CS 4641 | Machine Learning
Final Project

Introduction to the dataset

Dataset: Crowdsourced Mapping Dataset

This dataset contains geospatial imagery of land masses, taken between year 2014 and 2015. The labels for each land mass were crowdsourced and just like the imagery, may contain noise. Each sample is described by normalized difference vegetation index (NDVI) across a time series. The time series spans across 27 instances in time, so each sample is described with a vector of 27 floats. NDVI is a measure of the quantity of live green vegetation in an area. The possible labels with proportional frequencies are impervious (0.0919), farm (0.1367), forest (0.7047), grass (0.0423), orchard (0.005), water (0.0194).

The dataset was already split into a training set of 10546 samples and testing set of 300 samples. Because of the lengthy time to tune the hyperparameters for Deep Neural Nets on all 10546 training samples, I created a subset of samples, by first randomly shuffling the 10546 and taking the first 1000. Since, I intend to do a 10-fold cross validation, and the random sampling rarely selected more than 10 orchard labeled samples, I inserted an additional 17 orchard samples to the 1000. The specific samples can be found in extra_orchards.csv. The results from this report will be based on experiments done using the 1017 training set and 300 testing set.

This is a multi-label classification problem. Looking at a large confusion matrix may provide more information about the performance of each classifier. However, for the purposes of narrowing down the best hyperparameter setting and comparing different algorithms, this may present an overflow of information, and hinder the decision-making process. As such, a simple accuracy score will be the sole performance metric for this exercise.

Description of the algorithms

I will be using implementations from Sci-kit Learn. For the neural network, I did attempt to apply code from HW2 logistic regression and neural network code from another class. Both proved to be too time consuming when training on large datasets.

Decision Trees with Bagging via RandomForestClassifier from Sci-kit Learn
Decision trees attempt to arrive at a classification through asking a series of questions. Questions are represented as internal nodes, responses to questions as edges between nodes, and final predicted label encoded in the leaves. Thus, each response to a question either leads to another question or a classification.

The proper question to ask for each node is determined by measuring the information gain by asking that question. The complexity of a tree grows when more questions are asked, i.e. when the tree is deeper. Naturally, overfitting can occur. One solution is Bagging, which is to train many simple trees, or a forest, and take the most popular prediction across all trees.

Note: I will refer to Decision Trees with Bagging as Random Forests from now on.

Hyperparameters to be Tuned:
n_estimators, [100, 500]: Model is more complex with more trees.
max_depth [100, None]: Model is more complex with larger limit.
class_weight [inverse normalized frequency, None]: Model is more complex with weighting.

Non-linear SVMs via SVC from Sci-kit Learn
Support Vector Machines attempt to find the best decision boundary which can separate points of different labels. Best is defined as the boundary with the largest margin. Hypotheses for SVM project the feature transformation of each data point onto a vector perpendicular to the margin. Finding the hypothesis that maximizes the margin is equivalent to finding the optimal Lagrange multipliers, denoted by alpha, for each point in the dataset. These alphas

help define the boundary. Most points have alpha equal to 0. Only the points that lie on the edge of the margin have positive alphas, which contribute to decision boundary.

Note: When referring to linear SVMs, I will explicitly state “linear SVM”. Nonlinear SVMs, will be referenced as either “nonlinear SVM” or just “SVM” .

Hyperparameters to be Tuned:
kernel, [rbf, sigmoid, linear]: Model is more complex with nonlinear kernels.
gamma, [scale, auto]: Only applicable for nonlinear kernels, so model is more complex with gamma values.
C, regularization parameter: [1, 100, 1000]: There is an inversely proportional relationship between C and regularization strength. So, model is more complex with larger C values.
class_weight [inverse normalized frequency, None]: Model is more complex with weighting.

Neural Network via MultiLayeredPerceptron from Sci-kit Learn
A single perceptron has inputs, weights for inputs, and an output. The output is computed by taking the weighted sum of the inputs, using its weights, and then applying a non-linear activation function on the weighted sum.

