mathematics

MDPI

Article

Decision-Making on the Diagnosis of Oncological Diseases Using Cost-Sensitive SVM Classifiers Based on Datasets with a Variety of Features of Different Natures

Liliya A. Demidova

Institute of Information Technologies, Federal State Budget Educational Institution of Higher Education, MIREA—Russian Technological University, 78, Vernadsky Avenue, 119454 Moscow, Russia; liliya.demidova@rambler.ru

Abstract: This paper discusses the problem of detecting cancer using such biomarkers as blood protein markers. The purpose of this research is to propose an approach for making decisions in the diagnosis of cancer through the creation of cost-sensitive SVM classifiers on the basis of datasets with a variety of features of different nature. Such datasets may include compositions of known features corresponding to blood protein markers and new features constructed using methods for calculating entropy and fractal dimensions, as well as using the UMAP algorithm. Based on these datasets, multiclass SVM classifiers were developed. They use cost-sensitive learning principles to overcome the class imbalance problem, which is typical for medical datasets. When implementing the UMAP algorithm, various variants of the loss function were considered. This was performed in order to select those that provide the formation of such new features that ultimately allow us to develop the best cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. The experimental results proved the possibility of applying the UMAP algorithm, approximate entropy and, in addition, Higuchi and Katz fractal dimensions to construct new features using blood protein markers. It turned out that when working with the UMAP algorithm, the most promising is the application of a loss function on the basis of fuzzy cross-entropy, and the least promising is the application of a loss function on the basis of intuitionistic fuzzy cross-entropy. Augmentation of the original dataset with either features on the basis of the UMAP algorithm, features on the basis of the UMAP algorithm and approximate entropy, or features on the basis of approximate entropy provided the creation of the three best cost-sensitive SVM classifiers with mean values of the metric $MacroF_1 - score$ increased by 5.359%, 5.245% and 4.675%, respectively, compared to the mean values of this metric in the case when only the original dataset was utilized for creating the base SVM classifier (without performing any manipulations to overcome the class imbalance problem, and also without introducing new features).

Keywords: oncological disease; cost-sensitive SVM classifier; features; UMAP algorithm; loss function; entropy; fractal dimension

MSC: 68Q32; 68T05

check for updates

Citation: Demidova, L.A. Decision-Making on the Diagnosis of Oncological Diseases Using Cost-Sensitive SVM Classifiers Based on Datasets with a Variety of Features of Different Natures. *Mathematics* **2024**, *12*, 538. https://doi.org/10.3390/math12040538

Academic Editor: Liangxiao Jiang

Received: 5 January 2024
Revised: 28 January 2024
Accepted: 5 February 2024
Published: 8 February 2024

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

At present, the processes of digital transformation are becoming more and more apparent and sought-after in many spheres of human society, including the spheres of medicine and healthcare. First of all, digital transformation in the spheres of medicine and healthcare is a complex continuous process that involves a complete restructuring of the fundamental principles of the functioning of medical organizations at all hierarchy levels, as well as the concept of their work with patients [1]. Now, the implementation of innovative digital technologies is aimed at establishing high standards of healthcare

delivery and moving towards the "4P medicine" model [1,2], which integrates preventive, personalized, participatory and predictive aspects of medical practice.

Digital transformation in sphere of healthcare consists of the transition from standard approved clinical approaches recommended for use in the examination and treatment of patients, to personal and individual approaches, as well as the prevention of certain diseases through timely early diagnosis, the constant monitoring of patients, active involvement of patients in treatment process, etc. [2]. An important key aspect of the digital transformation in the sphere of healthcare involves creating the prerequisites for reducing morbidity and mortality, as well as for increasing the active life expectancy of a person. Advanced health monitoring technologies should help not only to detect certain diseases in their early stages, but also to prevent disease progression through the application of innovative treatments for certain diseases.

The design of intellectual analysis tools for processing medical data of large and ultra-large volumes with the involvement of advanced machine learning (ML) [3–7] and deep learning (DL) [8–12] technologies provides an opportunity to receive and analyze new previously hidden knowledge both directly in the medical sphere and in related ones. These technologies are actively applied in solving various problems of medical diagnosis and, in particular, in solving problems of diagnosis of oncological diseases (ODs). The logistic regression algorithm [13], k-nearest neighbors (kNN) algorithm [14], support vector machine (SVM) algorithm [15], random forest (RF) algorithm [16] and DL algorithms [8–10] are usually applied when creating classifiers to solve OD diagnosis problems. Such classifiers are created on the basis of datasets which accommodate information about both patterns with diagnosed ODs of various types and patterns with unconfirmed ODs (i.e., normal patterns) [13–20].

Recently, applied and computational mathematics tools have become increasingly in demand, especially in the sense of digital transformation in healthcare. At the present time, the integration of applied and computational mathematics into digital health platforms is an important factor for efficient analysis and processing of medical data. These tools provide more accurate detection of pathologies at early stages of development, which is critical for the successful prevention of diseases, including cancer [13,21–23]. The use of applied and computational mathematics methods in the digital transformation of healthcare not only improves the accuracy of diagnosis but also contributes to the formation of innovative methods of disease prevention and treatment. As a result, the basis for personalized and predictive healthcare strategies is created, including strategies of cancer prevention and diagnosis.

Oncological diseases (ODs) are considered to be one of the most critically dangerous diseases due to the potentially severe consequences for patients, especially with late diagnosis [24,25]. These consequences include severe pain and serious psychological distress. Treatment for ODs can take a very long time. In addition, it is associated with huge financial costs both on the part of patients and on the part of the state. ODs are dangerous immune violations that cause abnormal cells to divide and grow in some organ of the patient. In addition, these immune violations can very quickly cover the patient's entire body and lead to a sad outcome. Clearly, timely early diagnosis of ODs is crucial so that the doctor can promptly choose an effective method for treating the patient. Unfortunately, the problem of early diagnosis of ODs is challenging and extremely difficult, because obvious characteristic symptoms do not manifest themselves until late in the course of the disease, so even innovative treatments may not be effective.

One approach to early diagnosis of ODs involves analyzing the results of various tests, for example, gene tests (GTs) [13,26] and protein tests (PTs) [13,27,28]. In the last few years, experts have favored PT-based OD diagnostic tools involving blood protein markers [13–16,26]. While GTs are static, PTs are dynamic, so PTs, when performed in a timely manner, can detect the disease onset and monitor its progression [13]. In addition, PTs are performed non-invasively. Also, they are cheap. It is assumed that PT-based

diagnostic technologies provide the forecasting of risks of cancer development for 1–3 years in advance and, therefore, allow us to carry out advanced prevention and diagnosis of ODs.

Various protein markers are present in the blood. It is known that for different types of ODs, the values of blood protein markers (BPMs) differ from each other [13]. It can be reasonably assumed that taking into account the entire spectrum of BPMs should provide an increase in the accuracy of diagnostics of various diseases. The use of data mining (DM) tools with the involvement of ML and DL technologies will allow us to reveal the relationships between the values of BPMs hidden in PT data for different types of ODs [13,14,20].

The main problem that arises when solving medical diagnosis problems is the class imbalance of the dataset [14,20]. As a rule, the number of normal data patterns describing situations with the absence of any ODs greatly exceeds the number of pathologic ones describing situations of OD of one type or another. As a result, the pattern class of normal data represents the majority class, while the pattern classes of pathologic data represent the minority classes (minority classes). When solving the problem of early diagnosis of ODs, the target classes, i.e., the classes whose correct classification of patterns is most important, are the minority classes. In addition, the medical diagnosis problem itself is usually a multiclass classification problem.

The problem of creating a multiclass classifier on the basis of an imbalanced dataset is very difficult, as the classifier must be trained to accurately classify the patterns of different classes that are imbalanced. Currently, a lot of approaches to overcoming the class imbalance problem have been proposed [29,30]. The most commonly used approaches are those that implement various class balancing algorithms realizing the strategies of oversampling [31–34], undersampling [34,35] and their combinations, as well as approaches that implement cost-sensitivity learning (CSL) and take into account the cost of incorrect decisions [14,36]. In [29], the authors show that, currently, there is no universal approach to address class imbalance. They propose a taxonomy that covers methods to eliminate class imbalance such as through performing cost-sensitive classification or through data sampling. The authors show that a lot of DM problems are cost-sensitive and class-imbalanced. In [30], the authors note that it is common to use data-level approaches, algorithm-level approaches, ensemble approaches and hybrid approaches to deal with class imbalance. They present a systematic literature review and perform an analysis of the studies presented in more than 400 papers from 2002 to June 2017. This analysis emphasizes the significant impact that inherent problems in the data have on the results obtained from classification problems. In addition, the analysis covers methods for handling imbalanced data and methods used to deal with skewed data distributions. The authors reveal trends and gaps in this sphere of research and discuss directions for future research.

Obviously, in each specific case, when applying one or another method to overcoming the class imbalance problem, it is necessary to check whether undesirable effects have not appeared, for example, in the form of a loss of representative data during undersampling, the appearance of mistaken or redundant data during oversampling, a significant increase in time for classifier development, a significant decrease in accuracy for classifying patterns of the majority class, etc. [20]. In this regard, when evaluating the quality of the created classifier, it is advisable to perform a thorough analysis of different classification quality metrics using the test set, particularly metrics which allow taking the dataset imbalance into account; e.g., such metrics as $F_1 - score$ and balanced accuracy can be used for this purpose. In addition, it is advisable to use k-fold cross-validation to empirically assess the generalization ability of the created classifier.

The problem of working with data accommodating information on blood protein markers is addressed in a number of scientific papers [13,21–25,37,38]. In particular, in [13], the so-called CancerSEEK test based on the logistic classifier (LC) is considered. The authors of the pilot study [13] choose eight types of ODs described by the patterns of this dataset due to the fact that these types are most often found in residents of Western countries and, additionally, because in clinical practice, blood tests were not applied for the

early identification of ODs. As a result, the dataset contains information about patterns belonging to nine classes: eight of them correspond to eight types of ODs, and another one corresponds to patterns for which no OD has been diagnosed. Also, the authors consider each individual's sex, levels of eight proteins and facts of mutations in 1933 various genomic locations. Based on the results of the experiments, the authors argue that blood protein markers reflect most of the information about the localization of ODs, because mutations in genes are most often not tissue-specific.

Later, there appeared research that proposed approaches to the development of classifiers that diagnose ODs based on datasets that accommodate data only on blood protein marker values [14,20], that is, they do not take into account data about the values of gene markers. Thus, in [14], the authors propose a cost-sensitive three-class kNN classifier created on the basis of the three-class imbalanced dataset extracted from the dataset used in [13]. The new dataset accommodates only patterns describing information for 39 blood protein markers mapped to 39 features. The authors extract additional information hidden in the 39-dimensional data patterns using sample entropy and approximate entropy, form two new features and add them to the used dataset, hoping to improve the data classification quality. In the research [20] performed earlier by the author of this paper, the aspects of kNN [14,39] and SVM [15,39] classifier development using sampling class balancing tools are considered. In this study, oversampling strategies are applied to balance classes [31–34]. In addition, the research explored various approaches to the formation of new features based on the methods for calculating entropy [40–45], Hjorth parameters [46,47] and fractal dimension [48,49], as well as on the basis of the UMAP (Uniform Manifold Approximation and Projection) algorithm [50–53], which is a nonlinear dimension reduction algorithm. New features were created to be added to the original dataset in different combinations, as well as to independently use these combinations as datasets when developing classifiers.

In general, we can note the high interest of scientists and practitioners in developing approaches to diagnosing cancer based on blood protein markers. At the same time, ideas are proposed for the development of both binary classifiers aimed at identifying any one cancer disease and multiclass classifiers seeking to identify different classes of diseases, which is much more difficult.

The aim of this research is to develop efficient classifiers of ODs (in terms of providing high values of classification quality metrics) using modern DM tools and ML techniques. We propose to develop SVM classifiers [15,20,39] using cost-sensitive algorithms that allow for the different estimation of classification errors in majority and minority classes when working with the original dataset and the extended datasets created on the basis of the original dataset using different tools for forming new features. In particular, when forming new features, as in [20], it is planned to use:

- The UMAP algorithm [50–53], which implements the nonlinear dimensionality reduction of data;
- Methods for calculating the approximate entropy (AE) [45] as well as Higuchi fractal dimension (HFD) [48] and Katz fractal dimension (KFD) [48].

While working with the UMAP algorithm, the plan is to investigate how different loss functions affect the results of embedding the original dataset into a lower-dimensional space. Also, we plan to research how the choice of loss function (LF) affects the quality of the extended datasets and the quality of the SVM classifiers developed from them.

In addition, it is planned to perform a comparative analysis of the SVM classifiers created in this study with the SVM classifiers created in [20], both in terms of the classification quality metrics and the time taken to train and test the SVM classifiers.

The rest of the paper is organized as follows. Section 2 is devoted to a review of works related to the presented research. Section 3 summarizes the design aspects of cost-sensitive SVM classifiers used to address the class imbalance problem in datasets. Furthermore, the applied quality metrics for multiclass classification are emphasized. In addition, a brief description of the principles of the UMAP algorithm and the various LFs used in it, which affect the results of embedding the original dataset from a high-dimensional space

into a low-dimensional space, is also provided. Moreover, the methods for computing the approximate entropy, Higuchi fractal dimension and Katz fractal dimension are mentioned. Section 4 presents the experimental results. First, a brief background to this research is given; in particular, a description of the approach to generating datasets and the previously obtained results of the creation of multi-class classifiers using oversampling algorithms are discussed. Then, aspects of the analysis of the original three-class datasets on the basis of the UMAP algorithm using various LFs are considered. Furthermore, the results of developing cost-sensitive SVM classifiers based on various datasets are discussed. Section 5 discusses the obtained results. Section 6 provides conclusions and purposes for future research. The section Appendix A contains the names of concepts and their abbreviations in Table A1 and reference information on the datasets and the composition of their features in Table A2. The section Appendix B contains figures visualizing the graphical dependencies for the LFs used in the UMAP algorithm when the original dataset is embedded in two-dimensional space. Section Appendix C contains information on the results of statistical tests with the developed models.

2. Related Work

In a pilot study [13], the authors propose to evaluate the levels of proteins and mutations in extracellular deoxyribonucleic acid and apply this information in a nine-class CancerSEEK test based on LC. The dataset applied during the development of the CancerSEEK test is available in the supplementary materials to the paper [13] under the name aar3247_cohen_sm_tables-s1-s11.xlsx. It should be noted that new data are constantly being added to this dataset. A regularly updated version of this dataset is openly presented in the repository titled as Catalog of Somatic Mutations in Cancer (COSMIC) [54] under the title NIHMS982921-supplement-Tables_S1_to_S11.xlsx. In developing the CancerSEEK test, the authors use 1005 patterns of data from patients with clinically identified "Breast", "Colorectum", "Esophagus", "Liver", "Lung", "Ovary", "Pancreas" or "Stomach" ODs. In doing so, they propose to consider each individual's sex, levels of eight proteins and facts of mutations in 1933 various genomic locations. The authors believe that the facts of the gene mutations or the growth in the level of any of the eight proteins allows a pattern of data to be classified as a pattern with detectable OD. They use 10-fold cross-validation to calculate values of the classification quality metrics and show that for all types of ODs, the mean value of such metrics as sensitivity is about 70%; however, there are great differences by classes: the lowest value of this metric, equal to 33%, is found for the breast class, and the highest value of this metric, equal to 98%, is found for the ovarian class. Also, authors show that the value of this metric depends significantly on the stage of the disease: the lower the disease stage number, the lower the sensitivity metric value.

In [14], the authors develop a cost-sensitive three-class kNN classifier on the basis of imbalanced dataset patterns describing data only for 39 blood protein markers. Each pattern belongs to one of the following classes: "Normal", "Ovary" and "Liver". The authors refused the idea of developing a nine-class classifier, as was performed in [14], because of the poor separability of the patterns of the nine classes. They expand the original dataset containing 39 features with 2 new features formed using sample entropy and approximate entropy, and they use the extended 41-dimensional dataset to create a cost-sensitive kNN classifier. In this research, the dataset contained 897 patterns belonging to one of three classes in the such ratio as "Normal":"Ovary":"Liver" = 799:54:44. The values of metrics such as $Precision$, $Recall$, $MacroF_1 - score$ and AUC were equal to 0.807, 0.833, 0.819 and 0.920, respectively. The overall accuracy of the classifier was equal to 0.952.

In study [20], the author of this paper develops kNN [14,39] and SVM [15,39] classifiers using such class balancing tools as SMOTE (Synthetic Minority Oversampling Technique) [31], Borderline SMOTE-1 [32], Borderline SMOTE-2 [32] and ADASYN (ADaptive SYNthetic sampling approach) [33], which allow us to restore class balance based on oversampling strategies. The creation of classifiers is performed on the basis of both the original dataset and the new datasets designed on the basis of the original dataset using

different tools for forming new features. Thus, five methods of entropy calculation [40–45], such as approximate entropy (AE), sample entropy (SE), singular value decomposition entropy (SVDE), spectral entropy (SPE) and permutation entropy (PE); two methods of Hjorth parameter calculation [46,47], such as Hjorth complexity and Hjorth mobility (HC and HM); and three methods for calculating fractal dimensions [48,49], such as Higuchi fractal dimension (HFD), Katz fractal dimension (KFD) and Petrosian fractal dimension (PFD) were used to form potentially new features from the 39-dimensional patterns of the original dataset. Based on the results of the analysis of values of the mean and the standard deviation (SD), calculated for each class, as well as correlation assessments, the methods for calculating approximate entropy AE, Higuchi fractal dimension HFD and Katz fractal dimension KFD were selected for further use. In addition, the UMAP algorithm (Uniform Manifold Approximation and Projection) [50–53], which implements non-linear dimensionality reduction on the data by embedding them in a lower-dimensional space, was applied to form new features from 39-dimensional patterns of the original dataset. The dimensionality reduction was performed not only to 2-dimensional space (as it is usually carried out when solving data visualization problems in two-dimensional space), but also to other dimensions (from 3 to 38) in order to select the best dimensionality reduction option in the context of solving the problem of achieving higher data classification quality.

