CALORICH AI

An UBC MDS Capstone Project with TRIUMF, 2023

June 15, 2023

GENG, Crystal
MERIGO, Daniel
WONG, Kelvin
ZHANG, Peng

THE UNIVERSITY OF BRITISH COLUMBIA

Agenda

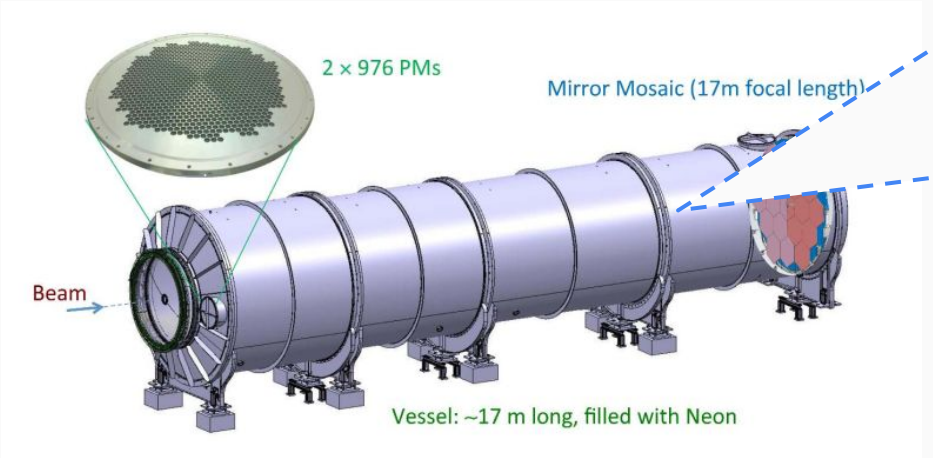
1. Background and Scope
2. Data and Pipeline
3. Feature Engineering
4. XGBRegressor Model
5. Multi-layer Perceptron
6. PointNet
7. Data Product
8. Q&A

1. Background and Scope

What's TRIUMF?

Image Credit: https://cerncourier.com/a/tales-of-triumf/

NA62 Experiment



Decay

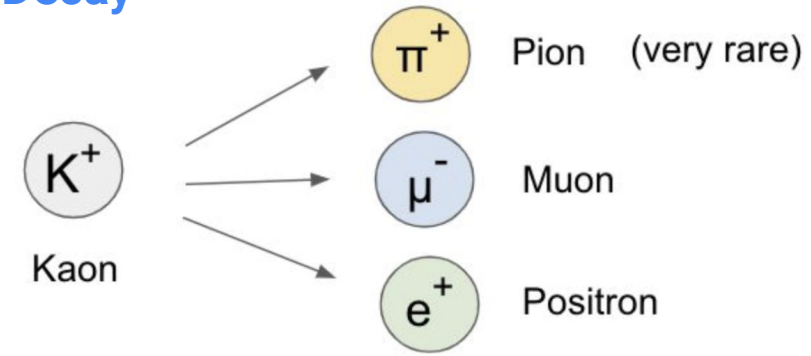


Image Credit: doi:10.1088/1748-0221/12/05/P05025

Goal: to differentiate between **pion** and **muon**

NA62 Experiment

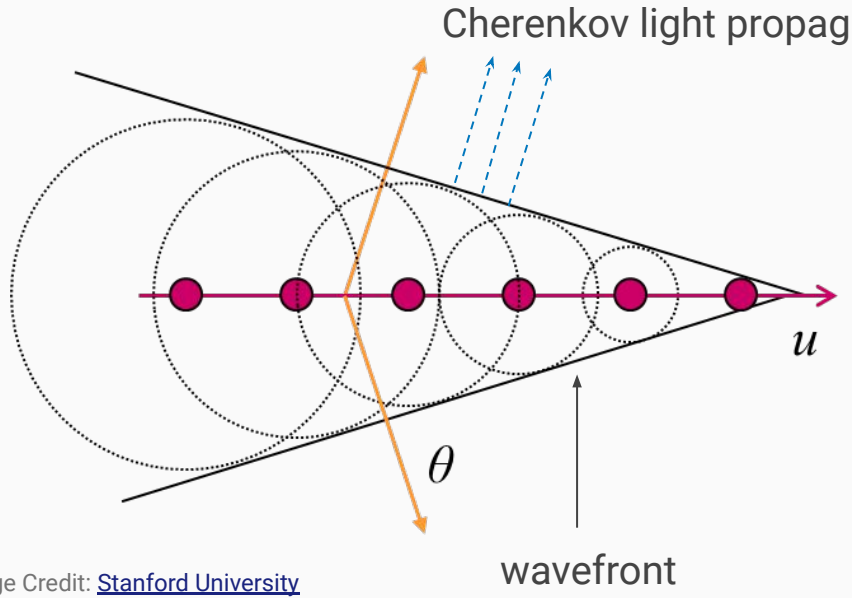
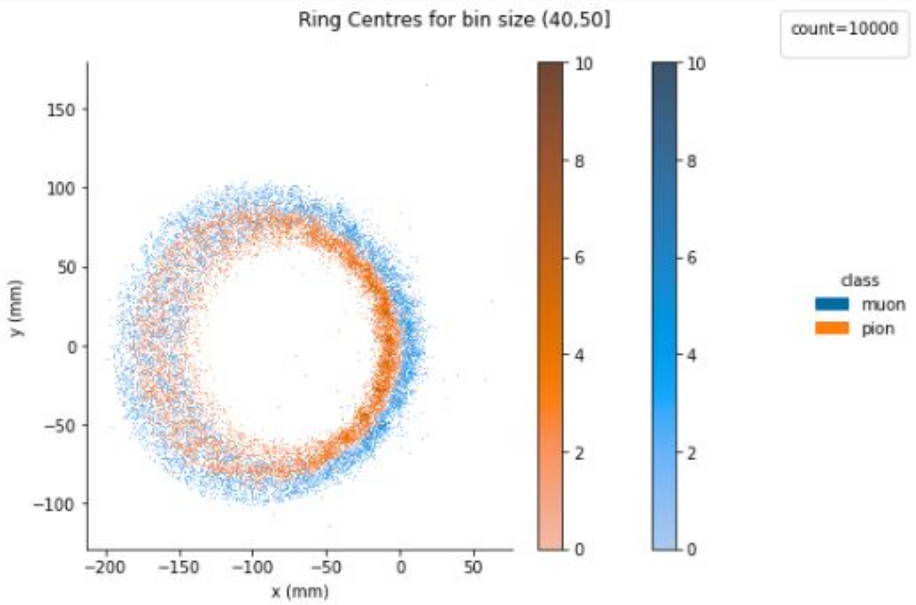


Image Credit: Stanford University

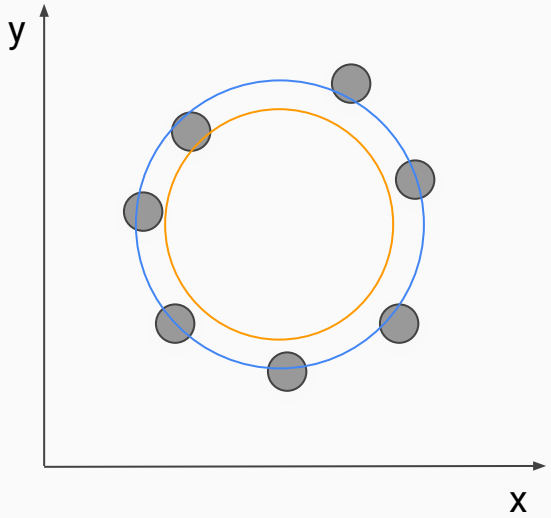
- Charged particles passes through a dielectric medium
- The particle travels at a speed higher than phase velocity of the medium
- This process emits photons that form a spherical wavefront
- This “ring image” can be detected by the RICH detector

Pion and Muon Differentiation



- Photon hits are captured by the detectors
- Different particles form rings with different sizes
- **Muon** has a larger ring radius than **pion** at a given momentum

State-of-the-art Method at TRIUMF



MLE is used in the SOTA algorithm

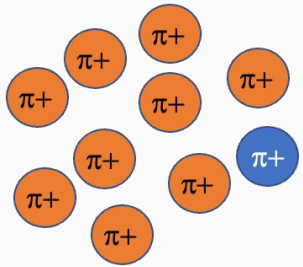
Likelihood of **muon** = 0.85

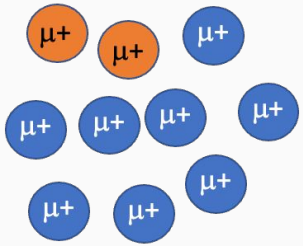
Likelihood of **pion** = 0.15

This particle is identified as a **muon**

Our Objectives

- **Last year's Capstone Project**:
  - Use **classification** methods to differentiate between pion and muon





$$\text{Pion Efficiency} = \frac{TP}{TP + FN}$$

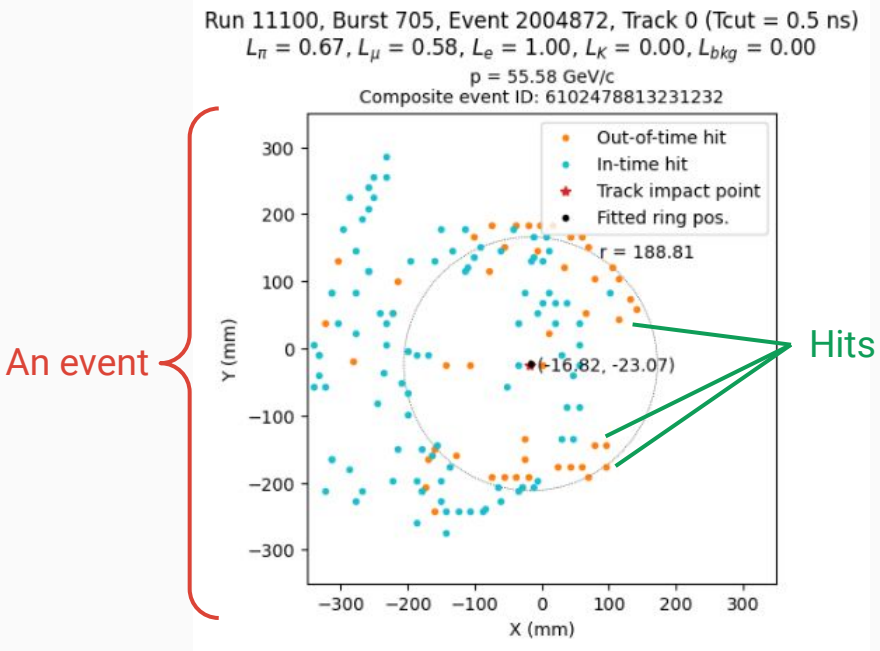
$$\text{Muon Efficiency} = \frac{FP}{FP + TN}$$

- **Our goal this year**:
  - Use **regression** methods to **predict** the fitted ring radius
  - Using the predicted radii to compute the decision boundary, with which **pion** and **muon** can be differentiated
  - **Approaches:**
    - XGBRegressor
    - Multi-layer Perceptron
    - PointNet

2. Data and Pipeline

Dataset

- An **event** refers to a decay (“ring circle”), and the **hit** refers to the sensor excited.
- We have 2 datasets, each with identical structure of **events** and **hits**.
- The primary dataset that we work with has **~2.4M** events and **~99M** hits.
- A supplementary dataset has **~43M** events and **~1.6B** hits.