Neural Networks are networks of perceptrons. Multiple perceptrons stack together to form a layer and outputs from perceptrons of a layer feed into perceptrons of another layer. The outputs of the final layer become the output for the entire network. These networks can estimate functions by adjusting the weights in each perceptron and various methods exist for updating those weights.

Note: I will refer to Neural Networks as DNN from now on.

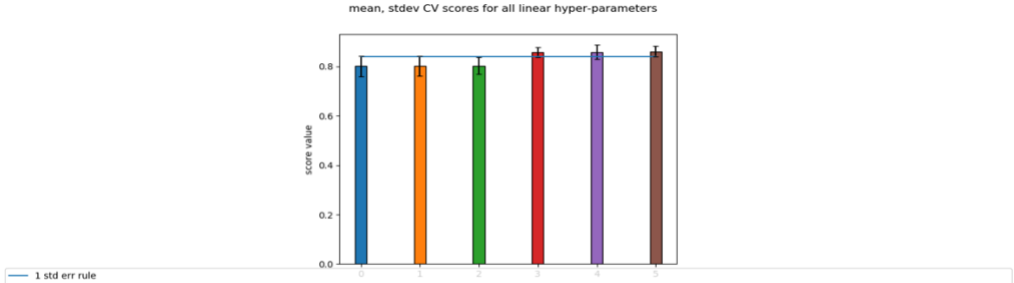
Hyperparameters to be Tuned:
hidden_layer_sizes, [(32, 24), (32, 24, 16)]: Model is more complex with more layers.
activation, [logistic, relu]: Model is more complex with nonlinear activation functions.
learning_rate_init, [0.001, 0.0001, 0.00001]: Model weights are more likely to avoid diverging with smaller learning rates but must also train from more iterations.
learning_rate, ["constant", "invscaling", "adaptive"]: Learning rate schedules help mitigate problems with slow weight updates and divergence of weight values.

Note: I initially tried to test layer sizes of (100, 100), (100, 200, 100), but the grid search was taking more than two hours. After lowering the sizes, the grid search takes just under two hours.

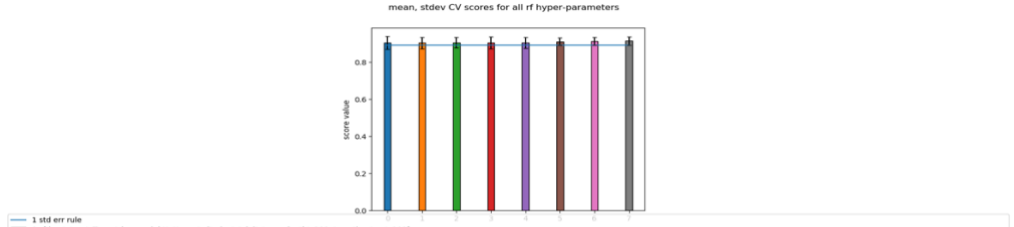
Tuning hyperparameters

A grid search with a 10-fold cross validation of the training data will be used for tuning hyperparameters. The train set is divided into 10 folds. All combinations of the hyperparameter settings mentioned in the previous section will do a single 10-fold cross validation trial. The mean and standard deviation of the 10 accuracies scored on the cross-validation folds are recorded. This is needed for selection of the hyperparameters to use for each classifier.

