



Development of accurate classification of heavenly bodies using novel machine learning techniques

Michał Wierzbiński¹ · Paweł Pławiak^{2,3} · Mohamed Hammad⁴ · U. Rajendra Acharya^{5,6,7}

Accepted: 10 February 2021 / Published online: 17 March 2021
© The Author(s) 2021

Abstract

The heavenly bodies are objects that swim in the outer space. The classification of these objects is a challenging task for astronomers. This article presents a novel methodology that enables an efficient and accurate classification of cosmic objects (3 classes) based on evolutionary optimization of classifiers. This research collected the data from Sloan Digital Sky Survey database. In this work, we are proposing to develop a novel machine learning model to classify stellar spectra of stars, quasars and galaxies. First, the input data are normalized and then subjected to principal component analysis to reduce the dimensionality. Then, the genetic algorithm is implemented on the data which helps to find the optimal parameters for the classifiers. We have used 21 classifiers to develop an accurate and robust classification with fivefold cross-validation strategy. Our developed model has achieved an improvement in the accuracy using nineteen out of twenty-one models. We have obtained the highest classification accuracy of 99.16%, precision of 98.78%, recall of 98.08% and F1-score of 98.32% using evolutionary system based on voting classifier. The developed machine learning prototype can help the astronomers to make accurate classification of heavenly bodies in the sky. Proposed evolutionary system can be used in other areas where accurate classification of many classes is required.

Keywords Celestial objects · Machine learning · Evolutionary systems · Genetic optimization · Heavenly bodies · Celestial bodies

1 Introduction

Recently, an accurate stellar object classification using photometric data has gained much popularity. It is an important field of research where lot of improvements are made every year. Hence, many tools have been developed to aid in this task. Some of them include SExtractor Bertin et al. (1996)—a widely used tool for star-galaxy separation. Due to the increase in the number of available sky surveys in optical and near-infrared spectra, and due to varying observing conditions and sensitivities, it is difficult to fine-tune these tools for specific database. The objective of this paper is to present a novel evolutionary system with highest performance in classifying three cosmic objects.

In this work, we chose to conduct our research using the Sloan Digital Sky Survey (SDSS) database Blanton et al. (2017). Astronomers, physicists and mathematicians from many countries took part in this project. Currently, a large part of the classification of objects from SDSS database have used advanced algorithms written in the Matlab environment SDSS (2015). Many researchers have used machine learning

✉ Paweł Pławiak
pławiak@pk.edu.pl; pławiak@iitis.pl;
pławiak.pawel@gmail.com

¹ Spyrosoft S.A., Krakow, Poland
² Department of Computer Science, Faculty of Computer Science and Telecommunications, Cracow University of Technology, Krakow, Poland
³ Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland
⁴ Information Technology Department, Faculty of Computers and Information, Menoufia University, Menoufia, Egypt
⁵ Department of Electronics and Computer Engineering, Ngee Ann Polytechnic, Singapore, Singapore
⁶ Department of Biomedical Engineering, School of Science and Technology, Singapore School of Social Sciences, Singapore, Singapore
⁷ Department of Bioinformatics and Medical Engineering, Asia University, Taichung City, Taiwan

methods to classify the heavenly bodies automatically. The details on their study are provided in Sect. 3. The drawbacks of these studies are given below:

1. Majority of previous studies have used either very complex frameworks or basic methods to build the classification model.
2. It is complex and time-consuming to handle huge data.
3. Handling of large number of features will make the model computationally more intensive.

In order to overcome the above-mentioned drawbacks in this work, we proposed a novel machine learning approach that utilized genetic algorithm to find the best model with optimal set of hyperparameters. First, we reduced the number of features using principal component analysis to speed up the computations. Next, we trained a set of 21 classifiers with their default parameters as a baseline. Then, using genetic algorithm, we optimized their hyperparameters. Resulting individual models were then combined into a single voting classifier, using genetic algorithm to find a combination that yielded the best results. During the learning process fivefold cross-validation (Mosteller and Tukey 1968) is used. The main novel contributions of this work are as follows:

1. Introduced new and efficient solution based on machine learning models to classify three classes of cosmic objects.
2. Developed a genetic algorithm optimization technique to obtain high classification accuracy with small number of features.
3. Achieved highest performance (over 98% accuracy score) using 15 out of 21 tested classifiers.
4. Employed various clinical parameters to evaluate the performance of the developed model.

The rest of this paper is structured as follows. Section 2 discusses the previous works. Section 3 describes the proposed model based on evolutionary optimization of classifier parameters; data collection and preprocessing, model design, training and validation, are also described in this section. Experimental results and discussion are provided in Sects. 4 and 5, respectively. Conclusions and future works are given in Sect. 6.

2 Previous work

There is a significant increase in research works related to stellar spectra detection and classification. Many researchers focused on star-quasar (Zhang et al. 2011; Jin et al. 2019; Zhang et al. 2009, 2013; Viquar et al. 2018), galaxy-quasar (Bailer-Jones et al. 2019) or star-galaxy Philip et al. (2002)

binary classification. Others (López et al. 2010, Becker et al. 2020) focused on multi-class classification of stars, galaxies and quasars Cabanac et al. (2002); Acharya et al. (2018). In these works, various methods have been applied to automatically classify the heavenly bodies accurately.

Many authors used classical machine learning algorithms such as support vector machines (SVM) or *k*-nearest neighbors (*k*NN) (Zhang et al. 2011, 2009, 2013; Tu et al. 2015,[12,15],Jin et al. 2019; Viquar et al. 2018). Others adopted deep learning techniques (Becker et al. 2020, 11) or developed their own novel solutions (Viquar et al. 2018).

Many databases related to sky survey data are freely available. Among them, most popular databases are SDSS (Zhang et al. 2011, 2013, 2009; Viquar et al. 2018, Acharya et al. 2018), Gaia (Bailer-Jones et al. 2019; Becker et al. 2020), WISE (Jin et al. 2019; Becker et al. 2020) and UKIDSS (Zhang et al. 2011, 2013). The summary of related works conducted using these databases is shown in Table 1

Zhang et al. (2011) operated on the data from UKIDSS database. Their best model, a LS-SVM proved to be a highly efficient and powerful in classifying the photometric data. Jin et al. (2019) used data from WISE database using two new color criteria ($yW1W2$ and $iW1zW2$), which were constructed to distinguish quasars from stars efficiently. In Zhang et al. (2009) a *k*NN algorithm is used to distinguish star and quasar sources. Authors of Zhang et al. (2013) used SVM classifier for the same purpose. They achieved very high accuracy scores using SDSS DR7 and UKIDSS DR7 catalogs of photometric data. Viquar et al. (2018) used the same database and asymmetric AdaBoost to classify quasars and stars. In Zhang et al. (2013), Zhang et al. used supervised and unsupervised methods on quasar-star classification problem. Philip et al. (2002) reported higher performance using difference-boosting neural network (DBNN) in star-galaxy classification, which is comparable to the SExtractor.

Authors of Acharya et al. (2018) demonstrated how multi-class classification of stellar sources can be scaled to billions of records by incorporating the scalability of the cloud. Multi-class classification is also performed by Cabanac et al. (2002). They showed that the first 10 eigenvectors of the Karhunen–Loève expansion or principal component analysis (PCA) provided a robust classification scheme for the identification of stars, galaxies and quasi-stellar objects from multi-band photometry.

López et al. (2010) developed an automatic multistage classification system based on Bayesian networks for the OMC (Optical Monitoring Camera) data. They focused on multi-class classification of different categories of stars. Becker et al. (2020) worked on similar problem. They proposed an end-to-end solution for automated source classification based on RNNs. They then compared the classification results with random forest classifier.

