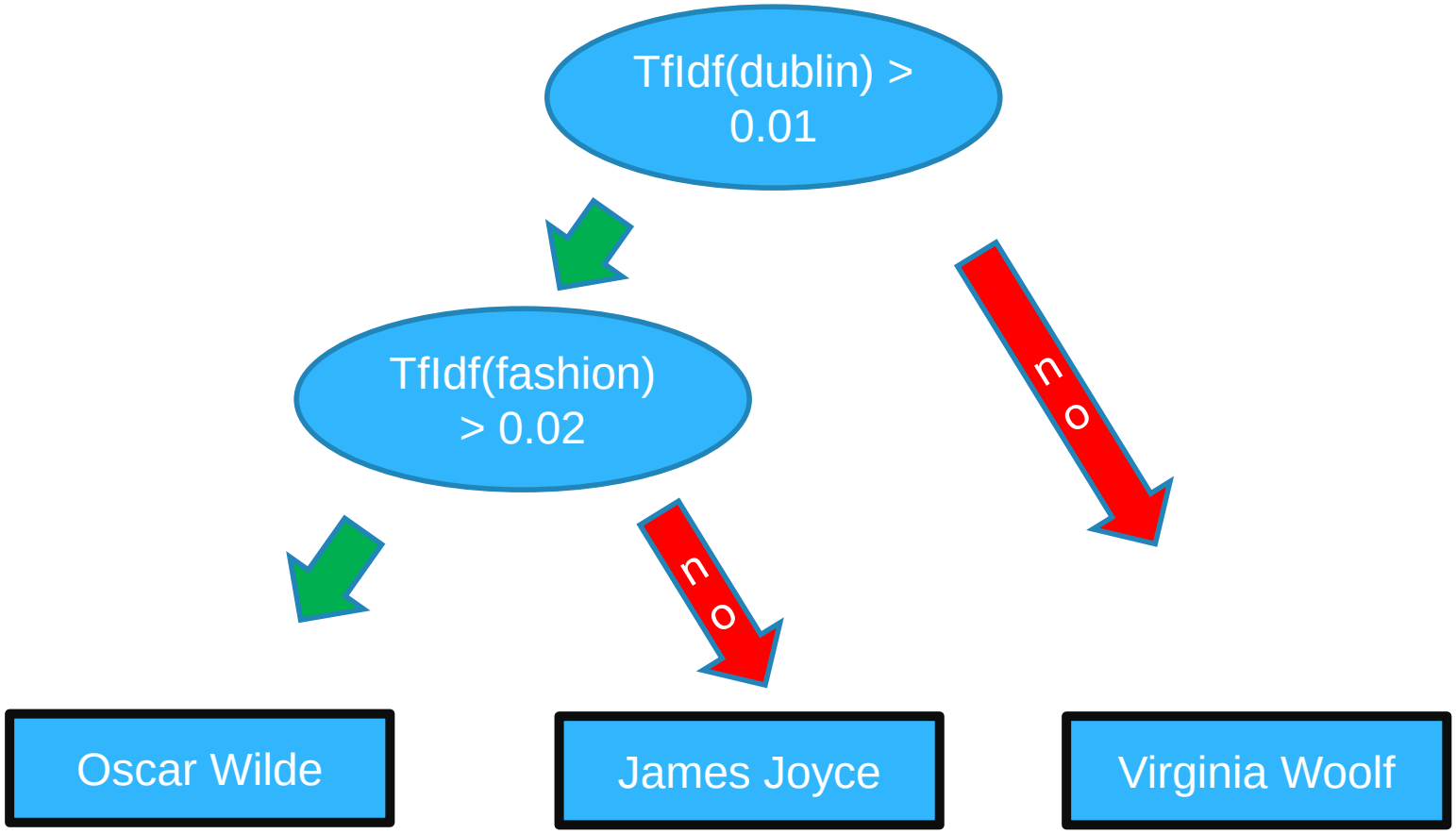
SEMINAR KLASSIFIKATION & CLUSTERING

DECISION TREES

Stefan Langer

stefan.langer@cis.uni-muenchen.de

Entscheidungsbäume (decision trees)



Featuretypen

- Kontinuierlich (größer, kleiner, gleich-Relation)
  - z.B. in unserem Fall TF-IDF
- Nominal / Bool
  - z.B. Wort existiert / existiert nicht) entspricht numerisch > 0

Entscheidungsalgorithmus

- Entscheidungsalgorithmus sehr einfach.
- Arbeite den Baum von oben nach unten ab, bis ein Blatt erreicht ist
- Blatt = Klasse

Baumaufbau: Rekursive Partitionierung

- Top-Down Algorithmus
- Es wird jeweils das Feature verwendet, das den Baum das Datenset am besten unterteilt.

