algorithms

MDPI

Article

Hyperparameter Black-Box Optimization to Improve the Automatic Classification of Support Tickets

Renato Bruni [1,*], Gianpiero Bianchi [2] and Pasquale Papa [2]

[1] Department of Computer Control and Management Engineering, "Sapienza" University of Rome, 00100 Rome, Italy
[2] Directorate for Data Collection, Italian National Institute of Statistics "Istat", 00100 Rome, Italy
* Correspondence: bruni@diag.uniroma1.it

Abstract: User requests to a customer service, also known as tickets, are essentially short texts in natural language. They should be grouped by topic to be answered efficiently. The effectiveness increases if this semantic categorization becomes automatic. We pursue this goal by using text mining to extract the features from the tickets, and classification to perform the categorization. This is however a difficult multi-class problem, and the classification algorithm needs a suitable hyperparameter configuration to produce a practically useful categorization. As recently highlighted by several researchers, the selection of these hyperparameters is often the crucial aspect. Therefore, we propose to view the hyperparameter choice as a higher-level optimization problem where the hyperparameters are the decision variables and the objective is the predictive performance of the classifier. However, an explicit analytical model of this problem cannot be defined. Therefore, we propose to solve it as a black-box model by means of derivative-free optimization techniques. We conduct experiments on a relevant application: the categorization of the requests received by the Contact Center of the Italian National Statistics Institute (Istat). Results show that the proposed approach is able to effectively categorize the requests, and that its performance is increased by the proposed hyperparameter optimization.

Keywords: machine learning; black-box optimization; Auto ML; classification; customer support; statistical surveys

Citation: Bruni, R.; Bianchi, G.; Papa, P. Hyperparameter Black-Box Optimization to Improve the Automatic Classification of Support Tickets. *Algorithms* **2023**, *16*, 46. https://doi.org/10.3390/a16010046

Academic Editor: Frank Werner

Received: 29 November 2022
Revised: 22 December 2022
Accepted: 1 January 2023
Published: 10 January 2023

1. Introduction

The assistance requests issued by the users to a customer service center are often called trouble or support *tickets*. These tickets generally need to be grouped by the topic of the question, so that each one is answered by an expert on the specific subject. This also aims to provide equity of treatment and prevent possible incoherence between different answers to similar questions. The effectiveness of the whole ticketing system increases when the described *semantic categorization* of the tickets can be done automatically: besides the obvious throughput boost and economic advantages, this will also provide a more consistent and unbiased subdivision. Ticket categorization problems arise in several different areas, and can be tackled by using machine learning techniques.

In the literature, these problems have been approached with a variety of techniques, more and more switching to deep learning in recent times. Quite often, the overall strategy requires a first phase to extract from the text of each ticket the *features* describing that ticket, and then another phase to determine the partitioning. The first phase generally uses *Text Mining*, which is the branch of Data Mining concerning the process of deriving high-quality information from texts, see for details, e.g., [1]. In our case, the aim is to describe each free-form textual ticket with a standardized data record. On the other hand, the second phase can be viewed as a classification operation. Classification is the supervised process that takes a training set of elements, each of which is labeled with a class value, and learns a

criterion to predict the class label of other unseen elements. In our case, it assigns a subject to each data record representing a ticket.

Focusing on recent works, Ref. [2] proposes a hierarchical multi-label classification method to classify the monitoring tickets on the basis of the problem encountered. The authors introduce a contextual hierarchy loss and make also use of knowledge from the domain experts. In [3] the authors propose a multi-purpose text classifier that can tackle tasks independently of domain and language, based on hyperparameter optimization. In [4] the authors propose the use of Convolutional Neural Networks to extract features from the text of IT tickets. In [5] the authors present a whole framework for knowledge extraction from call-center data and other textual data sources such as social networks, to obtain data patterns. Work [6] proposes a Financial Ticket Classification network based on weakly supervised fine-grained classification discriminative filter learning networks. The authors use a deep Convolution Neural Network to extract highly descriptive features of the tickets, and a large-margin softmax loss function to improve the classification accuracy. In [7] the authors use specifically designed features based on linguistic representation, and then various classifiers, to predict the ticket class label of low, medium, or high complexity, showing that even simple algorithms can deliver high-quality predictions when using appropriate linguistic features. The authors of [8] classify emails in four classes, and then compare Naive Bayes, Support Vector Machines (SVMs), and K-NN, integrated with some Natural Language Processing (NLP) techniques (Stop-words removal, Stemming, and feature extraction using TF-IDF and Word2vec). In [9] the authors present a classification of trouble tickets regarding telecommunications networks, designed to use past troubles for the resolution of the problems asked in the current trouble tickets. Finally, [10] classifies support tickets by using multiple methods of text classification and recognition, in particular by embedding documents into feature vectors and using these embeddings for finding similarities between them.

In any case, the categorization of textual tickets turns out to be a particularly challenging multi-class problem. Many words may be in common between messages dealing with completely different subjects, and on the other hand, messages dealing with the same subject may use completely different words. Hence, grasping the semantics may be tricky. Several approaches to multi-class classification exist, based on different models and paradigms. However, to obtain a practically useful subdivision of the tickets, the hyperparameters of the selected classification algorithm should be carefully tuned. Indeed, as recently highlighted by several researchers, the selection of these hyperparameters is often the weak link in many machine learning applications [11,12], because the difference in the classification performance between a "good" and a "bad" hyperparameter choice can be dramatic, especially for hard problems.