Table 1 Summary of state-of-the-art techniques developed using the sky survey databases

Authors	Classification task	Methods	Datasets
Zhang et al. (2011)	Binary (star, quasar)	LS-SVM	SDSS, UKIDSS
Zhang et al. (2013)	Binary (star, quasar)	SVM	DSS, UKIDSS
Philip et al. (2002)	Binary (star, galaxy)	Difference Boosting Neural Network (DBNN)	NDWF
Viquar et al. (2018)	Binary (star, quasar)	SVM, SVM-KNN, AdaBoost, Asymmetric AdaBoost	SDSS
Acharya et al. (2018)	Multi-class (star, galaxy, quasar)	kNN, SVM, Random Forest	SDSS
Lopez et al. (2010)	Multi-class (stars)	Bayesian Networks	OMC
Cabanac et al. (2002)	Multi-class (star, galaxy, quasar)	PCA	GISSEL PEGASE
Bailer-Jones et al. (2019)	Binary (galaxy, quasar)	Gaussian Mixture Model (GMM)	Gaia
Jin et al. (2019)	Binary (quasar, not-quasar)	SVM, XGBoost	Pan-STARSS WISE
Becker et al. (2020)	Multi-class (star)	Random Forest, Recurrent Neural Network (RNN)	OGLE-III Gaia WISE
Zhang et al. (2009)	Binary (star, quasar)	kNN	SDSS FIRST USNO-B1.0

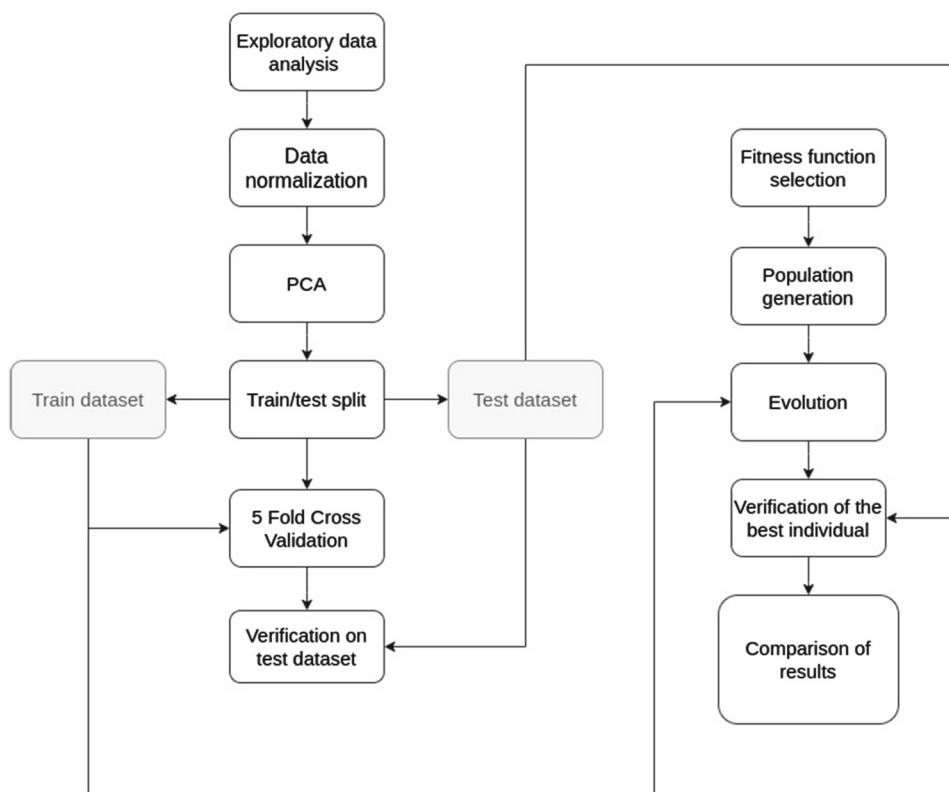
Authors of Bailer-Jones et al. (2019) used Gaussian mixture models to probabilistically classify objects in Gaia data release 2 (GDR2) using photometric and astrometric data. Their trained model is able to classify star, quasar and galaxy with high accuracy.

Genetic algorithms (GAs) are widely used in many fields as they are versatile and can provide very good results, especially when the search space is large, outperforming standard optimization techniques such as random or grid searches (Liashchynskiy and Liashchynskiy 2019). Constant optimizations are being developed since its first inception (De Jong et al. 1977). Wu Deng et al. proposed an algorithm that addressed premature convergence, low search ability and tendency to fall into local optima of quantum evolutionary algorithm (QEA)—improved QEA with multi-strategies namely MSIQDE (Deng et al. 2020). Authors used this algorithm to optimize hyperparameters of DBN model. Another usage example is provided in Deng et al. (2020). Authors developed improved QEA based on the niche

co-evolution strategy and enhanced particle swarm optimization (PSO)—IPOQEA. Proposed system is used to solve airport gate resource allocation problem. A new optimal mutation strategy based on the complementary advantages of five mutation strategies has been used in Deng et al. (2020) to develop an improved differential evolution algorithm with the wavelet basis function. This algorithm can improve the search quality while simultaneously accelerating convergence and help in avoiding the falling into local optimum. Song et al. proposed a multi-population parallel co-evolutionary differential evolution, named MPPCEDE, to optimize parameters of photovoltaic models. Sezer et al. in Sezer et al. (2017) developed a stock trading system which used optimized technical analysis parameters for creating buy–sell points using genetic algorithms.

This paper uses approach similar to that presented in Pławiak and Acharya (2020) by Pławiak and Acharya. They used a GA for parameter optimization coupled with k-fold cross-validation (CV) for arrhythmia detection using ECG signals.

Fig. 1 Learning and optimization pipeline developed for the classification



Similar to their work, we have built the hyperparameter optimization pipeline based on GA, which takes the raw data and performs its pre-processing, parameter optimization coupled with fivefold CV, yielding a list of classifiers with optimal parameters. Using those classifiers, a voting ensemble optimized using GA further increased the classification accuracy.

3 Materials and methods

Sloan Digital Sky Survey is a project that provides public database on observations of celestial objects Blanton et al. (2017).

A special 2.5-m-diameter telescope was used to observe celestial objects, which was built in New Mexico at the Apache Point Observatory in the USA. The telescope used a camera consisting of 30 charge-coupled device (CCD) chips with a resolution of 2048×2048 each. The chips were arranged in 5 rows with 6 in each row. Each row observes the space through various optical filters (u' , g' , r' , i' , z') with different wavelengths $u' = 354$ nm, $g' = 475$ nm, $r' = 622$ nm, $i' = 763$ nm and $z' = 905$ nm [25].

SDSS database consists of two main tables `PhotoObj` and `SpecObj`. The SkyServer and CASJOB portals provide web interfaces build for SQL query execution over those tables. We have collected data from aforementioned tables and randomly selected 10,000 records of cele-

tial bodies collected by SDSS Data Release 16 (DR16). While querying the data, we made sure to closely follow approach taken by Peng et al. (2012) and Jin et al. (2019). Query filters of `sciencePrimary = 1`, `Mode = 1` and `zWarning = 0` are applied. Each observation is described by 8 attributes (u' , g' , r' , i' , z' bands, `right ascension`, `declination` and `redshift`) and the class to which it belongs—`star`, `galaxy` or `quasar`.

Figure 1 presents the learning and optimization pipeline developed for the classification process.

The pipeline consists of the following steps:

(i) Exploratory data analysis is performed manually in order to understand the data better. It is a widely used practice and employed rich data visualization techniques to identify the issues with the data. Those issues might include: missing feature values, large number of outliers, different data scales and presence of categorical features. (ii) Initial data pre-processing is an important step for any machine learning project. Data need to be cleaned and scaled in order for our classifiers to learn the class relationships well. This step also contains dimensionality reduction techniques such as principal component analysis (PCA). Application of PCA allows us to reduce the number of features the model needs to process for accurate prediction (Pearson 1900). This in turn speeds up the learning process significantly. (iii) In train/test split step, dataset is shuffled and split into training and test sets. We used 75% of our data for training and the rest for testing. Strati-

Table 2 Summary of genetic algorithm parameters used for classification with optimization

Parameter	Value
Fitness function	Classification accuracy
Tournament size	3
Population size	50
Gene mutation probability	0.1
Gene crossover probability	0.5
No. of generations	15
Errored individual score	0.0
Elitism (individuals)	1

fied split is used to preserve the class balances. (iv) In this work, we have used 21 classifiers from the `scikit-learn` package Pedregosa et al. (2011) and have initialized their parameters by default setting. (v) After initialization of all classifiers, the classifiers are developed using fivefold CV. This helps us prevent over- or under-fitting of the classifiers. (vi) The verification of final classification performance on the test set which is our baseline study is done. (vii) Genetic parameter optimization consists of: (a) choosing proper fitness function and we chose accuracy as the fitness function. This function, however, can be replaced by other performance metrics such as precision, recall or F_1 score. (b) Population generation: population size and other details regarding the genetic algorithm are presented in Table 2. (c) Evolution: this step consists of cross-over, selection of best individuals, gene mutation and other related operations. It is performed together with fivefold CV on training dataset and verification on the test dataset. For every individual, the same steps used for baseline are applied. Elitism and multipoint gene mutation strategy is used. Elitism strategy helps us to keep the best individual Bhandari et al. (1996). This way we make sure that even if current generation yielded no better individuals, the best individual from previous generation will still participate in the next one. (d) Saving the best individual: after all generations have passed, the best individual will be saved. This individual can be then used for further applications. (viii) Finally, the optimized learning process is verified and compared with one without optimization. The computational complexity of genetic algorithm in terms of O notation is given by $O(gnm)$ with g indicating the number of generations; n and m denote the size of population and the individuals, respectively.

