Quality Assessment of NGS Data using Machine Learning

Alexander Hinzer

Hochschule für Technik und Wirtschaft Berlin
Robert Koch-Institut Berlin

07 April 2022

Background

- Quality assessment of sequencing data:
  - First step in the data analysis
  - Performed manually by biologists
  - Supported by software but not automated

Explore the possibility to use machine learning algorithms to enable the automation of quality assessment of Next Generation Sequencing data.

Outline

1. Sequencing Data and FastQC
2. Introduction to Machine Learning (ML)
3. ML classification algorithms
4. First Implementation & Results
5. Feature Engineering of FastQC Modules
6. Final Results & Pipeline Implementation
7. Conclusion

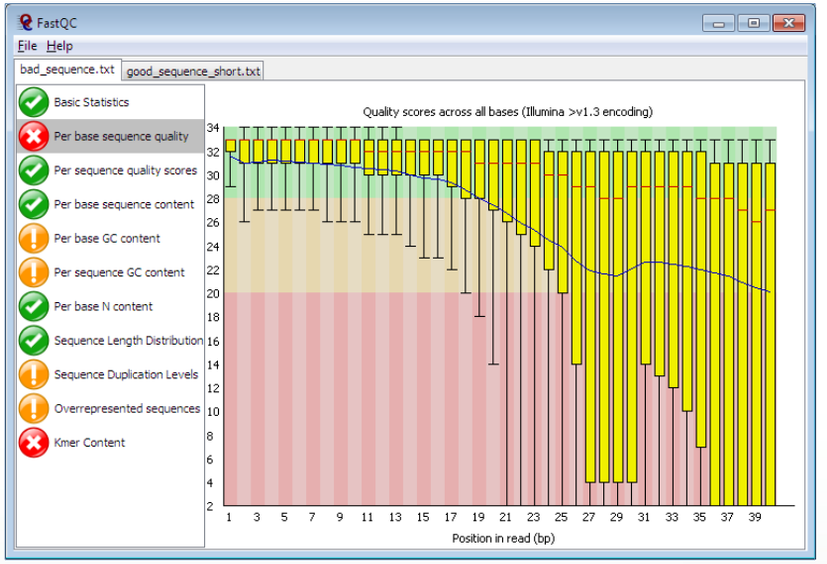
Next Generation Sequencing Data

- NGS systems produce files for the sequenced DNA
- FASTQ file format:
  - Sequences of base calls for all reads
  - Quality (Phred) score for each base call
  - Information per read (flowcell, lane, tile, position)

```
1  @SIM:1:FCX:1:15:6329:1045:GATTACT+GTCTTAAC 1:N:0:ATCCGA
2  TCGCACTCAACGCCCTGCATATGACAAGACAGAATC
3  +
4  <>;##=><9=AAAAAAAAAA9#:<#<;<<<????#=
```

Quality Control with FastQC

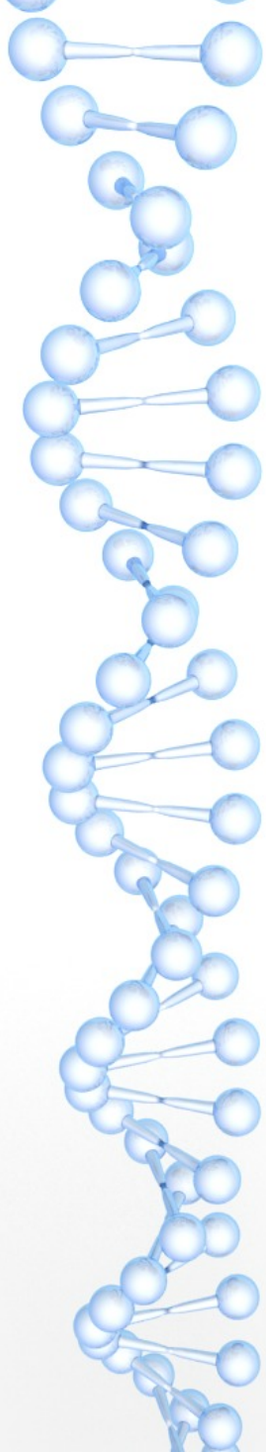
- FastQC for quality check of NGS data
- Provides analysis results in modules
- Assigns a status for each module (pass, warning, failure)
- Reports in GUI or file export



2. Introduction to Machine Learning

What is Machine Learning?

- Learn rules by analyzing example data
- Learned rules = *model*
  Example data = *training data*
- Supervised Learning: learn using known solutions, called *labels*
- *Classification*: determine class of a data point by analyzing its attributes, called *features*



2. Introduction to Machine Learning

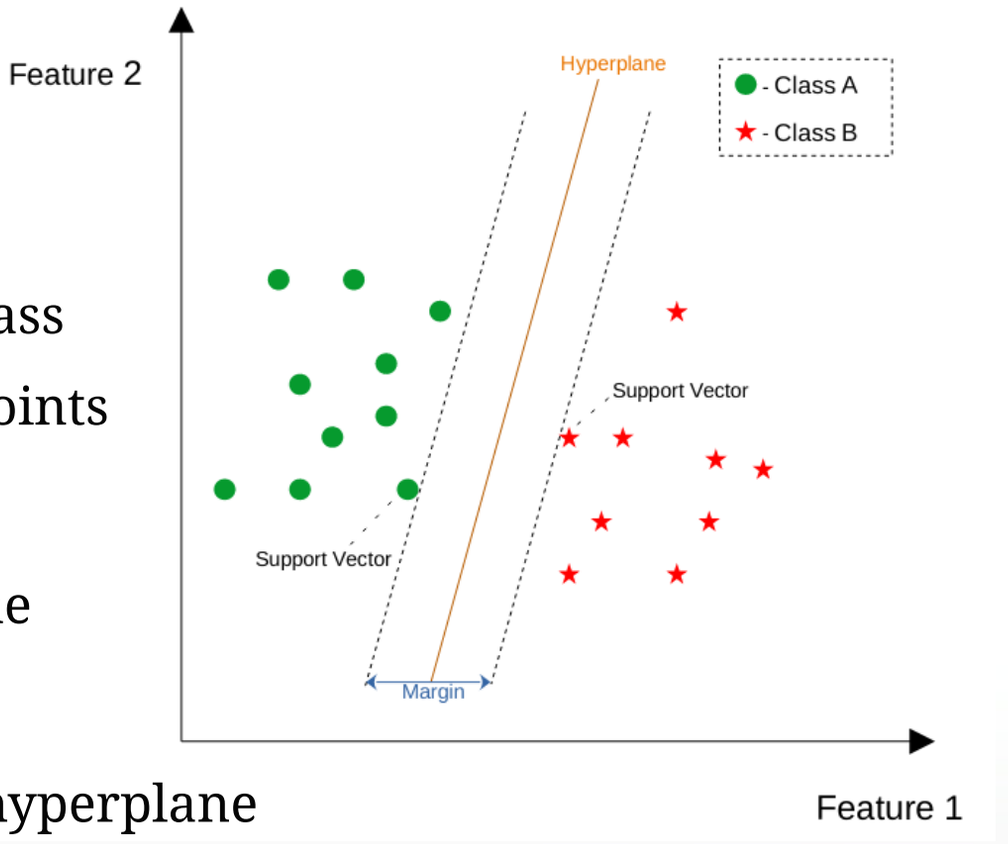
The Data

- 184 FASTQ files from the RKI
  - 88× E. Faecium, 78× S. Aureus, 18× E. Coli
- Quality assessments provided => labels: “good”/”ugly“
- FastQC analysis => features
  - Basic Statistics:
    - Total sequences, %GC, shortest and longest sequence length
  - Module statuses encoded as:
    - failure = 0, warning = 1, pass = 2

3. ML Classification Algorithms

Support Vector Machine I

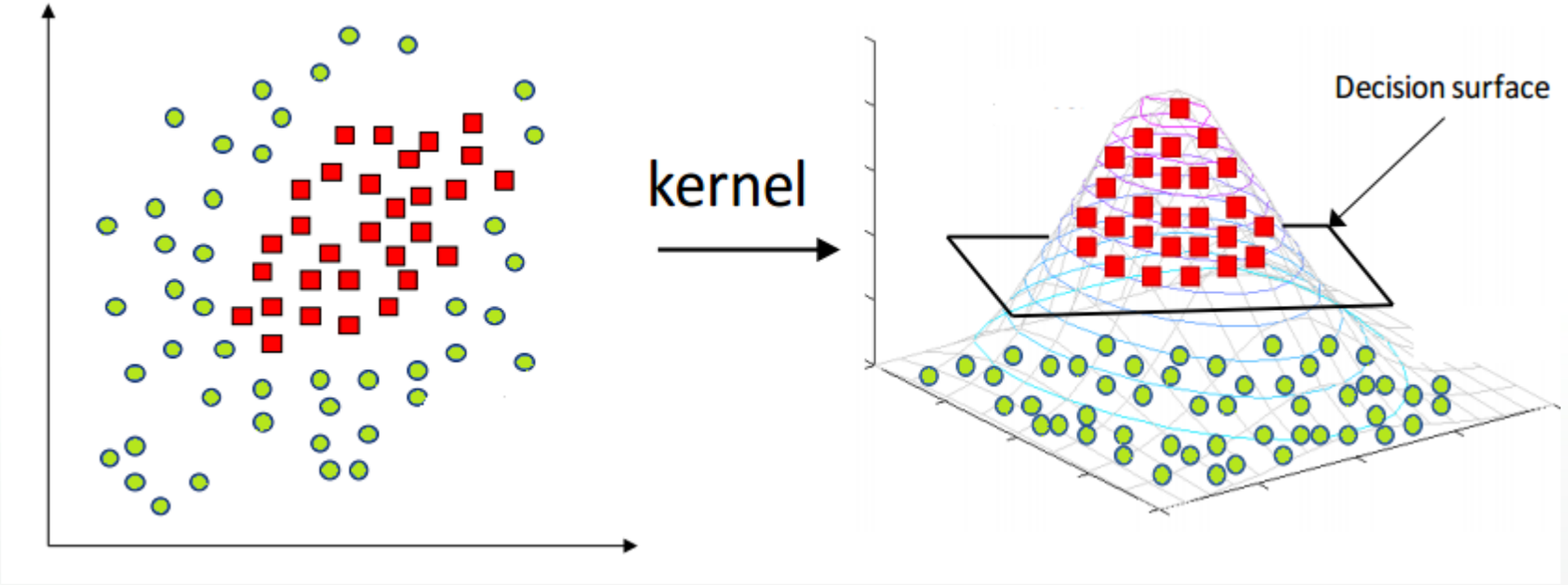
- Map datapoints in vector space
  - Dimension = number of features
- Find the *hyperplane*
  - Separates the datapoints of each class
  - Dimension = one lower than datapoints
- Support Vectors:
  - Datapoints closest to the hyperplane
- Maximize Margin:
  - Distance of support vectors to the hyperplane



