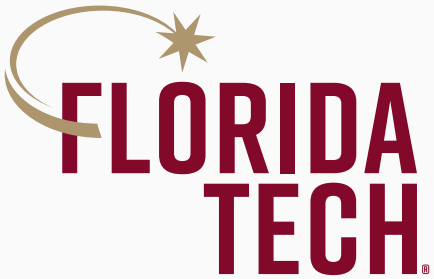
Using XGBoost to Correctly Pair Muons in Events with Two Dimuons from Off-shell Boson Decays

Stephen D. Butalla, Spencer Hirsch, & Marcus Hohlmann

April Meeting, American Physical Society
Florida Institute of Technology

April 25, 2023



Introduction & Motivation

- An attractive conception of dark matter (DM) is the "Hidden Valley" or "Hidden Sector" model, where a spectrum of DM particles, analogous to Standard Model (SM) matter, exist [1]
- Standard Model (SM) gauge group is extended by a "dark," $U(1)_D$ group

$$SU(3) \otimes SU(2) \otimes U(1) \otimes U(1)_D$$

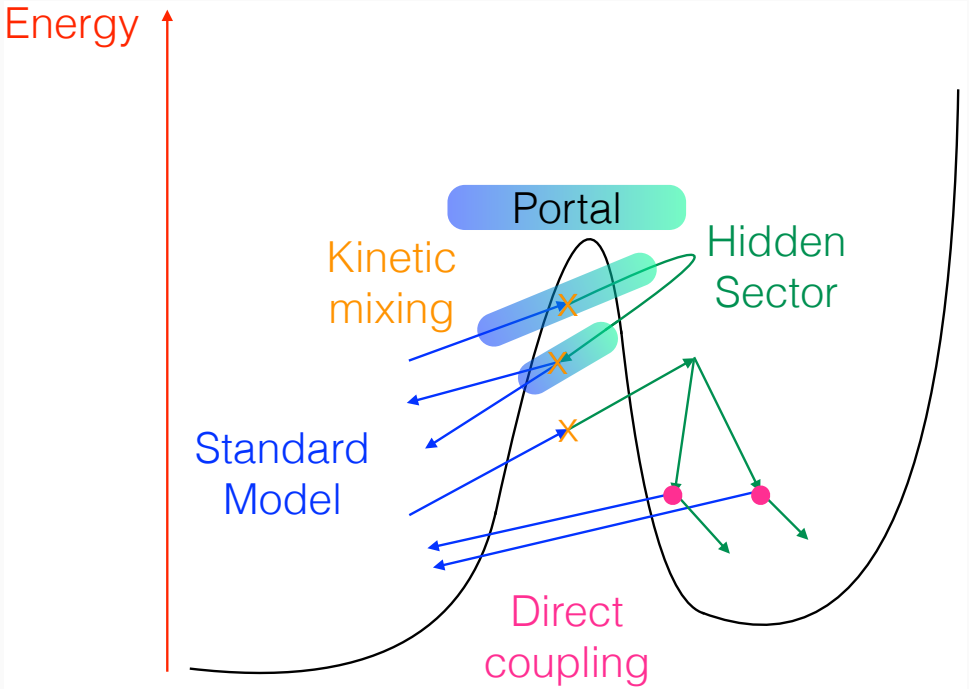
- **This extension produces a dark, massive, spin-1, Abelian mediator: the dark Z boson, Z_D**
- **Kinetic mixing** between $U(1)$ and $U(1)_D$ fields enables DM current and electromagnetic current interactions; spontaneous symmetry breaking allows DM-electroweak interaction

WORK IN PROGRESS

Interaction Lagrangian:

$$\mathcal{L} \subset -\frac{1}{4} B_{\mu\nu} B^{\mu\nu} - \frac{1}{4} Z_{D\mu\nu} Z_D^{\mu\nu} + \frac{\epsilon}{2\cos\theta} Z_{D\mu\nu} B^{\mu\nu} + \frac{1}{2} m_{Z_D}^2 Z_D^{\mu} Z_{D\mu}$$

where $B_{\mu\nu}$ and $Z_{D\mu\nu}$ are the SM and dark field strength tensors for $U(1)$ and $U(1)_D$, respectively, and ϵ is the kinetic mixing coefficient

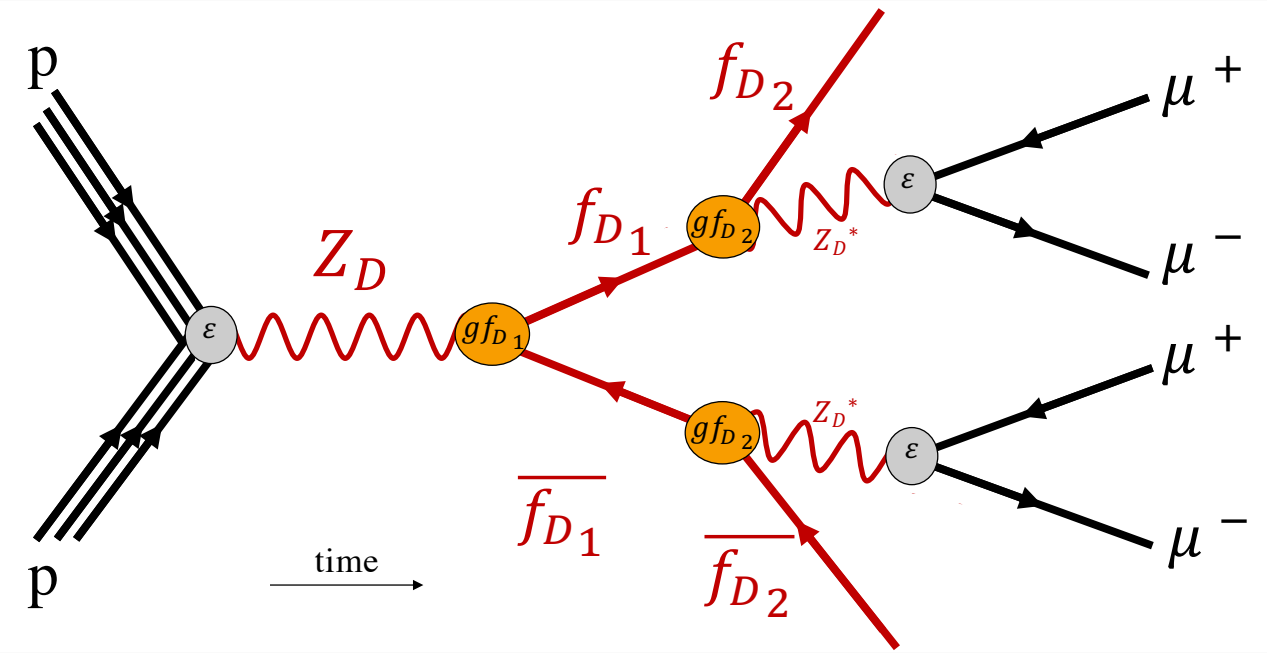


Introduction & Motivation

- The CMS Muon Dark Sector Working Group is engaged in a model-independent search for two new bosons that decay to four muons each
- One of our model-dependent searches interprets this process in the context of the Dark Fermion, or f_D model [2]

$$pp \rightarrow Z_D \rightarrow f_{D_1}\overline{f_{D_1}} \rightarrow f_{D_2}\overline{f_{D_2}}\mu^+\mu^-\mu^+\mu^-$$

