A&A, 693, A300 (2025)
https://doi.org/10.1051/0004-6361/202451309

Astronomy & Astrophysics

Optimised sampling of SDSS-IV MaStar spectra for stellar classification using supervised models

R. I. El-Kholy* and Z. M. Hayman

Department of Astronomy, Space Science, and Meteorology, Faculty of Science, Cairo University, Giza 12613, Egypt

Received 29 June 2024 / Accepted 17 December 2024

ABSTRACT

Context. Supervised machine learning models are increasingly being used for solving the problem of stellar classification of spectroscopic data. However, training these models calls for a large number of labelled instances, whereas their collection is usually costly in both time and expertise.

Aims. Active learning (AL) algorithms minimise training dataset sizes by keeping only the most informative instances. This paper explores the application of AL to sampling stellar spectra using data from a highly class-imbalanced dataset.

Methods. We utilised the MaStar Stellar Library from the SDSS DR17, along with its associated stellar parameter catalogue. A preprocessing pipeline that includes feature selection, scaling, and dimensionality reduction was applied to the data. Using different AL algorithms, we iteratively queried instances where the model or committee of models exhibits the highest uncertainty or disagreement, respectively. We assessed the effectiveness of the sampling techniques by comparing several performance metrics of supervised-learning models trained on the queried samples with randomly sampled counterparts. Evaluation metrics included specificity, sensitivity, and the area under the curve. In addition, we used Matthew's correlation coefficient, which accounts for class imbalance. We applied this procedure to the effective temperature, surface gravity, and iron metallicity, separately.

Results. Our results demonstrate the effectiveness of AL algorithms in selecting samples that produce performance metrics that are superior to random sampling and even stratified samples, with fewer training instances.

Conclusions. We find AL is recommended for prioritising instance labelling for astronomical-survey data by experts or crowdsourcing to mitigate the high time cost. Its effectiveness can be further exploited in selecting targets for follow-up observations in automated astronomical surveys.

Key words. methods: data analysis – methods: statistical – techniques: spectroscopic – surveys – stars: general

1. Introduction

Stellar spectra can be divided into seven main spectral classes according to the Harvard scheme of stellar spectral classification. The classes, O, B, A, F, G, K, and M follow a sequence represented by the effective temperature of stellar atmospheres, where the hottest stars belong to class O ($T_{\rm eff} \gtrsim 25\,000$ K) and the coolest belong to class M (2000 K < $T_{\rm eff}$ < 3500 K). Each of these main classes can be further divided into ten sub-classes from 0 to 9, where 0 represents the hottest stars within that class and 9 the coolest. Morgan and Keenan later proposed appending a luminosity class (Ia, Ib, II, III, IV, and V) to the main class and sub-class (e.g. our Sun is classed as G2V). The luminosity class depends on the surface gravity of stars, often represented as $\log g$ in stellar parameters catalogues, where luminous supergiants with the least $\log g$ values belong to class 'Ia' and dwarfs with the largest values belong to class 'V'. The modified system has become known as the MK classification system. A review of stellar spectral classification can be found in Giridhar (2010).

The stellar spectral classification of large numbers of stars is essential to studies of stellar populations and galactic formation history. In the past, stellar spectral classifications have been performed by human experts, who had to visually inspect each of the spectra. With the advancement of computational capabilities and the introduction of machine learning (ML) algorithms, more sophisticated techniques have been applied to classify stellar spectra. Among those are χ^2-minimisation, artificial neural networks (ANN), and principal component analysis (PCA; e.g. Gulati et al. 1994; Singh et al. 1998; Bailer-Jones et al. 1998; Manteiga et al. 2009; Gray & Corbally 2014; Kesseli et al. 2017; Fabbro et al. 2017). With the avalanche of stellar spectroscopy data pouring from telescope surveys, the use of ML algorithms has been increasing and this approach has been proven capable of reducing the error and improving the accuracy of stellar spectral classification (Sharma et al. 2019). However, for any supervised ML algorithm to be applied to the stellar classification problem, a large sample of labelled data has to be collected and curated for the training of the model, which is very costly in terms of both time and expertise. This has always been a limitation related to the use of supervised ML techniques and this is especially prominent in applications of deep learning (DL) frameworks. Attempts to tackle this problem by crowdsourcing the classification have been made, as in the case of the Galaxy Zoo Project (Lintott et al. 2010) which eventually went on to include many other applications[1] and they have indeed been effective to some extent. However, this approach in itself suffers from two limitations: (i) for certain tasks, many non-expert volunteers become uncertain of their answers, which might lead to inaccurate labelling that would eventually reflect in the poor performance of models trained using that data; and (ii) the crowdsourcing process does not resolve the problem of the time-cost completely. Some

* Corresponding author; relkholy@sci.cu.edu.eg

[1] https://www.zooniverse.org/

efforts have been employed to solving the first limitation by a careful curation of the questions and taking the confidence level of the volunteers into account, with some success (Song et al. 2018). However, this can also exacerbate the time-cost issue. Another, more effective approach that can minimise the size of the required training dataset, while keeping fewer high-quality instances for labelling is active learning (AL; Lughofer 2012). The use of AL algorithms has been shown to give favourable results in many astronomical applications, such as stellar population studies, photometric supernova classification, galactic morphology, and anomaly detection for time-domain discoveries (e.g. Solorio et al. 2005; Richards et al. 2011; Ishida et al. 2018, 2021; Walmsley et al. 2019).

In this work, we apply AL algorithms to a set of stellar spectra to study the efficiency of the sampling techniques in selecting instances that are informative and representative of the overall distribution of the data pool and investigate whether the performance of models trained using the selected instances is comparable to that of models trained on randomly-sampled instances or even stratified samples. We used the MaNGA Stellar Library (MaStar; Yan et al. 2019), which is highly imbalanced, from the seventeenth data release (DR17) of the Sloan Digital Sky Surveys (SDSS; Abdurro'uf et al. 2022). We started by applying a preprocessing pipeline to the data. We used random sampling to establish a baseline for comparison. We varied both the initial batch size and the number of additional instances sampled using each algorithm. Supervised ML algorithms were then trained using each sample and their performance on test sets was compared. Several metrics were applied for comparing the performances. This process was implemented for three stellar parameters: effective temperature, surface gravity (in terms of $\log g$), and iron metallicity. Finally, we tested the progression of the performance of spectral classification with an increase in the number of selected instances and demonstrate how AL sampling produces results superior to both random and stratified sampling, even with less than half the sample size.

The paper is structured as follows. In Sect. 2, we give an overview of the spectral dataset used in this work. In Sect. 3 we describe the preprocessing steps applied to the data, illustrate the AL algorithms employed, briefly illustrate each of the supervised learning models used for classification, and define the set of performance metrics used for model assessment. In Sect. 4, we present our results and discuss their potential interpretations. Finally, in Sect. 5, we give our summary and conclusions.

2. Data

In this work, we use the final version of the MaStar library from the SDSS DR17 (Abdurro'uf et al. 2022). MaStar is a large library of high-quality calibration empirical stellar library. The MaStar data were obtained using the Baryon Oscillation Spectroscopic Survey (BOSS) spectrograph (Smee et al. 2013; Drory et al. 2015), the same as the main MaNGA survey, mounted on the Apache Point Observatory 2.5m telescope (Gunn et al. 2006). The same fibre system used by the MaNGA survey was used as well. The targets of the MaStar library were chosen to cover a wide range of parameter space. The MaStar spectra cover a wavelength range of 3622−10 354 Å, with a spectral resolution of $R \sim 1800$. The first release of MaStar has been presented in Yan et al. (2019) and its final version will be detailed in Yan et al. (in prep.).

The empirical spectra are obtained by observing real stars; hence, they are not subject to many of the limitations of synthetic

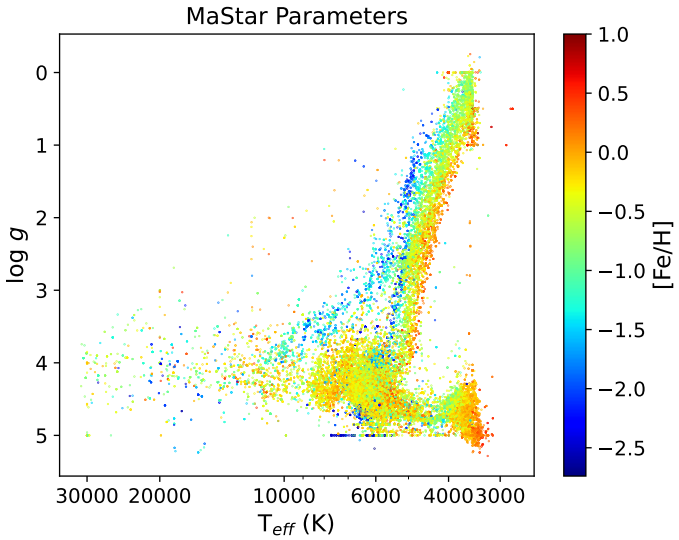

Fig. 1. MaStar library stellar parameter distribution after final quality cuts, colour-coded by the iron metallicity.

spectra produced by theoretical models (Kurucz 2011; Dupree et al. 2016). However, empirical spectral libraries are limited by the wavelength range, spectral resolution, and parameter-space coverage. This makes the high quality and wide coverage of the MaStar library particularly optimal for use in data-driven experiments. In this work, we only used the good-quality visit spectra included in the `mastar_goodspec` file[2]. This library has a flux calibration accuracy of up to 4% (Imig et al. 2022). SDSS DR17 also includes a value-added catalogue (VAC)[3] containing four sets of different stellar parameter measurements. Each measurement uses different methods, the details of which can be found in the respective papers (Chen et al. 2020; Hill et al. 2021, 2022; Imig et al. 2022; Lazarz et al. 2022). A detailed comparison will be presented in Yan et al. (in prep.). The same VAC also includes the median values of these methods (when available and robust), along with the uncertainties of these medians based on the quality assessments of each set of measurements. We rely on these median columns in our current work.

We applied our approach to three stellar parameters: effective temperature ($T_{\rm eff}$), surface gravity ($\log g$), and iron metallicity ([Fe/H]). We only included stars that have a median value available in the VAC for each of these parameters. After dropping unqualified visits, we ended up with 59 085 spectra (visits) of 24 162 unique stars. For this set of spectra, more than 85% have signal-to-noise ratios of S/N > 50, with an overall mean value of about 126. The resulting stellar parameter ranges are as follows:

1. $2800\,{\rm K} \lessapprox T_{\rm eff} \lessapprox 31\,000\,{\rm K}$,
2. $-0.25\,{\rm dex} \lessapprox \log g \lessapprox 5.25\,{\rm dex}$,
3. $-2.75\,{\rm dex} \lessapprox {\rm [Fe/H]} \lessapprox 1.00\,{\rm dex}$.

