

A Proofs

A.1 Proof of Theorem 1

Consider the density estimator for the samples populating the polytopes \hat{f} . Let n be the total number of samples and n_r be the number of data points within polytope Q_r . As stated in the theorem statement, we make the following assumptions:

1. The polytope bandwidth $h_n \rightarrow 0$ as $n \rightarrow \infty$.
2. n grows faster than the shrinkage of h_n , i.e., $n \cdot h_n \rightarrow \infty$ as $h_n \rightarrow 0$ in probability.

For simplicity, we first explore the one-dimensional distribution. The derivation can be readily extended to multi-dimensional scenarios. We consider any Gaussian kernel $\mathcal{G}'()$ with parameters chosen independently of the data satisfying two conditions. In lemma 3 and 4 we will show that the aforementioned class of estimators is consistent. The conditions for choosing the Gaussian kernel parameters are:

1. The center of the kernel can be any point z_r within the polytope Q_r as $n \rightarrow \infty$,
2. The kernel bandwidth σ_r is any non-negative number always bounded by the polytope bandwidth h_n as $n \rightarrow \infty$, i.e., $\sigma_r = C_r h_n$, where $0 < C_r \leq 1$.

Now the class conditional density estimate at a point x can be written as:

$$\hat{f}(x) = \frac{1}{n} \sum_{r \in \mathcal{P}} n_r \mathcal{G}'(x; \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*). \quad (19)$$

Lemma 3. *The class of estimators in (19) is an asymptotically unbiased class of estimators of the true density f .*

Proof. The polytope sample counts n_r can be considered as binomially distributed: $n_r \sim B(n, P_r)$, where $P_r = \int_{Q_r} dF$ is the probability of finding a training sample within the polytope r and F is the cumulative density function associated with the density f . This allows us to write: $\mathbb{E}[n_r] = n P_r$. Using the mean value theorem, we have that $P_r = h_n f(q_r)$, for some $q_r \in Q_r$. Note that, if we consider the multi-dimensional scenario for the mean value theorem here, the proof can be easily generalized for multi-dimensional case.

Now consider the expectation of \hat{f} with respect to the training distribution:

$$\mathbb{E}[\hat{f}(x)] = \sum_{r \in \mathcal{P}} \frac{n P_r \mathcal{G}'(x; \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*)}{n} = \sum_{r \in \mathcal{P}} h_n f(q_r) \mathcal{G}'(x; \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*). \quad (20)$$

Note that we are given the partitions of the feature space and we choose Gaussian parameters such that they are independent of the training data. Therefore, $\mathcal{G}'(x, \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*)$ is not a random variable. Now, as $n \rightarrow \infty$, $h_n \rightarrow 0$, and h_n can be considered an infinitesimal measure dz_r . Furthermore, as the bandwidth of the Gaussian is limited by the polytope bandwidth and the area under the Gaussian is 1, the kernel $\mathcal{G}'(x, \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*)$ becomes a dirac delta function evaluated at x as $h_n \rightarrow 0$. Therefore, in the limiting conditions, $\mathcal{G}'(x, \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*) \rightarrow \delta(x - z_r)$ and $f(q_r) \rightarrow f(z_r)$. Here z_r is a point such that $z_r \in Q_r$ for all n . As $n \rightarrow \infty$, the summation in (20) thus becomes an integral over the space \mathbb{R} . Therefore, in the limit we can write (20) as

$$\begin{aligned} \mathbb{E}[\hat{f}(x)] &= \int_{-\infty}^{\infty} f(z_r) \delta(x - z_r) dz_r \\ &= f(x) \end{aligned} \quad (21)$$

Therefore, $\hat{f}(x)$ is an asymptotically unbiased estimator of $f(x)$.