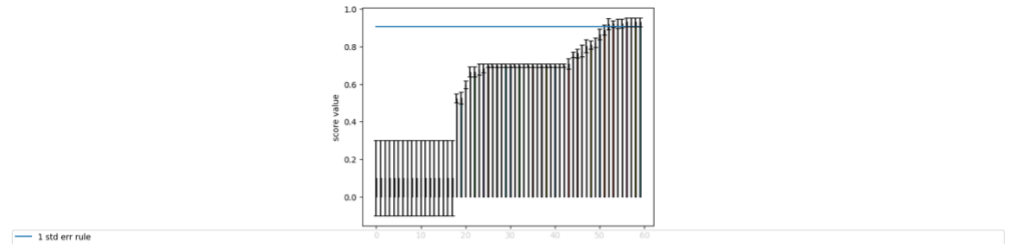
Hyperparameter selection is based on the 1 standard error rule. The hyperparameter selected is the simplest one within 1 standard deviation of the best hyperparameter. In the bar plots below, each bar indicates the mean for a single hyperparameter and the whiskers are the corresponding standard deviation. There is a blue horizontal line, which represents the bounds for 1 standard error limit within the best mean. The hyperparameters I selected follow this philosophy and are outlined in the table below the bar plots.



1 std err rule
0: {'C': 1, 'class_weight': {'farm': 4.677631578947368, 'forest': 1.0, 'orchard': 37.421052631578945, 'grass': 16.53488372093023, 'impervious': 8.8875, 'water': 54.69230769230769}, 'kernel': 'linear'}
1: {'C': 100, 'class_weight': {'farm': 4.677631578947368, 'forest': 1.0, 'orchard': 37.421052631578945, 'grass': 16.53488372093023, 'impervious': 8.8875, 'water': 54.69230769230769}, 'kernel': 'linear'}
2: {'C': 1000, 'class_weight': {'farm': 4.677631578947368, 'forest': 1.0, 'orchard': 37.421052631578945, 'grass': 16.53488372093023, 'impervious': 8.8875, 'water': 54.69230769230769}, 'kernel': 'linear'}
3: {'C': 1000, 'class_weight': None, 'kernel': 'linear'}
4: {'C': 1, 'class_weight': None, 'kernel': 'linear'}
5: {'C': 100, 'class_weight': None, 'kernel': 'linear'}

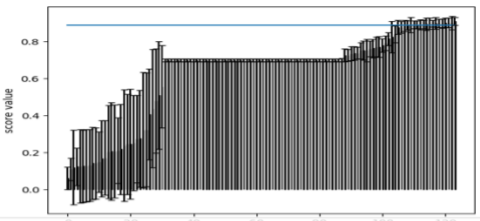


1 std err rule
0: {'bootstrap': True, 'class_weight': None, 'criterion': 'gini', 'max_depth': 100, 'n_estimators': 100}
1: {'bootstrap': True, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'n_estimators': 500}
2: {'bootstrap': True, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'grass': 16.53488372093023, 'orchard': 37.421052631578945, 'water': 54.69230769230769}, 'criterion': 'gini', 'max_depth': None, 'n_estimators': 100}
3: {'bootstrap': True, 'class_weight': None, 'criterion': 'gini', 'max_depth': 100, 'n_estimators': 500}
4: {'bootstrap': True, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'n_estimators': 100}
5: {'bootstrap': True, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'grass': 16.53488372093023, 'orchard': 37.421052631578945, 'water': 54.69230769230769}, 'criterion': 'gini', 'max_depth': 100, 'n_estimators': 100}
6: {'bootstrap': True, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'grass': 16.53488372093023, 'orchard': 37.421052631578945, 'water': 54.69230769230769}, 'criterion': 'gini', 'max_depth': None, 'n_estimators': 500}
7: {'bootstrap': True, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'grass': 16.53488372093023, 'orchard': 37.421052631578945, 'water': 54.69230769230769}, 'criterion': 'gini', 'max_depth': 100, 'n_estimators': 500}