The final parameter distribution is also shown in Fig. 1. Each of the three parameters was then used separately to classify the spectra into categories according to the ranges shown in Table 1. The resulting class distribution for each parameter is shown in Fig. 2. It is clear that the dataset is highly imbalanced for all three parameters where the imbalance ratio (IR) ranges are as follows:

[2] https://data.sdss.org/sas/dr17/manga/spectro/mastar/v3_1_1/v1_7_7/mastar-goodspec-v3_1_1-v1_7_7.fits.gz
[3] https://data.sdss.org/sas/dr17/manga/spectro/mastar/v3_1_1/v1_7_7/vac/parameters/v2/mastar-goodstars-v3_1_1-v1_7_7-params-v2.fits

Table 1. Classification ranges for separate stellar parameters, where XMP, MP, MR, and XMR correspond to extremely metal poor, metal poor, metal rich, and extremely metal rich, respectively.

	$T_{\rm eff}$ (K)		log g (dex)		[Fe/H] (dex)	
Class	Range	Class	Range	Class	Range
M	<3500	C1	<2.0	XMP	<−2
K	3500–5000	C2	2.0–3.0	MP	(−2)–(−1)
G	5000–6000	C3	3.0–3.5	MR	(−1)–0
F	6000–7500	C4	3.5–4.0	XMR	≥0
A	7500–10 000	C5	4.0–4.5		
B	10 000–25 000	C6	≥4.5		
O	≥25 000				

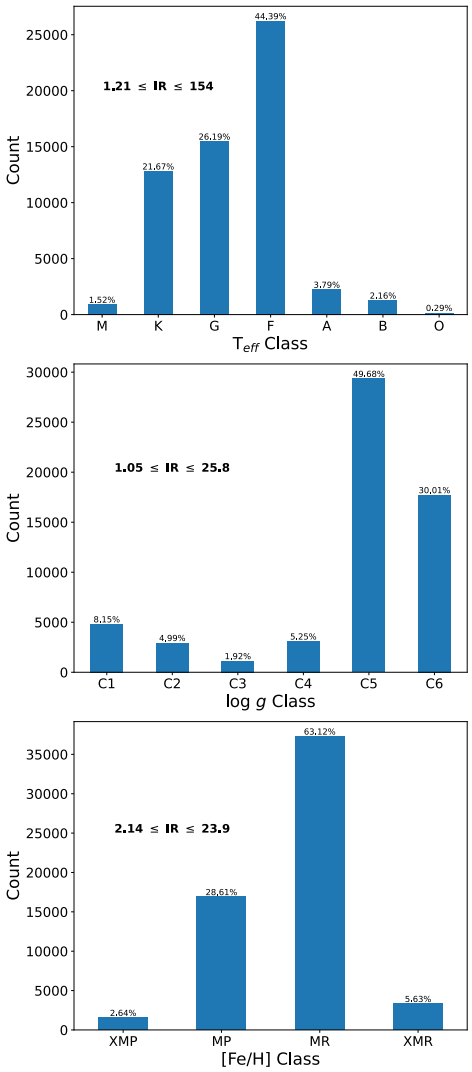

Fig. 2. Class distributions for each of the stellar parameters according to the ranges given in Table 1, demonstrating high imbalance ratios (IR) for all three parameters.

1. $T_{\rm eff}$: 1.21–154,
2. log g: 1.05–25.8,
3. [Fe/H]: 2.14–23.9.

Finally, Fig. 3 shows a sample spectrum for every class of each parameter.

3. Method

In this section, we describe the methods applied in this study. We first describe how we prepared the data for use in Sect. 3.1. Then, we applied the AL algorithms described in Sect. 3.2 to curate training samples. The output was iteratively used to train ML models, as outlined in Sect. 3.3, and the results were compared with a random-sampling benchmark, according to the metrics defined in Sect. 3.4. The pipeline of the study is shown as a flowchart in Fig. 4 and further details of the experiments and steps applied are described in Sect. 3.5. The Python code created for this work has been made available on GitHub[4].

3.1. Preprocessing

Before using any dataset to train an ML model, it has to be suitably prepared. To this end, we applied the following four-step preprocessing scheme:

1. Employ a feature-selection routine adapted from the algorithm used by Brice & Andonie (2019) as a first step toward a dimensionality reduction;
2. Split the dataset into training and testing sets;
3. Use a min-max normalisation to scale each of the selected features; and
4. Apply a PCA to further reduce the dimensionality.

Feature selection is a common approach to dimensionality reduction, where we extract the most relevant set of features to reduce the number of dimensions of the input space. It helps with speeding up the algorithm, while also discarding some of the noise inherent in the data. There is more than one way to achieve this, but here we apply the approach proposed by Brice & Andonie (2019); namely, we picked flux measurements around specific absorption lines. Brice & Andonie (2019) included the H δ (4102 Å) and Ca I (4227 Å) lines as they cover six of the seven main spectral classes. The idea is that the flux intensity of such lines is what determines the spectral class, while the width of the lines is what determines the luminosity class. Thus, it is not enough to include the flux measurement closest to the wavelength of the absorption line in question; a sufficiently wide region needs to be included around the wavelength to account for the line width in addition to the shifting of the spectrum due to radial velocity. We adopted the same procedure in this work, but since our model was applied not only to spectral and luminosity classes, but also to metallicity classes, instead of only using the two lines mentioned above, we included additional lines as listed below:

1. Ca II K (3934 Å) and Ca II H (3968 Å): key indicators in A-type stars, showing strength variances linked to temperature and luminosity effects (Gray 2009);
2. Fe I (4046 Å): often used with the hydrogen line H δ for temperature classification, especially in F-type and later-type stars (Gray 2009);
3. Two spectral lines can help us classify stars from B to M (Brice & Andonie 2019):
   – H δ (4102 Å): present in B-, A-, F-, and G-type stars;
   – Ca I (4227 Å): present in F-, G-, K-, and M-type stars,
4. *G*-band CH (4300 Å): prominent in late-G to K-type stars and is sensitive to surface gravity (Gray 2009);
5. He II (4686 Å): dominates O-type stars (Gray 2009);
6. TiO band (4955 Å): at least one TiO band is needed for M-type classification (Gray & Corbally 2014);

[4] https://github.com/rehamelkholy/StellarAL

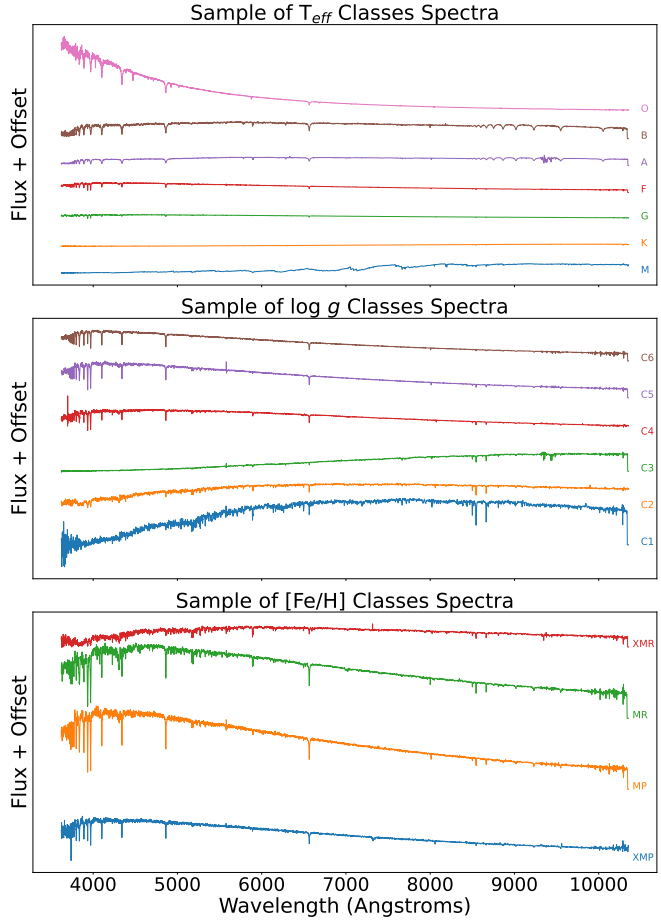

Fig. 3. Sample spectra from different classes for each stellar parameter.

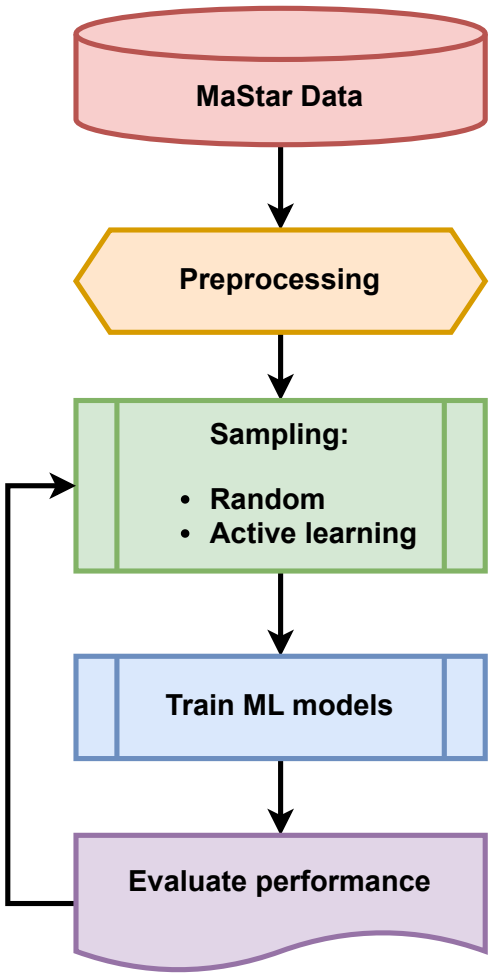

Fig. 4. Flowchart outlining the main steps undertaken in this study.

Table 2. Absorption lines included in the feature selection step, their central wavelengths, and feature ranges in ascending order, with a consistent wavelength spacing of about 0.83 Å.

Line	Central wavelength (Å)	Wavelength range (Å)
Ca II K	3934	3926.45–3940.94
Ca II H	3968	3960.96–3975.58
Fe I	4046	4038.31–4053.22
H δ	4102	4094.49–4109.60
Ca I	4227	4218.91–4234.48
G-band CH	4300	4292.40–4308.24
He II	4686	4677.35–4694.62
TiO band	4955	4945.38–4963.64
Fe II	5018	5008.41–5026.90
Fe I	5269	5258.96–5278.37

7. Fe I (5,269 Å) and Fe II (5018 Å): for metallicity classification (Santos et al. 2004).

Table 2 lists the set of spectral lines used, the central wavelength corresponding to each of them, and the wavelength range included to account for the spectral line in the reduced feature space. The flux measurements from all regions are combined at the end to create one flux array per spectrum. At this preprocessing step, we reduce the feature-space dimensionality from 4563 to 170.

