Full length article

Light curve classification with `DistClassiPy`: A new distance-based classifier

S. Chaini [a,b,*], A. Mahabal [c,d], A. Kembhavi [e], F.B. Bianco [a,f,g,h]

[a] Department of Physics and Astronomy, University of Delaware, Newark, DE 19716, USA
[b] Department of Physics, Indian Institute of Science Education and Research, Bhopal 462066, India
[c] Division of Physics, Mathematics and Astronomy, California Institute of Technology, Pasadena, CA 91125, USA
[d] Center for Data Driven Discovery, California Institute of Technology, Pasadena, CA 91125, USA
[e] Inter University Centre for Astronomy and Astrophysics (IUCAA), Pune 411007, India
[f] Joseph R. Biden, Jr. School of Public Policy and Administration, University of Delaware, DE 19716, USA
[g] University of Delaware, Data Science Institute, Newark, DE 19713, USA
[h] Vera C. Rubin Observatory, Tucson, AZ 85719, USA

ARTICLE INFO

ABSTRACT

The rise of synoptic sky surveys has ushered in an era of big data in time-domain astronomy, making data science and machine learning essential tools for studying celestial objects. While tree-based models (*e.g.* Random Forests) and deep learning models dominate the field, we explore the use of different distance metrics to aid in the classification of astrophysical objects. We developed `DistClassiPy`, a new distance metric based classifier. The direct use of distance metrics is unexplored in time-domain astronomy, but distance-based methods can help make classification more interpretable and decrease computational costs.

In particular, we applied `DistClassiPy` to classify light curves of variable stars, comparing the distances between objects of different classes. Using 18 distance metrics on a catalog of 6,000 variable stars across 10 classes, we demonstrate classification and dimensionality reduction. Our classifier meets state-of-the-art performance but has lower computational requirements and improved interpretability. Additionally, `DistClassiPy` can be tailored to specific objects by identifying the most effective distance metric for that classification.

To facilitate broader applications within and beyond astronomy, we have made `DistClassiPy` open-source and available at https://pypi.org/project/distclassipy/.

1. Introduction

Over the last few decades, time-domain astronomy has experienced rapid growth. This growth has been driven by the advent of large-scale sky surveys like the Sloan Digital Sky Survey (SDSS; York et al., 2000), the Catalina Real-Time Transient Survey (CRTS; Djorgovski et al., 2011) and the Zwicky Transient Facility (ZTF; Bellm et al., 2019) (see Djorgovski et al. 2013 for a comprehensive list of surveys), and accompanied by advances in computing power and data storage. Observing billions of astronomical objects over time allows us to detect changes in the night sky that were once impossible to see. The Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST; Ivezić et al., 2019), will observe over 37 billion objects over its 10-year lifespan. However, this opportunity for new discoveries comes with a data-intensive challenge. Manual classification of all objects is impossible, so we need machine learning methods to automate the classification and identification of objects of interest.

Machine learning refers to a class of computer algorithms in which the computer learns patterns within the data, eliminating the need for explicit manual programming. Machine learning models can be deployed for a variety of cases — for *e.g.*, clustering (unsupervised learning) or for classification and regression tasks (supervised learning). In the context of large datasets and the need for automation, it has become essential in modern astronomy. A critical task in time-domain astronomy is classifying astronomical objects based on how their brightness changes with time (called light curves), which is well suited to be performed by machine learning algorithms (see for example Eyer and Blake 2002, Thiebaut et al. 2002 and Mahabal et al. 2008; for more recent work see Cabral et al. 2020 and Förster et al. 2021). Because of the irregular sampling of astrophysical light curves, we generally extract features (such as statistical properties of the data, best-fit parameters to models, etc.) from the light curves, which are then fed to a classifier model. However, as the number of features increase, prediction becomes difficult and computationally expensive,
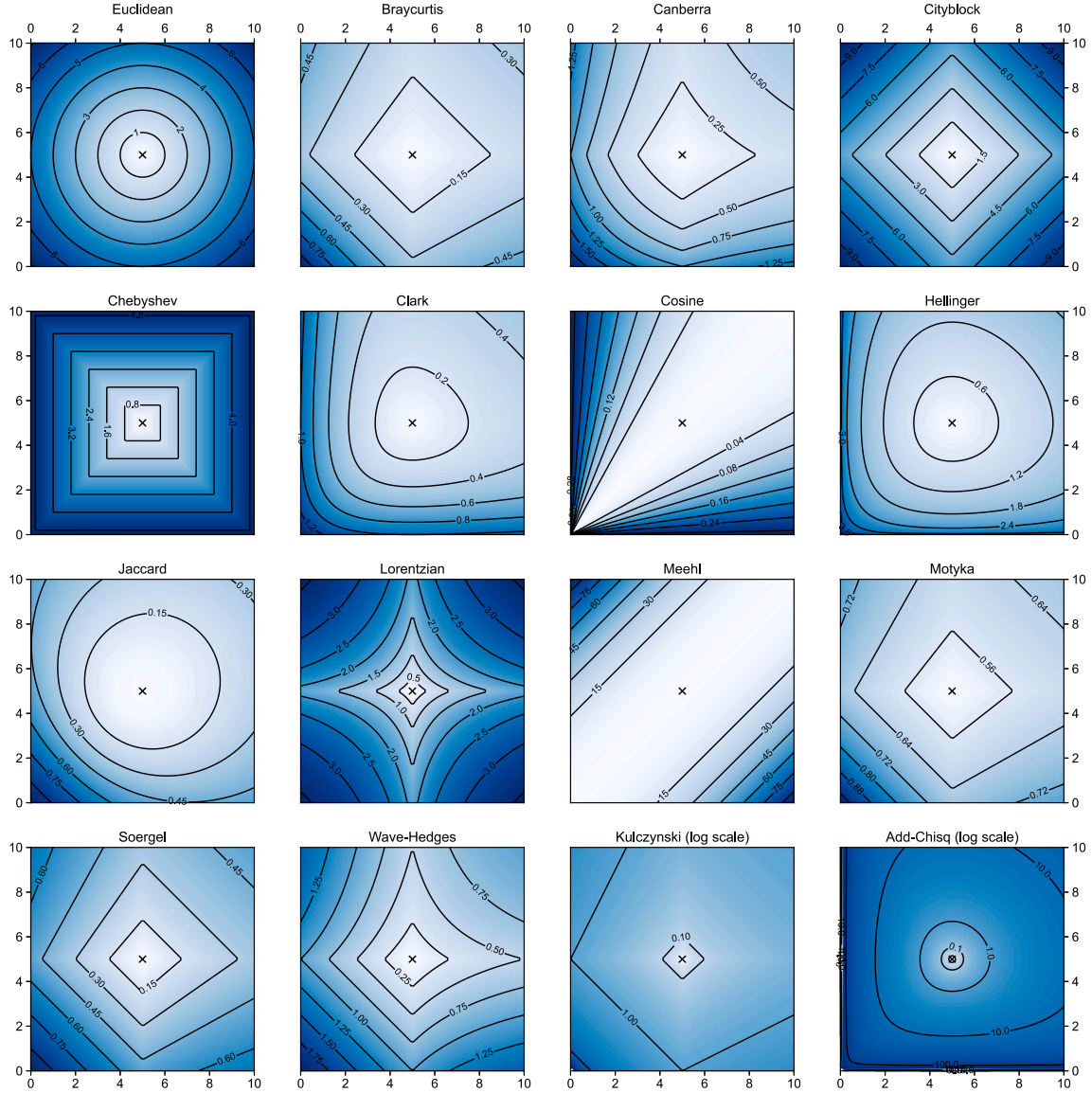

Fig. 1. A visualization of 16 (of the 18) distance metrics used throughout this work. Each subplot shows the equidistant loci measuring the distance from the central point $(5, 5)$. The color background denotes the distance values, with labeled contours. Contours differ for each subplot, as the range of values the distance can take varies by metric. To aid readability, we use a log-scale for the last two metrics — Kulczynski and Additive ChiSq due to high-power elements (in the metric definition) compressing the distance scale. The Correlation and Maryland Bridge metrics are not visualized here as they require vector inputs, and not 2-dimensional data points (see Appendix A).

and performance typically drops (Bishop, 2006), making dimensionality reduction a requirement. Furthermore, the use of more features may lead to data abstraction, and impair the interpretability of a model - resulting in the model behaving like a black-box.

Dimensionality reduction involves mapping the higher-dimensional data into a lower-dimensional space while ensuring the low-dimensional representation retains the properties of the original data for prediction.

In a physical space, a distance is a scalar quantity that tells us how far away two objects are from each other. In a feature space, the "distance" is inverse to the similarity of the objects. A distance metric[1] is the mathematical function or algorithm used to measure the distance between two points in a space. Using the appropriate metric and features, we expect that the distance between objects of

the same class should be smaller than between objects from different classes. These distances can then be used to separate and classify light curves.

In this paper, we compare the effectiveness of different distance metrics applied for dimensionality reduction and classification of light curves. Using a wealth of distances defined in statistics and mathematics (Cha, 2007; Deza and Deza, 2013; Tschopp and Hernandez-Rivera, 2017), we compiled a list of 18 distance metrics designed for data analysis, that have been defined in the appendix, and visualized in Fig. 1. We use these metrics to compare classification performance for three classification problems, described in Section 5. We also identify the most important features for each distance metric and classification task, and analyze the impact of limiting the feature space to these top features, thus reducing dimensionality and computational cost. The use of different distance metrics for dimensionality reduction and subsequent classification is an approach that, to our knowledge, has not been explored in time-domain astronomy before.

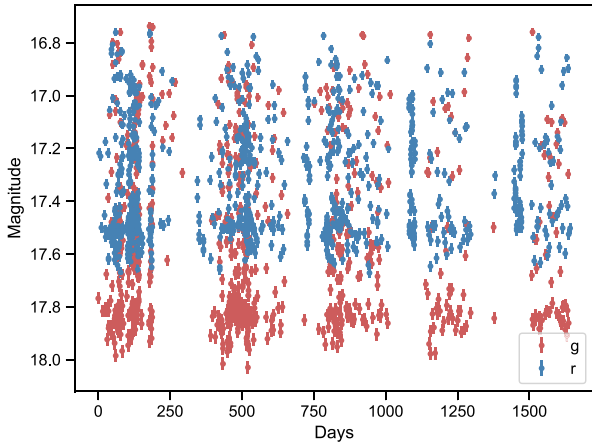[1] Also referred to as a distance measure in literature.

Fig. 2. An example light curve of a RR Lyrae Type ab (RA = 2.93, Dec = 44.62, period = 0.55 days), taken from the ZTF DR15. Each light curve consists of a series of magnitude (brightness) values as a function of time, which in our case is for the two filters — *g* and *r*. The sampling of ZTF is sparse, as common for ground-based surveys, with visible gaps due to seasonality, and, although this is a periodic variable, the periodicity is not obvious because of the sampling.

We show that our classifier, `DistClassiPy`, when used natively out of the box, offers similar performance to other models commonly used to classify light curves, but with lower computational costs. In addition, our model can be tailored according to scientific interests by knowledgeably selecting the distance and features to use based on a scientific goal and data characteristics, enhancing both the computational and performance properties of the model. This paper is accompanied by a Python package, `DistClassiPy`,[2] which is our distance metric classifier built on top of `scikit-learn` (Pedregosa et al., 2011), in addition to all code required to reproduce our results in this paper.[3] We introduce some of the scientific and technical background in Section 2, describe our data in Section 3, discuss our approach to dimensionality reduction in Section 4, then our approach to perform and evaluate the performance on our classification tasks (Section 5). We describe our results in Section 6 and conclude in Section 7.

2. Distances in machine learning

The concept of distance is intuitive. It tells us about the degree of proximity between two objects — the shorter the distance, the closer the objects. However, the distance depends on the path followed to join two objects. Because of this, we can have different types of distances, each calculated differently. Let us first define this mathematically:

Definition 2.1. The distance d between two points, in a set X, is a function $d : X \times X \to [0, \infty)$ that gives a distance between each pair of points in that set such that, for all $x, y, z \in X$, the following properties hold:

(1) $d(x, y) = 0 \iff x = y$ (identity of indiscernibles)
(2) $d(x, y) = d(y, x)$ (symmetry)
(3) $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

An example of a distance is the Euclidean distance, commonly used for measuring the distance between physical objects (the "as the crow flies" distance). A detailed discussion of distance metrics, along with the definition of each of the 18 metrics we used, are included in the appendix, Appendices A and B, respectively. Of these 18 distances, 16 are visualized in a 2-dimensional space in Fig. 1.

Distance metrics see their use in a variety of supervised and unsupervised machine learning algorithms (*e.g.*, *k*-Nearest Neighbors — Abu Alfeilat et al. 2019, Nayak et al. 2022 —, Kernel Density Estimation — He et al. 2013 —, hierarchical clustering — Murtagh and Contreras 2012).

In many machine learning tasks, we do not feed the data directly to a machine learning model. Instead, we extract or engineer *features* from the data and transform them into a new space, to which we will refer hereafter as the "feature space". The dimensionality of the feature space is determined by the number of features, denoted as n. The feature space itself exists as a real coordinate space, represented as $\mathbb{R}^n$. In this n-dimensional feature space, we can now define a variety of distance metrics: any function complying with Definition 2.1 is a valid distance metric. When equipped with a distance metric, the feature space takes on the characteristics of a metric space (as defined in Definition A.1). Since the distance value is always a positive real number, we can easily compare this value for different points even when dealing with high-dimensional data.

3. Data

3.1. Catalog and raw light curves

Our dataset consists of light curves of variable stars from the Zwicky Transient Facility (ZTF) Data Release 15 (DR15).[4] ZTF is a robotic synoptic facility located at the Palomar Observatory. It scans the sky, monitoring objects of magnitude,[5] $r <= 20.6$ (Bellm et al., 2019). ZTF observes in three wavelength bandpasses, *g*, *r* and *i* in the optical wavelength regime. However, the *i* band has a lower cadence as well as shallower observations, and so we restrict our data to the *g* and *r* bands. In our dataset, there are on average 382 points in the *g* band and 674 points in the *r* band for each light curve, which are collected over an average of 4.42 years (minimum 0.61 years, maximum 4.64 years, median 4.47 years).

The original catalog (Chen et al., 2020) consists of 781,602 identified variable stars in 10 classes (see Table 1) that represent stellar objects of different nature and that display different observational properties in the time domain. Examples of light curves in our dataset are shown in Fig. 2. We will briefly return to the topic of variable stars classification in Section 5; for a comprehensive review of variable stars classification and characterization, see Eyer and Mowlavi (2008). The number of objects in each class in Chen et al. (2020) is heavily imbalanced, with the catalog containing 369,707 Eclipsing W Ursae Majoris (EW) variables, but only 1,262 Cepheid (CEP) variables. Models trained on an imbalanced dataset have a hidden bias due to the relative frequencies of occurrence which teaches the model that predictions on minority classes carry a significant risk thus impacting the performance on these classes (Krawczyk, 2016). Thus, we select a random set of 1,000 objects of each class from Chen et al. (2020)'s dataset to develop our model. Our raw dataset consists of 10,000 light curves (1,000 for each of the 10 classes listed in Table 1). After cleaning the data and dropping outliers (as described in Section 3.3), we are left with 558 objects from each of the ten classes.