During data analysis, high correlation in the light band variables (u' , g' , r' , i' , z') is observed. The correlation found between magnitudes is to be expected, since the magnitudes contain information about the total brightness of an object and its spectral shape. Those 5 light-bands are substituted by lower number of variables produced by PCA

algorithm (Pearson 1900). This correlation can be observed in Fig. 2. The number of principal components set for this is 3. This helped us to keep over 99% of explained variance. Hence, the training and testing time is significantly reduced.

The final dataset used in the learning process contains 10,000 observations, of which each is described by 3 variables produced by the PCA algorithm, redshift, right ascension and declination. The entire dataset is then split into training and test datasets. The training set contained 75% of observations from the original dataset, and the test set contained rest of the samples.

In the learning process, the stratified fivefold CV is used. Finally, the models are tested using the test set. Stage II of the experiment involved using genetic parameter optimization on the training model. The datasets used for both stages are the same. The only difference is the learning process. In the stage II, we trained the model until we have reached the maximum number of generations in our evolutionary algorithm. The best individual is then verified using the same test set which is used in stage I.

4 Results

This section contains the results of learning and genetic parameter optimization processes.

The below tables and figures show the results obtained using both stages of learning processes. The classifier parameters in stage I are chosen “by hand”, using default values in many cases. Parameters of the classifiers for stage II of the experiment are obtained using the genetic algorithm. Baseline and search space configurations are provided in Appendix A.

The voting classifier Re and Valentini (2012) in the stage I consists of 11 estimators: (i) quadratic discriminant analysis, (ii) support vector machine of type Nu, (iii) radial basis function kernel support vector machine, (iv) poly kernel support vector machine, (v) decision tree classifier, (vi) random forest classifier, (vii) XGBoost classifier, (viii) bagging classifier, (ix) multilayer perceptron, (x) extra trees classifier and (xi) naive Bayes classifier.

Genetic parameter optimization reduced the number of those estimators to only 3: (i) gradient boosted trees classifier; (ii) random forest classifier and (iii) support vector machine with polynomial function kernel.

4.1 Accuracy score

It can be noted from Table 3 that before the optimization, the best results (99.01%) are achieved by XGBoost classifier (Chen and Guestrin 2016). Second and third best classifiers are gradient boosted trees (98.92%) Bühlmann (2012)

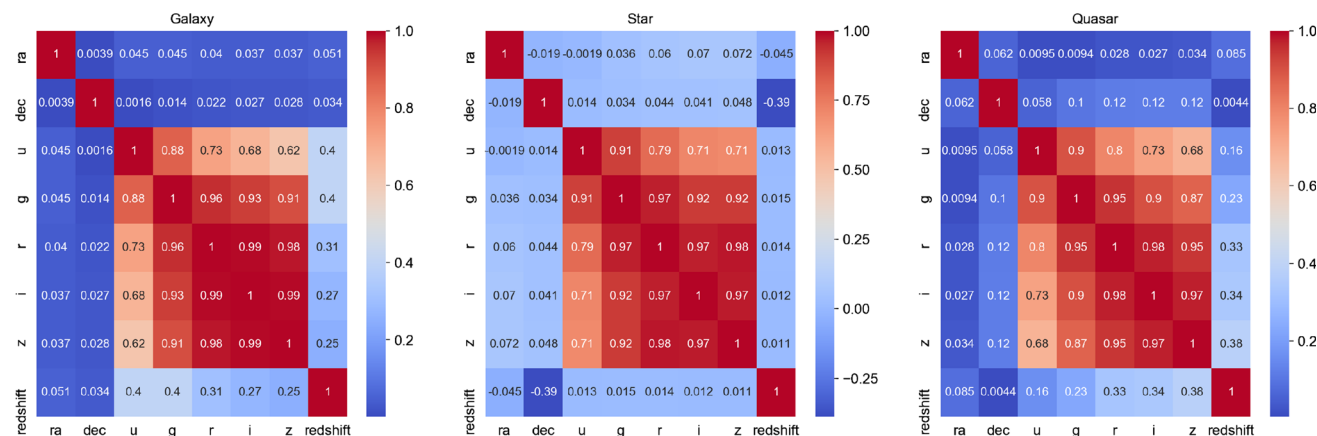


Fig. 2 Correlation matrices for each of the classes

Table 3 Summary of accuracies obtained using various classifiers before and after genetic parameter optimization

Classifier	Accuracy before optimization	Accuracy after optimization	Increase in performance
AdaBoost classifier	0.9857	0.9903	0.0046
Bagging classifier	0.9880	0.9785	-0.0095
Decision tree classifier	0.9857	0.9891	0.0034
Extra trees classifier	0.9836	0.9892	0.0056
Gaussian Naive Bayes	0.9785	0.9787	0.0002
Gradient boosting classifier	0.9889	0.9907	0.0018
KNN	0.8977	0.9249	0.0272
Linear discriminant analysis	0.9300	0.9303	0.0003
Linear SVM	0.9641	0.9824	0.0183
Logistic regression	0.9717	0.9849	0.0132
MLP classifier	0.9884	0.9860	-0.0024
Nu-SVM	0.9831	0.9871	0.0040
Passive aggressive classifier	0.9588	0.9744	0.0156
Poly Kernel SVM	0.8547	0.9907	0.1360
Quadratic discriminant analysis	0.9848	0.9849	0.0001
Random forest classifier	0.9875	0.9911	0.0036
RBF Kernel SVM	0.9603	0.9903	0.0300
Ridge classifier	0.9203	0.9232	0.0029
SGD classifier	0.9680	0.9851	0.0171
Voting classifier	0.9885	0.9916	0.0031
XGBoost classifier	0.9901	0.9901	0.0000
Average	0.9647	0.9778	0.0131

The best results are shown in bold

and multilayer perceptron (98.88%) (White and Rosenblatt 1963).

After genetic parameter optimization, the best classifier in terms of classification accuracy is voting classifier with 99.16% accuracy. The random forest classifier (99.11%) (Ho 1998) is the second best, and SVM classifier with poly-

mial kernel ex aequo with gradient boosted trees from the scikit-learn package—both achieved 99.07% classification accuracy.

Figure 3 shows the plot of the accuracies before and after the genetic parameter optimization.

The average accuracy before optimization is 96.5% and increased to 97.8% after the genetic optimization. The average increase in classification accuracy is 1.3%. It can also be noted that 7 out of 21 classifiers achieved the accuracy of more than 99%. Before optimization, only XGBoost classifier obtained above 99% accuracy. After optimization, this result is increased to 99.16% (voting classifier). Nineteen out of twenty-one classifiers have performed better after genetic optimization.

4.2 Precision score

Random forest classifier yielded the highest precision score of 98.61%. Following that, voting classifier and XGBoost classifier yielded the precision of 98.56 and 98.41%, respectively. The results of all classifiers are presented in Table 4.

After the optimization, extra trees classifier (Geurts et al. 2006), AdaBoost and the voting classifier yielded precision scores of 98.66%, 98.61% and 98.57%, respectively.

Figure 4 shows the plot of precision scores before and after the genetic parameter optimization.

The average precision before optimization is 95.6%. After genetic optimization, this value is increased to 96.4%. The average increase in precision score is 0.8%. Before optimization, the highest precision score is 98.61% (random forest). After genetic optimization, this result is increased to 98.66% by the extra trees classifier.

