

Supplement 1. Summary of the material from the study whose database were used (Dobruć-Sobczak *et al.*⁽¹⁾)

Materials and methods

Patients

This retrospective study was approved by the institutional bioethical review board of each participating institution: Maria Skłodowska-Curie Institute of Oncology in Warsaw (MSCI), Medical University of Lodz (MUL), Poznan University of Medical Sciences (PUMS) in Poland, and the Department of Imaging Diagnostics, Medical University of Warsaw (MUW)

The authors reviewed the database from January 2009 to July 2018, obtaining data on 428 patients admitted to tertiary referral centers for thyroidectomy due to the following Bethesda System for Reporting Thyroid Cytopathology (BSRTC) results: (a) suspicion of malignancy or neoplasm (BSRTC category IV–VI) or (b) nodular goiter with clinical symptoms (BSRTC category II). Patients with symptomatic purely cystic lesions were excluded. Thyroid ultrasound examination was performed again on admission to the Department of Surgery just before thyroidectomy. A total of 842 thyroid nodules were included and classified as benign or malignant on the basis of the final histopathological examination of the resected specimens.

Ultrasound examination

Ultrasound examinations were conducted at the Department of Oncological Endocrinology and Nuclear Medicine of MSCI in Warsaw, the Department of Endocrinology and Metabolic Diseases of the Medical University of Lodz, the Department of Endocrinology, Metabolism, and Internal Medicine of PUMS, and the Department of Imaging Diagnostics at the Medical University of Warsaw. Neck ultrasound examinations were performed with the use of linear transducers (7–18 MHz by Aplio XG, Toshiba Medical Systems, Japan; 5–12-MHz by iU22, Philips Medical Systems, Bothell, Wash), and 5–15 MHz with the AIXPLOERER system (Supersonic Imagine, Aix-en-Provence, France).

Ultrasound (US) examinations were performed by one of five sonographers (two radiologists, three endocrinologists) with 9 to 22 years of experience in thyroid imaging. Physicians were blinded to the FNAB results and the final postsurgical verification when

reassessing the US examinations and assigning the EU-TIRADS score. During US examination, transverse and longitudinal planes of both the gland and nodules were obtained with the patient in the supine position. The anteroposterior, transverse, and longitudinal diameters of the gland and nodules were measured on frozen images during examination and archived. US features of thyroid nodules were also prospectively recorded.

EU-TIRADS score

Retrospectively, all 842 nodules were classified according to the European Thyroid Association Guidelines for Ultrasound Malignancy Risk Stratification of Thyroid Nodules in Adults (EU-TIRADS). The following are the characteristics of the different levels of classification according to this system:

- EU-TIRADS 2 (benign category): purely cystic, entirely spongiform; risk of malignancy (RM): close to 0%, no biopsy recommended.
- EU-TIRADS 3 (low-risk category): ovoid, smooth isoechoic/hyperechoic, no highly suspicious characteristics; RM: 2%–4%, biopsy recommended only for nodules >20 mm.
- EU-TIRADS 4 (intermediate-risk category): ovoid, smooth, mildly hypoechoic, no highly suspicious characteristics; RM: 6%–17%, biopsy usually recommended for nodules >15 mm.
- EU-TIRADS 5 (high-risk category): at least one of the following highly suspicious characteristics: irregular shape (taller-than-wide), irregular margins, microcalcifications, marked hypoechoogenicity (and solid); RM: 26%–87%, biopsy recommended for nodules >10 mm.

Histopathological examination

Final histopathological diagnoses were obtained after thyroidectomy for all 842 analyzed nodules. Of the 229 malignant neoplasms, papillary thyroid carcinoma (PTC) was the most common (184), while hyperplastic lesions predominated among the benign lesions. Pathologists were blinded to the results of the ultrasound examination. Surgical specimens were immediately fixed in 10% buffered formalin. Representative sections were processed and routinely stained with H&E (hematoxylin and eosin) for microscopic examination.

1. Dobruć-Sobczak K, Adamczewski Z, Szczepanek-Parulska E, Migda B, Woliński K, Krauze A *et al.*: Histopathological verification of the diagnostic performance of the EU-TIRADS classification of thyroid nodules—results of a multicenter study performed in a previously iodine-deficient region. *J Clin Med* 2019; 8: 1781. doi: 10.3390/jcm8111781.

