Adaptive Weighted Voting KNN: A Hybrid Clustering Method for Mixed Data

Marina Lin, Clinton Morimoto

Tuesday 18th February, 2025

Abstract

Loan approval prediction is a critical task in financial systems as it directly impacts individuals' access to resources and institutions' risk management. These problems tend to be heterogeneous, skewed and contain challenging mixed-type datasets, which combine numerical and categorical variables. In this paper, we introduce the *Adaptive Weighted Voting K-Nearest Neighbors Algorithm*, which processes both numerical and nominal features for more accurate loan prediction. By experimenting with Euclidean and Mahalanobis distance strategies for numerical features and dissimilarity measures for categorical features, we discover novel correlations that allow more accurate prediction in complex datasets. Traditional algorithms tested on uniform datasets often fail to generalize to complex datasets with imbalanced class distributions. Our mixed data KNN algorithm incorporates Mahalanobis distance and weighted majority voting to overcome limitations in skewed class datasets. Experimental evaluations on robust loan datasets demonstrated an accuracy of 88.94%, a significant improvement performance of 2.66% compared to the conventional baselines, highlighting the efficacy of our novel algorithm in real-world applications.

1 Introduction

Loan approval is a critical problem that is fundamental to all financial systems, as it directly impacts individuals' access to funds and institutions' ability to manage risk effectively. The problem is assessing an applicant's current financial credit score and personal background and determining if loans will be returned in a timely manner. This decision is vital because misjudgments can lead to significant financial losses for institutions (if they do not get the funds back in a timely manner), and rejecting creditworthy applicants can limit economic growth and aspirations.

Out dataset involves binary classification for loan approvals, but such datasets are often highly heterogeneous, combining categorical variables (e.g., employment type, education level) with continuous variables (e.g., income, loan amount), and tend to be heavily skewed, with fewer approved loans than rejected ones. This is our motivation for pursuing this problem: the real-world complexity and factors that determine, which depend on various interrelated factors such as credit history, income stability, and loan purpose.

For performance evaluation, the input to our algorithm includes applicant features (nominal and numeric) such as income, education, age, loan purpose, and home ownership, to name a few. We propose a mixed data type K-Nearest Neighbors (KNN,) with adaptive weight adjustment for minority labels, to predict whether a loan is approved (1) or rejected (0).

1.1 Related Work

KNN, originally designed in the 1950s as a rudimentary classification algorithm, has seen several innovations in its core paradigm within the last several decades. The primary hyperparameter of the KNN algorithm is the choice of the k parameter — whilst the standard choice is the square root of instance count, other attempts are based on similar functions such as logarithms, direct fine tuning, or ensemble learning, with varying success based on the individual datasets [2].

An additional parameter of KNN is the choice of distance metric, wherein the primary choice is Euclidean distance. A different promising distance metric, however, is the Mahalanobis distance, which compares each data point to the feature distribution [3]. Going further, another popular technique for optimizing KNN is the mixed weighting method, which assigns inverse proportion weights and is especially useful for imbalanced-class datasets [1].

Each of these approaches provides a unique advantage to KNN, though only in particular circumstances, such as correlated feature sets for Mahalanobis distance and imbalanced class labels for mixed weighting. Our research attempts to unify a variety of these techniques, including the aforementioned two, as they fit well for our quantitative and categorical data.

2 Dataset and Features

2.1 Data Description

This dataset from Lo in 2024 is a synthetic version inspired by an original credit risk dataset on Kaggle [5]. Furthermore, there are additional variables based on Financial Risk for Loan Approval data. SMOTENC was used to simulate new data points to enlarge the instances. This dataset is complex in that it contains both categorical and continuous features, and all the features are detailed in Table 1. There are 13 features, including information about an individual's income, education, home ownership, and loan intent. The target variable is loan status which is a binary variable with either a loan approved indicated by 1 or a loan rejected indicated by 0.