For this reason, in the recent subfield of Auto Machine Learning, the values of those hyperparameters are searched by automatic procedures. This constitutes an important meta-learning step. For example, [13] presents hyperopt, a reusable software engine for hyperparameter optimization which could outperform domain experts in the tuning of Deep Belief Networks (DBNs). Work [14] observes that hyperparameter tuning may be very computationally demanding, and proposes a recommender system based on meta-learning to exactly identify when to use default values and when to tune hyperparameters, focusing the analysis on Support Vector Machines and extending it to decision trees. In [15], the author proposes a method to find the hyperparameter tuning for a deep neural network by using a univariate dynamic encoding algorithm. Work [16] presents a hyperparameter tuning framework that works on a small subset of training data using Bayesian optimization, and then it leverages the insights from the learning theory to seek more complex models by using directional derivatives. In [17] the authors propose MonkeyKing, a system that exploits past experience and collects new information to adjust parameter configurations of big data platforms. It can recommend key parameters and combine deep reinforcement learning (DRL) to optimize these key parameters to improve performance. Work [18] presents Hyperband, an algorithm for hyperparameter optimization speeding

up random search through adaptive resource allocation and early-stopping. The problem is formulated as a pure-exploration non-stochastic infinite-armed bandit problem. Finally, ref. [19] proposes an approach named Auto-CASH to select both the algorithm and the hyperparameters, learning them from prior experience by means of a reinforcement learning strategy, so as to be less dependent on human expertise.

In this work, we propose an approach for the automatic categorization of the tickets received by the Contact Center of the Italian National Statistics Institute (Istat) for assistance to users involved in surveys (e.g., Census of Population, Research, and Development Business survey, etc.). This Contact Center has been in operation since January 2016, and currently manages about 60 large statistical investigations and processes an average volume of about 80,000 requests per year. An efficient and accurate automatic categorization of these tickets is particularly needed. The proposed approach is composed of an initial Text Mining phase, which includes tokenization, stop-word elimination, Lemmatization with Part-Of-Speech recognition, and finally feature extraction by using Word2vec [20]. The tickets are written in the Italian language, however, the above text processing is based on the use of a dictionary. Hence, the language can be easily modified by simply changing the dictionary. After this, our approach contains a classification phase, in which we use both deep and traditional learning. In particular, we use Convolutional Neural Networks (CNN) [21] and Support Vector Machines (SVMs) [22]. Our approach is formal and data driven: all the relevant information is extracted from the data, except for the class labels of the training set which are of course externally defined. Hence, the proposed technique is not confined to the specific context described above but can be adapted to the categorization of other texts with different origins.

Moreover, we present an innovative technique for the determination of the hyperparameters of the two classifiers. We propose to view the hyperparameter choice as a higher-level optimization problem, where the hyperparameters are the decision variables and the objective is the performance of the classifier. Note that this approach could also be used in several other learning tasks. However, an explicit analytical model of this problem cannot be defined. Therefore, we propose to consider it a black-box model and solve it by means of derivative-free optimization techniques. These techniques do not need the explicit optimization model, in particular, the analytic expression of the objective function; they only need to numerically evaluate this objective function over a number of points, see e.g., ref. [23] for further details. Due to the discrete nature of the choices, we use for this task a recently proposed algorithm [24] based on primitive directions and non-monotone line search which can deal with integer decision variables.

Hence, the main contributions of this work include: (1) A methodology to solve the difficult multiclass classification problem of the categorization of tickets written in natural language, based on text mining and classification, which works at the formal level and is based on a data-driven approach. Thus, it works independently from the specific content of the tickets, and it could also be applied to the semantic categorization of other texts with different origins or in different languages. (2) An approach to the difficult problem of determining the hyperparameter configuration of a generic machine learning procedure. Again, this approach is purely formal, hence it can solve other problems of hyperparameter optimization, dealing with hyperparameters assuming integer, continuous, or even categorical values.

2. Feature Extraction from Textual Tickets

Each ticket is essentially a small text describing some problem(s) and/or asking some question(s) in natural language. The level may go from very elementary sentences to complex and involved periods. The mentioned semantic categorization of the tickets is needed to assign the tickets to the employees, so that each one is answered by an expert on the specific subject. This would pursue the advantages already described in Section 1. Moreover, this categorization helps in the design and the improvement of the process (e.g.,

the survey), since an abnormal frequency of tickets on a specific aspect may highlight the presence of problems (e.g., unclear questions, defective prompts, etc.).

In many cases, for example, requests received by telephone and recorded, there is no metadata supporting the semantic categorization. In other cases, such as emails, the metadata (e.g., the subject of the email) may be present, but they are often too generic, not reliable enough, or not concerning the desired categorization. In all the above cases tickets need to be categorized homogeneously and preferably automatically. Evidently, basic features of the tickets, like the length of the message or similar measures, are not meaningful for semantic categorization. Even the basic presence or absence of predetermined words is not enough, because in many cases similar words (answer, form, compile, fill, etc.) may describe completely different types of problems. Therefore, a more in-depth analysis is needed to extract the significance of a ticket from its text in natural language.

A scheme of the overall procedure described in this work is reported in the subsequent Figure 1, while the sequence of the steps is reported in Algorithm 1.

Algorithm 1: Categorization of Tickets.