The research in reference [20] is the first attempt to create datasets using different tools for generating new features by recovering data that are hidden in 39 features of the original dataset and then selecting the best tools for generating new features. In particular, the selection of new feature formation tools was founded on the correlation analysis of potentially new features among themselves, as well as on their ability to separate patterns that belong to different classes, using the mean and the SD values of features for each class. This approach to the creation of a group of datasets describing the subject area was first applied in the field of medical diagnostics, including the identification of ODs using blood protein markers. These datasets were further subjected to balancing using SMOTE, Borderline SMOTE-1, Borderline SMOTE-2 and ADASYN oversampling tools followed by choosing the best of them. The best oversampling tools were used in developing kNN and SVM classifiers followed by choosing the best classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. In this research, the best kNN classifier was created using the original dataset extended by the feature on the basis of approximate entropy, and the best SVM classifier was created using the original dataset extended by the feature on the basis of approximate entropy and 28 features on the basis of the UMAP algorithm. The mean values of the metric $MacroF_1 - score$ of kNN and SVM classifiers increased by 16.138% and 4.219%, respectively, in comparison with the mean values of this metric in the case when the original dataset was applied to create kNN and SVM classifiers. Thus, the mean values of the metric $MacroF_1 - score$ of the best kNN and SVM classifiers were 0.878 (with the SD value equal to 0.050) and 0.914 (with the SD value equal to 0.050), respectively. In addition, in [20], it was shown that it is promising to work with other considered tools of new feature formation, because their use in the datasets applied in the creation of classifiers also provided an improvement in data classification quality, although not as significant as the best classifiers mentioned above. Thus, the feasibility of using the proposed approach to form a group of datasets describing the subject area has been experimentally proven.

Despite the clearly significant results of the study performed in [20], we should note the drawbacks of the approach to solving the class imbalance problem, which implements work with oversampling tools that involve the synthesis of new patterns (perhaps never hypothetically possible). First, oversampling may lead to even more class mixing (in the case when classes are already poorly separable from each other) due to the impossibility of a priori accurate knowledge about the spatial geometry of data patterns because of the refusal (due to substantial time expenditures) to conduct additional research to study the spatial geometry of data patterns. However, in a sense, this disadvantage can be offset by evaluating the quality of the developed classifiers while screening out the unreliable ones. Second, oversampling always leads to an increase in the time cost of classifier development,

both because of the need to synthesize new patterns and because of the need to develop classifiers on significantly larger datasets.

Due to this, it is reasonable to consider approaches to overcoming the class imbalance problem, in which algorithms that take into account the cost sensitivity of wrong decisions are implemented. CSL, and, in particular, cost-sensitive algorithms, have been addressed and applied in a large number of research works. For example, such an approach is used in the aforementioned work [14] related to the development of a three-class kNN classifier for diagnosing ODs.

In [36], the authors note that CSL accounts for the misclassification cost and possibly costs of other types as well. The purpose of CSL is to minimize the total cost. CSL handles various classification errors differently. In particular, the classification cost of marking the positive data pattern as negative may not be equal to the classification cost of marking the negative data pattern as positive. Non-cost-sensitive learning does not account for the misclassification cost.

In [55], the authors show that the class imbalance problem (CIP) is one of the serious ML problems. Training on very imbalanced data leads to the fact that classifiers will be overloaded with data patterns from majority classes; therefore, the false negative rate will be high. They note that many methods are currently known to address the class imbalance problem, including sampling methods and CSL methods, but these methods are usually applied independently of each other. The authors propose two empirical methods on the basis of sampling methods and CSL methods. The first method suggests to create SVM classifiers conjoining sampling methods with CSL methods. The second method suggests to use CSL methods with a locally optimized cost matrix. The authors show that the first method allows reducing the misclassification costs, while the second method allows improving the classifier performance.

In [56], the authors argue that CSL has made significant efforts to address the CIP, but in practice, it is almost impossible to assess the misclassification cost exactly. Additionally, they show that the classification quality depends on the used feature subsets from the dataset and the values of the classifier parameters. The authors embed evaluation metrics such as AUC (Area Under Curve) and $G - mean$ (Geometric Mean) into the objective function to improve the classification quality of the cost-sensitive SVM classifier. They offer the method that tries to find the best combination of feature subsets, values of misclassification costs and values of the classifier parameters. The authors show that the proposed method is more efficient than commonly used sampling methods.

In [57], the authors propose robust cost-sensitive classifiers developed via the modification of the target functions of ML algorithms such as decision tree, random forest, extreme gradient boosting and logistic regression and apply them to efficiently predict medical diagnoses. Unlike sampling methods, the authors' approach does not change the original data distribution. The authors implement standard versions of the algorithms mentioned above and compare them with cost-sensitive versions. The cost-sensitive classifiers take into account the imbalanced class distribution during training, leading to more robust performance compared to classifiers on the basis of sampling methods.

In [58], the authors show that although methods such as cost-sensitive methods, sampling methods and ensemble learning methods can improve classification accuracy for minority class patterns, they are restricted by the problems of selection of cost parameter values and overfitting. The authors suggest a hybrid approach that includes data block construction, dimensionality reduction and ensemble creation with DL neural network classifiers. The effectiveness of the proposed hybrid approach is validated by experimental results using eight unbalanced datasets evaluated in terms of $Recall$, $G - mean$ and AUC.

In [59], the authors analyze CSL aspects and postulate its importance in medicine. They note that doctors are interested in models which can seek to minimize several types of healthcare-related costs such as the attribute cost (e.g., the diagnostic test cost) and the misclassification cost (e.g., the false negative test cost). They show that the diagnostic tests and the misclassification errors have high financial and human costs. The authors

propose ideas for dealing with CSL and its medical applications and provide an overview of research on CSL, including approaches and methods for the creation and evaluation of cost-sensitive classifiers.

Currently, especially in 2022–2023, there is a significant increase in the number of studies addressing aspects of the use of ML and DL technologies in the context of solving the problem of early diagnosis of cancer [3–7,11–14,20,26,37,38,60–66]. At the same time, many of them are aimed at solving the problem of diagnosing ODs on the basis of biomarkers, including blood protein markers [13,14,20–24,27,28,37,38,60]. However, most research solves the problem of binary classification with the identification of one specific disease, for example, breast [22], liver [12] or lung [37] cancer. It is obvious that the problem of multiclass classification is, on the one hand, more complex, but, on the other hand, the data used in its solution contain more complex dependencies, the restoration of which should help the rapid diagnosis of oncological diseases [13,14,20].

3. Materials and Methods

3.1. Aspects of Developing Cost-Sensitive SVM Classifiers

Various ML and DL algorithms can be used in the development of data classifiers, including multi-class classifiers, such as kNN [14,20,39,67,68], SVM [15,20,39,69,70], LR (Logistic Regression) [13,71] and RF [16,72,73], as well as algorithms on the basis of certain neural network architectures [8–12]. In addition, an increase in the quality of data classification can be achieved by applying cascade algorithms and ensembles on the basis of ML and DL algorithms. In this case, when developing certain classifiers, it is possible to use the default values of the parameters of the eponymous algorithms or to adjust the values of these parameters by applying, for example, grid search algorithms or well-established population optimization algorithms [74].

It should be noted that there are no universal approaches to classifier development which would guarantee that the classifier developed with their application will ensure high-quality classification for every task. In particular, the quality of data classification will depend both on the key features of the mathematical apparatus used in classifier development and on the specifics of the dataset used in classifier development, including its balance. In some cases, the time spent on classifier development, as well as on making classification decisions, may be of fundamental importance. Obviously, preference should be given to classifiers that provide high data classification quality with minimum time cost. For example, the kNN algorithm can be characterized as an algorithm that requires minimal time to develop a classifier as well as to make classification decisions, unlike the RF algorithm. The SVM algorithm is generally less time-consuming than the RF algorithm but more time consuming than the kNN algorithm. It is for this reason that the author of the study in [20] used the kNN and the SVM algorithms in the creation of the eponymous classifiers.

In this study, only the SVM algorithm is considered, and a cost-sensitive one at that. This choice is made in connection with the previously stated goal of the study. The rejection of the kNN algorithm used by the author of this study along with the SVM algorithm in [20] can be justified by the fact that the SVM classifiers provided a higher quality of data classification in the earlier study. Therefore, it was decided to conduct an additional study to see if the SVM classifiers could be developed with even higher quality by utilizing cost-sensitive learning principles.

Let $U = \{\langle x_1, y_1\rangle, \ldots, \langle x_s, y_s\rangle\}$ be the dataset applied in the creation of the SVM classifier, where x_i ($i = \overline{1,s}$) is the pattern described by q features; $x_i \in X$; X is the set of patterns; y_i is the class labels of the pattern x_i ($i = \overline{1,s}$); $y_i \in Y = \{1, \ldots, M\}$; Y is the set of pattern class labels; s is the number of patterns in the dataset U; M is number of classes in the dataset U [39].

Suppose that the SVM classifier is trained on S patterns. Additionally, the SVM classifier is tested on $s - S$ patterns. In this case, the k-fold cross-validation procedure can be used to assess the quality of the SVM classifier.

The base SVM algorithm assumes that $M = 2$, that is, the classification is binary, and implements binary data classification by constructing a hyperplane that separates the classes [39,62]. In this case, each data pattern $x_i \in X$ corresponds to a class label $y_i \in Y = \{-1; +1\}$ $(i = \overline{1,s})$.

When developing a binary SVM classifier, we must solve the problem of constructing a hyperplane that separates the classes. This problem can be reduced, accordingly with the Kuhn–Tucker theorem, to a quadratic programming problem that contains only dual variables λ_i $(i = \overline{1,S})$ [39,69]:

$$\begin{cases} \frac{1}{2} \cdot \sum\limits_{i=1}^{S} \sum\limits_{j=1}^{S} \lambda_i \cdot \lambda_j \cdot y_i \cdot y_j \cdot \kappa(x_i, x_j) - \sum\limits_{i=1}^{S} \lambda_i \to \min\limits_{\lambda}, \\ \sum\limits_{i=1}^{S} \lambda_i \cdot y_i = 0, \\ 0 \le \lambda_i \le C, \; i = \overline{1,S}, \end{cases} \tag{1}$$

where $\kappa(x_i, x_j)$ is the kernel function; C $(C > 0)$ is the value of the regularization parameter.

The kernel function $\kappa(x_i, x_j)$ can be linear, radial basis, polynomial or sigmoid, the last three of which carry out a transition to a higher-dimensional space than the original feature space in order to provide better separability of pattern classes from each other.

In the proposed study, as in [20], the radial basis function (RBF) kernel is applied. Such function can be defined as $\kappa(x_i, x_j) = exp\left(-\frac{(x_i - x_j) \cdot ((x_i - x_j))}{2 \cdot \sigma^2}\right)$, where σ $(\sigma > 0)$ is the kernel function parameter.

When creating a binary SVM classifier with an a priori defined RBF kernel function $\kappa(x_i, x_j)$, we must define the value of the parameter σ and the value of the regularization parameter C [62], which assures the minimum classification error. The task of searching for optimal values of these parameters can be solved on the basis of grid search (GS) algorithms or population optimization algorithms.

Support vectors, which are patterns of the original dataset located near the hyperplane that separates classes, are defined as a result of solving problem (1). They present all information about the class separation rules. For the support vectors, the values of the dual variables λ_i satisfy the condition $\lambda_i \neq 0$ [75].

The classification rule, according to which the membership class of pattern x is determined, has the following form [39,69]:

$$F(x) = sign\left(\sum_{i=1}^{S} \lambda_i \cdot y_i \cdot \kappa(x_i, x) + b\right) \tag{2}$$

where $b = \omega \cdot x_i - y_i$; $\omega = \sum_{i=1}^{S} \lambda_i \cdot y_i \cdot x_i$.

In order to form classification decisions in the case of multiclass classification, i.e., when the number of classes is greater than 2, either the OvR (One-vs-Rest) strategy or the OvO (One-vs-One) strategy is applied [75].

Since the purpose of this research is to create a multi-class cost-sensitive SVM classifier, we can use different values of the regularization parameter C_j $(j = \overline{1,M})$ for different classes by defining them as $C_j = weight_j \cdot C$, where $weight_j$ is the weight coefficient of the j-th class. Thus, $weight_j$ can be defined as $\frac{S}{M \cdot S_j}$, where S_j is the number of patterns in the j-th class; $\sum_{j=1}^{M} S_j = S$. Furthermore, we can consider arbitrary combinations of weights for different classes, assigning large weights $weight_j$ $\left(j = \overline{1,M}\right)$ to small classes. As a result, it will be possible to select such combinations of weights that guarantee high-quality data classification according to some quality metric at not the highest costs (penalties) for classification errors.

When evaluating the quality of multiclass classification, it is reasonable to use metrics such as $Accuracy$, $MacroPrecision$, $MacroRecall$ and $MacroF_1 - score$ [20,76]. Such met-

rics as $MacroPrecision$, $MacroRecall$ and $MacroF_1 - score$ are useful and effective when developing classifiers using imbalanced datasets [76].

We can calculate $Accuracy$, $MacroPrecision$ and $MacroRecall$ as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

$$MacroPrecision = \frac{1}{M}\sum_{m=1}^{M} \frac{TP_m}{TP_m + FP_m}, \tag{4}$$

$$MacroRecall = \frac{1}{M}\sum_{m=1}^{M} \frac{TP_m}{TP_m + FN_m}, \tag{5}$$

where TP is the total number of true positive patterns; TN is the total number of true negative patterns; FP is the total number of false positive patterns; FN is the total number of false negative patterns; M is the total number of classes; TP_m is the number of true positive patterns with the m-th class label; FP_m is the number of false positive patterns with the m-th class label; FN_m is the number of false negative patterns with the m-th class label.

The metric $MacroF_1 - score$ is based on metrics $MacroPrecision$ and $MacroRecall$. This metric allows us to simultaneously consider information about the precision and recall of the decisions formed by the classifier. It can be calculated as follows [20,76]:

$$MacroF_1 - score = 2 \cdot \frac{MacroPrecision \cdot MacroRecall}{MacroPrecision + MacroRecall}. \tag{6}$$

A high value of the metric $MacroF_1 - score$ means that the classifier has good performance on all the classes, whereas a low value of the metric $MacroF_1 - score$ means that the classes are poorly predictable [76].

In the proposed study, just like in [20], the SVM classifier having the maximum value of the metric $MacroF_1 - score$ is recognized as the best one.

3.2. Aspects of New Feature Generation

3.2.1. Generation of Features on the Basis of the UMAP Algorithm with Different Loss Functions

The UMAP algorithm carries out nonlinear dimensionality reduction by embedding patterns whose feature values are defined in high-dimensional space into lower-dimensional space. When performing such an embedding, the UMAP algorithm preserves local and global data structures that are defined in the high-dimensional space better [50,53] than similar algorithms, for example, the t-SNE (t-distributed stochastic neighbor embedding) algorithm [77].

Let $X = \{x_1, x_2, \ldots, x_s\}$ be some dataset. The UMAP algorithm embeds q-dimensional patterns x_i ($i = \overline{1,s}$) described by q features into h-dimensional space ($h \leq q$).

At the first stage, the UMAP algorithm constructs a fuzzy weighted undirected graph. At the second stage, the UMAP algorithm optimizes the LF [50].

At the first stage, k nearest neighbors [78] are found for each pattern x_i ($i = \overline{1,s}$), and then distances d_{il} ($l = \overline{1,k}$) to k nearest neighbors are calculated based on some distance metric (e.g., on the basis of the Euclidean metric). Next, the values ρ_i that define the distances to the nearest neighbor are found for each pattern x_i ($i = \overline{1,s}$).

Then, a binary search is carried out to look for the values σ_i that satisfy the condition:

$$\sum_{l=1}^{k} e^{\left(\frac{\rho_i - d_{il}}{\sigma_i}\right)} = \log_2 k. \tag{7}$$

An array $\mathcal{M}_i$ is formed for each pattern x_i ($i = \overline{1,s}$). The array's component μ_{ij} ($\mu_{ij} \in [0, 1]$) is a fuzzy number that defines how similar the i-th and the j-th patterns belonging to dataset X are. If the patterns x_i and x_j are not neighbors, the component μ_{ij} of

the array $\mathcal{M}_i$ is assumed to be 0. If the patterns x_i and x_j are neighbors, the component μ_{ij} of the array $\mathcal{M}_i$ is calculated as follows:

$$\mu_{ij} = e^{\left(\frac{\rho_i - d_{ij}}{\sigma_i}\right)}. \tag{8}$$

As a result, a weighted adjacency matrix $Matr \in \mathbb{R}^{s \times s}$ is formed in which the i-th row is defined on the basis of components from the array $\mathcal{M}_i$ ($i = \overline{1,s}$). The matrix $Matr$ is asymmetric. It defines the fuzzy weighted oriented graph that encodes the pairwise similarity of patterns x_i ($i = \overline{1,s}$).

At the second stage, the UMAP algorithm carries out the symmetrization of the matrix $Matr$ on the basis of the probabilistic t-conorm:

$$\mu_{ij} \leftarrow \mu_{ij} + \mu_{ji} - \mu_{ij} \cdot \mu_{ji} \ \left(i = \overline{1,s}\ ;\ j = \overline{1,s}\right), \tag{9}$$

where $\mu_{ll} = 0$ ($l = \overline{1,s}$).

Representations of q-dimensional patterns x_i ($i = \overline{1,s}$) in h-dimensional space as h-dimensional patterns y_i ($i = \overline{1,s}$) are computed using spectral embedding [50] ($h \leq q$). As a result, a dataset $Y = \{y_1, y_2, \ldots, y_s\}$ is generated.

