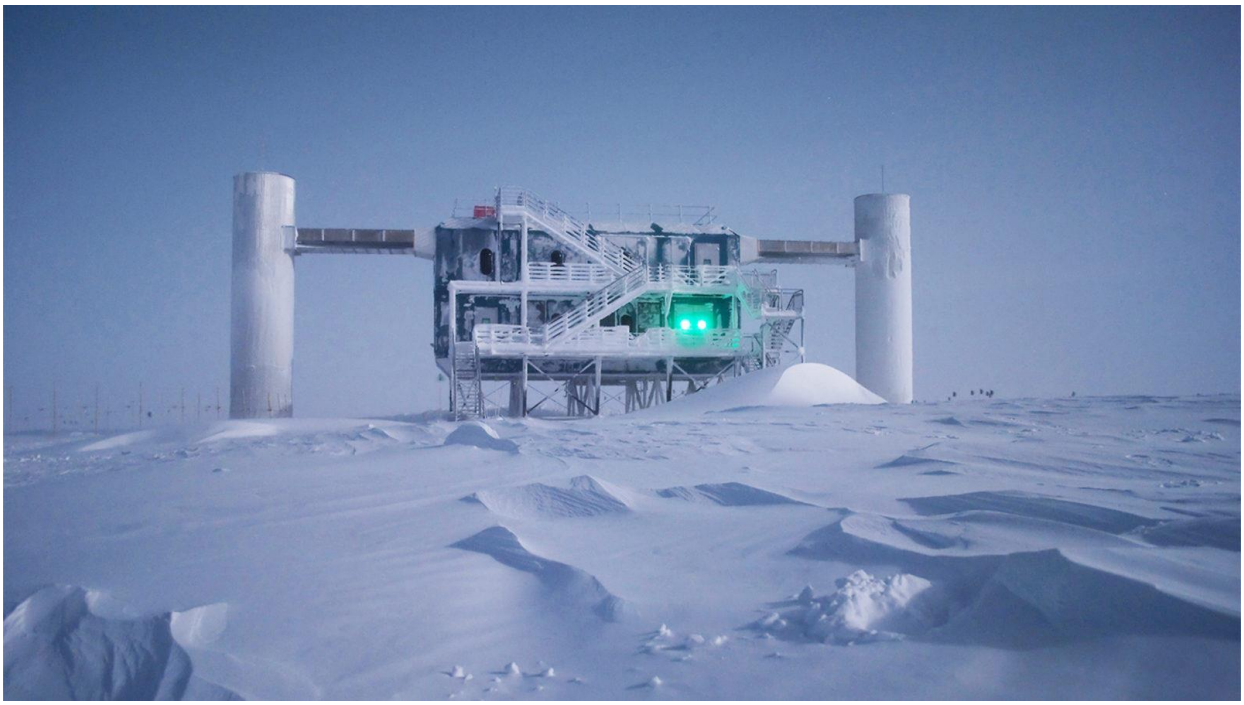


Predicting Pulse Charges caused By Neutrino Collisions Using Data from The IceCube Neutrino Observatory

Ilyas Saoud

DAT 490

Table of Contents

1. Abstract

The IceCube Neutrino Observatory is a large facility located at the South Pole with its sensors located deep under the surface to detect high-energy neutrinos to study how they interact with ice atoms. This project is aimed to analyze a large dataset of 131 million observations that are split into 660 batches, using three separate datasets. The goal is to gain a better understanding of high-energy neutrinos and push our findings to better aid further research that will contribute to our understanding of some of the most fascinating phenomena in the Universe such as black holes, and supernovas. The training dataset includes six variables, and the geometry dataset represents the location of 5,159 sensors. The training dataset of batch number one contains details of each event, including the sensor ID, time of the pulse, charge, and auxiliary information. The final dataset merges the training dataset of the first batch with the geometry dataset and the main train dataset. In this project, various machine learning models were used to predict the energy of our neutrinos and used separate methods to analyze our data before building the models. The Gradient Boosting Regression (GBR) and Linear Regression models had poor performances, with the predicted values being significantly off from the actual values. This indicates that the model is not well-suited for the data, not enough data, or that there are issues with the model's parameter settings. However, the neural network model performed significantly better and was able to capture more patterns of the data, resulting in better predictions and accuracy.

2. Project Plan

March 31, 2023

2.1 Profile of the Observatory and Background

2.1.1 Observatory Details:

Founded – Construction began in 2005, with completion in 2010.

Founders – The facility was built and is operated by an international collaboration of scientists and engineers from around the world.

Location – Amundsen-Scott South Pole station in Antarctica.

Categories – Scientific Research, Astrophysics, Neutrino Physics.

2.1.2 Address:

IceCube Neutrino Observatory

222 W. Washington Ave., Suite 500

Madison, WI 53703 USA

2.1.3 Company communication:

Website – https://icecube.wisc.edu/

Phone – (+1) 608-890-0265

2.1.4 Description:

The IceCube Neutrino Observatory is a state-of-the-art facility that is designed to detect high-energy neutrinos, subatomic particles that can provide valuable insights into some of the most important and intriguing phenomena in the universe, such as supernovae, black holes, and gamma-ray bursts. The observatory uses a cubic kilometer of ice located deep beneath the surface of the South Pole as a detector medium. The facility includes a network of sensors that are capable of detecting the faint flashes of light that are produced when neutrinos interact with ice atoms. By studying the characteristics of these interactions, scientists can learn more about the sources of high-energy neutrinos and the physics that governs their behavior. The IceCube Neutrino Observatory is a collaboration between over 300 scientists and engineers from around the world, representing 48 institutions in 12 countries.

2.1.5 Financials:

The main source of income – is grants, and research budgets from institutions and countries involved.

2.1.6 Key Executives:

Francis Halzen, *Principal Investigator and Spokesperson*

Olga Botner, *Deputy Spokesperson*

Darren Grant, *Deputy Spokesperson*

Albrecht Karle, *Data Center Director*

James Madsen, *Detector Engineer*

2.2 Business/Analysis Opportunity:

The IceCube data presents a vast amount of potential for predictive modeling and analysis of research questions. By analyzing the direction and characteristics of the high-energy neutrinos, as well as the location and activation patterns of the sensors, it may be possible to develop models that can predict the sources of these particles. This could lead to further advancements in our understanding of high-energy physics and the origins of cosmic rays. Additionally, analyzing the activation patterns of the sensors and the properties of the neutrinos could provide valuable insights into the fundamental properties of these particles. Overall, the

IceCube data has great potential for advancing scientific research and developing new technologies in a variety of fields.

2.3 Research Questions

Research questions are important because they help scientists to ask specific questions and focus on areas of interest. This can lead to discoveries and advances in the field of high-energy neutrino astronomy. The following research questions will be the base foundation for our project.

RQ1: Does the distribution of charges differ significantly between batches?

Testing whether the distribution of charges is different or the same between batches in the IceCube data is important because it can help us to identify any systematic biases or variations in the data collection process. If we find that the distribution of charges is different between batches, it could suggest that there are underlying factors affecting the data collection process, such as environmental conditions or technical issues. On the other hand, if we find that the distribution of charges is the same between batches, it suggests that the data collection process is consistent and reliable. This is a crucial step before splitting our data and begins fitting our training data to build a predictive model.

RQ2: Can we use our data to predict the charge of a sensor during an event?

Predicting the charge of the neutrino pulse in IceCube data can provide insights into the energies and fluxes of these particles, helping to identify their sources and test theories of high-energy astrophysics. Accurate charge predictions can also optimize the performance of the IceCube detector, which is crucial for detecting and studying these elusive particles. This has far-reaching implications for advancing our understanding of fundamental physics and astrophysics.

2.4 Hypotheses

H1: The batches were split for computational reasons, and as such, the distribution of charges should be similar across batches.

Since the batches were split due to a large number of observations, the charges may be distributed evenly across batches. We must test this hypothesis before training our data.

H2: There is a relationship between the independent features of our dataset and the amount of charge the sensor detects during an event.

Because high-energy sources of neutrinos – such as exploding stars – will typically generate more energy in the sensors than neutrinos from our sun, our charge variable will be dependent on all features, especially the direction feature that indicates the angles from where these neutrinos came from.

2.5 Data

The dataset contains 131 million observations or 'events' that are split into 660 batches. Each event represents the collision of the neutrinos – which are subatomic particles – that were detected by 5,159 sensors located under the surface. The project includes 3 separate datasets:

- **Train dataset:** This dataset represents each event (131 million observations). This dataset includes 6 variables of numerical type:
    - Batch_id – the id of the batch
    - Event_id – the id of the event
    - First/last_pulse_index – The first and last sensor activated.
    - Azimuth/zenith angles – The direction of the neutrinos in radians.
    - N_time_steps – number of sensors activated in order.
- **Geometry dataset:** This dataset represents the location of each of the 5,159 sensors, all numerical:
    - Sensor_id – the sensor identifier
    - x,y,z – positions of sensors
- **Train dataset of batch1:** This dataset includes details of each event, also all numerical:
    - Sensor_id – the sensor identifier
    - Time – time of the pulse in nanoseconds in the current event window
    - Charge – Amount of light in the pulse
    - Auxiliary – If True, the sensors were activated due to noise.

The final dataset will consist of merging our train dataset for the first batch with the geometry dataset, and the main Train dataset which consists of important features such as the azimuth/zenith angles which will be crucial for our research.

2.6 Measurements

In the IceCube Neutrino Observatory data, many measurements could be important for analysis. For example, the number of sensors activated in order, the position of sensors, and the time and amount of light in each pulse. These measurements can help us understand the behavior of neutrinos and how they interact with the detectors. It's important to carefully analyze and interpret these measurements to extract useful information from the data. Additionally, data preprocessing techniques such as normalization, scaling, and feature selection can be used to prepare the data for analysis. By carefully selecting and analyzing these measurements, we can gain insights into the behavior of neutrinos and use this information for further research and scientific discovery.

2.7 Methodology

For research question 1, one possible approach would be to perform a statistical test, such as a chi-square goodness-of-fit test to compare the distribution of charges across different batches or

maybe an ANOVA test. If the distribution of charges does not differ significantly between batches, then it may be appropriate to proceed with training without randomly ordering the data. For research question 2, we can develop a predictive model to estimate the charge of a sensor during an event using machine learning techniques such as deep learning algorithms. We can train the model on a subset of the data and validate its performance on another subset. To improve the accuracy of the model. We could use regression or build a neural network model for prediction.

2.8 Computational Methods and Outputs

For research question 1, one possible computational method would be to use statistical software packages such as R or Python to perform the chi-square goodness-of-fit test or ANOVA test. The output of these tests would be a p-value indicating whether there is a statistically significant difference in the distribution of charges between batches. If the p-value is below a certain significance level (0.05), we can reject the null hypothesis that the distributions are the same and conclude that the batches are not equivalent. For research question 2, we can use machine learning libraries such as TensorFlow or Keras to build and train deep learning models. The input to the model would be the sensor data, and the output would be the predicted charge of the sensor during an event. The model would be trained on a subset of the data, and the validation set would be used to test its performance. The Neural Network could consist of 2 hidden layers, and the input data will be preprocessed in a way that will produce the best accuracy. The output of the model would be a predicted charge value for each sensor, and the accuracy of the model would be measured by metrics such as mean squared error or coefficient of determination. This approach can be further enhanced by using feature engineering techniques to extract more meaningful features from the data and improve the accuracy of the predictive model.