3. ML Classification Algorithms

Support Vector Machine II

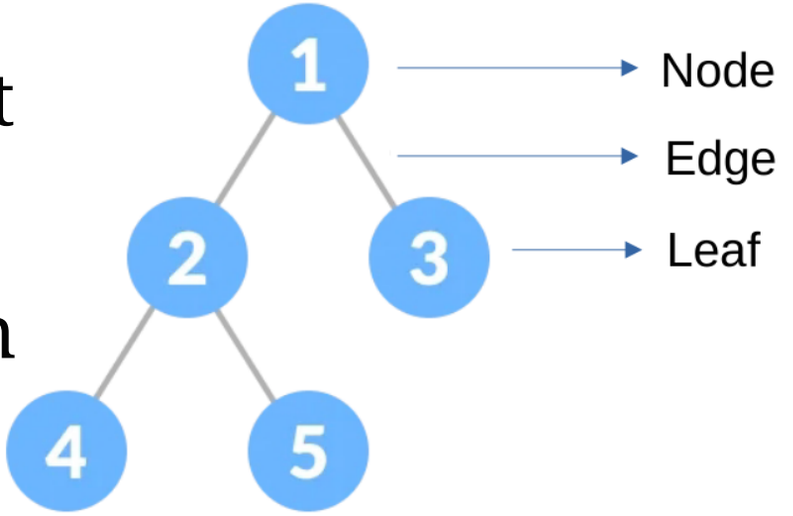
- SVMs can also apply non linear separation
  - Soft margin: tolerate outliers
  - Kernel function: transform data to higher dimension



3. ML Classification Algorithms

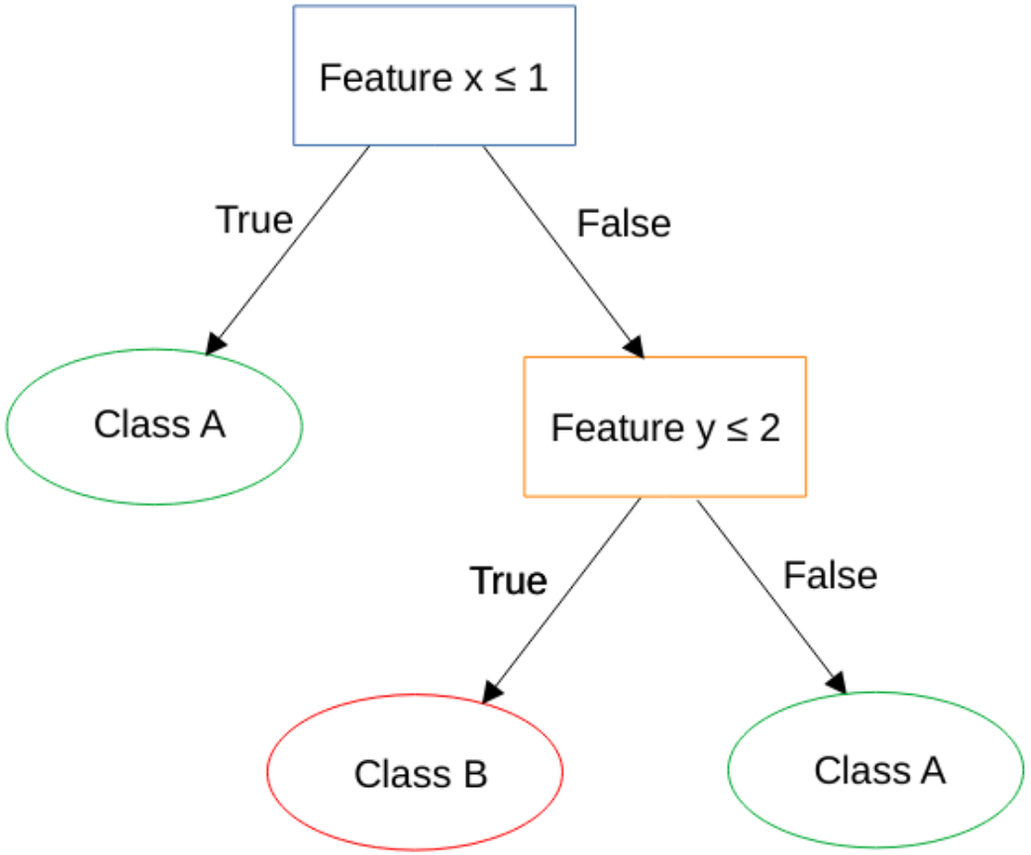
Decision Tree I

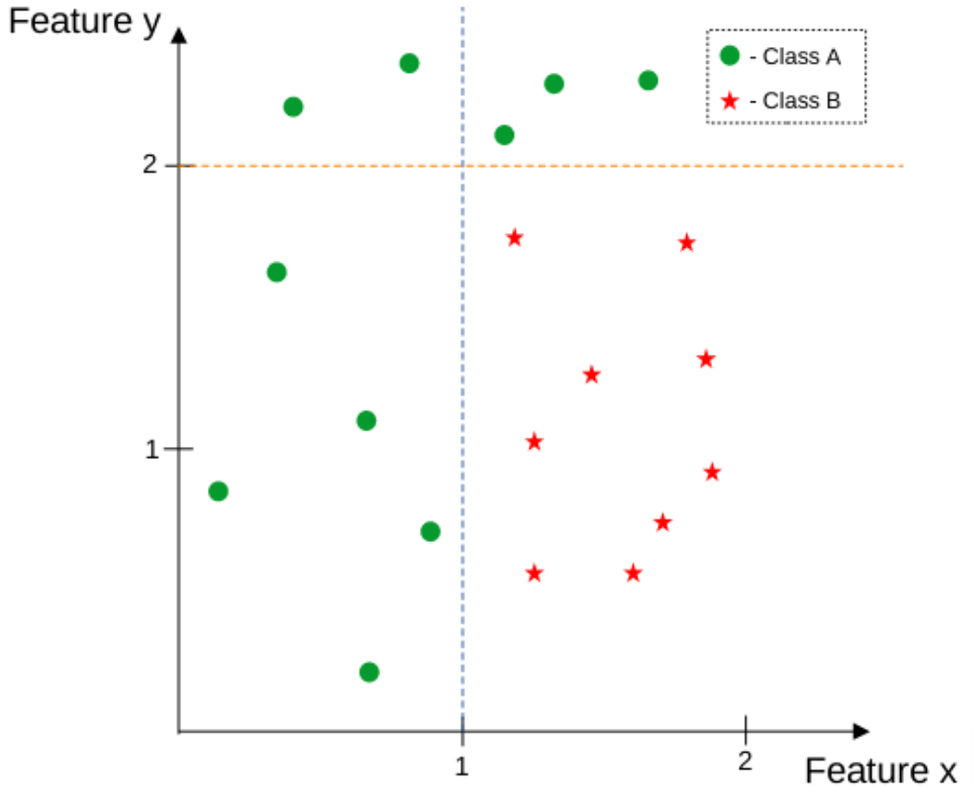
- Nodes apply a feature tests
- Edges: possible answers that split the data
- Pure Leaves: End points, uniform class
- Add nodes recursively until only pure leaves left



3. ML Classification Algorithms

Decision Tree II





3. ML Classification Algorithms

Decision Tree III

- Find the best feature test for the data at a node
- Measure of the ability to separate the data:
  - **Gini Impurity** $G(D) = 1 - \sum_{i=1}^{K} p_i^2$
- Pure leaf: G(D) = 0 ,
  Equal class distribution: G(D) = 0.5

3. ML Classification Algorithms

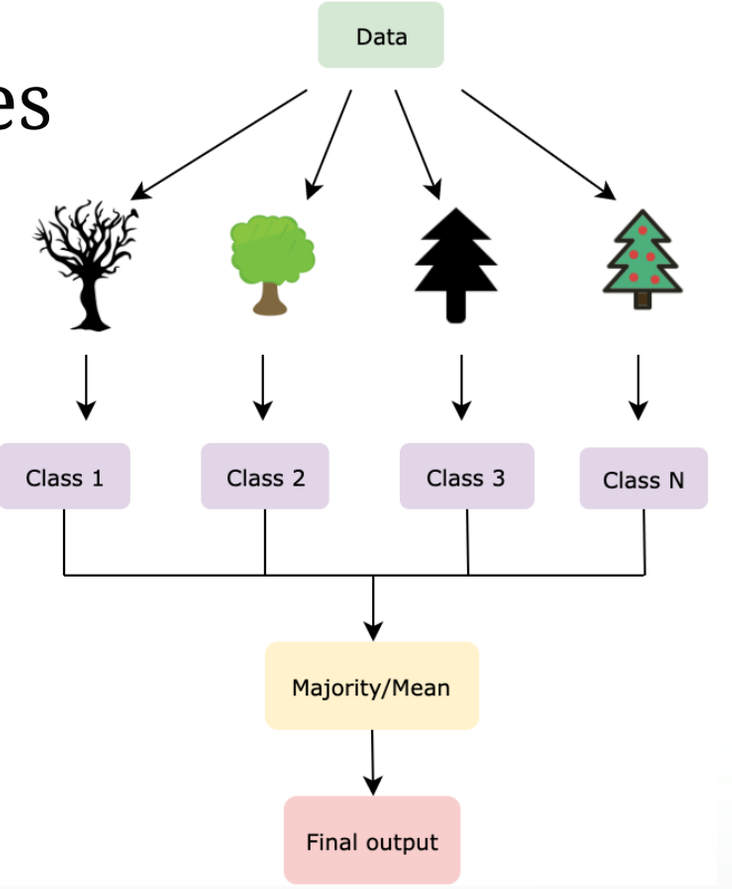
Decision Tree IV

- Intuitive classification procedure
- Problem: *Overfitting*
- Solution: *Pruning*
- Post-Pruning:
  - remove unimportant nodes
- Pre-Pruning: limit building of the tree
  - Fixed maximum depth
  - Fixed number of leaves

3. ML Classification Algorithms

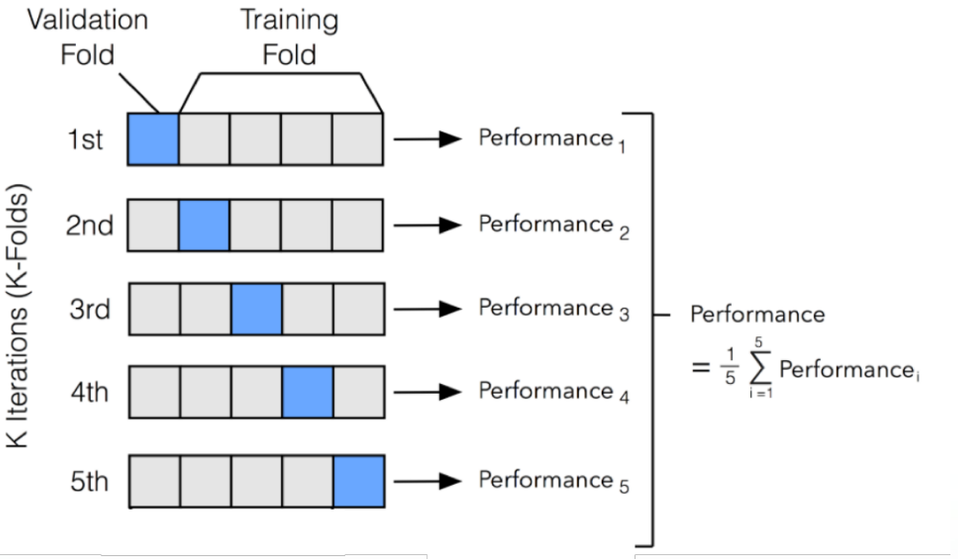
Random Forest

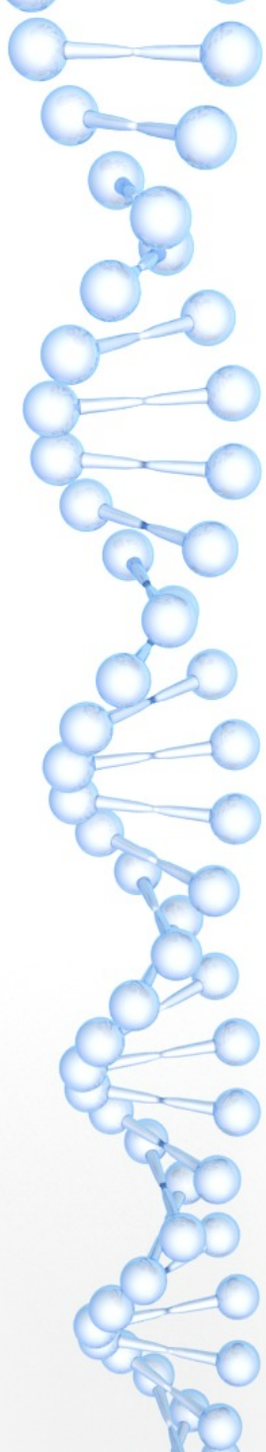
- Improves generalizability of d. trees
- Builds many random trees
  - Using bootstrap samples
    - Random subsets of training data
  - Limiting available features for nodes
- Final classification by
  - Majority voting
  - Soft voting