Since the last two preprocessing steps of the scheme outlined above include parameter fitting, the data have to be split first, as only training data can be used in the fitting process in order to prevent data leakage. Accordingly, 10% of the dataset is set aside for testing. Because the dataset is highly imbalanced, stratification during data splitting is necessary to ensure that the evaluation metrics obtained at the testing step accurately reflect the model performance. Moreover, to avoid any ambiguity that might result from multi-label stratification, this step is applied separately to a copy of the entire dataset for each of the three classification parameters: $T_{\rm eff}$, $\log g$, and [Fe/H]. This leaves us with 53 176 samples for training and 5909 for testing.

After splitting the dataset into training and testing sets, each feature is scaled using the equation:

$$f_{i,\rm scaled} = \frac{f_i - f_{i,\rm min}}{f_{i,\rm max} - f_{i,\rm min}}, \quad (1)$$

where f_i is the $i^{\rm th}$ flux measurement, $f_{i,\rm max}$ and $f_{i,\rm min}$ are the corresponding maximum and minimum flux measurements, respectively, and $f_{i,\rm scaled}$ is the scaled flux measurement. This step is crucial to provide a frame-of-reference for the model to compare feature values for different samples. However, as mentioned before, only the training set can be used in determining the minimum and maximum values for each feature. These values are then used to scale the testing set. This process can easily be handled by using the `MinMaxScaler` of

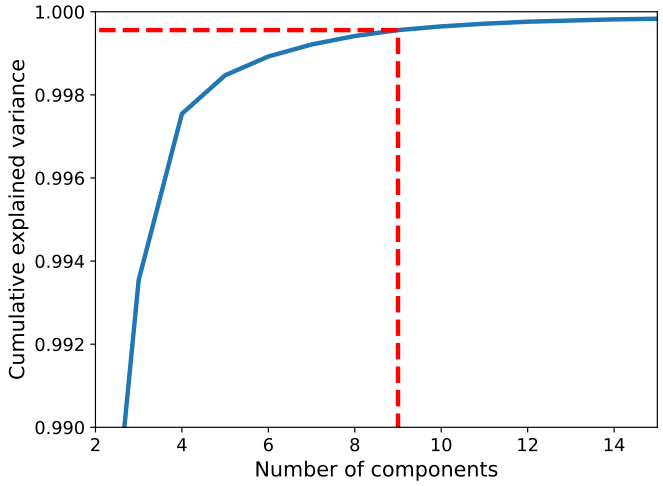

Fig. 5. Cumulative explained variance plotted against the number of PCA components, showing that more than 99.95% of the variance is explained by the first nine components.

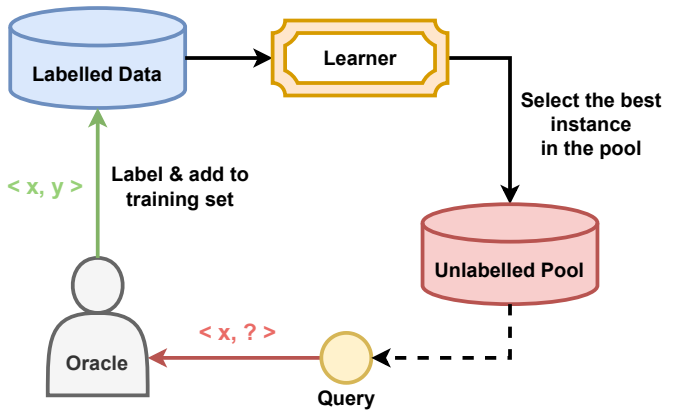

Fig. 6. Illustration of the pool-based active-learning scenario querying the most informative instance from a large pool of unlabelled data.

the `sklearn.preprocessing` module (Pedregosa et al. 2011), where the scaler is first fit by the training set and later used to transform the entire dataset.

Due to the higher computational expense of the AL algorithms described in Sect. 3.2, we had to minimise the number of features further. Thus, we used a PCA, which is a statistical method that defines a linear transform to reduce a dataset to its most essential features (i.e. principal components). These components are ordered according to the variance captured by each of them. Using this transform, an approximation of the dataset can be obtained by a few major components. A thorough description of the PCA method can be found in Ivezić et al. (2020) or Greenacre et al. (2022). In this work, we first applied a PCA to the entire dataset (after applying the feature scaling routine described above) to determine the number of principal components we would include in our model. As shown in Fig. 5, we found that more than 99.95% of the variance in the data is captured by the first nine components. Early trials also indicated that a higher number of features results in an increase in computational cost that cannot be justified by any slight improvement in the model's performance. After the dataset has been split and scaled separately for each parameter, PCA can be applied on the training sets and the resulting approximations are used to map the testing sets as well. This process was practically executed using the `PCA` class from the `sklearn.decomposition` module (Pedregosa et al. 2011).

3.2. Active learning approach

The role of a classification ML model is essentially to generate a mapping between input features and the class labels based on the features and labels of the training dataset. However, for this mapping to be as accurate as possible, large amounts of labelled training instances are required. The labelling process is often very expensive in terms of time and manpower. The collection of such data is currently one of the main challenges in ML applications (Dimitrakakis & Savu-Krohn 2008; Li et al. 2021). The solution to this problem would be to minimise the size of the needed training data while only keeping the most high-quality data. This could be achieved by careful selection of unlabelled instances to later be labelled by an annotator or expert, which is the goal of AL (Huang et al. 2014; Ghahramani et al. 2017).

Overall, AL algorithms can be categorised into three main scenarios: membership query synthesis, stream-based selective sampling, and pool-based AL; the latter is the most well-known of the three and the algorithms we use in this work belong to this category. A detailed discussion of the three different scenarios and the advantages and limitations of each can be found in Settles (2012) or Tharwat & Schenck (2023). The pool-based sampling approach selects instances from an existing pool of unlabelled data based on the active learner evaluation of the informativeness of some or all of the instances in the pool. The selected instance is then annotated by the oracle and added to the labelled training set. This process is iteratively repeated until a criterion is reached, which is usually a maximum number of iterations. Thus, this type of scenario generally includes two adjustable parameters: the initial labelled batch size and the number of additional instances to be queried. Figure 6 illustrates the pool-based sampling approach. In this work, we tested six different sampling strategies that can be divided into two categories. The first category is uncertainty sampling which includes three strategies based on three uncertainty measures: classification uncertainty, classification margin, and classification entropy. The second category is query by committee (QBC), which includes three strategies as well based on three disagreement measures: vote entropy, consensus entropy, and maximum disagreement. We give a brief definition of each strategy below, but a more thorough explanation can be found in Settles (2012).

On the one hand, the uncertainty sampling evaluates each instance in the unlabelled pool and presents the most informative one to be annotated and added to the labelled training set, where the evaluation of instances is based on an uncertainty measure (hence the name). The first measurement we try here is classification uncertainty defined by:

$$U(x) = 1 - P(\hat{x}|x), \tag{2}$$

where x is the instance to be predicted and $\hat{x}$ is the most likely prediction. The strategy selects the instance with the highest uncertainty. For the classification margin strategy, the difference in probability between the first and second most likely classes is calculated according to:

$$M(x) = P(\hat{x}_1|x) - P(\hat{x}_2|x), \tag{3}$$

where $\hat{x}_1$ and $\hat{x}_2$ are the first and second most likely classes, respectively. In this case, the strategy selects the instance with

the smallest margin, since it means that the learner is less decisive about the predicted class. Finally, the classification entropy is calculated using:

$$H(x) = -\sum_k p_k \log(p_k), \tag{4}$$

where p_k is the probability of the sample belonging to the kth class. This is proportional to the average number of guesses that has to be made to find the true class. Thus, the strategy selects the instance with the largest entropy.

On the other hand, QBC strategies are based on having several hypotheses (i.e. classifiers) about the data, and querying the instances based on measures of disagreement between the hypotheses. The first measure we try is vote entropy defined by:

$$E_{\rm vote}(x) = -\sum_y \frac{N(y|x)}{|C|} \log \frac{N(y|x)}{|C|}, \tag{5}$$

where $N(y|x)$ is the number of 'votes' the class y receives for instance x among the hypotheses in committee C, and $|C|$ is the committee size. This strategy selects the instance where $E_{\rm vote}$ is the largest, since it corresponds to the most uniform distribution of votes among classes. It is a 'hard' vote entropy measure; we also tried a 'soft' vote entropy measure referred to as consensus entropy, which accounts for the confidence of each committee member and is defined by:

$$E_{\rm cons}(x) = -\sum_y P(y|x) \log P(y|x), \tag{6}$$

where $P(y|x)$ is the average 'consensus' probability that y is the correct class according to the committee. Finally, the maximum disagreement measure is based on the Kullback-Leibler (KL) divergence (Kullback & Leibler 1951), which is a measure of the difference between two probability distributions. In other words, the disagreement is quantified as the average divergence of each classifier's prediction from that of the consensus C as follows:

$$D(x) = \frac{1}{|C|} \sum_{\theta \in C} {\rm KL}\left(P_\theta(Y|x) \parallel P_C(Y|x)\right), \tag{7}$$

where the KL divergence of committee member θ is defined by:

$${\rm KL}\left(P_\theta(Y|x) \parallel P_C(Y|x)\right) = \sum_y P_\theta(y|x) \log \frac{P_\theta(y|x)}{P_C(y|x)}. \tag{8}$$

As the name suggests, this strategy picks the instance with the maximum disagreement value, $D_{\rm max}$.

In this work, we used each of the AL strategies described above to iteratively sample instances for training supervised ML models. We applied this approach to each of the three stellar parameters separately, taking random sampling as a baseline for comparison. We use the Modular Active Learning framework for Python3 (`modAL`; Danka & Horvath 2018) to implement these strategies directly into our code. The pipeline of the entire experimental steps is detailed in Sect. 3.5.

3.3. Machine learning models

In this work, we used different supervised-learning algorithms and compare their performances according to the metrics described in Sect. 3.4. We applied three ML models: k-nearest neighbours (KNN), random forest (RF; Breiman 2001), and gradient boosting (GB; Friedman 2001); in addition to an ensemble model that combines their outputs. Some ML algorithms were only used in certain experiments, as detailed in Sect. 3.5. In what follows, we briefly introduce each algorithm.

KNN: The k-nearest neighbours is a clustering algorithm based on distance metrics. The standard Euclidean distance is most-commonly chosen as the distance metric measure. KNN can be applied to both regression and classification problems. Since it is one of the simpler algorithms, it offers a robust way to establish a baseline for classification accuracy. At its core, it is based on the assumption that if two data points are nearby each other, they belong to the same class. Then, k is a tunable parameter that represents the number of neighbouring points to be considered, such that the classification of a data point relies on the voting results of the k neighbours that are nearest to it in the multidimensional space.

Random forest: RF (Breiman 2001) is widely used for classification and regression problems because it is fast to train and scales well, while also maintaining competitive performance to other ML algorithms. It is an ensemble method consisting of randomly-generated decision trees. It uses bootstrap sampling techniques, which means that different decision trees are simultaneously trained on different sub-sets of the training data using random sub-sets of the features. Thus, while a decision tree usually overfits, RF is less prone to overfitting as it uses the average of the trees, which ultimately improves classification accuracy.