1. Extraction of **Relevant Terms** from the Corpus of 15,000 **Tickets**
   1.a Tokenization
   1.b Stop-words Elimination
   1.c Lemmatization and Part-Of-Speech recognition
2. Preparation of the **Records** (=Standardized Description) of the Tickets
   2.a Word Embedding to associate **Relevant terms** with vectors
   2.b Selection within the 31,000 **Relevant Terms**
   of the 424 **Most Relevant Terms**
   2.c Projection of the Tickets in the space of the **Most Relevant Terms**
   to obtain the **Records**
3. Labeling of 8076 **Records** by Human Experts
4. Training of the **Classifier** with the 8076 **Labeled Records**
   4.a Splitting of training and test sets
   4.b Optimization of the **Hyperparameters**: For every
   combination of **Hyperparameters V'** do the following:
   4.b.i Training with **Hyperparameters V'**
   4.b.ii Testing and evaluation of the performance with **Hyperp.V'**
   4.c Final training with the optimal **Hyperparameters**
5. Categorization of all the **Tickets** (193,419 **Records**) using
   the optimal **Hyperparameters**

After an initial Tokenization (individuation of the words within the sequence of characters) and Stop-words Elimination (elimination of useless parts, like articles, etc.), we perform Lemmatization with Part-Of-Speech recognition. This means that, for each word, we remove the inflectional ending to identify its basic lemma. This allows us to recognize together the different inflected forms of a word (e.g., plurals of nouns, tenses of the verbs, etc.). Moreover, we still keep track of which part of speech each word is (e.g., noun, verb, adjective, etc.). This Natural Language Processing (NLP) is done by using the Gensim python library from scikit learn [25]. Since the tickets are in the Italian language, we perform the above operations by using an Italian dictionary. However, the language can easily be changed by simply switching the underlying dictionary.

After this, we need to convert each ticket into a data record constituting a "standardized description" of the ticket. This feature extraction is done by using the word embedding algorithm Word2vec [20], still in the Gensim python library. This algorithm uses a shallow neural network to learn word associations in a large corpus of text, all the tickets in our case. Hence, it can detect synonymous words. In particular, Word2vec represents each

distinct word with a particular list of numbers called a vector. The vectors are generated in such a way that the cosine similarity between the vectors indicates the level of semantic similarity between the corresponding words. In more detail, we have assembled a corpus composed of the text of 15,000 real tickets received by the Contact Center of the Italian National Statistics Institute (Istat). After the above-described NLP steps, we identified in the word embedding operations a *vocabulary* with 31,000 relevant lemmas, among which we later selected the 424 most relevant lemmas (the top of the list). By projecting on this set of 424 lemmas, each ticket is now converted into a vector with 424 elements, each of which is computed as the frequency of each relevant lemma in that ticket (number of occurrences normalized by the size of the ticket).

Some of those vector representations of the tickets have been labeled by human experts of the Italian National Statistics Institute with 7 distinct classes, representing the subdivision that was wanted, obtaining so a training set of 8076 units. Note that this is a costly operation and no more than 8076 records could be labeled. This set of labeled records will constitute our source of information in the semantic categorization of new unseen tickets, as explained in the next Section, and the procedure will later be used to classify a total of 193,419 records.

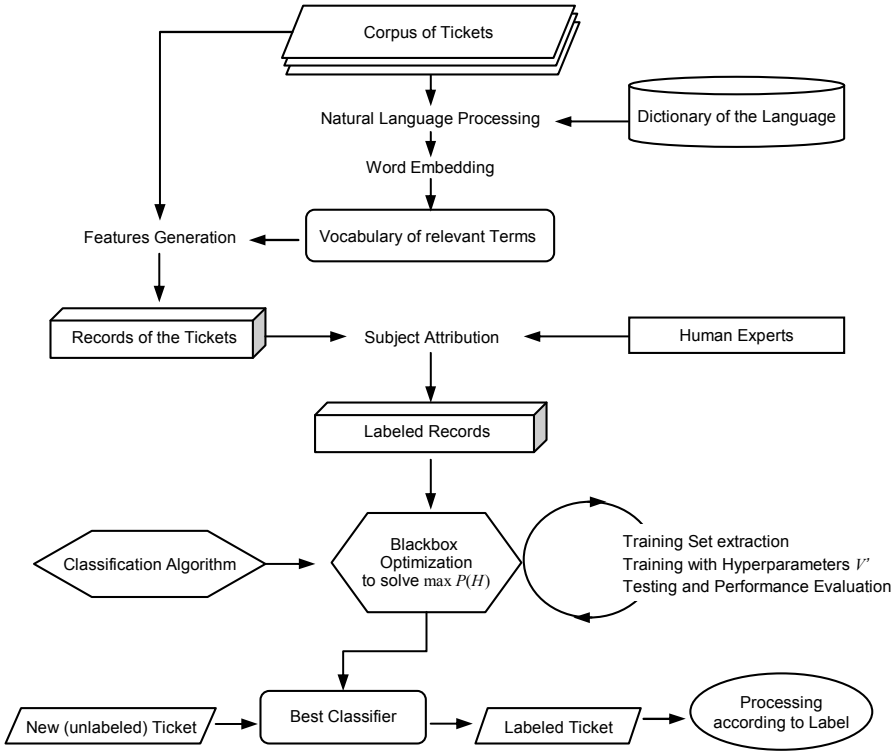

Figure 1. Overall scheme of the proposed approach, including both the text mining phase and the classification phase with hyperparameters optimization.

3. Optimizing the Hyperparameters of the Classifiers

After obtaining the training set of labeled tickets, we set up a classification phase, which can predict the class of unlabeled tickets by learning the classification criteria from the above training set. We perform experiments with both *deep* and *traditional* learning strategies. In particular, we use Convolutional Neural Networks (CNN) [21] and Support Vector Machines (SVM) [22] classifiers.

