Project Report: Locally Optimized Feature Weighted Ball Tree kNN - LOFWB-KNN

Anna Wisniewski and Dylan Tallis
Machine Learning 1
Dr. Selma Yilmaz
February 3, 2025

Abstract
The k-Nearest Neighbors (kNN) algorithm is widely used for classification due to its simplicity and effectiveness. However, traditional kNN suffers from limitations such as being unable to adjust to local neighborhood density and equal feature weighting in distance calculations, which means less relevant features are equally considered. This paper introduces an adaptive kNN variant that integrates local k selection [16], information-based feature weighting, and the time-efficient Ball Tree method [18] to improve classification performance. Feature importance is determined using a combination of Mutual Information (MI) [11] and Shapley values [22] to ensure that more informative features contribute more significantly to distance calculations. In order to be flexible to local conditions around an instance, the algorithm dynamically selects a local k value for each test instance based on the consistency of its nearest neighbors to prevent misclassification in heterogeneous datasets. The final classification is performed using majority voting among the k nearest neighbors. Experimental results demonstrate that the proposed approach outperforms traditional kNN and standard feature-weighted kNN methods on the tested UCI Machine Learning datasets, Mammographic Mass [12] and Blood Transfusion Service Center [4]. These findings suggest that incorporating adaptive feature weighting, local k-selection, and probabilistic feature importance estimation significantly enhances kNN's flexibility and classification accuracy.

Introduction

The k-Nearest Neighbors (kNN) algorithm is a widely used non-parametric classification method known for its simplicity and effectiveness. However, traditional kNN suffers from several limitations, particularly when applied to heterogeneous datasets where instance density and feature importance vary significantly. A fixed k value across the dataset often fails to capture local variations in the distribution of data, which may lead to decreased performance in classification. Additionally, standard kNN treats all features equally, though some may be more or less relevant to the classification task than others. We address these challenges by creating a more flexible and adaptive kNN variant capable of adjusting both the neighborhood size and feature importance to improve classification accuracy.

To address these limitations, we introduce an enhanced kNN algorithm that incorporates feature weighting based on mutual information (MI) [11] and Shapley values [22], as well as local selection of k [16]. The input to our algorithm consists of structured tabular data with numerical and categorical features, such as those found in common machine learning datasets. The algorithm first applies MI-based feature weighting and Shapley values to emphasize more relevant attributes when later performing the distance calculations before constructing a Ball Tree [18] to make neighbor searches more efficient. During prediction, our model dynamically selects the best local k based on the density and agreement of surrounding instances, which ensures adaptability if a dataset has irregular density.

We evaluate our proposed algorithm using UCI machine learning datasets, which are generally known as standardized benchmarks for comparing classification models. The chosen datasets are Mammographic Mass [12] and Blood Transfusion Service Center [4]. Testing our model on heterogeneous datasets with varying instance densities tests its ability to adapt to different data distributions. The primary evaluation metric is classification accuracy, measured on a test set and compared against standard kNN, fuzzy kNN, and Kumbure and Luukka's feature weighted [11], Minkowski distance, local means-based fuzzy kNN algorithm. By integrating adaptive neighbor selection and feature weighting, we develop a kNN variant that is

more robust, flexible, and capable of delivering improved performance in complex classification tasks.

Related Work

The original KNN algorithm was created in 1951, and in 1985 James Keller refined the algorithm and called it "fuzzy KNN" (FKNN). Keller's FKNNs addressed the drawback of assigning equivalent importance to each nearest neighbor, and used distance weighting to lower the error rate [8]. After the introduction of FKNNs, various algorithms have introduced enhancements to the classical FKNN method. Kumbure and Luukka employ a combination of feature weighting, Minkowski distance, and local means to improve accuracy of their local means-based fuzzy kNN (FWM-LMFKNN) [11]. They determined feature importance using both feature relevance and complementarity, which measure individual feature importance and combined feature importance, respectively, using MI as a measure of relevance. Kumbure and Luukka also include fuzzy entropy as a measure of uncertainty and opted to use Minkowski distance, rather than the Euclidean or Manhattan, as it is a better generalization to varying types of data. The researchers compared their algorithm's results with the accuracies of seven other baseline kNN models, each tested on 20 data sets. Their algorithm achieved the highest accuracy on sixteen of the twenty datasets but lacks the complexity introduced by Shapley values to better estimate the contribution of each feature [11].

