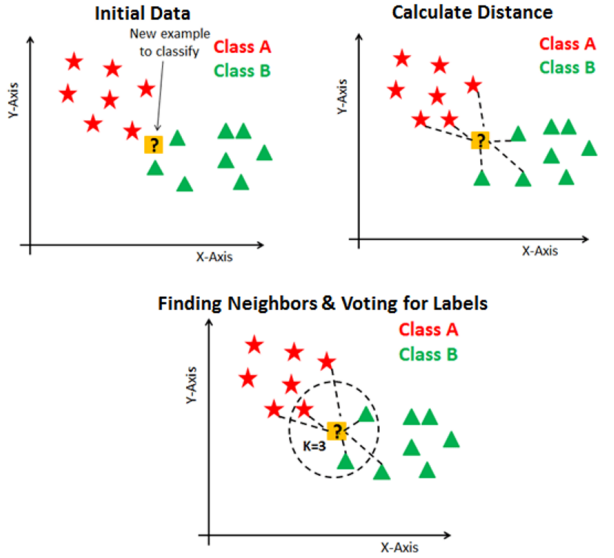
The three features that I used from the Monte Carlo data was de/dx CTOF which every data point will have, de/dx CND which an event may or may not have, and momentum which is guaranteed as well. For the KNN model, I did use the sklearn StandardScaler, which standardizes the feature by removing the mean and scaling to unit variance.[3] I did this because of the large difference in

[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

range between the de/dx feature which can be from 0 to 30, and the momentum which can only be from 0 to 2.

Methods/Models

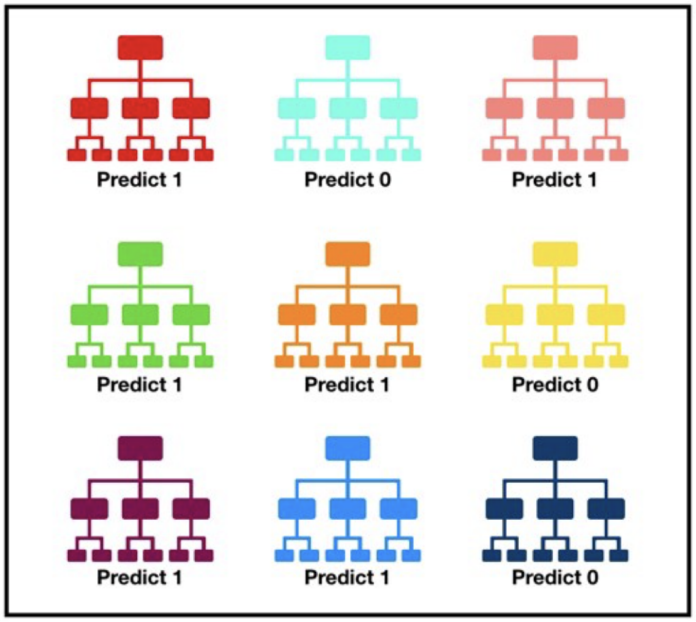
The three machine learning models that I worked on were K-Nearest Neighbors[4], Random Forest[5], and logistic regression[6]. K-nearest neighbors is a technique to perform classification by calculating the distance between the point of interest with the training data set. Once these distances are calculated, the algorithm looks for the K- nearest neighbors and whichever output has the most out of the K is how the model classifies that test data.



This is an example of using K=3 in the KNN algorithm. Since there are more green triangles within the 3 nearest neighbors, that is how that point is classified.

Next, I used the Random Forest classifier. This makes use of decision trees. Specifically in Random Forest, there are many decision trees that act as an ensemble. It is important to note that the large number of relatively uncorrelated models (trees) operating as the ensemble will outperform any of the individual trees. Below is a picture which illustrates this idea further.

