IML | EX.02 | OHAD DANIEL | 209006063

Classification with k-Nearest Neighbors (kNN)

Description of the experiment

In this part of the exercise, we investigated the use of a distance-based classifier, k-Nearest Neighbors (kNN), for predicting the U.S. state of each city based solely on its geographical coordinates (longitude and latitude). The dataset consists of roughly 30,000 samples, each representing the location of a city along with its corresponding state label. We split the data according to the assignment instructions: the train.csv file was used exclusively for training, while test.csv was used for the final evaluation.

To perform the classification efficiently, we implemented a custom KNN Classifier using the FAISS library. The classifier is composed of three main components. The fit function stores the training data and builds an FAISS index using either the L1 or L2 distance metric. The knn distance function queries this index to retrieve, for each test sample, its k nearest neighbors along with their distances. Finally, the predict function uses these neighbors to perform a majority vote over their labels, returning a predicted state for every test sample. This structure follows the standard supervised-learning workflow: training on labeled data, querying the model for nearest neighbors, and predicting based on those neighbors.

We examined all combinations of $k \in \{1, 10, 100, 1000, 3000\}$ and distance metric $\in \{\text{L1}, \text{L2}\}$. For each of the 10 resulting configurations, we trained a separate kNN model, ran it on the test set, computed its accuracy, and stored both the model and its hyperparameters for later comparison. This setup allowed us to systematically study how the choice of k and distance metric affects the model's performance.

Results and discussion

The results of the experiment are summarized in the following table, which reports the test-set accuracy for each combination of k and distance metric:

k	L1 Accuracy	L2 Accuracy
1	0.9670	0.9680
10	0.9617	0.9577
100	0.9231	0.9201
1000	0.7450	0.7417
3000	0.4018	0.3981

These results reveal several important trends. First, small values of k clearly perform best. When k = 1, the classifier achieves accuracy of approximately 0.968, and for k = 10 accuracy remains very high (around 0.96). This behavior is natural in this dataset: cities located near one another tend to belong to the same state, so the nearest neighbor(s) are strong indicators of the correct class. The large size of the dataset also means that each city has many nearby training points from the same state, making local information highly informative.

As k increases, however, the accuracy drops significantly. For k = 1000 and especially k = 3000, the classifier includes very distant cities from many states when computing the majority vote. Because these distant neighbors are not geographically related to the query point, the prediction becomes essentially an average over unrelated regions, leading to substantial performance degradation (as low as 0.40). This illustrates the classic tradeoff in kNN: very small k may overfit noise, but excessively large k destroys locality and weakens the classifier.

When comparing the two distance metrics, we observe that L2 distance consistently performs slightly better than L1, especially for small k. This outcome is intuitive: Euclidean distance (L2) corresponds more closely to real geographic distance on a map, while L1 measures movement along axis-aligned paths, which is less meaningful for spatial coordinates. Nevertheless, the difference between L1 and L2 is relatively small across all configurations.

Overall, the best-performing model is obtained with k = 1 and L2 distance, achieving a test accuracy of 0.968. This model will be used in subsequent parts of the assignment as required

---

Visualization of kNN Decision Boundaries

Figure 2 – L2 distance, k = 1 (kmax):
The decision boundaries are highly detailed and closely follow the true geographic structure, as each point is classified by its single nearest neighbor.

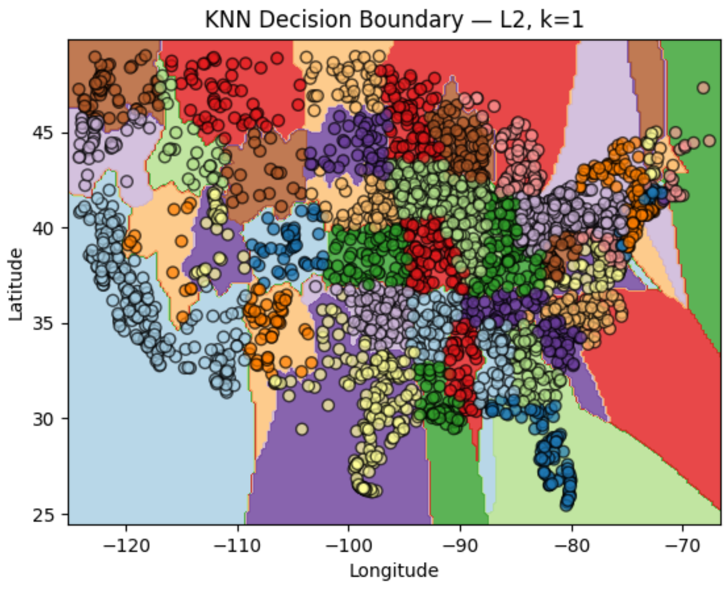


Figure 2 – L1 distance, k = 1 (kmax):
These boundaries create vertical and horizontal strips that do not follow the true curved geographic shapes of the states as well as with the distance.