Mutual information has also been utilized in conjunction with Shapley's values. In Vahedifar et al., mutual information, which measures how much information one variable contains about the target variable, is used to assign weights to neighbors. Shapley values were inspired by cooperative game theory, and measure the contributions of data points to the final prediction. The researchers proposed Information-Modified kNN uses a global k value and both MI and Shapley values to build a straightforward algorithm that outperforms the traditional KNN [22]. Vahedifar et al. compared their performance on twelve datasets against the performance of eight other baseline KNN models, and achieved the highest accuracy on eight and highest precision and recall on seven.

One of the compared algorithms is locally adaptive kNN, that addresses the overgeneralization of a global k value in Vahedifar et al. Pan et al. proposed a version of locally adaptive kNN based on discrimination class [16]. The traditional kNN algorithm has a fixed global k, is very sensitive to changes in k, and only considers the majority class, ignoring the information from the second majority class completely. The implementation of locally adaptive k addresses these problems by determining the optimal k for each test sample. The algorithm starts with a predefined max k and selects the smallest k such that the cumulative distance to the k neighbors is below a certain threshold, and does so by tracking the frequencies of not only the majority class but also the second majority. This allows the algorithm to adjust to varying data densities to choose the optimal k. The algorithm was able to significantly outperform the standard kNN algorithm [16]. Locally adaptive k is an algorithm enhancement, but can better outperform the standard kNN when combined with the previously mentioned algorithms. The combination of all enhancements, though, significantly increases runtime efficiency.

Many of the improvements made to the kNN algorithm are to improve accuracy, but others address computational speed and efficiency. Constructing nearest neighbor graphs is computationally expensive, especially with high dimensional data. Rajani et al. compared the implementations of kNN using k-d trees and ball trees [18]. KD-trees divide the space into axis-aligned hyperplanes which works well with evenly distributed data and lower dimension

dataset. Ball trees partition the space using hyperspheres which make it better for high dimensional datasets. KNN only needs to visit half the data points in a ball tree. Results comparing different implementations of KD and ball trees as opposed to brute force show that the use of ball trees significantly increases efficiency in runtime [18].

Our algorithm combines the accuracy improvements of MI [11][22], Shapley values [22], and locally adaptive k [16] while maintaining computational efficiency using a Ball Tree [18].

Dataset and Features

To evaluate our model's performance, two datasets from the UCI Machine Learning Repository were used: the Mammographic Mass dataset [12] and the Blood Transfusion Service Center dataset [4].

The Mammographic Mass dataset [12] contains 961 instances, each representing a case where a mass was detected in a mammogram. The goal is to classify the mass as benign or malignant based on five features: BI-RADS assessment (1–5), patient age, shape (round, oval, lobular, or irregular), margin (circumscribed, microlobulated, ill-defined, spiculated), and density (1–4). Missing values were handled by removing instances if they had no class value associated and imputing median for missing feature values. Categorical features were numerically encoded.

The Blood Transfusion Service Center dataset [4] consists of 748 instances, each representing a donor's history, with the goal of predicting whether the donor will donate blood again within six months. It includes four numerical features: recency (months since last donation), frequency (total donations), monetary (total volume of blood donated in c.c.), and time (months since first donation). No categorical data or missing values were present, but the numerical features operate on different scales. To prevent larger-valued features from dominating distance-based calculations, z-score normalization was applied.