- Mehrere mögliche Algorithmen, z.B.
  - Gini-Unreinheit (Gini-Index)
  - Entropy (information gain)

ID 3

- Beginne am Wurzelknoten
- Iteration: Berechne auf dem optimalsten der noch nicht benutzes Attribut die beste Unterteilung
- Optimal heißt: Beste Unterteilung aufgrund Information Gain oder Gini Impurity
- Wiederhole, bis der Baum aufgebaut ist

C4.5

- Aufbauend auf ID-3, behandelt aber auch kontinuierliche Werte (wie etwa TF-IDF) durch Aufteilung in Intervalle

Gini impurity

- Auch ‘GINI index’ nicht identisch mit Gini-Koeffizient
- Ein Maß dafür wie oft ein zufällig gewähltes Element einer Menge inkorrekt gelabelt würde, rein auf Basis der Verteilung von Labels.
- Minimal, wenn alle Elemente eines Knoten in zu einer Kategorie gehören

Gini impurity: Formel

In einer Menge von Objekten mit J Klassen sind f_i die Items, die mit der Klasse i gelabelt sind.

Dann ist die Gini-Unreinheit dieser Menge:

$$\sum_{i=1}^{J} p_i * (1 - p_i)$$

J: Die Zahl der Klassen
p_i ist die Wahrscheinlichkeit (= relative Häufigkeit) einer Klasse.

Ziel: Gini-Unreinheit optimieren, d.h. wie suchen den Split in zwei Untermengen, der die gewichtete Gini-Unreinheit beider Untermengen am niedrigsten macht.

Information gain (reduce Entropy)

Deutsch: Kullback-Leibler-Divergenz

Maximale Reduktion von Entropie, nach nachfolgender Formel:

$$-\sum_{i=1}^{J} p_i * log2(p_i)$$

Decision tree parameters in sklearn

- **criterion**: {"gini", "entropy"}, default="gini"
  - The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
  - Siehe nächste Folien
- **splitter**: {"best", "random"}, default="best"
  - The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

Parameter (weitere)

- **max_depth**: int, default=None
  - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split** int or float, default=2
  - The minimum number of samples required to split an internal node
  - If int, then consider min_samples_split as the minimum number.
  - If float, then min_samples_split is a fraction
- **min_samples_leaf**: int or float, default=1

  The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

  If int, then consider min_samples_leaf as the minimum number.

  If float, then min_samples_leaf is a fraction

Parameter (weitere)

- **min_weight_fraction_leaf**: float, default=0.0
  - The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features**: int, float or {"auto", "sqrt", "log2"}, default=None
  - The number of features to consider when looking for the best split.
  - If int, then consider max_features features at each split.
  - If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
  - If "auto", then max_features=sqrt(n_features).
  - If "sqrt", then max_features=sqrt(n_features).
  - If "log2", then max_features=log2(n_features).
  - If None, then max_features=n_features.

Parameter (weitere)

- **random_state**: int, RandomState instance or None, default=None
  - Controls the randomness of the estimator. The features are always randomly permuted at each split, even if splitter is set to "best". When max_features < n_features, the algorithm will select max_features at random at each split before finding the best split among them. But the best found split may vary across different runs, even if max_features=n_features. That is the case, if the improvement of the criterion is identical for several splits and one split has to be selected at random. To obtain a deterministic behaviour during fitting, random_state has to be fixed to an integer. See Glossary for details.
- **max_leaf_nodes**: int, default=None
  - Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

Parameter (weitere)

- **class_weight**: dict, list of dict or "balanced”, default=None
  - Weights associated with classes in the form {class_label: weight}. If None, all classes are supposed to have weight one
  - Note that for multioutput (including multilabel) weights should be defined for each class of every column in its own dict
  - The "balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))
  - For multi-output, the weights of each column of y will be multiplied.
  - Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

Parameter (weitere)