1 std err rule
0: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
1: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
2: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
3: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
4: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
5: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
6: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
7: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
8: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
9: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
10: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
11: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
12: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
13: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
14: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
15: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
16: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
17: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
18: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'sigmoid'}
19: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'sigmoid'}
20: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'sigmoid'}
21: {'C': 100, 'class_weight': None, 'gamma': 'scale', 'kernel': 'sigmoid'}
22: {'C': 1000, 'class_weight': None, 'gamma': 'scale', 'kernel': 'sigmoid'}
23: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'sigmoid'}
24: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'sigmoid'}
25: {'C': 1, 'class_weight': None, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
26: {'C': 1, 'class_weight': None, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
27: {'C': 1, 'class_weight': None, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
28: {'C': 1, 'class_weight': None, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
29: {'C': 1, 'class_weight': None, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
30: {'C': 1, 'class_weight': None, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
31: {'C': 100, 'class_weight': None, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
32: {'C': 100, 'class_weight': None, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
33: {'C': 100, 'class_weight': None, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
34: {'C': 100, 'class_weight': None, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
35: {'C': 100, 'class_weight': None, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
36: {'C': 100, 'class_weight': None, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
37: {'C': 1000, 'class_weight': None, 'degree': 3, 'gamma': 0.001, 'kernel': 'poly'}
38: {'C': 1000, 'class_weight': None, 'degree': 3, 'gamma': 0.0001, 'kernel': 'poly'}
39: {'C': 1000, 'class_weight': None, 'degree': 4, 'gamma': 0.001, 'kernel': 'poly'}
40: {'C': 1000, 'class_weight': None, 'degree': 4, 'gamma': 0.0001, 'kernel': 'poly'}
41: {'C': 1000, 'class_weight': None, 'degree': 5, 'gamma': 0.001, 'kernel': 'poly'}
42: {'C': 1000, 'class_weight': None, 'degree': 5, 'gamma': 0.0001, 'kernel': 'poly'}
43: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'sigmoid'}
44: {'C': 1, 'class_weight': None, 'gamma': 'auto', 'kernel': 'sigmoid'}
45: {'C': 1, 'class_weight': None, 'gamma': 'scale', 'kernel': 'sigmoid'}
46: {'C': 1000, 'class_weight': None, 'gamma': 'auto', 'kernel': 'sigmoid'}
47: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'rbf'}
48: {'C': 100, 'class_weight': None, 'gamma': 'auto', 'kernel': 'sigmoid'}
49: {'C': 1, 'class_weight': None, 'gamma': 'auto', 'kernel': 'rbf'}
50: {'C': 1, 'class_weight': None, 'gamma': 'scale', 'kernel': 'rbf'}
51: {'C': 1, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'rbf'}
52: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'rbf'}
53: {'C': 100, 'class_weight': None, 'gamma': 'auto', 'kernel': 'rbf'}
54: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'auto', 'kernel': 'rbf'}
55: {'C': 1000, 'class_weight': None, 'gamma': 'auto', 'kernel': 'rbf'}
56: {'C': 100, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'rbf'}
57: {'C': 100, 'class_weight': None, 'gamma': 'scale', 'kernel': 'rbf'}
58: {'C': 1000, 'class_weight': {'impervious': 8.8875, 'forest': 1.0, 'farm': 4.677631578947368, 'water': 54.69230769230769, 'grass': 16.53488372093023, 'orchard': 37.421052631578945}, 'gamma': 'scale', 'kernel': 'rbf'}
59: {'C': 1000, 'class_weight': None, 'gamma': 'scale', 'kernel': 'rbf'}