2.8.1 Output Summaries

Research Question 1:

The output summary for RQ1 may include a statistical analysis report, which would compare the distribution of charges across different batches. If the null hypothesis is rejected, indicating a significant difference between the batches, the report would suggest reordering the data randomly before proceeding with training. If the null hypothesis is accepted, indicating no significant difference between the batches, the report would recommend training the model without shuffling the data. An ANOVA result could look like the following:

ANOVA

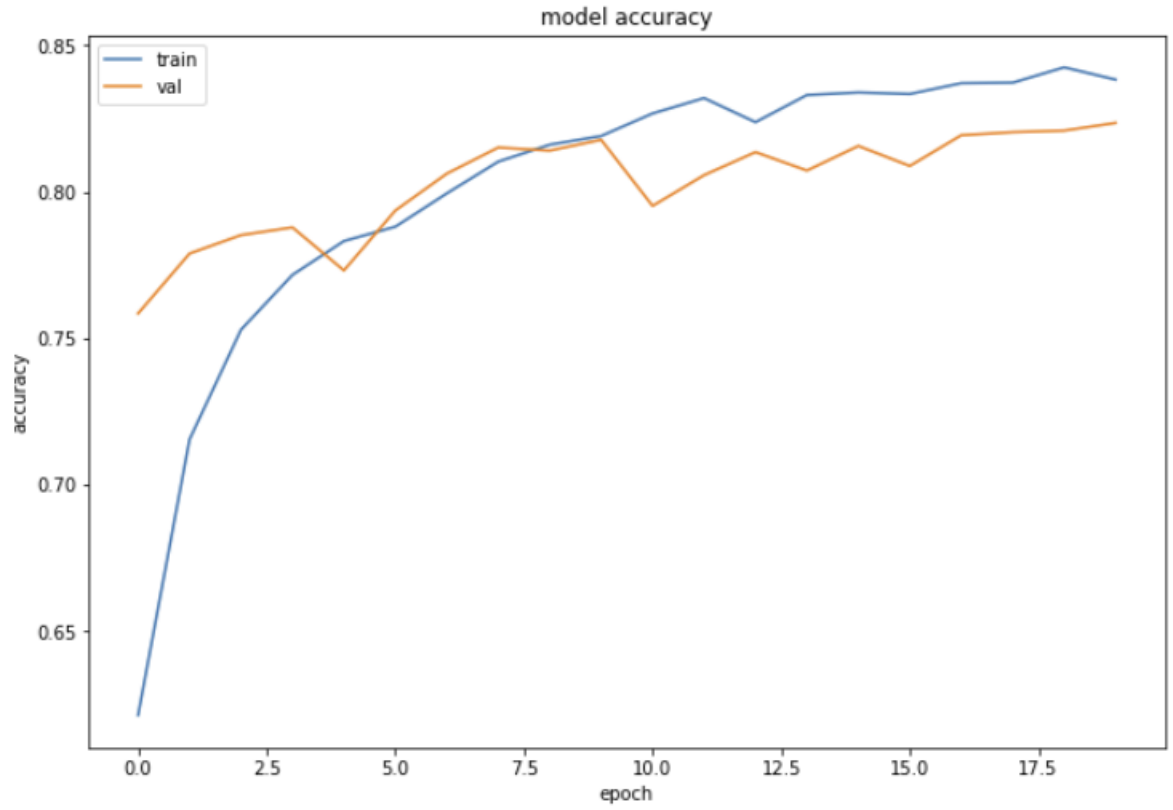
score

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	339.800	2	169.900	4.545	.020
Within Groups	1009.400	27	37.385		
Total	1349.200	29			

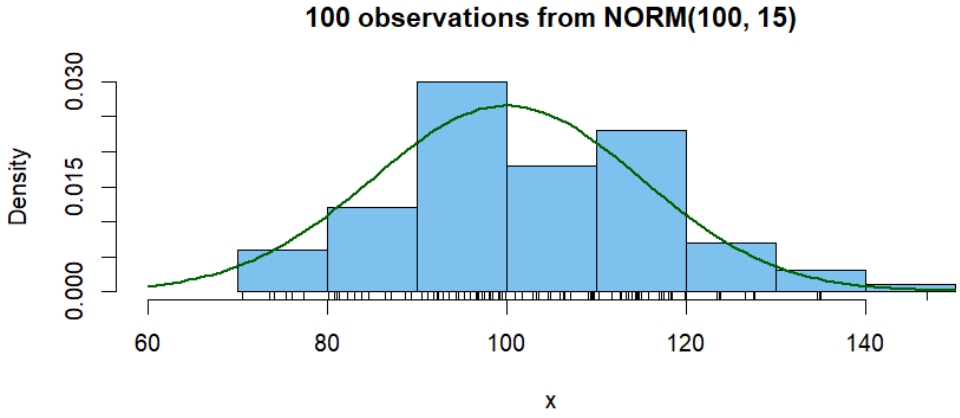
Research Question 2:

For RQ2, the output summary could include a predictive model with an estimated accuracy score, which would indicate how well the model can predict the charge of a sensor during an

event. The output could also include a confusion matrix to visualize the performance of the model, showing the true positive, true negative, false positive, and false negative predictions. Additionally, the summary may include feature importance rankings to highlight which variables have the most significant impact on the charge prediction. One of our results could be an accuracy score between our training and validation datasets. This example was taken from a previous project *(Figure 11 & Figure 12)*:



We could also include a density histogram that can show us the angles or directions where neutrinos of higher energy are coming from.



2.9 Observatory Implementation

The IceCube Neutrino Observatory would be able to improve its understanding of neutrino interactions and possibly discover new physics beyond the Standard Model. The results could lead to the development of more efficient neutrino detectors and contribute to the advancement of astrophysics and particle physics.

Regarding Research Question 1, if the charges are found to be distributed differently across batches, IceCube could modify its data collection and processing methods to improve the accuracy and reliability of its measurements. For example, they could re-randomize the data or adjust the batch sizes to ensure consistent and reliable results.

For Research Question 2, if a reliable predictive model is developed, IceCube could use it to estimate the charge of sensors during an event in real time. This would be a significant improvement over the current method of manual data processing, which is time-consuming and prone to human error. With the predictive model, IceCube could potentially detect and respond to events in a matter of seconds rather than hours or even days.

2.10 Literature Review

There has been a lot of hype lately surrounding AI applications such as ChatGPT, but AI applications have been used in our daily lives for over a decade now, ever since we broke through the hardware limitations that once held us back, we began taking advantage of the power of CPUs and GPUs to apply statistical models to many applications. The most used techniques in machine learning are prediction and classification. For our project, we will focus on predictive analysis. Machine learning relies heavily on data, but not just any data, we need good data to predict accurate results, from having a large number of observations to a healthy number of features that can act as independent variables.

Making decisions based on findings from our data can be a long process but a worthy one, and it's important to not skip any steps. Cao categorizes these steps as such: "They include reporting, statistical analysis, alerting, forecasting, predictive modeling, optimization, prescriptive analytics, and actionable knowledge delivery (delivering insights-based actions for business decision-making and operations)" [Cao et al. 2010]. Typically, you want to start with descriptive analytics to understand the data and describe what has happened in the past. Once we have that, we can move on to predictive analytics, here we use what we know, and the data from the past to predict what could happen, and finally, the most important step, that is prescriptive analytics, this is where we state what steps to take next to achieve our goal. It's important to note that the decision-making process doesn't end after taking prescriptive actions. We must monitor the outcome of any changes made; this could be done through A/B testing. It's also important to re-evaluate our decisions as new batches of data become available.

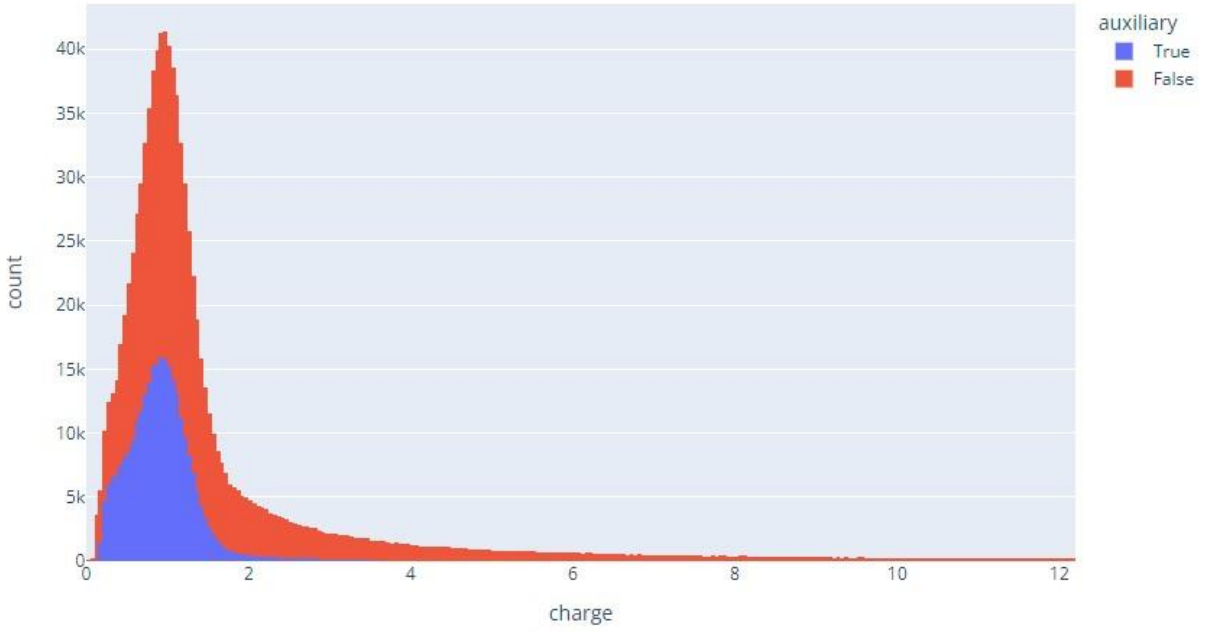
While data science can be used in all sorts of industries and businesses, one exciting ongoing challenge in the scientific field is happening at the IceCube Neutrinos Observatory. Here we have scientists observing the behavior of neutrinos – which are subatomic particles – using 5,159 sensors located under the surface of the laboratory located at the South Pole station in Antarctica. The dataset contains 131 million observations with many features that can lead to new research findings.

Our goal for this project is to predict the energy of neutrinos. Neutrinos' energy represents the amount of kinetic energy they possess due to their motion. The energy of a neutrino is typically measured in electronvolts (eV) or its multiples, such as Kiloelectronvolts (keV), megaelectronvolts (MeV), gigaelectronvolts (GeV), or Petaelectronvolts (PeV), depending on its energy level. The detection of high-energy neutrinos with energies up to several PeV indicates that these particles have been accelerated to extremely high speeds, possibly in the most energetic astrophysical environments in the universe, such as black holes or supernovae.

In 2013, the American Association for the Advancement of Science ran a study using data from the IceCube Observatory where they observed 28 high-energy neutrino events. Their goal was to prove that high-energy neutrinos exist, motivated by the generic predictions done in the past. They noted that some properties were “inconsistent with atmospheric backgrounds” but are “consistent with generic predictions for an additional component of extraterrestrial origin.” The study used the data to provide evidence for the existence of high-energy extraterrestrial neutrinos. While this may not directly relate to our project where we will be predicting energy levels through sensors, it does offer valuable insights and aids our understanding of the data and the possibilities which motivate further research in the future and can help us refine our analysis and therefore improve our predictions.

3. Exploratory Data Analysis

Let’s begin with our histogram. This is a commonly used data visualization tool that allows us to explore the distribution of a variable in a dataset. The histogram of charge, colored by Auxiliary, is an important visualization that can help to identify any anomalies or outliers in the dataset. The charge variable represents the amount of light in the pulse, and the auxiliary variable indicates whether the sensors were activated due to noise. When Auxiliary is equal to True, it indicates that the sensors were active due to noise.



