University of Bucharest
Faculty of Mathematics and Informatics
2025–2026

Practical Machine Learning
Pairwise Image Distribution Classification

Popescu Pavel–Yanis
Group 407 — Artificial Intelligence

Contents

1 Introduction

The Kaggle competition addressed in this project defines a binary classification task over pairs of images. Given two images, the objective is to decide whether they originate from the same underlying distribution. The problem is formulated as supervised learning and relies entirely on manually engineered feature representations, with emphasis placed on low-level pixel statistics rather than semantic content.

The task is tricky because of the strong visual similarity between the image pairs. The images are noisy, textures overlap, and clear semantic structures are pretty absent. Global image descriptors provide limited discriminative power, as feature distributions tend to overlap significantly. Informative differences appear locally, in small regions, making local comparison fit for this problem.

Given these constraints, classical machine learning methods combined with explicit feature engineering seem and really are appropriate. Distance-based models such as K-Nearest Neighbors and margin-based models such as Support Vector Machines allow direct control over similarity measures, regularization strength, and decision boundaries.

Model performance is evaluated using both error-based and ranking-based criteria. Mean Absolute Error (MAE) and Mean Squared Error (MSE) measure numerical prediction quality, while Spearman and Kendall correlations assess the preservation of relative sample ordering.

All experiments are implemented in Python. No external datasets and no pretrained models are used.

2 Dataset and Preprocessing

The data set contains pairs of grayscale images and defines a binary classification task. The training set has 6 078 samples, the validation set has 1 800 samples, and the test set has 4 104 samples. Each training and validation sample has two image identifiers and a binary label indicating whether the images originate from the same distribution. The test samples have only image identifiers and no labels.

The modeling is based on visual information taken from the image pairs. No semantic annotations or auxiliary metadata are provided. The training set is used for feature extraction, model fitting and hyperparameter tuning, while the validation set is used for evaluation. The test set is used only for the generation of the final prediction.

Images are processed at their original resolution, without resizing or having any geometric transformations done to them, to preserve spatial structure for local feature computation. Pixel intensities are normalized using a global mean and standard deviation computed only from the training images. The same normalization parameters are applied unchanged to the validation and the test data.

No data augmentation techniques are applied. In accordance with the competition rules, no external datasets and no pretrained models are used.

3 Exploratory Analysis and Motivation of Features

Exploratory analysis is performed to understand low-level differences between image pairs and to assess whether simple statistics provide class separation.

Global statistics derived from pixel-wise absolute differences are examined first. The mean and standard deviation of these differences are calculated for both classes. The resulting distributions exhibit quite a strong overlap, indicating that just global statistics are not enough for discrimination.

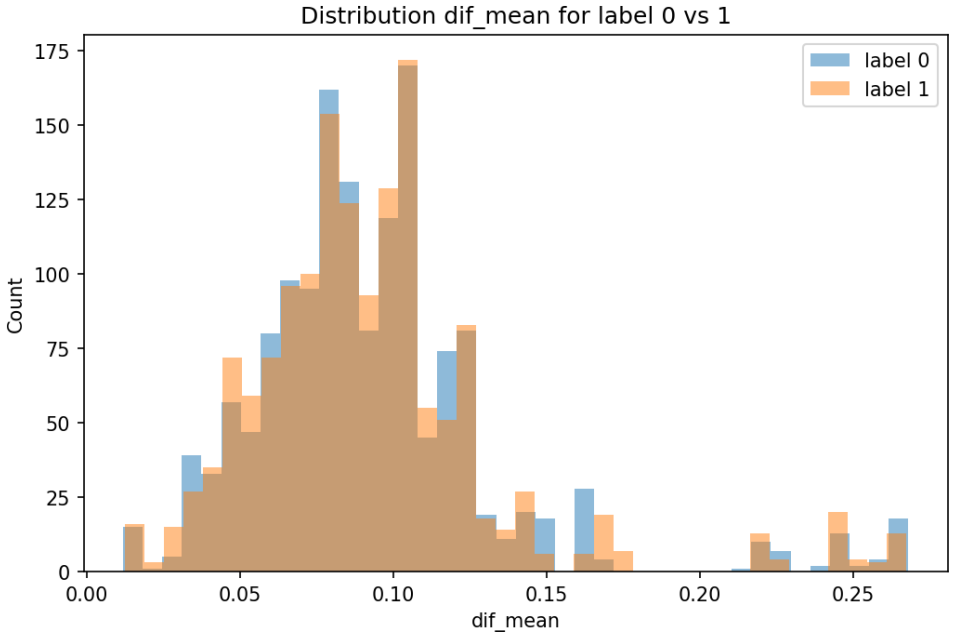

Figure 1: Distribution of mean absolute pixel-wise differences for the two classes.

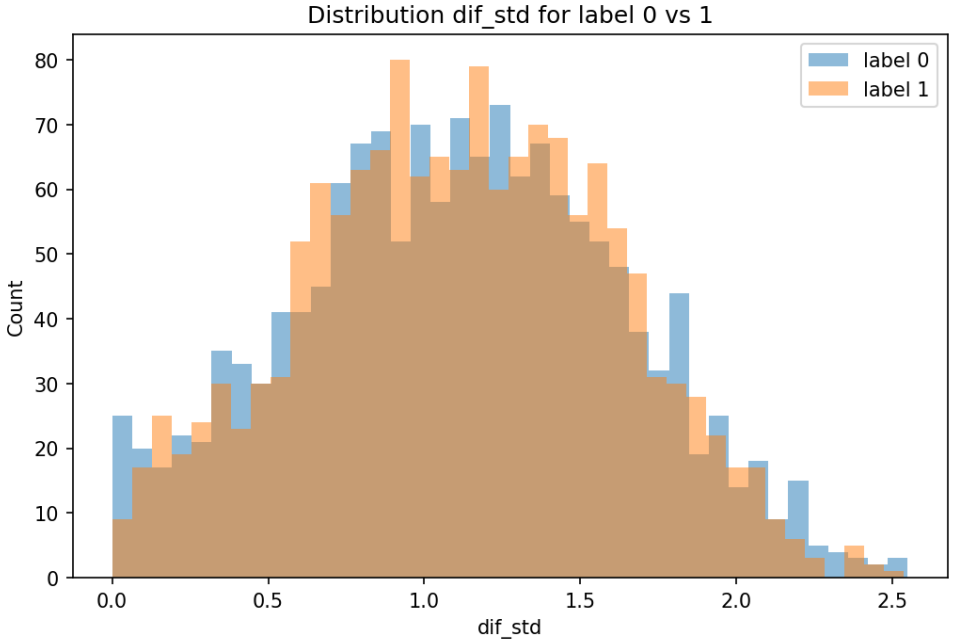

Figure 2: Distribution of the standard deviation of pixel-wise differences for the two classes.

To evaluate the relevance of spatial locality, local 8×8 window-based L2 distances are analyzed. These measures capture region-specific differences and show a slightly increased structure compared to the previous mentioned global statistics, but still show a pretty high variance and a lot of overlap. That means that local information is informative but still noisy when used in isolation.

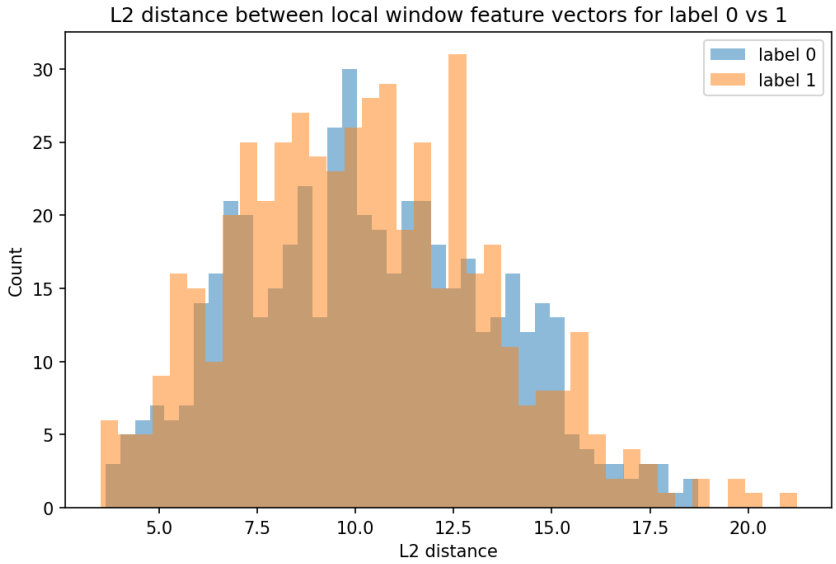

Figure 3: Distribution of local 8×8 window-based L2 distances for the two classes.

Therefore, global image statistics are not enough for this task, and relevant information appears locally but is distributed across the image. This motivates a feature representation that aggregates local measurements through spatial pooling, reducing noise while preserving spatial variation. The design is inspired by the pooling principles used in convolutional models, but remains fully explicit and hand-crafted.

The strong overlap observed across the exploratory distributions supports the use of distance-based and margin-based models. Methods such as KNN and SVM are appropriate for exploiting relative distances and decision margins in the high-dimensional feature space that results.

4 Feature Representation

Each image pair is represented by a fixed-length feature vector constructed from pixel-level interactions between the two images, being designed to capture local differences.

In an image pair, both images are normalized using a global mean and standard deviation computed only from the training set. This normalization ensures consistent intensity scaling across samples and avoids information leakage from the validation data. All the operations are applied the same way to both images in a pair.

Two interaction maps are computed: a difference map and a product map. The difference map highlights local intensity discrepancies between the two images, while the product map focuses on regions where both images exhibit similar values. Together, these maps provide complementary information about dissimilarity and alignment at the pixel level.

To reduce dimensionality and suppress the noise, the interaction maps are aggregated using spatial pooling. Each image is divided into blocks of size 16×16 that do not overlap with one another, while the mean pooling is applied independently within each block. This produces a stable spatial representation which preserves local structure while significantly reducing the feature dimensionality. The pooling strategy is inspired by convolutional architectures, but is handcrafted and without learned filters.

After the pooling part, the difference and product maps are flattened and concatenated. After this, a small set of global summary statistics is included to capture overall image relationships. These statistics consist of the mean and standard deviation of each image, the mean and maximum absolute difference on a pixel level, the L1 and L2 distances between the images, and the cosine similarity computed over the flattened image vectors.

The final feature vector combines pooled local information with global descriptors, producing a compact and expressive representation of each image pair. This representation is used across all reported KNN and SVM models described in the following sections.

5 Machine Learning Models

5.1 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a distance-based method that assigns labels based on the closest samples in feature space. Since there's no explicit model learned here, the performance depends directly on the feature representation and the distance measure.

In this task, image pairs are encoded as high-dimensional vectors combining pooled local interactions and global statistics. Due to the strong overlap observed in exploratory analyses, local neighborhood relationships are more informative than global decision boundaries. KNN exploits this structure by relying on relative distances between samples.

The behavior of KNN is controlled by the number of neighbors, the distance metric, and the neighbor weighting strategy. These are the hyperparameters which are explored experimentally. KNN is a strong distance-based baseline for evaluating the proposed feature representation.

5.2 Support Vector Machines

Support Vector Machines (SVM) are margin-based models that learn a global decision function by maximizing class separation while controlling model complexity through regularization.

Given that we have overlap present in the feature distributions, linear separation is not enough. Kernel functions are used to construct non-linear decision boundaries, making SVMs well suited for feature spaces combining local pooled information with global descriptors.

SVM performance is mainly controlled by its hyperparameters. The regularization parameter balances margin size and classification errors, while kernel-specific parameters determine boundary flexibility. These parameters are tuned in later experiments.