Gradient boosting: The GB algorithm (Friedman 2001) is a powerful ensemble model that combines multiple decision trees to create a stronger predictive model. In GB, each subsequent tree corrects the errors of the previous one. It optimises a specific objective function, typically a loss function, by minimising it through gradient descent. Overall, GB achieves higher performance and better generalisation with lower time cost than other ensemble learning methods, such as the stochastic forest algorithm and the support vector machine (SVM) of a single model (Zeraatgari et al. 2023).

Voting: This is an ensemble learning technique used in classification and regression tasks. It combines predictions from multiple base classifiers and selects the class label by voting, which leads to improved performance compared to individual classifiers. The voting classifier can be implemented using soft or hard voting. A hard voting classifier chooses the class with the highest frequency of votes, whereas a soft voting one averages the class probabilities across all base classifiers. Different base classifiers can be given different voting weights based on their individual performances. In this work, we combine KNN, RF, and GB in a soft voting classifier, giving RF and GB weights of 2 each and KNN a weight of 1.

3.4. Metrics

In this study, our aim is to compare the performances of different sampling methods, in addition to the performances of ML models. Accuracy is the most basic evaluation metric, and is given by:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{9}$$

where TP, FP, TN, and FN are the numbers of true positives, false positives, true negatives, and false negatives, respectively. However, the accuracy in this case does not give an realistic reflection of the model performance in the case of class imbalance. Hence, a more helpful pair of metrics can be employed for that; namely, sensitivity and specificity. Sensitivity, or the true positive rate (TPR), measures the ability of a model to classify

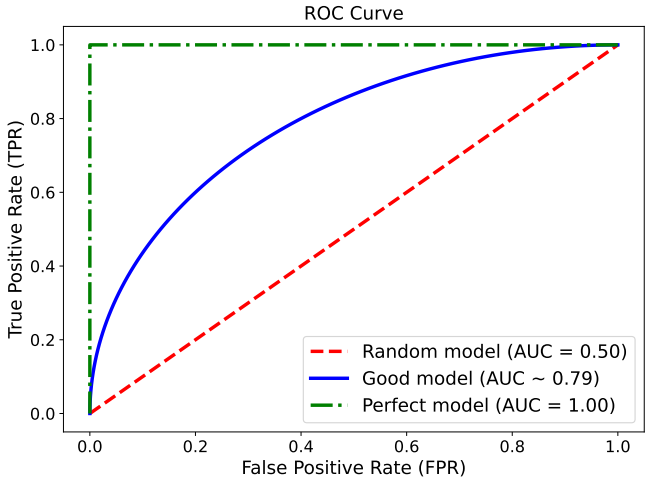

Fig. 7. Examples of different receiver operating characteristic (ROC) for models with different levels of performance, where the legend shows the area under the curve (AUC) value for each model.

positives correctly, given by:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{10}$$

while specificity, or the true negative rate (TNR), measures the ability of a model to classify negatives correctly. and is given by:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{11}$$

It is also conventional to use another metric related to both sensitivity and specificity, which is the area under the curve (AUC) of the receiver operating characteristic (ROC) curve, where TPR is plotted against the false positive rate (FPR) at different threshold values. Here, FPR is given by:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{specificity}. \tag{12}$$

AUC is a measure of the total 2D area under the ROC curve; and is a very good predictor of the overall performance of the classifier, where a baseline random model is expected to have an AUC $\sim$ 0.5 and a perfect model would have an AUC value of 1. Figure 7 shows examples of ROC curves for models with different levels of performance.

For a multi-class imbalanced dataset, sensitivity is of particular interest since it emphasises the ability of the model to correctly identify true positives of minority classes; whereas a model can score high specificity even if it can only classify the majority classes. To take that into account, we use all three metrics to compare the AL algorithms with random sampling. For all three metrics, we calculated the macro value for the metrics; that is, we evaluated the metric for each class separately and took the average as the final metric value. This is an added measure to give the performance of the model on minority classes the same weight as its performance on majority ones.

In addition, we used Matthew's correlation coefficient (MCC) as a fourth metric. It is defined by:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}. \tag{13}$$

Because MCC uses all four elements of the confusion matrix (TP, TN, FP, and FN) in the numerator, it does not get skewed by class imbalance, which makes it a more reliable metric for summarising the overall performance of the model across all classes. MCC ranges from -1 (total disagreement between predicted and true labels) to 1 (perfect prediction), where MCC = 0 indicates a near-random prediction. This makes it a very intuitive metric for understanding the performance of a classifier. A more thorough explanation of each of the chosen metrics can be found in Marsland (2014).

Even though we defined the metrics above in terms of binary classification to provide a clear concept of what they measure, these definitions are easily generalised to fit their application on a multi-class dataset. This is easily handled by making use of the `imblearn.metrics` module (Lemaître et al. 2017) to calculate both the sensitivity and specificity, along with the `sklearn.metrics` module (Pedregosa et al. 2011) for the AUC and MCC.

3.5. Pipeline

In the following, we outline the steps undertaken in the experiments performed in this work. The aim of the first experiment carried out was to compare the performances of models trained on samples queried by different approaches. Since the aim is not to optimise the ML model selected, but the sampling algorithm instead, we started by training each of the models described in Sect. 3.3 on the entire dataset to select the best-performing one. The chosen model was then used throughout the rest of the steps, except for QBC strategies, where we used a committee of three learners; namely, KNN, RF, and GB, all initialised using the same batch. We then evaluated each sampling strategy, separating the uncertainty sampling strategies from QBC strategies and using random sampling as a baseline in both cases. We varied the initial batch size, taking care to initialise all strategies using the same batch. We began by evaluating the initial model on the testing set before iteratively augmenting the training sample by querying the data pool and retraining the model. The model performance was reevaluated after each five queries using each of the metrics listed in Sect. 3.4. We performed 20 runs as described to account for the performance variance for some strategies and calculate the mean performance metrics for all runs. This experiment was repeated for each of the stellar parameters separately.

As a final step to statistically assess the impact of AL strategies on performance, we aggregated the results across all runs and batch sizes for uncertainty-sampling and QBC strategies separately and apply the Wilcoxon signed-rank test (Wilcoxon 1992) on each metric, repeating the steps for each stellar parameter. We set a predefined significance threshold of $p < 0.05$. In addition, we calculated the associated values of the Cohen's d impact index (Cohen 2013)[5], which is often used to quantify the practical significance of the difference between two means. Values of 0.2–0.5 indicate low significance, 0.5–0.8 indicate medium significance, and greater than 0.8 indicate large significance.

The aim of the second experiment is to assess the performance progress of a model trained on an AL-sampled set with the increase of the number of additional instances. This experiment is only carried out on effective temperature. We picked the highest-performing strategy from the first experiment to use with the same best-performing model chosen before and only

[5] Description based upon print version of record.

Table 3. Performance scores for different ML models when evaluated on the testing set after being trained on the entire training set for the three stellar parameters.

Parameter	Model	AUC	MCC	Sensitivity	Specificity
$T_{\rm eff}$	KNN	0.947	0.876	0.787	0.982
	RF	0.989	0.907	0.859	0.987
	GB	0.955	0.836	0.766	0.977
	Voting	0.989	0.891	0.825	0.985
$\log g$	KNN	0.884	0.591	0.629	0.929
	RF	0.949	0.683	0.707	0.942
	GB	0.928	0.627	0.647	0.932
	Voting	0.948	0.673	0.683	0.939
Fe/H	KNN	0.920	0.772	0.711	0.939
	RF	0.979	0.856	0.787	0.961
	GB	0.975	0.836	0.776	0.956
	Voting	0.980	0.845	0.781	0.958

sensitivity is used to assess the models in this experiment. For the sake of comparison, we used three baseline training sets:

1. the whole initial training set,
2. a random sample of 10% of the initial training set,
3. a stratified sample of 10% of the initial training set.

For both (2) and (3), 20 different samples were used to account for performance variance and the mean results are calculated along with their standard deviations. Finally, we ran the AL-sampling method five times (averaged at the end) to sample 5% of the initial training data pool and retrain and reevaluate the model every five queries.

4. Results and discussion

In this section, we present and discuss the results of the experiments carried out in this study. We begin by evaluating the performance of different ML models on the testing set after being trained on the entire training set. The results of these steps are shown in Table 3. It can be seen that RF outperforms the other three across all metrics, particularly sensitivity, for both effective temperature and surface gravity. For the iron metallicity, the voting models has a slightly better AUC score compared to RF, but the latter still scores higher on the other three metrics. It is also worth noting that the computational cost of the voting model is almost three times that of RF, since it is training the three member learners under the hood. Hence, we decided to use RF for all three parameters moving forward, except when using QBC strategies as mentioned before.

We can also see from Table 3 that the KNN model has the lowest overall scores across all three stellar parameters. This is because we do not perform any hyperparameter tuning for the ML models used, but rather keep the default values of the `Scikit-learn` library (Pedregosa et al. 2011). In the case of KNN, the most important hyperparameter is the number of neighbours, k, which has a default value of 5. Early trials with hyperparameter grid searches for some of the ML models used in this study indicated that a value of 18 achieves better performance scores for the KNN model. This suggests that the overpopulation of the feature space with majority classes makes it necessary to increase the number of neighbours taken into account to correctly classify instances that belong to minority classes.

Figure 8 shows the performance scores of different single-learner AL sampling strategies along with a random-sampling baseline for different initial batch sizes when applied to effective temperature. It can be seen that for all metrics, at least one uncertainty sampling strategy outperforms random sampling. In particular, all AL strategies significantly outperform the random baseline on sensitivity scores across all initial batch sizes. The best-performing strategy is clearly the classification margin strategy, but only for an initial batch size ($n_{\rm init}$) = 20. In comparing the first and second columns of sub-plots in the figure, we may notice that the performance scores, after adding 50 AL-sampled instances to an initial randomly chosen set of 20, is always higher than the corresponding scores when using an initial set of 100 randomly chosen instances. This demonstrates the effectiveness of AL sampling in achieving better scores with fewer training instances. We can also see that the larger the size of the initial training batch, the less pronounced the improvement in performance due to AL sampling. However, the improvement in sensitivity for all AL strategies is still evident, even with an initial batch of 500 instances. This offers a contrast with the plateauing of random-sampling sensitivity at the same initial batch size. Of course, the emphasis on sensitivity scores is due to the highly-imbalanced nature of the dataset, which makes sensitivity scores more representative of a model's ability to correctly identify minority classes.

Figure A.1 shows the performance scores of different QBC disagreement sampling strategies, along with a random sampling baseline adapted for QBC learning as well, for different initial batch sizes applied to $T_{\rm eff}$. We can see that some of the scores in this case are higher than those of the single RF model shown in Fig. 8. However, when we take into account the fact that the computational cost of a QBC model is almost linearly dependent on the number of learners in the committee (three in this case), the corresponding improvement in performance is diminished. Comparing the scores of QBC disagreement strategies with the random approach yields similar results to non-committee uncertainty sampling comparison with random sampling. Nevertheless, it is worth reiterating that AL strategies score higher than random sampling on sensitivity, even after increasing the initial batch size. It is clear that the vote-entropy strategy outperforms all others across all metrics. If computational resources were no issue, higher scores could be obtained via pre-calibration of single committee members. However, this again raises the need for labelled instances to perform such a calibration prior to training.