4.3 Recall score

Quadratic discriminant analysis Bose et al. (ddd), MLP and XGBoost classifier yielded results of 98.39%, 98.02% and

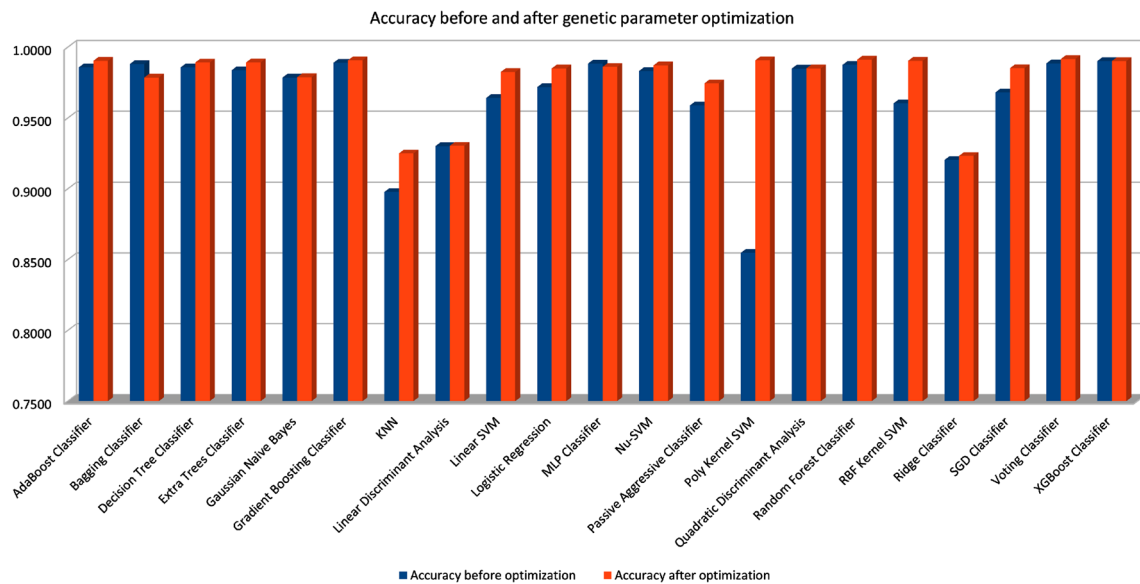


Fig. 3 Accuracies obtained using various classifiers before and after the genetic parameter optimization

Table 4 Summary of precision score obtained using various classifiers before and after the genetic parameter optimization

Classifier	Precision before optimization	Precision after optimization	Increase in performance
AdaBoost classifier	0.9712	0.9861	0.0149
Bagging classifier	0.9824	0.9742	-0.0082
Decision tree classifier	0.9665	0.9784	0.0119
Extra trees classifier	0.9809	0.9866	0.0057
Gaussian Naive Bayes	0.9579	0.9579	0.0000
Gradient boosting classifier	0.9784	0.9852	0.0068
KNN	0.8828	0.9092	0.0264
Linear discriminant analysis	0.9249	0.9249	0.0000
Linear SVM	0.9575	0.9742	0.0167
Logistic regression	0.9599	0.9723	0.0124
MLP classifier	0.9822	0.9661	-0.0161
Nu-SVM	0.9790	0.9726	-0.0064
Passive aggressive classifier	0.9347	0.9069	-0.0278
Poly Kernel SVM	0.8638	0.9857	0.1219
Quadratic discriminant analysis	0.9719	0.9730	0.0011
Random forest classifier	0.9861	0.9827	-0.0034
RBF Kernel SVM	0.9405	0.9814	0.0409
Ridge classifier	0.9197	0.9062	-0.0135
SGD classifier	0.9657	0.9476	-0.0181
Voting classifier	0.9856	0.9857	0.0001
XGBoost classifier	0.9841	0.9841	0.0000
Average	0.9560	0.9639	0.0079

The best results are shown in bold

97.91%, respectively, before the genetic algorithm parameter optimization. The summary of all of the classifiers is given in Table 5.

After genetic optimization, quadratic discriminant analysis, SVM with polynomial kernel function and logistic

regression (Cabrera 1994) yield the recall scores of 98.52%, 98.43% and 98.12%, respectively.

Figure 5 shows the plot of the recall scores before and after the genetic parameter optimization.

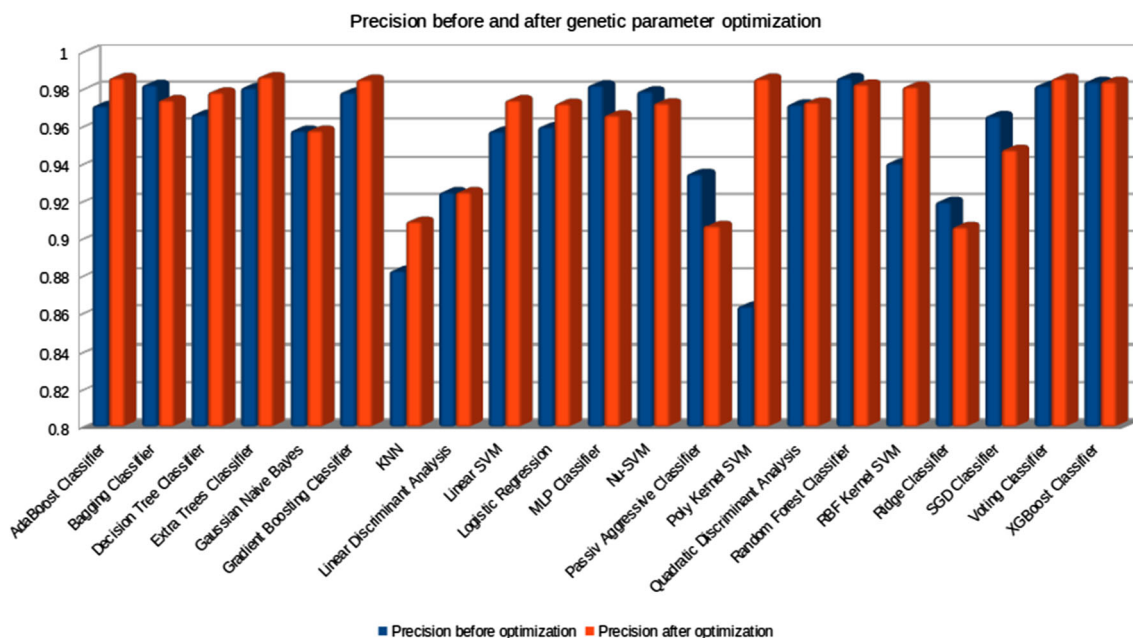


Fig. 4 Precision scores obtained using various classifiers before and after the genetic parameter optimization

Table 5 Summary of recall score obtained using various classifiers after the genetic parameter optimization

Classifier	Recall before optimization	Recall after optimization	Increase in performance
AdaBoost classifier	0.9694	0.9810	0.0116
Bagging classifier	0.9775	0.9595	-0.0180
Decision tree classifier	0.9697	0.9714	0.0017
Extra trees classifier	0.9693	0.9772	0.0079
Gaussian Naive Bayes	0.9741	0.9741	0.0000
Gradient boosting classifier	0.9785	0.9780	-0.0005
KNN	0.8632	0.8858	0.0226
Linear discriminant analysis	0.8657	0.8657	0.0000
Linear SVM	0.9342	0.9726	0.0384
Logistic regression	0.9520	0.9812	0.0292
MLP classifier	0.9802	0.9652	-0.0150
Nu-SVC	0.9659	0.9699	0.0040
Passive aggressive classifier	0.9114	0.9049	-0.0065
Poly Kernel SVM	0.8106	0.9843	0.1737
Quadratic discriminant analysis	0.9839	0.9852	0.0013
Random forest classifier	0.9736	0.9789	0.0053
RBF Kernel SVM	0.9286	0.9809	0.0523
Ridge classifier	0.8637	0.9065	0.0428
SGD classifier	0.9394	0.9167	-0.0227
Voting classifier	0.9763	0.9808	0.0045
XGBoost classifier	0.9791	0.9791	0.0000
Average	0.9413	0.9571	0.0158

The best results are shown in bold

The average pre-optimization recall score is 94.1%. After genetic optimization, it is increased to 95.7%. The average increase in the recall score is 1.6%. As a result of parameter

optimization, 17 out of 21 classifiers got better results after the optimization. In both conditions, quadratic discriminant analysis performed better than the rest of the classifiers.