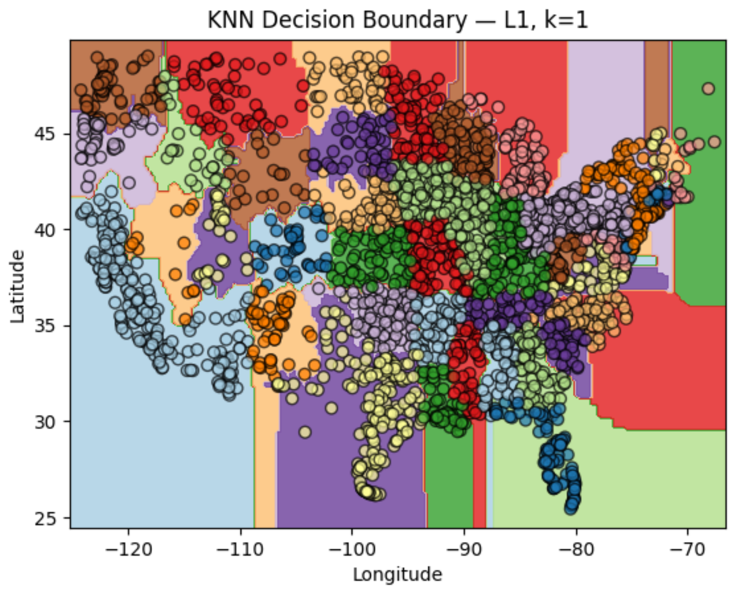
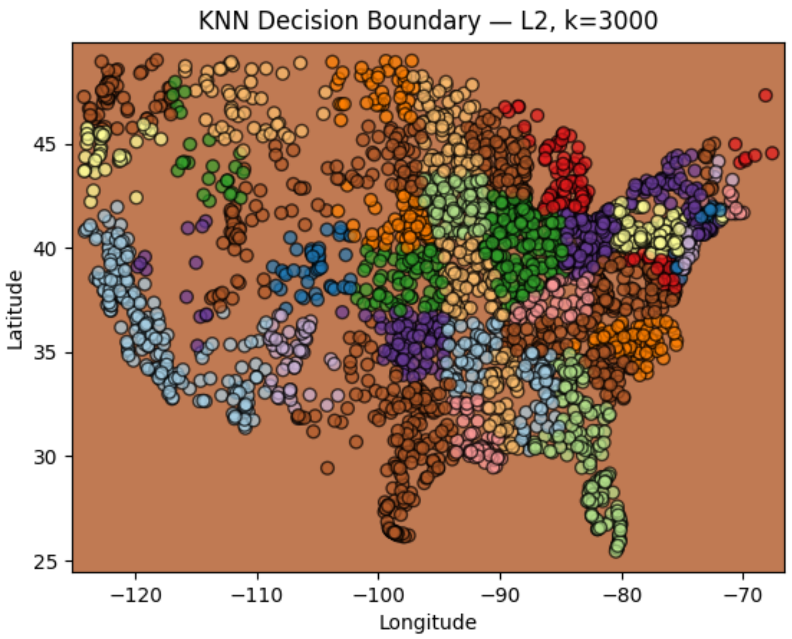


Figure 1 – L2 distance, k = 3000 (kmin):
The decision regions are very detailed but appear as blocky, axis aligned shapes due to the l1 distance. This creates vertical and horizontal strips that do not follow the true curved geographic shapes of the states as closely as in the l2 case.



a. Compare L2 with kmax (k=1) vs L2 with kmin (k=3000). What is different and why does kmax give better accuracy?

When using L2 distance with k = 1, the kNN classifier produces very fine-grained and highly localized decision boundaries. Each location is assigned the label of its single nearest training city, leading to detailed partitions that closely reflect the actual geographic structure of the U.S. states. This local behavior is well-suited to the dataset, since cities that are geographically close almost always belong to the same state. As a result, k=1 preserves meaningful local distinctions and achieves the highest accuracy.

