BioMedInformatics

MDPI

Article

A Biomedical Case Study Showing That Tuning Random Forests Can Fundamentally Change the Interpretation of Supervised Data Structure Exploration Aimed at Knowledge Discovery

Jörn Lötsch [1,2,*] and Benjamin Mayer [1]

[1] Institute of Clinical Pharmacology, Goethe-University, Theodor-Stern-Kai 7, 60590 Frankfurt am Main, Germany
[2] Fraunhofer Institute for Translational Medicine and Pharmacology ITMP, Theodor-Stern-Kai 7, 60596 Frankfurt am Main, Germany
* Correspondence: j.loetsch@em.uni-frankfurt.de

Abstract: Knowledge discovery in biomedical data using supervised methods assumes that the data contain structure relevant to the class structure if a classifier can be trained to assign a case to the correct class better than by guessing. In this setting, acceptance or rejection of a scientific hypothesis may depend critically on the ability to classify cases better than randomly, without high classification performance being the primary goal. Random forests are often chosen for knowledge-discovery tasks because they are considered a powerful classifier that does not require sophisticated data transformation or hyperparameter tuning and can be regarded as a reference classifier for tabular numerical data. Here, we report a case where the failure of random forests using the default hyperparameter settings in the standard implementations of R and Python would have led to the rejection of the hypothesis that the data contained structure relevant to the class structure. After tuning the hyperparameters, classification performance increased from 56% to 65% balanced accuracy in R, and from 55% to 67% balanced accuracy in Python. More importantly, the 95% confidence intervals in the tuned versions were to the right of the value of 50% that characterizes guessing-level classification. Thus, tuning provided the desired evidence that the data structure supported the class structure of the data set. In this case, the tuning made more than a quantitative difference in the form of slightly better classification accuracy, but significantly changed the interpretation of the data set. This is especially true when classification performance is low and a small improvement increases the balanced accuracy to over 50% when guessing.

Keywords: data science; artificial intelligence; machine-learning; digital medicine

Citation: Lötsch, J.; Mayer, B. A Biomedical Case Study Showing That Tuning Random Forests Can Fundamentally Change the Interpretation of Supervised Data Structure Exploration Aimed at Knowledge Discovery. *Biomedinformatics* **2022**, *2*, 544–552. https://doi.org/10.3390/biomedinformatics2040034

Academic Editor: David Barlow

Received: 28 September 2022
Accepted: 14 October 2022
Published: 18 October 2022

1. Introduction

In biomedical research, knowledge is increasingly being discovered using machine-learning techniques. Unsupervised and supervised methods are used to analyze whether a data set contains a structure relevant to the research subject. Among the supervised methods, the classification algorithms are of particular interest and have been described in detail elsewhere for this purpose [1]. The approach assumes that if a classifier can be trained to assign a case to the correct class better than by guessing, then the data contain a structure that is relevant to the class structure and the variables the classifier needs to perform this task contain relevant information about the class structure being addressed. In this setting, high classification performance is not necessarily the primary goal, but rather the focus is on supporting the hypothesis that the variables "X" collected contain relevant information to support the given class structure "y". This hypothesis must be rejected if a reasonably chosen classifier, such as random forests for tabular numerical data, cannot assign cases to classes better than by guessing.

Random forests [2,3] is a tree-based bagging classification algorithm that creates a set of distinct, uncorrelated, and often very simple decision trees [3]. The splits of the features are random, and the classifier refers to the majority vote for class membership provided by hundreds of simple decision trees. It is considered a powerful classifier on tabular numeric data compared to deep learning neural networks [4,5] and outperforms logistic regression [6]. Increasing the number of trees in the forests does not tend toward overfitting [7]. Moreover, random forests can be used without prior complex parameter settings [8]; removing uninformative features has been advised to suffice the tuning [9]. These are key advantages to alternative classifiers, such as k-nearest neighbors [10] that requires a valid distance measure often difficult to define [11], or support vector machines [12] that have regularization as a critical hyperparameter [13], or deep learning layered artificial neural networks, that while considered universal classifiers [14], need the number of layers and the number of neurons in each layer to be set.