Evaluate Performance with Cross-Validation

- Withhold data from model training to validate performance → less information
- Split data in *k* subsets
- Build *k* models
  - *k*-1 subsets for training
  - Remaining for validation
- Average performance results of all *k* runs





3. ML Classification Algorithms

Performance Measures

	Actually „good“	Actually „ugly“
Classified as „good“	true positive (tp)	false positive (fp)
Classified as „ugly“	false negative (fn)	true negative (tn)

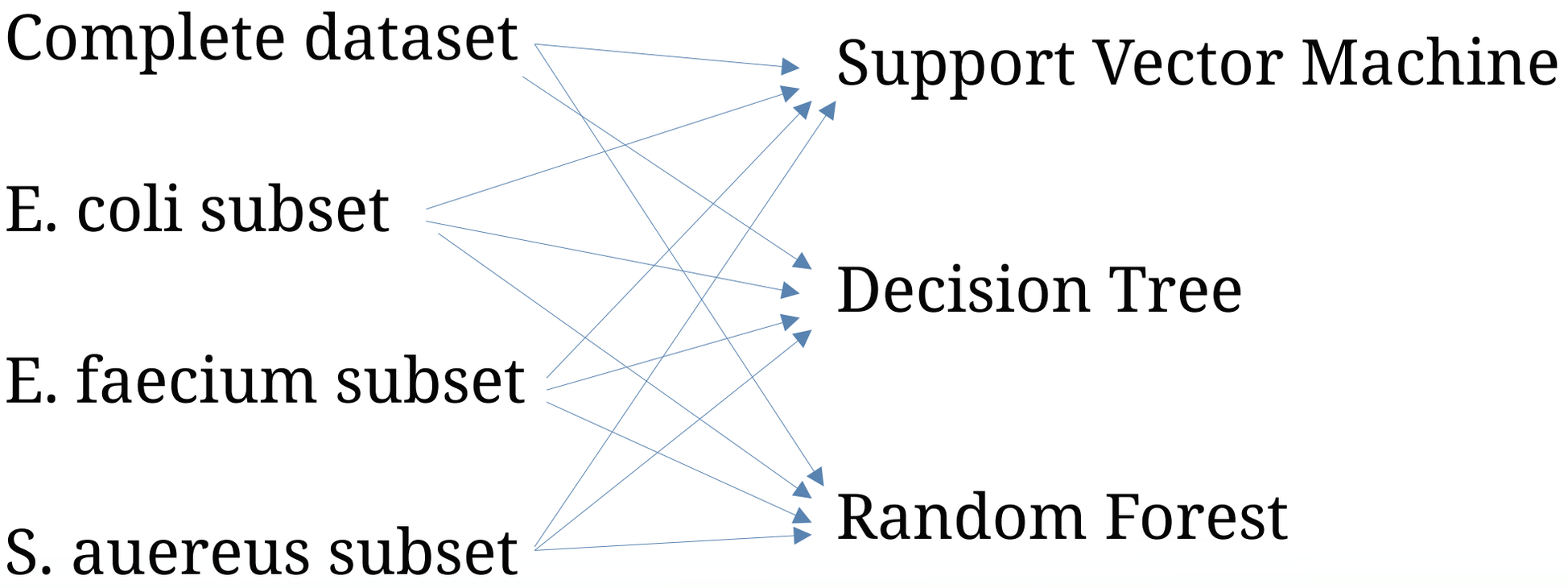
$$Precision = \frac{t_p}{t_p + f_p}$$

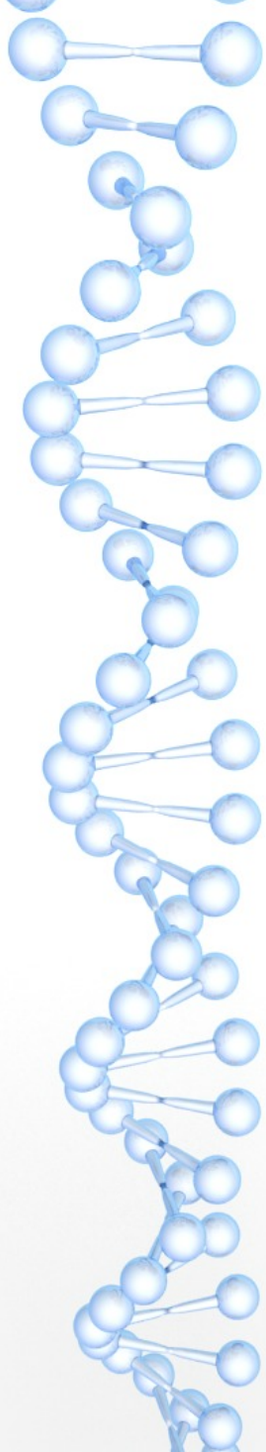
$$Recall = \frac{t_p}{t_p + f_n}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

4. First Implementation & Results

Trained Models





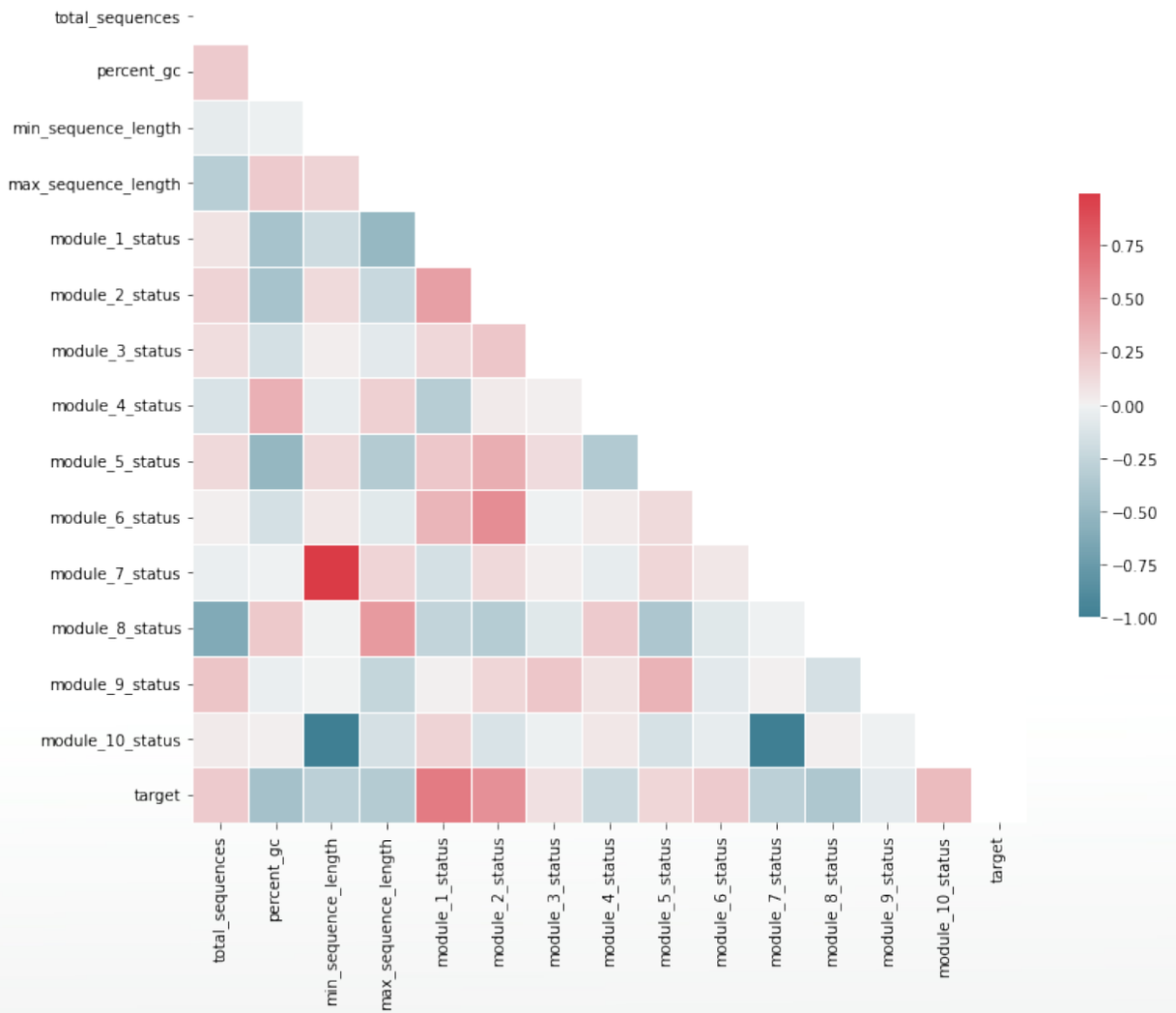
4. First Implementation & Results

Features

Feature Name	Explanation
total_sequences	Number of sequences
percent_gc	Total GC content
min_sequence_length	Length of shortest sequence
max_sequence_length	Length of longest sequence
module_1_status	Per Base Sequence Quality Status
module_2_status	Per Tile Sequence Quality Status
module_3_status	Per Sequence Quality Scores Status
module_4_status	Per Base Sequence Content Status
module_5_status	Per Sequence GC Content Status
module_6_status	Per Base N Content Status
module_7_status	Sequence Length Distribution Status
module_8_status	Sequence Duplication Levels Status
module_9_status	Overrepresented Sequences Status
module_10_status	Adapter Content Status

4. First Implementation & Results

Feature Correlation



4. First Implementation & Results

Support Vector Machine Peformance

Performance Measures for SVM of the whole dataset in a Cross Validation with 10 folds:

	Accuracy	Precision	Recall	F1-Score
Average	0.68	0.67	0.95	0.78
Deviation	0.14	0.11	0.14	0.1