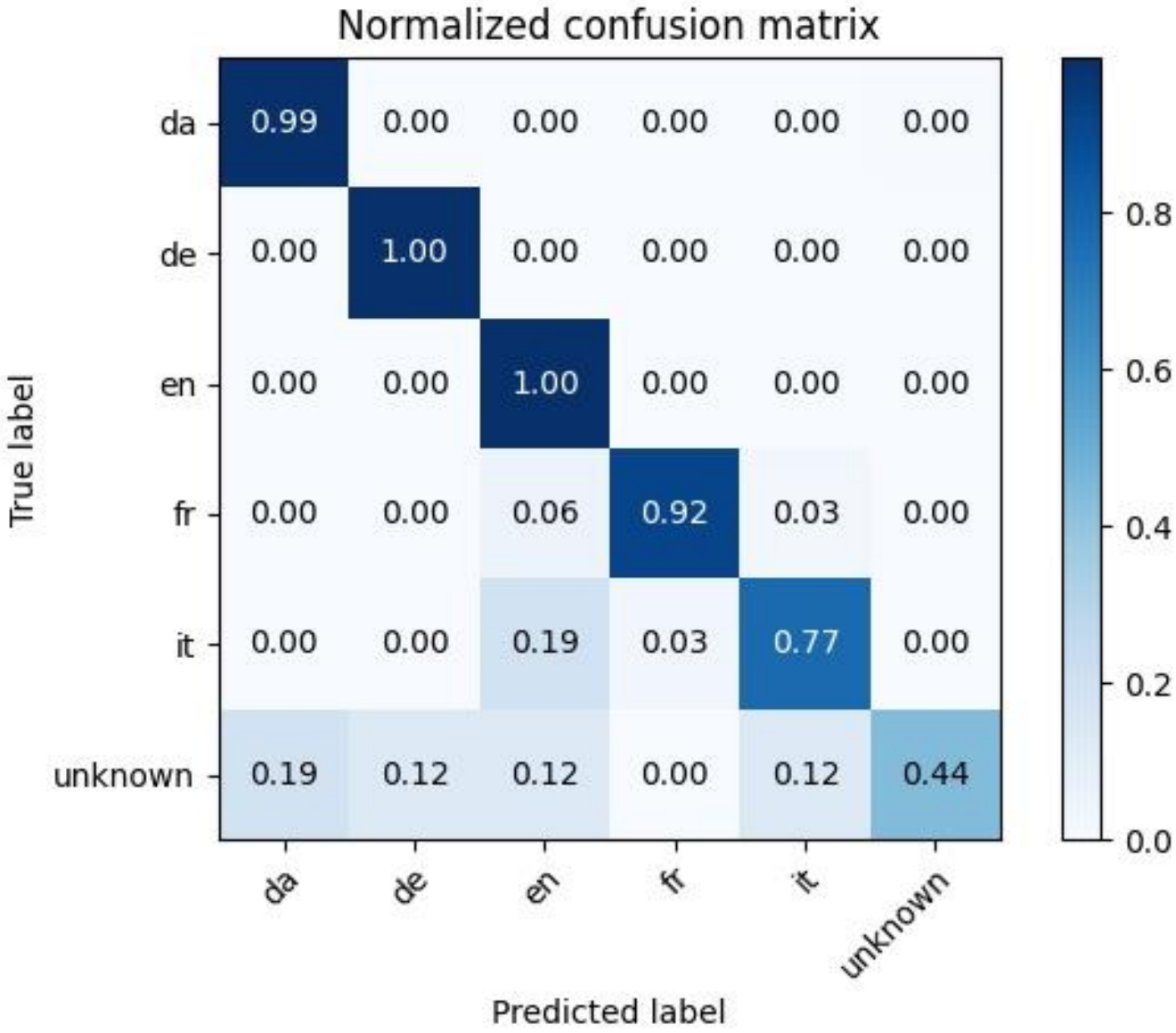
- **ccp_alpha**: non-negative float, default=0.0
  - Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed. See Minimal Cost-Complexity Pruning for details.
- **min_impurity_decrease**: float, default=0.0
  - A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
  - The weighted impurity decrease equation is the following:
  - N_t / N * (impurity - N_t_R / N_t * right_impurity - N_t_L / N_t * left_impurity) where N is the total number of samples, N_t is the number of samples at the current node, N_t_L is the number of samples in the left child, and N_t_R is the number of samples in the right child.
  - N, N_t, N_t_R and N_t_L all refer to the weighted sum, if sample_weight is passed.