In addition to the above data, we also used an entirely "hidden" set of ~500 new objects for each class as a final test set. The results of these final performance tests are discussed in Section 7.

3.2. Feature extraction

Since the ZTF is a ground-based telescope, the light curves obtained are unevenly sampled, sparse, noisy, and heteroskedastic. Objects

---

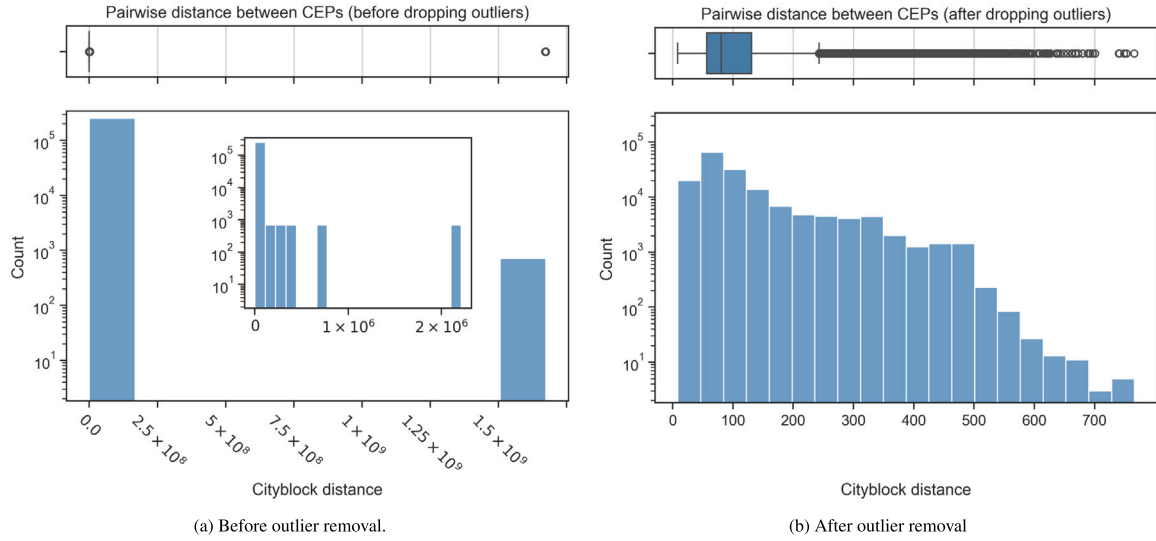(a) Before outlier removal.   (b) After outlier removal

Fig. 3. The distribution of pairwise Cityblock distances (d_{CB}) between all 558 Cepheid variable stars in our final dataset (class CEP): 155,403 unique pairs of CEP contribute one point each to this distribution. The left and right panels show the distribution of distances before and after outlier removal respectively. The top panels show a box and whiskers plot of the same, where the median is marked by a vertical line, the interquartile range by the box, the 10th and 90th percentile by the whiskers. All points beyond these percentile values (our definition of outlier, see Section 3.3) are plotted individually. In the left plots (before outlier removal) the original distribution spreads out to $d_{CB} > 10^9$ but its extremely sparse past $d_{CB} \geq 10^5$ (see insert which zooms into the $d_{CB} < 10^6$ region, and notice how in the top plot the box and whiskers are indistinguishable). However, outlier removal leads to a much more compact distribution of pair-wise distances (right) where the distances are contained between $0 < d_{CB} < 1,000$. Note that, as the distribution changes after the first cut, the functional definition of outliers does not, thus the tail of the distribution is plotted with individual points in the right-side figure.

from the same class have different noise levels and different magnitudes based on their distance from the Earth. This extrinsic diversity within a class makes the direct comparison of light curves difficult. Our scheme for feature extraction is domain-driven: we select features that measure expected behaviors of variable stars (*e.g.*, periodicity).

To extract features from the light curves, we use the `lc_classifier` module (Jainaga et al., 2021) in Python. While the context of the astrophysical target may be informative (*e.g.*, its coordinates to indicate if it is likely to be a Galactic or extra-galactic object), all of `lc_classifier`'s features are based only on the light curve data. Most features (53) are calculated separately for the *g* and *r* passband, while some features (8) are calculated jointly from the *g* and *r* observations. This gives us a total of 114 features for every light curve in our dataset.

A detailed description of all the features is provided in Sánchez-Sáez et al. (2021, see Table 2). Most features are based on common light curve statistics (*e.g.* amplitude — based on the difference of highest and lowest magnitudes), while some are based on parameters obtained after fitting a model to the light curve (*e.g.* multiband period — the period is obtained by fitting a periodogram).

3.3. Data cleaning

We remove objects from our dataset for which feature extraction failed for one or more features. For most classes, this removes 3%–8% of the objects. However, we find that the classes Mira, CEP, and SR have a higher fit failure rate for some features like η (Ratio of mean of square of successive mag differences to light curve variance; refer Kim et al. (2014, Table 3)) and the Mexican Hat Power Spectra (Arévalo et al., 2012), thus leaving us with only 640 Miras, 713 CEPs, and 741 SRs. To keep the dataset balanced, we once again randomly drop objects from each class such that each class has exactly 600 objects. However, we do *not* rebalance the hidden set of data — this is to obtain a more representative assessment of our model's performance.

Chen et al. (2020) state that their catalog does not constitute a robust classification, thus we expect our labels to be noisy and that there might be outliers in the dataset.

To assess the presence of outliers we calculate the pair-wise distances with the Cityblock metric in the 114-dimensional feature space

Table 1

The abbreviation code used along with the full class name for the 10 classes of variable stars used to test our distance metric classifier in this work, along with the original data size for that class in Chen et al. (2020), and the size after we resampled our data. Light curves for these classes were obtained from the Zwicky Transient Facility Data Release 15.

Abbreviation	Class Name	Original class size	Final class size
BYDra	BY Draconis	84,697	.
CEP	Cepheid	1610	.
DSCT	Delta Scuti	16,709	.
EA	Eclipsing Algol	49,943	.
EW	Eclipsing W Ursae Majoris	369,707	558
Mira	Mira	11,879	.
RR	RR Lyrae (Type ab)	32,518	.
RRc	RR Lyrae Type c	13,875	.
RSCVN	RS Canum Venaticorum	81,393	.
SR	Semiregular	119,261	.

among all classes. We remove the top 10% and bottom 10% of the distribution to eliminate outliers and misclassifications, as well as potential duplicates in the ZTF photometry. The process has been illustrated for the class CEP in Fig. 3(a). We see a prominent peak centered near 0 and a long tail that extends to 2.13×10^6 in the original distribution. After the cuts, the distribution of distances remains right-skewed, with a long tail. The distribution of pairwise Cityblock distances for CEPs after outlier removal is also illustrated in Fig. 3(b). This step is largely robust to the choice of distance metric used (*e.g.*, the Euclidean and Cityblock metrics lead to datasets with an overlap of >98%).

Finally, we have a 'clean' dataset of 558 objects of each class mentioned in Table 1.

4. Feature selection and dimensionality reduction

We are up to this point working in a 114-dimensional feature space. Working in high-dimensional spaces may hinder the performance of machine learning models due to the "curse of dimensionality" (Bishop, 2006), which leads to a decrease in performance and an increase in computational complexity. To address this, we further reduce the
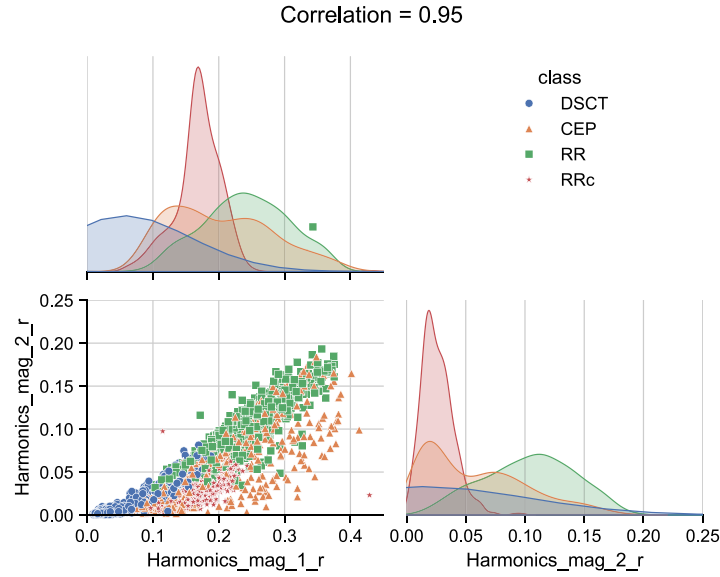
Fig. 4. Correlation between the first two components of a harmonic series fit to each light curve in the *r*-band. The bottom left panel shows the linear relationship between the two components, while the remaining two panels illustrate the distribution of the component per class. We find that these two components have a Pearson's linear correlation coefficient *r* = 0.95, and thus choose only `Harmonics_mag_1` as part of our feature selection step.

dimensionality of the 114 feature space using a variety of approaches. We refer to this step as feature selection, because we are selecting a subset of features from those created in Section 3.2.

4.1. Drop all g-band features

Our data consists of light curves measured in two wavelength passbands, *g* and *r*. We used `lc_classifier` to extract the aforementioned features from each band, and from the "multiband" light curve generated as the difference between the two, which represents the color of the transient.[6] Because the multiband features contain information from both *g* and *r* bands, we dropped all *g*-band features as a dimensionality reduction step. By removing only *g*-band features, we are not removing any information, since it is indirectly contained in the multiband features derived from both the *r* and *g* bands. We chose to remove *g*-band features instead of *r*-band features because ZTF is more sensitive in the *r*-band, and also has more observations in the *r*-band. This step reduces the dimensionality of the feature space from 114 to 60.

4.2. Dropping flags and number of points

We dropped features that are not physically motivated or are not well suited to measure distances. These include the number of points in a light curve,[7] and flags regarding the success of models fits (because of the low dynamic nature of their binary values). This removes 16 features.

4.3. Dropping highly correlated features

Features having high correlation (which we measure with Pearson's linear correlation coefficient, *r*) do not add much new information to the classification. So, for every set of highly correlated features (*i.e.*, $r > 0.9$),[8] we keep only one from that set.

For example, Fig. 4 shows the correlation between two features for the case of multi-class classification (see Section 5) — the amplitude of the first two components of a harmonic series fit to each light curve in the *r*-band, for a subset of our 10 classes. Because this correlation is high ($r > 0.9$), we only choose one of these features (`Harmonics_mag_1`).

After dropping the features as described above, we are left with 31 of the 114 feature dimensional space. A correlation matrix of the original 114 features and of the final set of 31 features is shown in Fig. 5.

5. Classification

5.1. Classification problems

Variable stars, and most stars which vary to some degree, are classified based on their observational features into classes. Their variability is generally connected to physical properties such as mass, metallicity (or chemical composition), age, and environment, with the goal of understanding their observational properties, including their variability, as an expression of physical processes within the star (or star system in the case, for example, of eclipsing binaries). A wealth of classes have been identified in astrophysical studies (Eyer and Mowlavi, 2008). To demonstrate the potential of our classifier in the classification of variable stars' light curves, we examine three typical, progressively more complex classification scenarios:

1. One *vs.* Rest Classification: EA, rest[9]
2. Binary Classification: RSCVN, BYDra
3. Multi-class Classification: CEP, RR, RRc, DSCT

For each classification task, we also run a Random Forest Classifier (RFC; Breiman, 2001), a well-known machine learning model which uses ensembles of decision trees models to classify a test object. We choose RFC as our benchmark comparison because most astrophysical services use RFC as the (or one of the) classifiers of choice (*e.g.* Hinners et al. 2018, Sánchez-Sáez et al. 2021, Cheung et al. 2021). Fig. 6 shows the canonical hierarchical classification scheme for the

---

[6] Since magnitude is a log scale, the difference of the magnitudes is the ratio of brightness in the two bandpasses.
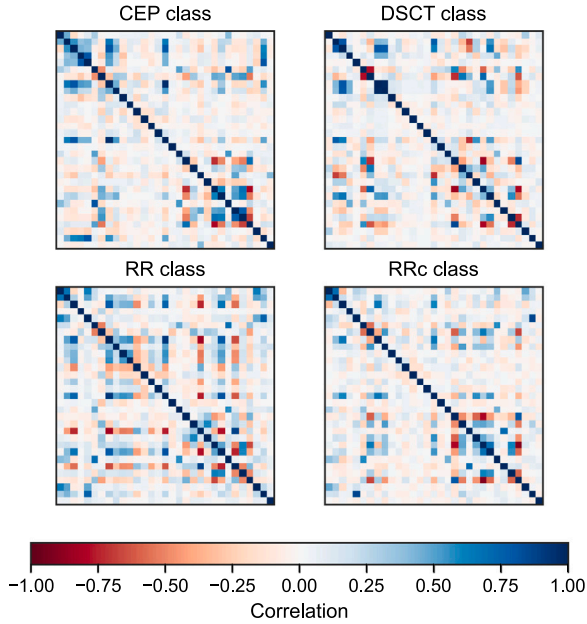
[7] Note that we already remove light curves with few points to avoid introducing bias in feature extraction.

[8] The threshold 0.9 was empirically found to be suitable to balance the trade-off between reducing redundancy and retaining informative features.

---

[9] Note: Rest includes all other classes combined.
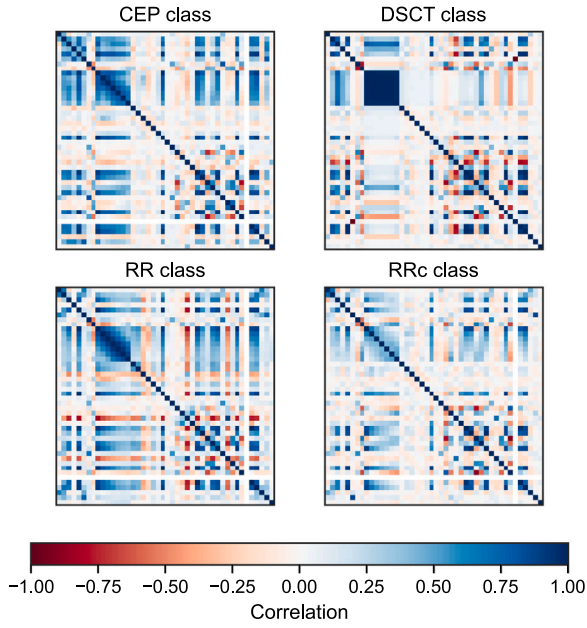
(a) Before feature selection.

(b) After feature selection.

Fig. 5. Correlation matrices of our features space before and after dimensionality reduction for four exemplary labels: CEP, DSCT, RR, RRc (see Table 1 for a definition of each class), which we use for the multi-class classification problem. Each cell in the plot represents the Pearson's linear correlation coefficient, r, between the corresponding feature on the x and y axis. By definition, the diagonal has a correlation $r = 1$. No correlation ($r = 0$) is mapped to the color white, positive correlation ($r > 0$) to shades of blue, while negative correlation ($r < 0$) to shades of red. Our original set of 114 features per object (a) is computed using lc_classifier. These consisted of features calculated in the r, g, and $g-r$ (multiband) light curves. In (a), the correlation between features is evident for each class in the block structure of the correlation matrix. We dropped g-based features, features that are not physically motivated, and finally, features that have a very high correlation. This leads to the correlation matrices shown in (b). Our final feature space has 31 features.