where $m_{f_{D1}} > m_{f_{D2}}$ and f_{D_2} is the lightest, stable dark fermion



The Feynman diagram for the f_D model

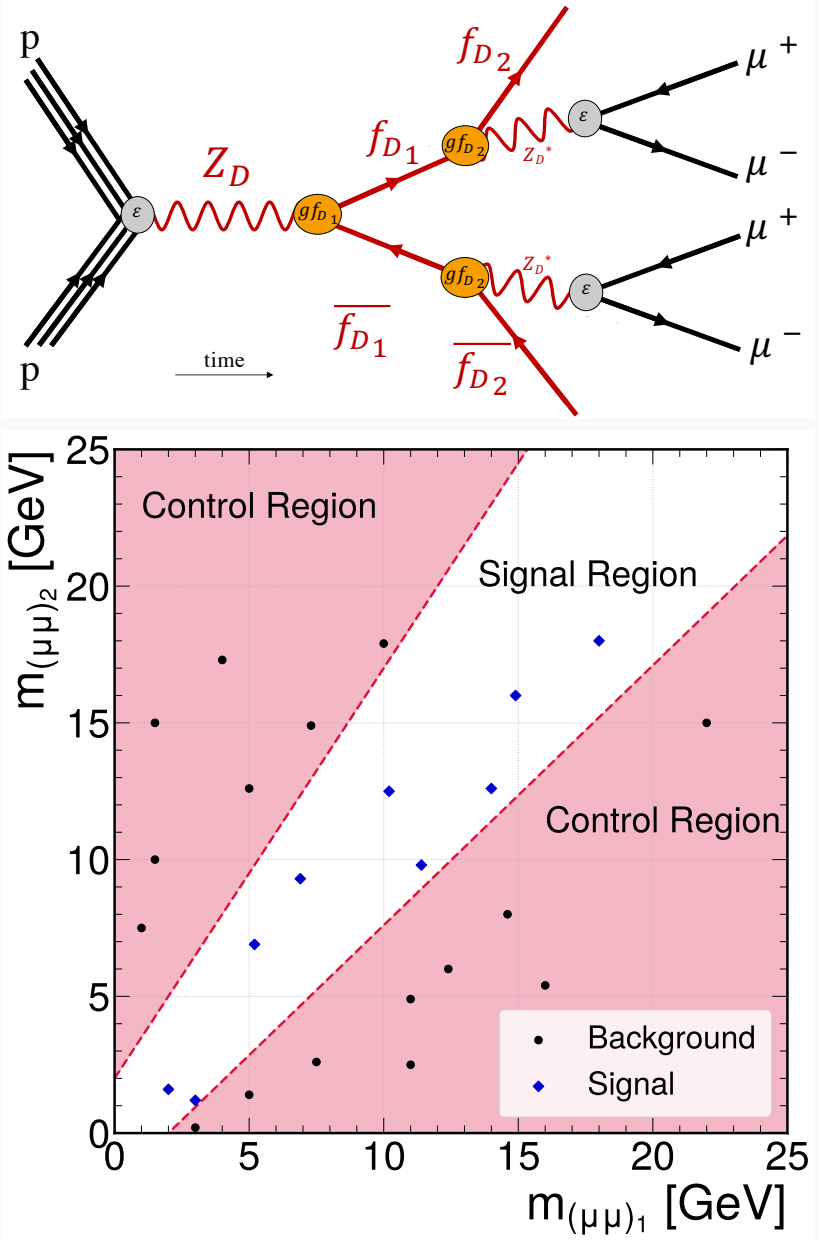
- The final state will consist of two f_{D_2} particles that escape detection, and four muons, paired into two dimuons, which decay from **off-shell** Z_D particles

WORK IN PROGRESS

The Problem

- **The final state muons are produced from off-shell dark bosons, meaning the invariant masses of the two reconstructed Z_D vertices will not necessarily be equal**
- The current analysis uses the signal mass window approach to determine a signal region
- **We need a different approach to pair the muons into dimuons since we cannot require equal invariant masses for the dimuons**
- To solve this problem, we developed a machine learning (ML) model to **correctly** pair each muon into a dimuon
- The model is trained on Monte-Carlo generated and reconstructed data of the f_D model
- The resultant model will eventually be used on real data collected by the CMS experiment

WORK IN PROGRESS

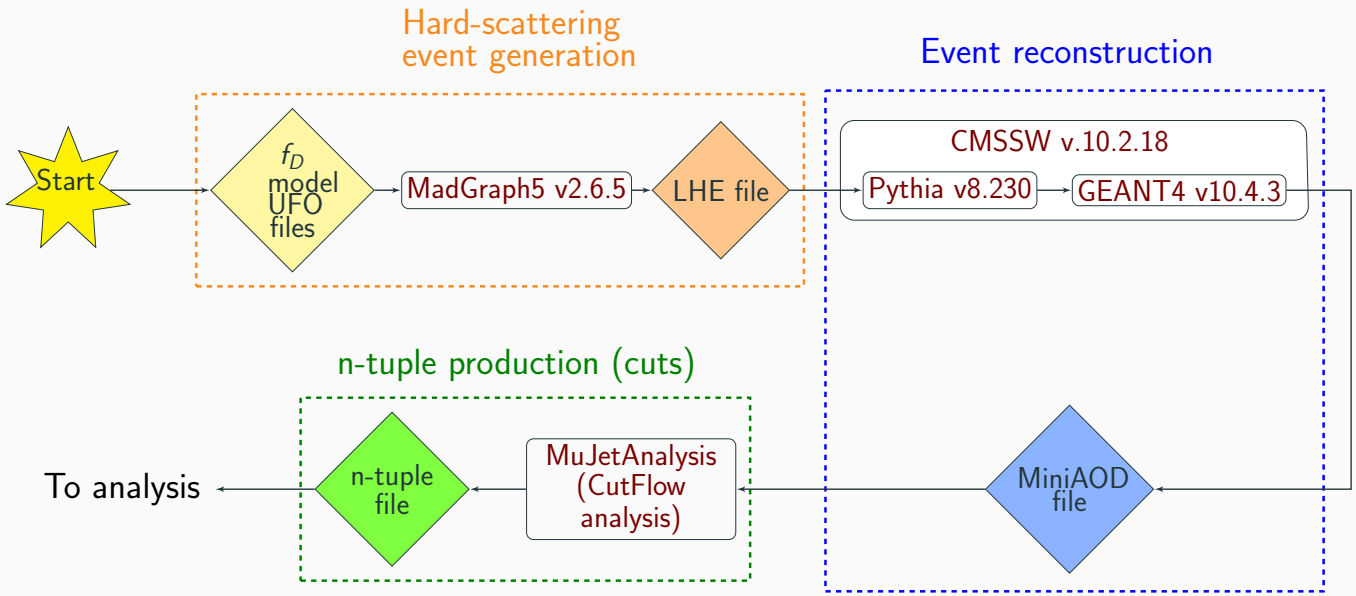


A sketch of the mass signal window method

Monte Carlo Simulation and Reconstruction

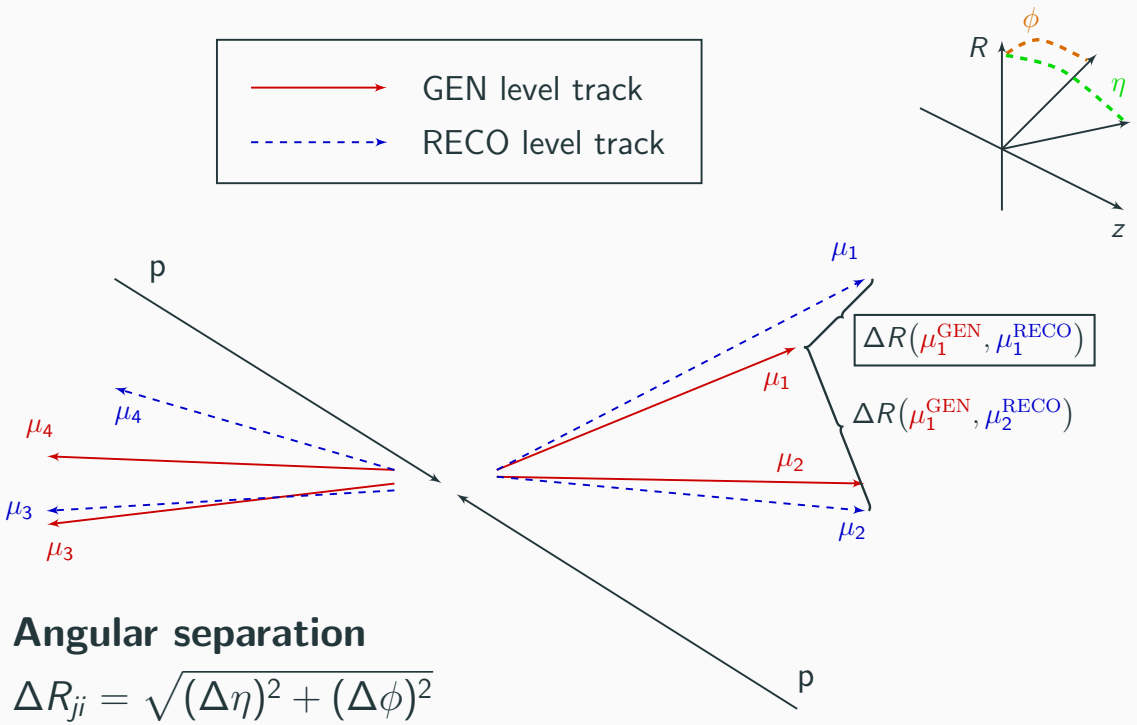
WORK IN PROGRESS

Flow chart of the sample generation process



- Samples of 10^4 events are generated and reconstructed for $85 \leq m_{Z_D} \leq 400$ GeV, with $5 \leq m_{f_{D1}} \leq 60$ GeV ($m_{f_{D2}} = 2$ GeV for all samples)
- Kinematic and geometric cuts placed during n-tuple production (e.g., $p_T > 8$ GeV and $|\eta| < 2.4$ for all 4 muons)
- Further cuts are placed to remove events that fail reconstruction (charge, η, ϕ, etc.)
- Roughly 4000 events survive the selection in each sample