- Great Recall (true positive rate)
- Acceptable Precision & F1-Score

4. First Implementation & Results

Decision Tree Peformance

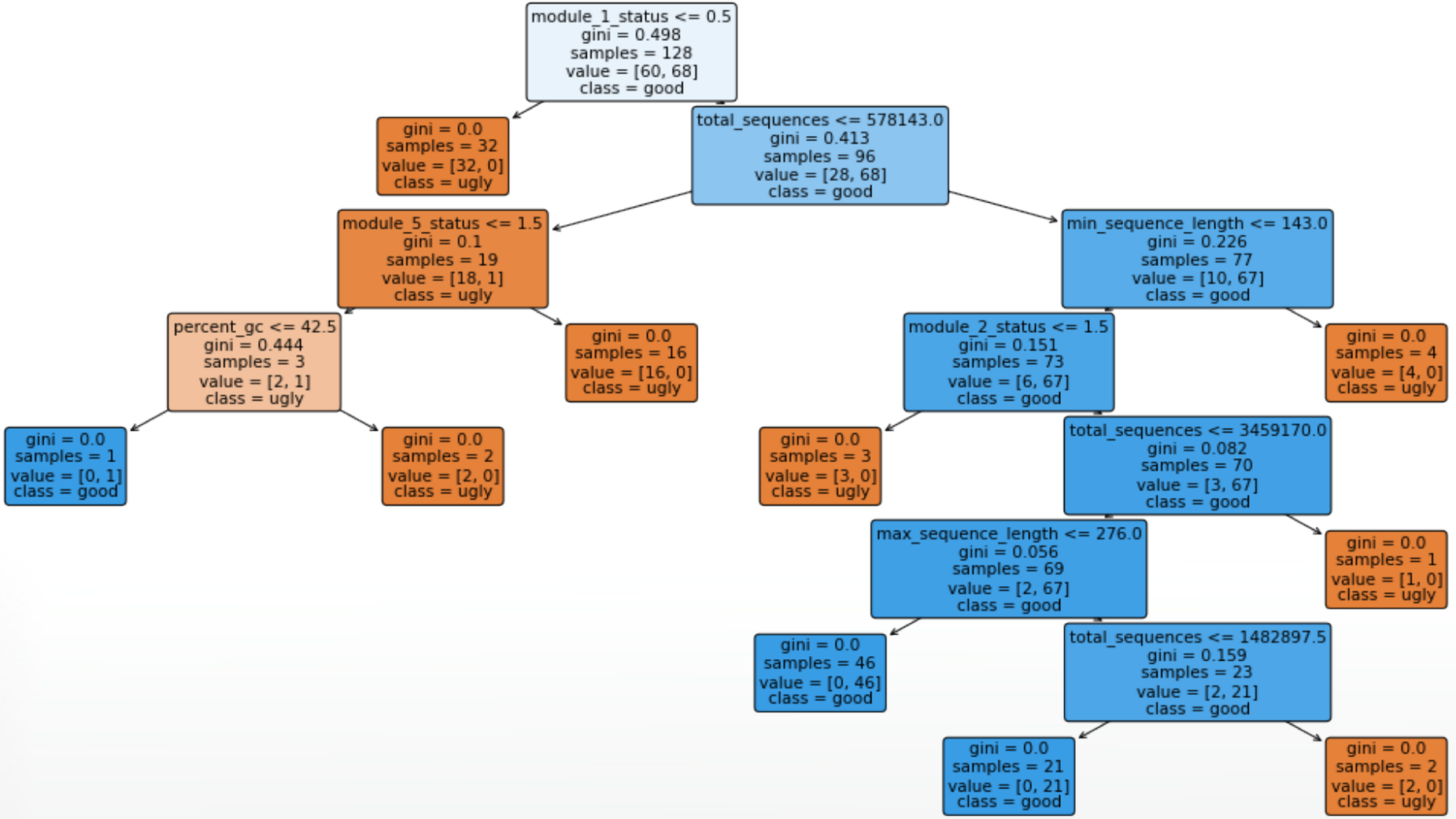
Performance Measures for Decision Tree of the whole dataset in a Cross Validation with 10 folds:

	Accuracy	Precision	Recall	F1-Score
Average	0.93	0.93	0.98	0.96
Deviation	0.08	0.11	0.05	0.06

- Great Overall Performance
- Lower deviations than SVM Cross Validation

4. First Implementation & Results

Decision Tree Insights



4. First Implementation & Results

Random Forest Peformance

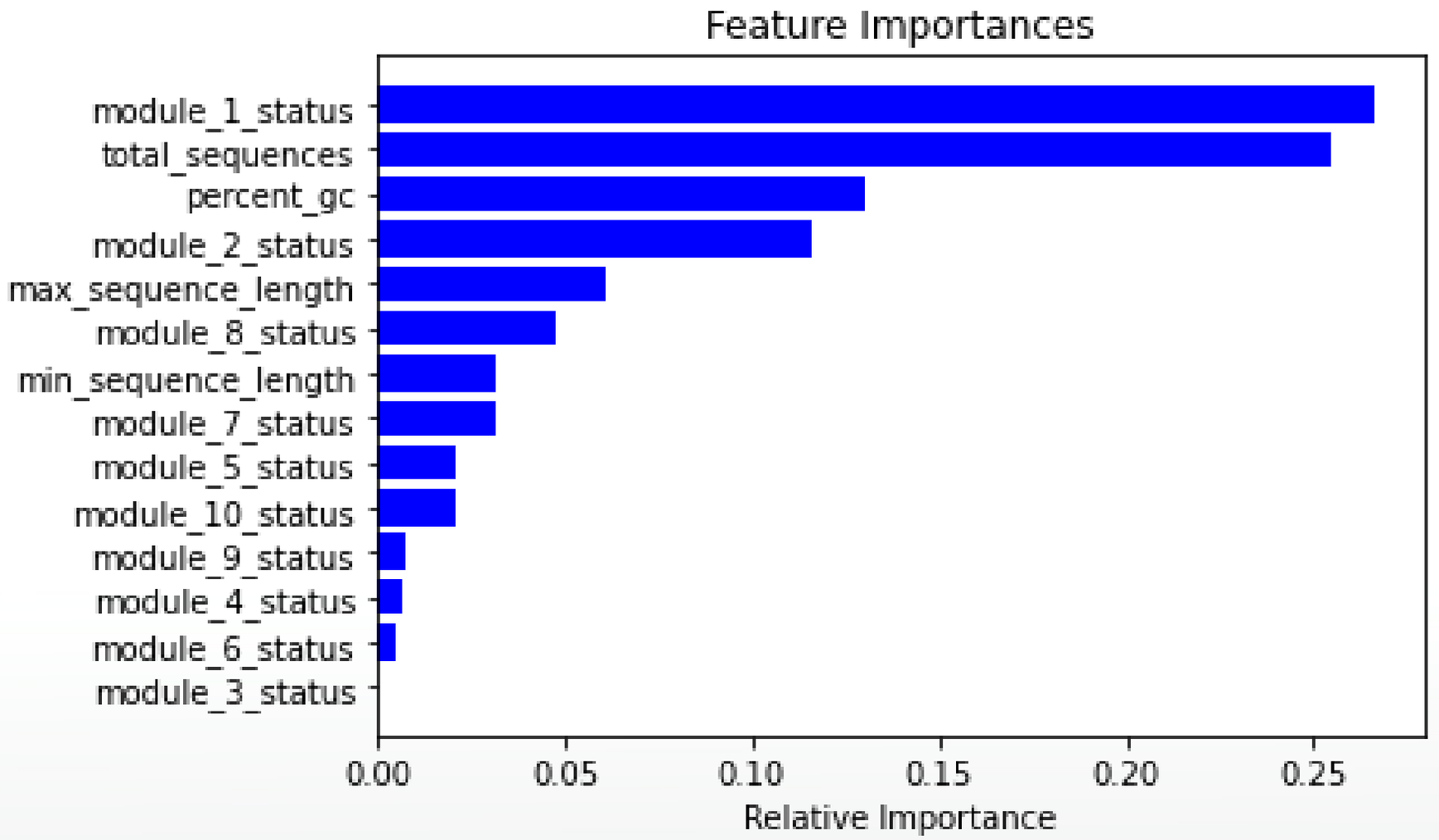
Performance Measures for Random Forest (with 100 decision trees) of the whole dataset in a Cross Validation with 10 folds (compared to Decision Tree Performance):

	Accuracy	Precision	Recall	F1-Score
Average	0.96 (+3%)	0.96 (+3%)	0.98 (+/-0)	0.97 (+/-0)
Deviation	0.06 (-2%)	0.07 (-4%)	0.05 (+/-0)	0.05 (-1%)

- Slight improvement of performance
- Lower deviations than Random Forest

4. First Implementation & Results

Random Forest Insights

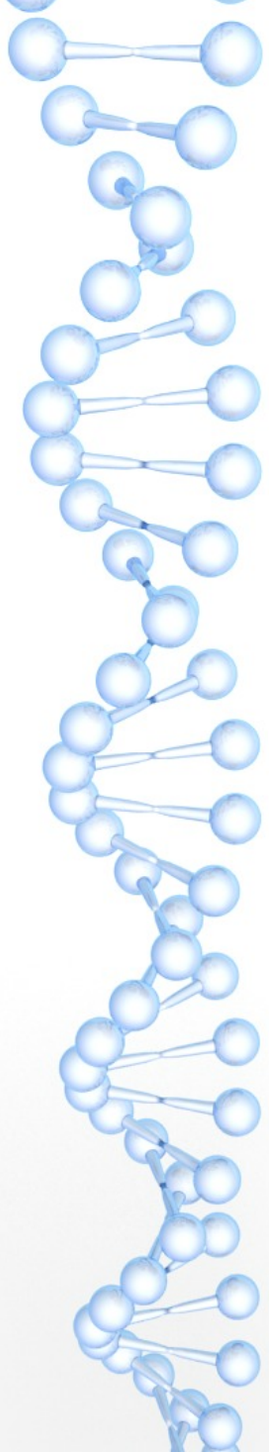


4. First Implementation & Results

Organism Specific Models

Model	Accuracy	Precision	Recall	F_1-Score
Complete (SVM)	0.68	0.67	0.95	0.78
E. faecium (SVM)	0.86	0.86	0.98	0.91
S. aureus (SVM)	0.62	0.73	0.62	0.63
E. coli (SVM)*	1.0*	0.4*	0.4*	0.4*
Complete (Decision Tree)	0.93	0.93	0.98	0.95
E. faecium (Decision Tree)	0.98	1.0	0.96	0.97
S. aureus (Decision Tree)	0.96	0.97	0.98	0.97
E. coli (Decision Tree)*	1.0*	0.2*	0.2*	0.2*
Complete (Random Forest)	0.96	0.96	0.98	0.97
E. faecium (Random Forest)	1.0	1.0	1.0	1.0
S. aureus (Random Forest)	0.99	1.0	0.98	0.99
E. coli (Random Forest)*	1.0*	0.2*	0.2*	0.2*