Table 2

An example of the median set for 4 r-band features and 4 classes from the multi-class classification case. This is just a subsection of the complete data, the full table is available online at https://github.com/sidchaini/LightCurveDistanceClassification.

Class	r_{period}	$r_{amplitude}$	r_{R21}	r_{skew}
CEP	3.86	0.25	0.27	−0.16
RR	0.56	0.35	0.45	−0.52
RRc	0.33	0.20	0.15	0.01
DSCT	0.09	0.08	0.17	−0.14

ten classes included in our work, based on Eyer and Mowlavi (2008, Figure 1). Note that this is a physical, rather than a phenomenological classification, so that, while in general they display marked similarity, proximity in this classification scheme does not necessarily imply proximity in the photometric feature space.

Our analysis will focus on the more comprehensive multi-class classification, where we look at four classes of Cepheid-like pulsating variables: Cepheid (CEP), RR Lyrae Type ab (RR), RR Lyrae Type c (RRc), and Delta Scuti (DSCT). We discuss in detail these results in Section 6. But we also report the performance of our method on a specific binary case and One *vs.* Rest case for comparison, keeping in mind that the results obtained when choosing one or two classes may differ significantly for any other class or class combination. In the One *vs.* Rest classification, we isolate Eclipsing Algols (EA) from all other classes. In the binary classification, we separate two classes of rotating variables (RS Canum Venaticorum, RSCVN, and BY Draconis, BYDra). This is a particularly challenging classification because while physically different, both classes exhibit photometrically similar behaviors, with variability at the $\sim 0.1 - 0.2$ mag level, and with varying phase, magnitude slope, amplitude, and period (Bopp, 1980).

5.2. *Distclassipy* classification algorithm

We use a custom algorithm, which we have named DistClassiPy, to classify the light curves. Our method draws inspiration from the k−Nearest Neighbors (k−NN; (Cover and Hart, 1967; Fix and Hodges, 1989)) algorithm. In addition to the classification that k−NN provides, DistClassiPy also offers a quantification of the uncertainty in the prediction. We have implemented this in Python using the scikit-learn API (Buitinck et al., 2013).

The detailed training algorithm is outlined in Algorithm 1.

5.2.1. Training

We compute the median and standard deviation for each feature per class, where we use the median set as a representative for each class.[10] Using the median here also makes our calculations more resilient to outliers. An illustration of the median values for four representative features can be found in Table 2. The distance metric is not selected in the training step.

Algorithm 1: Training Step.

1: **for** each class C in the training set **do**
2:     **for** each feature F in class C **do**
3:         Calculate the median M_F^C of feature F in class C.
4:         Calculate the standard deviation σ_F^C of feature F in class C.
5:     **end for**
6:     Save the median set $\{M_F^C\}$ and standard deviation set $\{\sigma_F^C\}$ for class C.
7: **end for**

---

[10] The median and standard deviation can also be replaced by other statistical measures of central tendency and dispersion.
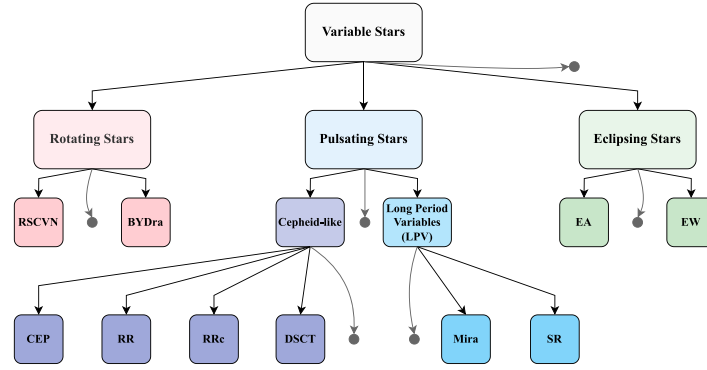
Fig. 6. The canonical hierarchy tree for the ten classes included in our work. This is a physical classification, based on Eyer and Mowlavi (2008).
Note: the gray dot represents the objects which belong to none of the other classes, which are not represented in our dataset.
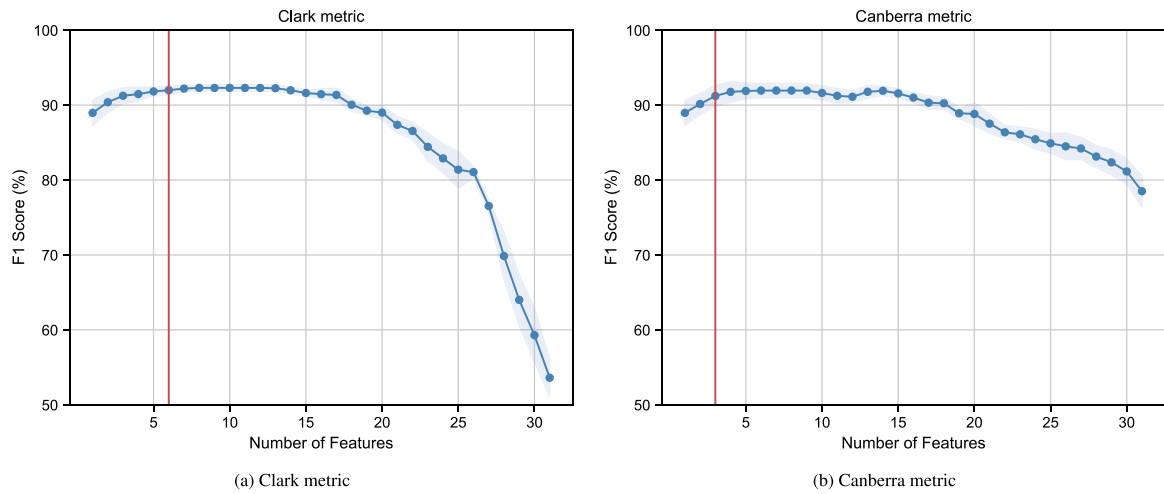

(a) Clark metric

(b) Canberra metric

Fig. 7. How `DistClassiPy` performance changes for the multi-class classification problem with the number of features used. Here we show our final step in dimensionality reduction, the Sequential Feature Selection (SFS) process, for two sample distance metrics: Clark (Fig. 7(a)) and Canberra (Fig. 7(b)). The performance of the classifier, measured as the mean F_1 score in a 5-fold cross validation run, is shown as a function of the number of features included in the classification (with the standard deviation shown as a filled area). The SFS algorithm iteratively adds one feature at a time from $n=1$ to $n=31$ such that the F_1 score is maximized at each step. We then select a final set of n_{final} features such that n_{final} is the lowest number of features whose F_1 score is within 1σ of the maximum F_1 score (denoted by a red vertical line in each panel).

5.2.2. Predicting

Our approach to classifying a test object involves several steps. First, we select a distance metric and proceed to scale our data. This scaling is inspired by the Mahalanobis distance (Mahalanobis, 1936): we generalize the idea of measuring how many standard deviations away a test object is from the median set for each class calculated in the training step, for the chosen distance metric.

For each class, we scale both the test object and the class set by the standard deviation for that class. We then compute the distance between the test object and the median set in units of standard deviations.

Once we have calculated all the distances, we identify the predicted class for the test object as the one for which the object's distance to the median is minimum.

The detailed prediction algorithm is outlined in 2.

5.2.3. Scoring

To evaluate the performance of our classifier's predictions, we use two different scoring methods. The F_1 score is given by

$$F_1 = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})},\qquad(1)$$

Algorithm 2: Prediction Step.

1: Choose a distance metric.
2: **for** each test object **do**
3:   **for** each class C in the training set **do**
4:     **for** each feature F in class C **do**
5:       Scale the test object feature by dividing it by σ_F^C, which was calculated in the training step.
6:       Scale the median set $\{M_F^C\}$ features by dividing them by σ_F^C.
7:     **end for**
8:     Calculate the distance D_C between the scaled test object and the median set $\{M_F^C\}$ for class C, scaled by $\{\sigma_F^C\}$.
9:     Save D_C for class C.
10:   **end for**
11:   Choose the class $C_{\min}$ for which the distance $D_{C_{\min}}$ is the smallest.

12:   Assign class $C_{\min}$ as the predicted class for the test object.
13: **end for**

---

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The higher the
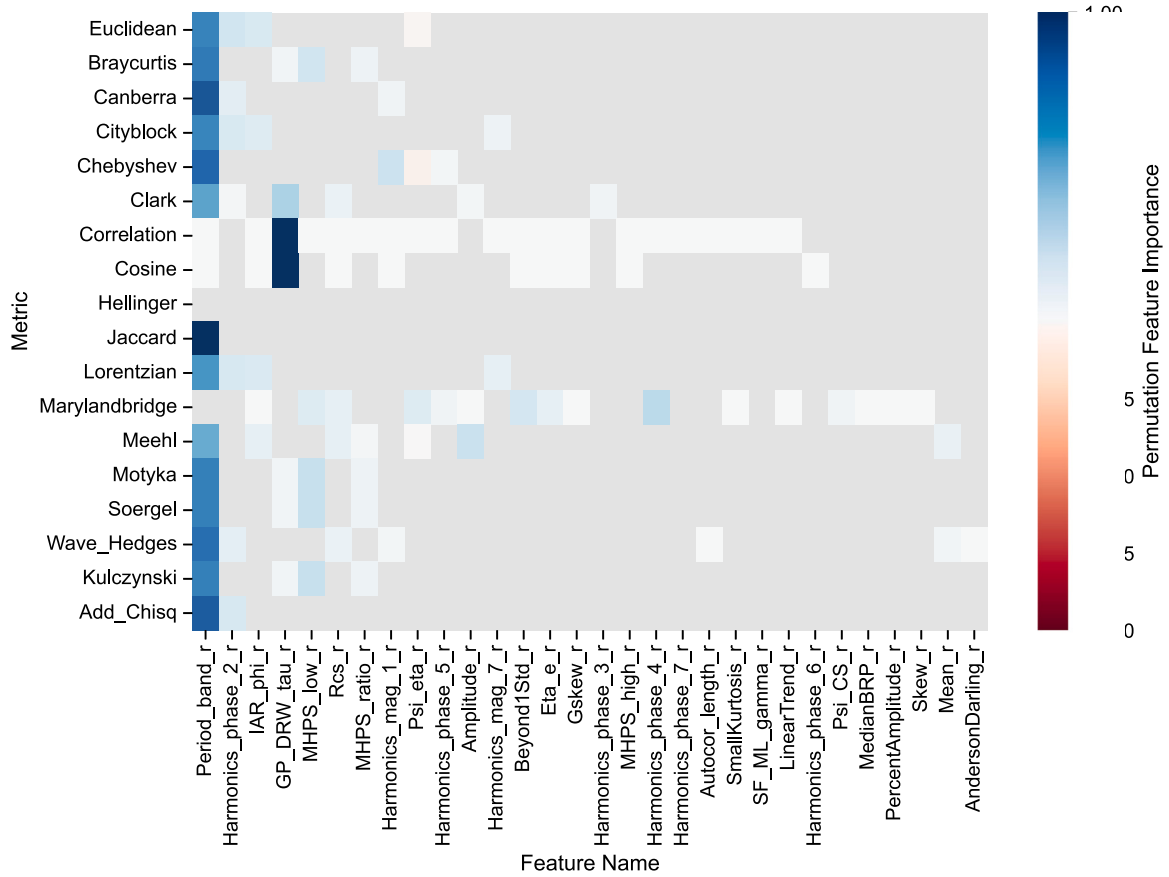
Fig. 8. `DistClassiPy` permutation feature importance for the multi-class classification task for all metrics. We only show the importance for features that were selected for the classification with at least three metrics (see Fig. 9), and all other features are shown in gray. The `Period_band_r` appears as important for nearly all metrics (but not for the Maryland Bridge metric, which, as discussed in Fig. 1 and Appendix A, is a vector-based metric). A negative importance (*e.g.*, `Psi_Eta_r` with Chebyshev) denotes the fact that, although using the SFS it was selected as one of the top four best features, when using all 31 features, it led to a decrease in performance. The Cosine and Correlation metrics stand out, with classification importance nearly entirely placed on the `GP_DRW_tau_r` feature: the relaxation time for a damped random walk model of the light curve, originally designed in Graham et al. (2017) for quasar classification.

F_1, the better the classification. For more than two classes, we take the average (*macro* mode) of the F_1 scores for each class. Because we consistently maintain class balance, we do not require any weights when calculating the F_1 score.

5.3. Classification confidence

First, as is standard in the machine learning community, we assess the stability of our classification results with cross-validation (Kohavi, 1995): we split our dataset into 5-folds, use four folds as our training set and one fold as our testing set, and average the five F_1 scores to obtain the final score. This assesses the confidence of the result for a given dataset and the influence of data outliers, but does not provide a direct measure of the uncertainty associated with the use of a specific distance metric.

Therefore, we additionally developed three confidence parameters for use with the classifier. Our goal was to create a distance-based measure for uncertainty (or confidence) in the classification method, not just the data. We tested the performance of the following three distance-derived confidence measures:

- Inverse of total distance: We invert the distance (d) calculated in the prediction step to give a classification confidence, such that a low distance value corresponds to a higher confidence value. Note that this distance is calculated in n dimensions corresponding to the n features, and depends on the metric being used. Furthermore, it has already been scaled by the standard deviation during computation (2). Since the distance scales are different
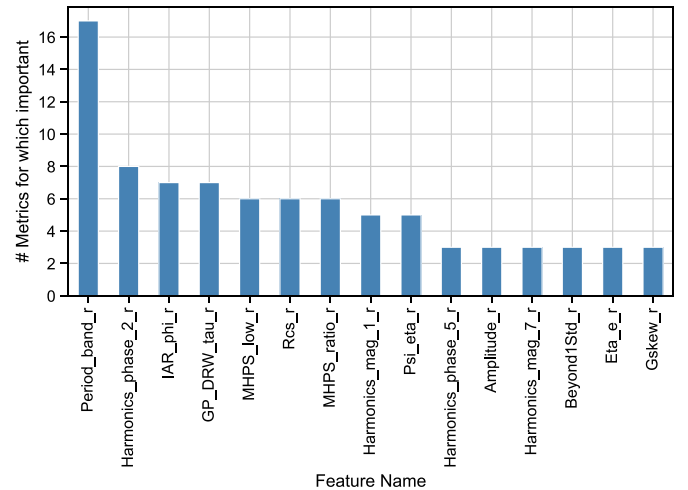

Fig. 9. The top features for the multi-class classification task across all metrics. This plot shows 15 features that were included in the final set of n_{final} features for ≥ 3 metrics. All but one metric (the Maryland Bridge metric) identified `Period_band_r` as important.

