Supporting Information for "Identifying deep moonquake nests using machine learning model on single lunar station on the far side of the Moon"

Josipa Majstorović[1], Philippe Lognonné[1], Taichi Kawamura[1], Mark Panning[2]

[1]Université Paris Cité, Institut de physique du globe de Paris, CNRS, Paris, France

[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

Contents of this file

1. Text S1 to S3
2. Figures S1 to S24

This file contains supplement text, and figures for the manuscript "Identifying deep moonquake nests using machine learning model on single lunar station on the far side of the Moon". Supplemental Text S1 provides a detailed description of P- and S- travel times calculating for deep moonquake nests using several lunar models. Supplemental Text S2 details about the Random Forest algorithm and the particularity of its training and testing. Supplemental Text S3 provides additional information on the t-distributed stochastic neighbor embedding (t-sne) method. Supplemental Figure S1 show travel time $t_{sp} = t_s - t_p$ for around 20 nests and seven existing lunar models. Supplemental Figure

Corresponding author: J. Majstorović, Université Paris Cité, Institut de physique du globe de Paris, CNRS, Paris, France (josipa.majstorovic@protonmail.com)

S4 illustrates distributions of features used to train machine learning model for two nests, A1 and A8. Supplemental Figure S2 shows features used as an input data for machine learning model. Supplemental Figures S5 to S13 show statistics of the Random Forest models on the test dataset that are trained to dissociate between A1 and A8 nest. Supplemental Figure S13 shows prediction values and negatively labelled data points for the base Random Forest model trained to dissociate between A1 and A8 nests. Supplemental Figure S14 and S15 illustrate 2-D graphic manifold of the space spanned by the 8-D feature space of A1 and A8 nests. Supplemental Figures S16 to S19 show statistic of the Random Forest model trained and tested on combination of nests A1-A8-A18, A1-A8-A18-A6, A1-A8-A18-A6-A14, A8-A18-A6-A14, respectively. Supplemental Figure S20 illustrates 2D graphic manifold representation of the 8D feature space spanned by five nests A1-A8-A18-A6-A14. Supplemental Figure S21 show nests ratios for different sets. Supplemental Figures S22 to S24 illustrates 2-D manifolds of the feature spanned by different sets.

Text S1. Calculation of travel times of seismic waves between sources and station can be obtained using two programming packages: Python package Obspy and its module 'taup' and TauP Java package, both based on the paper Crotwell, Owens, Ritsema, et al. (1999). In our experiment we placed single station on the far side of the Moon in the Schrödinger Crater, while our sources are located nests from the paper Lognonné, Gagnepain-Beyneix, and Chenet (2003). Also, we utilize the existing lunar interior models from papers Garcia et al. (2019) (ISSI 1, ISSI 2, ISSI 3), Garcia, Gagnepain-Beyneix, Chevrot, and Lognonné (2011), Khan, Connolly, Pommier, and Noir (2014), Matsumoto et al. (2015), Weber, Lin, Garnero, Williams, and Lognonné (2011). We first calculate epicentral distances between

nests and station, then travel time of P- and S- seismic waves for the seven models using Python and Java packages. This leaves us with: $t'_{p;i,j}, t'_{s;i,j}, t''_{p;i,j}, t''_{s;i,j}$ where i indicates nest, j indicates lunar model, t' and t'' travel times calculated using Python and Java, respectively. Next, we calculate the average over P- and S- wave travel times for two programming packages, leaving us with $t_{p;i,j} = (t'_{p;i,j} + t''_{p;i,j})/2$, $t_{s;i,j} = (t'_{s;i,j} + t''_{s;i,j})/2$. Further, we calculate the travel time difference, $t_{sp;i,j} = t_{s;i,j} - t_{p;i,j}$ values that we plot in Figure S1, with i running over the y-axis for all nests and j running over the x-axis over all lunar models. We can notice that some combination of nests and lunar models don't have travel time t_{sp} estimation, and some underestimate or overestimate it, when compared to the average value per nest. Due to these discrepancies we decide to further work with only four models: ISSI 2, ISSI 3 Garcia et al. (2019), Garcia PEPI 2011 (Garcia et al., 2011), Khan JGR 2014 (Khan et al., 2014).

Text S2. As discussed in the main manuscript, Random Forest is a machine learning algorithm that consist of ensemble of randomised decision trees. A decision tree consists of decision (internal) nodes, followed by inequality branches, and leaf nodes that hold the final prediction of individual trees shown in Figure S3. Thus, within each tree the beginning is at root node that doesn't have incoming branches. Next in line are internal nodes where based on the available features/attributes and inequality operations, the decision whether the feature is smaller or larger than some threshold is made. These translate to leaf nodes, which represent all possible outcomes. The hyperparamters that define a RF structure and need to be defined before a training process are: the number of decision trees, the maximum depth of trees, the measure that maximises diversity between classes, the minimum samples in the internal node, and the minimum number of samples in leaf

node for it to be considered, the maximum number of features when looking for the best split, the maximum number of leaf nodes, the maximum samples to be draw from the main training dataset when training each decision tree. We proceed to test the learning robustness of our base model that is trained with normalised features (shown in Figure S6) by carry out several experiments: a) changing the randomness of the bootstrapping initialization of the samples that are used when building decision trees, the randomness of the feature sampling when considering for the best split at each internal node, as well as the randomness of the training and test dataset split (see Figure S7); b) changing the optimal number of decision trees (see Figure S8); c) equalizing the size of nests within the dataset by randomly downsampling the largest nest A1 to be the same size as A8 (see Figure S9); d) reducing the number of input feature data to five most important from the base model ($cos(\gamma)$, $\Delta t_{Perigee}$, d, $\Delta t_{AscendingNode}$, $e_{i+1} - e_i$) (see Figure S10). In all test beside those in experiment a), we keep the random state fixed. Finally, the results do not vary between different tests, indicating that in all above configurations models are able to learn how to classify two nest with the similar performances. Further, we notice that in experiment c) the statistic for A8 nest improved compared to the base model statistic, indicating that having a balanced classes while training a ML is important. Moreover, we observe that the model trained with the fewer features statistically perform worst than the base model. This might be because all eight features are uncorrelated, thus equally important for model learning. Next, we cross-validate our base model. A cross-validation is a technique to assess how the model will generalize to an independent data set by using the resampling technique. A resampling technique uses different portions of the training data to train and validate model during several iteration. Usually, the training

dataset is divided into k equally sized folds, and then k iterations is performed. In each iteration $(k-1)$ folds are used for training, and one fold is used for validation. During the cross-validation, the test set is kept aside. Eventually, the full dataset is divided into three sets: training (55%), validation (20%), test (25%), where training and validation set change in each iteration. We calculate cross-validation with $k = 5$, while keeping the base model parameters. The choice of $k = 5$, has been proven to be a good practice (Witten & James, 2013). This test produces 5 models with the same performance as indicated with ROC curves (see Figure S11). This implies that the base model generalizes well, and to not over fit the results. Moreover, we also perform the grid search over several other RF parameters, besides the number of decision trees. Grid search represents a set of many models, where each model is build with unique set of parameters, and each is trained and tested with the same datasets. The tested parameters are: the maximum features ('auto','sqrt'), the maximum depth of the decision trees (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None), the minimum samples in the internal nodes (2, 5, 10), the minimum number of samples in leaf node (1, 2, 4), the bootstraping technique on or off. However, it seems that the model that performed the best during the grid search (the minimum samples in internal node equal to 10, the minimum samples in leaf node equal to 2, the maximum features equal to 'auto', the maximum depth equal to 110, with bootstraping turned on) does not perform better than our base model (see Figure S12). Even though our base RF model as a final output predicts a class (A1 or A8), it also associates each class prediction with the class probability. This value ranges between 0 and 1, where 1 indicated that a model is absolutely certain that a given event belongs to a predicted class. From our base model the correctly predicated classes score 81% cases higher than

0.80 for test dataset (see Figure S13A). This suggest that model is confident in its prediction. The mislabelled prediction values show uniform distribution of values between 0.5 to 1 (see Figure S13B). We can further examine these mislabelled events from the perspective of the feature input data. We choose three features, $cos(\gamma)$, $\Delta t_{Perigee}$, d, with high importance value. We could argue that the mislabelled data points from both nests show characteristics which better suits the opposite class (see Figure S13C-E). Yet, the decision is not defined using only one feature. To gain an insight how all eight features contribute into separating two nests, we calculate the 2D manifold of their feature space using t-sne method (van der Maaten & Hinton, 2008). Visualisation of the feature space that is color-coded based on two nests, show us that two classes are well but not perfectly separated (see Figure S14). If we further color 2D manifold space with the values of the individual features, we notice that $\Delta t_{AscendingNode}$ feature might be the most responsible for imperfect split between nests (see Figure S15).

Text S3. One of the statistical dimensionality reduction algorithm that helps to visual high-dimensional data is t-distributed Stochastic Neighbor Embedding (t-sne) algorithm (van der Maaten & Hinton, 2008). It is an unsupervised non-linear reduction technique, since it allows us to separate data that cannot be separated by any straight line. Once it is applied on the input data, first, it starts by calculating the probability distribution of neighbours around each points. The term neighbour stands for the set of points that are closest to a given point. In the input original high-dimensional space the probability is modeled as a Gaussian distribution. Second, the algorithm models the probability of neighbours around given points in the lower-dimensional space using a Student's t-distribution. Third, the algorithm minimizes the divergence, usually Kullback–Leibler

divergence, between two probabilities using gradient descent. The result is a lower-dimensional manifold of the data, that still preserves the pairwise similarities between original data points, optimized to a stable state. This optimisation process generates clusters and sub-clusters of similar data points that become visually better understand in the lower-dimensional space by keeping the relationship of the data from the higher-dimensional space. There are several t-sne hyperparameters that need to be adjusted by the user, and the most importnat one is perplexity. The preplexity parameter defines the number of influential neighbours used to calculate the Gaussian probabilities around given point in the high-dimensional space. Its value range from 5 - 50 (Wattenberg et al., 2016), and can significantly impact the resulting mapping of the input data. For our implementation of the t-sne algorithm we use Scikit-learn machine learning Python library (Pedregosa et al., 2011). After trying some combinations we choose to work with the given set of parameters: n_components=2, perplexity=30, n_iter=5000, verbose=1, random_state=133, while keeping the rest of the them as default by the package implementation.

References

Crotwell, H. P., Owens, T. J., Ritsema, J., et al. (1999). The taup toolkit: Flexible seismic travel-time and ray-path utilities. *Seismological Research Letters*, *70*, 154–160.

Garcia, R. F., Gagnepain-Beyneix, J., Chevrot, S., & Lognonné, P. (2011). Very preliminary reference moon model. *Physics of the Earth and Planetary Interiors*, *188*(1-2), 96–113.

Garcia, R. F., Khan, A., Drilleau, M., Margerin, L., Kawamura, T., Sun, D., ... others (2019). Lunar seismology: An update on interior structure models. *Space Science Reviews*, *215*, 1–47.

Khan, A., Connolly, J. A. D., Pommier, A., & Noir, J. (2014). Geophysical evidence for melt in the deep lunar interior and implications for lunar evolution. *Journal of Geophysical Research: Planets*, *119*(10), 2197-2221. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014JE004661` doi: https://doi.org/10.1002/2014JE004661

Lognonné, P., Gagnepain-Beyneix, J., & Chenet, H. (2003). A new seismic model of the moon: implications for structure, thermal evolution and formation of the moon. *Earth and Planetary Science Letters*, *211*(1-2), 27–44.

Matsumoto, K., Yamada, R., Kikuchi, F., Kamata, S., Ishihara, Y., Iwata, T., ... Sasaki, S. (2015). Internal structure of the moon inferred from apollo seismic data and selenodetic data from grail and llr. *Geophysical Research Letters*, *42*(18), 7351-7358. Retrieved from `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015GL065335` doi: https://doi.org/10.1002/2015GL065335

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605. Retrieved from http://jmlr.org/papers/v9/vandermaaten08a.html

Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to use t-sne effectively. *Distill*. Retrieved from http://distill.pub/2016/misread-tsne doi: 10.23915/distill.00002

Weber, R. C., Lin, P.-Y., Garnero, E. J., Williams, Q., & Lognonné, P. (2011). Seismic detection of the lunar core. *Science*, *331*(6015), 309-312. Retrieved from https://www.science.org/doi/abs/10.1126/science.1199375 doi: 10.1126/science.1199375

Witten, D., & James, G. (2013). *An introduction to statistical learning with applications in r*. springer publication.

