Introduction to Machine Learning: Part III

Prof. Sean Dobbs[1] & Daniel Lersch[2]

April 23, 2020

[1] (sdobbs@fsu.edu)
[2] (dlersch@jlab.org)

About this Lecture

- **Part I:**
  - Introduction to DataFrames
  - Basic concepts of machine learning

    (with focus on feedforward neural networks)
- **Part II:**
  - Machine learning in (physics) data analysis
  - Performance evaluation
- **Part III:**
  - Algorithm tuning
  - Hyper parameter optimization
- **Part IV:**
  - Custom neural networks with Tensorflow
  - Transition to Deep Learning

The individual contents might be subject to change

This Lecture will...

... NOT turn you into a machine learning specialist

This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

... utilize the **scikit-learn** library

This Lecture will...

... NOT turn you into a machine learning specialist

... NOT cover all aspects of machine learning

... give a (very) brief overview only (i.e. further reading is definitely required)

... introduce a few machine learning algorithms

... utilize the **scikit-learn** library

... most likely contain several errors ($\rightarrow$ Please send a mail to `dlersch@jlab.org`)

Homework and Literature

- Machine learning can be learned best by simply doing it!

Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework (most likely posted on Thursday) aims to perform a simple analysis and getting familiar with machine learning

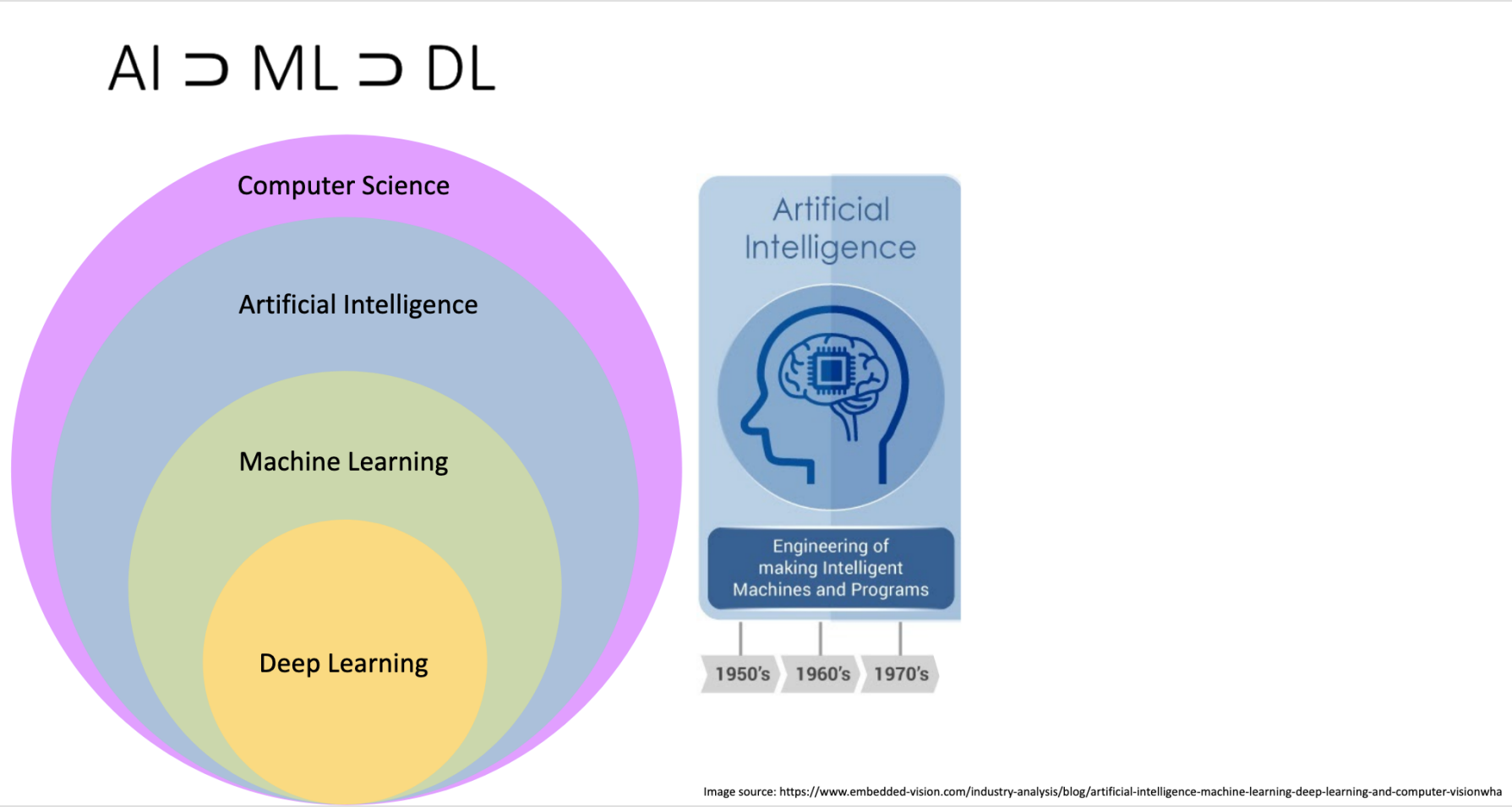
Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework (most likely posted on Thursday) aims to perform a simple analysis and getting familiar with machine learning
- Helpful literature:
  - The **scikit-learn** documentation
  - Talks from the **deep learning for science school 2019**[3]
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron
  - The internet is full of good (but also very bad!) literature[4] $\rightarrow$ browse with caution
  - The slides of the lecture are available at: `http://hadron.physics.fsu.edu/~dlersch/ml_slides/`

[3] Very good and detailed explanation of (deep) neural networks

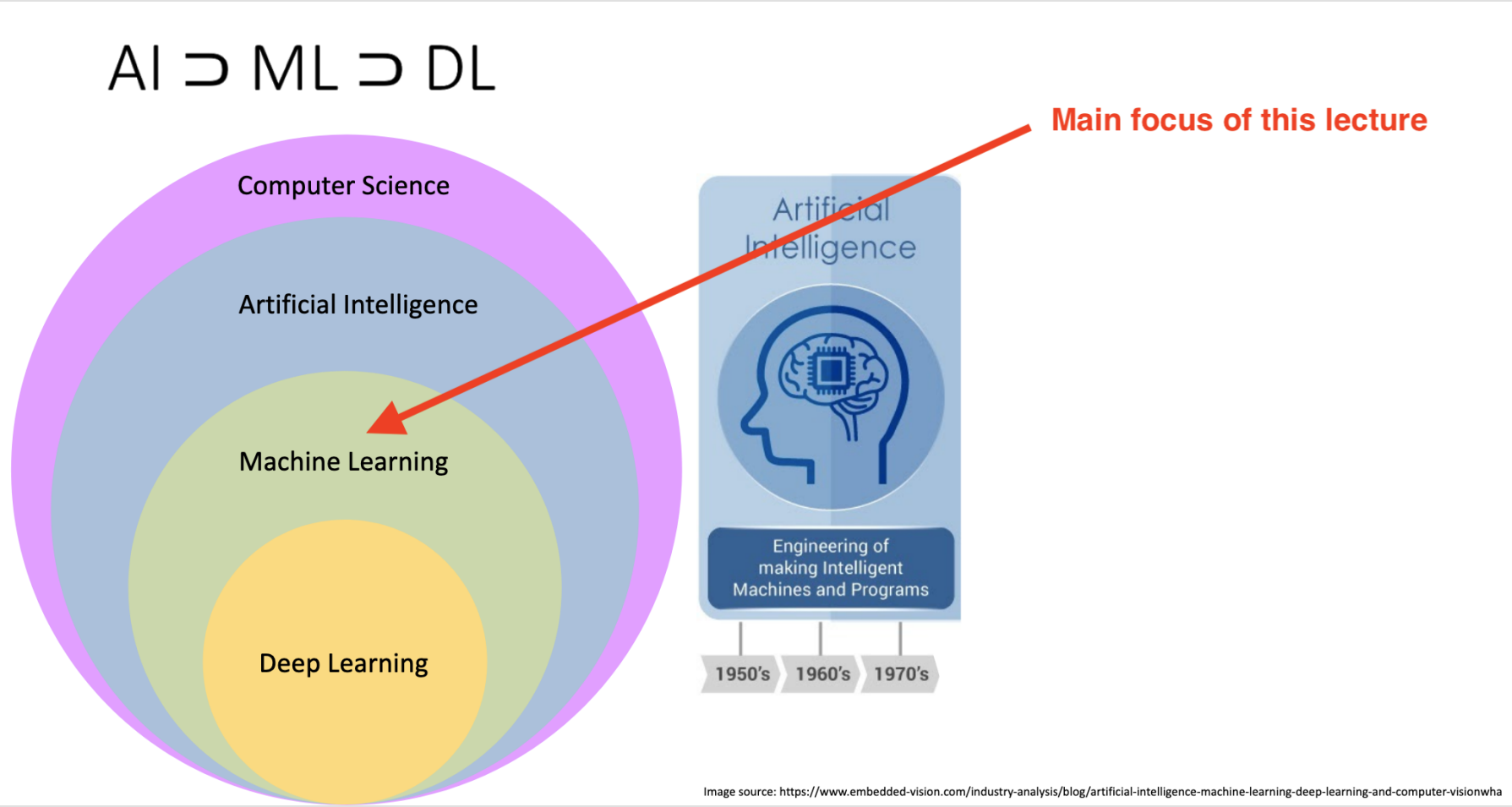
[4] Any document claiming that there is a quick way to understand machine learning without any theory / math is considered as bad

AI, ML and DL



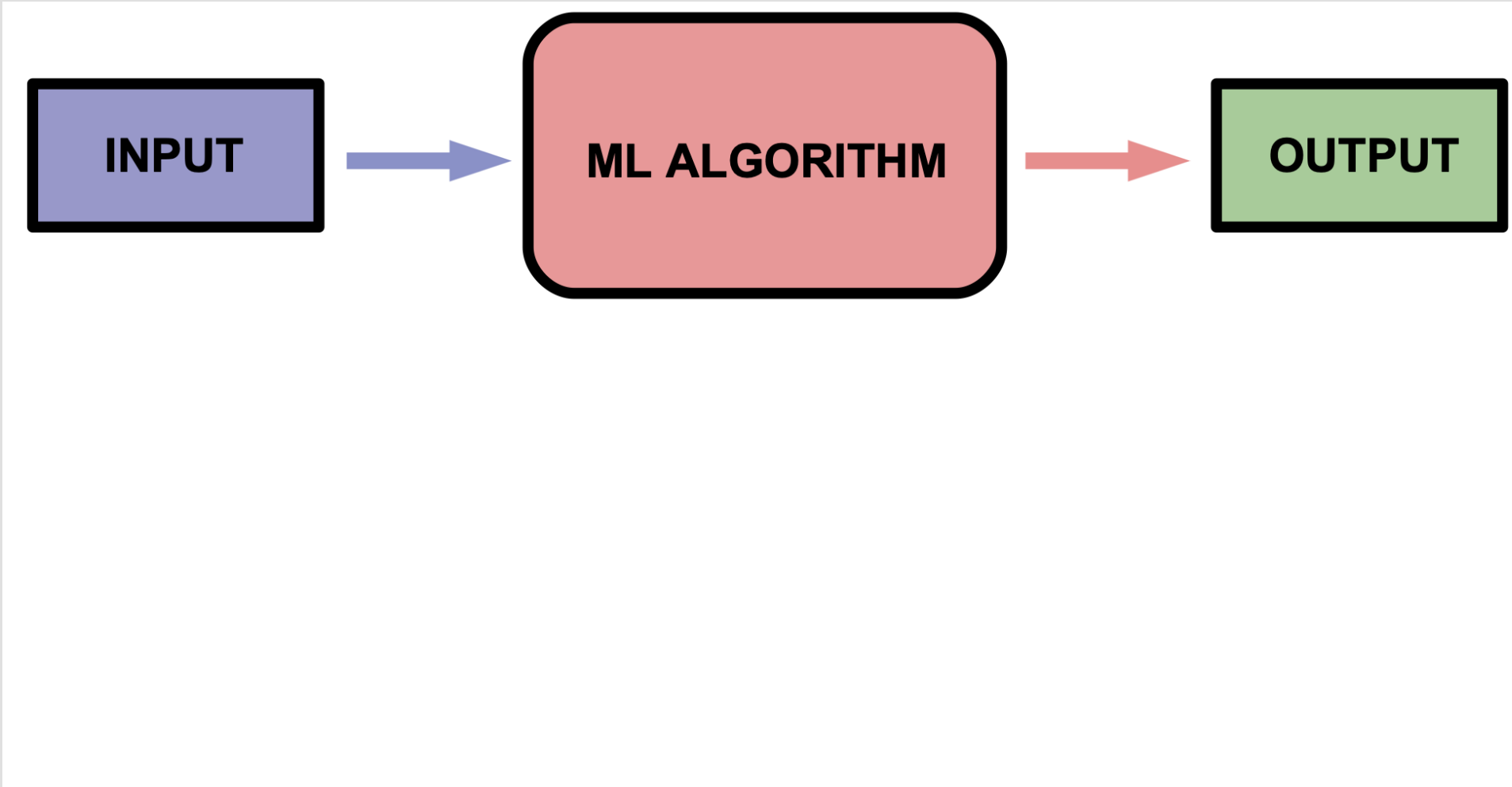
Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