Table 1: Dataset Features with Description and Type

Column	Description	Type
person age	Age of the person	Float
person gender	Gender of the person	Categorical
person education	Highest education level	Categorical
person income	Annual income	Float
person emp exp	Years of employment experience	Integer
person home ownership	Home ownership status (e.g., rent, own, mortgage)	Categorical
loan amnt	Loan amount requested	Float
loan intent	Purpose of the loan	Categorical
loan int rate	Loan interest rate	Float
loan percent income	Loan amount as a percentage of annual income	Float
cb person cred hist length	Length of credit history in years	Float
credit score	Credit score of the person	Integer
previous loan defaults on file	Indicator of previous loan defaults	Categorical
loan status	Loan approval status: 1 = approved; 0 = rejected	Integer

We describe the class distribution of labels (between 0 for loans not approved and 1 for loans approved) in Figure 1. There are 5585 training instances with label 0 and 1615 training instances with label 1. Similarly, there are 1393 test instances with label 0 and 407 test instances with label 1, clearly indicating a skewed dataset, which is a key challenge we are trying to address.

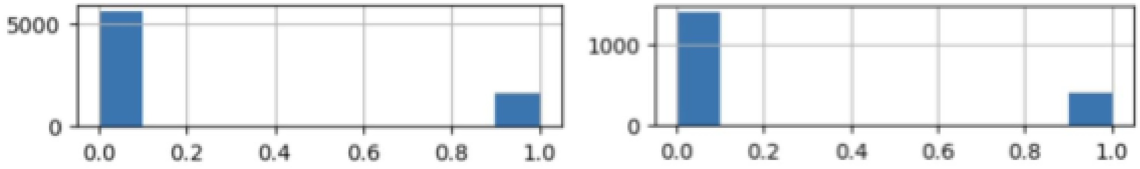

Figure 1: Class Label Distribution for Train (left) and Test (right).

2.2 Preprocessing and Train/Test

In order to best facilitate the KNN algorithm, we normalized all numeric attributes into a 0–1 range. Similarly, we set all binary categorical variables to either a numeric 0 or 1. As person education is an ordinal categorical variable, we converted it into a 0–0.25–0.5–0.75–1 numeric variable. We kept all remaining categorical variables the same — they were not transformed to numeric. Due to the computational time complexity of the KNN algorithm, we removed 80% of the instances within the dataset randomly — there were no missing values for any attributes. Lastly, we used a standard 80%–20% train-test split for our data.

3 Methodology

In this section, we introduce the adaptive *Weighted Voting KNN*, a novel hybrid approach designed to address the limitations of traditional uniform clustering methods when applied to mixed-type datasets. The standard algorithm uses Euclidean distance for numerical features and a dissimilarity measure for categorical features. We extend this by using a more robust metric known as Mahalanobis distance as well as a minority class voting aggregation strategy. In the following sections, we detail the mathematical foundation of KNN, the dissimilarity measure, and our new algorithm.

3.1 K-Nearest Neighbors (KNN) Algorithm for Mixed Data

Since KNN is an algorithm for classification and regression, it works for both numerical and categorical data. We begin by introducing KNN for numerical features. KNN assigns a class to a given point based on the majority vote from its *k* nearest neighbors, determined by minimizing the Euclidean distance between points.

We define Euclidean Distance as:

$$e(x,y) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$$

where *x* and *y* are points in the feature space, and x_j, y_j are the numerical values of a feature *j*.

Next, we introduce the dissimilarity measure used for categorical features:

$$c(x,y) = \begin{cases} 0, & \text{if } x = y \\ 1, & \text{if } x \neq y \end{cases}$$

where *x* and *y* are categorical feature values. If *x* equals *y* the dissimilarity is 0, otherwise the dissimilarity is 1. To formulate a baseline mixed data KNN algorithm, we combine the distance functions for numerical and categorical features. For a mixed dataset, the total distance between two points *x* and *y* is calculated as:

$$d(x,y) = \sqrt{\sum_{j\in F_n} (x_j - y_j)^2 + \sum_{j\in F_c} c(x_j, y_j)}$$

where F_n is the set of numerical feature indices, F_c is the set of categorical feature indices, and $c(x_j, y_j)$ is the dissimilarity function for categorical features defined above. The algorithm classifies as follows: for a test point *x*, identify its *k* nearest neighbors based on the combined distance measure *d*(*x,y*), then assign *x* the class label that occurs most frequently among its *k* nearest neighbors.