□

Lemma 4. *The variance of the class of estimators in (19) asymptotically goes to 0.*

399 *Proof.* For binomially distributed samples n_r , we can write, $\mathbb{V}ar[n_r] = nP_r(1 - P_r)$. Therefore,
 400 we can estimate the variance of $\hat{f}(x)$ as:

$$\begin{aligned}
 \mathbb{V}ar[\hat{f}(x)] &= \sum_{r \in \mathcal{P}} \frac{nP_r(1 - P_r)}{n^2} \{\mathcal{G}'(x; \mu_r, \Sigma_r) \mathbb{1}(r = r_x^*)\}^2 \\
 &= \frac{h_n f(z_{r_x^*})(1 - h_n f(z_{r_x^*}))}{n} \left\{ \frac{1}{\sqrt{2\pi}\sigma_{r_x^*}} \exp\left(-\frac{(x - \mu_{r_x^*})^2}{2\sigma_{r_x^*}^2}\right) \right\}^2 \\
 &= \frac{h_n f(z_{r_x^*})(1 - h_n f(z_{r_x^*}))}{nC_{r_x^*}^2 h_n^2} \left\{ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_{r_x^*})^2}{2\sigma_{r_x^*}^2}\right) \right\}^2 \\
 &\leq \frac{f(z_{r_x^*})(1 - h_n f(z_{r_x^*}))}{2\pi C_{r_x^*}^2 (nh_n)}
 \end{aligned} \tag{22}$$

401 Equation 22 offers several interesting insights:

- 402 1. As $h_n \rightarrow 0$, $z_{r_x^*} \rightarrow x$ and $f(z_{r_x^*})(1 - h_n f(z_{r_x^*})) \rightarrow f(x)$. Therefore, the variance becomes
 403 directly dependent on $f(x)$. There is more variability at regions with higher density.
- 404 2. The variance is also higher if $C_{r_x^*}^2 \ll 1$. Therefore, $C_{r_x^*}$ should be as close to 1 as possible
 405 and nh_n should be as high as possible for lower variance. Moreover, variance can be
 406 reduced at the expense of higher bias with $C_{r_x^*} > 1$. Most importantly note that, the effect
 407 of $C_{r_x^*}$ cancels out in the numerator and the denominator while estimating the posteriors in
 408 Equation 7.
- 409 3. As $h_n \rightarrow 0$ the estimation becomes unbiased (see Equation 20), but the estimation variance
 410 becomes unbounded. Therefore, for bounded and decreasing variance the condition $nh_n \rightarrow$
 411 ∞ is necessary.

412 □

413 Lemma 3 and 4 together completes the proof of Theorem 1.

414 A.2 Proof of Theorem 2

415 We first expand $\hat{g}_y(\mathbf{x})$:

$$\begin{aligned}
 \hat{g}_y(\mathbf{x}) &= \frac{\hat{f}_y(\mathbf{x}) \hat{P}_Y(y)}{\sum_{k=1}^K \hat{f}_k(x) \hat{P}_Y(k)} \\
 &= \frac{\tilde{f}_y(\mathbf{x}) \hat{P}_Y(y) + \frac{b}{\log(n)} \hat{P}_Y(y)}{\sum_{k=1}^K (\hat{f}_k(\mathbf{x}) \hat{P}_Y(k) + \frac{b}{\log(n)} \hat{P}_Y(k))}
 \end{aligned}$$

As the inference point \mathbf{x} becomes more distant from training samples (and more distant from all of the Gaussian centers), we have that $\mathcal{G}(\mathbf{x}, \hat{\mu}_r, \hat{\Sigma}_r)$ becomes smaller. Thus, $\forall y$, $\tilde{f}_y(\mathbf{x})$ shrinks. More formally, $\forall y$,

$$\lim_{d_{\mathbf{x}} \rightarrow \infty} \tilde{f}_y(\mathbf{x}) = 0$$

416 We can use this result to then examine the limiting behavior of our posteriors as the inference point \mathbf{x}
 417 becomes more distant from the training data:

$$\begin{aligned}
 \lim_{d_{\mathbf{x}} \rightarrow \infty} \hat{g}_y(\mathbf{x}) &= \lim_{d_{\mathbf{x}} \rightarrow \infty} \frac{\tilde{f}_y(\mathbf{x}) \hat{P}_Y(y) + \frac{b}{\log(n)} \hat{P}_Y(y)}{\sum_{k=1}^K (\tilde{f}_k(\mathbf{x}) \hat{P}_Y(k) + \frac{b}{\log(n)} \hat{P}_Y(k))} \\
 &= \frac{(\lim_{d_{\mathbf{x}} \rightarrow \infty} \tilde{f}_y(\mathbf{x})) \hat{P}_Y(y) + \frac{b}{\log(n)} \hat{P}_Y(y)}{\sum_{k=1}^K (\lim_{d_{\mathbf{x}} \rightarrow \infty} \tilde{f}_k(\mathbf{x})) \hat{P}_Y(k) + \frac{b}{\log(n)} \hat{P}_Y(k)} \\
 &= \frac{\hat{P}_Y(y)}{\sum_{k=1}^K \hat{P}_Y(k)} \\
 &= \hat{P}_Y(y)
 \end{aligned}$$

