THIRD YEAR LABORATORY

STUDY OF MUON NEUTRINO OSCILLATIONS USING MICROBOONE DATA

This experiment has a significant computer programming component in python.

1 Aims

1. **Introduction to neutrino physics at MicroBooNE:** To use and understand data analysis techniques used in neutrino analyses. These techniques can be applied both to general neutrino experiments and some that are specific to MicroBooNE.
2. **Neutrino Oscillations:** To understand the theory behind neutrino oscillations and how neutrino oscillations can be observed and measured in practice.
3. **MicroBooNE detector:** Understand the MicroBooNE detector as well as general modern neutrino detectors.

2 Objectives

In this experiment, we learn about the basics of neutrino physics and then combine this knowledge with computing techniques to investigate neutrino oscillations. This experiment assumes basic knowledge of the python syntax.

1. You will learn some basic neutrino physics and observe some digital images of neutrino events from MicroBooNE. With these images, you will get the chance to look through different events and attempt to recognise when you see a neutrino event, a shower and/or a track.
2. You will use data frames to sort and analyse the data. This is where you will start doing some coding in python, and will provide good experience in data manipulation within a particle physics context.
3. You will be given a brief introduction to machine learning, including implementing decision trees.
4. You apply 'cuts' to the provided data, where you select events desired in our sample and remove background events. Applying effective cuts requires an understanding of the variables describing the data. After applying these requirements, you will write a python program to calculate the reconstructed neutrino energy.
5. You will plot the neutrino energy for different data sets as histograms and scale them correctly according to their weight.
6. You will apply an oscillation analysis to the simulated sample while doing a χ^2 test between the data and the simulation.

3 Background reading

Potential sources for background reading:

- A good summary document on neutrino oscillations written by Dr Steve Boyd can be found here.

- The MicroBooNE website also contains valuable information about the experiment, and can be found here.
- *PDG Review of Particle Physics* is an exceptionally rigorous review documenting the current state of Particle Physics, including Neutrino Masses, Mixing and Oscillations (page 285). The document can be found here.

The relevant PDFs from these sites are also saved locally under the *Background Reading* directory.

4 Neutrinos

Neutrinos are neutral leptons with spin quantum number 1/2 that only interact weakly and were originally assumed to be massless in the standard model [11]. The first indication of neutrinos came in studies of β decays in the early 1900s. There are two separate type of β decays; $\beta^{\pm}$, with β^{+} decays being when a proton within the atomic nucleus decays to a neutron, emitting a positron and neutrino: $p \rightarrow n + e^{+} + \nu_e$

Lepton	Charge	Spin	Mass (GeV/c^2)	Generation
electron (e^{-})	-1	1/2	0.00051	1
muon (μ^{-})	-1	1/2	0.10565	2
tau (τ^{-})	-1	1/2	1.77686	3
electron-neutrino (ν_e)	0	1/2	Non-zero	1
muon-neutrino (ν_μ)	0	1/2	Non-zero	2
tau-neutrino (ν_τ)	0	1/2	Non-zero	3

Table 1: Table of general lepton properties, including the different **flavours**: electron, muon, and tau

Early experiments on β decays found that the emitted electron does not have a discrete energy as predicted, but instead follows a continuous distribution. In 1930 Wolfgang Pauli proposed a solution to this problem in which the nucleus also emits a tiny undetectable particle with no charge and spin quantum number 1/2 to conserve angular momentum. By 1934, Enrico Fermi published a famous paper detailing his β decay theory in which he incorporated the neutrino [10]. It was not before 1956 that Los Alamos physicists Frederick Reines and colleague Clyde Cowan confirmed the existence of Neutrinos via the detection of antineutrinos [7].

4.1 The Solar neutrino problem

Nuclear fusion is another reaction known for its production of electron-neutrinos, which occurs in abundance within our own Sun. Due to their lack of interaction with matter, it was understood that neutrinos would pass through the Sun and reach Earth unperturbed. With this in mind, the 'expected electron-neutrino' flux arriving at Earth from the Sun was determined. This computation was shown to be at odds with experimental data, as approximately a third of the expected flux was detected across multiple experiments [4].

A solution to this discrepancy between the data and theory was the introduction of **neutrino oscillation**, in which the individual neutrino flavours can oscillate between one another ($\nu_e \rightarrow \nu_\mu$, $\nu_\mu \rightarrow \nu_\tau$, etc). This was verified by multiple neutrino experiments and resulted in the 2015 Nobel Prize being awarded to Takaaki Kajita and Arthur McDonald for '*the discovery of neutrino oscillations, which shows that neutrinos have mass*' [1].

4.2 Neutrino Oscillation

As stated above, the existence of neutrino oscillation necessitates that they must have mass, as the theory posits that **neutrino flavour eigenstates** are considered as combinations of **mass eigenstates**. We will not go into detail on how this works, but if you wish, you can explore this topic on your own. Let us, for now, assume that neutrinos have a non-zero mass. Neutrino oscillation, the phenomenon of one flavour of neutrino (ν_e, ν_μ, ν_τ) oscillating into another flavour, was proposed by Bruno Pontecorvo in 1967 [9]. In this framework it is important to distinguish between flavour eigenstates as (ν_e, ν_μ, ν_τ), produced in **weak interactions** and the mass eigenstates which are the those with definite masses (ν_1, ν_2, ν_3). The flavour eigenstates can be thought of as linear superpositions of the mass states described by the PMNS matrix [8], U^{PMNS}:

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{pmatrix} = U^{\mathrm{PMNS}} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3. \end{pmatrix} \tag{1}$$

In the two-flavour approximation, we can write the relation between the flavour eigenstates and the mass eigenstates as [6]

$$\begin{pmatrix} \nu_\alpha \\ \nu_\beta \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \end{pmatrix} \tag{2}$$

From this, we can find the time propagation of a weak state as

$$|\nu(t=0)\rangle = |\nu_\alpha\rangle = \cos\theta|\nu_1\rangle + \sin\theta|\nu_2\rangle, \tag{3}$$

$$|\nu(t)\rangle = e^{iq_1\cdot x}\cos\theta|\nu_1\rangle + e^{iq_2\cdot x}\sin\theta|\nu_2\rangle, \tag{4}$$

where $q_{1,2}$ are the time propagated four momenta of the two states $E_{1,2}t - P\cdot x$.