6 Evaluation and Hyperparameter Search

All models are evaluated using the training–validation split defined by the competition. Training data is used only for feature extraction, model fitting, and hyperparameter tuning, while the validation set is used just for evaluation. No validation data is used during training or preprocessing.

Model performance is measured with error-based and ranking-based metrics. MAE and MSE quantify prediction accuracy, while Spearman and Kendall correlations measure how well the relative ordering of the samples is preserved. Using both types of metrics allows accuracy and ranking quality to be assessed together.

Hyperparameters are selected using grid search. For each model, all predefined hyperparameter combinations are evaluated on the validation set under the same protocol, ensuring fair comparison between configurations.

For KNN, the search covers the number of neighbors, distance metrics, and neighbor weighting strategies. Two scaling variants are evaluated, and both broad and refined searches are used to identify promising regions of the hyperparameter space.

For SVM, the same feature representation is used throughout. The grid search focuses on the regularization parameter and kernel-specific parameters, with the same search space applied to all SVM runs.

All reported results are produced using this protocol. The best-performing configurations on the validation set are selected, while weaker configurations are kept only for comparison.

7 KNN Experiments and Results

This section reports validation results for K-Nearest Neighbors (KNN) using the final feature representation from Section 4 (mean pooling with 16×16 blocks plus global statistics). Two variants are evaluated, sharing the same features and being different only by the scaling applied to the feature vectors:

- **KNN + StandardScaler**
- **KNN + RobustScaler**

For each variant, a broad grid search (v1) is followed by a refined search (v2). All metrics are computed on the validation set: `validation accuracy`, `MAE`, `MSE`, `Spearman`, and `Kendall`.

7.1 KNN + StandardScaler (refined search, v2)

The refined search explores neighborhood size (k), distance metric, and neighbor weighting. The first table reports three representative configurations: one that maximizes validation accuracy, one that minimizes MAE/MSE, and one that maximizes Spearman/Kendall. The second table lists the top configurations ranked by validation accuracy.

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	457	distance	1.000000	0.695556	0.480376	0.233148	0.439640	0.359066
cosine	NaN	305	distance	1.000000	0.659444	0.478697	0.232163	0.419669	0.342755
cosine	NaN	471	distance	1.000000	0.691111	0.480475	0.233204	0.443028	0.361833

Table 1: KNN (StandardScaler, v2): configurations optimizing individual validation metrics.

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	457	distance	1.000000	0.695556	0.480376	0.233148	0.439640	0.359066
cosine	NaN	461	distance	1.000000	0.695000	0.480430	0.233187	0.440243	0.359559
cosine	NaN	457	uniform	0.681639	0.694444	0.482570	0.234224	0.434141	0.357462
cosine	NaN	467	distance	1.000000	0.693889	0.480415	0.233157	0.442779	0.361630
cosine	NaN	463	distance	1.000000	0.693333	0.480385	0.233140	0.442093	0.361070
cosine	NaN	447	distance	1.000000	0.693333	0.480368	0.233162	0.436656	0.356629
cosine	NaN	455	uniform	0.680816	0.692778	0.482595	0.234249	0.432483	0.356086
cosine	NaN	481	distance	1.000000	0.692778	0.480599	0.233309	0.440893	0.360090
cosine	NaN	465	distance	1.000000	0.692222	0.480426	0.233169	0.442198	0.361155
cosine	NaN	443	distance	1.000000	0.692222	0.480332	0.233139	0.436703	0.356668
cosine	NaN	443	uniform	0.680322	0.691667	0.482516	0.234203	0.431589	0.355433
cosine	NaN	469	distance	1.000000	0.691667	0.480467	0.233205	0.441684	0.360736
cosine	NaN	445	uniform	0.679829	0.691667	0.482511	0.234191	0.432579	0.356253
cosine	NaN	461	uniform	0.682461	0.691667	0.482626	0.234265	0.435002	0.358164
cosine	NaN	459	distance	1.000000	0.691667	0.480419	0.233185	0.439599	0.359033

Table 2: KNN (StandardScaler, v2): top configurations by validation accuracy.

The following figures show how each metric varies across the refined hyperparameter space.

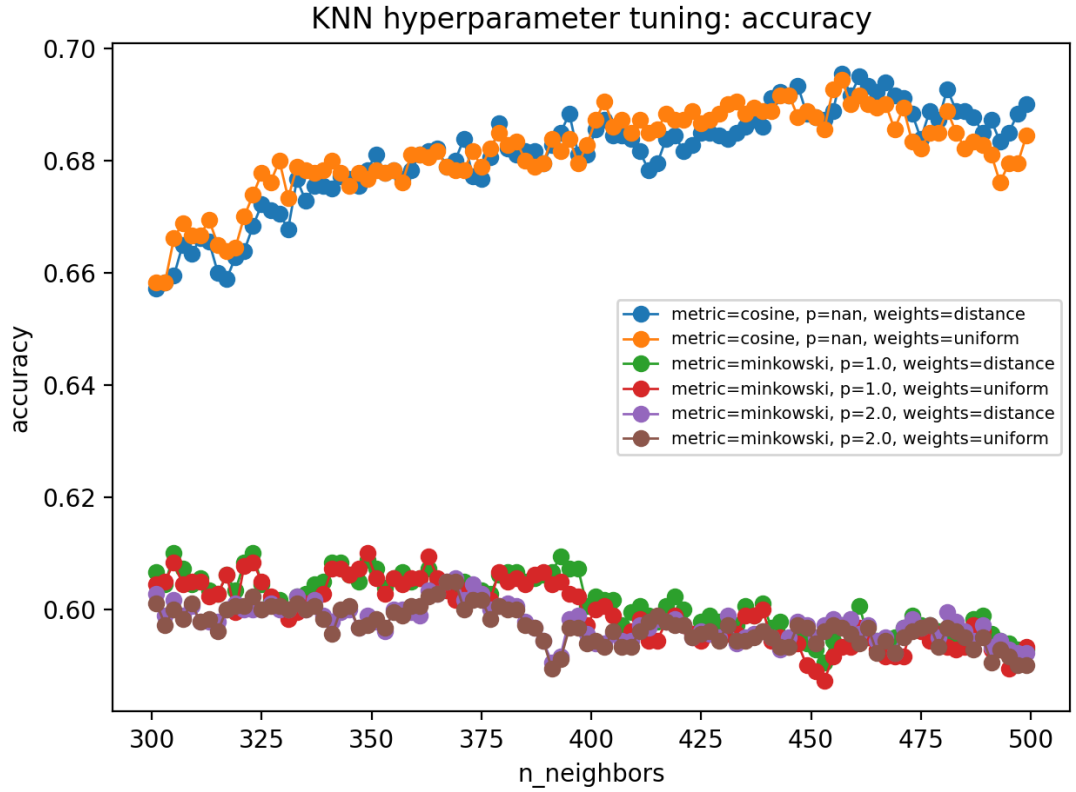

Figure 4: KNN (StandardScaler, v2): validation accuracy across tested configurations.

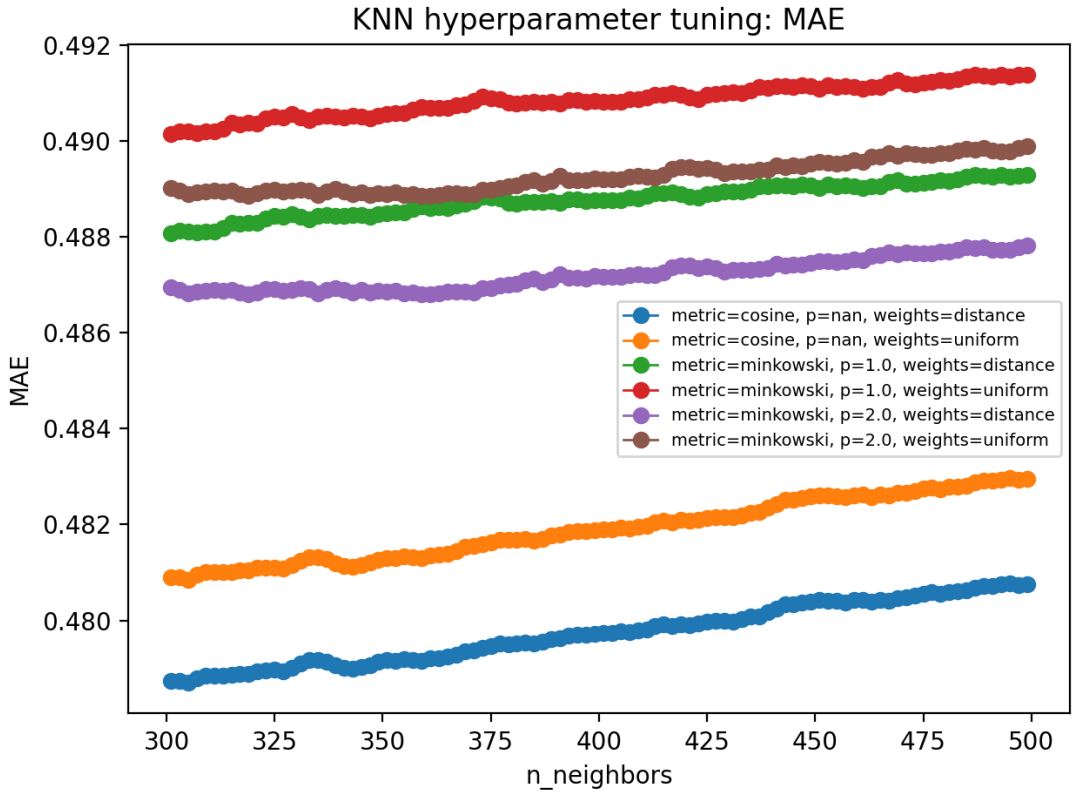

Figure 5: KNN (StandardScaler, v2): MAE across tested configurations.

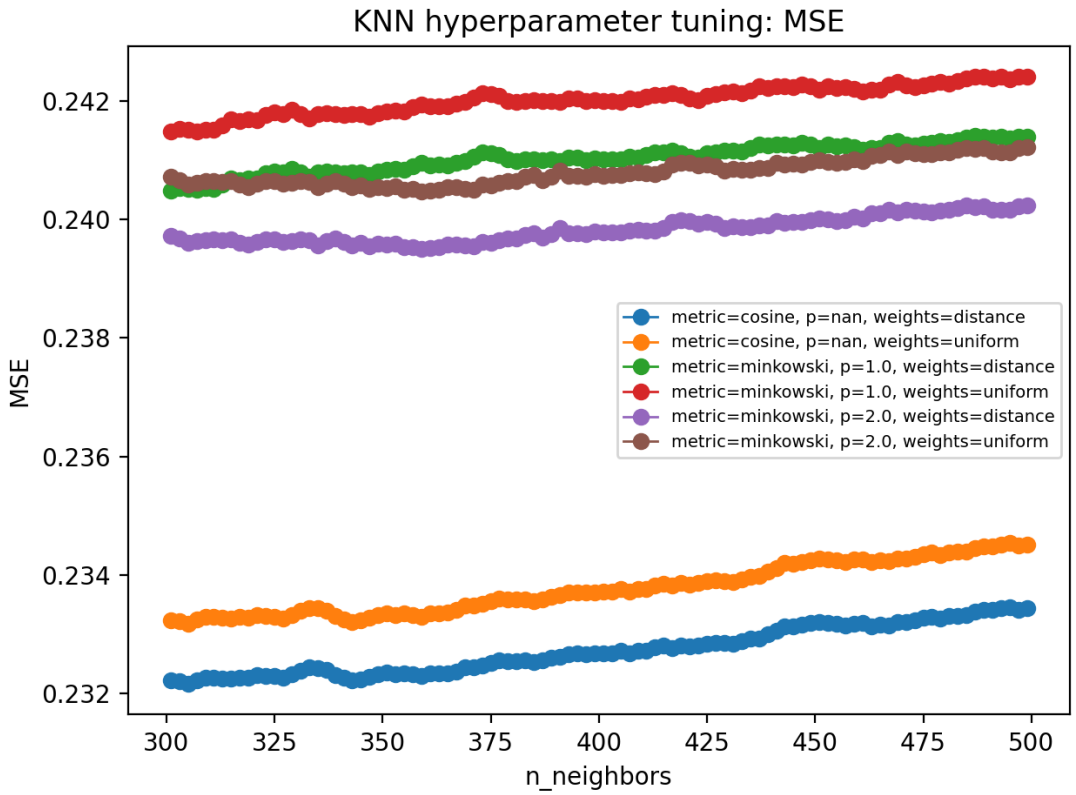

Figure 6: KNN (StandardScaler, v2): MSE across tested configurations.