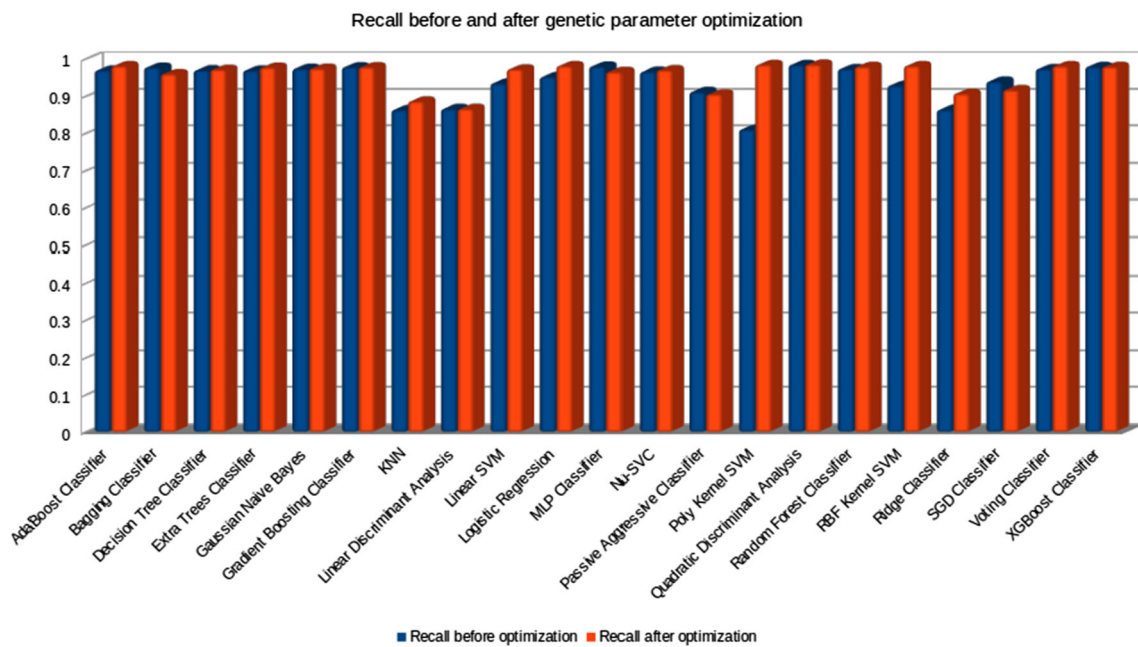


Fig. 5 Recall scores obtained using various classifiers before and after the genetic parameter optimization

4.4 F1-score

The XGBoost classifier before optimization yielded the highest F1-score of 98.16%. MLP model and bagging classifier provided the F1-score of 98.11% and 97.99%, respectively. The F1-scores before and after the genetic optimization are shown in Table 6.

It can be noted from Table 6 that after genetic parameter optimization, SVM classifier with polynomial kernel function, AdaBoost and voting classifier provided the F1-scores of 98.5%, 98.35% and 98.32%, respectively.

Figure 6 shows the plot of F1-scores before and after the genetic parameter optimization.

The average value of F1-score before optimization is 94.7%. After genetic optimization, this value is increased to 96%. The average increase in F1 score is 1.3%. Before the optimization, XGBoost yielded the highest F1-score of 98.4%. After optimization, SVM classifier with polynomial kernel yielded the F1-score of 98.5%. It can be noted that for many classifiers F1-score is improved.

5 Discussion

Table 7 provides the summary of comparison with other similar works (Viquar et al. 2018; Zhang et al. 2011, 2013; Acharya et al. 2018; Zhang et al. 2009) developed for the automated detection of heavenly bodies using the same SDSS database.

It can be noted from Table 7 that most of the previous works (Zhang et al. 2009, 2013, 2011; Viquar et al. 2018) have performed binary classification and obtained high performance.

Recently, Acharya et al. (2018) have classified three classes using random forest classifier and reported the classification accuracy of 94%. To the best of our knowledge, we are the first group to achieve over 99% accuracy for three-class classification of heavenly bodies. In future, we intend to use the whole dataset of 4 million objects to train the model which may improve the classification performance. We can also use genetic algorithms to reduce the number of features and select only those, which would improve our accuracy score. Yet another option will be to use genetic parameter and feature optimization with asymmetric AdaBoost classifier as proposed by Viquar et al. (2018).

Advantages of the proposed system are:

1. Obtained highest classification accuracy.
2. Proposed a novel model based on genetic algorithm.
3. Model is simple to use and robust as it is developed using fivefold cross-validation..

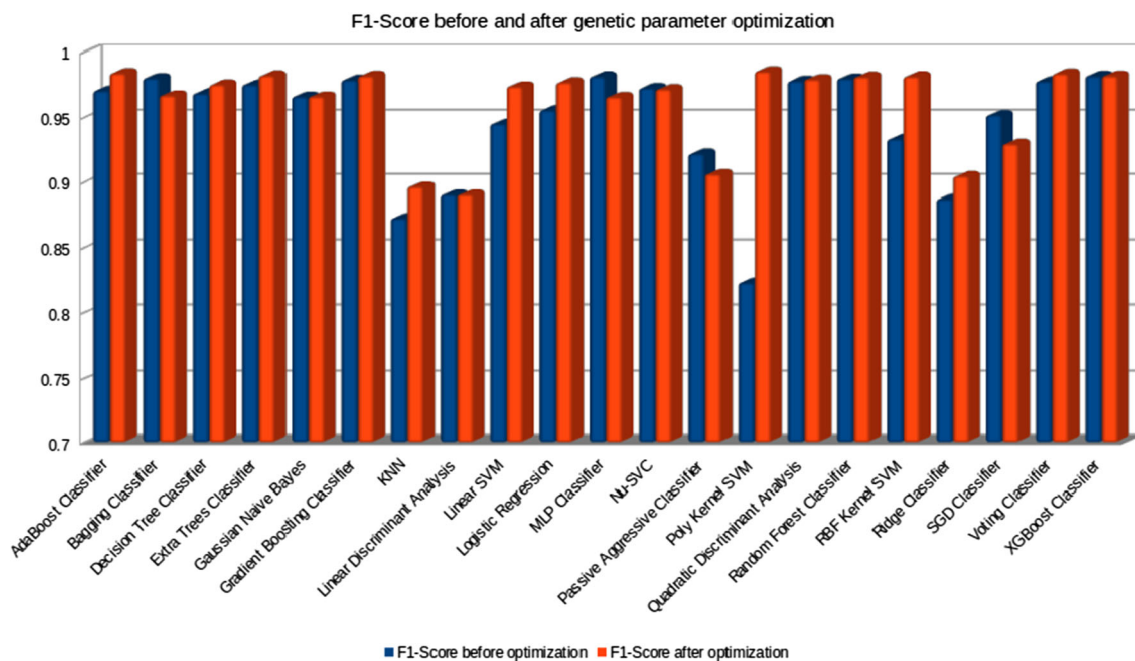
Limitations of our work are:

1. A small number of photometric records (10,000 instances) are analyzed. The challenge for the astronomers is to accurately classify at various scales. Our approach would need to be scaled several order of magnitudes to meet those needs.

Table 6 Summary of F1-score obtained using various classifiers before and after the genetic parameter optimization

Classifier	F1-Score before optimization	F1-Score after optimization	Increase in performance
AdaBoost classifier	0.9703	0.9835	0.0132
Bagging classifier	0.9799	0.9665	-0.0134
Decision tree classifier	0.9681	0.9748	0.0067
Extra trees classifier	0.9749	0.9817	0.0068
Gaussian Naive Bayes	0.9656	0.9656	0.0000
Gradient boosting classifier	0.9785	0.9815	0.0030
KNN	0.8724	0.8964	0.0240
Linear discriminant analysis	0.8903	0.8903	0.0000
Linear SVM	0.9447	0.9734	0.0287
Logistic regression	0.9553	0.9765	0.0212
MLP classifier	0.9811	0.9654	-0.0157
Nu-SVC	0.9722	0.9713	-0.0009
Passive aggressive classifier	0.9216	0.9059	-0.0157
Poly Kernel SVM	0.8227	0.9850	0.1623
Quadratic discriminant analysis	0.9777	0.9789	0.0012
Random forest classifier	0.9796	0.9808	0.0012
RBF Kernel SVM	0.9329	0.9811	0.0482
Ridge classifier	0.8871	0.9043	0.0172
SGD classifier	0.9514	0.9291	-0.0223
Voting classifier	0.9808	0.9832	0.0024
XGBoost classifier	0.9816	0.9816	0.0000
Average	0.9471	0.9598	0.0128