418 B Simulations

419 We construct five types of binary class simulations:

- 420 • *Gaussian XOR* is a two-class classification problem with equal class priors. Conditioned
 421 on being in class 0, a sample is drawn from a mixture of two Gaussians with means
 422 $\pm[0.5, -0.5]^\top$ and standard deviations of 0.25. Conditioned on being in class 1, a sample is
 423 drawn from a mixture of two Gaussians with means $\pm[0.5, -0.5]^\top$ and standard deviations
 424 of 0.25.
- 425 • *Spiral* is a two-class classification problem with the following data distributions: let K
 426 be the number of classes and $S \sim \text{multinomial}(\frac{1}{K} \mathbf{1}_K, n)$. Conditioned on S , each feature
 427 vector is parameterized by two variables, the radius r and an angle θ . For each sample,
 428 r is sampled uniformly in $[0, 1]$. Conditioned on a particular class, the angles are evenly
 429 spaced between $\frac{4\pi(k-1)t_K}{K}$ and $\frac{4\pi(k)t_K}{K}$, where t_K controls the number of turns in the
 430 spiral. To inject noise along the spirals, we add Gaussian noise to the evenly spaced angles
 431 $\theta' : \theta = \theta' + \mathcal{N}(0, 0.09)$. The observed feature vector is then $(r \cos(\theta), r \sin(\theta))$.
- 432 • *Circle* is a two-class classification problem with equal class priors. Conditioned on being
 433 in class 0, a sample is drawn from a circle centered at $(0, 0)$ with a radius of $r = 0.75$.
 434 Conditioned on being in class 1, a sample is drawn from a circle centered at $(0, 0)$ with a
 435 radius of $r = 1$, which is cut off by the region bounds. To inject noise along the circles, we
 436 add Gaussian noise to the circle radii $r' : r = r' + \mathcal{N}(0, 0.01)$.
- 437 • *Sinewave* is a two-class classification problem based on sine waves. Conditioned on being
 438 in class 0, a sample is drawn from the distribution $y = \cos(\pi x)$. Conditioned on being in
 439 class 1, a sample is drawn from the distribution $y = \sin(\pi x)$. We inject Gaussian noise to
 440 the sine wave heights $y' : y = y' + \mathcal{N}(0, 0.01)$.
- 441 • *Polynomial* is a two-class classification problem with the following data distributions:
 442 $y = x^a$. Conditioned on being in class 0, a sample is drawn from the distribution $y = x^1$.
 443 Conditioned on being in class 1, a sample is drawn from the distribution $y = x^3$. Gaussian
 444 noise is added to variables $y' : y = y' + \mathcal{N}(0, 0.01)$.

Table 1: Hyperparameters for RF and KGF.

Hyperparameters	Value
n_estimators	500
max_depth	∞
min_samples_leaf	1
$O(n)$	$1 + \lfloor \log(n)/3 \rfloor$
b (bias)	$\exp(-10^{\sqrt{d}})$
λ	1×10^{-6}

Table 2: Hyperparameters for ReLU-net and KGN.

Hyperparameters	Value
number of hidden layers	4
nodes per hidden layer	1000
optimizer	Adam
learning rate	3×10^{-4}
$O(n)$	$\log_2(n)$
b (bias)	$\exp(-10^{\sqrt{d}})$
λ	1×10^{-6}

445 C Pseudocodes

In this section, we provide the pseudocode for our proposed algorithms.

Algorithm 1 Fit a KGX model.