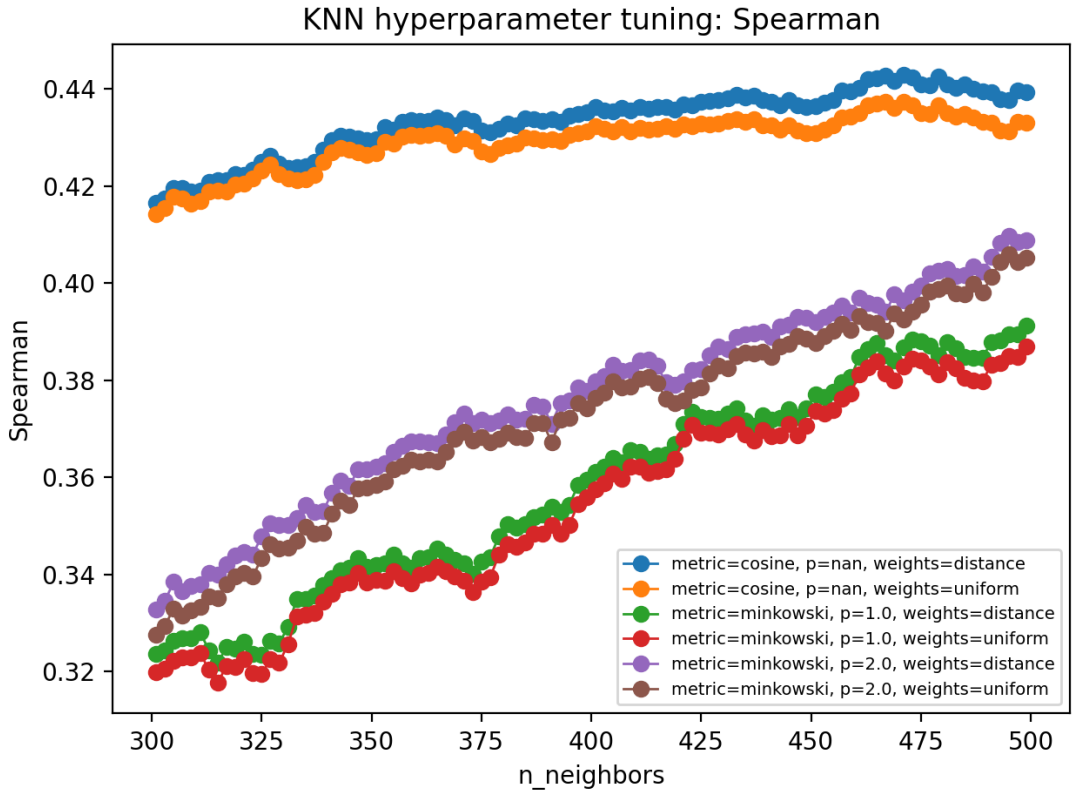

Figure 7: KNN (StandardScaler, v2): Spearman correlation across tested configurations.

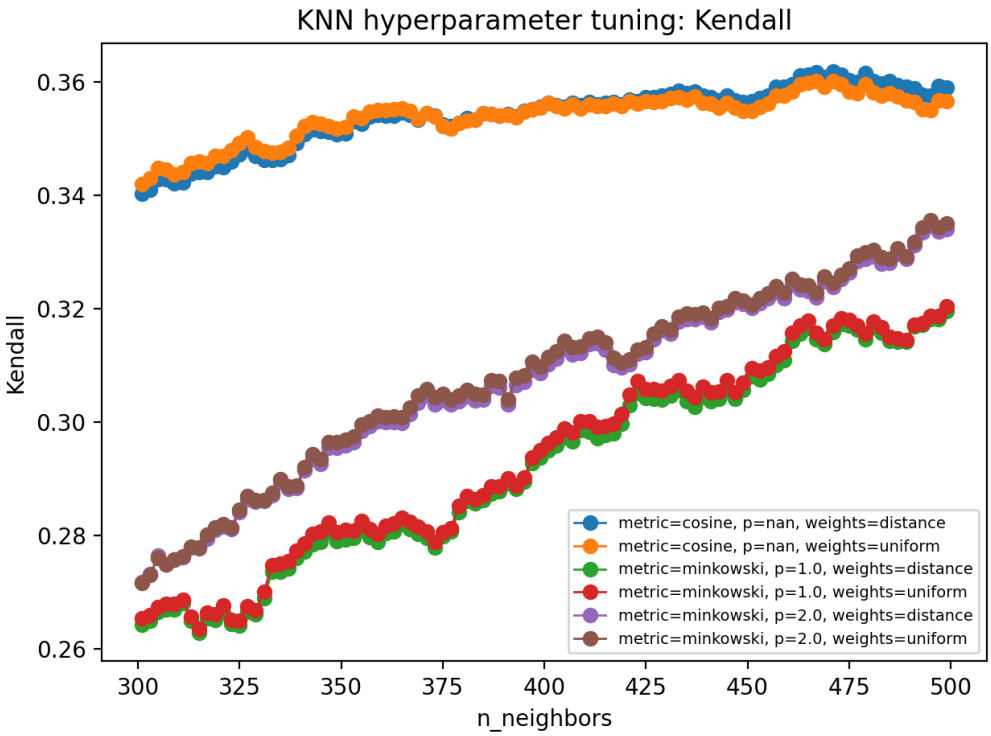

Figure 8: KNN (StandardScaler, v2): Kendall correlation across tested configurations.

7.2 KNN + RobustScaler (refined search, v2)

RobustScaler is relevant because several global features (e.g., L1/L2 distance, max absolute difference) are heavy-tailed. The same refined search protocol is applied. The tables report the same summaries as above.

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	1363	distance	1.000000	0.697222	0.484545	0.236575	0.439595	0.359028
cosine	NaN	1201	distance	1.000000	0.687778	0.483342	0.235553	0.438829	0.358402
cosine	NaN	1331	distance	1.000000	0.692778	0.484262	0.236325	0.442279	0.361220

Table 3: KNN (RobustScaler, v2): configurations optimizing individual validation metrics.

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	1363	distance	1.000000	0.697222	0.484545	0.236575	0.439595	0.359028
cosine	NaN	1307	distance	1.000000	0.696667	0.484133	0.236220	0.441537	0.360614
cosine	NaN	1305	distance	1.000000	0.696111	0.484137	0.236225	0.440850	0.360053
cosine	NaN	1361	distance	1.000000	0.696111	0.484515	0.236547	0.440373	0.359664
cosine	NaN	1299	distance	1.000000	0.695556	0.484087	0.236183	0.440784	0.359999
cosine	NaN	1311	distance	1.000000	0.695000	0.484177	0.236258	0.441011	0.360184
cosine	NaN	1297	distance	1.000000	0.695000	0.484071	0.236168	0.440726	0.359952
cosine	NaN	1365	distance	1.000000	0.695000	0.484538	0.236568	0.440140	0.359473
cosine	NaN	1309	distance	1.000000	0.694444	0.484173	0.236256	0.440598	0.359847
cosine	NaN	1301	distance	1.000000	0.694444	0.484102	0.236194	0.440977	0.360156
cosine	NaN	1313	distance	1.000000	0.694444	0.484171	0.236251	0.441263	0.360390
cosine	NaN	1369	distance	1.000000	0.694444	0.484565	0.236591	0.440065	0.359412
cosine	NaN	1379	distance	1.000000	0.693889	0.484638	0.236656	0.439488	0.358940
cosine	NaN	1329	distance	1.000000	0.693889	0.484257	0.236323	0.441492	0.360577
cosine	NaN	1377	distance	1.000000	0.693889	0.484612	0.236633	0.439666	0.359085

Table 4: KNN (RobustScaler, v2): top configurations by validation accuracy.

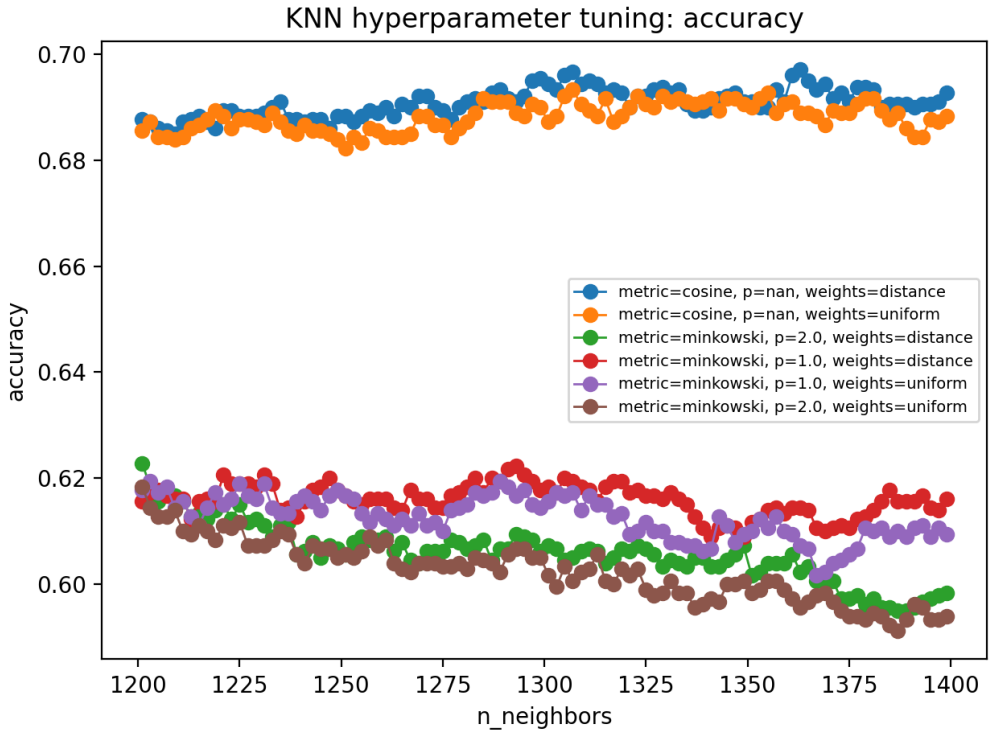

Figure 9: KNN (RobustScaler, v2): validation accuracy across tested configurations.

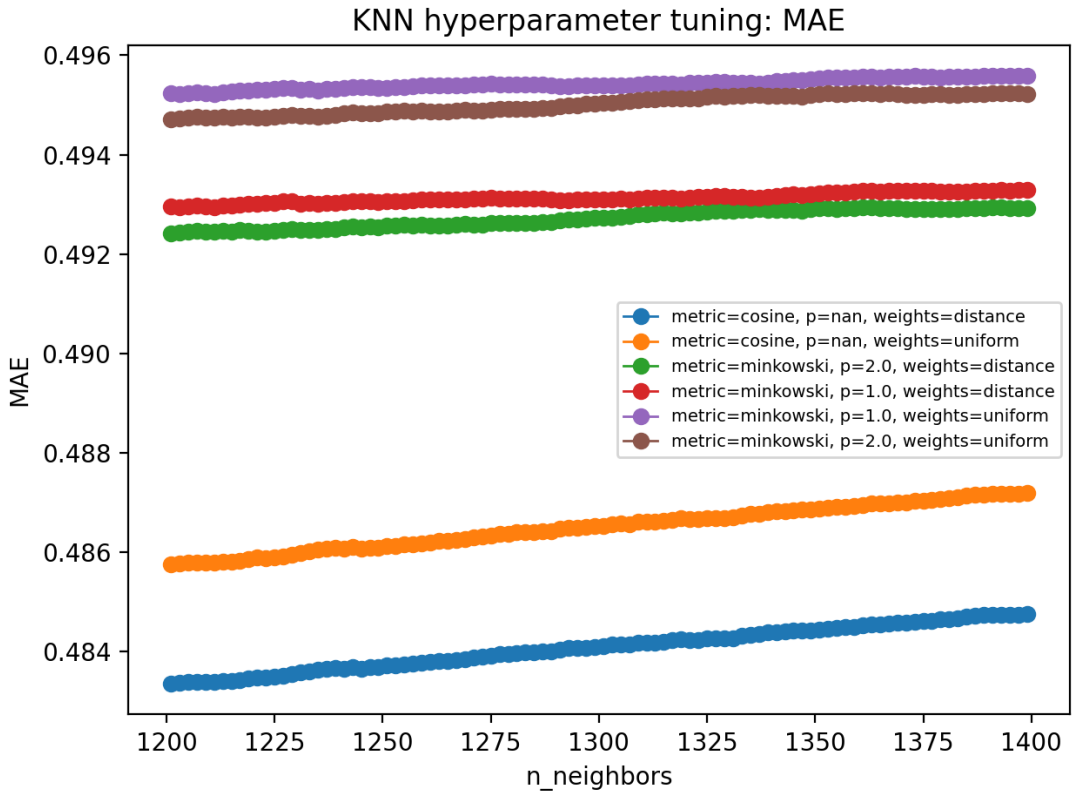

Figure 10: KNN (RobustScaler, v2): MAE across tested configurations.

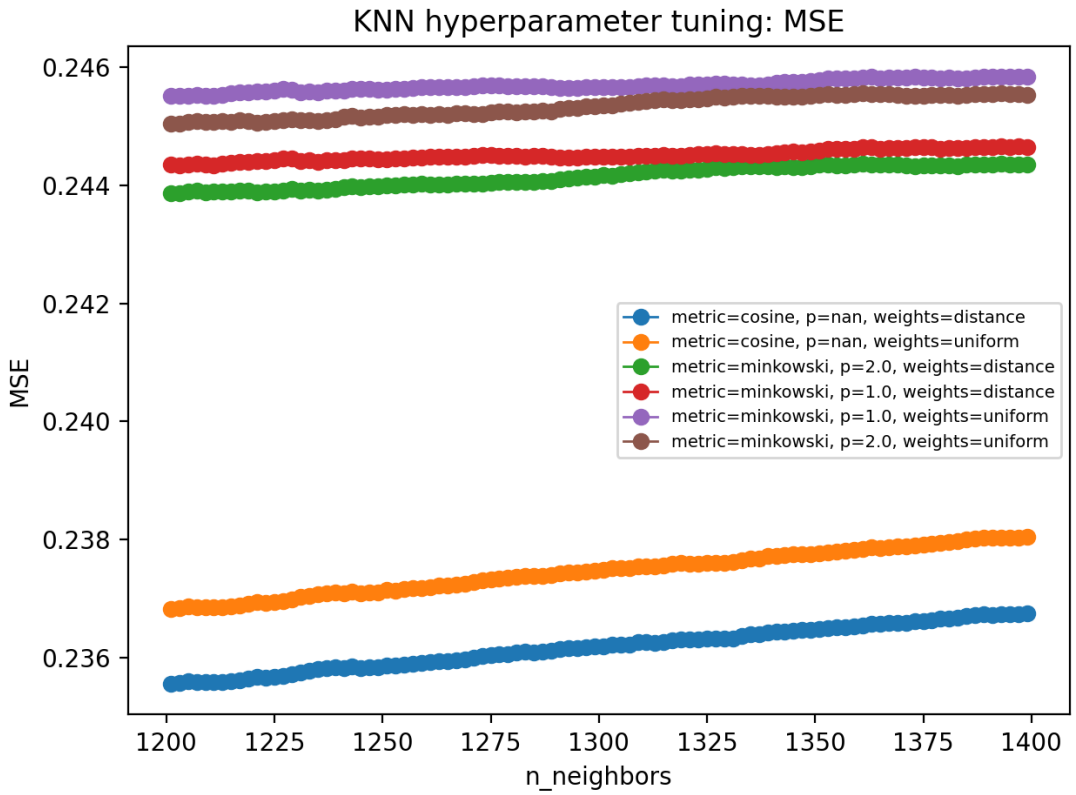

Figure 11: KNN (RobustScaler, v2): MSE across tested configurations.

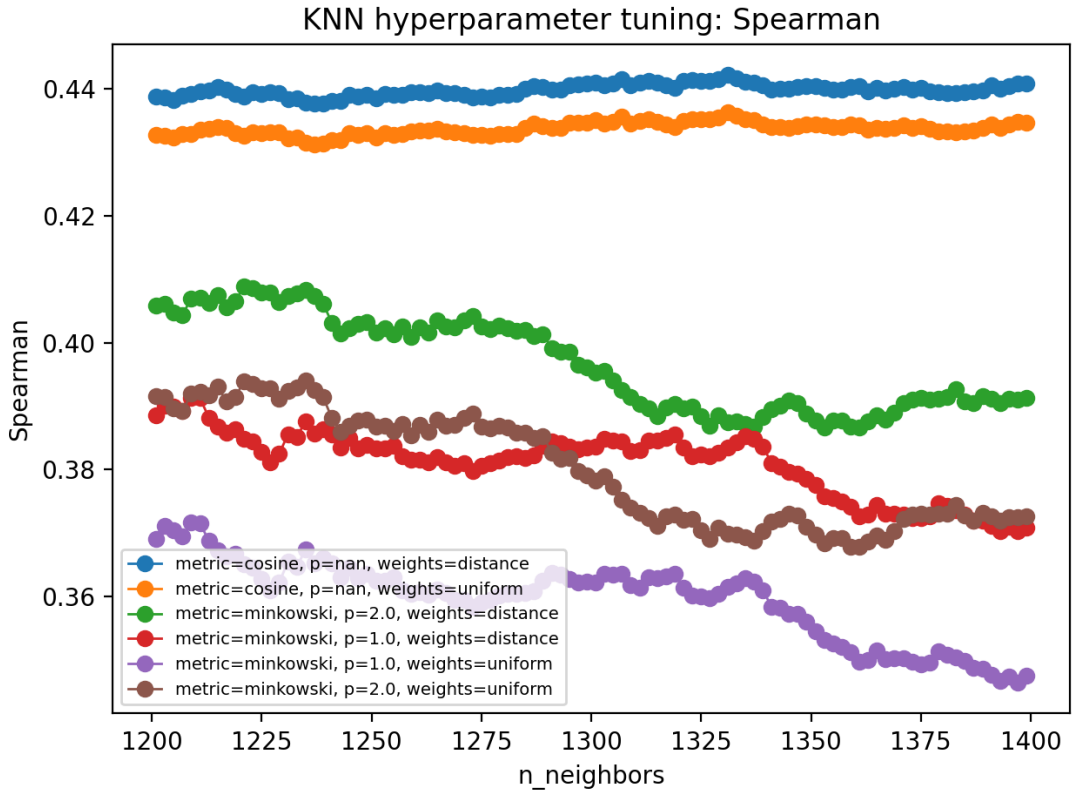

Figure 12: KNN (RobustScaler, v2): Spearman correlation across tested configurations.

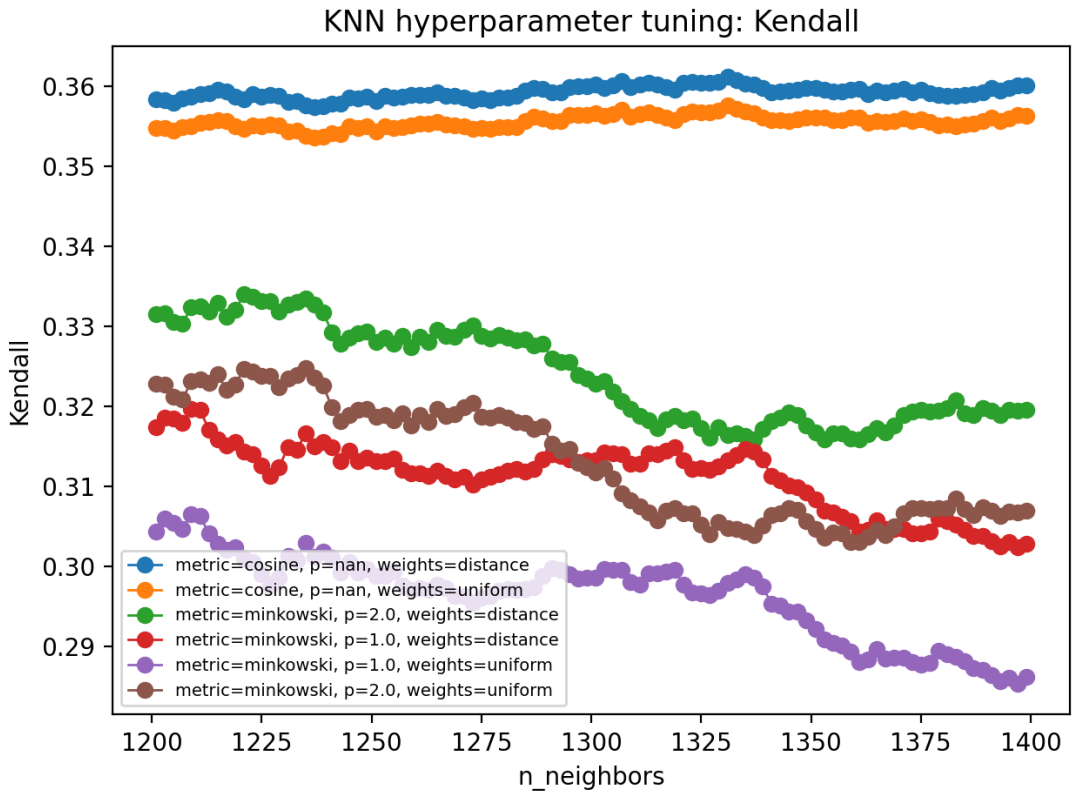

Figure 13: KNN (RobustScaler, v2): Kendall correlation across tested configurations.

7.3 Broad vs refined search (tuning)

The broad searches (v1) document the initial exploration of the hyperparameter space and motivate the need for refined ranges used in v2. The tables below report the metric-wise optimal configurations obtained from the broad search. Each row corresponds to the single hyperparameter configuration that optimizes one validation metric over the entire v1 search space: maximum validation accuracy, minimum MAE, minimum MSE, and maximum ranking correlation (Spearman/Kendall). These configurations don't have to coincide. They just highlight the trade-offs between error-based and ranking-based objectives.

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	401	uniform	0.683778	0.687222	0.481894	0.233720	0.432306	0.356184
cosine	NaN	11	distance	1.000000	0.572222	0.462935	0.251070	0.191797	0.156653
cosine	NaN	201	distance	1.000000	0.623889	0.477538	0.232131	0.380609	0.310854
cosine	NaN	1501	distance	1.000000	0.683889	0.487285	0.238998	0.449489	0.367110

Table 5: KNN (StandardScaler, v1): metric-wise optimal configurations from the broad search (best validation accuracy, best MAE, best MSE, best Spearman/Kendall).

metric	p	neighbors no.	weights	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
cosine	NaN	1301	distance	1.000000	0.694444	0.484102	0.236194	0.440977	0.360156
cosine	NaN	5	distance	1.000000	0.581111	0.450150	0.266874	0.181826	0.149157
cosine	NaN	201	distance	1.000000	0.630000	0.472716	0.229070	0.404069	0.330013
cosine	NaN	801	distance	1.000000	0.691111	0.479774	0.232603	0.447185	0.365227

Table 6: KNN (RobustScaler, v1): metric-wise optimal configurations from the broad search (best validation accuracy, best MAE, best MSE, best Spearman/Kendall).

StandardScaler (v1).

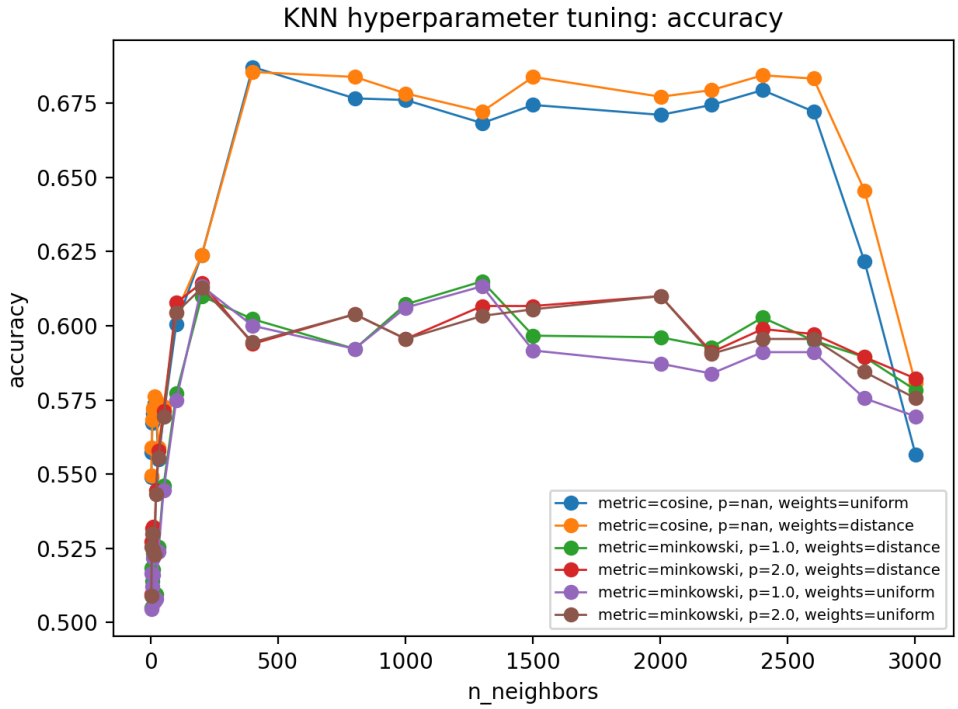

Figure 14: KNN (StandardScaler, v1): validation accuracy across tested configurations.

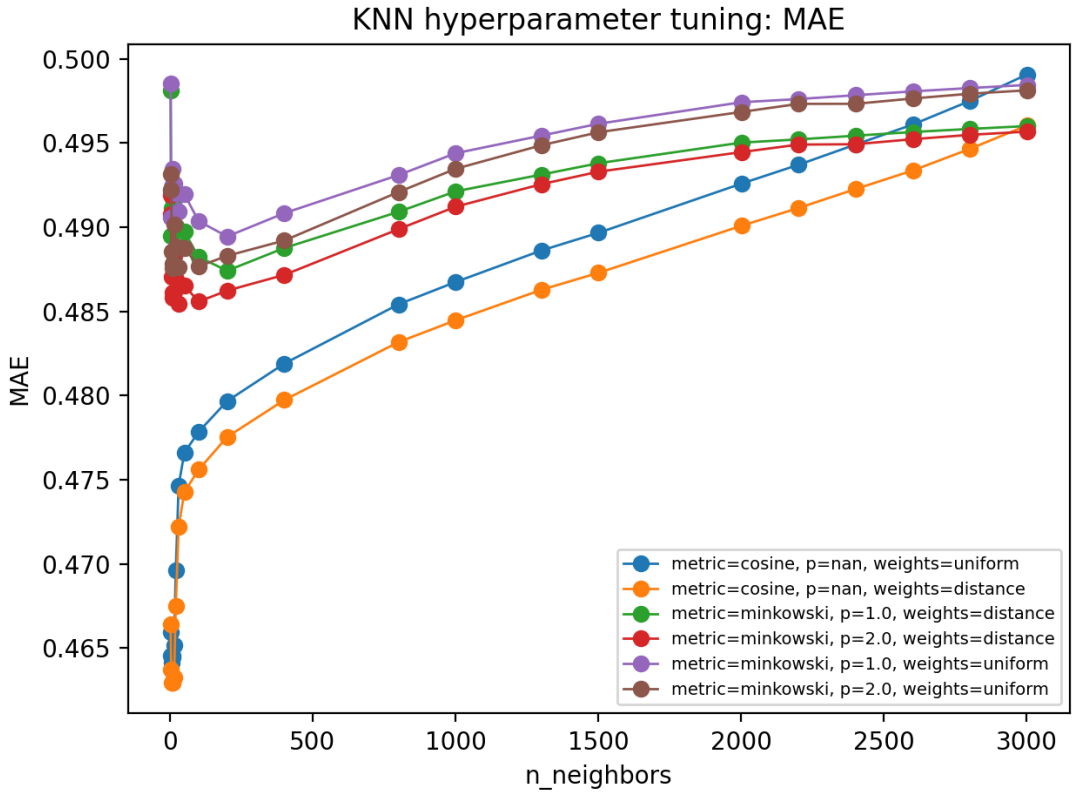

Figure 15: KNN (StandardScaler, v1): MAE across tested configurations.

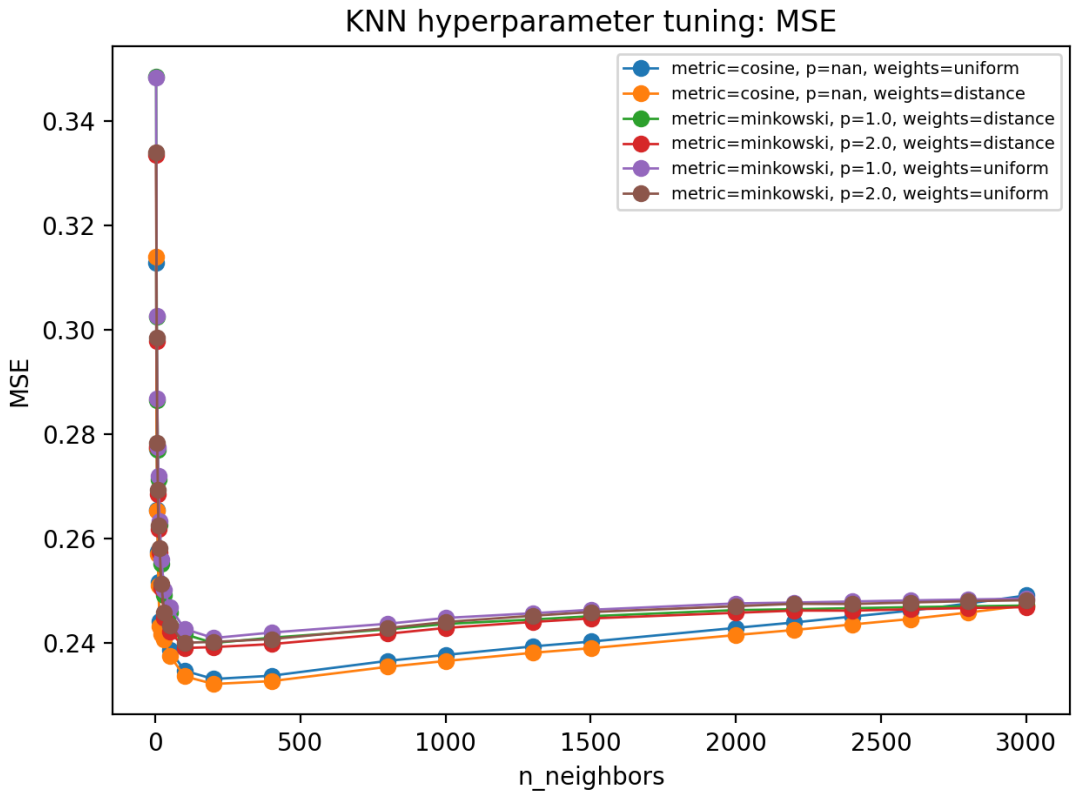

Figure 16: KNN (StandardScaler, v1): MSE across tested configurations.

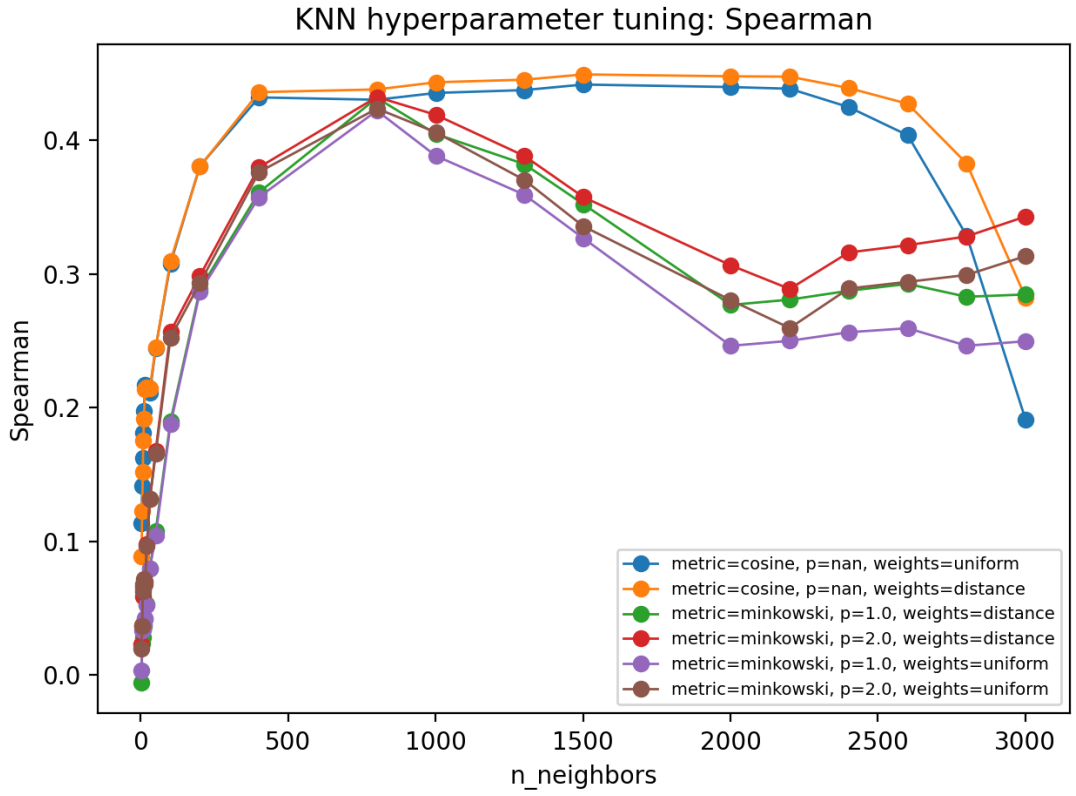

Figure 17: KNN (StandardScaler, v1): Spearman correlation across tested configurations.

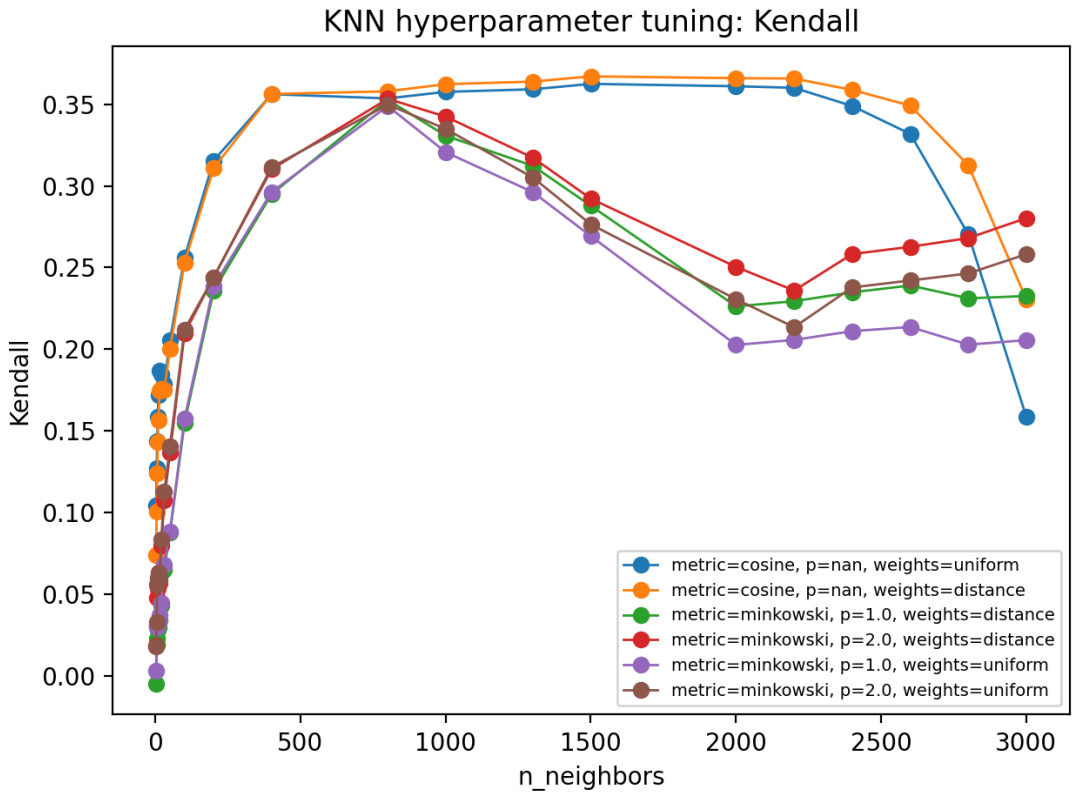

Figure 18: KNN (StandardScaler, v1): Kendall correlation across tested configurations.

RobustScaler (v1).

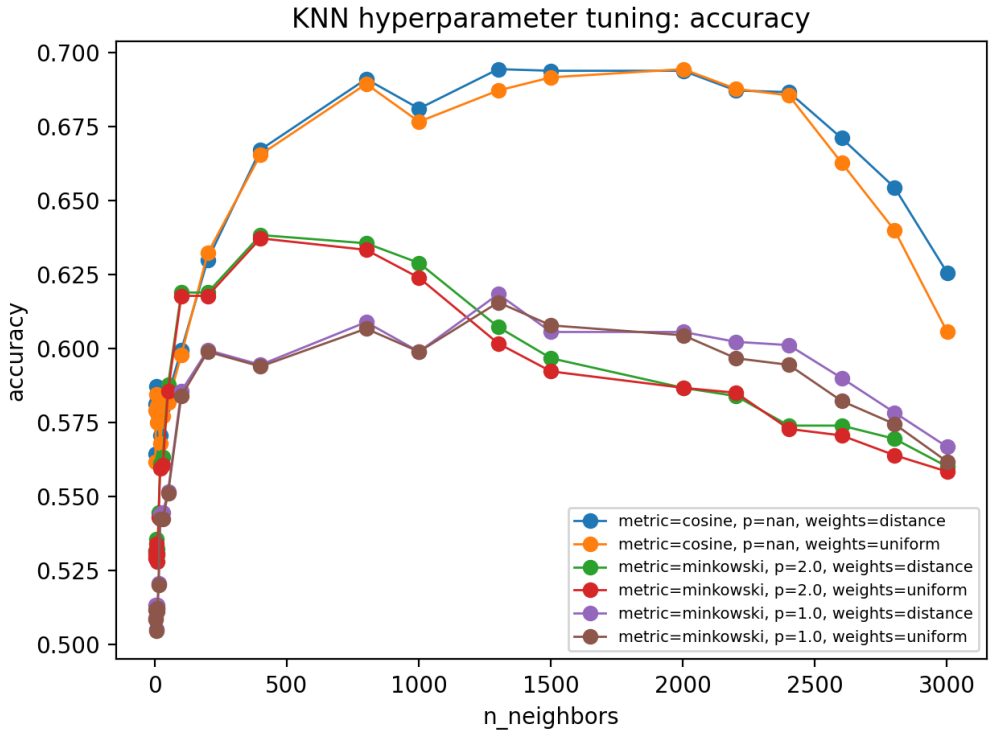

Figure 19: KNN (RobustScaler, v1): validation accuracy across tested configurations.

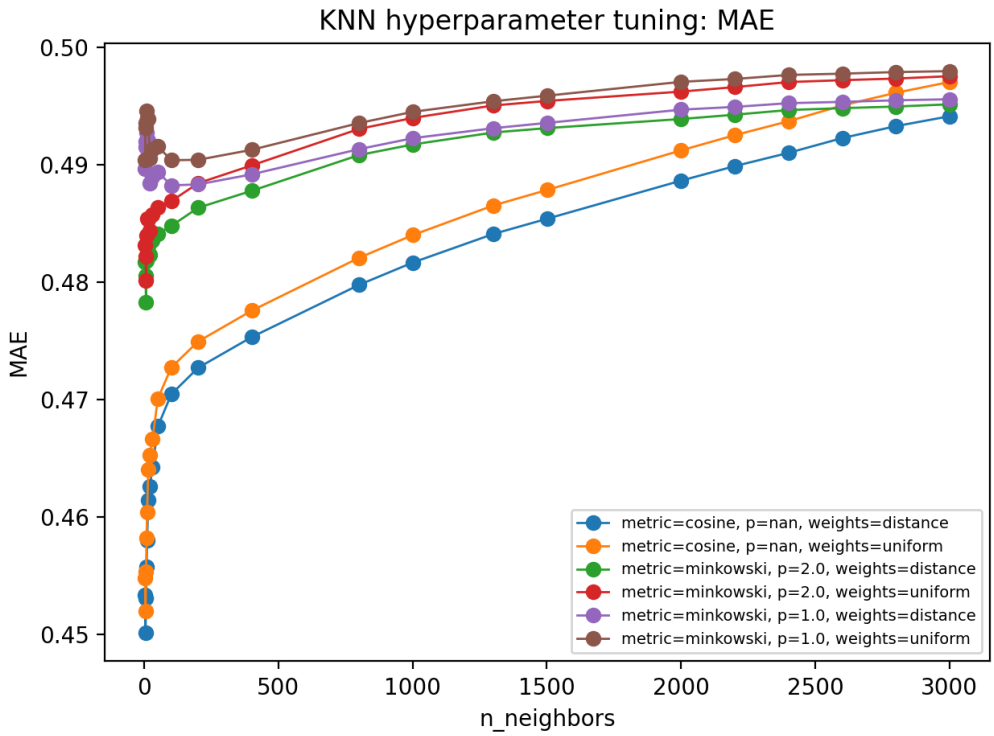

Figure 20: KNN (RobustScaler, v1): MAE across tested configurations.

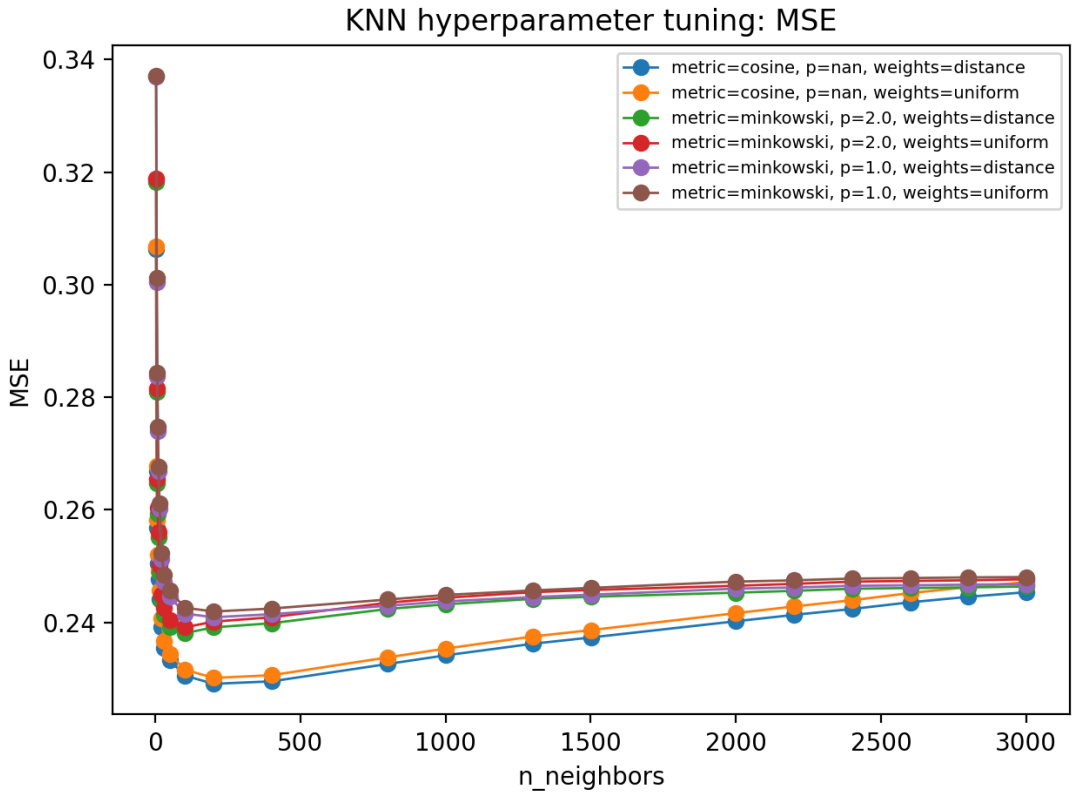

Figure 21: KNN (RobustScaler, v1): MSE across tested configurations.

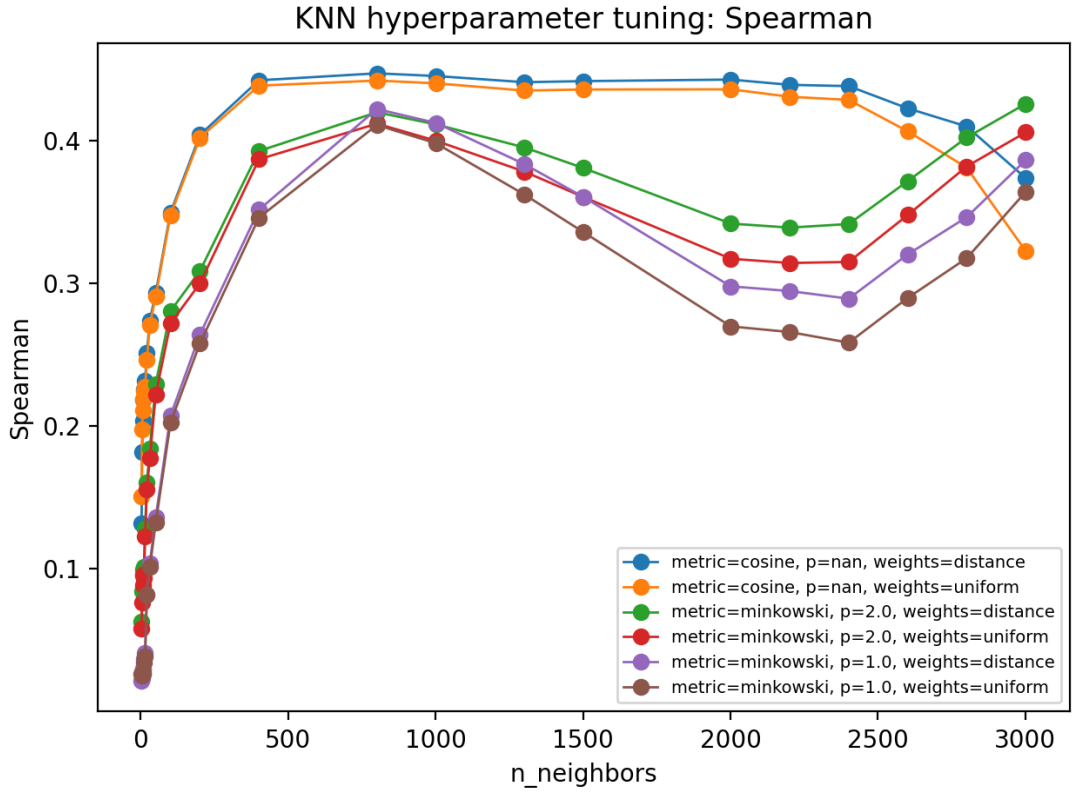

Figure 22: KNN (RobustScaler, v1): Spearman correlation across tested configurations.

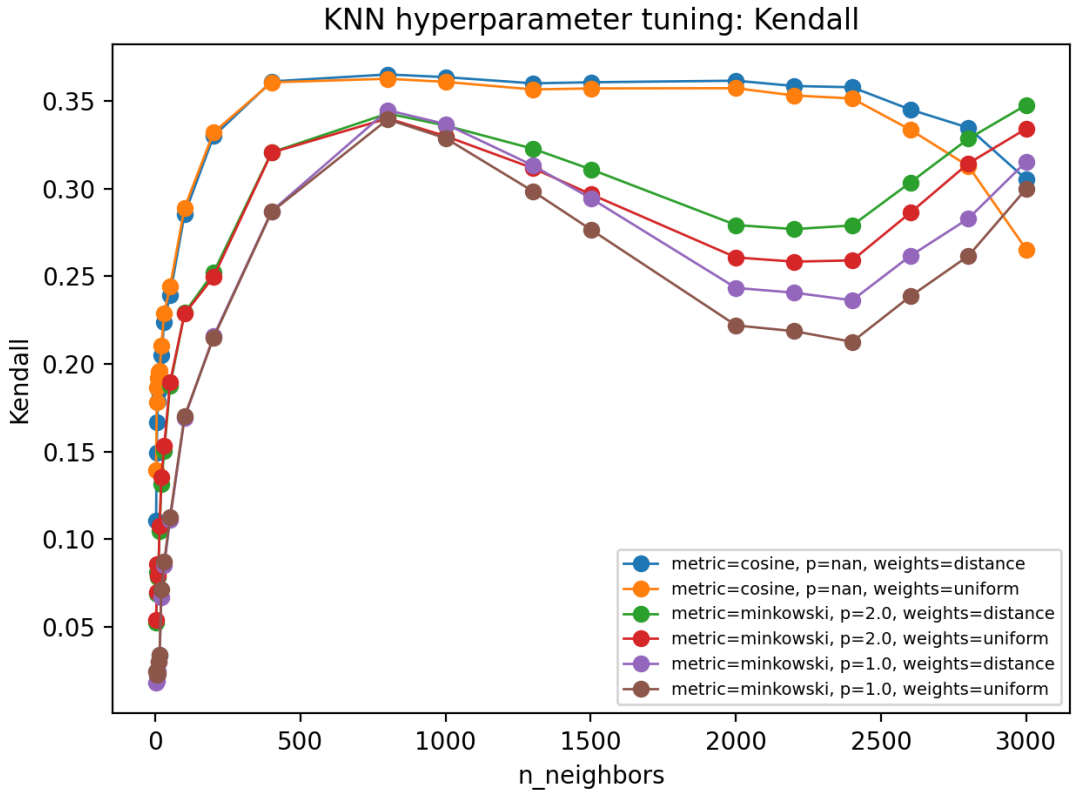

Figure 23: KNN (RobustScaler, v1): Kendall correlation across tested configurations.

8 SVM Experiments and Results

This section reports the validation results obtained with Support Vector Machines (SVM) using the same final feature representation described in Section 4 (mean pooling with 16×16 blocks combined with global statistics). In contrast to KNN, SVM learns a global decision function by maximizing the margin between classes while controlling model complexity through regularization.

All SVM experiments use the radial basis function (RBF) kernel. No feature augmentation or additional preprocessing is applied beyond the normalization already described in Section 2. The feature vectors are identical to those used for KNN, ensuring that differences in performance are attributable to the learning model rather than the representation.

8.1 Hyperparameter search and model selection

A single grid search is being done over the SVM hyperparameter space, exploring combinations of the regularization parameter C and the kernel width γ. The same grid and evaluation protocol are used for all SVM runs. Validation performance is assessed using the required metrics: `validation accuracy`, `MAE`, `MSE`, `Spearman`, and `Kendall`.

From this grid search, two representative SVM configurations are retained for analysis:

- **SVM (best configuration)**: $C = 0.25$, $\gamma = 0.0001$, corresponding to the configuration that achieves the strongest overall validation performance across the reported metrics.
- **SVM (alternative configuration)**: $C = 0.25$, $\gamma = 0.0003$, corresponding to the third-ranked configuration from the same grid search.

Both models use the same kernel, feature representation, and training protocol. The only difference is the value of γ, which controls the locality of the RBF kernel.

8.2 Best configurations per-metric

This table summarizes the hyperparameter configurations obtained from the grid search that optimize each validation metric individually. Each row corresponds to the single configuration that maximizes or minimizes the given metric over the entire search space.

C	gamma	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
0.250000	0.0001	0.713393	0.705000	0.400863	0.195747	0.475455	0.388351
0.150000	0.0003	0.721783	0.701111	0.379683	0.190017	0.485396	0.396468
0.200000	0.0003	0.725403	0.701667	0.380233	0.189983	0.486166	0.397089

Table 7: SVM (RBF kernel): metric-wise optimal configurations obtained from the grid search (best validation accuracy, best MAE/MSE, best Spearman/Kendall).

8.3 Top configurations by validation accuracy

The table below lists the top configurations ranked by validation accuracy. This view is useful in assessing the stability of performance across close hyperparameter values and in identifying whether the best-performing model is an isolated optimum or part of a broader high-performing region.

C	gamma	train accuracy	validation accuracy	MAE	MSE	Spearman	Kendall
0.250000	0.0001	0.713393	0.705000	0.400863	0.195747	0.475455	0.388351
0.150000	0.001	0.739552	0.703333	0.384459	0.191340	0.482292	0.393922
0.250000	0.0003	0.730010	0.702778	0.380885	0.190178	0.485627	0.396657
0.300000	0.0001	0.721290	0.702778	0.386954	0.191657	0.480090	0.392119
0.200000	0.0003	0.725403	0.701667	0.380233	0.189983	0.486166	0.397089
0.150000	0.0003	0.721783	0.701111	0.379683	0.190017	0.485396	0.396468
0.350000	0.0003	0.734781	0.700556	0.381517	0.190291	0.485311	0.396404
0.600000	0.0003	0.743008	0.700556	0.383589	0.190810	0.483352	0.394793
0.350000	0.0001	0.718987	0.699444	0.381989	0.191156	0.481016	0.392906
0.650000	0.0001	0.724745	0.699444	0.382026	0.191184	0.481570	0.393332
0.400000	0.0001	0.719645	0.699444	0.380843	0.191043	0.481457	0.393310
0.500000	0.0003	0.740375	0.699444	0.383027	0.190580	0.484404	0.395657
0.300000	0.0003	0.731820	0.698889	0.381286	0.190530	0.484224	0.395523
0.700000	0.0001	0.726719	0.698889	0.382287	0.191309	0.481490	0.393259
0.550000	0.0001	0.723593	0.698889	0.381549	0.191059	0.482423	0.394037

Table 8: SVM (RBF kernel): top configurations ranked by validation accuracy.

8.4 Grid search

To show the hyperparameter tuning, the following figures visualize how the validation metrics vary across the (C, γ) search space. Each figure corresponds to one evaluation metric and is derived from the same grid search results reported in the tables above.

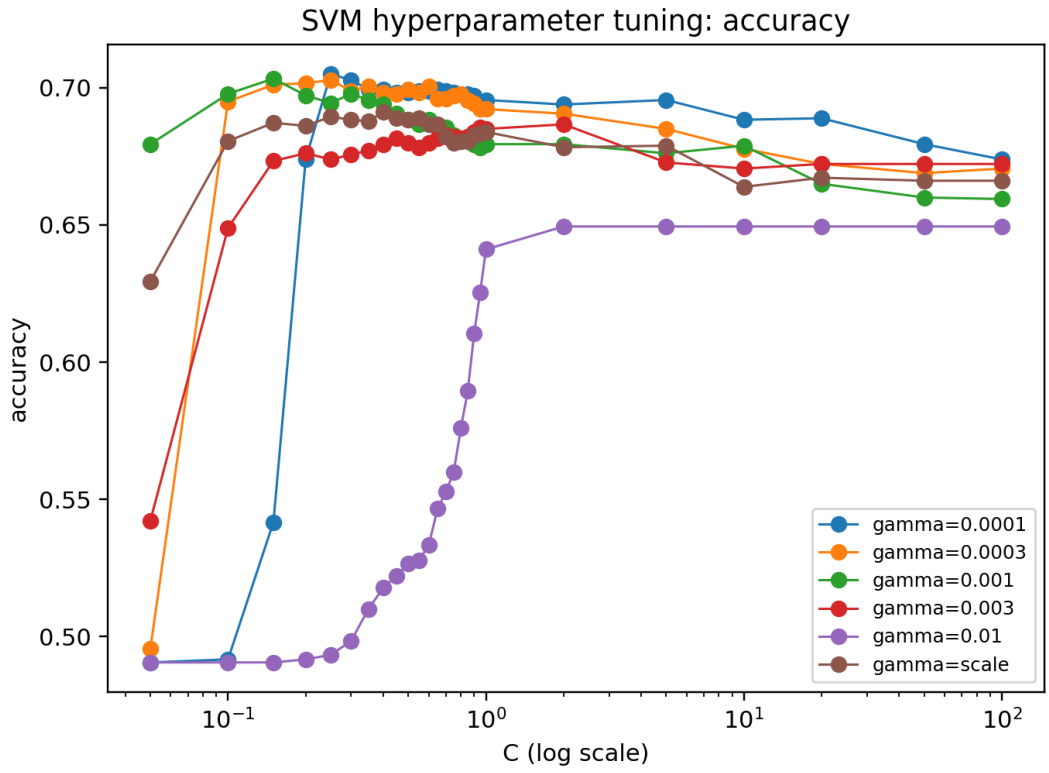

Figure 24: SVM: validation accuracy across the (C, γ) grid.

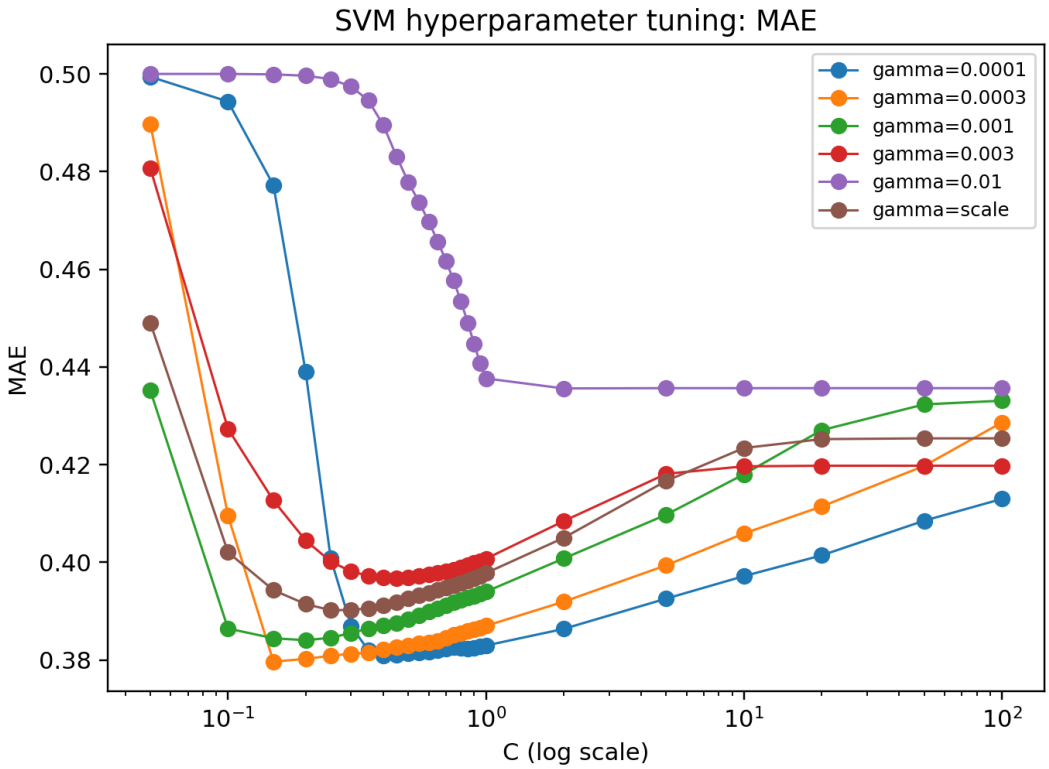

Figure 25: SVM: MAE across the (C, γ) grid.

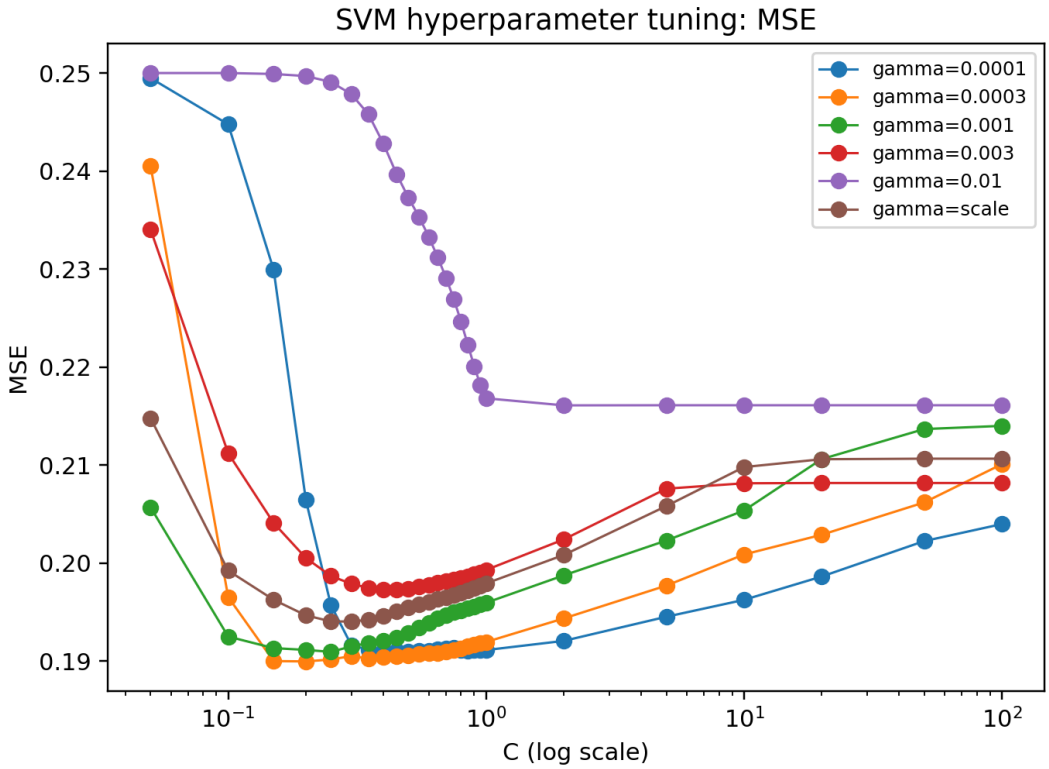

Figure 26: SVM: MSE across the (C, γ) grid.

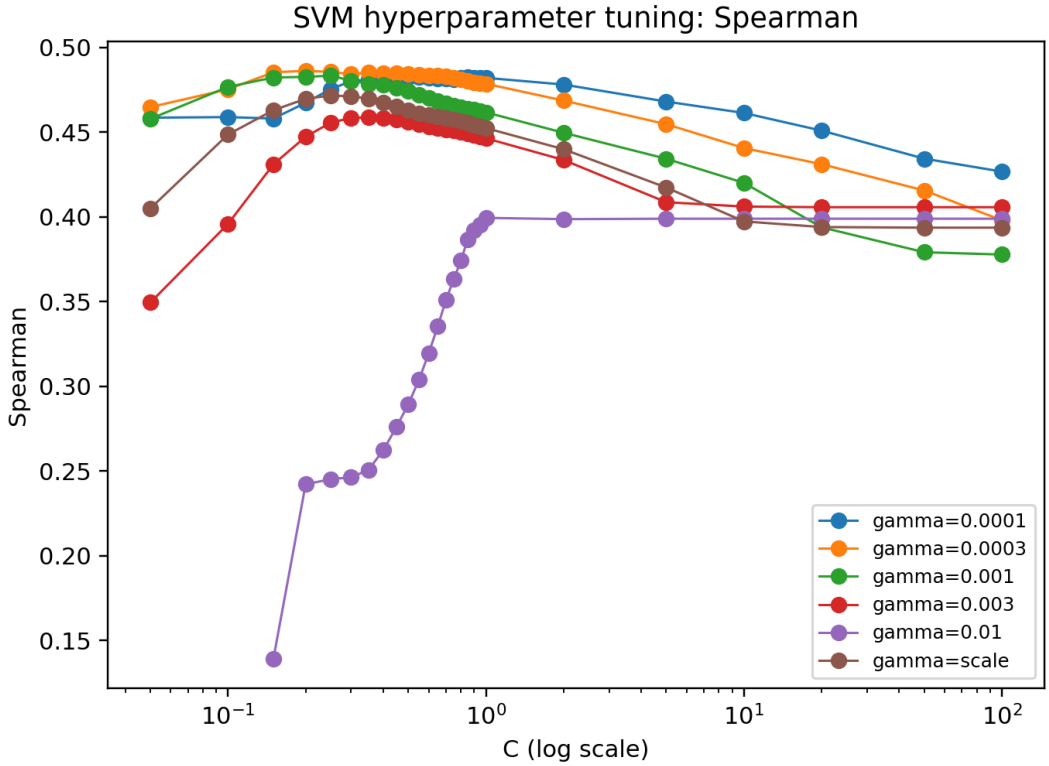

Figure 27: SVM: Spearman correlation across the (C, γ) grid.

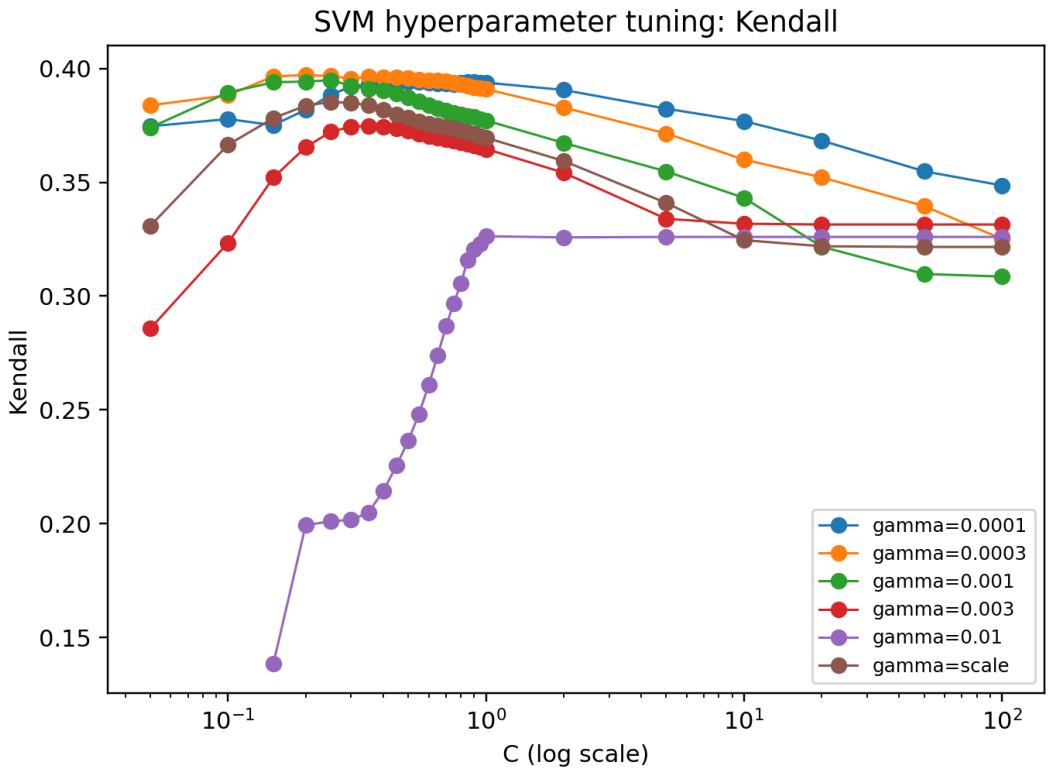

Figure 28: SVM: Kendall correlation across the (C, γ) grid.

8.5 Comparison of SVM configurations

Comparing the two SVM configurations shows that reducing γ leads to improved validation performance across metrics. This behavior implies a smoother decision boundary that generalizes better in a high-dimensional feature space. The configuration with $\gamma = 0.0001$ outperforms the alternative configuration with $\gamma = 0.0003$ on the validation set and is therefore selected as the final SVM model for the comparative analysis.

9 Comparisons and Conclusion

The comparison shows a different kind of behaviors on the validation set and on the fully evaluated Kaggle test set. On the validation data, the SVM with RBF kernel has the highest accuracy and the strongest Spearman and Kendall correlations, basically indicating better sample ordering and more consistent ranking performance. In this setting, KNN shows a slightly lower accuracy, but competitive MAE and MSE values and stable behavior across many neighborhood sizes.

However, as it was showed on the Kaggle test set, for the private score, the situation changes. Both of the KNN variants generalize better than SVM, with the KNN + RobustScaler configuration achieving the highest public score, followed by KNN + StandardScaler. The SVM configurations obtain slightly lower scores, despite their stronger validation performance. This obviously suggests that the distance-based KNN model, when combined with robust feature scaling, is, after all, less sensitive to validation-specific patterns and adapts better to unseen data.

A clear trade-off is, therefore, observed. SVM performs better on the validation set and provides stronger ranking consistency, while KNN generalizes better on the private Kaggle test data. This difference indicates mild overfitting in the SVM models and highlights the robustness of KNN when operating on the proposed feature representation. For this reason, KNN shows a stronger generalization, while the SVM remains the most balanced model on validation metrics.