Particle Identification and Performance Evaluation

Daniel Lersch

September 21, 2020

dlersch@jlab.org

Overview

This lecture consists of two parts

- **1. Part: How to**
  - i) Example analysis on a toy data set
  - ii) Definition and comparison of basic evaluation metrics
- **2. Part: Hands-On**
  - i) Perform your own analysis on different toy data sets
  - ii) Train and evaluate your own classifier with scikit

This Lecture...

... will mainly focus on supervised learning with labeled data

... covers only a small fraction of all available classification metrics

... does NOT turn you into an AI specialist

... aims to give you a rough idea about particle identification with machine learning

... uses a generated and simple (in terms of complexity) data set

... will not deal with machine learning in great detail (done in "‘ML for Beginners“ by Thomas Stibor)

... includes material mainly from:

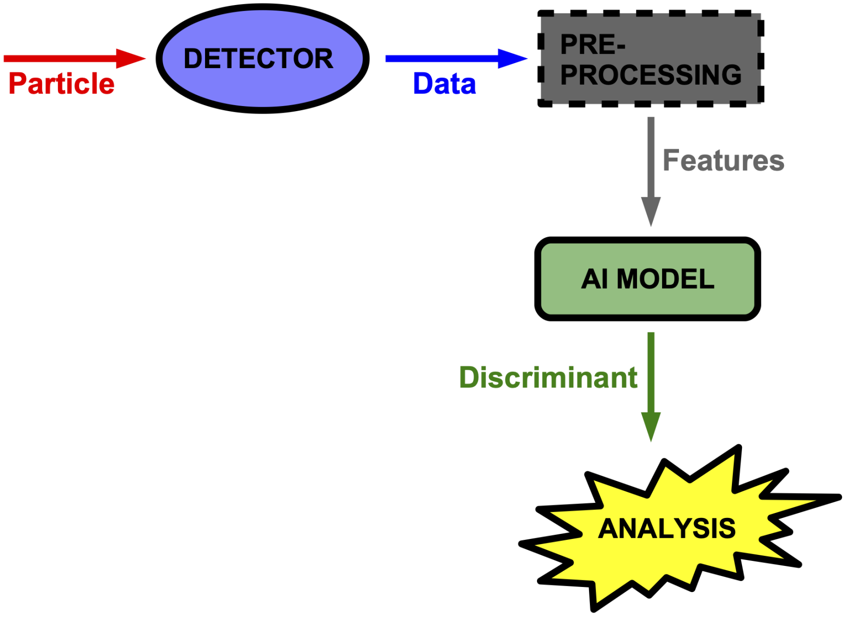
- Wikipedia
- Apache Spark Documentation
- Scikit Documentation

... most likely contains several errors → please report them

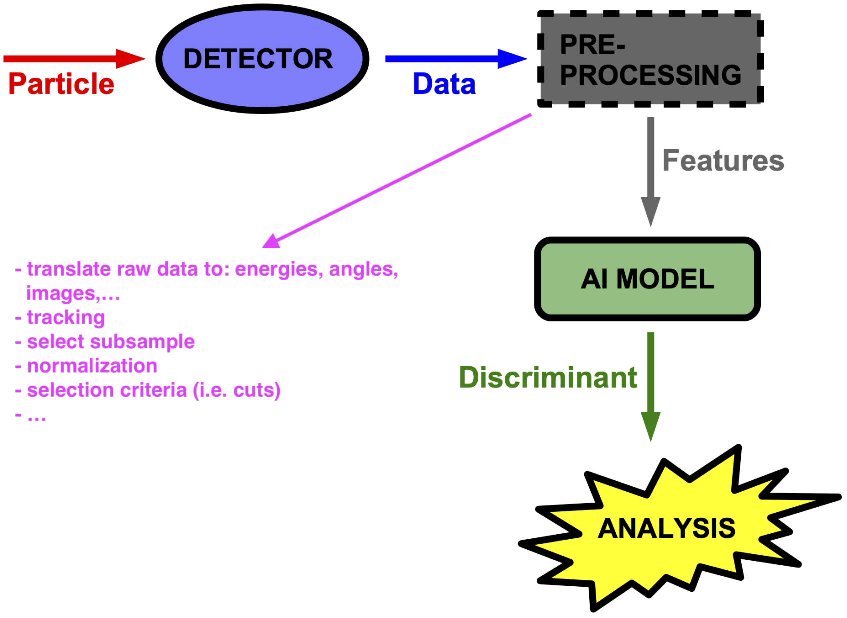
Part I: How to

Picture taken from here

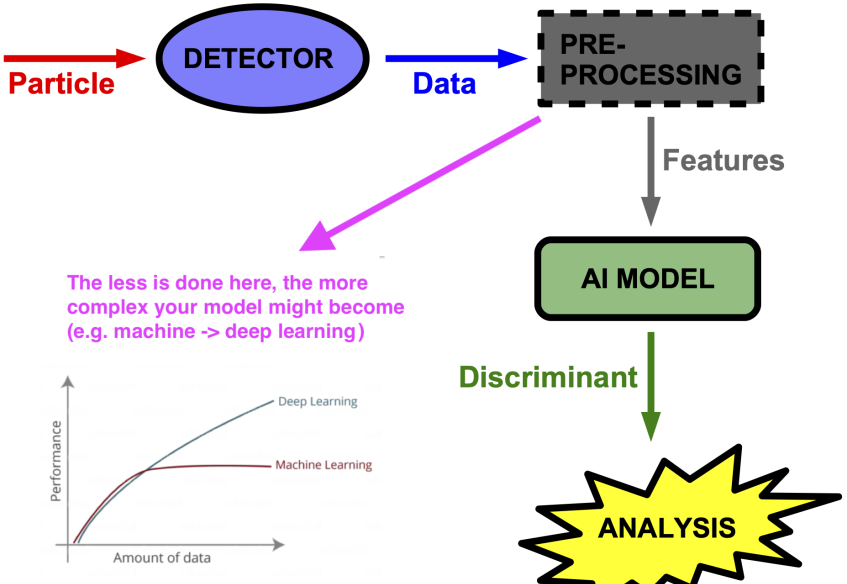
Particle Identification - PID



Particle Identification - PID

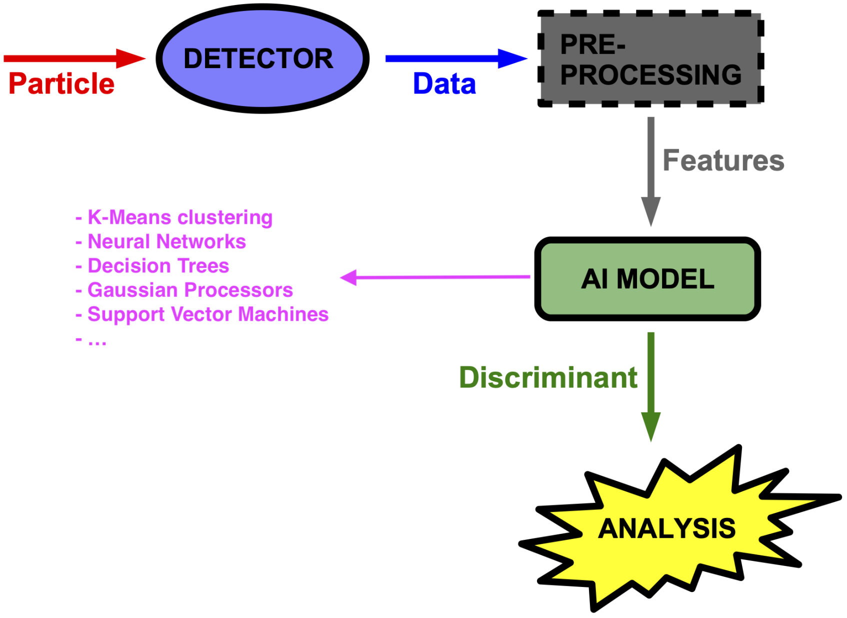


Particle Identification - PID

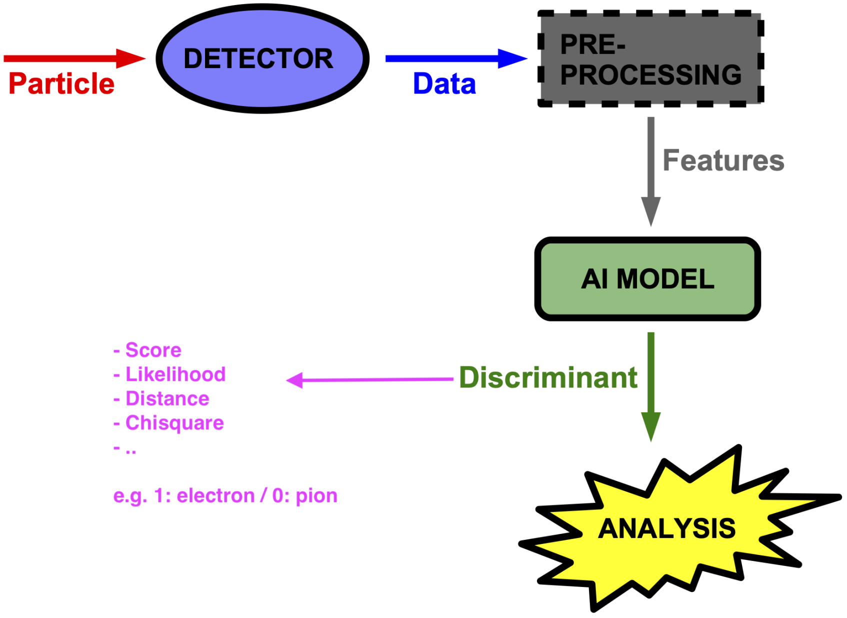


picture taken from deep learning for science
school (2019), talk by Mustafa Mustafa “Neural Networks I”

Particle Identification - PID



Particle Identification - PID



In a Nutshell

- Particle identification aims to solve a classification problem

In a Nutshell

- Particle identification aims to solve a classification problem
- The AI model (or classifier) simply represents a function f_{model}

$$f_{model} : \text{Features (information from detector)} \mapsto \text{Discriminant} \tag{1}$$

In a Nutshell

- Particle identification aims to solve a classification problem
- The AI model (or classifier) simply represents a function f_{model}

$$f_{model} : \text{Features (information from detector)} \mapsto \text{Discriminant} \tag{1}$$

- f_{model} is determined by:
  - Choice of model / algorithm
  - Provided data
  - Training (conditions)

In a Nutshell

- Particle identification aims to solve a classification problem
- The AI model (or classifier) simply represents a function f_{model}

$$f_{model} : \text{Features (information from detector)} \mapsto \text{Discriminant} \tag{1}$$

- f_{model} is determined by:
  - Choice of model / algorithm
  - Provided data
  - Training (conditions)
- Need to judge quality of f_{model}
  - How well does f_{model} solve the underlying classification problem?
  - Can f_{model} be applied on data sets other than the training data?