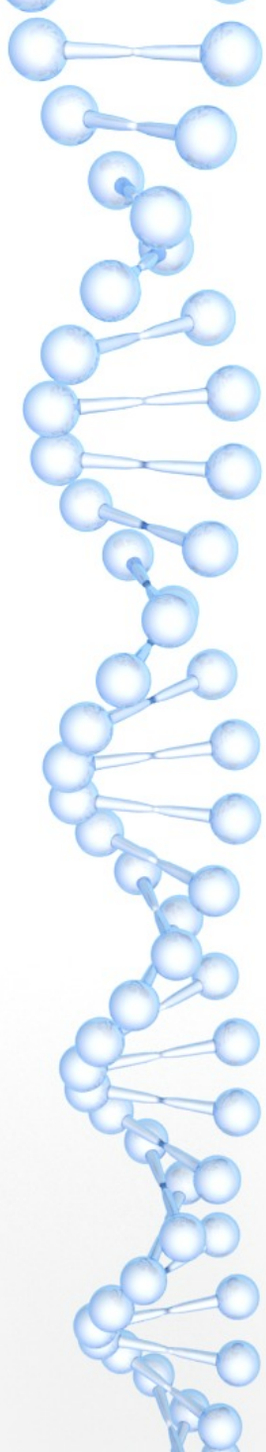
* size of E. coli dataset was too small for useful performance measures



4. First Implementation & Results

Conclusion of the First Part

- Machine learning can be used to assess the quality of NGS data
- Performances of the models were (almost) perfect
- Used data was very clear cut
- Different data could show worse performances



Part II

1. Feature Engineering

2. Pipeline for NGS data quality evaluation

5. Feature Engineering of FastQC Modules

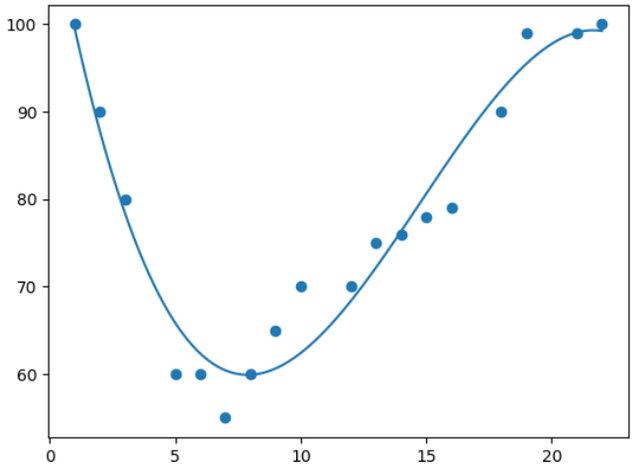
What is Feature Engineering?

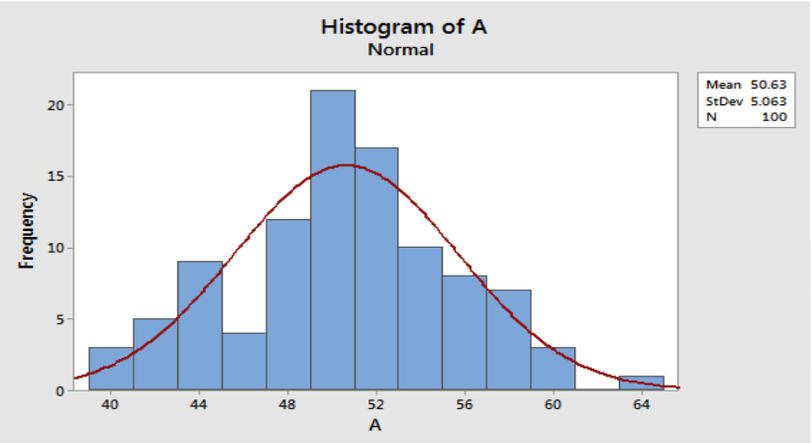
- ML needs features as input
- Features are numerical values
- Features represent the data in a way relevant to the task

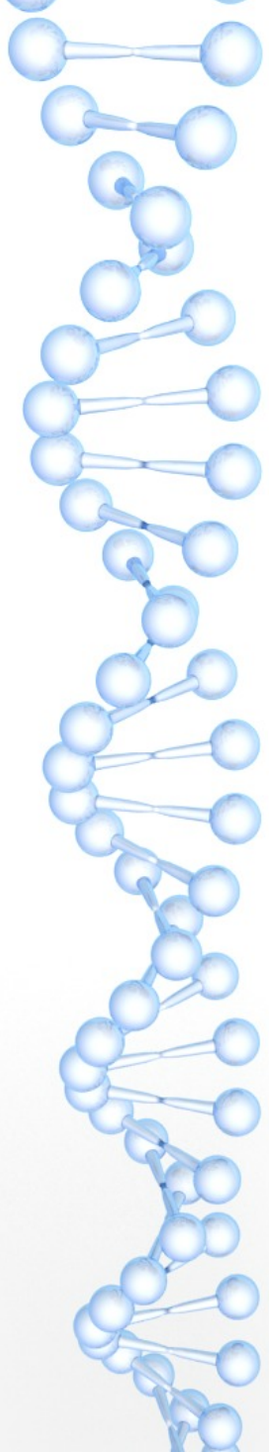
- Domain knowledge and mathematical/statistical operations transform raw data to features

Feature Engineering of Continuous Data

- Summarize continuous data in a few features
- Fit the data to polynomial functions
- Analyze distribution of data
  - Calculate distribution parameters







5. Feature Engineering of FastQC Modules

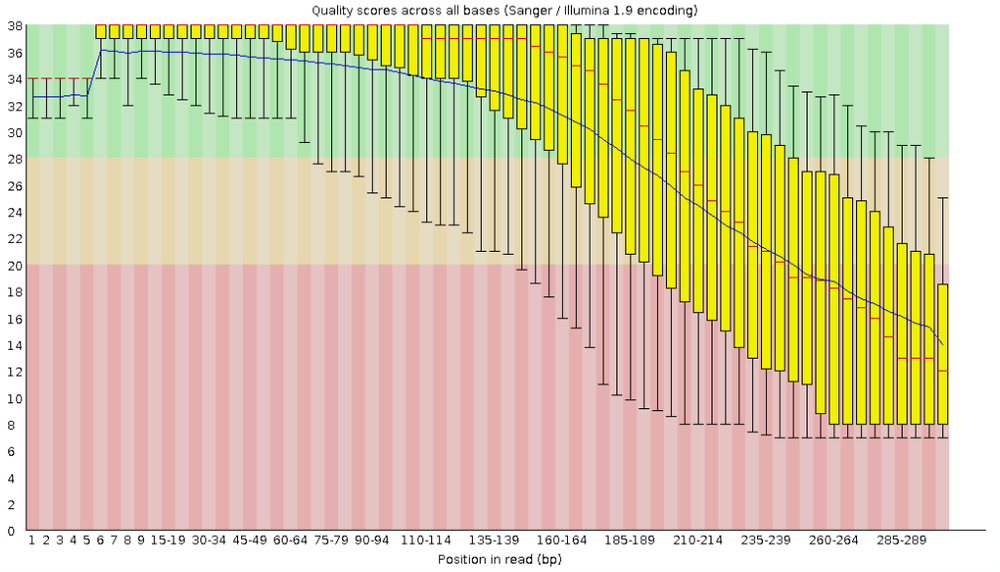
Module 0 - Basic Statistics

- Features already extracted in part I
  - *total_sequences*
  - *percent_gc*
  - *min_sequence_length*
  - *max_sequence_length*

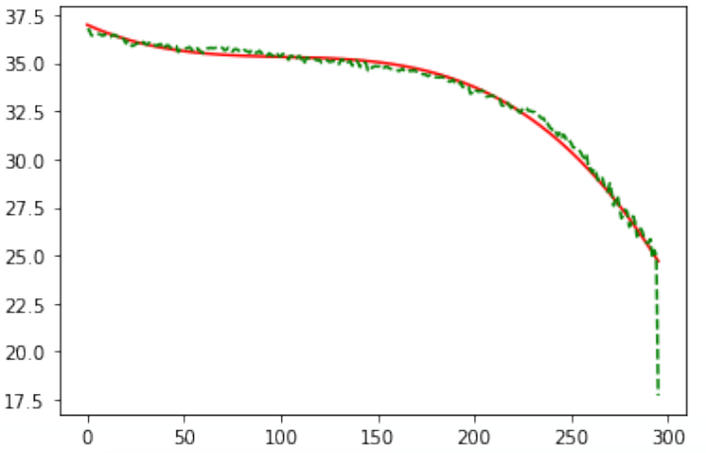
5. Feature Engineering of FastQC Modules

Module 1 – Per Base Sequence Quality

- Box-Whisker Plot (median, **mean**, quartiles, 10/90-percentiles) of the qualities at each position



$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$



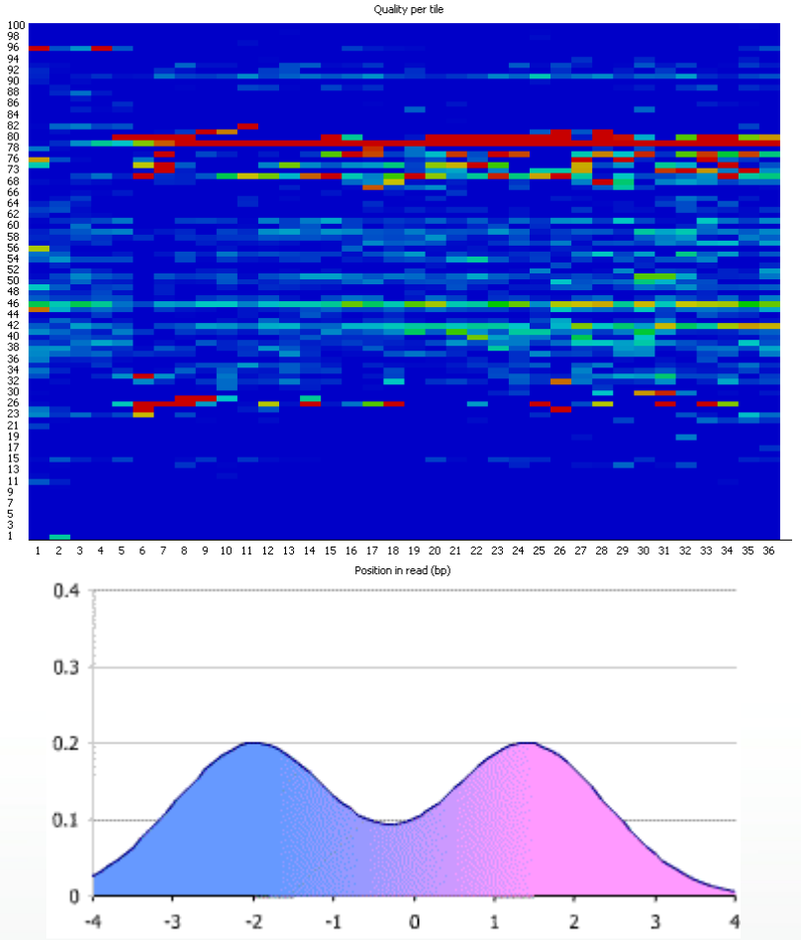
- Features:

  module_1_a0, *module_1_a1*, *module_1_a2* and *module_1_a3*

5. Feature Engineering of FastQC Modules

Module 2 – Per Tile Sequence Quality

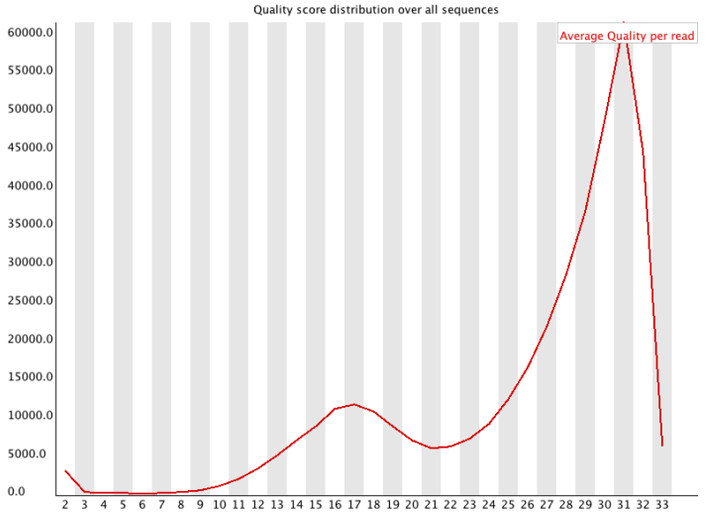
- Shows deviations of quality in each tile from the average quality of a position
- Summarize data per tile
  - Bimodal normal distribution split by 0
- Features:
  - *module_2_std_neg*
    module_2_std_pos