The histogram *(Figure 1)* comes from our charge variable where the data consists of 1 million observations, however, I only plotted data points that fall below a z-score of 0.5 *(z-score < 0.5)*. This allows us to view our histogram better. Let's compare some charge statistics between batch1's data points and our randomly sampled data points:

Batch1

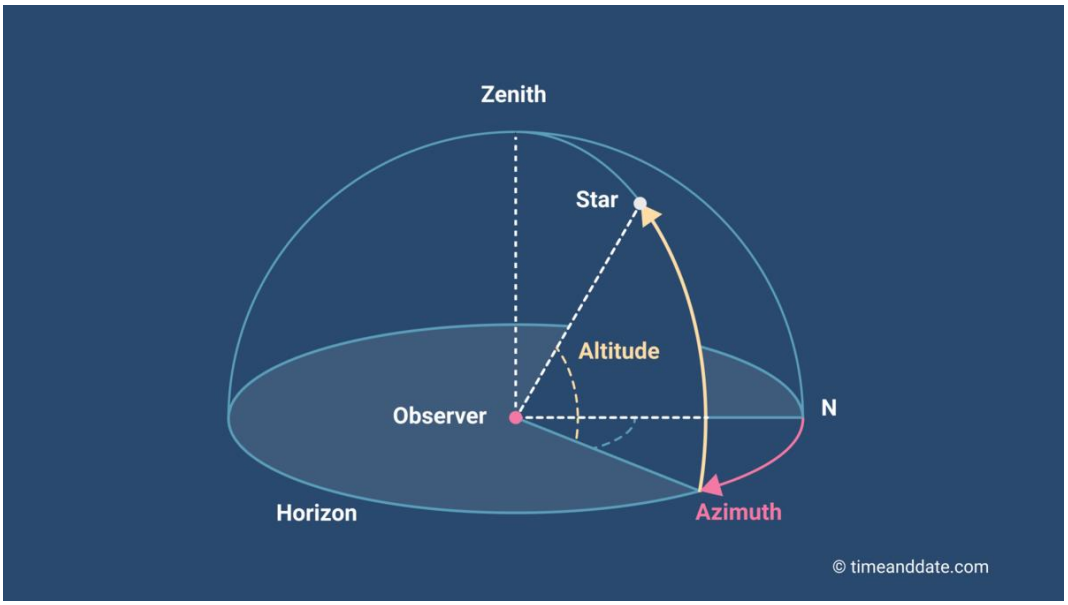
```
count    3.279242e+07
mean     3.908981e+00
std      1.628897e+01
min      2.500000e-02
25%      7.750000e-01
50%      1.075000e+00
75%      1.775000e+00
max      2.762025e+03
Name: charge, dtype: float64
```

Sample

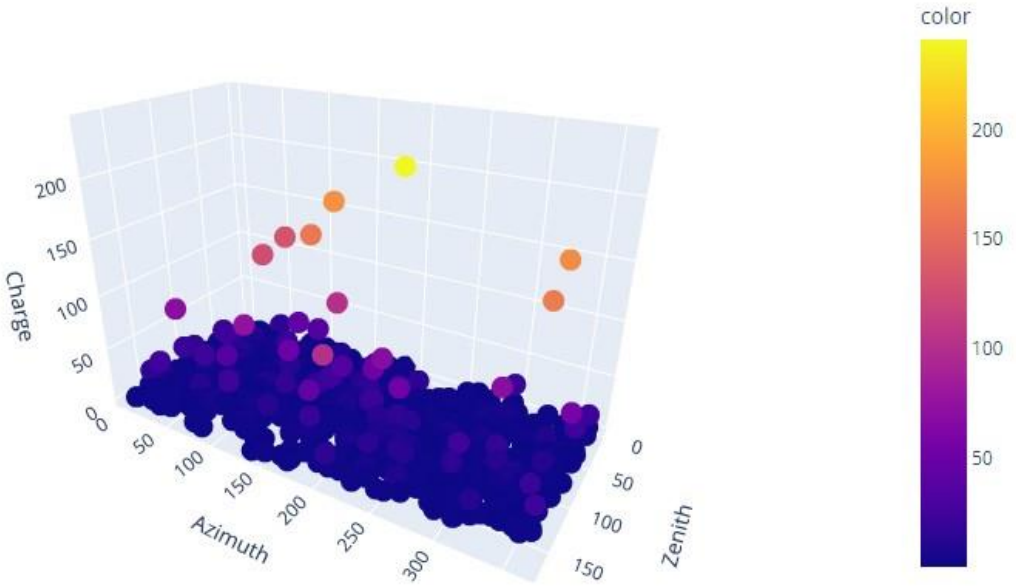
```
count    1000000.000000
mean           3.942740
std           16.490711
min            0.025000
25%            0.775000
50%            1.075000
75%            1.775000
max         2211.524902
Name: charge, dtype: float64
```

From just our observation here, there is no significant difference statistically. Which makes our sample data a good representation of the population's histogram distribution.

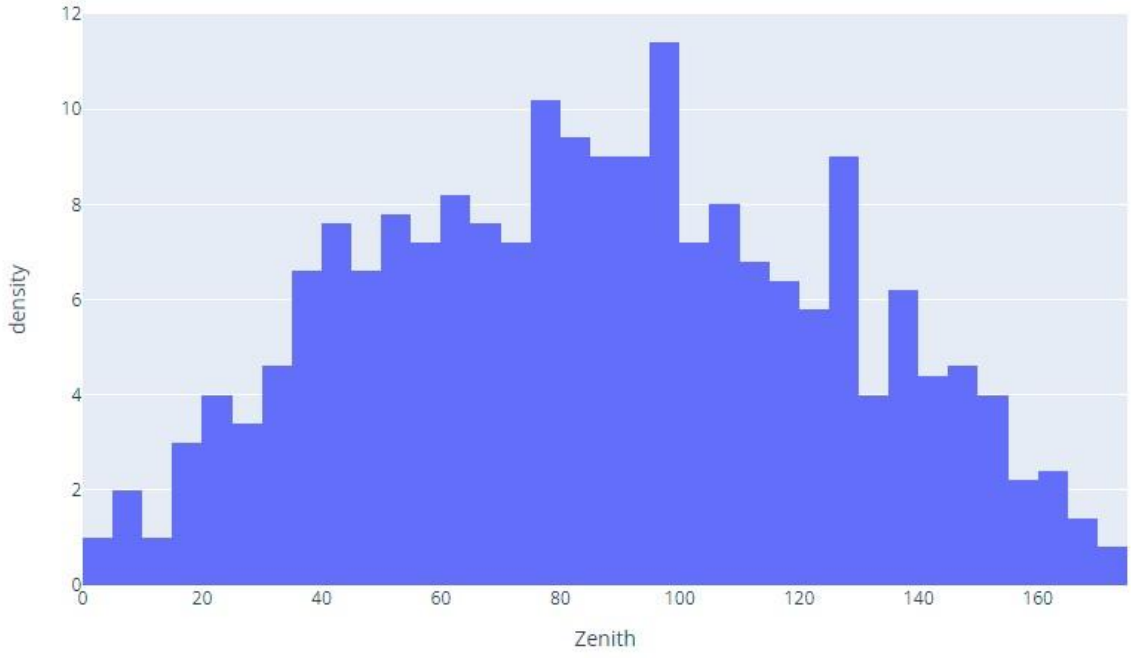
As I mentioned in our Project Plan, our ultimate goal is to predict the energy of neutrinos. Neutrinos' energy represents the amount of kinetic energy they possess due to their motion. In our data, we're dealing with many variables, but one very interesting and important feature is the direction that points towards the source of these neutrinos which are represented in two angles: Azimuth and Zenith. Here is a representation of these two angles in radians *(Figure 2)*.

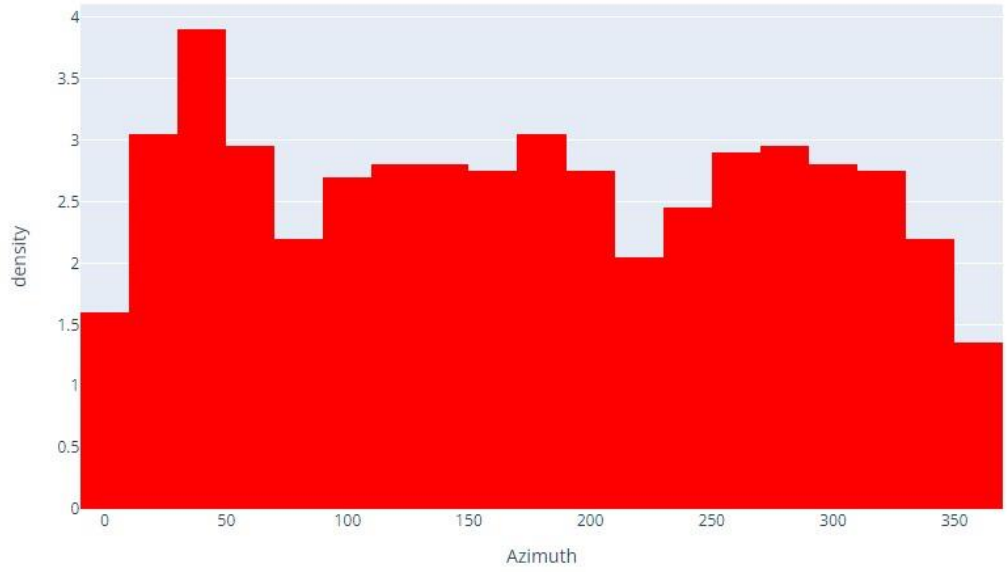


If there is one thing that IceCube Observatory scientists get excited about, that would be the detection of high-energy neutrinos with energies up to several PeV, which indicates that these particles have been accelerated to extremely high speeds, possibly in the most energetic astrophysical environments in the universe, such as black holes or supernovas! It's important to note that our charge value is measured by photoelectrons (p.e), *"a pulse with charge 2.7 p.e. could quite likely be the result of two or three photons hitting the photomultiplier tube around the same time."* This can't be directly converted to the voltage without additional information such as the conversion factor of the detector or photomultiplier tube. Let's visualize a 3-dimensional representation of these angles (converted to degrees) and some charges *(Figure 3)*. Let's also remember that the mean value of these charges is 3.9 P.E.



It's a bit difficult to visualize any trends in this 3-dimensional graph, but we do see that the majority of these points lie at the bottom where the mean is, which makes sense. Let's visualize the Zenith and the Azimuth *(Figure 4 & Figure 5)* angles and see what sort of distribution they hold.