Supplement 2: Technical description

Introduction

The purpose of this document is to present detailed characteristics of selected Python libraries and programming tools used in data analysis and machine learning. The document supplements the main article by providing additional technical information that helps clarify the implementation and operation of classifiers and validation methods.

The documentation focuses on the practical aspects of using Python functions and classes, describing technical parameters, default values, and operating principles of individual methods. It is intended for readers with a basic knowledge of Python who are new to machine learning, as well as for those who seek detailed information necessary for the correct implementation of classification models in the Python environment.

General libraries

Pandas

A library for manipulating data in Python, especially tabular and spreadsheet data. Key objects:

- DataFrame – a two-dimensional table with indexes and named columns
- Series – a one-dimensional data vector with labels

Application: loading data (e.g., CSV, Excel files), cleaning, transforming, filtering, and merging datasets. In this study, the library was used to load data and adapt it for use in classifiers.

Seaborn

A visualization library built on matplotlib, focusing on statistical data analysis. Key features:

- `sns.heatmap()` – visualization of correlation matrices
- `sns.pairplot()` – pairs of scatter plots and histograms for multiple variables
- `sns.boxplot()` – analysis of distributions and outliers

In this study, the `heatmap()` function was used to determine correlations between individual features.

Tools and auxiliary functions

Scikit-learn

A popular library for machine learning in Python, implementing methods for classification, regression, clustering, dimensionality reduction, feature selection, and cross-validation.

The following methods were used:

- StratifiedKFold – a method used for cross-validation, which

divides the dataset into K equal parts and then iteratively uses different parts to test and train the model. In the case of StratifiedKFold, divisions are made in such a way that the proportions of classes in each fold are similar to those in the entire dataset.

- RFECV – a feature selection method that works based on recursive feature elimination (RFE) combined with cross-validation. In each iteration, the model is trained on a subset of features, and the least significant feature (based on model weight or significance measures) is removed. The process is repeated until a minimum number of features remain, while cross-validation is used in each iteration to evaluate the performance of a given subset. It is used to reduce dimensionality, improve model interpretability, and decrease the risk of overfitting. The method implemented in this library takes the following parameters:
 - estimator – base model used to evaluate feature importance and for classification in the elimination process. RFECV uses this model to calculate which features are important. In this study, SVC with a linear kernel type was selected.
 - step – specifies how many features are removed per iteration. In the present implementation, this parameter has a value of 1.
 - cv – specifies the cross-validation method used to evaluate the quality of the model for a given set of features. In the present implementation, this parameter has a value of 5.
 - scoring – defines the quality metric by which RFECV evaluates the performance of the model. In the case studied, the most important aspect was to avoid false negatives, so recall was selected as the quality metric.
- GridSearchCV – used for automatic model hyperparameter tuning. GridSearchCV checks all combinations of specific parameters (from the so-called parameter grid) and evaluates the model in each configuration using cross-validation. The result is the best combination of parameters that yields the highest model quality. The method is used to optimize model parameters, e.g., classifiers, returning a model with the best hyperparameter configuration. In this study, the method was used to select hyperparameters for logistic regression and SVC and Random Forest classifiers. The implementation of this function requires the following parameters to be specified:
 - estimator – the model for which optimal hyperparameters are sought. In this study, these were logistic regression, SVC, and Random Forest.
 - param_grid – the list of parameters checked was different for each classifier. For logistic regression, it was the selection of the C parameter from the values 1.1, 1.25, 1.5. For SVC, it was the selection of the C value from the values 1 and 10, and the kernel type, 'linear' and 'rbf'. For Random Forest, it was the selection of the number of trees (`n_estimators`) from the values 200 or 500; the `max_features` parameter from auto, sqrt, and log2; `max_depth` from the values 4, 5, 6, 7, 8; and criterion from gini or entropy.
 - scoring – a model quality assessment function used to select the best parameters. Recall was selected for all classifiers.
 - cv – the number of folds in cross-validation; in all implementations, it was 10.
 - verbose – the level of detail in messages displayed during GridSearchCV operation. In all implementations, the value

of this parameter was 2, providing detailed information about each combination of parameters and progress.