Input:

- (1) θ ▷ Parent learner (random forest or deep network model)
(2) $\mathcal{D}_n = (\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{n \times d} \times \{1, \dots, K\}^n$ ▷ Training data

Output: \mathcal{G} ▷ a KGX model

```

1: function KGX.FIT( $\theta, \mathbf{X}, \mathbf{y}$ )
2:   for  $i = 1, \dots, n$  do ▷ Iterate over the dataset to calculate the weights
3:     for  $j = 1, \dots, n$  do
4:        $w_{ij} \leftarrow \text{COMPUTEWEIGHTS}(\mathbf{x}_i, \mathbf{x}_j, \theta)$ 
5:     end for
6:   end for
7:
8:
9:    $\{Q_r\}_{r=1}^{\tilde{p}} \leftarrow \text{GETPOLYTOPES}(\mathbf{w})$  ▷ Identify the polytopes by clustering the samples with
   similar weight
10:
11:    $\mathcal{G}.\{\tilde{w}_k\}_{k=1}^K \leftarrow 0$  ▷ Initialize the counts for each class
12:   for  $r = 1, \dots, \tilde{p}$  do ▷ Iterate over each polytope
13:     for  $k = 1, \dots, K$  do
14:        $\mathcal{G}.\tilde{w}_{rk} \leftarrow \text{COUNTWEIGHTS}(\{\mathbf{w}_{rs}\}_{s=1}^{\tilde{p}}, k)$  ▷  $w_{rk}$  is the number of weighted input
       samples in  $Q_r$  with label  $k$ 
15:        $\mathcal{G}.\tilde{w}_k \leftarrow \mathcal{G}.\tilde{w}_k + \mathcal{G}.\tilde{w}_{rk}$  ▷ Update the total count for each class
16:     end for
17:    $\mathcal{G}.\hat{\mu}_r, \mathcal{G}.\hat{\Sigma}_r \leftarrow \text{ESTIMATEPARAMETERS}(\mathbf{X}, \{\mathbf{w}_{rs}\}_{s=1}^{\tilde{p}})$  ▷ Fit Gaussians using weighted
   MLE
18:   end for
19:   return  $\mathcal{G}$ 
20: end function

```

446

447 D Hardware and Software Configurations

- 448 • Operating System: Linux (ubuntu 20.04), macOS (Ventura 13.2.1)
- 449 • VM Size: Azure Standard D96as v4 (96 vcpus, 384 GiB memory)
- 450 • GPU: Apple M1 Max
- 451 • Software: Python 3.8, scikit-learn $\geq 0.22.0$, tensorflow-macos ≤ 2.9 , tensorflow-metal \leq
- 452 0.5.0.

Algorithm 2 Computing weights in KGF

Input:

- (1) $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{1 \times d}$ ▷ two input samples to be weighted
- (2) θ ▷ parent random forest with T trees

Output: $w_{ij} \in [0, 1]$ ▷ compute similarity between i and j -th samples.

```
1: function COMPUTEWEIGHTS( $\mathbf{x}_i, \mathbf{x}_j, \theta$ )
2:    $\mathcal{I}_i \leftarrow \text{PUSHDOWNTREES}(\mathbf{x}_i, \theta)$  ▷ push  $\mathbf{x}_i$  down  $T$  trees and get the leaf numbers it end up in.
3:    $\mathcal{I}_j \leftarrow \text{PUSHDOWNTREES}(\mathbf{x}_j, \theta)$  ▷ push  $\mathbf{x}_j$  down  $T$  trees and get the leaf numbers it end up in.
4:    $l \leftarrow \text{COUNTMATCHES}(\mathcal{I}_i, \mathcal{I}_j)$  ▷ count the number of times the samples end up in the same leaf
5:    $w_{ij} \leftarrow \frac{l}{T}$ 
6:   return  $w_{ij}$ 
7: end function
```

Algorithm 3 Computing weights in KGN

Input:

- (1) $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{1 \times d}$ ▷ two input samples to be weighted
- (2) θ ▷ parent deep-net model