AI, ML and DL

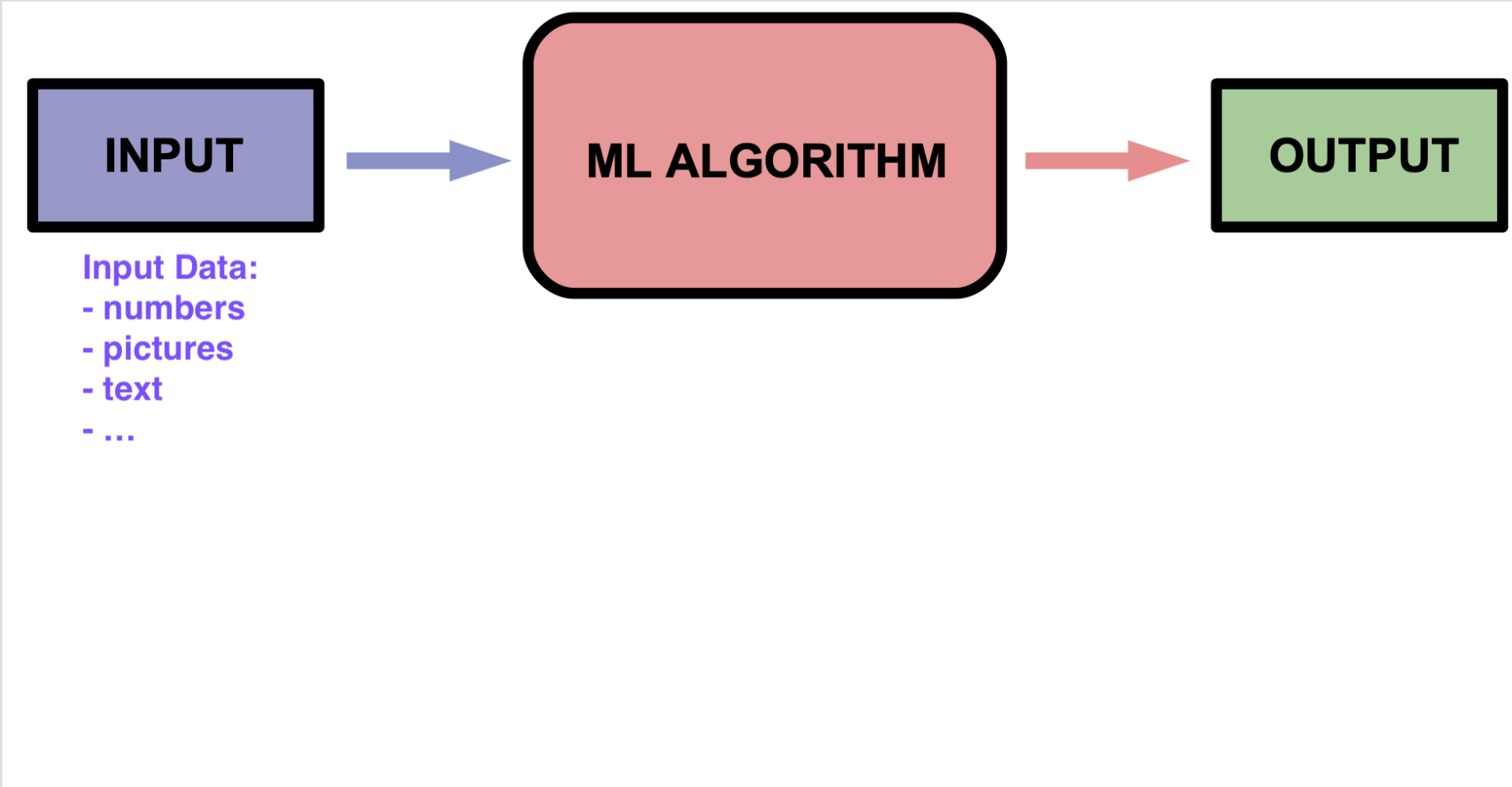


Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

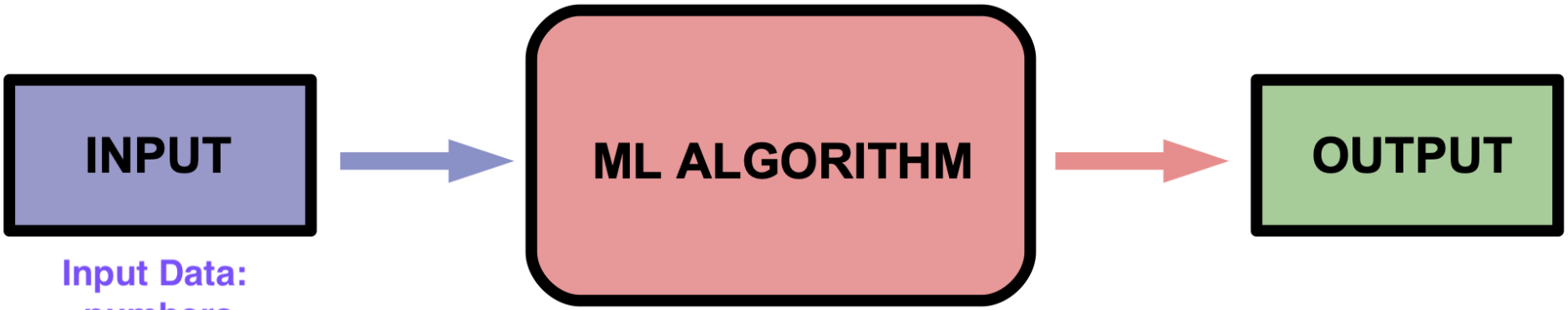
Basic Components of Machine Learning



Basic Components of Machine Learning



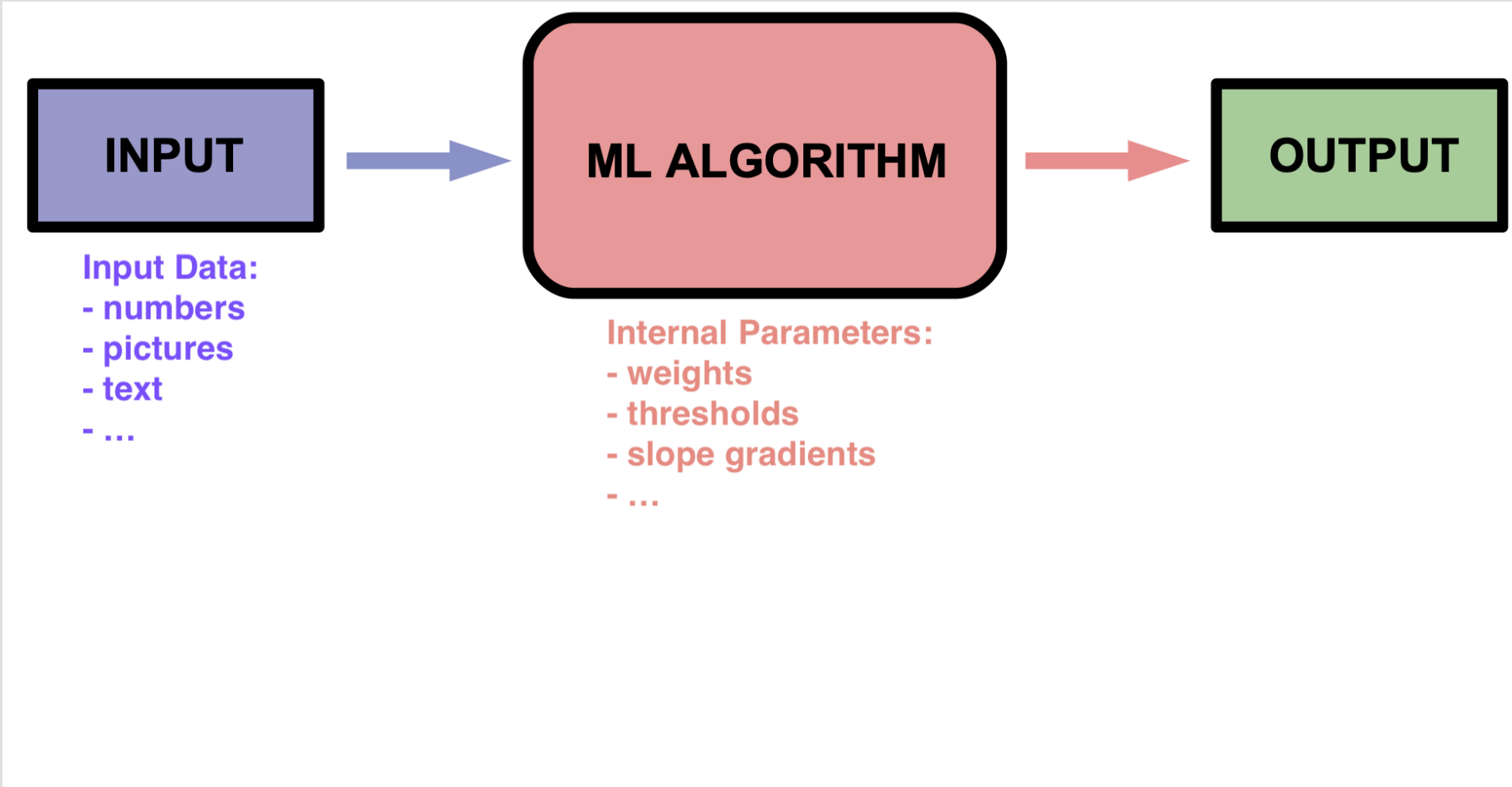
Basic Components of Machine Learning



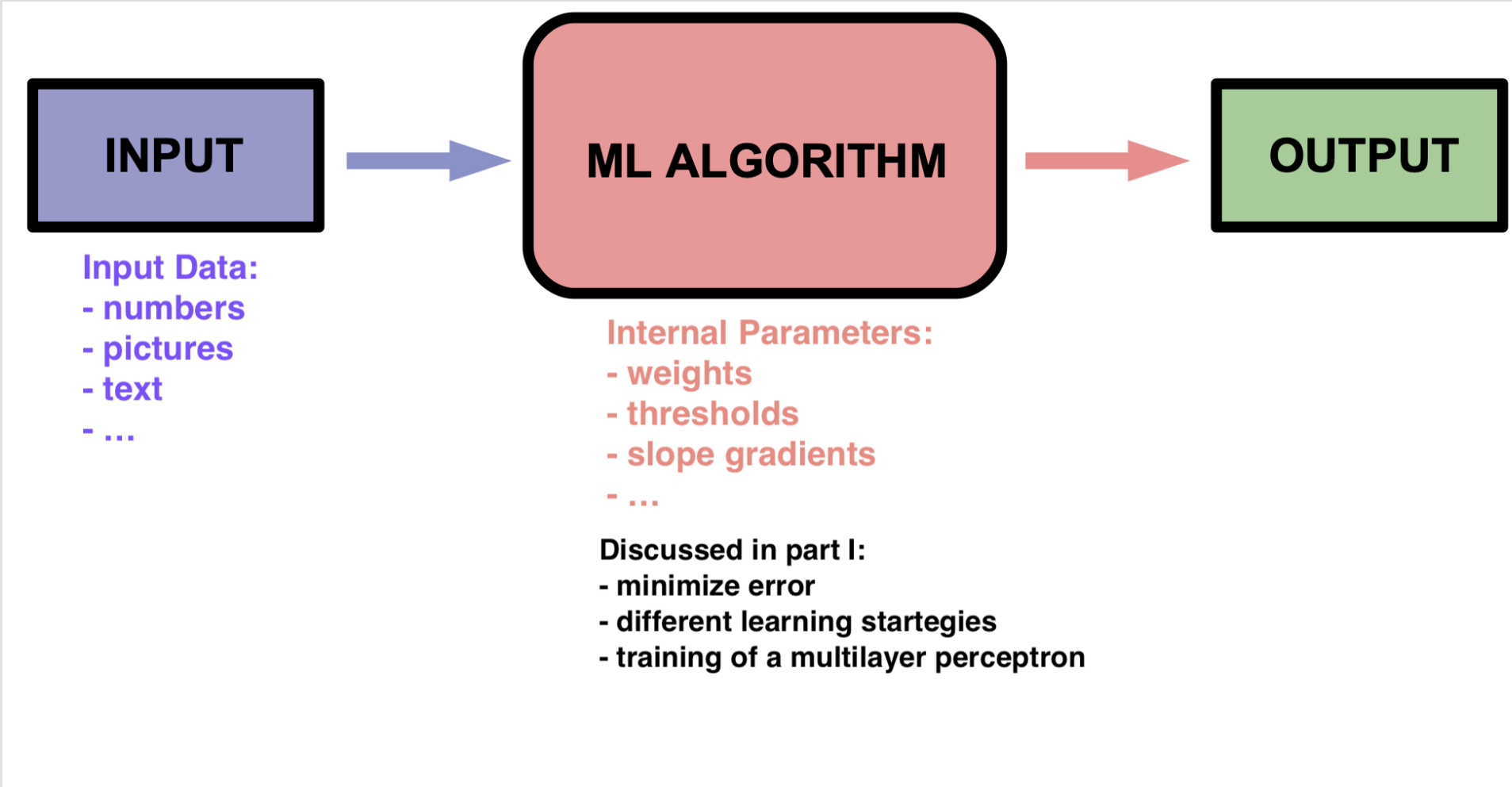
Input Data:
- numbers
- pictures
- text
- ...

Introduced in part I: DataFrames —> handle and manipulate data

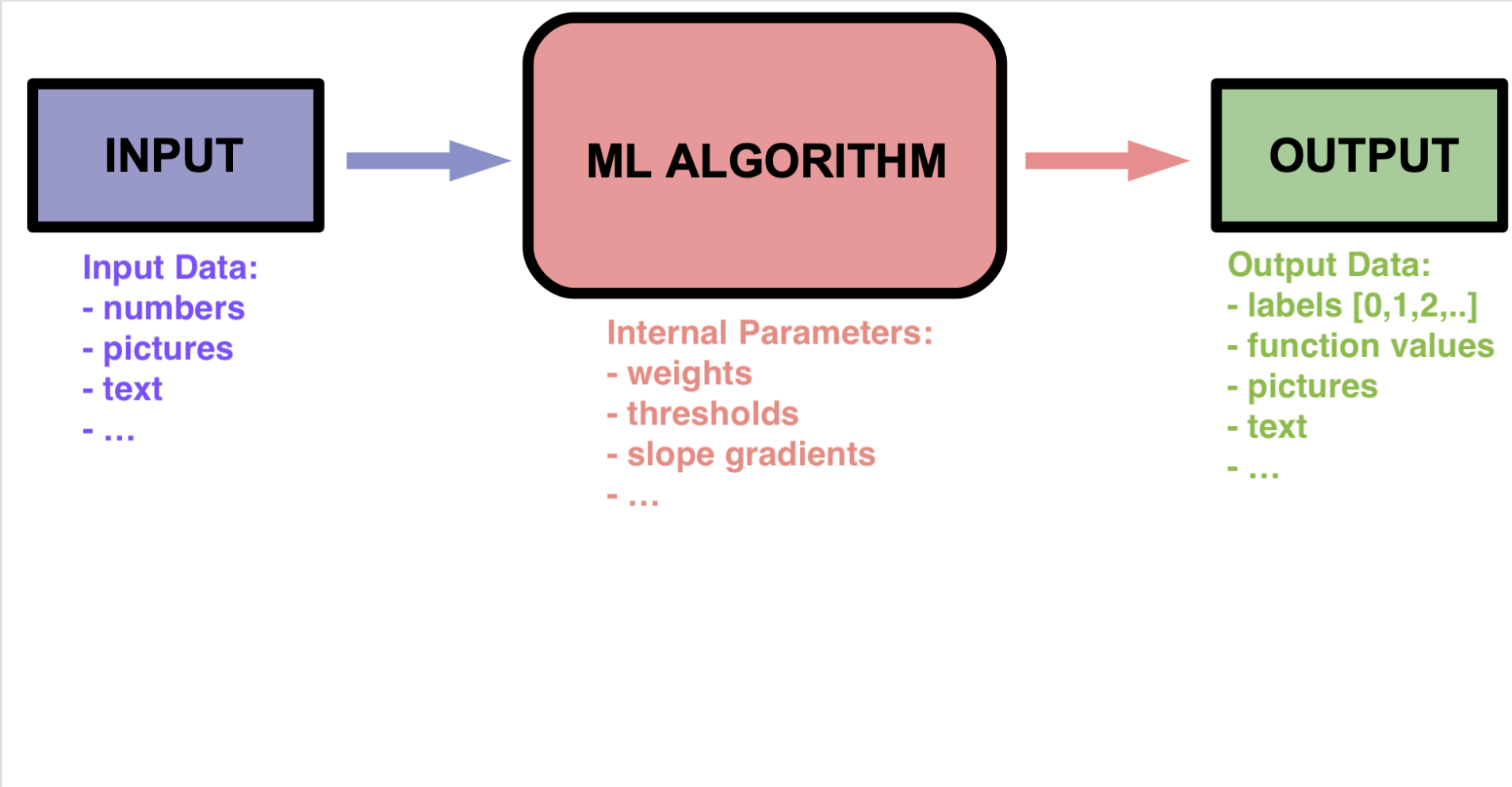
Basic Components of Machine Learning



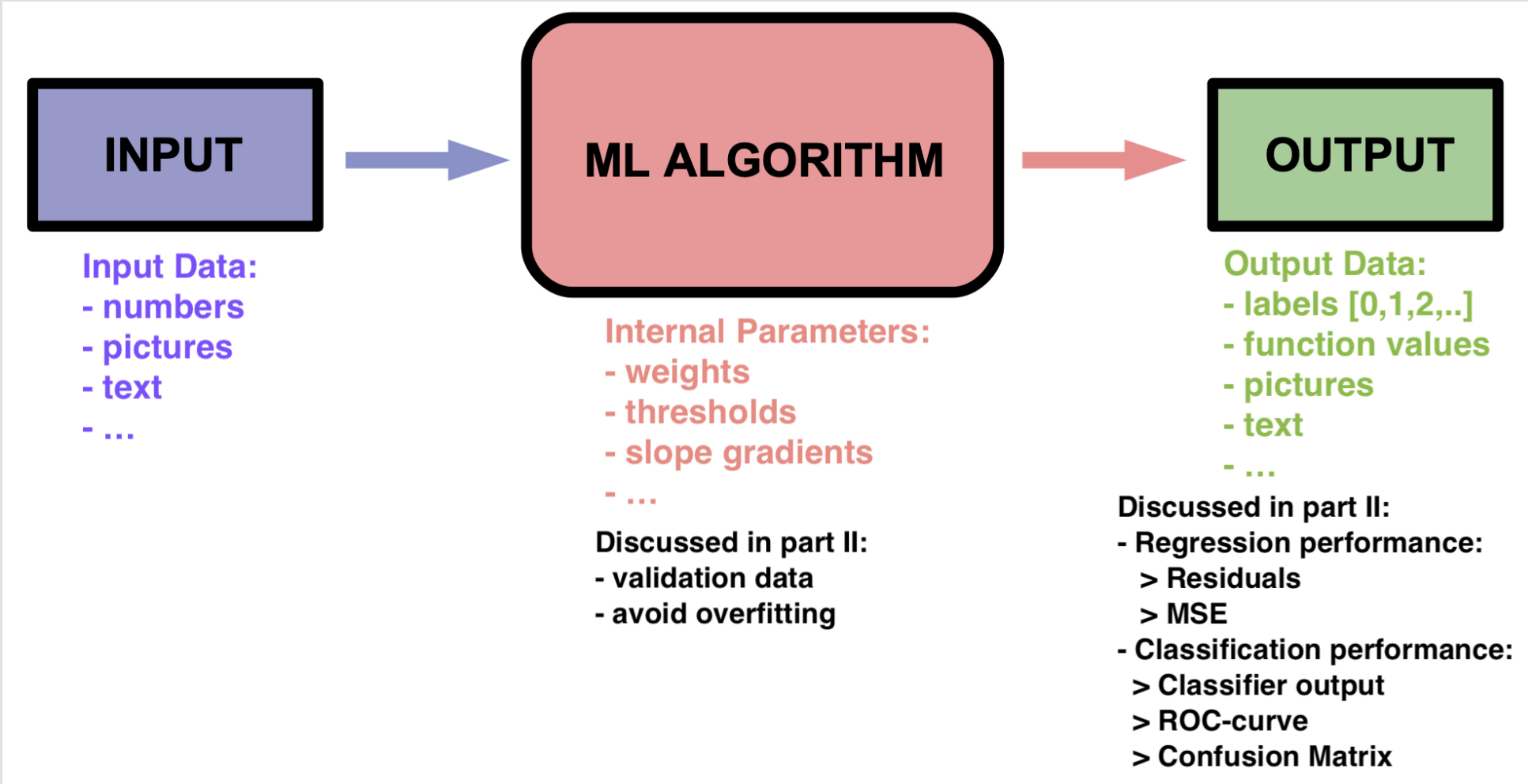
Basic Components of Machine Learning



Basic Components of Machine Learning



Basic Components of Machine Learning



Classification with Machine Learning II

- In Part II: Discussed in detail a classification analysis, using a MLP
- But sometimes a MLP is not the best choice for the underlying problem / data
- Introduce X more classification algorithms which are an alternative to MLPs

Random Forest Classifier: A very brief Introduction

- As the name might suggest: A random forest is composed of several decision trees

[5] e.g. via bootstrapping

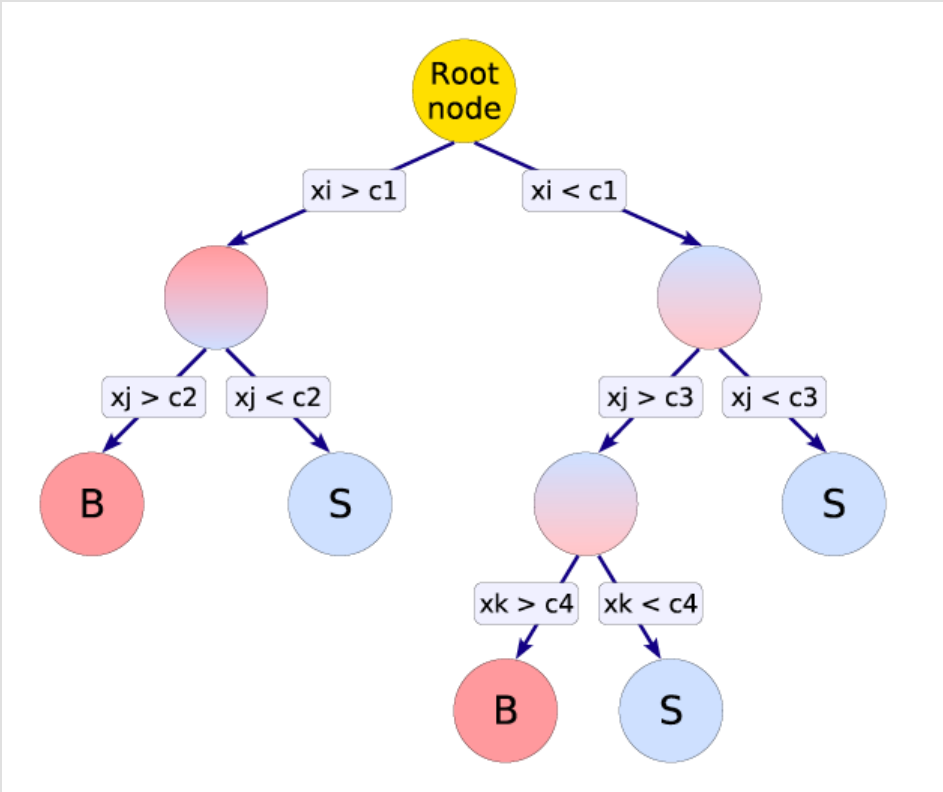
Random Forest Classifier: A very brief Introduction

- As the name might suggest: A random forest is composed of several decision trees
- Classifier ensemble $\leftrightarrow$ Collection of different classifier (all of same type in this case)

[5] e.g. via bootstrapping

Random Forest Classifier: A very brief Introduction

- As the name might suggest: A random forest is composed of several decision trees
- Classifier ensemble $\leftrightarrow$ Collection of different classifier (all of same type in this case)
- The thresholds in each tree are adjusted with respect to maximum separation / purity on the corresponding (sub-) node $\rightarrow$ The splitting features are selected randomly in random forest classifier



Random Forest Classifier: A very brief Introduction

- As the name might suggest: A random forest is composed of several decision trees
- Classifier ensemble $\leftrightarrow$ Collection of different classifier (all of same type in this case)
- The thresholds in each tree are adjusted with respect to maximum separation / purity on the corresponding (sub-) node $\rightarrow$ The splitting features are selected randomly in random forest classifier
- In a random forest, each tree is trained on a individual data set, generated[5] from the provided training data $\Rightarrow$ allows for parallelization during training

[5] e.g. via bootstrapping

Random Forest Classifier: A very brief Introduction

- As the name might suggest: A random forest is composed of several decision trees
- Classifier ensemble $\leftrightarrow$ Collection of different classifier (all of same type in this case)
- The thresholds in each tree are adjusted with respect to maximum separation / purity on the corresponding (sub-) node $\rightarrow$ The splitting features are selected randomly in random forest classifier
- In a random forest, each tree is trained on a individual data set, generated[5] from the provided training data $\Rightarrow$ allows for parallelization during training
- The detailed implementation of a random forest will not be discussed here $\leftrightarrow$ Already done in the homework