The performance scores of uncertainty sampling strategies compared with random sampling applied to surface gravity with different initial batch sizes are shown in Fig. A.2. Random sampling performance is comparable to AL uncertainty sampling strategies for most metrics. However, the classification margin strategy outperforms all the others across all metrics, even with an increasing initial batch size. The differences in MCC and sensitivity scores between margin sampling and random sampling particularly highlights the effectiveness of the strategy in mitigating the impact of class imbalance. Finally, the effect of increasing the initial batch size is similar to that discussed above. Figure A.3 shows scores of QBC sampling applied to surface gravity with increasing initial batch sizes. Unlike the case of $T_{\rm eff}$, there is no noticeable improvement in performance compared to uncertainty sampling strategies shown in Fig. A.2. Again, this might be due to the use of committee members without prior hyperparameter tuning, which would require an initial labelled set for performing grid searches. It is worth noting that the vote-entropy strategy outperforms all others. The difference is particularly significant for MCC and sensitivity.

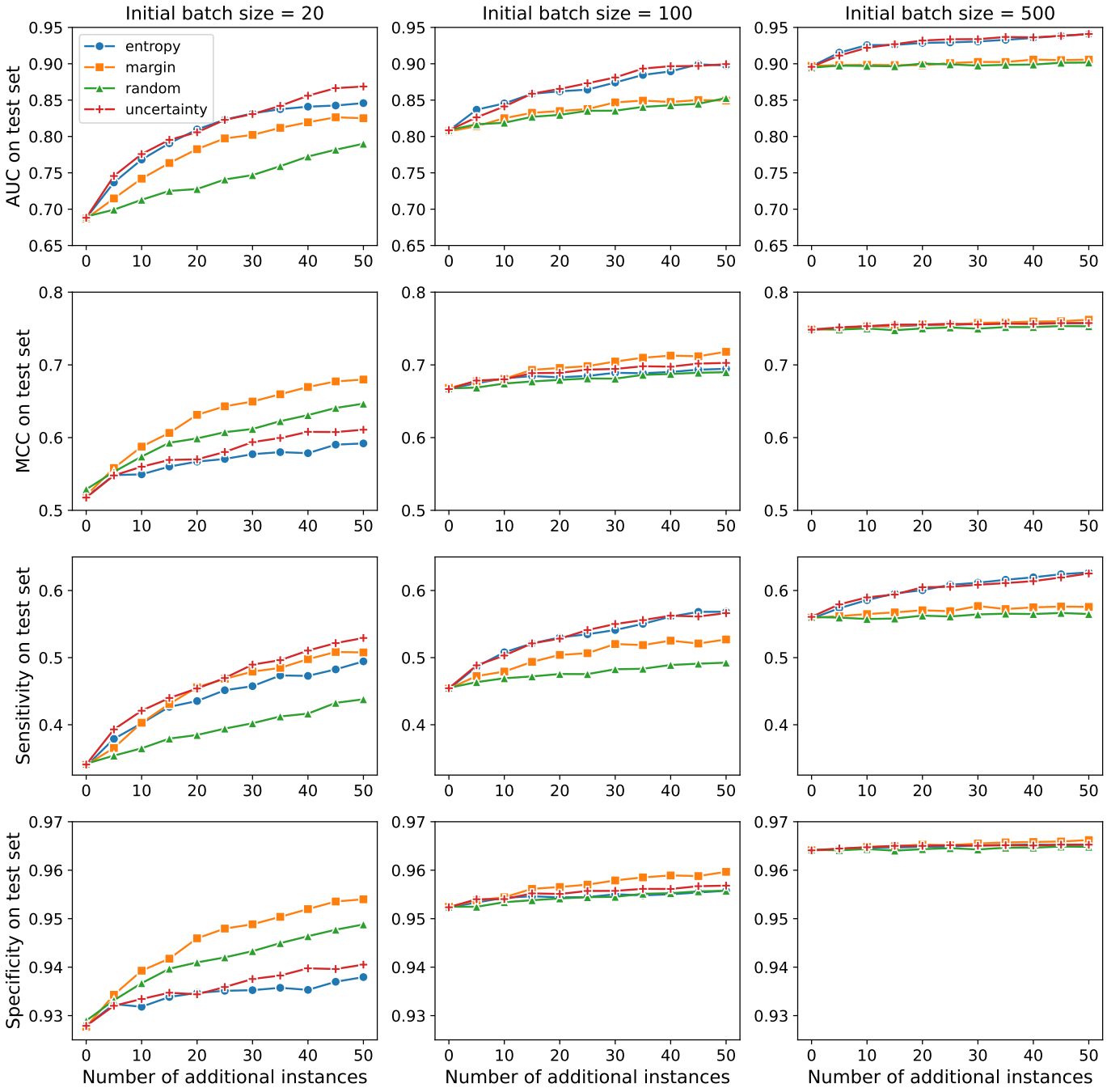

Fig. 8. Performance scores for uncertainty sampling strategies along with random sampling for different initial batch sizes applied to $T_{\rm eff}$.

In Fig. A.4, we show the performance scores of uncertainty sampling strategies compared with random sampling when applied to iron metallicity with different initial batch sizes. Compared to the sensitivity scores when the model is applied to both $T_{\rm eff}$ and $\log g$, we can see that the improvement with the increase in additional instances is much lower for most strategies in the case of [Fe/H]. This could be due to the existence of chemically peculiar (CP) stars in the minority classes of the testing set. CP stars mainly belong to spectral classes A and B (Ghazaryan et al. 2018, 2019); and their existence in the testing set will necessitate a higher number of training instance before the model can start to correctly identify rare classes. This is evident when we look at the improvement of sensitivity scores when we start with a batch size of 500 instances. It is worth noting that this impact will not be so pronounced if we choose the 'micro' instead of 'macro' values for the metrics (see Sect. 3.4). The figure also shows that uncertainty-margin sampling still outperforms all others in MCC, sensitivity, and specificity, even with the increase of $n_{\rm init}$.

The last plots for the first part of this study are shown in Fig. A.5; namely, the performance scores of QBC sampling strategies applied to iron metallicity with increasing initial batch sizes. When compared to Fig. A.4, we can see that QBC does not offer any improvement upon single-learner uncertainty sampling in any metric, regardless of the additional computational cost of training a QBC model. Some models show an erratic behaviour at lower numbers of additional instances with $n_{\rm init}$ = 20. This could be due to the KNN member of the committee, which

Table 4. Statistical impact of different AL strategies on classification metrics compared to random sampling, where the p-value is calculated for the Wilcoxon's signed-rank test.

Strategy	AUC		MCC		Sensitivity		Specificity	
	p-value	Cohen's d	p-value	Cohen's d	p-value	Cohen's d	p-value	Cohen's d
$T_{\rm eff}$: Non-committee strategies								
Entropy	2×10^{-6}	2.09	2.15×10^{-2}	−0.53	2×10^{-6}	1.90	1.21×10^{-3}	−0.75
Margin	5.86×10^{-4}	0.81	3.6×10^{-5}	1.19	4×10^{-6}	1.49	8.2×10^{-5}	1.11
Uncertainty	2×10^{-6}	2.27	6.22×10^{-1}	−0.17	2×10^{-6}	2.22	5.32×10^{2}	−0.46
$T_{\rm eff}$: Committee strategies								
Consensus	2×10^{-6}	1.67	8.51×10^{-4}	−0.87	2×10^{-6}	2.02	3.22×10^{-4}	−0.91
Disagreement	6×10^{-6}	1.33	7.30×10^{-3}	−0.52	3.22×10^{-4}	0.92	7.30×10^{-3}	−0.59
Vote	4×10^{-6}	1.52	1.05×10^{-4}	0.99	2×10^{-6}	2.22	3.6×10^{-5}	1.04
$\log g$: Non-committee strategies								
Entropy	3.62×10^{-2}	−0.53	6.48×10^{-1}	−0.11	1.89×10^{-1}	0.34	6.74×10^{-1}	−0.24
Margin	2.15×10^{-2}	0.56	3.95×10^{-4}	1.18	1.05×10^{-4}	1.16	3.15×10^{-3}	0.85
Uncertainty	7.56×10^{-1}	−0.13	2.02×10^{-1}	0.26	1.53×10^{-2}	0.67	3.88×10^{-1}	0.10
$\log g$: Committee strategies								
Consensus	2.31×10^{-1}	−0.30	2.02×10^{-1}	−0.45	2.45×10^{-1}	0.22	3.62×10^{-2}	−0.65
Disagreement	3.49×10^{-1}	−0.11	1.89×10^{-1}	−0.25	6.74×10^{-1}	−0.03	2.15×10^{-2}	−0.488
Vote	4.8×10^{-5}	1.36	4.8×10^{-5}	1.41	4×10^{-6}	1.73	1.21×10^{-3}	1.04
[Fe/H]: Non-committee strategies								
Entropy	1.72×10^{-2}	−0.50	5.83×10^{-2}	−0.61	5.32×10^{-2}	−0.31	2.15×10^{-2}	−0.75
Margin	2.45×10^{-1}	−0.25	2×10^{-6}	2.28	8.51×10^{-4}	0.58	4×10^{-6}	1.61
Uncertainty	7.30×10^{-3}	−0.46	1.33×10^{-1}	−0.49	8.70×10^{-2}	−0.26	3.62×10^{-2}	−0.63
[Fe/H]: Committee strategies								
Consensus	7.30×10^{-3}	−0.52	2.10×10^{-4}	−1.09	8.26×10^{-2}	−0.42	2.61×10^{-4}	−0.94
Disagreement	4.09×10^{-1}	−0.14	3.15×10^{-3}	−0.75	7.01×10^{-1}	−0.15	1.53×10^{-2}	−0.62
Vote	2.94×10^{-1}	−0.16	9.27×10^{-1}	0.05	2.61×10^{-1}	0.18	2.94×10^{-1}	0.17

requires a larger class population to achieve stabilisation (particularly in cases where CP stars are included). This is particularly evident for random sampling because it is less likely to populate neighbourhoods of rare classes with fewer instances. It is worth mentioning that the vote-entropy strategy still outperforms all others across all metrics and initial batch sizes.

Taking a closer look at Figs. A.1 and A.3 together, we can see an 'elbow' feature in the first column ($n_{\rm init}$ = 20) across all metrics at 15 additional instances, which is equivalent to a total training set size of 35 instances. However, this feature does not appear in Figs. 8 or A.2, indicating that it can be attributed to the use of QBC. The most likely reason is that each member of the committee requires the feature space to be populated in a different way to improve performance. This translates to requiring a higher number of training instances to improve the collective performance of the committee. This is even corroborated by the fact that we do not see a similar feature in Fig. A.5 because the number of iron metallicity classes is lower (4 compared to 7 and 6 for $T_{\rm eff}$ and $\log g$, respectively).