In general, to use each of these classifiers, a tuple of values $V = (v_1, ..., v_n)$ must be assigned to a set H of *hyperparameters*. Note that, for simplicity, we call here "hyperparameters" all the values needed by the classifier which are not learned by data, not distinguishing between the categories of those values. On the contrary, the values learned by data (e.g., neuron weights in a neural network) are generally called only "parameters"

and will not be set in our hyperparameters optimization but will be determined during the training of the classifier, which constitutes an "inner loop" of our procedure (see also Figure 1).

Generally speaking, the majority of the hyperparameters are bounded to take integer values (e.g., the number of neurons in a neural network), even if some can vary with continuity within a given interval (e.g., the kernel coefficient γ in a support vector machine), and some are categorical (e.g., the choice of the activation function of a neuron). Thus, each value v_i must belong to its feasible domain D_i, and consequently, we can define the set $\mathcal{H}$ of all the feasible tuples of hyperparameter values. The choice of a tuple of values, that is, the choice of a point in $\mathcal{H}$, determines the behavior of the classifier, and so it may dramatically affect its prediction performance P. As a matter of fact, small variations in hyperparameter values may sometimes determine a huge variation in P. Therefore, the values assigned to the hyperparameters must be selected very carefully, and they cannot simply be "default" values or similar.

We propose to view the choice of these values as a higher-level optimization problem, where the hyperparameters H are the decision variables and the objective is the performance P of the classifier, evaluated by choosing a performance measure appropriate to the specific classification case.

$$\begin{aligned} &\max \ P(H) \\ &H \in \mathcal{H} \end{aligned} \tag{1}$$

In our case with 7 classes, the accuracy (defined as the overall percentage of correct class predictions over the test set) was deemed to be the appropriate measure, even if in other cases one may prefer precision, sensitivity, F-1 score, etc. However, the above problem has a main difficulty. Even though P clearly depends on H, this dependency cannot be expressed in analytical form. This is not only due to the complexity of such a relationship, but also because the performance P depends not solely on the rules governing the classifier, but also on the data handled by it. Hence, an explicit optimization model cannot be written for Problem 1.

To solve despite this issue, we propose to adopt a black-box optimization approach and use a derivative-free algorithm. Similar algorithms do not need the explicit optimization model, in particular, the analytic expression of the objective function; they only need to numerically evaluate the objective function over a number of points (i.e., tuples of values of the hyperparameters). In practice, to evaluate a set of hyperparameter values V', we run the classifier on the dataset using values V', doing both training and testing (using 5-fold cross-validation in our experiments), and so we obtain the classification performance P' corresponding to V'.

Due to the discrete nature of the majority of the choices, we use for this task a recently proposed algorithm [24] based on primitive directions and non-monotone line search which can deal with integer variables. Indeed, many hyperparameters assume integer values, or even categorical values which we encode with integers (e.g., the choice of the activation function of a neuron). Also, continuous hyperparameters are discretized, as it is often done, for example in a standard grid search. In particular, such an algorithm can guarantee theoretical convergence to a local optimum of Equation (1) by locally exploring the behavior of the objective function corresponding to variations in the optimization variables. This is obtained in practice by means of an iterative procedure evaluating the objective function over a sequence of points $(V', V'', \dots)$. Each new point is obtained from the previous one V^i by choosing a direction from a set of directions $D(V^i)$ which is specific to the current position, and by exploring the chosen direction by means of a non-monotone line search, to find a new point V^{i+1} that both guarantees a sufficiently large movement along the search direction and an improvement in the value of the objective function.

To avoid getting stuck in points of the integer lattice that cannot be further improved, the set of possible directions is enriched by considering the so-called primitive directions as soon as the algorithm gets stuck. When moving over those points, the algorithm maintains the integrality of the variables and respects box constraints. Possible constraints delimiting

the feasible set are handled by using a penalty approach. The iterations are stopped either when the improvements become numerically negligible, or when an overall maximum number of hyperparameters evaluations is reached, or when a maximum number of evaluations without improvements is obtained. This algorithm is repeatedly restarted from several different starting points, to improve the quality of the final solution (multi-start technique).

To evaluate the advantages of the above optimization approach, we compare it to the selection of the values of the hyperparameters by using a standard technique called *grid search*, which is very often used in the literature. A grid search is the simple evaluation of all the combinations of the possible hyperparameter values. This corresponds to a complete enumeration approach in the solution of problem 1. If not interrupted, it clearly guarantees completeness, hence in theory it always reaches the optimal solution, however, this may require in practice very long times, often excessive.

For each tested classifier, we list below all the values of the hyperparameters that were considered in our experiments. Due to the time required by each hyperparameter evaluation (full training and testing with 5-fold cross-validation), we could experiment with only the sets of values for the hyperparameter that were deemed sufficiently promising. However, in principle, our technique can handle any set of hyperparameter values, provided that the required computations can be carried out in practice. Furthermore, since the number of points tested in a grid search tends to be exponential (all the possible combinations), for mere computational reasons we are forced to reduce in the grid search the number of values tested for each hyperparameter with respect to those considered in the optimization approach.

In particular, we consider two variants of the grid search, which we call "Grid Search" and "Extended Grid Search". The first is intended to run in a reasonable time, thus the size of the search space, fixed in advance, must be quite smaller than that of the Black Box procedure.