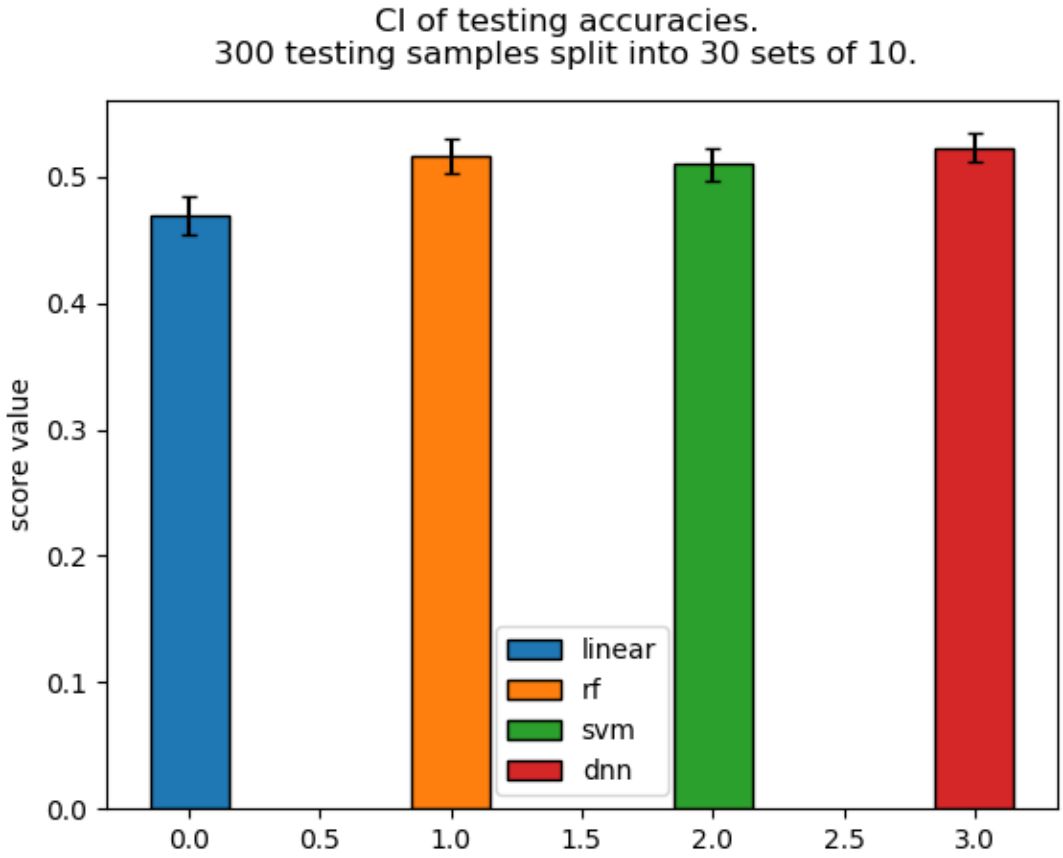
mean, stdev CV scores for all dnn hyper-parameters



1 std err rule
0: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
1: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
2: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
3: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
4: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
5: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
6: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
7: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
8: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
9: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
10: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
11: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
12: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
13: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
14: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
15: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
16: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
17: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
18: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
19: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
20: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
21: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
22: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
23: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
24: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
25: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
26: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
27: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
28: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
29: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
30: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
31: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
32: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
33: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
34: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
35: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
36: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
37: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
38: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
39: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
40: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
41: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
42: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
43: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
44: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
45: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
46: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
47: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
48: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
49: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
50: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
51: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
52: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
53: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
54: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
55: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
56: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
57: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
58: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
59: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
60: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
61: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
62: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
63: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
64: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
65: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
66: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
67: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
68: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
69: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
70: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
71: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
72: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
73: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
74: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
75: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
76: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
77: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
78: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
79: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
80: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'invscaling', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
81: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
82: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
83: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 0.0001, 'max_iter': 10000, 'solver': 'adam'}
84: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 1e-05, 'max_iter': 10000, 'solver': 'adam'}
85: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 0.0001, 'max_iter': 10000, 'solver': 'adam'}
86: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 1e-05, 'max_iter': 10000, 'solver': 'adam'}
87: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 1e-05, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
88: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
89: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
90: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
91: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
92: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
93: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
94: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
95: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
96: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
97: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
98: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
99: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.0001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
100: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 1e-05, 'max_iter': 10000, 'solver': 'adam'}
101: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 1e-05, 'max_iter': 10000, 'solver': 'adam'}
102: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'max_iter': 10000, 'solver': 'lbfgs'}
103: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 0.001, 'max_iter': 10000, 'solver': 'adam'}
104: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'max_iter': 10000, 'solver': 'lbfgs'}
105: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'max_iter': 10000, 'solver': 'lbfgs'}
106: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'max_iter': 10000, 'solver': 'lbfgs'}
107: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
108: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
109: {'activation': 'logistic', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 0.001, 'max_iter': 10000, 'solver': 'adam'}
110: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
111: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
112: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
113: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
114: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 0.001, 'max_iter': 10000, 'solver': 'adam'}
115: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.5, 'solver': 'sgd'}
116: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
117: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
118: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.7, 'solver': 'sgd'}
119: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
120: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_iter': 10000, 'power_t': 0.3, 'solver': 'sgd'}
121: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 0.0001, 'max_iter': 10000, 'solver': 'adam'}
122: {'activation': 'relu', 'hidden_layer_sizes': (32, 24, 16), 'learning_rate_init': 0.0001, 'max_iter': 10000, 'solver': 'adam'}
123: {'activation': 'relu', 'hidden_layer_sizes': (32, 24), 'learning_rate_init': 0.001, 'max_iter': 10000, 'solver': 'adam'}