Data Pre-processing



Angular separation

$$\Delta R_{ji} = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$$
$$= \sqrt{(\eta_j - \eta_i)^2 + (\phi_j - \phi_i)^2},\ j \neq i$$

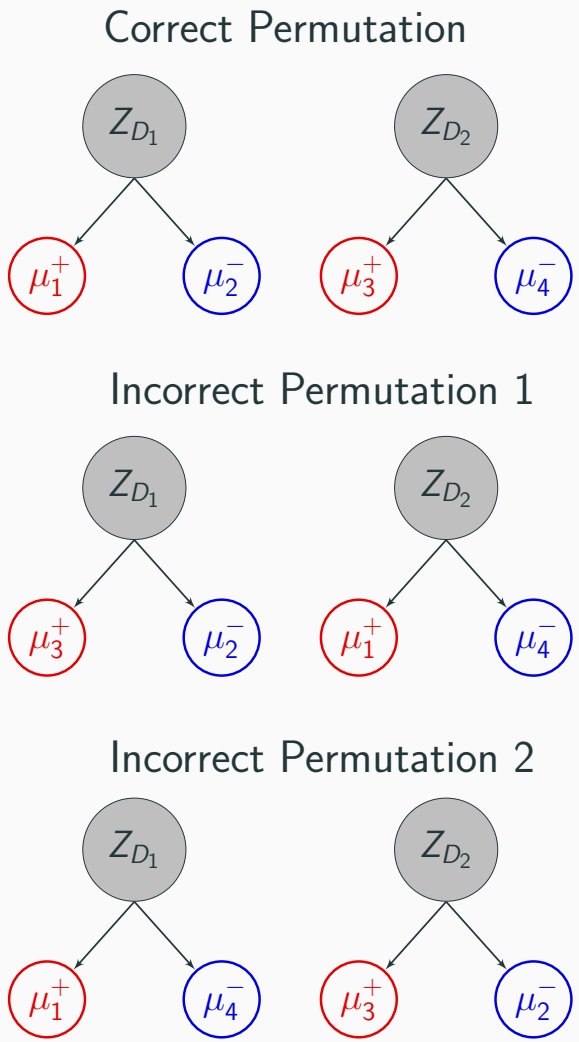
- **For each dataset, we pair the generator (GEN, or "MC truth") level data to the fully reconstructed (RECO) level muon objects, using the following algorithm:**
  1. ΔR is calculated between each GEN and RECO level muon
  2. A GEN muon is chosen at random and matched with the RECO muon with the smallest ΔR
  3. The remaining RECO muon with like charge is paired with the remaining GEN muon
  4. One of the oppositely charged μ selected randomly, then step 2 is repeated
  5. The last RECO muon is paired with the remaining GEN muon

WORK IN PROGRESS

The Final Dataset

WORK IN PROGRESS

- We then **generate** two additional, incorrect permutations which are used as "incorrectly reconstructed dimuons" for training the model
- Requiring each dimuon to be composed of a positive and negative muon, this comes to two additional permutations
- Binary labels assigned to correct (1) and incorrect permutations (0)
- The dataset contains **all** permutations and their associated observables:
  - For each muon: p_T, η, ϕ, Q
  - For each dimuon: ΔR, $\Delta\phi$, invariant mass, m_{inv}
- Each dataset is then randomly shuffled and split into a 70%/30% train/test datatset

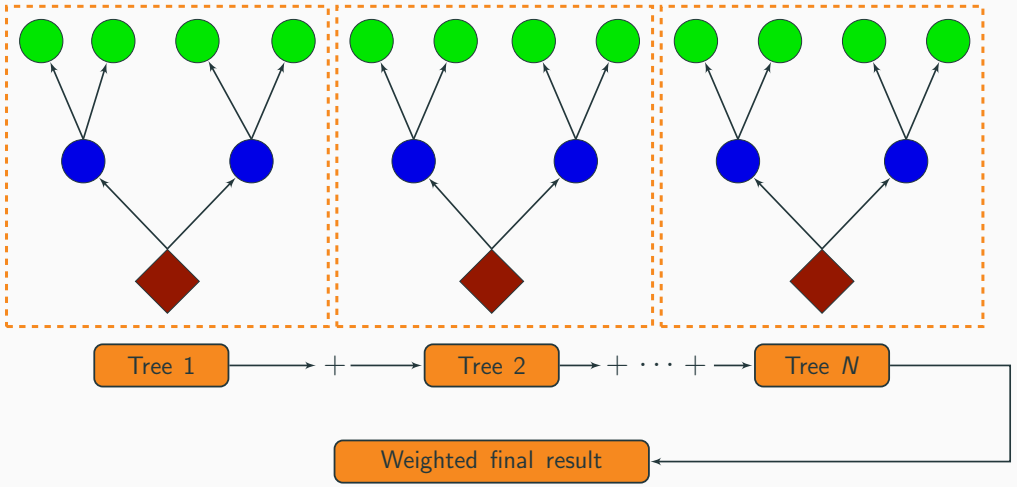


Positive muons swapped to different parent particles

Negative muons swapped to different parent particles

The XGBoost Model

- eXtreme Gradient Boosted decision tree: a "boosted," ensemble decision tree algorithm (supervised machine learning algorithm)
  - Individual **decision trees (DTs) are randomly created** and the **input data are randomly sampled** ("boosting")
  - Each successive DT generated such that it minimizes error/impurity
  - Uses gradient descent for optimizing a loss function
  - Parallelized computation and better caching (improvement to gradient boosting algorithm)
- For the base model all default hyperparameters are used (learning rate, $\tilde{\eta} = 0.3$; maximum tree depth, $D_{\max} = 6$, L_1 and L_2 regularization, $\alpha = 0$, $\lambda = 1$, respectively, etc.)



An (extremely) simplified schematic of an XGBoost model

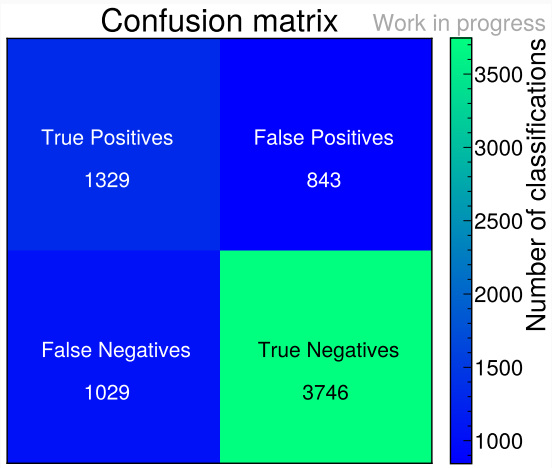
WORK IN PROGRESS

Training/Testing Results for All Mass Points

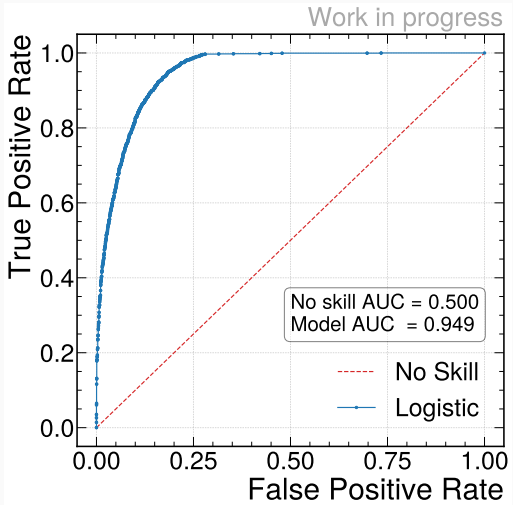
- The initial approach: **train a separate model on each of the 150 different mass points** ($85 \leq m_{Z_D} \leq 400$ GeV, $5 \leq m_{f_{D1}} \leq 60$ GeV, and $m_{f_{D2}} = 2$ GeV)
- Statistics for the testing performance metrics are displayed below for **all of the 150 mass points**