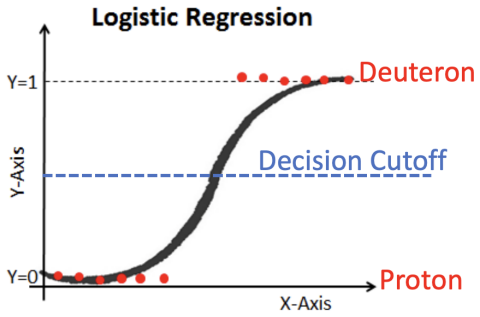
[4] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[5]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html



In this example of Random Forest, there are 9 different decision trees that each make a prediction on the test data. There are 6 trees that classify this as a 1 and only 3 that classify it as a 0, making the prediction a 1.

The last model I used was logistic regression. This model fits a sigmoid function to the data. For my case, a "1" represented a deuteron and a "0" represented a proton. During the training of the model, the different weights are modified to create the best model.

$$f(x) = \frac{1}{1+e^{-(x)}}$$



$x = w_0 + \text{dedxCTOF}*w_1 + \text{dedxCND}*w_2 + \text{momentum}*w_3$

In this figure, the formula on the left is the sigmoid function which produces the curve on the right. The "x" of the exponential function is a sum of the features multiplied by the weights that the model fits. During the testing of this model, if the model has a prediction > 0.5 that is considered a deuteron, while a model prediction <0.5 is considered a proton by the model.

Evaluation

I chose KNN at first because of the large distinction in the data between the proton and deuteron which I believed would create a good model since the data is relatively well separated. Below is a table of the accuracies of some of the different models I used. The parameters that I tuned were the number of neighbors, switching between 3 and 5, the variables that I was using as features, and finally the scaling of the data. I had some models that would only use de/dx CTOF and momentum, and some models that would use de/dx CTOF, de/dx CND, and momentum. Recall that this is because some particles do not necessarily need to have a hit in the CND part of the detector and therefore the value would be null. In the model where I did use de/dx CND, I discarded any particle that did not have a value for that, so my training and test set for these models was slightly smaller.

# Neighbors	De/dx CTOF	De/dx CND	Momentum	Accuracy (no scaling)	Accuracy (scaling)
3	yes	no	yes	93.8	94.7
5	yes	no	yes	93.8	94.7
3	yes	yes	yes	93.8	94.7
5	yes	yes	yes	94.9	97.4

As I described in the introduction, the problem I am really after is how I can find misidentified deuterons in the data that are really protons. Because of this, it is very important to look at the confusion matrix to see how many protons my model is classifying as a deuteron because this will hurt my analysis because I am including non-DVCS events. Therefore, a good model would much rather incorrectly identify deuterons as protons rather than classify protons as deuterons. By looking at the confusion matrix we can verify whether or not this is happening.

Accuracy = 97.5	Predicted Label		
Actual Label		Proton	Deuteron
	Proton	324	13
	Deuteron	4	354

This confusion matrix is of the KNN model that produced a 97.5% accuracy on the test data. This model truly is performing well not only because of the accuracy but also because we can see that there are only 13 protons classified as deuterons which is what we are trying to limit happening. This means if we were to implement this model into our analysis codes, we are using 13 protons that we think are deuterons which is not good for our analysis.

The next algorithm I used was Random Forest Classification. For this, I also created two separate models with slightly different features, one that included just de/dx CTOF and momentum, and a second that also included de/dx CND. Once again, the model that included de/dx CND had a slightly smaller training set. Below is a table documenting the accuracy of each of these models.

Model	De/dx CTOF	De/dx CND	momentum	Accuracy
Random Forest	yes	no	yes	94.3%
Random Forest	yes	yes	yes	98.2%

The decision matrix looks like the following. Once again, we can see that there is a tendency to classify protons as deuterons which is detrimental when we are doing the physics analysis. However, with an accuracy as high as this one, we can assume that this will help in the analysis even though we are including some protons as deuterons it is undoubtedly better than without this machine learning model.

Accuracy = 98.2	Predicted Label		
Actual Label		Proton	Deuteron
	Proton	361	10
	Deuteron	3	363

Lastly, I tried logistic regression because using foresight and recognizing that I will need to implement this into my analysis codes, I do not want to use a slow algorithm like KNN when I am doing my physics analysis. Something like logistic regression is much easier to implement because I only need to extract the different weights for the variables and use that in my analysis.