Its advantages make random forests a reference classifier for tabular numerical data. Initial investigation of whether the data contain a structure that reflects the class structure and can be used for classification is often performed with random forests. The present case study reports an occasional failure of random forests to classify a simple data set using the default hyperparameter settings of random forests implementations in the R [15] and Python [16] programming languages. The context of this observation was data exploration using unsupervised methods. As machine learning becomes more prevalent in biomedical research and "out-of-the-box" solutions become widely available, it is important to be aware of its pitfalls. The fact that an important decision about whether to accept or reject a research hypothesis may depend on the voting of random forests is the motivation for describing the following case.

2. Materials

2.1. Data Set

A biomedical data set was available from an assessment of pain sensitivity to various experimental noxious stimuli collected in a quantitative sensory testing study in n = 125 healthy Caucasian volunteers (69 men, 56 women, aged 18 to 46 years, mean 25 $\pm$ 4.4 years) [17]. From that study it was known that blunt pressure pain had a comparatively large effect size with respect to sex differences in pain perception (Figure 1) with an estimate of Cohen's d [18] = 0.83. Reassessing the sex difference verified statistical significance (t-test [19]: t = 4.7025, degree of freedom = 122.93, $p = 6.783 \times 10^{-6}$). In an actual reanalysis of this data set, a focus of assigning the subjects' sex from the acquired pain information was pursued, reversing the widely accepted fact that pain perception is gender specific [20].

2.2. Experimentation

2.2.1. Programming Environment

Programming was performed in the R language [15] using the R software package [21], version 4.1.2 for Linux, available free of charge from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/ (accessed on 9 October 2022), and in the Python language [16] using Python version 3.8.13 for Linux, available free of charge at https://www.python.org (accessed on 1 March 2022). The main R packages used for the experiments were R package "randomForest" (https://cran.r-project.org/package=randomForest (accessed on 9 October 2022) [23]), "caret" (https://cran.r-project.org/package=caret (accessed on 9 October 2022) [24]) and "pROC" (https://cran.r-project.org/package=pROC (accessed on 9 October 2022) [25]). The main packages used for the Python-based data analysis were the numerical Python package "numpy" (https://numpy.org (accessed on 9 October 2022) [26]), "pandas" (https://pandas.pydata.org (accessed on 9 October 2022) [27,28]), fundamental algorithms for scientific computing in Python "SciPy" (https://scipy.org (accessed on 9 October 2022) [29]) and "scikit-learn" (https://scikit-learn.org/stable/ (accessed on 9 October 2022) [30]). Experiments were performed on 1–64 cores/threads on an

AMD Ryzen Threadripper 3970X (Advanced Micro Devices, Inc., Santa Clara, CA, USA) computer with 256 GB random access memory (RAM) running Ubuntu Linux 22.04.1 LTS (Canonical, London, UK)).