for different metrics, this value is only useful for comparison of confidences among different objects for a given metric. However, for a given metric, this is very easily interpretable, as something
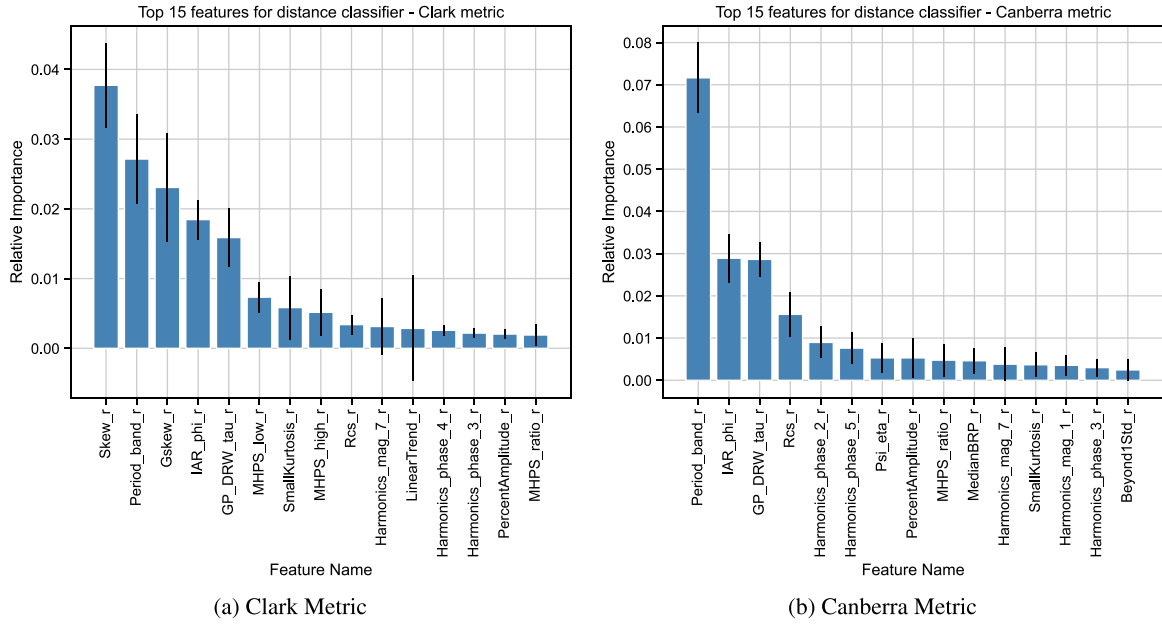
(a) Clark Metric

(b) Canberra Metric

Fig. 10. The relative importance of the top 15 features for the multi-class classification task when using the Clark (a) and Canberra metrics (b). The feature importance is determined by computing the Permutation Feature Importance. For each metric, the top 15 features are shown, although only six and three appear significant (at 3σ, respectively). Interestingly, the top feature for the Clark metric is `Skew_r` (the skew of the distribution of brightness values in the *r* band), which is not selected as important with the Canberra metric.

having a zero distance would imply an almost infinite confidence.

$$c_d = \frac{1}{d}$$

- Inverse of scaled distances: For a given feature (*i*), we compute a one-dimensional distance (d_i) between the test object and the median set. Because each feature has a different level of variance, we then scale it with the standard deviation (σ_i) associated with that feature within the reference class, and then finally invert it.

$$c_{\text{scaled}} = \frac{1}{\Sigma_i \frac{d_i}{\sigma_i}}$$

Scaling the distances with feature-specific standard deviations, we can ensure that the resulting confidence value can be compared across different metrics. However, this distance is not applicable for all metrics because not all distance metrics are defined in one dimension.

- Kernel Density Estimate (c_{KDE}) Probability: Kernel Density Estimate (KDE) is a statistical technique to estimate the probability density from given data (Silverman, 2018). It is given by:

$$f_{\text{KDE}}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right),$$

where x is the continuous variable over which density is estimated, N is the number of data points, h is the bandwidth parameter, x_i are the points in the dataset and K is the kernel function (for which we choose a Gaussian).

We thus use this KDE to generate the probability density for each class given its features. Finally, to calculate the KDE probability for a test object (x_{test}), we plug its features into the KDE for that class, and get the confidence for a test object belonging to that class (c_{KDE})

$$c_{\text{KDE}} = f_{\text{KDE}}(x_{\text{test}})$$

5.4. Sequential feature selection and classification

We have already reduced our feature space from 114 to 31 features. To increase the robustness of our method and decrease the computational cost, we further reduce our dimensionality by selecting the
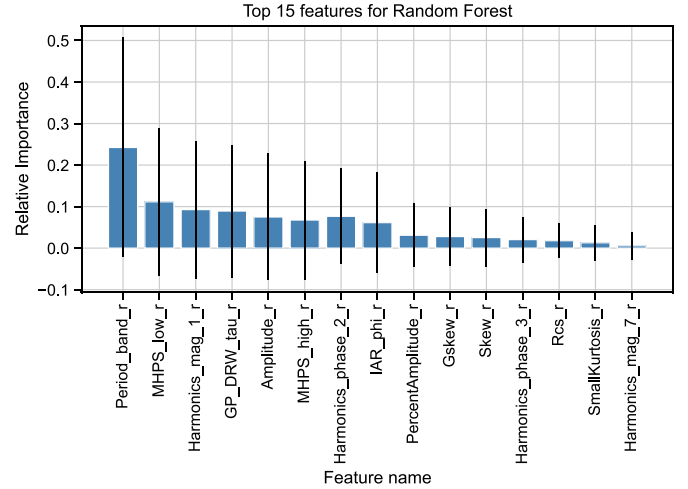

Fig. 11. The top 15 features for a Random Forest Classifier for the multi-class classification task mentioned in Section 5.1. The feature importance is determined by computing the Gini Importance (Breiman et al., 1984). We note that none of the features' importance is statistically significant.

most effective classification features for each class and each problem. This process is computationally expensive (as described below) and the final set of features to be used depends on the metric used and the classification task. However, this computation can be performed once for each problem, as we did for the three classifications we explored in our work. Thereafter, the user of our algorithm can choose the features that best match their research based on their scientific interest and object sample, with a net computational gain. Here, we describe our strategy, and the detailed results are discussed in Section 6.3.

We use the Forward Sequential Feature Selection (SFS) strategy (Ferri et al., 1994). Forward SFS works by adding one feature at a time: we start by choosing the single feature that, alone, maximizes our 5-fold cross-validation F_1 score. We continue adding features and increasing the dimensionality of the feature space, each time selecting the next
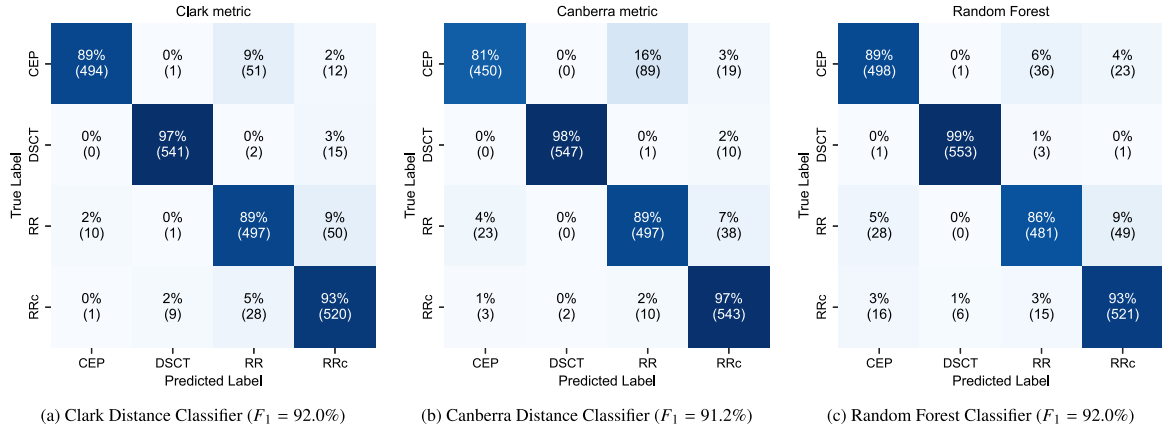
Fig. 12. Cross-Validation Results: Confusion matrix of our `DistClassiPy` model using Clark (*left*) and Canberra (*center*) distances (two of the top-performing distance metrics) compared to the performance of a Random Forest Classifier (*right*) for the multi-class classification task of Cepheids (CEP), Delta Scuti (DSCT), RR Lyrae ab (RR) and RR Lyrae c (RRc). The respective F_1 scores are shown in each subplot: all three models achieve comparable performance ($F_1 \sim 92\%$). Each cell in each plot shows the percentage and the number of objects corresponding to that combination of label (y) and prediction (x). The diagonal corresponds to correct classifications.

feature that maximizes F_1. We use Forward SFS over Backward SFS because of the high computational requirements of Backward SFS which make it prohibitive in a high dimensional space. Recursive Feature Elimination (RFE) is another option for the selection of features, but RFE requires a built-in feature importance attribute to the model, which is not implemented at this time in `DistClassiPy`. A comparison of the selection performed by RFE, compared to SFS, is left for future work.

The total number of possible feature combinations are,

$$\text{No. of combinations} = \sum_{i=0}^{n} \binom{n-i}{1}$$

$$= \frac{n(n+1)}{2} \tag{2}$$

$$= \mathcal{O}(n^2) \tag{3}$$

With $n = 31$, we perform a total of 465 calculations for each distance metric (18) and classification task (3). This provides us with the feature importance ranking and allows us to track how performance varies with the number of included features, enabling us to optimize the dimensionality of our feature space: in each case, we choose a final set of n_{final} features such that n_{final} is the lowest number of features whose F_1 score is within 1σ of the maximum F_1 score. An example of SFS is shown in Fig. 7, for the Canberra and Clark distance metrics in the multi-class classification case.

The feature importance is measured as the relative permutation feature importance, an algorithm-agnostic method (originally introduced in random forests; Breiman, 2001) where the importance of a feature is measured as the drop in performance when the feature is randomly shuffled during testing. Feature importance for the multi-class classification is shown in Fig. 8 for each distance metric.

In Fig. 9 we show the number of distance metrics (out of our 18 metrics) for which a given feature is selected. The 15 features shown were important for ≥ 3 metrics and are considered as a "super-set" of important features. We emphasize that the user should select the most effective distance metric and subset of features, performing a feature selection such as the one described above, specifically for their data and classification problem, as the classification goal (*e.g.*, purity *vs* completeness in selecting a specific class of objects, *vs* separating two specific subclasses, etc.), as well as data quality (including sparsity, signal-to-noise ratio, noise property, number of bands, etc.) will influence the selection of features and ultimate performance of the model.

The top features for the Clark and the Canberra metrics are shown in Fig. 10 for the multi-class classification case. For comparison, we also show the feature importance results of the RFC in Fig. 11. We note that none of the RFC features' importance reaches statistical significance.

5.5. Random forest classifier

We use a Random Forest Classifier (RFC, Breiman, 2001) as a benchmark to compare our method. A Random Forest involves training an ensemble of decision trees, each performing binary decisions on one feature at a time. This method, therefore, does not leverage or require the definition of a distance. The RFC was run in the `scikit-learn` implementation with the following hyperparameter choices: the maximum depth was set to 3 to avoid overfitting, the forest was composed of 100 estimators (trees), with bootstrapped samples (to reduce the variance) and considering $\sqrt{N}$ features at each split. As for the distance-based models, the RFC was run with a 5-fold cross-validation scheme on the entire dataset.

6. Results

In this section, we report and discuss the performance of our model (`DistClassiPy`) and the comparison model (RFC), as well as the results of our investigation into the classification's confidence and feature selection.

6.1. *Distclassipy classification performance*

The best cross-validation result of our distance metric classifier (DMC) from `DistClassiPy` across all distance metrics and feature subsets for each classification task are as follows: for the Multi-class Classification, DMC = 92% (RFC = 92%); for the One *vs.* Rest, DMC = 94% (RFC = 95%; for the Binary classification of rotating stars, DMC = 68% (RFC = 70%. A comprehensive report of the performance of our model, including the distribution of performance values across different distance metric and feature selection choices, is presented in Table 3. The full spreadsheet is available online.[11] Confusion matrices for the multi-class classification task for the distance metric classifier (using the top two distance metrics: Clark and Canberra) and RFC are shown in Fig. 12. From these, we see that performance is mostly very similar. However, for some classes like RR and RRc, the Canberra distance classifier outperforms the RFC.

Finally, we test our final classifiers on the hidden set introduced in Section 3.1, which consists of 500 objects, entirely unseen in training. For each classification, we test the best model (best distance metric and features subset) on a hidden dataset, first rebalanced and then unbalanced, where each class has a representation similar to the fraction of

---

[11] Cross-ValResultsURL

Table 3

Summary table of the performance of `DistClassiPy` for each classification task. The statistics are reported across the selection of distance metrics and feature subsets. The RFC results are shown for comparison next to the highest distance metric classifier performance for each classification task. The complete set of results is available online as a spreadsheet here.

		Multi-class		Binary		One-vs-rest	
		DistClassiPy	RFC	DistClassiPy	RFC	DistClassiPy	RFC
F_1 Score (%)	Max	92	92	68	70	94	95
	75 Percentile	91		66		93	
	50 Percentile	90		65		93	
	25 Percentile	82		64		90	
	Min	75		62		89	
No. of features	Max	20		25		17	
	75 Percentile	7		3		5	
	50 Percentile	4		3		4	
	25 Percentile	4		2		2	
	Min	1		1		1	

objects in the initial dataset. We confirmed the model performance for all three classification tasks, and found the results to be mostly within 1σ expectations from our cross-validation results for the rebalanced dataset. With the best distance metric selected on the cross-validated data, for each classification task the performance on the hidden set is $F_1 = 93\%$ with the Clark metric on the multi-class classification (1881 objects), $F_1 = 68\%$ with the Euclidean metric on the binary classification (1000 objects) and $F_1 = 92\%$ with the Kulczynski metric for the One *vs.* Rest classification (976 objects; Section 5). For the unbalanced dataset, the performances change slightly. In the multi-class classification, the unbalanced dataset contains 494 RR, 253 DSCT, 210 RRc, 24 CEP. The performance of the Canberra-based model improves slightly from $F_1 = 91\%$ to $F_1 = 93\%$, while the performance of the Clark-based model drops to $F_1 = 86\%$ from 92%. This is primarily due to confusion between the RR and RRc classes, which are the most common labels. However, for comparison, the RFC performance drops significantly more, from $F_1 = 92\%$ to $F_1 = 76\%$. In the imbalanced One *vs.* Rest classification, the performance drops by a few points, from $F_1 = 93\%$ (with Euclidean distance) to $F_1 = 90\%$. Similarly, the RFC performance drops from $F_1 = 95\%$ to $F_1 = 89\%$. The binary classification is naturally balanced with 500 BYDra and 480 RSCVN.

The reader should however be reminded that our classification is based on the labels found in Chen et al. (2020) and that while Chen et al. (2020) use DR2 ZTF data, our work is based on data from DR15. A possible implication, since DR2 data contains fewer points per light curve, is that label noise contributes to apparent inaccuracy in our scores.

The complete set of results is available online as a spreadsheet.[12] The confusion matrices for the multi-class classification of the hidden set are displayed in Fig. 13 for the Clark and Canberra metric.

For the remainder of the discussion, we will focus solely on the multi-class classification task.