3.2 Novel Distance and Weighted Voting Scheme

For datasets with non-uniform distributions, global distance measures such as Euclidean distance may lead to suboptimal clustering results and not enough minority class representation. To address this limitation, we integrate a weighted voting scheme into the KNN algorithm. This novel extension involves two new enhancements: 1) using a Mahalanobis Distance for numerical features and 2) incorporating a weighted voting mechanism.

First, we define the Mahalanobis distance as a measure of the distance between a point and a distribution for numerical features. It accounts for the correlations between features and the variances within the data and can be defined as:

$$D_M(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}$$

where Σ is the covariance matrix. The algorithm adapts to the shape and distribution of the data by using Mahalanobis distance, improving its ability to handle complex mixed datasets.

To incorporate a weighted voting system, we assign weights to the *k*-nearest neighbors of a test point *x* based on their class membership. The weights are determined using the frequency of each class in the training data. Let c_{minor} and c_{major} represent the minority and majority class labels, respectively. The weights $w(y_i)$ for class y_i are assigned as follows:

$$w(y_i) = \begin{cases} \mathscr{H}, & \text{if } y_i = c_{\text{minor}} \\ h, & \text{otherwise.} \end{cases}$$

where *H (fancy denoted H above)* = 2 and *h* = 1 for this particular dataset. This ratio is chosen based on the ratio of c_{minor} and c_{major} and a grid search on a subsample of the data shows *H* = 2 and *h* = 1 performs the best. For a test point *x*, the weights of the *k*-nearest neighbors are accumulated for each class *c*:

$$w_c = \sum_{(d_i, y_i) \in N_k(x)} \nVdash [y_i = c] \cdot w(y_i)$$

where d_i is the distance of the *i*-th neighbor to *x*, y_i is the class label of the *i*-th neighbor, $N_k(x)$ is the set of the *k*-nearest neighbors of *x*, $w(y_i)$ is the weight assigned to the class of neighbor y_i, and ⊮[·] is an indicator function that equals 1 if the condition inside is true and 0 otherwise.

The predicted class *y (denoted as y hat below)* is determined by selecting the class with the highest accumulated weight:

$$\hat{y} = \arg\max_c w_c$$

By accumulating the weights of neighbors for each class, the algorithm accounts for class imbalances and ensures a fair contribution of minority class instances to the prediction. The combined algorithm is fully described in Algorithm 1.

Algorithm 1 Weighted Mixed K-Nearest Neighbors (WM-KNN)

```
Input: Training dataset with numerical and categorical features, test dataset, $k$ neighbors
Output: Predicted labels for all test data points
Step 1: Calculate weights for each class in the training dataset:
    • Determine class counts using the training labels.
    • Assign a higher weight to the minority class ($w$ = 2.0) and default weight to other
      classes ($w$ = 1.0).
Step 2: For each test data point $x$:
    • Compute distances to all training points.
    • Identify the $k$-nearest neighbors based on the computed distances.
Step 3: Perform weighted voting to determine the predicted label:
    • Accumulate weights for each class $c$ among the $k$-nearest neighbors
    • Assign the class with the highest accumulated weight.
```

```
Step 4: Repeat Steps 2–3 for all test data points.
Step 5: Evaluate the performance of the algorithm using multiple metrics.
```

4 Experimental Results

4.1 Performance Metrics

To evaluate clustering algorithms, we use the following metrics to assess the performance of our model:

- $\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$
- $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$
- $\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$
- $\text{F1-Score} = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

where TP is the true positive rate, TN is the true negative rate, FP is the false positive rate, FN is the false negative rate.

4.2 Results

In all experiments, k = 85 was employed as the square root of the total number of training instances under a mathematical approximation as used in similar papers [2]. This was chosen over the other methods, such as the elbow method or silhouette score, because it is extremely computationally expensive given the number of instances we are testing. We present confusion matrices for all the experiments conducted below as well as Table 2 compares the performance of our best-performing adaptive weighted voting mixed KNN algorithm with Mahalanobis distance against other baseline proposed algorithms across various metrics.