[5] e.g. via bootstrapping

(Linear) Support Vector Machine - (Linear) SVM

- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)

(Linear) Support Vector Machine - (Linear) SVM

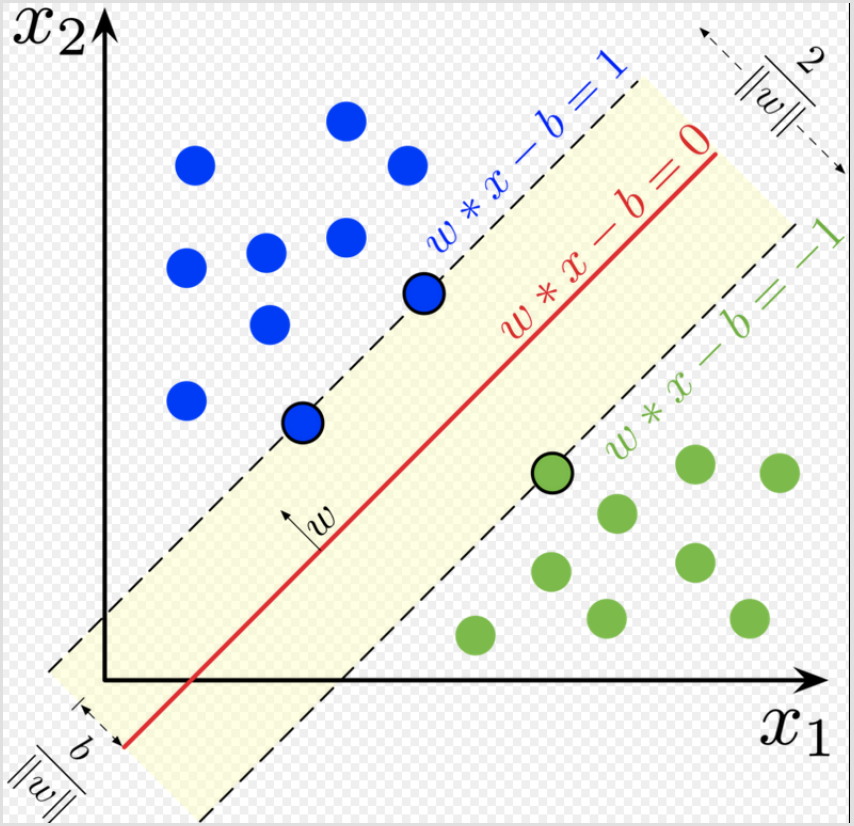
- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- A data set with N events would result in: $\vec{x_1} = \begin{pmatrix} x_1(1) \\ \vdots \\ x_1(n) \end{pmatrix}, \ldots, \vec{x_N} = \begin{pmatrix} x_N(1) \\ \vdots \\ x_N(n) \end{pmatrix}$

(Linear) Support Vector Machine - (Linear) SVM

- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- A data set with N events would result in: $\vec{x_1} = \begin{pmatrix} x_1(1) \\ \vdots \\ x_1(n) \end{pmatrix}, \ldots, \vec{x_N} = \begin{pmatrix} x_N(1) \\ \vdots \\ x_N(n) \end{pmatrix}$
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)

(Linear) Support Vector Machine - (Linear) SVM

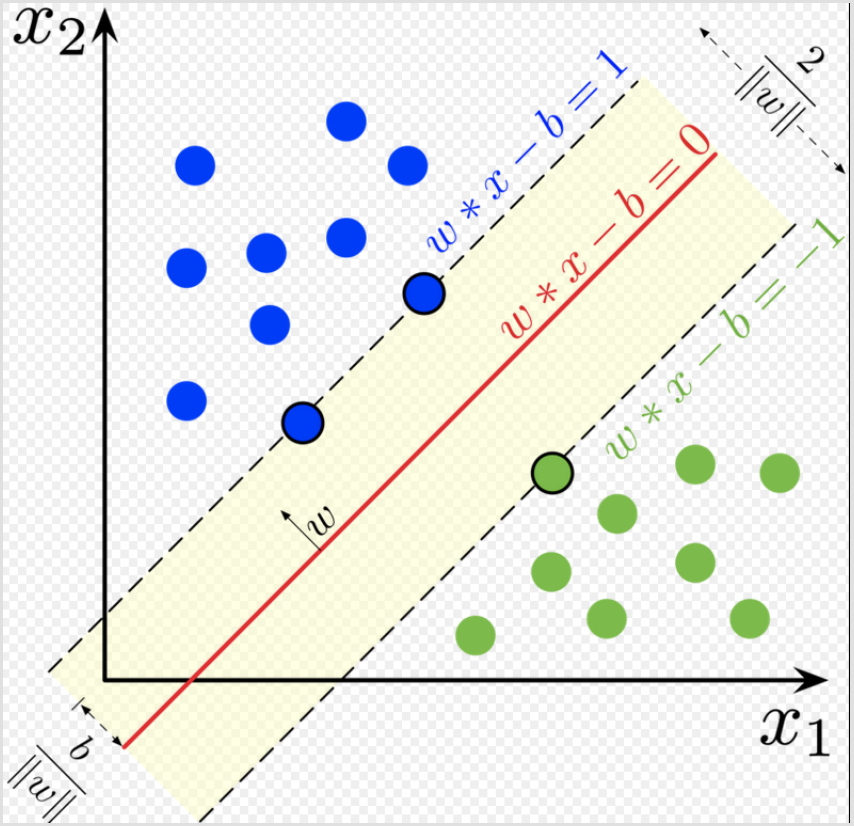
- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)



Picture taken from Larhman @ wikipedia

(Linear) Support Vector Machine - (Linear) SVM

- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)

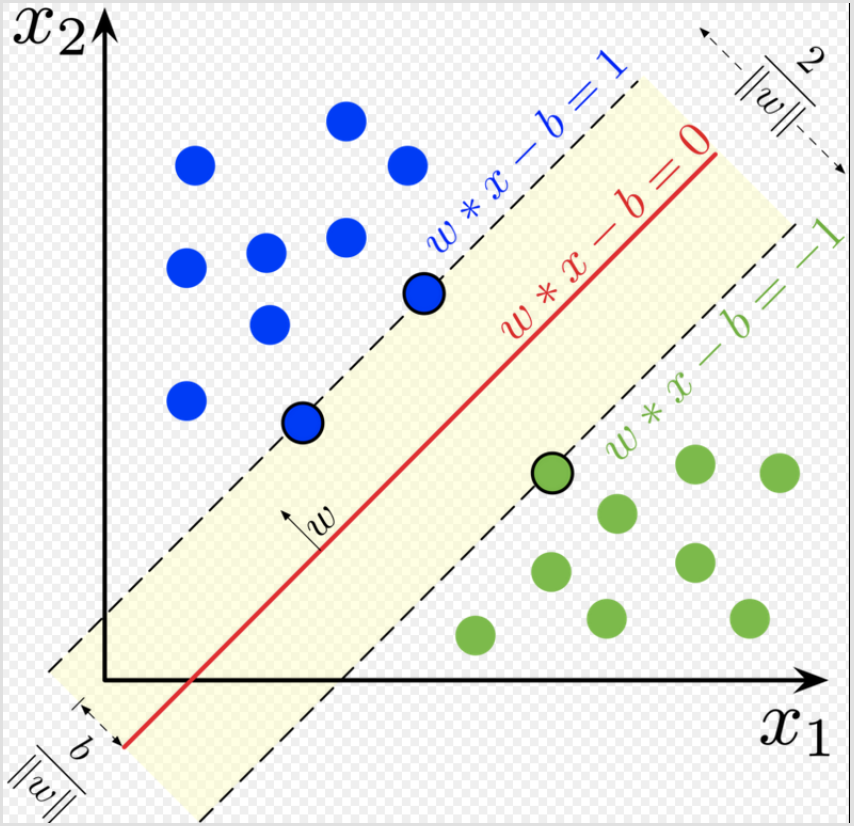


Picture taken from Larhman @ wikipedia

- Data points closest (dashed lines) to separation hyperplane (red solid line) $\rightarrow$ support vectors (they "support" the decision)

(Linear) Support Vector Machine - (Linear) SVM

- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)

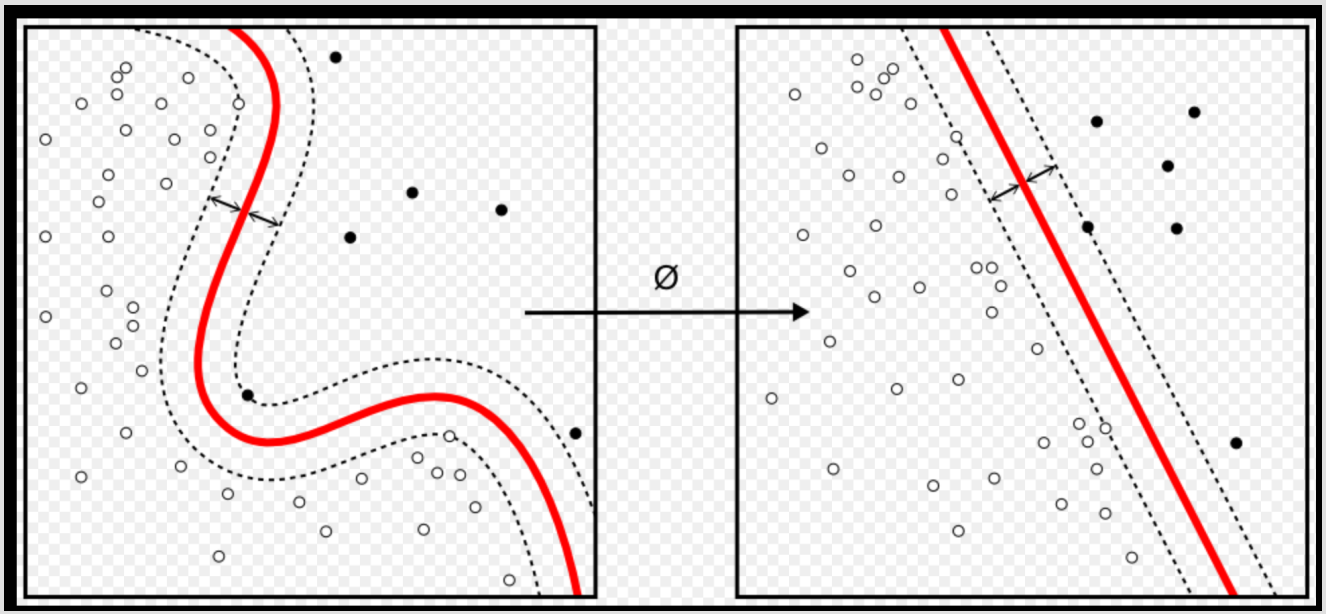


Picture taken from Larhman @ wikipedia

- Data points closest (dashed lines) to separation hyperplane (red solid line) $\rightarrow$ support vectors (they "support" the decision)
- Depending on whether then data is linearly separable or not, one has to allow margins

(Linear) Support Vector Machine - (Linear) SVM

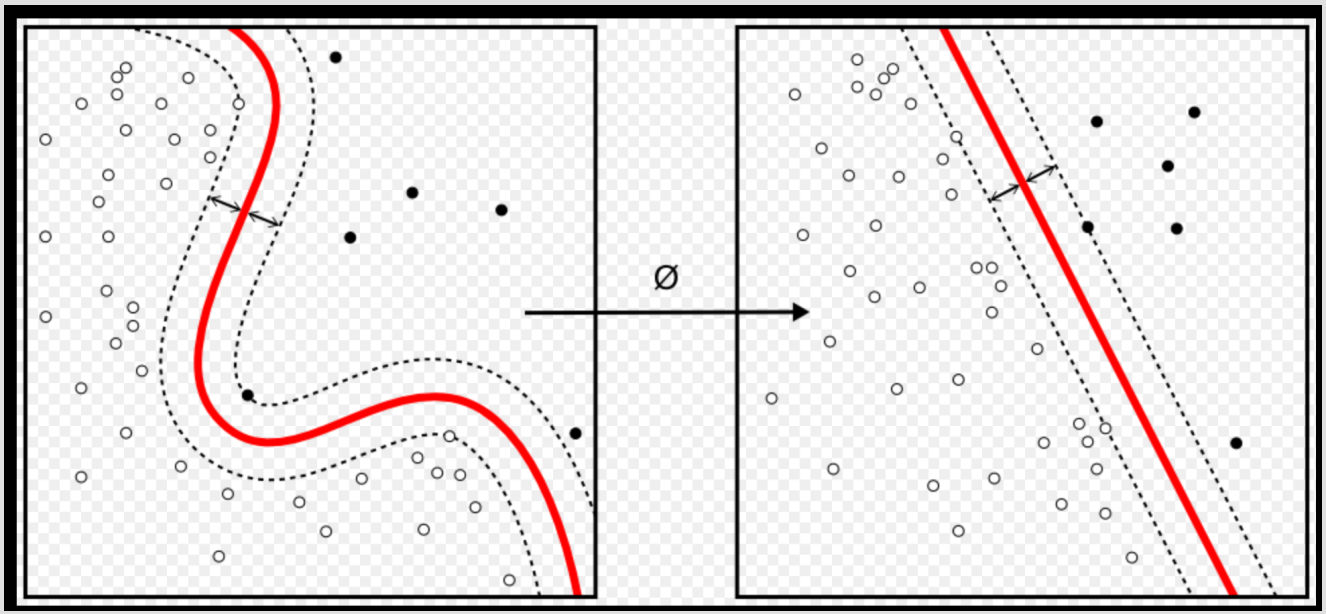
- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)
- Data points closest (dashed lines) to separation hyperplane (red solid line) $\rightarrow$ support vectors (they "support" the decision)
- Depending on whether then data is linearly separable or not, one has to allow margins
- If data is highly non-linear, one might have to include a kernel $\rightarrow$ increase feature dimensionality



Picture taken from Alisneaky @ wikipedia

(Linear) Support Vector Machine - (Linear) SVM

- Data is represented as n-dimensional vectors (n $\equiv$ feature dimension)
- **Goal of the SVM:** Find a hyperplane that separates the data best (according to the provided labels)
- Data points closest (dashed lines) to separation hyperplane (red solid line) $\rightarrow$ support vectors (they "support" the decision)
- Depending on whether then data is linearly separable or not, one has to allow margins
- If data is highly non-linear, one might have to include a kernel $\rightarrow$ increase feature dimensionality



Picture taken from Alisneaky @ wikipedia

- Linear SVM $\rightarrow$ Assume linearly separable data, no specific kernel needed

Include Linear SVM in Analysis

- Setup linear SVM in scikit

```
from sklearn.svm import LinearSVC

my_linsvm  = LinearSVC(
    max_iter = 1000, #-->Set the max. number of iterations
    C=1.0, #--> Steer performance, large / low margin
    tol=1e-6
)
```

Include Linear SVM in Analysis

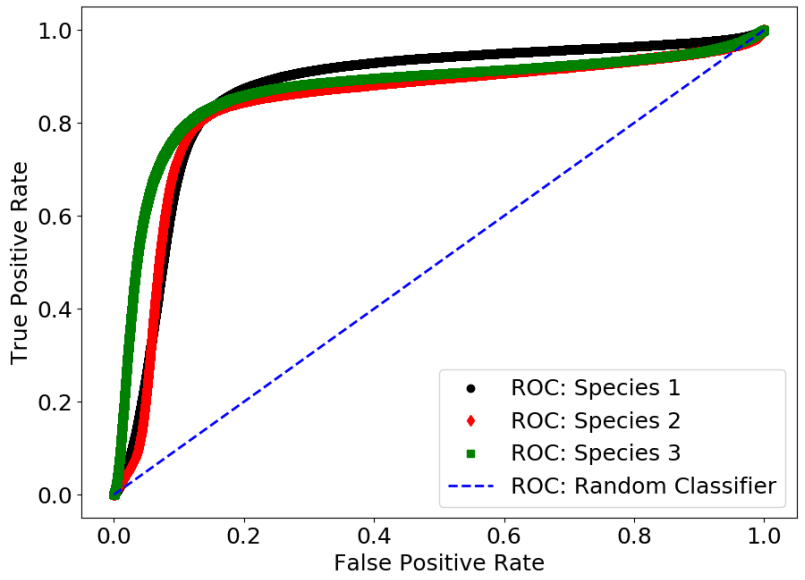
- Setup linear SVM in scikit
- Fit the data and obtain predicted labels

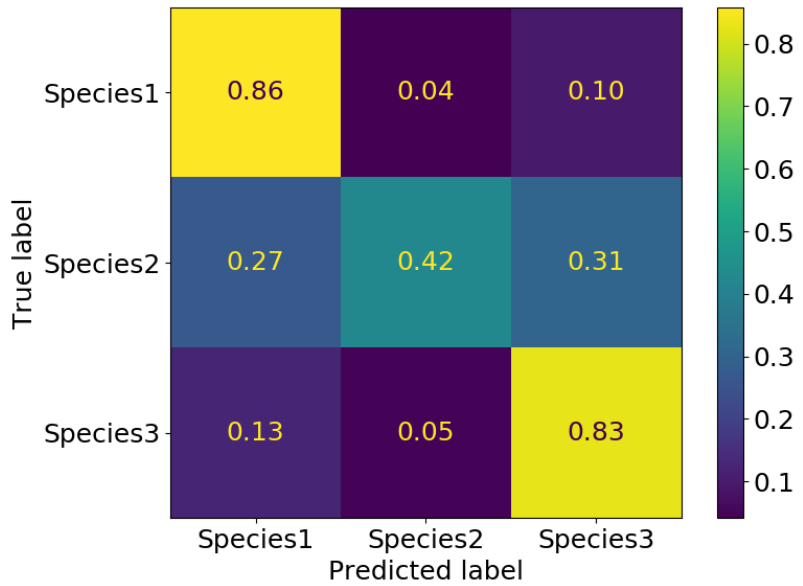
```
#Train the model:
my_linsvm.fit(x_train,y_train)

#Get the predictions:
predictions = my_linsvm.predict(X)
```

Include Linear SVM in Analysis

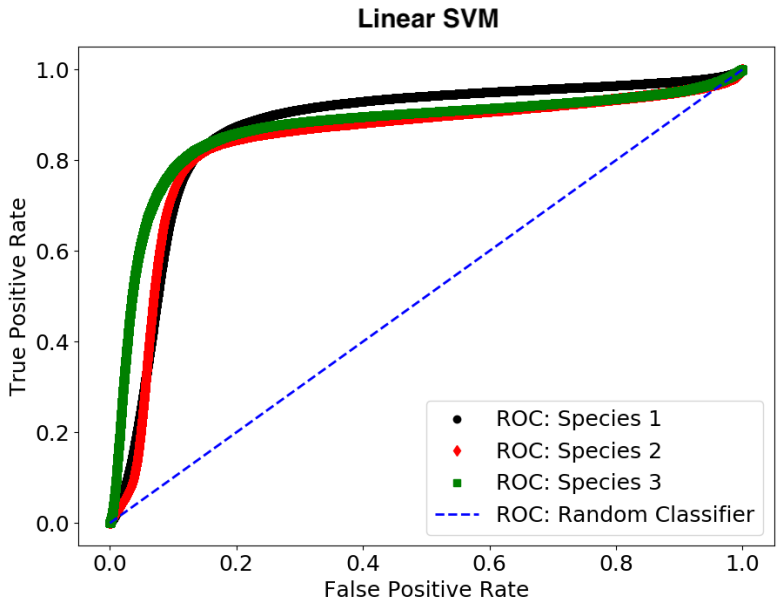
- Setup linear SVM in scikit
- Fit the data and obtain predicted labels
- Check performance