Not only is this extremely easy, it won't slow down the analysis code at all because it is merely a simple mathematical operation.

I once again implemented two different models for this, with just modifying the amount of features I was using. For the model that only uses de/dx CTOF and momentum, I received an 81.5% accuracy on the test set while in the model that also uses de/dx CND, I got a 91.7% accuracy. The confusion matrix on the right shows the logistic regression with 91.7% accuracy. We can see that there is a tendency to classify actual protons as deuterons. It classified 41 protons as deuterons which means if we implement this in our analysis codes we are including many non real DVCS events since the particle is actually a proton.

Model	De/dx CTOF	De/dx CND	momentum	Accuracy
Logistic Regression	yes	no	yes	81.5%
Logistic Regression	yes	yes	yes	91.7%

Accuracy = 81.5%	Predicted Label		
		Proton	Deuteron
Actual Label	Proton	175	39
	Deuteron	42	182

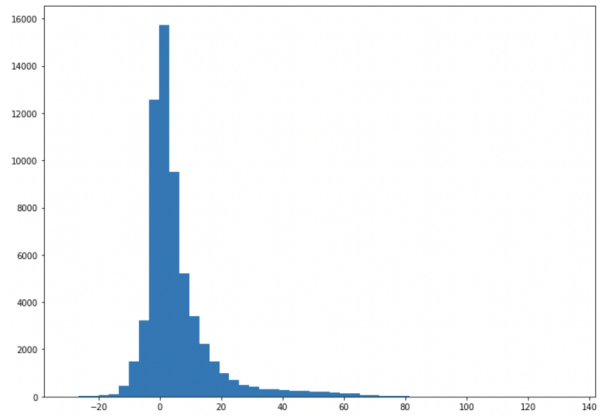
Accuracy = 91.7%	Predicted Label		
		Proton	Deuteron
Actual Label	Proton	330	41
	Deuteron	20	346

I did additional evaluation of my model using real data that we received from Jefferson Lab. There is a variable that Jefferson Lab creates for each particle called Chi2PID which is a measure of how confident Jefferson Lab is in how they identified a particle. For example a large Chi2PID means that Jefferson Lab algorithms for particle identification were not very certain in that identification, while a smaller Chi2PID shows much more certainty. I wanted to do a test to see if there is a correlation between misidentified deuterons and their associated Chi2PID value. In other words, I would take real data and run it on my model and see for particles my model calls a proton but the data calls a deuteron and see if those particles have larger Chi2PID.

I did this test with the Logistic Regression model taking the weights and plugging into the sigmoid function and using 0.5 as my cutoff for identification. Using the first Logistic Regression model which uses only the de/dxCTOF and momentum for the classification I got an average of 7.1 Chi2PID for model predicted deuterons and 15.7 for model predicted protons. This does show that there is a correlation and that there is a high probability that the Jefferson Lab algorithm may be misidentifying protons as deuterons which is evident because those are the

particles that they are less confident in using their current techniques for identifying particles. Below is a histogram showing the distribution of Chi2PID for model predicted deuterons on the left and model predicted protons on the right.