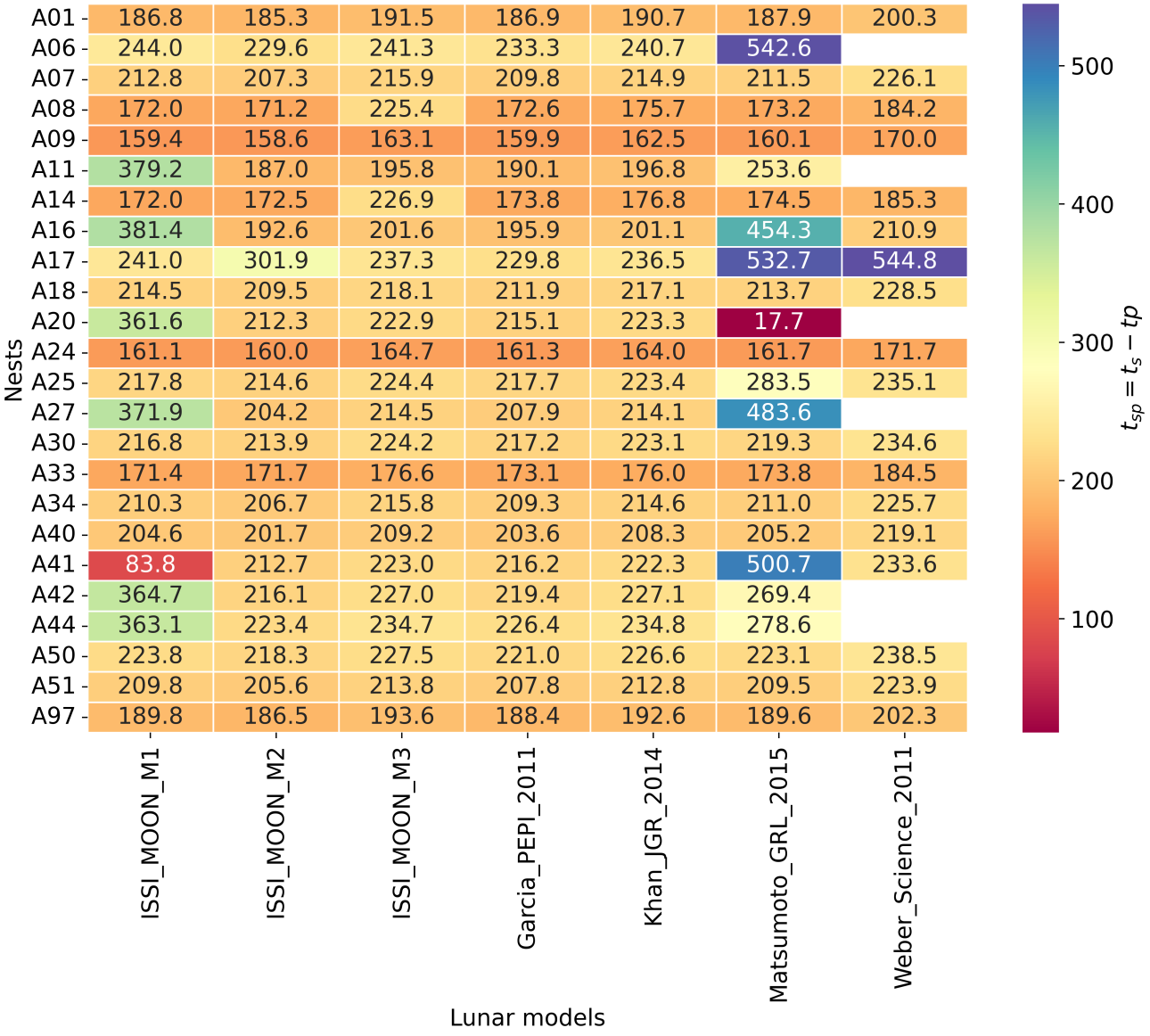

Figure S1. Travel time $t_{sp} = t_s - t_p$ calculations, where t_s and t_p stands for S- and P- waves travel times, respectively, between the range of nests and the station placed on the far side of the Moon in the Schrödinger crater. Calculation are done for seven existing lunar model, from papers Garcia et al. (2019) (ISSI M1, ISSI M2, ISSI M3), Garcia et al. (2011), Khan et al. (2014), Matsumoto et al. (2015), Weber et al. (2011).

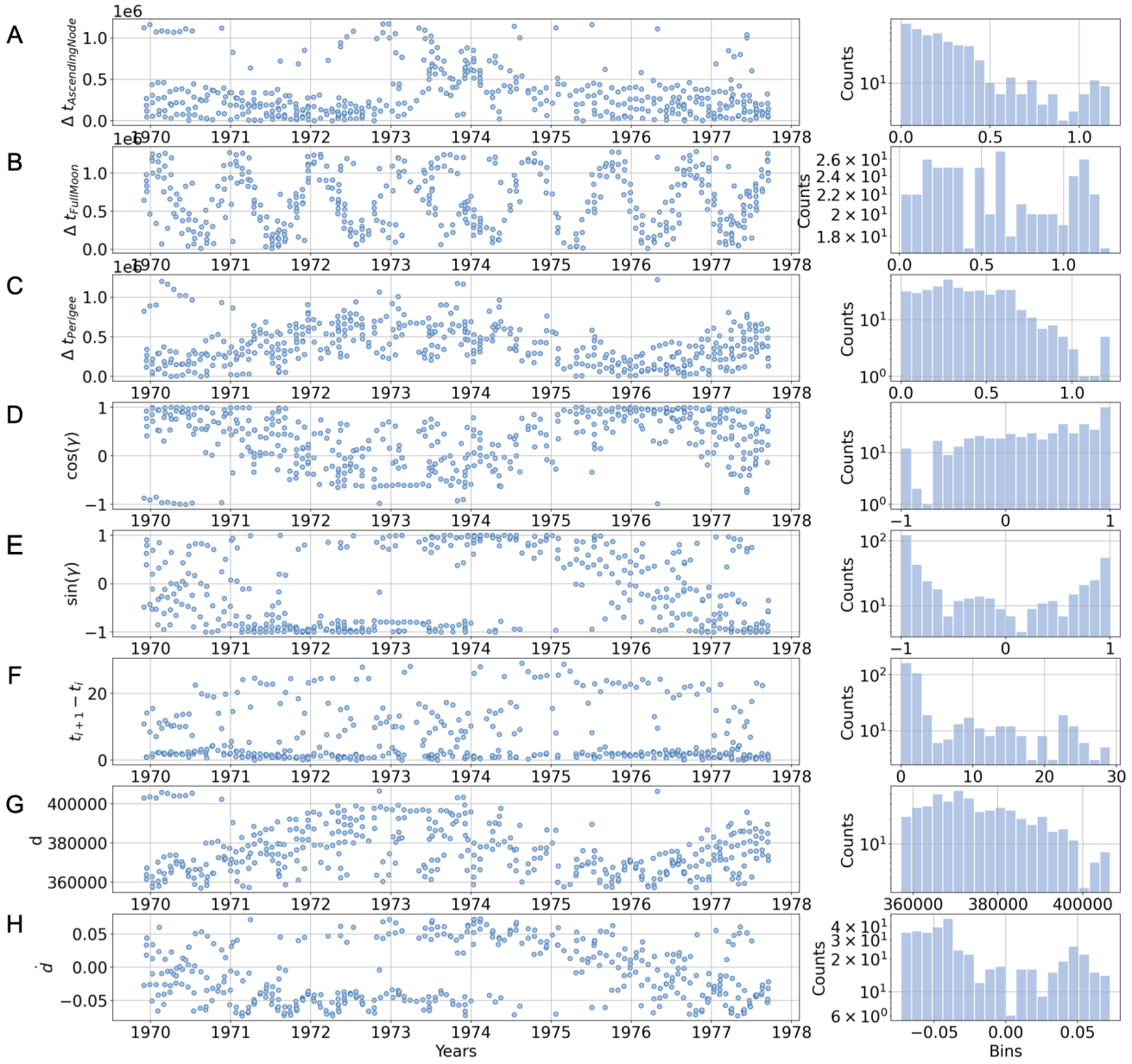

Figure S2. Time evolution of features during the Apollo mission and their histograms for A1 nest. First three features are time difference between quakes and A) the instance when Moon was passing through ascending node, B) Full Moon phase, C) instance when Moon was in perigee; next D) $\cos\gamma$, E) $\sin\gamma$ where γ is the true anomaly angle, indicating the position of the Moon in the orbit; F) time difference between two quakes, G) distance between Moon and Earth at the quake occurrence, H) the rate of distance change from G.

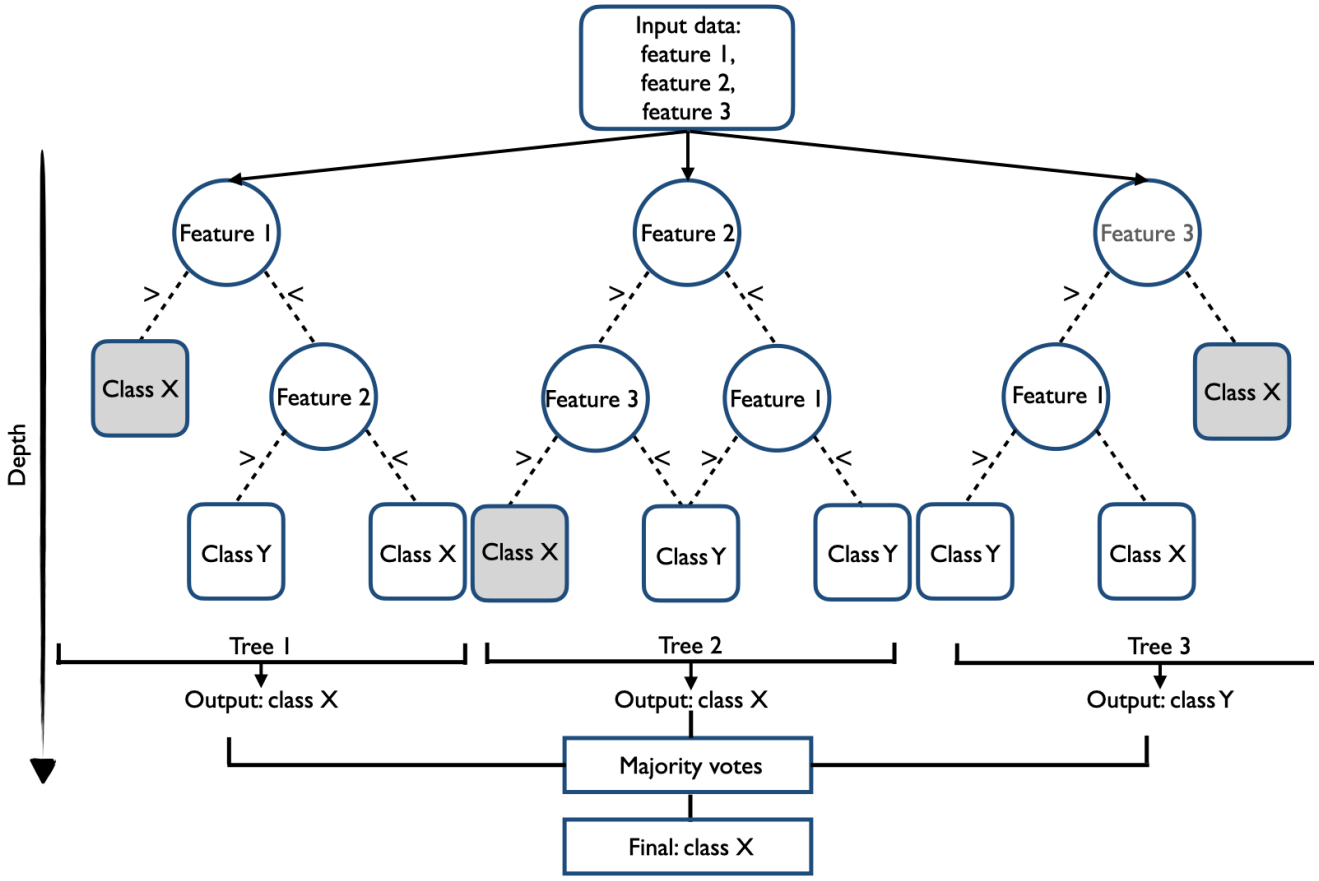

Figure S3. Schematic representation of a Random Forest algorithm. In this example, the model is trained with 3 features, it consists of 3 decision trees with a maximum tree depth equal to 3. A tree consists of decision nodes (circles), followed by inequality branches (dashed lines) , and leaf nodes (rectangles). The prediction is taking place in each tree by yes or no questions, while the final prediction is made upon majority voting considering individual tree predictions.

