CS 6350 - Machine Learning

Final Project Report

Rebecca Corley — GitHub Link [1]

[1]Department of Physics and Astronomy, University of Utah

December 20, 2023

1 Introduction

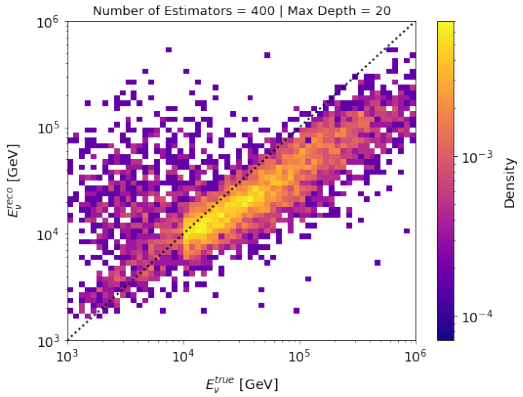
1.1 Background

Located within the Antarctic ice at the South Pole, the IceCube Neutrino Observatory is the world's largest neutrino detector. Neutrinos are elementary particles - subatomic particles that are not composed of other particles. When a neutrino interacts with an ice molecule, a secondary shower of particles are produced. These secondary particles travel faster than the speed of light through ice producing Cherenkov radiation, emitting a blue light picked up by the detector cameras, which is known as an "event." The detectors pick up the observables of direction, energy, and the pattern of light produced by an event. Neutrino interactions produce different patterns, tracks and cascades, corresponding to electron and muon neutrinos respectively. An event's direction, energy, and pattern allow us to work backwards to understand more about prospective astrophysical neutrino sources and what type, or flavor of neutrino, among other interesting physics. This process of working backwards is known as "event reconstruction."

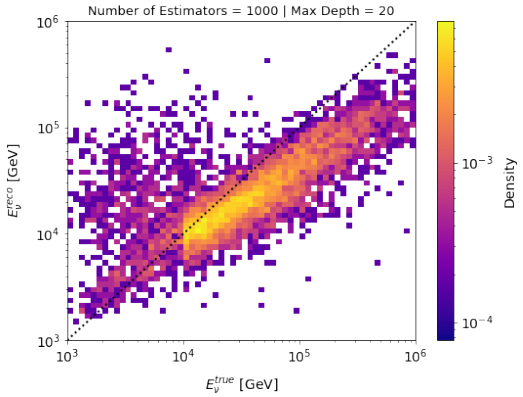
1.2 Motivation

Due to the influx of signals, about 2,600 signals per second, finding interesting events is like finding a needle in a haystack, which has made implementing machine learning algorithms an invaluable tool for particle physics experiments. Ongoing work is focused on improving reconstruction biases and detector resolution. In this work, we are developing a Convolutional Neural Network (CNN)-based reconstruction method for track-like neutrino events. This project is based on existing work for cascade-like events [1]. However, there are additional challenges to overcome in track reconstruction. Muons escape differently than electrons that form cascade events, making it difficult to collect all of the information required for track reconstruction. Monte Carlo (MC) based information will be used to train the neural network. Reasonable agreement between data and MC simulated events is expected. The expected outcomes of this project are to identify how well the CNN-based event reconstruction works for track-like events, and to compare and review machine learning reconstruction methods for the IceCube Neutrino Observatory.

This is a contribution to a larger long-term project that I am working to develop a CNN for track-like neutrino events, and compare the reconstruction results to other machine learning models, including Random Forest and Boosted Decision Tree. For the purpose of this course, I develop a



(a) number of estimators = 400 and max depth = 20.



(b) number of estimators = 1000 and maximum depth = 20.

Figure 1: 2D histograms comparing true and reconstructed method using random forest regressor. The dashed black line is the ideal center densities.

Random Forest model and vary the algorithm parameters and investigate bias and variance. The code for this work can be found on my private GitHub Repository. [1]

2 Data and Preliminary Work

My first steps were exploring the IceCube server environment and preforming exploratory analysis on the data. The IceCube collaboration works on a particular server environment based in Madison, Wisconsin, known as IceTray. IceTray is the framework responsible for managing interactions between users and IceCube-specific data, which is in the format of I3 files. The I3 files are used for both experimental acquired data and for Monte Carlo simulated data, both of which undergo serial processing of the data. In this project, I3 files were converted to numpy files using a separate script. Data in IceTray is accessed using a remote connection and edited using vim through the terminal.

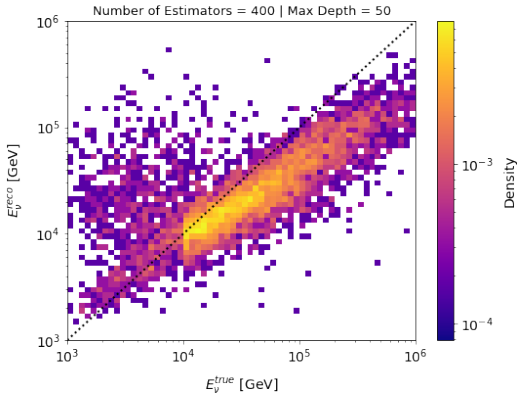
Once I learned to navigate the IceTray environment, I began working with IceCube data and generating plots to visualize different particles and observables. Provided the length constraint, these were intentionally left out. The data was generated via a Monte Carlo simulation. I investigated data of muons which are produced from muon neutrinos in track-like events. I explored observables of energy, spatial distribution by zenith and azimuthal angles, and interaction types of charged-current or neutral-current events. Next I implemented the Random Forest algorithm, which is discussed in further detail in the next section.

3 Algorithm Architecture and Design Choices

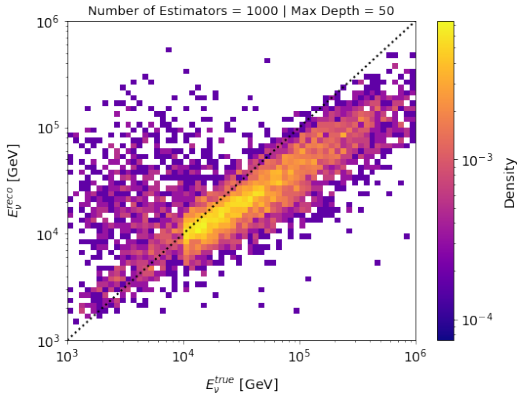
This code can be found on my GitHub. **Please note: Due to collaboration guidelines, I cannot publish data, so code will not run. This was approved by Dr. Shandian Zhe in writing. Please reach out if there are further concerns.**

I chose to use the random forest algorithm as part of this work because it seemed to be a

[1] `https://github.com/corley-rebecca/CS-6350`



(a) number of estimators = 400 and max depth = 50.



(b) number of estimators = 1000 and maximum depth = 50.

Figure 2: 2D histograms comparing true and reconstructed method using random forest regressor. The maximum depth was increased to 50. The dashed black line is the ideal center densities.

natural starting point as it is a widely used and versatile technique in machine learning. A random forest represents an ensemble learning approach that combines predictions that come from several decision trees to generate reconstructions that are more precise and consistent. This method falls under supervised learning algorithms, applicable to tasks involving classification and regression.

Using the scikit-learn random forest regressor, [2] I built my machine learning model for to reconstruct muon neutrino energies. I focused on varying the number of estimators and maximum depth. The number of estimators was varied at intervals 400, 800, and 1000, and the maximum depth was varied at intervals 20, 50, and 100. I exempt 800 in these figures to save space. The parameters of the algorithm were evaluated using bias and variance. In Figure 1, I show the two extreme values, 400 and 1000 estimators, for maximum depth equal to 20. The black dashed line through the middle is where I expect the highest density to center - it should be linear. A careful eye will note that Figure 2b is slightly below this line. Next the maximum depth was increased to 50 and again ran for estimators 400, 800, and 1000. Again, it appears that the highest number of estimators causes more of the data to fall below the black line, which is shown in Figure 2. Lastly, the max depth was increased to 100, as seen in Figure 3.

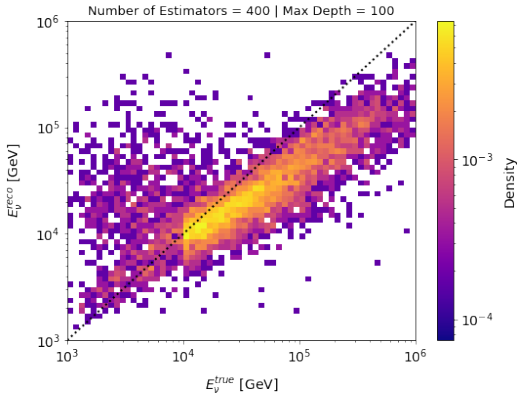
4 Reconstruction Performance

I generated plots for the true value vs the ratio of the true to reconstructed value. This allowed me to investigate the bias, resolution or variance, and medians of each run.

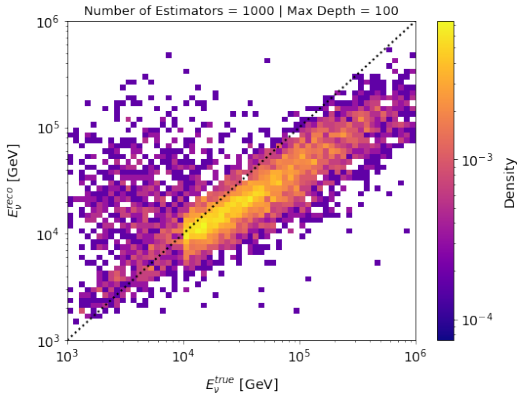
4.1 Energy Bias

Bias is defined as the model's limitation or inability, leading to discrepancies or errors between the model's predicted or reconstructed value and the true value. These discrepancies represent the differences between predicted and expected values. Bias arises from systematic errors from incorrect assumptions within the machine learning process. If we let the true value be Y and the estimator of Y is $\hat{Y}$, and $E(\hat{Y})$ is the expected value of the estimator, then bias is calculated using:

$$Bias(\hat{Y}) = E(\hat{Y}) - Y$$



(a) number of estimators = 400 and max depth = 100.



(b) number of estimators = 1000 and maximum depth = 100.

Figure 3: 2D histograms comparing true and reconstructed method using random forest regressor. The maximum depth was increased to 100. The dashed black line is the ideal center densities.

Looking at Figure 4 and Figure 5, we can see the bias as the distance from zero. At higher energies there is a higher bias where more assumptions were made to generate the target function, and at lower energies the bias is lower where fewer assumptions were made and it is closer to the training data at zero. In the negative region, where the reconstructed energy is less than the true energy, this is deemed as under fitting.

4.2 Energy Resolution

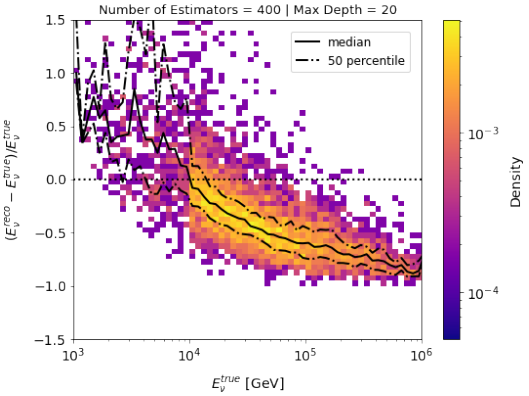
Variance quantifies the dispersion of data around its average position. We also refer to variance as resolution. In machine learning, variance denotes the extent to which a predictive model's performance fluctuates upon training with various subsets of the training data. It specifically gauges the model's variability and its adaptability to different subsets within the training data, which shows its sensitivity and capacity for adjustment to new training subsets.

The spread of the distribution appears symmetric, which could imply that our statistics are more Gaussian. In Figure 4 and Figure 5, we see a higher variance towards lower energies and a lower variance at higher energies.

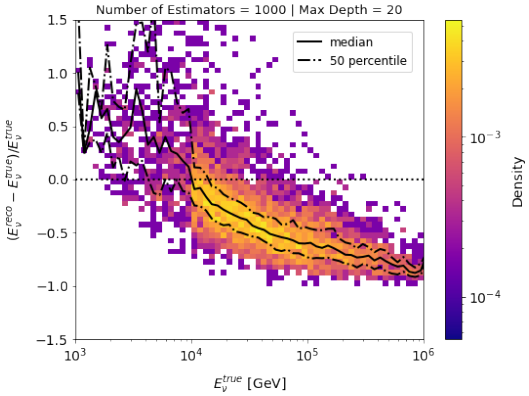
To further investigate the bias and resolution, I overlay medians of each ratio plot seen in Figure 7 where I show two of these figures. I found this result to be quite shocking as I expected more variance in each plot.

5 Concluding Remarks and Future Steps

I think that the random forest was implemented correctly, but I look forward to optimization and improvements to create better agreeance between the predicted and true data. I am surprised that the medians in each plot are so similar despite ranging the parameters. The sparsity in the lower energy regions of Figure 4 and Figure 5 is likely due to the need for increased statistics by generating significantly more Monte Carlo events, which is computationally expensive. I would also adjust the training verses validation data set sizes. After improvements on the random forest method, I will implement a boosted decision tree alongside a CNN (which is currently under development) to compare the bias and resolution of each technique to determine which machine learning method

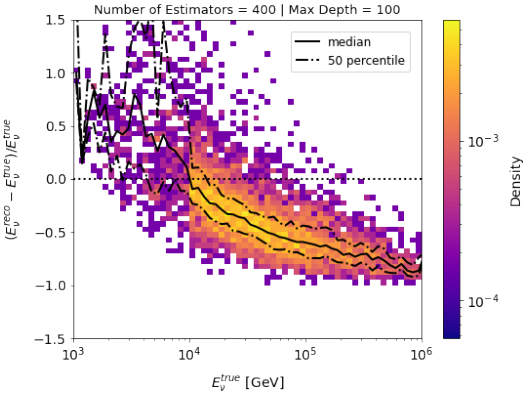


(a) number of estimators = 400 and max depth = 20.

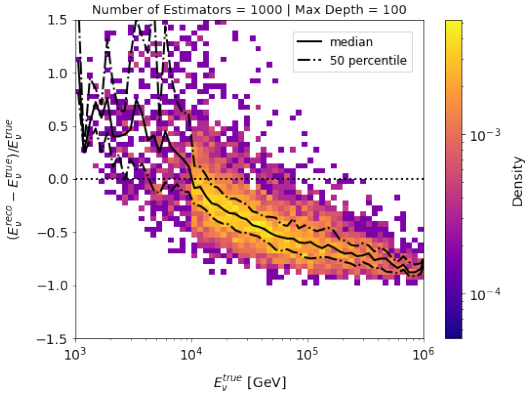


(b) number of estimators = 1000 and maximum depth = 20.

Figure 4: Graphs showing the true value against the ratio of reconstructed to true value for maximum depth of 20.

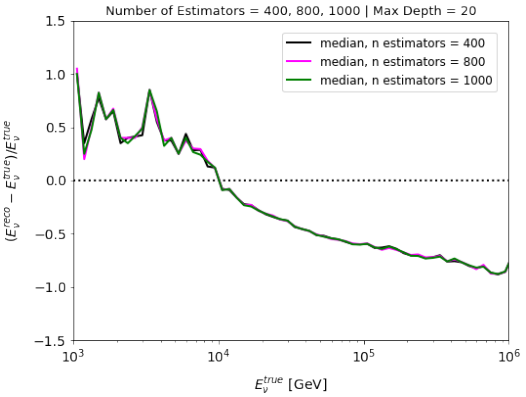


(a) number of estimators = 400 and max depth = 100.

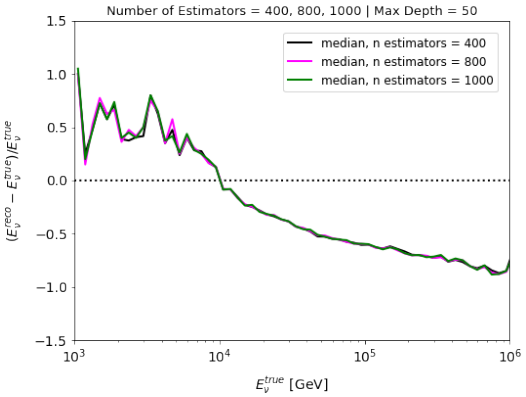


(b) number of estimators = 1000 and maximum depth = 100.

Figure 5: Graphs showing the true value against the ratio of reconstructed to true value for maximum depth of 100.



(a) number of estimators = 400, 800, 1000 and max depth = 20.



(b) number of estimators = 400, 800, 1000 and maximum depth = 50.

Figure 6: Graphs showing the comparison of medians.

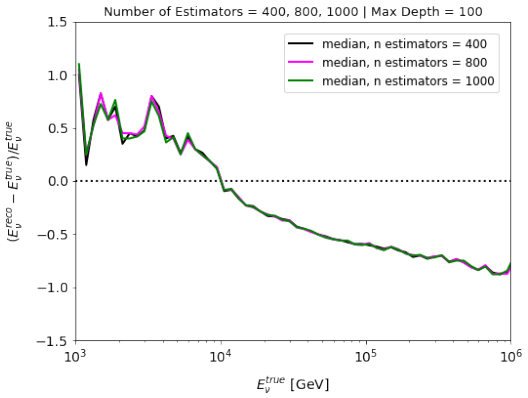


Figure 7: Graphs showing the comparison of medians.

produces the most efficient and accurate results of track-like reconstructed events detected by the IceCube Neutrino Observatory.

References

[1] R. A. et. al. A convolutional neural network based cascade reconstruction for the IceCube neutrino observatory. *Journal of Instrumentation*, 16(07):P07041, jul 2021. doi: 10.1088/1748-0221/16/07/p07041. URL `https://doi.org/10.1088%2F1748-0221%2F16%2F07%2Fp07041`.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.