The second is intended only as a validation of our Black Box procedure, and not as a practically useful alternative. This is because the grid search clearly guarantees to find the best solution within its search space, and so, when its search space is the same as the Black Box procedure, we can say that the Black Box procedure is running fine if it finds a solution comparable to the one given by the grid search. On the other hand, the time required by such an extended grid search would be excessive for most practical cases. In our experiments, the size of the search space of this extended grid search is the same as the Black Box whenever the running time allows that, and it has been reduced when the running time was really excessive (more than 10 days). This issue is further discussed in Section 4.

3.1. Convolutional Neural Networks

The deep strategy used for our classification is a Convolutional Neural Network (CNN). In general, an artificial Neural Network is a collection of connected nodes called artificial neurons, which resemble the neurons in a biological brain. These artificial neurons are typically organized in layers, according to different architectures, and the resulting network can perform classification tasks by training the neuron input weights to minimize misclassification errors. In particular, CNN includes internal layers that perform convolutions. Such convolutional filters are sometimes called kernels. These networks are often used in image recognition or classification because they behave similarly to neurons in the visual cortex of animals. However, their classification capability can also be successfully applied to texts (see, e.g., [26]), since this type of architecture can extract "deep" features of the data, i.e., distinctive local motifs, regardless of their position in the data record.

In more detail, we build a CNN with three layers of type 2D convolutional, each followed by a 2D Max Pooling layer, and one Dense layer in output. We choose this architecture as a good compromise between speed and performance, since the solution of

problems 1 requires us to train and execute the network a large number of times with many different sets of hyperparameter values. The hyperparameter tested for this CNN are:

- Embedding_dim: width of the kernel matrix for the 2D convolution window. The considered values are (200; 300; 400; 500), both for the optimization approach and for the grid search approach. For the extended grid search, to keep the computational time within 250 h (>10 days) and explore in detail the other hyperparameters, we are forced to consider only (200; 300) which are however the most promising values.
- Filter_sizes: height of the 2D convolution window; in other words, the number of words we want our convolutional filters to cover. We specify 3 different values for the three layers. The values tested with the optimization approach are ([3,4,5]; [5,4,3]; [3,5,7]; [7,5,3]). For the grid search approach, for the computational reasons explained above, we use only ([3,4,5]). For the extended grid search we use ([3,4,5]; [5,4,3]).
- Num_filters: the number of output filters in the convolution. We consider for all the approaches only the value 512, since preliminary experiments showed not much sensitivity to this parameter.
- Optimizer: the optimization technique used in the gradient descent when training the network. For all the approaches we consider (adam; adamax; RMSprop; sgd).
- Loss_function: the function used to calculate the error when training the network. For our multi-class problem, we selected only categorical cross-entropy as the loss function in all the approaches, since it was deemed the most suitable.
- Activation_Conv: activation function for the convolutional layers. The activation functions of a neuron determine whether it should be activated ("fired") or not, based on the inputs received. Many activation functions exist, see also [27]. They must also be computationally efficient because they are calculated across many neurons for each data sample. For the optimization approach, we use (linear; relu; elu; selu; softsign; softplus; sigmoid; hard_sigmoid; exponential; tanh). For the grid search approach, we slightly limit the choice to the most promising and use (relu; softsign; sigmoid; exponential; tanh). For the extended grid search, we use (relu; elu; softsign; sigmoid; exponential; tanh).
- Activation_Dense: activation function for the final dense layer. For the optimization approach, we use (relu; softmax; sigmoid). For the grid search approach, for the computational reasons explained above, we only use (softmax). For the extended grid search, we use (softmax; sigmoid).
- Epochs: an epoch is one complete pass through the training data. Generally, a network is trained for multiple epochs; as a compromise between speed and performance, we select 5 for all the approaches.
- Class_weights: specifies how the errors in the different classes are weighed in the loss function. In keras the possible options are as follows: 1 allows to specify individual weights, in particular, weights proportional to the class frequencies; 2 provides balanced class weights, that is, weights are inversely proportional to the class frequencies; 3 provides uniform class weights, an error that has the same importance for any class. For both the optimization approach and the grid search approaches we use all possibilities (1; 2; 3).

3.2. Support Vector Machines

The traditional (non-deep) strategy that we select for our classification is Support Vector Machines (SVMs). SVMs are supervised learning models that build a deterministic linear classifier. They are based on finding a separating hyperplane that maximizes the margin between the extreme training data of opposite classes. New examples are then mapped into that same space and predicted to belong to a class, on the basis of which side of the hyperplane they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, by implicitly mapping their inputs to a higher dimensional space, see also [22,28]. The hyperparameters tested for SVMs are:

- Kernel: the type of kernel used. For both the optimization approach and the grid search approaches we use (Linear; Radial Basis Function (RBF)). In the case of RBF, it is important to define the penalty parameter c of the error term and the kernel coefficient γ.
- Coefficient γ: is the inverse of the standard deviation of the radial basis kernel, and can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. For the optimization approach and for the extended grid search we use $\gamma = (2^{-19}, 2^{-18.7}, \ldots, 2^{20})$ with step $2^{i+0.3}$ for a total of 131 values. For the grid search we use $\gamma = (2^{-19}, \ldots, 2^{20})$ but step 2^{i+1}, for a total of 40 values.
- Penalty c: is the regularization parameter of the error term. This value allows one to trade off training error vs. model complexity. For the optimization approach and for the extended grid search we use $C = (2^{-8}, 2^{-7.7}, \ldots, 2^{23})$ with step $2^{i+0.3}$ for a total of 95 values. For the grid search we use $C = (2^{-8}, \ldots, 2^{23})$ with step 2^{i+1} for a total of 32 values.
- Class_weights: specifies how the errors in the different classes are weighed during the training. The possible options are: (1) weights proportional to the class frequencies; (2) weights inversely proportional to the class frequencies; (3) provides uniform class weights, an error that has the same importance for any class. For both the optimization approach and the grid search approaches we use all possibilities (1, 2, and 3).