- `cross_val_score` – a function used to quickly assess model quality using cross-validation. It automatically divides the data into a specified number of folds, trains the model on a portion of the data, and tests it on the remainder. It returns a list of results for each fold, allowing estimation of the average effectiveness of the model and its variance. In this study, the described method was used for each classifier. The values of the implemented parameters are listed below. A description of their significance for each of the classifiers can be found later in this document.
 - Logistic Regression
 - default parameters
 - Decision Tree Classifier
 - criterion: entropy
 - Random Forest Classifier
 - criterion: gini
 - max_depth: 8
 - max_features: sqrt
 - n_estimators: 200
 - K-Nearest Neighbors Classifier
 - n_neighbors: 2
 - SVC
 - kernel: linear
 - C: 1

Models and classifiers

All classifiers described below were implemented using methods available in the Python scikit-learn library.

Support Vector Classifier (SVC)

Support Vector Machine (SVM) creates a decision boundary in feature space that maximizes the margin separating samples from different classes. The key element is the selection of so-called support vectors, which are samples closest to the boundary. Thanks to kernel functions, the model can perform non-linear classification, e.g., using an RBF kernel that maps data into a higher-dimensional space.

Technical parameters:

- kernel – type of kernel function
- C – regularization parameter
- gamma – influence of individual samples

Technical parameters selected in the course of the study:

- kernel: linear
- C: 1
- gamma: in the case of the linear kernel type, this parameter is ignored

Logistic Regression

Logistic regression estimates the probability of class membership using a sigmoid function. The decision boundary is linear, and the output values are interpreted as probabilities. In practice, the model finds a linear combination of features that best separates the classes.

Technical parameters:

- solver – optimization algorithm
- penalty – type of regularization
- C – inverse of regularization strength. Technical parameters selected in the study:
 - solver: lbfgs
 - penalty: l2
 - C: 1.0

Random Forest classifier

Random Forest is an ensemble algorithm that builds multiple decision trees on random subsamples of data and features. The classification result is obtained by majority vote.

Thanks to its randomness, the model reduces the risk of overfitting and is resistant to noise in the data.

Technical parameters:

- criterion – function for evaluating the quality of splits
- n_estimators – number of trees
- max_depth – maximum tree depth
- max_features – number of features used in node division.

Technical parameters selected in the study:

- criterion: gini
- max_depth: 8
- max_features: sqrt
- n_estimators: 200

Decision Tree classifier

A decision tree divides data into smaller subsets based on conditional questions about feature values. The splitting criterion (e.g., entropy, Gini) determines which features best separate the classes. The process continues until the maximum depth is reached or stopping criteria are met.

Technical parameters:

- criterion – function for evaluating the quality of the split
- max_depth – maximum depth of the tree
- min_samples_split – minimum number of samples required for splitting. Technical parameters selected in the study:
 - criterion: entropy
 - max_depth: None
 - min_samples_split: 2

K-Nearest Neighbors Classifier

The KNN algorithm classifies a sample based on the votes of its closest neighbors in the feature space. The number of neighbors, k , determines how the decisions are averaged. For $k=1$, classification consists of assigning the same class as the closest sample.

Distances between points can be measured using various metrics (e.g., Euclidean, Manhattan).

Technical parameters:

- `n_neighbors` – number of neighbors
- `metric` – type of metric
- `weights` – method of weighting votes

Technical parameters selected in the course of the study:

- `n_neighbors`: 4
- `metric`: Minkowski, equivalent to Euclidean metric
- `weights`: uniform, meaning that each neighbor contributes equally to the vote

Sources

General libraries

Pandas: <https://pandas.pydata.org/docs/> Seaborn: <https://seaborn.pydata.org/>

Matplotlib (pyplot): https://matplotlib.org/stable/api/pyplot_summary.html Scikit-learn (sklearn): <https://scikit-learn.org/stable/documentation.html>

Tools and support features

StratifiedKFold:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

RFECV (Recursive Feature Elimination with Cross-Validation):

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html GridSearchCV:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

`cross_val_score`:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

Models and classifiers

LogisticRegression:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

SVC (Support Vector Classifier):

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html> RandomForestClassifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

DecisionTreeClassifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html> KNeighborsClassifier:

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>