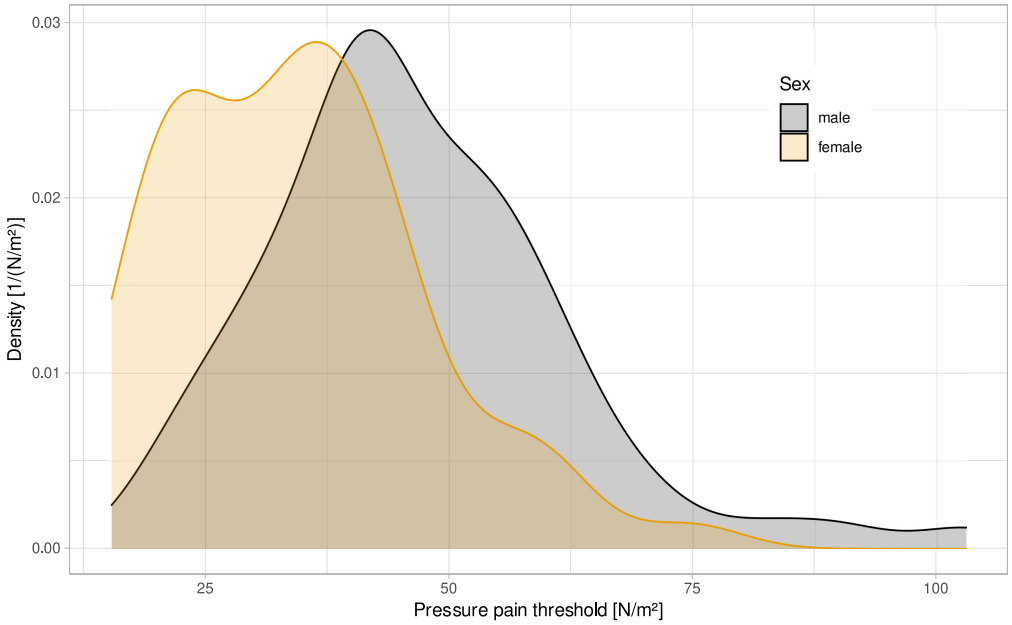

Figure 1. Pressure pain data set: class-wise distributions. Density plot showing the distribution of the raw data of pain thresholds to blunt pressure stimuli, color coded for the two sexes. The figure has been created using the R software package (version 4.2.1 for Linux; https://CRAN.R-project.org/ (accessed on 9 October 2022) [21]) and the library "ggplot2" (https://cran.r-project.org/package=ggplot2 (accessed on 9 October 2022) [22]).

2.2.2. Design of the Experiments

In a 1000-fold cross-validation scenario on training data subsets comprising 2/3 of the original data drawn by random Monte-Carlo re-sampling [31] from the original data. The trained classifiers were then applied to the remaining 1/3 of the cases that have not been used for training. The success of classifier training was quantified by calculating the balanced accuracy [32], which corresponds to the area under the receiver operating characteristic curve (roc-auc) [33]. These calculations were done using the R libraries "caret" and "pROC".

3. Results

3.1. Random Forests Classification with Default Hyperparameters

An initial attempt to assign subjects to the correct sex based on their pain threshold to blunt pressure stimuli using a random forests classifier was performed using the default parameters of the R library "randomForest", i.e., $n = 500$ decision trees with sqrt(d) features = 1 and no restriction on the number of nodes or the depth of the trees (Listing 1).

Listing 1: R code for classification with random forests using the default settings of the library "randomForest" (https://cran.r-project.org/package=randomForest (accessed on 9 October 2022) [23]).

```
# without tuning
  rf <-
    randomForest::randomForest(as.factor(sex) ~ .,
      data = training
    )
```

This led to a 95 % confidence interval of both, balanced accuracy and roc-auc, that included the value of 0.5, i.e., the classification could not be considered as better than mere guessing (Table 1). This was unexpected given the significant difference determined by a *t*-test, the clear separation of the distributions of pain thresholds (Figure 1) between the two sexes, and therefore prompted further investigation.

Table 1. Performance of non-tuned and tuned random forests and of two ad hoc classifiers consisting of a simple brute force split rule or a Bayesian decision rule. Performance was evaluated as balanced accuracy (BA) and, for random forests, also as area under the receiver operator characteristic (auc-roc). The latter was not calculated for the ad hoc classifiers because the assignment probability was either not implemented (split) or not queried (Bayes). Implementations of random forests in the R ("randomForest") and Python ("RandomForestClassifier") programming languages were both used.