We begin by describing the standard mixed data KNN baseline in Figure 2 using no weighted voting and a standard Euclidean distance calculation. We found an accuracy of 86.28% on the testing set, signaling a competitive baseline. However, we noticed that the class label of 1 (which is approved loans) is the minority class, resulting in worse performance as seen through the precision for loans approved. We aim to improve the overall accuracy as well as minority class classification.

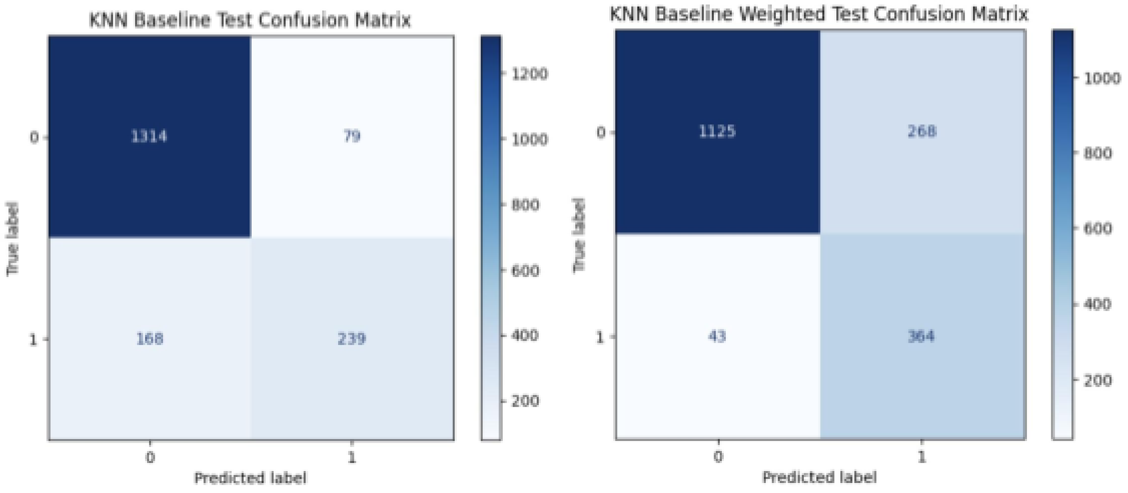

Figure 2: Test Confusion Matrix for Mixed KNN Baseline with Euclidean Distance (left) and Weighted Mixed KNN with Euclidean Distance (right).

Next, we attempted to apply our weighting scheme to our Euclidean distance baseline and achieved an accuracy of 82.72%. We discovered that it decreased the overall performance in terms of accuracy through the loans not approved label (0, majority class) with a much lower precision of 80.76%. It did improve in minority class prediction, however, as seen in a high precision of 89.43% for label 1, which is the loans approved.

Afterward, we experimented with the Mahalanobis distance metric without weighting described in the methodology for the numeric features in Figure 3. Even though the overall accuracy appears to be similar to the baseline using Euclidean distance and no weighting at 86.22% for the testing, we discovered that in this case, the majority class prediction improved, as seen in an improved precision for label 0 of 99.21% compared to the baseline of 94.33%. However, the minority class prediction significantly worsened for testing, as seen in a precision of 41.77% for class label 1. Additionally, the recall for the minority label was 93.92% which is significant compared to the Euclidean baseline.