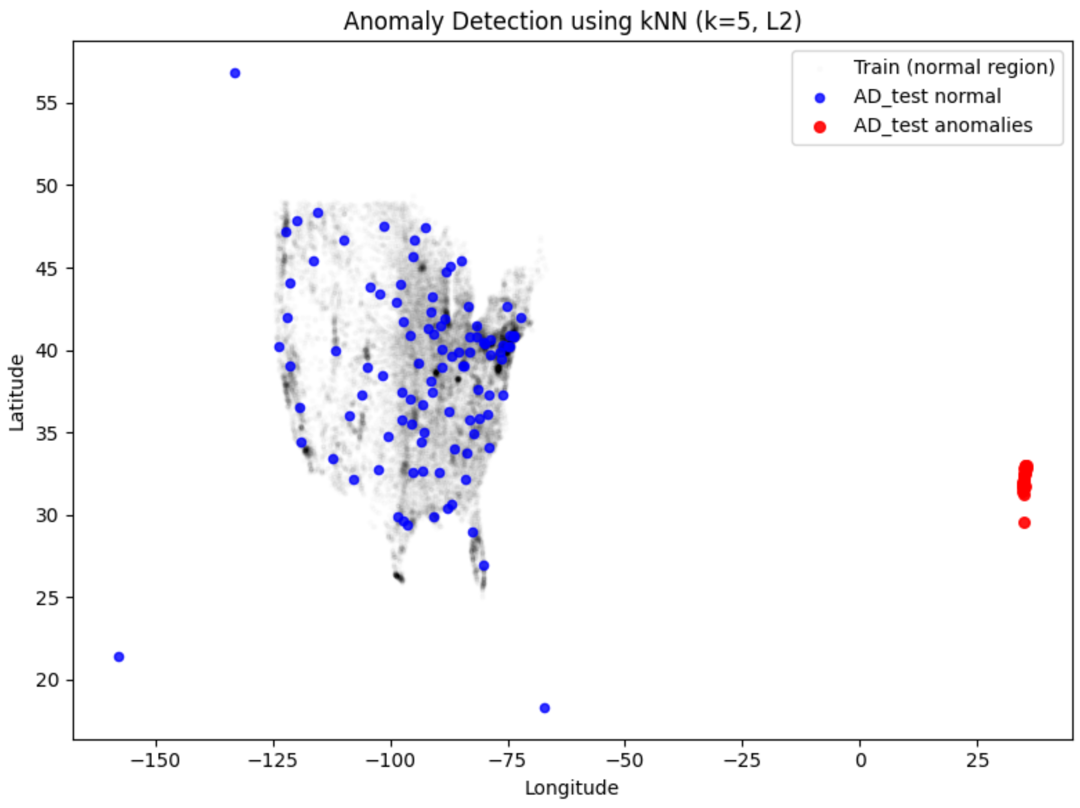
In contrast, when k = 3000, each prediction is influenced by thousands of neighbors spread across the entire map. This causes the decision boundaries to become extremely smooth and coarse: large contiguous regions receive identical labels, and local structure is almost completely lost. Because many of these distant neighbors belong to irrelevant or geographically unrelated states, the model tends to misclassify many test points. Therefore, k_min produces oversmoothed, inaccurate boundaries, while k_max better captures the underlying spatial organization and yields superior accuracy.

b. Compare L2, kmax (k=1) vs L1, kmax (k=1). How does the distance metric affect the classification space?

When k = 1, both L1 and L2 rely on the nearest neighbor, so the overall structure of the decision map is similar: the space is divided into many small, detailed regions that correspond to the city-level geometry of state borders. However, the shape of these regions differs because the distance metrics induce different geometries. L2 distance considers Euclidean circles as neighborhoods, resulting in more rounded and smooth boundaries. L1 distance, in contrast, uses Manhattan geometry, where distance level-sets form diamond-shaped regions aligned with the coordinate axes. This produces sharper, more rectangular, and more axis-aligned partitions. Although the difference in accuracy is small, L2 is slightly more appropriate for geographical coordinates, which explains its better overall performance.

Anomaly Detection Using kNN

The plot shows the training data (black) forming the geographic shape of the United States, while the AD_test samples appear either inside this dense region as normal points (blue) or far outside it in a tight cluster (red), representing the detected anomalies.