Both datasets were split into 80% training and 20% testing while preserving class distributions. They were saved as CSV files for consistency across experiments.

Methods

Our proposed model augments the k-Nearest Neighbors (kNN) algorithm by incorporating feature weighting using Shapley values [22] based on an embedded Naive Bayes model and mutual information (MI) [11], using a Ball Tree [18] for efficient neighbor searches, and dynamically selecting a local k [16] for each test instance based on consistency of nearby data. The proposed modifications address some of the limitations found in traditional kNN by making sure that more relevant features are more highly weighted in distance calculations and that k value can adapt to local data density variations.

The algorithm begins with a training phase in which a Naive Bayes model is trained on the dataset to prepare for calculating Shapley values [22] to weight features. The type of Naive Bayes model used depends on the nature of the dataset: Multinomial Naive Bayes [13] is applied when the dataset primarily consists of categorical variables, while Gaussian Naive Bayes [6] is used when the dataset contains mostly continuous numerical features. Naive Bayes is based on Bayes' theorem, which expresses how the probability of a class given observed features can be computed using prior probabilities and proportion of features:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

In the context of machine learning, A represents the class and B represents the observed feature value, while $P(A|B)$ corresponds to the probability of a particular class given the value of the

feature of the current test instance. *P(B|A)* represents the probability of observing that feature value within a specific class, *P(A)* represents the prior probability of the class occuring in the dataset, and *P(B)* represents the probability of that feature value occuring.

For Gaussian Naive Bayes, instead of using raw probabilities for feature likelihoods (*P(B|A)*), the likelihood of a continuous feature is computed using a Gaussian distribution with class-specific means and variances:

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}$ is the variance of the feature within class *A*. This makes Gaussian Naive Bayes more appropriate for datasets with continuous numeric attributes.

This model is then used to compute Shapley values [2], which estimate how much each feature contributes to classification decisions. The Shapley value for a particular feature *i* is calculated by checking how the model's predictions change when that feature is included or excluded from subsets of features. This process involves testing different feature combinations and averaging their contributions across all possible subsets. The goal is to quantify how much each feature influences the probability distribution of the output class. Given a set of features *F* and a class label, the Shapley value for feature *i* is calculated as:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (v(S \cup \{i\}) - v(S))$$

In this formula, *S* represents a subset of features excluding *j*, and *f(S)* represents the class probability predicted by the Naive Bayes model when only the subset *S* is used. Basically, this formula evaluates how much the presence of feature *j* improves classification accuracy on the training set compared to cases where it is absent, with the final score being an average contribution across all possible feature subsets. Shapley values are a great way to weight features because they provide information directly about which features are most useful in classification. Once these values are extracted, they are used in tandem with MI to weight features before any distances are calculated.

MI [14] measures how much information a feature provides about the class label by comparing the feature's probability distribution to the joint distribution of the feature and the class. Unlike simple correlation, which only captures linear relationships, MI accounts for more complex dependencies, which makes it particularly useful when a feature-class relationship is nonlinear. Intuitively, mutual information measures how much a feature "tells us" about the class label.

Mathematically, mutual information is computed by comparing the probability of the feature and class occurring together to what we would expect if they were independent. If observing a feature changes the likelihood of a class more than what we would expect by chance, the feature is deemed more informative. This ensures that features highly associated with a particular class receive greater weight in our model. MI between feature *x* and class *y* is given by the following:

$$\text{Mutual Information} = \sum_{y \in Y} \sum_{x \in X} p(x, y) \cdot log \frac{p(x, y)}{p(x)p(y)}$$

In this formula, *p(x,y)* is the joint probability of feature *x* and class *y*, while *p(x)* and *p(y)* are their individual probabilities. This formula quantifies how much knowing the value of *x* reduces

uncertainty about *y*, similarly to Shapley values. If mutual information is high, then *x* strongly influences the class label and should be weighted accordingly.

After they are computed, Shapley values and MI scores are combined into a single weighting factor which is their product over the sum of the products of both values for all features. These factors are then used to scale the dataset before we construct the Ball Tree.

A Ball Tree [3] is a data structure that partitions the dataset into nested hyperspheres, which reduces the complexity of kNN searches from O(N) to O(logN). The Ball Tree is built recursively by dividing the dataset into nested hyperspheres, or "balls." It begins by selecting a pivot point and computing the median distance from this pivot to all other points. The data is then split into two groups: one containing points inside the median distance (the left subtree) and one containing points outside it (the right subtree). This process is repeated recursively until each leaf node contains only a small number of points, typically one or a few data instances. The resulting tree structure allows nearest-neighbor searches to efficiently discard large sections of the dataset that are unlikely to contain the closest points. The distance metric we use is weighted Euclidean distance. We chose to use the Ball Tree to speed up searches using our model since finding a local k value for each point slows the testing step down.

During classification, instead of using a globally fixed k [22], the model dynamically determines the best k value for each test instance [16]. The optimal k is selected based on classification consistency, ensuring that the chosen set of neighbors forms a stable classification decision. The smallest k value with a high consistency of classification decision will be chosen for the region around that instance.

Finally, the test instances are classified using a majority vote within the neighbors, as in the original kNN algorithm. Each of the k nearest neighbors contributes one vote to its corresponding class, and the class with the most votes is assigned as the final prediction. By integrating weighted feature importance, efficient neighbor retrieval using a Ball Tree, and locally adaptive k selection, this approach improves upon traditional kNN, particularly for datasets with varying feature relevance and density distributions.

Results and Evaluation

We wrote simple KNN and FKNN algorithms to compare our algorithm's performance.

Figure 1

Accuracies of LOFWB-KNN, KNN, and FKNN

Dataset	LOFWB-KNN	KNN	FKNN
Mammogram	80.83	79.79	77.20
Blood	78.67	74.67	73.33

The accuracy of our LOFWB-KNN showed over a one point increase on the mammogram dataset and a four point increase on the blood dataset (Figure 1). To further analyze the performance of our proposed method, we compute precision and recall for each dataset. These metrics provide deeper insights into the classification performance beyond mere accuracy.

Figure 2

Mammogram Dataset Confusion Matrices

LOFWB-KNN		Predicted	
		Benign	Malignant
Actual	Benign	81	15
	Malignant	22	75

KNN		Predicted	
		Benign	Malignant
Actual	Benign	81	15
	Malignant	24	73

FKNN		Predicted	
		Benign	Malignant
Actual	Benign	77	19
	Malignant	25	72

Figure 3
Mammogram Dataset Measures

Model	Precision	Recall
LOFWB-KNN	0.833	0.773
KNN	0.830	0.753
FKNN	0.791	0.742

Although our algorithm's accuracy was only approximately one percentage point more accurate than KNN, the second highest (Figure 1), the results in Figure 3 indicate that our model achieves the highest precision and recall among the three methods, highlighting its effectiveness in detecting relevant patterns in mammogram data. In the context of the mammogram data, both precision and recall are important, as a false positive could lead to unnecessary treatment and a false negative could prevent the patient from receiving the treatment they need, both potentially life-threatening mistakes. For both measures, our algorithm scored highest, followed by the traditional KNN. Our LOFWB-KNN's precision is slightly higher than KNN's by 0.003 and our recall showed a larger improvement over KNN by 0.02.

Figure 4
Blood Dataset Confusion Matrices

LOFWB-KNN		Predicted	
		Not Donate	Donate
Actual	Not Donate	111	8
	Donate	24	7

KNN		Predicted	
		Not Donate	Donate
Actual	Not Donate	104	15
	Donate	23	8

FKNN		Predicted	
		Not Donate	Donate
Actual	Not Donate	102	17
	Donate	23	8

Figure 5

Blood Dataset Measures

Model	Precision	Recall
LOFWB-KNN	0.467	0.226
KNN	0.348	0.258
FKNN	0.320	0.258

We observe similar trends in the accuracy and precision of LOFWB-KNN on the blood dataset in Figure 1 and Figure 5. Although, comparatively, the recall suffered slightly, this measure holds less significance in the context of the blood dataset. False negatives indicate a patient donating blood after being predicted not to, so while the prediction was wrong, the outcome was preferable.

Figure 6

Accuracies of LOFWB-KNN and FWM-LMFKNN

Dataset	LOFWB-KNN	FWM-LMFKNN
Mammogram	80.83	77.31
Blood	78.67	77.65

The accuracy of our LOFWB-KNN can also be compared to the recorded accuracies of Kumbure and Luukka's FWM-LMFKNN on the mammogram and blood datasets in Figure 6. It must be recognized, though, that the methods used for data preprocessing and train/test splits may differ, since we were unable to replicate their model. Nevertheless, when comparing, our LOFWB-KNN achieved higher accuracies on both datasets. The accuracies can be better visualized in Figure 7 and Figure 8.

Figure 7

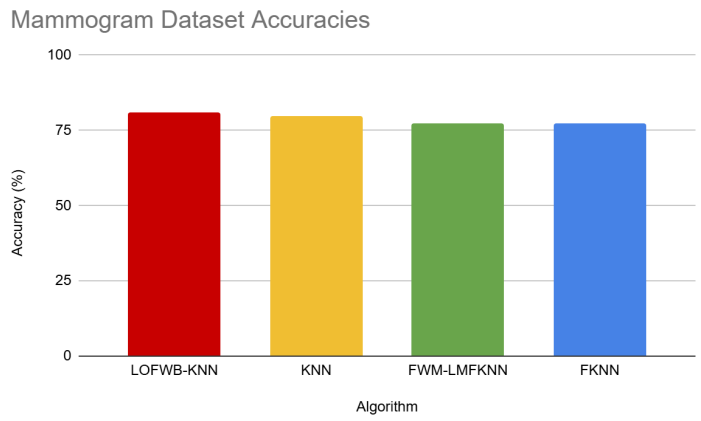
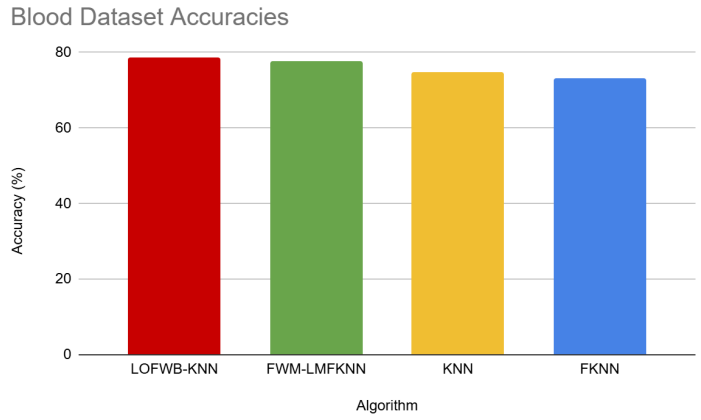


Figure 8



Discussions and Conclusions

Our proposed algorithm, which integrates Ball Tree, Mutual Information, Shapley values, and local k selection, achieves higher accuracy across both the Mammogram Mass [12] and Blood Transfusion Service Center datasets [4] compared to standard KNN and FKNN. Taking into account differences in dataset splits, our LOFWB-KNN also outperformed Kumbure and Luukka's FWM-LMFKNN. This improvement shows how effective feature weighting and local adaptability are in enhancing classification performance.

The confusion matrix analysis further demonstrates that our approach maintains a balance between precision and recall, compared to both traditional KNN and FKNN. The combination of Mutual Information and Shapley values allows for more informed weighting which reduces the impact of irrelevant or redundant information. Moreover, the implementation of a Ball Tree improves neighbor search efficiency and optimizes computational complexity. The incorporation of local k selection ensures the model is able to dynamically adapt to local data distributions, further improving its robustness.

While the current results are promising, future work could focus on small enhancements to continue to boost accuracy. In our research, we found algorithms that, rather than relying on distance based voting, employed a localized Naive Bayesian approach. Although including local Naive Bayes did not make sense for our algorithm, it would be interesting to test the effect of its addition to a modified version of our LOFWB-KNN. Additionally, the specific approach to weighting with MI and Shapley values could be altered to improve accuracy on higher dimensional data or specific types of datasets, such as unbalanced data.

Contributions

Anna: Wrote most of Methods, Datasets and Features, and Introduction

Dylan: Wrote most of Related Work, Experiments/Results/Discussion, and Conclusion/Future Work

Both: Worked on code collaboratively - can't specify specific parts since we each did some work on each part of code
If not stated, assume roughly equal contribution.

References

[1] *accuracy_score*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[2] *API reference*. (n.d.). SHAP. Retrieved February 4, 2025, from https://shap.readthedocs.io/en/latest/api.html

[3] *BallTree*. (n.d.). Scikit Learn. Retrieved February 4, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html

[4] *Blood transfusion service center*. (2008, October 2). UC Irvine Machine Learning Repository. Retrieved February 6, 2025, from https://archive.ics.uci.edu/dataset/176/blood+transfusion+service+center

[5] *confusion_matrix*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[6] *GaussianNB*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[7] *inv*. (n.d.). SciPy. Retrieved February 6, 2025, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.inv.html
[8] Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(4), 580-585. https://doi.org/10.1109/tsmc.1985.6313426
[9] *KNeighborsClassifier*. (n.d.). Scikit Learn. Retrieved February 4, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[10] *LabelEncoder*. (n.d.). Scikit Learn. Retrieved February 4, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[11] Mailagaha Kumbure, M., & Luukka, P. (2024). Local means-based fuzzy k-nearest neighbor classifier with minkowski distance and relevance-complementarity feature weighting. *Granular Computing*, *9*(4). https://doi.org/10.1007/s41066-024-00496-0
[12] *Mammographic mass*. (2007, October 28). UC Irvine Machine Learning Repository. Retrieved February 6, 2025, from https://archive.ics.uci.edu/dataset/161/mammographic+mass
[13] *MultinomialNB*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[14] *Mutual_info_classif*. (n.d.). Scikit Learn. Retrieved February 4, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html
[15] *NumPy*. (n.d.). NumPy. Retrieved February 6, 2025, from https://numpy.org/
[16] Pan, Z., Wang, Y., & Pan, Y. (2020). A new locally adaptive k-nearest neighbor algorithm based on discrimination class. *Knowledge-Based Systems*, *204*, 106185. https://doi.org/10.1016/j.knosys.2020.106185
[17] *Pandas documentation*. (n.d.). Pandas. Retrieved February 6, 2025, from https://pandas.pydata.org/docs/index.html
[18] Rajani, N., McArdle, K., & Dhillon, I. S. (2015). Parallel k nearest neighbor graph construction using tree-based data structures. *HPGM: High Performance Graph Mining. "1st High Performance Graph Mining Workshop, Sydney, 10 August 2015."* https://doi.org/10.5821/hpgm15.1
[19] *StandardScaler*. (n.d.). Scikit Learn. Retrieved February 4, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[20] *StratifiedKFold*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
[21] *train_test_split*. (n.d.). Scikit Learn. Retrieved February 6, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[22] Vahedifar, M. A., Akhtarshenas, A., Sabbaghian, M., Rafatpanah, M. M., & Toosi, R. (2024, May 14). Information modified k-nearest neighbor. Retrieved February 6, 2025, from https://arxiv.org/abs/2312.01991