Finally, Table 4 summarises the statistics describing the impact of the different AL strategies on the classification performance, each compared to random sampling. For each of the metrics listed in Sect. 3.4, we calculated the p-value of the Wilcoxon's signed-rank test and Cohen's d impact index. A p-value of less than 0.05 indicates a statistically significant impact, and a positive Cohen's d shows superiority of the AL strategy over random sampling and vice versa. These results were calculated by aggregating across all runs and initial batch sizes for each parameter-strategy-metric combination. In addition to validating our findings based on Figs. 8 and A.1–A.5, the following points are worth noting:

- The classification margin strategy results in highly significant ($p \leq 8.51 \times 10^{-4}$) improvements in MCC and sensitivity across all parameters.
- The vote-entropy strategy shows highly significant ($p \leq 1.21 \times 10^{-3}$) improvements across all metrics for both $T_{\rm eff}$ and $\log g$ classification.
- All AL uncertainty strategies show higher sensitivity results for both $T_{\rm eff}$ and $\log g$ classification.

In the second part of this study, we used an RF model along with the uncertainty-margin sampling strategy to track the progress of the sensitivity score on the test set with the increase in the number of instances queried by the AL algorithm. The results, for the instance where this model was applied to effective temperature, are shown in Fig. 9. We also included three baselines to reference for comparison, corresponding to RF models trained on three different sets: the whole training pool (100%), a random sample of 10%, and a stratified sample of 10% as well. For the AL strategy, we used only ten instances as a random initial batch in this experiment. Based on these results, on one hand, it seems

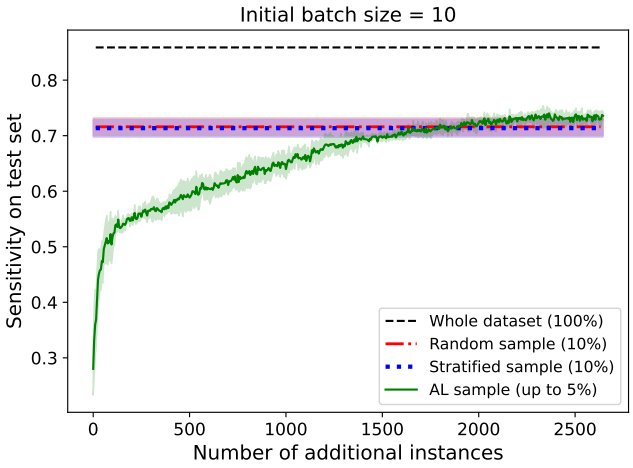

Fig. 9. Progress of the sensitivity score on the test set with the increase in the number of additional instances queried, using an RF model trained on a sample selected by the uncertainty-margin strategy with an initial batch size of ten instances applied to $T_{\rm eff}$, with several baselines drawn for reference showing the score using the whole training set pool, a random sample of 10%, and a stratified sample of 10% as well.

that stratification does not add any improvement over random sampling. This demonstrates that a sampling strategy more effective than stratification is needed to achieve higher performance scores with fewer training instances and less computational cost, even disregarding the fact that stratified sampling requires the entire data pool to be labelled prior to instance selection. On the other hand, it is clear that the AL approach outperforms both samples with only half the training-set size. We can also see that the variance in the AL approach sensitivity starts to increase after the first score jump at around 100 instances. This is because the feature space of the training sample is widening but has not yet accumulated enough instances to cover the finer details of each class. However, when the training sample reaches a size of around 1500 instances, we can see the variance starting to significantly decrease. In spite of this, the sensitivity score of the uncertainty-margin algorithm is still trending upwards to the end of the curve. This indicates that further improvement can be safely expected when we increase the number of sampled instances even further, when more computational resources are available.

Other AL algorithms and ML models can be utilised to perform the above experiments as regression problems, rather than classification ones, if we wish to estimate the parameters associated with each spectrum instead of classifying it. At the moment, our computational resources do not permit us to perform the study in this format. However, this work is meant as a proof-of-concept for the effectiveness of the AL approach, in general, in curating training sets for ML models.

5. Conclusions

The results shown in this paper demonstrate the effectiveness of AL in curating training sets for supervised ML models with the objective of achieving the best possible stellar spectral classification performance while reducing labelling costs in terms of time and expertise. Compared to classical classification approaches, there are several interesting conclusions. They are as follows:

1. AL algorithms significantly improve the performance of stellar spectral classification compared to random or stratified sampling methods by iteratively selecting the most informative instances to annotate.
2. AL reduces the size of the labelled training set required for achieving the same performance as random sampling, making it more cost-effective and efficient.
3. AL algorithms are more robust against class imbalance, which is often the case in stellar spectra datasets, consequently ensuring that rarer stellar classes would be represented adequately and, thus, classified correctly.
4. AL sampling strategies are scalable and can be practically used on large datasets, indicating that it can be integrated into stellar survey data processing pipelines.
5. Models trained on samples curated using AL methods exhibit better generalisation results with fewer instances, which is evident when evaluated on unseen testing data. This makes them more reliable in real-survey applications.

Therefore, the AL approach for automating stellar spectra data curation and classification is feasible, accurate, and cost-effective. Based on the findings of this study, we recommend the integration of AL algorithms in citizen science projects to accelerate the annotation process even further. They can also be used in automated astronomical surveys to optimise the selection of spectra for follow-up observations and analysis. Future works will be aimed at: (1) adapting the approach used here for multi-label classification in order to further minimise the amount of training data needed; (2) investigating the percentage of data required to achieve the same performance scores obtained by using the entire dataset; (3) merging data from different surveys to use in curating a comprehensive training sample using AL and making it publicly available to use for automated stellar classification in future surveys; and (4) evaluating AL algorithms available for regression problems to leverage them in curating pipelines for estimating stellar atmospheric parameters. All of the above is contingent on increasing the availability of computational resources.

Acknowledgements. To prepare the code required for performing this work, we used each of the following open-source Python (Van Rossum 2020) libraries: `NumPy` (Harris et al. 2020), `pandas` (Pandas Development Team 2024); (McKinney 2010), `Matplotlib` (Hunter 2007), `Scikit-learn` (Pedregosa et al. 2011), `Imbalanced-learn` (Lemaître et al. 2017), `Astropy` (Robitaille et al. 2013; Price-Whelan et al. 2018, 2022), and `modAL` (Danka & Horvath 2018). In this work, we have used the SDSS database extensively. 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 website is `www.sdss4.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, Center for Astrophysics | Harvard & Smithsonian, the Chilean Participation Group, the French Participation Group, 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, Observatário 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. We are grateful to the anonymous referee for their insightful comments and suggestions, which have significantly improved the quality and clarity of this manuscript.

References

Abdurro'uf, Accetta, K., Aerts, C., et al. 2022, ApJS, 259, 35
Bailer-Jones, C. A. L., Irwin, M., & Von Hippel, T. 1998, MNRAS, 298, 361
Breiman, L. 2001, Mach. Learn., 45, 5
Brice, M. J., & Andonie, R. 2019, AJ, 158, 188
Chen, Y.-P., Yan, R., Maraston, C., et al. 2020, ApJ, 899, 62
Cohen, J. 2013, Statistical Power Analysis for the Behavioral Sciences (Burlington: Elsevier Science)
Danka, T., & Horvath, P. 2018, arXiv e-prints [arXiv:1805.00979]
Dimitrakakis, C., & Savu-Krohn, C. 2008, Cost-Minimising Strategies for Data Labelling: Optimal Stopping and Active Learning (Springer Berlin Heidelberg), 96
Drory, N., MacDonald, N., Bershady, M. A., et al. 2015, AJ, 149, 77
Dupree, A. K., Avrett, E. H., & Kurucz, R. L. 2016, ApJ, 821, L7
Fabbro, S., Venn, K. A., O'Briain, T., et al. 2017, MNRAS, 475, 2978
Friedman, J. H. 2001, Ann. Stat., 29, 1189
Ghahramani, Z., Gal, Y., & Islam, R. 2017, Deep Bayesian Active Learning with Image Data
Ghazaryan, S., Alecian, G., & Hakobyan, A. A. 2018, MNRAS, 480, 2953
Ghazaryan, S., Alecian, G., & Hakobyan, A. A. 2019, MNRAS, 487, 5922
Giridhar, S. 2010, Spectral Classification: Old and Contemporary (Berlin, Heidelberg: Springer), 165
Gray, R. O. 2009, Stellar Spectral Classification, eds. C. J. Corbally, & A. J. Burgasser, Princeton series in astrophysics (Princeton: Princeton University Press)
Gray, R. O., & Corbally, C. J. 2014, AJ, 147, 80
Greenacre, M., Groenen, P. J. F., Hastie, T., et al. 2022, Nat. Rev. Methods Primers, 2, 100
Gulati, R. K., Gupta, R., Gothoskar, P., & Khobragade, S. 1994, ApJ, 426, 340
Gunn, J. E., Siegmund, W. A., Mannery, E. J., et al. 2006, AJ, 131, 2332
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357
Hill, L., Thomas, D., Maraston, C., et al. 2021, MNRAS, 509, 4308
Hill, L., Thomas, D., Maraston, C., et al. 2022, MNRAS, 517, 4275
Huang, S.-J., Jin, R., & Zhou, Z.-H. 2014, IEEE Trans. Pattern Anal. Mach. Intell., 36, 1936
Hunter, J. D. 2007, Comput. Sci. Eng., 9, 90
Imig, J., Holtzman, J. A., Yan, R., et al. 2022, AJ, 163, 56
Ishida, E. E. O., Beck, R., González-Gaitán, S., et al. 2018, MNRAS, 483, 2
Ishida, E. E. O., Kornilov, M. V., Malanchev, K. L., et al. 2021, A&A, 650, A195
Ivezić, Ž., Connolly, A. J., VanderPlas, J. T., & Gray, A. 2020, Statistics, Data Mining, and Machine Learning in Astronomy, eds. A. J. Connolly, J. VanderPlas, A. Gray, & J. T. Vanderplas (Princeton: Princeton University Press)
Kesseli, A. Y., West, A. A., Veyette, M., et al. 2017, ApJS, 230, 16
Kullback, S., & Leibler, R. A. 1951, Ann. Math. Statist., 22, 79
Kurucz, R. L. 2011, Can. J. Phys., 89, 417
Lazarz, D., Yan, R., Wilhelm, R., et al. 2022, A&A, 668, A21
Lemaître, G., Nogueira, F., & Aridas, C. K. 2017, JMLR, 18, 1
Li, K., Li, G., Wang, Y., et al. 2021, in 2021 IEEE 37th International Conference on Data Engineering (ICDE) (IEEE)
Lintott, C., Schawinski, K., Bamford, S., et al. 2010, MNRAS, 410, 166
Lughofer, E. 2012, Pattern Recognit., 45, 884
Manteiga, M., Carricajo, I., Rodríguez, A., Dafonte, C., & Arcay, B. 2009, AJ, 137, 3245
Marsland, S. 2014, Machine Learning: An Algorithmic Perspective (London: Chapman and Hall/CRC)
McKinney, W. 2010, in Proceedings of the 9th Python in Science Conference, SciPy (SciPy)
Pandas Development Team 2024, https://doi.org/10.5281/ZENODO.3509134
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, JMLR, 12, 2825
Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., et al. 2018, AJ, 156, 123
Price-Whelan, A. M., Lim, P. L., Earl, N., et al. 2022, ApJ, 935, 167
Richards, J. W., Homrighausen, D., Freeman, P. E., Schafer, C. M., & Poznanski, D. 2011, MNRAS, 419, 1121
Robitaille, T. P., Tollerud, E. J., Greenfield, P., et al. 2013, A&A, 558, A33
Santos, N. C., Israelian, G., & Mayor, M. 2004, A&A, 415, 1153
Settles, B. 2012, Active Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning (San Rafael, USA: Morgan & Claypool), 18
Sharma, K., Kembhavi, A., Kembhavi, A., et al. 2019, MNRAS, 491, 2280
Singh, H. P., Gulati, R. K., & Gupta, R. 1998, MNRAS, 295, 312
Smee, S. A., Gunn, J. E., Uomoto, A., et al. 2013, AJ, 146, 32
Solorio, T., Fuentes, O., Terlevich, R., & Terlevich, E. 2005, MNRAS, 363, 543
Song, J., Wang, H., Gao, Y., & An, B. 2018, KBS, 159, 244
Tharwat, A., & Schenck, W. 2023, Mathematics, 11, 820
Van Rossum, G. 2020, The Python Library Reference, release 3.8.2 (USA: Python Software Foundation)
Walmsley, M., Smith, L., Lintott, C., et al. 2019, MNRAS, 491, 1554
Wilcoxon, F. 1992, Individual Comparisons by Ranking Methods (New York: Springer), 196–202
Yan, R., Chen, Y., Lazarz, D., et al. 2019, ApJ, 883, 175
Zeraatgari, F. Z., Hafezianzadeh, F., Zhang, Y., et al. 2023, MNRAS, 527, 4677