6.2. Computational requirements

To understand how the computational time scales with dataset size and feature dimensions, we monitored the total time taken for training and prediction for different numbers of data points and features using synthetic data generated with `scikit-learn` — the specifications for our hardware and software are outlined in Appendix C. We compared the timings of `DistClassiPy` using the metrics Canberra and Clark. It is worth noting that the Canberra metric is implemented in the `scipy.spatial.distance` package, and we thus expect it to be computationally optimized. On the other hand, the Clark metric is not available within `Scipy` (to our knowledge) and we coded it ourselves: we expect it may be sub-optimal in terms of computational costs. The timings of an RFC with 100 trees is also tracked. We ran each method five times, and report the mean here. Our results are illustrated in Fig. 14 and Fig. 15.

We find that `DistClassiPy` is remarkably scalable and shows minimal computation times even for large and high-dimensional datasets. Even when dealing with a dataset size of 10,000 objects and 500 features, the computation times for `DistClassiPy` remain as low as 0.0677 ± 0.0016 seconds (Canberra) and 0.1538 ± 0.0023 seconds (Clark) compared to 14.3563 ± 0.4978 seconds for RFC. These make `DistClassiPy` very suitable for current and future astrophysical surveys, with increasing sample sizes (*e.g.*, the Vera C. Rubin LSST is expected to characterize 17B stars, many of which are variable to the depth limit of the survey, and 48B astrophysical objects altogether).

Finally, `DistClassiPy` and RFC utilize a comparable amount of RAM (the distance metric classifier utilized 574.5 ± 4.8 MiB compared to RFC's 568.4 ± 16.3 MiB on our machine.[13] for the dataset having 10,000 objects and 500 features) during training, as per our tests using the Python package `memory_profiler`.

In conclusion, `DistClassiPy` performs comparably to the RFC (state-of-the-art), while being faster and utilizing a comparable amount of memory.

We now look at other assets that `DistClassiPy` can offer us to help understand classification — confidence and robustness.

6.3. Feature importance

We find that whether a feature is important or not depends not only on the classification task, but also the distance metric being used. In other words, each metric has features with which it works best, and often, these features differ for different metrics. We show in Fig. 16 how the important features change for the multi-class classification task for three metrics — Canberra, Clark, and Soergel.

This flexibility allows for the distance metric classifier to be adapted to the specific research goal and resources. In a scenario for a particular classification task, if some features are easier or faster to calculate than others, one could use `DistClassiPy` with a distance metric which is the most optimal for these features.

6.4. Confidence of the distance metric classification

In Section 5.3, we introduced three different confidence methods with the goal of measuring the classification reliability for each metric. Here, we compare these three methods. The question we ask is - does a confident prediction imply a correct prediction? Although it is difficult to answer this in absolute, we can instead employ the following strategy: we treat the confidence as a pseudo-probability, and then see what the performance would be if we used this probability as a class prediction (instead of the distance metric classifier). Finally, we

---

[12] HiddenSetResultsURL

[13] Note: These numbers are meant for comparison when run on our machine whose specifications are given in Appendix C
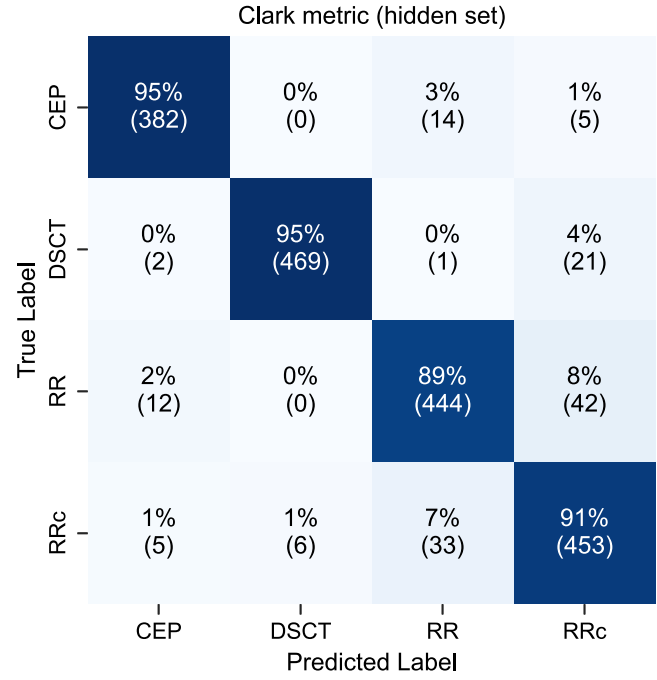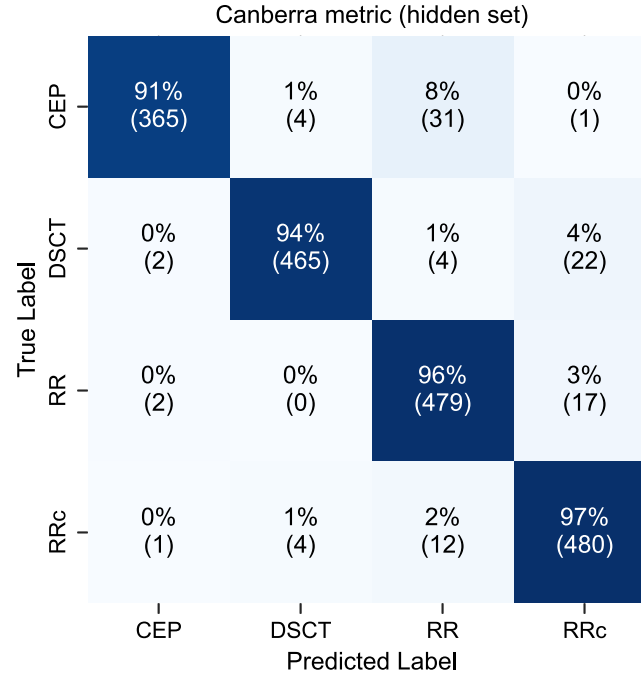
(a) Clark Distance Classifier on the *hidden* set ($F_1 = 92.7\%$)

(b) Canberra Distance Classifier on the *hidden* set ($F_1 = 94.7\%$)

Fig. 13. Hidden Set Results: Confusion matrices (as in Fig. 12) for our `DistClassiPy` model on the *hidden* set using Clark (*left*) and Canberra (*right*) distances for the multi-class classification task of Cepheids (CEP), Delta Scuti (DSCT), RR Lyrae ab (RR), and RR Lyrae c (RRc). Both Clark and Canberra perform more than 1 σ better than their scores from cross-validation ($91.99\% \pm 0.61\%$ and $91.22\% \pm 1.38\%$, respectively).

evaluate the F_1 score for the prediction for each of the three confidence methods Fig. 17. We note that, because the first confidence method, c_d is just $1/d$, where d is the distance metric classifier distance, this prediction is the same as the `DistClassiPy` prediction.

The F_1 scores for predictions from the method c_d is the most reliable, and works well for all distance metrics ($72\% \leq F_{1,c_d} \leq 91\%$). c_{KDE} slightly outperforms the c_d method for a few metrics (*e.g.,*

Wave-Hedges), but dramatically fails for some others (*e.g.,* Maryland-bridge). However, all of the metrics for which c_{KDE} underperforms ($F_{1,c_{\mathrm{KDE}}} \leq 64\%$) are the lowest five performers overall, while for all others $F_{1,c_{\mathrm{KDE}}} \geq 87\%$. Meanwhile, scores for c_{scaled} are the poorest for all distance metrics. Based on this we conclude that, although the simplest, the confidence parameter c_d is a more reliable indicator of overall classification confidence and we thus use c_d in `DistClassiPy`.
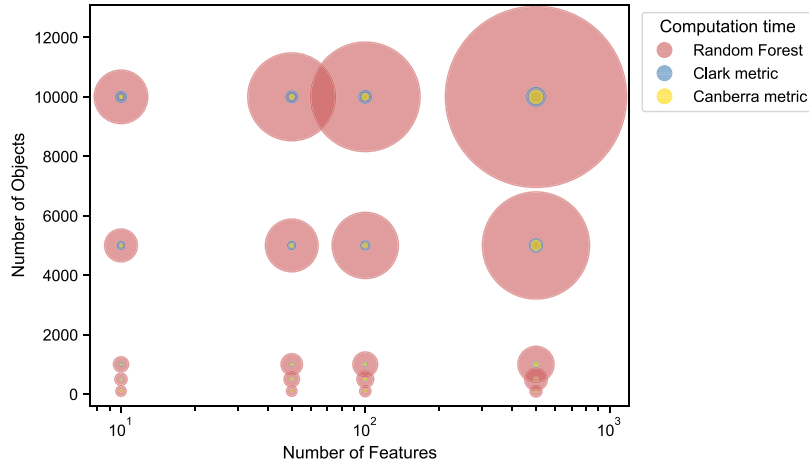
(a) Visual comparison of computation time across a Random Forest, and `DistClassiPy` with Clark and Canberra metric.
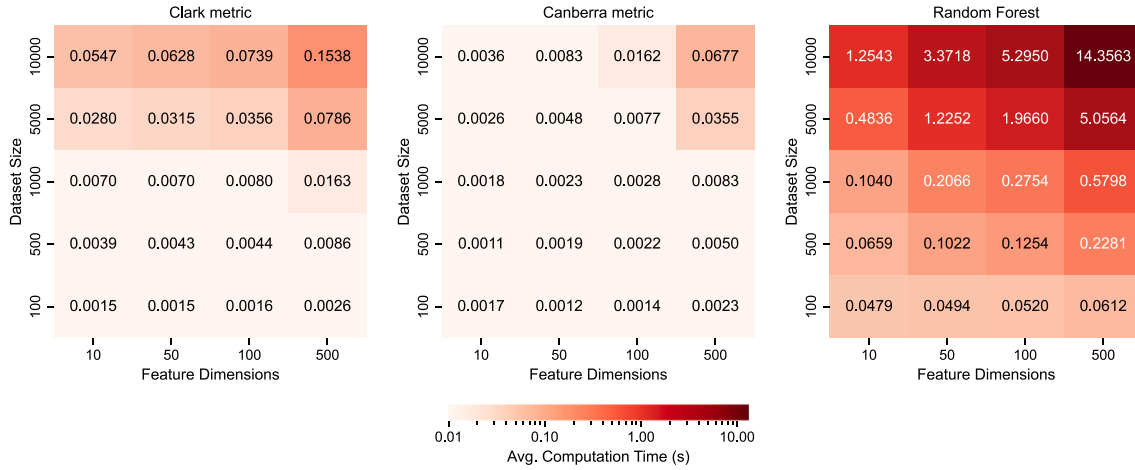
(b) Clark Distance Classifier

(c) Canberra Distance Classifier

(d) Random Forest Classifier

Fig. 14. Top: A visual comparison of computational times between our method, `DistClassiPy` and RFC as a function of the dataset size (number of objects; *y*-axis) and number of features (*x*-axis). For all plots, the computational costs of fitting the model on the training data and predicting on the testing data (not including preprocessing) are evaluated on the entire dataset five times, and the results shown are the average over these five iterations. The area of each circle is proportional to the computational time required for training and generating the predictions. We see that `DistClassiPy` is always faster than RFC (with 100 trees), and the difference is even more prominent in larger datasets. Bottom: A comparison of computational times (in seconds) between `DistClassiPy` and RFC as a function of variable dataset size and number of features. We see that `DistClassiPy` is always at least an order of magnitude faster than RFC (with 100 trees), even when using custom metrics (*e.g.* Clark) which are not in-built in `SciPy` (like, *e.g.*, Canberra).

An interesting point to note is that, unlike the probabilistic outputs from something like RFC, our distance metric classifier with c_d does not assume a fixed confidence level shared across all classes. The distance metric classifier does not impose a strict boundary on how far away a test object can be, and we suspect this may be useful for dealing with outliers and anomalies. However, this is not something we will explore in this work, and we end our discussion on confidences.

6.5. Robustness of the distance metric classification

We are interested in determining the robustness of our classifier — how well does our classifier perform with different types of data? We split the test set into batches, splitting the data by a given feature's distribution into four quantiles, ensuring that each quantile contains an equal number of objects. Then, we evaluate the accuracy for each quantile separately, only using the final set of features from the SFS (Section 4). Since this accuracy is now a measure of robustness, we call it the robustness score. This method is applied to all features from our "super-set" (as derived in Fig. 9). For the multi-class classification

task, robustness scores for the Clark and Canberra metric have been visualized as a heatmap in Fig. 18. We note that, although all features in the "super-set" of the 15 best features are not used by the classifier itself, we still use them to split our quantiles, before dropping the unused feature based on the distance metric. This helps us compare the robustness for different metrics.

In addition to obtaining the robustness scores for a given distance metric across all features, we can also compare the robustness of all distance metrics across a given feature. In Fig. 19, we look at the robustness score for the multi-class classification task across all the 18 distance metrics when splitting our test data based on `Period_band_r` (the *r*-band period, which is one of the most important features for the multi-class classification, as we saw earlier in Fig. 9).

The performance for different quantiles of the *r*-band period varies for different distance metrics. For example, the Jaccard metric provides a highly accurate prediction (99%) for objects having short *r*-band periods (such that they belong in Quantile 1), but not so much (81%; Q4) for objects with a long period (from Quantile 4). Conversely, the Meehl metric has a low accuracy (82%; Q1) for objects with shorter
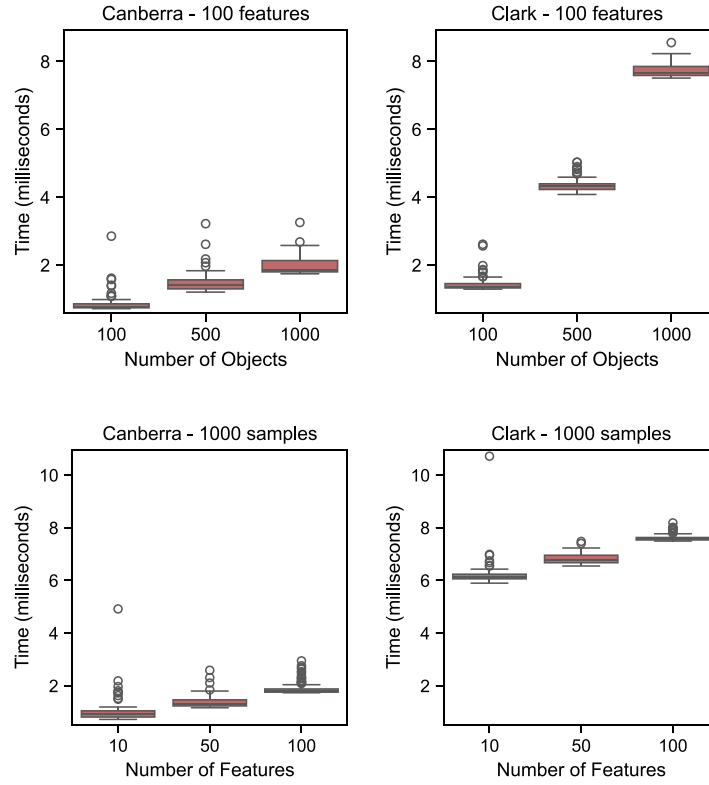
Fig. 15. The computational time of `DistClassiPy` using the Canberra and Clark metrics (top and bottom rows respectively) as a function of sample size using 100 features (left plots) and of number of features using 1000 objects (right plots).
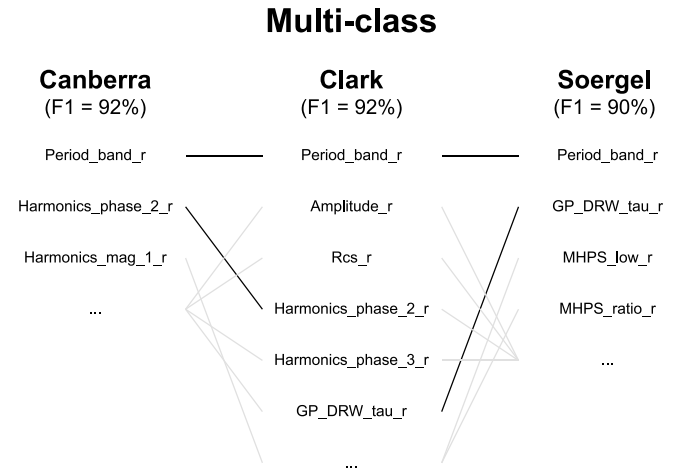

Fig. 16. How feature importance changes for three different distance metrics (Canberra, Clark, and Soergel) in the multi-class classification. Each of these metrics has a similar performance, but there is only one feature that is common among all of them: `Period_band_r` — the *r*-band period.

periods, but has some of the best performance (93%; Q4) for objects with longer periods. The fact that performance for different quantiles varies for different metrics may have remarkable implications. This can enable a dynamic approach for classification - when a test object is discovered, a different distance metric might be suitable based on what values its features have. This dynamism is something that cannot be achieved with traditional classifiers.