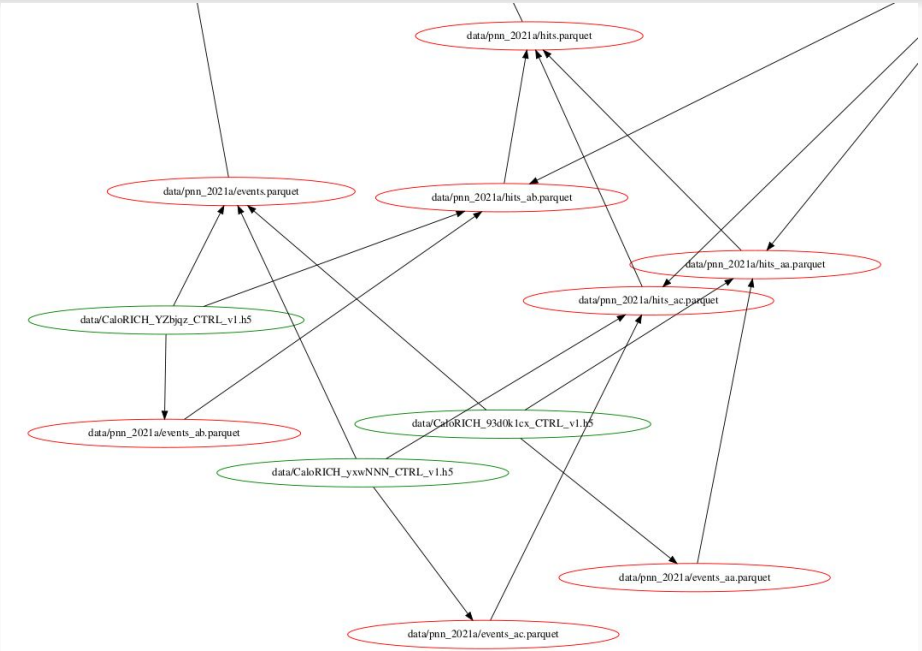


Time: Our Worst Enemy

- **Time!**
  - Specifically **loading time** and **training time** for the models we tried
- Last year's approach:
  - Implement feature engineering **within each of the model**
    - Code repetition
    - Wasted computing resources over the same features across models
    - Slower model iteration because of the back-and-forth time

Our Approach

- We use scripts to extract the data and do feature engineering **outside of the models**
- We use **Polars** instead of Pandas
- We generate a series of intermediate Parquet files for different hyperparameters and use **Makefile** to manage the generation



Our Approach

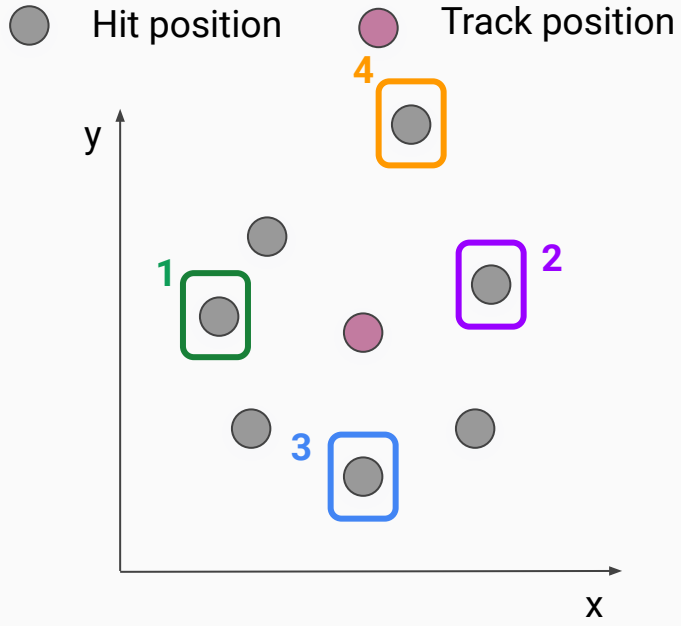
- Originally we run the `make` command locally
- We then use Kaggle to generate and host the intermediate files
  - 12 hour limit
- Now we use a dedicated private server to generate the files and upload to a cloud storage for our team to download

Calorich Data Output (Version: 2023-06-09)

```
[4.0K 10-Jun-2023 00:18]  .
├── [4.0K 09-Jun-2023 21:36]  pnn_2021a
│   ├── [565M 03-Jun-2023 04:06]  events_aa.parquet
│   ├── [1.4G 06-Jun-2023 02:18]  events_aa_with_hit_features_[cut_off_time=0.1].parquet
│   ├── [1.4G 06-Jun-2023 04:24]  events_aa_with_hit_features_[cut_off_time=0.2].parquet
│   ├── [1.4G 06-Jun-2023 06:26]  events_aa_with_hit_features_[cut_off_time=0.3].parquet
│   ├── [1.4G 06-Jun-2023 08:30]  events_aa_with_hit_features_[cut_off_time=0.4].parquet
│   ├── [1.4G 06-Jun-2023 10:40]  events_aa_with_hit_features_[cut_off_time=0.5].parquet
│   ├── [1.4G 06-Jun-2023 12:44]  events_aa_with_hit_features_[cut_off_time=0.6].parquet
│   ├── [1.4G 06-Jun-2023 14:49]  events_aa_with_hit_features_[cut_off_time=0.7].parquet
│   ├── [1.4G 06-Jun-2023 16:56]  events_aa_with_hit_features_[cut_off_time=0.8].parquet
│   ├── [1.4G 06-Jun-2023 19:00]  events_aa_with_hit_features_[cut_off_time=0.9].parquet
│   ├── [1.4G 06-Jun-2023 21:06]  events_aa_with_hit_features_[cut_off_time=1.0].parquet
│   ├── [651M 03-Jun-2023 04:10]  events_ab.parquet
│   ├── [1.6G 06-Jun-2023 23:26]  events_ab_with_hit_features_[cut_off_time=0.1].parquet
│   ├── [1.6G 07-Jun-2023 01:45]  events_ab_with_hit_features_[cut_off_time=0.2].parquet
│   ├── [1.6G 07-Jun-2023 04:07]  events_ab_with_hit_features_[cut_off_time=0.3].parquet
│   ├── [1.6G 07-Jun-2023 06:28]  events_ab_with_hit_features_[cut_off_time=0.4].parquet
│   ├── [1.6G 07-Jun-2023 08:50]  events_ab_with_hit_features_[cut_off_time=0.5].parquet
│   ├── [1.6G 07-Jun-2023 11:13]  events_ab_with_hit_features_[cut_off_time=0.6].parquet
│   ├── [1.6G 07-Jun-2023 13:37]  events_ab_with_hit_features_[cut_off_time=0.7].parquet
│   ├── [1.6G 07-Jun-2023 15:58]  events_ab_with_hit_features_[cut_off_time=0.8].parquet
│   ├── [1.6G 07-Jun-2023 18:20]  events_ab_with_hit_features_[cut_off_time=0.9].parquet
│   ├── [1.6G 07-Jun-2023 20:44]  events_ab_with_hit_features_[cut_off_time=1.0].parquet
│   ├── [643M 03-Jun-2023 04:13]  events_ac.parquet
│   ├── [1.6G 07-Jun-2023 23:10]  events_ac_with_hit_features_[cut_off_time=0.1].parquet
│   ├── [1.6G 08-Jun-2023 01:29]  events_ac_with_hit_features_[cut_off_time=0.2].parquet
│   ├── [1.6G 08-Jun-2023 03:48]  events_ac_with_hit_features_[cut_off_time=0.3].parquet
│   ├── [1.6G 08-Jun-2023 06:07]  events_ac_with_hit_features_[cut_off_time=0.4].parquet
│   ├── [1.6G 08-Jun-2023 08:28]  events_ac_with_hit_features_[cut_off_time=0.5].parquet
│   ├── [1.6G 08-Jun-2023 10:48]  events_ac_with_hit_features_[cut_off_time=0.6].parquet
│   ├── [1.6G 08-Jun-2023 13:10]  events_ac_with_hit_features_[cut_off_time=0.7].parquet
│   ├── [1.6G 08-Jun-2023 15:31]  events_ac_with_hit_features_[cut_off_time=0.8].parquet
│   ├── [1.6G 08-Jun-2023 17:52]  events_ac_with_hit_features_[cut_off_time=0.9].parquet
│   ├── [1.6G 08-Jun-2023 20:12]  events_ac_with_hit_features_[cut_off_time=1.0].parquet
│   ├── [1.9G 03-Jun-2023 04:15]  events.parquet
│   ├── [4.6G 09-Jun-2023 21:26]  events_with_hit_features_[cut_off_time=0.1].parquet
│   ├── [4.6G 09-Jun-2023 21:27]  events_with_hit_features_[cut_off_time=0.2].parquet
│   ├── [4.6G 09-Jun-2023 21:28]  events_with_hit_features_[cut_off_time=0.3].parquet
│   ├── [4.6G 09-Jun-2023 21:29]  events_with_hit_features_[cut_off_time=0.4].parquet
│   ├── [4.6G 09-Jun-2023 21:30]  events_with_hit_features_[cut_off_time=0.5].parquet
│   ├── [4.6G 09-Jun-2023 21:30]  events_with_hit_features_[cut_off_time=0.6].parquet
│   ├── [4.6G 09-Jun-2023 21:31]  events_with_hit_features_[cut_off_time=0.7].parquet
│   ├── [4.6G 09-Jun-2023 21:32]  events_with_hit_features_[cut_off_time=0.8].parquet
│   ├── [4.6G 09-Jun-2023 21:33]  events_with_hit_features_[cut_off_time=0.9].parquet
│   ├── [4.6G 09-Jun-2023 21:34]  events_with_hit_features_[cut_off_time=1.0].parquet
│   ├── [ 14G 03-Jun-2023 04:23]  hits_aa.parquet
│   ├── [ 16G 03-Jun-2023 04:31]  hits_ab.parquet
│   ├── [ 16G 03-Jun-2023 04:40]  hits_ac.parquet
│   └── [ 53G 03-Jun-2023 06:20]  hits.parquet
├── [103M 03-Jun-2023 00:14]  events.parquet
├── [263M 03-Jun-2023 00:37]  events_with_hit_features_[cut_off_time=0.1].parquet
├── [263M 03-Jun-2023 00:59]  events_with_hit_features_[cut_off_time=0.2].parquet
```

