Project - Advanced Statistics for Physics Analysis

Multivariate analysis in particle physics and separation of signal from background using a Deep Neural Network

Student

Fabio Semenzato

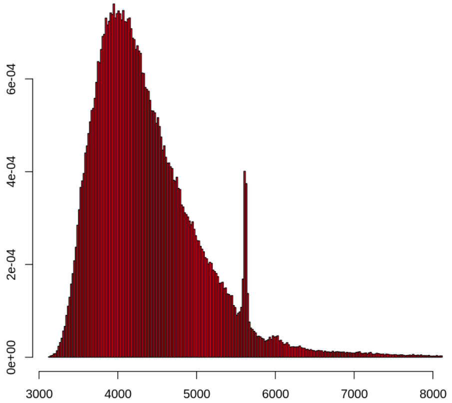
Introduction

Multivariate analysis in particle physics | and separation of signal from background using a Deep Neural Network

- Lots of data
- Lots of features

- Many background data
- Few interesting events

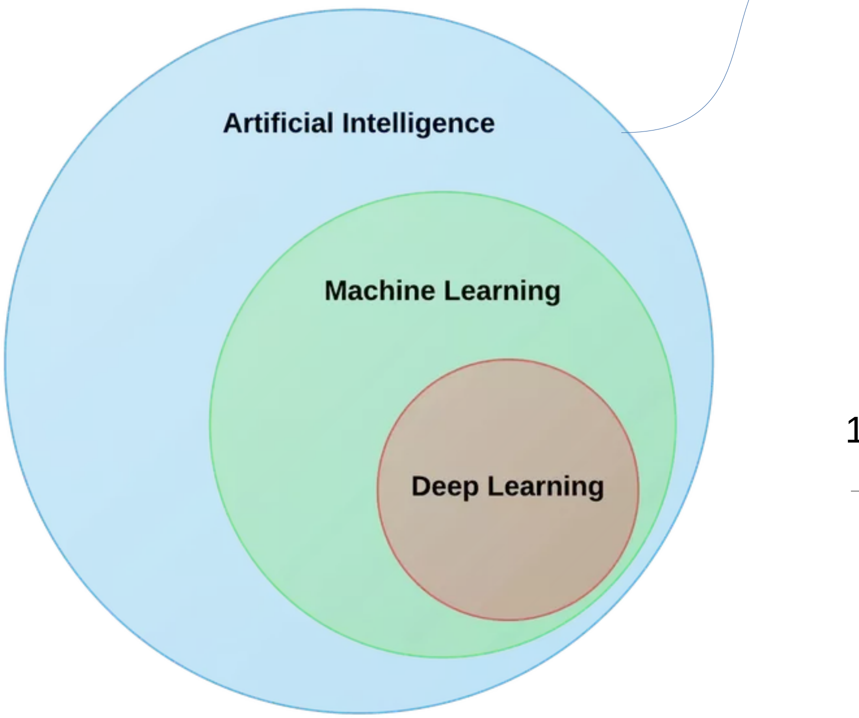
	lcstar_MM_F	Lambda_c_MM_F	Lambda_b0_MM_F	lc_p_ProbNNp_F	lcZDecLSigma_F	lcstarZDecLSigma_F	lcDecTime_F	lcstarDecTime_F	lbDecTime_F
0	2667.468	2308.289	4500.151	0.9987966	3.7409363	0.64662420	0.3047157	0.10524536	1.146613
1	2697.567	2308.289	4531.314	0.9987966	3.6852982	0.25769106	0.2985286	0.03501043	1.157879
2	2670.196	2308.289	4521.702	0.9987966	3.7632043	0.22716732	0.3074197	0.04779404	1.167280
3	2698.279	2308.289	4551.235	0.9987966	3.7061355	-0.12590024	0.3010732	-0.02033100	1.178410
4	2800.471	2286.779	5256.411	0.9909659	6.3935475	-0.64229697	0.3180268	-0.05100102	3.035080
5	2741.149	2279.552	4214.323	0.9739404	4.1540904	-1.19302950	0.6868992	-0.28974180	1.920715
6	2776.752	2279.670	4418.528	0.7637424	-1.8190205	0.46489272	-0.3238385	0.10154445	1.491546
7	2776.752	2279.670	4813.781	0.7637424	-2.1368237	-0.06115993	-0.4398269	-0.01484417	1.371934
8	2776.752	2279.670	4540.744	0.7637424	-1.7646810	0.36948892	-0.3399683	0.08543953	1.302051
9	2704.588	2285.433	4292.723	0.9611492	0.8521455	-1.03829300	0.1111045	-0.26846078	3.279141



Objectives

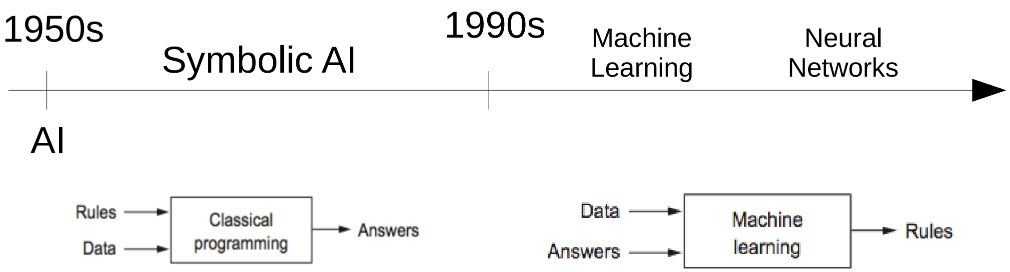
- Find the best fraction of background samples to be used during training
- Find the best parameters to be used for the Boosted Decision Tree
- Find the best parameters to be used for the Deep Neural Network
- Predict the labels of the data acquired in the experiment

Artificial Intelligence



“The effort to automate intellectual tasks normally performed by humans.”

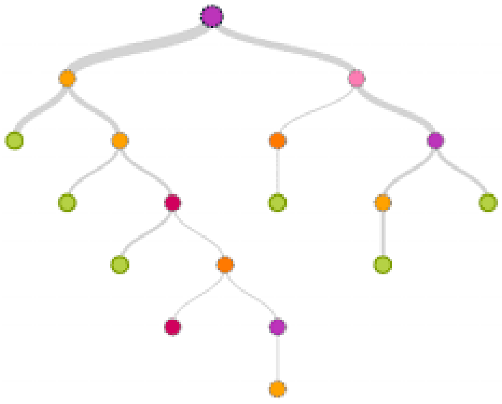
F. Chollet



Decision Tree

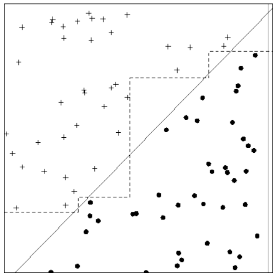
Steps:

- Find best separation line among features and values
- In each branch thus created, the process is repeated until termination condition
- At the end of each termination branch there's a leaf, telling to which group the sample belongs



Cons:

- Decision trees love orthogonal decision boundaries



Boosting

"Boosting […] refers to any Ensemble method that can combine severak weak learners into a strong learner."

A. Gèron

Wisdom of the crowd

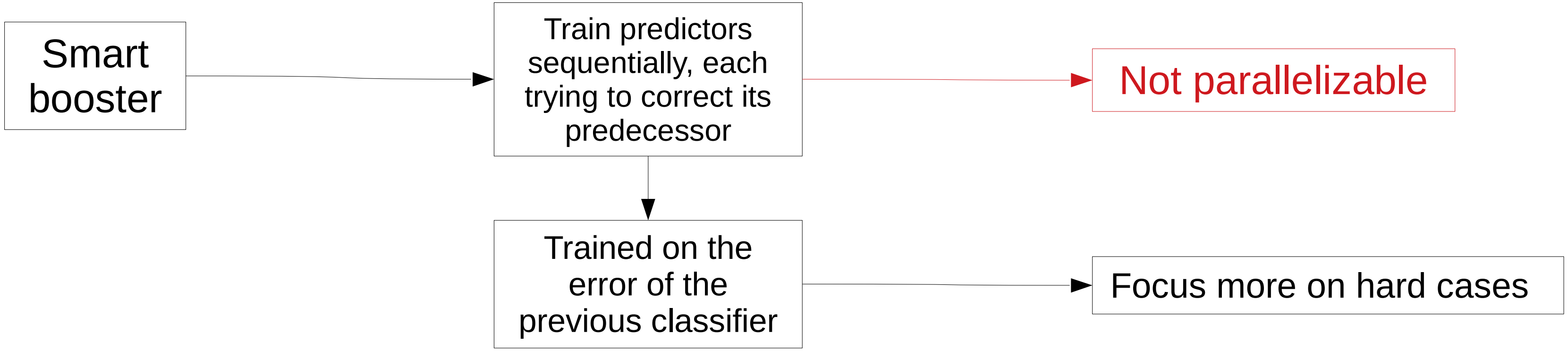
Blind boosters:

- Random forest classifier

Smart boosters: → Cannot be parallelized

- AdaBoost
- Gradient boosting

Smart Boosters



Adapting Boosting (AdaBoost):

- Increase loss weights of the sample misclassified by the previous classifier
- In the final prediction, predictors have weights depending on their overall accuracy on the training set

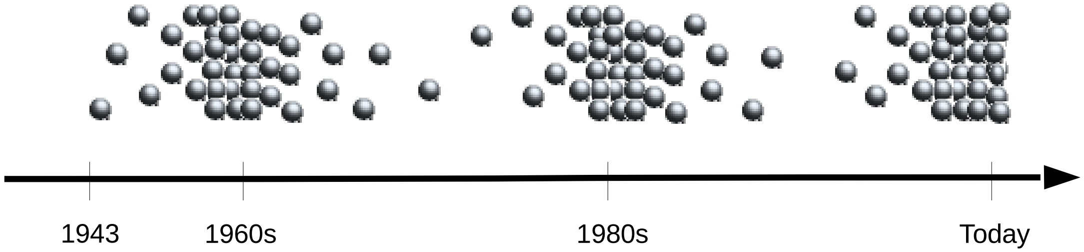
Gradient Boosting:

- Trains predictors on misclassified samples by the previous classifier

Neural Networks

People and Funds
over time

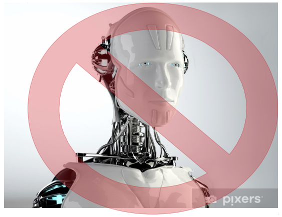
Initially based on brain's architecture



Nowadays whole different thing

1943: A neurophysiologist and a matematician introduce the first Artificial Neural Network