6.6. Interpretability

One more key advantage that `DistClassiPy` offers is better interpretability compared to methods like RFC and Neural Networks. When a test object is classified using the distance metric classifier, we know exactly what features are being used (unlike some deep learning methods), and we also know which features are more important. Furthermore, the decision making process is also fairly straightforward, unlike an RFC, whose constituent trees often have arbitrary decision boundaries. The distance metric classifier essentially finds the class to which the test object is "closest", in a scaled feature-space. This can also be directly visualized for lower dimensions for a given metric space, and this may play a significant role in understanding a particular classification.

7. Conclusions

Time series classification is a non-trivial problem in machine learning. In ground-based astronomy, light curve classification is even more challenging due to the sparsity and heteroscedasticity of the data. We developed a distance metric classifier `DistClassiPy` that uses distance metrics computed on features derived from light curves. Tested on multiple classification tasks, our classifier produces results comparable to other state-of-the-art feature-based methods like Random Forest Classifiers (RFCs), with $F_1 = 92\%$ and $F_1 = 94\%$ on a multi-class and a One *vs.* Rest classification respectively. We achieve a performance of $F_1 = 68\%$ on a particularly challenging binary classification that attempts to separate RSCVN and BYDra, two star types that exhibit photometrically similar and highly variable behaviors (Bopp, 1980), and for which, too, the RFC performance is comparable $F_1 = 70\%$. This cross-validated performance was confirmed by running the model on a "hidden" dataset completely unseen in training and that, unlike
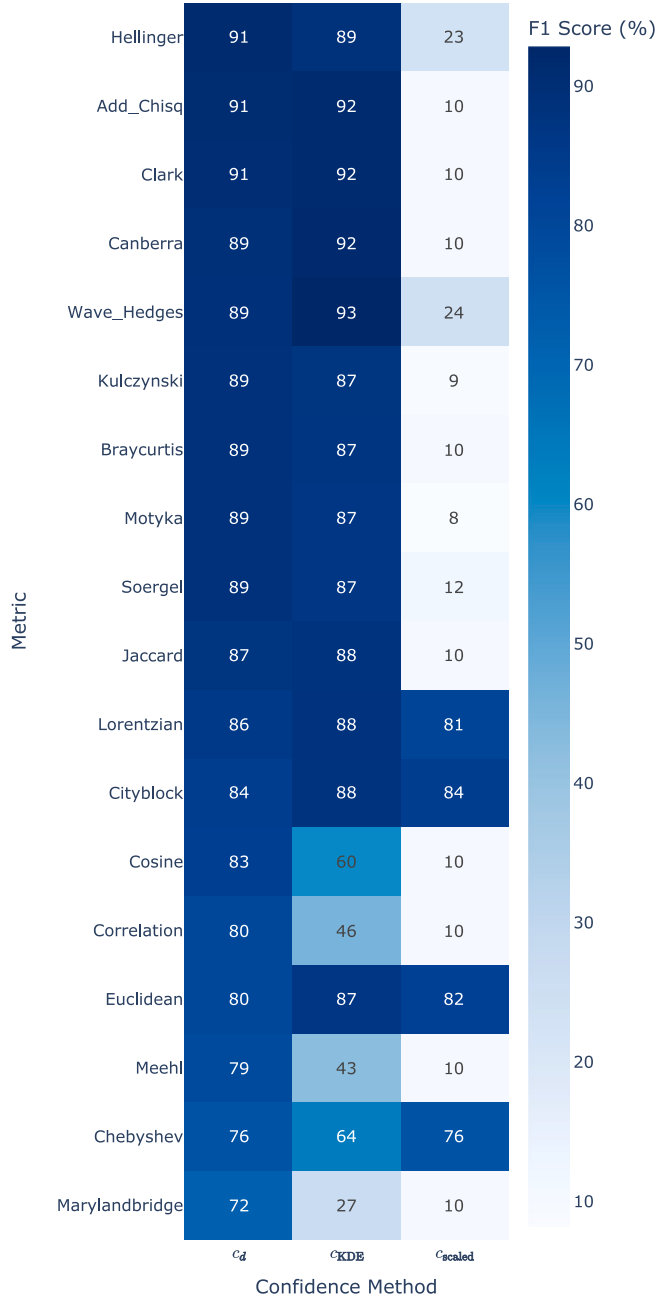
Fig. 17. Here, we compare the effectiveness of three confidence methods mentioned in Section 5.2.3 for our 18 metrics, for the multi-class classification task. We do this by treating our confidences as a pseudo-probability, and then using this pseudo-probability for predicting a test object. Finally, we calculate the accuracy for each method using these predictions, which has been plotted above. This accuracy score helps tell whether a confident prediction is an accurate prediction. From this plot, we see that c_{scaled} and c_{KDE} are ineffective methods, and we thus choose c_d as the confidence method for `DistClassiPy`.

the original dataset, is not rebalanced to have an even representation of all classes. For all three classification tasks, we confirm the performance obtained in cross-validation testing: $F_1 = 93\%, 92\%,$ and 68% respectively for the multi-class, One *vs.* Rest, and binary classification.

At the same time, `DistClassiPy` is faster, more interpretable, and suited to be tailored to specific science goals with additional performance enhancements and computational advantages. In our tests,

it performed 50× faster than RFC on small datasets (10 features by 100 objects), and over 150× faster on larger datasets (500 features by 10,000 objects) demonstrating impressive scalability properties. The dynamism in the distance metrics allows for a choice suited to the available features and the test particle's characteristics within the feature space. Finally, because the features are a result of domain-sensitive dimensionality reduction, the user can leverage relevant knowledge in the context of the problem.

`DistClassiPy` is made available to the users as open-source code. The package is accessible through PyPI[14] with the goal of broadening its potential applications not only in astronomy but also in other classification scenarios across various fields.

In future work, we plan to extend `DistClassiPy` to transient classification, explore its use for anomaly detection, and test more distance metrics, including metrics typically reserved for comparison of statistical distributions, *e.g.*, Earth Mover's Distance, Bhattacharya distance.

CRediT authorship contribution statement

S. Chaini: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **A. Mahabal:** Writing – review & editing, Visualization, Supervision, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **A. Kembhavi:** Writing – review & editing, Supervision, Resources. **F.B. Bianco:** Writing – review & editing, Visualization, Supervision, Resources, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Our dataset, along with all our code is available on https://github.com/sidchaini/LightCurveDistanceClassification.

In addition, our classifier package DistClassiPy is available on https://pypi.org/project/distclassipy and https://github.com/sidchaini/DistClassiPy.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT to generate LaTeX code templates. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

---

[14] https://pypi.org/project/distclassipy/
[15] https://www.lsstcorporation.org/science-collaborations
[16] https://lsst-tvssc.github.io/

(a) Clark metric   (b) Canberra metric
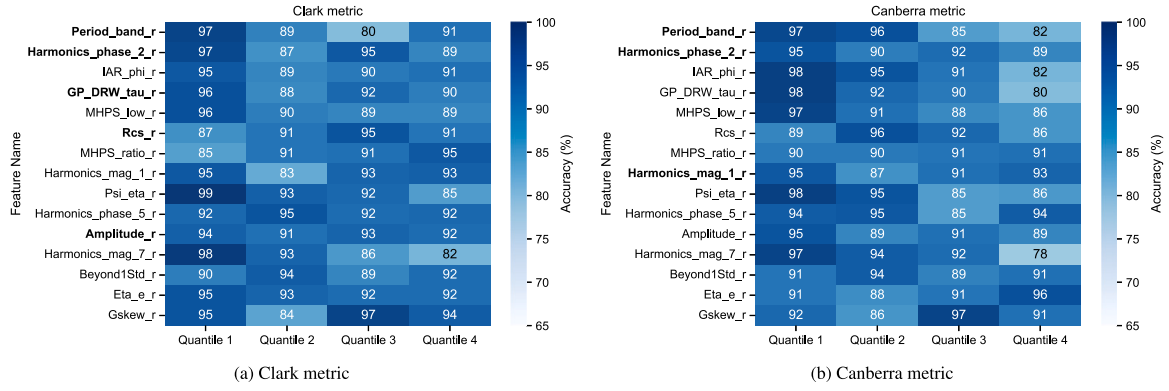
Fig. 18. Robustness scores for the multi-class classification task, with the distance metric classifier using the Clark (Fig. 18(a)) and Canberra metrics (Fig. 18(b)). The robustness score is calculated by splitting our test set into quantiles, and then evaluating the classification accuracy for each quantile. Note that, although each quantile is used to split the dataset, only a subset of these features are actually used in the classification (boldfaced). We still include all features from the super-set of features for the quantile split, to better compare performance for different metrics. For *e.g.*, although the Canberra metric (b) performs better in general, Clark metric (a) performs better with objects having larger periods (quantile 4, top row). The color maps limits for Fig. 18 and Fig. 19 have been cut to the range [65,100] to increase the readability of differences in performance, but both models presented in this figure perform better than 78% at any quantile with any feature.
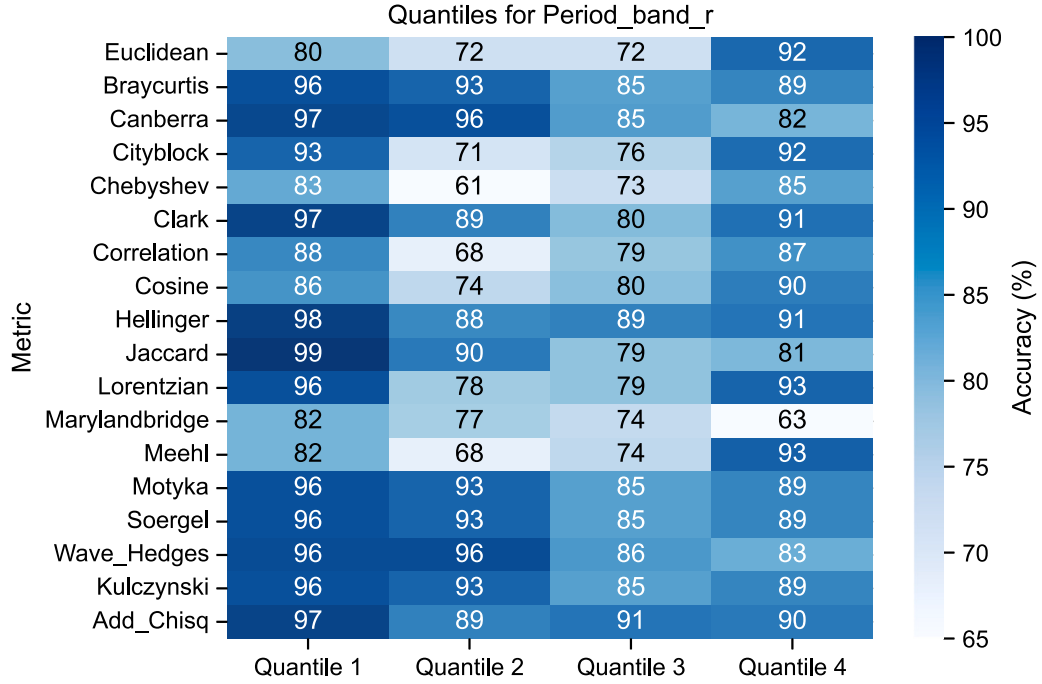

Fig. 19. Robustness for objects with different periodicity (`Period_band_r`) for the multi-class classification task with the distance metric classifier. The *r*-band period is the most important feature for most metrics. We compare how robust each of our metrics is to this feature. This figure is designed like Fig. 18. For variables having a shorter period (and belonging to Quantile 1), the Jaccard metric gives the most accurate classification. On the other hand, for variables with a longer period (and belonging to Quantile 4), the Meehl metric gives the most accurate classification.

Appendix A. Distance: A mathematical perspective

Earlier, in the paper, we defined the distance as:

Definition 2.1. The distance d between two points, in a set X, is a function $d : X \times X \rightarrow [0, \infty)$ that gives a distance between each pair of points in that set such that, for all $x, y, z \in X$, the following properties hold:

(1) $d(x, y) = 0 \iff x = y$ (identity of indiscernibles)
(2) $d(x, y) = d(y, x)$ (symmetry)
(3) $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

From the three axioms in Definition 2.1, another condition (*non-negativity*) can be shown to hold:

Corollary.

$$d(x, y) \geq 0, \text{for all } x, y \in X \tag{A.1}$$

Example. The Euclidean distance between two points in $\mathbb{R}^2$ is an example of a distance. Similarly, we can define a distance between any set of points such that Definition 2.1 is satisfied.

Definition A.1. A metric space (X, d) is a set X equipped with a metric d.

Example. The 2-dimensional $\mathbb{R}^2$ plane with the Euclidean distance is an example of a metric space. Similarly, we can also define metrics on matrices, functions, sets of points or any other mathematical object as long as it satisfies Definition A.1.

Now, we note the difference between the terms *distance* and *distance metric* (often shortened to just *metric*).

Definition A.2. The term *metric* refers to the way to calculate the distance between any two elements of the set. Meanwhile, the term *distance* refers to the scalar value obtained after using a particular metric on a pair of elements in the set.

Example. The Euclidean metric for $\mathbb{R}^2$ is given by $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, while the Euclidean distance between the points $(0, 0)$ and $(1, 1)$ is given by $d = \sqrt{2}$.

Appendix B. List of metrics

We used the following metrics for all distance calculations in this paper. For a detailed description of each metric, see Deza and Deza (2013).

1. **Euclidean metric**:

$$d(u, v) = \sqrt{\sum (v_i - u_i)^2} \tag{B.1}$$

The Euclidean metric is the most widely used metric and represents the length of the shortest straight line between two points in a multidimensional space.

2. **Cityblock metric**:

$$d(u, v) = \sum \left| v_i - u_i \right| \tag{B.2}$$

The Cityblock metric gives the shortest path between 2 points if we restrict movement to the grid. Named after the grid layout of cities, this is the shortest distance for a taxi to traverse between two places. It is also known as the Manhattan metric.