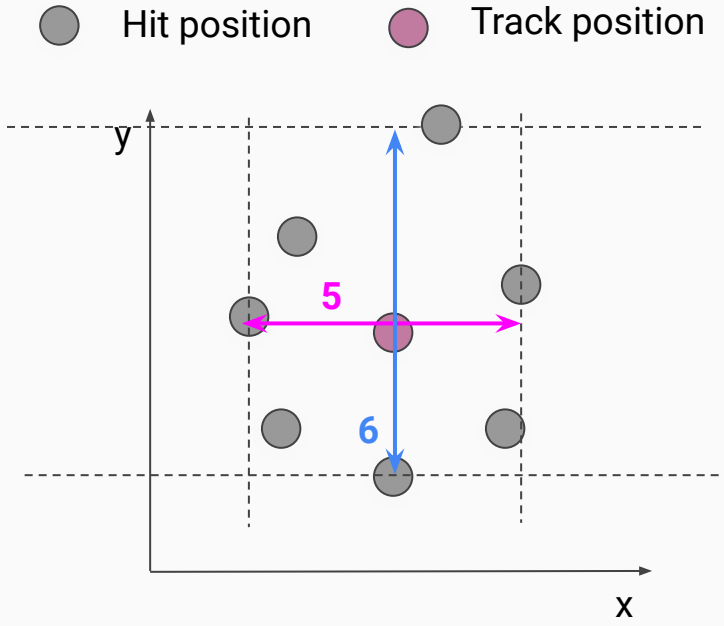
3. Feature Engineering

Engineered Features - Hit Positions



1. Minimum hit position x
2. Maximum hit position x
3. Minimum hit position y
4. Maximum hit position y

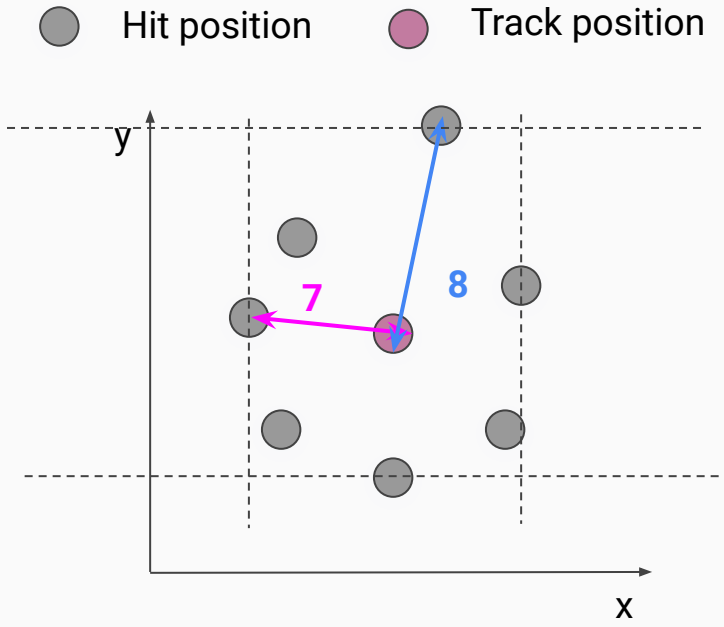
Engineered Features - Width of x and y Positions



5. Max minus min hit positions x

6. Max minus min hit positions y

Engineered Features: Hit Distances



7. Min hit position - track position

8. Max hit position - track position

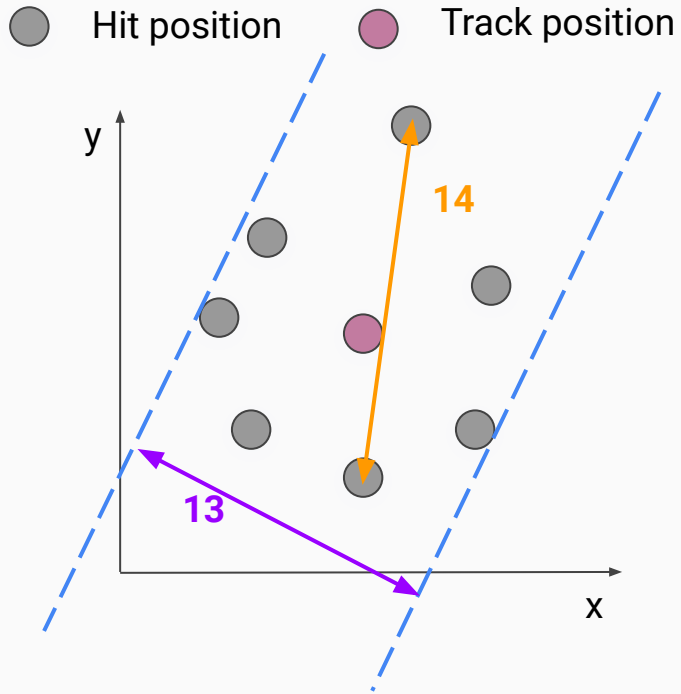
9. Mean hit position - track position

10. Median hit position - track position

11. 25% and 75% quantiles of hit position - track position

12. Root mean square hit position - track position

Engineered Features : Hull Width and Diameter

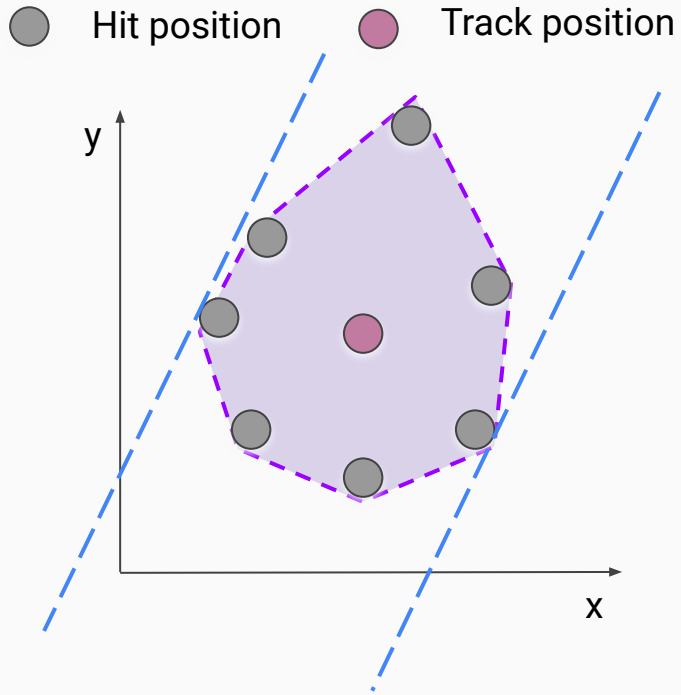


13. Hull width: shortest distance of two parallel lines encapsulating all points

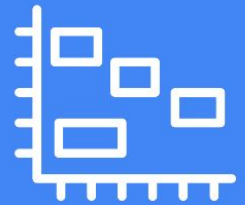
14. Hull diameter: longest distance among all the points

15. Hull diameter and width difference: the difference between hull diameter and width

Engineered Features : Hull Area

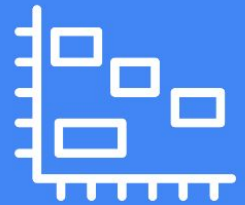


16: Hull area: area of the convex hull, i.e., the polygon that encapsulates all the points