Classifier	**Tuning**	**Performance Measure**		
		Package	**RandomForest (R)**	**RandomForest Classifier (Python)**
RF	non-tuned	BA	0.56 (0.42–0.68)	0.55 (0.43–0.67)
		auc-roc	0.59 (0.45–0.72)	0.58 (0.45–0.7)
	tuned	BA	0.65 (0.53–0.77)	0.67 (0.55–0.78)
		auc-roc	0.73 (0.59–0.85)	0.72 (0.58–0.85)
Split	ad hoc	BA	0.65 (0.54–0.76)	-
Bayes decision	ad hoc	BA	0.67 (0.55–0.79)	-

3.2. Investigation of the Classification Failure

First, it was checked whether the R library "randomForest" accidentally failed on this data set. Therefore, the classification attempt was repeated using the Python implementation "RandomForestClassifier" from the "sklearn.ensemble" package (Textbox 2). Again, the balanced accuracy and the roc-auc were close to the value of 0.5 (50%), and their confidence interval spanned the value 0.5, indicating a failed classification (Table 1)).

Second, a simple splitting rule was built by brute force by testing each possible splitting of the pressure pain threshold of the training data set for the best accuracy for discriminating the sexes. The classification performance of these rules was evaluated on the test data subset. In addition, a Bayesian boundary [34] in the pain threshold between sexes was computed using our R library "AdaptGauss" (https://cran.r-project.org/package=AdaptGauss (accessed on 9 October 2022) [35]), and used as another simple classifier. These two ad hoc classifiers had no problem assigning sexes based on pressure pain threshold with balanced accuracy and a roc-auc better than guessing, that is, with a lower limit of the 95% confidence interval 0.5 (Table 1).

3.3. Random Forests Classification with Tuned Hyperparameters

Tuning the hyperparameters of the implementation of the package "randomForest" for the number of trees (100 to 1500 in increments of 100) and the number of nodes in each tree (1–8) (Figure 2) indicated that the classifier should be run with n = 1300 trees and a maximum of n = 3 nodes.

A rerun of the classifier training and testing with the tuned hyperparameters (code not shown) provided the desired result of balanced accuracy and roc-auc for sex assignment from the pain threshold variable better than guessing, i.e., with confidence intervals to the right of the 0.5 value (Table 1). The Python implementation (Listing 2) resulted in n = 700 trees with a maximum depth of only one split. Again, the desired classification better than chance was achieved, almost exactly as with the R implementation "randomForest" (Table 1).

Listing 2: Python code for hyperparameter tuning and classification with random forests using the "RandomForestClassifier" method imported from the "scikit-learn"package (https://scikit-learn.org/stable/ (accessed on 9 October 2022) [30]).

```
import pandas as pd
import numpy as np
from sklearn.metrics import balanced_accuracy_score, roc_auc_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# %% Classifier tuning
forest = RandomForestClassifier(random_state=0)
param_grid = {'bootstrap': [True, False],
              'max_depth': np.arange(1,11).tolist(),
              'max_features': [1],
              'min_samples_leaf': np.arange(1,21).tolist(),
              'min_samples_split': np.arange(2,21).tolist(),
              'n_estimators': np.arange(100, 1500, 100).tolist()}
grid_search = GridSearchCV(forest, param_grid=param_grid,
                           scoring="balanced_accuracy", verbose=0, n_jobs=-1)
grid_search.fit(df1, y)
bootstrap_rf,  max_depth_rf, max_features_rf, min_samples_leaf_rf,
        min_samples_split_rf, n_estimators_rf = grid_search.best_params_.values()

# %% Classification CV
nIter = 1000
BA_RF_default, ROC_RF_default,BA_RF_tuned, ROC_RF_tuned = [], [], [], []
for i in range(nIter):
    # %split data into train and test
    X_train, X_test, y_train, y_test = train_test_split(
        df1, y, test_size=0.33, random_state=i)
    pd.DataFrame(y_test).value_counts()

    forest = RandomForestClassifier(random_state=0)
    forest.fit(X_train, y_train)
    y_pred = forest.predict(X_test)
    y_pred_proba = forest.predict_proba(X_test)
    BA_RF_default.append(
        balanced_accuracy_score(y_test, y_pred))
    ROC_RF_default.append(roc_auc_score(
        y_test, y_pred_proba[:, 1]))

    forest = RandomForestClassifier(random_state=0, bootstrap=bootstrap_rf,
        max_depth=max_depth_rf, max_features=max_features_rf,
        min_samples_leaf=min_samples_leaf_rf,
        min_samples_split=min_samples_split_rf,
        n_estimators=n_estimators_rf)
    forest.fit(X_train, y_train)
    y_pred = forest.predict(X_test)
    y_pred_proba = forest.predict_proba(X_test)
    BA_RF_tuned.append(
        balanced_accuracy_score(y_test, y_pred))
    ROC_RF_tuned.append(roc_auc_score(
        y_test, y_pred_proba[:, 1]))
```