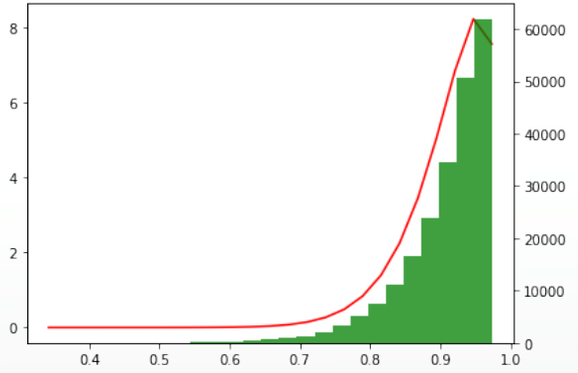


5. Feature Engineering of FastQC Modules

Module 3 – Per Sequence Quality Scores

- Distribution of the quality scores across all reads
- Can be fit to a beta distribution
- Beta distribution parameters:
  - α, β
- Features:
  - *module_3_alpha, module_3_beta*

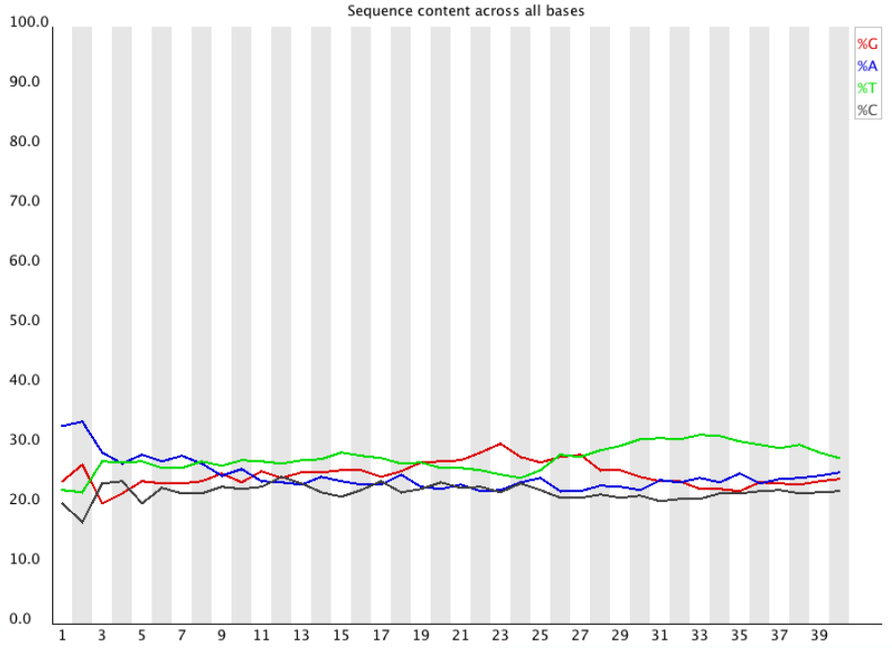




5. Feature Engineering of FastQC Modules

Module 4 – Per Base Sequence Content

- Proportion of each base per position
- Combine this and the next GC module
- Combine changes of proportion for all bases → normal distribution

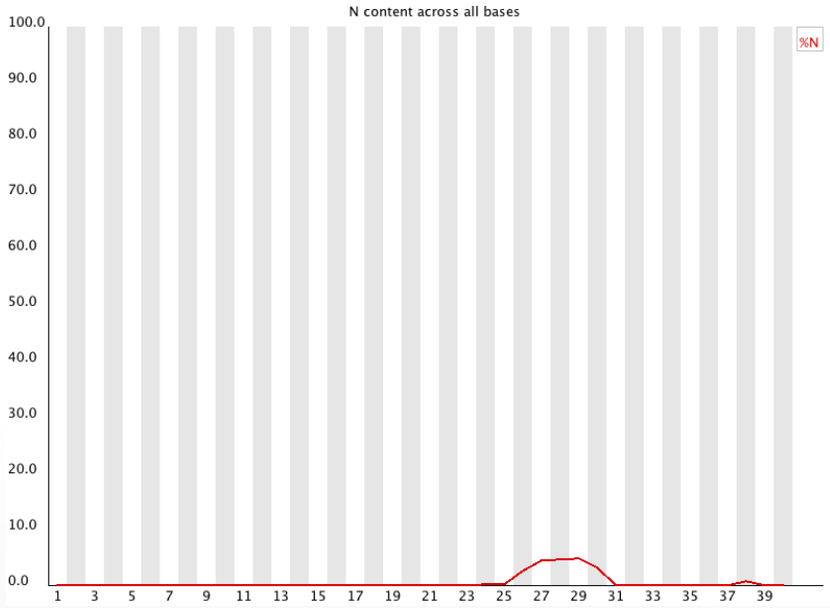


- Features: *module_4_diff_mean*, *module_4_diff_std*

5. Feature Engineering of FastQC Modules

Module 6 – Per Base N Content

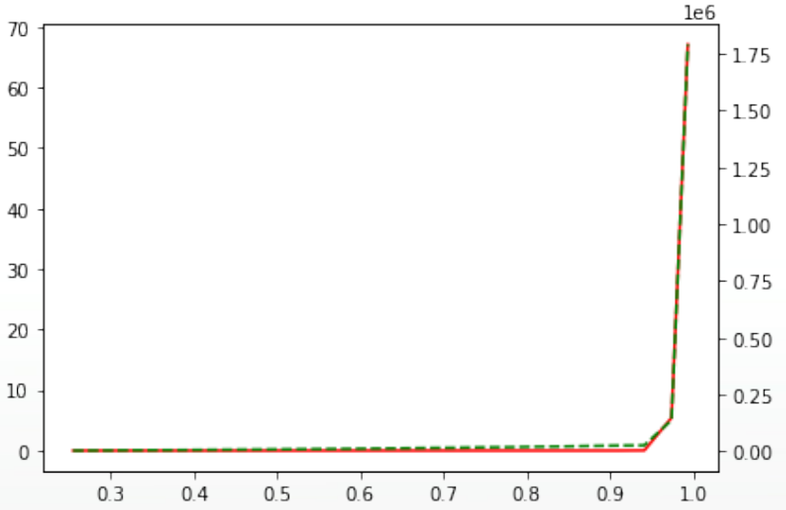
- Percentages of unsuccessful base calls per position
- Total percentage of N sufficient
- Feature:
  - *module_6_n_content*