4. XGBRegressor Model and Analyses

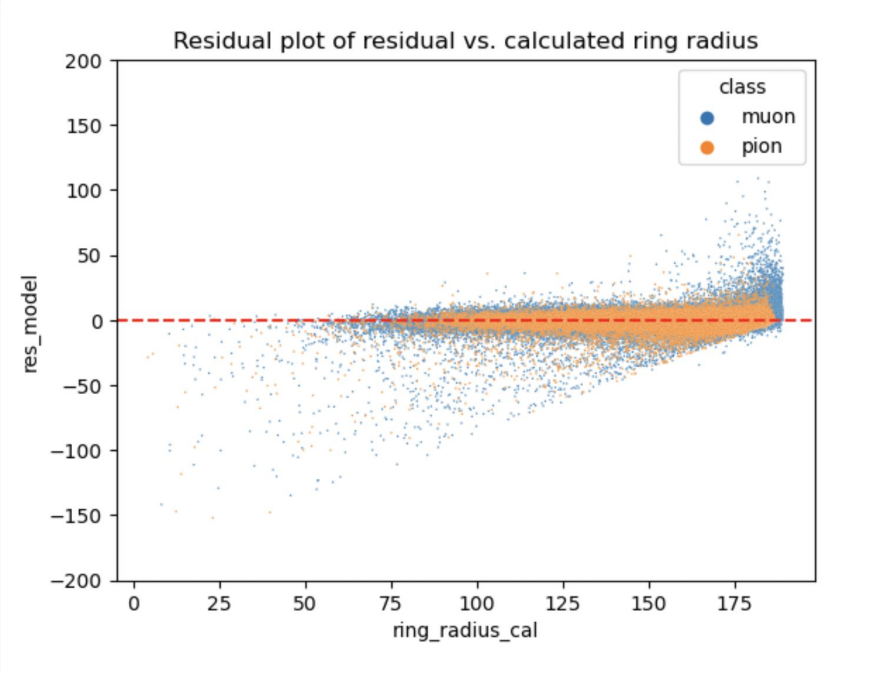
XGBRegressor



- Model of choice: **XGBRegressor**
- **Features used** (18 in total):
    - Total number of in-time hits
    - Aforementioned engineered features
- **Target (y):**
    - Theoretical ring radius calculated based on momentum and mass
    -

$$r = F_M \cdot N \cdot p \cdot \sqrt{1 - \frac{m^2 + p^2}{N^2 p^2}} \cdot \frac{1}{\sqrt{m^2 + p^2}}$$

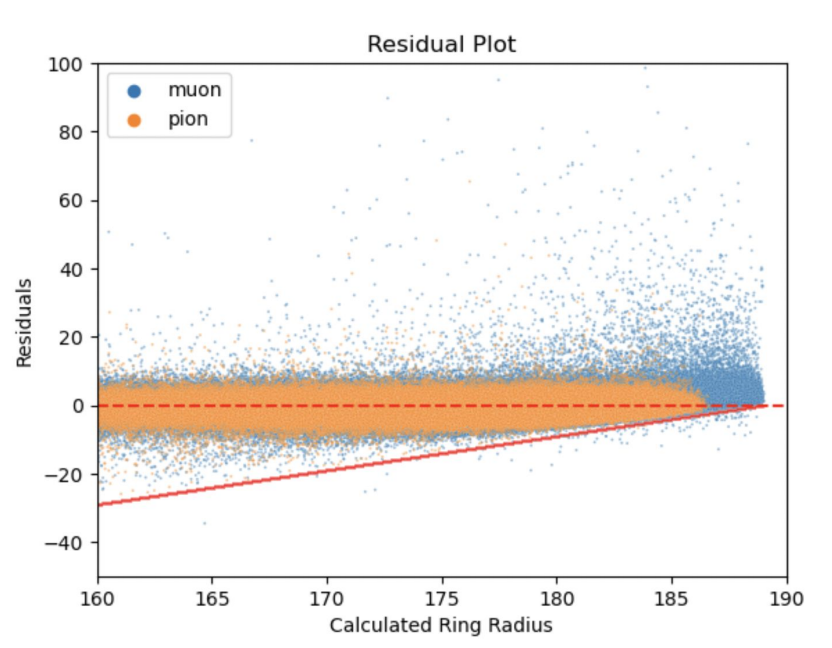
XGBRegressor Performance



- **Performance:**
  - Training R^2: 0.938
  - Test R^2: 0.939
- **Residual plot:**
  - Observed a cut-off bias at **higher end** of theoretical ring radius
  - More points **above 0** for residual
  - The model seems to **underestimate** the ring radius for higher values

XGBRegressor Residual Analysis

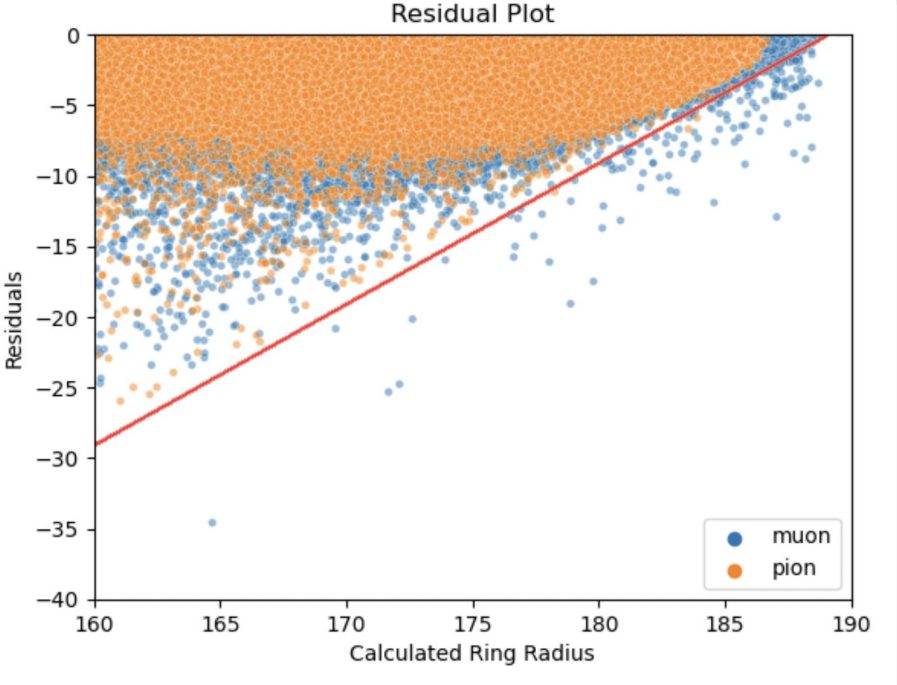
* residual is calculated by theoretical - predicted



- The difference between **calculated ring radius** of each example and the **maximum** value is calculated, shown as the red line
- Points above this line indicate that the predicted radius is **smaller than** the max calculated ring radius

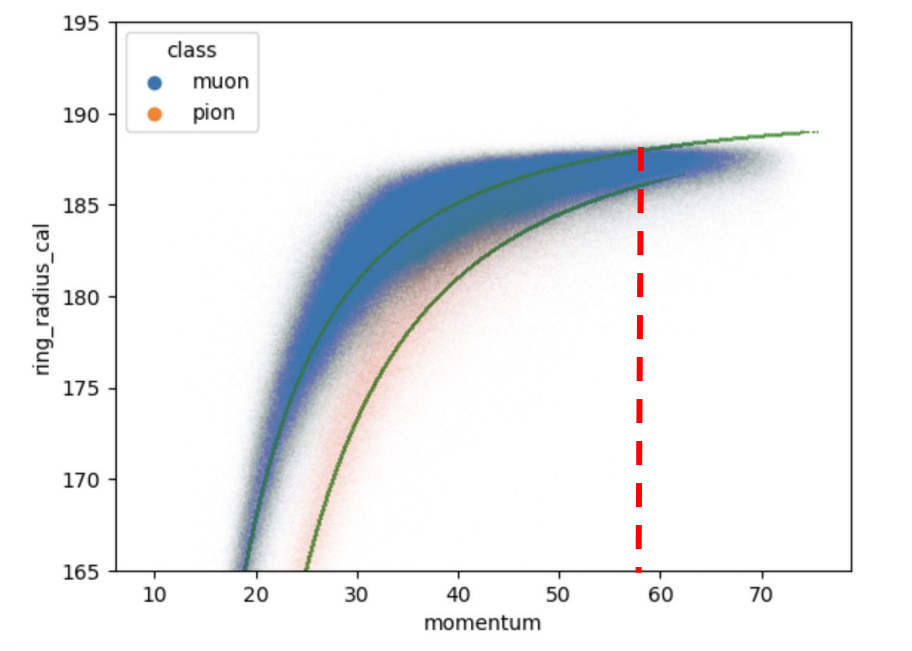
XGBRegressor Residual Analysis

* residual is calculated by theoretical - predicted



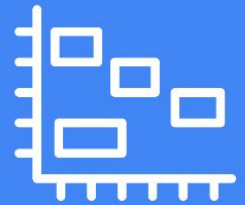
- Total number of points **below** the line:
  - **Muon**: 172
  - **Pion**: 5
- Total number of points **above** the line:
  - **Muon:** 2158463
  - **Pion**: 215642

Ring Radius vs. Track Momentum

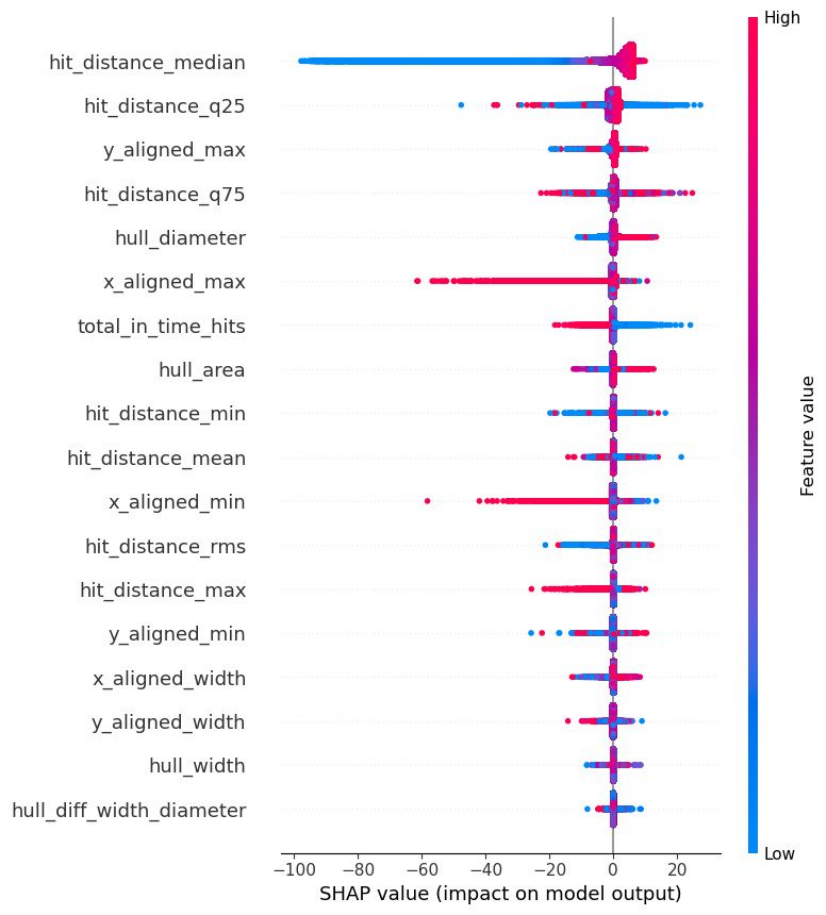


- **Green lines**: theoretical ring radius vs. track momentum
- **Blue points**: predicted ring radius vs. track momentum for **muons**
- **Orange points**: predicted ring radius vs. track momentum for **pions**
- Can see that the model **underestimates** radius for **muons** for momentum > 57 GeV/c