The best result is shown in bold

**Fig. 6** F1-scores obtained using various classifiers before and after the genetic parameter optimization

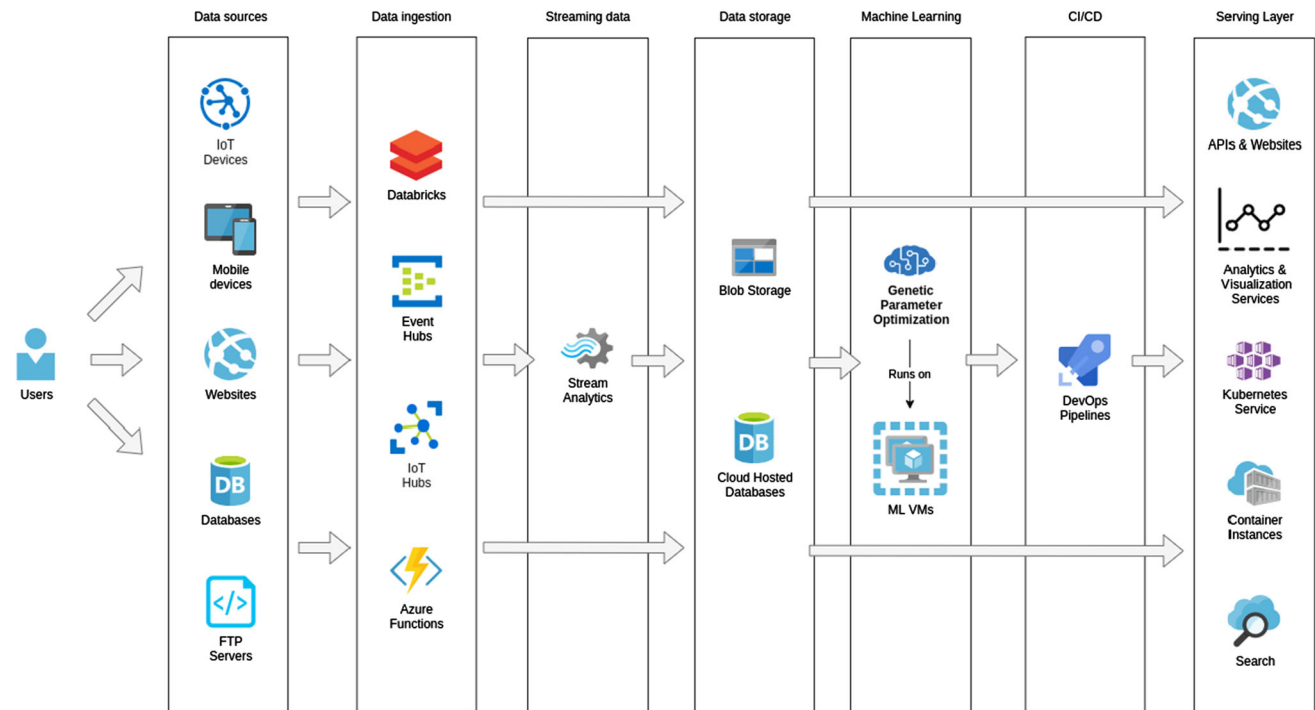
2. It is computationally expensive to find the optimal set of classifiers and their parameters. This approach on large-scale data may not be suitable as high computa-

tional complexity would require parallel task distribution among many nodes, thus increasing the cost.

Table 7 Summary of comparison with similar other works for automated celestial object classification using the same database

Authors	Classification task	Best classifier	Accuracy	Precision	Recall	F1-Score
Zhang et al. (2009)	Binary (star, quasar)	k NN	0.9493	–	–	–
Zhang et al. (2013)	Binary (star, quasar)	SVM	0.9811	–	–	–
Zhang et al. (2011)	Binary (star, quasar)	LS-SVM	0.9881	–	–	–
Viquar et al. (2018)	Binary (star, quasar)	Asymmetric AdaBoost	0.9995	1.0000	1.0000	1.0000
Acharya et al. (2018)	Multi-class (star, galaxy, quasar)	Random Forest	0.9400	–	–	–
This work	Multi-class (star, galaxy, quasar)	Voting Classifier with genetic parameter optimization	0.9916	0.9878	0.9808	0.9832

The best results are shown in bold

**Fig. 7** An illustration of the proposed genetic parameter optimization methodology usage in real-world scenario on Azure Cloud

The disadvantage of proposed methodology is its computational complexity. Hence, we intend to explore the possibility of using cloud environment. An example of cloud architecture (based on Microsoft Azure) that incorporates our system is shown in Fig. 7. This methodology is not limited to astronomy and can be extended to other applications as well. Proposed architecture can take data from different sources, store them and perform machine learning model optimization using our approach. Elastic scaling of the cloud resources is necessary when the data size is huge. To further leverage fully manage cloud services, we could run our evolutionary optimization pipeline on Azure Batch Service [38] instead of using virtual machines. This will give us dynamic scaling capabilities, and hence, we need to pay for the infrastructure only when we use it. After training and evaluating the model, the cost will be further reduced. Running our pipeline

on Azure Batch gives us the ability to run the whole process not only on demand but also automatically. The data used for the testing can be used later to train the model as well. This will make our system more robust and accurate.

6 Conclusion

In this work, we have proposed a novel method of optimizing multi-class classification task using machine learning techniques and genetic algorithm. This approach helps to find the optimal parameters for the classifiers and achieved the highest accuracy of 99%. (Seven out of twenty-one classifiers have achieved the accuracy score of over 99% using our approach.) In future, the proposed model can be used to classify more classes of heavenly bodies and also can

be used for healthcare applications like detection of cardiac ailments, brain abnormalities and other physiological malfunctioning. Various state-of-the-art deep learning techniques can be employed to increase the performance using more data.

Acknowledgements Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions. SDSS-IV acknowledges support and resources from the Center for High-Performance Computing at the University of Utah. The SDSS web site is www.sdss.org. SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration including the Brazilian Participation Group, the Carnegie Institution for Science, Carnegie Mellon University, the Chilean Participation Group, the French Participation Group, Harvard-Smithsonian Center for Astrophysics, Instituto de Astrofísica de Canarias, The Johns Hopkins University, Kavli Institute for the Physics and Mathematics of the Universe (IPMU)/University of Tokyo, the Korean Participation Group, Lawrence Berkeley National Laboratory, Leibniz Institut für Astrophysik Potsdam (AIP), Max-Planck-Institut für Astronomie (MPIA Heidelberg), Max-Planck-Institut für Astrophysik (MPA Garching), Max-Planck-Institut für Extraterrestrische Physik (MPE), National Astronomical Observatories of China, New Mexico State University, New York University, University of Notre Dame, Observatório Nacional/MCTI, The Ohio State University, Pennsylvania State University, Shanghai Astronomical Observatory, United Kingdom Participation Group, Universidad Nacional Autónoma de México, University of Arizona, University of Colorado Boulder, University of Oxford, University of Portsmouth, University of Utah, University of Virginia, University of Washington, University of Wisconsin, Vanderbilt University, and Yale University.

Author Contributions Michał Wierziński conceived and designed the analysis, collected the data, performed the analysis, wrote the paper; Paweł Pławiak conceived and designed the analysis, wrote the paper; Mohamed Hammad wrote the paper; U. Rajendra Acharya wrote the paper.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix

Appendix A. Baseline and search space configurations

Table 8 presents the baseline configuration of the classifiers together with their search space configurations. Where possible, the `random_state` parameter was always set to constant value of 42 for reproducibility purposes. The functions from `numpy` (Harris et al. 2020) package were used to generate numerical parameter value ranges.