Evaluation set

```
Number of training data_records: 39077
Number of classified data_records: 4881
```

DecisionTreeClassifier: lang

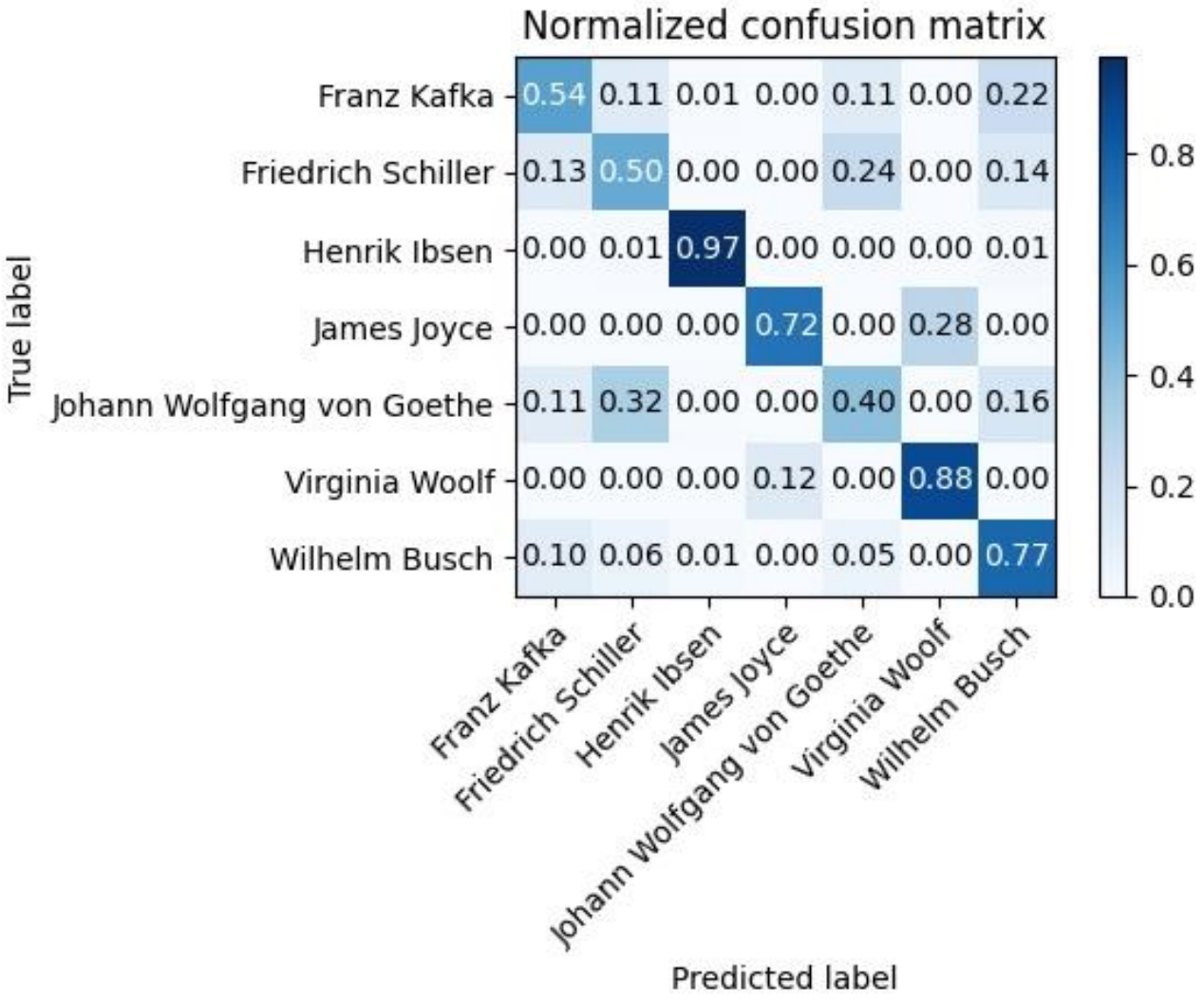
```
#Counts:
Number of unique classes in data_records: 6
Number of unique classes found: 6
#Performance:
Seconds used for training: 266
Seconds used for classification: 6
#Classification report:
              precision    recall  f1-score   support

          da       0.99      0.99      0.99       838
          de       1.00      1.00      1.00      1443
          en       0.99      1.00      1.00      2517
          fr       0.97      0.92      0.94        36
          it       0.77      0.77      0.77        31
     unknown       0.64      0.44      0.52        16

    accuracy                           0.99      4881
   macro avg       0.89      0.85      0.87      4881
weighted avg       0.99      0.99      0.99      4881
```