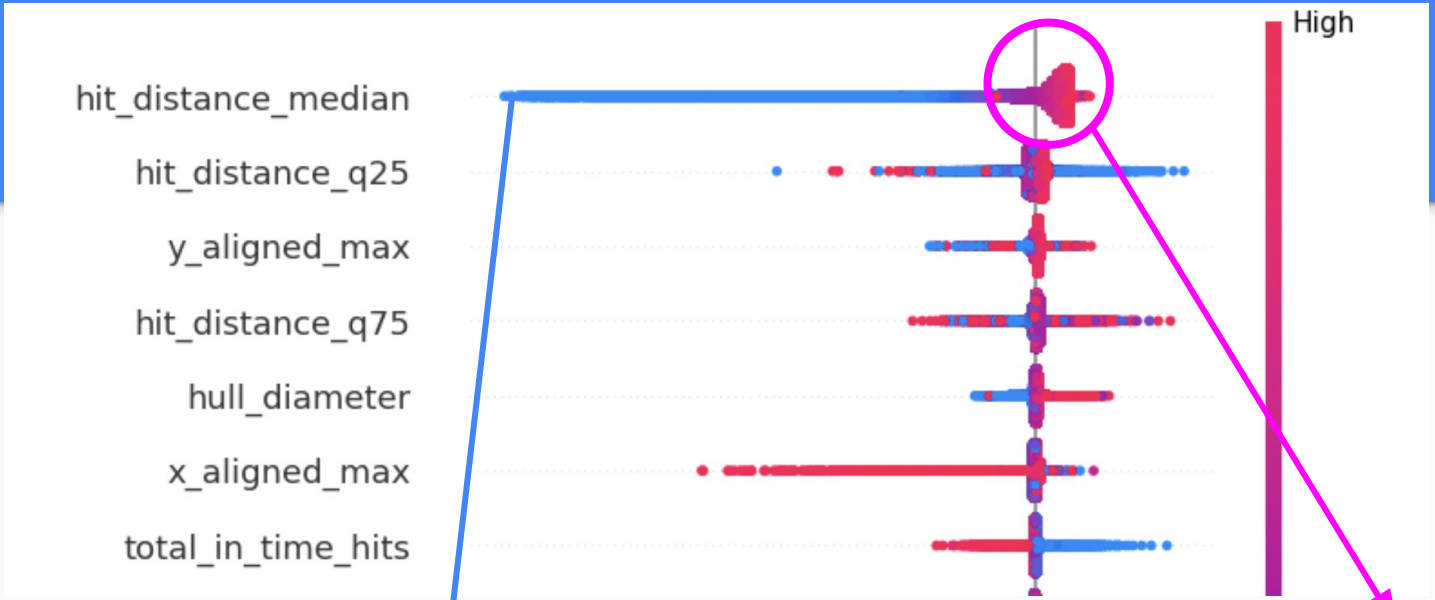
XGBRegressor SHAP Analysis



- SHAP (SHapley Additive exPlanations) analysis is used to explain the predictions of machine learning models
- It provides a way to understand the **contribution of each feature** to the model's output.
- SHAP values represent the **marginal contribution** of a feature to the **expected prediction** compared to a baseline prediction.



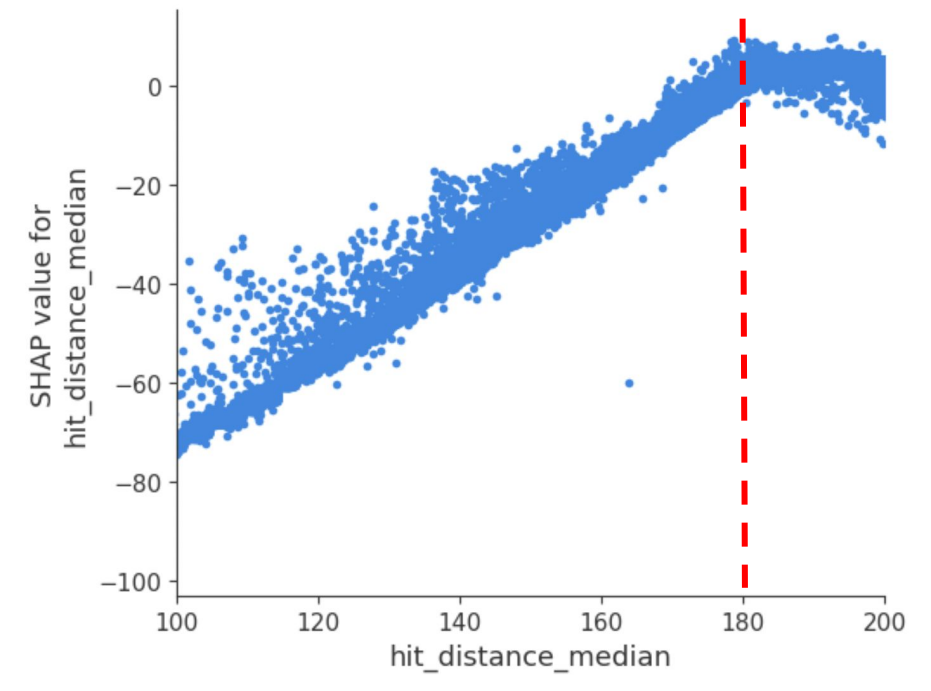
- Features are ranked based on their importance, from top to bottom
- The bar on the right-hand-side indicates feature values (e.g., value of hit_distance_median)
- The x-axis indicates the SHAP value:
  - A negative SHAP indicates the feature contributes negatively to the prediction
  - A positive SHAP indicates the feature contributes positively to the prediction



Low feature values are associated with negative SHAP values, and the bar extends very far to the left.

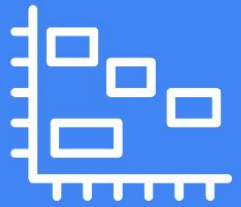
High feature values are associated with positive SHAP values, but the bar does not extend as far as the left side.
The values rather “pile up”, which corresponds to the “leveling effect” we see in residual plot

XGBRegressor SHAP Analysis

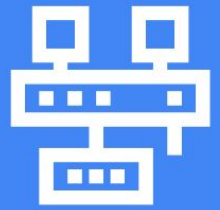


- Scatter plot showing SHAP value vs. feature value for hit_distance_median
- SHAP value starts to plateau at **~ 180 mm**
- This is consistent with the cut-off we observe in the residual plot
- It shows that hit_distance_median is **underestimating** the ring radius for **radius > 180 mm**

XGPRegressor Model Conclusion

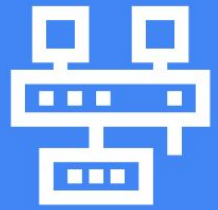


- The ring radius is **underestimated** for larger values
- The model starts to underestimate the prediction for radius **over 180 mm**
- This effect seems to be attributed to the **upper limit of max calculated radius** (~ 189 mm)
- Hyperparameter optimization/feature selection/regularization did not alleviate this problem
- We are interested in seeing if this is model-related or data-related



5. Multi-layer Perceptron

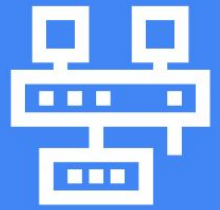
Multi-layer Perceptron (MLP) Regression



- **Input**: same as XGB model
- **Target**: calculated ring radius
- StandardScaler applied
- Loss function: nn.MSELoss
- Optimizer: optim.Adam
- Epochs = 30

```
# Define the model
model = torch.nn.Sequential(
    nn.Linear(18, 1024),
    nn.ReLU(),
    nn.Linear(1024, 512),
    nn.ReLU(),
    nn.Linear(512, 256),
    nn.ReLU(),
    nn.Linear(256, 64),
    nn.ReLU(),
    nn.Linear(64, 12),
    nn.ReLU(),
    nn.Linear(12, 1)
)
```