The base UMAP algorithm implements optimization using an LF [79] that is a weighted fuzzy cross-entropy with reduced repulsion:

$$L(Matr, Y) = \sum_{i=1}^{s} \sum_{j=1}^{s} \left(\mu_{ij} \ln \frac{\mu_{ij}}{\nu_{ij}} + \frac{\sum_{k=1}^{s} \mu_{ik}}{2s} \ln \left(\frac{1 - \mu_{ij}}{1 - \nu_{ij}} \right) \right), \tag{10}$$

where $Matr \in \mathbb{R}^{s \times s}$ is a symmetric adjacency matrix containing fuzzy values that determine the pairwise similarity of patterns of high dimensionality (i.e., of dimensionality q) from the dataset X; $Y \in \mathbb{R}^{s \times h}$ is the representation of s patterns of low dimensionality (i.e., of dimensionality h); $\mu_{ij} \in [0,1]$ is the number that defines the fuzzy similarity of the i-th and the j-th patterns of high dimensionality that belong to the dataset X; $\nu_{ij} \in [0,1]$ is the number that defines the fuzzy similarity of the i-th and the j-th patterns of low dimensionality that belong to the dataset Y.

Pairwise similarity of the i-th and the j-th patterns of low dimensionality that belong to the dataset Y is defined as:

$$\nu_{ij} = \left(1 + a d_{ij}^{2b}\right)^{-1}, \tag{11}$$

where d_{ij} is the distance between the i-th and the j-th patterns of low dimensionality that belong to the dataset Y, calculated based on some distance metric (e.g., based on the Euclidean metric); a and b are coefficients fitted using the nonlinear least squares method (11) on the curve:

$$\psi_{ij} = \begin{cases} 1, & d_{ij} \leq d_{min} \\ e^{(d_{min} - d_{ij})}, & d_{ij} > d_{min} \end{cases}, \tag{12}$$

where d_{ij} is the distance between the i-th and j-th patterns of low dimensionality that belong to the dataset Y; d_{min} is the parameter ($d_{min} \in (0,\ 1]$) that influences the density of clusters created in the low-dimensional space during the optimization of the LF.

In the proposed study, as in the earlier study [20], the Euclidean metric is used as the distance metric.

The base UMAP algorithm applies the stochastic gradient descent (SGD) algorithm to optimize the LF (10) [50]. Pattern representations y_i ($i = \overline{1,s}$) of low dimensionality from the dataset Y are refined at each iteration of the SGD algorithm during minimization of the LF (10).

In addition to the LF (10), other LFs can be used.

We will additionally use the LFs described in [53] and presented below.

The LF based on fuzzy cross-entropy can be written as [53]

$$L_1(Matr, Y) = \sum_{i=1}^{s}\sum_{j=1}^{s}\left(\mu_{ij}\ln\frac{\mu_{ij}}{\nu_{ij}} + (1-\mu_{ij})\ln\left(\frac{1-\mu_{ij}}{1-\nu_{ij}}\right)\right). \quad (13)$$

The LF based on symmetric fuzzy cross-entropy can be written as [53]

$$L_2(Matr, Y) = \sum_{i=1}^{s}\sum_{j=1}^{s}\left((\mu_{ij}-\nu_{ij})\ln\left(\frac{\mu_{ij}(1-\nu_{ij})}{\nu_{ij}(1-\mu_{ij})}\right)\right). \quad (14)$$

The LF based on intuitionistic fuzzy cross-entropy can be written as [53]

$$L_3(Matr, Y) = \sum_{i=1}^{s}\sum_{j=1}^{s}\left(\mu_{ij}\ln\frac{\mu_{ij}}{\frac{1}{2}\mu_{ij}+\frac{1}{2}\nu_{ij}} + (1-\mu_{ij})\ln\left(\frac{1-\mu_{ij}}{1-\frac{1}{2}(\mu_{ij}+\nu_{ij})}\right)\right). \quad (15)$$

In the following, for convenience of presentation, we will refer to the LF (10) based on weighted fuzzy cross-entropy with reduced repulsion as L_4.

LFs (10) and (13)–(15) determine how the embedding of q-dimensional patterns x_i ($i = \overline{1,s}$) into h-dimensional space ($h \leq q$) will look like.

It should be noted that in the author's study [50], function (10) is stated as an LF, but in fact, in the author's version of the UMAP algorithm software (v. 0.5.5) library in Python [70], the LF is not explicitly specified, and the optimization process itself when performing pattern embedding from a high-dimensional space to a low-dimensional space can be described precisely by the LF (10), as proven by the authors of study [79].

The main parameters involved in the work of the UMAP algorithm are the parameters k and d_{min}.

k is the number of nearest neighbors that are found for each pattern in the high-dimensional space. This parameter is responsible for controlling balance between local and global structures in the data. Low values of k (*n_neighbors* in the software library [80]) will force the UMAP algorithm to pay attention to a very local structure. Large values of k will force the UMAP algorithm to pay attention to larger neighborhoods of each pattern during assessment of the topological structure of the data, but in this case, it is possible to lose small detail structure wanting to cover more data. Traditionally, $k = 15$. In our research, we will enumerate values for parameter k from the range [10, 20] with a step of 5.

d_{min} is the threshold distance ($d_{min} \in (0, 1]$). This parameter influences the density of clusters created in the low-dimensional space during the optimization of the LF. It is responsible for controlling how densely UMAP algorithm packs points together. It defines the minimum distance between the patterns in the low-dimensional space. Low values of d_{min} (*min_dist* in the software library [80]) will lead to clumpier embeddings. Larger values of d_{min} will make it possible to avert from packing patterns together and focus on the preserving of the broad topological structure. Traditionally, $d_{min} = 0.1$. In our research, we will enumerate values for parameter d_{min} from the range [0.1, 0.3] with a step of 0.1.

The UMAP algorithm also works with the parameter h, which determines the dimension of low-dimensional space. Traditionally, $h = 2$. In our research, we will enumerate values for parameter h from the range [2, 38] with a step of 2 to investigate the performance of the UMAP algorithm when embedding data in spaces with dimensions other than 2.

3.2.2. Generation of Features on the Basis of the Approximate Entropy, the Higuchi Fractal Dimension and the Katz Fractal Dimension

The study carried out in [20] has shown the feasibility of using the methods of calculating the approximate entropy AE [45], Higuchi fractal dimensionality HFD [48] and Katz fractal dimensionality KFD [48] for the creation of new features. These are the methods that will be used in the proposed research. They are described in detail in [20].

3.2.3. Computational Complexity of Developing Classifiers

The assessment of the computational complexity of developing the proposed SVM classifiers can be performed as follows. Let s be the number of patterns in the dataset and q be the number of features.

The computational complexity of the standard SVM classifier training has both a quadratic component and a cubic one [81]. It increases at least like s^2 when the value of the regularization parameter C is small and like s^3 when the value of the regularization parameter C is large [81,82]. In general, the computational complexity of the standard SVM classifier training can be estimated as $O(qs^3)$.

The computational complexity of the UMAP algorithm realization can be estimated as $O(qs^2)$ [50,83].

The computational complexity of the approximate entropy calculation method can be estimated as $O(q^2)$ [84,85]. Because we calculate the approximate entropy for each pattern in the dataset, the total computational complexity can be estimated as $O(q^2s)$.

The computational complexity of the methods for calculating the Higuchi fractal dimension and the Katz fractal dimension can be estimated as $O(q^2)$ [48]. Because we calculate the fractal dimensions for each pattern in the dataset, the total computational complexity can be estimated as $O(q^2s)$.

Thus, the main computational complexity comes from training the SVM classifier. However, state-of-the-art SVM realizations typically have computational complexity that scales between $O(s)$ and $O(s^{2.3})$, if we assume that the number of features q is not large compared to the number patterns s [86]. We can improve the computational complexity to $O(s)$ using parallel mixture [87]. Also, we can use such modern solvers as the Pegasos SVM [88] and the quantum SVM [89] to improve the computational complexity of standard SVM classifier training. Hence, this complexity can be reduced to quadratic (and even lower), both taking into account the specifics of the dataset and through the use of modern solvers. Though, these are only empirical observations and not theoretical guarantees [82].

Working with fractal dimensions does not make a significant contribution to the computational complexity of the developed classifiers. The computational complexity of the UMAP algorithm implementation and the computational complexity of the approximate entropy calculation method (taking into account working with all patterns in the dataset) are comparable.

Thus, the computational complexity of developing one SVM classifier in this research in the worst case is determined as $O(s^3)$, but in some cases it can be estimated as $O(s^2)$. When working with modern solvers, we can obtain computational complexity of $O(s^2)$ (if the UMAP algorithm is used when creating a dataset) and even less (if only entropy and fractal dimension calculation methods are used when creating a dataset).

4. Experimental Studies

All experimental studies were performed in the interactive cloud environment Google Colab. We used the Python 3.10 programming language for software development, since it allows us to work with a large number of different software libraries, including libraries that implement ML algorithms.

A three-class dataset on oncological diseases accommodating data on 39 serum protein markers (39 features) for 910 patterns was used in experimental studies. The following classes are considered: “Normal” (corresponding to the case when no OD was detected), “Liver” and “Ovary”. Each protein marker in this dataset corresponds to a feature in the dataset. A list of serum protein markers can be found in [20]. This dataset is extracted from the original nine-class dataset, which contains 1817 patterns of classes such as “Normal”, “Breast”, “Colorectum”, “Esophagus”, “Liver”, “Lung”, “Ovary”, “Pancreas” and “Stomach”. Such a dataset with nine classes is publicly available in the COSMIC repository [54]. However, as shown in [14,20], the nine-class dataset is characterized by both poor separability of classes from each other and significant imbalance. In this regard, an attempt to work to restore class balance through the use of sampling strategies will not give the desired

result. The use of oversampling strategies can lead to an even greater deterioration of the situation in the context of class separability as a result of even greater mixing of patterns of different classes, and the use of undersampling strategies can lead to the unreasonable deletion of representative patterns belonging to the "Normal" class, which is the biggest one, that is, the majority class. In [14,20], it is shown that the three-class dataset is also poorly balanced.

In [20], the visualization results using the UMAP algorithm in two-dimensional space for the original nine-class dataset and for the three-class dataset are presented. At the same time, it is assumed that the class separation in the three-class dataset should be better than in the original one. In this regard, it was the three-class dataset that was chosen for further research in [20]. It should be noted that the class ratio in a three-class dataset is as follows: "Normal":"Liver":"Ovary" = 812:44:54.

The UMAP algorithm has a library implementation in Python [80], proposed by the authors of this algorithm [50]. The authors argue that UMAP is a stochastic algorithm, so they use elements of randomness both to speed up the approximation steps and during the solution of optimization problems using the SGD algorithm.

Figure 1 shows the two-dimensional visualization for the three-class dataset using the library implementation of the UMAP algorithm with default parameter values of *n_neighbors* = 15, *min_dist* = 0.1, *metric* = *'euclidean'* and *random_state* = 42, where *n_neighbors* is the number of neighbors taken into account when assessing local and global properties of a diverse data structure [80]; *min_dist* is the parameter that determines the minimum distance at which data patterns can be located in low-dimensional space [80]; *random_state* is the parameter responsible for the initialization of UMAP algorithm and the reproducibility of results [80]; *metric* is the parameter that defines the metric used when calculating the distance between patterns [80]. Each two-dimensional point in Figure 1 corresponds to a 39-dimensional data pattern. The points corresponding to different classes are marked with different colors.

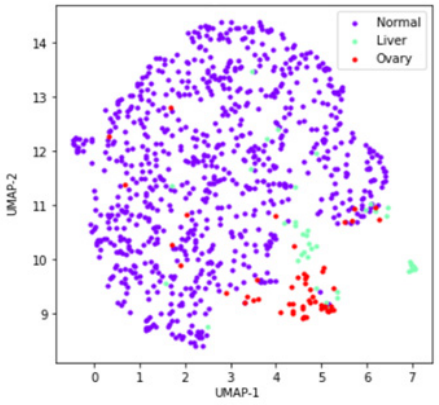

Figure 1. Two-dimensional visualization for three-class dataset using the library implementation of the UMAP algorithm [80] with the default parameter values (*n_neighbors* = 15, *min_dist* = 0.1, *metric* = *'euclidean'*, *random_state* = 42). The points corresponding to different classes are marked with different colors.

Since the dataset has three classes, the classification problem is multi-class, and its solution includes the development of a multi-class classifier.

4.1. Brief Background of This Study

Previously, in [20], approaches to the creation of three-class kNN and SVM classifiers based on datasets generated in various ways were investigated. The original dataset was tested for feature correlation. This test showed that there was no strong correlation between all the features. The correlation index values for all pairs of features, except one, turned out to be less than 0.6. Only for one pair of features with numbers 34 and 35

(sHER2/sEGFR2/sErbB2 (pg/mL) and sPECAM-1 (pg/mL)), the value of the correlation index was 0.604. Thus, the feasibility of using all features when performing further research was proven. Also, different options for extracting features from those in question were previously analyzed as well. In particular, five methods for calculating entropies such as approximate entropy (AE), sample entropy (SE), singular value decomposition entropy (SVDE), spectral entropy (SPE) and permutation entropy (PE); two methods for calculating Hjort parameters such as Hjorth complexity and Hjorth mobility (HC and HM); and three methods for calculating fractal dimensions such as Higuchi fractal dimension (HFD), Katz fractal dimension (KFD) and Petrossian fractal dimension (PFD) were considered. These methods were applied to the three-class dataset whose feature values were not subjected to preliminary scaling to [0, 1] to generate potential new features.

The values of entropies, Hjorth parameters and fractal dimensions were combined according to pattern class labels. Then, for each class, the mean value and the SD value of the potential new feature were calculated.

For each potential new feature, based on the results of the analysis of values of the mean and the SD for each class, the following conclusions were made. The biggest differences between the classes are observed for potentially new features based on the entropies AE and SE, as well as based on the fractal dimensions HFD and KFD. At the same time, the SD values for the aforementioned potentially new features are not large (and only for the feature based on the fractal dimension KFD they are slightly larger). It was these four potential new features that were selected for further consideration and examined for correlation with each other. We discovered that the potential new features based on the entropies AE and SE highly correlate with each other (the correlation score value is equal to 0.931). The feature based on the sample entropy SE was removed from consideration since it had a lesser correlation with the target feature that defines the pattern class label [20]. We also discovered that the potential new features based on the fractal dimensions HFD and KFD weakly correlate with each other (the correlation score value is equal to 0.141).

Thus, the preliminary analysis showed the feasibility of using the new feature on the basis of the approximate entropy AE as well as new features on the basis of the fractal dimensions HFD and KFD.

The additional experiments showed a slight excellence of the feature on the basis of the approximate entropy AE over the feature on the basis of the sample entropy SE in terms of maximizing the mean value of the metric $MacroF_1 - score$.

In addition, the library implementation of the UMAP algorithm [63] was used in [20] to embed a three-class 39-dimensional dataset into spaces of lower dimensions h ($h = 2, \ldots, 38$) to form new features.

Thus, the new generated features belonged to one of the following three groups:

- Feature on the basis of the approximate entropy AE;
- Features on the basis of the fractal dimensions HFD and KFD;
- Features on the basis of the UMAP algorithm.

We created the new datasets either by appending various combinations of new features of the three groups mentioned above to the original dataset or by appending various combinations of new features of the first two groups mentioned above to the features of the third group formed on the basis of the UMAP algorithm. As a result, we developed classifiers of 12 types: we developed classifiers of 4 types once (since we did not use the UMAP algorithm in the creation of the corresponding datasets) and classifiers of another 8 types over and over again (since we used the UMAP algorithm in the creation of the corresponding datasets for $h = 2, \ldots, 38$, where h is the dimension of the space into which the original dataset is embedded).

Figure 2 schematically shows all 12 ways for creating datasets. Table A2 in Appendix A contains information on the composition of features for all 12 datasets.

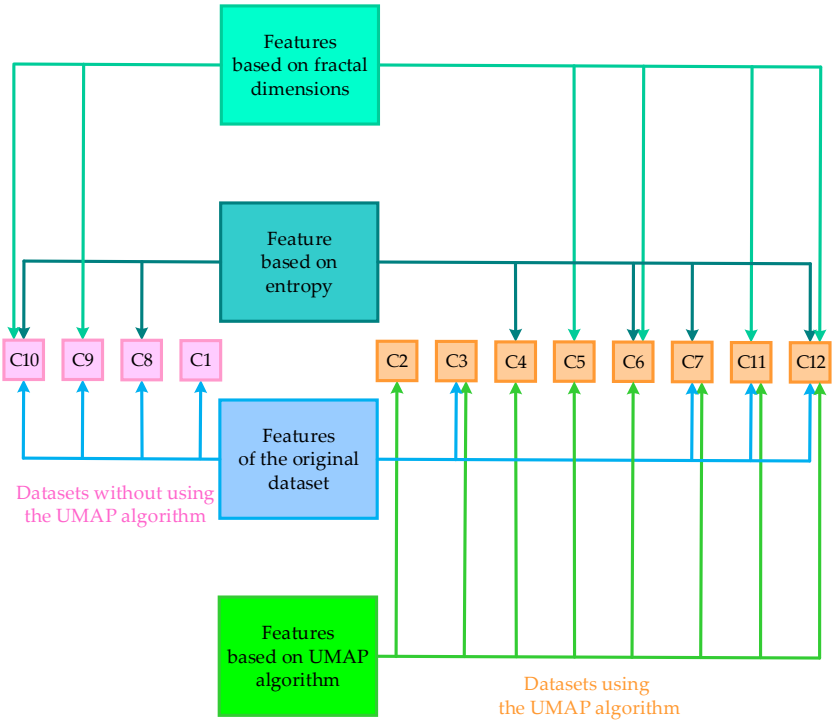

Figure 2. Scheme of creating of all 12 datasets applied in the research for the development of classifiers.