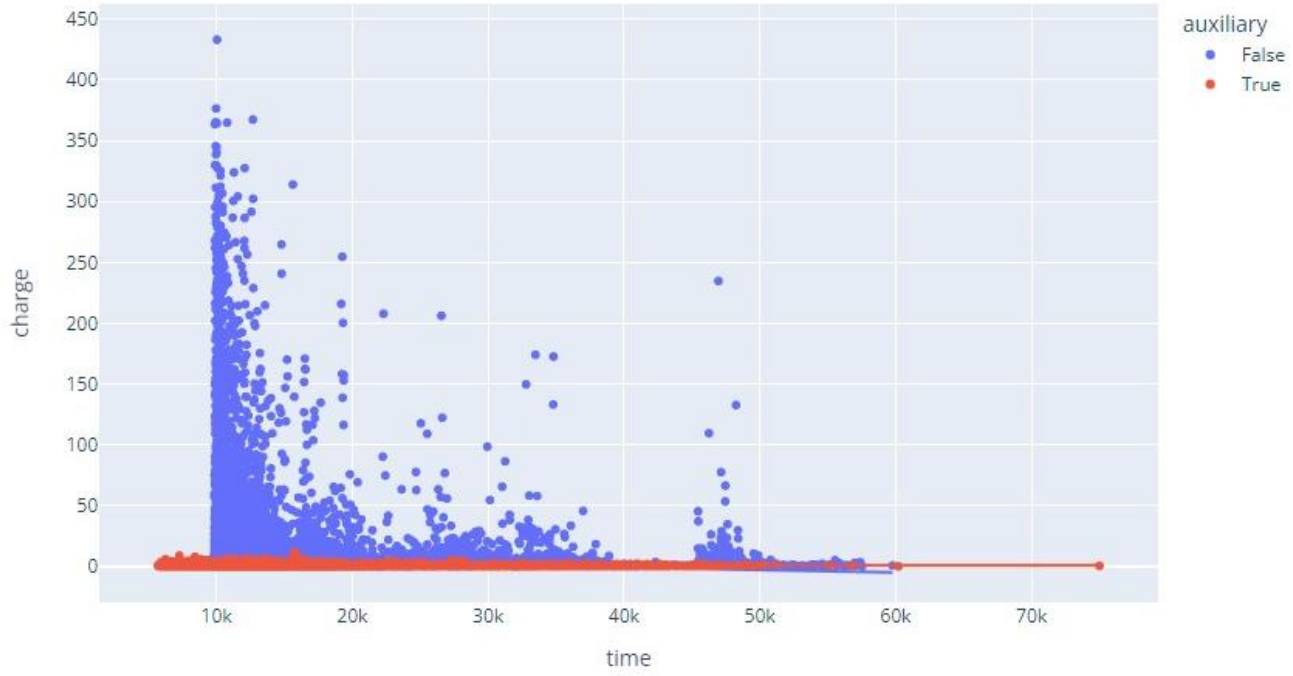




Here we can see that Zenith has a normal distribution, while Azimuth has a uniform distribution. We can expect most neutrinos – with noise and without noise – in the universe to come from a Zenith angle of 87 degrees, with a std of 39.

```
count   1000.000000
mean      86.733698
std       38.644032
min        2.799585
25%       57.075637
50%       87.267466
75%      115.732929
max      174.528113
```

Finally, I'd like to explore the relationship between time and charge. Here the time is in nanoseconds, this represents the time it took for the pulse to be activated in this event time window. The authors of this dataset mentioned that "*the actual time at which a pulse is recorded is not as important as the relative timing of that pulse with respect to other pulses within the same event."* Let's explore this time vs. charge relationship *(Figure 6)*.



Here we can see that the majority of the high-energy neutrinos lie in the range of 8,000 nanoseconds and 15,000 nanoseconds. This could be useful when performing some feature engineering later on, before building a model that will predict our charge variable. Here we can once again compare the differences of the charge grouped by the auxiliary feature. We can see that the activations due to noise have very low charge levels. This chart allows us to see the relationship between time and the charge, however, let's keep in mind that the mean is still 3.9 PE. This can be better presented in our other plots, but this gives us more information regarding high-energy neutrinos and activation time.

4. Methodology

RQ1: Does the distribution of charges differ significantly between batches?

The dataset contains 131 million observations or 'events' that are split into 660 batches. To ensure the quality of our batch, we must ensure that the distribution of our 'charge' variable between all the batches is not significantly different. One possible approach would be to perform a statistical test. Because our variable of interest is continuous, the ANOVA test will allow us to determine if there is a statistically significant difference in the mean charges across the different batches. If the null hypothesis of equal means is rejected, then post-hoc tests can be conducted to determine which batches have significantly different mean charges. When our ANOVA is conducted and the null hypothesis of equal means is rejected, it can only tell us that at least one group has a significantly different mean from the others, but it does not indicate which specific groups are different. Therefore, a post-hoc test can determine which groups have significantly different means.

For our ANOVA test, we will be conducted at a significant level of 0.05 between all 660 batches. Due to a large number of batches, some of the means may be significantly different from each

other due to chance alone, even if the null hypothesis of equal means is true. Therefore, it's important to use a post-hoc test that adjusts for multiple comparisons to reduce the risk of type I error (i.e., falsely rejecting the null hypothesis). As for our post-hoc test, Tukey's Honestly Significant Difference (HSD) test can be conducted. The test will compare all possible pairs of means and calculate the minimum significant difference needed between any two means to reject the null hypothesis of equal means.

RQ2: Can we use our data to predict the charge of a sensor during an event?

Once we perform our ANOVA test and decide on a good batch of data, we will then move on to building our predictive model. Before we begin training our dataset, we must perform some preprocessing and potentially feature engineering, we might find it helpful to view our EDA to add, remove, or modify our variables. For the nature of our data, preprocessing can consist of one-hot encoding, label encoding, and feature scaling. Some statistical models such as decision trees and random forests don't require feature scaling, but for most of our models, we will be performing all three preprocessing steps. One-hot encoding turns categorical variables that have no order effect into a binary vector. Label encoding also deals with categorical variables onto numerical variables, but here the order matters. For example, 0 can represent 'weak' and 1 can represent 'strong.' Feature scaling standardizes the range of numerical features in data. This technique is used to ensure that all features contribute equally to the model and to prevent features with large values from dominating the model. Let's explore some of our options.

4.1 Modeling Techniques:

- Linear Regression
    - A widely used algorithm for regression problems that models the relationship between a dependent variable and one or more independent variables. It fits a straight line to the data by minimizing the sum of the squared residuals between the predicted and actual values. For our case, we're going to use Multiple Linear Regression.
- Gradient Boosting Regressor
    - An advanced algorithm for regression problems that combines the predictions of multiple weak models, typically decision trees, to improve the accuracy of the final prediction. It trains each tree on the residuals of the previous tree to iteratively improve the model.
- Support Vector Machines (SVM)
    - A versatile class of algorithms that can be used for classification or regression problems. In regression, SVMs aim to find the hyperplane that maximizes the margin between the predicted values and the actual values. This algorithm can handle non-linear relationships between the dependent and independent variables.
- Neural Networks
    - A powerful class of algorithms that are capable of modeling complex non-linear relationships between the dependent and independent variables. Neural networks consist of layers of interconnected nodes that perform mathematical operations on the input data to produce a prediction. They can be used for both regression and classification problems.

For our project, we're going to explore all of these methods and see which one will perform the best. We will analyze their training and validation accuracy scores and other results for comparison.

5. Visualization and Analysis

RQ1: Does the distribution of charges differ significantly between batches?

5.1 ANOVA

As we discussed in our methodology paper, to confidently work with one of our data sets, we must first ensure that there is no significant difference between the batches. The full dataset contains 660 batches, where each batch contains about 32 million observations. That is too large a number to work with for this project, mostly due to CPU limitations, and although we have access to a powerful P100 GPU, the training would take 10+ hours to complete. The first step is computing an ANOVA test, to compare the charge variable – that is the variable we're trying to predict – and see if there's a significant difference between the batches. Here are the results:

```
F-Statistic: 368.69
p-value: 0.00000
The means of the populations are not equal at a significance level of 0.05.
```

Although the p-value is 0 – which means at least one of the batches is significantly different – after performing a Tukey's Honestly Significant Difference (HSD) test, I found that almost all batches differ <0.01 in means between each other as seen below. Keep in mind this is only a small snippet of the full output. Each batch was compared to all the other 659 batches.

5.2 Tukey's HSD

```
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
  group1    group2  meandiff  p-adj   lower   upper  reject
-----------------------------------------------------
  Batch 1  Batch 10    0.025   -0.0  0.0175  0.0325   True
  Batch 1 Batch 100   0.0242   -0.0  0.0167  0.0317   True
  Batch 1  Batch 11  -0.0085 0.0049 -0.0159  -0.001   True
  Batch 1  Batch 12  -0.0083 0.0073 -0.0158 -0.0008   True
  Batch 1  Batch 13   0.0278   -0.0  0.0203  0.0353   True
  Batch 1  Batch 14  -0.0354   -0.0 -0.0429 -0.0279   True
  Batch 1  Batch 15  -0.0044 0.9984 -0.0119  0.0031  False
  Batch 1  Batch 16   0.0106    0.0  0.0031  0.0181   True
  Batch 1  Batch 17    0.012    0.0  0.0045  0.0195   True
  Batch 1  Batch 18  -0.0007    1.0 -0.0082  0.0068  False
  Batch 1  Batch 19  -0.0176   -0.0 -0.0251 -0.0101   True
  Batch 1   Batch 2  -0.0174   -0.0 -0.0249 -0.0099   True
  Batch 1  Batch 20  -0.0027    1.0 -0.0102  0.0048  False
  Batch 1  Batch 21   0.0146    0.0  0.0071  0.0221   True
  Batch 1  Batch 22   0.0164    0.0  0.0089  0.0239   True
```

This difference is acceptable, so for the next few statistical models, we will be working with the batch number one, and from this batch, we will randomly sample out 250,000 observations without replacement.

RQ2: Can we use our data to predict the charge of a sensor during an event?

Datasets

	event_id	sensor_id	time	charge	auxiliary	x	y	z	first_pulse_index	last_pulse_index	azimuth	zenith	n_time_steps
0	2782231	1979	13602	0.475	True	-324.39	-93.43	-504.09	28655830	28655897	4.736250	0.289624	67
1	863181	4119	14898	0.875	True	-268.90	354.24	-162.27	10302046	10302084	3.065164	2.693593	38
2	2045145	84	8444	0.525	True	-132.80	-501.45	91.58	7012095	7012197	4.795914	2.319830	102
3	3264616	1008	11273	1.325	False	-43.27	-267.52	-316.87	10607759	10607808	2.748960	1.816309	49
4	197119	1720	11669	0.825	False	371.56	-92.18	-179.25	32157778	32157861	5.449773	1.552117	83
...	...	...	...	...	...	...	...	...	...	...	...	...	...

This dataset is used for our statistical models, containing 250,000 observations. To create such a dataset, we had to merge the batch dataset with the sensor geometry dataset, and the training dataset which consisted of all 131,953,923 events. Let's review our variables once again.

Dataset	**Description**
Train dataset	This dataset represents each event (131 million observations). This dataset includes 6 variables of numerical type: o Batch_id – the id of the batch. o Event_id – the id of the event. o First/last_pulse_index – The first and last sensor activated. o Azimuth/zenith angles – The direction of the neutrinos in radians.
Geometry dataset	This dataset represents the location of each of the 5,159 sensors, all numerical: o Sensor_id – the sensor identifier. o x,y,z – positions of sensors.
Train dataset of batch1:	This dataset includes details of each event, also all numerical: o Sensor_id – the sensor identifier o Time – time of the pulse in nanoseconds in the current event window. o Charge – Amount of light in the pulse o Auxiliary – If True, the sensors were activated due to noise.

5.3 Statistical Models

The dataset was split into training and testing, where 70 percent of the data was used for training, and 30 percent was used for validation. Both the Multiple Linear Regression and the Gradient boosting Regressor (GBR) performed horribly. GBR model did not require the data to be standardized – a technique that is used to ensure that all features contribute equally to the model and to prevent features with large values from dominating the model – but the Multiple Linear Regression and the Neural Networked did need to be standardized. As for SVM, our CPU could not handle the computations.