Exercise 1: From this you should be able to derive the probability of the two-flavour oscillation formula $P(\nu_\alpha \to \nu_\beta) = |\langle\nu_\beta|\nu(t)\rangle|^2 = \sin^2(2\theta)\sin^2(\frac{\Delta m^2_{21}L}{4E})$, where $\Delta m^2_{21} = m^2_2 - m^2_1$.

Hint: *first show that* $q_i \cdot x = \frac{m_i^2 L}{2E}$ *via the relativistic energy-momentum relation, using Taylor expansions were needed.*

Using natural units, this probability can be written as:

$$P(\nu_\alpha \to \nu_\beta) = \sin^2(2\theta)\sin^2(1.27\frac{\Delta m^2_{21}(\mathrm{eV})^2 L(\mathrm{km})}{E(\mathrm{GeV})}) \tag{5}$$

This will be useful in later exercises.

Write down what each of the variables in equation 5 mean. What is the significance of the two separate sine components? What effect do they have on the resultant probability?

5 An Introduction to MicroBooNE

In this experiment, you will analyse data from the neutrino detector MicroBooNE. MicroBooNE is a 170-ton liquid argon time projection chamber (TPC). These detectors are some of the best detectors suited for modern neutrino experiments and therefore you should ensure you have an

understanding of their operation and why they are the preferred method for emerging physics analyses. The MicroBooNE detector operates in a muon neutrino beam at Fermilab which is created by 'firing' protons at a Beryllium target.

5.1 Liquid-argon TPCs

Liquid-argon TPCs use a large container of liquid argon to analyse neutrino interactions. We cannot detect neutrinos directly as they have no charge and do not emit any Cherenkov or Bremsstrahlung radiation. Therefore, what we do is detect charged particles that have been created during an interaction of a neutrino with the argon and infer the neutrino's properties. An example of this is 'quasi-elastic scattering' where a neutrino interacts with a neutron from the liquid argon atoms producing a proton and a corresponding charged lepton. This interaction can be seen in Figure 1. This is a charged current (CC) interaction as it is mediated by a W boson and will be our signal in this analysis.

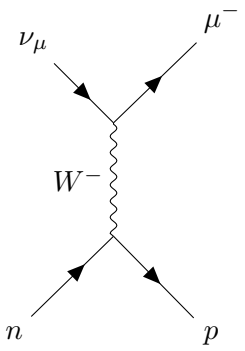

Figure 1: The quasi-elastic neutrino scattering of a neutron. This is a charged current interaction mediated by a W boson.

After a neutrino interaction has occurred within the argon, there will be both scintillation photons emitted and ionization electrons from the charged particles hitting the argon atoms. To analyse the charged particles involved in the neutrino interaction, an electric field is applied to the detector that will drift the particles to one side of the detector. The detector has three collection wire planes that together with the time projection obtained from the detection of scintillation photons can be used to obtain a 3-dimensional picture of a neutrino interaction. A diagram of the process can be seen in Figure 2, and an animation demonstrating the process can be seen here, courtesy of FermiLab.

Exercise 2: What other interactions can neutrinos undergo? Are any of these applicable within the detector? If you know how, draw those applicable as Feynmann diagrams.

6 Initial setup

6.1 Installing packages

When running through the code, you may find that your local machine does not have the required python packages. If you are using the 3rd year lab machines, this can be easily resolved. First, open the command terminal and using the command `pip install` type:

```
pip install jupyterlab
    ...     numpy
    ...     uproot3
    ...     pandas
    ...     seaborn
    ...     scipy
    ...     scikit-learn
    ...     matplotlib
```

There may be more packages required for your code to run. If you have any issues with this, please contact any of the lab support.

6.2 Downloading and using the lab scripts

If you are using the 3rd year lab machines, it is recommended that you save all data from this lab within your 'onedrive' folder. First, download the `uboone_lab_student_copy` repository from github, found here and read through the `README.md`. Extract the repository to whatever directory you wish to work from.

Then you will need to download a large data file: `bnb_run3_mc_larcv.h5`. This will be provided either via a usb stick or onedrive link. Insert this file into the above directory in the 'data' folder. If the files data_flattened.pkl & MC_EXT_flattened.pkl within the data folder appear to be very small (bytes rather than MB), it means they have been downloaded incorrectly and should be downloaded manually from their github pages linked above.

Once you have all of this installed, you should be able to open up jupyterlab within command prompt by typing: `python -m jupyterlab`. Then use the interface to navigate to the relevant directory with your lab data.

7 Event Displays

Event displays are a useful tool used by MicroBooNE researchers to visualise the data we obtain from reconstructed interactions. In these event displays, you can see coloured pictures of what is happening in the detector. The colour map from blue to red indicates the amount of charge deposited by the particles involved in an interaction. By observing these events, one can distinguish different interactions and properties such as how shower-like or track-like an interaction was.

Exercise 3:

- **Go to the event_display.ipynb file, run the code and look at the event display produced. You can select which wire plane to observe the event from, and can use the in-built plotting tools to look at individual interactions or 'tracks'. Use the function `list_entries` to look at different 'images' from the dataset.**
- **Due to MicroBooNE's location above ground, it detects an excess of cosmic muons. These muons are consequently very easy to identify within the event displays (be-**

ing the most abundant particle tracks). What characteristics do they all have in common? Justify these characteristics using your knowledge of Muons[1].