1. What can you tell about the anomalies your model found? How are they different from the normal data? Explain.

The anomalies detected by the kNN model consist of points whose locations are significantly farther from all training examples than the rest of the AD_test data. Since the training set represents real U.S. cities with realistic longitude–latitude coordinates, it forms a dense spatial region corresponding to the map of the United States. Normal AD_test points appear inside or near this region, meaning their five nearest neighbors in the training set are relatively close, resulting in low anomaly scores.

In contrast, the anomalous points form a tight cluster far to the east of the United States, in a location where no training points exist. Because these points are geographically isolated, even their five nearest neighbors in the training set are very far away, yielding much higher distance sums (anomaly scores). This clear spatial separation is what causes the model to classify them as anomalies.

In summary, the anomalies differ from the normal points primarily in their geographical isolation: they lie far outside the spatial distribution of U.S. cities represented in the training data. Their unusually large distances from all known locations make them stand out as outliers, which is exactly what the kNN-based anomaly detection method is designed to identify.

Decision Trees

Visualization of kNN Decision Boundaries

Max Depth	Max Leaf Nodes	Train Acc	Val Acc	Test Acc
1	50	0.113	0.114	0.114
1	100	0.113	0.114	0.114
1	1000	0.113	0.114	0.114
2	50	0.189	0.186	0.189
2	100	0.189	0.186	0.189
2	1000	0.189	0.186	0.189
4	50	0.363	0.357	0.357
4	100	0.363	0.357	0.357
4	1000	0.363	0.357	0.357
6	50	0.593	0.580	0.582
6	100	0.593	0.580	0.582
6	1000	0.593	0.580	0.582
10	50	0.848	0.836	0.831
10	100	0.927	0.918	0.911
10	1000	0.942	0.931	0.923
20	50	0.855	0.845	0.836
20	100	0.951	0.942	0.931
20	1000	1.000	0.980	0.977
50	50	0.855	0.845	0.836
50	100	0.951	0.942	0.931
50	1000	1.000	0.980	0.977
100	50	0.855	0.845	0.836
100	100	0.951	0.942	0.931
100	1000	1.000	0.980	0.977

Decision Tree Results

As the maximal depth increases, the decision trees become more expressive and are able to capture the geographic structure of the states more accurately, leading to higher validation and test accuracy. Small depths (1–4) underfit the data, resulting in very low accuracy, while very large depths with many leaf nodes tend to overfit the training set but still achieve strong generalization due to the clear spatial separation in the dataset. The best-performing models are those with high depth and many leaf nodes (e.g., max_depth ≥ 20, max_leaf_nodes = 1000), achieving test accuracy of approximately 0.98.

Questions

Q1. Choose the tree with the best validation accuracy. What is its test accuracy?

The tree with the highest validation accuracy uses
max_depth = 20 (or 50 or 100) and max_leaf_nodes = 1000,
achieving:
• Validation accuracy: 0.980
• Test accuracy: 0.977

All trees with these hyperparameters achieve identical results because the depth constraint is no longer active when max_leaf_nodes is large enough.

Q2. Does the selected tree generalize well? Is the validation set sufficient for choosing the model?

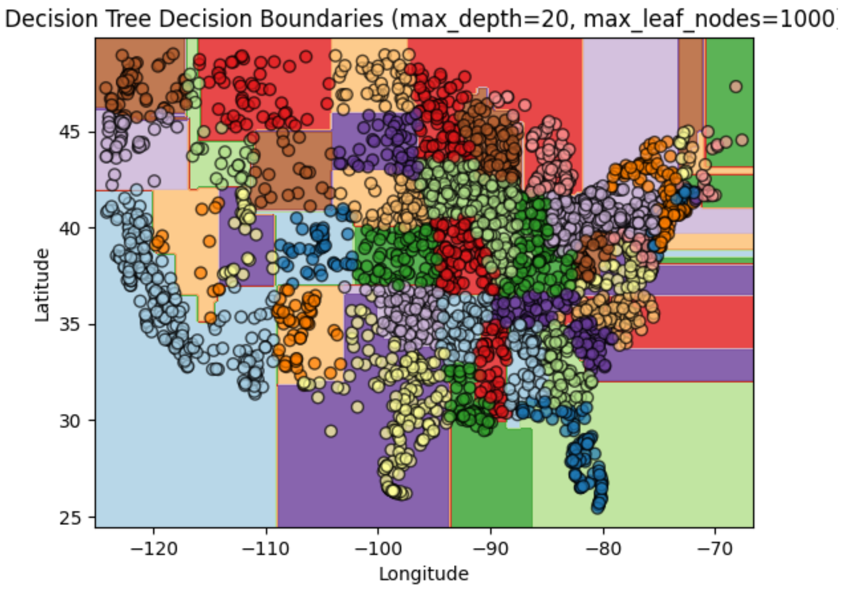
Yes.
The selected tree generalizes very well: although it achieves 100% accuracy on the training set, it still reaches 0.977 on the test set, which closely matches the validation accuracy (0.980).
Furthermore, trees with lower validation accuracy consistently show lower test accuracy.
This indicates that the validation set provides a reliable estimate of generalization and is sufficient for selecting the best model.