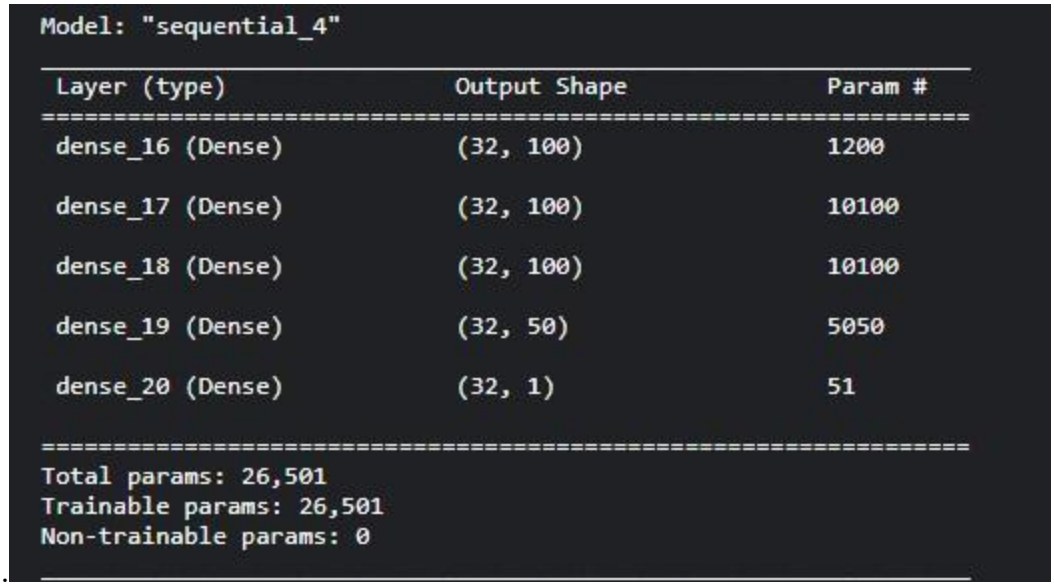
Modeling Technique	**MSE**	**R2**
Gradient Boosting Regressor	222	0.072
Multiple Linear Regression	235	0.017
Multilayer Perceptron (MLP)	141.4	0.48

The Multilayer Perceptron (MLP) or Full-Connected Neural Network (FCNN) had the best performance, training with 1,000,000 observations – compared to only 200,000 for the GBR and MLR models:

Epoch 150/150

21875/21875 - 71s - loss: 123.7373 - val_loss: 141.4487 - 71s/epoch - 3ms/step

The model consisted of 26,501 parameters, and 4 hidden layers, all using the 'Relu' activation function.



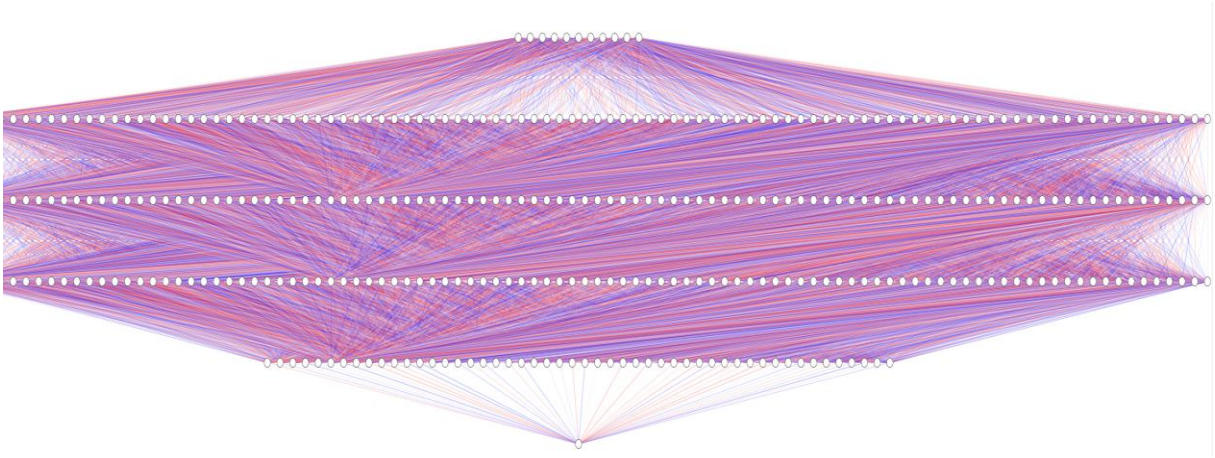
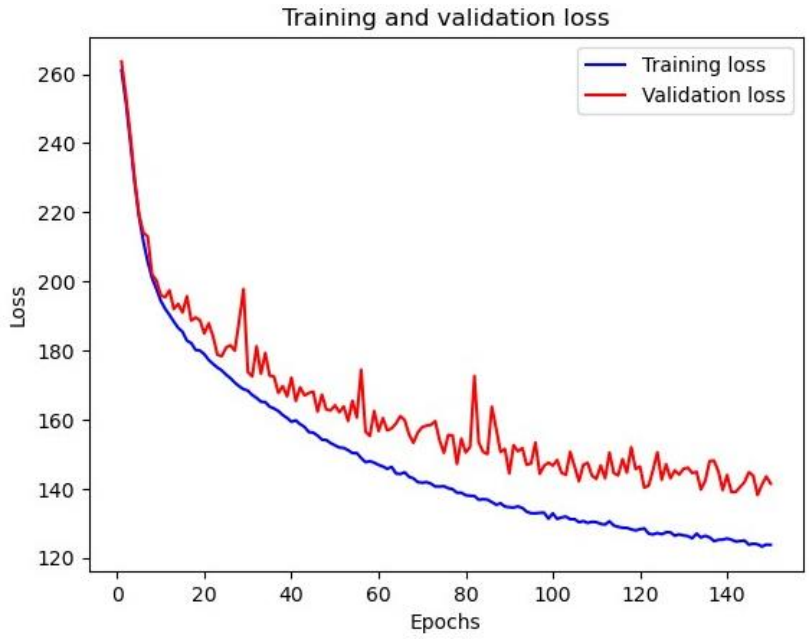


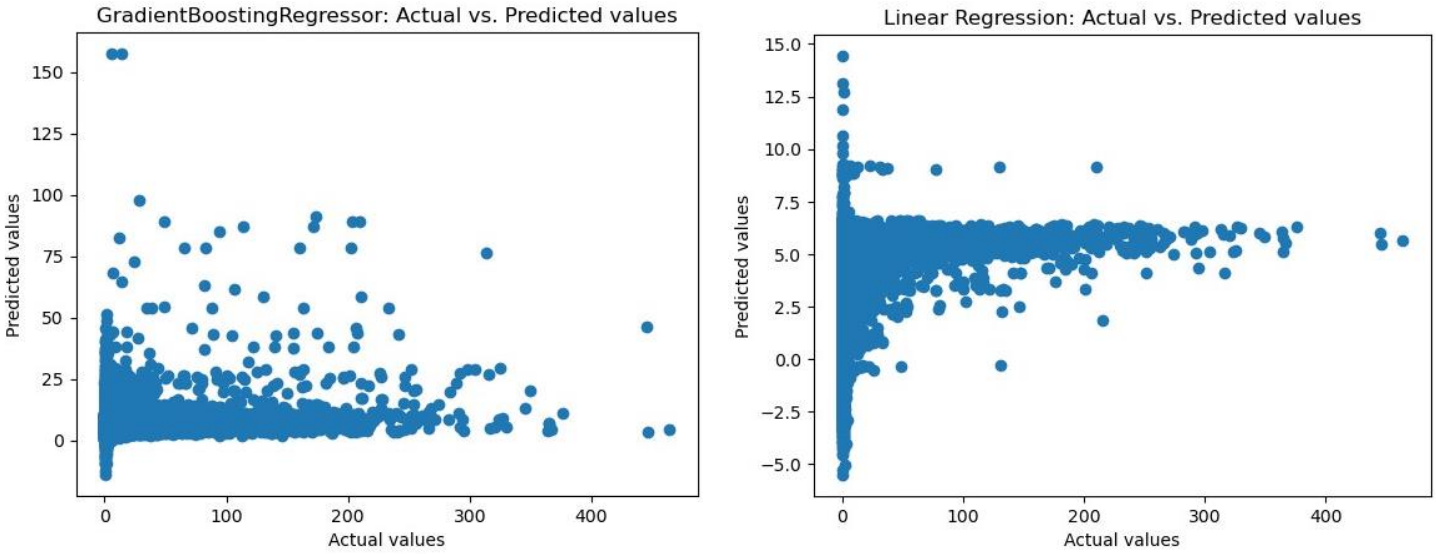
Figure 7 shows a representation of our neural network, moving from top to bottom, the first layer represents our input layer, which consists of 11 nodes, there are also four hidden layers, where the first three consist of 100 nodes and the fourth hidden layer has 50 nodes, finally, the output layer consists of one node which represents the predicted charge variable. This model was trained for approximately 1.5 hours utilizing a P100 GPU. Below *(Figure 8)* is a graph visualizing our loss vs epoch, this is split into training, and validation loss where the blue line represents the training loss, and the red line represents the validation loss.



.

It appears that the model ceased to make significant progress after 100 epochs, however, it is possible that continuing to add more epochs could lead to further improvement and minimize our losses.

As for our GBR and Linear Regression models, looking at the Actual values vs the Predicted values plots *(Figure 9 & Figure 10)*, we can see that the predicted values were so off. Ideally, when you compare the actual values with the predicted values using a scatter plot, you would like to see a linear relationship between the two. This would indicate that the model is accurately capturing the patterns in the data and that its predictions are in line with the actual values. Our GBR and Linear Regression models are so off from the actual values, it may indicate that the model is not well-suited for the data, not enough data, or that there are issues with the model's parameter settings



6. Ethical Recommendations

Scientific experiments and research have long been tied to ethical procedures that ensure the protection of human subjects, animal welfare, and the integrity of the scientific process. Ethical considerations are essential to ensure that scientific research is conducted in a manner that is both safe and ethical. The main idea is to ask the right ethical questions, while we won't get into the different types of ethical theories in this paper, the goal here is to think of all possible outcomes from our findings and reports that can negatively impact others. In addition to considering the potential negative impacts of their research, scientists must also be aware of their own biases and limitations. This requires a deep understanding of the ethical principles that guide scientific inquiry, as well as a commitment to ongoing education and reflection.

In our project, we're using data given by the IceCube Observatory to predict the charge variable. There are some ethics that we must consider before sharing our results and findings. First, let's explore how our findings may be beneficial. Predicting the charge variable has the potential to contribute significantly to the field of particle physics. The results may lead to a better understanding of the nature of neutrinos and their interactions with matter, which can have a large positive impact on fields such as astrophysics, cosmology, and high-energy physics. However, we must also consider the potential negative impacts our results could have. For example, if we release our results without properly validating them, and they contain errors, it could have implications for other researchers. They may use our flawed results as a basis for their research, leading to incorrect conclusions. Inaccurate or incomplete results can break many

ethical laws, and so we must do our best to avoid such errors by reevaluating our work and ensuring that the research can be replicated by others, given the data and instructions, and yielding the same results.

Another thing to consider is the ethical implications of the process of how data is collected. The IceCube Observatory data is obtained through the use of photomultiplier tubes, which detect light produced by neutrinos interacting with ice. The tubes are inserted 1450 meters under the ice surface serving as one large detector located at the South Pole. Although the data collection process is not harmful to us or animals, there are concerns about the environmental impact of the detector's long-term construction and operation.

7. Challenges

Working on the statistical models, the biggest challenge faced was the poor performance of the Gradient Boosting Regressor and Multiple Linear Regression. There are a few causes that may cause this. A neural network may perform better than in gradient boosting or regression if the relationship between variables is non-linear, the dataset is large – which it is - if the patterns are complex, or if feature engineering is difficult. Neural networks can capture non-linear relationships, handle large datasets, learn features automatically, and capture complex patterns. A possible cause as to why the models performed poorly could be the lack of data which can cause the failure of capturing non-linear relationships. Unfortunately, the data was limited to only 1,000,000 rows from the first batch due to the lack of computational power; The cloud GPU P100 was the only graphics processing unit available.