Appendix A: Additional figures

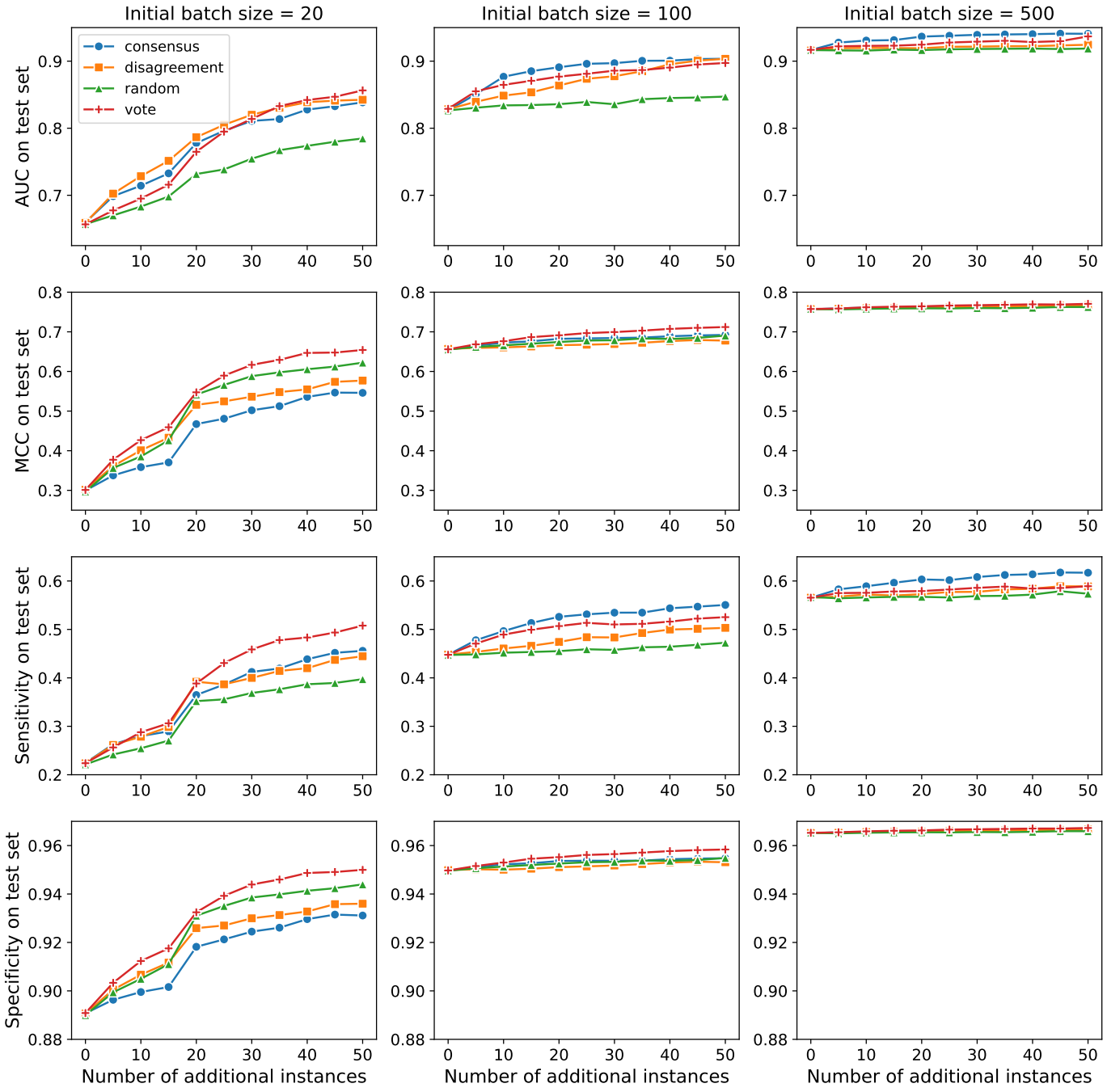

Fig. A.1. Performance scores for QBC disagreement sampling strategies along with random sampling for different initial batch sizes applied to $T_{\rm eff}$.

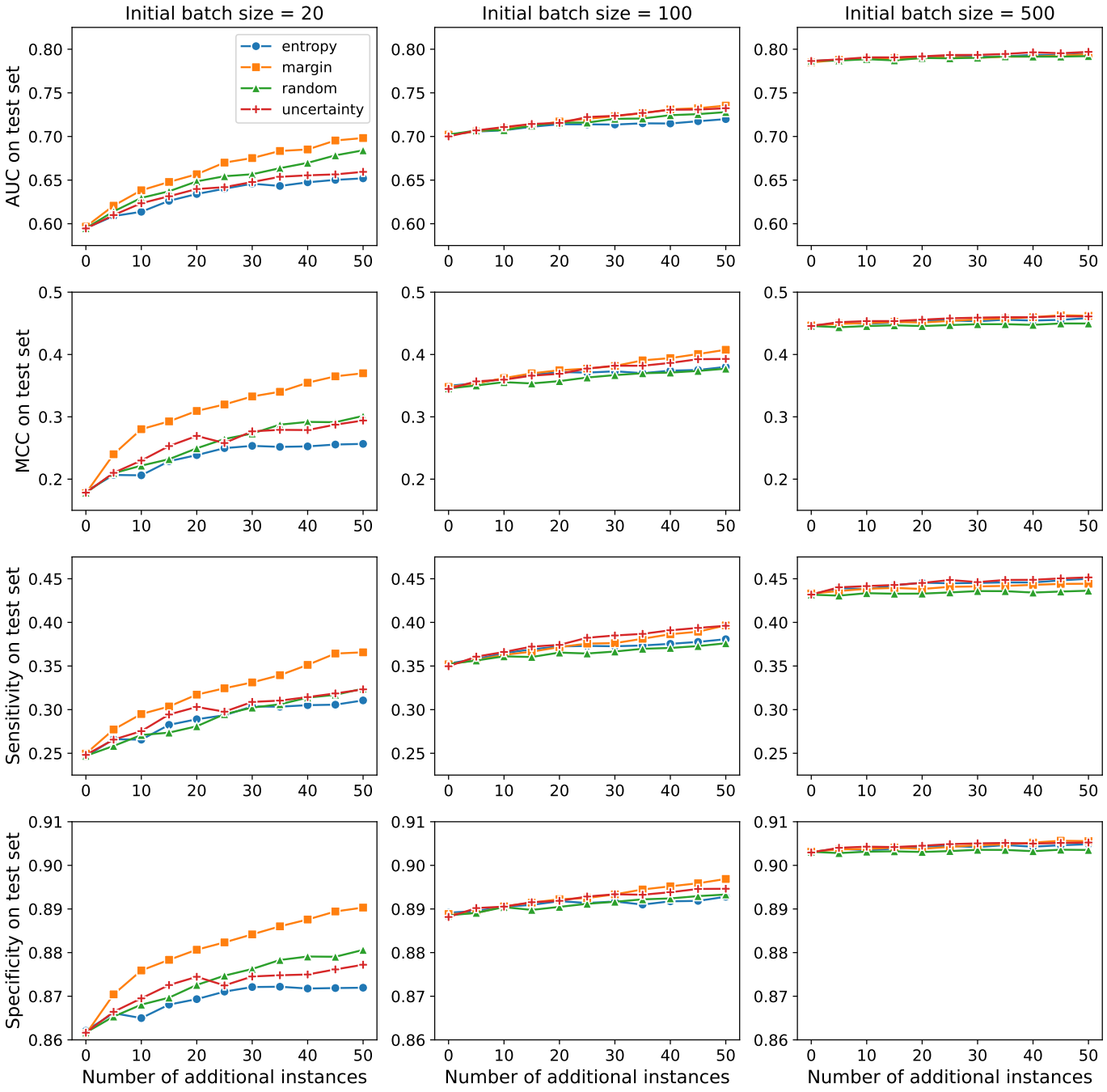

Fig. A.2. Performance scores for uncertainty sampling strategies along with random sampling for different initial batch sizes applied to $\log g$.