3. **Bray–Curtis metric**:

$$d(u, v) = \frac{\sum |u_i - v_i|}{\sum |u_i + v_i|} \tag{B.3}$$

The Bray–Curtis metric is commonly used in Ecology and Biology, and denotes the dissimilarity in the (biological) species composition of two sites.

4. **Canberra metric**:

$$d(u, v) = \sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|} \tag{B.4}$$

The Canberra metric is a weighted version of the Cityblock metric, with a denominator that is slightly different from the Bray–Curtis metric. The form of the denominator makes this metric sensitive to points that are closer to the origin.

5. **Chebyshev metric**:

$$d(u, v) = \max \left(|u_i - v_i| \right) \tag{B.5}$$

The Chebyshev metric is also known as the chessboard metric. It represents the minimum number of moves needed by the King chess piece to move from one point to another.

6. **Clark metric**:

$$d(u, v) = \left(\frac{1}{n} \sum \left(\frac{u_i - v_i}{|u_i| + |v_i|} \right)^2 \right)^{\frac{1}{2}} \tag{B.6}$$

The Clark metric is a variation of the Euclidean metric, but with a normalizing factor similar to the Canberra metric.

7. **Cosine metric**:

$$d(u, v) = \frac{\sum \left((u_i - \bar{u}) \cdot (v_i - \bar{v}) \right)}{\sqrt{\sum \left((u_i - \bar{u}) \cdot (u_i - \bar{u}) \right)} \sqrt{\sum \left((v_i - \bar{v}) \cdot (v_i - \bar{v}) \right)}} \tag{B.7}$$

The Cosine metric is based on the cosine similarity, which depends only on the angle between the two vectors in the feature space, and not their magnitude.

8. **Hellinger metric**:

$$d(u, v) = \sqrt{2 \sum \left(\sqrt{\frac{u_i}{\bar{u}}} - \sqrt{\frac{v_i}{\bar{v}}} \right)^2} \tag{B.8}$$

The Hellinger metric is a metric that is commonly used to compare two probability distributions but can also be used with numeric data.

9. **Jaccard metric**:

$$d(u, v) = \frac{\left(\sum u_i - v_i \right)^2}{\sum u_i^2 + \sum v_i^2 - \sum u_i v_i} \tag{B.9}$$

The Jaccard metric is based on the Jaccard similarity coefficient, which is a measure of similarity between 2 sets. The similarity coefficient can be converted into a distance.

10. **Lorentzian metric**:

$$d(u, v) = \sum \ln \left(1 + |u_i - v_i| \right) \tag{B.10}$$

Because the Lorentzian metric is given by a natural log, it is more sensitive to changes when the input vector is small, but less sensitive to changes when the input vector is large.

11. **Meehl metric**:

$$d(u,v) = \sum_{1 \le i \le n-1} \left(u_i - v_i - u_{i+1} + v_{i+1}\right)^2 \tag{B.11}$$

The Meehl metric mixes terms from consecutive dimensions, thus making the ordering of the dimensions significant.

12. **Soergel metric**:

$$d(u,v) = \frac{\sum |u_i - v_i|}{\sum \max\{u_i, v_i\}} \tag{B.12}$$

The Soergel metric is another modified version of the Cityblock metric but has only a max term in the denominator. Note that the summation is performed independently on the numerator and denominator before division.

13. **Wave–Hedges metric**:

$$d(u,v) = \sum \frac{|u_i - v_i|}{\max\{u_i, v_i\}} \tag{B.13}$$

The Wave-Hedges metric is a slight modification of the Soergel metric, where the summation is done after division.

14. **Kulczynski metric**:

$$d(u,v) = \frac{\sum |u_i - v_i|}{\sum \min\{u_i, v_i\}} \tag{B.14}$$

The Kulczynski metric is similar to the Soergel metric, with the difference that its denominator consists of a minimum instead of a maximum.

15. **Additive Symmetric χ^2 metric**:

$$d(u,v) = \Sigma \frac{(u_i - v_i)^2 (u_i + v_i)}{u_i v_i} \tag{B.15}$$

The additive symmetric χ^2 metric is a form of a weighted Euclidean metric, where terms between the two vectors are mixed.

16. **Correlation metric**:

$$d(\vec{u}, \vec{v}) = 1 - \frac{(\vec{u} - \bar{u}) \cdot (\vec{v} - \bar{v})}{\|(\vec{u} - \bar{u})\|_2 \|(\vec{v} - \bar{v})\|_2} \tag{B.16}$$

The Correlation metric is based on the Pearson correlation coefficient between 2 vectors.

17. **Maryland Bridge metric**:

$$d(u,v) = 1 - \frac{1}{2}\left(\frac{\sum u_i v_i}{\sum u_i^2} + \frac{\sum u_i v_i}{\sum v_i^2}\right) \tag{B.17}$$

The Maryland Bridge metric is based on the Maryland Bridge similarity. This metric does not satisfy axiom 3 (triangle inequality) in Definition 2.1 and it is therefore not a distance metric. However, it has been used in literature, particularly in the field of genomics to measure similarity between genomes (Mirkin and Koonin, 2003). Thus, it is included as a utility in our package.

18. **Motyka metric**:

$$d(u,v) = \frac{\sum \max\{u_i, v_i\}}{\sum (u_i + v_i)} \tag{B.18}$$

The Motyka metric has a summation term in the denominator, which acts as a scaling factor of the maximum term in the numerator. This metric does not satisfy axiom 1 (identity of indiscernibles) in Definition 2.1 (and it is not positive defined). However, it is a metric that commonly appears in the literature when measuring similarity between data, so it is included as a utility in our package.

Appendix C. Hardware and software specifications

The results described in this work, including the computational time estimates, were produced on machines with the hardware specifications reported in Table C.4. The Python libraries and corresponding versions used in this work are given in Table C.5.

Table C.4
Hardware Information.

Specification	Details
Processor	ARM
CPU	Apple M1 Pro
Physical cores	8
Total cores	8
RAM	16 GB
Operating System	Darwin 23.1.0
Machine	arm64
Platform	macOS-14.1.1-arm64-arm-64 bit

Table C.5
Software and Versions.

Library	Version	Reference
Python	3.12.2	Van Rossum and Drake (2009)
DistClassiPy	0.1.5	Chaini et al. (2024)
NumPy	1.26.4	Harris et al. (2020)
Pandas	2.2.1	Wes McKinney (2010), The pandas development team (2020)
scikit-learn	1.4.1.post1	Pedregosa et al. (2011)
SciPy	1.12.0	Virtanen et al. (2020)
statistical-distances	0.9.1	Zielezinski (2023)
matplotlib	3.8.3	Hunter (2007)
JSON	2.0.9	Pezoa et al. (2016)
tqdm	4.65.0	Casper da Costa-Luis et al. (2024)
seaborn	0.13.2	Waskom (2021)
mlxtend	0.23.1	Raschka (2018)
Jupyter	4.1.2	Kluyver et al. (2016)

References

Abu Alfeilat, H.A., Hassanat, A.B., Lasassmeh, O., Tarawneh, A.S., Alhasanat, M.B., Eyal Salman, H.S., Prasath, V.S., 2019. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. Big Data 7 (4), 221–248. doi:10.1089/big.2018.0175.

Arévalo, P., Churazov, E., Zhuravleva, I., Hernández-Monteagudo, C., Revnivtsev, M., 2012. A mexican hat with holes: Calculating low-resolution power spectra from data with gaps: Power spectrum of data with gaps. Mon. Not. R. Astron. Soc. 426 (3), 1793–1807. doi:10.1111/j.1365-2966.2012.21789.x, arXiv:1207.5825.

Bellm, E.C., Kulkarni, S.R., Graham, M.J., Dekany, R., Smith, R.M., Riddle, R., Masci, F.J., Helou, G., Prince, T.A., Adams, S.M., Barbarino, C., Barlow, T., Bauer, J., Beck, R., Belicki, J., Biswas, R., Blagorodnova, N., Bodewits, D., Bolin, B., Brinnel, V., Brooke, T., Bue, B., Bulla, M., Burruss, R., Cenko, S.B., Chang, C.-K., Connolly, A., Coughlin, M., Cromer, J., Cunningham, V., De, K., Delacroix, A., Desai, V., Duev, D.A., Eadie, G., Farnham, T.L., Feeney, M., Feindt, U., Flynn, D., Franckowiak, A., Frederick, S., Fremling, C., Gal-Yam, A., Gezari, S., Giomi, M., Goldstein, D.A., Golkhou, V.Z., Goobar, A., Groom, S., Hacopians, E., Hale, D., Henning, J., Ho, A.Y.Q., Hover, D., Howell, J., Hung, T., Huppenkothen, D., Imel, D., Ip, W.-H., Ivezić, Ž., Jackson, E., Jones, L., Juric, M., Kasliwal, M.M., Kaspi, S., Kaye, S., Kelley, M.S.P., Kowalski, M., Kramer, E., Kupfer, T., Landry, W., Laher, R.R., Lee, C.-D., Lin, H.W., Lin, Z.-Y., Lunnan, R., Giomi, M., Mahabal, A., Mao, P., Miller, A.A., Monkewitz, S., Murphy, P., Ngeow, C.-C., Nordin, J., Nugent, P., Ofek, E., Patterson, M.T., Penprase, B., Porter, M., Rauch, L., Rebbapragada, U., Reiley, D., Rigault, M., Rodriguez, H., van Roestel, J., Rusholme, B., van Santen, J., Schulze, S., Shupe, D.L., Singer, L.P., Soumagnac, M.T., Stein, R., Surace, J., Sollerman, J., Szkody, P., Taddia, F., Terek, S., Van Sistine, A., van Velzen, S., Vestrand, W.T., Walters, R., Ward, C., Ye, Q.-Z., Yu, P.-C., Yan, L., Zolkower, J., 2019. The Zwicky transient facility: System overview, performance, and first results. Publ. Astron. Soc. Pac. 131 (995), 018002. doi:10.1088/1538-3873/aaecbe, arXiv:1902.01932.

Bishop, C.M., 2006. Pattern recognition and machine learning. In: Information Science and Statistics, Springer, New York.

Bopp, B.W., 1980. By draconis and RS canum venaticorum stars: The discoveries of classical photometry and spectroscopy. Highlights Astron. 5, 847–848. doi:10.1017/S1539299600004846.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32. doi:10.1023/A:1010933404324.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification And Regression Trees, first ed. Routledge, doi:10.1201/9781315139470.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G., 2013. API design for machine learning software: Experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122.

Cabral, J.B., Ramos, F., Gurovich, S., Granitto, P.M., 2020. Automatic catalog of RR Lyrae from ~14 million VVV light curves: How far can we go with traditional machine-learning? Astron. Astrophys. 642, A58. doi:10.1051/0004-6361/202038314, arXiv:2005.00220.

Casper da Costa-Luis, Larroque, S.K., Altendorf, K., Mary, H., richardsheridan, Korobov, M., Noam Yorav-Raphael, Ivanov, I., Bargull, M., Rodrigues, N., Chen, G., Dektyarev, M., mjstevens777, Pagel, M.D., Zugnoni, M., JC, CrazyPython, Newey, C., Lee, A., pgajdos, Todd, Malmgren, S., redbug312, Desh, O., Nechaev, N., Górny, M., Nordlund, M., MapleCCC, McCracken, J., 2024. Tqdm: A fast, extensible progress bar for Python and CLI. doi:10.5281/ZENODO.595120.

Cha, S.-H., 2007. Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Models Methods Appl. Sci. 1 (4), 300–307.

Chaini, S., Mahabal, A., Kembhavi, A., Bianco, F.B., 2024. DistClassiPy: Distance-based light curve classification. Astrophysics Source Code Library, record ascl:2403.002.

Chen, X., Wang, S., Deng, L., de Grijs, R., Yang, M., Tian, H., 2020. The Zwicky transient facility catalog of periodic variable stars. Astrophys. J. Suppl. Ser. 249 (1), 18. doi:10.3847/1538-4365/ab9cae, arXiv:2005.08662.

Cheung, S.-H., Villar, V.A., Chan, H.-S., Ho, S., 2021. A new classification model for the ZTF catalog of periodic variable stars. Res. Notes AAS 5 (12), 282. doi:10.3847/2515-5172/ac4159, arXiv:2112.04010.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Trans. Inform. Theory 13 (1), 21–27. doi:10.1109/TIT.1967.1053964.

Deza, M.M., Deza, E., 2013. Encyclopedia of Distances. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-30958-8.

Djorgovski, S.G., Drake, A.J., Mahabal, A.A., Graham, M.J., Donalek, C., Williams, R., Beshore, E.C., Larson, S.M., Prieto, J., Catelan, M., Christensen, E., McNaught, R.H., 2011. The Catalina real-time transient survey (CRTS). arXiv:1102.5004 [astro-ph].

Djorgovski, S.G., Mahabal, A., Drake, A., Graham, M., Donalek, C., 2013. Sky surveys. In: Planets, Stars and Stellar Systems: Volume 2: Astronomical Techniques, Software, and Data. Springer Netherlands, Dordrecht, pp. 223–281. doi:10.1007/978-94-007-5618-2_5.

Eyer, L., Blake, C., 2002. Automated classification of variable stars for ASAS data. In: Aerts, C., Bedding, T.R., Christensen-Dalsgaard, J. (Eds.), IAU Colloq. 185: Radial and Nonradial Pulsationsn As Probes of Stellar Physics. In: Astronomical Society of the Pacific Conference Series, vol. 259, p. 160, arXiv:astro-ph/0110071.

Eyer, L., Mowlavi, N., 2008. Variable stars across the observational HR diagram. J. Phys.: Conf. Ser. 118 (1), 012010.

Ferri, F., Pudil, P., Hatef, M., Kittler, J., 1994. Comparative study of techniques for large-scale feature selection* *this work was suported by a SERC grant GR/E 97549. the first author was also supported by a FPI grant from the Spanish MEC, PF92 73546684. In: Machine Intelligence and Pattern Recognition, vol. 16, Elsevier, pp. 403–413. doi:10.1016/B978-0-444-81892-8.50040-7.

Fix, E., Hodges, J.L., 1989. Discriminatory analysis. Nonparametric discrimination: consistency properties. Int. Stat. Rev. / Rev. Int. Stat. 57 (3), 238. doi:10.2307/1403797.

Förster, F., Cabrera-Vives, G., Castillo-Navarrete, E., Estévez, P.A., Sánchez-Sáez, P., Arredondo, J., Bauer, F.E., Carrasco-Davis, R., Catelan, M., Elorrieta, F., Eyheramendy, S., Huijse, P., Pignata, G., Reyes, E., Reyes, I., Rodríguez-Mancini, D., Ruz-Mieres, D., Valenzuela, C., Alvarez-Maldonado, I., Astorga, N., Borissova, J., Clocchiatti, A., De Cicco, D., Donoso-Oliva, C., Graham, M.J., Kurtev, R., Mahabal, A., Maureira, J.C., Molina-Ferreiro, R., Moya, A., Palma, W., Pérez-Carrasco, M., Protopapas, P., Romero, M., Sabatini-Gacitúa, L., Sánchez, A., Martín, J.S., Sepúlveda-Cobo, C., Vera, E., Vergara, J.R., 2021. The automatic learning for the rapid classification of events (alerce) alert broker. The Astronomical Journal 161 (5), 242. doi:10.3847/1538-3881/abe9bc, arXiv:2008.03303.