Table 8 A sample long table

Classifier	Baseline parameters	Search space configuration
AdaBoost		<ul style="list-style-type: none"> base_estimator: DecisionTreeClassifier with max_depth=3, DecisionTreeClassifier with max_depth=1 n_estimators: np.logspace(1, 3, base=10, num=25, dtype=int) learning_rate: np.logspace(-3, 2, base=10, num=25) algorithm: "SAMME", "SAMME.R"
Bagging Classifier		<ul style="list-style-type: none"> base_estimator: Perceptron(), DecisionTreeClassifier() n_estimators: np.logspace(2, 4, base=10, num=25, dtype=int) max_samples: np.logspace(0, 2, base=10, num=25, dtype=int) max_features: np.logspace(0, 2, base=10, num=25, dtype=int) bootstrap: True, False bootstrap_features: True, False oob_score: True, False
Decision Tree		<ul style="list-style-type: none"> criterion: "gini", "entropy", "splitter": "best", "random" max_depth: np.arange(1, 10) min_weight_fraction_leaf: np.logspace(-3, 0, base=10, num=20) max_leaf_nodes: np.arange(1, 10) presort: True, False min_samples_split: np.arange(2, 10) min_samples_leaf: np.arange(1, 10) max_features: "auto", "log2", None
Gradient Boosting Tree		<ul style="list-style-type: none"> loss: "deviance", "exponential" learning_rate: np.logspace(-2, 1, num=20, base=10) n_estimators: np.logspace(1, 3.5, num=20, base=10, dtype=int) criterion: "friedman_mse" min_samples_split: np.arange(2, 10) min_samples_leaf: np.arange(1, 10) max_depth: np.arange(3, 9)
kNN		<ul style="list-style-type: none"> neighbors: np.arange(1, 12, step=2, dtype=int) weights: "uniform", "distance" algorithm: "auto", "ball_tree", "brute" leaf_size: np.unique(np.logspace(0, 2, base=10, num=20, dtype=int)) p: 1, 2 metric: "euclidean", "manhattan", "chebyshev", "minkowski"
LDA		<ul style="list-style-type: none"> solver: "svd", "lsqr", "eigen" shrinkage: None, "auto" n_components: None store_covariance: True, False
Linear SVM	loss : "hinge"	<ul style="list-style-type: none"> penalty: "l1", "l2" loss: "hinge", "squared_hinge" dual: True, False C: np.logspace(-9, 6, num=30, base=10) multi_class: "ovr" fit_intercept: True, False intercept_scaling: np.logspace(-6, 3, num=40, base=10) class_weight: None, "balanced" max_iter: 10000
Logistic Regression	multi_class : "auto"	<ul style="list-style-type: none"> penalty: "l1", "l2", "elasticnet", "none" dual: True, False C: np.logspace(-4, 2, num=25, base=10) fit_intercept: True, False intercept_scaling: np.logspace(-3, 3, num=40, base=10) class_weight: "balanced" solver: "newton-cg", "lbfgs", "liblinear", "sag", "saga" multi_class: "ovr", "multinomial", "auto" class_weight: "balanced" warm_start: False max_iter: 1000
MLP	max_iter : 1000	<ul style="list-style-type: none"> hidden_layer_sizes: A list of combinations of different sizes of the network's layers. Every combination of up to 3 layer sizes in the following list: 10, 20, 40, 50, 100, 200 was generated activation: "relu", "identity", "logistic", "tanh" solver: "adam", "lbfgs", "sgd" alpha: np.logspace(-5, 1, base=10, num=25) batch_size: np.unique(np.logspace(0, 4, base=10, num=25, dtype=int)) learning_rate: "constant", "invscaling", "adaptive" learning_rate_init: np.logspace(-4, 1, base=10, num=25) power_t: np.logspace(-3, 0, base=10, num=25) max_iter: 1000 momentum: np.logspace(-3, 0, base=10, num=25) neesterovs_momentum: True, False early_stopping: True beta_1: 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9 beta_2: np.logspace(-1, -0.001, base=10, num=25) epsilon: 1e-08 n_iter_no_change: 5

Table 8 continued

Classifier	Baseline parameters	Search space configuration
Naive Bayes		<ul style="list-style-type: none"> •var_smoothing: np.logspace(-15, 1, base=10, num=50)
nu-SVM	<ul style="list-style-type: none"> •mu: 0.05 	<ul style="list-style-type: none"> •nu: np.logspace(-9, 6, num=30, base=10) •kernel: "rbf", "poly", "sigmoid", "linear" •gamma: np.logspace(-9, 6, num=30, base=10) •degree: np.arange(3, 10) •coef0: np.logspace(-9, 6, num=30, base=10) •shrinking: True, False •probability: False, True •tol: 0.001 •cache_size: 20, 50, 100, 200 •class_weight: None, "balanced" •verbose: False •max_iter: 100000
Passive Aggressive Classifier		<ul style="list-style-type: none"> •loss: "hinge", "squared_hinge" •shuffle: True, False •early_stopping: True •warm_start: False •class_weight: None, "balanced" •average: False, 1, 2, 4, 5, 6, 7, 10, 15, 20, 30, 50
Poly kernel SVM	<ul style="list-style-type: none"> •kernel: "poly" 	<ul style="list-style-type: none"> •kernel: "poly" •gamma: np.logspace(-9, 6, num=30, base=10) •degree: np.arange(3, 10) •coef0: np.logspace(-9, 6, num=30, base=10) •shrinking: True, False •probability: False, True •tol: 0.001 •cache_size: 20, 50, 100, 200 •class_weight: None, "balanced" •verbose: False •max_iter: 100000
QDA		<ul style="list-style-type: none"> •reg_param: np.logspace(-15, 5, base=10, num=25) •store_covariance: False, True
Random Forest		<ul style="list-style-type: none"> •on_estimators: np.logspace(2, 3.5, num=10, base=10, dtype=int) •criterion: "gini", "entropy" •bootstrap: True, False •oob_score: True, False •min_samples_split: np.arange(2, 10) •min_samples_leaf: np.arange(1, 10) •max_features: "auto", "log2", None
RBF kernel SVM	<ul style="list-style-type: none"> •kernel: "rbf" 	<ul style="list-style-type: none"> •kernel: "rbf" •C: np.logspace(-9, 5, num=20, base=10) •gamma: np.logspace(-9, 5, num=20, base=10) •degree: np.arange(3, 10) •coef0: np.logspace(-9, 5, num=20, base=10) •shrinking: True, False •probability: False, True •tol: 0.001 •cache_size: 20, 50, 100, 200, 500, 1000 •class_weight: None •verbose: False •max_iter: -1
Ridge Classifier		<ul style="list-style-type: none"> •alpha: np.logspace(-3, 4, base=10, num=25) •fit_intercept: True, False •normalize: True, False •copy_X: True, False •max_iter: 3000 •class_weight: None, "balanced" •solver: "auto", "svd", "cholesky", "lsqr", "sparse_cg", "sag", "saga" •loss: "hinge", "log", "modified_huber", "squared_hinge", "perceptron", "squared_loss", "huber", "epsilon_insensitive", "squared_epsilon_insensitive" •penalty: "l1", "l2", "elasticnet" •alpha: np.logspace(-5, 0, base=10, num=25) •l1_ratio: np.logspace(-2, 0, base=10, num=25) •fit_intercept: True, False •shuffle: True, False •epsilon: np.logspace(-2, 1, base=10, num=25) •learning_rate: "optimal", "constant", "invscaling", "adaptive" •eta0: 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.15, 0.2 •power_t: np.logspace(-2, 0, base=10, num=25) •early_stopping: True •on_iter_no_change: 5 •class_weight: None, "balanced" •average: False, 1, 2, 4, 5, 6, 7, 10, 15, 20, 30, 50
XGB	<ul style="list-style-type: none"> •max_depth: 4 	<ul style="list-style-type: none"> •eta: np.logspace(-3, 0, num=25, base=10) •gamma: np.logspace(-9, 5, num=25, base=10) •max_depth: np.arange(4, 10) •min_child_weight: np.logspace(-2, 5, num=25, base=10) •max_delta_step: np.arange(10) •subsample: np.logspace(-1, 0, num=10, base=10) •max_delta_step: np.logspace(-1, 0, num=10, base=10) •lambda: np.logspace(0, 1, num=10, base=10) •alpha: np.logspace(-9, 1, num=25, base=10) •colsample_bytree: 0.5, 0.7, 0.9, 1 •objective: "multi:softprob" •on_estimators: np.logspace(1, 4, num=25, base=10, dtype=int)