8. Recommendations

Some many other statistical models and techniques should be explored. Some commonly used models include decision trees, Random Forest, and Support Vector Machines (SVMs). Decision trees and Random Forest are tree-based models that can handle both regression and classification problems. SVMs are powerful models that can handle non-linear data by transforming the data into a higher dimensional space. Another further step worth noting is to perform feature engineering, which has not been a big part of our process, but it is highly recommended to capture more information – it is one of the possible reasons why our neural network model performed much better. One recommendation is to use a cloud TPU – available on the Kaggle website – and train the network with more data from the first batch, or multiple batches, this could significantly increase our accuracy. It is also worth considering fine-tuning the hyperparameters of our neural network model. Hyperparameters, such as the learning rate, batch size, and the number of hidden layers, can have a significant impact on the performance of our model. Fine-tuning these hyperparameters can help us optimize the model's accuracy and reduce the risk of overfitting.

9. References

Center for Ethics in Financial Services. (2012). Ethical theories. [PDF]. DePaul University Driehaus College of Business. Retrieved from https://dsef.org/wp-content/uploads/2012/07/EthicalTheories.pdf

IceCube. (n.d.). Retrieved April 23, 2023, from https://icecube.wisc.edu

Cao, Longbing. "Data Science: A Comprehensive Overview." ACM Computing Surveys, vol. 50, no. 3, 2018, pp. 1–42, https://doi.org/10.1145/3076253.

Aartsen, M. G., et al. "Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector." Science (American Association for the Advancement of Science), vol. 342, no. 6161, 2013, pp. 947–947, https://doi.org/10.1126/science.1242856.