In a Nutshell

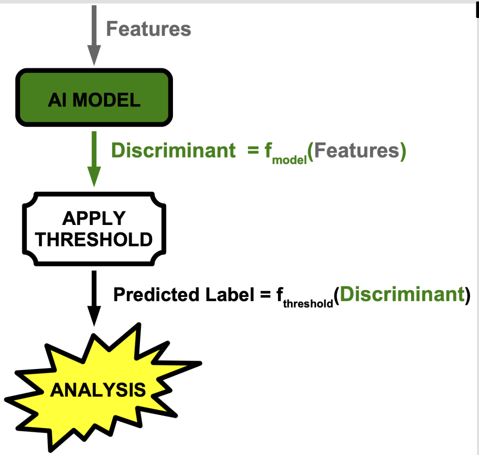
- Particle identification aims to solve a classification problem
- The AI model (or classifier) simply represents a function f_{model}

$$f_{model} : \text{Features (information from detector)} \mapsto \text{Discriminant} \tag{1}$$

- f_{model} is determined by:
  - Choice of model / algorithm
  - Provided data
  - Training (conditions)
- Need to judge quality of f_{model}
  - How well does f_{model} solve the underlying classification problem?
  - Can f_{model} be applied on data sets other than the training data?

$\Rightarrow$ Need performance metrics to address these questions properly

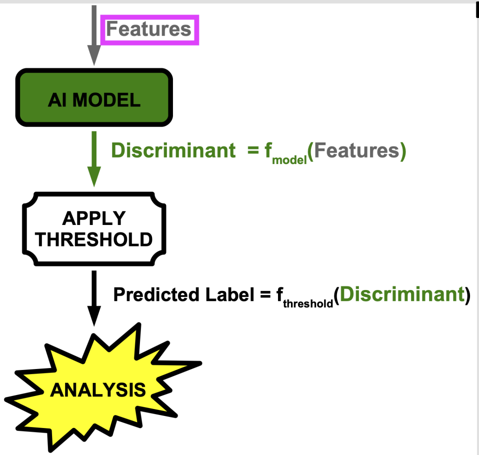
Binary Classification



Example:

- One event with 2 possible particle types (e.g. 1 and 2)

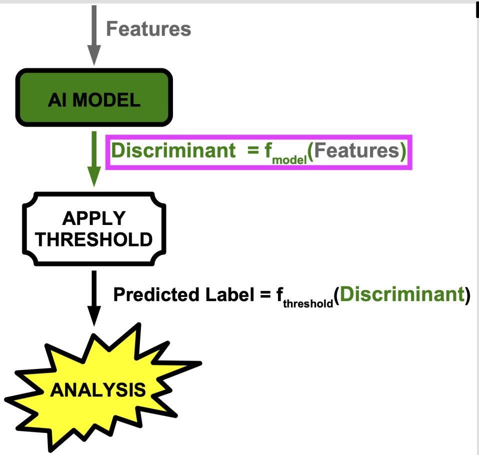
Binary Classification



Example:

- One event with 2 possible particle types (e.g. 1 and 2)
- Event is characterized by an n-dim feature vector $\vec{v}_{feat}$

Binary Classification

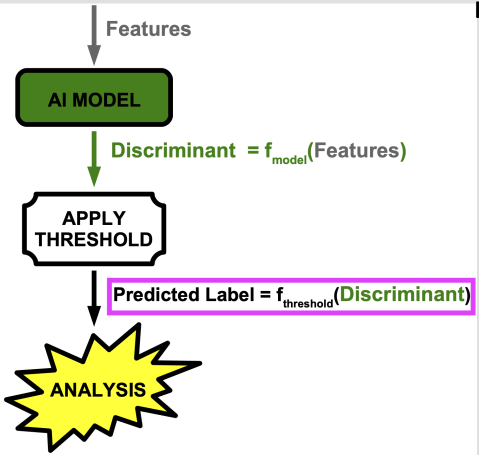


Example:

- One event with 2 possible particle types (e.g. 1 and 2)
- Event is characterized by an *n*-dim feature vector $\vec{v}_{feat}$
- Discriminant D:

  $D = f_{model}(\vec{v}_{feat}) \in \mathbb{R}$

Binary Classification



Example:

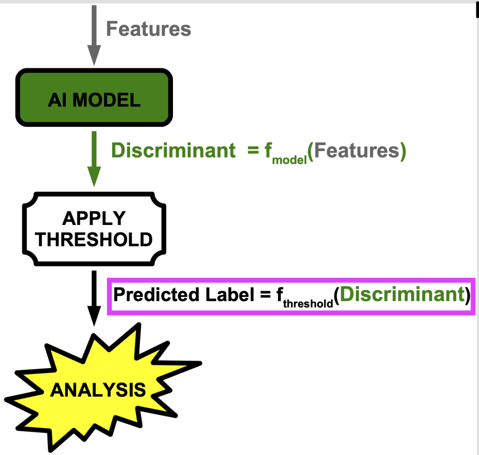
- One event with 2 possible particle types (e.g. 1 and 2)
- Event is characterized by an *n*-dim feature vector $\vec{v}_{feat}$
- Discriminant D:

  $$D = f_{model}(\vec{v}_{feat}) \in \mathbb{R}$$

- Threshold function:

  $$f_{threshold}(D, t) = \begin{cases} 1, & \text{if } D \geq t, \\ 2 & \text{else} \end{cases}$$

Binary Classification



Example:

- One event with 2 possible particle types (e.g. 1 and 2)
- Event is characterized by an *n*-dim feature vector $\vec{v}_{feat}$
- Discriminant D:

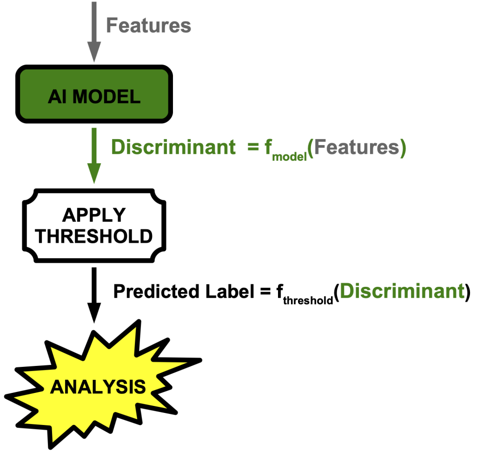
  $$D = f_{model}(\vec{v}_{feat}) \in \mathbb{R}$$

- Threshold function:

  $$f_{threshold}(D, t) = \begin{cases} 1, & \text{if } D \geq t, \\ 2 & \text{else} \end{cases}$$

- We find: $f_{threshold}(D, 0.5) = 1$
  $\Rightarrow$ The event is labeled as particle type 1

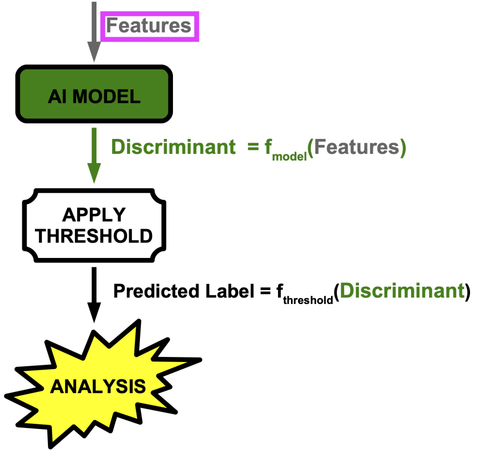
Multiclass Classification



Example:

- One event with *m* possible particle types (e.g. 1, 2, 3,..., m)

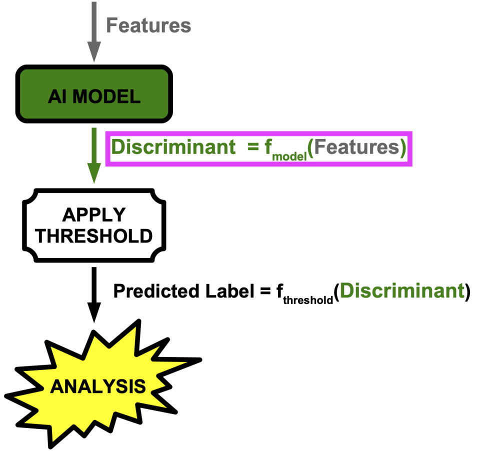
Multiclass Classification



Example:

- One event with m possible particle types (e.g. 1, 2, 3,..., m)
- Event is characterized by an n-dim feature vector $\vec{v}_{feat}$

Multiclass Classification

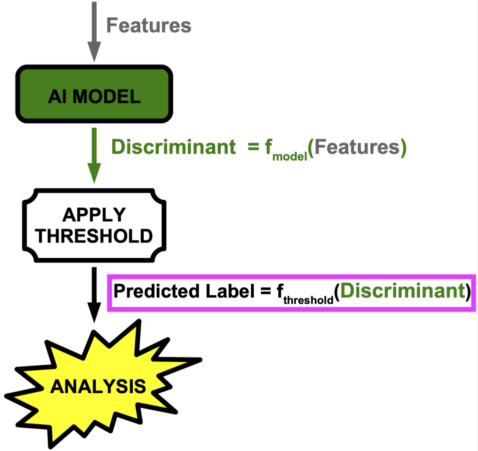


Example:

- One event with m possible particle types (e.g. 1, 2, 3,..., m)
- Event is characterized by an n-dim feature vector $\vec{v}_{feat}$
- Discriminant $\vec{D}$:

$$\begin{pmatrix} D_1 \\ \vdots \\ D_m \end{pmatrix} = \vec{D} = f_{model}(\vec{v}_{feat}) \in \mathbb{R}^m$$

Multiclass Classification



Example:

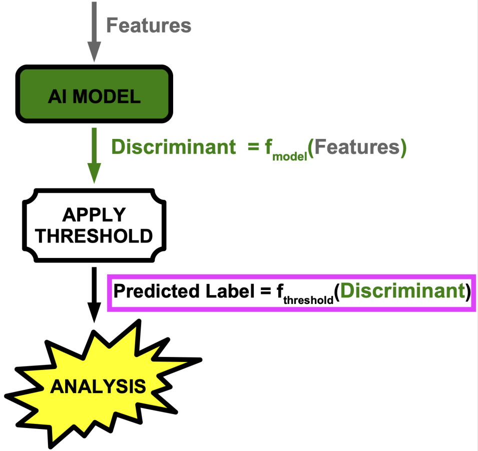
- One event with m possible particle types (e.g. 1, 2, 3,..., m)
- Event is characterized by an n-dim feature vector $\vec{v}_{feat}$
- Discriminant $\vec{D}$:

$$\begin{pmatrix} D_1 \\ \vdots \\ D_m \end{pmatrix} = \vec{D} = f_{model}(\vec{v}_{feat}) \in \mathbb{R}^m$$

- Threshold function:

$$f_{threshold}(\vec{D}) \equiv i \text{ for } D_i = max[\vec{D}]$$

Multiclass Classification



Example:

- One event with m possible particle types (e.g. 1, 2, 3,..., m)
- Event is characterized by an n-dim feature vector $\vec{v}_{feat}$
- Discriminant $\vec{D}$:

$$\begin{pmatrix} D_1 \\ \vdots \\ D_m \end{pmatrix} = \vec{D} = f_{model}(\vec{v}_{feat}) \in \mathbb{R}^m$$

- Threshold function:

$$f_{threshold}(\vec{D}) \equiv i \text{ for } D_i = max[\vec{D}]$$

- We find: $f_{threshold}(\vec{D}) = 2$
  $\Rightarrow$ The event is labeled as particle type 2

Threshold Functions

- Different threshold functions available $\leftrightarrow$ Binary / Multiclass classification ?
- Shown below are three examples of possible threshold functions:

i) $f_{threshold}(\vec{D}) \equiv i$ for $D_i = max[\vec{D}]$

ii) $f_{threshold}(\vec{D}, t) \equiv i$ for $D_i = max[\vec{D} - t \cdot \mathbf{1}]$

iii) $f_{threshold}(\vec{D}, t) \equiv i$ for $D_i = max[\frac{1}{t} \cdot \vec{D}]$

Example Analysis

- Throughout this lecture (and the hands-on session) we will look at a toy data set:

#Events	#Species	#Features	Labeled ?
$\sim 257\,\mathrm{k}$	3	6	yes

Example Analysis

- Throughout this lecture (and the hands-on session) we will look at a toy data set:

#Events	#Species	#Features	Labeled ?
$\sim 257\,\mathrm{k}$	3	6	yes

- Want to understand / explain PID analysis steps, with the help of this toy data

Example Analysis

- Throughout this lecture (and the hands-on session) we will look at a toy data set:

#Events	#Species	#Features	Labeled ?
$\sim 257\,\mathrm{k}$	3	6	yes

- Want to understand / explain PID analysis steps, with the help of this toy data
- **Goal:** Classify events in the data, using the provided features

Example Analysis

- Throughout this lecture (and the hands-on session) we will look at a toy data set:

#Events	#Species	#Features	Labeled ?
$\sim 257\,\mathrm{k}$	3	6	yes

- Want to understand / explain PID analysis steps, with the help of this toy data
- **Goal:** Classify events in the data, using the provided features
- **Approach:** Use scikit machine learning algorithm(s)

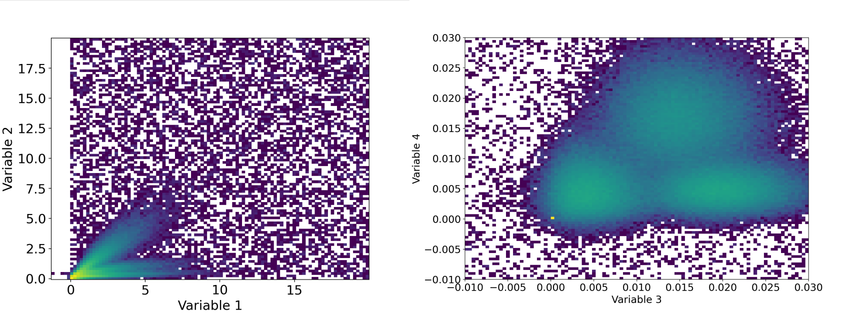
Example Analysis

- Throughout this lecture (and the hands-on session) we will look at a toy data set:

#Events	#Species	#Features	Labeled ?
$\sim 257\,\mathrm{k}$	3	6	yes

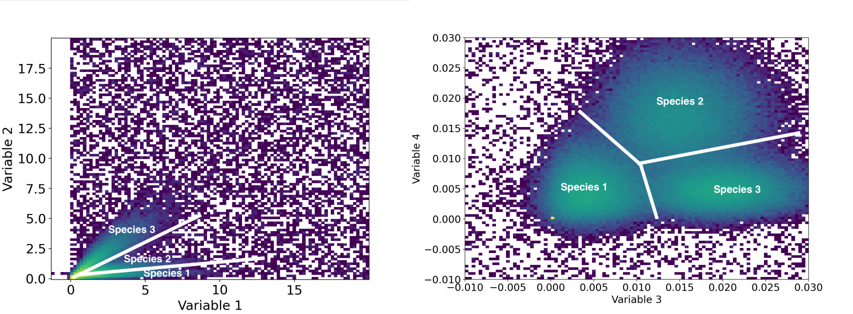
- Want to understand / explain PID analysis steps, with the help of this toy data
- **Goal:** Classify events in the data, using the provided features
- **Approach:** Use scikit machine learning algorithm(s)
- **Issue:** Evaluate performance of the algorithm(s) properly

Example Analysis: The Data Set I



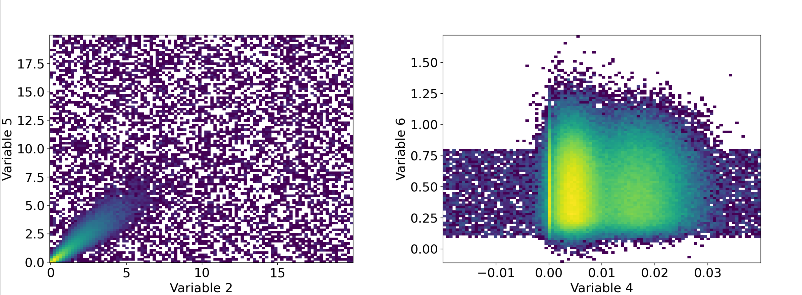
- <u>This is the first thing you should do:</u> Look at your input features!
- Variables show different correlations, depending on the species $\rightarrow$ Ideal for PID
- Variables show different ranges

Example Analysis: The Data Set I



- This is the first thing you should do: Look at your input features!
- Variables show different correlations, depending on the species → Ideal for PID
- Variables show different ranges

Example Analysis: The Data Set II



- This is the first thing you should do: Look at your input features!
- Variables show different correlations, depending on the species → Ideal for PID
- Variables show different ranges
- Variable 5 $\sim$ Variable 2
- Variable 6 is just a flat random distribution

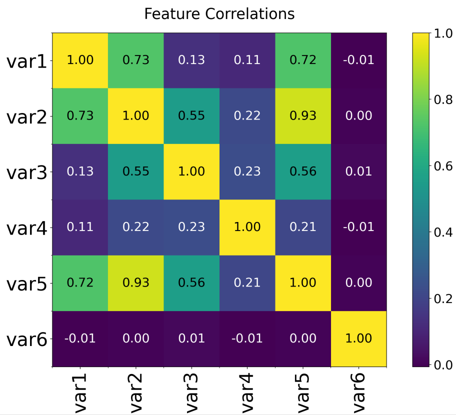
Example Analysis: The Data Set III

- This data set is labeled:

Species	Label
1	0
2	1
3	2

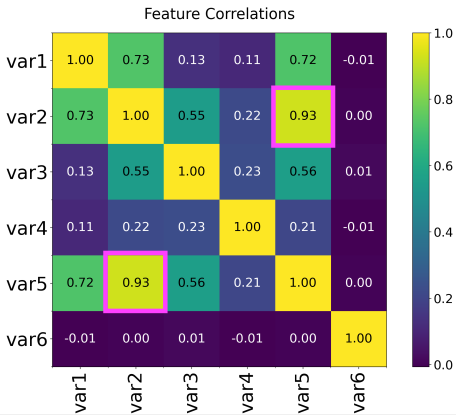
- Labeled data allows to perform supervised training
- But this data set is designed such that one might perform unsupervised learning as well (e.g. clustering)

Example Analysis: The Correlation Matrix



- Different methods to calculate feature correlations, e.g. Spearman vs. Pearson
- Off-diagonal elements...
  - ... close to one indicate redundancy → no information gain
  - ... close to zero indicate no correlation

Example Analysis: The Correlation Matrix

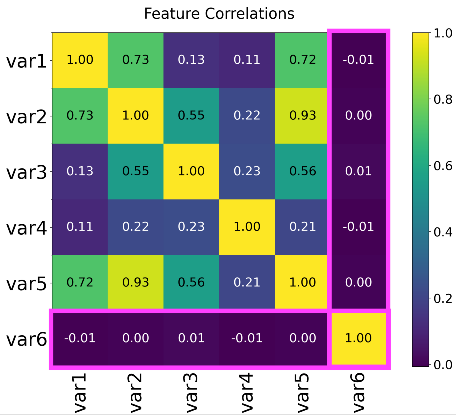


- Different methods to calculate feature correlations, e.g. Spearman vs. Pearson
- Off-diagonal elements...