1960s:

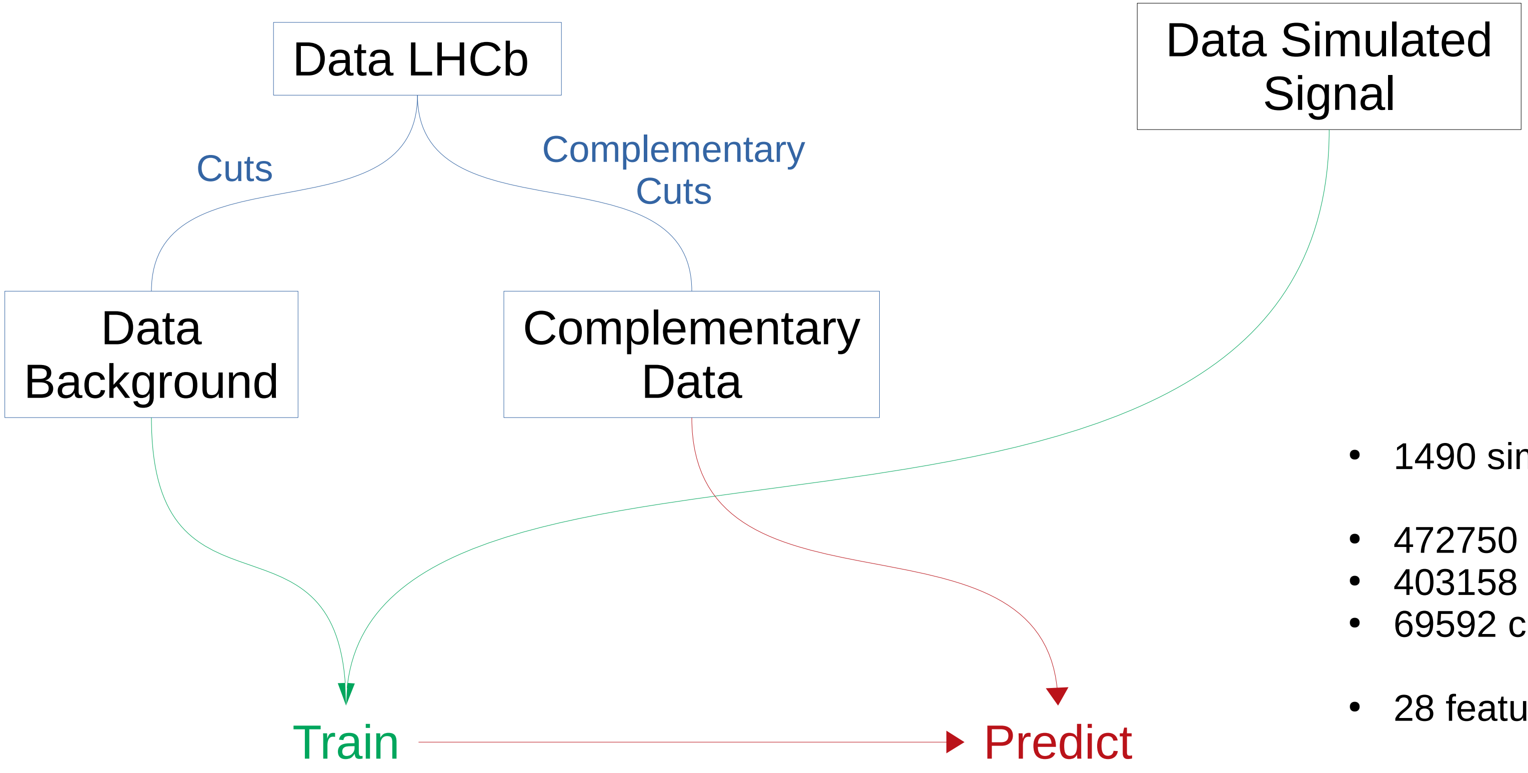


1980s: SVM
Random Forest

Neural networks

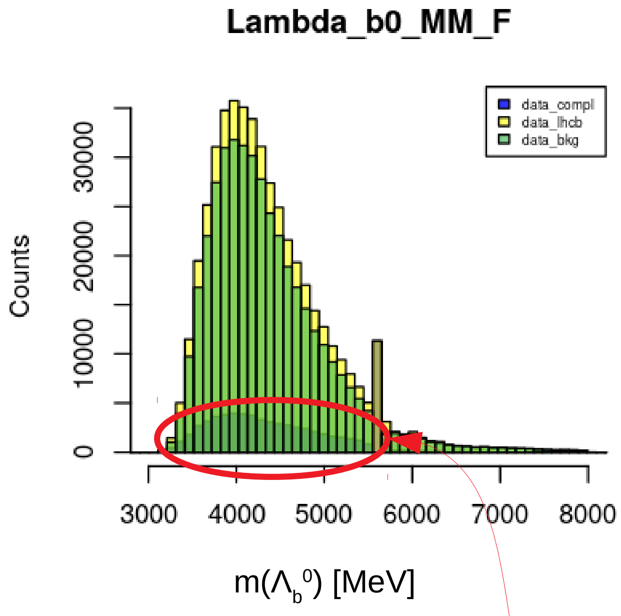
Nowadays:
- More data avilable
- More computing power
- Training algorithms improves
-More funds

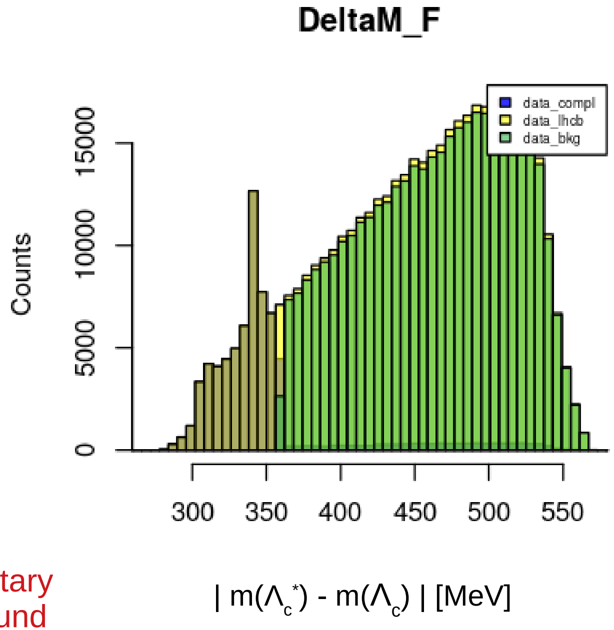
Data description



- 1490 simulated signal data
- 472750 LHCb data
- 403158 background data
- 69592 complementary data
- 28 features

Data cuts





$m(\Lambda_b^0) < 5550$ MeV

or

$m(\Lambda_b^0) > 5680$ MeV

N.B.: many complementary data belong to background category

Cuts have been joined through “and” operator instead of “or”

$| m(\Lambda_c^*) - m(\Lambda_c) | > 360$ MeV

Unusable Features

Don’t represent the complementary data

“Broken” features:

- Lambda_b0_BKGCAT_F
- lcstar_BKGCAT_F
- Lambda_c_BKGCAT_F

Background

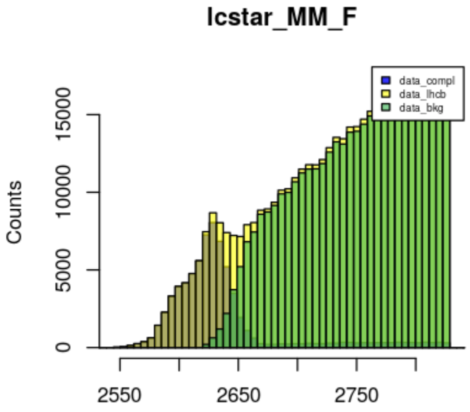
	pair1_3pi_F	Lambda_b0_BKGCAT_F	lcstar_BKGCAT_F	Lambda_c_BKGCAT_F	au_pion0_ProbNNpi_F
0	852.4460	4.5915e-41	2.897943e-08	4.5915e-41	0.8897798
1	852.4460	4.5915e-41	2.897943e-08	4.5915e-41	0.8897798
2	852.4460	4.5915e-41	2.897943e-08	4.5915e-41	0.8897798
3	878.7152	4.5915e-41	2.897943e-08	4.5915e-41	0.5428184
4	236.9293	4.5915e-41	2.897943e-08	4.5915e-41	0.9975619
5	716.9175	4.5915e-41	2.897943e-08	4.5915e-41	0.9868126
6	603.5337	4.5915e-41	2.897943e-08	4.5915e-41	0.9437394
7	603.5337	4.5915e-41	2.897943e-08	4.5915e-41	0.9868126
8	234.9984	4.5915e-41	2.897943e-08	4.5915e-41	0.9679128
9	299.0599	4.5915e-41	2.897943e-08	4.5915e-41	0.9481228

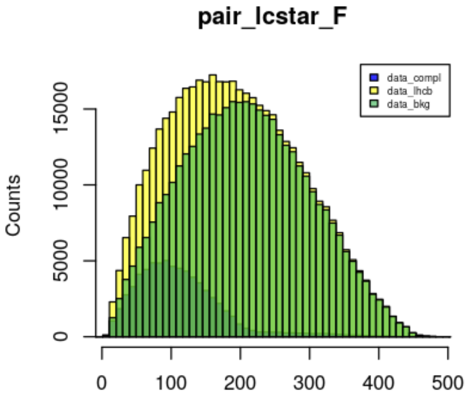
Simulated signal

	pair1_3pi_F	Lambda_b0_BKGCAT_F	lcstar_BKGCAT_F	Lambda_c_BKGCAT_F	tau_pion0_ProbNNpi_F
0	754.2185	10	0	0	0.9960517
1	571.5563	10	0	0	0.9691991
2	812.8149	10	0	0	0.8418677
3	748.3558	10	0	0	0.9889790
4	566.7918	10	0	0	0.7840963
5	281.1272	10	0	0	0.9668434
6	356.0302	10	10	10	0.7380860
7	1036.1909	10	10	10	0.9932866
8	559.2961	10	10	10	0.9977948
9	848.2168	10	0	0	0.9907691