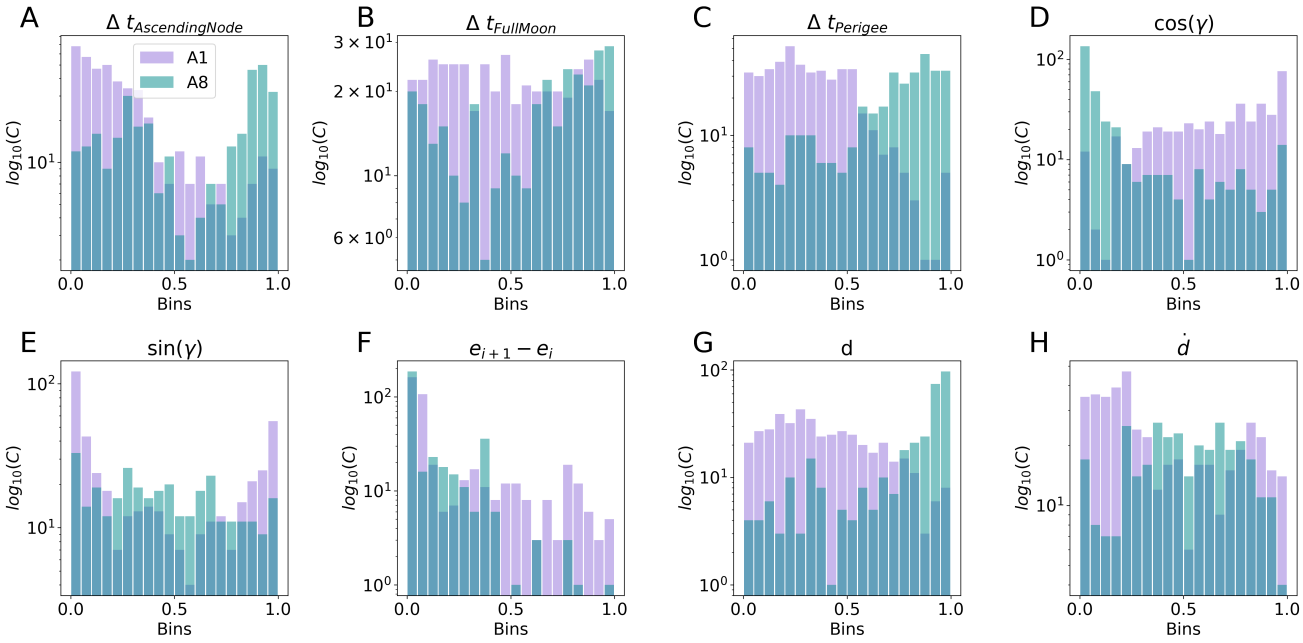

Figure S4. Distributions of eight features used for training Random Forest model for dissociating between nests A1 and A8.

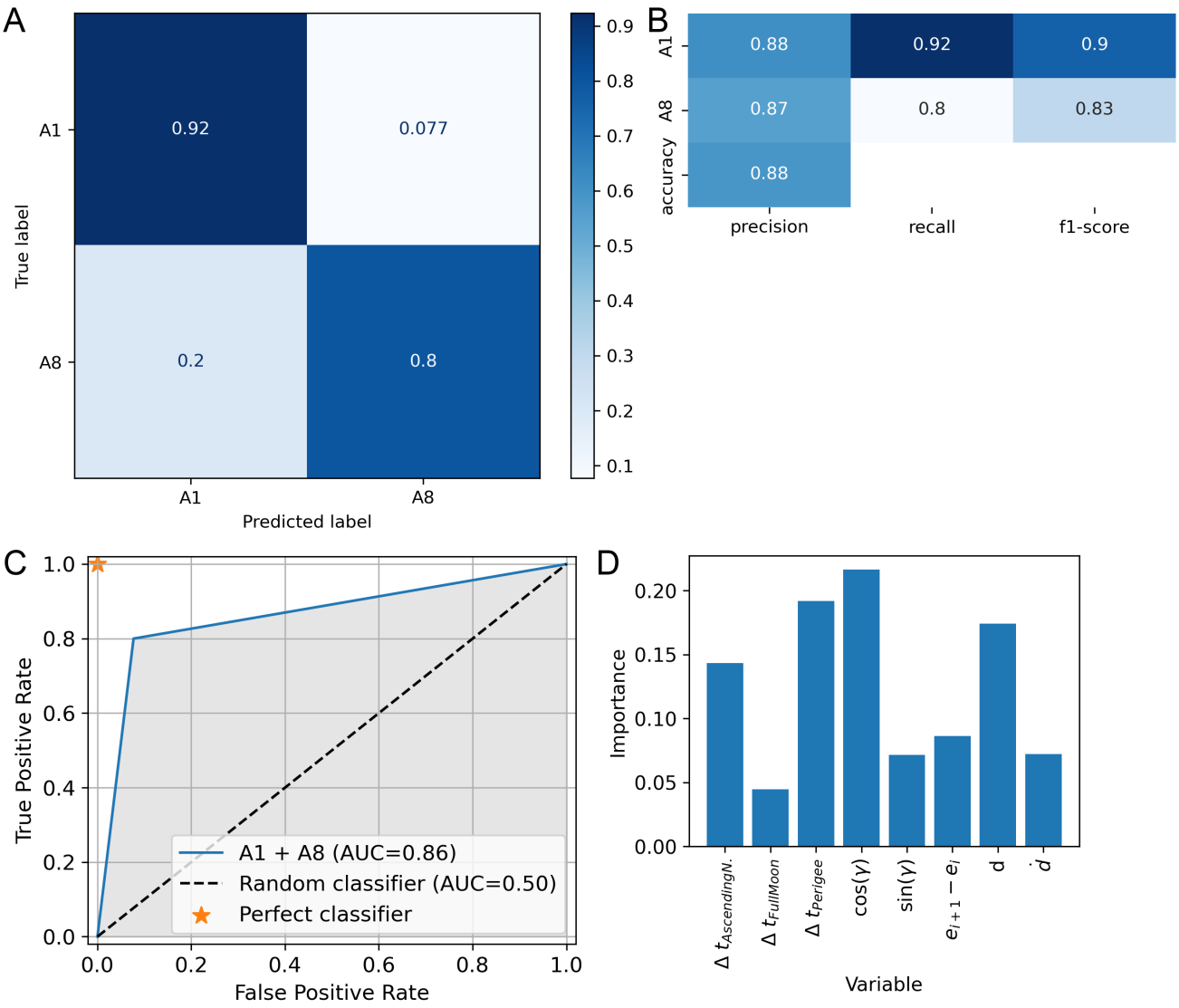

Figure S5. Statistics performance of the base Random Forest model on the dataset trained to dissociate between nests A1 and A8 using raw feature data without normalisation: A) confusion matrix, B) precision, recall, f1-score per nests and accuracy of the model, C) receiver operating characteristic (ROC) curve, D) feature importance for the model to make decisions.

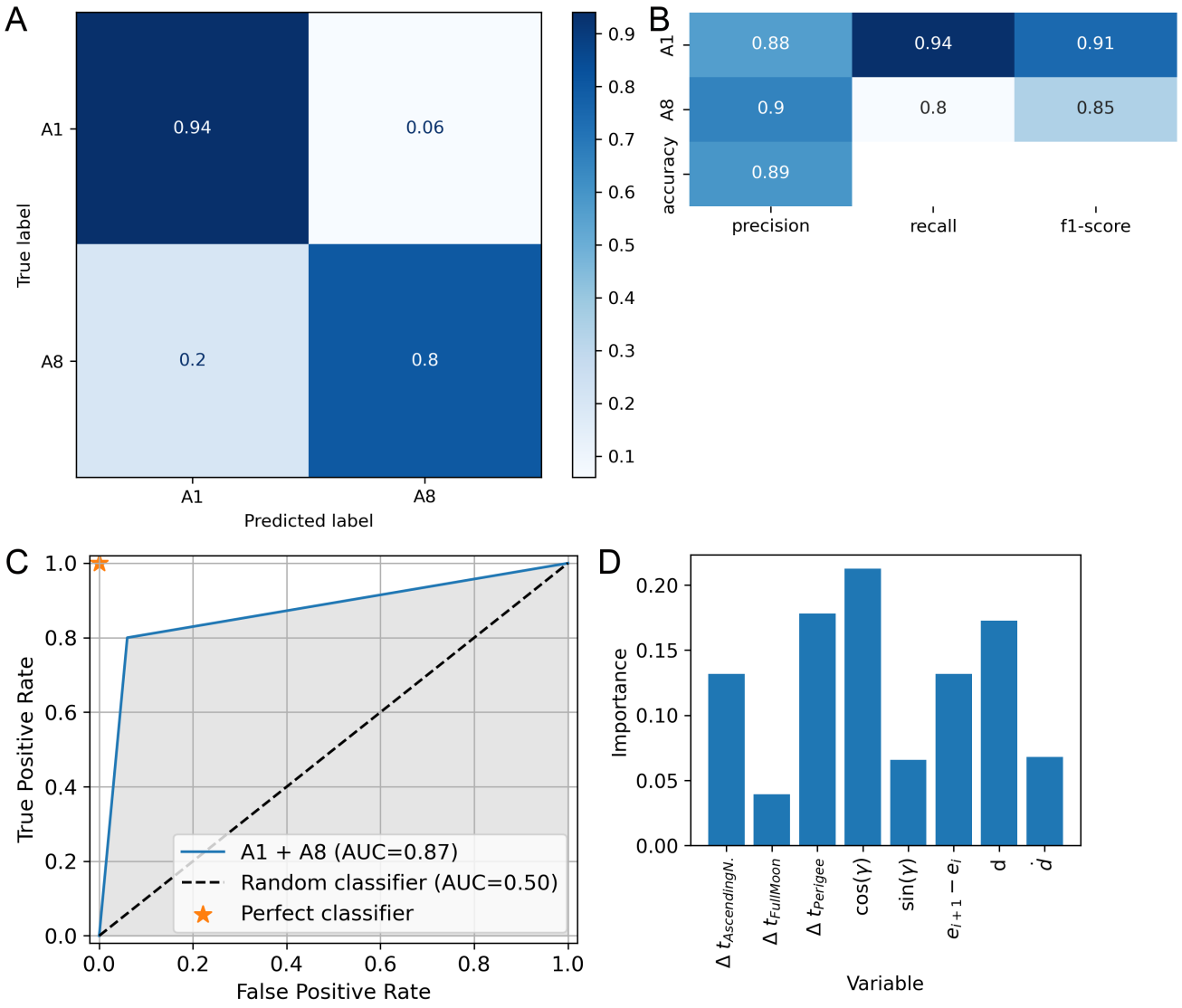

Figure S6. Same as Figure S5 but using feature data that are normalised between 0 and 1.