10. Appendix

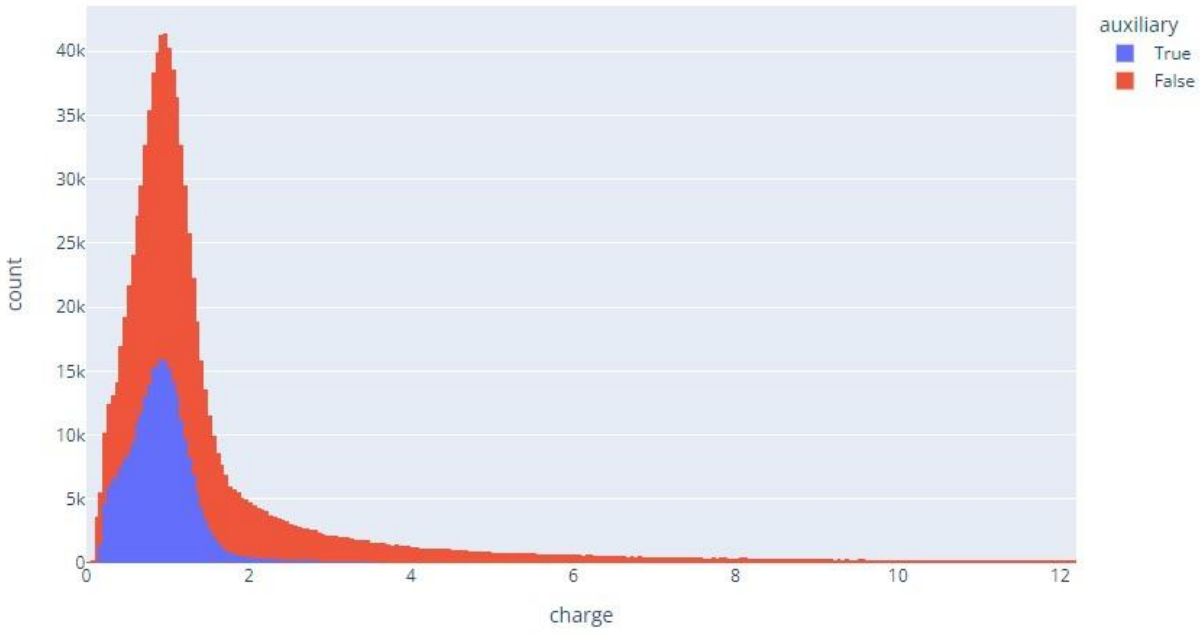


Figure 1. Charge Vs. Count Histogram Graph

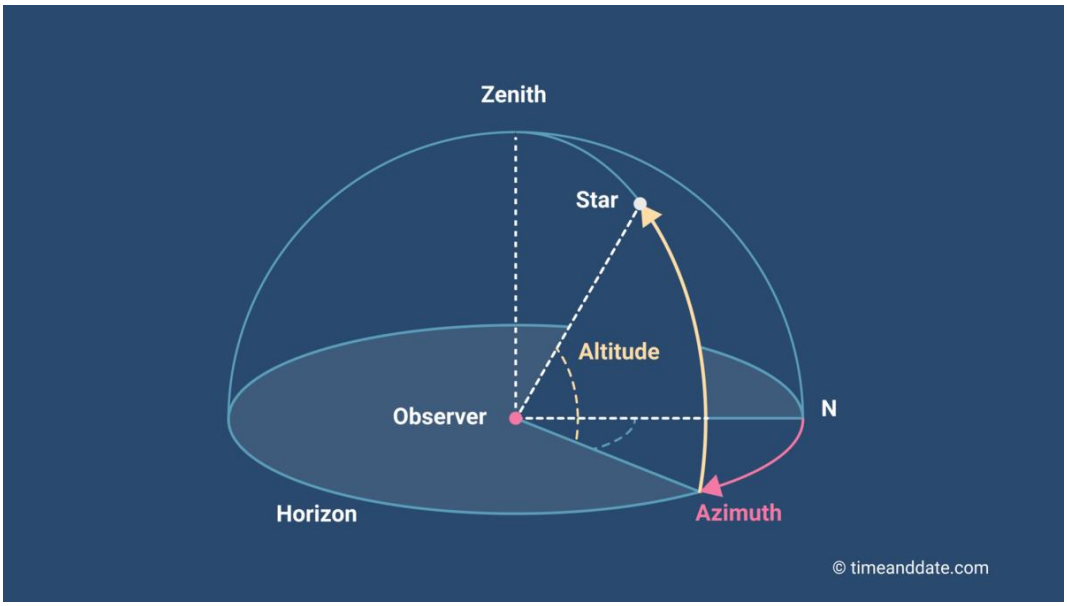


Figure 2. Angles Visualization

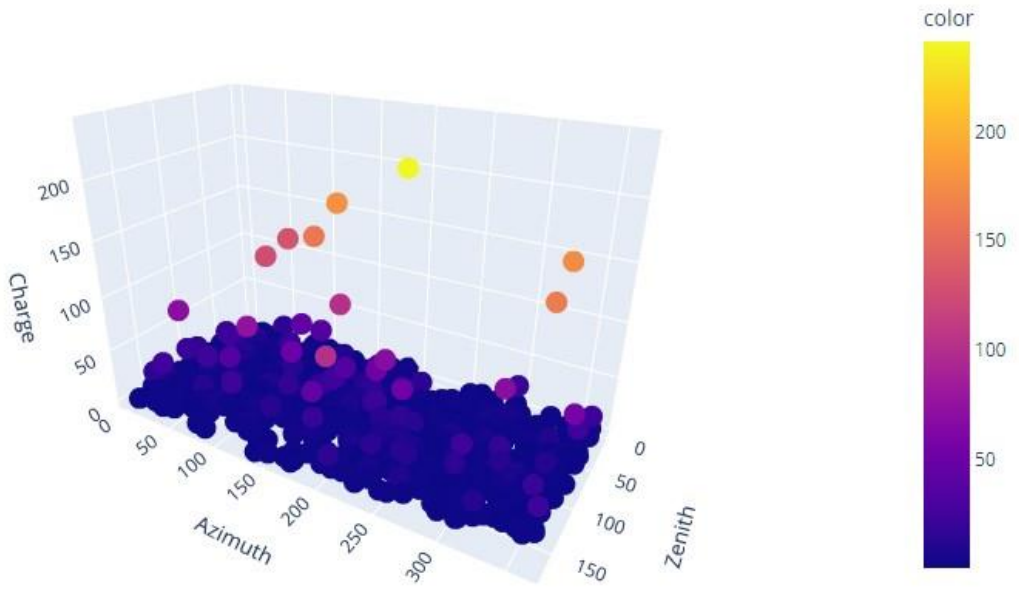


Figure 3. Azimuth and Zenith angles Vs. Charge 3D Scatter Plot

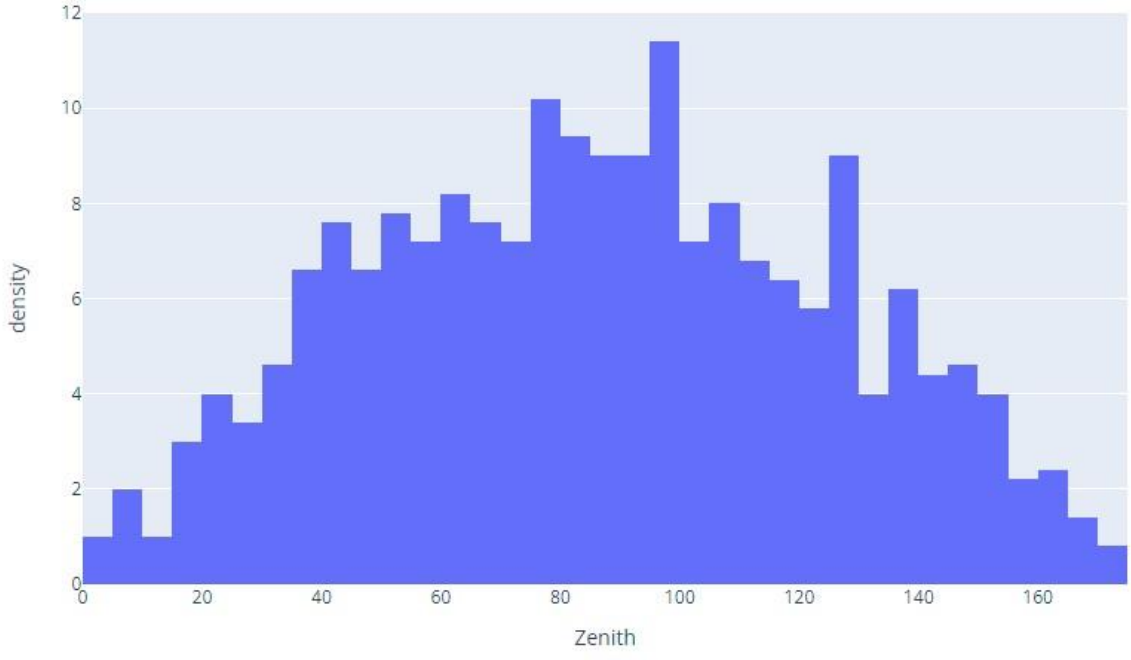


Figure 4. Zenith Vs. Density Histogram Graph

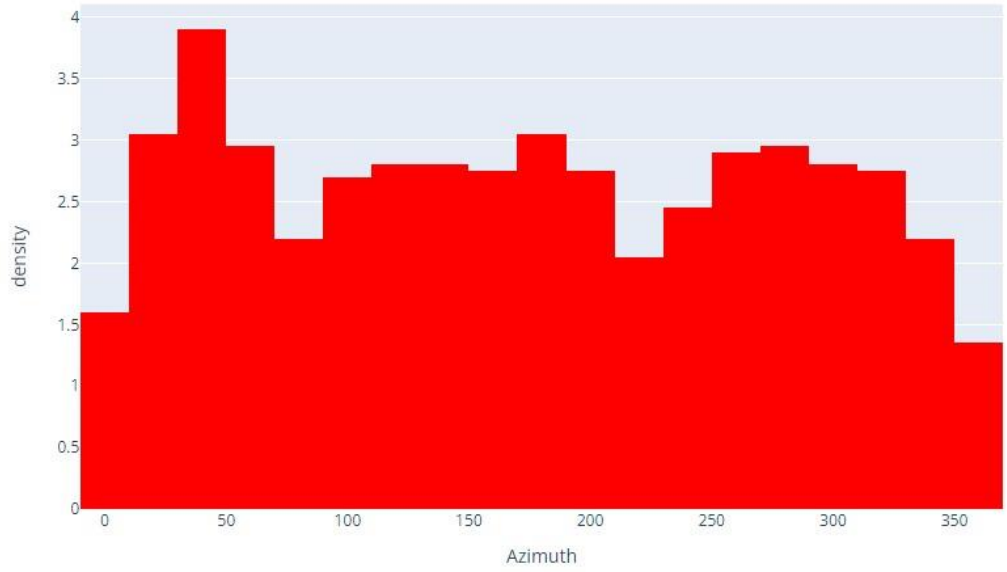


Figure 5. Azimuth Vs. Density Histogram Graph

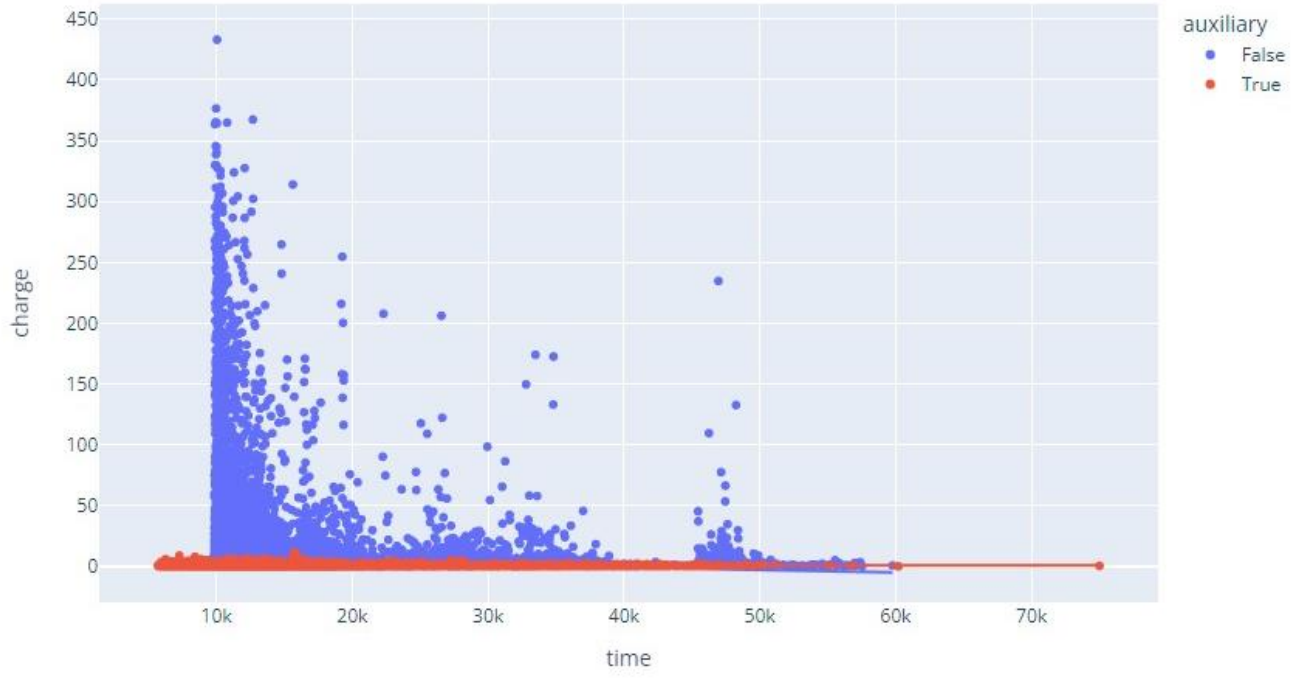


Figure 6. Time Vs Charge Scatter plot

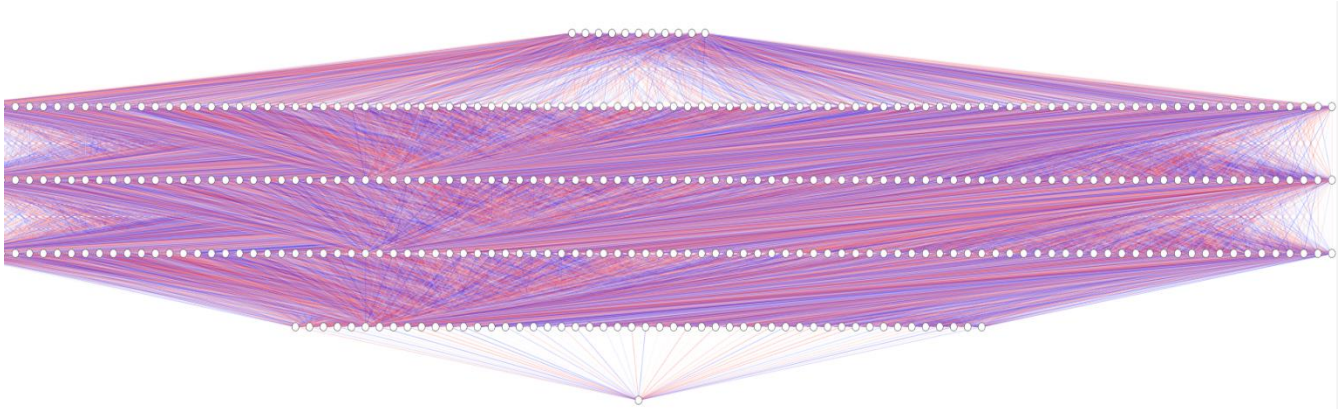
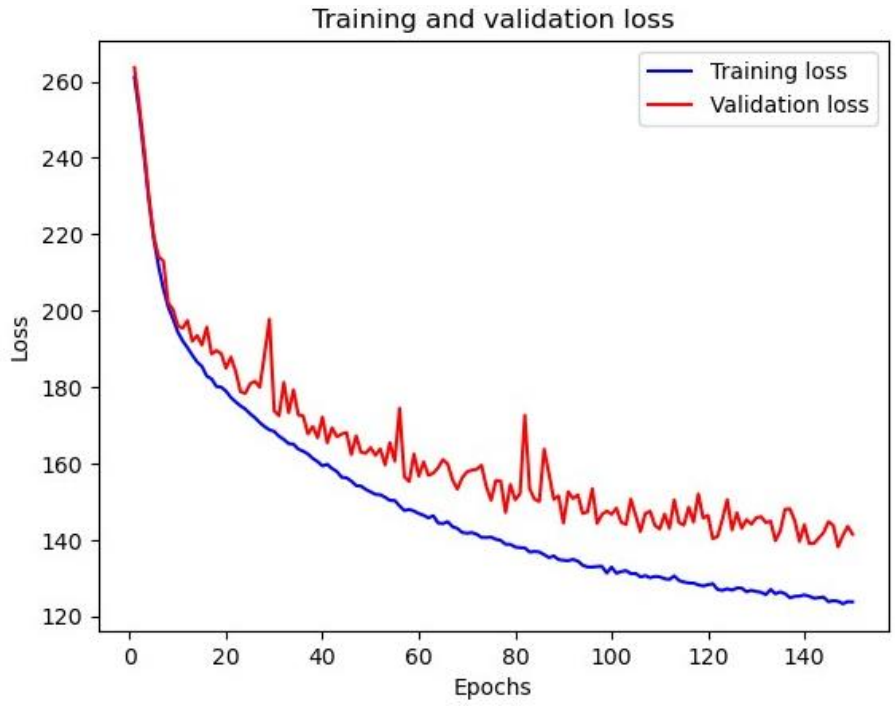


Figure 7. Visualization of our Neural Network



.

Figure 8. Epochs Vs. Loss graph

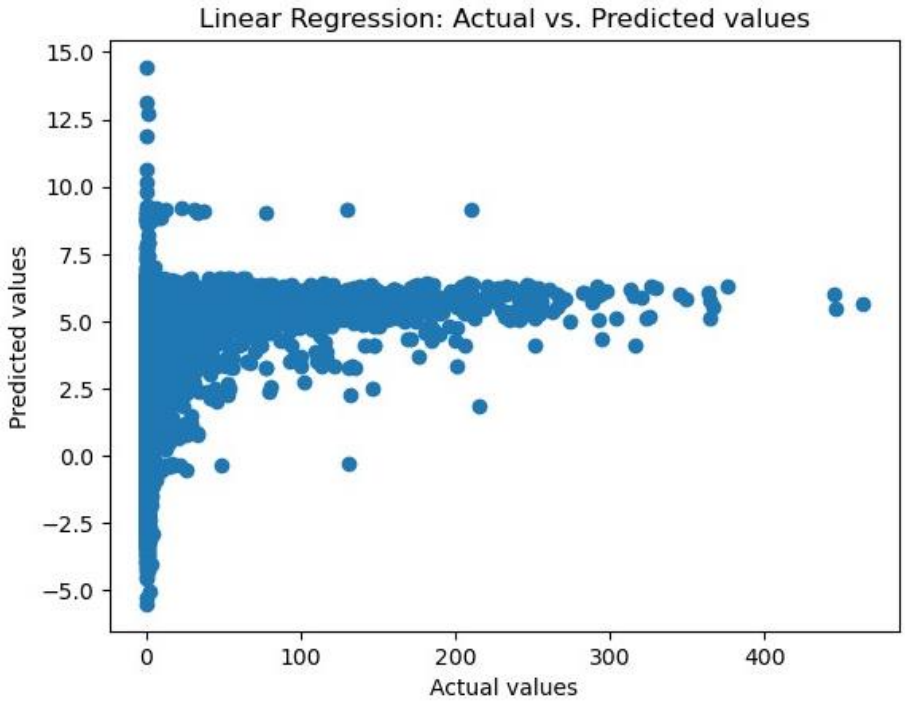


Figure 9 Linear Regression's Predicted Values Vs. Actual Values.