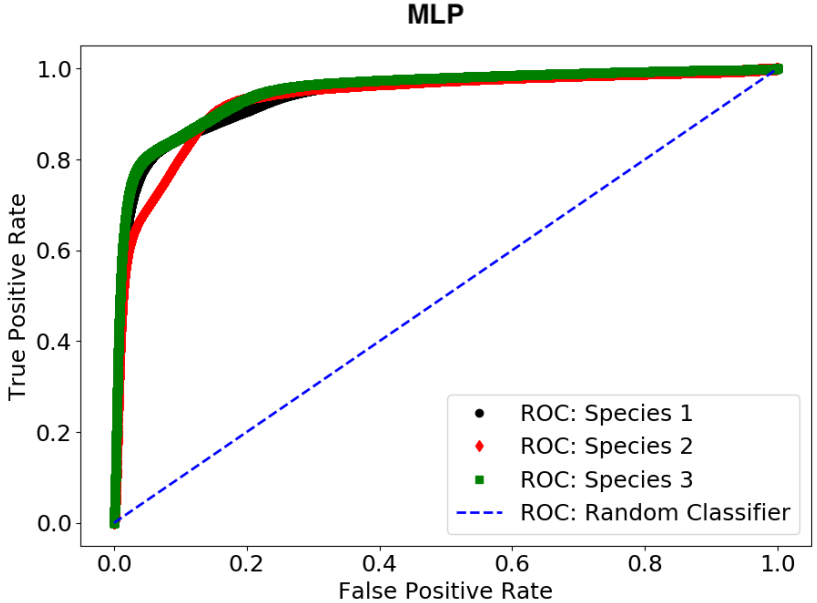




Include Linear SVM in Analysis

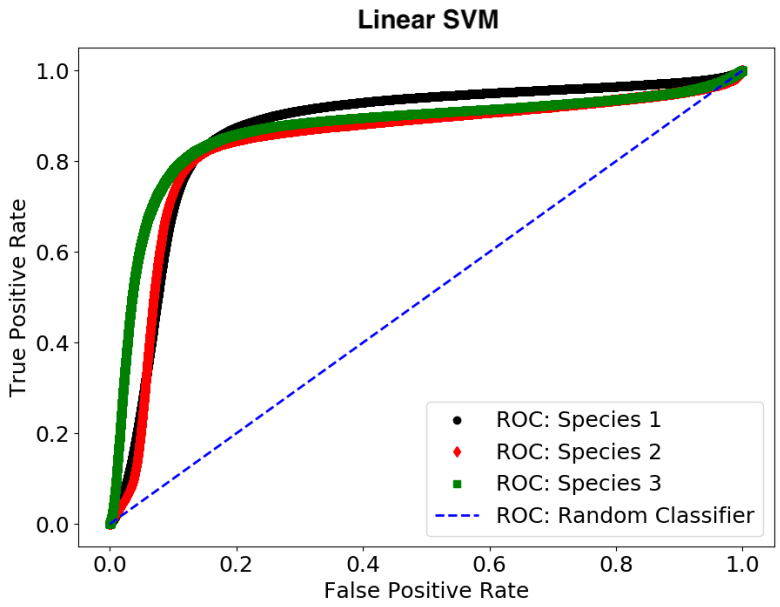
- Setup linear SVM in scikit
- Fit the data and obtain predicted labels
- Check performance
- Compare performance with other algorithms

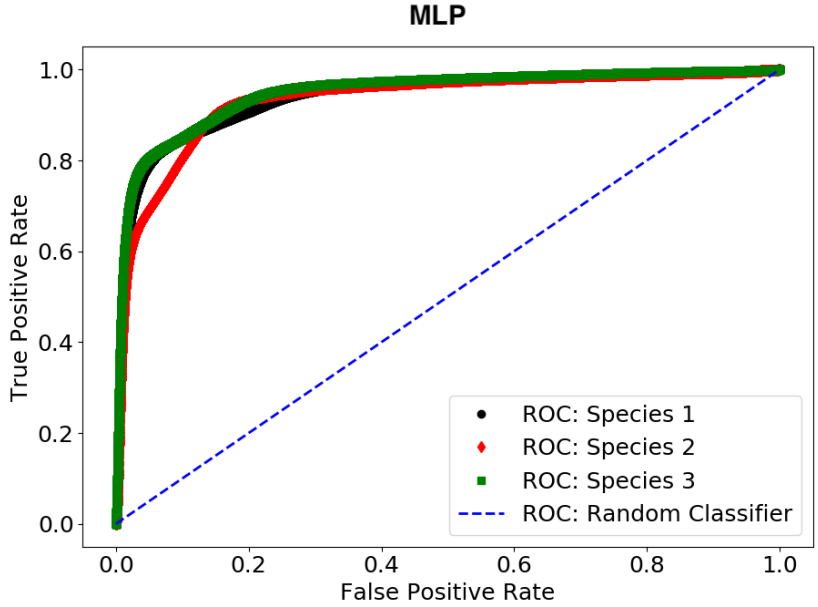




Include Linear SVM in Analysis

- Setup linear SVM in scikit
- Fit the data and obtain predicted labels
- Check performance
- Compare performance with other algorithms

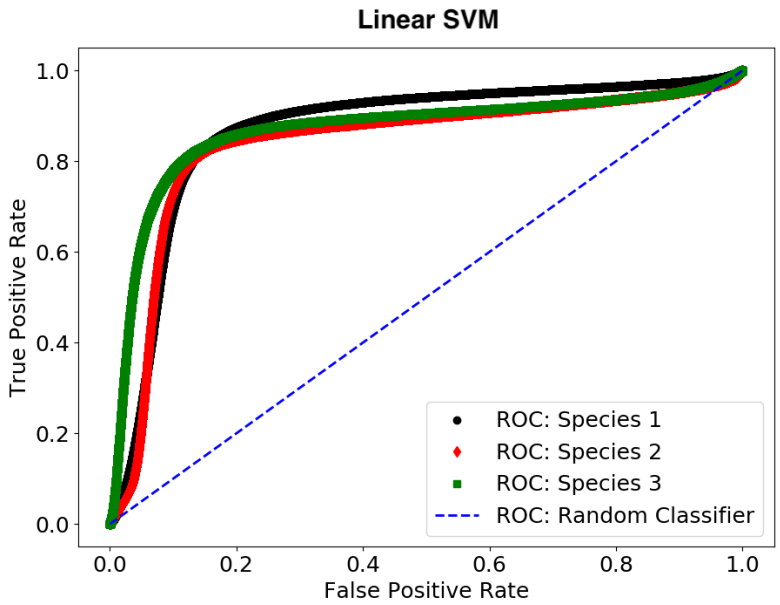


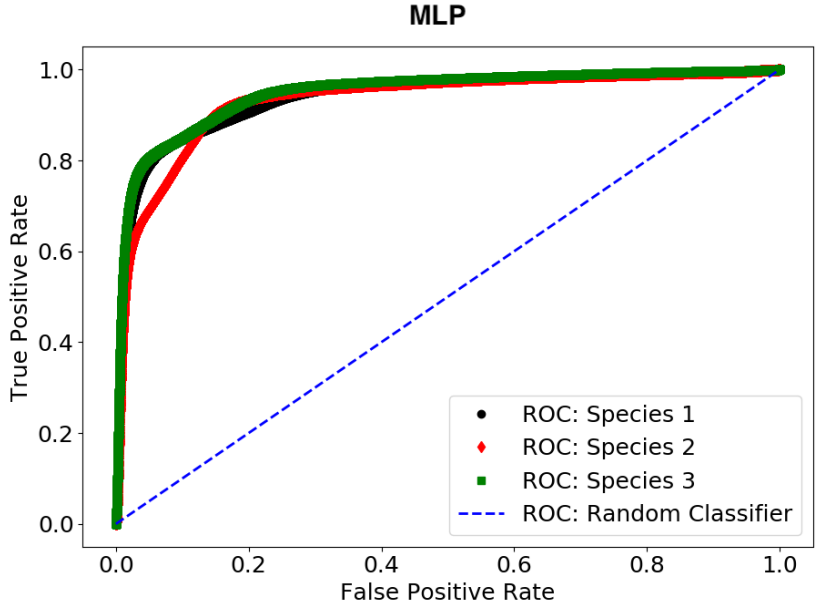


- Linear SVM is not best choice for classification $\rightarrow$ indicates that data might not be linearly separable $\rightarrow$ include kernel

Include Linear SVM in Analysis

- Setup linear SVM in scikit
- Fit the data and obtain predicted labels
- Check performance
- Compare performance with other algorithms





- Linear SVM is not best choice for classification $\rightarrow$ indicates that data might not be linearly separable $\rightarrow$ include kernel
- Is there another way to compare classifier (performances) $\rightarrow$ Evaluation metrics

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  - i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?
  - ii) False Positive Rate $\leftrightarrow$ How many background events are wrongly identified as signal?

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  - i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?
  - ii) False Positive Rate $\leftrightarrow$ How many background events are wrongly identified as signal?
  - Ideally[7], those quantities are universal, i.e. independent of the relative abundance between signal / background events $\rightarrow$ They purely depend on the underlying algorithm

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

[7] i.e. sufficient statistics

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  - i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?
  - ii) False Positive Rate $\leftrightarrow$ How many background events are wrongly identified as signal?
  - Ideally[7], those quantities are universal, i.e. independent of the relative abundance between signal / background events $\rightarrow$ They purely depend on the underlying algorithm
- There are many other metrics that can be directly derived from i) and ii)

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

[7] i.e. sufficient statistics

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  - i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?
  - ii) False Positive Rate $\leftrightarrow$ How many background events are wrongly identified as signal?
  - Ideally[7], those quantities are universal, i.e. independent of the relative abundance between signal / background events $\rightarrow$ They purely depend on the underlying algorithm
- There are many other metrics that can be directly derived from i) and ii)
- Scikit provides a huge metrics library
  `https://scikit-learn.org/stable/modules/classes.html`

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

[7] i.e. sufficient statistics

Evaluation Metrics

- Introduced the ROC-curve and confusion matrix as performance monitoring tools
- It can be sometimes helpful to summarize the performance within one single number[6]
- Already introduced:
  - i) True Positive Rate $\leftrightarrow$ How many signal events are identified correctly?
  - ii) False Positive Rate $\leftrightarrow$ How many background events are wrongly identified as signal?
  - Ideally[7], those quantities are universal, i.e. independent of the relative abundance between signal / background events $\rightarrow$ They purely depend on the underlying algorithm
- There are many other metrics that can be directly derived from i) and ii)
- Scikit provides a huge metrics library
  `https://scikit-learn.org/stable/modules/classes.html`
- On the following slides: introduce a few of the most common metrics

[6] However, a (multidimensional) distribution generally provides more information and should always included into any analysis

[7] i.e. sufficient statistics

The Purity

- A common definition of the purity with respect to a species *i* is:

$$\text{Purity i} = \frac{\#\text{Events with species i \& identified as species i}}{\#\text{Events identified as species i}} \tag{1}$$

The Purity

- A common definition of the purity with respect to a species *i* is:

$$\text{Purity i} = \frac{\#\text{Events with species i \& identified as species i}}{\#\text{Events identified as species i}} \tag{1}$$

- Purity describes "cleanliness" of data set after classification

The Purity

- A common definition of the purity with respect to a species i is:

$$\text{Purity i} = \frac{\#\text{Events with species i \& identified as species i}}{\#\text{Events identified as species i}} \tag{1}$$

- Purity describes "cleanliness" of data set after classification
- Using the basic definitions of true positive rate and false positive rate, one might rewrite the equation above to:

$$\text{Purity i} = \frac{\text{TPR(i)}}{TPR(i) + \left(\frac{1-R(i)}{R(i)}\right) \cdot FPR(i)} \tag{2}$$

with: TPR(i) / FPR(i) being the true / false positive rate for species i and: $R(i) = \sum_i \frac{\#\text{Events with species i}}{\#\text{Events}}$, the relative abundance of species i within the given data set

The Purity: Example

- Lets take a look at the data set we have analyzed in this lecture:
  Species 3 covers about 36% of the entire data

The Purity: Example

- Lets take a look at the data set we have analyzed in this lecture:
  Species 3 covers about 36% of the entire data
- In part II we trained a mlp classifier, from its ROC-curve we can extract for example:
  TPR(species 3) ≈ 0.92 and FPR(species 3) ≈ 0.2

The Purity: Example

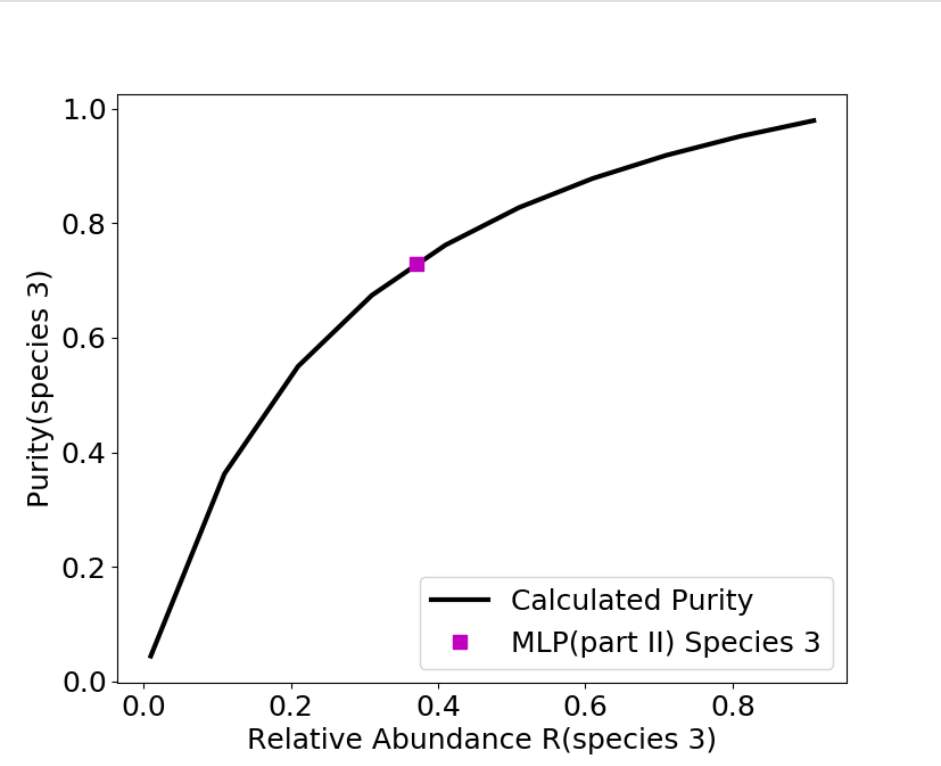
- Lets take a look at the data set we have analyzed in this lecture:
  Species 3 covers about 36% of the entire data
- In part II we trained a mlp classifier, from its ROC-curve we can extract for example:
  TPR(species 3) ≈ 0.92 and FPR(species 3) ≈ 0.2
- Using the equation from the previous slide yields:
  Purity(sepcies 3) = TPR(3) / (TPR(3) + FPR(3) $\cdot$ [1-R(3)]/R(3))
  $= 0.92$ / (0.92 + [1-0.36]/0.36 $\cdot$ 0.2) $\approx 72\,\%$

The Purity: Example

- Lets take a look at the data set we have analyzed in this lecture:
  Species 3 covers about 36% of the entire data
- In part II we trained a mlp classifier, from its ROC-curve we can extract for example:
  TPR(species 3) ≈ 0.92 and FPR(species 3) ≈ 0.2
- Using the equation from the previous slide yields:
  Purity(sepcies 3) = TPR(3) / (TPR(3) + FPR(3) $\cdot$ [1-R(3)]/R(3))
  $= 0.92$ / (0.92 + [1-0.36]/0.36 $\cdot$ 0.2) $\approx 72\,\%$
- However, one has to be careful when using the purity, because it depends on the relative abundances, summarized in $R(i)$

The Purity: Example

- Lets take a look at the data set we have analyzed in this lecture:
  Species 3 covers about 36% of the entire data
- In part II we trained a mlp classifier, from its ROC-curve we can extract for example:
  TPR(species 3) ≈ 0.92 and FPR(species 3) ≈ 0.2
- Using the equation from the previous slide yields:
  Purity(sepcies 3) = TPR(3) / (TPR(3) + FPR(3) $\cdot$ [1-R(3)]/R(3))
  $= 0.92$ / (0.92 + [1-0.36]/0.36 $\cdot$ 0.2) $\approx 72\,\%$
- However, one has to be careful when using the purity, because it depends on the relative abundances, summarized in $R(i)$



The F1-Score

- Deduced from F-measure
- The F1-Score basically folds in the true positive rate (or efficiency) and the purity

$$\text{F1-Score (species i)} = 2 \cdot \frac{\text{TPR(i)} \cdot \text{Purity(i)}}{\text{TPR(i)} + \text{Purity(i)}} \tag{3}$$

- By definition, the F1-Score also depends on the relative abundances $R(i)$

The Accuracy

- The accuracy is given by:

$$\text{Accuracy} = \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} \#\text{Events with species i \& identified as i} \quad (4)$$

$$= \frac{1}{\#\text{Events}} \sum_{i}^{\#\text{Species}} \text{TPR(i)} \cdot N(i) \quad (5)$$

$$= \sum_{i}^{\#\text{Species}} \text{TPR(i)} \cdot R(i) \quad (6)$$

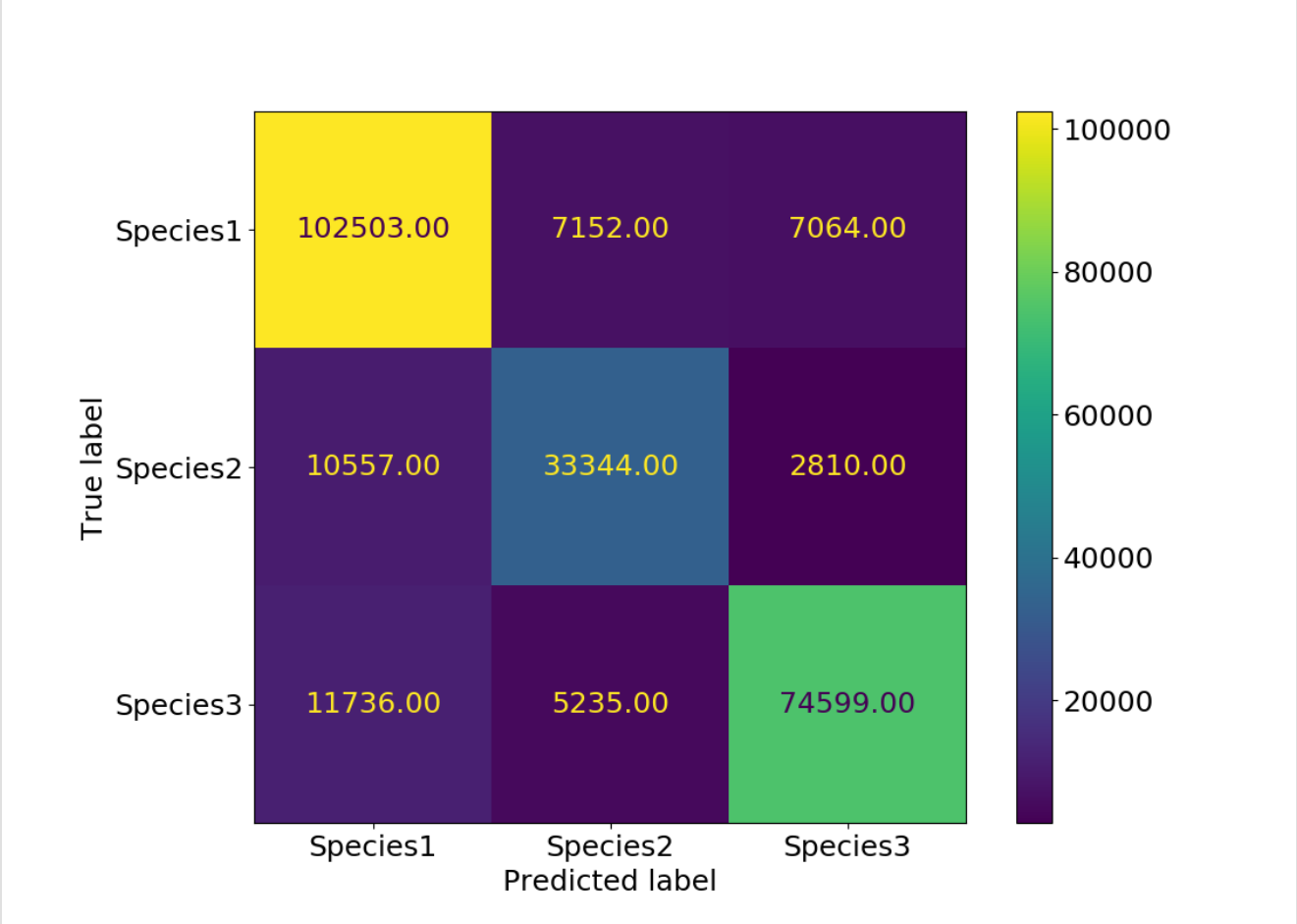
- Or: Accuracy $\propto$ Tr(confusion matrix)
- Again, the relative abundance is folded in
- The accuracy is less reliable on highly imbalanced data, i.e. R(j) >> R(i)