Normalized confusion matrix
True label
Predicted label
da
de
en
fr
it
unknown
0.99 0.00 0.00 0.00 0.00 0.00
0.00 1.00 0.00 0.00 0.00 0.00
0.00 0.00 1.00 0.00 0.00 0.00
0.00 0.00 0.06 0.92 0.03 0.00
0.00 0.00 0.19 0.03 0.77 0.00
0.19 0.12 0.12 0.00 0.12 0.44
0.8
0.6
0.4
0.2
0.0

Decision tree classifier: Authors

```
#Counts:
Number of unique classes in data_records: 7
Number of unique classes found: 7
#Performance:
Seconds used for training: 340
Seconds used for classification: 5
#Classification report:
```

	precision	recall	f1-score	support
Franz Kafka	0.55	0.54	0.55	280
Friedrich Schiller	0.47	0.50	0.48	266
Henrik Ibsen	0.98	0.97	0.98	897
James Joyce	0.68	0.72	0.70	682
Johann Wolfgang von Goethe	0.41	0.40	0.41	228
Virginia Woolf	0.90	0.88	0.89	1901
Wilhelm Busch	0.77	0.77	0.77	627
accuracy			0.80	4881
macro avg	0.68	0.68	0.68	4881
weighted avg	0.80	0.80	0.80	4881


Normalized confusion matrix
True label
Predicted label
Franz Kafka
Friedrich Schiller
Henrik Ibsen
James Joyce
Johann Wolfgang von Goethe
Virginia Woolf
Wilhelm Busch
0.54 0.11 0.01 0.00 0.11 0.00 0.22
0.13 0.50 0.00 0.00 0.24 0.00 0.14
0.00 0.01 0.97 0.00 0.00 0.00 0.01
0.00 0.00 0.00 0.72 0.00 0.28 0.00
0.11 0.32 0.00 0.00 0.40 0.00 0.16
0.00 0.00 0.00 0.12 0.00 0.88 0.00
0.10 0.06 0.01 0.00 0.05 0.00 0.77
0.0
0.2
0.4
0.6
0.8

Random Forest

Ensemble-Methode auf Basis von Entscheidungsbäumen

- Mehrere Entscheidungsbäume
- Training der Entscheidungsbäume aufgrund einer randomisierten Aufteilung der Trainingsdaten
- Training der Entscheidungsbäume aufgrund einer randomisierten Auswahl der Features, die herangezogen werden
- Klassenzuweisung aufgrund einer Mehrheitsentscheidung

Random forest hyperparameters

n_estimators: int, default=100 (earlier versions: 10)

The number of trees in the forest.

bootstrap: bool, default=True

Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

random_state: int, RandomState instance or None, default=None

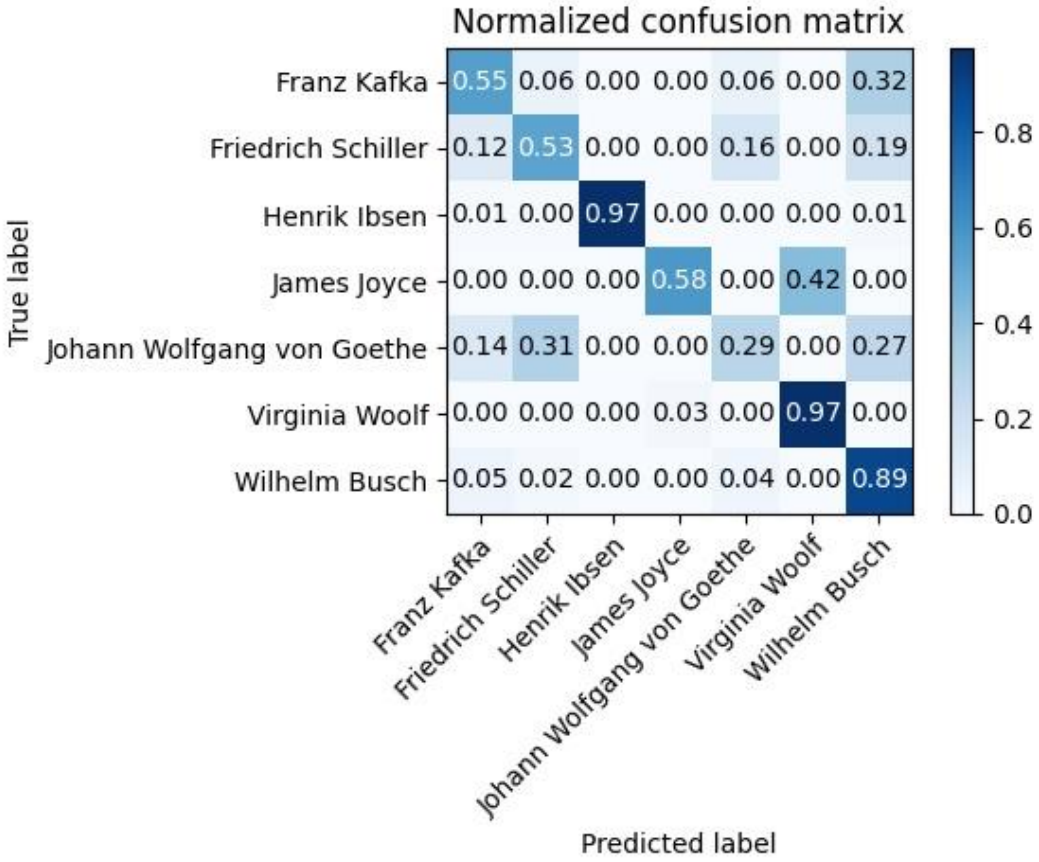
Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap=True) and the sampling of the features to consider when looking for the best split at each node (if max_features < n_features).

RF: authors

```
#Info:
Classifier: RandomForestClassifier, est 20
Label: author
Text label: text
Dense|LSA: False|False

#Counts:
Number of training data_records: 39077
Number of classified data_records: 4881
Number of unique classes in data_records: 7
Number of unique classes found: 7

#Performance:
Seconds used for training: 129
Seconds used for classification: 47

#Classification report:
```

	precision	recall	f1-score	support
Franz Kafka	0.668	0.475	0.555	280
Friedrich Schiller	0.599	0.489	0.538	266
Henrik Ibsen	0.997	0.972	0.984	897
James Joyce	0.956	0.547	0.696	682
Johann Wolfgang von Goethe	0.545	0.237	0.330	228
Virginia Woolf	0.859	0.994	0.922	1901
Wilhelm Busch	0.656	0.944	0.774	627
accuracy			0.828	4881
macro avg	0.754	0.665	0.686	4881
weighted avg	0.832	0.828	0.813	4881

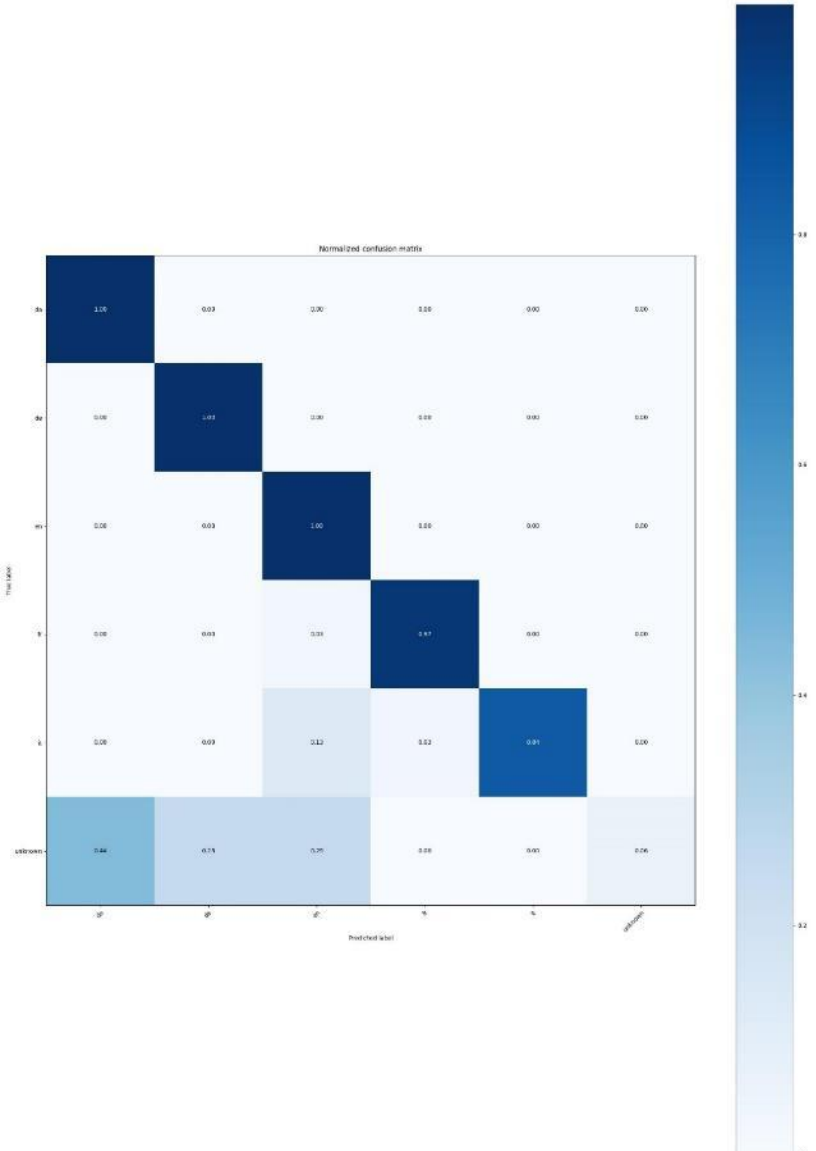
RF: authors confusion matrix



RF: language

```
#Info:
Classifier: RandomForestClassifier
Label: lang
Text label: text
Dense|LSA: False|False

#Counts:
Number of training data_records: 39077
Number of classified data_records: 4881
Number of unique classes in data_records: 6
Number of unique classes found: 6

#Performance:
Seconds used for training: 67
Seconds used for classification: 40

#Classification report:
              precision    recall  f1-score   support

          da      0.992     0.999     0.995       838
          de      0.997     0.999     0.998      1443
          en      0.996     1.000     0.998      2517
          fr      0.972     0.972     0.972        36
          it      0.963     0.839     0.897        31
     unknown      0.500     0.062     0.111        16

    accuracy                          0.995      4881
   macro avg      0.903     0.812     0.829      4881
weighted avg      0.994     0.995     0.994      4881
```

RF: language confusion matrix



RF: news

```
#Info:
Classifier: RandomForestClassifier
Label: category
Text label: headline,short_description
Dense|LSA: False|False

#Counts:
Number of training data_records: 51197
Number of classified data_records: 12824
Number of unique classes in data_records: 34
Number of unique classes found: 34

#Performance:
Seconds used for training: 263
Seconds used for classification: 31
```

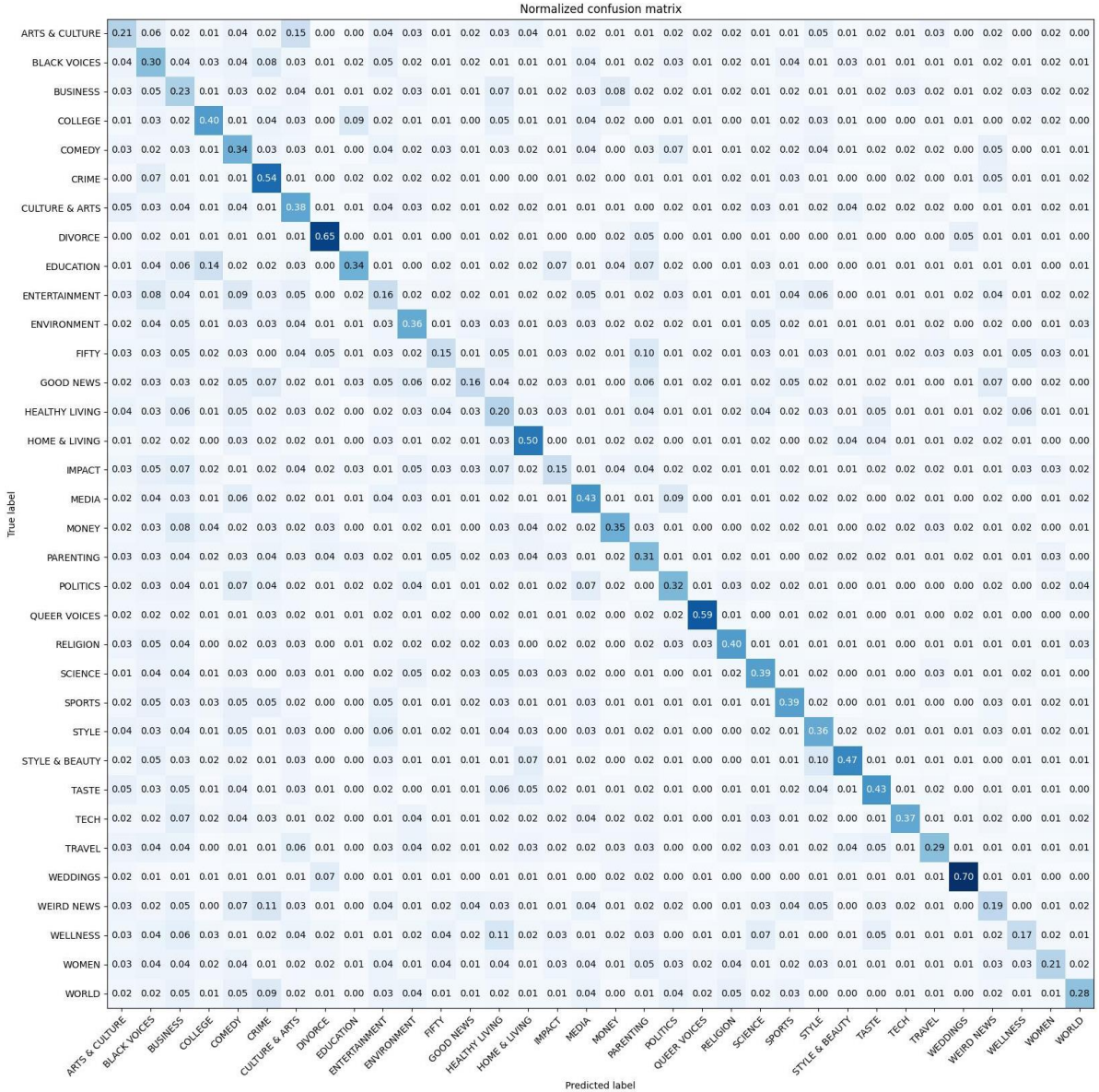
accuracy			**0.35**	**12824**
macro avg	0.35	0.34	0.34	12824
weighted avg	0.36	0.35	0.35	12824