4. Experimental Results

The derivative-free optimization algorithm described in Section 3 to solve the black-box problem 1 has been implemented in Python, and the CNN and SVM classifiers have been realized in the same language by using keras library [29] form scikit learn [25]. We design our experiments to determine the effectiveness of an automatic approach for ticket categorization problems, given the difficulties of a real-world case, and to compare the black-box approach proposed for the optimization of the hyperparameters to a standard grid search.

We consider the case of the tickets received by the Contact Center of the Italian National Statistics Institute (Istat) for assistance to users involved in surveys. This Contact Center has been in operation since January 2016, and currently manages about 60 statistical investigations and processes an average volume of about 80,000 requests per year. The ticket categorization operation is currently performed by human experts, and though it is often performed in a more coarse-grained manner, it requires a considerable amount of work. In our experiments, we have focused on the tickets arising from some economic surveys, such as the "Research and Development Business" survey and the "Information and Communication Technologies usage in Enterprises" survey. This type of survey contains many of the critical issues that users may encounter during the compilation, due to the complexity of the questionnaires. Indeed, they have the following characteristics: they require detailed quantitative information; they use tabular forms for filling out many questions; they contain a large number of prompts, many of which are blocking (also known as *hard prompts*). Therefore, this dataset is well representative of the various cases of requests for assistance.

We prepared a training set of 8076 tickets, whose class has been determined by experts in the field using 7 distinct classes, which are listed below:

1. General information: tickets that require generic information on the survey, like topic or use of the data, etc.
2. Usability: tickets relating to the difficulties in accessing the site or which highlight problems relating to the usability of the electronic questionnaire (e.g., non-editable fields, methods of sending the questionnaire, etc.) or tickets requesting the reopening of the questionnaire already sent, to proceed with the correction of data entered incorrectly.

3. Information on questions: tickets requesting assistance in completing a specific question in the questionnaire (e.g., the question is not applicable to the case of the user, or the case of the user is not present in the questionnaire).
4. Interaction with Istat: tickets that highlight a criticality in the communication process that took place between Istat and end users (e.g., requests for information on the reference year of the survey, the obligation to reply, the deadline for completion, confirmation of the sending of the questionnaire, extensions for the completion of the questionnaire).
5. Eligibility: tickets that show difficulties in understanding whether the unit has the characteristics to be part of the sample or not and therefore is actually required to participate in the survey.
6. Indeterminable / Unclassifiable: tickets with ambiguous content, or tickets with several problems highlighted at the same time, so they cannot be classified.
7. Rest: tickets describing well-defined problems not belonging to the previous classes, however, these problems have a small frequency (they appear rarely). So, instead of using several other small classes, they have been put all into one class: "the rest of the tickets".

The class subdivision of the mentioned training set is the following: 239 in class 1, 219 in class 2, 712 in class 3, 2714 in class 4, 1200 in class 5, 418 in class 6, and 2574 in class 7.

To perform the classification task, we have selected 4,038 records as a training set, that is 50% of the total dataset, and the rest has been used for testing. The extraction has been randomly performed 5 times, and all performance results are averaged on the 5 trials. The resulting 7-class problem is not trivial. Just as an example, preliminary tests with some "default" hyperparameter values obtain less than 50% in accuracy.

As reported in Tables 1 and 2, by considering all the values of the hyperparameters described in the previous Section, in the case of CNN we obtain 5760 possible hyperparameter configurations for the black-box optimization approach, 240 hyperparameter configurations for the grid search approach, and 576 hyperparameter configurations for the extended grid search. Hence, the search space of the extended grid search is one-tenth of the black box, however, it was impossible to extend it further, since this already requires 250 h of computation (more than 10 days). Indeed, if the grid search was extended to the full search space of the black box, the required time could be roughly estimated as $(375{,}000/240) * 5760 = 9{,}000{,}000$ s, which is about 104 days, in the case of CNN.

Table 1. Comparison of black-box optimization and 2 variants of grid search for CNN.

	Black Box	**Grid Search**	**Ext. Grid Search**
Hyperparameter configurations	5760	240	576
Evaluated points	212	240	576
Enumeration percentage	2.76%	100%	100 %
Time in sec.	318,500	375,000	901,000 (>10 days)
Solution accuracy	89.72%	86.15%	89.75%

Table 2. Comparison of black-box optimization and 2 variants of grid search for SVM.

	Black Box	**Grid Search**	**Ext. Grid Search**
Hyperparameter configurations	49,780	1280	49,780
Evaluated points	291	1280	49,780
Enumeration percentage	0.58%	100%	100%
Time in sec.	3490	15,360	597,360 (∼7 days)
Solution accuracy	74.53%	71.87%	74.53%

In the case of SVMs, we obtain 49,780 possible parameter configurations for the black-box optimization approach and for the extended grid search, and 1280 parameter

configurations for the grid search. In this case, the extended grid search can cover the same search space of the optimization approach.

In conclusion, the size of the search space of the optimization approach may be quite larger than that of the grid search. However, only a small fraction of the points of this search space is actually evaluated by the optimization approach (see Evaluated points in Tables 1 and 2), so this last approach can find a solution in less time. The tables also report the percentage of points evaluated by each approach (Enumeration percentage).

In the case of our 7-class problem, we select, as an appropriate performance metric, the accuracy, defined as the overall percentage of correct class predictions over the test set. Hence, in our experiments, we pursue maximum accuracy both during the black-box optimization and during the grid search.

Tables 3 and 4 report respectively the best hyperparameter configurations obtained in the case of CNN and SVM. Clearly, the extended grid search guarantees us to find the best accuracy within its search space, and it was intended to validate the result of the optimization approach. It turns out that the solution obtained by the black-box optimization approach is, for both classifiers, fully comparable with that of the extended grid search. In particular, it is slightly different but with almost the same value in the case of CNN, and it is exactly the same in the case of SVM. However, the time required by the optimization procedure is only a small fraction of that of the extended grid search.

Table 3. Best hyperparameter configuration for CNN.

	Black Box	**Grid Search**	**Ext. Grid Search**
Embedding_dim	300	300	300
Filter_sizes	[5,4,3]	[3,4,5]	[5,4,3]
Num_filters	512	512	512
Optimizer	Adam	Adamax	Adamax
Loss_function	Cat. Cross entropy	Cat. Cross entropy	Cat. Cross entropy
Activation_Conv	Elu	Tanh	Elu
Activation_Dense	Sigmoid	Softmax	Sigmoid
Epochs	5	5	5
Class_weights	2	1	2

Table 4. Best hyperparameter configuration for SVM.

	Black Box	**Grid Search**	**Ext. Grid Search**
Kernel	RBF	RBF	RBF
γ	42	32	42
c	10	8	10
Class_weights	2	2	2

On the other hand, the comparison between the optimization approach and the version of grid search requiring a more reasonable time gives slightly different results. Therefore, the advantages of the proposed optimization approach with respect to the grid search can be described as follows.

1. If we allow a reasonably comparable amount of time for the two approaches, then the optimization approach can explore a much larger search space, hence it can likely find a better hyperparameter configuration that will lead to better predictive performance.
2. If, on the other hand, we allow the same size of the search space for the two approaches, then we have two possible subcases:
    - either we need to use a search space small enough to allow the termination of the grid search in a reasonable time, and in this case, the optimization approach can probably find the same best configuration of the grid search (or a slightly suboptimal one, given the nature of the technique) but using considerably less computational effort,

- or we chose a search space large enough to consider all interesting hyperparameter configurations, and in that case, the grid search may simply become computationally infeasible.

Finally, to further test our approach, we have applied this procedure on a large dataset of 193,419 unlabeled tickets arising from several surveys of economic scope in the period 2016–2019. Although those data were not labeled, and so the exact accuracy is not computable, the results have been judged very satisfactory from a practical point of view, and show that an automatic treatment of these tickets is feasible and useful. This was determined by means of analyses conducted by experts of the field over a random sample of those tickets: the accuracy for the sample was judged to be aligned to that obtained for the 8076 labeled records.

5. Conclusions

The semantic categorization of tickets received by a contact or customer center is an important practical problem. Its solution can provide several advantages, improving both the efficiency of the ticketing system and the quality of the answers, and it may also help in designing or improving the procedures object of the tickets. However, such a categorization is a particularly difficult task, since it requires an automatic extraction of the meaning of the text, followed by a classification, which is often multiclass. This work proposes an approach to this problem, based on text mining and classification, which can use either deep or traditional learning algorithms. The proposed methodology works at the formal level using a data-driven approach, and thus it could also be applied to the semantic categorization of other texts with different origins or in different languages.

Experiments on real-world data from the Contact Center of the Italian National Statistics Institute (Istat) confirm that automatic ticket categorization is practically feasible and very useful. In particular, experiments on the classification of tickets whose real class was known, show that the proposed approach can reach a very good accuracy in very reasonable times.

However, suitable values must be found for the hyperparameters of the classifier, otherwise, the performance may be considerably degraded. Therefore, this work also proposes a technique for the automatic determination of such hyperparameters. The problem is viewed as a higher-level optimization problem where the hyperparameters are the decision variables and the objective is the performance of the classifier. Since an explicit analytical model of this problem cannot be defined, it has been considered as a black-box and it has been solved by means of derivative-free optimization techniques. These techniques do not need the analytic expression of the objective function; they only need to numerically evaluate this objective function over a number of points. In particular, we have used a derivative-free approach that can deal with integer decision variables. All the possible hyperparameter values have been converted into integers, the categorical ones by encoding them with integer values, and the continuous ones by discretizing them. Again, this approach is purely formal, hence it can solve other problems of hyperparameter optimization, even for different machine learning strategies. Future work includes the integration of the proposed hyperparameter optimization methodology in other machine learning techniques, such as those described in [30–32].