Table 1: Training results for all mass points

	Average	Median	Std. dev.	Min.	Max.
Accuracy	0.959	**0.996**	0.075	0.688	1.00
Matthew's correlation coefficient (MCC)	0.907	**0.990**	0.170	0.272	1.00
Area under the receiver operating characteristic curve	0.981	**1.00**	0.045	0.761	1.00
Execution time [s]	1.19	0.895	0.735	0.127	3.19
Recall	0.939	0.993	0.121	0.442	1.00
Precision	0.937	0.994	0.109	0.550	1.00



Example confusion matrix for $m_{Z_D} = 140$ GeV, $m_{f_{D1}} = 55$ GeV



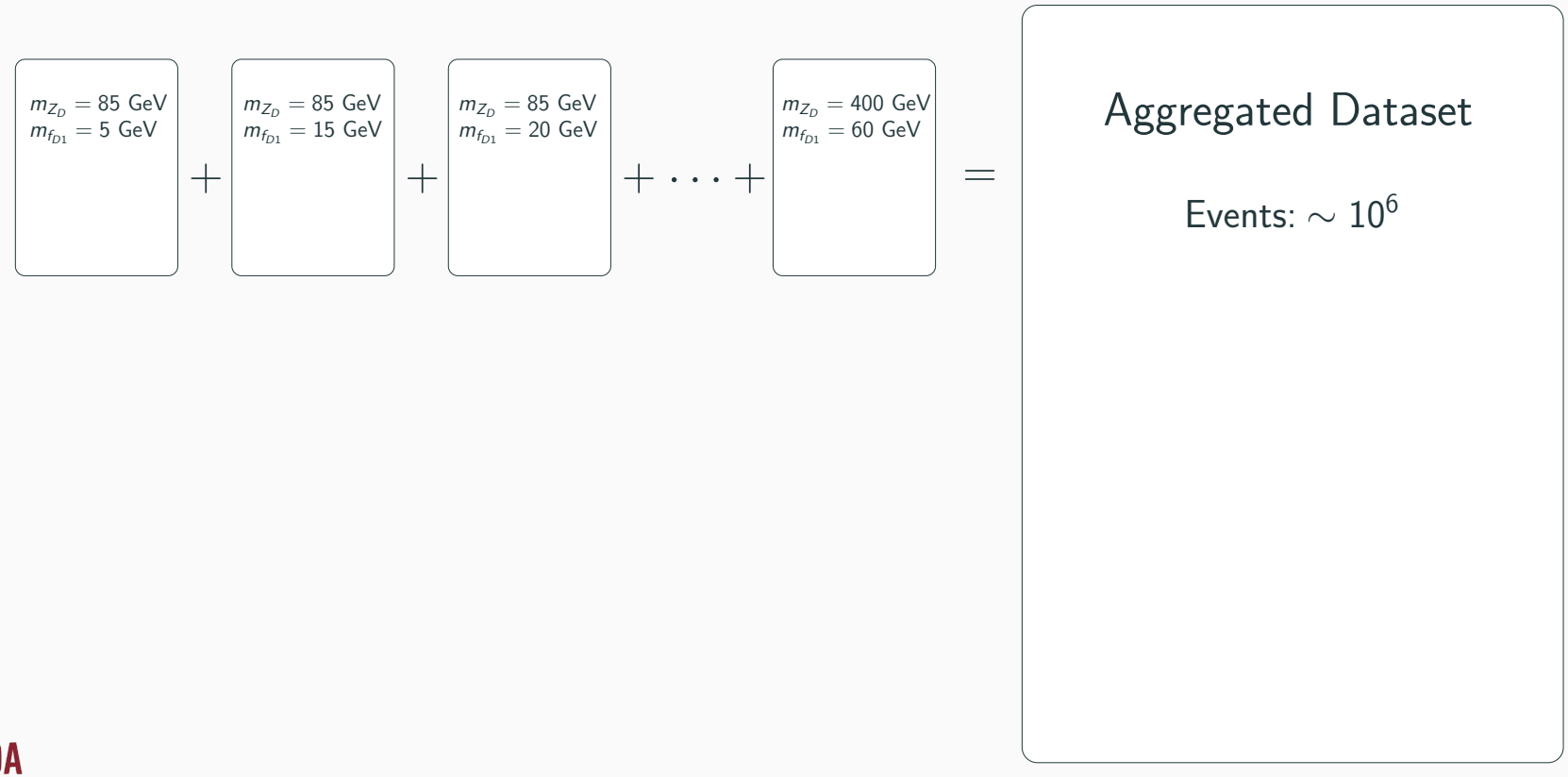
Example receiver operating curve (ROC) curve for $m_{Z_D} = 125$ GeV, $m_{f_{D1}} = 55$ GeV

WORK IN PROGRESS

A Different Approach

- In reality, we don't know the mass of Z_D
- To capture all of the variability in the kinematics and physics over the mass range $85 \leq m_{Z_D} \leq 400$ GeV and $5 \leq m_{f_{D1}} \leq 60$ GeV, **we aggregate all of the mass points into a single dataset**
- The dataset is split into a 70/30 train/test split
- We also perform hyperparameter tuning via a grid search to optimize performance

WORK IN PROGRESS



Training and Hyperparameter Optimization Results

WORK IN PROGRESS

- Pretty good performance with such a diverse input dataset; **∼97% accuracy!**
- Hyperparameter tuning provides minimal improvement in performance
  - ∼ 0.3% increase in accuracy, but ∼ 55% increase in computation time

Table 2: Training results for aggregated dataset

	Default	Optimal
Learning rate ($\tilde{\eta}$)	0.3	0.4
Maximum dept ($D_{\max}$)	6	10
L_1 regularization (α)	0	5
L_2 regularization (λ)	1	0

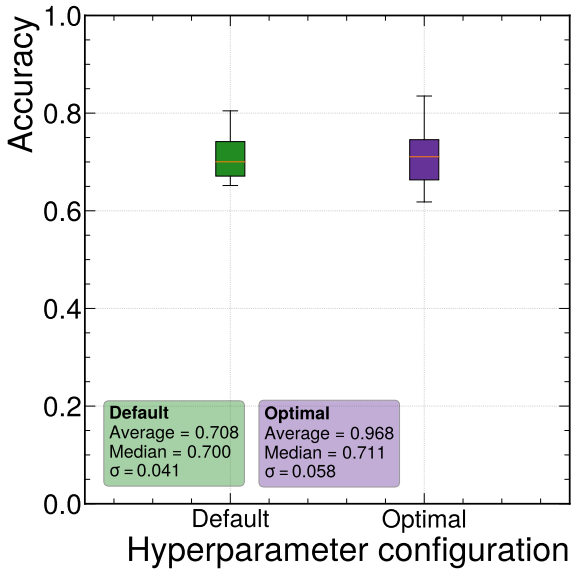
Table 3: Training results for aggregated dataset