RF: news details

#Classification report:

	precision	recall	f1-score	support
ARTS & CULTURE	0.16	0.21	0.18	282
BLACK VOICES	0.20	0.30	0.24	392
BUSINESS	0.15	0.23	0.18	380
COLLEGE	0.30	0.40	0.34	212
COMEDY	0.24	0.34	0.28	399
CRIME	0.38	0.54	0.44	409
CULTURE & ARTS	0.29	0.38	0.33	391
DIVORCE	0.61	0.65	0.63	419
EDUCATION	0.36	0.34	0.35	198
ENTERTAINMENT	0.15	0.16	0.15	387
ENVIRONMENT	0.31	0.36	0.33	355
FIFTY	0.15	0.15	0.15	273
GOOD NEWS	0.20	0.16	0.18	305
HEALTHY LIVING	0.17	0.20	0.18	386
HOME & LIVING	0.44	0.50	0.47	386
IMPACT	0.23	0.15	0.18	400
MEDIA	0.37	0.43	0.40	395
MONEY	0.37	0.35	0.36	324
PARENTING	0.28	0.31	0.29	391
POLITICS	0.36	0.32	0.34	420
QUEER VOICES	0.68	0.59	0.63	415
RELIGION	0.50	0.40	0.44	440
SCIENCE	0.41	0.39	0.40	414
SPORTS	0.42	0.39	0.41	410
STYLE	0.35	0.36	0.36	413
STYLE & BEAUTY	0.55	0.47	0.50	391
TASTE	0.46	0.43	0.44	397
TECH	0.53	0.37	0.43	416
TRAVEL	0.42	0.29	0.35	405
WEDDINGS	**0.70**	**0.70**	**0.70**	**400**
WEIRD NEWS	0.24	0.19	0.21	408
WELLNESS	0.29	0.17	0.21	407
WOMEN	0.33	0.21	0.26	400
WORLD	0.46	0.28	0.35	404

RF: news confusion matrix



RF: sentiment

```
#Info:
Classifier: RandomForestClassifier
Label: sentiment
Text label: text
Dense|LSA: False|False

#Counts:
Number of training data_records: 40000
Number of classified data_records: 10000
Number of unique classes in data_records: 2
Number of unique classes found: 2

#Performance:
Seconds used for training: 192
Seconds used for classification: 28

#Classification report:
              precision    recall  f1-score   support

    negative      0.725     0.829     0.774      4985
    positive      0.802     0.688     0.740      5015

    accuracy                          0.758     10000
   macro avg      0.764     0.759     0.757     10000
weighted avg      0.764     0.758     0.757     10000
```

RF: sentiment confusion matrix