Q3. Are 50 leaf nodes enough for perfect accuracy?

No, 50 leaf nodes are not enough.
Even though there are 50 U.S. states, a leaf node predicts only a single state, and decision trees split the space using axis-aligned cuts. Because:
• state boundaries are highly irregular,
• many states consist of multiple disconnected or narrow regions,
• axis-aligned rectangles cannot match these shapes with a single leaf,

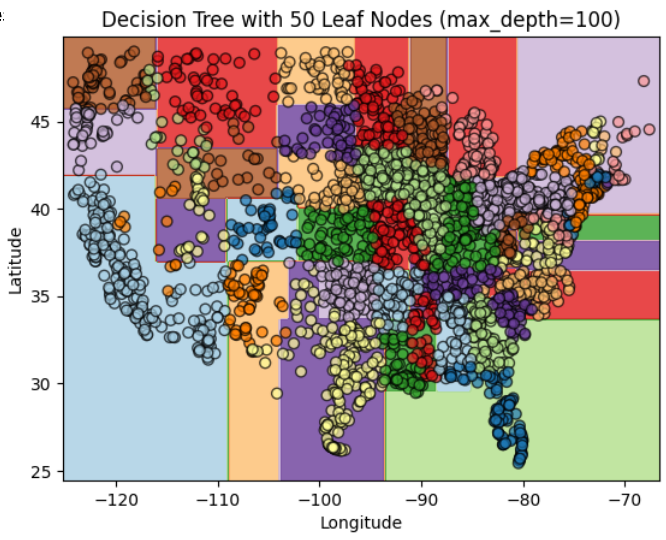
significantly more than 50 leaves are required to model the geographic structure accurately.
Thus, 50 leaf nodes are insufficient to achieve perfect accuracy.

Q4. How trees see the world — shape of the decision regions



The decision tree creates axis-aligned rectangular regions for each class.
Because tree splits are restricted to vertical and horizontal cuts in the longitude–latitude plane, each state is represented as a collection of rectangular blocks rather than smooth geographic shapes. This produces a blocky, grid-like structure in which complex state boundaries are approximated by multiple axis-aligned rectangles.
Interestingly, the resulting decision regions sometimes resemble the real map of the United States. This is partly coincidental: many U.S. states (especially in the Midwest and West) have boundaries that are rectangular and aligned with longitude and latitude lines. Since decision trees split the space using axis-aligned cuts, some states naturally match these rectangular partitions. However, states with irregular shapes are still broken into multiple rectangular blocks.

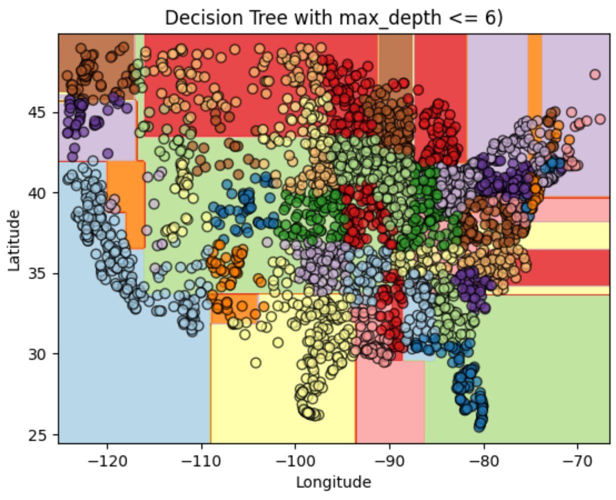
Q5: Restricted leave



Although the 50-leaf tree performs worse numerically (0.836 test accuracy versus 0.977 for the best model), the visual difference between the two decision maps is not as dramatic as one might expect. This is because many U.S. states have roughly rectangular, axis-aligned boundaries. Consequently, even a tree with only 50 leaves can capture much of the coarse geographic structure.