Table 8 continued

Classifier	Baseline parameters	Search space configuration
X-Tree		<p>on_estimators: np.logspace(2, 3.5, num=10, base=10, dtype=int) • criterion: "gini", "entropy" • bootstrap: True, False • oob_score: True, False • min_samples_split: np.arange(2, 10) • min_samples_leaf: np.arange(1, 10) • max_features: "auto", "log2", None</p>
Voting Classifier	<p>• voting: "hard"</p> <p>• estimators: nu-SVM, Poly kernel SVM, RBF kernel SVM, Decision Tree, Random Forest, Bagging Classifier, XGB, MLP, Bagging, X-Tree, Naive Bayes, QDA, all with the parameters as above</p>	<p>• voting: "hard" • estimators: A list where each element consisted of random choice of classifiers. Classifiers were chosen by randomly sampling (without replacement) the list of all of the pre-trained models. Total of 300 estimator choices were generated</p>

References

- Acharya V, Bora P, Karri N, Nazareth A, Anusha S, Rao S (2018) Classification of sdss photometric data using machine learning on a cloud. *Curr Sci* 115:249–257. <https://doi.org/10.18520/cs/v115/i2/249-257>
- Bagging Bühlmann P (2012) Boosting and ensemble methods. *Handb Comput Stat*. https://doi.org/10.1007/978-3-642-21551-3_33
- Bailer-Jones C, Fouesneau M, Andrae R (2019) Quasar and galaxy classification in gaia data release 2. *Mon Notices R Astron Soc* 490:5615–5633. <https://doi.org/10.1093/mnras/stz2947>
- Becker I, Pichara K, Catelan M, Protopapas P, Aguirre C, Nikzat F (2020) Scalable end-to-end recurrent neural network for variable star classification. *Mon Notices R Astron Soc* 493:2981–2995. <https://doi.org/10.1093/mnras/staa350>
- Bertin E, Arnouts S (1996) SExtractor: software for source extraction. *Astron Astrophys Suppl Ser*. <https://doi.org/10.1051/aas:1996164>
- Bhandari D, Murthy C, Pal S (1996) Genetic algorithm with elitist model and its convergence. *Int J Pattern Recognit Artif Intell*. <https://doi.org/10.1142/S0218001496000438>
- Blanton M, Bershadsky M, Abolfathi B, Albareti F, Prieto C, Almeida A, Alonso-Garcia J, Anders F, Anderson S, Andrews B, Aquino-Ortiz E, Aragon-Salamanca A, Argudo-Fernandez M, Armengaud E, Aubourg E, Avila-Reese V, Badenes C, Bailey S, Barger K, Zou H (2017) Sloan digital sky survey iv: mapping the milky way, nearby galaxies, and the distant universe. *Astron J* 154:28
- Bose S, Pal A, SahaRay R (2015) Generalized quadratic discriminant analysis. *Pattern Recognit*. <https://doi.org/10.1016/j.patcog.2015.02.016>
- Cabanac R, De Lapparent V, Hickson P (2002) Classification and redshift estimation by principal component analysis. *Astron Astrophys*. <https://doi.org/10.1051/0004-6361:20020665>
- Cabrera A (1994) Logistic regression analysis in higher education: an applied. *Perspective* 10:225–256
- Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system, pp 785–794. <https://doi.org/10.1145/2939672.2939785>
- De Jong K, Fogel D, Schwefel H-P (1997) A history of evolutionary computation *Handb Evolut Comput* A2.3:1–12
- Deng W, Liu H, Xu J, Zhao H, Song Y (2020) An improved quantum-inspired differential evolution algorithm for deep belief network. *IEEE Trans Instrum Meas*. <https://doi.org/10.1109/TIM.2020.2983233>
- Deng W, Xu J, Zhao H, Song Y (2020) A novel gate resource allocation method using improved pso-based qea. *IEEE Trans Intell Transp Syst*. <https://doi.org/10.1109/TITS.2020.3025796>
- Deng W, Xu J, Song Y, Zhao H (2020) Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. *Appl Soft Comput*. <https://doi.org/10.1016/j.asoc.2020.106724>
- Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. *Mach Learn* 63:3–42. <https://doi.org/10.1007/s10994-006-6226-1>
- Gunn J, Carr M, Rockosi C, Sekiguchi M, Berry K, Elms B, Haas E, Ivezić Z, Lupton R, Pauls G, Simcoe R, Hirsch R, Sanford D, Wang S, York D, Annis J, Bartozek L, Boroski W, Brinkman J (1998) The sloan digital sky survey photometric camera. *Astron J*. <https://doi.org/10.1086/300645>
- Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with NumPy. *Nature* 585(7825):357–362. <https://doi.org/10.1038/s41586-020-2649-2>

- Ho T (1998) The random subspace method for constructing decision forests. *IEEE Trans Pattern Anal Mach Intell* 20:832–844. <https://doi.org/10.1109/34.709601>
- Jin X, Zhang Y, Zhang J, Zhao Y, Wu X-B, Fan D (2019) Efficient selection of quasar candidates based on optical and infrared photometric data using machine learning. *Mon Notices R Astron Soc* 485:4539–4549. <https://doi.org/10.1093/mnras/stz680>
- Liashchynskiy P, Liashchynskiy P (2019) Grid search, random search, genetic algorithm: a big comparison for nas. [arXiv:1912.06059](https://arxiv.org/abs/1912.06059)
- López M, Sarro L, Solano E, Gutierrez-Sanchez R, Deboscher J (2010) Supervised star classification system for the omc archive https://doi.org/10.1007/978-3-642-11250-8_151
- Microsoft (2020) Batch—cloud-scale job scheduling and compute management. <https://azure.microsoft.com/en-us/services/batch/>. Access 29 May 2020
- Mosteller F, Tukey J (1968) Data analysis, including statistics. In: Lindzey G, Aronson E (eds) *Revised handbook of social psychology*, vol 2. Addison Wesley, pp 80–203
- Pearson K (1900) On lines and planes of closest fit to points in space. *Philos Mag* 2:559–572. <https://doi.org/10.1080/14786440109462720>
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E, Louppe G (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830
- Peng N, Zhang Y, Zhao Y, Wu X-B (2012) Selecting quasar candidates using a support vector machine classification system. *Mon Notices R Astron Soc* 425:2599–2609. <https://doi.org/10.1111/j.1365-2966.2012.21191.x>
- Philip N, Wadadekar Y, Kembhavi A, Kouneiher J (2002) A difference boosting neural network for automated star-galaxy classification. *Astron Astrophys*. <https://doi.org/10.1051/0004-6361:20020219>
- Pławiak P, Acharya UR (2020) Novel deep genetic ensemble of classifiers for arrhythmia detection using ecg signals. *Neural Comput Appl* 32:11137–11161. <https://doi.org/10.1007/s00521-018-03980-2>
- Re M, Valentini G (2012) Ensemble methods: a review. *Adv Mach Learn Data Min Astron* 563–594
- SDSS (2015) Jpeg images on skyserver. <https://www.sdss.org/dr15/imaging/jpg-images-on-skyserver/>. Access 22 Jan 2019
- Sezer OB, Ozbayoglu M, Dogdu E (2017) A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters, *Procedia Computer Science* 114, 473–480, complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS October 30 - November 1, 2017. Chicago, Illinois, USA. <https://doi.org/10.1016/j.procs.2017.09.031>
- Tu L, Wei H, Ai L (2015) Galaxy and quasar classification based on local mean-based k-nearest neighbor method 285–288. <https://doi.org/10.1109/ICEIEC.2015.7284540>
- Viquar M, Basak S, Dasgupta A, Agrawal S, Saha S (2018) Machine learning in astronomy: a case study in quasar-star classification. *Proc IEMIS* 3(2019):827–836. https://doi.org/10.1007/978-981-13-1501-5_72
- White B, Rosenblatt F (1963) Principles of neurodynamics: perceptrons and the theory of brain mechanisms. *Am J Psychol* 76:705. <https://doi.org/10.2307/1419730>
- Zhang Y, Zhao Y, Zheng H (2009) Automated classification of quasars and stars. *Proc Int Astron Union* 5:147–147. <https://doi.org/10.1017/S1743921310006083>
- Zhang Y, Zhao Y, Zheng H, Wu X-B (2013) Classification of quasars and stars by supervised and unsupervised methods. *Proc Int Astron Union* 8:333–334. <https://doi.org/10.1017/S1743921312017176>
- Zhang Y, Zhao Y, Peng N (2011) LS-SVM applied for photometric classification of quasars and stars. In: Evans IN, Accomazzi A, Mink, DJ, Rots AH (eds) *Astronomical data analysis software and systems XX*. Astronomical Society of the Pacific Conference Series, vol 442

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.