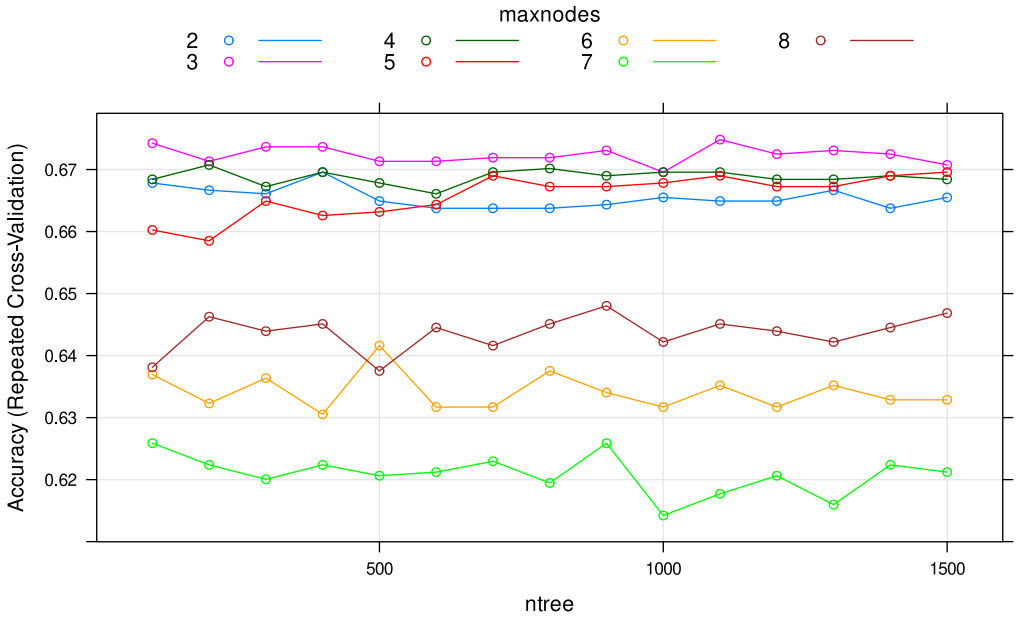

Figure 2. Results of tuning the number of trees (ntree) and the maximum number of nodes (maxnodes) of the random forests R implementation. The figure has been created using the R software package (version 4.2.1 for Linux; https://CRAN.R-project.org/ (accessed on 9 October 2022) [21]); it is the printout of the grid tuning using the "caret" library (https://cran.r-project.org/package=caret (accessed on 9 October 2022) [24]), which calls for plotting the "lattice" package (https://cran.r-project.org/package=lattice (accessed on 9 October 2022) [36]).

4. Discussion

In this example case, random forests unexpectedly but consistently failed across programming environments on a seemingly simple classification problem when the default parameters of the software implementations were used. After tuning the hyperparameters, the expected classification success was achieved, i.e., classification performance exceeded guessing. Failure with the default settings can become particularly relevant when supervised learning is used to recognize data structures rather than to create a powerful classifier. A typical problem in biomedical data analysis is met in a set of individuals with a particular class label, e.g., a diagnosis, and some measurements for each case. The first question is whether these measurements have a structure that is relevant to the class structure and can be used to classify the subjects. To answer this question, a powerful machine-learned classifier can be trained on a subset of the data. If this classifier is able to classify the cases not used in learning such that this classification is better than guessing, this indicates that there is structure in the data that supports the class structure. Classification accuracy that exceeds guessing is sufficient for this structural assessment. This was thwarted in the present example case, where failure to classify would have led to false rejection of the hypothesis that the data contained a structure relevant to the subjects' sex. Only after tuning, the hypothesis could be accepted.