Graham, M.J., Djorgovski, S.G., Drake, A.J., Stern, D., Mahabal, A.A., Glikman, E., Larson, S., Christensen, E., 2017. Understanding extreme quasar optical variability with CRTS - I. Major AGN flares. Mon. Not. R. Astron. Soc. 470 (4), 4112–4132. doi:10.1093/mnras/stx1456, arXiv:1706.03079.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with numpy. Nature 585 (7825), 357–362. doi:10.1038/s41586-020-2649-2, arXiv:2006.10256.

He, Y., Chen, W., Chen, Y., Mao, Y., 2013. Kernel density metric learning. In: 2013 IEEE 13th International Conference on Data Mining. IEEE, Dallas, TX, USA, pp. 271–280. doi:10.1109/ICDM.2013.153.

Hinners, T., Tat, K., Thorp, R., 2018. Machine learning techniques for Stellar light curve classification. Astron. J. 156 (1), 7. doi:10.3847/1538-3881/aac16d, arXiv:1710.06804.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9 (3), 90–95. doi:10.1109/MCSE.2007.55.

Ivezić, Ž., Kahn, S.M., Tyson, J.A., Abel, B., Acosta, E., Allsman, R., Alonso, D., AlSayyad, Y., Anderson, S.F., Andrew, J., Angel, J.R.P., Angeli, G.Z., Ansari, R., Antilogus, P., Araujo, C., Armstrong, R., Arndt, K.T., Astier, P., Aubourg, É., Auza, N., Axelrod, T.S., Bard, D.J., Barr, J.D., Barrau, A., Bartlett, J.G., Bauer, A.E., Bauman, B.J., Baumont, S., Becker, A.C., Becla, J., Beldica, C., Bellavia, S., Bianco, F.B., Biswas, R., Blanc, G., Blazek, J., Blandford, R.D., Bloom, J.S.,

Bogart, J., Bond, T.W., Borgland, A.W., Borne, K., Bosch, J.F., Boutigny, D., Brackett, C.A., Bradshaw, A., Brandt, W.N., Brown, M.E., Bullock, J.S., Burchat, P., Burke, D.L., Cagnoli, G., Calabrese, D., Callahan, S., Callen, A.L., Chandrasekharan, S., Charles-Emerson, G., Chesley, S., Cheu, E.C., Chiang, H.-F., Chiang, J., Chirino, C., Chow, D., Ciardi, D.R., Claver, C.F., Cohen-Tanugi, J., Cockrum, J.J., Coles, R., Connolly, A.J., Cook, K.H., Cooray, A., Covey, K.R., Cribbs, C., Cui, W., Cutri, R., Daly, P.N., Daniel, S.F., Daruich, F., Daubard, G., Daues, G., Dawson, W., Delgado, F., Dellapenna, A., de Peyster, R., de Val-Borro, M., Digel, S.W., Doherty, P., Dubois, R., Dubois-Felsmann, G.P., Durech, J., Economou, F., Eracleous, M., Ferguson, H., Figueroa, E., Fisher-Levine, M., Focke, W., Foss, M.D., Frank, J., Freemon, M.D., Gangler, E., Gawiser, E., Geary, J.C., Gee, P., Geha, M., Gessner, C.J.B., Gibson, R.R., Gilmore, D.K., Glanzman, T., Glick, W., Goldina, T., Goldstein, D.A., Goodenow, I., Graham, M.L., Gressler, W.J., Gris, P., Guy, L.P., Guyonnet, A., Haller, G., Harris, R., Hascall, P.A., Haupt, J., Hernandez, F., Herrmann, S., Hileman, E., Hoblitt, J., Hodgson, J.A., Hogan, C., Huang, D., Huffer, M.E., Ingraham, P., Innes, W.R., Jacoby, S.H., Jain, B., Jammes, F., Jee, J., Jenness, T., Jernigan, G., Jevremović, D., Johns, K., Johnson, A.S., Johnson, M.W.G., Jones, R.L., Juramy-Gilles, C., Jurić, M., Kalirai, J.S., Kallivayalil, N.J., Kalmbach, B., Kantor, J.P., Karst, P., Kasliwal, M.M., Kelly, H., Kessler, R., Kinnison, V., Kirkby, D., Knox, L., Kotov, I.V., Krabbendam, V.L., Krughoff, K.S., Kubánek, P., Kuczewski, J., Kulkarni, S., Ku, J., Kurita, N.R., Lage, C.S., Lambert, R., Lange, T., Langton, J.B., Guillou, L.L., Levine, D., Liang, M., Lim, K.-T., Lintott, C.J., Long, K.E., Lopez, M., Lotz, P.J., Lupton, R.H., Lust, N.B., MacArthur, L.A., Mahabal, A., Mandelbaum, R., Marsh, D.S., Marshall, P.J., Marshall, S., May, M., McKercher, R., McQueen, M., Meyers, J., Migliore, M., Miller, M., Mills, D.J., Miraval, C., Moeyens, J., Monet, D.G., Moniez, M., Monkewitz, S., Montgomery, C., Mueller, F., Muller, G.P., Arancibia, F.M., Neill, D.R., Newbry, S.P., Nief, J.-Y., Nomerotski, A., Nordby, M., O'Connor, P., Oliver, J., Olivier, S.S., Olsen, K., O'Mullane, W., Ortiz, S., Osier, S., Owen, R.E., Pain, R., Palecek, P.E., Parejko, J.K., Parsons, J.B., Pease, N.M., Peterson, J.M., Peterson, J.R., Petravick, D.L., Petrick, M.E.L., Petry, C.E., Pierfederici, F., Pietrowicz, S., Pike, R., Pinto, P.A., Plante, R., Plate, S., Price, P.A., Prouza, M., Radeka, V., Rajagopal, J., Rasmussen, A.P., Regnault, N., Reil, K.A., Reiss, D.J., Reuter, M.A., Ridgway, S.T., Riot, V.J., Ritz, S., Robinson, S., Roby, W., Roodman, A., Rosing, W., Roucelle, C., Rumore, M.R., Russo, S., Saha, A., Sassolas, B., Schalk, T.L., Schellart, P., Schindler, R.H., Schmidt, S., Schneider, D.P., Schneider, M.D., Schoening, W., Schumacher, G., Schwamb, M.E., Sebag, J., Selvy, B., Sembroski, G.H., Seppala, L.G., Serio, A., Serrano, E., Shaw, R.A., Shipsey, I., Sick, J., Silvestri, N., Slater, C.T., Smith, J.A., Smith, R.C., Sobhani, S., Soldahl, C., Storrie-Lombardi, L., Stover, E., Strauss, M.A., Street, R.A., Stubbs, C.W., Sullivan, I.S., Sweeney, D., Swinbank, J.D., Szalay, A., Takacs, P., Tether, S.A., Thaler, J.J., Thayer, J.G., Thomas, S., Thukral, V., Tice, J., Trilling, D.E., Turri, M., Van Berg, R., Berk, D.V., Vetter, K., Virieux, F., Vucina, T., Wahl, W., Walkowicz, L., Walsh, B., Walter, C.W., Wang, D.L., Wang, S.-Y., Warner, M., Wiecha, O., Willman, B., Winters, S.E., Wittman, D., Wolff, S.C., Wood-Vasey, W.M., Wu, X., Xin, B., Yoachim, P., Zhan, H., 2019. LSST: From science drivers to reference design and anticipated data products. Astrophys. J. 873 (2), 111. doi:10.3847/1538-4357/ab042c, arXiv:0805.2366.

Jainaga, I.R., Arredondo, J., Valenzuela, C., Rodriguez, D., Sanchez, J., Heise, P.H., 2021. Alercebroker/Lc_classifier: Release 1.2.3-P. doi:10.5281/ZENODO.5275453.

Kim, D.-W., Protopapas, P., Bailer-Jones, C.A.L., Byun, Y.-I., Chang, S.-W., Marquette, J.-B., Shin, M.-S., 2014. The EPOCH project: I. periodic variable stars in the EROS-2 LMC database★. Astron. Astrophys. 566, A43. doi:10.1051/0004-6361/201323252, arXiv:1403.6131.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In: Loizides, F., Schmidt, B. (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press, pp. 87–90.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI '95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1137–1143.

Krawczyk, B., 2016. Learning from imbalanced data: Open challenges and future directions. Prog. Artif. Intell. 5 (4), 221–232. doi:10.1007/s13748-016-0094-0.

Mahabal, A., Djorgovski, S.G., Williams, R., Drake, A., Donalek, C., Graham, M., Moghaddam, B., Turmon, M., Jewell, J., Khosla, A., Hensley, B., 2008. Towards real-time classification of astronomical transients. In: Bailer-Jones, C.A.L. (Ed.), Classification and Discovery in Large Astronomical Surveys. In: American Institute of Physics Conference Series, vol. 1082, pp. 287–293. doi:10.1063/1.3059064, arXiv:0810.4527.

Mahalanobis, P.C., 1936. On the generalized distance in statistics. Proc. Natl. Inst. Sci. India 2 (1), 49–55. doi:10.1007/s13171-019-00164-5.

Mirkin, B., Koonin, E., 2003. A top-down method for building genome classification trees with linear binary hierarchies. In: DIMACS series in Discrete Mathematics and Theoretical Computer Science, vol. 61, American Mathematical Society, pp. 97–112.

Murtagh, F., Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. Wiley Interdiscip. Rev.: Data Mining Knowl. Discov. 2 (1), 86–97.

Nayak, S., Bhat, M., Subba Reddy, N.V., Ashwath Rao, B., 2022. Study of distance metrics on k - nearest neighbor algorithm for star categorization. J. Phys. Conf. Ser. 2161 (1), 012004. doi:10.1088/1742-6596/2161/1/012004.

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., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830. doi:10.5555/1953048.2078195, arXiv:1201.0490.

Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., Vrgoč, D., 2016. Foundations of JSON schema. In: Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp. 263–273.

Raschka, S., 2018. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. J. Open Source Softw. 3 (24), 638. doi:10.21105/joss.00638.

Sánchez-Sáez, P., Reyes, I., Valenzuela, C., Förster, F., Eyheramendy, S., Elorrieta, F., Bauer, F.E., Cabrera-Vives, G., Estévez, P.A., Catelan, M., Pignata, G., Huijse, P., De Cicco, D., Arévalo, P., Carrasco-Davis, R., Abril, J., Kurtev, R., Borissova, J., Arredondo, J., Castillo-Navarrete, E., Rodriguez, D., Ruz-Mieres, D., Moya, A., Sabatini-Gacitúa, L., Sepúlveda-Cobo, C., Camacho-Iñiguez, E., 2021. Alert classification for the ALeRCE broker system: The light curve classifier. Astron. J. 161 (3), 141. doi:10.3847/1538-3881/abd5c1, arXiv:2008.03311.

Silverman, B., 2018. Density Estimation for Statistics and Data Analysis, first ed. Routledge, doi:10.1201/9781315140919.

The pandas development team, 2020. Pandas-dev/pandas: Pandas. doi:10.5281/zenodo.3509134, Zenodo.

Thiebaut, C., Boer, M., Roques, S., 2002. Steps towards the development of an automatic classifier for astronomical sources. In: Starck, J.-L., Murtagh, F.D. (Eds.), Astronomical Data Analysis II. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 4847, pp. 379–390. doi:10.1117/12.461027.

Tschopp, M.A., Hernandez-Rivera, E., 2017. Quantifying Similarity and Distance Measures for Vector-Based Datasets: Histograms, Signals, and Probability Distribution Functions. US Army Research Laboratory.

Van Rossum, G., Drake, F.L., 2009. Python 3 Reference Manual. CreateSpace, Scotts Valley, CA.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17, 261–272. doi:10.1038/s41592-019-0686-2, arXiv:1907.10121.

Waskom, M.L., 2021. seaborn: Statistical data visualization. J. Open Source Softw. 6 (60), 3021. doi:10.21105/joss.03021.

Wes McKinney, 2010. Data structures for statistical computing in Python. In: van der Walt, S., Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference. pp. 56–61. doi:10.25080/Majora-92bf1922-00a.

York, D.G., Adelman, J., Anderson, J., Anderson, S.F., Annis, J., Bahcall, N.A., Bakken, J.A., Barkhouser, R., Bastian, S., Berman, E., Boroski, W.N., Bracker, S., Briegel, C., Briggs, J.W., Brinkmann, J., Brunner, R., Burles, S., Carey, L., Carr, M.A., Castander, F.J., Chen, B., Colestock, P.L., Connolly, A.J., Crocker, J.H., Csabai, I., Czarapata, P.C., Davis, J.E., Doi, M., Dombeck, T., Eisenstein, D., Ellman, N., Elms, B.R., Evans, M.L., Fan, X., Federwitz, G.R., Fiscelli, L., Friedman, S., Frieman, J.A., Fukugita, M., Gillespie, B., Gunn, J.E., Gurbani, V.K., de Haas, E., Haldeman, M., Harris, F.H., Hayes, J., Heckman, T.M., Hennessy, G.S., Hindsley, R.B., Holm, S., Holmgren, D.J., Huang, C.-h., Hull, C., Husby, D., Ichikawa, S.-I., Ichikawa, T., Ivezić, Ž., Kent, S., Kim, R.S.J., Kinney, E., Klaene, M., Kleinman, A.N., Kleinman, S., Knapp, G.R., Korienek, J., Kron, R.G., Kunszt, P.Z., Lamb, D.Q., Lee, B., Leger, R.F., Limmongkol, S., Lindenmeyer, C., Long, D.C., Loomis, C., Loveday, J., Lucinio, R., Lupton, R.H., MacKinnon, B., Mannery, E.J., Mantsch, P.M., Margon, B., McGehee, P., McKay, T.A., Meiksin, A., Merelli, A., Monet, D.G., Munn, J.A., Narayanan, V.K., Nash, T., Neilsen, E., Neswold, R., Newberg, H.J., Nichol, R.C., Nicinski, T., Nonino, M., Okada, N., Okamura, S., Ostriker, J.P., Owen, R., Pauls, A.G., Peoples, J., Peterson, R.L., Petravick, D., Pier, J.R., Pope, A., Pordes, R., Prosapio, A., Rechenmacher, R., Quinn, T.R., Richards, G.T., Richmond, M.W., Rivetta, C.H., Rockosi, C.M., Ruthmansdorfer, K., Sandford, D., Schlegel, D.J., Schneider, D.P., Sekiguchi, M., Sergey, G., Shimasaku, K., Siegmund, W.A., Smee, S., Smith, J.A., Snedden, S., Stone, R., Stoughton, C., Strauss, M.A., Stubbs, C., SubbaRao, M., Szalay, A.S., Szapudi, I., Szokoly, G.P., Thakar, A.R., Tremonti, C., Tucker, D.L., Uomoto, A., Vanden Berk, D., Vogeley, M.S., Waddell, P., Wang, S.-i., Watanabe, M., Weinberg, D.H., Yanny, B., Yasuda, N., SDSS Collaboration, 2000. The sloan digital sky survey: Technical summary. AJ 120 (3), 1579–1587. doi:10.1086/301513, arXiv:astro-ph/0006396.

Zielezinski, A., 2023. Statistical-distance: Distance measures to compare two probability density functions.