Output: $w_{ij} \in [0, 1]$ ▷ compute similarity between i and j -th samples.

```
1: function COMPUTEWEIGHTS( $\mathbf{x}_i, \mathbf{x}_j, \theta$ )
2:    $\mathcal{A}_i \leftarrow \text{PUSHDOWNNETWORK}(\mathbf{x}_i, \theta)$  ▷ get activation modes  $\mathcal{A}_i$ 
3:    $\mathcal{A}_j \leftarrow \text{PUSHDOWNNETWORK}(\mathbf{x}_j, \theta)$  ▷ get activation modes  $\mathcal{A}_j$ 
4:    $l \leftarrow \text{COUNTMATCHES}(\mathcal{A}_i, \mathcal{A}_j)$  ▷ count the number of times the two samples activate the activation paths in a similar way
5:    $w_{ij} \leftarrow \frac{l}{N}$  ▷  $N$  is the total number of activation paths
6:   return  $w_{ij}$ 
7: end function
```

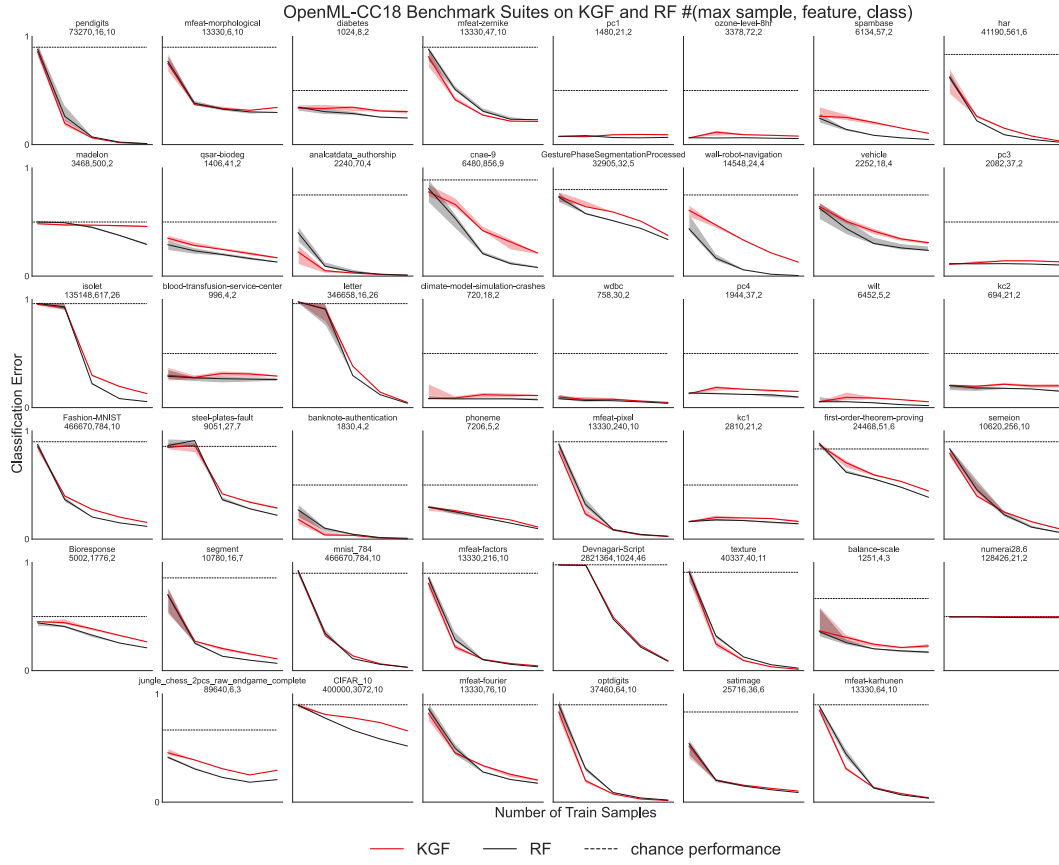


Figure 4: **Extended results on OpenML-CC18 datasets.** Classifications errors for KGF and RF with max sample size, number of features, and number of classes mentioned in the title for each panel.