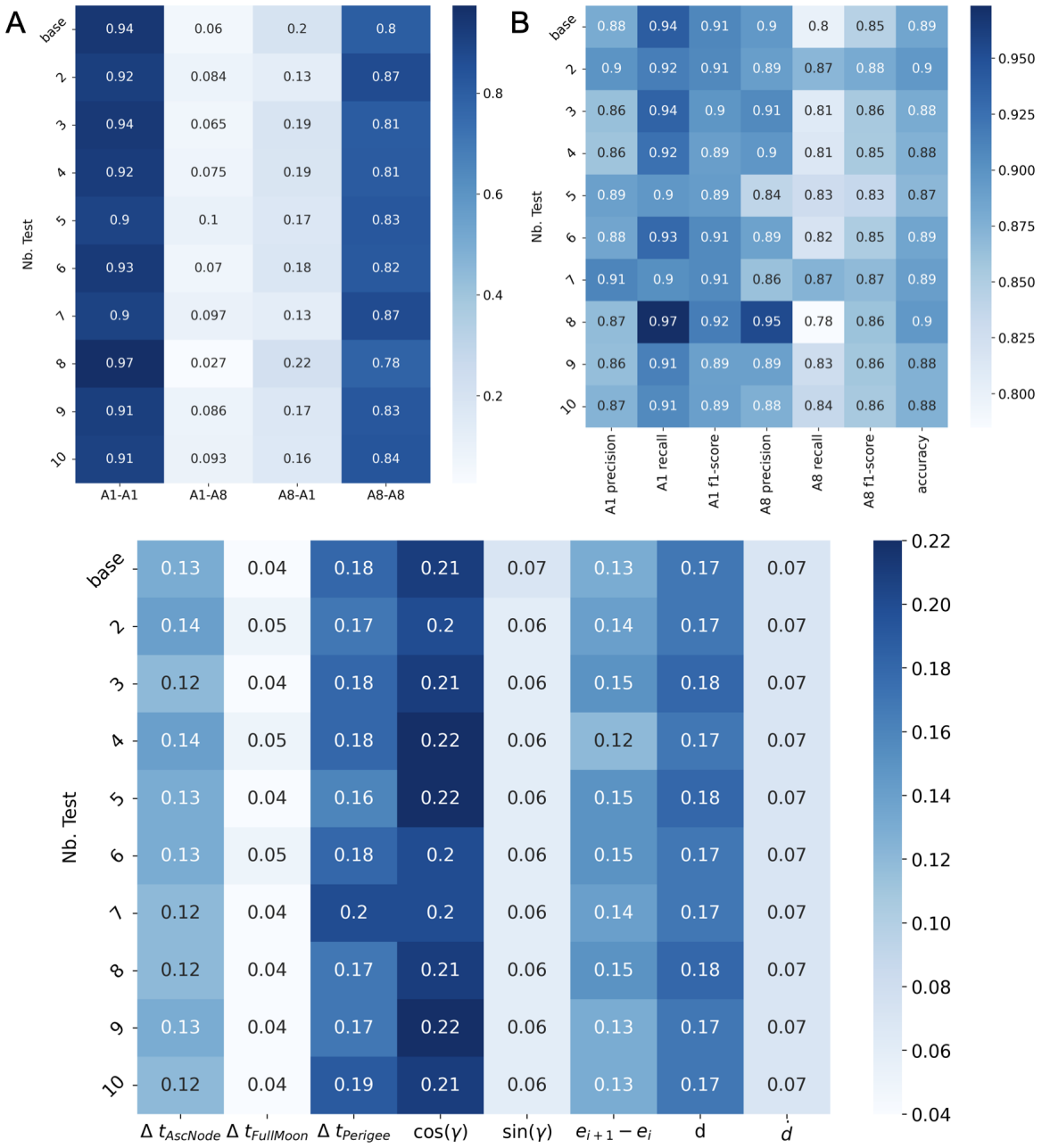

Figure S7. Statistics performance of the Random Forest models while changing the randomness of the data split and decision tree initialisation compared to the base model shown in Figure S6. The models are trained to dissociated between nests A1 and A8. The randomness is changed from 600 to 1400 from test 1 to test 10 by step of 100. Statistics are: A) confusion matrix, B) precision, recal, f1-score and accuracy of the model, C) the importance of the feature used by the models to make a correct classification.

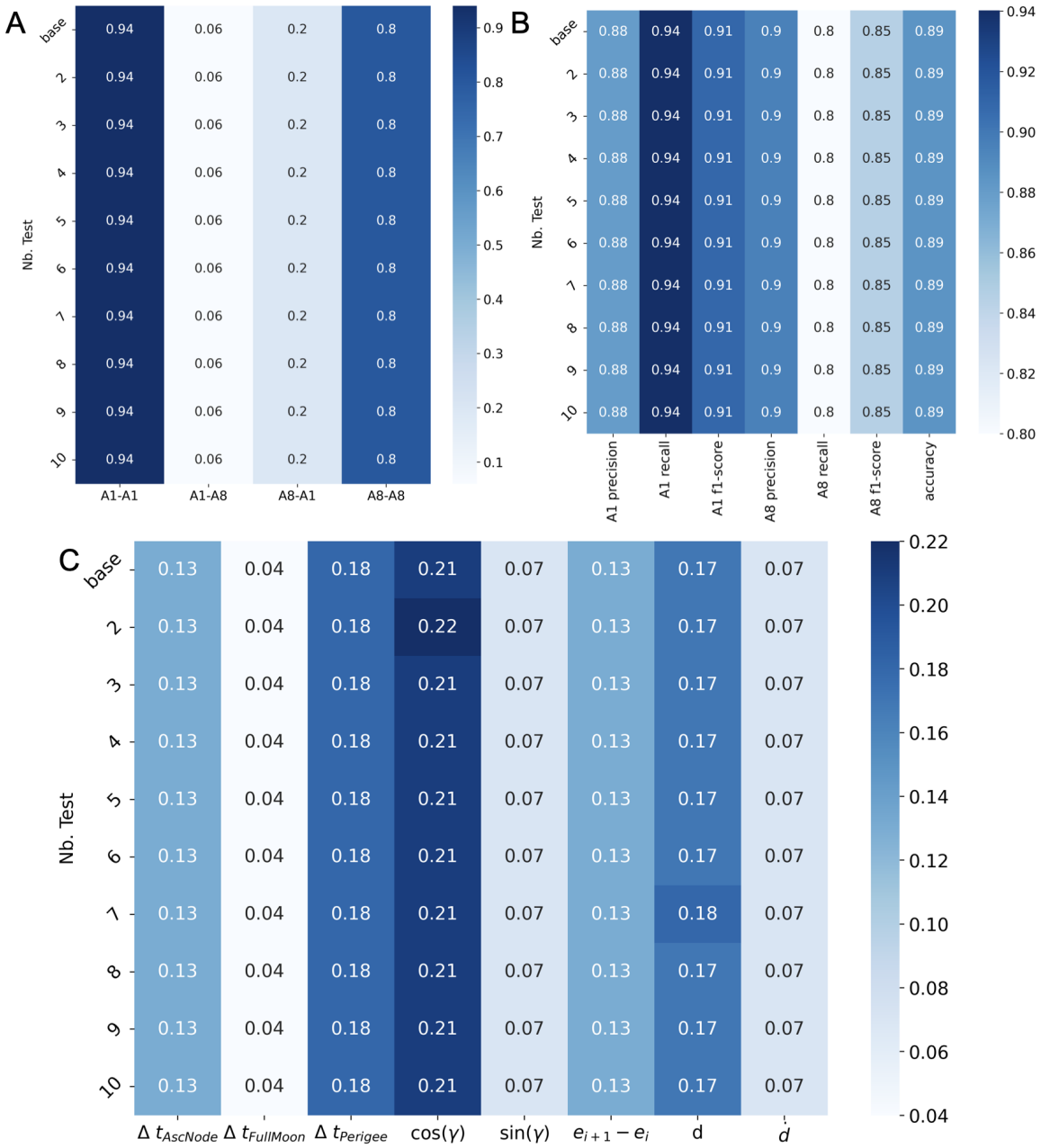

Figure S8. Same as Figure S7 while keeping the randomness fixed, but changing the number of trees used to build Random Forest model.

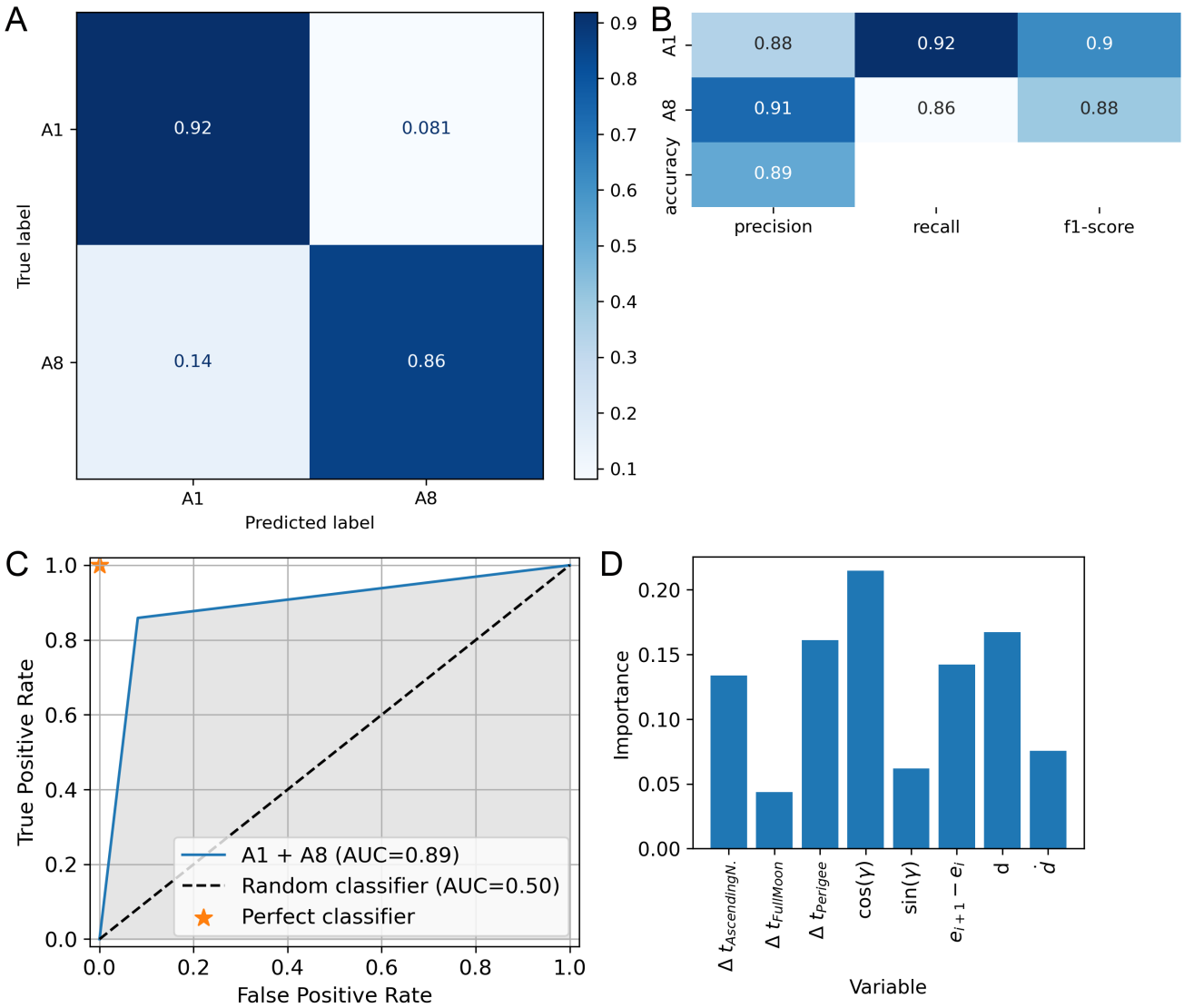

Figure S9. Same as Figure S6, but using the balanced dataset, thus having the same number of A1 and A8 events.