- **Look at different events and try to recognise specific features that can be used to characterise different particle interactions.**

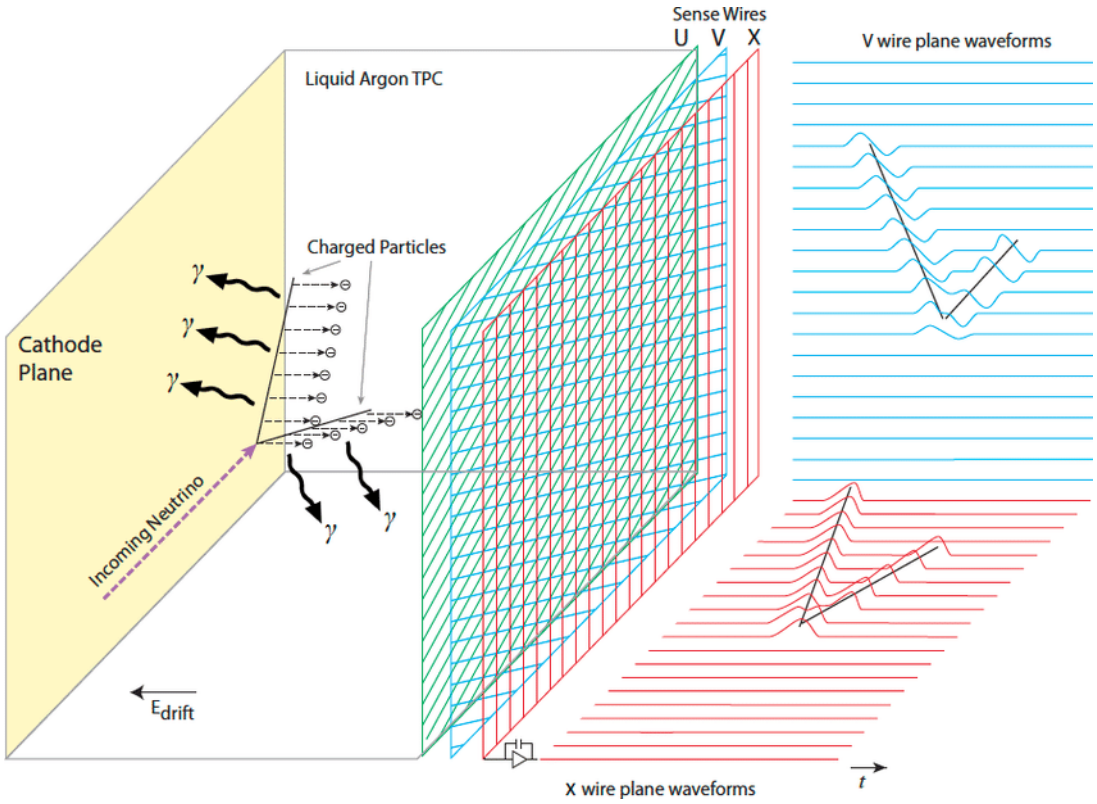

Figure 2: A schematic of the operation of a Liquid Argon Time Projection Chamber. The time projection from the scintillating photons together with the three wire planes are combined to produce a three-dimensional reconstruction of an interaction [2].

8 Variables

In the samples we will be using, many variables are defined, some of which we will not need. In Table 2 a list of the variables we will take into account is given with a description of what they do.

[1] This paper may be useful for seeing some examples of what particles look like within the MicroBooNE detector.

Variable Name	Description
$\mathrm{vertex}_{\mathrm{x,y,z}}$ `reco_nu_vtx_sce_x,y,z`	Reconstructed position of the vertex of an interaction in the detector (cm).
$\mathrm{track}^{\mathrm{start}}_{\mathrm{x,y,z}}$ `trk_sce_start_x,y,z_v`	Reconstructed position of the start of an identified track in the detector. (cm)
$\mathrm{track}^{\mathrm{end}}_{\mathrm{x,y,z}}$ `trk_sce_end_x,y,z_v`	Reconstructed position of the end of an identified track in the detector (cm).
$\mathrm{L}_{\mathrm{track}}$ `trk_len_v`	The distance between the longest track in an event and the reconstructed neutrino vertex (cm).
$\mathrm{d}_{\mathrm{track}}$ `trk_llr_pid_score_v`	Log likelihood score used to find the ID of the particle. For example, if it is a muon or a proton.
$\mathrm{L}_{\nu,\mathrm{cosmic}}$ `_closestNuCosmicDist`	Distance between our detected Neutrino track and closest cosmic muon track within the detector.
$\mathrm{S}_{\mathrm{topological}}$ `topological_score`	A score which determines how much a signal in the detector looks like a track. This is also a variable that is determined by machine learning, where the machine has been trained with track-like events.
$\mathrm{S}_{\mathrm{track}}$ `trk_score_v`	A score which determines how much a signal in the detector looks like a track. This is also a variable that is determined by machine learning, where the machine has been trained with track-like events.
E_{n} `trk_energy_tot`	Reconstructed energy of the neutrino in the interaction.

Table 2: A list of variables (and their names within the data structure) we are going to look at in this experiment and a description of their function.

9 Setting up

If you need to install any of the packages imported, you can type “pip install package” into an empty cell in the notebook or within your terminal, where “package” is the name of the package.

9.1 Running the example code

You are given a template for this lab in the form of *MainAnalysis.ipynb*, which will give you access to a couple of useful scripts as well as the code structure for some components of the lab.

Exercise 4: Run the example code. Look through the code and the figures that are produced and see if you can understand what the code does.

The samples used in this Python file are called “MC_EXT” and “data”. The sample “MC_EXT” is a combination of the predicted distribution of the neutrino events from our detector, developed using *Monte Carlo* simulations; and real data taken from MicroBooNE when the muon neutrino beam is turned off (interactions occurring due to EXTernal particles). The sample “data” is the real data taken at MicroBooNE.

10 Particle Identification within MicroBooNE

As shown in **exercise 3**, recognising different particle interactions within the detector for each and every event is a process that (by-hand) would take a very long time, but is key to parti-

cle physics analysis (what use is a detector if we don't know what its detecting?). Therefore, automation processes are implemented to increase the speed at which interactions within the detector can be 'classified' or categorised. These classification processes are relatively complex and use a variety of tools to identify as many particles within the detector as possible.

As is the case in most areas of cutting-edge research, machine learning is a tool that is used extensively throughout particle physics. Here we will be implementing a very simple machine learning method known as **Random Decision Forests** to try and classify our particle interactions, as well as determine what variables are most important in distinguishing between them.