Author Contributions: Methodology, R.B., G.B. and P.P.; Software, G.B. and P.P.; Writing—original draft, R.B., G.B. and P.P.; Supervision, R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Sapienza University research grants number RM1161550376E40E and RM120172B870E2E2.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank F. Bianchi and F. Scalfati for their contribution to the experiments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aggarwal, C.C. *Machine Learning for Text*; Springer: Berlin/Heidelberg, Germany, 2018.
2. Zeng, C.; Zhou, W.; Li, T.; Shwartz, L.; Grabarnik, G.Y. Knowledge Guided Hierarchical Multi-Label Classification Over Ticket Data. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 246–260. [CrossRef]
3. Tellez, E.S.; Moctezuma, D.; Miranda-Jímenez, S.; Graff, M. An automated text categorization framework based on hyperparameter optimization. *Knowl.-Based Syst.* **2018**, *149*, 110–123. [CrossRef]
4. Han, J.; Akbari, M. Vertical Domain Text Classification: Towards Understanding IT Tickets Using Deep Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence 2018, New Orleans, LA, USA, 2–7 February 2018.
5. Mateo, R.M.A. A Knowledge Extraction Framework for Call Center Analytics. In *Proceedings of the 18th Online World Conference on Soft Computing in Industrial Applications (WSC18). Advances in Intelligent Systems and Computing*; Ane, B., Cakravastia, A., Diawati, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 864.
6. Zhang, H.; Dong, B.; Feng, B.; Yang, F.; Xu, B. Classification of Financial Tickets Using Weakly Supervised Fine-Grained Networks. *IEEE Access* **2020**, *8*, 129469–129477. [CrossRef]
7. Revina, A.; Buza, K.; Meister, V.G. IT Ticket Classification: The Simpler, the Better. *IEEE Access* **2020**, *8*, 193380–193395. [CrossRef]
8. Putong, M.W.; Suharjito, S. Classification Model of Contact Center Customers Emails Using Machine Learning. *Adv. Sci. Technol. Eng. Syst. J.* **2020**, *5*, 174–182. [CrossRef]
9. Yayah, F.C.; Ghauth, K.I.; Ting, C.-Y. The automated machine learning classification approach on telco trouble ticket dataset. *J. Eng. Sci. Technol.* **2021**, *16*, 4263–4282.
10. Tolciu, D.-T.; Săcărea, C.; Matei, C. Analysis of patterns and similarities in service tickets using natural language processing. *J. Commun. Softw. Syst.* **2021**, *17*, 29–35. [CrossRef]
11. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art. *Knowl.-Based Syst.* **2021**, *212*, 106622. [CrossRef]
12. Kotthoff, L.; Thornton, C.; Hoos, H.H.; Hutter, F.; Leyton-Brown, K. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *J. Mach. Learn. Res.* **2017**, *18*, 1–5.
13. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS) 2011*; NeurIPS: La Jolla, CA, USA, 2011; ISBN 978-161839599-3.
14. Mantovani, R.G.; Rossi, A.L.D.; Alcobaça, E.; Vanschoren, J.; de Carvalho, A.C.P.L.F. A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers. *Inf. Sci.* **2019**, *501*, 193–221. [CrossRef]
15. Yoo, Y. Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches. *Knowl.-Based Syst.* **2019**, *178*, 74–83. [CrossRef]
16. Joy, T.T.; Rana, S.; Gupta, S.; Venkatesh, S. Fast hyperparameter tuning using Bayesian optimization with directional derivatives. *Knowl.-Based Syst.* **2020**, *205*, 106247. [CrossRef]
17. Du, H.; Han, P.; Xiang, Q.; Huang, S. MonkeyKing: Adaptive Parameter Tuning on Big Data Platforms with Deep Reinforcement Learning. *Big Data* **2020**, *8*, 270–290. [CrossRef]
18. Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* **2018**, *18*, 1–52.
19. Mu, T.; Wang, H.; Wang, C.; Liang, Z.; Shao, X. Auto-CASH: A meta-learning embedding approach for autonomous classification algorithm selection. *Inf. Sci.* **2022**, *591*, 344–364. [CrossRef]
20. Mikolov, T.; Le, Q.V.; Sutskever, I. Exploiting Similarities among Languages for Machine Translation. *arXiv* **2013**, arXiv:1309.4168.
21. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Proc. Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1090–1098. [CrossRef]
22. Chang, C.-C.; Lin, C.J. Training v-support vector classifiers: Theory and algorithms. *Neural Comput.* **2001**, *13*, 2119–2147. [CrossRef]
23. Boukouvala, F.; Misener, R.; Floudas, C.A. Global optimization advances in Mixed-Integer Nonlinear Programming, MINLP, and Constrained Derivative-Free Optimization, CDFO. *Eur. J. Oper. Res.* **2016**, *252*, 701–727. [CrossRef]
24. Liuzzi, G.; Lucidi, S.; Rinaldi, F. An algorithmic framework based on primitive directions and nonmonotone line searches for black-box optimization problems with integer variables. *Math. Program. Comput.* **2020**, *12*, 673–702. [CrossRef]
25. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
26. Yoon, K. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014. [CrossRef]
27. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv* **2018**, arXiv:1811.03378.
28. Vapnik, V. *The Nature of Statistical Learning Theory*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1995.
29. Oswal, B.V. CNN-Text-Classification-Keras, GitHub Repository. 2016. Available online: https://github.com/bhaveshoswal/CNN-text-classification-keras (accessed on 1 December 2022).
30. Bruni, R.; Bianchi, G. Effective Classification using Binarization and Statistical Analysis. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2349–2361. [CrossRef]

31. Bruni, R.; Bianchi, G.; Dolente, C.; Leporelli, C. Logical Analysis of Data as a Tool for the Analysis of Probabilistic Discrete Choice Behavior. *Comput. Oper. Res.* **2019**, *106*, 191–201. [CrossRef]
32. Bruni, R.; Bianchi, G. Website categorization: A formal approach and robustness analysis in the case of e-commerce detection. *Expert Syst. Appl.* **2019**, *142*, 113001. [CrossRef]