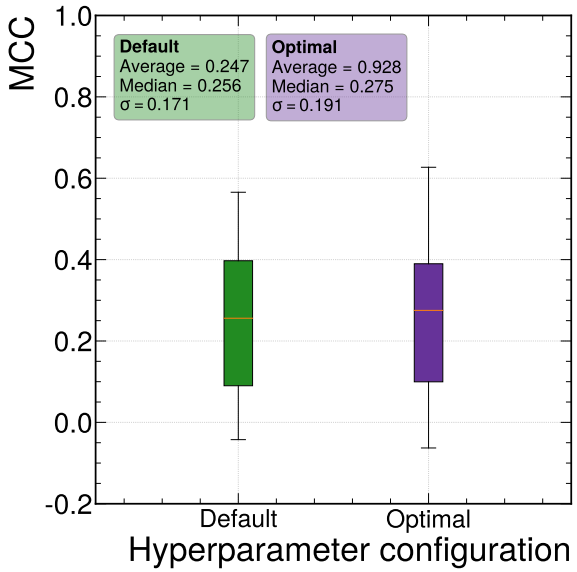
	MCC	AUC	Accuracy	Time (s)	F_1	Precision	Recall
Default hyperparamers	0.934	0.997	0.971	62.3	0.956	0.960	0.952
Optimal hyperparameters	0.942	0.998	0.974	96.3	0.961	0.963	0.960
% difference between opt. and def.	0.857	0.100	0.309	54.6	0.523	0.313	0.840

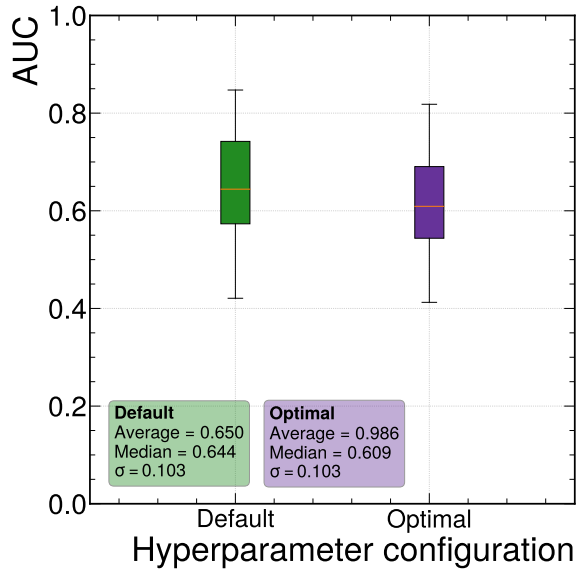
Evaluating Aggregated Model Performance on all Mass Points

- Two trained models, **using the aggregated dataset**, with the **default** and **optimal** hyperparameters used to evaluate **all 150 individual samples**
- Distributions reflect the prediction metrics **for all 150 mass points**

WORK IN PROGRESS







Parametrizing the dataset

- The performance metrics distributions shown on the previous slide are much poorer than the testing metrics
  - The model is probably over-training on the aggregated dataset
  - Hyperparameter tuning doesn't help with the over-training
- To improve results, we tried **parametrizing** [4] the dataset
  - For each event, we add m_{Z_D}, $m_{f_{D1}}$, and $m_{f_{D2}}$

WORK IN PROGRESS

	selpT0	selpT1	selpT2	selpT3	selEta0	selEta1	selEta2	selEta3	selPhi0	selPhi1	...	selCharge3	dPhi0	dPhi1	dRA0	dRA1	invMassA0	invMassA1	mZD	mfD1	mfD2
0	27.203215	9.378386	8.010424	6.051712	0.210876	0.168951	0.035462	0.212726	0.934701	0.827434	...	1.0	0.107267	-0.114202	0.115169	0.210866	1.838784	1.469289	85.0	5.0	2.0
1	27.203215	6.051712	8.010424	9.378386	0.210876	0.212726	0.035462	0.168951	0.934701	2.377698	...	1.0	-1.442997	1.436062	1.442999	1.442253	16.949601	11.463321	85.0	5.0	2.0
2	8.010424	9.378386	27.203215	6.051712	0.035462	0.168951	0.210876	0.212726	2.263496	0.827434	...	1.0	1.436062	-1.442997	1.442253	1.442999	11.463321	16.949601	85.0	5.0	2.0
3	20.921888	4.583872	9.627180	3.834181	0.548159	0.366273	-0.378856	-0.574116	2.699803	2.602110	...	-1.0	0.097693	0.154653	0.206463	0.249087	2.023875	1.514236	85.0	5.0	2.0
4	9.627180	4.583872	20.921888	3.834181	-0.378856	0.366273	0.548159	-0.574116	-1.586711	2.602110	...	-1.0	-4.188820	4.441167	4.254578	4.580771	12.571529	17.762714	85.0	5.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3006661	46.458000	66.191505	13.230703	29.568367	0.917999	-0.588407	-0.122839	-0.936213	1.526946	-2.017315	...	-1.0	3.544260	3.074131	3.851109	3.179915	142.163427	42.853988	400.0	60.0	2.0
3006662	29.568367	13.230703	66.191505	46.458000	-0.936213	-0.122839	-0.588407	0.917999	-1.676499	1.397632	...	-1.0	-3.074131	-3.544260	3.179915	3.851109	42.853988	142.163427	400.0	60.0	2.0
3006663	75.562950	29.421379	37.761433	17.095875	0.021898	-0.272916	-1.011629	-0.083937	-2.240009	-2.681245	...	-1.0	0.441236	-0.572003	0.530665	1.089862	24.909462	28.321558	400.0	60.0	2.0
3006664	75.562950	37.761433	29.421379	17.095875	0.021898	-1.011629	-0.272916	-0.083937	-2.240009	-0.040200	...	-1.0	-2.199810	-3.213048	2.430502	3.218601	111.325321	45.026509	400.0	60.0	2.0
3006665	17.095875	29.421379	37.761433	75.562950	-0.083937	-0.272916	-1.011629	0.021898	0.531803	-2.681245	...	-1.0	3.213048	2.199810	3.218601	2.430502	45.026509	111.325321	400.0	60.0	2.0

3006666 rows × 25 columns

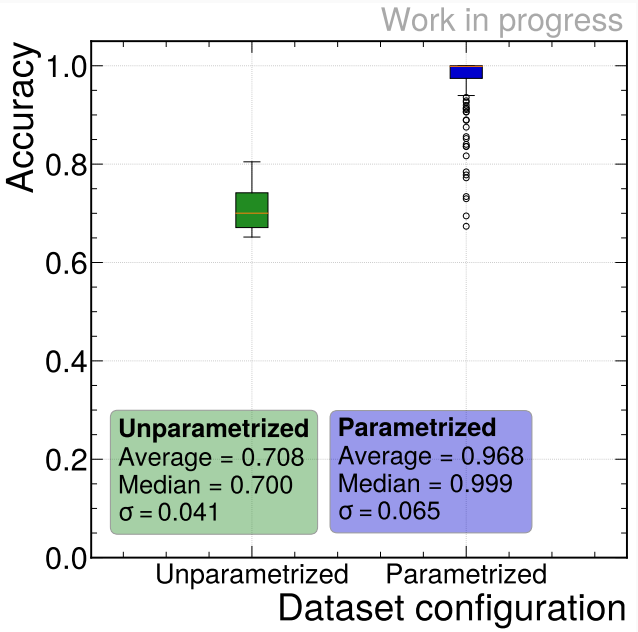
A screenshot of the parametrized dataset used for training the XGBoost model

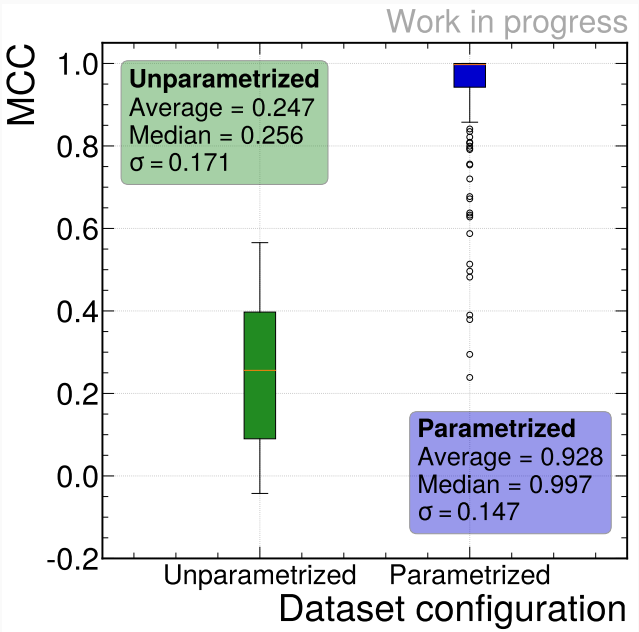
- A new model is trained on the **aggregated** and **parametrized** dataset

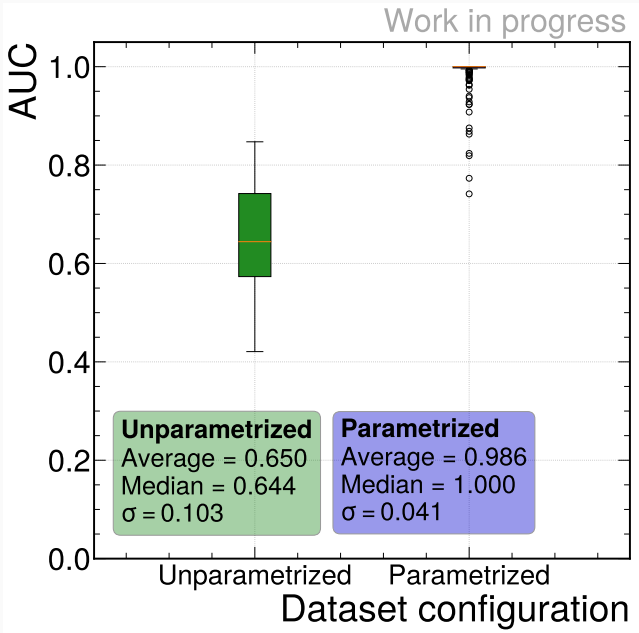
Evaluation results of aggregated, parametrized dataset-trained model

- Models are trained on the aggregated and parametrized dataset (both w/ default hyperparameters) and used to predict pairing for **each of the 150 mass points**
- Comparison of performance metric distributions of models trained on unparametrized and parametrized datasets with default hyperparameters
- All of the 150 mass points are represented in the box plots

WORK IN PROGRESS







Summary & Conclusion

The ML approach to correctly pair muons to off-shell dark bosons works!

- Low-level observables for each muon (p_T, η, ϕ, Q), and high-level observables for each dimuon (ΔR, $\Delta\phi$, m_{inv}) are used as input for training
- XGBoost models trained on individual mass point datasets show a 95.9% average accuracy for the test dataset
- An aggregated dataset of all individual mass points was constructed and used to train a model; test dataset prediction accuracy: 97.1%
- Hyperparameter tuning on the aggregated dataset showed marginal improvement ($\sim$ 0.3% increase in accuracy but 50% increase in computation time)
- **The aggregated, parametrized dataset yielded the highest performance metrics when testing individual mass points (the most important case)**
  - Average unparametrized accuracy: 70.8%
  - **Average parametrized accuracy: 96.8%**
- For analyzing real data, we now have a very accurate model that can accept three free parameters: m_{Z_D}, $m_{f_{D1}}$, and $m_{f_{D2}}$
  - These parameters can be chosen based off of simulation or theoretical considerations

WORK IN PROGRESS

References

[1] D. Curtin, R. Essig, S. Gori & J. Shelton, "Illuminating dark photons with high-energy colliders," *Journal of High Energy Physics*, **2015**(157), 2015. *Journal of High Energy Physics*, **118**(2009), 2009.

[2] M. Rahmani et al., "Vector-Portal Search for Long Lived Dark Matter Particles," talk presented at the *2020 APS April Meeting 2020*, April 19, 2020.

[3] The ATLAS Collaboration, "Search for resonant production in the fully leptonic final state in proton–proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," 2022, arXiv:2207.03925.

[4] P. Baldi, et al., "Parameterized Machine Learning for High-Energy Physics," *The European Physical Journal C*, **76**(235), arXiv:1601.07913, 2016.

Backup

Dark Photon — EM Current Coupling

- Consider just the kinetic terms of the interaction Lagrangian:

$$\mathcal{L}_{\text{kin}} = -\frac{1}{4} B_{\mu\nu} B^{\mu\nu} - \frac{1}{4} X_{D\mu\nu} X_D^{\mu\nu} + \frac{\epsilon}{2\cos\theta} X_{D\mu\nu} B^{\mu\nu}$$

- Diagonalizing the terms in the above expression by rotating the field [**?**]:

$$\begin{pmatrix} \Omega_0^\mu \\ \Omega_1^\mu \end{pmatrix} = \begin{pmatrix} (1-\epsilon^2)^{-1/2} & 0 \\ -\epsilon(1-\epsilon^2)^{-1/2} & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} Z_D^\mu \\ A^\mu \end{pmatrix}$$

- Writing the interaction Lagrangian in terms of currents so we can substitute the diagonalized result:

$$\mathcal{L} = e_{\text{EM}} J_{\text{EM}\mu} \Omega_0^\mu + e_D J_{D\mu} \Omega_1^\mu$$

where Ω_i are Abelian gauge bosons

- Substituting the matrix/vector product of $\left(\Omega_0^\mu \; \Omega_1^\mu\right)^T$ into the Lagrangian above and requiring $\sin\theta = 0$ and $\cos\theta = 1$

$$\boxed{\mathcal{L} = (1-\epsilon^2)^{-1/2} (e_D J_{D\mu} - e\epsilon J_{\text{EM}\mu}) Z_D^\mu + e J_{\text{EM}\mu} A^\mu}$$

- Showing that the dark photon Z_D is coupled to the EM current of SM particles (and the dark sector current)

Dark Photon — Z Boson Coupling

- The same procedure can be conducted to determine the couplings to the weak bosons
- Diagonalizing the SM neutral fields W^3, B^μ, and our new dark photon field, $X_{\mu\nu}$ [**?**]:

$$\begin{pmatrix} W^3_\mu \\ B_\mu \\ X_\mu \end{pmatrix} = \begin{pmatrix} \cos\theta_W & \sin\theta_W & -\epsilon\sin\theta_W \\ -\sin\theta_W & \cos\theta_W & -\epsilon\cos\theta_W \\ \epsilon\tan\theta_W & 0 & 1 \end{pmatrix} \begin{pmatrix} Z_\mu \\ A_\mu \\ Z_{D\mu} \end{pmatrix}$$

  where θ_W is the Weinberg weak mixing angle
- After some algebra, the interaction Lagrangian takes the form [**?**]:

$$\mathcal{L} = -e_{\mathrm{EM}}\epsilon J^\mu Z_{D\mu} + e_D \tan\theta_W J^\mu_D Z_\mu + e_D J^\mu_D Z_{D\mu}$$

  thus showing the coupling of the Z_D to SM EM current and the Z_D coupling to the SM Z boson

- Angular separation, $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$, calculated between each GEN and RECO level muon

Step 1: Randomly select a GEN muon and properly match it with a RECO muon

(Pseudo)-random number generator [0, 3] → 2 →

		+	−	+	−
		GEN μ_0	GEN μ_1	GEN μ_2	GEN μ_3
−	RECO μ_0	ΔR_{00}	ΔR_{01}	ΔR_{02}	ΔR_{03}
+	RECO μ_1	ΔR_{10}	ΔR_{11}	ΔR_{12}	ΔR_{13}
−	RECO μ_2	ΔR_{20}	ΔR_{21}	ΔR_{22}	ΔR_{23}
+	RECO μ_3	ΔR_{30}	ΔR_{31}	ΔR_{32}	ΔR_{33}

→ min = ΔR_{12} →

RECO μ_1 = GEN μ_2

- A GEN muon is chosen at random (0–3), and is matched with the RECO muon that has a minimum ΔR

Step 2: Remaining GEN muon with same charge is paired with other RECO muon

		+	−	+	−
		GEN μ_0	GEN μ_1	GEN μ_2	GEN μ_3
−	RECO μ_0	ΔR_{00}	ΔR_{01}	ΔR_{02}	ΔR_{03}
+	RECO μ_1	ΔR_{10}	ΔR_{11}	ΔR_{12}	ΔR_{13}
−	RECO μ_2	ΔR_{20}	ΔR_{21}	ΔR_{22}	ΔR_{23}
+	RECO μ_3	ΔR_{30}	ΔR_{31}	ΔR_{32}	ΔR_{33}

→ RECO μ_1 = GEN μ_2

↓

RECO μ_3 = GEN μ_0

- One of the oppositely charged muons are selected randomly, and the minimum ΔR process is performed again

Step 3: Random opposite charge GEN muon randomly selected, minimum ΔR between GEN and RECO muon used to pair

(Pseudo)-random number generator (1 or 3) → 1 →

		+	−	+	−
		GEN μ_0	GEN μ_1	GEN μ_2	GEN μ_3
−	RECO μ_0	ΔR_{00}	ΔR_{01}	ΔR_{02}	ΔR_{03}
+	RECO μ_1	ΔR_{10}	ΔR_{11}	ΔR_{12}	ΔR_{13}
−	RECO μ_2	ΔR_{20}	ΔR_{21}	ΔR_{22}	ΔR_{23}
+	RECO μ_3	ΔR_{30}	ΔR_{31}	ΔR_{32}	ΔR_{33}

→ min = ΔR_{01} →

RECO μ_0 = GEN μ_1

- The last RECO muon is paired with the remaining GEN muon

Step 4: Remaining oppositely charged GEN muon paired with other RECO muon

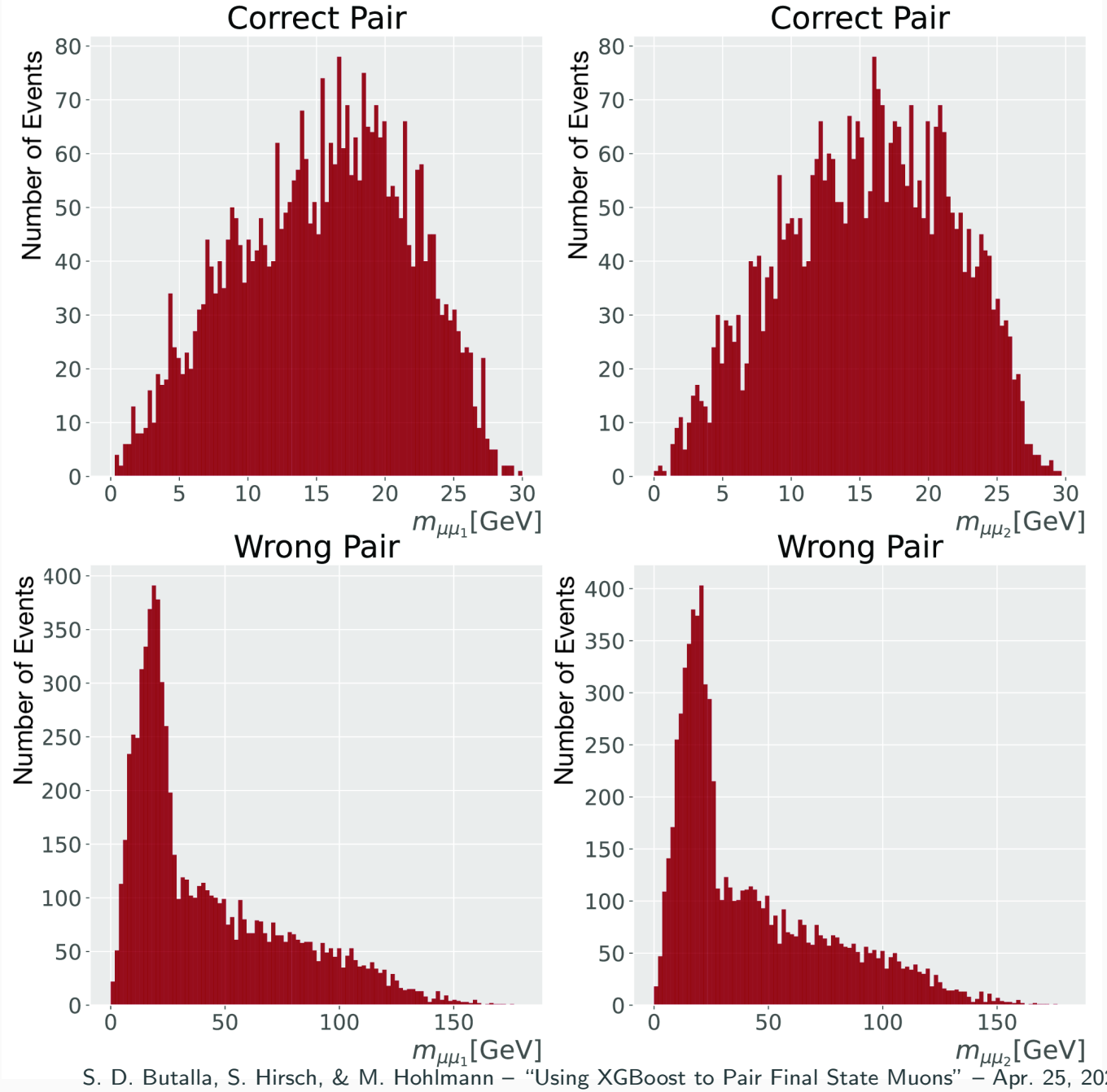
		+	−	+	−
		GEN μ_0	GEN μ_1	GEN μ_2	GEN μ_3
−	RECO μ_0	ΔR_{00}	ΔR_{01}	ΔR_{02}	ΔR_{03}
+	RECO μ_1	ΔR_{10}	ΔR_{11}	ΔR_{12}	ΔR_{13}
−	RECO μ_2	ΔR_{20}	ΔR_{21}	ΔR_{22}	ΔR_{23}
+	RECO μ_3	ΔR_{30}	ΔR_{31}	ΔR_{32}	ΔR_{33}

→ RECO μ_0 = GEN μ_1

↓

RECO μ_2 = GEN μ_3

Dimuon Invariant Mass Distributions



Review of statistical metrics for binary classification problems

- We need metrics to quantitatively assess the performance of any predictive model
- Suppose we have two categories that we need to place things in (binary classification problem), being group 0 (negative classification) and group 1 (positive classification)
- In our case, **we are classifying whether event data contain properly reconstructed (correctly-ordered) final-state muons (true/false) or not**
- At the lowest level, we have the confusion matrix; for a binary classification problem, the matrix elements are:

$$C = \begin{bmatrix} \mathrm{TP} & \mathrm{FP} \\ \mathrm{FN} & \mathrm{TN} \end{bmatrix}$$

  where TP = true positive, FP = false positive (the so-called "Type I error")
  FN = false negative (the so-called "Type II error"), TN = true negative
  Note: `scikit-learn` **fills its confusion matrix out-of-order!**

- From here, we can construct more sophisticated metrics:
  - Accuracy (also known as the "Hohlmann Metric")
  - Precision
  - Recall
  - F_1 score
  - Receiver operating characteristic (ROC) curve
  - Area under the (ROC) curve (AUC)
  - And many more...

Accuracy

- Exactly how it sounds: **the ratio of the truly classified events out of all classifications performed**
- **Note: this includes both categories!** So the true positives and true negatives for classifying data into both category 0 and 1
- Mathematical expression:

$$A = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} = \frac{C_{00} + C_{01}}{\sum_i \sum_j C_{ij}}$$

with $A \in [0, 1]$

Matthew's Correlation Coefficient

- An improved version of the F_1 score (the definition of the coefficient includes all elements of the confusion matrix; F_1 only includes 3)
- Measures the magnitude of correlation between true positives/negatives, and false positives/negatives
- Essentially equivalent to the chi-square test for a 2×2 table

$$|M| = \sqrt{\chi^2/n}$$

- Mathematical expression:

$$M = \frac{\mathrm{TP}(\mathrm{TN}) - \mathrm{FP}(\mathrm{FN})}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

with $M \in [-1, 1]$

Precision

- **Out of all of the positive classifications (group 1), the ratio of the truly classified events out of the true and false positives**
- The name is quite fitting, as it really describes how precise the model is at classifying the data and not erroneously classifying the data on "accident"
- Mathematical expression:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{C_{00}}{C_{00} + C_{01}}$$

with $P \in [0, 1]$

Recall

- **The ratio of the truly classified data in group 1 to the truly classified and erroneously classified (false negative) data**
- Gives an idea of how often you are correctly classifying the data into group 1 (positive classification) out of all of the data in group 1 that exist
- Mathematical expression:

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{C_{00}}{C_{00} + C_{10}}$$

with $R \in [0, 1]$

F_1 score

- **The harmonic mean of the recall and precision**
- Provides an "average" of the recall and precision

$$\begin{aligned} F_1 &= \frac{2}{R^{-1} + P^{-1}} \\ &= \frac{2RP}{P + R} \\ &= \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})} \end{aligned}$$

with $F_1 \in [0, 1]$

- Another related (but less commonly used) statistic is the Fowlkes-Mallows index:

$$FM = \sqrt{\frac{\mathrm{TP}^2}{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})}}$$

with $FM \in [0, 1]$

  - Provides roughly the same information as the F_1 score

ROC curves and AUC

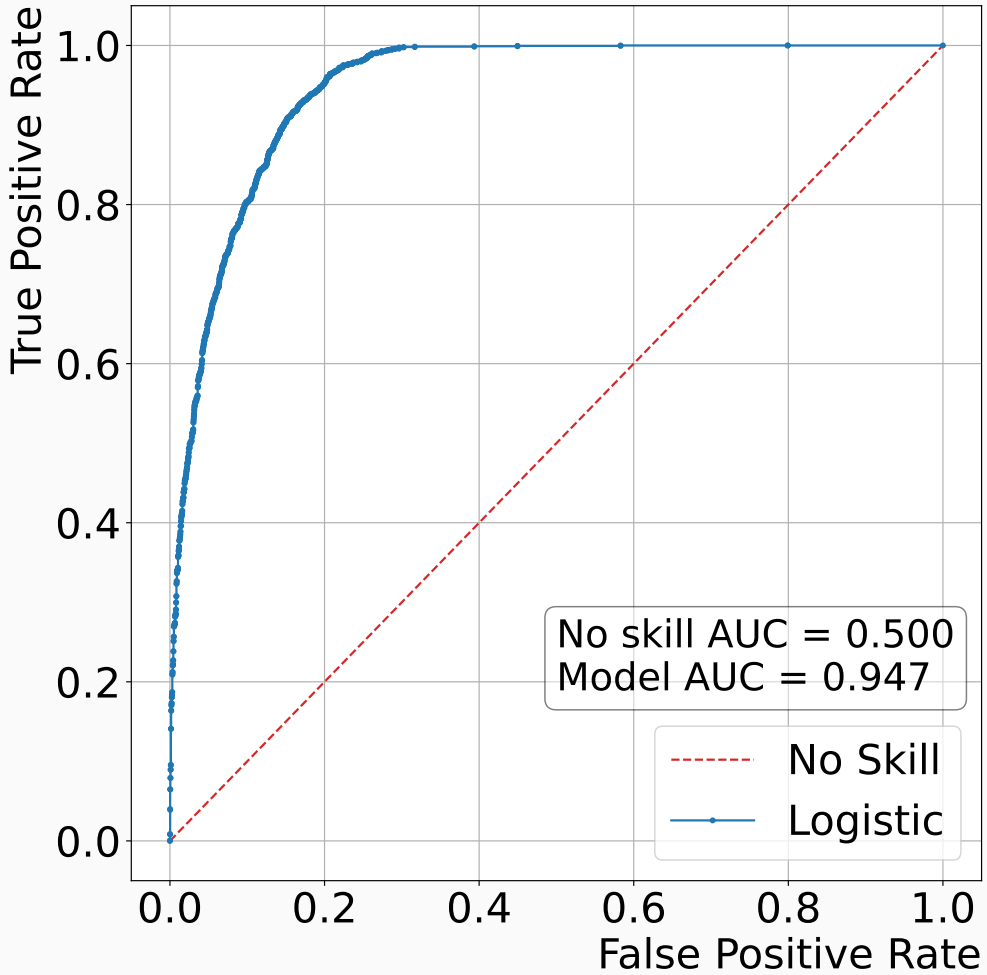
- The true (ordinate) and false (abscissa) positive rates plotted against each other at different thresholds
- True positive rate (TPR):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- False positive rate (FPR):

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}}$$

- Can also be interpreted as a plot of the sensitivity as a function of the FPR at different thresholds
- Area under the curve is the integral of the ROC curve (AUC $\in [0,\ 1]$)
- AUC provides a compact, single statistic that can be used to evaluate the performance of the model



Example ROC curve for
$m_{Z_D} = 125$ GeV, $m_{s_D} = 40$ GeV

Hyperparameter tuning

- Hyperparameter: a manually chosen, user-defined value that is used as input to the model and specifies the model architecture or facets of the underlying mathematical machinery.

  Examples:

  1. In deep neural nets: the learning rate (LR), number of layers, neuron activation function, etc.
  2. In boosted decision trees: the LR, maximum tree depth, L_1 and L_2 penalty values, etc.
- Most models are extremely sensitive to the values of the hyperparameters $\Rightarrow$ hyperparameters usually need to be tuned to achieve the best performance
- Tuning can be done manually (e.g., grid search, random search, or intuition) or automatically (e.g., Bayesian tuning)
- We consider 5 parameters:
  - Objective/loss function: The function that evaluates the performance of the model by comparing the actual and predicted value (**this is what is being minimized by the optimization algorithm!**)
  - L_1 (lasso) and L_2 (ridge) regularization parameters: a "penalty" term added to the loss function which helps prevent overfitting (the L_i name come from the L_i norms)
  - Maximum tree depth: the maximum "height" of the tree; maximum number of nodes from the root to the furthest leaf node
  - Learning rate: the "step size" of the optimization algorithm
- **The process of optimizing the performance is always a tradeoff between the metric gauging the performance and the execution time**

Hyperparameter descriptions

- Objective/loss function: The function that evaluates the performance of the model by comparing the actual and predicted value (**this is what is being minimized by the optimization algorithm!**):
  - General form:

$$L(\mathbf{y}, \hat{\mathbf{y}}, \phi) = \sum_i l(y_i, \hat{y}_i) + f(\phi)$$

    where $\mathbf{y}$ is the true label and $\hat{\mathbf{y}}$ is the prediction ($\in [0, 1]$), f is (are) some penalty function(s), and ϕ is the vector of variables for that (those) penalty function(s)
  - Binary logistic regression:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_i \left[-y_i \ln \hat{y}_i + (1 - y) \ln(1 - \hat{y}_i) \right]$$

  - Squared error (SE) regression (**default**):

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \sum_i \left[\ln(y_i + 1) - \ln(\hat{y}_i + 1) \right]^2$$

  - Hinge function:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = \sum_i \max\left(0, 1 - y_i \cdot \hat{y}_i\right)$$

  - **NOTE: the default loss function is best suited for regression, not classification!**
  - Both the hinge and binary logistic regression functions are purposed for binary classification (binary logistic regression = binary classification)

Hyperparameter descriptions

- L_1 (parameter denoted by α) and L_2 (parameter denoted by λ) regularization parameters:

$$L'(\mathbf{y}, \hat{\mathbf{y}}, \alpha, \lambda) = L(\mathbf{y}, \hat{\mathbf{y}}) + \alpha \sum_j |w_j| + \frac{\lambda}{2} \sum_j w_j^2$$

  where w_j are the weights of each leaf node
- Maximum tree depth: the maximum "height" of the tree; maximum number of nodes from the root to the furthest leaf node
- Learning rate (η): the "step size" of the optimization algorithm; directly impacts the weights at leaf nodes:

$$w_j = \eta \frac{\sum_k \frac{\partial L}{\partial y_k}}{\sum_k \frac{\partial^2 L}{\partial y_k^2} + \lambda + \alpha}$$

- We want to optimize the output of the model with respect to these five hyperparameters (i.e., achieve the highest metric value with the correct combination of hyperparameters)
- This is just an optimization problem with an N-dimensional hyperplane ($N = 5$ here)
- A 5D grid is set up, and the following hyperparameter values are iterated over

  (default in red):

  1. Objective function: Binary logistic, squared error, hinge
  2. L_1 regularization: $\alpha \in [0, 1, 2, 3, 4, 5]$
  3. L_2 regularization: $\lambda \in [0, 1, 2, 3, 4, 5]$
  4. Maximum tree depth: $m \in [3, 6, 10, 12, 15]$
  5. Learning rate: $\eta \in [0.1, 0.3, 0.4, 0.5, 0.6]$

Feature Importance

- The model is trained on the aggregated, parametrized dataset
- This bar plot displays the feature importance, i.e., the proportion that each observable contributes to making the decision (correct permutation or incorrect permutation)