The developed classifiers subsequently have the same names as the datasets on the basis of which they are developed. Thus, the designation C7 in Figure 2 corresponds to a dataset obtained by supplementation of the original three-class dataset with the feature on the basis of the approximate entropy AE, as well as the features on the basis of the UMAP algorithm for a certain dimension h ($h = 2,\ldots, 38$) of space in which the original three-class dataset is embedded. Based on dataset C7 for a certain space dimension h, homonymous classifier C7 is developed. All 12 considered three-class datasets are unbalanced, since the three-class dataset C1 extracted from the original nine-class dataset is unbalanced.

It should be noted that all the principles formulated above for forming new groups of datasets can be applied when developing classifiers on the basis of any ML algorithms, since they only have an impact on the stage of preparing datasets applied in the creation of classifiers.

In [20], the CIP was overcome by applying oversampling algorithms such as SMOTE [31], Borderline SMOTE-1 [32], Borderline SMOTE-2 [32] and ADASYN [33]. Then, the best oversampling algorithms in terms of maximizing the mean value of the metric $MacroF_1 - score$ were selected. In this case, all datasets were either immediately used to develop classifiers or were first subjected to oversampling based on SMOTE, Borderline SMOTE-1, Borderline SMOTE-2 and ADASYN algorithms.

We found that Borderline SMOTE-1 is the best oversampling algorithm for developing kNN classifiers. Also, we found that the base SMOTE algorithm is the best oversampling algorithm for developing SVM classifiers.

The choice of these different oversampling algorithms was justified by the fact that these particular algorithms made it possible to provide the best classification quality assessed using the metric $MacroF_1 - score$ for the kNN and SVM classifiers.

The following conclusions were made based on the experimental results:

- SVM classifiers outperform kNN classifiers, both in the absence of oversampling and in the case of its use, in terms of maximizing the mean value of the metric $MacroF_1 - score$.

- In the case of oversampling, SVM classifiers provided the best classification quality assessed using the metric $MacroF_1 - score$, but the time spent on their development increased significantly.

In this regard, it was decided to continue researching the capabilities of SVM classifiers. In order to solve the CIP, it was decided to use the cost-sensitive algorithms instead of oversampling algorithms. If a positive result is obtained from experiments using cost-sensitive algorithms, we will be able to significantly reduce the time spent on creating SVM classifiers and, possibly, improve the data classification quality.

The method for finding the best SVM classifiers is described in detail in [20]. It involves searching through a grid of parameter values that provide the maximum value of the metric $MacroF_1 - score$.

In particular, we applied a grid search to find the values of parameters such as *C* and *gamma*, that, respectively, determine the regularization parameter and the parameter of the RBF kernel. The values of parameter *C* and parameter *gamma* varied in the range [0.4, 2] with a step of 0.1. We used the default values from the software implementation of the SVM algorithm in the scikit-learn library of Python as the values of the remaining parameters.

We used the metric $MacroF_1 - score$ as the main classification quality metric. This was performed to minimize the negative impact of the existing class imbalance in the original dataset C1 on the classification performance.

We applied a grid search to find the optimal values of the SVM classifier parameters [20] using stratified 10-fold cross-validation [90,91] with three-time repetition and the multi-class OvO strategy. We calculated the mean value of the metric $MacroF_1 - score$ with the corresponding SD value. Also, we calculated the mean values of the metrics *Accuracy*, *MacroPrecision* and *MacroRecall* with the corresponding SD values for the best classifiers. In addition, we determined the hyperparameter values for the best classifiers.

Table 1 demonstrates, for reference, the mean values of the metric $MacroF_1 - score$ and the corresponding SD values for the best classifiers of 12 types created without the application of oversampling algorithms [20]. Based on the information in Table 1, we can select best potential types of classifiers, which, in the future, first of all, should be paid attention to when developing classifiers based on CSL principles.

Table 1. Characteristic values of the best classifiers of different types created without the application of oversampling algorithms and cost-sensitive algorithms on the basis of the metric $MacroF_1 - score$.

Characteristic	Classifiers											
	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**
Mean	**0.877**	0.837	**0.864**	0.799	0.840	0.846	**0.873**	**0.885**	**0.878**	**0.880**	**0.866**	**0.876**
Standard deviation	0.078	0.095	0.074	0.101	0.090	0.090	0.075	0.079	0.078	0.074	0.078	0.080

Values greater than 0.850 are highlighted in bold.

As we can see, classifiers C3, C7, C8, C9, C10, C11 and C12 have the mean values of the metric $MacroF_1 - score$ exceeding 0.850. The mean values of the metric $MacroF_1 - score$ of these classifiers are highlighted in bold in Table 1. In the future, it will be interesting to evaluate the behavior of classifiers of these types if CSL principles will be applied for their development.

We found that that even the best SVM classifiers do not have good quality metric values due to class imbalance: the values of the metric $MacroF_1 - score$ are small, while all classifiers make many errors on patterns belonging to minority classes [20].

4.2. Experiments to Implement the Concept of This Study

When performing experiments in this study, it was decided to:

- Apply the principles of cost-sensitive algorithms for developing classifiers;
- Explore the possibility of generating new features using the UMAP algorithm, which uses five variants of the LF.

We plan to select the parameter values of the UMAP algorithm on the basis of the GS algorithm.

It should be noted that, as in [20], methods for calculating the approximate entropy AE, the Higuchi fractal dimension HFD and the Katz fractal dimension KFD will be used to generate new features.

We plan to use the following variants for the LF:

- Variant available in the library implementation in Python [80];
- Four variants described in [53] and available in the library implementation in Python [92].

Unfortunately, the LF in the library implementation in Python [80] is set implicitly, although the authors really use an SGD algorithm to solve an optimization problem when embedding the dataset from the high-dimensional space into the low-dimensional space; that is, in fact, the LF (13) declared by the authors in [50] is not used. The authors of study [79] tried to explicitly write down the formula for the LF, relying on the library implementation in Python [80]. They concluded that the LF resembles formula (10). In this case, with some error, they reproduced the work of the UMAP algorithm in the library implementation in Python [80]. In what follows, we will call the implicit LF used in the library implementation [80] as L_0 and the LF (10) as L_4.

The LFs in [92] are defined in accordance with (10) and (13)–(15). In this case, the optimization problem is solved using full gradient descent (FGD). The parameter value of *random_state* equal to 42 is used only when initializing the embedding of patterns into low-dimensional space.

We suggest using the following methodology when performing experiments.

1. Create datasets C1–C12 in accordance with the scheme presented in Figure 2, for a fixed combination of the parameter values (h, LF type, *n_neighbors*, *min_dist*), where h is the dimension of the low-dimensional space (h is selected from the range [2, 38] with a step of 2); the LF type is defined as one of the types in the list [L_0, L_1, L_2, L_3, L_4]; *n_neighbors* is the number of nearest neighbors of the data pattern in the high-dimensional space (*n_neighbors* is selected from the range [10, 20] with a step of 5); *min_dist* is the threshold distance that influences the density of clusters created in the low-dimensional space (*min_dist* is selected from the range [0.1, 0.3] with a step of 0.1). During the experiments, a walk through the grid is carried out for combinations of parameter values (h, LF type, *n_neighbors*, *min_dist*).
2. Develop cost-sensitive SVM classifiers based on datasets C1–C12 for a fixed combination of penalty values ($weight_1$, $weight_2$, $weight_3$) for the misclassification of patterns of different classes.
3. Assess the data classification quality using the cost-sensitive SVM classifiers based on datasets C1–C12 on the basis of stratified 10-fold cross-validation. Select classifiers that maximize the mean value of the metric $MacroF_1 - score$ (6). Analyze of the mean values of the metrics *Accuracy* (3), *MacroPrecision* (4) and *MacroRecall* (5). Assess the time spent training and testing the cost-sensitive SVM classifiers.

At step 1 of the methodology, we prepare different datasets that vary from each other in the composition of features. Additional information on the formation of such datasets is also reflected in [20].

At step 2 of the methodology, we can use the options for setting penalties for classification errors proposed in Section 3.1. Selecting fine values involves a certain amount of time. It can be performed by walking through the grid or by analyzing the most intuitively selected combinations of penalty values based on the following considerations: the smaller the number of patterns in a class, the greater the penalty for misclassifying patterns of this class. At the same time, considering various combinations of penalties and maximizing the mean value of the metric $MacroF_1 - score$, it is advisable to choose combinations with smaller penalties with similar mean values of the metric $MacroF_1 - score$. For the three-class dataset under study, we preferred the combination of penalties in the form ($weight_1$, $weight_2$, $weight_3$) = (1, 10, 10).

At step 3 of the methodology, it makes sense to pay special attention to the mean values of the metrics $MacroF_1 - score$ and $MacroRecall$: we maximize the first in order to select the best classifier, and the second allows us to assess whether there is an increase in the data classification quality in terms of sensitivity to identifying truly existing diseases. In addition, with similar values of the data classification quality for SVM classifiers, it makes sense to give preference to those whose training and testing time will be lesser (paying attention to time costs makes sense, for example, when comparing the cost-sensitive classifiers and classifiers created using oversampling technologies). It should be noted that when choosing classifiers for further use, it is advisable to pay attention to the results of certain statistical tests.

Figure 1 shows the two-dimensional visualization of the three-class dataset using the UMAP algorithm, for which there is a library implementation in Python [80], with the values of parameters *n_neighbors*, *min_dist*, *random_state* and *metric* set as default. These are the values of the parameters that were used in [20]. It should be noted that, unfortunately, this library implementation does not provide an explicit output of the values of the LF, and the results of the embedding, even with fixed values of the parameters *n_neighbors* and *min_dist*, depend on the initialization of the UMAP algorithm using the parameter *random_state.* In addition, the final results of the UMAP algorithm's application depend on the version of the library implementation, which is in a state of constant improvement, as well as on the version of the libraries associated with it, including the Numba library [93], which is responsible for parallelizing calculations and allows for speeding up the operation of the UMAP algorithm.

In the proposed study for the UMAP algorithm with LFs L_0, L_1, L_2, L_3 and L_4, the GS was performed for the values of parameters *n_neighbors* and *min_dist*, providing development of the best cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. In our research, we enumerate values for the parameter *n_neighbors* in the range [10, 20] with a step of 5, and enumerate values for the parameter *min_dist* in the range [0.1, 0.3] with a step of 0.1. In addition, we enumerate values for the parameter that defines the space dimension h in the range [2, 38] with a step of 2.

The tuple of parameter values (*n_neighbors*, *min_dist*) which allowed us to obtain a cost-sensitive SVM classifier of a specific type with the maximum possible mean value of the metric $MacroF_1 - score$ was considered the best.

It should be emphasized that the creation of cost-sensitive SVM classifiers was carried out. At the same time, different ratios of penalties for classification errors were considered for different classes, but in the end, a ratio of the form 1:10:10 was chosen, respectively, for the classes "Normal", "Liver" and "Ovary".

With this ratio of penalties for classification errors, the best cost-sensitive classifier C1 has a mean value of the metric $MacroF_1 - score$ equal to 0.907 (with the SD value equal to equal to 0.052). In what follows, we will consider this mean value of the metric $MacroF_1 - score$ to be the base (threshold) value. It is with this value that we will compare the mean values of the metric $MacroF_1 - score$ of cost-sensitive SVM classifiers of other types in order to select the truly best one, that is, superior to classifier C1, developed based on a cost-sensitive SVM algorithm.

Table 2 shows the main characteristics of the base classifier C1 [20], created on the basis of the original three-class 39-dimensional dataset not subjected to any manipulation, balanced classifier C1 using SMOTE algorithm [20], balanced classifier C7 using the SMOTE algorithm (at $h = 28$) [20] and the base cost-sensitive classifier C1, developed on the basis of the original three-class 39-dimensional dataset, which was not subjected to any manipulation.

Table 2. Characteristics of such SVM classifiers as the base C1, balanced C1, balanced C7 (at $h = 28$) and the base cost-sensitive C1.

Characteristic	Classifiers			
	Base C1 [20]	**Balanced C1 Using SMOTE Algorithm [20]**	**Balanced C7 (at *h*=28) Using SMOTE Algorithm [20]**	**Base Cost-Sensitive C1**
Number of features in the dataset	39	39	68	39
gamma	1.2	1	0.7	0.8
C	2.0	0.4	0.7	0.4
$MacroF_1 - score$ (mean/SD)	0.877/0.078	0.910/0.064	0.914/0.050	0.907/0.052
Accuracy (mean/SD)	0.973/0.015	0.977/0.015	0.978/0.012	0.977/0.013
MacroRecall (mean/SD)	0.843/0.088	0.907/0.081	0.907/0.065	0.907/0.066
MacroPrecision (mean/SD)	0.950/0.053	0.929/0.058	0.937/0.048	0.923/0.053
Training time (mean/SD), s.	0.123/0.008	0.886/0.214	0.489/0.021	0.169/0.023
Quality metrics calculation time (mean/SD), s.	0.007/0.001	0.013/0.004	0.008/0.001	0.011/0.003

It should be noted that the best kNN and SVM classifiers created on the basis of oversampling strategies and presented in [20] outperformed the cost-sensitive kNN classifier developed in [14] in terms of maximizing the mean value of the metric $MacroF_1 - score$. In addition, the best SVM classifier [20] outperformed the kNN classifier [20] in terms of the same indicator. In this regard, the main attention in the proposed study is paid specifically to the aspects of creating cost-sensitive SVM classifiers that have a higher data classification quality than previously developed classifiers.

As can be seen from Table 2, the base cost-sensitive classifier C1 has a higher mean value of the metric $MacroF_1 - score$ than the base classifier C1. However, this value in case of the cost-sensitive classifier C1 is less than that of classifier C1 balanced using the SMOTE algorithm [20] and classifier C7 balanced using the SMOTE algorithm (at $h = 28$) [20]. At the same time, the time spent on developing and testing base cost-sensitive classifier C1 is comparable to a similar time for the base classifier C1.

Figure 3a–e show the two-dimensional visualization of the three-class dataset using the UMAP algorithm with LFs L_0, L_1, L_2, L_3 and L_4 for tuples of parameter values (*n_neighbors*, *min_dist*) providing development of the best cost-sensitive SVM classifiers, called C3, for $h = 2$ in terms of maximizing the mean value of the metric $MacroF_1 - score$. It should be said that, generally speaking, the best SVM classifiers, called C3, can be obtained in spaces of dimension h other than 2. Each two-dimensional point in Figure 3a–e corresponds to a 39-dimensional data pattern. The points corresponding to different classes are marked with different colors.

Figure A1a–d, from Appendix B, show graphical dependencies for LF L_1, L_2, L_3 and L_4, obtained by constructing embeddings of the original 39-dimensional three-class dataset into the two-dimensional space, presented in Figure 3b–e. It should be noted that the ability to analyze and display the values of the LF L_0 is not provided by the library implementation [80]; therefore, graphical dependency is not shown.

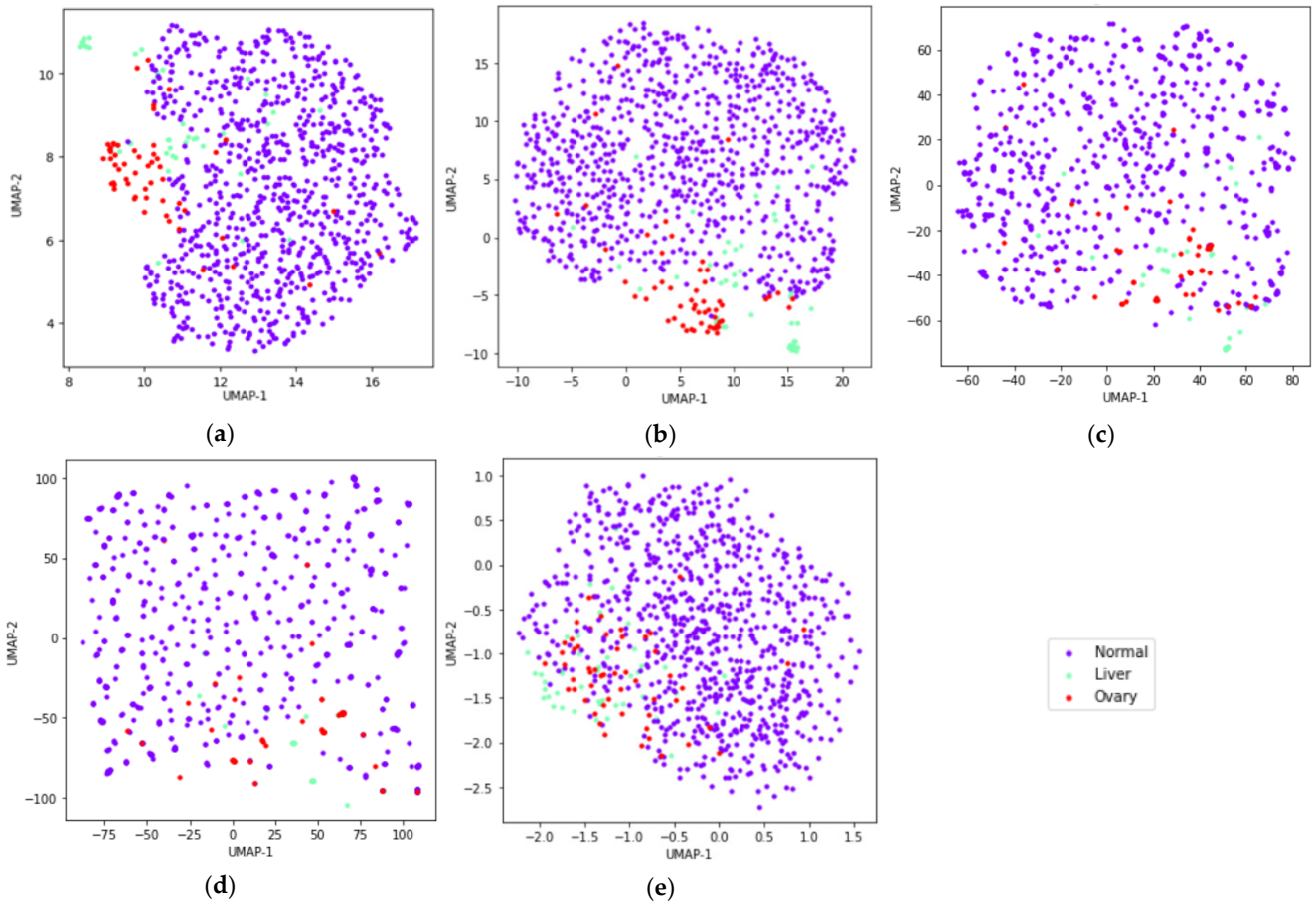

Figure 3. Two-dimensional visualization of the 39-dimensional three-class dataset using the UMAP algorithm with various LFs with parameter values *n_neighbors* and *min_dist*, ensuring the creation of the best cost-sensitive classifiers C3 (at $h = 2$) in the terms of maximizing the mean value of the metric $MacroF_1 - score$: (**a**) L_0: implicitly set LF (*n_neighbors* = 10; *min_dist* = 0.3; $MacroF_1 - score$: mean = 0.924; SD = 0.047); (**b**) L_1: LF based on fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$: mean = 0.918; SD = 0.049); (**c**) L_2: LF based on symmetric fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$: mean = 0.916; SD = 0.047); (**d**) L_3: LF based on intuitionistic fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.3; $MacroF_1 - score$: mean = 0.914; SD = 0.055); (**e**) L_4: LF based on weighted fuzzy cross-entropy with FGD (*n_neighbors* = 20; *min_dist* = 0.2; $MacroF_1 - score$: mean = 0.915; SD = 0.044). The points corresponding to different classes are marked with different colors.