Accuracy for the MLP

- Look again at the MLP from lecture part II

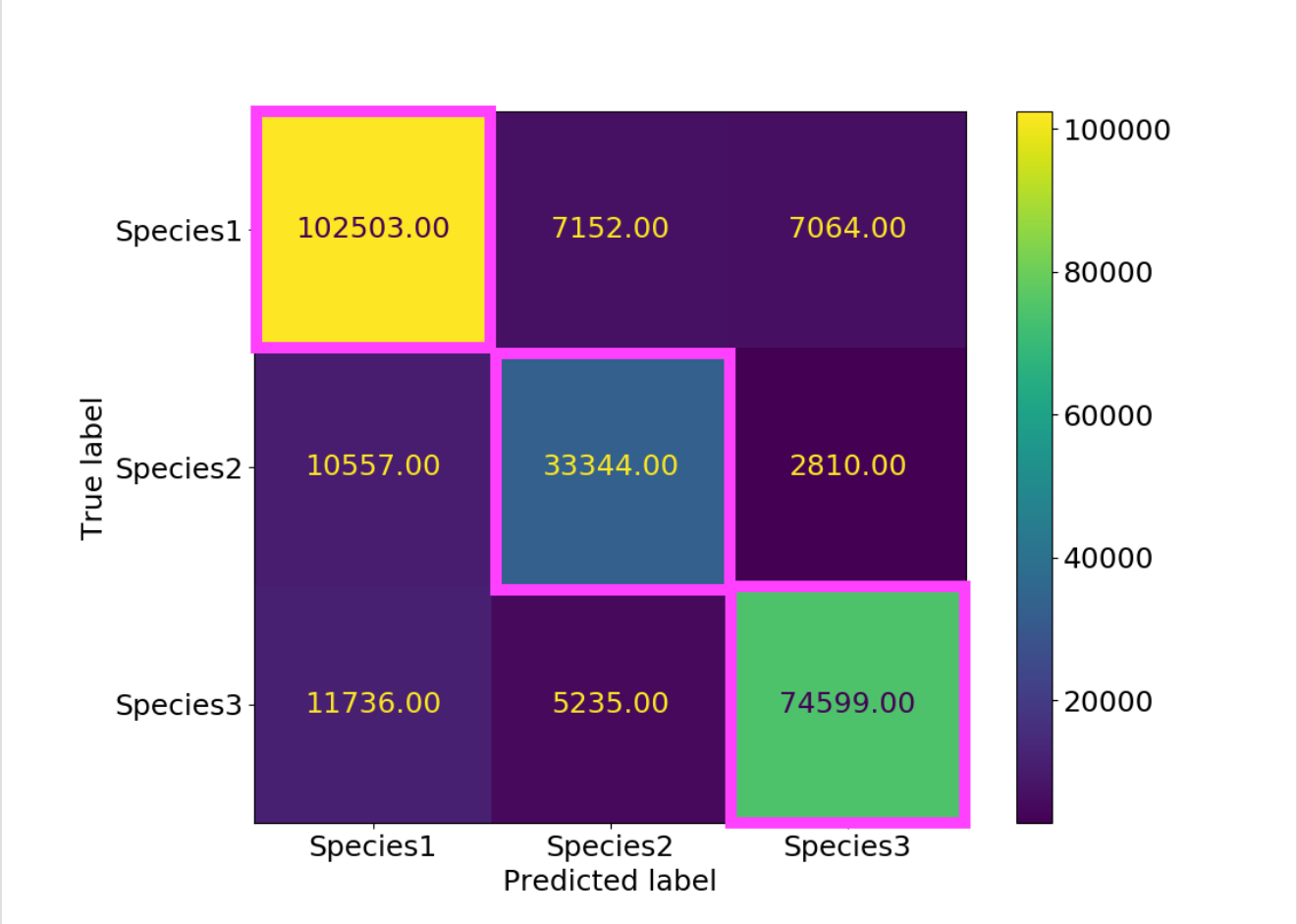
Accuracy for the MLP

- Look again at the MLP from lecture part II
- The unnormalized confusion matrix is:



Accuracy for the MLP

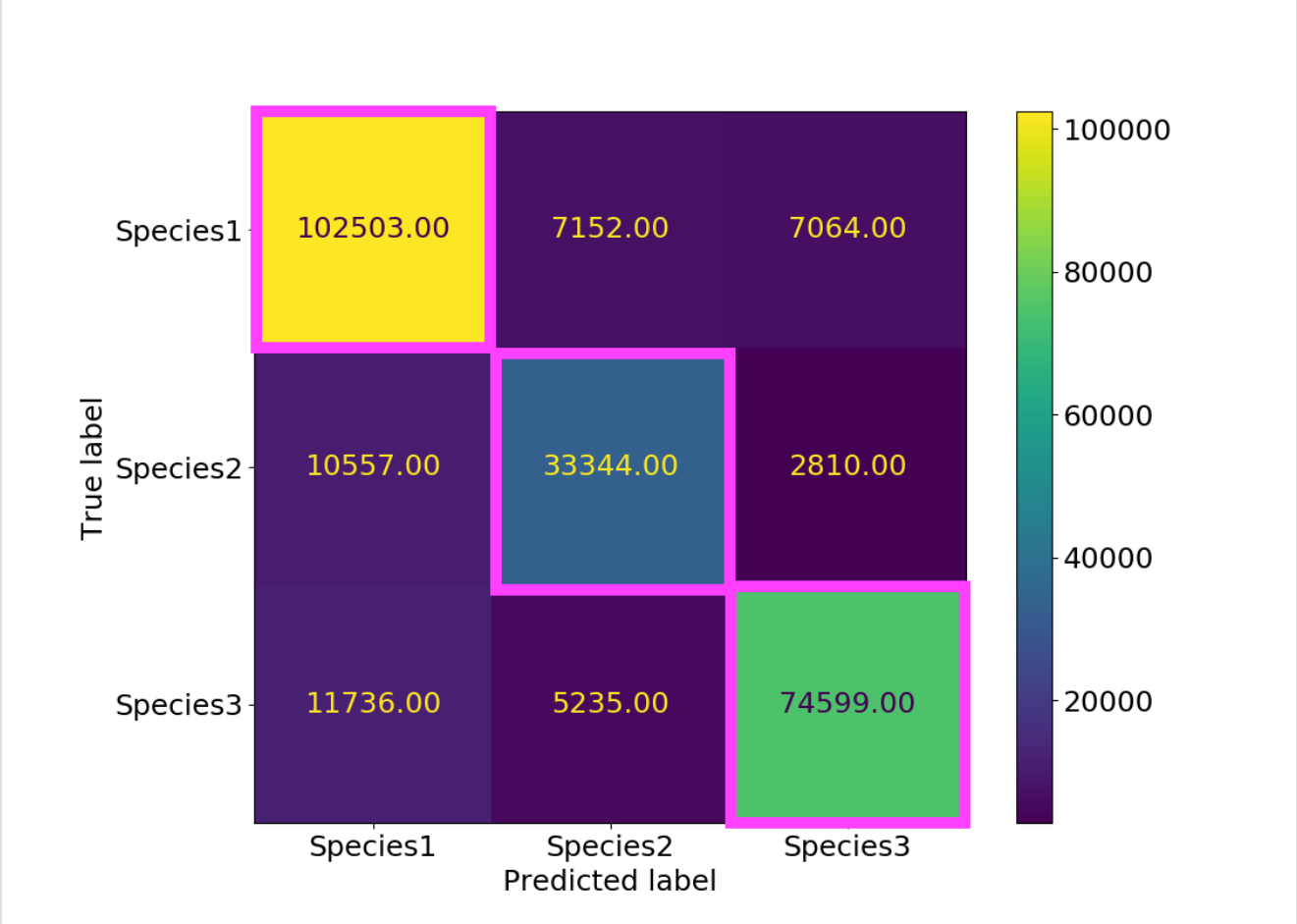
- Look again at the MLP from lecture part II
- The unnormalized confusion matrix is:



- The trace of this matrix is: TR = 102,503 + 33,344 + 74,599 = 210,446

Accuracy for the MLP

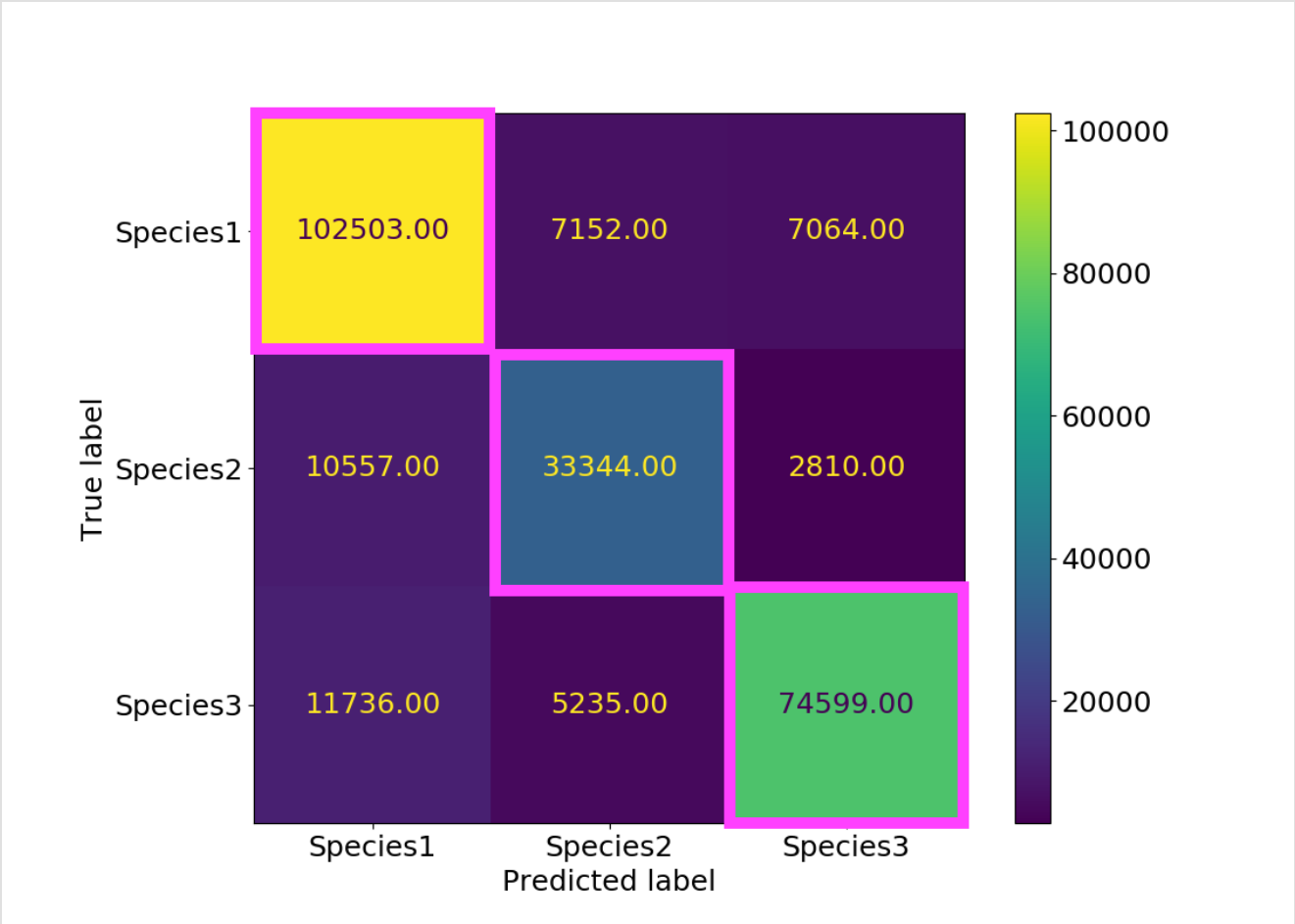
- Look again at the MLP from lecture part II
- The unnormalized confusion matrix is:



- The trace of this matrix is: TR = 102,503 + 33,344 + 74,599 = 210,446
- The total number of events in the given data set is: #Events = 255,000

Accuracy for the MLP

- Look again at the MLP from lecture part II
- The unnormalized confusion matrix is:



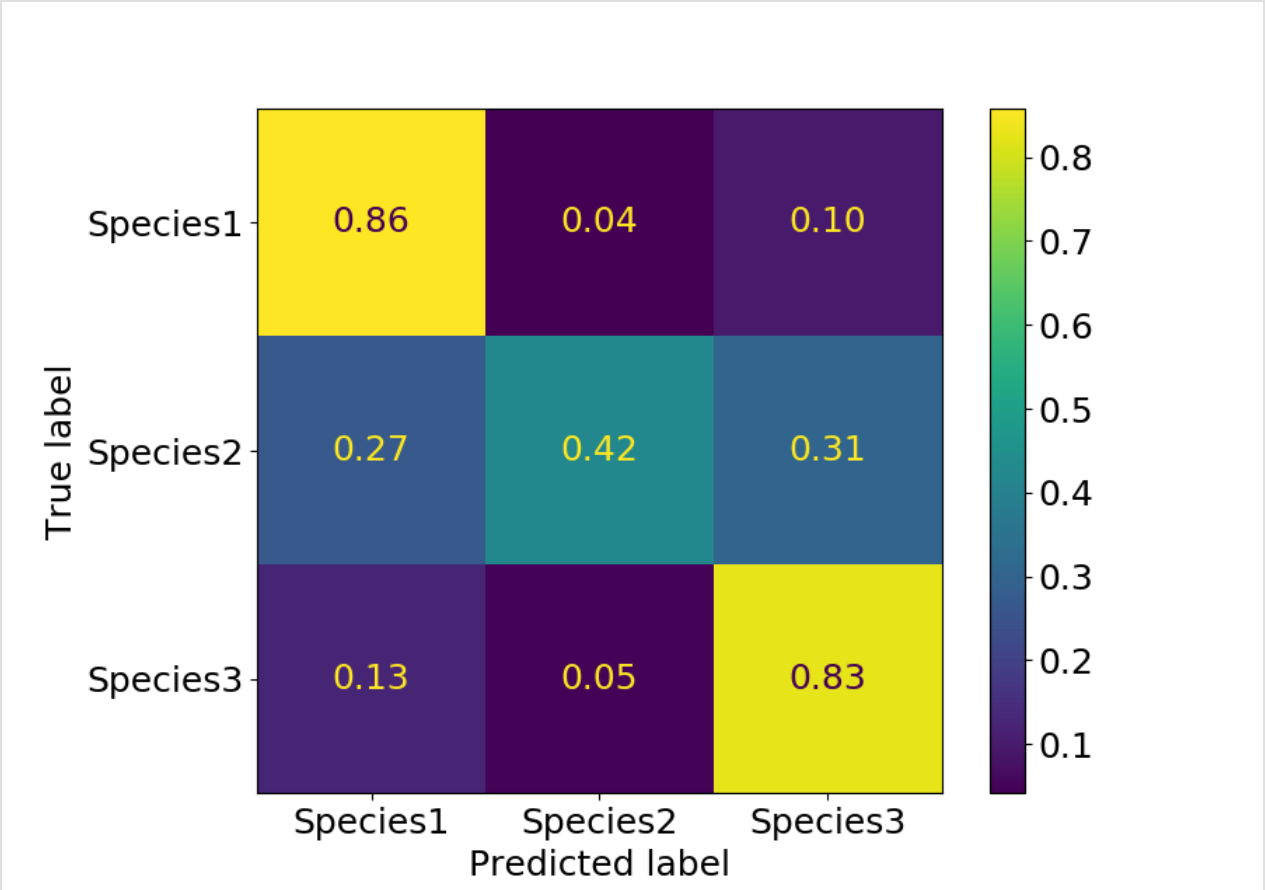
- The trace of this matrix is: TR = 102,503 + 33,344 + 74,599 = 210,446
- The total number of events in the given data set is: #Events = 255,000
- Using Eq. 4 from slide 14 yields: Accuracy = TR / #Events ≈ 82%
  → consistent with the value reported by scikit

Accuracy for the Linear SVM

- Now look at linear SVM with a normalized confusion matrix

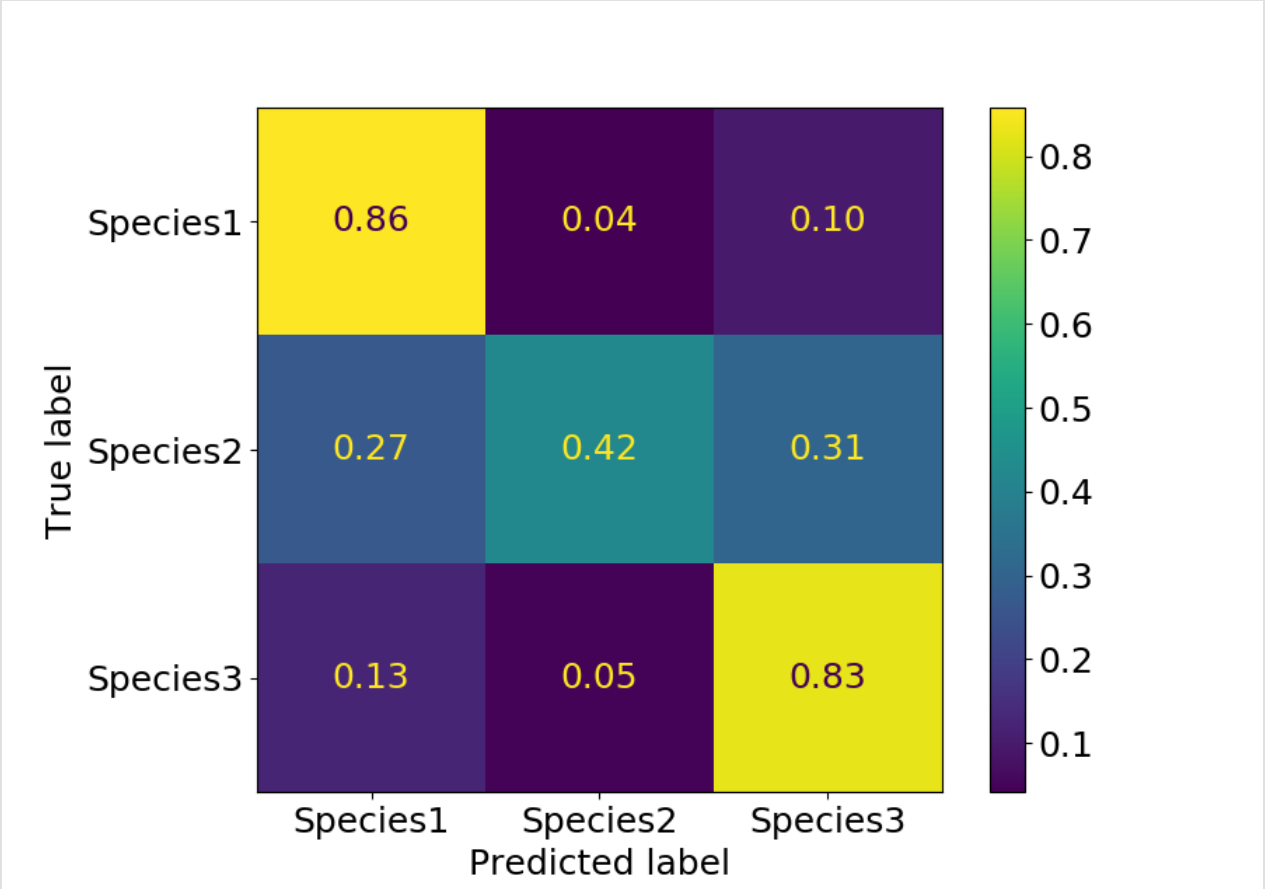
Accuracy for the Linear SVM

- Now look at linear SVM with a normalized confusion matrix



Accuracy for the Linear SVM

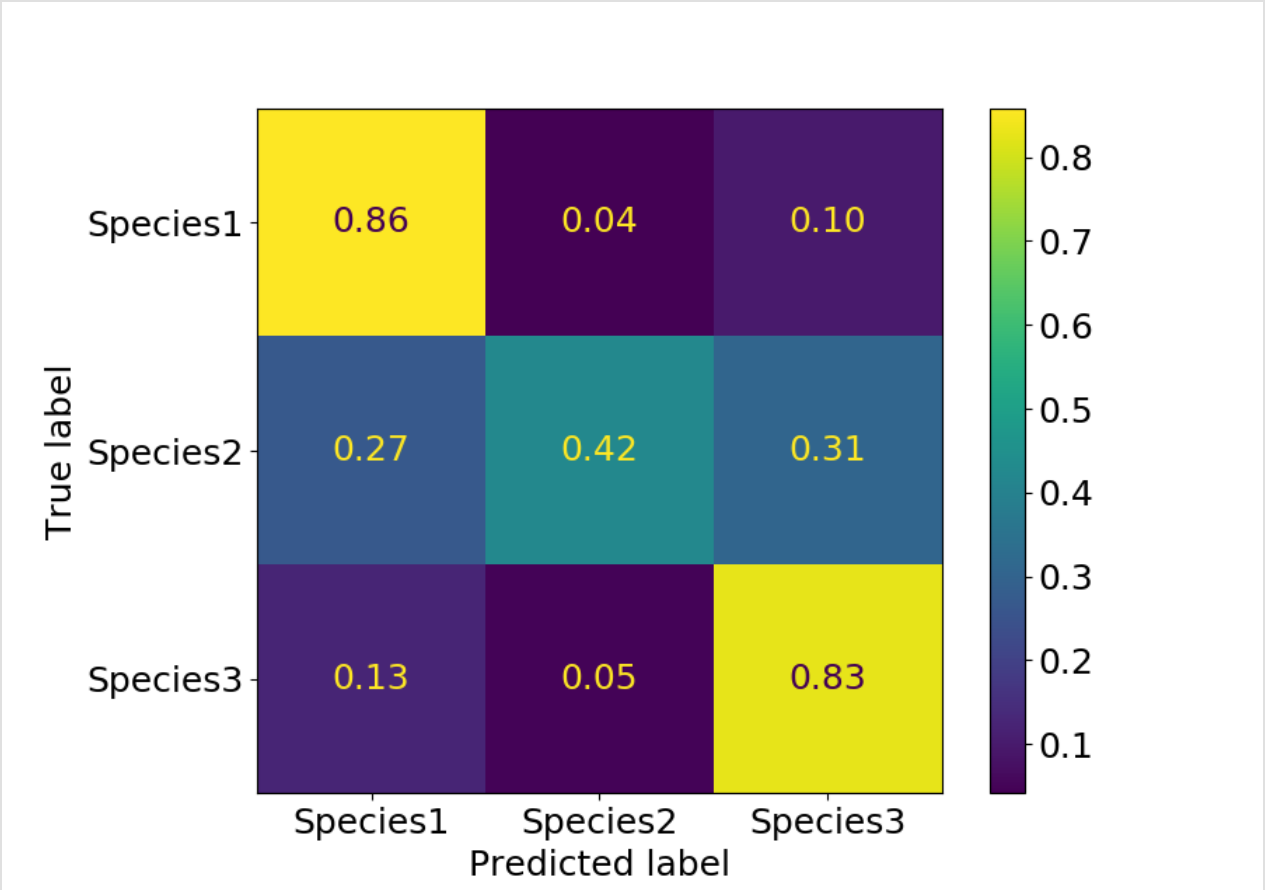
- Now look at linear SVM with a normalized confusion matrix



- In this case we need to compute the weighted trace (Eq. 6 on slide 14)

Accuracy for the Linear SVM

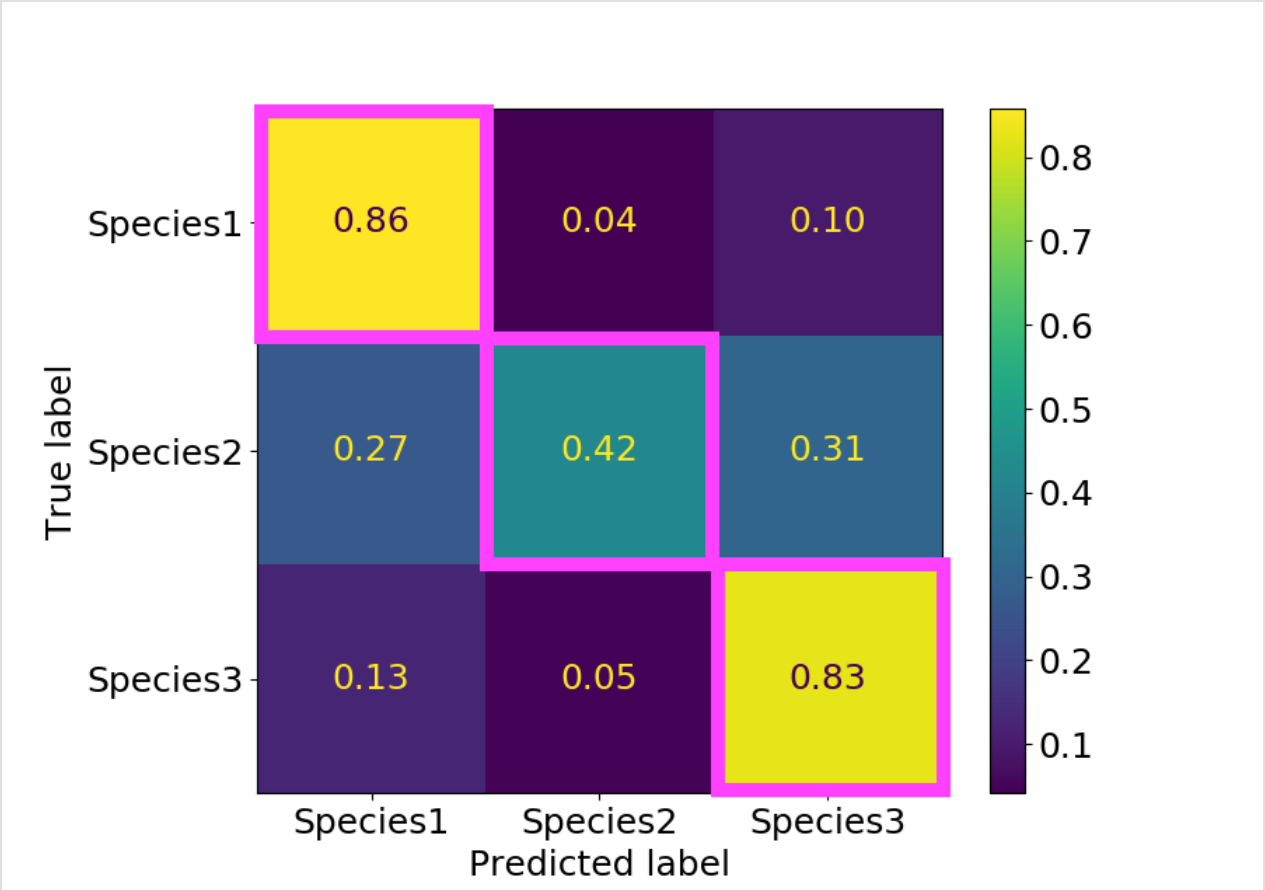
- Now look at linear SVM with a normalized confusion matrix



- In this case we need to compute the weighted trace (Eq. 6 on slide 14)
- The relative abundances are: $R(1) = 0.46$, $R(2) = 0.18$ and $R(3) = 0.36$

Accuracy for the Linear SVM

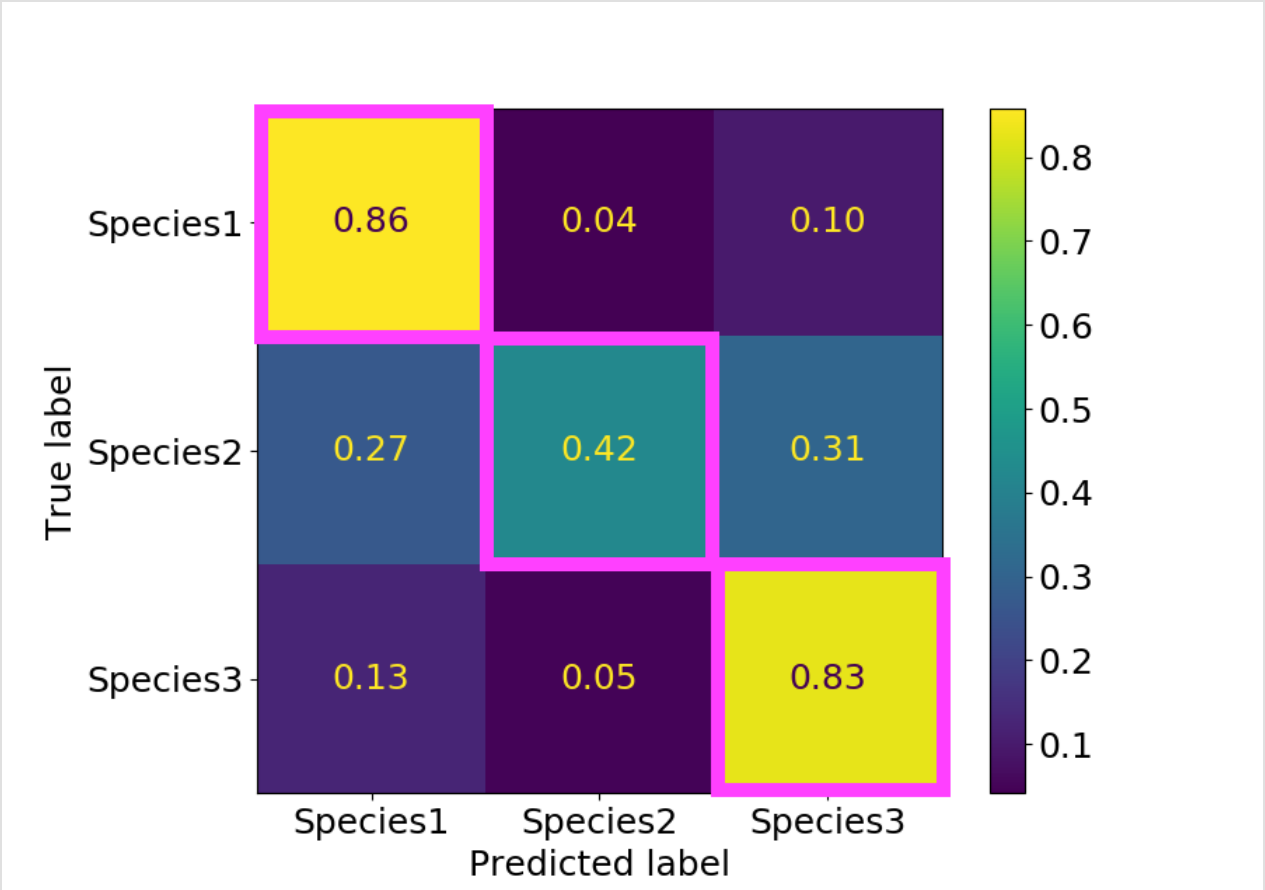
- Now look at linear SVM with a normalized confusion matrix



- In this case we need to compute the weighted trace (Eq. 6 on slide 14)
- The relative abundances are: $R(1) = 0.46$, $R(2) = 0.18$ and $R(3) = 0.36$
- Which leads to the Accuracy $= 0.46 \cdot 0.86 + 0.18 \cdot 0.42 + 0.36 \cdot 0.83 \approx 77\%$

Accuracy for the Linear SVM

- Now look at linear SVM with a normalized confusion matrix



- In this case we need to compute the weighted trace (Eq. 6 on slide 14)
- The relative abundances are: $R(1) = 0.46$, $R(2) = 0.18$ and $R(3) = 0.36$
- Which leads to the Accuracy $= 0.46 \cdot 0.86 + 0.18 \cdot 0.42 + 0.36 \cdot 0.83 \approx 77\%$
- Note that contribution for species 2 is suppressed due to R(2)

The Matthews Correlation Coefficient (MCC)

- Like the accuracy, the MCC can be computed from entries within the (unnormalized) confusion matrix:[8]

$$\mathrm{MCC} = \frac{\sum_k \sum_l \sum_m (C_{kk} C_{lm} - C_{kl} C_{mk})}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k' \neq k} \sum_{l'} C_{k'l'})} \cdot \sqrt{\sum_k (\sum_l C_{lk})(\sum_{k' \neq k} \sum_{l'} C_{k'l'})}} \tag{7}$$

- In a very simplified picture, the MCC combines the true positive rate, false positive rate and purity[9]
- The MCC is a common / preferable choice for imbalanced data

[8] Formula taken from **wikipedia**

[9] This is exact true for binary classification

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

- Note the relative difference between the individual accuracy values / mcc values

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

- Note the relative difference between the individual accuracy values / mcc values
- So, which metric to use?

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

- Note the relative difference between the individual accuracy values / mcc values
- So, which metric to use?
- Well, it depends...

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

- Note the relative difference between the individual accuracy values / mcc values
- So, which metric to use?
- Well, it depends...
  - ... on the data set you are looking at
    (imbalanced, balanced, specifically looking at one species,...)

Example

- Compare performance metrics of mlp and linear svm on the given data set

Model	F1-Score	Accuracy	MCC
MLP	0.8	0.82	0.72
Linear SVM	0.71	0.77	0.62

- Note the relative difference between the individual accuracy values / mcc values
- So, which metric to use?
- Well, it depends...
  - ... on the data set you are looking at (imbalanced, balanced, specifically looking at one species,...)
  - ... on which information you would like to retrieve (e.g. care more about purity, or efficiency, or overall classification performance?)

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training
- **Examples:**

Model	Fit Parameter	Hyper Parameter
$pol(N) = p_N x^N + p_{N-1} x^{N-1} + \cdots + p_0$	$p_N, \cdots, p_0$	N, fit iterations,...

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training
- **Examples:**

Model	Fit Parameter	Hyper Parameter
$pol(N) = p_N x^N + p_{N-1} x^{N-1} + \cdots + p_0$	$p_N, \cdots, p_0$	N, fit iterations,...
Multilayer Perceptron	Biases, Weights	#Neurons, #Layers, #Epochs, learning rate, ...

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training
- **Examples:**

Model	Fit Parameter	Hyper Parameter
$pol(N) = p_N x^N + p_{N-1} x^{N-1} + \cdots + p_0$	$p_N, \cdots, p_0$	N, fit iterations,...
Multilayer Perceptron	Biases, Weights	#Neurons, #Layers, #Epochs, learning rate, ...
Random Forest	Thresholds in trees	#Trees, depth of each tree,...

Hyper Parameter Optimization (HPO)

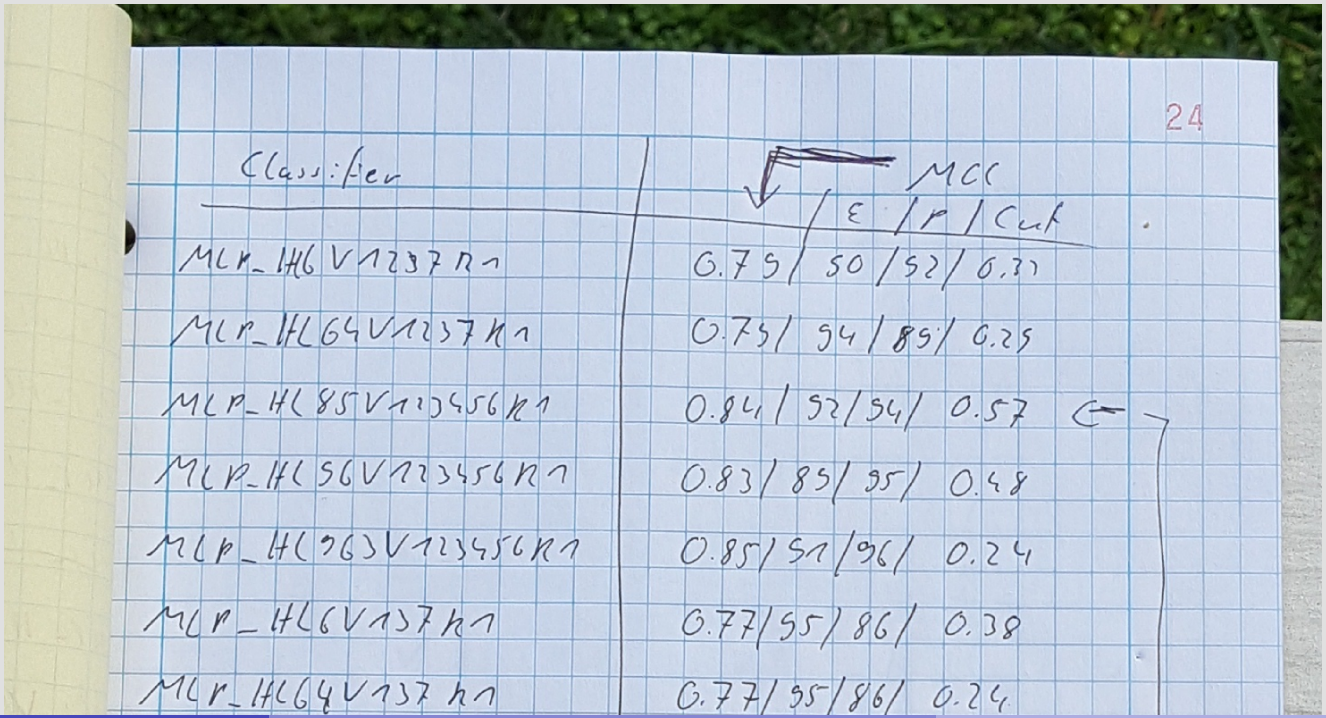
- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training

- Main goal of HPO:
  - Find the set of model parameters which minimizes the loss / error
  - Fulfill given constraints (i.e. avoid overfitting or demand minimal computing time)

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training

- Main goal of HPO:
  - Find the set of model parameters which minimizes the loss / error
  - Fulfill given constraints (i.e. avoid overfitting or demand minimal computing time)
- Manual tuning your model
  - is considered to be HPO
  - Painful



Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training

- Main goal of HPO:
  - Find the set of model parameters which minimizes the loss / error
  - Fulfill given constraints (i.e. avoid overfitting or demand minimal computing time)
- Manual tuning your model
  - is considered to be HPO
  - Painful
- There are many algorithms / frameworks on the market:
  - Parameter grid search: Simple, but ineffective (helpful when you know the parameter ranges)
  - Random parameter search
  - Bayesian optimization: More sophisticated
  - ...