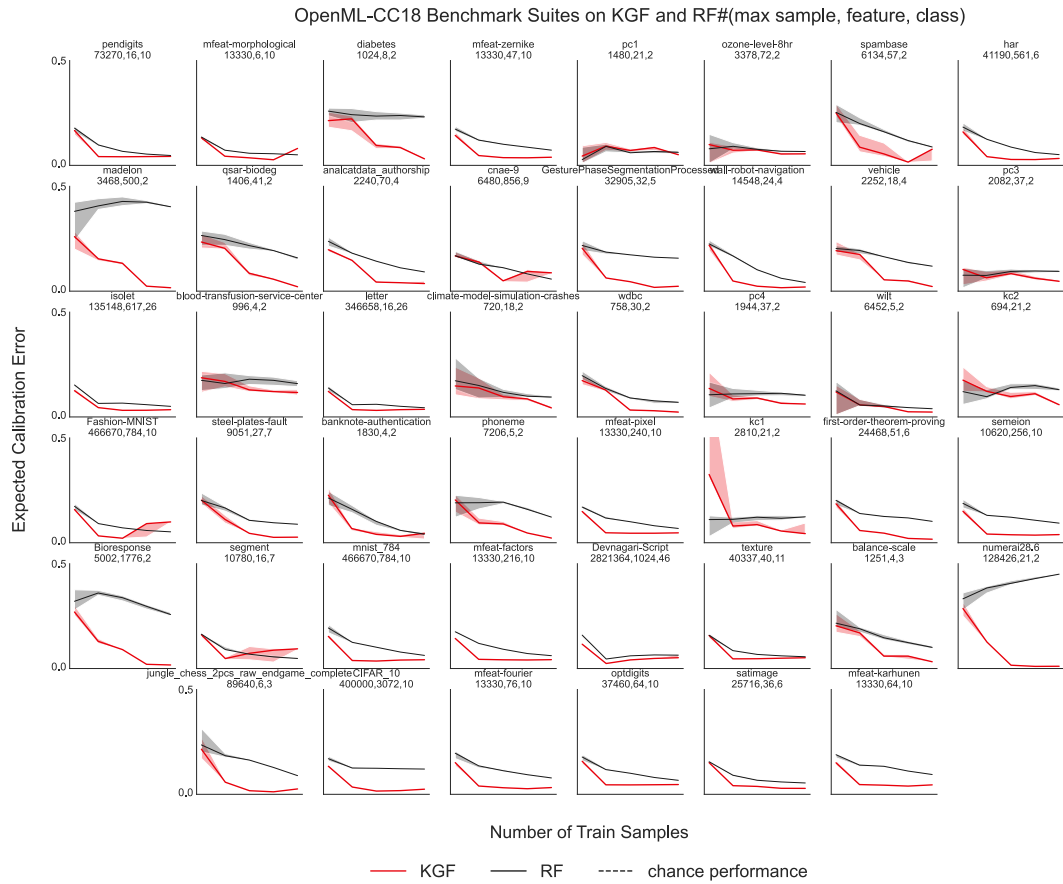


Figure 5: **Extended results on OpenML-CC18 datasets.** Calibration errors for KGF and RF with max sample size, number of features, and number of classes mentioned in the title for each panel.