The best cost-sensitive classifier C3 on the basis of the library implementation [80] has parameter values *n_neighbors* and *min_dist* (Figure 3a) different from the default values (Figure 1). At the same time, the mean values of the metric $MacroF_1 - score$ for the best cost-sensitive classifier C3 and the cost-sensitive classifier C3 with the default values of the parameters *n_neighbors* and *min_dist* are equal to 0.924 (with the SD value equal to 0.047) and 0.913 (with the SD value equal to 0.045), respectively, i.e., the best cost-sensitive classifier C3 outperformed the cost-sensitive classifier C3 with default parameter values by 1.205%. It should be noted that in both cases, the used default value of the parameters *random_state* was equal to 42. We decided to check how choosing the value of parameter *random_state* affects the final quality of the cost-sensitive classifier C3. In order to do this, we varied the values of the parameter *random_state* from 1 to 50 with a step of 1. Unfortunately, it turned out that only in 11 cases out of 50, which is 22%, we were able to receive a mean values of the metric $MacroF_1 - score$ of no less than 0.907. Moreover, only in 7 cases out of 50, which is only 14%, we were able to receive mean values of the metric $MacroF_1 - score$ of no less than 0.908: three of them turned out to be equal to 0.908, one of them turned

out to be equal to 0.909, two of them turned out to be equal to 0.911 and only one of these, which is only 2%, turned out to be 0.924. In this regard, the following conclusions can be deduced: Indeed, we can reduce the dimension of space even to 2, obtaining in some cases mean values of the metric $MacroF_1 - score$ of no less than 0.907, which is not bad, because in this case, it is possible to reduce the dimension of space from 39 to 2. However, such cases occur rarely, and searching for them is associated with the additional time expenditures. The case where the mean value of the metric $MacroF_1 - score$ is 0.924 turned out to be the only one. The remaining cases in which it was possible to obtain mean values of the metric $MacroF_1 - score$ of no less than 0.907 did not occur often, and the corresponding mean values of the metric $MacroF_1 - score$ turned out to be significantly less than the found maximum mean value of 0.924. So, obviously, one should not expect that simply iterating over the values of the parameter *random_state* will quickly lead us to the desired result, namely, to mean values of the $MacroF_1 - score$ of no less than 0.924. We can say that the use of the SGD algorithm in the problem under consideration, although it makes it possible to reduce the time spent searching for a solution, in most cases leads to finding only certain local extrema. Therefore, to find better solutions, i.e., solutions close to the global extremum, repeated runs of the SGD algorithm are required. So, choosing a different value for the parameter *random_state* other than 42 does not guarantee that we will obtain a mean value of the metric $MacroF_1 - score$ for cost-sensitive classifier C3 no worse than the mean value of the metric $MacroF_1 - score$ for cost-sensitive classifier C1. We will look for classifiers that are no worse in terms of the mean value of the metric $MacroF_1 - score$ than the cost-sensitive classifier C1, assuming that, perhaps, with equal mean values of the $MacroF_1 - score$ we will be able to decrease the number of features in the datasets on the basis of which cost-sensitive classifiers will be developed. A smaller number of features in the dataset that is applied for a certain classifier development, with all other characteristics being equal for the compared classifiers, can be considered a positive property of this classifier.

As can be seen from Figure 3a–e, only four LFs, named L_0, L_1, L_2 and L_4, provide good separation of classes in the two-dimensional space. They are used in the formation of two-dimensional datasets, a visualization of which is presented in Figure 3a–c,e.

However, for the purity of the experiment and the formation of convincing conclusions, we conducted experiments with all five LFs, without abandoning the LF L_3, for all values of the space dimension h indicated above, i.e., for values h from 2 to 38 with a step of 2.

It should be noted, based on the results obtained when developing cost-sensitive classifiers C3 (at $h = 2$) based on different LFs, that the best cost-sensitive classifiers C3 have parameter values recommended for use by default in the library implementation of the UMAP algorithm in only two out of five cases [80]. It can be assumed that the best cost-sensitive SVM classifiers on the basis of the UMAP algorithm result in spaces with dimension h different from 2 and may have parameter values of *n_neighbors* and *min_dist* different from those that are recommended to be used by default in the library implementation of the UMAP algorithm [80].

Next, the responses to two research questions (RQs) will be presented.

Question 1. Which types of cost-sensitive SVM classifiers are the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ when using LFs L_0, L_1, L_2, L_3 and L_4 for different combinations of values of parameters *n_neighbors* (in the range [10, 20] with a step of 5) and *min_dist* (in the range [0.1, 0.3] with a step of 0.1), as well as different values of the space dimension h (from 2 to 38 with a step of 2)?

Question 2. How often do cost-sensitive SVM classifiers of each type have mean values of the metric $MacroF_1 - score$ no lower than the base (threshold) mean value of the metric $MacroF_1 - score$ inherent to the base cost-sensitive classifier C1 and equal to 0.907, when using different LFs for different combinations of values of parameters *n_neighbors* (from 10 to 20 with a step of 5) and *min_dist* (from 0.1 to 0.3 with a step of 0.1), as well as different values of the space dimension h (in the range [2, 38] with a step of 2)?

4.2.1. Identifying the Best Cost-Sensitive SVM Classifiers and Analysis of Their Characteristics

In order to answer Question 1, Table 3 shows the names of the best cost-sensitive SVM classifiers and their characteristics, such as the mean value and the SD value of the metric $MacroF_1 - score$, as well as the dimension of space h (in the case of using the UMAP algorithm) in the format classifier name/mean/standard deviation/space dimension.

Table 3. Names of the best cost-sensitive SVM classifiers and their characteristics.

Tuple of Parameter Values (*n_neighbors*, *min_dist*)	Loss Functions				
	L_0	L_1	L_2	L_3	L_4
(10, 0.1)	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-
(10, 0.2)	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-
(10, 0.3)	**C3/0.924/0.047/2** *	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-
(15, 0.1)	C8/0.918/0.047/-	C7/0.919/0.046/22	C7/0.917/0.048/12	C8/0.918/0.047/-	C8/0.918/0.047/-
(15, 0.2)	C8/0.918/0.047/-	C7/0.919/0.042/8	C7/0.919/0.042/6	C8/0.918/0.047/-	C7/0.918/0.046/6
(15, 0.3)	C8/0.918/0.047/-	C7/0.920/0.044/28	*C7/0.921/0.048/26*	C8/0.918/0.047/-	C8/0.918/0.047/-
(20, 0.1)	C8/0.918/0.047/-	C8/0.918/0.047/-	C8/0.918/0.047/-	C3/0.918/0.045/12	C7/0.920/0.047/34
(20, 0.2)	C8/0.918/0.047/-	C7/0.920/0.052/30	C8/0.918/0.047/-	C8/0.918/0.047/-	C11/0.919/0.054/8
(20, 0.3)	C8/0.918/0.047/-	*C7/0.921/0.045/28* ***	***C7/0.923/0.047/28*** **	C8/0.918/0.047/-	*C3/0.921/0.045/6*

* the classifier, which took first place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold; ** the classifier, which took second place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold italic font; *** the classifiers, sharing third place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, are highlighted in italics.

We can see from Table 3 that the best cost-sensitive SVM classifiers for the considered tuples of the parameter values (*n_neighbors*, *min_dist*) turned out to be cost-sensitive classifiers C3, C7, C8 and C11. At the same time, cost-sensitive classifiers C7 and C8 most often took the lead, 11 and 30 times, respectively. Cost-sensitive classifiers C3 and C11 were leaders three times and one time, respectively.

In Table 3, the cost-sensitive SVM classifier, which is the absolute leader in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold. This is the cost-sensitive classifier C3 with values of the mean and the SD of the metric $MacroF_1 - score$ equal to 0.924 and 0.047, respectively. The dataset used in the development of this classifier was created by supplementing the original three-class 39-dimensional dataset with new features generated on the basis of the library implementation of the UMAP algorithm [80] (i.e., using the LF L_0). Based on the results of previously performed experiments with the LF L_0, we can conclude that its use when implementing the UMAP algorithm does not give the expected effect when working with spaces of dimensions h different from 2, with a fixed value of the parameter *random_state*, which affects the results of stochastic gradient descent. By default, in the library implementation of the UMAP algorithm [80], the value of the parameter *random_state* is 42. Previously performed experiments with different values of the parameter *random_state* at $h = 2$ did not improve the quality of cost-sensitive classifiers C3, so the feasibility of similar experiments with other values of the space dimension h is questionable, but it is related to large time expenditures.

In Table 3, the cost-sensitive SVM classifier, which took second place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold italic font. This is the cost-sensitive classifier C7 with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.923 and 0.047, respectively. This classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE as well as features on the basis of the UMAP algorithm at $h = 28$ using LF L_2.

In Table 3, three cost-sensitive SVM classifiers, sharing third place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, are highlighted in italics. These are such cost-sensitive SVM classifiers as the following:

- Classifier C7, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.045, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE, as well as features on the basis of the UMAP algorithm at $h = 28$ using the LF L_1);
- Classifier C7, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.048, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE, as well as features on the basis of the UMAP algorithm at $h = 26$ using the LF L_2);
- Classifier C3, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.045, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with features on the basis of the UMAP algorithm at $h = 6$ using the LF L_4).

In this case, obviously, preference should be given to the cost-sensitive classifier C3, because it allows us to work with low-dimensional dataset h (at $h = 6$) in the UMAP algorithm, unlike the other two cost-sensitive classifiers C7, for which the dimension of the low-dimensional space h in the UMAP algorithm is equal to 26 or 28.

As can be seen from Table 3, the LF turned out to be the most successful and reliable L_1 in the context of its use for the formation of new features: for all analyzed tuples of parameter values (*n_neighbors*, *min_dist*), using this function allowed us to develop cost-sensitive classifiers C7, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in five out of nine experiments. In this regard, we can conclude that the LF L_1 successfully copes with the problem of reducing dimensionality when embedding the original dataset into spaces of different dimensions h (both small and large) and can be recommended for further use when working with the UMAP algorithm.

Second place in the success rating was shared by LFs L_2 and L_4. The use of the LF L_2 made it possible to develop the cost-sensitive classifiers C7, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in four out of nine experiments.

The use of the LF L_4 allowed to develop two cost-sensitive classifiers C7, one cost-sensitive classifier C3 and one cost-sensitive classifier C11, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in four out of nine experiments. It should be noted that although one cost-sensitive classifier C7 (at $h = 6$) has the same mean value of the metric $MacroF_1 - score$ as the cost-sensitive classifier C8, we will assume that the cost-sensitive classifier C7 (at $h = 6$) is the winner, because it has a slightly lower standard deviation value (it is equal to 0.46), while the cost-sensitive classifier C8 has a standard deviation value of 0.47. However, 6 more additional features were used during development of the cost-sensitive classifier C7.

The LFs L_0 and L_3 in the proposed study did not show significant success in solving the problem of generating new features that would improve the data classification quality: the successes of these LFs, according to the experimental results given in Table 3, can rather be called random (single, characteristic only of individual tuples of parameter values (*n_neighbors, min_dist*). Working with these LFs can lead to a substantial increase in time expenditure without any guarantee that the results expected from them will be obtained. It should be emphasized that even the success of the LF L_0 is only partial: the results depend significantly on how successfully the initialization of the UMAP algorithm was performed.

Table 4 shows the main characteristics of the five best cost-sensitive classifiers from Table 3, ranked in the first three places, as well as the main characteristics of the base cost-sensitive classifier C1 (Table 2).

Table 4. Characteristics cost-sensitive the winning classifiers of the rating and the base cost-sensitive classifier C1.

Characteristics	Classifiers					
	Base Cost-Sensitive C1	**C3 (at *h*=2)**	**C7 (at *h*=28)**	**C7 (at *h*=26)**	**C7 (at *h*=28)**	**C3 (at *h*=6)**
Number of features in the dataset	39	41	67	65	67	45
Loss function	-	L_0	L_1	L_2	L_2	L_4
n_neighbors	-	10	20	15	20	20
min_dist	-	0.3	0.3	0.3	0.3	0.3
gamma	0.8	0.5	0.7	0.5	0.5	0.4
C	0.4	0.8	0.6	1	1	1.3
$MacroF_1 - score$ (mean/SD)	0.907/0.052	0.924/0.047	0.921/0.045	0.921/0.048	0.923/0.047	0.921/0.045
Accuracy (mean/SD)	0.977/0.013	0.980/0.012	0.979/0.011	0.980/0.012	0.981/0.010	0.979/0.012
MacroRecall (mean/SD)	0.907/0.066	0.920/0.061	0.911/0.056	0.913/0.065	0.916/0.062	0.915/0.060
MacroPrecision (mean/SD)	0.923/0.053	0.943/0.046	0.944/0.052	0.943/0.053	0.945/0.050	0.941/0.045
Training time (mean/SD), s.	0.169/0.023	0.238/0.041	0.133/0.013	0.125/0.012	0.130/0.008	0.231/0.030
Quality metrics calculation time (mean/SD), s.	0.011/0.003	0.017/0.007	0.011/0.002	0.012/0.001	0.013/0.001	0.017/0.007

As can be seen from Table 4, all winning cost-sensitive classifiers surpassed the base cost-sensitive classifier C1 in terms of maximizing of the mean value of the metric $MacroF_1 - score$ (note that it was previously decided to use the mean value of the metric $MacroF_1 - score$ of the base cost-sensitive classifier C1 as the base (threshold) values for comparison). In addition, all winning cost-sensitive classifiers outperformed the base classifier C1 (Table 2, [20]); the classifier C1, balanced using the SMOTE algorithm (Table 2, [20]); and the classifier C7, balanced using the SMOTE algorithm (at $h = 28$) (Table 2, [20]) in terms of maximizing of the mean value of the metric $MacroF_1 - score$. In this case, there is a decrease in the SD value for the metric $MacroF_1 - score$, especially compared to the same value of the base classifier C1 (Table 2, [20]).

In addition, we can notice an increase in the mean value of the metric *MacroRecall* with a decrease in the SD value for the metric *MacroRecall*, especially compared to the same value of the base classifier C1 (Table 2, [20]).

It should be noted that the winning cost-sensitive SVM classifiers are created on the basis of datasets whose number of features is greater than the number of features in the original 39-dimensional dataset.

Table 5 shows, for reference, the mean values of the metric $MacroF_1 - score$ and the corresponding SD values for the best cost-sensitive classifiers of 12 types. Bold font in Table 5 indicates information on the cost-sensitive classifiers that outperformed the base cost-sensitive classifier C1 (information for which is marked in bold italics) in terms of maximizing the mean value of the metric $MacroF_1 - score$.

We can notice that cost-sensitive classifiers of all types, with the exception of the cost-sensitive classifier C4, improved their mean values of the metric $MacroF_1 - score$ compared to similar classifiers, during the development of which no manipulations were used to overcome the problem of class imbalance (Table 2, [20]). First of all, it should be noted that classifiers C1, C3, C7, C8, C10, C11 and C12, whose mean values of the metric $MacroF_1 - score$ exceeded the similar value for the base classifier C1 (Table 2, [20]), turned out to be more than 0.900.

Table 5. Characteristic values for the best cost-sensitive classifiers of different types based on the metric $MacroF_1 - score$.

Characteristic	Classifiers											
	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**
Mean	***0.907*** *	0.874	**0.924** **	0.799	0.851	0.849	**0.923**	**0.918**	0.898	**0.915**	**0.919**	**0.913**
Standard deviation	***0.052***	0.079	**0.047**	0.108	0.065	0.067	**0.047**	**0.047**	0.063	**0.053**	**0.054**	**0.065**

* information for the base cost-sensitive classifier C1 is highlighted in italics; ** information for classifiers that outperformed the base cost-sensitive classifier C1 in terms of maximizing the mean value of the metric $MacroF_1 - score$ is highlighted in bold.

4.2.2. Identification of the Best Loss Functions in the UMAP Algorithm and Analysis of Their Capabilities in the Context of the Formation of New Features

In order to answer Question 2 in Table 6 for each tuple of parameter values (*n_neighbors*, *min_dist*) it is shown how many times a cost-sensitive classifier of a certain type performed no worse than the base cost-sensitive classifier C1 in terms of maximizing the mean value of the metric $MacroF_1 - score$. Moreover, the percentage of successfulness to the total number of experiments is indicated for each tuple of parameter values (*n_neighbors*, *min_dist*). The total number of experiments is 19, because the dimension of space h in the UMAP algorithm varies from 2 to 39 with a step of 2. Table 6 provides information only about those cost-sensitive classifiers that that were no worse than the base cost-sensitive classifier C1 more than once (in all experiments). It can be noted that, according to the information from Table 5, the cost-sensitive classifier C12 outperformed the base cost-sensitive classifier C1, but it did this only once (when using the LF L_0 in the UMAP algorithm (at $h = 2$, *n_neighbors* = 10 and *min_dist* = 0.3).