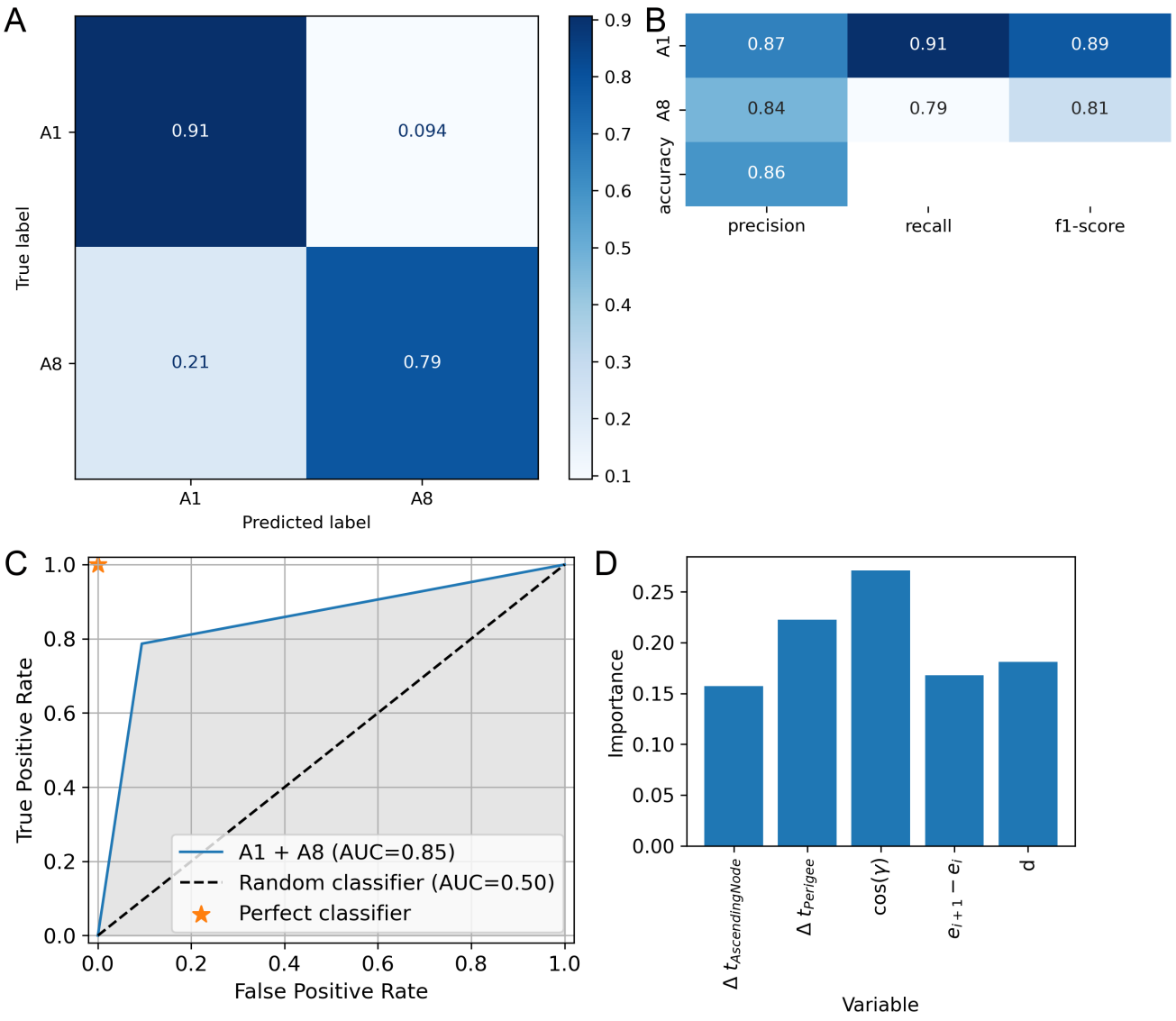

Figure S10. Same as Figure S6, but keeping only five out of eight features: $\Delta t_{AscendingNode}$, $\Delta t_{Perigee}$, $\cos(\gamma)$, $e_{i+1} - e_i$, d.

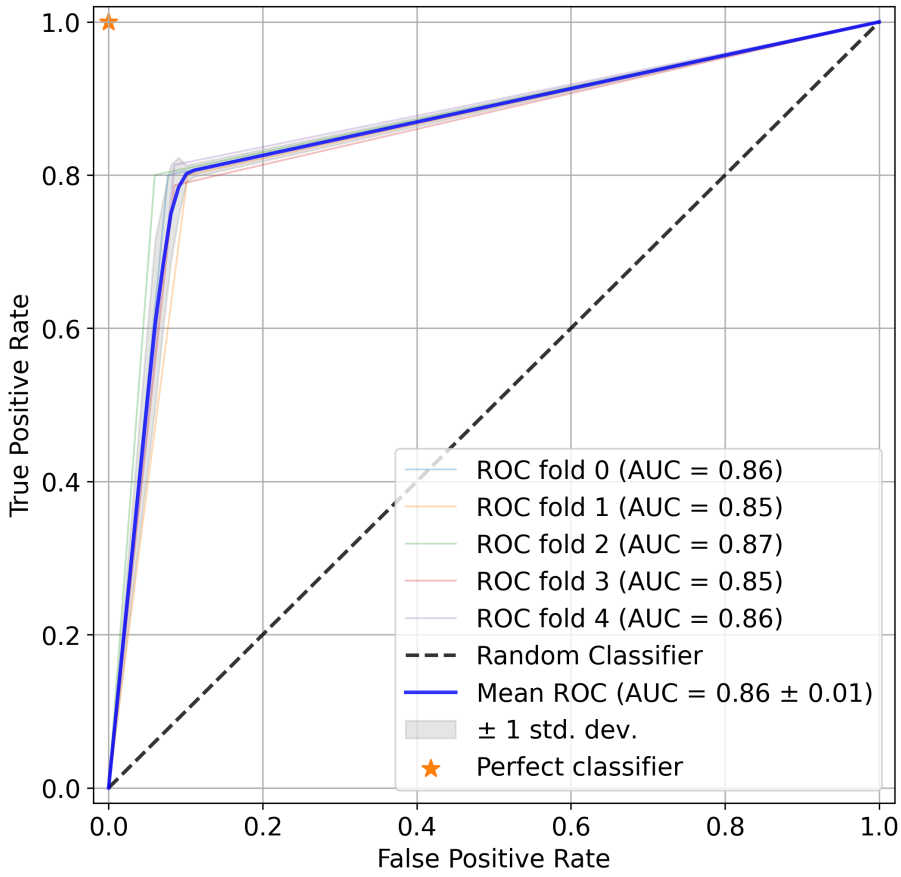

Figure S11. 5-fold cross-validation of the base RF model shown in Figure S6 with the mean and standard deviation.

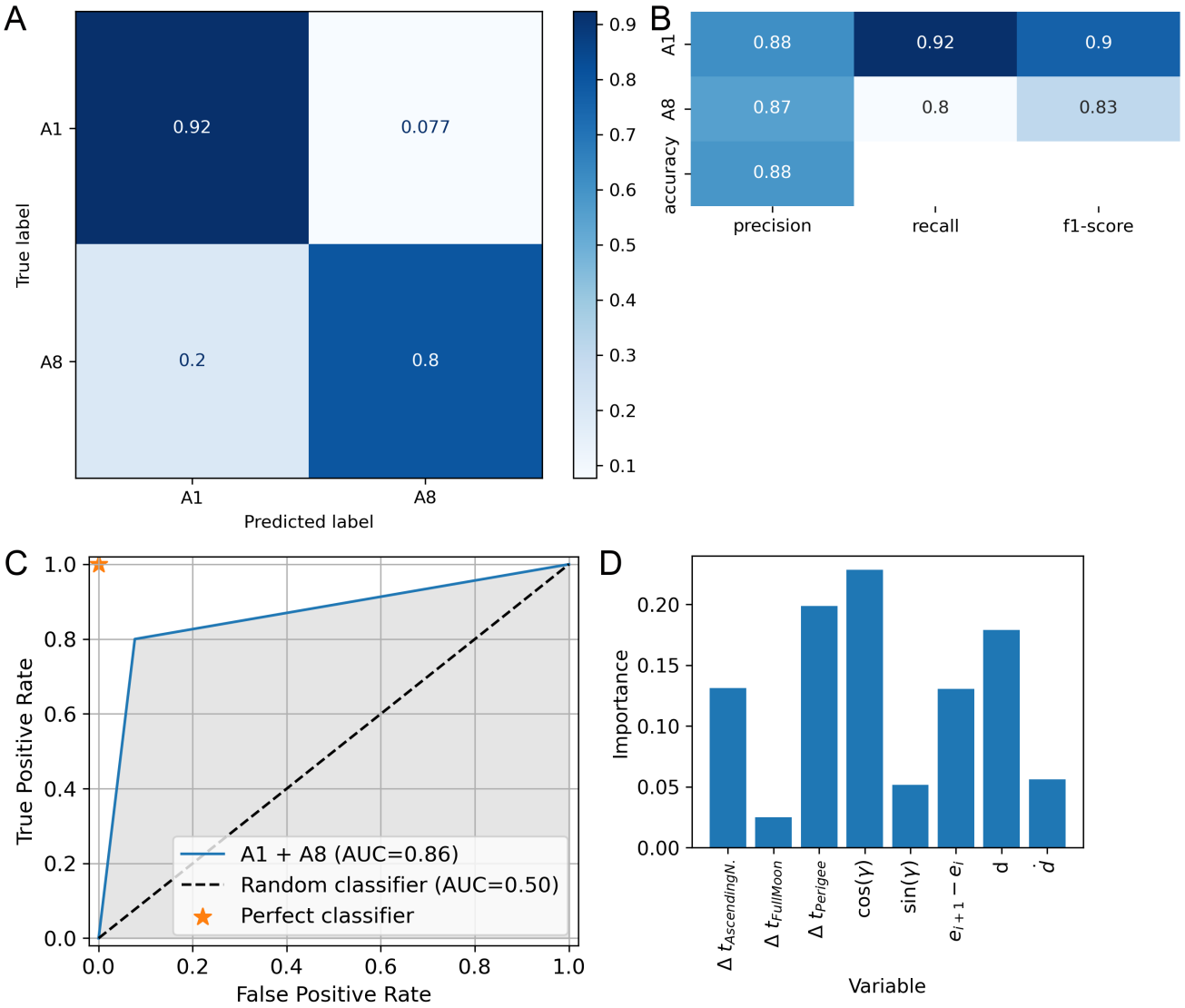

Figure S12. Same as Figure S6, but for the best performing model from the grid search analysis.

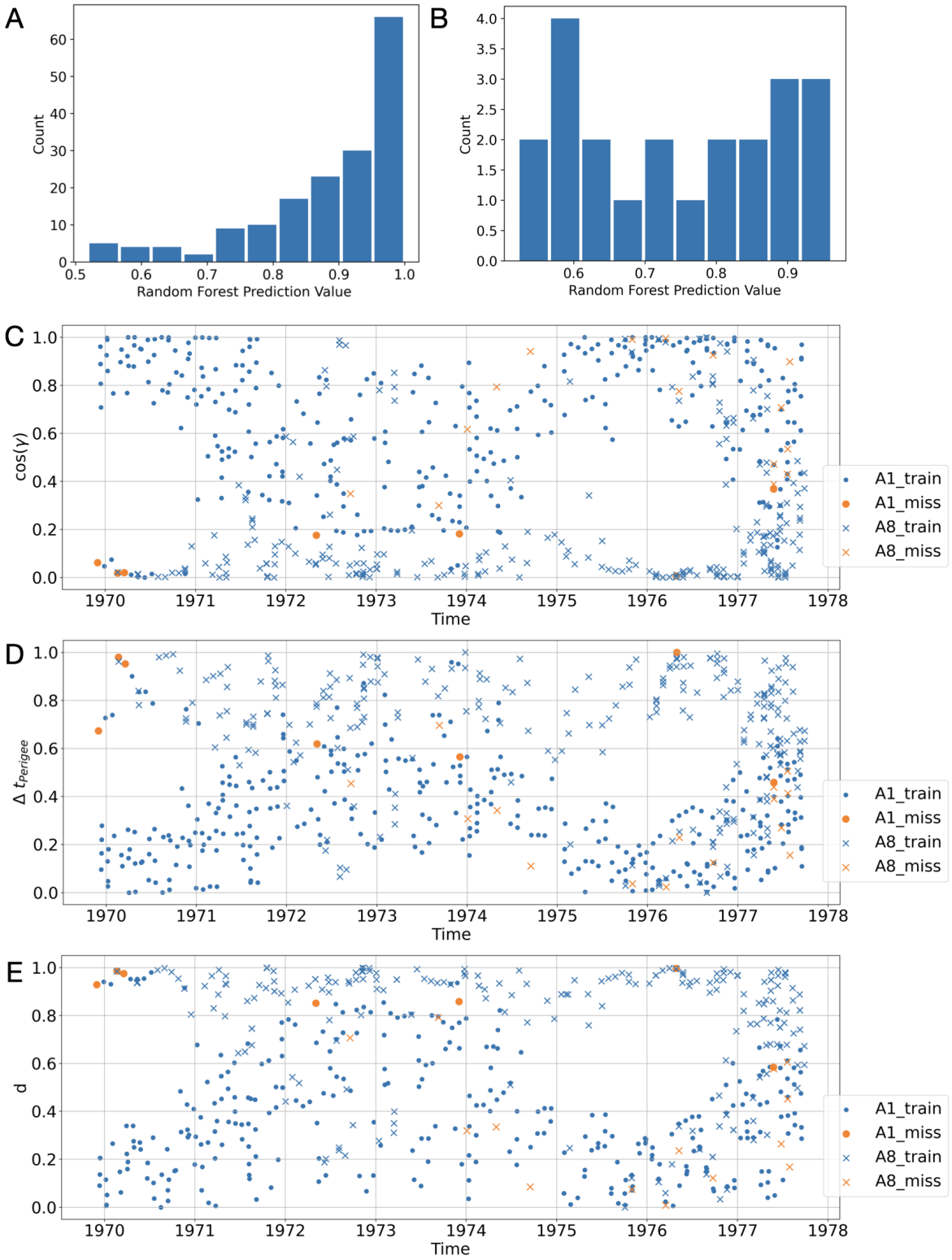

Figure S13. Classification results: A) prediction values for positively classified events, B) prediction values for negatively classified events. Events used for training and events used for testing but got mislabeled form the perspective of features: C) $\cos(\gamma)$, D) $\Delta t_{Perigee}$, E) d.

Figure S14. 2-D manifold of the feature space spanned by nests A1 and A8.

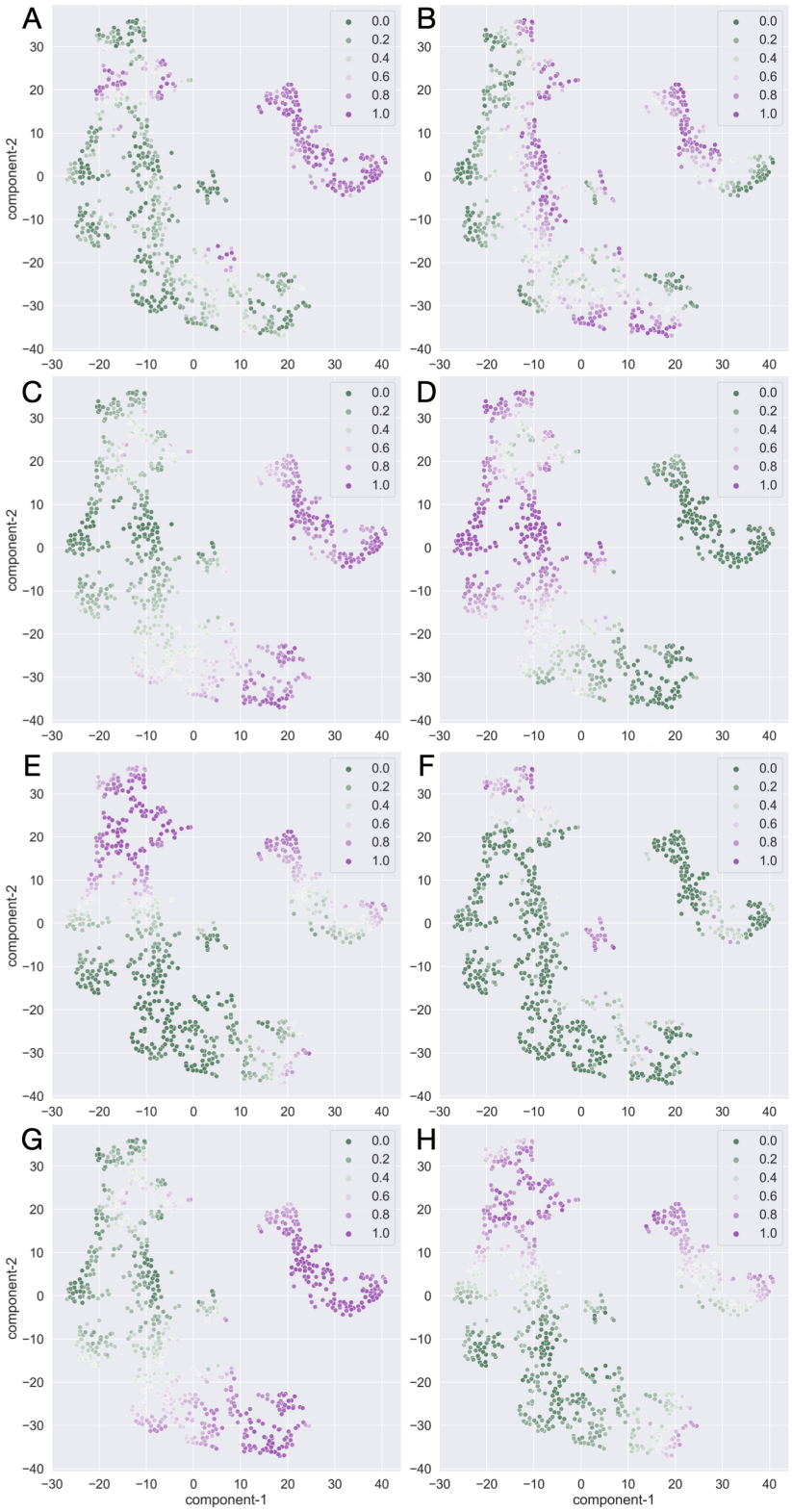

Figure S15. 2-D manifold of the feature space spanned by nests A1 and A8 colored by the features: A) $\Delta t_{AscendingNode}$, B) $\Delta t_{FullMoon}$, C) $\Delta t_{Perigee}$, D) $\cos(\gamma)$, E) $\sin(\gamma)$, F) $e_{i+1} - e_i$, G) d, H) $\dot{d}$.

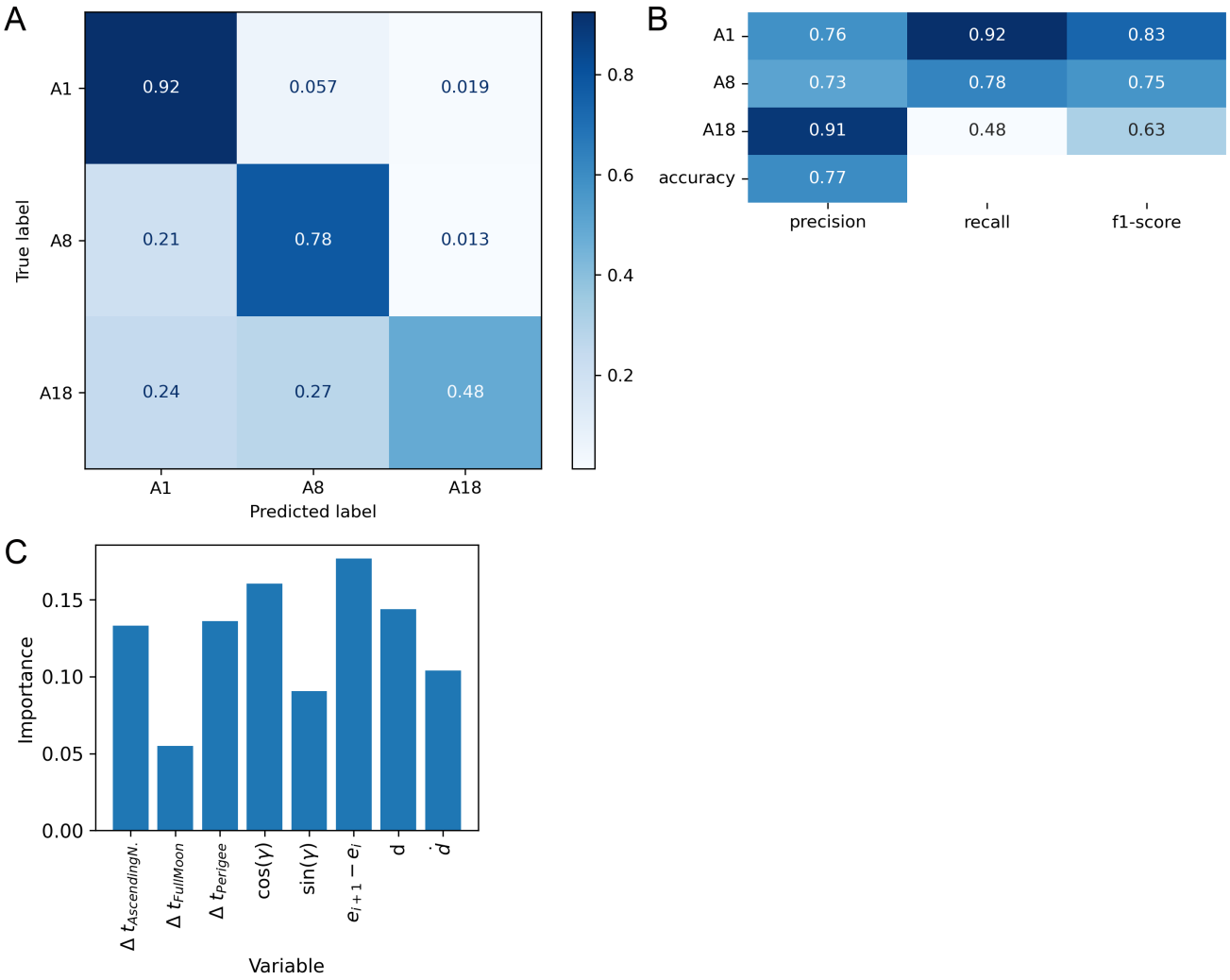

Figure S16. Same as Figure S6, but classifying three nests A1, A8 and A18.

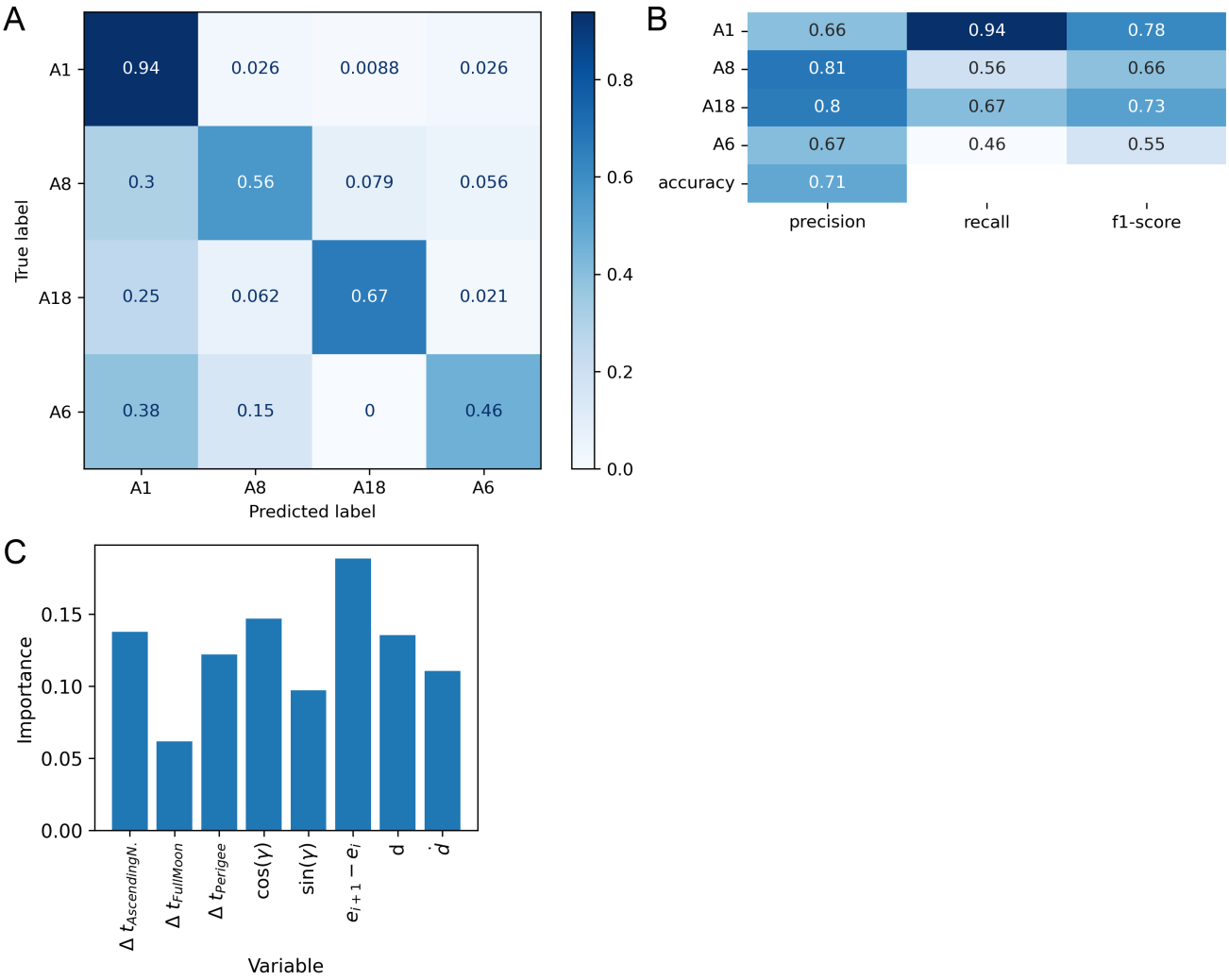

Figure S17. Same as Figure S6, but classifying four nests A1, A8, A18, A6.