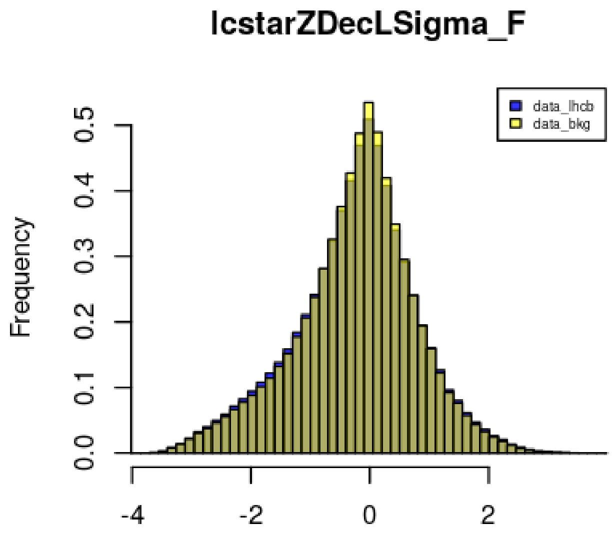
“Biased” features:

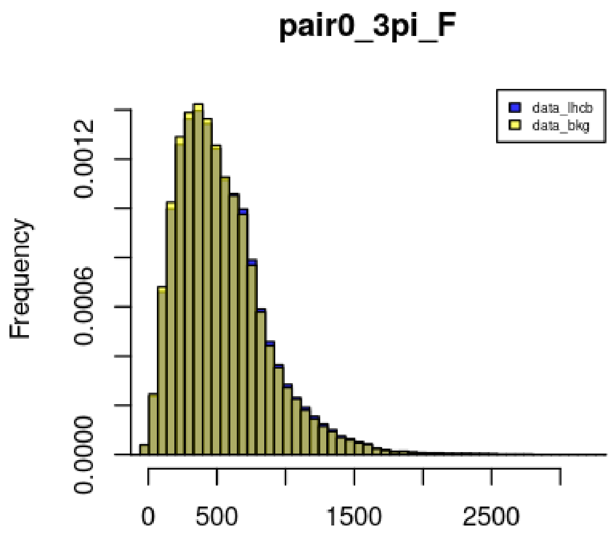
- lcstar_MM_F → $m(\Lambda_c^{*,+}$ (2625 MeV))
- pair_lcstar_F → Pair mass Λ_c^*
- Features used for cuts

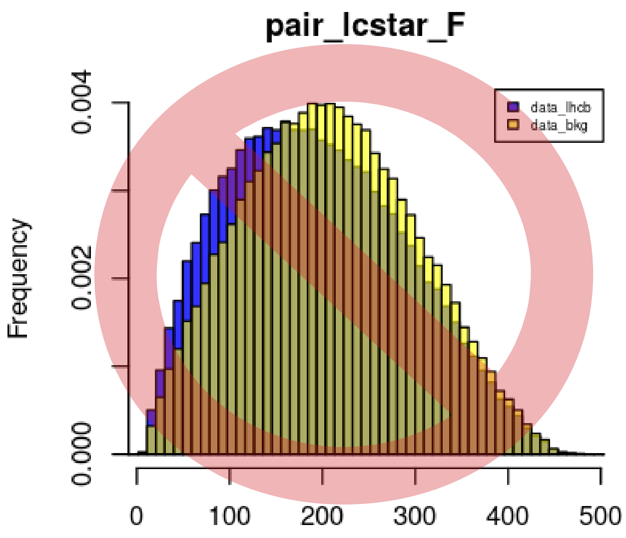




Usable Features







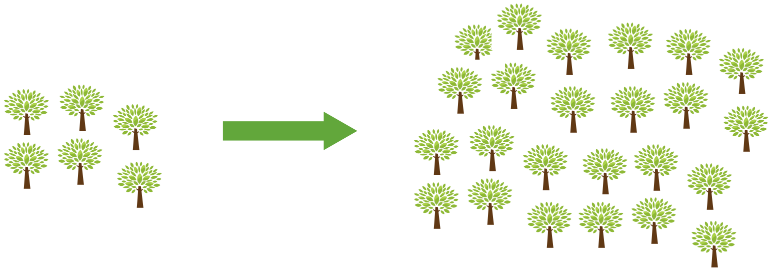
Key concept: Frequency discrepancies must be possibly caused by signals, not by cuts

Expected # of signal events is
7.000 out of 470.000 → 1.4%

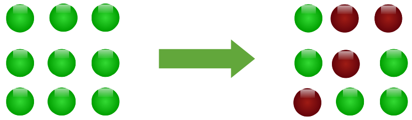
Random Forest

Parameters to be tuned:

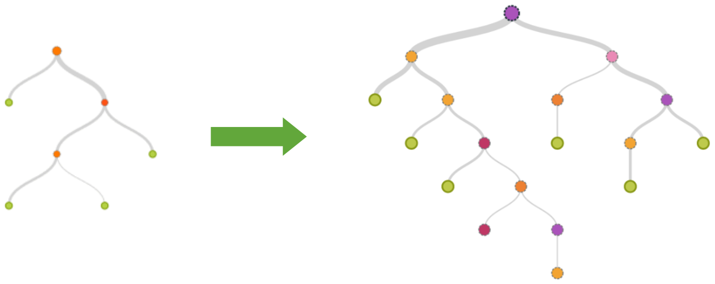
- Random Forest algorithm:

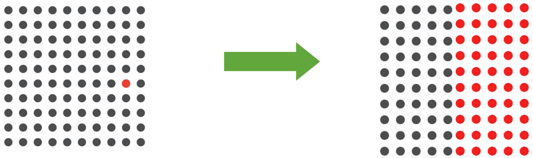


- # of trees in the forest
- # of features randomly chosen for each tree
- Max # of nodes each tree can have
- Minimum size of terminal nodes
- Whether the importance of each tree is uniform or based on its accuracy

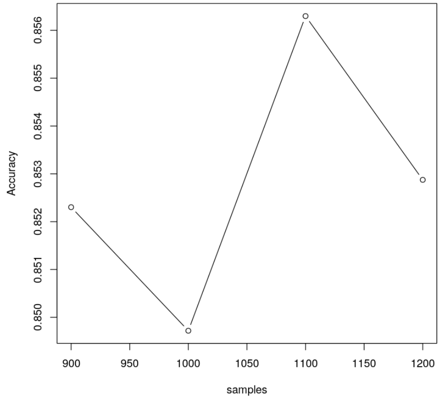


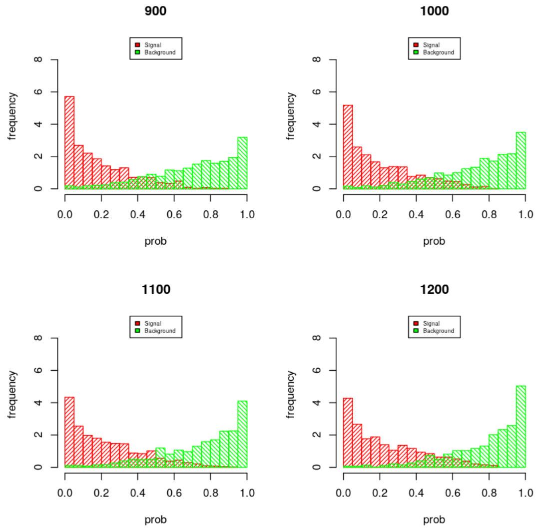
- # of background samples to be used during training



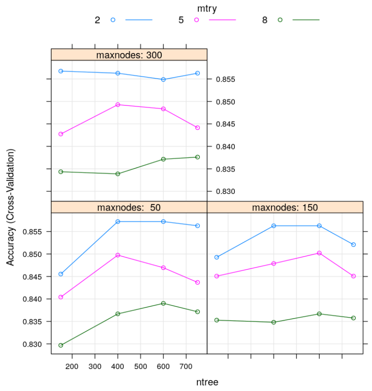


Random Forest – Best Parameters





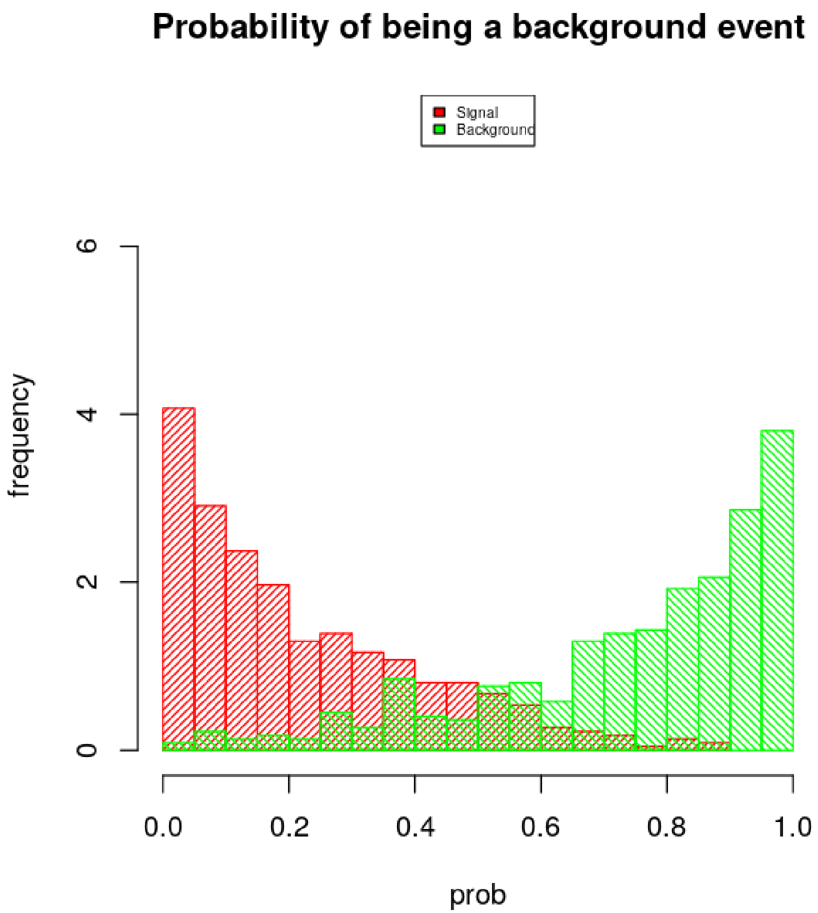
Best # of background samples → 1100
(with # of signal samples → 1043)



- Best # of trees: 400
- Best # features: 5
- Best # nodes: 150
- Best terminal nodes size: 150
- Best importance choice: FALSE

Random Forest - Result

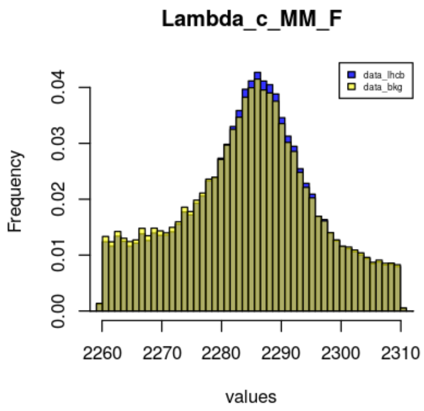
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 378  48
         1  69 399

               Accuracy : 0.8691
                 95% CI : (0.8452, 0.8905)
```

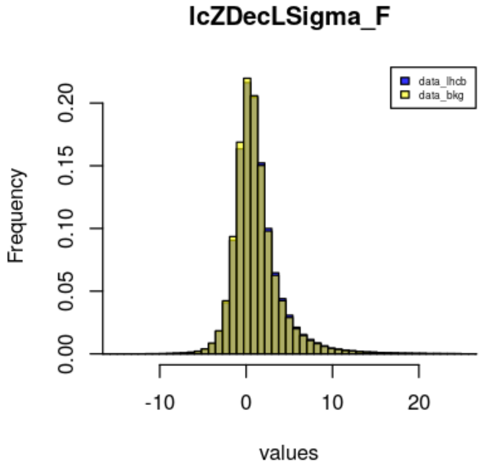


Useful Data

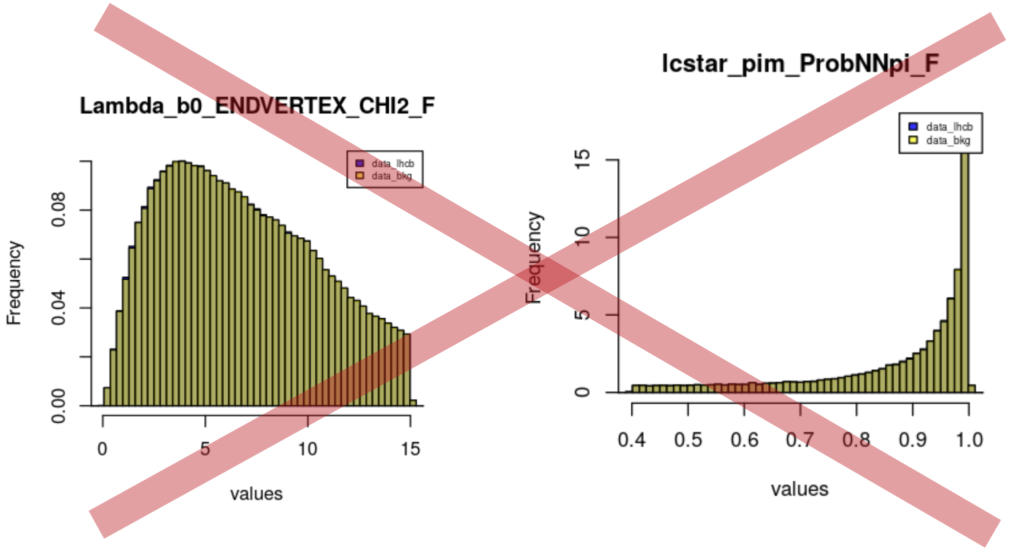
Useful features: features that present an asymmetry in the frequency

Useful





Useless



Deleted features: tau_pion0_ProbNNpi_F, tau_pion1_ProbNNpi_F, tau_pion2_ProbNNpi_F, lcstar_pim_ProbNNpi_F, lcstar_pip_ProbNNpi_F, Lambda_b0_ENDVERTEX_CHI2_F, Lambda_c_ENDVERTEX_CHI2_F, lcstar_ENDVERTEX_CHI2_F"

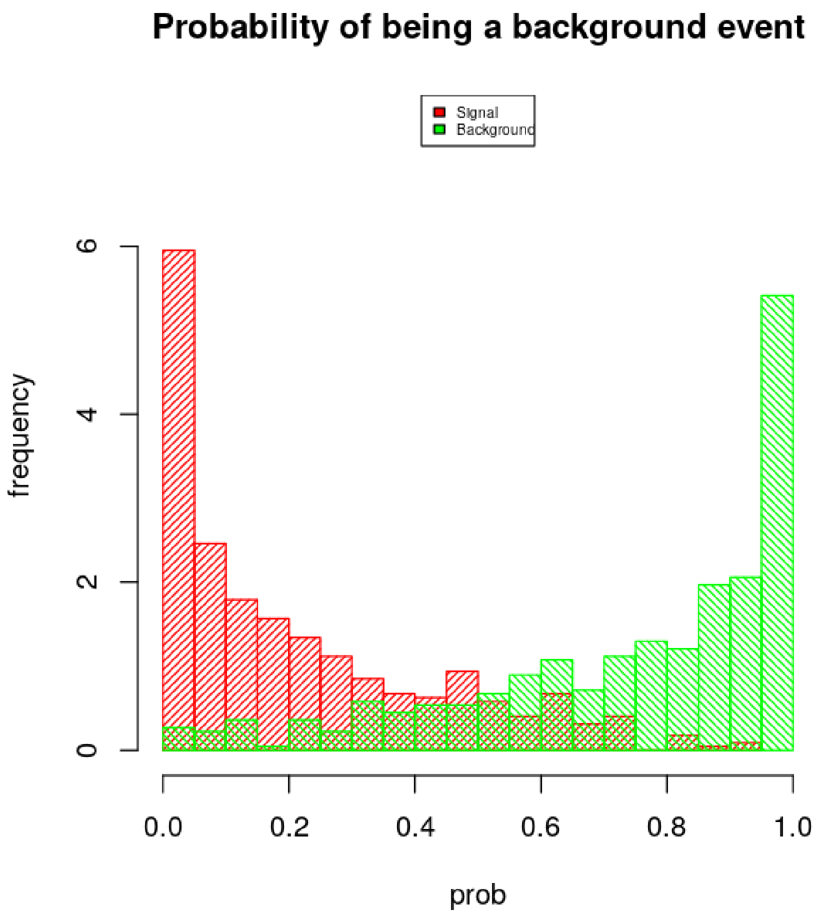
Random Forest - Useful

- Best # of background samples → 1100 1100
  (with # of signal samples → 1043)

- Best # of trees: 135 400 ← Original ones
- Best # features: 3 5
- Best # nodes: 30 150
- Best terminal nodes size: 27 150
- Best importance choice: TRUE FALSE

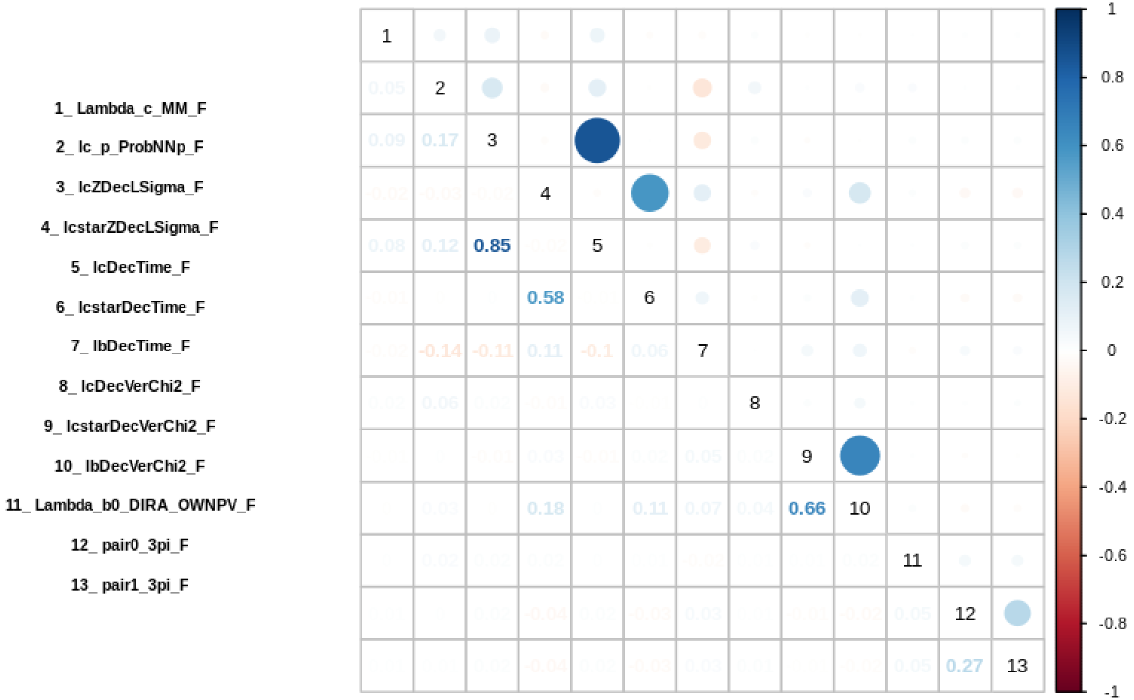
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 367  60
         1  80 387

               Accuracy : 0.8434
                 95% CI : (0.8179, 0.8666)
```

86.9% → 84.3%



Optimized Data

Optimized features: features that present a correlation less than 0.8 in module



Correlated Features:

lcZDecLSigma_F ⟷ lcDecTime_F

↓ Decay length

↓ Decay time

Deleted Feature: lcZDecLSigma_F

*Data used: Data LHCb

Random Forest - Optimized

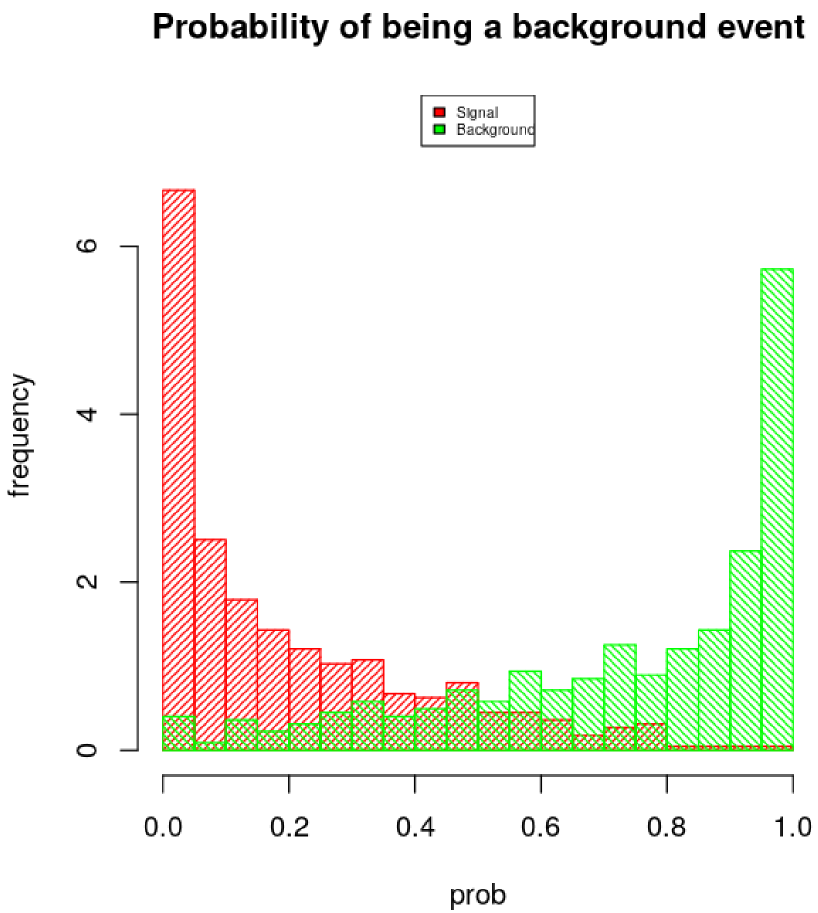
- Best # of background samples → 1100 1100
  (with # of signal samples → 1043)

- Best # of trees: 135 400
- Best # features: 3 5
- Best # nodes: 40 150
- Best terminal nodes size: 30 150
- Best importance choice: TRUE FALSE

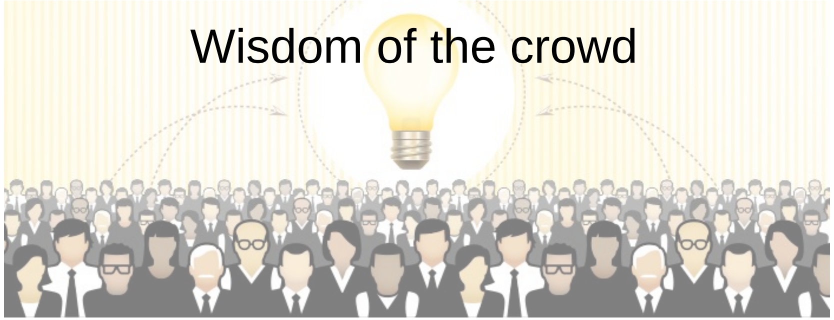
Original ones

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 357  49
         1  90 398

               Accuracy : 0.8445
                 95% CI : (0.8191, 0.8677)
```

86.9% → 84.5%

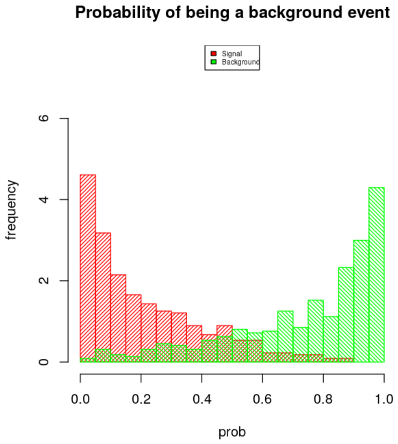


Ensemble Forest




```
params_list <- list(all_datas = list(mtry = 4:6,
                                     ntree = seq(300,500,50),
                                     maxnodes = seq(130,170,10),
                                     importance = c(TRUE,FALSE,FALSE),
                                     nodesize = seq(130,170,10)
                                     ),
                    few_datas = list(mtry = 3:4,
                                     ntree = seq(110,150,10),
                                     maxnodes = seq(30,60,5),
                                     importance = c(TRUE,FALSE),
                                     nodesize = seq(20,40,5)
                                     )
                    )
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 372  46
         1  75 401

               Accuracy : 0.8647
                 95% CI : (0.8405, 0.8864)
```



86.9% → 86.5%

Even if the forests are different, they don't learn different patterns

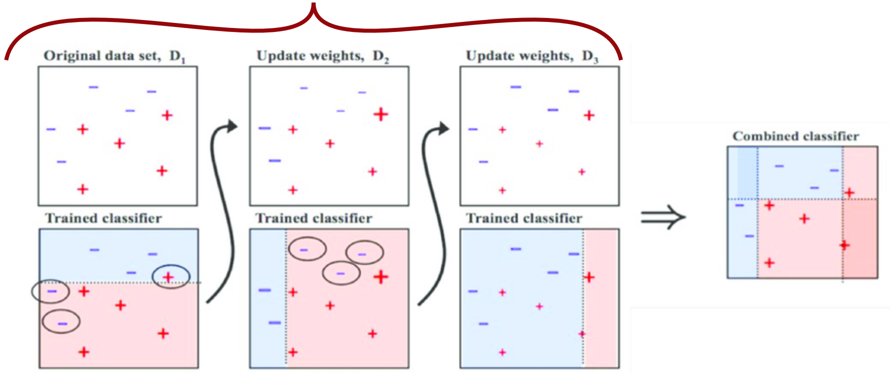
AdaBoost

Parameters to be tuned:

- Maximum depth of each tree
- Maximum number of rounds

- Best maximum depth: 4
- Best maximum # of rounds: 350

# of rounds



AdaBoost - Results

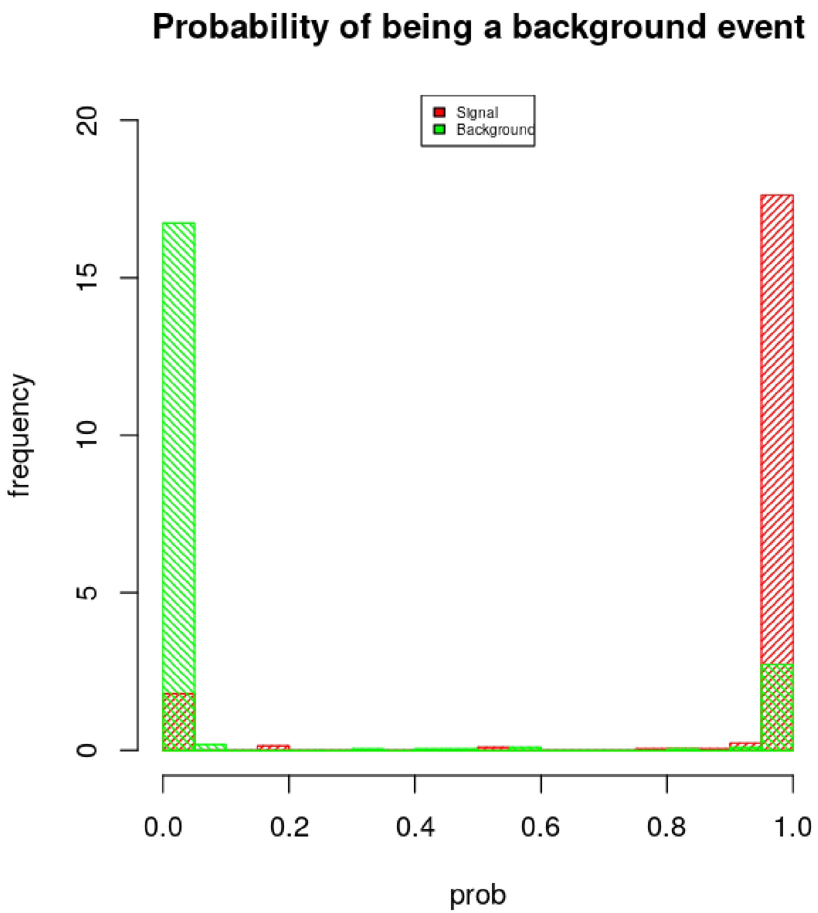
AdaBoost:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 381  43
         1  66 404

               Accuracy : 0.8781
                 95% CI : (0.8548, 0.8988)
```

Random Forest:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 378  48
         1  69 399

               Accuracy : 0.8691
                 95% CI : (0.8452, 0.8905)
```

86.9% → 87.8% → + 0.9%

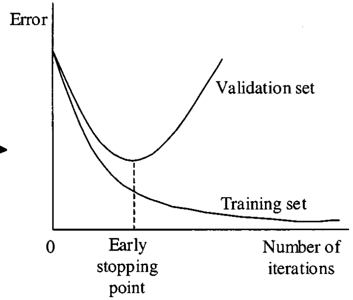


Gradient Boosting

Parameters to be tuned:

- Maximum depth of each tree
- Maximum number of rounds
- Scaling contribution for the next tree
- Minimum loss reduction required to make a further partition on a leaf node
- # of rounds of patience before early stopping the algorithm if the validation accuracy stops increasing

- Best maximum depth: 2
- Best maximum # of rounds: 300
- Best scaling contribution: 0.57
- Best minimum loss reduction: 0
- Best early stopping # of rounds: 50



Gradient Boosting - Results

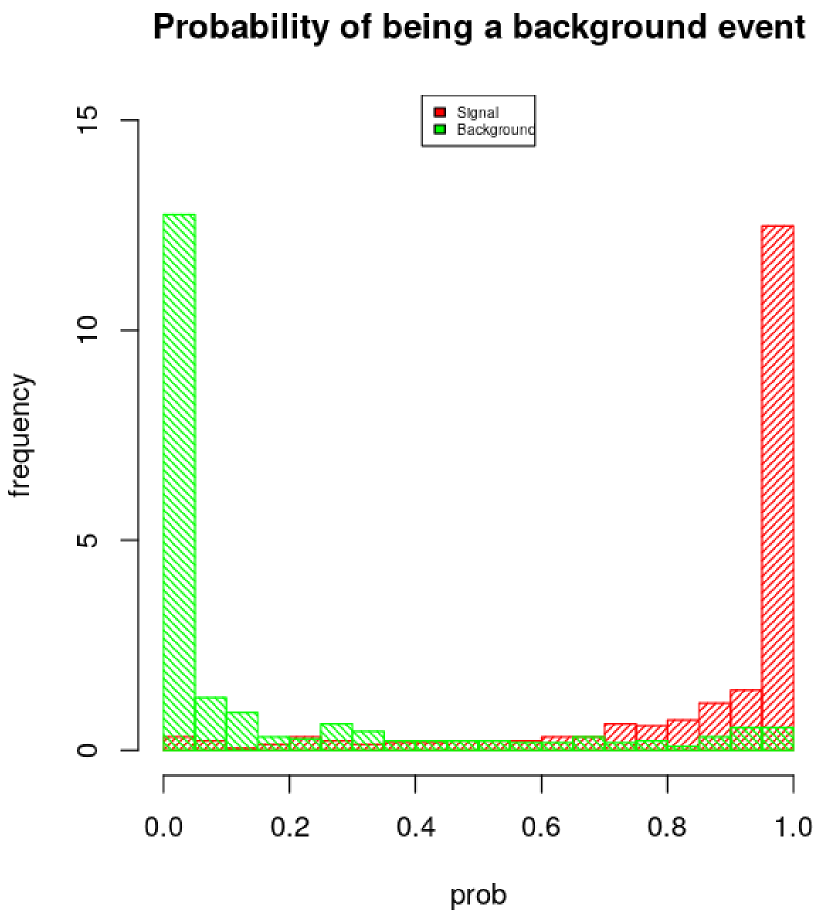
Gradient Boosting:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 385  44
         1  62 403

               Accuracy : 0.8814
                 95% CI : (0.8584, 0.9019)
```

AdaBoost:

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 381  43
         1  66 404

               Accuracy : 0.8781
                 95% CI : (0.8548, 0.8988)
```

87.8% → 88.1% → + 0.3%



Neural Networks - Parameters

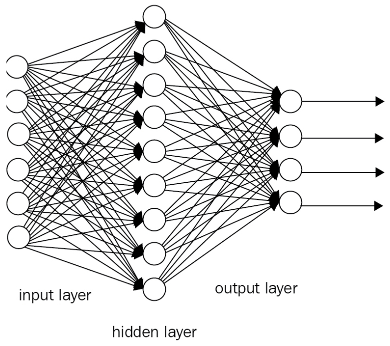
Parameters to be tuned:

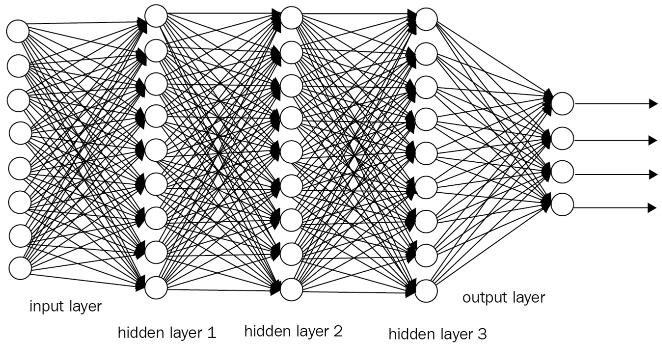
- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions

Neural Networks - Parameters

Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions





High # of layers and neurons

Higher prediction power

Higher risk of overfitting

Low # of layers and neurons

Lower risk of overfitting

Lower prediction power

Neural Networks - Parameters

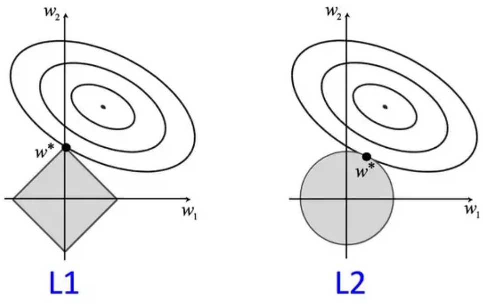
Reduce risk of overfitting

Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions

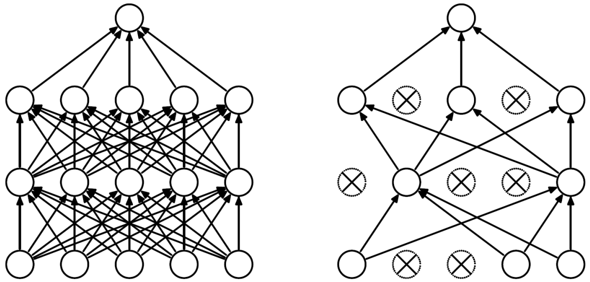
- L1 or L2 regularizer

  Add l1 or l2 norm of the weights to the loss function



- Dropout

  Probability p (to be tuned) of dropping out each neuron



- Batch normalization

Neural Networks - Parameters

Key problem: gradient saturation

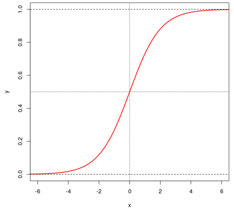
Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions

- Sigmoid

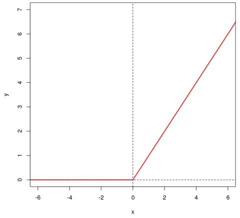
$$f(x) = \frac{1}{1 + e^{-x}}$$

Gradient saturation



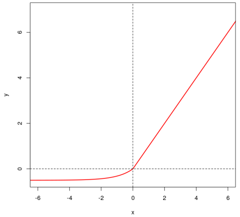
- ReLU

$$f(x) = \max(0, x)$$



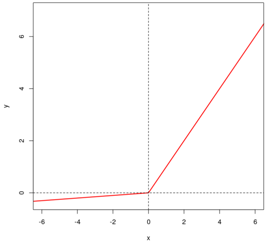
- ELU

$$f(x) = \begin{cases} x & \text{if} \quad x > 0 \\ \alpha \cdot (e^x - 1) & \text{if} \quad x < 0 \end{cases} \qquad 0 < \alpha < 1$$



- Leaky ReLU

$$f(x) = \begin{cases} x & \text{if} \quad x > 0 \\ a \cdot x & \text{if} \quad x < 0 \end{cases} \qquad 0 < a < 1$$

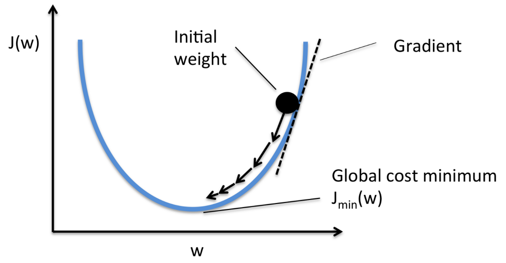


Neural Networks - Parameters

Key problems:
- don’t fall into local minima
- stabilize around global minimum

Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions



- Nesterov Accelerated Gradient optimizer

  Calculate the gradient of the cost function not at the local position but slightly ahead in the direction of the momentum

- RMSProp optimizer

  Accumulate gradients from the most recent iterations, with an exponential decay average of the squares

- Adam optimizer

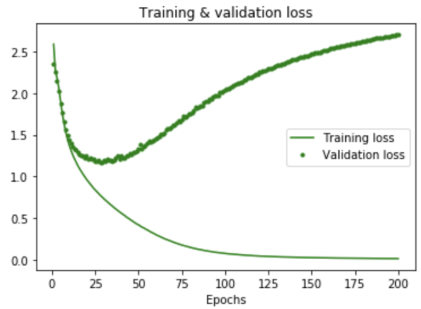
  ADAptive Moment estimation: like RMSProp but takes into consideration the normal (not squared) average as well

Neural Networks - Parameters

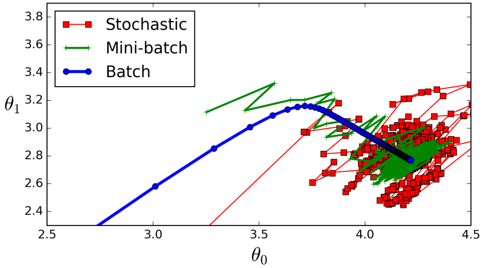
Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer
- Activation function
- Optimizer
- Number of epochs
- Batch size
- Callback functions as validation accuracy stops increasing

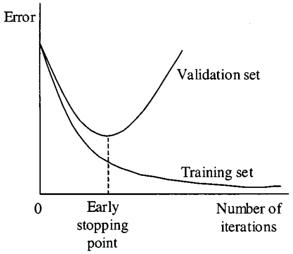
Risk of overfitting/underfitting



How do I reach the global minimum



- Early stopping
- Reducing learning rate



Neural Networks – Best Parameters

Parameters to be tuned:

- Number of layers
- Number of neurons per layer
- Regularizer: Batch Normalization + Dropout (p = 0.5)
- Activation function: ReLU
- Optimizer: RMSProp
- Number of epochs: 400
- Batch size: 25
- Callback functions:
  - Early stopping: patience 100
  - Reducing LR: patience 25, factor 0.6

Voting Classifier

1 Layer:	700
2 Layers:	90 + 180
3 Layers:	250 + 130 + 80
4 Layers:	100 + 70 + 50 + 30
5 Layers:	100 + 70 + 50 + 30 + 20

Neural Networks – Summary Models

```
Model
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 700)               15400
_________________________________________________________________
dropout (Dropout)            (None, 700)               0
_________________________________________________________________
batch_normalization_v1 (BatchNormal (None, 700)        2800
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 701
=================================================================
Total params: 18,901
Trainable params: 17,501
Non-trainable params: 1,400
```

```
Model
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 90)                1980
_________________________________________________________________
dropout_1 (Dropout)          (None, 90)                0
_________________________________________________________________
batch_normalization_v1_1 (BatchNorm (None, 90)         360
_________________________________________________________________
dense_3 (Dense)              (None, 180)               16380
_________________________________________________________________
dropout_2 (Dropout)          (None, 180)               0
_________________________________________________________________
batch_normalization_v1_2 (BatchNorm (None, 180)        720
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 181
=================================================================
Total params: 19,621
Trainable params: 19,081
Non-trainable params: 540
```

```
Model
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 250)               5500
_________________________________________________________________
dropout_3 (Dropout)          (None, 250)               0
_________________________________________________________________
batch_normalization_v1_3 (BatchNorm (None, 250)        1000
_________________________________________________________________
dense_6 (Dense)              (None, 130)               32630
_________________________________________________________________
dropout_4 (Dropout)          (None, 130)               0
_________________________________________________________________
batch_normalization_v1_4 (BatchNorm (None, 130)        520
_________________________________________________________________
dense_7 (Dense)              (None, 80)                10480
_________________________________________________________________
dropout_5 (Dropout)          (None, 80)                0
_________________________________________________________________
batch_normalization_v1_5 (BatchNorm (None, 80)         320
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 81
=================================================================
Total params: 50,531
Trainable params: 49,611
Non-trainable params: 920
```

```
Model
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 100)               2200
_________________________________________________________________
dropout_10 (Dropout)         (None, 100)               0
_________________________________________________________________
batch_normalization_v1_10 (BatchNor (None, 100)        400
_________________________________________________________________
dense_15 (Dense)             (None, 70)                7070
_________________________________________________________________
dropout_11 (Dropout)         (None, 70)                0
_________________________________________________________________
batch_normalization_v1_11 (BatchNor (None, 70)         280
_________________________________________________________________
dense_16 (Dense)             (None, 50)                3550
_________________________________________________________________
dropout_12 (Dropout)         (None, 50)                0
_________________________________________________________________
batch_normalization_v1_12 (BatchNor (None, 50)         200
_________________________________________________________________
dense_17 (Dense)             (None, 30)                1530
_________________________________________________________________
dropout_13 (Dropout)         (None, 30)                0
_________________________________________________________________
batch_normalization_v1_13 (BatchNor (None, 30)         120
_________________________________________________________________
dense_18 (Dense)             (None, 20)                620
_________________________________________________________________
dropout_14 (Dropout)         (None, 20)                0
_________________________________________________________________
batch_normalization_v1_14 (BatchNor (None, 20)         80
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 21
=================================================================
Total params: 16,071
Trainable params: 15,531
Non-trainable params: 540
```

```
Model
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 100)               2200
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
batch_normalization_v1_6 (BatchNorm (None, 100)        400
_________________________________________________________________
dense_10 (Dense)             (None, 70)                7070
_________________________________________________________________
dropout_7 (Dropout)          (None, 70)                0
_________________________________________________________________
batch_normalization_v1_7 (BatchNorm (None, 70)         280
_________________________________________________________________
dense_11 (Dense)             (None, 50)                3550
_________________________________________________________________
dropout_8 (Dropout)          (None, 50)                0
_________________________________________________________________
batch_normalization_v1_8 (BatchNorm (None, 50)         200
_________________________________________________________________
dense_12 (Dense)             (None, 30)                1530
_________________________________________________________________
dropout_9 (Dropout)          (None, 30)                0
_________________________________________________________________
batch_normalization_v1_9 (BatchNorm (None, 30)         120
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 31
=================================================================
Total params: 15,381
Trainable params: 14,881
Non-trainable params: 500
```

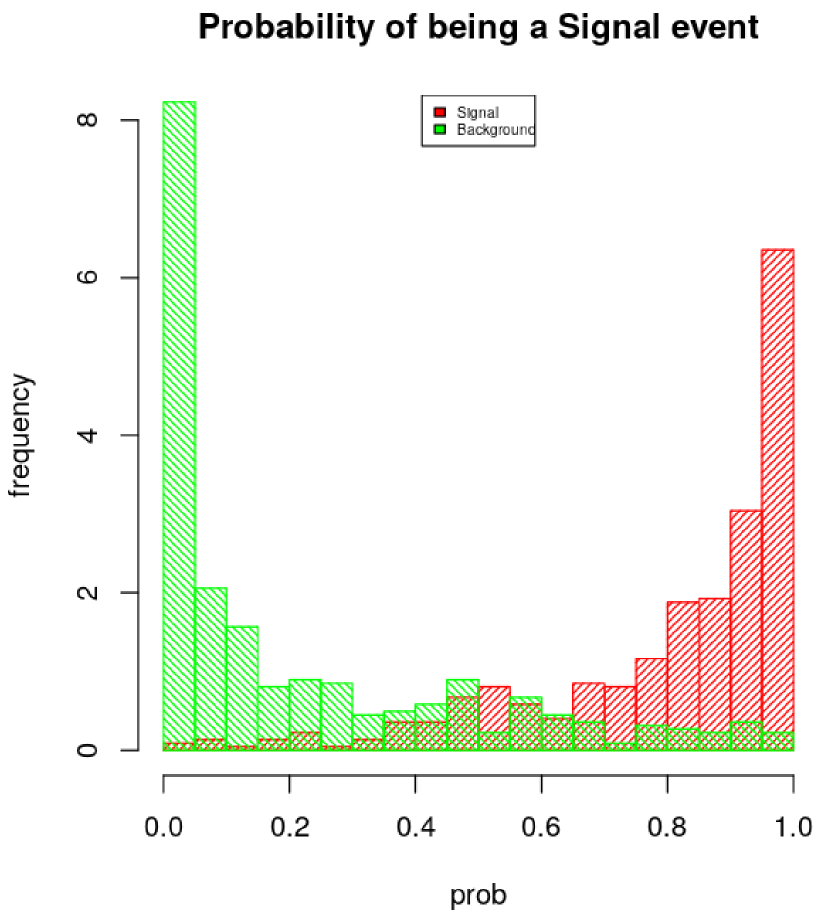
Neural Networks – Results

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 376  49
         1  71 398

               Accuracy : 0.8658
                 95% CI : (0.8417, 0.8874)
```

86.6% < 86.9% < 88.1%

Neural Network — Random Forest — Gradient Boosting

No reasons why Neural Networks should perform better than Random Forests

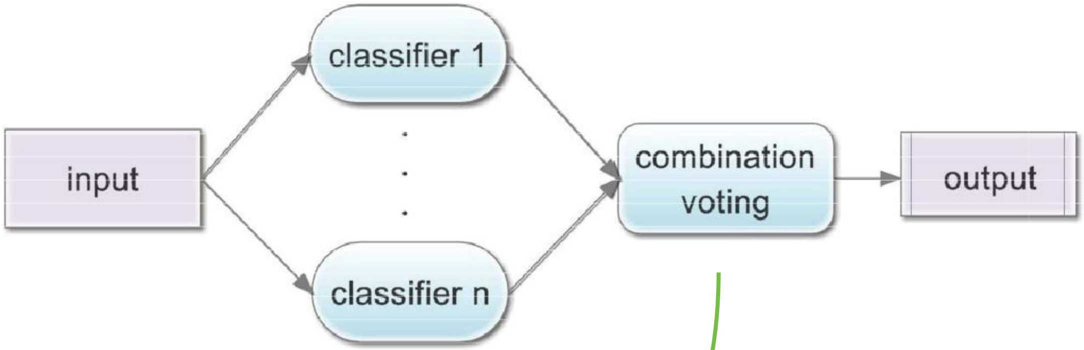


Final Entsemble

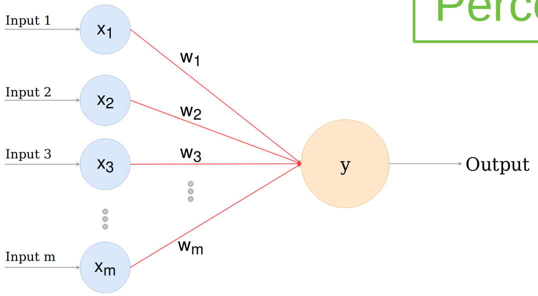
Classifiers used:

- Random forest - all features
- Random forest - useful features
- Random forest - optimized features
- Ensemble Forest
- AdaBoost
- Gradient Boosting
- Neural Networks (5 of them considered separately)

Ensemble



Perceptron



Constraint: $\sum_i w_i = 1$

Weighted average!

Cons: fewer training samples for the classifiers, some of them set aside for the training of the perceptron

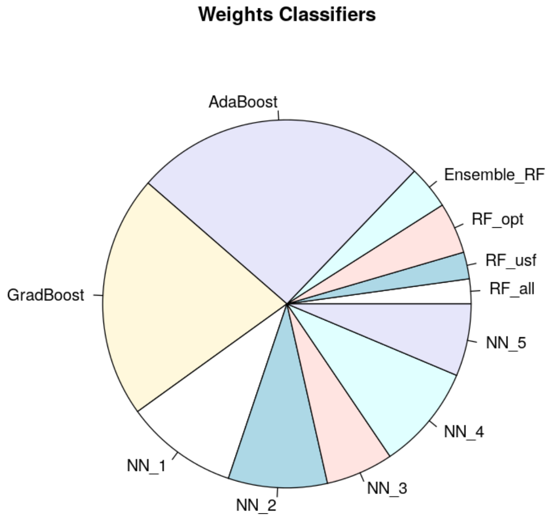
Final Ensemble - Results

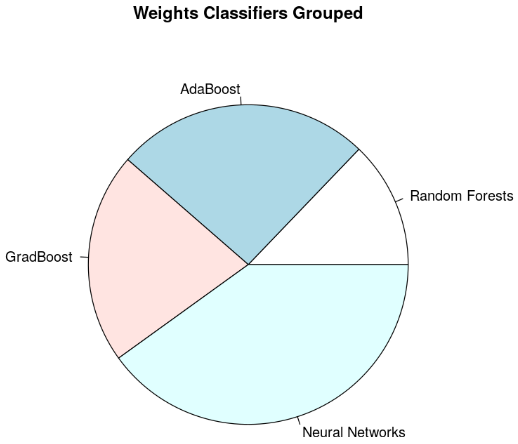
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 396  48
         1  51 399

               Accuracy : 0.8893
                 95% CI : (0.8668, 0.9091)
```

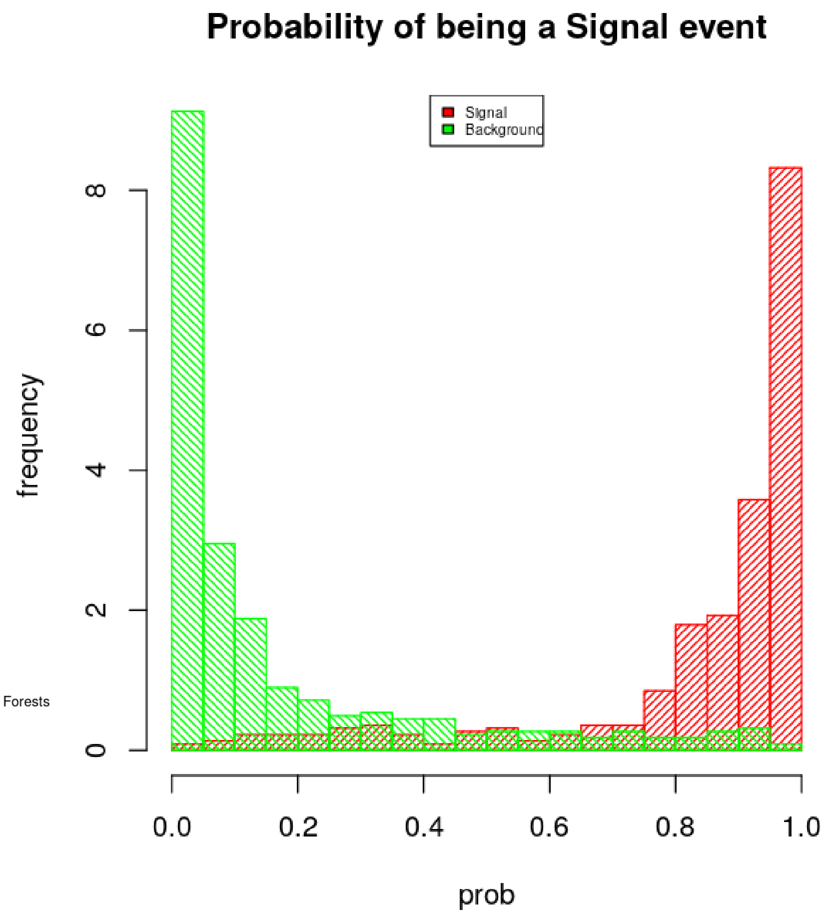
88.1% → 88.9% → + 0.8%

Gradient Boosting — Final Ensemble

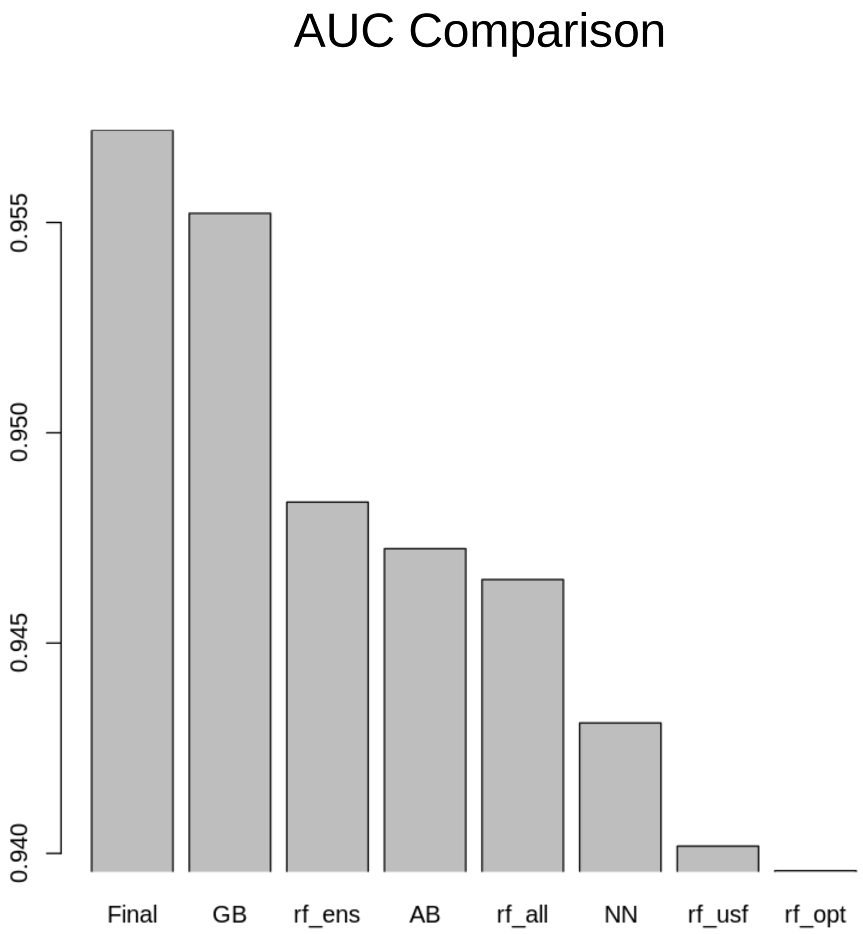
Best Result!

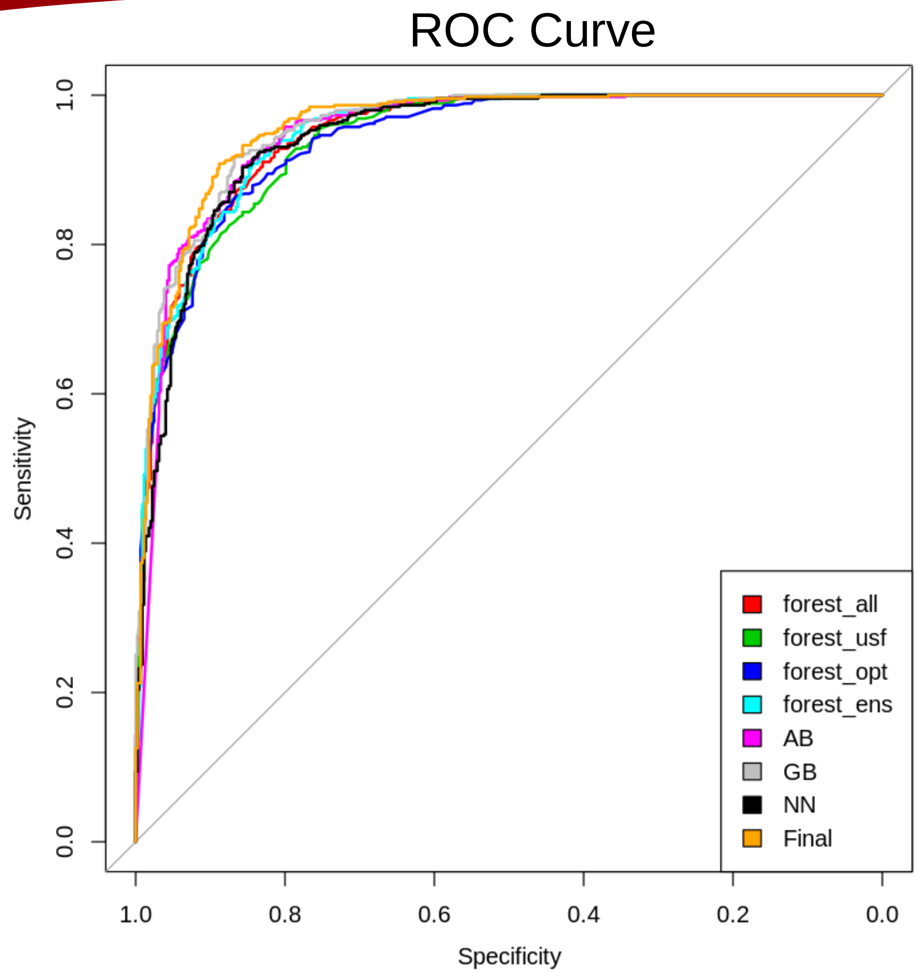






Classifiers Comparison

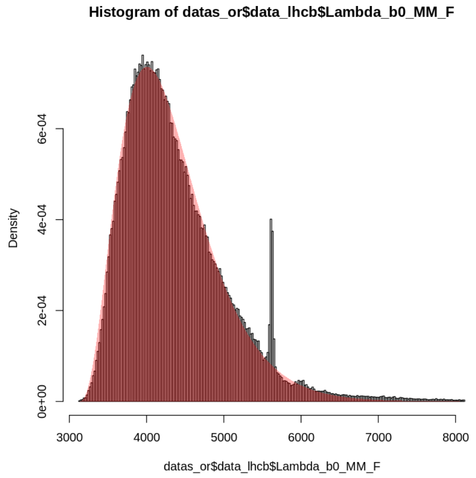




Theoretical # of interesting events

How to:

1) Fit all data (*feature $m(\Lambda_b^0)$*) with the most suitable distribution

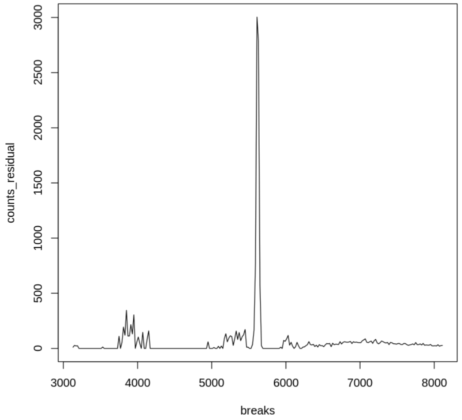


Regions where signals are present have been left out

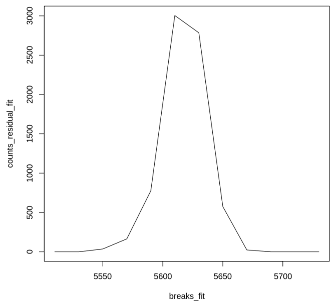
Gamma distribution

$\alpha \simeq 3.84 \quad \beta \simeq 0.0032$

2) Difference between fitted distribution and data, constrained to positive values



3) Cut with the cuts used selecting the background and sum the counts

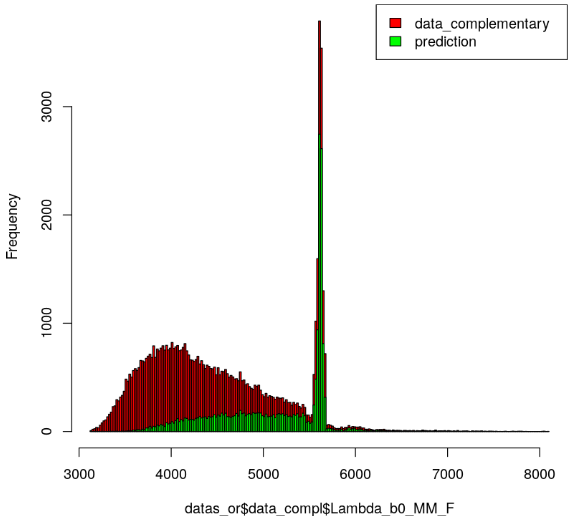


Theoretical # of interesting events:

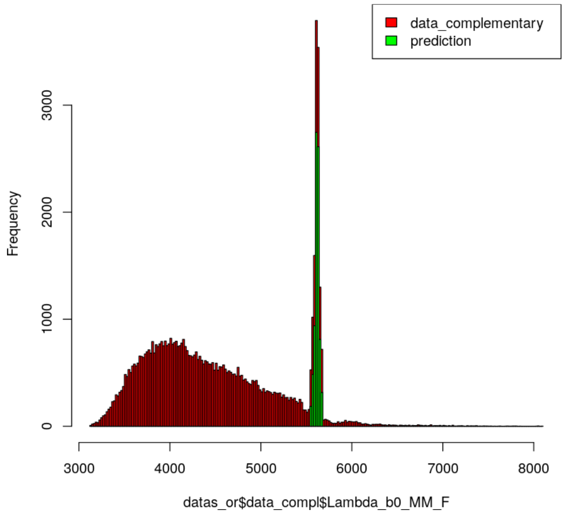
7363

Final Prediction

All complementary data



Complementary data with cuts



Counts $m(\Lambda_b^0)<5550$ MeV

11.200

Total $m(\Lambda_b^0)<5550$ MeV

55.733

79.9%

< 88.9% !!

Theoretical number

Sum of counts: 7713 → 7363