Selected Hyperparameters	Linear SVM Index = 4	RandomForest Index = 0	SVM Index = 54	DNN Index = 114
Hyperparameters	C: 1 class_weight: None kernel: linear	class_weight: None max_depth: 100 n_estimators: 100	C: 1000 class_weight: inverse normalized frequency gamma: auto kernel: rbf	activation: relu hidden_layer_size: (32, 24, 16) learning_rate_init: 0.001 solver: adam
Mean, Stdev. of CV Accuracy Score	0.858 (+/-0.057)	0.904 (+/-0.069)	0.930 (+/-0.046)	0.894 (+/-0.064)

Justify the claim that the data has a non-trivial distribution and Comparing algorithm performance.



The classifiers are once trained on the entire training set, no cross validation. The testing set is divided into 30 disjoint subsets of size 10, and the trained classifiers are evaluated on the accuracy performance score on each of the 30 sets. The plot above shows the mean across the 30 evaluations and the standard deviations.

Notice that the CI for linear SVM is lower than the others, nor does it intersect the others. This suggests, there is a significant difference, between linear SVM performance and nonlinear classifier performance, and justifies the non-trivial distribution of the data.

Time to run grid search with 10-fold cross validations from training data:

D:HH:MM:SS	Linear SVM	Random Forest	SVM	DNN
Subtotals	0:00:00:47.05	0:00:1:15.3	0:00:00:40.18	0:1:24:25.23
Total Time	**0:01:22:07.76 = 1 hour, 22 minutes, 07.76 seconds.**			

As stated before, Random Forest, nonlinear SVM, and DNN significantly outperform the linear SVM. Between the nonlinear classifiers, a significant difference cannot be observed, as the CI intersect.

Since performance is similar, the next deciding factor is simplicity of the model. Looking at just the selected hyperparameters, it looks like Random Forest has the least complex configuration, thus should be selected as the classifier to use on this and similar data.

Conclusion

Taking training and tuning time into consideration, Random Forest would still be the classifier I would use for similar types of data. Its simplicity gives it greater merit over nonlinear SVM and its time consumption is much better than DNN.

Acknowledgements
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV.fit
https://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforest#sklearn.ensemble.RandomForestClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html?highlight=mlpclassifier#sklearn.neural_network.MLPClassifier
https://python-graph-gallery.com/8-add-confidence-interval-on-barplot/
https://www.journaldev.com/15638/python-pickle-example#python-pickle-dump
https://stackoverflow.com/questions/6081008/dump-a-numpy-array-into-a-csv-file
https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope