SKiP: SVM weighted by K-Nearest-Neighbors and class Probability for weakening outliers *

UNIST CSE36301 Project Report / Team 4

Jeonghoon Park KangJun Lee Jaemin Kim Doyeol Oh
Department of Computer Science & Engineering, UNIST
{hoonably, suri7897, jm611, ohdoyoel}@unist.ac.kr

1 Introduction

Outliers are a major concern in machine learning, as they can significantly degrade model performance. They can be broadly divided into two types: (1) well-labeled samples that lie far from the main data distribution (Feature outliers), and (2) mis-labeled samples caused by incorrect or incomplete labeling (Label outliers). Both types can distort the decision boundary, especially in Support Vector Machines (SVM) [1], which treat all samples equally in the loss term.

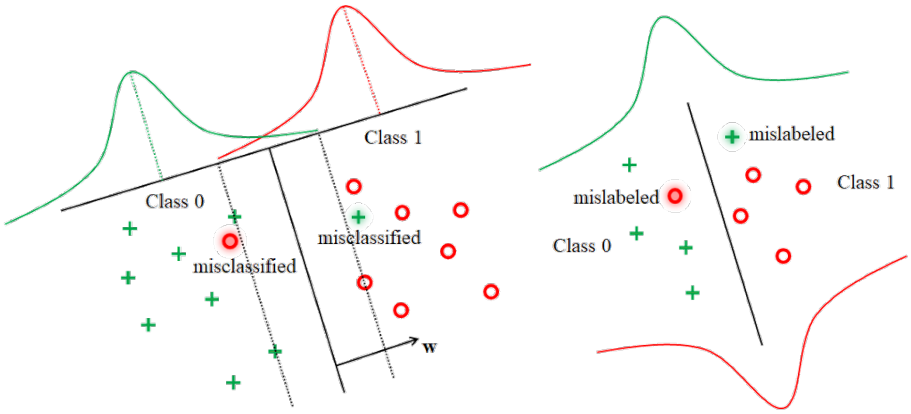


Figure 1: Two types of outliers: (Left) a correctly labeled sample far from the main distribution (Feature outliers), and (Right) a mislabeled sample near the decision boundary (Label outliers).

Traditional soft-SVM minimizes both the margin width and total hinge loss,

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{1}$$

subject to $y_i(w^\top x_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0,$

but the penalty term $C\sum_i \xi_i$ does not differentiate between highly reliable samples and noise samples. Consequently, even a small number of outliers can significantly distort the separating hyperplane, as illustrated in Fig. 1, where both distant but correctly labeled samples and mislabeled boundary samples affect the decision boundary in undesirable ways.

To make SVM more robust against these two types of errors, we introduce SKiP, SVM weighted by K-Nearest-Neighbors and class Probability for weakening outliers. The core idea is to incorporate

*Codes are available on `https://github.com/hoonably/SKiP`

two complementary reliability measures into the slack penalty: a class-conditional probability p_i describing how well a sample aligns with the global distribution of its class, and a KNN-based label consistency measure n_i capturing how well its label agrees with that of its local neighbors. Whereas p_i is effective for identifying Feature outliers that lie far from the distribution, n_i provides sensitivity to labeling errors that occur near the margin.

Instead of multiplying these two terms, we adopt a stable additive design and define the sample-dependent penalty coefficient as

$$C_i = C \cdot \frac{p_i + n_i}{2}.$$

This formulation encourages the SVM to assign lower influence to samples that are globally unlikely or locally inconsistent, thereby jointly addressing both distributional outliers and mislabeled boundary points. By combining global probability information with local label consistency in a balanced manner, SKiP offers a robust and unified extension of soft-SVM capable of mitigating both label and feature outliers effectively.

2 Related Work

There have been several attempts to adapt the constant C in SVMs by assigning sample-dependent weights to the slack penalties. One representative line of work is the Fuzzy SVM (FSVM) [2] by Lin Wang, where each sample is assigned a fuzzy membership that reflects its reliability. While FSVM reduces the influence of noisy samples, its membership values are typically determined heuristically or through global distance-based criteria, and thus it does not explicitly incorporate local label consistency.

For another approach, Platt introduced the probability-weighted SVM [3], which modifies the slack coefficient using class-conditional probabilities typically obtained through sigmoid-based probability calibration (e.g., Platt scaling). Such probability estimates, however, rely solely on global probability calibration and do not incorporate information about local label reliability, making them potentially sensitive to miscalibrated outputs.

Cost-sensitive SVM [4] is also introduced by Masnadi-Shirazi et al.. This line of work reformulates the SVM objective to incorporate class-dependent misclassification costs, enabling the classifier to adjust the decision boundary according to asymmetric error penalties. Such methods are particularly useful in imbalanced or risk-sensitive settings where one type of error is more costly than another. However, cost-sensitive SVMs operate at the class level and do not account for instance-wise reliability. As a result, they are unable to adapt to sample-specific noise or label uncertainty, limiting their effectiveness in scenarios where reliability varies within the same class.

For the KNN-based SVM, Zhang et al. proposed KNN-SVM [5], which integrates local neighborhood information into the SVM decision function. Their method first applies KNN to identify a sample's local neighborhood and then uses this local structure to adjust the classification decision, effectively leveraging local class distributions to refine the SVM boundary. The core idea is to combine the global margin-maximizing property of SVM with locally adaptive decision rules derived from KNN. However, the method primarily focuses on modifying the decision rule rather than incorporating neighborhood reliability into the optimization process itself, and thus does not provide instance-specific weighting within the SVM loss formulation.

3 Method

Real-world datasets commonly suffer from two types of corruption: *feature outliers*, where samples lie far from the class distribution, and *label outliers*, where samples are assigned incorrect labels. Each type of noise affects the SVM optimization in different ways. To handle these outliers, we proposed three SVM models. The probability-weighted SVM (Prob-SVM) addresses feature outliers by introducing probabilistic term, p_i . However, Prob-SVM remains vulnerable to mislabeled samples because it does not consider local label consistency. Conversely, the KNN-based SVM (KNN-SVM) reduces the influence of label outliers by assigning knn-based term, n_i. but it does not account for feature-level deviations from the class distribution.

To jointly handle both types of noise, we introduce SKiP, which combines probabilistic reliability with KNN-based local consistency, leveraging the complementary strengths of both approaches.

3.1 Prob-SVM

The Naive-SVM minimizes Eq. (1), where all samples are penalized equally, making the model sensitive to outliers. To address this, we introduce a class-conditional probability weight $p_i = p(y_i)p(x_i \mid y_i)$, which scales each sample's penalty by its posterior likelihood. The modified objective function is:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + \sum_i C\, p_i\, \xi_i. \tag{2}$$

Here, inspired by LDA, the class-conditional likelihood $p(x_i \mid y_i)$ is estimated under a Gaussian assumption.

$$p(y_i|x_i) \sim \mathcal{N}(x_i \mid \mu_{y_i}, \Sigma_{y_i}).$$

However, for categorical features, outliers are unlikely to occur because their values are restricted to a fixed set of categories. Therefore, we set $p_{ij} = 1$ for categorical feature. This setting was applied to the Titanic dataset.

3.2 KNN-SVM

Although probability-weighted naive SVM improves robustness against feature outliers, it still relies solely on the reliability estimated from the probabilistic model. To additionally handle label outliers, we consider a simple KNN-based variant, **KNN-SVM**, in which the instance-specific penalty is determined by the KNN consistency term n_i. The optimization problem becomes

$$\min_{w,b} \frac{1}{2}\|w\|^2 + \sum_i C\, n_i\, \xi_i, \tag{3}$$

subject to

$$y_i(w^\top x_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0.$$

here,

$$n_i = \frac{1}{K} \sum_{x_k \in \mathcal{N}_K(x_i)} \mathbb{I}(y_k = y_i),$$

where $\mathcal{N}_K(x_i)$ denotes the K nearest neighbors of x_i, and $\mathbb{I}(\cdot)$ is the indicator function. Here, $\mathcal{N}_K(x_i)$ denotes the K nearest neighbors of x_i, and $\mathbb{I}(\cdot)$ is the indicator function. Samples surrounded by inconsistent labels receive lower weights, making the classifier less sensitive to local label outliers.

3.3 SKiP

To address both feature outliers and label outliers, we propose SKiP, a model that integrates a KNN-based local consistency term with a probability-based reliability term.

$$\min_{w,b} \frac{1}{2}\|w\|^2 + \sum_i C\, (\frac{p_i + n_i}{2})\xi_i, \tag{4}$$

subject to $y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$

where p_i represents the class-conditional probability weight, and n_i is the KNN-based label consistency term.

The motivation of adopting an additive combination of p_i and rather than a multiplicative form is to enhance robustness against unreliable reliability estimates. Since p_i and n_i each capture different aspects of samples, an inaccurate assumption in either term can adversely affect the final weight. If the two were multiplied, such errors would be amplified and could lead to excessive down-weighting of samples. By using an additive formulation, the two measures can complement each other, preventing performance degradation caused by noise in only one of them and yielding a more stable and reliable weighting scheme.

To further support this design choice, we provide additional experiments in the **Appendix C** that compare alternative strategies for combining p_i and n_i. Specifically, we examine both additive and multiplicative formulations. Also, we examined rescaled p_i and n_i using min–max normalization to the range $[0, 1]$ before aggregation.

4 Experiments

4.1 Datasets

In this experiment, we use three base datasets: the Wine [6], Iris [7], and Titanic datasets [8]. The motivation for selecting these datasets is as follows: two of them (Wine, and Iris) exhibit moderately Gaussian-like feature distributions, whereas the Titanic dataset does not follow a Gaussian distribution and additionally contains categorical features.

In addition, computing the probabilistic term in our model requires estimating class means and variances for all features, which incurs a computational cost proportional to the number of features. Since the training complexity of a linear SVM scales as $O(nd)$, where n is the number of samples and d is the feature dimension, reducing the dimensionality is beneficial for both probability estimation and SVM optimization. Therefore, to obtain a consistent and efficient low-dimensional representation across datasets, we apply Principal Component Analysis (PCA) and retain components that explain 95% of the total variance. With these preprocessing steps and dataset configurations, we obtain a total of eight datasets used in our experiments.

4.2 Noise Injection

To systematically define noise within the data, we utilized the Mahalanobis distance relative to the normal distribution of each cluster. Specifically, we investigated the robustness of the SVM by introducing noise where $k\%$ of the original dataset is created to cross the decision boundary. We employed an arbitrary classifier, denoted as the `base_model`, to establish this decision boundary.

Consequently, we defined **Feature outliers** as data points that satisfy two simultaneous conditions:

1. The point's Mahalanobis distance exceeds the 99^{th} $(= \tau)$ percentile of the cluster's distribution.
2. The point crosses the decision boundary of the `base_model`.

This definition ensures that the injected noise represents statistically significant outliers that also disrupt the classification boundary.

Initially, we attempted to generate outliers using a naive rejection sampling strategy, where candidate points were iteratively generated and retained only if they exceeded the distance threshold. However, this approach proved computationally prohibitive due to the low acceptance probability of valid noise samples.

To mitigate this high computational cost, we introduced a **Cholesky-decomposed outlier injection** method. Unlike the iterative approach, this method directly constructs samples satisfying the distributional constraints, resulting in an approximate 100 times $(= \frac{1}{1-\tau})$ speedup. Detailed implementation and mathematical derivations are provided in **Appendix A**.

For **Label outliers**, we aimed to simulate label contamination within the dataset. To generate this noise, we randomly sampled $k\%$ subset of existing data points and flipped their class labels. Subsequently, we applied slight perturbations to these feature vectors to introduce marginal spatial deviations from the original instances.

4.3 Experiments

To evaluate the effectiveness of the proposed method, we conduct comparative experiments using four SVM variants: (1) the baseline naive SVM, (2) the probability-weighted SVM, (3) the KNN-weighted SVM, and (4) the proposed SKiP model, which incorporates both probabilistic and KNN-based weighting schemes.

We conducted experiments on three datasets (Iris, Wine, Titanic) with PCA, injecting noise levels ranging from 0% to 20% into both the labels and the feature space to simulate label corruption and feature outliers. Model performance was assessed by comparing the classification accuracy under these varying noise conditions. For all models, the regularization parameter C was selected from $\{0.1, 1, 10, 100, 1000, 10000\}$. For k-based methods such as KNN-SVM and SKiP-SVM, we additionally tuned $k \in \{3, 5, 7, 10, 15\}$. Among all configurations, we report the performance achieved with the best-performing hyperparameter setting.

All experiments were conducted on a single server equipped with 2 $\times$ AMD EPYC 9354 CPUs (32 cores / 64 threads each; 64 cores / 128 threads in total). With parallel computing enabled, the full experimental pipeline required approximately two hours to complete.

4.4 SVM Evaluation at Gaussian distribution (Iris, Wine Dataset)

Table 1: Classification accuracy under label and feature outliers on Iris and Wine datasets.

Dataset	Outlier (%)		Accuracy (%)			
	Label	Feature	Naive-SVM	Prob-SVM	KNN-SVM	SKiP
Iris	0	0	86.7	86.7	90.0	**93.3**
		10	**87.9**	75.8	81.8	84.8
		20	80.6	86.1	**91.7**	88.9
	10	0	80.0	80.0	**83.3**	**83.3**
		10	78.8	69.7	81.8	**87.9**
		20	75.0	66.7	81.8	**83.3**
	20	0	63.3	**70.0**	**70.0**	**70.0**
		10	69.7	66.7	72.7	**75.0**
		20	63.9	66.7	72.7	**75.0**
Wine	0	0	**100.0**	**100.0**	**100.0**	**100.0**
		10	**97.4**	89.7	94.9	**97.4**
		20	**93.0**	88.4	90.7	**93.0**
	10	0	83.3	83.3	**86.1**	83.3
		10	92.3	84.6	86.0	**94.9**
		20	**93.0**	86.0	89.7	**93.0**
	20	0	83.3	80.6	**86.1**	**86.1**
		10	**79.5**	70.5	74.4	**79.5**
		20	**74.4**	72.1	72.1	**74.4**

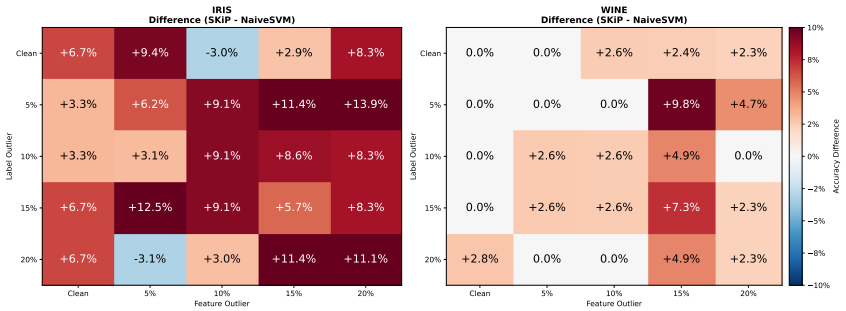


Figure 2: Accuracy difference (SKiP − Naive-SVM) under varying label and feature outlier levels on the *Iris* (left) and *Wine* (right) datasets.

As shown in Table 1, for both the Iris and Wine datasets, our SKiP model achieves the best performance across nearly all outlier configurations. Furthermore, as illustrated in Fig. 2, the SKiP model consistently outperforms the naive SVM across almost all batches.

In contrast to the naive SVM, both KNN-SVM and the SKiP variants exhibit a much smaller degradation in accuracy as noise levels increase. Although Prob-SVM often shows competitive performance, it suffers from sharp drops at certain noise combinations. This vulnerability, however, is largely dismissed in SKiP, since our formulation uses an additive weighting scheme rather than the multiplicative form used in Prob-SVM. As a result, SKiP is less sensitive to the cases where Prob-SVM experiences substantial performance loss.

Furthermore, in several regions—for example, the $(10\%, 10\%)$ noise setting in the Iris dataset—both KNN-SVM and Prob-SVM show noticeable degradation, yet SKiP still achieves comparatively higher accuracy. This improvement arises from the fact that SKiP integrates the two complementary weights (local decision reliability and instance similarity) through addition, allowing it to retain robustness even when one of the individual components becomes unreliable. In such cases, the two weighting terms effectively compensate for each other, producing a synergistic effect that leads to more stable and resilient predictions under noise.

4.5 SVM Evaluation at Non-Gaussian distribution (Titanic Dataset)

Table 2: Classification accuracy under label and feature outliers on Titanic datasets.

Dataset	Outlier (%)		Accuracy (%)			
	Label	Feature	Naive-SVM	Prob-SVM	KNN-SVM	SKiP
Titanic	0	0	77.7	77.7	79.3	**79.9**
		10	74.0	73.5	**74.5**	**74.5**
		20	72.9	72.0	72.9	**73.4**
	10	0	72.1	72.1	72.6	**73.2**
		10	**68.4**	63.8	**68.4**	**68.4**
		20	59.8	57.5	**67.8**	**67.8**
	20	0	65.9	65.9	**69.3**	68.2
		10	63.8	60.7	**65.8**	64.8
		20	55.1	55.1	**66.4**	**66.4**

Interestingly, as shown in Table 2, despite the fact that the Titanic dataset deviates substantially from a Gaussian distribution, SKiP still exhibits stable and competitive performance. In contrast, Prob-SVM—whose probabilistic weighting scheme implicitly relies on Gaussian-like assumptions—shows notable degradation on this non-Gaussian dataset, particularly due to the presence of categorical features. One might therefore expect SKiP to inherit similar limitations, given that it incorporates the probabilistic reliability term from Prob-SVM. However, SKiP's formulation is further reinforced by the KNN-based similarity component, which performs reliably on this dataset.

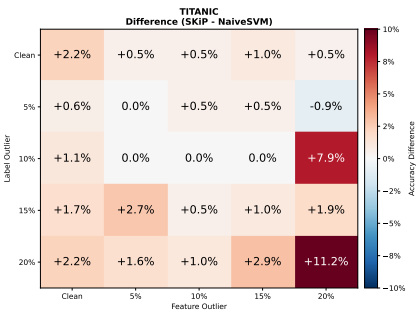


Figure 3: Accuracy difference (SKiP − Naive-SVM) under varying label and feature outlier levels on the *titanic*.

The robustness of SKiP becomes even more evident when examining the accuracy differences in Fig. 3. Across a wide range of label–feature noise combinations, SKiP consistently outperforms the naive SVM, often by a substantial margin—particularly in high-noise scenarios such as the $(20\%, 20\%)$ setting. These improvements demonstrate that the two weighting mechanisms in SKiP operate in a complementary manner, enabling the model to maintain strong predictive accuracy despite significant distributional mismatch.

4.6 Support Vector Analysis

To further understand the structural differences among the weighting schemes, we analyze the number of support vectors (SVs) produced by each method. In the dual formulation of soft-margin SVMs,

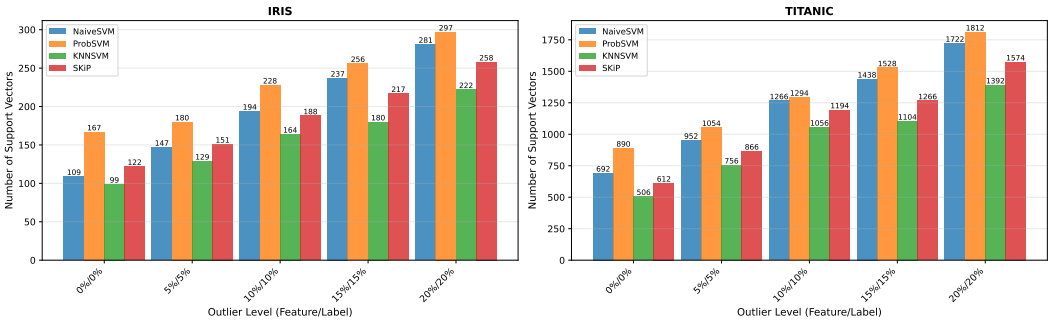


Figure 4: Comparison of the number of Support Vectors (SVs) across varying noise levels on Iris and Titanic datasets.

each sample satisfies the box constraint

$$0 \leq \alpha_i \leq C_i, \tag{5}$$

and becomes a support vector whenever $\alpha_i > 0$. Points with $0 < \alpha_i < C_i$ lie on the margin, whereas $\alpha_i = C_i$ indicates a point inside the margin or a misclassified sample. Thus, the per-sample upper bound C_i has a direct structural influence on SV formation: reducing C_i restricts the feasible range of α_i, and if the unconstrained optimum would exceed the reduced bound, the constraint becomes active and the sample becomes an SV. Furthermore, due to the dual feasibility condition

$$\sum_i \alpha_i y_i = 0, \tag{6}$$

adjusting C_i for one sample reshapes the optimal distribution of α_j across other samples, so the number of SVs does not change monotonically with C_i and may increase or decrease depending on the data geometry.

From this viewpoint, the observed differences among methods can be interpreted through their weighting structures. The KNN-based weighting n_i exhibits a *discrete-like* behavior: samples whose local neighborhoods disagree with their labels often yield n_i values that are very close to zero. Since each method enforces the constraint $\alpha_i \leq Cn_i$, the probability that $C_i = Cn_i$ becomes extremely small is relatively high. Consequently, many ambiguous or noisy points are effectively suppressed, contributing little to the decision boundary and resulting in the smallest number of SVs among the compared methods.

In contrast, the probabilistic weighting p_i is derived from Gaussian class-conditional likelihoods. Because Gaussian densities are continuous and rarely collapse to exactly zero—even in high-dimensional settings—the resulting weights satisfy $p_i > 0$ almost surely after normalization. Thus, the effective bounds $C_i = Cp_i$ seldom become extremely small. Combined with the dual coupling constraints, this continuous structure often leads to a larger number of active constraints and therefore to a larger number of SVs, sometimes even exceeding the Naive-SVM despite the fact that $C_i \leq C$ for all samples.

The SKiP method occupies an intermediate position. When using the averaging rule

$$C_i^{\text{SKiP}} = C \cdot \frac{p_i + n_i}{2}, \tag{7}$$

samples strongly suppressed by KNN-SVM (i.e., $n_i \approx 0$) receive partially restored weights through the influence of the continuous probabilistic term p_i, while highly representative samples with both large p_i and n_i retain large effective bounds. This mixture prevents the extreme sparsification characteristic of KNN-SVM, yet avoids the broad activation pattern of Prob-SVM. As a result, the number of SVs produced by SKiP consistently lies between those of the KNN-based and probabilistic models.

Overall, the SV statistics reflect fundamental differences in how each weighting scheme manages noisy or ambiguous samples and constructs the resulting decision boundary. These patterns highlight that SV analysis provides a meaningful lens through which to interpret the behavior of weighted SVM formulations.

5 Conclusion

In this work, we introduced SKiP, a weighted SVM framework that integrates a probabilistic reliability term with a KNN-based local consistency measure to jointly address feature outliers and label corruption. Across Gaussian and non-Gaussian datasets, SKiP achieved consistently stable performance compared to Naive-SVM, Prob-SVM, and KNN-SVM, demonstrating the advantages of its additive weighting design. Analysis of support vector structures further showed that SKiP forms a balanced decision boundary without excessive sparsity or instability. Overall, SKiP provides a practical and robust extension of soft-margin SVM under diverse noise conditions.

Limitations Although SKiP demonstrates improved robustness across diverse noise conditions, several limitations remain. First, the probabilistic component of SKiP is derived under a Gaussian assumption when estimating class-conditional likelihoods. Consequently, in datasets that deviate significantly from Gaussian behavior or contain substantial categorical structure, the reliability estimates may become less informative. While the additive integration with the KNN-based term alleviates this issue, SKiP still inherits a degree of sensitivity from its probabilistic foundation.

Second, the use of KNN-based local similarity requires additional neighborhood computations during training. This process increases the overall computational burden compared to standard soft-margin SVM and may limit the practicality of SKiP in large-scale applications.

Finally, SKiP is fundamentally built upon the soft-margin SVM framework, which itself can be computationally demanding. The reliance on SVM optimization means that SKiP retains the scalability challenges characteristic of margin-based classifiers.

Nevertheless, despite these limitations, design of SKiP allows it to retain strong robustness under label corruption and feature perturbations. This dual-weight structure enables SKiP to outperform conventional SVM variants in noisy or irregular settings, highlighting its effectiveness as a practical and resilient extension of soft-margin SVM.

Future work Shalev-Shwartz et al. introduced a stochastic gradient–based optimization algorithm for training SVMs called **Pegasos** [9] . At each iteration t, Pegasos selects a training sample (x_t, y_t) and performs a stochastic subgradient update with learning rate $\eta_t = 1/(\lambda t)$:

$$w_{t+1} = \begin{cases} (1 - \eta_t \lambda)\, w_t + \eta_t\, y_t\, x_t, & \text{if } y_t(w_t^\top x_t) < 1, \\ (1 - \eta_t \lambda)\, w_t, & \text{otherwise.} \end{cases}$$

A promising extension is to integrate our SKiP weighting mechanism into the Pegasos framework. Let p_t denote the probabilistic reliability weight (as in Prob-SVM) and n_t denote the KNN-based similarity weight. SKiP combines these two components additively, so we define

$$\alpha_t = \frac{p_t + n_t}{2}.$$

By incorporating α_t into the update, the Pegasos step becomes

$$w_{t+1} = \begin{cases} (1 - \eta_t \lambda)\, w_t + \eta_t\, \alpha_t\, y_t\, x_t, & \text{if } y_t(w_t^\top x_t) < 1, \\ (1 - \eta_t \lambda)\, w_t, & \text{otherwise.} \end{cases}$$

This modification allows SKiP to be trained using gradient-based updates while keeping its dual weighting structure. Since Pegasos enables fast and efficient training even on large-scale datasets, integrating the reliability and similarity terms into the Pegasos update can produce a more fast trainable version of SKiP on large-scale datasets. We believe this direction offers a promising path for future work.

Furthermore, the computational overhead introduced by the KNN-based similarity term suggests another promising direction for future improvement. Incorporating approximate nearest neighbor (ANN) search algorithms, such as HNSW [10], could substantially accelerate neighborhood queries while preserving high-quality locality information. Exploring ANN-based formulations may therefore enhance the scalability of SKiP, particularly in high-dimensional or large-scale settings.

References

[1] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998. doi: 10.1109/5254.708428.

[2] Chun-Fu Lin and Sheng-De Wang. Fuzzy support vector machines. *IEEE Transactions on Neural Networks*, 13(2):464–471, 2002. doi: 10.1109/72.991432.

[3] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10, 06 2000.

[4] Hamed Masnadi-Shirazi, Nuno Vasconcelos, and Arya Iranmehr. Cost-sensitive support vector machines. *CoRR*, abs/1212.0975, 2012. URL `http://arxiv.org/abs/1212.0975`.

[5] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2126–2136, 2006. doi: 10.1109/CVPR.2006.301.

[6] S. Aeberhard, D. Coomans, and O. de Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. Technical Report Tech. Rep. no. 92-02, Department of Computer Science and Department of Mathematics and Statistics, James Cook University of North Queensland, 1994.

[7] R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. DOI: https://doi.org/10.24432/C56C76.

[8] R. J. M. Dawson. The "unusual episode" data revisited. *Journal of Statistics Education*, 3(3): 1–14, 1995. Dataset: Titanic passenger survival data.

[9] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

[10] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016. URL `http://arxiv.org/abs/1603.09320`.

Appendix

A Cholesky-decomposed outlier injection

Algorithm 1: Generation of Feature Outliers.

Input: Training data $(X_{\text{train}}, y_{\text{train}})$; feature–noise rate k
Output: Augmented dataset $(\tilde{X}, \tilde{y})$

Estimate, for each class c, the mean μ_c and covariance matrix Σ_c.
For each class c, compute a Cholesky factor $\Sigma_c = L_c L_c^\top$.
$N_{\text{noise}} \leftarrow k \cdot |X|$
for $i = 1$ **to** N_{noise} **do**
  Select a class c uniformly at random.
  Sample latent vector $z \sim \mathcal{N}(0, I_d)$.
  Sample excess distance $E \sim \text{Exp}(2/\tau_c)$.
  Compute target squared distance $\gamma^2 \leftarrow \tau_c + E$.
  Calculate scaling factor $\alpha \leftarrow \sqrt{\frac{\gamma^2}{\|z\|_2^2}}$.
  Construct outlying point $x_{\text{new}} \leftarrow \mu_c + L_c(\alpha \cdot z)$.
  (if it crosses decision boundary) Append (x_{new}, c) to $(\tilde{X}, \tilde{y})$.
return $(\tilde{X}, \tilde{y})$

Here, we use the decision boundary produced by the scikit-learn SVM implementation as our baseline.

The primary motivation for this transformation is to enable precise and efficient control over the outlier generation process. In the original feature space, directly modifying x to satisfy a specific Mahalanobis distance is non-trivial due to the covariance structure. However, the proposed transformation maps the complex Mahalanobis distance to the simple Euclidean norm in the latent space ($\|z\|^2$). This equivalence allows us to generate a random seed z, deterministically scale its magnitude to meet the outlier threshold, and transform it back to x, thereby guaranteeing the desired distance without iterative checking.

For the validity of the proposed noise injection algorithm, we must demonstrate that if a latent vector z is sampled from a standard normal distribution $\mathcal{N}(0, I)$, the transformation $x = \mu + Lz$ yields a random vector x that strictly follows the target distribution $\mathcal{N}(\mu, \Sigma)$.

Formally, we proved the following theorem:

Theorem 1. *Let $\Sigma \in \mathbb{R}^{d\times d}$ be a positive definite covariance matrix with Cholesky decomposition $\Sigma = LL^T$, where $L \in \mathbb{R}^{d\times d}$ is a lower triangular matrix. If $z \in \mathbb{R}^d$ is a random vector following a standard normal distribution, $z \sim \mathcal{N}(0, I_d)$ where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix, and we define $x = \mu + Lz$ with $\mu \in \mathbb{R}^d$, then $x \in \mathbb{R}^d$ follows a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$.*

Proof. Let the probability density function (PDF) of the d-dimensional standard normal vector z be given by:

$$f_z(z) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} z^T z\right)$$

We define the affine transformation $x = \mu + Lz$. Since Σ is positive definite, L is invertible. Thus, we can express z as a function of x:

$$z = L^{-1}(x - \mu)$$

By the change of variables formula, the PDF of x, denoted as $f_x(x)$, is related to $f_z(z)$ by the Jacobian determinant of the inverse transformation:

$$f_x(x) = f_z(L^{-1}(x - \mu)) \left|\det\left(\frac{\partial z}{\partial x}\right)\right|$$

The Jacobian matrix of the transformation $z = L^{-1}(x - \mu)$ with respect to x is L^{-1}. Therefore, the Jacobian determinant term is:

$$\left|\det\left(\frac{\partial z}{\partial x}\right)\right| = |\det(L^{-1})| = \frac{1}{|\det(L)|}$$

From the property of determinants, $\det(\Sigma) = \det(LL^T) = \det(L)\det(L^T) = (\det(L))^2$. Consequently, $|\det(L)| = |\Sigma|^{1/2}$. Substituting this back into the equation:

$$\left|\det\left(\frac{\partial z}{\partial x}\right)\right| = \frac{1}{|\Sigma|^{1/2}}$$

Now, substituting $z = L^{-1}(x-\mu)$ into the exponential term of $f_z(z)$:

$$z^T z = (L^{-1}(x-\mu))^T(L^{-1}(x-\mu)) = (x-\mu)^T(L^{-1})^T L^{-1}(x-\mu)$$

Since $(L^{-1})^T L^{-1} = (L^T)^{-1}L^{-1} = (LL^T)^{-1} = \Sigma^{-1}$, the exponent simplifies to:

$$z^T z = (x-\mu)^T\Sigma^{-1}(x-\mu)$$

Finally, combining all terms, we obtain the PDF for x:

$$f_x(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

This is effectively the probability density function of the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. □

With Thm.(1), we can convert input data x into latent vector z, which follows normal distribution. Then, we can proceed to determine the scaling factor α required to project the latent vector z into the outlier region. By definition, a point is classified as Feature outliers if its squared Mahalanobis distance strictly exceeds the threshold τ (corresponding to the 99^{th} percentile). For the transformed vector $z' = \alpha z$, this condition implies:

$$\|z'\|^2 = \|\alpha z\|^2 = \alpha^2\|z\|^2 > \tau \quad \Longrightarrow \quad \alpha > \sqrt{\frac{\tau}{\|z\|^2}}.$$

A naive approach to satisfy this inequality is to apply a minimal deterministic scaling, such as $\alpha = \sqrt{1.05\tau/\|z\|^2}$. However, this method forces the generated noise to concentrate excessively near boundary defined by the 99^{th} percentile, failing to capture the diversity of outliers likely to be encountered in real-world scenarios.

To mitigate this issue and ensure a broader coverage of the feature space, we introduce a stochastic excess term E. We define the target squared distance γ^2 as the threshold τ augmented by E, where $E \sim \text{Exp}(\lambda)$. We calibrated the rate parameter λ of the exponential distribution so that the expected squared distance of the noise becomes $\frac{3}{2}\tau$. Deriving λ from this expectation constraint:

$$\mathbb{E}[\gamma^2] = \tau + \mathbb{E}[E] = \tau + \frac{1}{\lambda} = \frac{3}{2}\tau \quad \Longrightarrow \quad \lambda = \frac{2}{\tau}.$$

Substituting this back, we obtain the final formulation for the robust scaling factor α:

$$\alpha = \sqrt{\frac{\tau + E}{\|z\|^2}}, \quad \text{where } E \sim \text{Exp}\left(\frac{2}{\tau}\right).$$

B Can feature outliers Improve SVM Performance?

Contrary to the common intuition that noise invariably degrades model performance, we empirically observed that injecting feature outliers occasionally resulted in higher classification accuracy compared to the baseline. We hypothesize that this improvement stems from a data-driven regularization effect, where the expanded variance of the training data leads to a more robust decision boundary. In this section, we provide a theoretical justification for this phenomenon.

B.1 Proof of Covariance Inflation via Noise Sampling

First, we demonstrate that the covariance of the injected feature outliers is strictly "larger" (in the positive definite sense) than the covariance of the original distribution.

Let the original data follow a distribution with mean μ and covariance Σ. The squared Mahalanobis distance is defined as $D^2 = (x-\mu)^T\Sigma^{-1}(x-\mu)$. The feature outliers are generated from the region where $D^2 > \tau$, with τ being a high threshold (e.g., the 99^{th} percentile). To analyze the covariance structure, we apply a whitening transformation:

$$z = \Sigma^{-1/2}(x-\mu) \quad \sim \quad \mathcal{N}(0, I_d).$$

Under this transformation, the condition $D^2 > \tau$ is equivalent to $\|z\|^2 > \tau$.

Now, consider the covariance of the noise, denoted as $\text{Cov}(X \mid D^2 > \tau)$. Using the linearity of the covariance operator under affine transformations:

$$\text{Cov}(X \mid D^2 > \tau) = \Sigma^{1/2}\text{Cov}(z \mid \|z\|^2 > \tau)\Sigma^{1/2}.$$

Due to the spherical symmetry of the standard normal distribution, the off-diagonal elements of $\text{Cov}(z \mid \|z\|^2 > \tau)$ are zero. The diagonal elements are identical and can be derived from the expected squared norm. Since $\|z\|^2 = \sum_{i=1}^{d} z_i^2$, by symmetry:

$$\text{Var}(z_i \mid \|z\|^2 > \tau) = \mathbb{E}[z_i^2 \mid \|z\|^2 > \tau] = \frac{1}{d}\mathbb{E}[\|z\|^2 \mid \|z\|^2 > \tau].$$

For a standard normal distribution, the unconditional expectation is $\mathbb{E}[\|z\|^2] = d$. However, when conditioned on the tail region $\|z\|^2 > \tau$, the expected squared norm is strictly greater than the global mean:

$$\mathbb{E}[\|z\|^2 \mid \|z\|^2 > \tau] > \mathbb{E}[\|z\|^2] = d.$$

Consequently, the variance of the whitened noise components exceeds 1:

$$\text{Cov}(z \mid \|z\|^2 > \tau) = \beta I_d, \quad \text{where } \beta = \frac{1}{d}\mathbb{E}[\|z\|^2 \mid \|z\|^2 > \tau] > 1.$$

Mapping back to the original feature space, we obtain:

$$\text{Cov}(X \mid D^2 > \tau) = \beta\Sigma^{1/2} I_d \Sigma^{1/2} = \beta\Sigma \succ \Sigma.$$

This proves that the covariance of the injected noise is a scalar multiple of the original covariance, scaled by a factor $\beta > 1$, thereby strictly increasing the overall variance of the augmented dataset.

B.2 Connection to Margin and Generalization

The inflation of covariance directly impacts the Support Vector Machine (SVM) optimization. Consider the variance of the decision function projection for a weight vector w:

$$\text{Var}(w^T x) = w^T \Sigma w.$$

With the injection of Feature outliers, the effective covariance becomes $\tilde{\Sigma} \approx (1-\rho)\Sigma + \rho(\beta\Sigma)$, which satisfies $\tilde{\Sigma} \succ \Sigma$, where ρ implies percentage of noise. This implies:

$$w^T\tilde{\Sigma}w > w^T\Sigma w.$$

Therefore, the causal chain of the performance improvement can be summarized as follows:

$$\Sigma \uparrow \quad \longrightarrow \quad \text{Var}(w^T x) \uparrow \quad \longrightarrow \quad \text{Margin} \uparrow \quad \longrightarrow \quad \text{Generalization} \uparrow .$$

In conclusion, the injection of Feature outliers acts as a robust mechanism for variance inflation, potentially preventing overfitting to the dense core of the distribution. However, it is crucial to note that this improvement is observed only in specific cases where the noise intensity is moderate.

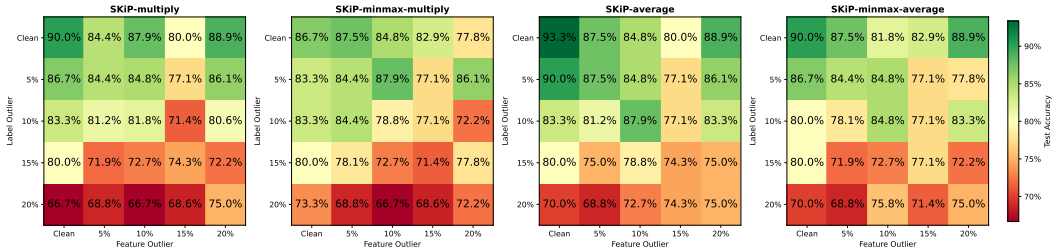
The benefit of noise injection relies on a delicate balance. While moderate noise effectively expands the feature space and encourages a more generalized boundary, excessive noise can be detrimental. If the injected outliers are sufficiently aggressive to drastically alter the set of **Support Vectors (SVs)**, they may distort the optimal hyperplane rather than stabilizing it. Thus, feature outliers serves as a regularizer only when it inflates the data variance without fundamentally corrupting the structural integrity of the original support vectors.

C Ablation on SKiP Weight Aggregation Strategies

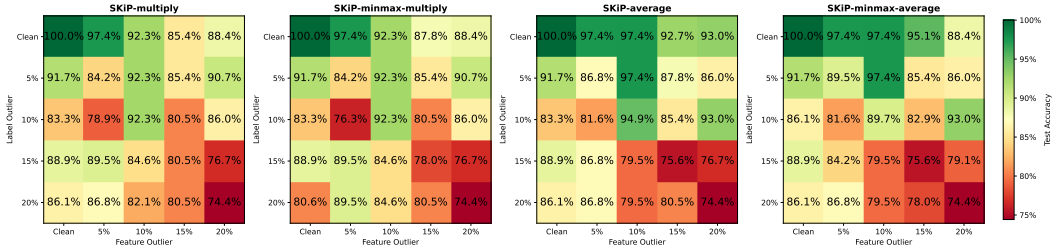
C.1 Experiments

For comparing SKiP model, we further implement four variants to examine different strategies for combining the probabilistic term p_i and the KNN-based term n_i. Specifically, we consider both multiplicative and arithmetic-mean formulations to investigate how the two weighting components interact. In addition, to ensure that the two terms have comparable influence, we also apply min–max normalization (scaling each term to the range $[0, 1]$), resulting in four SKiP variants.

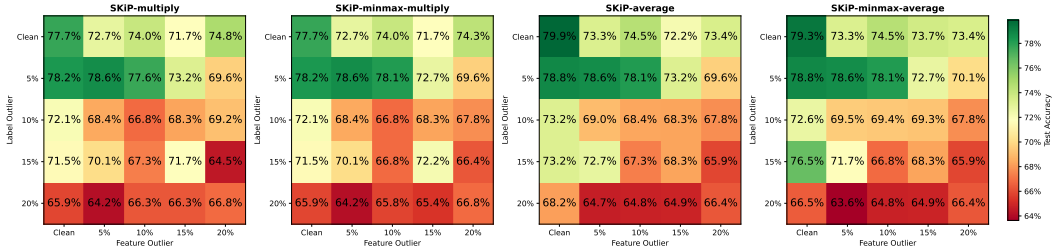
C.2 Results



(a) Iris



(b) Wine



(c) Titanic

Figure 5: Evaluation of multiplicative, averaged, and min–max–normalized SKiP weighting schemes under label and feature perturbations across three datasets

- **SKiP–Multiply**: This variant directly uses the multiplicative form

$$C_i = C\, p_i n_i,$$

  where the instance weight is defined as the product of the probability- and KNN-derived terms.
- **SKiP–MinMax–Multiply**: In this variant, both p_i and n_i are first scaled into the range $[0, 1]$ using min–max normalization, i.e., $\tilde{p}_i = \mathrm{MinMax}(p_i)$ and $\tilde{n}_i = \mathrm{MinMax}(n_i)$. The instance weight is then computed via

$$C_i = C\, \tilde{p}_i \tilde{n}_i.$$

- **SKiP–Average(baseline)**: This variant employs the arithmetic mean of the two terms:

$$C_i = C\, \frac{p_i + n_i}{2}.$$

- **SKiP–MinMax–Average**: Here, the min–max normalized terms $\tilde{p}_i$ and $\tilde{n}_i$, each rescaled to the interval $[0, 1]$, are averaged to obtain

$$C_i = C\,\frac{\tilde{p}_i + \tilde{n}_i}{2}.$$

In Figure 5, we can observe that the multiply formulation tends to be unstable: because the final weight is defined as the product of the probabilistic reliability and the KNN-based similarity, poor performance in either component substantially degrades the combined weight. As a result, when the probabilistic term becomes unreliable—for example, in non-Gaussian datasets such as Titanic—the multiply scheme fails to leverage the stronger KNN term, leading to inconsistent performance across noise levels.

In contrast, the average formulation shows noticeably more robust and stable performance across all datasets. Since the two components contribute additively, this scheme avoids the "weakest link" issue inherent in the multiplicative design and allows one component to compensate when the other becomes unreliable.

The min–max variants exhibit only marginal differences from their unnormalized counterparts. While min–max normalization adjusts the scale of each weighting component, it does not meaningfully alter relative sample ordering, which explains the minimal performance change. Nevertheless, the use of min–max normalization introduces a potential concern, and its practical implications should be interpreted with caution.

C.3 Potential Issues with Min–Max Scaling

In SKiP, the weight p_i is defined as a Gaussian-based probability assigned to each data point. Since it is a probability induced by the normal model, it necessarily satisfies

$$0 < p_i < 1.$$

Our concern is not the boundedness of p_i itself, but what happens when min–max scaling is applied to these values. Min–max normalization uses the empirical extrema of the dataset,

$$\tilde{p}_i = \frac{p_i - p_{\min}}{p_{\max} - p_{\min}}, \qquad p_{\min} = \min_j p_j, \quad p_{\max} = \max_j p_j,$$

and forces the smallest observed Gaussian probability to become exactly zero:

$$p_i = p_{\min} \quad \Rightarrow \quad \tilde{p}_i = 0.$$

This implies that if a new point x_j is drawn such that it lies farther from the class mean than the most extreme sample in the training set, then its Gaussian-based probability can become smaller than the empirical minimum. Consequently,

$$p_j < p_{\min} \quad \Rightarrow \quad \tilde{p}_j < 0 \quad \Rightarrow \quad \tilde{p}_j = 0 = \tilde{p}_i,$$

So, as a result, using min–max scaling can disrupt the assumption that p_i is drawn from a Gaussian-based probability distribution. Even when we adopt a practical safeguard by imposing a numerical floor,

$$p_i < 10^{-6} \;\Rightarrow\; p_i = 10^{-6},$$

this does not fully resolve the issue. Such clipping merely prevents extremely small probabilities from becoming numerically indistinguishable from zero, but it cannot restore the original Gaussian-derived semantics once the min–max transformation anchors the empirical minimum at $\tilde{p} = 0$ and makes the scale dataset-dependent. Therefore, the resulting weights should still be interpreted as relative scores rather than model-consistent probabilities.

D Other Model Comparision

In this section, we compare SKiP with several classical classifiers, including KNN, decision trees, and logistic regression, using an RBF–kernel SVM setup. As shown in Table 3, adopting the RBF kernel enables the SVM-based models to operate with expressive capacity comparable to nonlinear baselines such as KNN and decision trees, providing a fair evaluation across heterogeneous model classes. This setting further verifies that the SKiP framework is naturally compatible with kernelized SVMs and consistently retains its performance advantages beyond the linear case.

Table 3: Performance comparison of baseline models and SVM variants under noise perturbations.

Dataset	Outlier (%)		Accuracy (%)			
	Label	Feature	KNN	Decision Tree	Logistic Regression	SKiP
Iris	0	0	**96.7**	86.7	93.3	93.3
		10	**84.8**	81.8	81.8	**84.8**
		20	83.3	**88.9**	**88.9**	86.1
	10	0	83.3	**86.7**	**86.7**	**86.7**
		10	81.8	**84.8**	81.8	**84.8**
		20	80.6	**83.3**	**83.3**	80.6
	20	0	66.7	**70.0**	66.7	**70.0**
		10	66.7	**75.8**	66.7	66.7
		20	69.4	72.2	**75.0**	69.4
Wine	0	0	97.2	94.4	**100.0**	**100.0**
		10	97.4	79.5	97.4	**100.0**
		20	93.0	83.7	**95.3**	93.0
	10	0	80.6	80.6	**83.3**	**83.3**
		10	89.7	79.5	92.3	**94.9**
		20	86.0	74.4	**90.7**	**90.7**
	20	0	80.6	80.6	**83.3**	**83.3**
		10	74.4	74.4	79.5	**84.6**
		20	72.1	65.1	69.8	**76.7**
Titanic	0	0	**83.8**	81.0	81.6	82.7
		10	81.1	80.1	74.5	**82.7**
		20	79.9	76.6	68.2	**81.8**
	10	0	**76.0**	**76.0**	72.1	73.2
		10	**74.5**	73.5	65.8	**74.5**
		20	72.9	**74.3**	62.6	72.0
	20	0	64.8	65.4	67.0	**69.3**
		10	67.3	67.3	66.3	**71.4**
		20	**71.5**	69.6	56.1	71.0