10.1 Data visualisation and manipulation

First, lets look at our data and learn how to use `pandas`, a python library that allows for intuitive manipulation and analysis of data. You should have already seen `pandas` in action within the template, printing off the 'head' of our MicroBooNE and Monte Carlo data.

You may notice that we have some variables in our Monte Carlo dataframe that are not described in table 2, these being *weight & category*. Category tells us the type of interaction seen within the detector as shown in table 3, and weight is a scaling factor applied to each Monte Carlo event provided by the simulation framework.

Category	Interaction
4	Cosmic
5	Out Fid. Vol.
7	EXT
10	ν_e CC
21	ν_μ CC
31	ν NC

Table 3: Different `categories` with respect to their interactions

Exercise 5:

- **You've already been told that $CC\ \nu_\mu$ (Charged current muon neutrinos) are our signal events, but what make up the rest of our background events? Identify the different interactions/event types displayed in the table 3. The more abstract sounding event types (Out. Fid. Vol., EXT) are discussed in many MicroBooNE papers.**
- **We will initially look at our data using the `pairplot` function from the `seaborn` package. `pairplot` shows pair-wise relationships in the dataset, which can be useful for trying to visualise correlations. Take some time to understand what the `pairplot` graph is showing you.**
- **Our signal events drown out any interesting trends that may be visible in the background events. Using pandas, create a dataframe that is *deep copy* of `MC_EXT_VIS` and remove all signal (category == 21) events from this dataframe. Plot another `pairplot` using this new dataframe.**
- **Are any trends/correlations visible within the data now? Why might it still be difficult to pick out trends by eye? How could this be resolved for visualisation purposes?**
- **Bonus 1: Try to apply some of your ideas to make visualising trends more clear.**[2]

[2]Be careful when playing around/reducing the size of dataframes in jupyter notebook! It's very easy to carry forward modifications to your dataframe that may be unintentional.

- **Bonus 2: Add new variables to your plots to see if you can notice any other trends.**

10.2 Decision Trees

Exercise 5 should have demonstrated that while visualising data can be useful for better understanding it, when working with large data sets and many parameters it becomes much more difficult to find trends or correlations by eye. This is were the application of machine learning becomes appropriate to find trends, correlations, and other dependencies/relations within our data that can help us better classify our particles.

Due to the nature of this lab, we will be looking at a rather simple example of machine learning, specifically **decision trees**. Decision trees (in this context) produce a collection of simple rules for partitioning our variable space that allow for the separation of our different interactions. For example, if we applied a selection partition on `topological_score` at 0.4 for our initial Monte Carlo data, we would see that the majority of our muon neutrino CC interactions would be on one side of our selection.

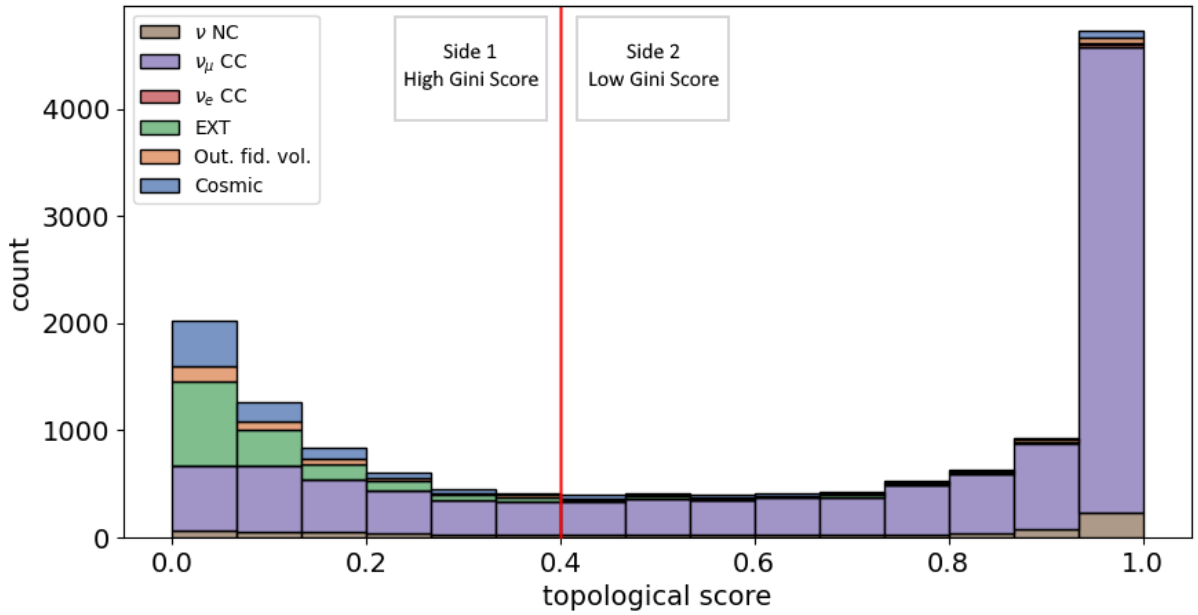

Figure 3: A plot of our topological score, now as a histogram for our different particle interactions, with the proposed partition at `topological_score = 0.4`. The hypothetical *Gini* scores for each side are represented in the plots as well.

Decision trees allow for such selection partitions to be made in sequence for large sets of data, across multiple variables for classification purposes. As this process of producing selection partitions is done automatically, we need a way to determine how good these partitions are at separating our different interactions. Decision trees do this by applying a 'criterion' for these selections that determine the 'accuracy' or how well the partitions classify the data, such as *Gini* or *Entropy*.

We will be using the *Gini* criterion: $G_{1,2} = 1 - \sum_{class} p^2_{class}$, where p_{class} is the probability of finding said class on one side of a partition (sides 1 & 2). *Gini* looks to minimise its own value with each partition, with 100% accuracy being when all partitions lead to $G_N = 0$. This is a very brief explanation of *Gini*, so I would **highly** recommend looking at slides on Decision trees provided by Dr. Guillermo Hamity from the University of Edinburgh for a more in-depth look at how exactly such processes work and some excellent visualisation of the Decision Tree

process, but for today we are just hoping to get to grips with implementing decision trees. His slides can be found here[3].