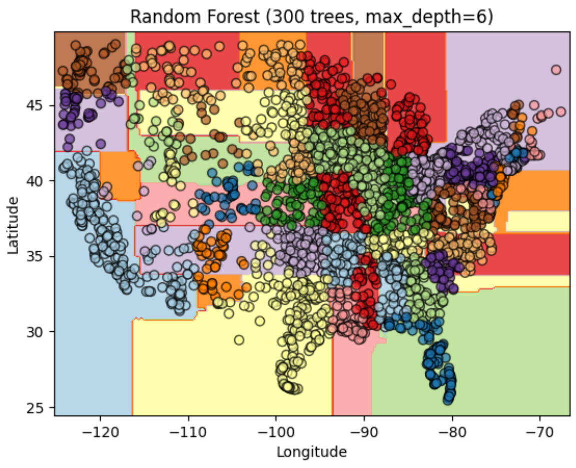
However, the 50-leaf tree still produces a noticeably coarser partition: large rectangular regions cover multiple states, and fine-grained distinctions between neighboring states are lost. In contrast, the best model (with 1000 leaves) generates a much more detailed and accurate decision map.
Thus, while the overall shape remains recognizable, the restricted tree lacks the resolution needed for precise classification.

Q6: Restricted depth.



When the maximal depth is restricted to 6, the tree can perform only a limited number of axis-aligned splits. As a result, the decision space becomes very coarse: the map is divided into large rectangular regions that group many states together. Fine-grained boundaries disappear, and the model cannot approximate the true geographic structure of the states.
Compared to the high-capacity tree from Q4 (with 1000 leaves and large depth), which produces a highly detailed and fragmented decision map, the depth-6 tree generates only a small number of broad partitions. The result is a much simpler and less expressive division of the space.

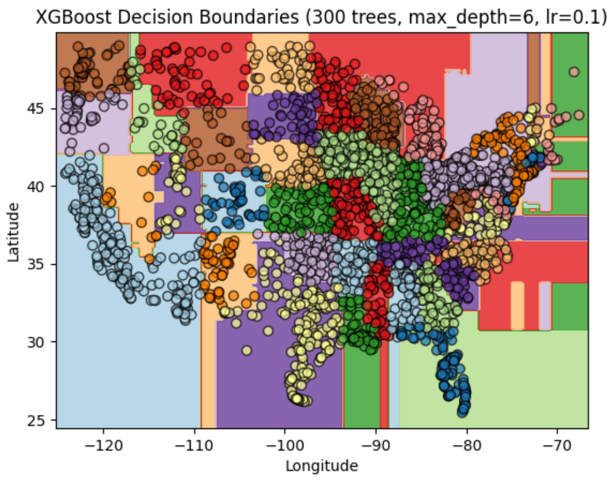
Q7.Random Forests.



The random forest consisting of 300 trees with maximal depth 6 produces a more refined decision boundary than a single depth-6 decision tree. Each shallow tree partitions the space coarsely, but because the forest aggregates many slightly different partitions, the resulting decision map becomes smoother and more detailed than that of an individual tree. This leads to improved generalization and more expressive boundaries.

However, the random forest is still less expressive than the best deep tree from Q1. Since all trees in the forest are restricted to depth 6, the model cannot capture fine-grained geographic structures in the same way that a deep, high-leaf-count tree can.
The visualization clearly shows this: the forest creates smoother boundaries than the shallow tree, but still lacks the high-resolution detail seen in the deep decision tree.

Q8.Experimenting with XGBoost.



The XGBoost model with 300 trees, max depth 6 and learning rate 0.1 achieves train accuracy of 0.992, validation accuracy of 0.974 and test accuracy of 0.965. Compared to the random forest with the same depth and number of trees, the XGBoost decision boundaries are less noisy and slightly smoother, and the model achieves a slightly higher test accuracy.
This happens because boosting trains trees sequentially, where each tree focuses on correcting the mistakes of the previous ensemble, while random forests train trees independently using bagging. As a result, in this task XGBoost is marginally more successful than the random forest, both in terms of accuracy and in the quality of the decision boundaries.