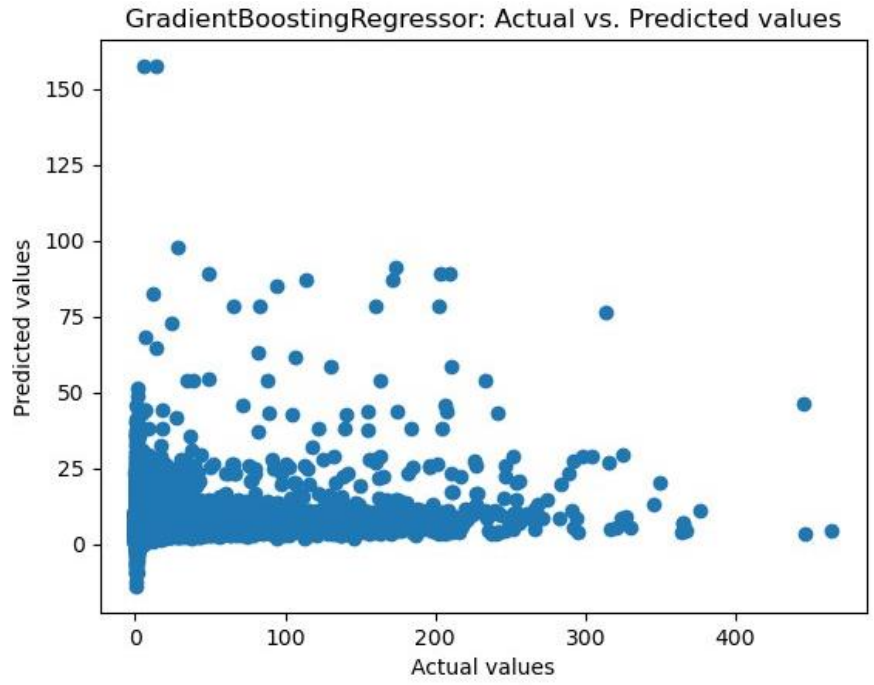


Figure 10. Gradient Boosting Regressor's Predicted Values Vs. Actual Values

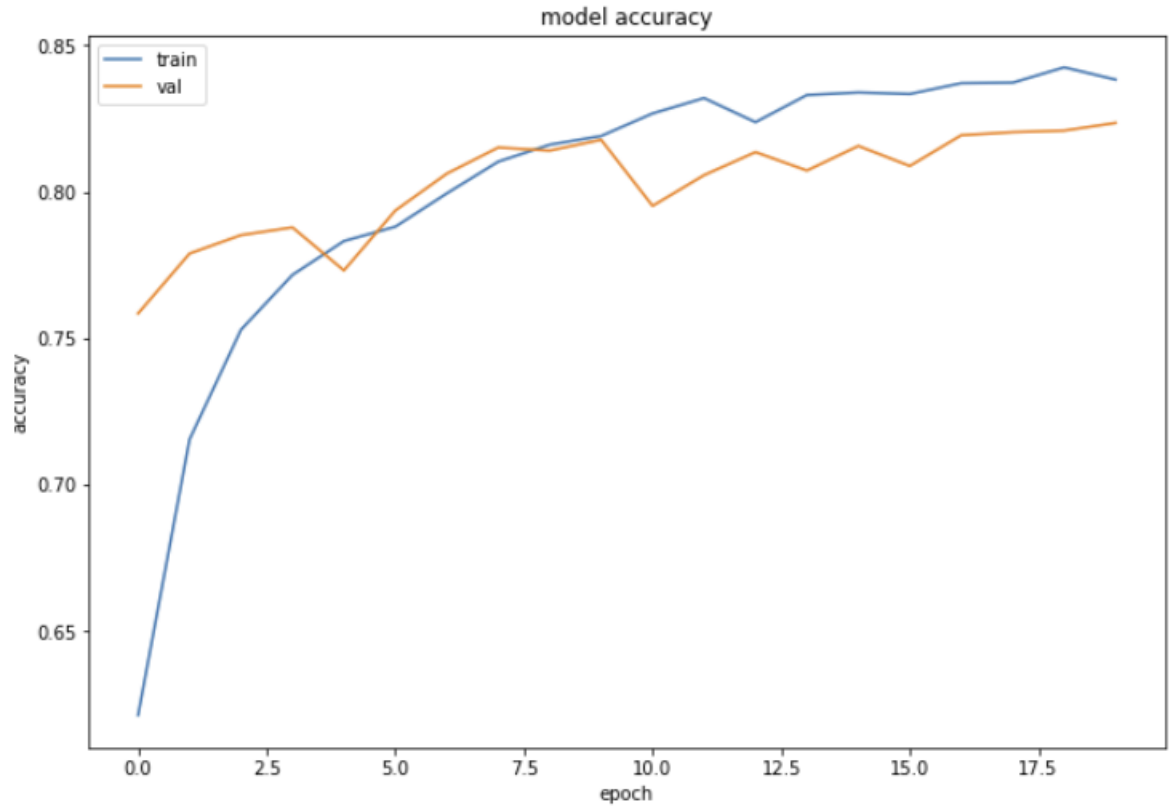


Figure 11. Output Sample of a Neural Network Epochs Vs. Accuracy Graph

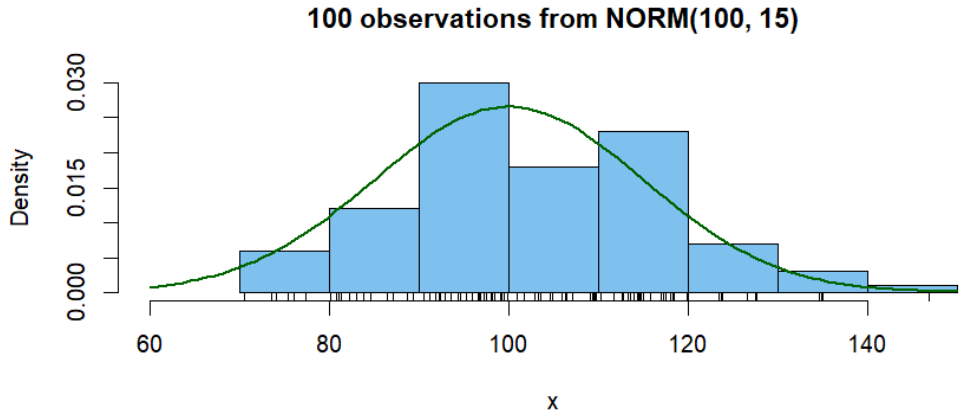


Figure 12. Output Sample of a Density Distribution

11. **Code**

11.1 Statistical Models

```
import numpy as np
import statsmodels.stats.multicomp as mc
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

print("\n\n... LOAD SAMPLE SUBMISSION DATAFRAME FROM PARQUET FILE ...\n")
ss_df = pd.read_parquet(os.path.join(DATA_DIR, "sample_submission.parquet"))
display(ss_df)

print("\n... BASIC DATA SETUP STARTING ...\n")
print("\n\n... LOAD TRAIN META DATAFRAME FROM PARQUET FILE ...\n")

train= pd.read_parquet(os.path.join(DATA_DIR, "train_meta.parquet"))
train["n_time_steps"] = train["last_pulse_index"]-train["first_pulse_index"]
display(train)

print("\n\n... LOAD SENSOR GEOMETRY DATAFRAME FROM CSV FILE ...\n")
s_geo_df = pd.read_csv(os.path.join(DATA_DIR, "sensor_geometry.csv"))
display(s_geo_df)

print("\n\n... LOAD BATCH1 DATAFRAME FROM TRAIN PARQUET FILE ...\n")
batch1= pd.read_parquet(os.path.join(DATA_DIR, 'train/batch_1.parquet'))
display(batch1)

print("\n\n\n... BASIC DATA SETUP FINISHED ...\n\n")


def remove_outliers(x):
    Q1 = np.percentile(x, 25)
    Q3 = np.percentile(x, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return x[(x > lower_bound) & (x < upper_bound)]

```

```
def get_batches(data_dir):
    n = 100
    charges = []
    for i in range(1, n+1):
        batch_file = os.path.join(data_dir,
   'train/batch_{}.parquet'.format(i))
        batch_data = pd.read_parquet(batch_file)

        batch_charge = batch_data['charge'].to_numpy()
        batch_charge = remove_outliers(batch_charge)

        batch_charge = np.random.choice(batch_charge, size=250000,
   replace=False)
        charges.append(batch_charge)
    return charges

charges = get_batches(DATA_DIR)

f_statistic, p_value = f_oneway(*charges)

if p_value < 0.05:
    print("The means of the populations are not equal at a significance level
   of 0.05.")
else:
    print("The means of the populations are equal at a significance level of
   0.05.")


# Concatenating all batches into a single array
all_charges = np.concatenate(charges)

#Creating a list of labels for each batch
labels = []
for i in range(len(charges)):
    labels += ['Batch {}'.format(i+1)] * len(charges[i])

# Performing Tukey's HSD test
tukey_result = mc.pairwise_tukeyhsd(all_charges, labels, alpha=0.05)

# Printing the results
print(tukey_result)

# merging Data
batch1 = batch1.reset_index()
```

```
batch1[['x', 'y', 'z']] = s_geo_df.loc[batch1.sensor_id].reset_index()[['x', 'y', 'z']]
batch1 = batch1.reset_index(drop=True)
batch1[['first_pulse_index','last_pulse_index','azimuth','zenith','n_time_steps']] = train.loc[batch1.event_id].reset_index()[['first_pulse_index','last_pulse_index','azimuth','zenith',   'n_time_steps']]

data = batch1.sample(n=250000, random_state=42)
data = data.reset_index(drop=True)
data

#Saving our data
data.to_csv('batch1_data_large.csv', index=False)

  # Preprocessing
  le = preprocessing.LabelEncoder()

  X = data[["sensor_id", "time", "auxiliary", "x", "y", "z", "first_pulse_index", "last_pulse_index", "azimuth", "zenith", "n_time_steps"]]
  y = data["charge"]

  X['auxiliary'] = le.fit_transform(X['auxiliary'])
  X

  std = StandardScaler()
  X_std = std.fit_transform(X)

  X = X.to_numpy()
  y = y.to_numpy()
  X,y,X_std

  # Splitting our Data for train/test
  X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                   test_size = 0.3,
                                                   random_state=0)

  #GBR

  gbr = GradientBoostingRegressor(random_state=0)
  gbr.fit(X_train, y_train)

  mse = mean_squared_error(y_test, gbr.predict(X_test))
  r2 = r2_score(y_test, gbr.predict(X_test))
```

```
129.
130.  mse, r2
131.
132.  # Splitting our Data for train/test Standardized.
133.
134.  X_train,X_test,y_train,y_test = train_test_split(X_std,y,
135.                                                   test_size = 0.3,
136.                                                   random_state=0)
137.

138.  plt.scatter(y_test, gbr.predict(X_test))
139.  plt.xlabel("Actual values")
140.  plt.ylabel("Predicted values")
141.  plt.title("GradientBoostingRegressor: Actual vs. Predicted values")
142.  plt.show()
143.
144.  # Linear Regression
145.
146.  reg = LinearRegression()
147.  reg.fit(X_train, y_train)
148.
149.  mse = mean_squared_error(y_test, reg.predict(X_test))
150.  r2 = r2_score(y_test, reg.predict(X_test))
151.
152.  mse, r2
153.
154.  y_pred = reg.predict(X_test)
155.
156.  plt.scatter(y_test, y_pred)
157.  plt.xlabel("Actual values")
158.  plt.ylabel("Predicted values")
159.  plt.title("Linear Regression: Actual vs. Predicted values")
160.  plt.show()
161.
```

11.2 Neural Network Code

```
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
from tensorflow import keras
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error,r2_score

data = pd.read_csv("/kaggle/input/batch1-large/batch1_data_large.csv")
data

def remove_outliers(x):
    Q1 = np.percentile(x, 25)
    Q3 = np.percentile(x, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return x[(x > lower_bound) & (x < upper_bound)]


le = preprocessing.LabelEncoder()

X = data[["sensor_id", "time", "auxiliary", "x", "y", "z", "first_pulse_index",
"last_pulse_index", "azimuth", "zenith", "n_time_steps"]]
y = data["charge"]

X['auxiliary'] = le.fit_transform(X['auxiliary'])
X

std = StandardScaler()
X_std = std.fit_transform(X)

X = X.to_numpy()
y = y.to_numpy()
X,y,X_std, X_std.shape


X_train,X_test,y_train,y_test = train_test_split(X_std,y,
                                                 test_size = 0.3,
                                                 random_state=0)



model = keras.Sequential([
    layers.Dense(100, activation="relu", input_shape=(11,)),
    layers.Dense(100, activation="relu"),
    layers.Dense(100, activation="relu"),
    layers.Dense(50, activation="relu"),
    layers.Dense(1)
])
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                loss='mean_squared_error')


model.summary()

# creating a tf.data.Dataset from our training data
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
val_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test))

history = model.fit(
    train_dataset,
    epochs=200,
    verbose=2,
    validation_data=val_dataset
)

## Saving model
#save_locally =
tf.saved_model.SaveOptions(experimental_io_device='/job:localhost')
#model.save('./model', options=save_locally) # saving in Tensorflow's
"SavedModel" format



training_loss = history.history['loss']
validation_loss = history.history['val_loss']

# Ploting the training and validation loss
epochs = range(1, len(training_loss) + 1)
plt.plot(epochs, training_loss, 'b', label='Training loss')
plt.plot(epochs, validation_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R2 score:", r2)
```