Thus, when random forests is used for supervised structure discovery in data sets and for feature selection [37], classification performance better than chance is the first requirement. Knowledge discovery by applying feature selection techniques assumes that if a classifier can be trained to assign a case to the correct class better than by guessing, the variables needed by the classifier to accomplish this task contain relevant information about the class structure being addressed. However, this depends on the success of the classification. It is not of interest in this context with which feature an unsuccessful classifier attempted its classification task. In this case study, tuning provided exactly the required classification success of the random forests classifier.

The most important parameter for tuning the random forests seemed to be the tree depth in the present data, which had to be limited to a few decisions of maxnodes = 3 and max_depth = 1 for the R and Python implementations, respectively. Both refer to a limit on tree growth. In contrast, the number of trees seemed less important (Figure 2). Changing the number of trees between 100 and 1500 had indeed little effect in additional cross-validated tuning attempts (not shown). This is consistent with previous results of assessing the impact of the number of trees in the forest [7]. However, with only one variable, this observation cannot be generalized from present experiments. Using random forests for only one variable seems unusual, but it is not discouraged, and it may well be that a single variable remains in a data set after feature selection.

In the present analyses, standard implementations of random forests in the R and Python programming languages were used with consistent results. This is the common data science environment. However, less frequently used modifications of random forests might be less affected, but were not tested here. An example is the combined use of the Kolmogorov–Gabor polynomial [38] and the random forests algorithm to increase classification accuracy [39]. There, each variable vector is represented as polynomial members and random forests are used to find the coefficients. However, the Python code mentioned by the authors is not freely available, and reimplementing the method would far exceed the purpose of this case report. Further modifications of random forests can also better avoid the present case of initial classification failure, such as several proposals summarized in [40]. However, the present work has highlighted a pitfall in standard data analysis when default parameters of common implementations of random forests are used for data exploration. The use of sophisticated modifications likely implies that parameters are carefully chosen rather than left at default values, so the problem presented here may not arise in these particular applications of random forests. However, for default implementations, the problem seems to occur with other implementations as well. Using the R package "ranger" (https://CRAN.R-project.org/package=ranger (accessed on 9 October 2022) [41]) instead of the package "randomForest", the default parameters provided a balanced accuracy of only 0.56 (95% confidence interval: 0.43–0.69). This underlines the relevance of the present observation performed with this case.

5. Concluding Remarks

The present case study showed that random forests occasionally need hyperparameter tuning. While tuning is implemented in data science workflows, encouraged by the reputation of random forests for being a robust classifier that does not require complicated parameter settings, tuning may be omitted to save time or when in a supervised approach to data structure exploration high classification performance is not the main goal while it is sufficient to prove that the data contain information relevant to the class structure if the classification is better than guessing. In this case study, tuning made more than a quantitative difference in terms of slightly better classification accuracy. In contrast, it made a qualitative difference in the interpretation of the data set, namely the difference between judging the input data structure as supporting or not supporting the class structure of the data set. This is especially relevant for data sets where classification performance is low and where a small improvement will increase balanced accuracy to over the 50% guessing limit. In the present case, tuning the R implementation of random forests increased the balanced accuracy of assigning subjects to their correct sex from 56% to 65%, and in the Python implementation the change was from 55% to 67%; however, the 95% confidence intervals in the tuned versions were to the right of the value of 50% that characterizes guessing-level classification. The tuning effect goes beyond quantitative improvement in classification performance but affects the interpretation of a data set based on supervised learning fundamentally. Therefore, the present report addresses the use of supervised data analysis for knowledge discovery, highlighting that tuning of random forests when used for supervised knowledge discovery in biomedical data can decide about the acceptance or the rejection of a research hypothesis.