  ... close to one indicate redundancy $\rightarrow$ no information gain

  ... close to zero indicate no correlation

Example Analysis: The Correlation Matrix



- Different methods to calculate feature correlations, e.g. Spearman vs. Pearson
- Off-diagonal elements...
  - ... close to one indicate redundancy → no information gain
  - ... close to zero indicate no correlation

Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data → supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data $\rightarrow$ supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

- Used 25% of entire data for validation (explained later in detail)

Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data → supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

- Used 25% of entire data for validation (explained later in detail)
- Ignored variable 5 and 6

Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data → supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

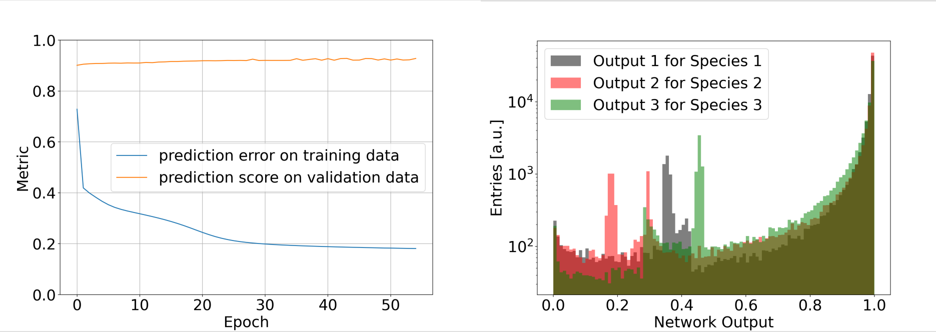
- Used 25% of entire data for validation (explained later in detail)
- Ignored variable 5 and 6
- **Pre-processing:** Normalized remaining input features between 0 and 1

Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data → supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

- Used 25% of entire data for validation (explained later in detail)
- Ignored variable 5 and 6
- **Pre-processing:** Normalized remaining input features between 0 and 1
- First things to check right after training: training curve(s) and output distributions

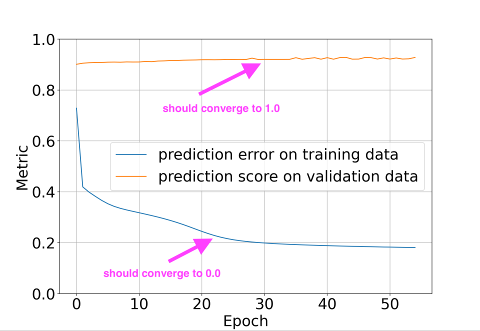


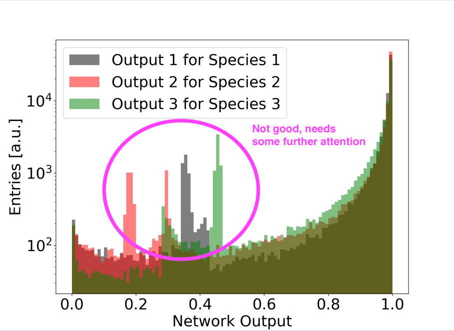
Example Analysis: Training a Multilayer Perceptron (MLP)

- Use labels in data → supervised training of MLP

#Inputs	#Hidden Layers	#Neurons in Hidden Layer	#Outputs
4	1	5	3

- Used 25% of entire data for validation (explained later in detail)
- Ignored variable 5 and 6
- **Pre-processing:** Normalized remaining input features between 0 and 1
- First things to check right after training: training curve(s) and output distributions





Example Analysis: Using the MLP

- Applied MLP on entire toy data:

#Events	Labeled as
$\sim 85\,\mathrm{k}$	1
$\sim 85\,\mathrm{k}$	2
$\sim 87\,\mathrm{k}$	3

$\rightarrow$ Is this good / bad?

- Need metrics to judge performance properly
- Our data is labeled $\rightarrow$ impact on metrics we can use

Labeled Data

- Events are tagged according to particle type (e.g. 1: e^-, 2: π^-, ...)
- Consequently, one knows:
  i) The abundance of each particle type in the entire data set (e.g. $10\,\mathrm{k}\ e^-$)
  ii) The relative abundance between the different particles (e.g. $N(e^-) = 0.1N(\pi^-)$)
- Most common training procedure used here is supervised training (one could perform unsupervised training of course)

True and False Positive Rate I

The building Blocks of Performance Evaluation

$$\text{True Positive Rate(i)} = \frac{\#\text{Events CORRECTLY identified as species i}}{\#\text{Events labeled as species i}} \tag{2}$$

$$\text{False Positive Rate(i)} = \frac{\#\text{Events FALSELY identified as species i}}{\#\text{Events NOT labeled as species i}} \tag{3}$$

True and False Positive Rate I

The building Blocks of Performance Evaluation

$$\text{True Positive Rate(i)} = \frac{\sum_{j=1}^{\#\text{Events}} \delta(\text{Predicted Label j} - i) \times \delta(\text{True Label j} - i)}{\sum_{j=1}^{\#\text{Events}} \delta(\text{True Label j} - i)} \qquad (2)$$

$$\text{False Positive Rate(i)} = \frac{\sum_{j=1}^{\#\text{Events}} \delta(\text{Predicted Label j} - i) \times [1.0 - \delta(\text{True Label j} - i)]}{\sum_{j=1}^{\#\text{Events}} [1.0 - \delta(\text{True Label j} - i)]} \qquad (3)$$

True and False Positive Rate II

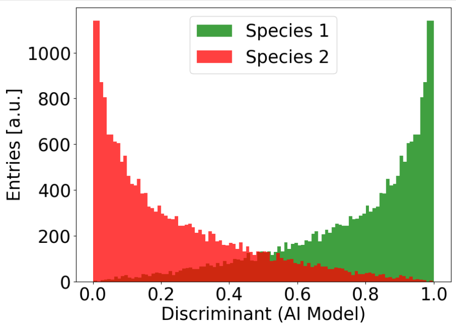
The building Blocks of Performance Evaluation

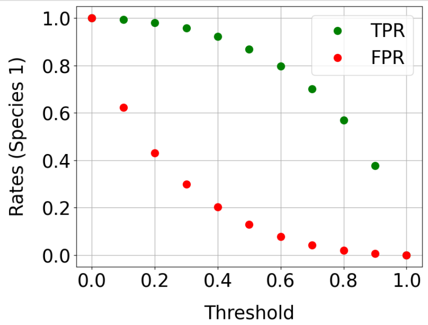
- Analogously, one can define the True Negative and False Negative Rate
- The True Positive Rate (TPR) and False Negative Rate (FNR) are...
  - ... universal, i.e. they do not[1] depend on relative abundances between the different particle types
  - ... characteristic for the used classifier
- The most important evaluation metrics are directly derived from the TPR and FPR

[1] Given enough statistics for each species and each feature distribution!

Receiving Operator Characteristics (ROC)

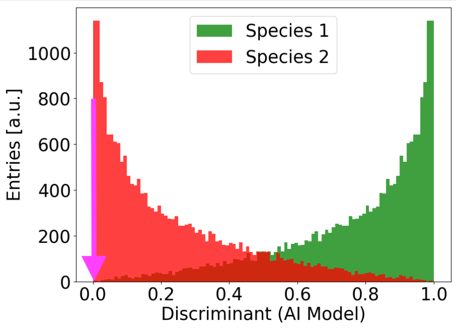
- Suppose a binary classification problem with two particle species (1 and 2)
- Trained AI Model to solve this problem
- **Basic Question:** What is the model actually doing?
- **Approach:** Perform a threshold scan

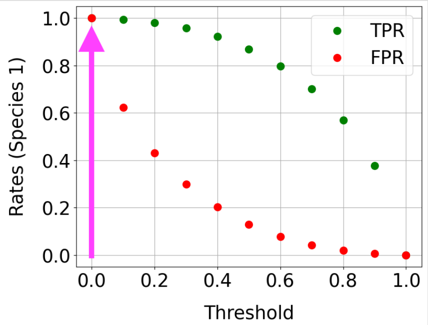




Receiving Operator Characteristics (ROC)

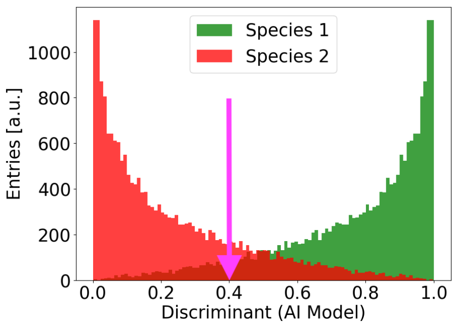
- Suppose a binary classification problem with two particle species (1 and 2)
- Trained AI Model to solve this problem
- **Basic Question:** What is the model actually doing?
- **Approach:** Perform a threshold scan

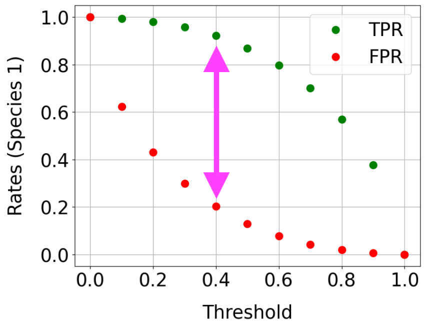




Receiving Operator Characteristics (ROC)

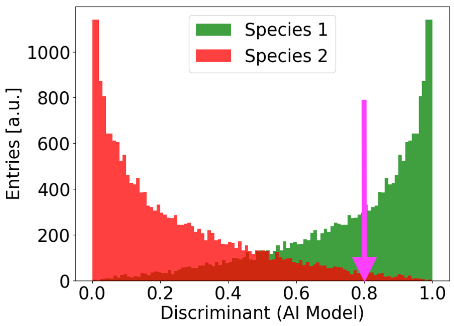
- Suppose a binary classification problem with two particle species (1 and 2)
- Trained AI Model to solve this problem
- **Basic Question:** What is the model actually doing?
- **Approach:** Perform a threshold scan

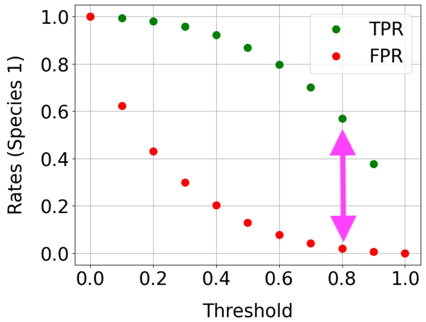




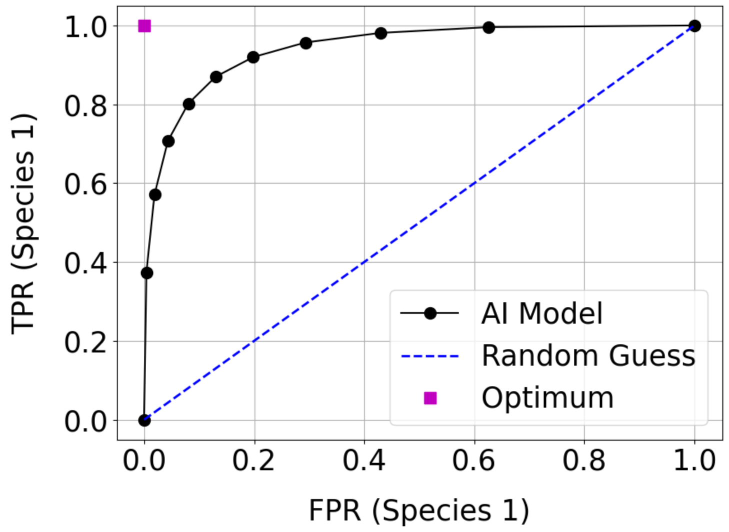
Receiving Operator Characteristics (ROC)

- Suppose a binary classification problem with two particle species (1 and 2)
- Trained AI Model to solve this problem
- **Basic Question:** What is the model actually doing?
- **Approach:** Perform a threshold scan

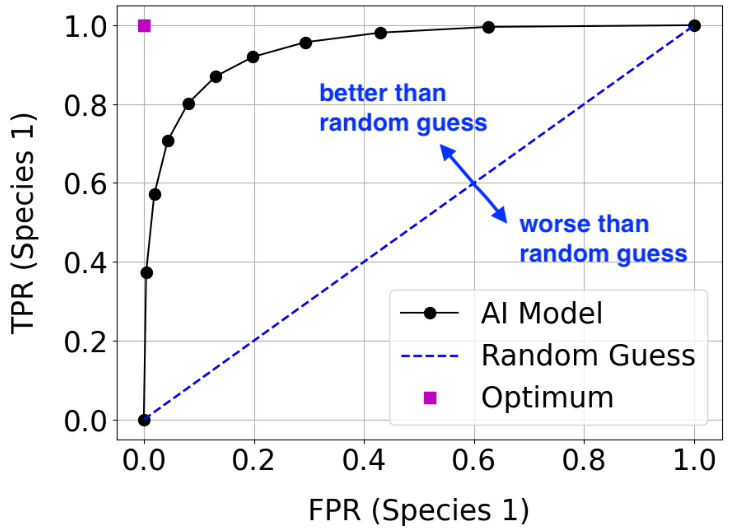




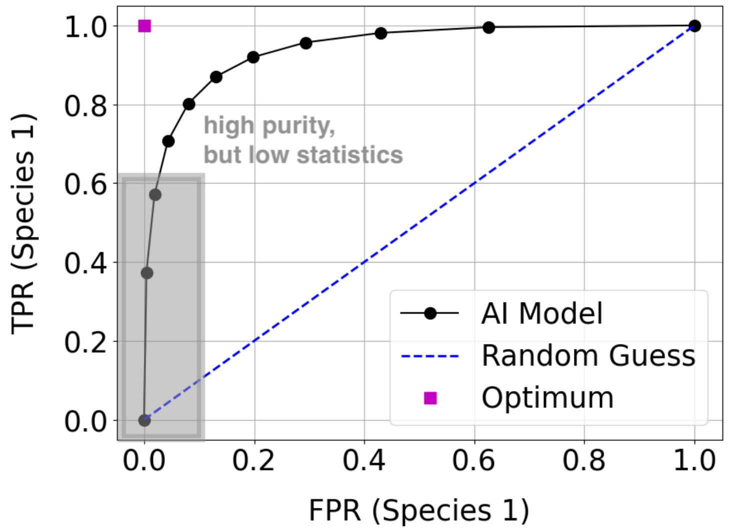
The ROC-Curve



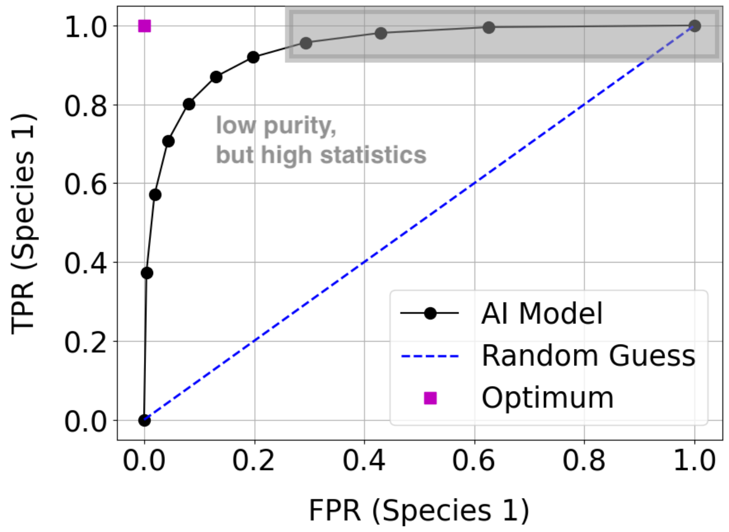
The ROC-Curve



The ROC-Curve

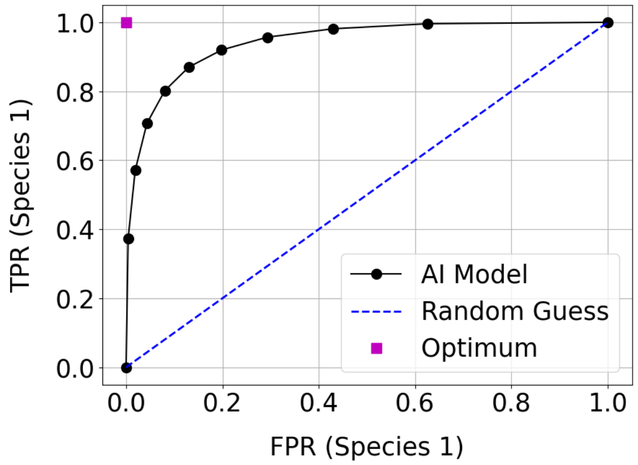


The ROC-Curve

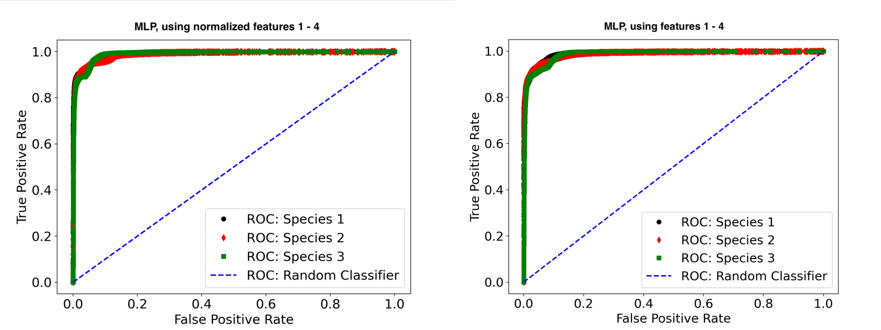


AUC - Area Under ROC

- Area under the ROC-Curve is another performance metric
- AUC $= 1.0 \leftrightarrow$ Optimal classifier
- AUC $= 0.0 \leftrightarrow$ Bad classifier
- Found here: AUC $= 0.94$

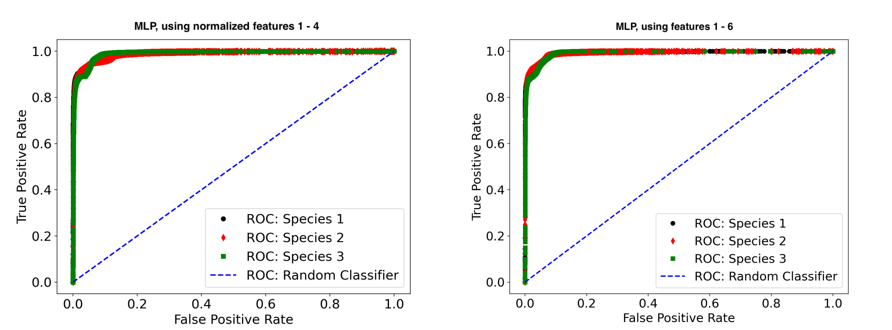


Comparing ROC-Curves for different Training Setups



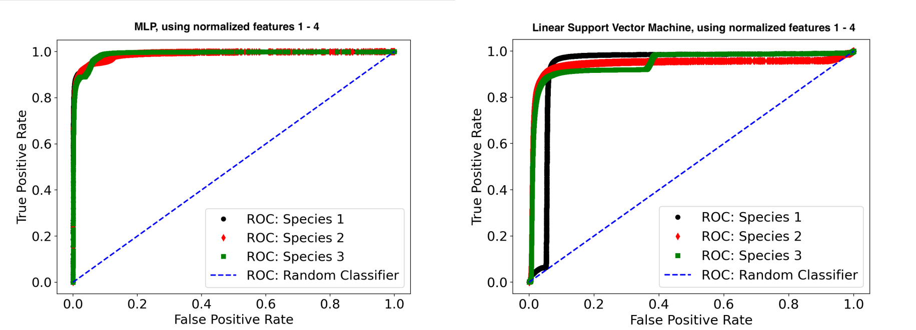
- Identify three particle species using differently trained MLP models
- ROC-curves allow to compare the classification performance between
  - i) Particle species
  - ii) Different models
- AUC for all curves shown here ~ 0.99

Comparing ROC-Curves for different Training Setups



- Identify three particle species using differently trained MLP models
- ROC-curves allow to compare the classification performance between
  - i) Particle species
  - ii) Different models
- AUC for all curves shown here ~ 0.99

Comparing ROC-Curves for different Classifier



- Identify three particle species using two different classification models
- ROC-curves allow to compare the classification performance between
  - i) Particle species
  - ii) Different models
- AUC(MLP) ~ 0.99 / AUC(lin. svm) ~ 0.93

The Confusion Matrix

- Right after the ROC, the second most important monitoring tool

The Confusion Matrix

- Right after the ROC, the second most important monitoring tool
- Nearly all performance measures (accuracy, F1 score, purity, mcc, efficiency,...) are directly derived from this matrix

The Confusion Matrix

- Right after the ROC, the second most important monitoring tool
- Nearly all performance measures (accuracy, F1 score, purity, mcc, efficiency,...) are directly derived from this matrix
- The elements in the confusion matrix $\hat{C}$ are defined[2] as:

$$c_{ij}(t) \equiv \sum_{k=0}^{N-1} \delta(L_{true,k} - \ell_i) \times \delta(L_{pred,k}(t) - \ell_j) \tag{4}$$

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{else} \end{cases} \tag{5}$$

- With:

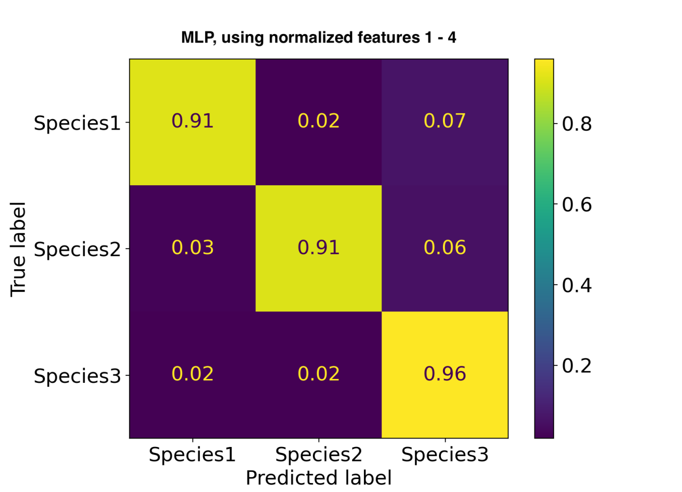
$L_{true,k}$	true label of event k
$L_{pred,k}(t)$	predicted label of event k, using the threshold t
ℓ_i	the label corresponding to species i

- **NOTE:** The definition of the above equation depends on which axis holds the true / predicted label

[2] Directly derived from TPR and FPR

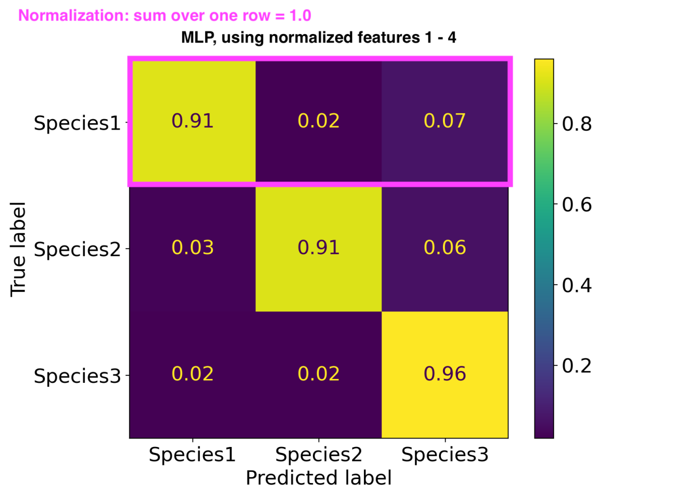
Confusion Matrix from the Example Analysis

- Shown below is the confusion matrix for a neural network classifying three particle species (see previous slides)
- **Ideally:** All diagonal elements should be one and all off-diagonal elements should be zero



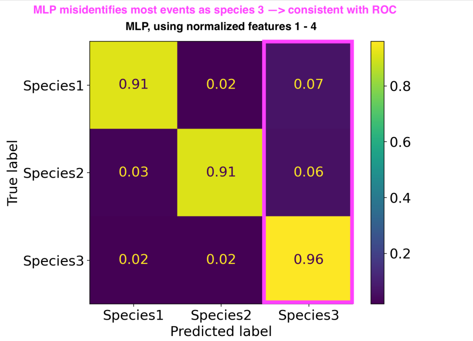
Confusion Matrix from the Example Analysis

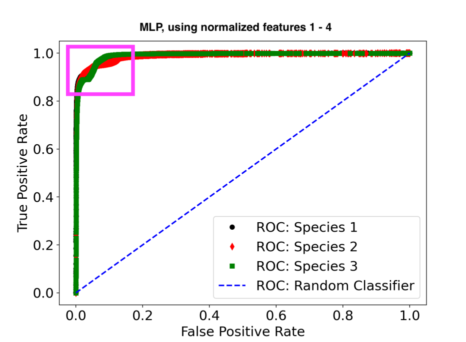
- Shown below is the confusion matrix for a neural network classifying three particle species (see previous slides)
- **Ideally:** All diagonal elements should be one and all off-diagonal elements should be zero



Confusion Matrix from the Example Analysis

- Shown below is the confusion matrix for a neural network classifying three particle species (see previous slides)
- **Ideally:** All diagonal elements should be one and all off-diagonal elements should be zero





⇒ Always check for consistency between different metrics!

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$\text{Accuracy} \equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) \tag{6}$$

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$\begin{aligned} \text{Accuracy} &\equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) && (6) \\ &= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} && (7) \end{aligned}$$

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$\text{Accuracy} \equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) \tag{6}$$

$$= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} \tag{7}$$

$$= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} TPR(i) \cdot \#\text{Events with species i} \tag{8}$$

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$
\begin{aligned}
\text{Accuracy} &\equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) && (6)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} && (7)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} TPR(i) \cdot \#\text{Events with species i} && (8)\\
&= \sum_{i}^{\#\text{Species}} TPR(i) \cdot R(i) && (9)
\end{aligned}
$$

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$
\begin{aligned}
\text{Accuracy} &\equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) && (6)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} && (7)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} TPR(i) \cdot \#\text{Events with species i} && (8)\\
&= \sum_{i}^{\#\text{Species}} TPR(i) \cdot R(i) && (9)
\end{aligned}
$$

- Where $R(i)$ denotes the abundance ratio of species i (e.g. 20% protons)

The Accuracy

- The accuracy can be calculated from the weighted trace of the confusion matrix

$$\begin{aligned}
\text{Accuracy} &\equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) && (6)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} && (7)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} TPR(i) \cdot \#\text{Events with species i} && (8)\\
&= \sum_{i}^{\#\text{Species}} TPR(i) \cdot R(i) && (9)
\end{aligned}$$

- Where $R(i)$ denotes the abundance ratio of species i (e.g. 20% protons)
- The accuracy varies between **0: bad performance** and **1: ideal performance**

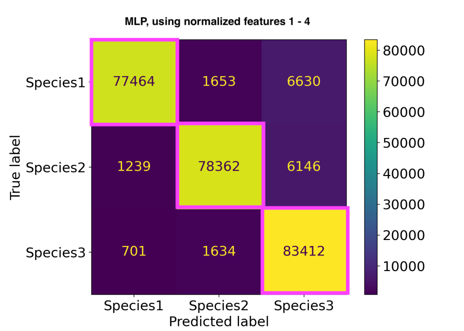
The Accuracy

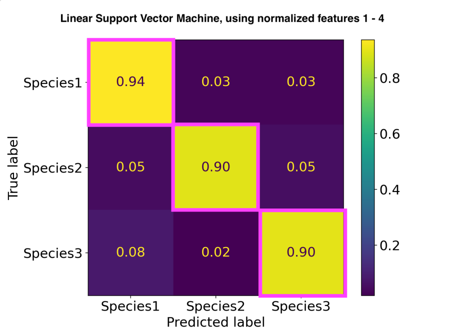
- The accuracy can be calculated from the weighted trace of the confusion matrix

$$
\begin{aligned}
\text{Accuracy} &\equiv \frac{1}{\#\text{Events}} Tr(\hat{C}) && (6)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} c_{ii} && (7)\\
&= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} TPR(i) \cdot \#\text{Events with species i} && (8)\\
&= \sum_{i}^{\#\text{Species}} TPR(i) \cdot R(i) && (9)
\end{aligned}
$$

- Where $R(i)$ denotes the abundance ratio of species i (e.g. 20% protons)
- The accuracy varies between **0: bad performance** and **1: ideal performance**
- Balanced Accuracy $= \frac{1}{\#\text{Species}} \cdot \sum_{i}^{\#\text{Species}} TPR(i)$

Accuracy from the Example Analysis





- $257\,\mathrm{k}$ events with three species: $R(1) = R(2) = R(3) = \frac{1}{3}$
- Using the formulas from the previous slides yield:

  Accuracy (MLP) $= (77,464 + 78,362 + 83,412)/257\,\mathrm{k} \approx 93\%$

  Accuracy (LSVM) $= 0.333{\cdot}0.94 + 0.333{\cdot}0.90 + 0.333{\cdot}0.90 \approx 0.91\%$

$\Rightarrow$ Obtain same values when using the accuracy function from scikit

The Precision and F1-Score

- The precision[3] can be thought of as 'cleanliness' of the classified data set

$$\text{Precision for species i} \equiv \frac{\#\text{Events CORRECTLY identified as species i}}{\#\text{Events identified as species i}} \tag{10}$$

[3] Sometimes also referred to as purity

The Precision and F1-Score

- The precision[3] can be thought of as 'cleanliness' of the classified data set

$$\text{Precision for species i} \equiv \frac{\text{TPR(i)}}{\text{TPR(i)} + \frac{1-R(i)}{R(i)} \times \text{FPR(i)}} \tag{10}$$

[3]Sometimes also referred to as purity

The Precision and F1-Score

- The precision[3] can be thought of as 'cleanliness' of the classified data set

$$\text{Precision for species i} \equiv \frac{\text{TPR(i)}}{\text{TPR(i)} + \frac{1-R(i)}{R(i)} \times \text{FPR(i)}} \tag{10}$$

- The F1-Score is deduced from F-measure and folds the TPR together with the purity

$$\text{F1-Score for species i} \equiv 2 \cdot \frac{\text{TPR(i)} \cdot \text{Precision(i)}}{\text{TPR(i)} + \text{Precision(i)}} \tag{11}$$

- Like the accuracy, these metrics also depend on the relative abundance $R(i)$

[3] Sometimes also referred to as purity

The Precision and F1-Score

- The precision[3] can be thought of as 'cleanliness' of the classified data set

$$\text{Precision for species i} \equiv \frac{\text{TPR(i)}}{\text{TPR(i)} + \frac{1-R(i)}{R(i)} \times \text{FPR(i)}} \tag{10}$$

- The F1-Score is deduced from F-measure and folds the TPR together with the purity

$$\text{F1-Score for species i} \equiv 2 \cdot \frac{\text{TPR(i)} \cdot \text{Precision(i)}}{\text{TPR(i)} + \text{Precision(i)}} \tag{11}$$

- Like the accuracy, these metrics also depend on the relative abundance $R(i)$
- Both, precision and F1 show values between **0: bad performance** and **1: ideal performance**

[3]Sometimes also referred to as purity

The Matthews Correlation Coefficient (MCC)

- Like the accuracy, the MCC can be computed from entries within the (unnormalized) confusion matrix:[4]

$$\mathrm{MCC} = \frac{\sum_k \sum_l \sum_m (C_{kk} C_{lm} - C_{kl} C_{mk})}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k' \neq k} \sum_{l'} C_{k'l'})} \cdot \sqrt{\sum_k (\sum_l C_{lk})(\sum_{k' \neq k} \sum_{l'} C_{k'l'})}} \tag{12}$$

- In a very simplified picture, the MCC combines the true positive rate, false positive rate and purity[5]

$$\text{MCC (Binary Classification)} = \frac{\mathrm{TPR(i)} \cdot \mathrm{TPR(j)} - \mathrm{FPR(i)} \cdot \mathrm{FPR(j)}}{\sqrt{\frac{\mathrm{TPR(i)}}{\mathrm{Precision(i)}} \cdot \frac{\mathrm{TPR(j)}}{\mathrm{Precision(j)}}}} \tag{13}$$

- The MCC is a common / preferable choice for imbalanced data
- Unlike the previously introduced metrics, the MMC might vary[6] between **-1: bad performance** and **1: ideal performance**

[4] Formula taken from **wikipedia**

[5] This is exact true for binary classification

[6] Different for binary or multiclass classification.

Comparing Metrics for the Example Analysis

- Compare performance metrics of differently trained neural networks and the linear support vector machine on the given data set

Model	Precision (averaged)	F1-Score (averaged)	Accuracy	MCC
MLP(1)	0.93	0.93	0.93	0.89
MLP(2)	0.92	0.92	0.92	0.88
MLP(3)	0.93	0.93	0.93	0.89
LSVM	0.91	0.91	0.91	0.87

MLP(1): Use all features for training

MLP(2): Use only features 1-4 for training

MLP(3): Use only normalized features 1-4 for training

- Different versions of MLP show similar performance and are somewhat better than the LSVM model

Balanced vs. Imbalanced Data

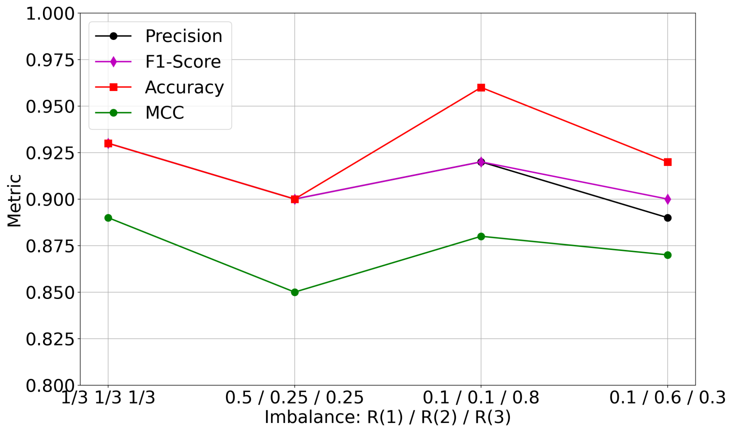
- Balanced data: $R(i) = \frac{1}{\#\text{Species}}, \forall i$
- Imbalanced data: $\exists i : R(i) \neq \frac{1}{\#\text{Species}}$
- TPR and FPR (and therefore the ROC-Curve) ideally[7] do not depend on balance in data
- **BUT:** Accuracy, Purity, MCC, F1-Score do[8] $\Rightarrow$ Take into consideration when evaluating your model(s) on different data sets

[7] Given sufficient statistics of course for each label and distribution

[8] By definition, because R is folded in.

Imbalanced Toy Data

- Generated toy data with imbalance between species
- Applied MLP (trained on balanced data!) on different toy sets



Which Metric to use?

- As usual: It depends on...

... what you intend to find out about your classifier

  - Are you interested in global performance? (e.g. accuracy)
  - Do you need to know the performance for a certain species only? (e.g. precision)

... imbalance in your data

... available statistics $\rightarrow$ e.g. ROC-Curve simply not available

- But in general: It helps to compare different metrics
- Do not trust single numbers only $\rightarrow$ also look at covariance matrix (your best friend!) and the ROC-curve (if possible)

Should I train on imbalanced Data?

- Again, it depends on...

... what you want to do $\rightarrow$ Do you want to analyze "real" data with a known imbalance $\rightarrow$ train your classifier appropriately

... the training data you have $\rightarrow$ might be highly imbalanced and you have no other data

... the resources you have (time, computing power, etc.)

- Best option (if resources available):

$\Rightarrow$ Train on different data sets and compare performances $\leftrightarrow$ Systematic X-check

- Sufficient (most of the time):

$\Rightarrow$ Train on balanced data $\rightarrow$ Let model pick up all feature distributions equally and check if model generalizes well enough on imbalanced sets

Summary: Metrics for labeled Data

- TPR and FPR are the building blocks for evaluating a classification algorithm
- Introduced a few (but not all) metrics
  - ROC-Curve
  - AUC
  - Accuracy
  - Precision
  - F1-Score
  - MCC
- There are many more
- Think about which information you need for a proper evaluation → Choose metric accordingly

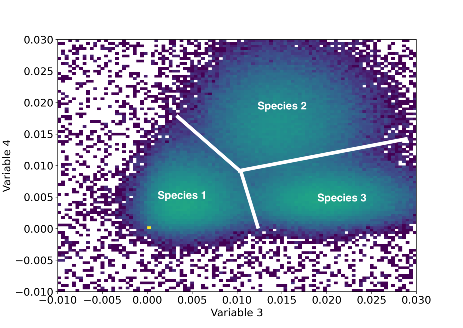
Unlabeled Data

- Events are not labeled, i.e. the particle type is a priori not known

⇒ The metrics introduced earlier are not directly applicable

- **However:** One might have some measures to roughly define a particle species (e.g. energy deposits in a detector)
- If the training data is unlabeled as well:
  - Perform unsupervised training (e.g. clustering algorithms)
  - Label data by yourself, e.g. autoencoder neural networks

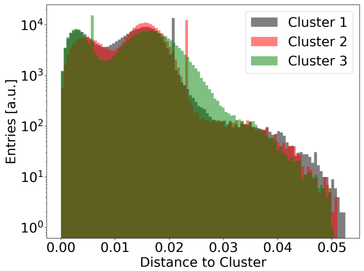
Example Analysis with unlabeled Toy Data

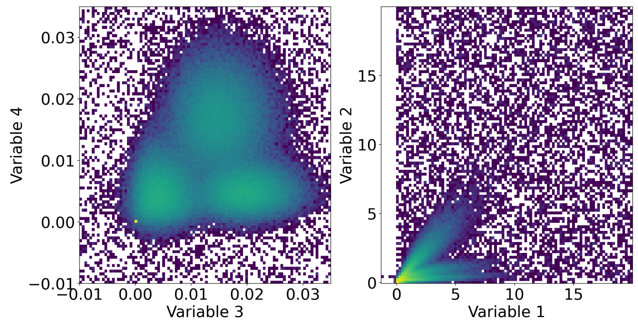


- Suppose that our (balanced) toy data has no labels
  - No information which event corresponds to which species
  - Do not know the abundance of each individual species
- The correlation between variable 3 and 4 suggests that we might perform a cluster analysis $\rightarrow$ unsupervised learning

Example Analysis: kMeans-Clustering

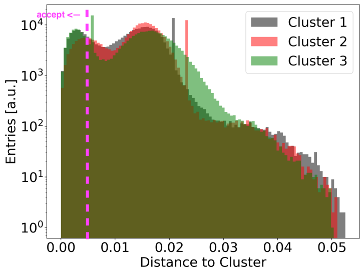
- Trained kMeans-algorithm with three cluster centers and 300 iterations
- Used variables 3 and 4 only
- Compute distance to each cluster
  $\rightarrow$ Our discriminant

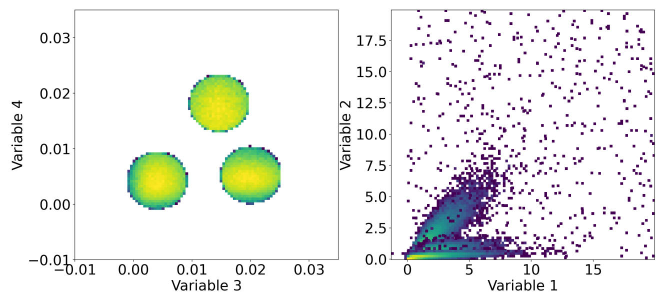




Example Analysis: kMeans-Clustering

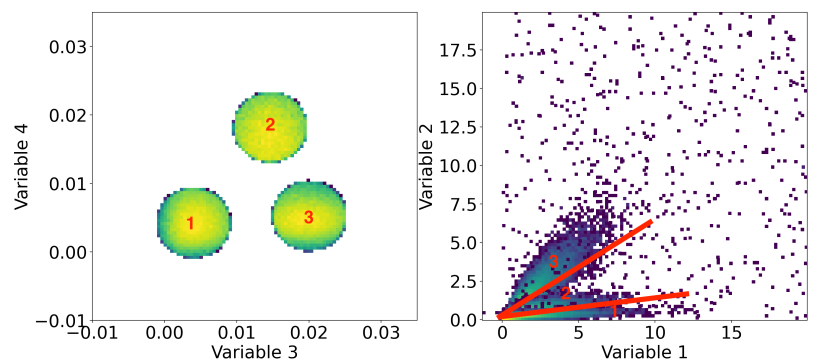
- Trained kMeans-algorithm with three cluster centers and 300 iterations
- Used variables 3 and 4 only
- Compute distance to each cluster
  → Our discriminant





ALWAYS check the input features after classification

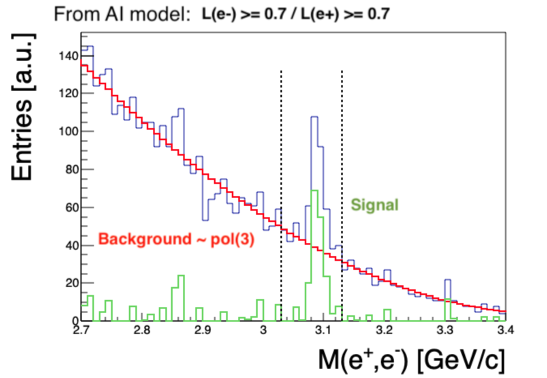
Using Yields



- Could use correlations between variable 2 and 1 for further analysis $\leftrightarrow$ They have not been used for training
- Red lines in top right panel indicate hypothetical selection criteria to extract yields for each cluster / blob
- Define metrics based on these yields

Using Yields: Example from GlueX PID

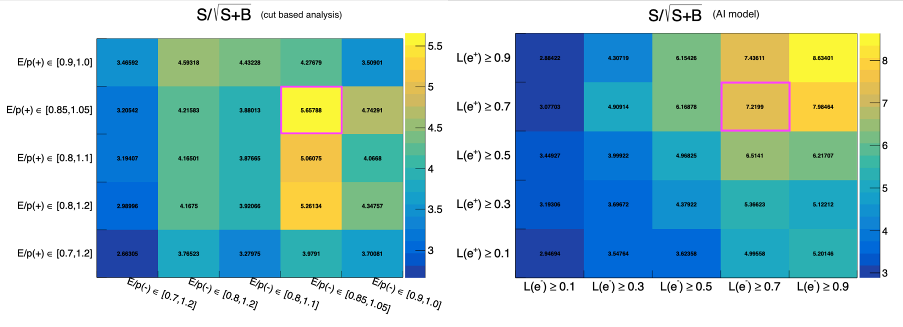
- **Goal:** Identify leptons in GlueX $\gamma p \rightarrow e^+e^-p$ data (measured $\rightarrow$ no labels)
- **Approaches:** AI model and cut based analysis
- Compare approaches by looking at dilepton mass[9] and determine signal (S) and background (B) contributions
- **Calculate FOM (Figure Of Merit)**: $S/\sqrt{S+B}$



[9] NOT part of the model input features

Using Yields: Example from GlueX PID

- **Goal:** Identify leptons in GlueX $\gamma p \to e^+e^-p$ data (measured $\to$ no labels)
- **Approaches:** AI model and cut based analysis
- Compare approaches by looking at dilepton mass[9] and determine signal (S) and background (B) contributions
- **Calculate FOM (Figure Of Merit):** $\boldsymbol{S/\sqrt{S+B}}$



[9] NOT part of the model input features

Generalization and Stability I

- **Question:** How well does the trained model generalize $\rightarrow$ Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze

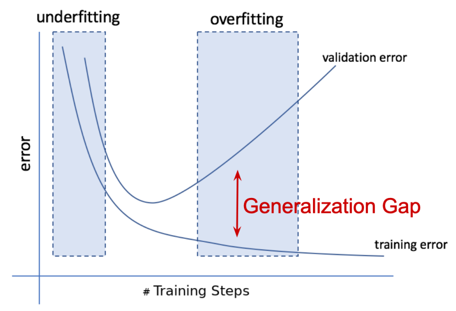
Generalization and Stability I

- **Question:** How well does the trained model generalize → Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Use validation data

Picture taken from Brenda Ngs introductory talk at the: deep learning for science school 2019

Generalization and Stability I

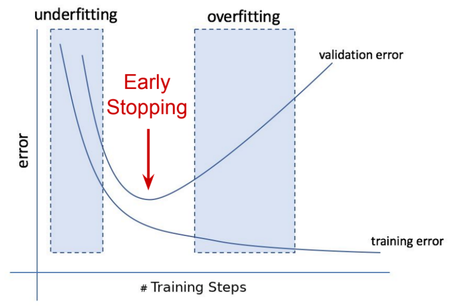
- **Question:** How well does the trained model generalize → Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Use validation data



Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

Generalization and Stability I

- **Question:** How well does the trained model generalize → Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Use validation data



Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

Generalization and Stability II

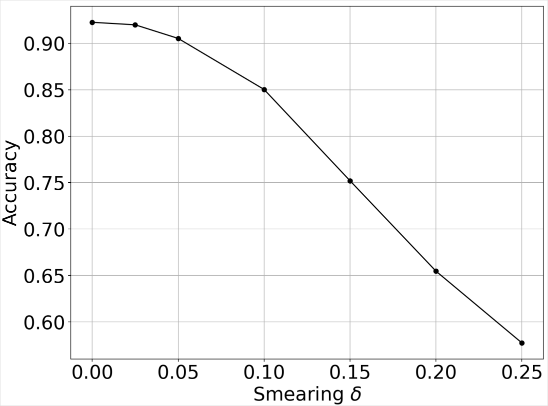
- **Question:** How well does the trained model generalize → Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze

Generalization and Stability II

- **Question:** How well does the trained model generalize $\rightarrow$ Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Apply smearing: features $\mapsto$ features $\times$ Gauss$(1, \delta)$ to training / validation data and monitor performance

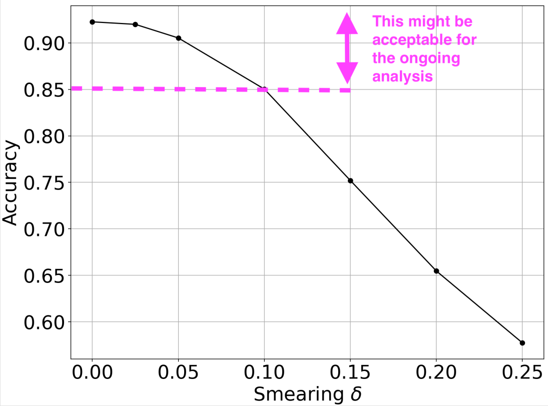
Generalization and Stability II

- **Question:** How well does the trained model generalize $\rightarrow$ Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Apply smearing: features $\mapsto$ features $\times$ Gauss$(1, \delta)$ to training / validation data and monitor performance
- Shown below: MLP accuracy on toy data for different δ



Generalization and Stability II

- **Question:** How well does the trained model generalize → Response on "unknown" data
- The model has been trained under certain conditions which might not be reflected by the data we want to analyze
- **Approach:** Apply smearing: features $\mapsto$ features $\times$ Gauss$(1, \delta)$ to training / validation data and monitor performance
- Shown below: MLP accuracy on toy data for different δ



Summary and Outlook (Part I)

- Introduced metrics to evaluate the performance of (any) classification algorithm
- Different metrics provide different information
- Choice of metrics depends on which question one tries to answer and the data set
  - Global vs. individual performance (for one species)
  - Labeled vs. unlabeled data
  - Balanced vs. imbalanced data
- Looked a distributed data only (no images), but the approaches shown here are applicable to any data set / classification problem
- **Always:**
  - Use and compare different metrics
  - Look at the classifier output distributions
  - Check features before / after classification
  - Have a critical view on your results $\leftrightarrow$ NEVER trust your classifier blindly
- Second part of this lecture: hands-on session

Part II: Hands-On

Picture taken from: `http://screenrant.com/things-you-did-not-know-about-wile-e-coyote/`

The (Toy) Data Set

- The data (.csv files) are stored at the FSU cluster:

  http://hadron.physics.fsu.edu/~dlersch/GlueX_PANDA_EIC_ML_Workshop/
- The naming scheme for the files is:

  hands_on_data_P1_P2_P3.csv

  where Pi refers to the relative abundance of species i
- **Example:** hands_on_data_02_07_01.csv

  $\rightarrow$ 20% of all particles in this data refer to species 1, 70% refer to species 2 and 10% refer to species 3

Scripts and Tools

- There are three options to join this hands-on

Option 1 (classic)

1. Go to: http://hadron.physics.fsu.edu/~dlersch/GlueX_PANDA_EIC_ML_Workshop/
2. Download python scripts from the folder: **Repl_Files**
3. Run everything on your local machine / cluster / ...

Option 2 (fancy)

1. Go to: http://hadron.physics.fsu.edu/~dlersch/GlueX_PANDA_EIC_ML_Workshop/
2. Download jupyter notebooks from the folder: **Notebooks**
3. Run everything on your local machine / Google collab / Binder / ....

Option 3 (easy) [Many thanks to Cristiano Fanelli for bringing this option up!]

1. Go to: http://repl.it/@daniel49/HandsOnSession
2. Click the **Fork** button
3. Follow instructions in **main.py**

- Options 1 and 2 require python ≥ 3.6 plus the corresponding libraries
- Option 3 requires internet only
- Material will be available for ~ 1 week