Table 6. Names of the best cost-sensitive SVM classifiers and their win rates.

Tuple of Parameter Values (*n_neighbors*, *min_dist*)	Loss Functions				
	L_0	L_1	L_2	L_3	L_4
(10, 0.1)	C3: 0 (0%) C7: 5 (26.316%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 2 (10.526%) C7: 18 (94.737%) C8: 19 (100%) C10: 9 (47.368%) C11: 2 (10.526%)	C3: 2 (10.526%) C7: 18 (94.737%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 1 (5.263%) C7: 2 (10.526%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 3 (15.789%) C7: 7 (36.842%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)
(10, 0.2)	C3: 1 (5.263%) C7: 7 (36.842%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 3 (15.789%) C7: 18 (94.737%) C8: 19 (100%) C10: 0 (0%) C11: 2 (10.526%)	C3: 2 (10.526%) C7: 17 (89.474%) C8: 19 (100%) C10: 2 (10.526%) C11: 0 (0%)	C3: 0 (0%) C7: 1 (5.263%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 3 (15.789%) C7: 3 (15.789%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)
(10, 0.3)	C3: 1 (5.263%) C7: 11 (57.895%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)	C3: 2 (10.526%) C7: 18 (94.737%) C8: 19 (100%) C10: 2 (10.526%) C11: 2 (10.526%)	C3: 2 (10.526%) C7: 17 (89.474%) C8: 19 (100%) C10: 1 (5.263%) C11: 0 (0%)	C3: 0 (0%) C7: 2 (10.526%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 3 (15.789%) C7: 3 (15.789%) C8: 19 (100%) C10: 0 (0%) C11: 2 (10.526%)
(15, 0.1)	C3: 1 (5.263%) C7: 4 (21.053%) C8: 19 (100%) C10: 1 (5.263%) C11: 0 (0%)	C3: 3 (15.789%) C7: 18 (94.737%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)	C3: 3 (15.789%) C7: 18 (94.737%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)	C3: 0 (0%) C7: 4 (21.053%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 4 (21.053%) C7: 15 (78.947%) C8: 19 (100%) C10: 0 (0%) C11: 3 (15.789%)
(15, 0.2)	C3: 2 (10.526%) C7: 8 (42.105%) C8: 19 (100%) C10: 1 (5.263%) C11: 0 (0%)	C3: 2 (10.526%) C7: 18 (94.737%) C8: 19 (100%) C10: 3 (15.789%) C11: 2 (10.526%)	C3: 0 (0%) C7: 19 (100%) C8: 19 (100%) C10: 5 (26.316%) C11: 2 (10.526%)	C3: 1 (5.263%) C7: 6 (31.579%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 4 (21.053%) C7: 9 (47.368%) C8: 19 (100%) C10: 0 (0%) C11: 5 (26.316%)
(15, 0.3)	C3: 0 (0%) C7: 4 (21.053%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 0 (0%) C7: 18 (94.737%) C8: 19 (100%) C10: 4 (21.053%) C11: 2 (10.526%)	*C3: 1 (5.263%)* C7: 19 (100%) C8: 19 (100%) C10: 8 (42.105%) C11: 2 (10.526%)	C3: 1 (5.263%) C7: 4 (21.053%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 6 (31.579%) C7: 7 (36.842%) C8: 19 (100%) C10: 1 (5.263%) C11: 4 (21.053%)

Table 6. *Cont.*

Tuple of Parameter Values (*n_neighbors*, *min_dist*)	Loss Functions				
	L_0	L_1	L_2	L_3	L_4
(20, 0.1)	C3: 1 (5.263%) C7: 3 (15.789%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 3 (15.789%) C7: 19 (100%) C8: 19 (100%) C10: 6 (31.579%) C11: 3 (15.789%)	C3: 1 (5.263%) C7: 15 (78.947%) C8: 19 (100%) C10: 4 (21.053%) C11: 1 (5.263%)	C3: 2 (10.526%) C7: 2 (10.526%) C8: 19 (100%) C10: 0 (0%) C11: 2 (10.526%)	C3: 9 (47.368%) C7: 16 (84.211%) C8: 19 (100%) C10: 1 (5.263%) C11: (36.842%)
(20, 0.2)	C3: 1 (5.263%) C7: 6 (31.579%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 4 (21.053%) C7: 19 (100%) C8: 19 (100%) C10: 4 (21.053%) C11: 3 (15.789%)	C3: 2 (10.526%) C7: 18 (94.737%) C8: 19 (100%) C10: 0 (0%) C11: 1 (5.263%)	C3: 1 (5.263%) C7: 1 (5.263%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 15 (78.947%) C7: 17 (89.474%) C8: 19 (100%) C10: 2 (10.526%) C11: 13 (68.421%)
(20, 0.3)	C3: 1 (5.263%) C7: 7 (36.842%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 5 (26.316%) C7: 19 (100%) C8: 19 (100%) C10: 3 (15.789%) C11: 5 (26.316%)	C3: 1 (5.263%) C7: 19 (100%) C8: 19 (100%) C10: 11 (57.895%) C11: 2 (10.526%)	C3: 1 (5.263%) C7: 2 (10.526%) C8: 19 (100%) C10: 0 (0%) C11: 0 (0%)	C3: 14 (73.684%) C7: 16 (84.211%) C8: 19 (100%) C10: 4 (21.053%) C11: 11 (57.895%)

Analysis of the results given in Table 6 confirms the clear advantage of the LF L_1: its use allows us to develop cost-sensitive SVM classifiers of different types that exceed the base cost-sensitive classifier C1 in terms of maximizing of the mean value of the metric $MacroF_1 - score$.

In addition, analysis of the results given in Table 6 allows us to notice that, usually, the number of successful cost-sensitive classifiers C3 for each tuple of parameter values (*n_neighbors*, *min_dist*) is no more than three. Most often, such situations arise when the dimensionality h of the low-dimensional space in the UMAP algorithm is two, four or six. However, when using the LF L_4 in the UMAP algorithm, the number of successful cost-sensitive classifiers C3 for each tuple of parameter values (*n_neighbors*, *min_dist*) is always at least 3, and for tuples (*n_neighbors*, *min_dist*) taking values (20, 0.2) and (20, 0.3), the number of successful cost-sensitive classifiers C3 is 15 and 14, respectively (i.e., such classifiers are successful with different dimensions h of space (both small and large)).

It should be noted that all cost-sensitive SVM classifiers, indicated in Table 6, are developed based on datasets whose number of features is greater than the number of features in the original 39-dimensional dataset.

5. Discussion

Experimental results of creating SVM classifiers using the cost-sensitive SVM algorithm confirmed that high data classification quality can be achieved through modification of the original dataset by adding different combinations of new features on the basis of the approximate entropy AE and the fractal dimensions KFD and HFD, as well as on the basis of the UMAP algorithm. The most successful in terms of maximizing the mean value of the metric $MacroF_1 - score$ turned out to be classifiers C3, C7 and C8, developed, respectively, on the basis of the original three-class 39-dimensional datasets, supplemented, respectively, with new features on the basis of the UMAP algorithm; on the basis of the UMAP algorithm and the approximate entropy AE; as well as only on the basis of the approximate entropy AE.

All winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the base SVM classifier (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 5.359%, 5.245%, 4.789%, 4.675%, 4.333% and 4.105%, respectively.

All winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the base cost-sensitive classifier C1 (Table 2) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 1.874%, 1.764%, 1.323%, 1.2135, 0.882% and 0.662%, respectively.

Also, all winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the best classifier C1 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing of the mean value of the metric $MacroF_1 - score$ by 1.538%, 1.429%, 0.989%, 0.879%, 0.549% and 0.330%, respectively.

In addition, five out of the six winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5)—namely classifiers C3, C7, C11, C8 and C10, presented in descending order of the mean values of the metric $MacroF_1 - score$—outperformed the best classifier C7 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 1.094%, 0.985%, 0.547%, 0.438% and 0.109%, respectively. Only classifier C12 turned out to be worse than the best classifier C7 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 0.109%.

We can see that the advantage of the best cost-sensitive classifiers over the best classifiers C1 and C7 on the basis of the SMOTE algorithm is not very large. However, it was possible to significantly reduce the time expenditures for developing and testing classifiers (Table 4) compared to similar time estimates obtained when developing classifiers using the SMOTE algorithm, which implements the strategy of oversampling new data patterns (Table 2, [20]).

So, for example, we can compare the total time spent on the training and testing of the cost-sensitive classifiers C7, recognized as the best in the proposed study and in [20]. They are created on the basis of the original 39-dimensional dataset, supplemented with features created on the basis of the approximate entropy AE and the UMAP algorithm. At the same time, in [20], the library implementation of the UMAP algorithm with the default parameter values is applied, and in the proposed study for the UMAP algorithm, the value enumeration of the parameters *n_neighbors* and *min_dist* is implemented for five LFs with the choice of the best variant. However, the CIP is solved differently in the proposed study and in [20]. The best SVM classifier in [20] is the classifier C7, in which the CIP was solved using the oversampling SMOTE algorithm. One of the best cost-sensitive SVM classifiers in the proposed study is the cost-sensitive classifier C7 (Table 4), for which the CIP was solved using the CSL concept. In this case, the LF L_2 was used in the UMAP algorithm, and the parameters *n_neighbors* and *min_dist* took the values 20 and 0.3, respectively. The total time spent on the training and testing of the cost-sensitive classifier C7 in the proposed study turned out to be only 1.1 times longer than the time to develop the base classifier C1 and 1.26 times less than the time to develop the base cost-sensitive classifier C1, while it took 3.48 times less than the same time for classifier C7 built using the SMOTE algorithm and being the best in [20], as well as 6.29 times less than the same time for classifier C1 built using the SMOTE algorithm [20].

The function L_1 that implements the calculation of fuzzy cross-entropy with FGD should be recognized as the best function in the context of working with different LFs in the UMAP algorithm in order to form new features that complement the original dataset and ensure the development of classifiers with high data classification quality. The function L_3 that implements the calculation of intuitionistic fuzzy cross-entropy with FGD and then the LF L_0 that implements the calculation of implicitly set LF should be recognized as the worst LFs. Such conclusions were made based on the efficiency of these LFs in terms of embedding of the original 39-dimensional dataset into spaces of arbitrary dimensions h (from 2 to 38 with a step of 2) in the context of the formation of new datasets and the further development of a classifier with high data classification quality, superior to the quality of the base cost-sensitive classifier C1. Thus, the function L_1 was successful in these terms for all space dimensions h, while the successes of the function L_3 were solitary. The L_0 function's successes were also solitary. Although the use of the function L_0 made it possible to obtain the best classifier C3 in the terms of maximizing the mean value of the metric $MacroF_1 - score$, that is, the absolute winning classifier in our rating (Table 3), the use of this LF is associated with selecting the value of another parameter, namely the *random_state*

parameter, affecting the final results of UMAP algorithm. This leads to additional time expenditures without a guaranteed expected result. The LFs L_2 and L_4 turned out to be less successful than the LF L_1 in the terms under consideration but more successful than the LFs L_0 and L_3. However, their use in the UMAP algorithm made it possible to obtain the winning classifiers in our rating (Table 3), so it is advisable to use them (in the absence of significant restrictions on the time spent on the development of high-quality classifiers).

In order to statistically test the superiority of the developed classifiers, which solve the CIP in different ways, over other classifiers, we applied the Wilcoxon signed rank test [94,95] to the obtained quality estimates of various classifiers. To obtain statistically representative results, we repeated the evaluation of each pair of classifiers using stratified 10-fold cross-validation [90,91] with three-time repetition: each time, the datasets were divided into 10 blocks, and the classifiers were evaluated on 10 different parts of each dataset. This was performed three times. Thus, each of the resulting classifier quality distributions contained a total of 30 values. The distribution obtained for the base classifier C1 based on the original dataset was compared with all other distributions obtained for other classifiers in this study. When assessing the quality of the classifiers, we considered the metrics $MacroF_1 - score$ and $Recall$, as well as estimates of the training time and testing time of the classifiers. The values of metrics $MacroF_1 - score$ and $Recall$ should be maximized. Estimates of training time and testing time for classifiers should be minimized.

According to the null hypothesis H_0, the two compared distributions did not have statistically significant differences [94,95]. The p-value was set to 0.05. The obtained results are presented in Tables A4–A7 in Appendix C. We compared the classifiers developed in this study with the base SVM classifier based on the original dataset [20], with the SVM classifier based on the original dataset and the SMOTE algorithm [20], and also with the SVM classifier based on the original dataset expanded using features based on the approximate entropy AE, the UMAP algorithm and the SMOTE algorithm [20]. In Tables A4–A7, the "=" sign means that there are no statistically significant differences between the compared distributions of the classifiers, the "+" sign means that the classifier in the row header is superior to the classifier in the column header, and the "–" sign means the opposite.

According to Table A4, classifiers C3 (at $h = 2$) with L_0, C7 (at $h = 28$) with L_1, C7 (at $h = 26$) with L_2, C7 (at $h = 28$) with L_2, C3 (at $h = 6$) with L_4 and balanced C7 (at $h = 28$) using the SMOTE algorithm [20] surpassed the base classifier C1 as measured by the metric $MacroF_1 - score$. The base cost-sensitive classifier C1 and the balanced classifier C1 using the SMOTE algorithm [20] have no statistically significant differences from the base C1 by this metric. When comparing the classifiers developed in the proposed study using the same metric with classifiers developed using the SMOTE algorithm [20], it was possible to reveal only the superiority of the C3 classifier (at $h = 2$) with L_0.

According to Table A5, all classifiers that solve the CIP in one way or another outperformed the base classifier C1 as measured by the metric $Recall$. When comparing the values of the same metric of the classifiers developed in the proposed study with classifiers developed using the SMOTE algorithm [20], it was also possible to reveal only the superiority of the C3 classifier (at $h = 2$) with L_0.

According to Table A6, all classifiers that solve the CIP in one way or another lose to the base classifier C1 in training time (which was expected). When comparing the training time of the classifiers developed in the proposed study with the classifiers developed using the SMOTE algorithm [20], the superiority of the cost-sensitive classifiers is observed. At the same time, the balanced classifier C7 (at $h = 28$) using the SMOTE algorithm [20] outperformed the balanced classifier C1 using the SMOTE algorithm [20] in training time (possibly due to better separability of classes).

According to Table A7, all classifiers that solve the CIP in one way or another lose to the base classifier C1 in terms of testing time (which was expected). When comparing the testing time of the classifiers developed in the proposed study with the same time of the balanced classifier C1 using the SMOTE algorithm [20], the superiority of the cost-sensitive

classifiers is observed, with the exception of the classifier C3 (at $h = 2$) with L_0 that lost. At the same time, the balanced classifier C7 (at $h = 28$) using the SMOTE algorithm [20] outperformed all classifiers in testing time except the base classifier C1, which it lost to.

In general, the following should be noted. All classifiers that solve the CIP in one way or another and are developed on the basis of modified datasets are superior to the base classifier C1 by the metrics $MacroF_1 - score$ and $Recall$.

The best cost-sensitive classifiers developed in the proposed study outperform the base cost-sensitive C1 in terms of metrics $MacroF_1 - score$ and $Recall$; however, they do not have statistical differences among themselves in these metrics. In terms of training time and classifier testing time, all these classifiers are statistically different. Therefore, when choosing a classifier, one can focus, for example, on the minimal time required to train a classifier. Thus, classifier C7 (at $h = 26$) with L_2 provides minimal time of training.

In general, limitations on the applicability of the proposed approach may be due to the following reasons. First, we may experience limitations caused by the computational complexity of developing SVM classifiers using standard implementations of the SVM algorithm. However, this problem can be solved using modern SVM solvers [87–89]. Secondly, certain problems may be caused by the very nature of the used dataset. Data should be subjected to exploratory analysis and, if possible, cleared of omissions, outliers and similar defects. In the case of data of very poor quality, their preprocessing can lead to an even greater imbalance of classes, up to the loss of a significant part of the patterns belonging to minority classes. In this regard, a qualitative solution to the imbalance problem using CSL algorithms or, for example, oversampling algorithms will be questionable. In addition, the nature of the used dataset may be such that the data of different classes in it will initially be poorly separable, for example, due to the poor separability of patterns determined by blood protein markers according to their membership in different classes. In this regard, both attempts to develop SVM classifiers based on the original dataset and attempts to generate new features, for example, based on approximation entropy and the UMAP algorithm, will be unsuccessful: new features will not improve the quality of dividing data patterns into different classes. Third, it should be noted that additional experiments are needed when determining penalties for misclassifying patterns of different classes in the case of CSL or when determining the class ratio that should be achieved after restoring the balance, for example, using oversampling algorithms.

In the proposed study, we used a dataset, the properties of which were previously studied in detail in [13,14,20], and in [20] it was noted that there was no strong correlation both between the features of the original dataset and when introducing those new features approved for use.

In the proposed study, we used a dataset, the properties of which were previously studied in detail in [13,14,20], including in [20], where it was noted that there was no strong correlation both between the features of the original dataset and when introducing those approved for use new features.

It should be noted that the combination of the CSL principles and the approach proposed in [20] to the creation of datasets by forming new features using various technologies with their acquisition and application as a new dataset or adding to the original dataset may be considered appropriate. In this case, varying the values of the parameters *n_neighbors* and *min_dist*, as well as working with several LFs in the UMAP algorithm, ultimately made it possible to obtain qualitatively better cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$.