Author Contributions: Conceptualization, J.L.; Data curation, J.L.; Formal analysis, J.L.; Funding acquisition, J.L.; Investigation, J.L.; Validation, B.M.; Visualization, J.L.; Writing—original draft, J.L. and B.M. All authors have read and agreed to the published version of the manuscript.

Funding: J.L. was supported by the Deutsche Forschungsgemeinschaft (DFG LO 612/16-1).

Institutional Review Board Statement: The study from which the data set originates followed the Declaration of Helsinki and was approved by the Ethics Committee of Medical Faculty of the Goethe—University, Frankfurt am Main, Germany (approval number 150/11). Permission for anonymized report of data analysis results obtained from the acquired information was included in the informed written consent.

Informed Consent Statement: Permission for anonymized report of data analysis results obtained from the acquired information was included in the informed written consent.

Data Availability Statement: The data set is available as "PressureSex.csv" on https://github.com/JornLotsch/RFtuning (accessed on 9 October 2022).

Conflicts of Interest: The authors have declared that no competing interest exist.

References

1. Spiliopoulou, M.; Schmidt-Thieme, L.; Janning, R. *Data Analysis, Machine Learning and Knowledge Discovery*; Studies in Classification, Data Analysis, and Knowledge Organization, Springer International Publishing: Berlin/Heidelberg, Germany, 2013.
2. Ho, T.K. Random decision forests. In Proceedings of the Third International Conference on Document Analysis and Recognition, ICDAR '95, Montreal, QC, Canada, 14–16 August 1995; Volume 1, p. 278.
3. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
4. Svetnik, V.; Wang, T.; Tong, C.; Liaw, A.; Sheridan, R.P.; Song, Q. Boosting: An Ensemble Learning Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Model.* **2005**, *45*, 786–799. [CrossRef]
5. Xu, H.; Kinfu, K.A.; LeVine, W.; Panda, S.; Dey, J.; Ainsworth, M.; Peng, Y.C.; Kusmanov, M.; Engert, F.; White, C.M.; et al. When are Deep Networks really better than Decision Forests at small sample sizes, and how? *arXiv* **2021**, arXiv:2108.13637.
6. Couronné, R.; Probst, P.; Boulesteix, A.L. Random forest versus logistic regression: A large-scale benchmark experiment. *BMC Bioinform.* **2018**, *19*, 270. [CrossRef] [PubMed]
7. Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J.C.; Sheridan, R.P.; Feuston, B.P. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1947–1958. [CrossRef]
8. Huang, B.F.; Boutros, P.C. The parameter sensitivity of random forests. *BMC Bioinform.* **2016**, *17*, 331. [CrossRef] [PubMed]
9. Kuhn, M.; Johnson, K. *Feature Engineering and Selection: A Practical Approach for Predictive Models*; CRC Press: Boca Raton, FL, USA, 2019.
10. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.* **1967**, *13*, 21–27. [CrossRef]
11. Bryant, V. *Metric Spaces: Iteration and Application*; Cambridge University Press: Cambridge, UK, 1985.
12. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
13. Bennett, K.P.; Campbell, C. Support vector machines: Hype or hallelujah? *ACM SIGKDD Explor. Newsl.* **2000**, *2*, 1–13. [CrossRef]
14. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
15. Ihaka, R.; Gentleman, R. R: A Language for Data Analysis and Graphics. *J. Comput. Graph. Stat.* **1996**, *5*, 299–314. [CrossRef]
16. Van Rossum, G.; Drake, F.L., Jr. *Python Tutorial*; Centrum voor Wiskunde en Informatica: Amsterdam, The Netherlands, 1995; Volume 620.
17. Doehring, A.; Küsener, N.; Flühr, K.; Neddermeyer, T.J.; Schneider, G.; Lötsch, J. Effect sizes in experimental pain produced by gender, genetic variants and sensitization procedures. *PLoS ONE* **2011**, *6*, e17724. [CrossRef]
18. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*; Routledge: New York, NY, USA, 1988. 9780203771587. [CrossRef]
19. Student. The Probable Error of a Mean. *Biometrika* **1908**, *6*, 1–25. [CrossRef]
20. Mogil, J.S. Sex differences in pain and pain inhibition: Multiple explanations of a controversial phenomenon. *Nat. Rev. Neurosci.* **2012**, *13*, 859–866. [CrossRef] [PubMed]
21. Team, R.D.C. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2008.
22. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis*; Springer: New York, NY, USA, 2009.
23. Liaw, A.; Wiener, M. Classification and Regression by randomForest. *R News* **2002**, *2*, 18–22.
24. Kuhn, M. *Caret: Classification and Regression Training*; Astrophysics Source Code Library: Houghton, HI, USA, 2018.
25. Robin, X.; Turck, N.; Hainard, A.; Tiberti, N.; Lisacek, F.; Sanchez, J.C.; Müller, M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinform.* **2011**, *12*, 77. [CrossRef]
26. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]

27. McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference (SciPy 2010), Austin, TX, USA, 28 June–3 July 2010; pp. 56–61.
28. The Pandas Development Team Pandas-dev/pandas: Pandas. *Zenodo* 2020. [CrossRef]
29. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
30. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
31. Good, P.I. *Resampling Methods: A Practical Guide to Data Analysis*; Birkhäuser: Boston, MA, USA, 2006.
32. Brodersen, K.H.; Ong, C.S.; Stephan, K.E.; Buhmann, J.M. The Balanced Accuracy and Its Posterior Distribution. In Proceedings of the Pattern Recognition (ICPR), 2010 20th International Conference on, Istanbul, Turkey, 23–26 August 2010; pp. 3121–3124. [CrossRef]
33. Sokolova, M.; Japkowicz, N.; Szpakowicz, S. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. In *AI 2006: Advances in Artificial Intelligence, Lecture Notes in Computer Science*; Sattar, A., Kang, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4304, pp. 1015–1021. [CrossRef]
34. Bayes, M.; Price, M. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philos. Trans.* **1763**, *53*, 370–418. [CrossRef]
35. Ultsch, A.; Thrun, M.C.; Hansen-Goos, O.; Lötsch, J. Identification of Molecular Fingerprints in Human Heat Pain Thresholds by Use of an Interactive Mixture Model R Toolbox (AdaptGauss). *Int. J. Mol. Sci.* **2015**, *16*, 25897–25911. [CrossRef] [PubMed]
36. Sarkar, D. *Lattice: Multivariate Data Visualization with R*; Springer: New York, NY, USA, 2008.
37. Lotsch, J.; Ultsch, A. Random Forests Followed by Computed ABC Analysis as a Feature Selection Method for Machine Learning in Biomedical Data. In *Advanced Studies in Classification and Data Science*; Imaizumi, T., Okada, A., Miyamoto, S., Sakaori, F., Yoshiro, Y., Vichi, M., Eds.; Springer: Singapore, 2020; pp. 57–69.
38. Ivakhnenko, A.G. Polynomial Theory of Complex Systems. *IEEE Trans. Syst. Man Cybern.* **1971**, *4*, 364–378. [CrossRef]
39. Tkachenko, R.; Duriagina, Z.; Lemishka, I.; Izonin, I.; Trostianchyn, A. Development of machine learning method of titanium alloy properties identification in additive technologies. *East.-Eur. J. Enterp. Technol.* **2018**, *3*, 23–31. [CrossRef]
40. Tripoliti, E.E.; Fotiadis, D.I.; Manis, G. Modifications of the construction and voting mechanisms of the Random Forests Algorithm. *Data Knowl. Eng.* **2013**, *87*, 41–65. [CrossRef]
41. Wright, M.N.; Ziegler, A. Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *J. Stat. Softw.* **2017**, *77*, 1–17. [CrossRef]