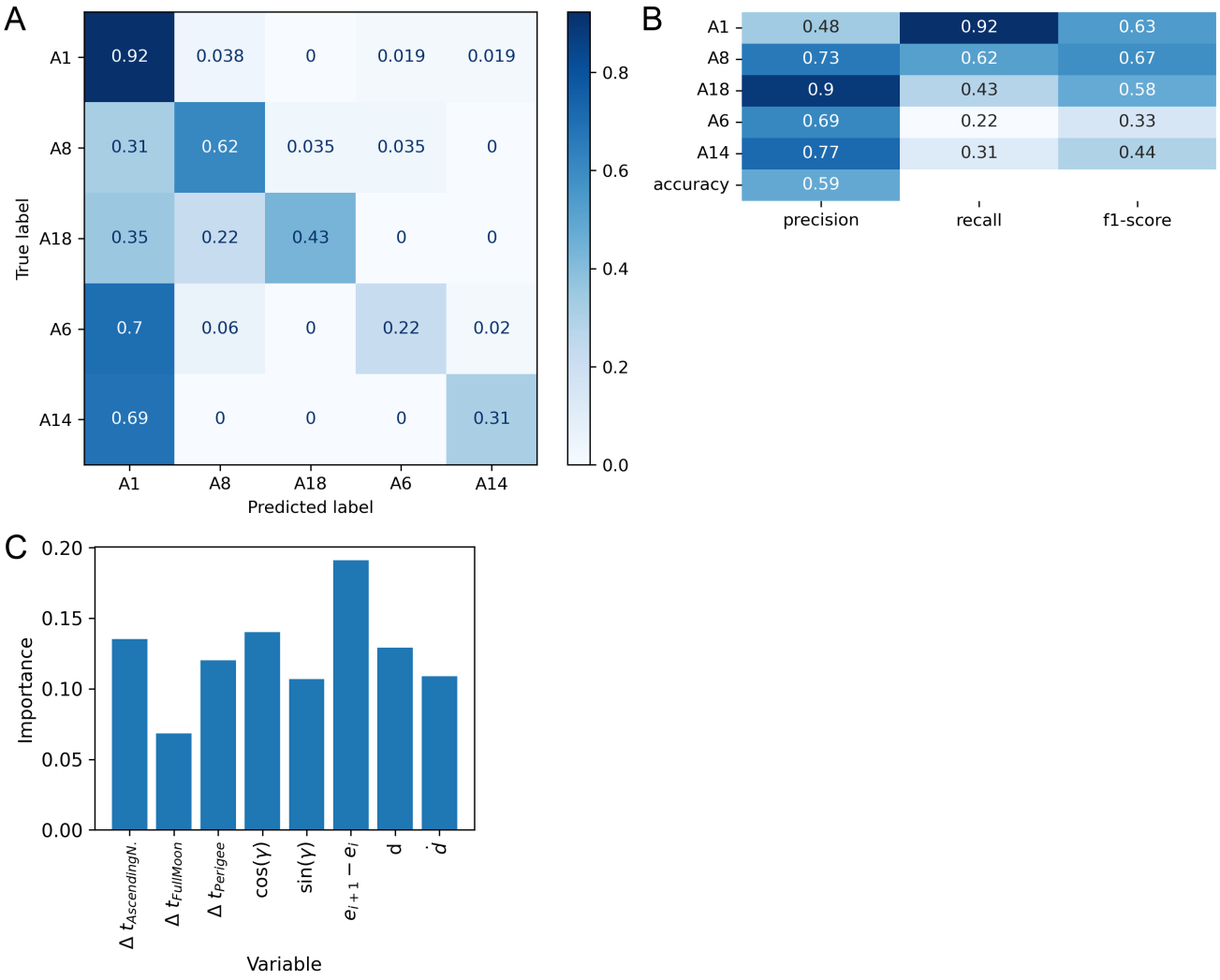

Figure S18. Same as Figure S6, but classifying five nests A1, A8, A18, A6, A14.

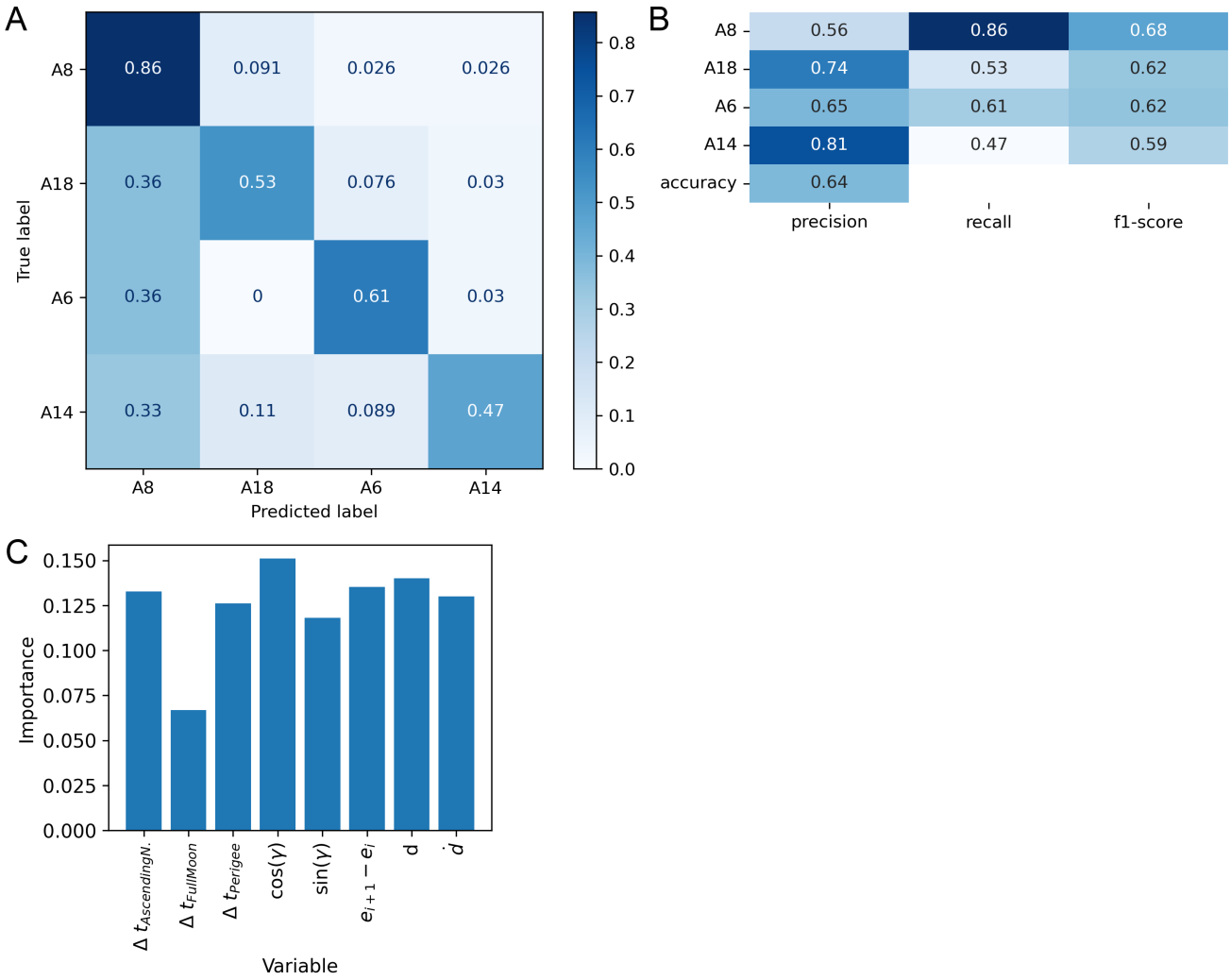

Figure S19. Same as Figure S6, but classifying four nests A8, A18, A6, A14.

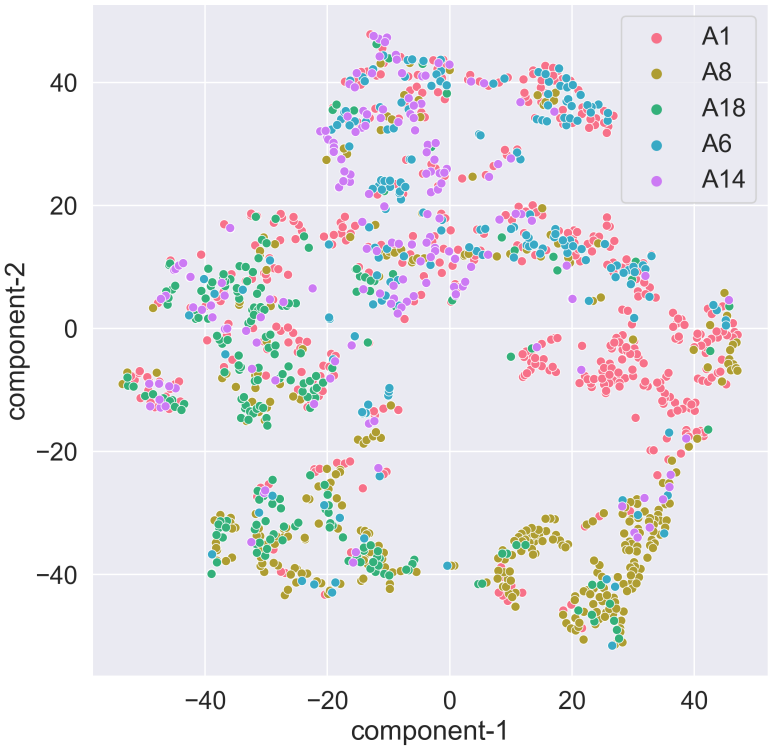

Figure S20. 2-D manifold of the feature space spanned by nests A1, A8, A18, A6, A14.

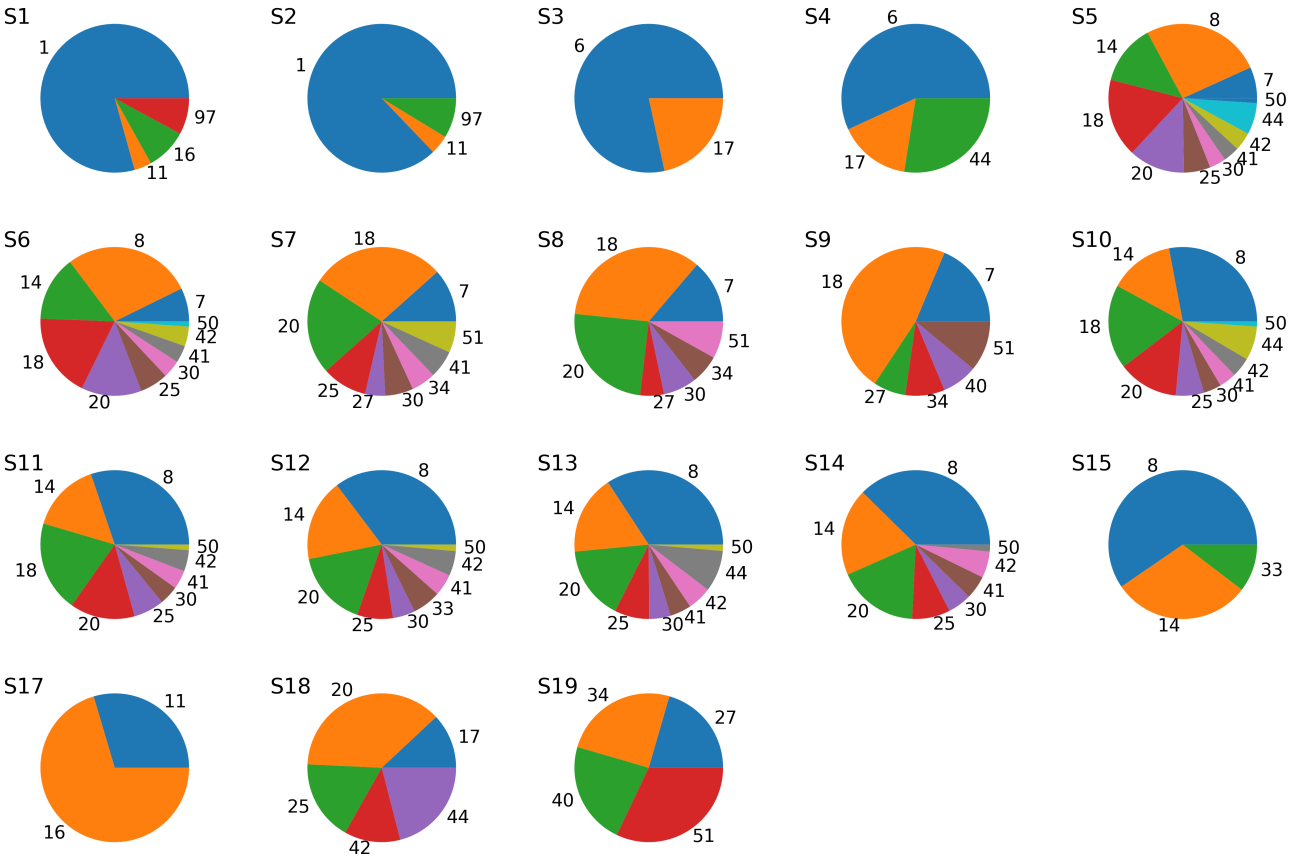

Figure S21. Pie charts for all sets shown in Figure 1 displaying the composition of the set and the contribution of each nest within the each set.