6. Conclusions

In this research, we investigated a previously suggested approach [20] for the diagnosis of cancer using blood protein markers through creation of the SVM classifiers on the basis of datasets with a variety of features of different nature. These features may correspond to blood protein markers or be constructed using methods for calculating entropy and fractal dimensions, as well as using the UMAP algorithm. These medical datasets are imbalanced.

To overcome the class imbalance problem, the concept of cost-sensitive learning was implemented, the use of which allowed the best developed SVM classifiers to outperform the base SVM classifier in data classification quality and the best SVM classifiers developed on the basis of the oversampling strategy, not only in data classification quality but also in the time spent on their development. The most successful in terms of maximizing the mean value of the metric $MacroF_1 - score$ are the following cost-sensitive SVM classifiers, listed in descending order of successfulness: C3, C7, C8, C11 and C10. The UMAP algorithm was applied to create new features in datasets used to develop classifiers C3, C7 and C11. The approximate entropy was applied to create a new feature in datasets used to develop classifiers C7, C8 and C10. The Katz and Higuchi fractal dimensions were applied to create new features in datasets used to develop classifiers C10 and C11. Each time, new features supplemented the original dataset. We showed that to create additional features on the basis of the UMAP algorithm, it is advisable to use the LFs L_1, L_2 and L_4, defined explicitly by formulas (13), (14) and (10). The use of an implicitly defined LF, which we called L_0, applied in the library implementation [80], is complicated because of the impossibility of explicitly estimating the values for the LF, although it cannot be considered unambiguously inexpedient.

The purpose of further research is to explore ways to improve data classification quality by forming new features on the basis of various dimensionality reduction algorithms, such as UMAP [53], t-SNE [77], TriMAP (Triplet Manifold Approximation) [96] and PaCMAP (Pairwise Controlled Manifold Approximation) [97], for which both the initialization of the initial embedding of patterns into low-dimensional space and the optimization of the embedding of patterns into low-dimensional space are performed in different manners. We also plan to implement a simultaneous combination of CSL and oversampling technologies with the selection of the best combinations of penalties and the best class proportions when synthesizing new data patterns.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in [54].

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A

Table A1. Names of concepts and their abbreviations.

Name of Concept	Abbreviation
ADASYN	ADaptive SYNthetic sampling approach
AE	Approximate Entropy
AUC	Area Under Curve
CIP	Class Imbalance Problem
COSMIC	Catalog of Somatic Mutations in Cancer
CSL	Cost-Sensitivity Learning
BPM	Blood Protein Marker
FGD	Full Gradient Descent
GT	Gene Test
HC	Hjorth Complexity
HFD	Higuchi Fractal Dimension
HM	Hjorth Mobility
KFD	Katz Fractal Dimension
DL	Deep Learning
DM	Data Mining
kNN	k-Nearest Neighbors
LF	Loss Function

Table A1. *Cont.*

Name of Concept	Abbreviation
LR	Logistic Regression
ML	Machine Learning
OD	Oncological Disease
OvO	One-vs-One strategy
OvR	One-vs-Rest strategy
PaCMAP	Pairwise Controlled Manifold Approximation
PFD	Petrosian Fractal Dimension
PT	Protein Test
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Over-Sampling Technique
SVDE	Singular Value Decomposition Entropy
SVM	Support Vector Machine
RBF	Radial Basis Function
RF	Random Forest
RQ	Research Question
t-SNE	T-Distributed Stochastic Neighbor Embedding
TriMAP	Triplet Manifold Approximation
SE	Sample Entropy
SD	Standard Deviation
SPE	SPectral Entropy
UMAP	Uniform Manifold Approximation and Projection

Table A2. Datasets and the composition of their features.

Dataset Name	The Composition of Features
C1	FOD *
C2	FUMAP **
C3	FOD, FUMAP
C4	FUMAP, FAE ***
C5	FUMAP, FHKFR ****
C6	FUMAP, FAE, FHKFR
C7	FOD, FUMAP, FAE
C8	FOD, FAE
C9	FOD, FHKFR
C10	FOD, FAE, FHKFR
C11	FOD, FUMAP, FHKFR
C12	FOD FUMAP, FAE, FHKFR

* FOD are the Features of the Original Dataset; ** FUMAP are the Features on the basis of the UMAP algorithm; *** FAE is the Feature on the basis of Approximate Entropy; **** FHKFR are the Features on the basis of Higuchi and Katz Fractal Dimensions.

Table A3. Basic algorithms and description of their changeable parameters.

Algorithm	Parameter	Parameter Value or Range with Step of Change
SVM	C is the regularization parameter (C in the scikit-learn library of Python)	[0.4, 2] with a step of 0.1
	σ ($\sigma > 0$) is the parameter of the RBF kernel (*gamma* in the scikit-learn library of Python)	[0.4, 2] with a step of 0.1
UMAP	k in the number of nearest neighbors that are found for each pattern in the high-dimensional space (*n_neighbors* in the software library [80])	[10, 20] with a step of 5
	d_{min} is the threshold distance ($d_{min} \in (0, 1]$) that influences the density of clusters created in the low-dimensional space (*min_dist* in the software library [80])	[0.1, 0.3] with a step of 0.1
	h is the dimension of the low-dimensional space (*n_components* in the software library [80])	[2, 38] with a step of 2
	metric is the distance metric in the software library [80]	Euclidean
	random_state is the parameter responsible for initialization of UMAP algorithm and reproducibility of results in the software library [80]	42

Appendix B

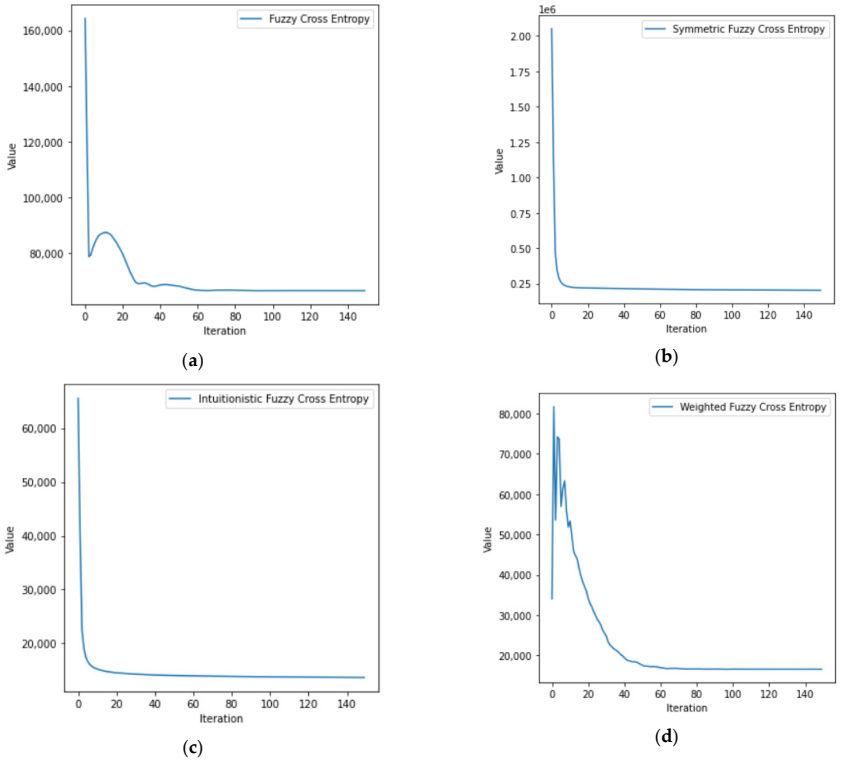

Figure A1. Graphical dependencies for LFs obtained by constructing embeddings of the original 39-dimensional three-class dataset in the two-dimensional space on the basis of the UMAP algorithm with various LFs with parameter values *n_neighbors* and *min_dist*, ensuring the creation of the best cost-sensitive classifiers C3 (at $h = 2$) in the terms of maximizing the mean value of the metric $MacroF_1 - score$) presented in Figure 3. (**a**) L_1: LF based on fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$: mean = 0.918; SD = 0.049); (**b**) L_2: LF based on symmetric fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$: mean = 0.916; SD = 0.047); (**c**) L_3: LF based on intuitionistic fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.3; $MacroF_1 - score$: mean = 0.914; SD = 0.055); (**d**) L_4: LF based on weighted fuzzy cross-entropy with FGD (*n_neighbors* = 20; *min_dist* = 0.2; $MacroF_1 - score$: mean = 0.915; SD = 0.044).

Appendix C

Table A4. Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis of the metric $MacroF_1 - score$.

Classifier	**Base C1**		**Balanced C1 Using SMOTE Algorithm [20]**		**Balanced C7 (at h = 28) Using SMOTE Algorithm [20]**	
Classifier	**Sign**	***p*-Value**	**Sign**	***p*-Value**	**Sign**	***p*-Value**
base cost-sensitive C1	=	0.071	=	0.950	=	0.761
C3 (at $h = 2$) with L_0	+	0.002	=	0.200	+	0.034
C7 (at $h = 28$) with L_1	+	0.005	=	0.214	=	0.594
C7 (at $h = 26$) with L_2	+	0.004	=	0.252	=	0.462
C7 (at $h = 28$) with L_2	+	0.004	=	0.147	=	0.795
C3 (at $h = 6$) with L_4	+	0.008	=	0.178	=	0.795
balanced C1 using SMOTE algorithm [20]	=	0.125	Not	Not	=	0.795
balanced C7 (at $h = 28$) using SMOTE algorithm [20]	+	0.021	=	0.795	Not	Not

Table A5. Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis of the metric *Recall*.

Classifier	**Base C1**		**Balanced C1 Using SMOTE Algorithm [20]**		**Balanced C7 (at h = 28) Using SMOTE Algorithm [20]**	
Classifier	**Sign**	***p*-Value**	**Sign**	***p*-Value**	**Sign**	***p*-Value**
base cost-sensitive C1	+	0.001	=	0.740	=	0.102
C3 (at $h = 2$) with L_0	+	4.571×10^{-5}	=	0.153	+	0.039
C7 (at $h = 28$) with L_1	+	2.194×10^{-4}	=	0.331	=	0.810
C7 (at $h = 26$) with L_2	+	3.405×10^{-4}	=	0.365	=	0.576
C7 (at $h = 28$) with L_2	+	2.682×10^{-4}	=	0.207	=	0.420
C3 (at $h = 6$) with L_4	+	2.309×10^{-4}	=	0.283	=	0.909
balanced C1 using SMOTE algorithm [20]	+	0.004	Not	Not	=	0.724
balanced C7 (at $h = 28$) using SMOTE algorithm [20]	+	2.043×10^{-4}	=	0.724	Not	Not

Table A6. Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis time of training.

Classifier	**Base C1**		**Balanced C1 Using SMOTE Algorithm [20]**		**Balanced C7 (at h = 28) Using SMOTE Algorithm [20]**	
Classifier	**Sign**	***p*-Value**	**Sign**	***p*-Value**	**Sign**	***p*-Value**
base cost-sensitive C1	−	1.863×10^{-9}	+	1.863×10^{-9}	+	1.863×10^{-9}
C3 (at $h = 2$) with L_0	−	1.863×10^{-9}	+	1.863×10^{-9}	+	1.863×10^{-9}
C7 (at $h = 28$) with L_1	−	1.863×10^{-9}	+	1.863×10^{-9}	+	1.863×10^{-9}
C7 (at $h = 26$) with L_2	−	3.725×10^{-9}	+	1.863×10^{-9}	+	1.863×10^{-9}
C7 (at $h = 28$) with L_2	−	3.725×10^{-9}	+	1.863×10^{-9}	+	1.863×10^{-9}
C3 (at $h = 6$) with L_4	−	5.588×10^{-9}	+	1.863×10^{-9}	+	3.239×10^{-6}
balanced C1 using SMOTE algorithm [20]	−	1.863×10^{-9}	Not	Not	−	1.863×10^{-9}
balanced C7 (at $h = 28$) using SMOTE algorithm [20]	−	1.863×10^{-9}	+	1.863×10^{-9}	Not	Not

Table A7. Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis time of testing.

Classifier	**Base C1**		**Balanced C1 Using SMOTE Algorithm [20]**		**Balanced C7 (at h = 28) Using SMOTE Algorithm [20]**	
Classifier	**Sign**	***p*-Value**	**Sign**	***p*-Value**	**Sign**	***p*-Value**
base cost-sensitive C1	−	2.608×10^{-8}	+	0.013	−	1.863×10^{-8}
C3 (at $h = 2$) with L_0	−	1.106×10^{-4}	−	0.002	−	0.003
C7 (at $h = 28$) with L_1	−	2.608×10^{-8}	+	0.0128	−	1.863×10^{-8}
C7 (at $h = 26$) with L_2	−	3.725×10^{-9}	+	1.863×10^{-8}	−	2.608×10^{-8}
C7 (at $h = 28$) with L_2	−	3.725×10^{-9}	+	1.863×10^{-8}	−	2.608×10^{-8}
C3 (at $h = 6$) with L_4	−	2.608×10^{-8}	+	0.013	−	1.863×10^{-8}
balanced C1 using SMOTE algorithm [20]	−	1.863×10^{-9}	Not	Not	−	1.863×10^{-9}
balanced C7 (at $h = 28$) using SMOTE algorithm [20]	−	0.002	+	1.863×10^{-9}	Not	Not

References

1. 2021 Global Health Care Outlook. Available online: https://www2.deloitte.com/cn/en/pages/life-sciences-and-healthcare/articles/2021-global-healthcare-outlook.html (accessed on 4 January 2024).
2. Slim, K.; Selvy, M.; Veziant, J. Conceptual innovation: 4P Medicine and 4P surgery. *J. Visc. Surg.* **2021**, *158*, S12–S17. [CrossRef] [PubMed]
3. Brar, A.; Zhu, A.; Baciu, C.; Sharma, D.; Xu, W.; Orchanian-Cheff, A.; Wang, B.; Reimand, J.; Grant, R.; Bhat, M. Development of diagnostic and prognostic molecular biomarkers in hepatocellular carcinoma using machine learning: A systematic review. *Liver Cancer Int.* **2022**, *3*, 141–161. [CrossRef]
4. Li, P.; Xu, S.; Han, Y.; He, H.; Liu, Z. Machine learning-empowered cis-diol metabolic fingerprinting enables precise diagnosis of primary liver cancer. *Chem. Sci.* **2023**, *14*, 2553–2561. [CrossRef]
5. Ma, J.; Bo, Z.; Zhao, Z.; Yang, J.; Yang, Y.; Li, H.; Yang, Y.; Wang, J.; Su, Q.; Wang, J.; et al. Machine Learning to Predict the Response to Lenvatinib Combined with Transarterial Chemoembolization for Unresectable Hepatocellular Carcinoma. *Cancers* **2023**, *15*, 625. [CrossRef] [PubMed]
6. Fu, Y.; Si, A.; Wei, X.; Lin, X.; Ma, Y.; Qiu, H.; Guo, Z.; Pan, Y.; Zhang, Y.; Kong, X.; et al. Combining a machine-learning derived 4-lncRNA signature with AFP and TNM stages in predicting early recurrence of hepatocellular carcinoma. *BMC Genom.* **2023**, *24*, 89. [CrossRef] [PubMed]
7. Iseke, S.; Zeevi, T.; Kucukkaya, A.S.; Raju, R.; Gross, M.; Haider, S.P.; Petukhova-Greenstein, A.; Kuhn, T.N.; Lin, M.; Nowak, M.; et al. Machine Learning Models for Prediction of Posttreatment Recurrence in Early-Stage Hepatocellular Carcinoma Using Pretreatment Clinical and MRI Features: A Proof-of-Concept Study. *AJR Am. J. Roentgenol.* **2023**, *220*, 245–255. [CrossRef] [PubMed]
8. Chaudhary, K.; Poirion, O.B.; Lu, L.; Garmire, L.X. Deep learning-based multi-omics integration robustly predicts survival in liver cancer. *Clin. Cancer Res.* **2018**, *24*, 1248–1259. [CrossRef] [PubMed]
9. Lv, J.; Wang, J.; Shang, X.; Liu, F.; Guo, S. Survival prediction in patients with colon adenocarcinoma via multi-omics data integration using a deep learning algorithm. *Biosci. Rep.* **2020**, *40*, BSR20201482. [CrossRef] [PubMed]
10. Lee, T.Y.; Huang, K.Y.; Chuang, C.H.; Lee, C.Y.; Chang, T.H. Incorporating deep learning and multi-omics autoencoding for analysis of lung adenocarcinoma prognostication. *Comput. Biol.* **2020**, *87*, 107277. [CrossRef]
11. Nam, D.; Chapiro, J.; Paradis, V.; Seraphin, T.P.; Kather, J.N. Artificial intelligence in liver diseases: Improving diagnostics, prognostics and response prediction. *JHEP Rep.* **2022**, *4*, 100443. [CrossRef]
12. Kawka, M.; Dawidziuk, A.; Jiao, L.R.; Gall, T.M.H. Artificial intelligence in the detection, characterisation and prediction of hepatocellular carcinoma: A narrative review. *Transl. Gastroenterol. Hepatol.* **2022**, *7*, 41. [CrossRef]
13. Cohen, J.D.; Li, L.; Wang, Y.; Thoburn, C.; Afsari, B.; Danilova, L.; Douville, C.; Javed, A.A.; Wong, F.; Mattox, A.; et al. Detection and localization of surgically resectable cancers with a multi-analyte blood test. *Science* **2018**, *359*, 926–930. [CrossRef]
14. Song, C.; Li, X. Cost-Sensitive KNN Algorithm for Cancer Prediction Based on Entropy Analysis. *Entropy* **2022**, *24*, 253. [CrossRef] [PubMed]
15. Huang, S.; Cai, N.; Pacheco, P.P.; Narrandes, S.; Wang, Y.; Xu, W. Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. *Cancer Genom. Proteom.* **2018**, *15*, 41–51. [CrossRef]
16. Toth, R.; Schiffmann, H.; Hube-Magg, C.; Büscheck, F.; Höflmayer, D.; Weidemann, S.; Lebok, P.; Fraune, C.; Minner, S.; Schlomm, T.; et al. Random forest-based modelling to detect biomarkers for prostate cancer progression. *Clin. Epigenet.* **2019**, *11*, 148. [CrossRef]
17. Pan, L.Y.; Liu, G.J.; Lin, F.Q.; Zhong, S.L.; Xia, H.M.; Sun, X.; Liang, H.Y. Machine Learning Applications for Prediction of Relapse in Childhood Acute Lymphoblastic Leukemia. *Sci. Rep.* **2017**, *7*, 7402. [CrossRef]
18. Abreu, P.H.; Santos, M.S.; Abreu, M.H.; Andrade, B.; Silva, D.C. Predicting Breast Cancer Recurrence using Machine Learning Techniques: A Systematic Review. *ACM Comput. Surv.* **2017**, *49*, 52. [CrossRef]
19. Savareh, B.A.; Aghdaie, H.A.; Behmanesh, A.; Bashiri, A.; Sadeghi, A.; Zali, M.; Shams, R. A machine learning approach identified a diagnostic model for pancreatic cancer through using circulating microRNA signatures. *Pancreatology* **2020**, *20*, 1195–1204. [CrossRef]
20. Demidova, L.A. A Novel Approach to Decision-Making on Diagnosing Oncological Diseases Using Machine Learning Classifiers Based on Datasets Combining Known and/or New Generated Features of a Different Nature. *Mathematics* **2023**, *11*, 792. [CrossRef]
21. Li, G.; Hu, J.; Hu, G. Biomarker Studies in Early Detection and Prognosis of Breast Cancer. *Adv. Exp. Med. Biol.* **2017**, *1026*, 27–39. [CrossRef]
22. Loke, S.Y.; Lee, A.S.G. The future of blood-based biomarkers for the early detection of breast cancer. *Eur. J. Cancer.* **2018**, *92*, 54–68. [CrossRef] [PubMed]
23. Killock, D. CancerSEEK and destroy—A blood test for early cancer detection. *Nat. Rev. Clin. Oncol.* **2018**, *15*, 133. [CrossRef] [PubMed]
24. Kalinich, M.; Haber, D.A. Cancer detection: Seeking signals in blood. *Science* **2018**, *359*, 866–867. [CrossRef]
25. Mansur, A.; Vrionis, A.; Charles, J.P.; Hancel, K.; Panagides, J.C.; Moloudi, F.; Iqbal, S.; Daye, D. The Role of Artificial Intelligence in the Detection and Implementation of Biomarkers for Hepatocellular Carcinoma: Outlook and Opportunities. *Cancers* **2023**, *15*, 2928. [CrossRef] [PubMed]