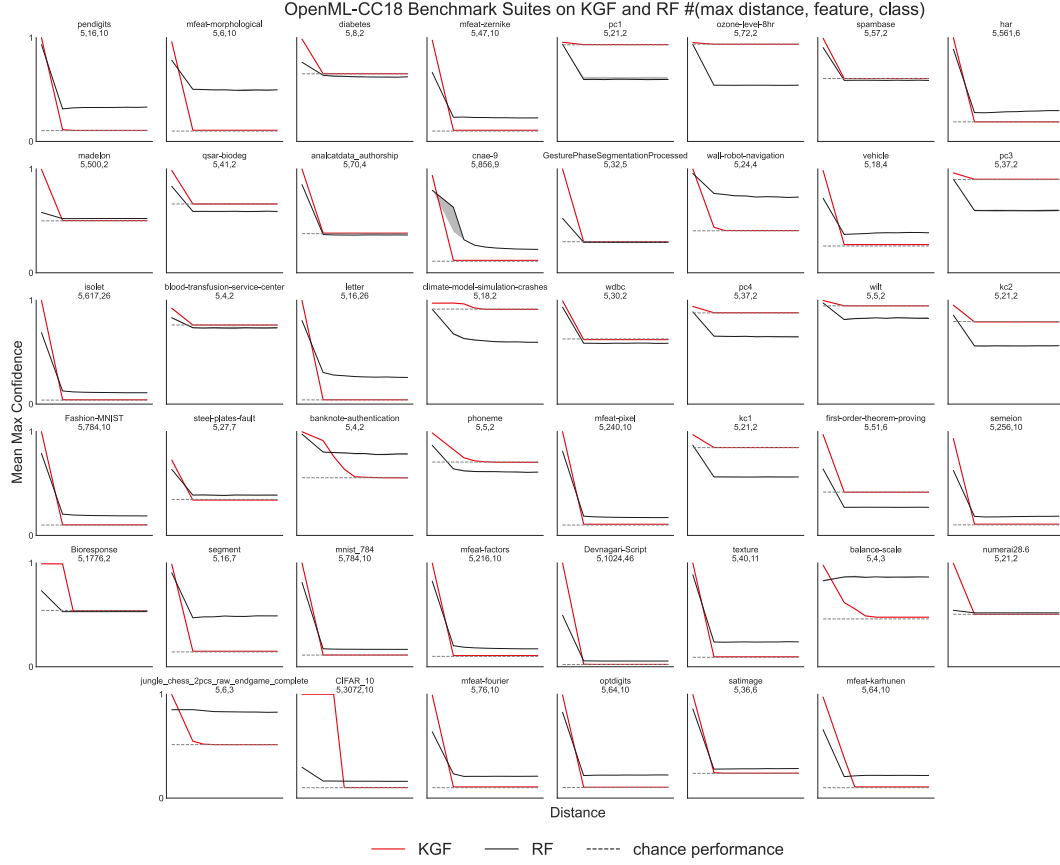


Figure 6: **Extended results on OpenML-CC18 datasets.** Mean max confidence for KGF and RF with max distance from the data origin, number of features, and number of classes mentioned in the title for each panel. The dashed line indicates the maximum of the empirical priors for different class labels.