Multi-layer Perceptron (MLP) Regression

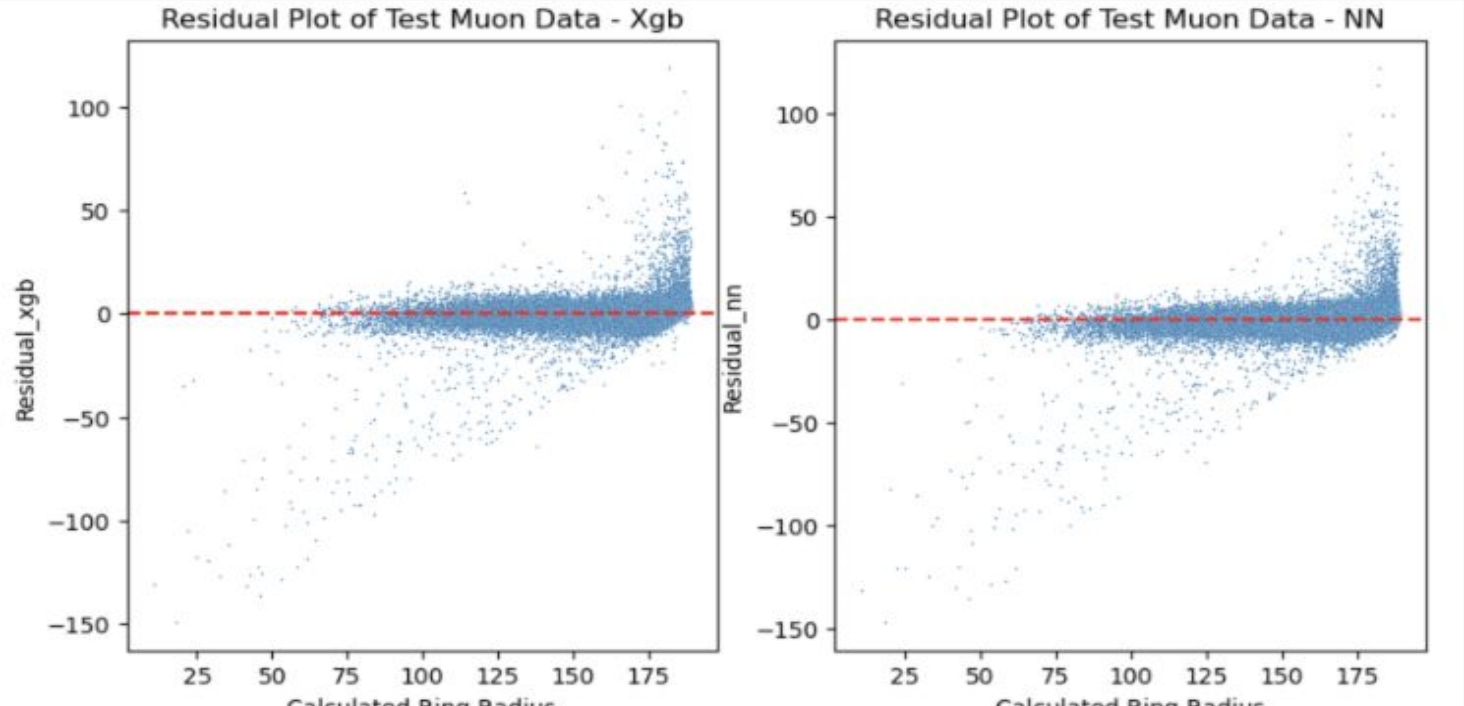


Model	r2_train	MAE_train	r2_val	MAE_val	r2_test	MAE_test
XGBRegressor	0.962	1.542	0.945	1.590	0.943	1.591
Regression Network	/	/	/	/	**0.939**	**1.740**

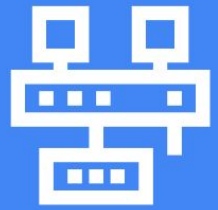
Model	r2_pion	MAE_pion
XGBRegressor	0.905	2.374
Regression Network	**0.912**	**2.365**

- MLP slightly less accurate than XGB
- Flexible extension based on MLP, such as Quantile regression

Multi-layer Perceptron (MLP) Regression



MLP Quantile Regression

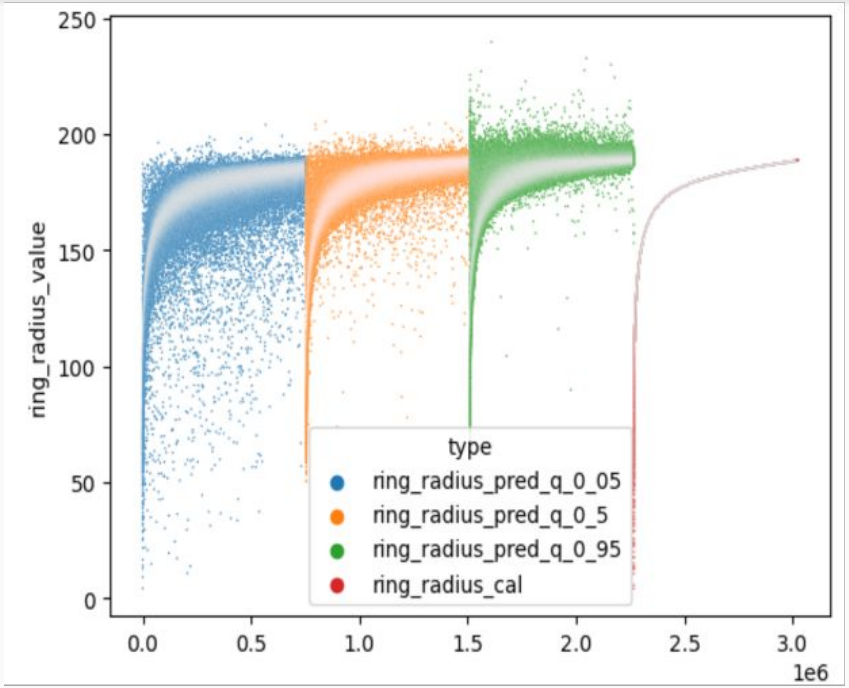


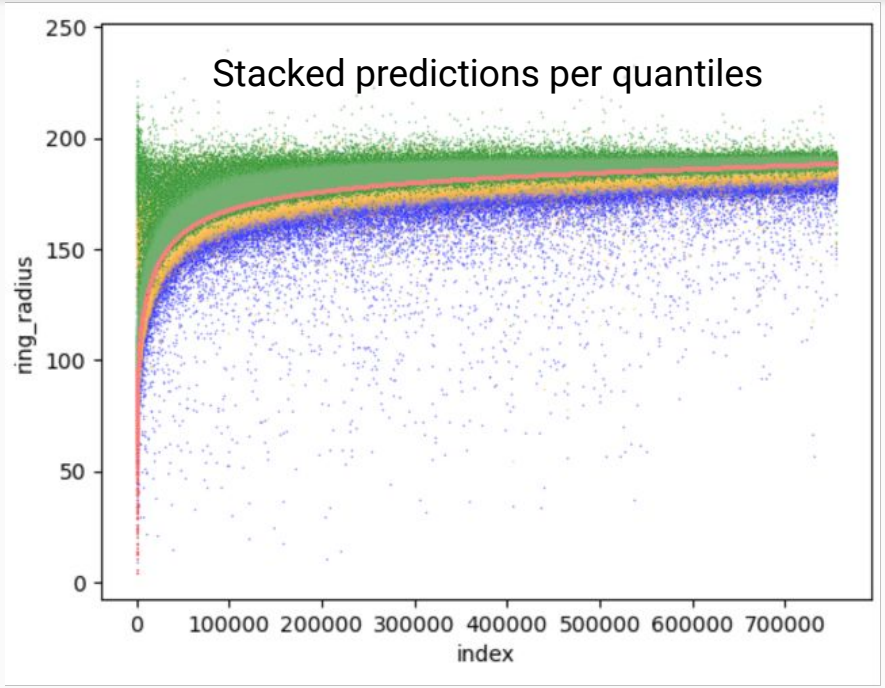
```python
def quantile_loss(preds, target, quantile):
    assert not target.requires_grad
    assert preds.size(0) == target.size(0)
    losses = []

    errors = target - preds
    losses.append(torch.max((quantile - 1) * errors, quantile * errors).unsqueeze(1))
    loss = torch.mean(torch.sum(torch.cat(losses, dim=1), dim=1))
    return loss
```

```python
def trainer(model, optimizer, trainloader, validloader, epochs=5, patience=5, q=0.5, verbose=True):
    """Training wrapper for PyTorch network."""

    train_loss = []
    valid_loss = []
```

Quantile parameter

MLP Quantile Regression





6. PointNet

What is PointNet

- Network Architecture developed in Stanford
- Handles **3-Dimensional point clouds**
- Can identify shapes based on the point cloud
- Can perform **Semantic Segmentation**
- Used in computer vision (LiDAR)

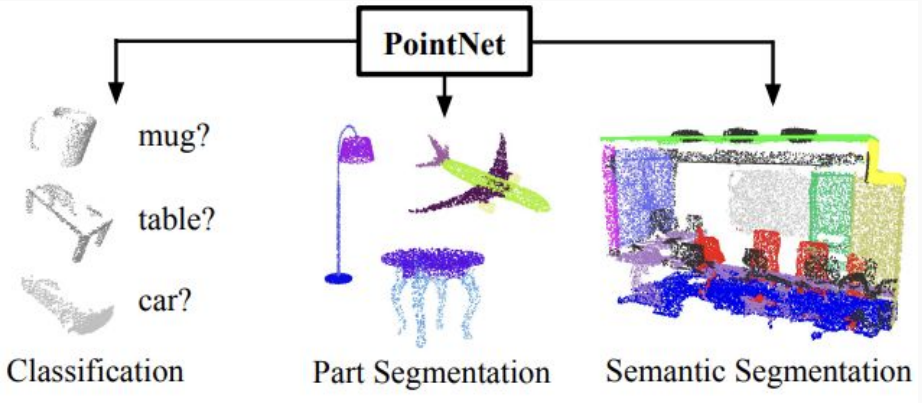
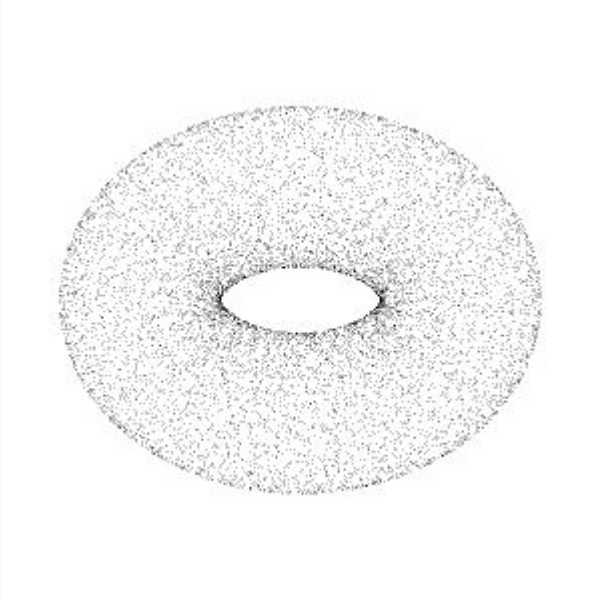


Image Credit: Stanford

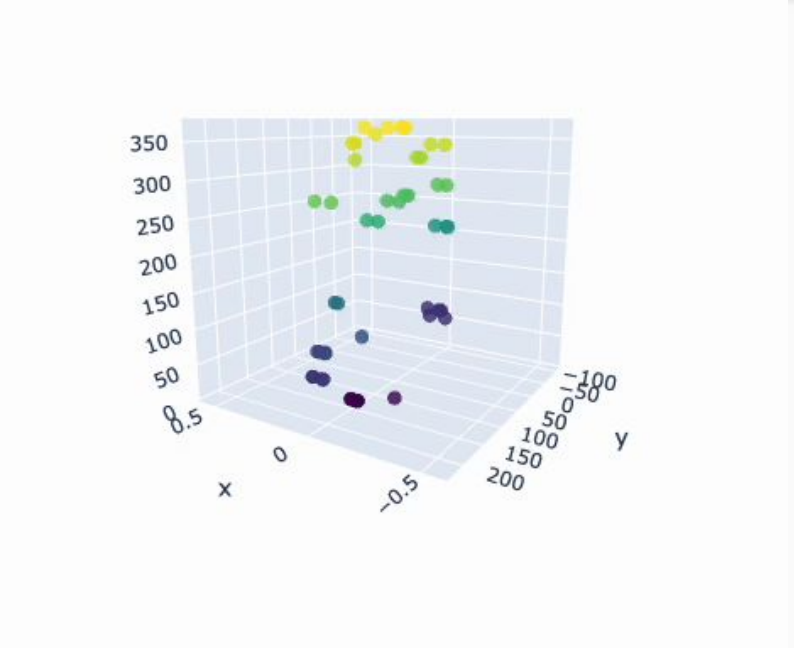
Why PointNet

- The **order of the points do not matter**
- The points are related to each other
- Invariant to transformations
- Can be modified to perform regression analysis
  - Adding a **fully connected output layer** with a single output



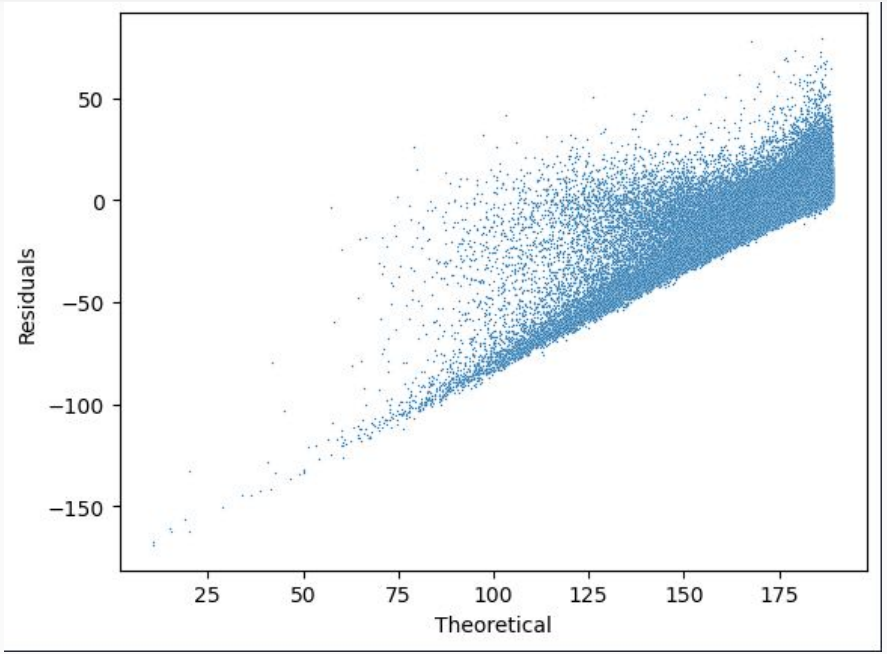
Experiment

- Used 2 different variations of PointNet architecture
- Trained on **2M** point clouds with **50** points each
- Added the **3rd Dimension**



Comparison with Simpler Models

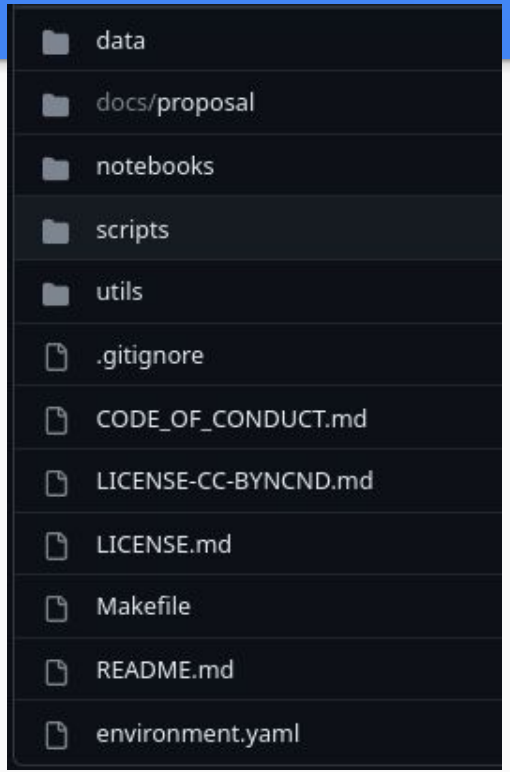
Model	r2_pion	MAE_pion
XGBRegressor	0.905	2.374
Regression Network	**0.912**	**2.365**
PointNet	**0.199**	**6.312**



7. Data Product

Data Product

- Key concern in academia: **reproducibility**
  - Conda environment with Makefile
  - (TBC) Docker/Singularity environment
- Full code to wrangle data from source file
- Models with trained weights
  - XGBRegressor
  - MLP neural network
  - PointNet



Thank you!

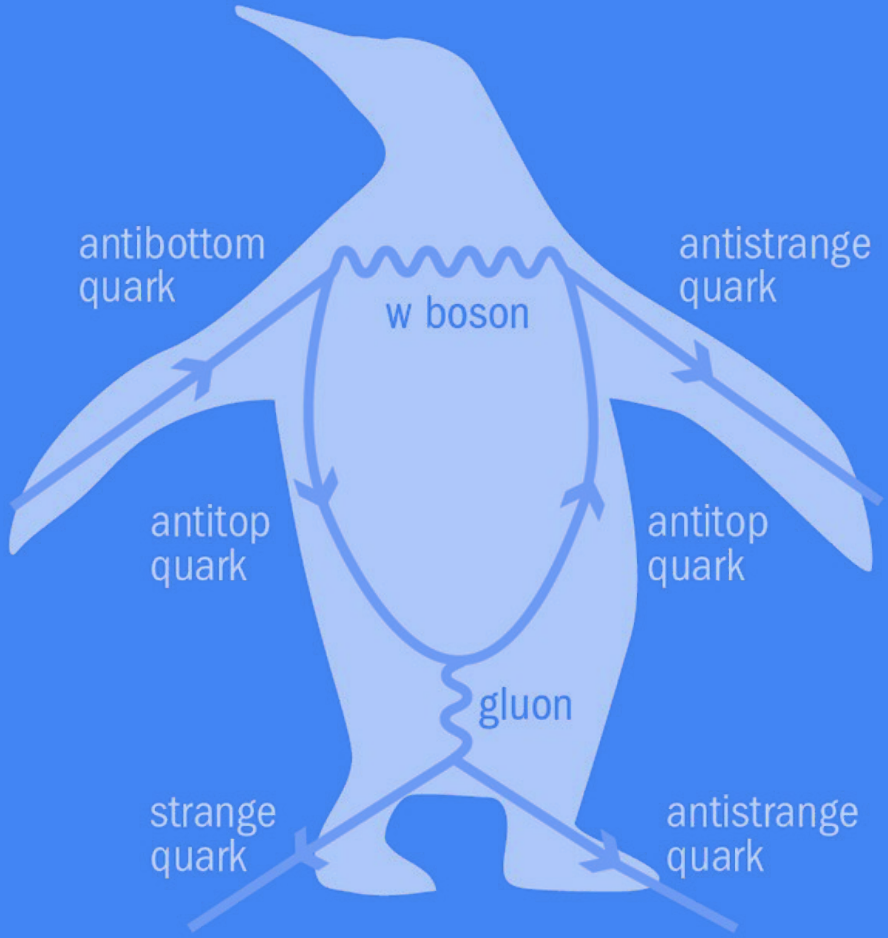
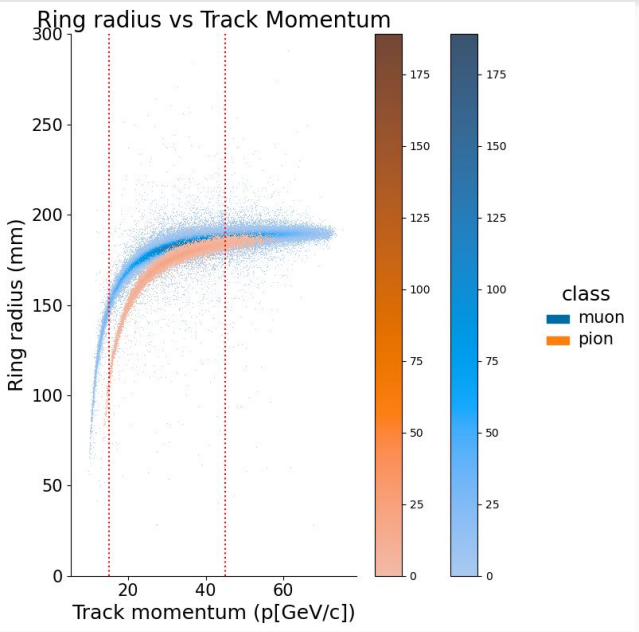


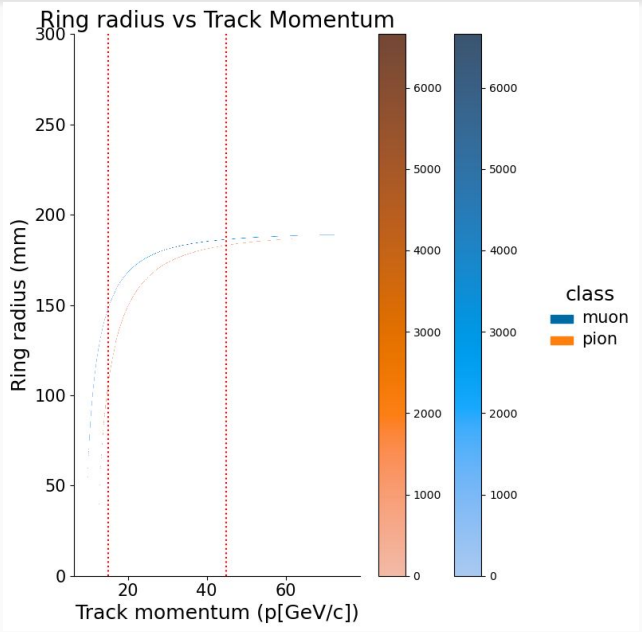
Image Credit: CERN

Appendix

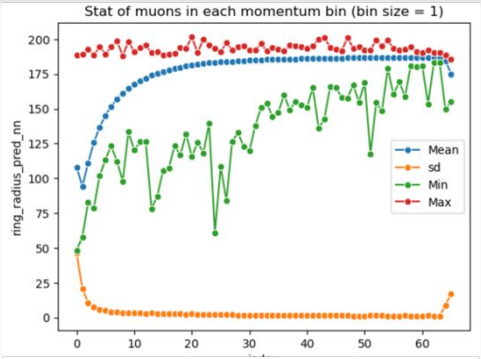
MLP Regression Resolution Analysis

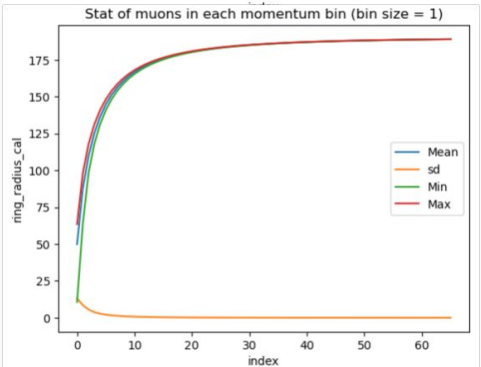


Fitted ring radius

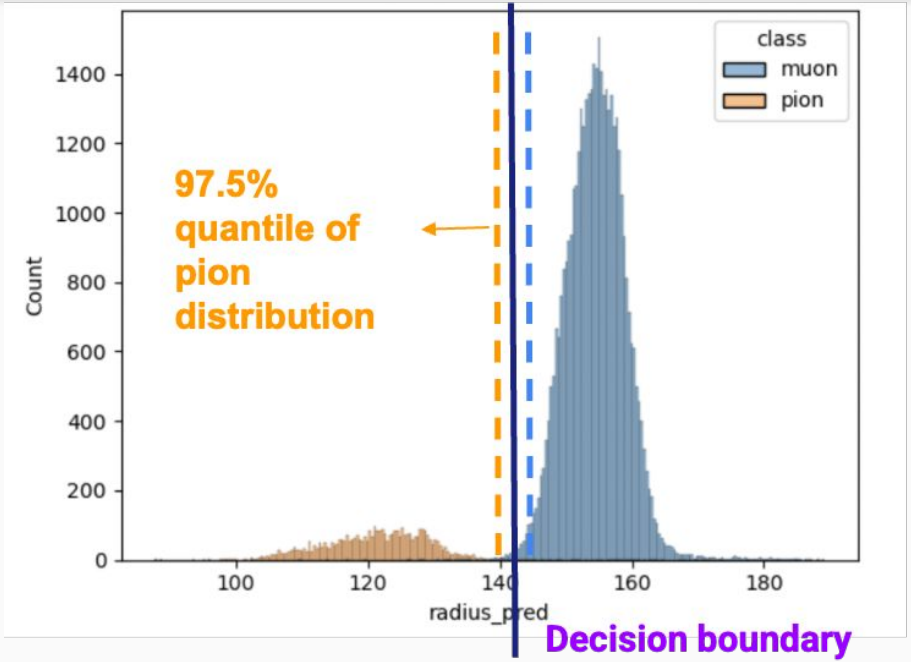


Calculated ring radius





MLP Regression Resolution Analysis

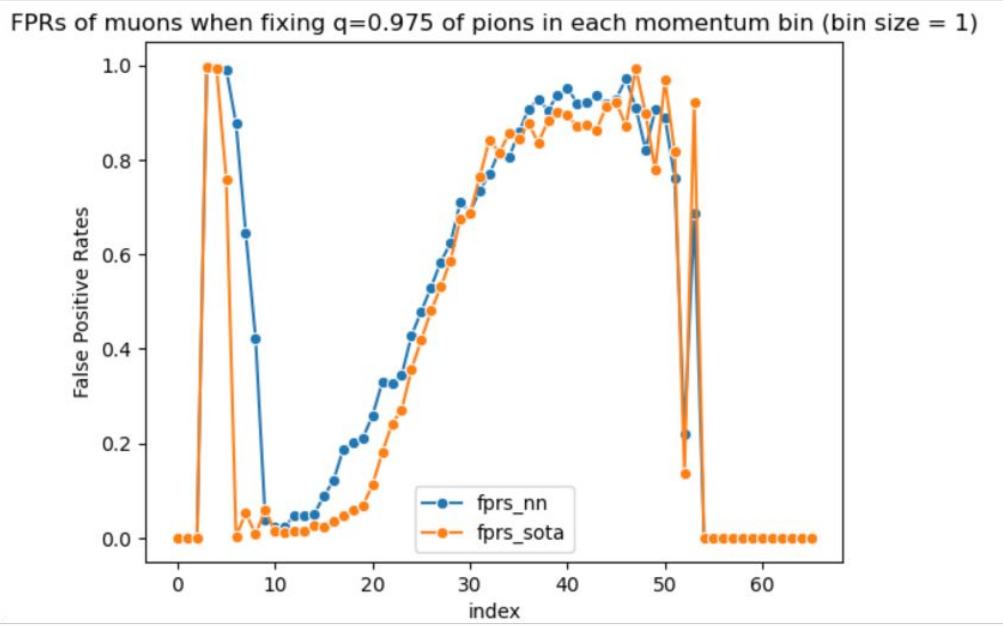


For a specific momentum bin

- **Positive** class: pion
- Define a **decision boundary**

  ie. 97.5% quantile of pion (pion efficiency/**True Positive Rate**)
- Calculate muon efficiency / **False Positive Rate**

MLP Regression Resolution Analysis



- Momentum bins from (9, 10] to (74, 75]
- Fixed pion efficiency (TPR) as 97.5%
- Plot muon efficiencies (FPR) for MLP and SOTA model

PointNet Architecture

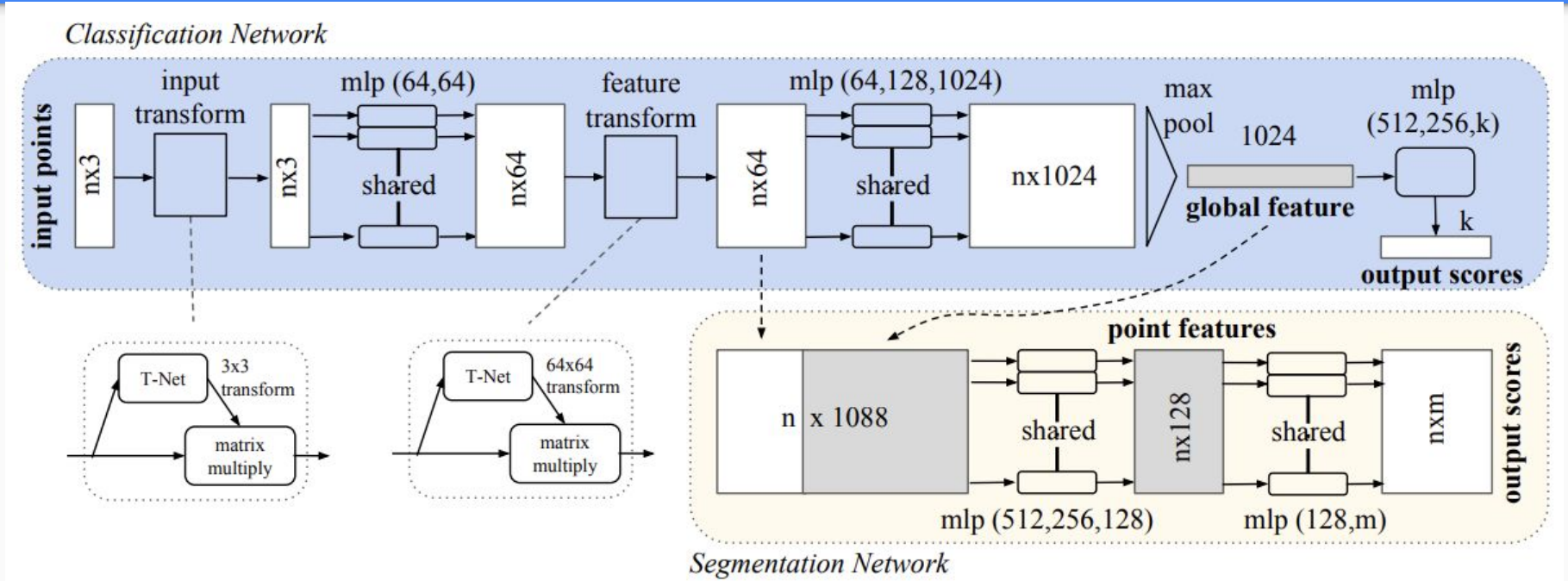


Image Credit: Stanford

PointNet Architecture