A problem with decision trees is that they are very prone to **overfitting**, this is when the partitions are tailored very specifically for **only** one dataset due to too many partitions being allowed (too much training), meaning that you're model may be less accurate for another dataset of the same simulation. To counteract this issue, we will use a method of **bootstrap aggregation** or "bagging" called **random decision forests**. This is the process of producing multiple independent decision trees which will make different random choices for each tree across different features to ensure that they're distinct, and aggregating their results.

Exercise 6:

- **To simplify the classification process, generate a new dataframe,** `MC_EXT_ML`**, that doesn't have any muon or electron neutrino events. This means we're only considering** *Cosmic, Out Fid Vol, EXT and* ν *NC* interactions**.**

For the decision trees to work, it needs 'training data' (X), which will be the interaction data we want to train or model on (`MC_EXT_ML` without the `category` column) and 'truth data' (y), or the true classifications of our interactions (the `category` column). The training and truth data have been separated for you already.

- **Display the** `shape` **of X and y. Do these shapes make sense? Print them out and investigate to better understand the data.**
- **A good way to check for overfitting is to produce a 'test dataset'. We will use a slice of the training data that is kept aside (the model isn't trained on it) to validate that our random forest model works adequately on data it hasn't been trained on. Use the function** `train_test_split` **to produce 4 datasets:** `x_train, x_test, y_train, y_test`**, with the split between training and test being 80/20.** ***Hint: Look up the documentation for scikit-learn functions when they're in use to understand what they're doing. This is generally good practice regardless of what you're doing.***
- **Now you can produce your** `RandomForestClassifier` **and fit/train it to your data! Create your model** `rf` **with 1000 trees, and a max_depth of 8, using the** ***Gini*** **criterion. Fit your** `RandomForestClassifier` **to your data using** `rf.fit`**.**
- **The code for determining the accuracy of your results for both your training and testing datasets has been provided. Comment on the accuracy.**
- **Bonus: What do you think would occur if you provided more trees/depth[4] for your** `RandomForest` **model? You can try different depths and number of trees if you'd like, but don't spend too much time waiting for code to run!**

Now that you have a model that can be used to predict an interaction based on your variables, how well does it perform? Accuracy isn't the best way of quantifying this, so now lets produce a **confusion matrix**. A confusion matrix plots the probability that your truth data matches what your model predicts for your dataset. For a 100% accurate model, we would expect the diagonal boxes of a confusion matrix to contain 1s. An example can be seen in figure 4.

- **Use the function** `ConfusionMatrixDisplay` **to produce a confusion matrix for model** `rf`**, using** `y_test` **&** `y_pred` **datasets.**
- **Comment on your results. Why might some interactions be confused with others by your model? Use your knowledge of each interaction category to justify the**

[3]The link should start the PDF at page 28. Jump forward to this page if this doesn't work on your device.
[4]depth here means number of partitions

models (in)accuracy for each category.

Our model allows us to determine how **important** each variable is in classifying our data, as in which variables made the most impact on allowing the model to distinguish between interactions. This can be useful to know when planning to manipulate our real MicroBooNE data to single out certain interactions.

This is just a basic introduction to ML, with much more complex models implemented in the field to do what we've just done. MicroBooNE analysis often uses boosted decision trees and more complex neural networks to characterise tracks seen within the data, and determine what variables in real data affect said classifications. There are a myriad other uses and implementations of ML throughout physics, but sadly we don't have time to go into them here.

If you're curious at what a simple neural network model would look like when trying to complete the same task you've just done, look at the output of `keras_example.ipynb`, which is a basic neural network deployed to achieve the same goal of classification.

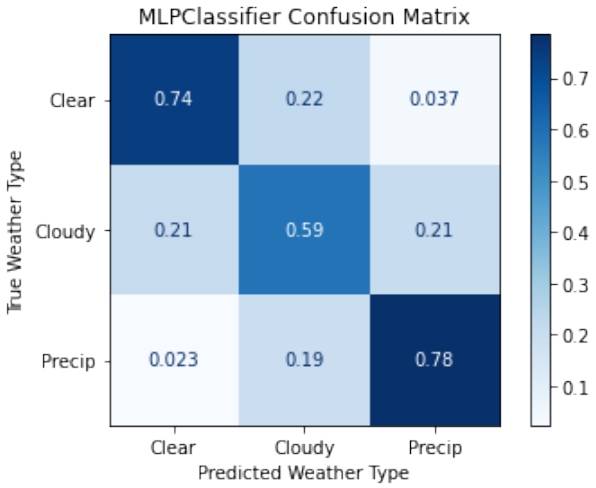

Figure 4: A confusion matrix for a model that predicts weather patterns.

11 Selection

From this section and onward, we will refer to limits and restrictions applied to the variables as selection cuts. We want to reduce the events induced by background processes while keeping as many signal $CC\ \nu_\mu$ events as possible. One major component of the background (as described in **Exercise 3**) is cosmic rays. These interactions come from high-energy cosmic radiation. They will traverse most of the detector and usually be detected near the edge of the detector coordinates.

Exercise 7:

- **Generate histograms of the various variables outlined in the previous section. Use `seaborn.histplot` and the example given within the template to help you.**
- **Once all variables have been plotted as histograms, identify appropriate cuts that can be used to remove background and apply them using the `Selections` function provided.**

Suggestions:

- A good first cut to make is to reduce the total track energy to be less than a couple GeV, as most values larger than this are caused by tracks that are mis-interpreted or beyond the scope of what we're interested.
- Some of the variables detail the position of a vertex or the start/end points of a track. As described above, a lot of cosmic interactions occur near the edges of the detector (as should be clear in your variable histograms). Therefore, an initial cut that would make sense is to remove these interactions near the edge of the detector with cuts, consequently only considering the *fiducial volume*[5] of the detector. You might have to adjust the bin width and limits on the axes to better represent the variables and make appropriate cuts.

11.1 Efficiency

It is worth monitoring the efficiency of the selection cuts you made. After each cut you lose some information, so you should see what selection cuts affect your efficiency the most. The efficiency of the selection cuts is defined as

$$\text{efficiency} = \frac{\text{Events surviving the selections}}{\text{Total number of events}}. \tag{6}$$

You should remove as many background events as possible while keeping a relatively high efficiency.

11.2 Purity

What fraction of all events is your signal? Purity describes it. You can use your simulated sample to assess the purity. The purity is defined as

$$\text{purity} = \frac{\text{The total number of signal events that pass your selection cuts}}{\text{Total events in selected sample}}. \tag{7}$$

Exercise 8: Find a way to keep purity and efficiency high during your selection cuts. How does each cut effect the efficiency and purity?

12 Neutrino Energy

From the formula derived in Section 4.2, it is clear that the two parameters we can determine from the experiment are L and E. Since MicroBooNE has a defined baseline $L \approx 470m$, we will plot the number of events for various neutrino energies. Later on, we will use this plot to apply an oscillation analysis to the data. The systematic uncertainty in the experiment can be approximated to a flat 15% for each bin.

Exercise 9:

- **Produce a histogram of the reconstructed neutrino energy (`trk_energy_tot`) for the predicted (Monte Carlo) sample. You should have done this in exercise 7.**
- **On top of this histogram, add the systematic uncertainty of the bins. Bonus: Include statistical uncertainty in your overall uncertainty of the bins.**

[5] The fiducial volume of a detector is a sub-volume that will contain a specific number of events (usually 90%) that are present in the detector.

- **We want to compare the distribution of the reconstructed neutrino energy from our MicroBooNE data with that obtained from the simulated sample. Apply your MicroBooNE data to the histogram so that they can be visually compared.**

Your result should look something like figure 5, but feel free to experiment with your cuts to see if you can improve the purity and efficiency!

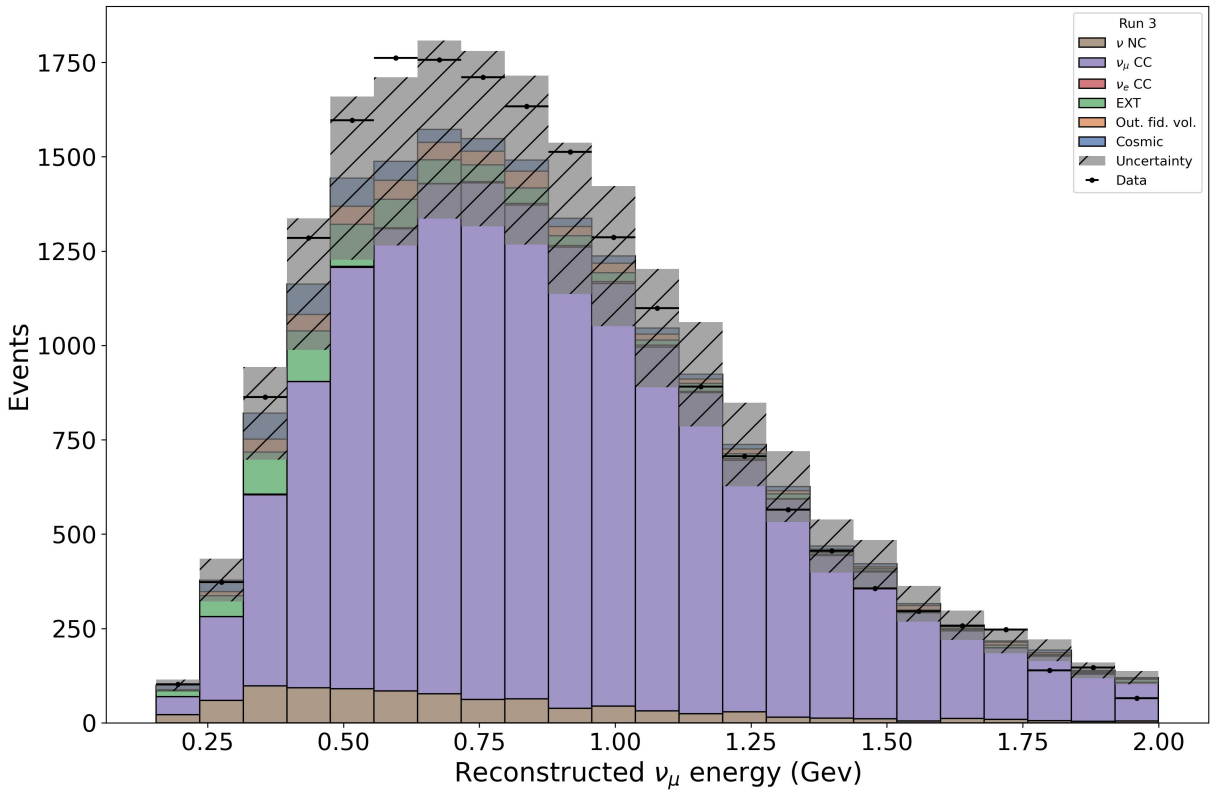

Figure 5: Reconstructed energy resolution for muon neutrinos, showing both MicroBooNE data and our Monte-Carlo Simulation Data.

13 Oscillation and fitting

In figure 5, you should see an excess of MicroBooNE (real) data compared to our Monte Carlo (simulated) data. The simulated data does not account for neutrino oscillation, so using the two-flavour oscillation probability derived in **exercise 1**, we can develop our own 'oscillation analysis', and see if including oscillations into our simulated data improves it's fit with respect to the real data.

We will do this by considering the occurrence of **muon neutrino disappearance** within our data, where the muon neutrinos (ν_μ) oscillate into another neutrino flavour that MicroBooNE couldn't detect: $P(\upsilon_\mu \rightarrow \upsilon_\beta)$. This oscillation probability can be applied to the data in figure 5 as a scaling factor, with the 'goodness' of fit determined via the minimisation of χ^2 between the real and simulated data.

Exercise 10:

- **Write a function representing the two-flavour oscillation from exercise 1 within this new context. Recall that you are no longer looking for the probability of oscillation**

from $\nu_\alpha \rightarrow \nu_\beta$, but rather the 'disappearance' (or probability to *not* oscillate) for our muon neutrinos.

- **What input variables are required for your oscillation function? The values for two of these parameters are not given and must be determined. One of these parameters are given to you, which one is it?**
- **Write a function that will apply the muon disappearance probability to your simulated data. This can be interpreted as the "scaling" of our reconstructed muon neutrino energy with respect to the different energy values per bin. Try plotting your scaled simulation energy as was done in exercise 9, or as shown in figure 6**
- **Write a function for Pearson's χ^2 that acts on the simulated data having undergone oscillation, and the real data. This is defined as:**

$$\chi^2_{\text{Pearson}} = \sum_{i=1} \left(\frac{(\mu_i(\theta) - M_i)^2}{\mu_i(\theta)} \right). \tag{8}$$

Where $\mu_i(\theta)$ are the number of events in the ith bin for your Monte Carlo data (or MC_ext) file, and M_i are the number of events in the ith bin of your MicroBooNE data.

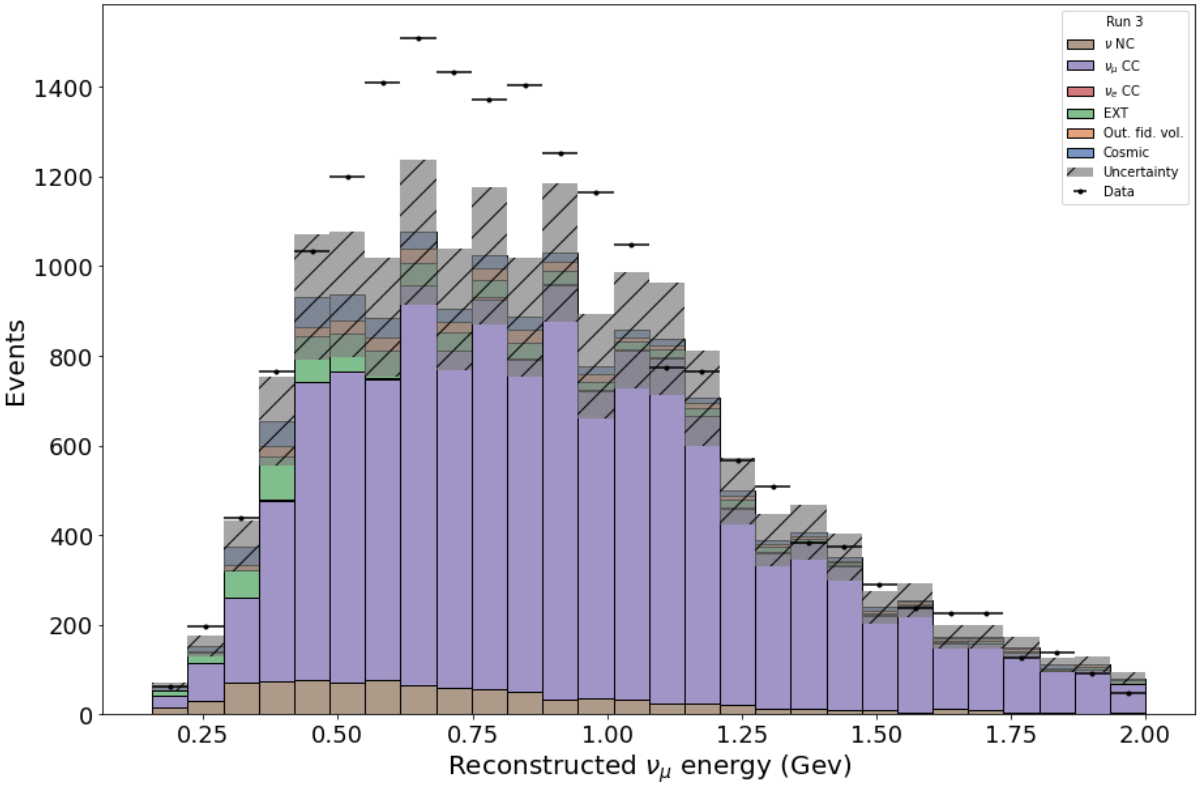

Figure 6: A demonstration of the simulated data modified by oscillation. Here it is clear that the parameters chosen do not produce a good fit between the simulated and real data, resulting in a large χ^2.

14 Oscillation parameters

From **exercise 10**, you should know the two parameters that we are allowed to vary within the oscillation function of which we don't have 'set values'. These parameters impact the oscilla-

tion probability and consequently the shape of our energy curves as shown in Figure 6. So we can determine the most optimal parameters by varying each of them across a range and determining χ^2 for each combination.

Exercise 11:

- **Find the values of χ^2 for varying oscillation parameters Δm^2_{21} and $\sin^2(2\theta)$, and make a 2D contour plot of the *reduced* χ^2 with the oscillation parameters as the axes. Use matplotlib's `contourf` function to help you. For reduced χ^2, here we will consider the degrees of freedom as the number of bins - 2. Why do we include -2?**
- **What characteristics can you see in your χ^2 plots? If no trends are visible, change your range of oscillation parameters.**
- **Bonus: Minimize χ^2 using a minimiser and extract the oscillation parameters that give the best fit for the Monte Carlo data against the real data.[6] What do these values tell you about said parameters? Does this make sense within context to the applied oscillation?**

Due to the nature of this analysis, we can't determine exact values for our desired parameters. Instead we can exclude certain parameter pairings based on their values of χ^2, producing 'exclusion zones' of high χ^2. These zones can be defined by **confidence levels** which we shall visualise as contour lines on our plots. Confidence levels act as 'upper limits' on our parameters, with a CL of 95% being equivalent to a 95% probability that our parameters allow for our simulated data to accurately describe our true data.

Exercise 12:

- **Plot lines corresponding to confidence levels of 90, 95 and 99% using matplotlibs `contour` function. These can be found in Ref [11], but can also be determined with *Chi-Square Tables* which are available online.**
- **Find the oscillation parameters that give the smallest value of χ^2 within your range and plot the point on your contour or CL plot.**

Hint: Restrain your oscillation parameters at least to between $10^{-4} \rightarrow 10^0$ for $\sin^2(2\theta)$ and $10^{-2} \rightarrow 10^2$ for Δm^2_{21}, but play around with them a bit, as you may struggle to see certain features if your range is too large!