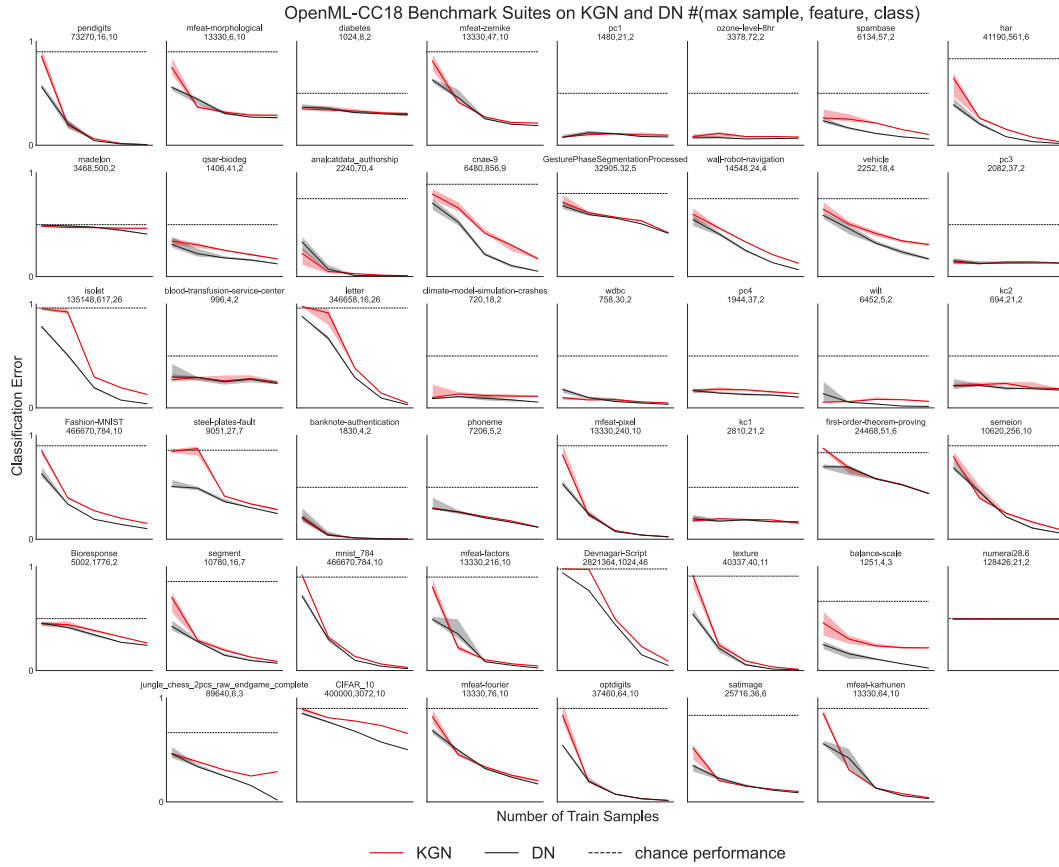


Figure 7: **Extended results on OpenML-CC18 datasets.** Classifications errors for KGN and RELU-net with max sample size, number of features, and number of classes mentioned in the title for each panel.



Figure 8: **Extended results on OpenML-CC18 datasets.** Calibration errors for KGN and RELU-net with max sample size, number of features, and number of classes mentioned in the title for each panel.

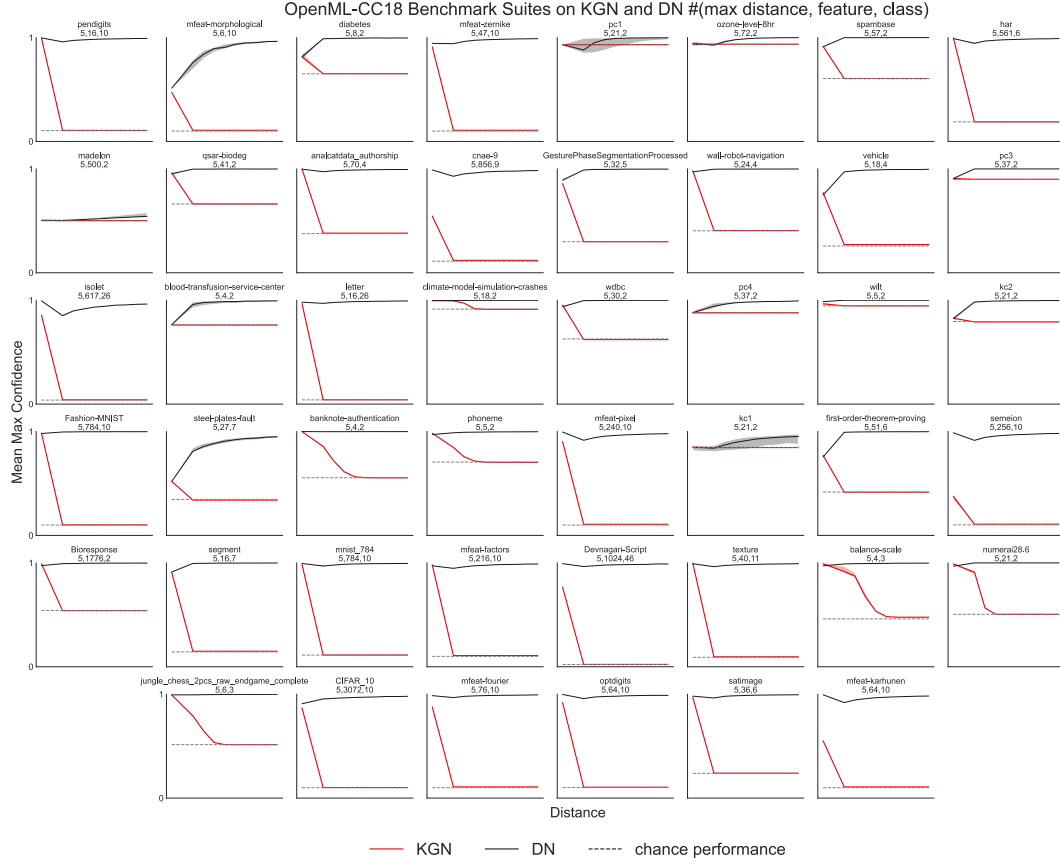


Figure 9: **Extended results on OpenML-CC18 datasets.** Mean max confidence for KGN and RLU-net with max distance from the data origin, number of features, and number of classes mentioned in the title for each panel. The dashed line indicates the maximum of the empirical priors for different class labels.