Hyper Parameter Optimization (HPO)

- Hyper parameters: Parameters that determine the architecture of your model and its performance during / after training

- Main goal of HPO:
  - Find the set of model parameters which minimizes the loss / error
  - Fulfill given constraints (i.e. avoid overfitting or demand minimal computing time)
- Manual tuning your model
  - is considered to be HPO
  - Painful
- There are many algorithms / frameworks on the market:
  - Parameter grid search: Simple, but ineffective (helpful when you know the parameter ranges)
  - Random parameter search
  - Bayesian optimization: More sophisticated
  - ...
- No matter which method you decide to use $\rightarrow$ Always include a validation data set

Parameter Grid Search for a MLP

- Look again at the classification data (three species, defined by three features each)

[10] Or minimizes the corresponding loss

Parameter Grid Search for a MLP

- Look again at the classification data (three species, defined by three features each)
- **Goal:** Try to find the mlp parameter setting which maximizes[10] the classification performance on the given data set

[10] Or minimizes the corresponding loss

Parameter Grid Search for a MLP

- Look again at the classification data (three species, defined by three features each)
- **Goal:** Try to find the mlp parameter setting which maximizes[10] the classification performance on the given data set
- **Approach:** Use grid search $\rightarrow$ Vary a given set of hyper parameters within specified ranges $\rightarrow$ Test ALL possible combinations of hyper parameter settings

[10] Or minimizes the corresponding loss

Parameter Grid Search for a MLP

- Look again at the classification data (three species, defined by three features each)
- **Goal:** Try to find the mlp parameter setting which maximizes[10] the classification performance on the given data set
- **Approach:** Use grid search $\rightarrow$ Vary a given set of hyper parameters within specified ranges $\rightarrow$ Test ALL possible combinations of hyper parameter settings

Hyper Parameter	Settings
Activation Function	linear, tanh, relu
Epochs	100, 200, 700
Learning Rate	0.005, 0.01, 0.02
Architecture	(5), (10), (5,3)

[10] Or minimizes the corresponding loss

Parameter Grid Search for a MLP

- Look again at the classification data (three species, defined by three features each)
- **Goal:** Try to find the mlp parameter setting which maximizes[10] the classification performance on the given data set
- **Approach:** Use grid search $\rightarrow$ Vary a given set of hyper parameters within specified ranges $\rightarrow$ Test ALL possible combinations of hyper parameter settings

Hyper Parameter	Settings
Activation Function	linear, tanh, relu
Epochs	100, 200, 700
Learning Rate	0.005, 0.01, 0.02
Architecture	(5), (10), (5,3)

$\Rightarrow$ 4 hyper parameters with 3 settings each $\rightarrow 3^4 = 81$ settings to test
$\rightarrow$ This might take a while...

[10] Or minimizes the corresponding loss

Setting up and performing the Grid Search

- Split data into training and validation sets

Setting up and performing the Grid Search

- Split data into training and validation sets

```
from sklearn.model_selection import train_test_split

#Load the DataFrame:
data = '/Volumes/BunchOfStuff/classifier_testData/fsu_ml_data3.csv'
data_df = pd.read_csv(data)

#Get features and labels:
X = data_df[['var1','var2','var3']].values
Y = data_df['label'].values

#Divide data into training and validation sets
x_train, x_val, y_train, y_val = train_test_split(
        X, #--> features
        Y, #--> targets
        test_size=0.25, #--> Percentage of data that is used for validation
        random_state=0)
```

Setting up and performing the Grid Search

- Split data into training and validation sets
- Define metric to judge the performance of your algorithm
  $\Rightarrow$ Accuracy, Purity, log-loss, Matthews Correlation Coefficient,...

Setting up and performing the Grid Search

- Split data into training and validation sets
- Define metric to judge the performance of your algorithm
  $\Rightarrow$ Accuracy, Purity, log-loss, Matthews Correlation Coefficient,...
- Setup algorithm that shall be optimized

Setting up and performing the Grid Search

- Split data into training and validation sets
- Define metric to judge the performance of your algorithm
  $\Rightarrow$ Accuracy, Purity, log-loss, Matthews Correlation Coefficient,...
- Setup algorithm that shall be optimized

```
def setup_mlp(architecture,n_epochs,act_func,l_rate):
        my_mlp = MLPClassifier(
        hidden_layer_sizes=architecture, #--> Updated for each param. setting
        activation=act_func, #--> Updated for each param. setting
        solver='sgd',
        shuffle=True,
        validation_fraction=0.1, #--> part of the training data
        #is used for early stopping
        early_stopping=True,
        max_iter = n_epochs, #--> Updated for each param. setting
        learning_rate_init=l_rate, #--> Updated for each param. setting
        learning_rate='invscaling', #--> Decreasing learning rate
        #will be discussed later
        warm_start=True,
        tol=1e-6
    )

    return my_mlp
    #--> In our example, this function will be called 81 times...
```

Setting up and performing the Grid Search

- Split data into training and validation sets
- Define metric to judge the performance of your algorithm
  $\Rightarrow$ Accuracy, Purity, log-loss, Matthews Correlation Coefficient,...
- Setup algorithm that shall be optimized
- Loop over all 81 hyper parameter settings and

a) Train the mlp on the training data

b) Determine performance of mlp on validation data

Setting up and performing the Grid Search

- Split data into training and validation sets
- Define metric to judge the performance of your algorithm
  $\Rightarrow$ Accuracy, Purity, log-loss, Matthews Correlation Coefficient,...
- Setup algorithm that shall be optimized
- Loop over all 81 hyper parameter settings and

a) Train the mlp on the training data

b) Determine performance of mlp on validation data

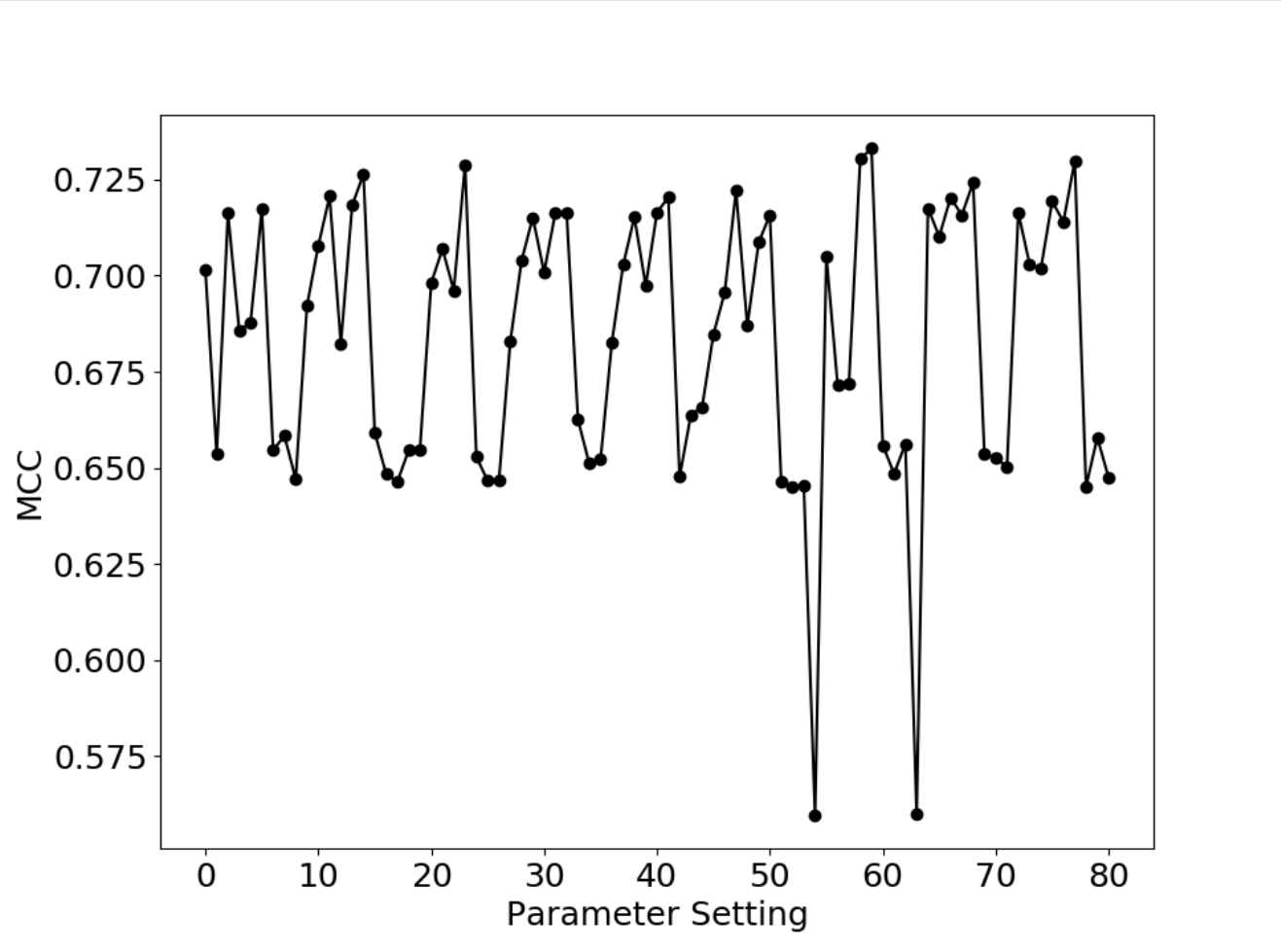
- Finally, pick algorithm that shows highest[11] performance

[11] In some cases not the best choice

Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting

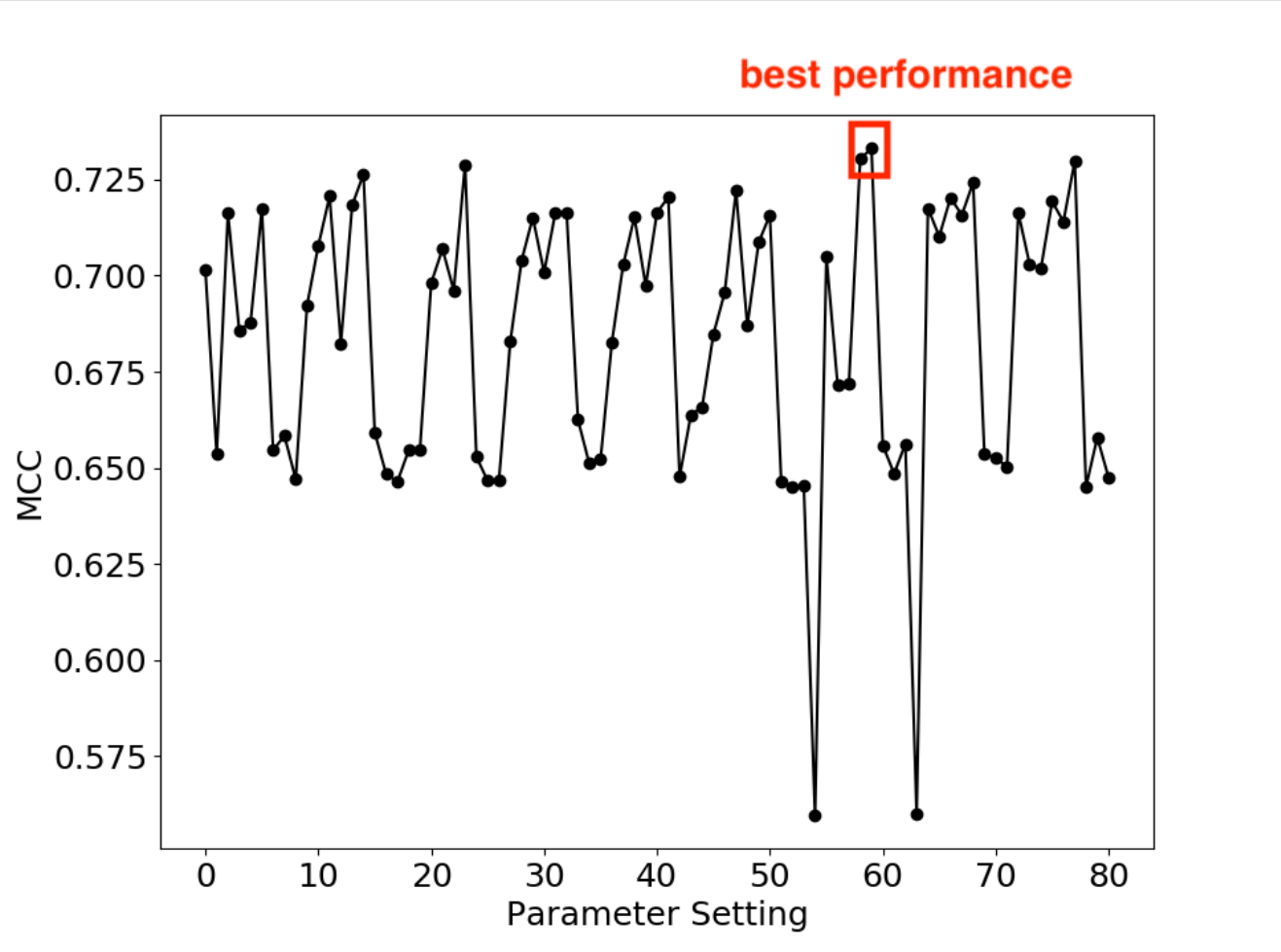
  $\leftrightarrow$ Understand model behavior with respect to parameter settings / ranges



Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting

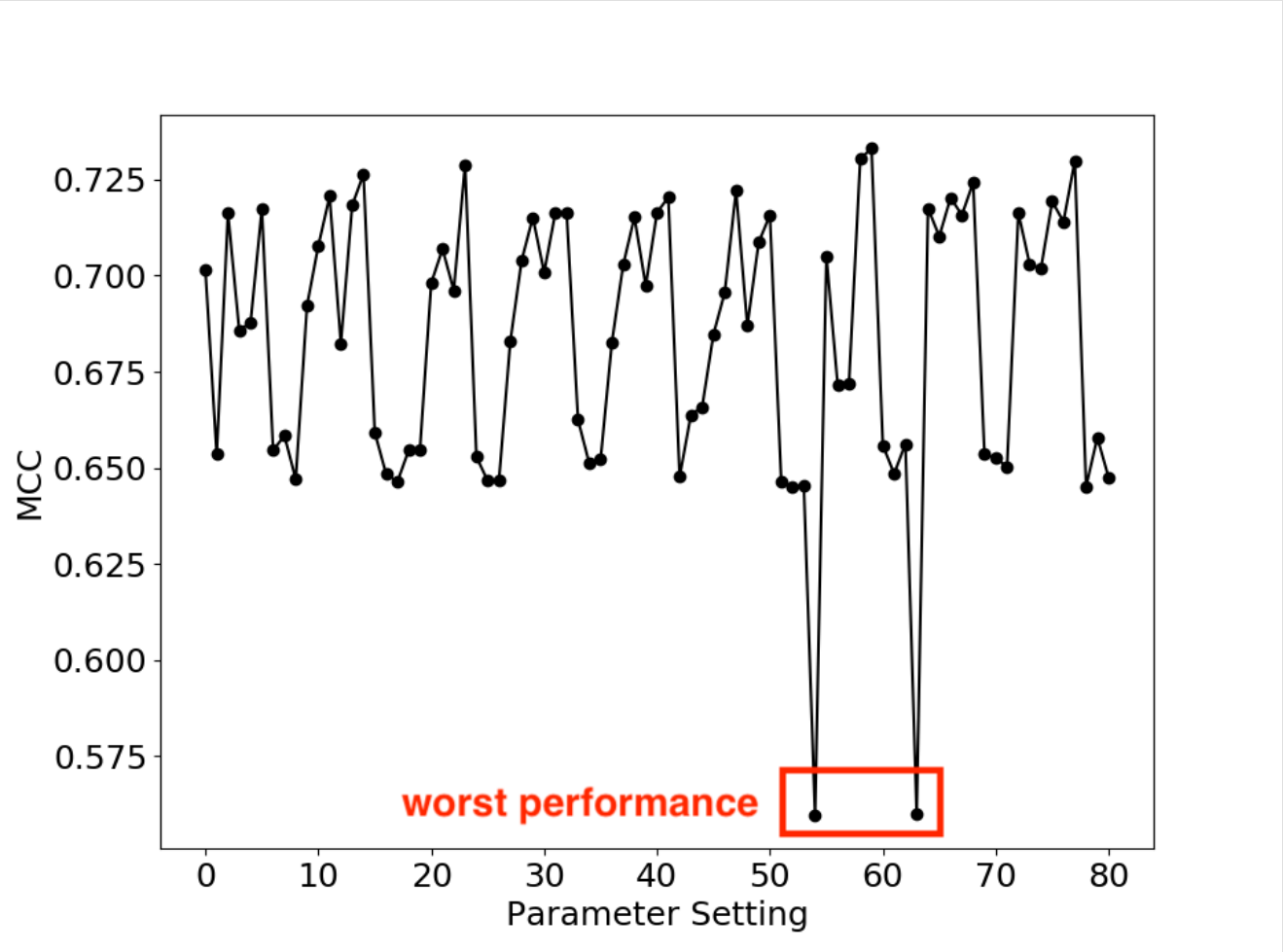
  $\leftrightarrow$ Understand model behavior with respect to parameter settings / ranges



Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting

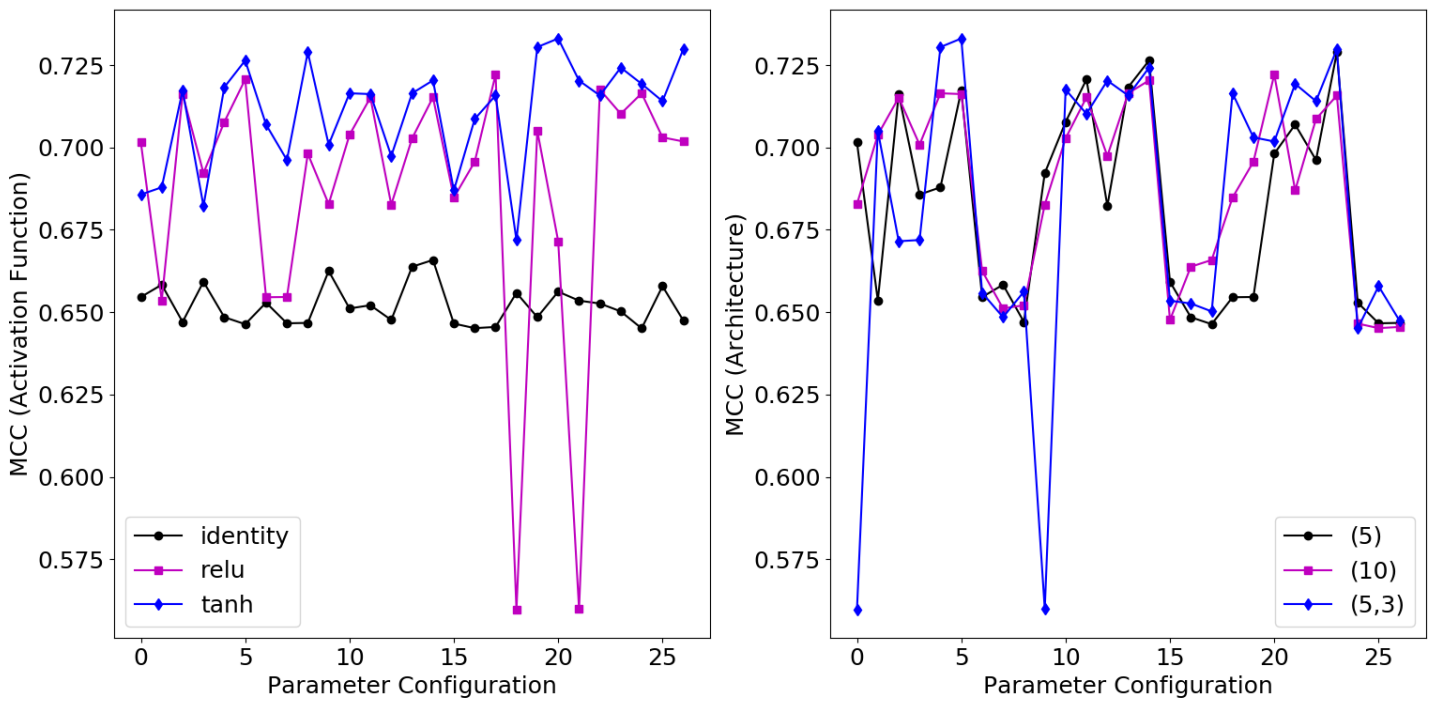
  $\leftrightarrow$ Understand model behavior with respect to parameter settings / ranges



Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings

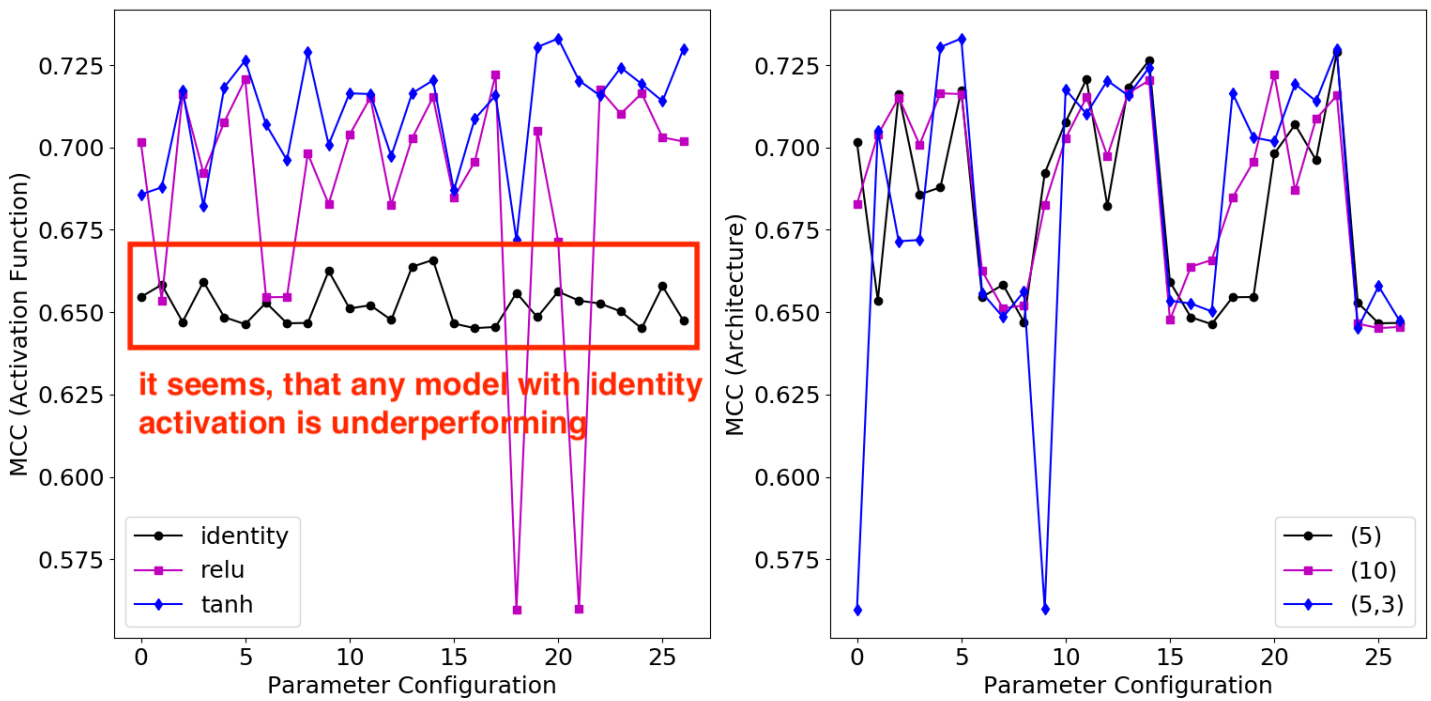
  $\leftrightarrow$ Study significance of parameter with respect to performance



Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings

  $\leftrightarrow$ Study significance of parameter with respect to performance



Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings
- Compare individual models

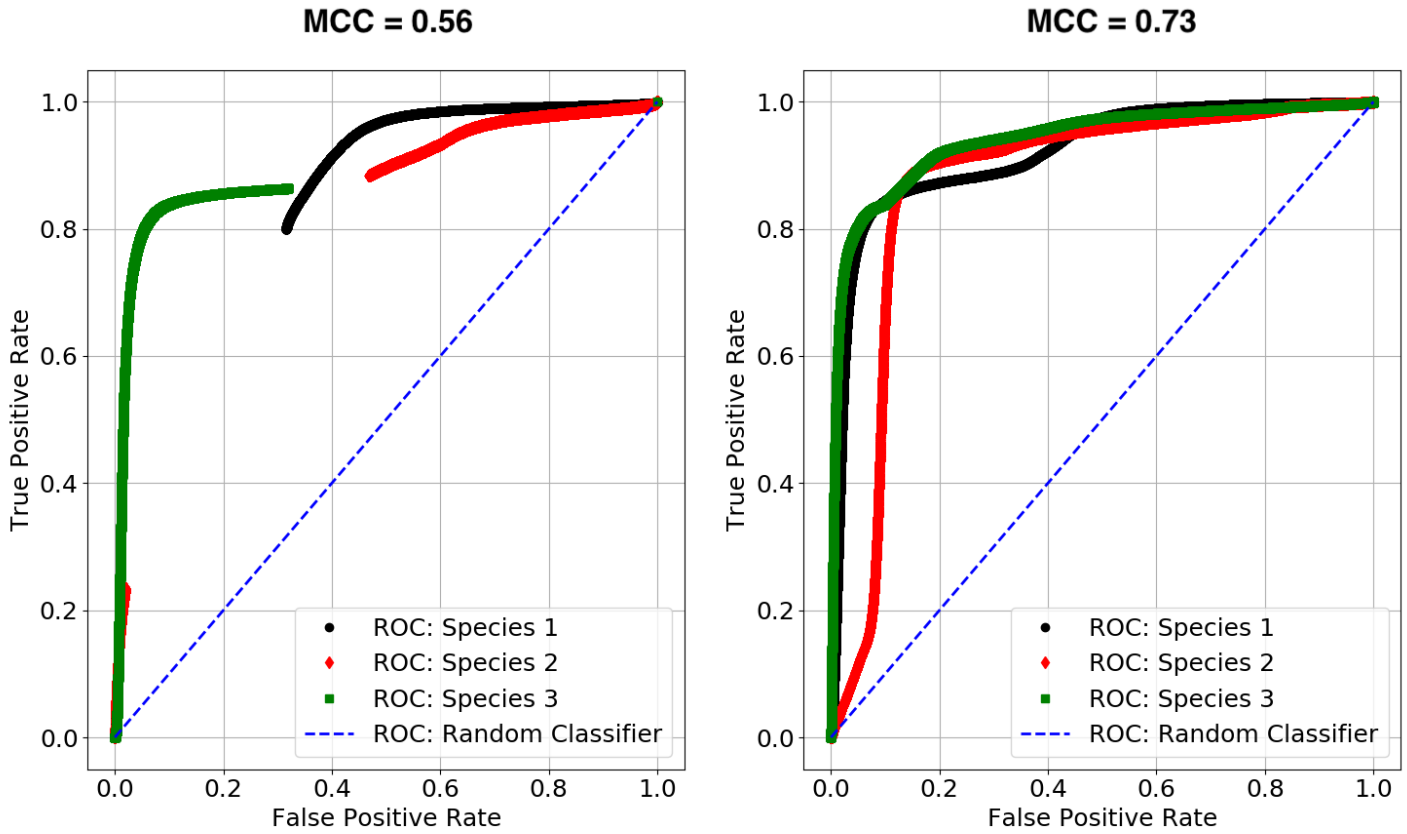
  $\leftrightarrow$ Look for striking differences

Architecture	Max. # Epochs	Learning rate	Act. Func.	MCC
(5,3)	100	0.005	relu	0.56
(5,3)	100	0.05	tanh	0.73

[12] Use the Matthews Correlation Coefficient (MCC) here

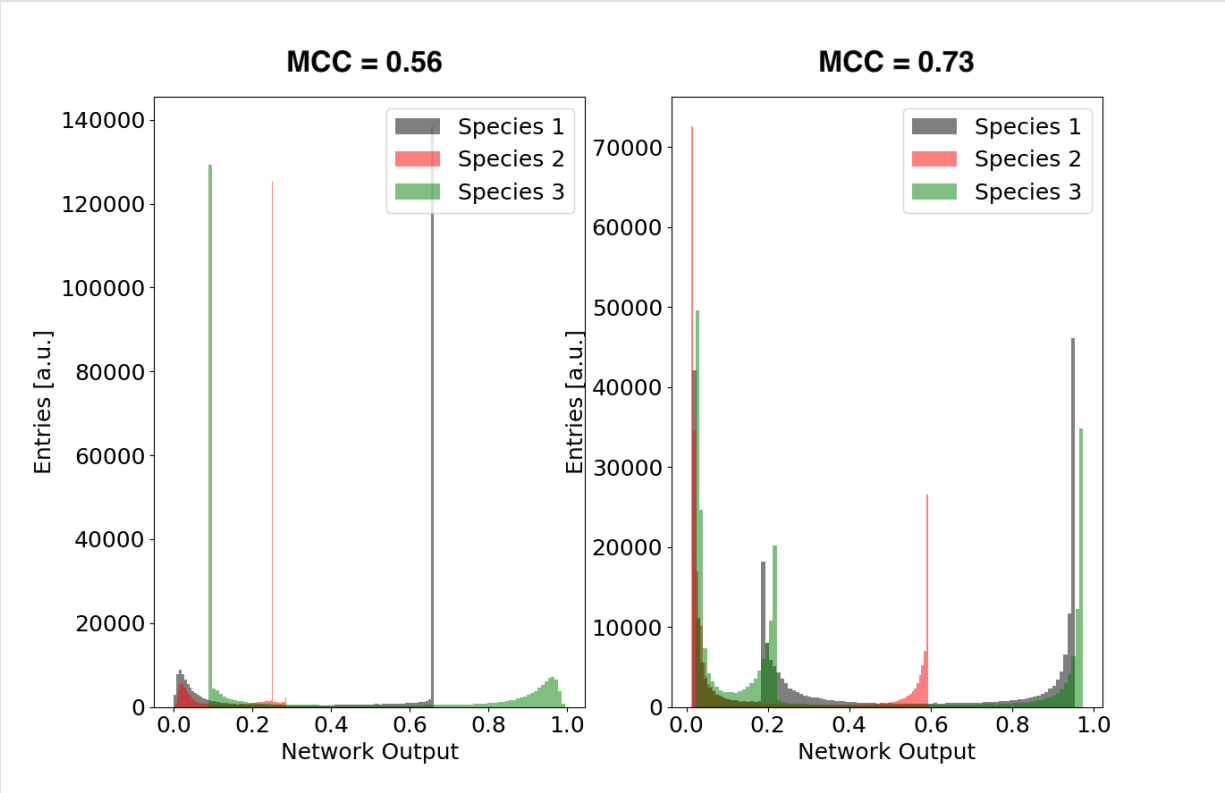
Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings
- Compare individual models
- Compare ROC-curves



Results of MLP Parameter Grid Search

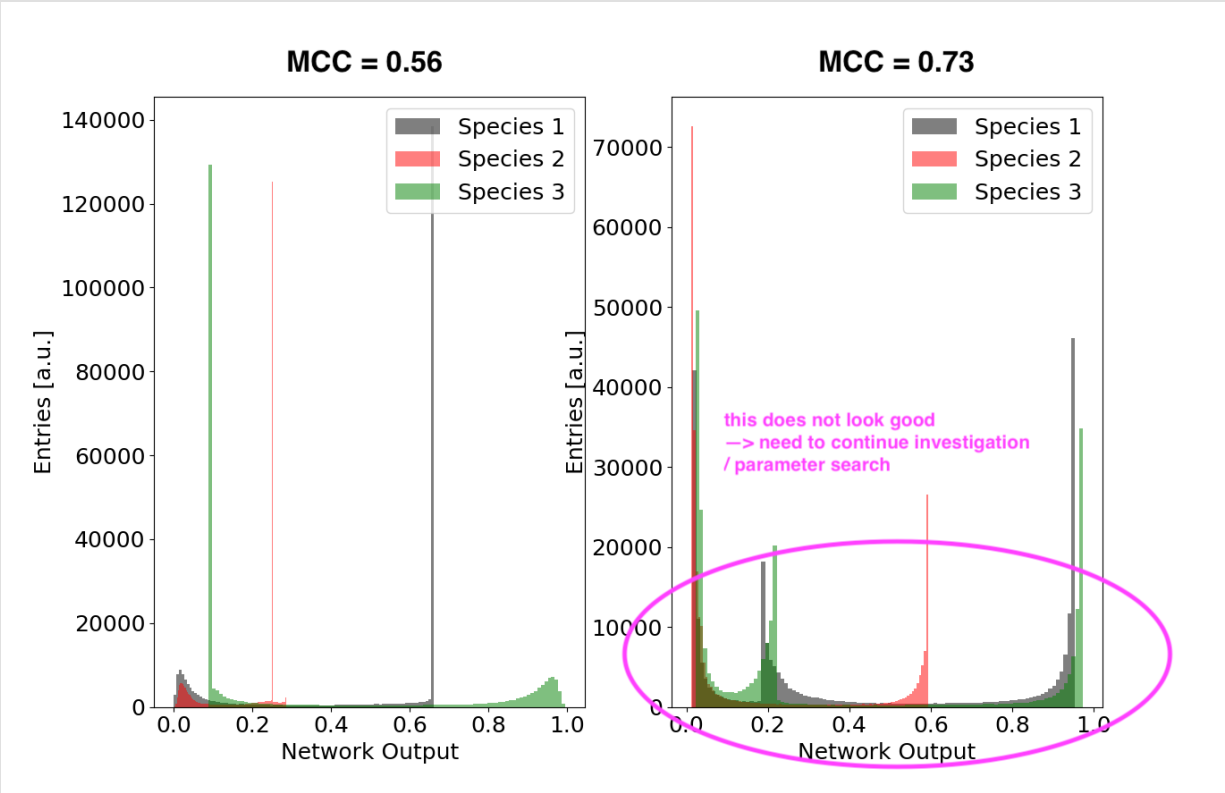
- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings
- Compare individual models
- Compare ROC-curves
- And ALWAYS check the model response



[12] Use the Matthews Correlation Coefficient (MCC) here

Results of MLP Parameter Grid Search

- Check performance[12] vs. parameter setting
- Look at performance for specific hyper parameter settings
- Compare individual models
- Compare ROC-curves
- And ALWAYS check the model response



[12] Use the Matthews Correlation Coefficient (MCC) here

Summary Parameter Grid Search

- Performed parameter grid search on mlp

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:
  - Number of parameter tests = $\prod_{hp=1}^{\#\text{Hyper Params}} (\#\text{Settings for } hp)$

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:
  - Number of parameter tests = $\prod_{hp=1}^{\#\text{Hyper Params}} (\#\text{Settings for } hp)$
  - (Computing) time consuming, depending on the number of searches

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:
  - Number of parameter tests = $\prod_{hp=1}^{\#\text{Hyper Params}} (\#\text{Settings for } hp)$
  - (Computing) time consuming, depending on the number of searches
  - No guarantee that optimal model is found within the specified limits

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:
  - Number of parameter tests = $\prod_{hp=1}^{\#\text{Hyper Params}} (\#\text{Settings for } hp)$
  - (Computing) time consuming, depending on the number of searches
  - No guarantee that optimal model is found within the specified limits

$\Rightarrow$ It might happen that you search for a long time, without finding the proper settings

Summary Parameter Grid Search

- Performed parameter grid search on mlp
- Checked 81 different configurations $\rightarrow$ took $\sim$ 10min on my laptop
- Found "winning" model with MCC = 0 .73, but response function shows weird structures
- Grid search is simple and easy to implement, but:
  - Number of parameter tests = $\prod_{hp=1}^{\#\text{Hyper Params}} (\#\text{Settings for } hp)$
  - (Computing) time consuming, depending on the number of searches
  - No guarantee that optimal model is found within the specified limits

$\Rightarrow$ It might happen that you search for a long time, without finding the proper settings

- Grid search is helpful, if you now your model pretty well and / or have somewhat narrowed down the individual parameter limits $\Rightarrow$ Fine tuning

Summary Part III

- Briefly discussed two more classification algorithms
  - Random Forest classifier
  - Linear support vector machine
- Discussed performance evaluation metrics
  - Purity, F1-Score, accuracy and MCC
  - These are NOT all available metrics $\rightarrow$ There are many more which are frequently used (i.e. AUC)
- Hyper Parameter Optimization (HPO)
  - Tune model on given data set
  - Different approaches available
  - Discussed simple grid search
- Next part:
  - A few more words on HPO
  - Transition to deep learning