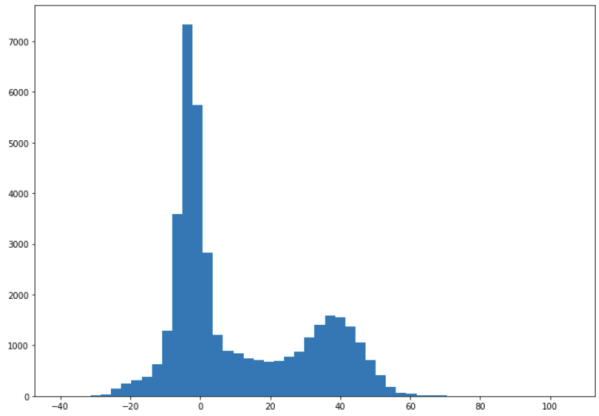
Chi2 of Model Predicted Deuterons



Chi2PID

Average 7.1

Chi2 of Model Predicted Protons

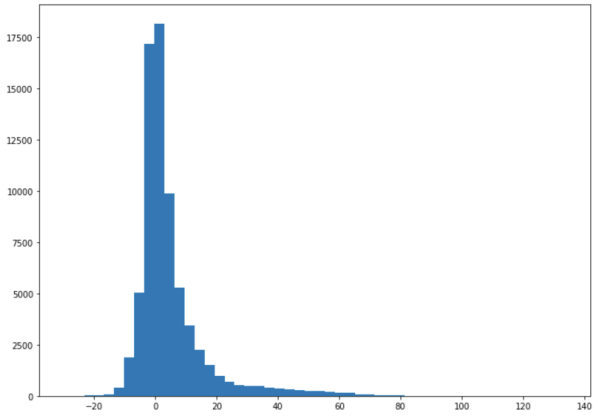


Chi2PID

Average = 15.7

Using the Logistic Regression model that uses de/dx CTOF, de/dx CND, and momentum I received even more promising results, depicted below. Model predicted deuterons had an average of 6.9 Chi2PID and model predicted protons had a Chi2PID average of 18.6. Looking at the spread of all of the Chi2PID you can see the large shoulder for the Chi2 of the predicted protons at about 40. A Chi2PID of 40 is extremely large and shows that the particles that Jefferson Lab had very little confidence in were mislabeled protons according to our model.

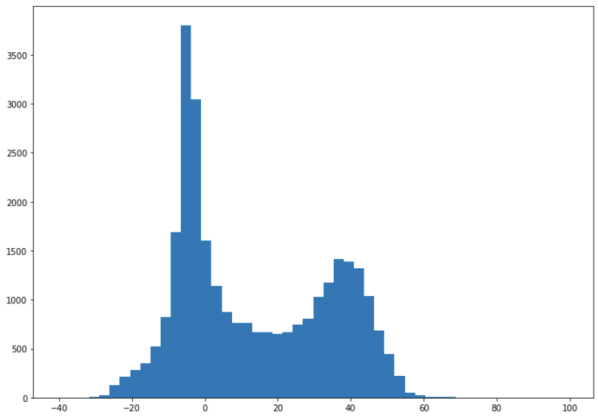
Chi2 of Model Predicted Deuterons



Chi2PID

Average 6.9

Chi2 of Model Predicted Protons



Chi2PID

Average = 18.6

Conclusion:

In conclusion, I was able to succeed in creating an accurate model to help solve the problem of mislabeled protons as deuterons in the data. Clearly, Random Forest was the best model. I think this is the case because the ensemble of subtrees was able to very accurately model the complex data because, while the data was Monte Carlo, there still were some random points that did not fit the basic trend that may have hurt the other models. KNN was another great algorithm because by looking at the nearest 3 or 5 nearest neighbors, the outliers in the data became less important and those big bunches of data that had less overlap between the proton and deuteron which I showed in the 2D histograms became more important. The Logistic Regression performed the least well because of the many outliers which I think skewed the weights of the data.

This work has many implications with regards to my research. We are always interested in ways we can enhance our particle identification. By using these models for particle selection, we can be even more confident in how we are determining particles for DVCS events moving forward. While I have not been able to implement this in the real analysis codes yet, I am cautiously optimistic that they will drastically improve our results. There is still work to be done however. We are still in discussion with our collaborators about which, if any, other properties of the data we can use to enhance the model further besides de/dx CTOF, de/dx CND, and momentum.

References:

CLAS12 EventBuilder. CLAS12 EventBuilder. (2020, July 2). Retrieved from https://clasweb.jlab.org/wiki/index.php/CLAS12_EventBuilder

Higinbotham et al. (2021). Letter of Intent: Exploring fundamental properties of 3He through the 3He (e, e' d) process in CLAS12. *Applied Physics Letters*.

LogisticRegression https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

KNeightborClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Matplotlib https://matplotlib.org/

Numpy https://numpy.org/

Pandas https://pandas.pydata.org/docs/

RandomForestClassifier.
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Seaborn. https://seaborn.pydata.org/

Standard Scaler.
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html