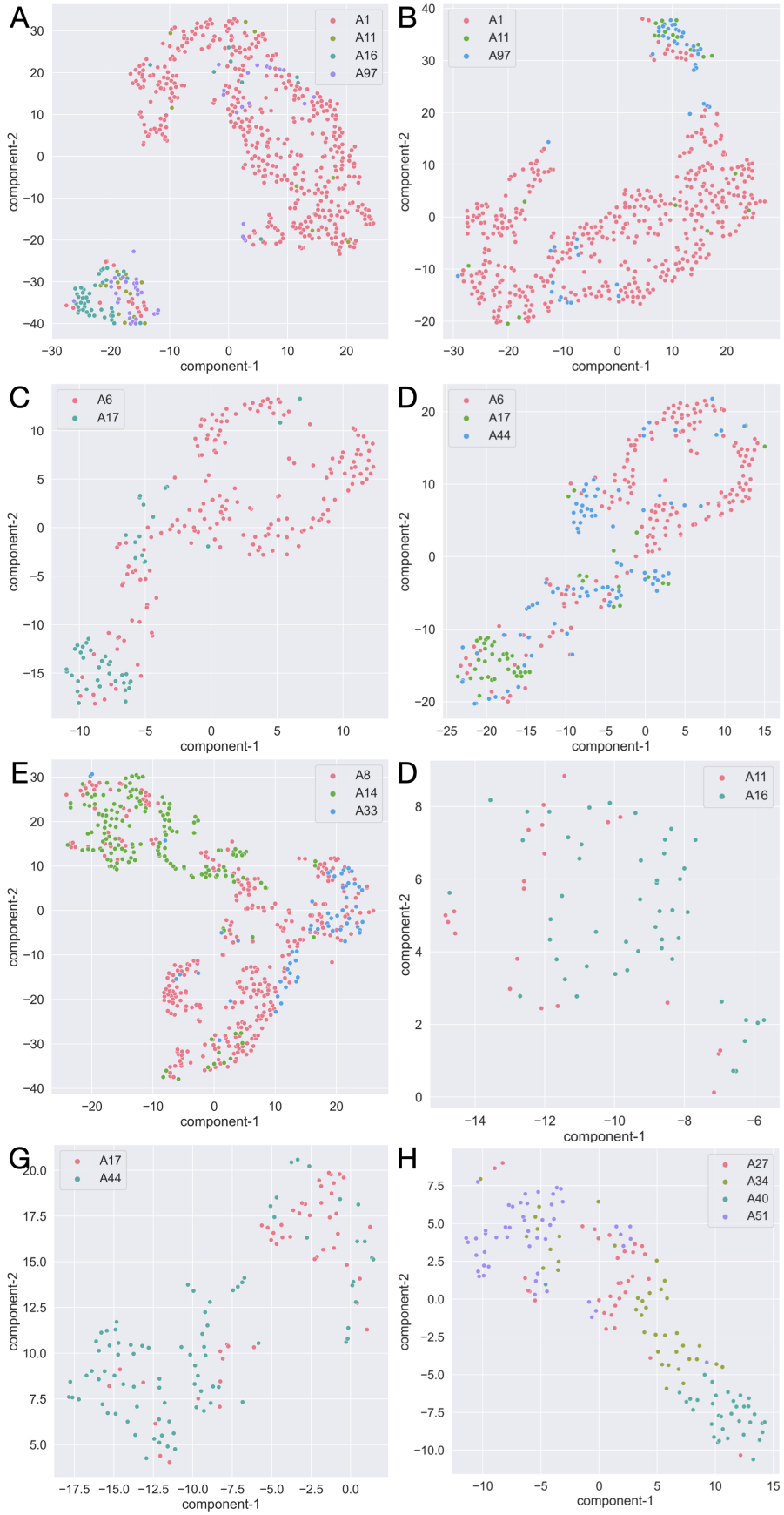

Figure S22. 2-D manifold of the feature space spanned by nests belonging to defined sets S_i: A) S1, B) S2, C) S3, D) S4, E) S12, F) S13, G) S14, H) S15.

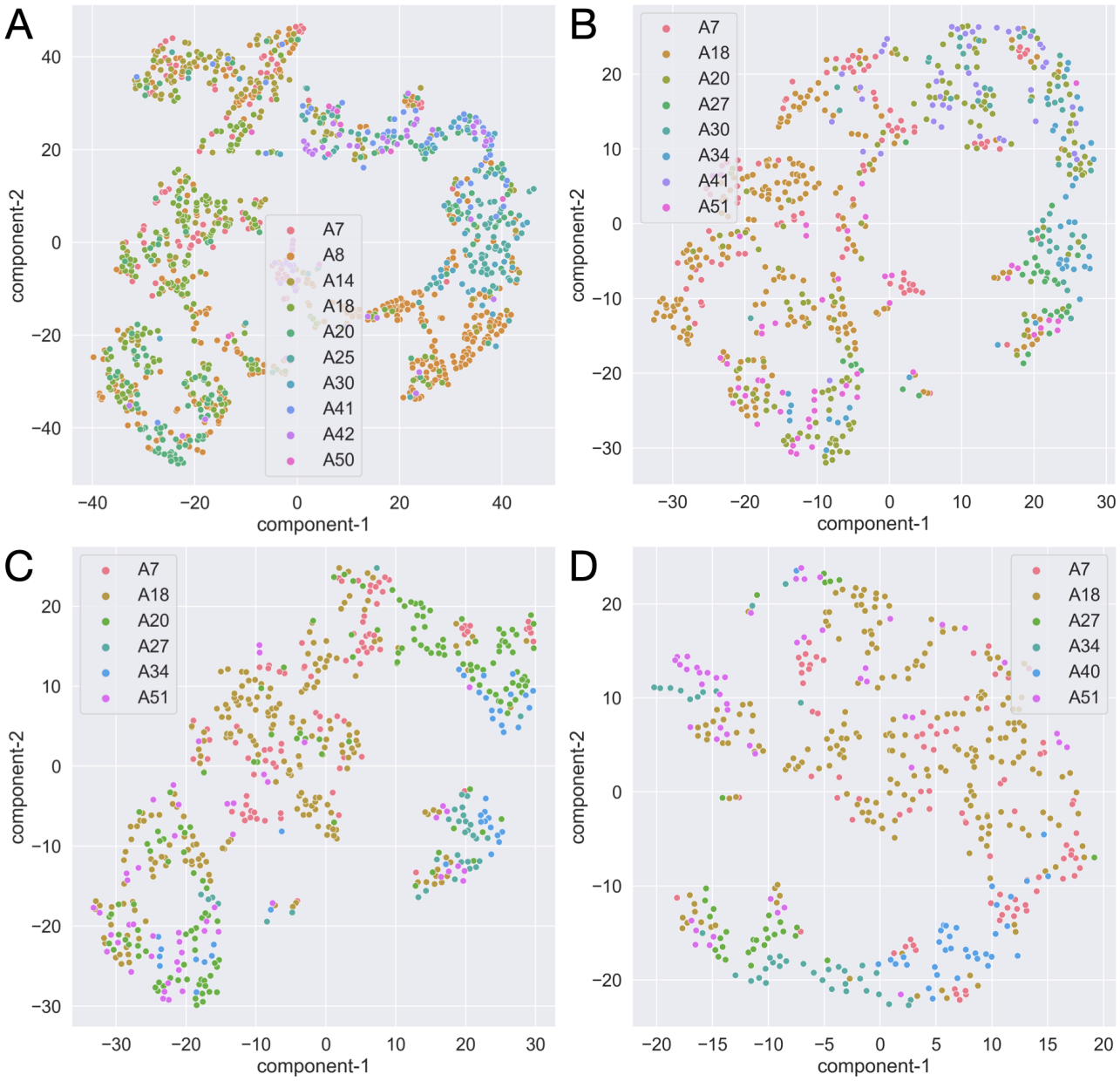

Figure S23. 2-D manifold of the feature space spanned by nests belonging to defined sets S_i: A) S5, B) S6, C) S7, D) S8

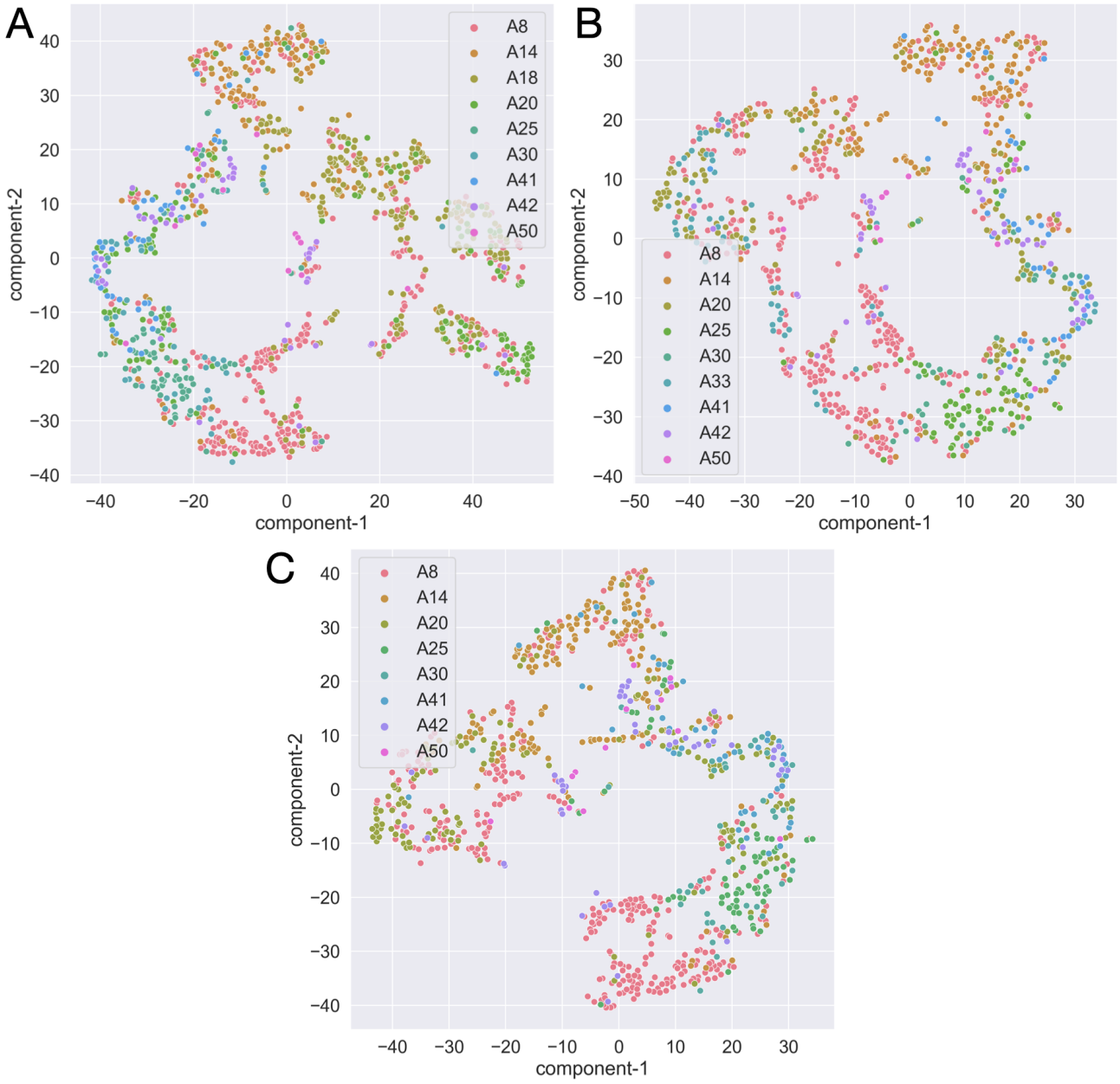

Figure S24. 2-D manifold of the feature space spanned by nests belonging to defined sets S_i: A) S9, B) S10, C) S11.