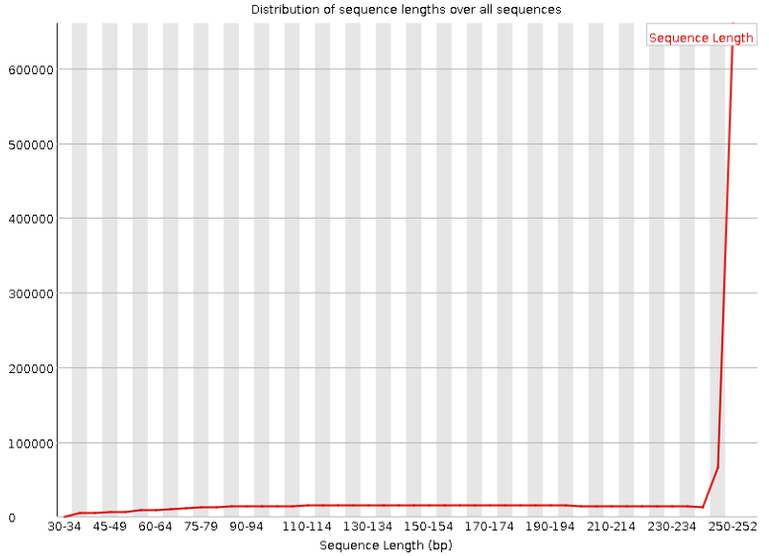


5. Feature Engineering of FastQC Modules

Module 7 – Sequence Length Distribution

- Shows distribution of the sequence lengths
- Fit beta distribution



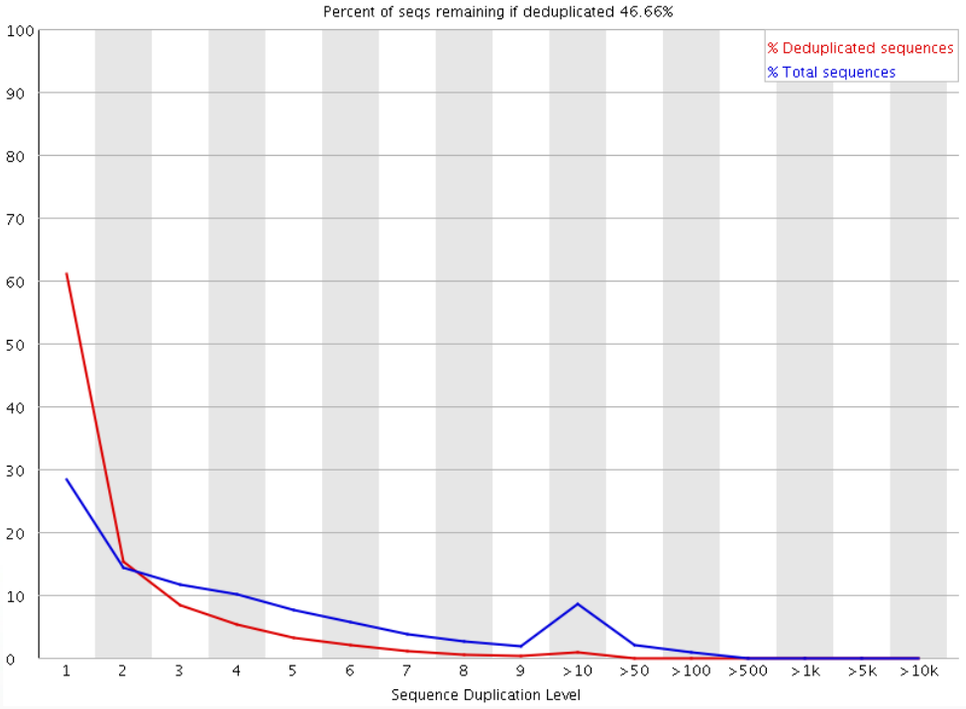


- Features:
  - *module_7_alpha*
    module_7_beta

5. Feature Engineering of FastQC Modules

Module 8 – Sequence Duplication Levels

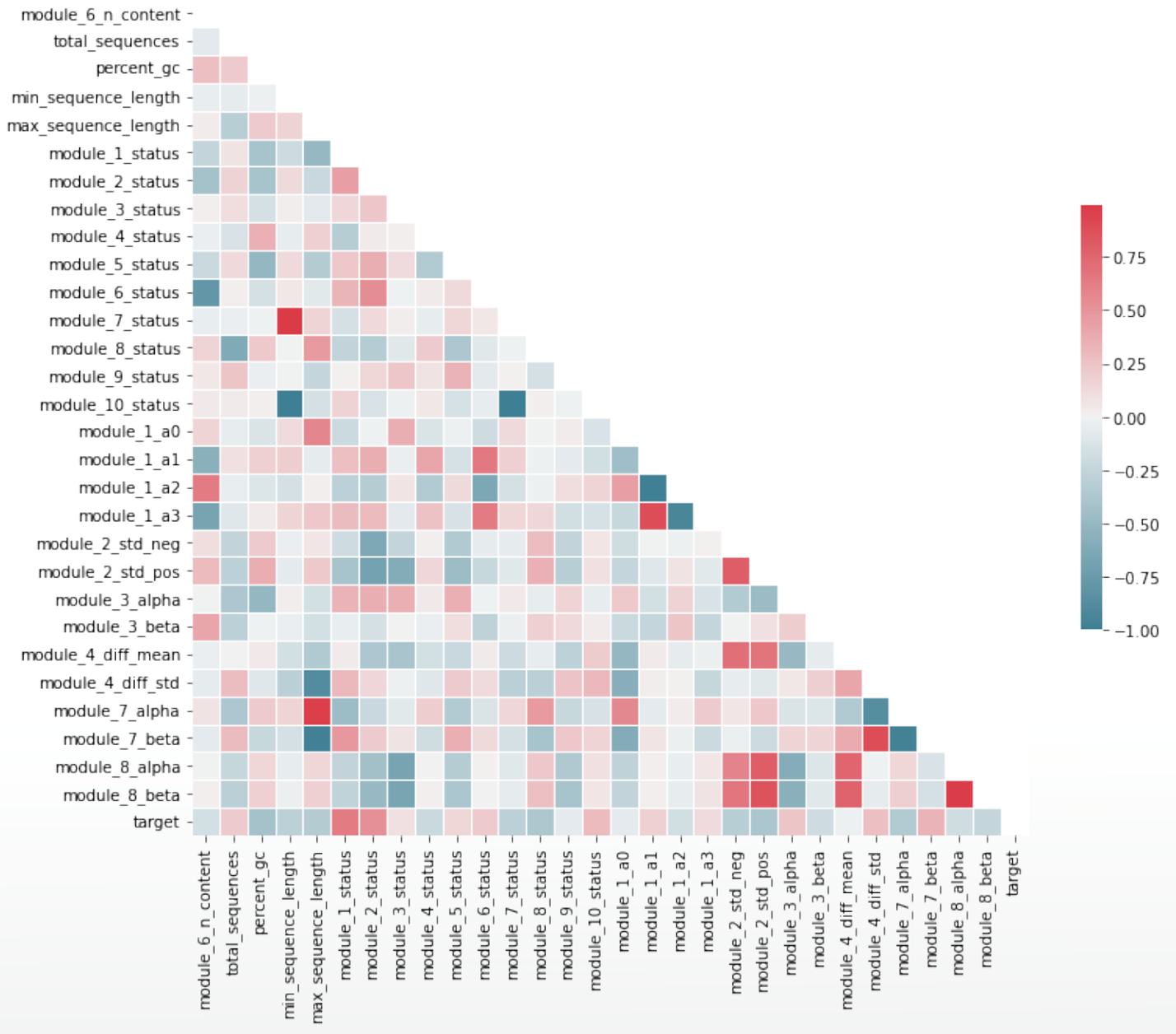
- Shows distribution of duplication levels of sequences
- Fit Beta distribution
- Features
  - *module_8_alpha*
    module_8_beta

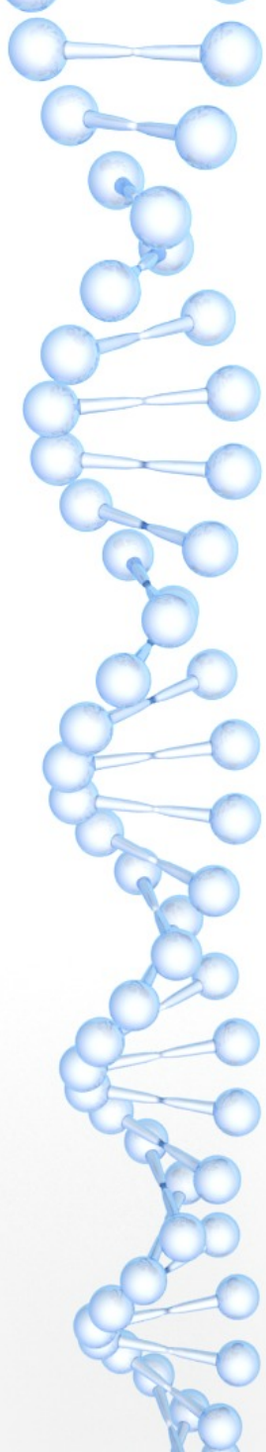


Module 9 – Overrepresented Sequences
Module 10 – Adapter Content

- Overrepresented Sequences already represented in module 8
- Adapter content no valuable information for given data
- No further features

Feature Correlation





6. Final Results and Pipeline Implementation

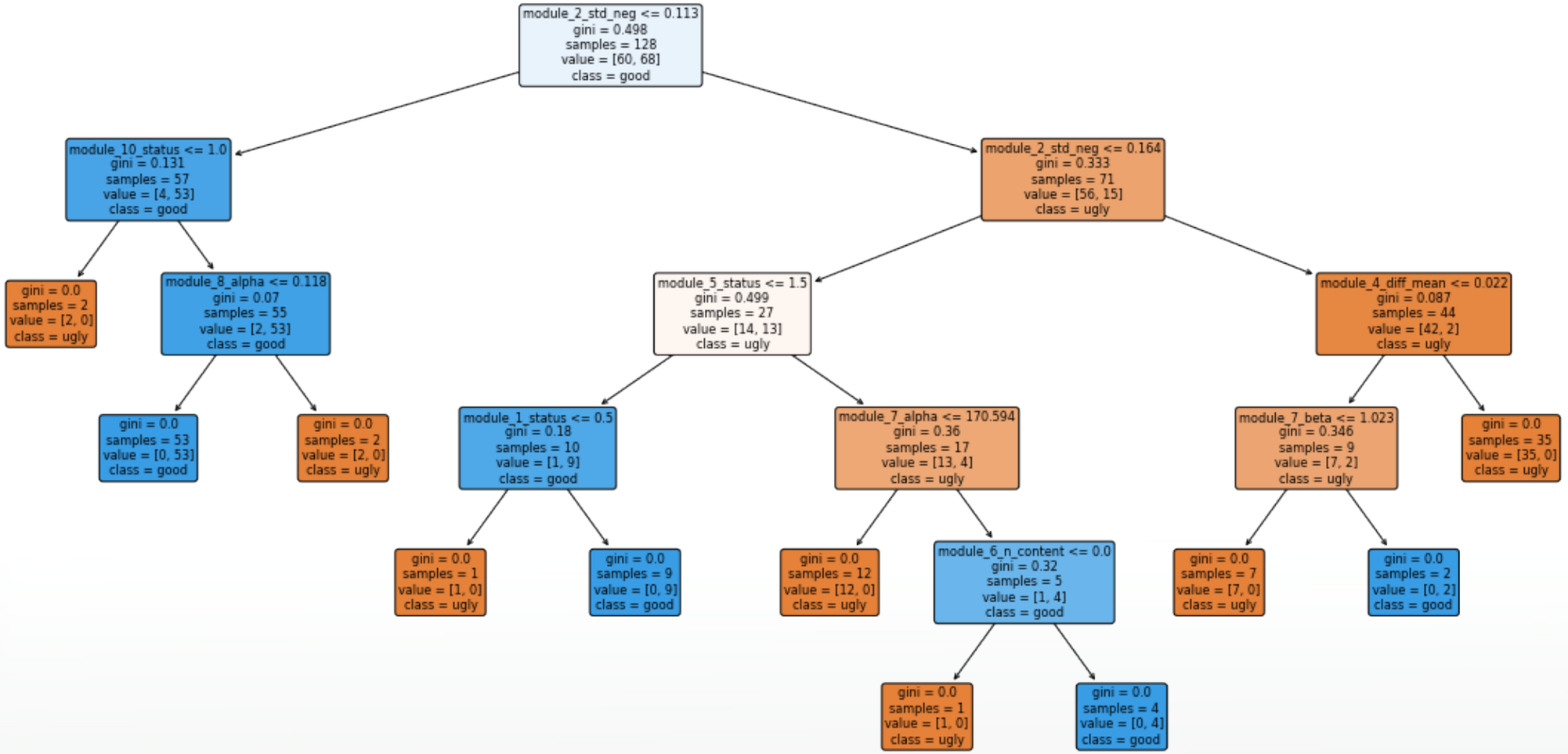
Model Performances

Performance Measures for the models of the whole dataset in a Cross Validation with 10 folds
First row: new models, second row: old models

	Accuracy	Precision	Recall	F1-Score
SVM	0.69 (0.68)	0.68 (0.67)	0.97 (0.95)	0.79 (0.78)
Decision Tree	0.89 (0.93)	0.89 (0.93)	0.95 (0.98)	0.91 (0.95)
Random Forest	0.93 (0.96)	0.93 (0.96)	0.96 (0.98)	0.94 (0.97)