15 3+1 neutrino model

The 3+1 neutrino model assumes the existence of a fourth sterile neutrino, with a much larger mass than the other neutrinos that isn't able to be detected within the MicroBooNE detector (sound familiar?). The PMNS matrix from Equation 1 is modified by adding a 4th eigenstate:

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \\ \nu_s \end{pmatrix} = \begin{pmatrix} U_{e1} & U_{e2} & U_{e3} & U_{e4} \\ U_{\nu 1} & U_{\nu 2} & U_{\nu 3} & U_{\nu 4} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} & U_{\tau 4} \\ U_{s1} & U_{s2} & U_{s3} & U_{s4} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \end{pmatrix}, \tag{9}$$

where the relevant matrix elements we need are given by

[6] `scipy.optimize.curve_fit` will be very beneficial here, but feel free to find your own solutions.

$$|U_{e4}|^2 = \sin^2\theta_{14}, \tag{10}$$

$$|U_{\mu 4}|^2 = \sin^2\theta_{24}\cos^2\theta_{14}. \tag{11}$$

The probability of oscillation is now given as:

$$\begin{aligned} P(\alpha \to \beta) &= \delta_{\alpha\beta} - 4(\delta_{\alpha\beta} - U^*_{\alpha 4}U_{\alpha 4})U_{\beta 4}U^*_{\beta 4}\sin^2\left(1.27\frac{\Delta m^2_{14}L}{E}\frac{(\mathrm{eV}^2)(\mathrm{km})}{(\mathrm{GeV})}\right) \\ &= \delta_{\alpha\beta} - \sin^2(2\theta_{\alpha\beta})\sin^2\left(1.27\frac{\Delta m^2_{14}L}{E}\right). \end{aligned} \tag{12}$$

Exercise 13:

- **Derive the probability of $\nu_\mu \to \nu_\mu$ disappearance for 3+1 model, and consequently find a relation for $\sin^2 2\theta_{\mu\mu}$ with respect to $\sin^2(\theta_{24})$ and $\cos^2(\theta_{14})$.**
- **Derive the probability of $\nu_e \to \nu_e$ disappearance for 3+1 model, and consequently find a relation for $\sin^2 2\theta_{ee}$ with respect to $\sin^2(\theta_{14})$.**
- **Derive the probability of $\nu_\mu \to \nu_e$ appearance for 3+1 model, and consequently find a relation for $\sin^2 2\theta_{\mu e}$ with respect to $\sin^2(\theta_{24})$ and $\sin^2(2\theta_{14})$.**

Hint: Use trig identities (specifically double angle identities) to help you.

15.1 LSND and MiniBooNE

The MicroBooNE experiment searches for possible neutrino oscillations involving a fourth sterile state [5]. This has been motivated by previous experiments such as MiniBooNE and LSND which found anomalies in their results that can indicate a fourth neutrino favour. The "allowed" parameter space that these experiments determined for this fourth sterile state can be seen plotted at the end of the notebook.

These experiments were looking for ν_e appearance, while the signal in our data is exclusively ν_μ, and we have been considering said signal's disappearance [3]. The MiniBooNE and LSND data are stored as 'MiniBooNEdata.csv' and 'LSNDdata.csv', respectively.

Exercise 14:

- **Using the value for $\sin^2 2\theta_{14}$ = $\sin^2 2\theta_{ee}$ = 0.24 result from MicroBooNE [5], compare your result with LSND and MiniBooNE. This requires that you convert the muon disappearance parameter space into the electron appearance parameter space.***
- **If your cuts and analysis are well-tuned, you should see that your confidence levels overlap or exclude the results from MiniBooNE and LSND. What does this tell you about the previous MiniBooNE and LSND anomalies?**

**Hint:* In practice, this means replacing your previous oscillation parameter of $sin^2 2\theta$ with $\sin^2 2\theta_{\mu e}$. To do this, derive $\sin^2 2\theta_{\mu e} = 4|U_{\mu 4}|^2|U_{e4}|^2$ with respect to parameters we know:

- $\sin^2 2\theta_{\mu\mu}$ (which can be considered as your original oscillation parameter range)
- $\sin^2 2\theta_{ee} = 0.24$

References

[1] The nobel prize in physics. https://www.nobelprize.org/prizes/physics/2015/press-release/, 2015.

[2] R. Acciarri et al. Design and construction of the microboone detector. *JINST*, 12, 2016.

[3] Matthew Adams and othrs. Direct comparison of sterile neutrino constraints from cosmological data, ν_e disappearance data and $\nu_\mu \rightarrow \nu_e$ appearance data in a 3+1 model. *The European Physical Journal C*, 80, aug 2020.

[4] John N. Bahcall and Raymond Davis. Solar neutrinos: A scientific puzzle. *Science*, 191(4224):264–267, 1976.

[5] Mary Bishai et al. Search for a sterile neutrino in a 3+1 framework using the wire-cell inclusive charged-current ν_e selection from microboone. https://microboone.fnal.gov/wp-content/uploads/MICROBOONE-NOTE-1116-PUB.pdf, 2022.

[6] Dr Steve Boyd. Neutrino oscillations. https://warwick.ac.uk/fac/sci/physics/staff/academic/boyd/stuff/neutrinolectures/lec_oscillations.pdf, 2020.

[7] C. L. Cowan et al. Detection of the free neutrino: a confirmation. *Science*, 124:103–104, 1956.

[8] P. Lipari. Introduction to neutrino physics. In *1st CERN-CLAF School of High-Energy Physics*, pages 115–199, 5 2001.

[9] B. Pontecorvo. Neutrino Experiments and the Problem of Conservation of Leptonic Charge. *Zh. Eksp. Teor. Fiz.*, 53:1717–1725, 1967.

[10] Fred L. Wilson. Fermi's theory of beta decay. *American Journal of Physics*, 36:1150–1160, 1968.

[11] P A Zyla et al. Review of Particle Physics. *Progress of Theoretical and Experimental Physics*, 2020. 083C01.