26. Hao, Y.; Jing, X.Y.; Sun, Q. Joint learning sample similarity and correlation representation for cancer survival prediction. *BMC Bioinform.* **2022**, *23*, 553. [CrossRef]
27. Núñez, C. Blood-based protein biomarkers in breast cancer. *Clin. Chim. Acta.* **2019**, *490*, 113–127. [CrossRef]
28. Du, Z.; Liu, X.; Wei, X.; Luo, H.; Li, P.; Shi, M.; Guo, B.; Cui, Y.; Su, Z.; Zeng, J.; et al. Quantitative proteomics identifes a plasma multi protein model for detection of hepatocellular carcinoma. *Sci. Rep.* **2020**, *10*, 15552. [CrossRef]
29. Siers, M.J.; Md Zahidul, I. Class Imbalance and Cost-Sensitive Decision Trees: A Unified Survey Based on a Core Similarity. *ACM Trans. Knowl. Discov. Data.* **2020**, *15*, 4. [CrossRef]
30. Rekha, G.; Tyagi, A.K.; Reddy, V.K. A Wide Scale Classification of Class Imbalance Problem and its Solutions: A Systematic Literature Review. *J. Comput. Sci.* **2019**, *15*, 886–929. [CrossRef]
31. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
32. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Advances in Intelligent Computing*; Huang, D.S., Zhang, X.P., Huang, G.B., Eds.; ICIC 2005. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3644, pp. 878–887. [CrossRef]
33. He, H.; Bay, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328. [CrossRef]
34. Swana, E.F.; Doorsamy, W.; Bokoro, P. Tomek Link and SMOTE Approaches for Machine Fault Classification with an Imbalanced Dataset. *Sensors* **2022**, *22*, 3246. [CrossRef]
35. Tomek, I. Two modifications of CNN. *IEEE Trans. Syst. Man Cybern.* **1976**, *6*, 769–772. [CrossRef]
36. Ling, C.X.; Sheng, V.S. Cost-Sensitive Learning. In *Encyclopedia of Machine Learning*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2011. [CrossRef]
37. Xu, R.; Wang, J.; Zhu, Q.; Zou, C.; Wei, Z.; Wang, H.; Ding, Z.; Meng, M.; Wei, H.; Xia, S.; et al. Integrated models of blood protein and metabolite enhance the diagnostic accuracy for Non-Small Cell Lung Cancer. *Biomark. Res.* **2023**, *11*, 71. [CrossRef]
38. Luan, Y.; Zhong, G.; Li, S.; Wu, W.; Liu, X.; Zhu, D.; Feng, Y.; Zhang, Y.; Duan, C.; Mao, M. A panel of seven protein tumour markers for effective and affordable multi-cancer early detection by artificial intelligence: A large-scale and multicentre case-control study. *EClinicalMedicine* **2023**, *61*, 102041. [CrossRef]
39. Demidova, L.A. Two-stage hybrid data classifiers based on SVM and kNN algorithms. *Symmetry* **2021**, *13*, 615. [CrossRef]
40. Zanin, M.; Zunino, L.; Rosso, O.A.; Papo, D. Permutation Entropy and Its Main Biomedical and Econophysics Applications: A Review. *Entropy* **2012**, *14*, 1553–1577. [CrossRef]
41. Zhang, A.; Yang, B.; Huang, L. Feature Extraction of EEG Signals Using Power Spectral Entropy. In Proceedings of the International Conference on BioMedical Engineering and Informatics, Sanya, China, 27–30 May 2008; Volume 2, pp. 435–439. [CrossRef]
42. Weng, X.; Perry, A.; Maroun, M.; Vuong, L.T. Singular Value Decomposition and Entropy Dimension of Fractals. *arXiv* **2022**, arXiv:2211.12338. [CrossRef]
43. Pincus, S.M. Approximate entropy as a measure of system complexity. *Proc. Natl. Acad. Sci. USA* **1991**, *88*, 2297–2301. [CrossRef]
44. Pincus, S.M.; Gladstone, I.M.; Ehrenkranz, R.A. A regularity statistic for medical data analysis. *J. Clin. Monit. Comput.* **1991**, *7*, 335–345. [CrossRef]
45. Delgado-Bonal, A.; Marshak, A. Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy* **2019**, *21*, 541. [CrossRef] [PubMed]
46. Hjorth, B. EEG Analysis Based on Time Domain Properties. *Electroencephalogr. Clin. Neurophysiol.* **1970**, *29*, 306–310. [CrossRef]
47. Galvão, F.; Alarcão, S.M.; Fonseca, M.J. Predicting Exact Valence and Arousal Values from EEG. *Sensors* **2021**, *21*, 3414. [CrossRef]
48. Shi, C.-T. Signal Pattern Recognition Based on Fractal Features and Machine Learning. *Appl. Sci.* **2018**, *8*, 1327. [CrossRef]
49. Bykova, M.O.; Balandin, V.A. Methodological features of the analysis of the fractal dimension of the heart rate. *Russ. Technol. J.* **2023**, *11*, 58–71. [CrossRef]
50. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
51. Becht, E.; McInnes, L.; Healy, J.; Dutertre, C.A.; Kwok, I.W.H.; Ng, L.G.; Ginhoux, F.; Newell, E.W. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* **2019**, *37*, 38–44. [CrossRef] [PubMed]
52. Dorrity, M.W.; Saunders, L.M.; Queitsch, C.; Fields, S.; Trapnell, C. Dimensionality reduction by UMAP to visualize physical and genetic interactions. *Nat. Commun.* **2020**, *11*, 1537. [CrossRef]
53. Demidova, L.A.; Gorchakov, A.V. Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm. *J. Imaging* **2022**, *8*, 113. [CrossRef]
54. COSMIC | Catalogue of Somatic Mutations in Cancer. Available online: https://cancer.sanger.ac.uk/cosmic (accessed on 4 January 2023).
55. Thai-Nghe, N.; Gantner, Z.; Schmidt-Thieme, L. Cost-sensitive learning methods for imbalanced data. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
56. Cao, P.; Zhao, D.; Zaiane, O. An optimized cost-sensitive SVM for imbalanced data learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 280–292.

57. Mienye, I.D.; Sun, Y. Performance analysis of cost-sensitive learning methods with application to imbalanced medical data. *Inform. Med. Unlocked* **2021**, *25*, 100690. [CrossRef]
58. Tsai, H.-H.; Yang, T.-W.; Wong, W.-M.; Chou, C.-F. A Hybrid Approach for Binary Classification of Imbalanced Data. *arXiv* **2022**, arXiv:2207.02738. [CrossRef]
59. Freitas, A.; Brazdil, P.; Costa-Pereira, A. Cost-Sensitive Learning in Medicine. In *Data Mining and Medical Knowledge Management: Cases and Applications*; Berka, P., Rauch, J., Zighed, D.A., Eds.; IGI Global: Hershey, PA, USA, 2009; pp. 57–75. [CrossRef]
60. Gupta, R.; Kleinjans, J.; Caiment, F. Identifying novel transcript biomarkers for hepatocellular carcinoma (HCC) using RNA-Seq datasets and machine learning. *BMC Cancer* **2021**, *21*, 962. [CrossRef] [PubMed]
61. Lee, T.; Rawding, P.A.; Bu, J.; Hyun, S.; Rou, W.; Jeon, H.; Kim, S.; Lee, B.; Kubiatowicz, L.J.; Kim, D.; et al. Machine-Learning-Based Clinical Biomarker Using Cell-Free DNA for Hepatocellular Carcinoma (HCC). *Cancers* **2022**, *14*, 2061. [CrossRef] [PubMed]
62. Sato, M.; Tateishi, R.; Moriyama, M.; Fukumoto, T.; Yamada, T.; Nakagomi, R.; Kinoshita, M.N.; Nakatsuka, T.; Minami, T.; Uchino, K. Machine Learning–Based Personalized Prediction of Hepatocellular Carcinoma Recurrence After Radiofrequency Ablation. *Gastro Hep. Adv.* **2022**, *1*, 29–37. [CrossRef]
63. An, C.; Yang, H.; Yu, X.; Han, Z.Y.; Cheng, Z.; Liu, F.; Dou, J.; Li, B.; Li, Y.; Li, Y.; et al. A Machine Learning Model Based on Health Records for Predicting Recurrence After Microwave Ablation of Hepatocellular Carcinoma. *J. Hepatocell. Carcinoma* **2022**, *9*, 671–684. [CrossRef] [PubMed]
64. Ding, W.; Wang, Z.; Liu, F.Y.; Cheng, Z.G.; Yu, X.; Han, Z.; Zhong, H.; Yu, J.; Liang, P. A Hybrid Machine Learning Model Based on Semantic Information Can Optimize Treatment Decision for Naïve Single 3-5-cm HCC Patients. *Liver Cancer* **2022**, *11*, 256–267. [CrossRef]
65. Hsu, P.Y.; Liang, P.C.; Chang, W.T.; Lu, M.Y.; Wang, W.H.; Chuang, S.C.; Wei, Y.J.; Jang, T.Y.; Yeh, M.L.; Huang, C.I.; et al. Artificial intelligence based on serum biomarkers predicts the efficacy of lenvatinib for unresectable hepatocellular carcinoma. *Am. J. Cancer Res.* **2022**, *12*, 5576–5588.
66. Ge, S.; Xu, C.R.; Li, Y.M.; Zhang, Y.L.; Li, N.; Wang, F.T.; Ding, L.; Niu, J. Identification of the Diagnostic Biomarker VIPR1 in Hepatocellular Carcinoma Based on Machine Learning Algorithm. *J. Oncol.* **2022**, *2022*, 2469592. [CrossRef]
67. Xing, W.; Bei, Y. Medical Health Big Data Classification Based on KNN Classification Algorithm. *IEEE Access* **2020**, *8*, 28808–28819. [CrossRef]
68. Mohanty, S.; Mishra, A.; Saxena, A. Medical Data Analysis Using Machine Learning with KNN. In *International Conference on Innovative Computing and Communications*; Gupta, D., Khanna, A., Bhattacharyya, S., Hassanien, A.E., Anand, S., Jaiswal, A., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2020; Volume 1166. [CrossRef]
69. Chapelle, O.; Vapnik, V.; Bousquet, O.; Mukherjee, S. Choosing multiple parameters for support vector machines. *Mach. Learn.* **2002**, *46*, 131–159. [CrossRef]
70. Yu, W.; Liu, T.; Valdez, R.; Gwinn, M.; Khoury, M.J. Application of support vector machine modeling for prediction of common diseases: The case of diabetes and pre-diabetes. *BMC Med. Inform. Decis. Mak.* **2010**, *10*, 16. [CrossRef]
71. Schober, P.; Vetter, T.R. Logistic Regression in Medical Research. *Anesth. Analg.* **2021**, *132*, 365–366. [CrossRef] [PubMed]
72. Dai, B.; Chen, R.-C.; Zhu, S.-Z.; Zhang, W.-W. Using Random Forest Algorithm for Breast Cancer Diagnosis. In Proceedings of the 2018 International Symposium on Computer, Consumer and Control (IS3C), Taichung, Taiwan, 6–8 December 2018; pp. 449–452. [CrossRef]
73. Acharjee, A.; Larkman, J.; Xu, Y.; Cardoso, V.R.; Gkoutos, G.V. A random forest based biomarker discovery and power analysis framework for diagnostics research. *BMC Med. Genom.* **2020**, *13*, 178. [CrossRef] [PubMed]
74. Cheng, S.; Liu, B.; Ting, T.O.; Qin, Q.; Shi, Y.; Huang, K. Survey on data science with population-based algorithms. *Big Data Anal.* **2016**, *1*, 3. [CrossRef]
75. Liu, J.-Y.; Jia, B.-B. Combining One-vs-One Decomposition and Instance-Based Learning for Multi-Class Classification. *IEEE Access* **2020**, *8*, 197499–197507. [CrossRef]
76. Grandini, M.; Bagli, E.; Visani, G. Metrics for Multi-class Classification: An Overview. *arXiv* **2020**, arXiv:2008.05756.
77. Van der Maaten, L.; Hinton, G.E. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
78. Dong, W.; Moses, C.; Li, K. Efficient k-nearest neighbor graph construction for generic similarity measures. In Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1 April 2011; pp. 577–586.
79. Damrich, S.; Hamprecht, F.A. On UMAP's true loss function. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12.
80. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. Available online: https://umap-learn.readthedocs.io/en/latest/_modules/umap/umap_.html (accessed on 4 January 2024).
81. Bottou, L.; Chapelle, O.; DeCoste, D.; Weston, J. *Support Vector Machine Solvers Large-Scale Kernel Machines*; MIT Press: Cambridge, MA, USA, 2007; pp. 1–27.
82. Tsang, I.W.; Kwok, J.T.; Cheung, P.-M. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *J. Mach. Learn. Res.* **2005**, *6*, 363–392.
83. umap. Available online: https://github.com/lmcinnes/umap/issues/8 (accessed on 4 January 2024).
84. Tomčala, J. New Fast ApEn and SampEn Entropy Algorithms Implementation and Their Application to Supercomputer Power Consumption. *Entropy* **2020**, *22*, 863. [CrossRef]
85. Batu, T.; Dasgupta, S.; Kumar, R.; Rubinfeld, R. The complexity of approximating the entropy. In Proceedings of the 17th IEEE Annual Conference on Computational Complexity, Montreal, QC, Canada, 21–24 May 2002; pp. 17–26. [CrossRef]

86. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods—Support Vector Learning*; Schölkopf, B., Burges, C.J., Smola, A.J., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 185–208.
87. Collobert, R.; Bengio, S.; Bengio, Y. A parallel mixture of SVMs for very large scale problems. *Neural Comput.* **2002**, *14*, 1105–1114. [CrossRef]
88. Shalev-Shwartz, S.; Singer, Y.; Srebro, N. Pegasos: Primal estimated sub-gradient solver for SVM. *Math. Program.* **2011**, *127*, 3–30. [CrossRef]
89. Gentinetta, G.; Thomsen, A.; Sutter, D.; Woerner, S. The complexity of quantum support vector machines. *Quantum* **2024**, *8*, 1225. [CrossRef]
90. Prusty, S.; Patnaik, S.; Dash, S.K. SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. *Front. Nanotechnol.* **2022**, *4*, 972421. [CrossRef]
91. Slamet, W.; Herlambang, B.; Samudi, S. Stratified K-fold cross validation optimization on machine learning for prediction. *Sink. J. Dan Penelit. Tek. Inform.* **2022**, *7*, 2407–2414. [CrossRef]
92. umap-losses. Available online: https://github.com/worldbeater/umap-losses (accessed on 4 January 2024).
93. Numba: A High Performance Python Compiler. Available online: https://numba.pydata.org/ (accessed on 4 January 2024).
94. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
95. Gorchakov, A.V.; Demidova, L.A.; Sovietov, P.N. Analysis of Program Representations Based on Abstract Syntax Trees and Higher-Order Markov Chains for Source Code Classification Task. *Future Internet* **2023**, *15*, 314. [CrossRef]
96. Amid, E.; Warmuth, M.K. TriMap: Large-scale Dimensionality Reduction Using Triplets. *arXiv* **2019**, arXiv:1910.00204v2. [CrossRef]
97. Wang, Y.; Huang, H.; Rudin, C.; Shaposhnik, Y. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *J. Mach. Learn. Res.* **2021**, *22*, 9129–9201.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.