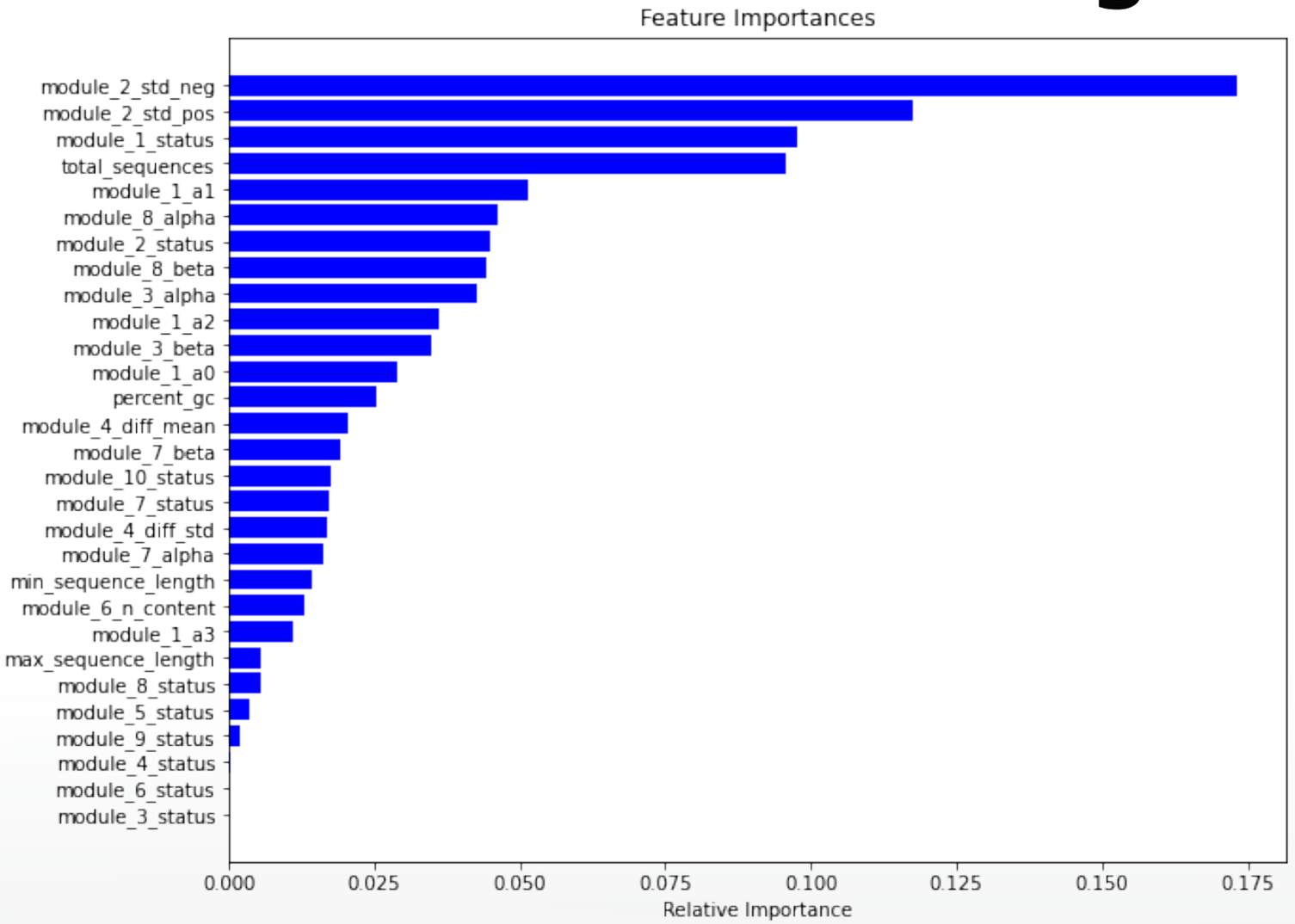
6. Final Results and Pipeline Implementation

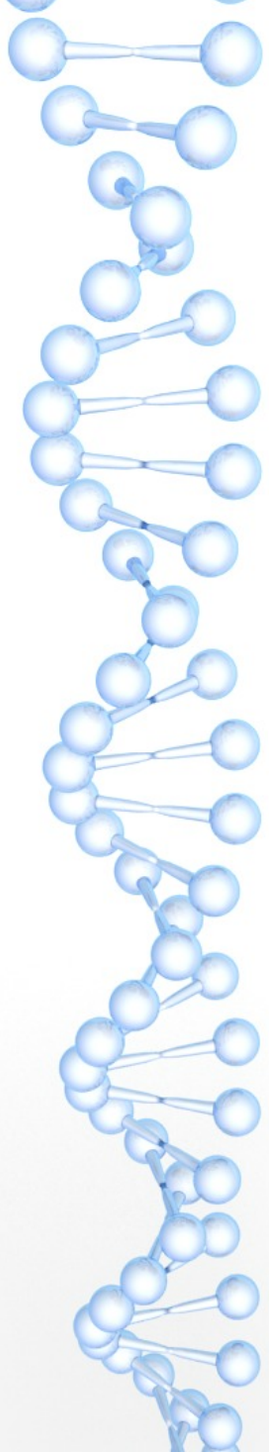
Decision Tree Insights



6. Final Results and Pipeline Implementation

Random Forest Insights





6. Final Results and Pipeline Implementation

Organism Specific Models

Model	Accuracy	Precision	Recall	F_1-Score
Complete (SVM)	0.69 (0.68)	0.68 (0.67)	0.97 (0.95)	0.79 (0.78)
E. faecium (SVM)	0.78 (0.86)	0.81 (0.86)	0.92 (0.98)	0.82 (0.91)
S. aureus (SVM)	0.81 (0.62)	0.83 (0.73)	0.93 (0.62)	0.87 (0.63)
E. coli (SVM)*	1.0 (1.0)*	0.2 (0.4)*	0.2 (0.4)*	0.2 (0.4)*
Complete (Decision Tree)	0.89 (0.93)	0.89 (0.93)	0.95 (0.98)	0.91 (0.95)
E. faecium (Decision Tree)	0.91 (0.98)	0.94 (1.0)	0.92 (0.96)	0.92 (0.97)
S. aureus (Decision Tree)	0.95 (0.96)	0.98 (0.97)	0.94 (0.98)	0.96 (0.97)
E. coli (Decision Tree)*	0.9 (1.0)*	0.1 (0.2)*	0.1 (0.2)*	0.1 (0.2)*
Complete (Random Forest)	0.93 (0.96)	0.93 (0.96)	0.96 (0.98)	0.94 (0.97)
E. faecium (Random Forest)	0.95 (1.0)	0.95 (1.0)	0.98 (1.0)	0.96 (1.0)
S. aureus (Random Forest)	0.96 (0.99)	0.97 (1.0)	0.98 (0.98)	0.97 (0.99)
E. coli (Random Forest)*	0.9 (1.0)*	0.0 (0.2)*	0.0 (0.2)*	0.0 (0.2)*

Pipeline Implementation

- Python Script I - Training:
  - FastQC analysis
    Data extraction
    Data preparation
    Feature engineering
    Model training
- Python Script II - Evaluation
  - Integrated in Nextflow process
  - Input: FASTQ file, Output: quality evaluation

Conclusion

- Engineered features represent complex data of modules
- These features separate the classes well
- Second implementation performs slightly worse
- Nextflow process allows integration into pipelines
- Decision tree and feature importance offer new insights

Machine learning can be used to automate the quality assessment of NGS data.

Sources

Simon Andrews. *Babraham Bioinformatics - FastQC Documentation*. url: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (accessed 30.03.2022)

David P. Clark, Nanette Jean. Pazdernik und Michelle R. McGehee. *„Next Generation Sequencing“. In: Molecular biology.* Academic Cell, 2019.

Géron Aurélien. *Hands-on machine learning with SCIKIT-learn, Keras, and tensorflow: Concepts, tools, and Techniques*. 2. Ed. O’REILLY MEDIA, 2019.

William S Noble. *„What is a support vector machine?“* In: Nature Biotechnology 24.12 (Dec. 2006), p. 1565–1567. doi: 10.1038/nbt1206-1565.

Müller Andreas C. und Sarah Guido. *Introduction to machine learning with python: A guide for data scientists*. 2. Ed. O’Reilly Media, Inc., 2017.

Fatih Karabiber. *Gini impurity*. url: https://www.learndatasci.com/glossary/gini-impurity/ (accessed 30. 03. 2022).

Michael L. Waskom. *„seaborn: statistical data visualization“.* In: Journal of Open Source Software 6.60 (2021), p. 3021. doi: 10.21105/joss.03021. url: https://doi.org/10.21105/joss.03021.

Pablo Duboue. *The Art of Feature Engineering: Essentials for Machine Learning*. 1. Ed. Cambridge University Press, 2020.

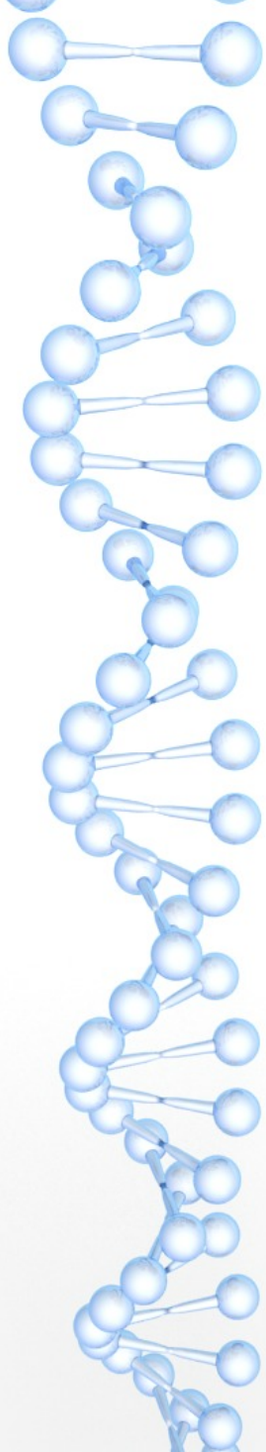
J. D. Hunter. *„Matplotlib: A 2D graphics environment“*. In: Computing in Science & Engineering 9.3 (2007), p. 90–95. doi: 10.1109/MCSE.2007.55.

Pauli Virtanen u. a. *„SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“*. In: Nature Methods 17 (2020), p. 261–272. doi: 10.1038/s41592-019-0686-2.

F. Pedregosa u. a. *„Scikit-learn: Machine Learning in Python“*. In: Journal of Machine Learning Research 12 (2011), p. 2825–2830.

Image Sources

Slide 5 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (accessed 05-Apr-2022)
Slide 6 - https://www.shutterstock.com/de/image-illustration/robot-child-machine-learninig-book-sketch-1524562841 (accessed 05-Apr-2022)
Slide 9 - https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d (accessed 05-Apr-2022)
Slide 13 - http://search.coolclips.com/m/vector/vc066196/man-pruning-a-tree/ (accessed 05-Apr-2022)
Slide 14 - https://blog.paperspace.com/random-forests/ (accessed 05-Apr-2022)
Slide 15 - http://ethen8181.github.io/machine-learning/model_selection/img/kfolds.png (accessed 05-Apr-2022)
Slide 28 - https://cdn.analyticsvidhya.com/wp-content/uploads/2018/07/data-engineer.jpg (accessed 05-Apr-2022)
Slide 29 - https://www.w3schools.com/python/python_ml_polynomial_regression.asp (accessed 05-Apr-2022)
Slide 29 - https://statisticsbyjim.com/basics/histograms/ (accessed 05-Apr-2022)
Slide 31 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (accessed 05-Apr-2022)
Slide 31 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (accessed 05-Apr-2022)
Slide 32 - https://en.wikipedia.org/wiki/Multimodal_distribution#/media/File:Bimodal.png (accessed 05-Apr-2022)
Slide 33 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help (accessed 05-Apr-2022)
Slide 34 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (accessed 05-Apr-2022)
Slide 35 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ (accessed 05-Apr-2022)
Slide 36 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help (accessed 05-Apr-2022)
Slide 37 - https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help (accessed 05-Apr-2022)
Slide 44 - https://www.nextflow.io/index.html (accessed 05-Apr-2022)
Slide 44 - https://de.wikipedia.org/wiki/Datei:Python-logo-notext.svg (accessed 05-Apr-2022)



Questions