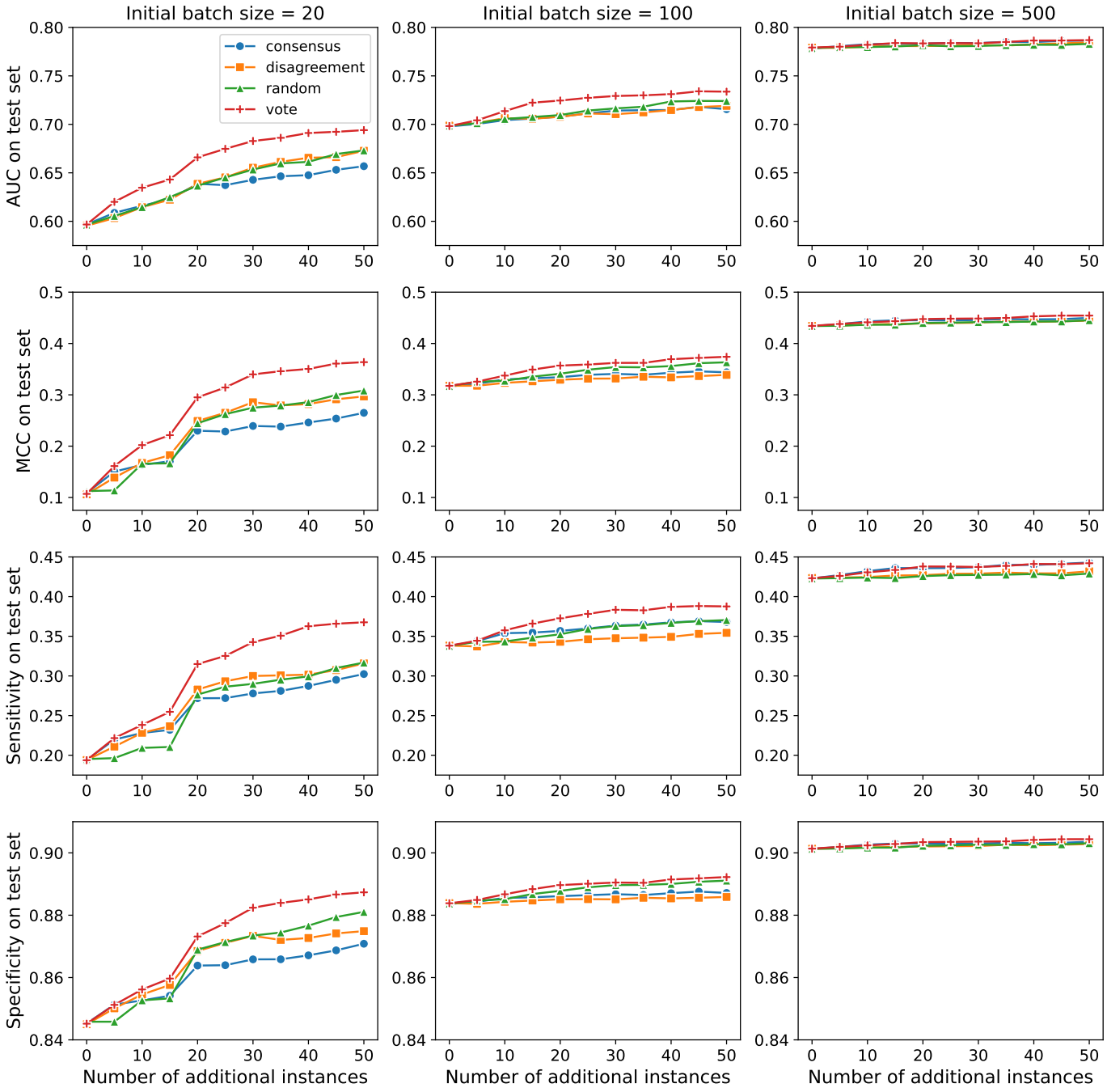

Fig. A.3. Performance scores for QBC disagreement sampling strategies along with random sampling for different initial batch sizes applied to $\log g$.

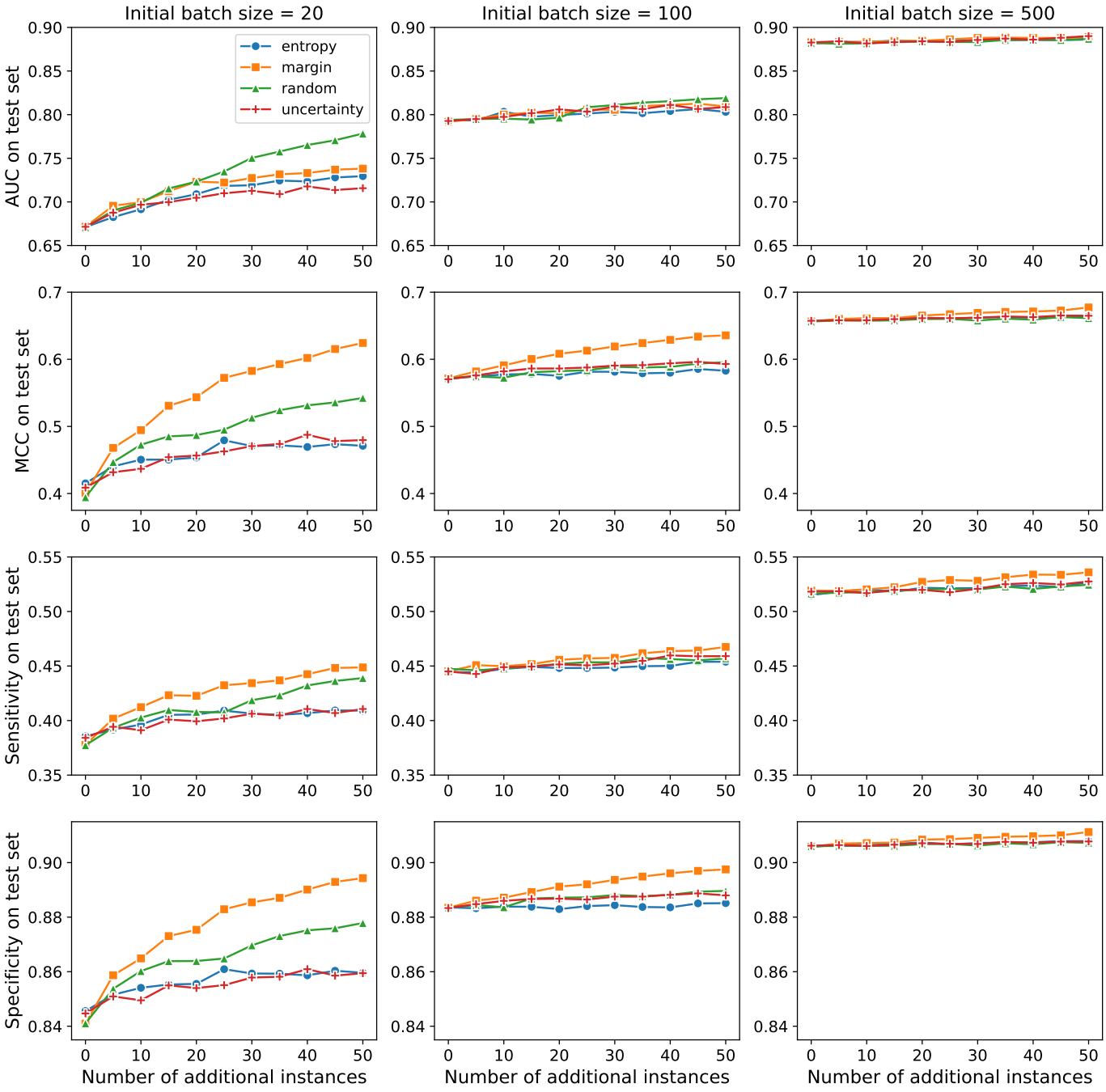

Fig. A.4. Performance scores for uncertainty sampling strategies along with random sampling for different initial batch sizes applied to [Fe/H].

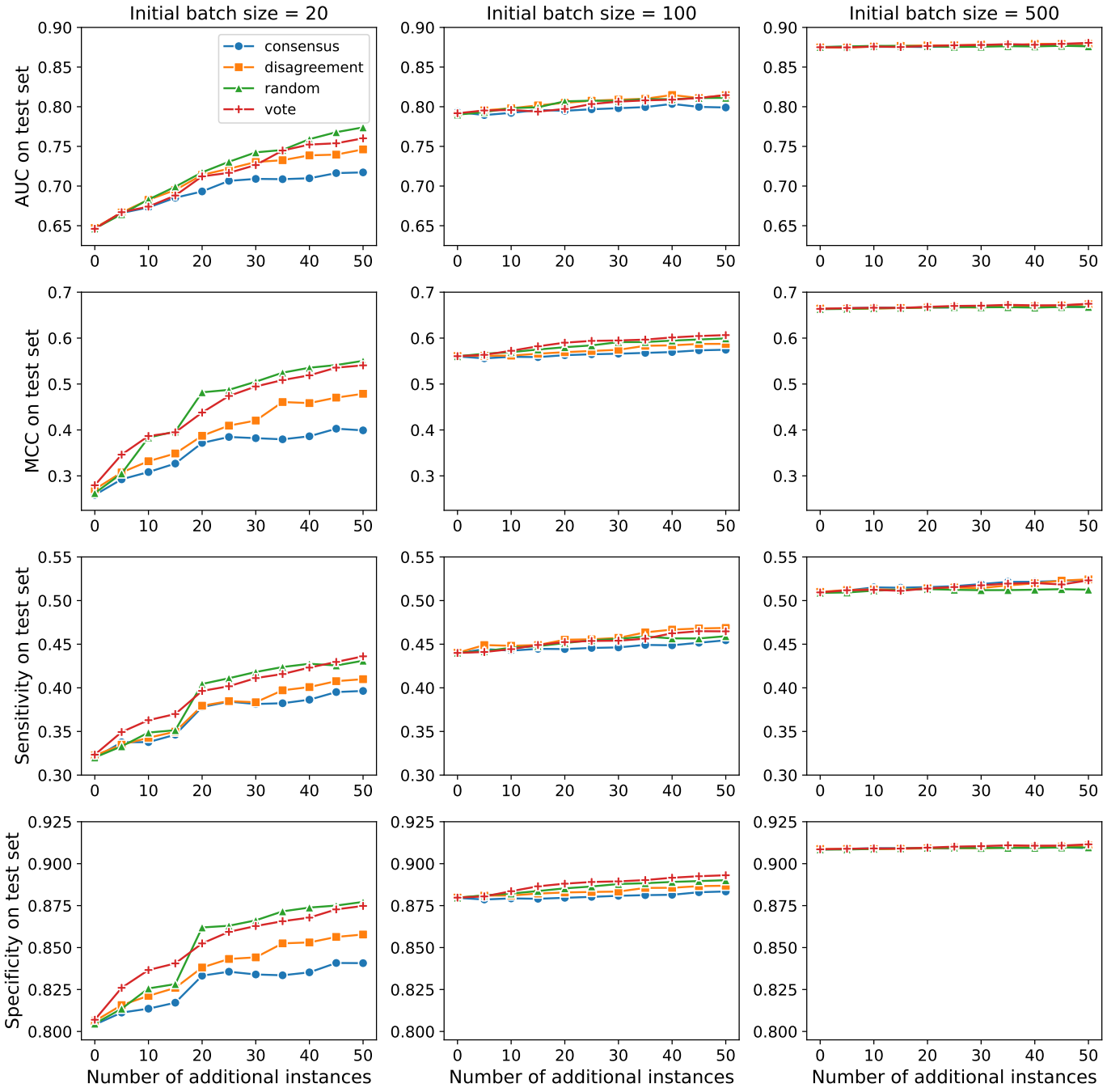

Fig. A.5. Performance scores for QBC disagreement sampling strategies along with random sampling for different initial batch sizes applied to [Fe/H].