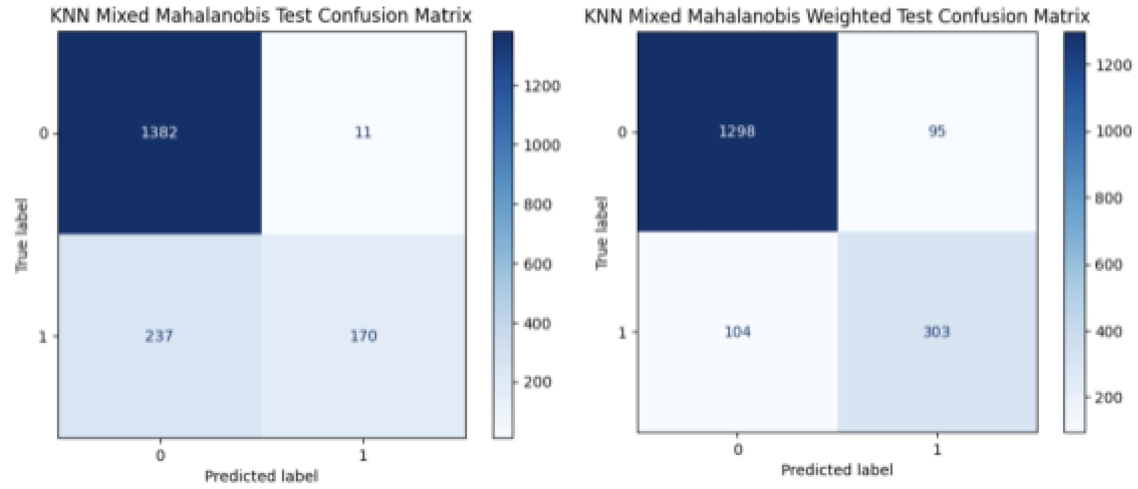

Figure 3: Testing Confusion Matrix for Mixed KNN with Mahalanobis Distance (left) and Weighted Matrix for Mixed KNN with Mahalanobis Distance (right).

Finally, we combined both of our techniques to develop our most robust algorithm, a weighted Mahalanobis distance-based KNN algorithm for mixed data types in Figure 3. Using this algorithm, we achieved our best accuracy of 88.94%, which is 2.66% better than the conventional baseline. Focusing on the other metrics, we noticed that the classification for the majority class prediction (class label 0 or not approved loans) worsened slightly, achieving a precision of 93.18%, however, the incorporation of the weighted voting significantly improved the minority class label prediction with a precision of 72.44%. However, the minority class prediction was not as good as the Euclidean weighting. We also see an improved recall score of the majority class of 92.58%. Accuracy and F1-score, as indicated by the bolded values in Table 2, indicate strong performance of our algorithm overall above multiple competitive baselines. The other metrics were very competitive against all the other algorithms achieving a good balance between majority and minority class prediction.

Table 2: Comparison of Different KNN Variants on Testing Data

Metric	Euclidean	Euclidean Weighted	Mahalanobis	Mahalanobis Weighted
Accuracy (Test %)	86.28	82.72	86.22	88.94
Precision (Test, 0)	94.33	80.76	99.21	93.18
Precision (Test, 1)	58.72	89.43	41.77	74.44
Recall (Test, 0)	88.66	96.32	85.36	92.58
Recall (Test, 1)	75.16	57.59	93.92	76.13
F1-Score (Test, 0)	91.41	87.32	91.77	92.88
F1-Score (Test, 1)	65.93	69.62	57.82	75.28

Previous work has focused a lot on uniform datasets, but our novel algorithm and its novel mixture of techniques can be easily generalized to improve performance in other skewed datasets over a wide range of applications, including financial portfolio optimization, healthcare, and market strategies.

4.3 Discussion

The Euclidean algorithm baseline saw a fairly even distribution of adjacency matrix-derived metrics— tending to devalue, however, most metrics for the minority class. On the other hand, Euclidean Weighted saw a reversal of class-based recall, while seeing much more skewed precision and the overall lowest accuracies. Sole Mahalanobis was somewhat opposite to Euclidean Weighted, reversing the distributions of precision whilst most strongly skewing recall and F1-scores, seeing an accuracy comparable to base Euclidean. Lastly, combining Mahalanobis distance and the mixed weighting technique gave the strongest accuracy by an added 2.66% — with fairly even precision, recall, and F1-score despite a clear inclination favoring the majority class.

On the whole, we noticed the fascinating outcome that while Mahalanobis and mixed weighting individually failed to have much of an effective impact on overall train/test accuracy, their combination yielded far greater results than the standard algorithm on our dataset. This may have been due in part to our specific dataset, which best enabled the conditions for the optimality of these techniques, but also suggests confirmation of our research objective — determining if combining these two techniques would yield superior results.

5 Conclusion

In this work, we introduced the Adaptive Weighted Voting KNN algorithm, a novel approach designed to address the challenges of skewed, mixed-type datasets in classification tasks. By combining Mahalanobis distance for numerical features, a dissimilarity measure for categorical features, and an innovative weighted voting scheme to prioritize minority classes, our algorithm demonstrates significant improvements over conventional methods. We also discovered that standard weighting on Euclidean measurement significantly hinders overall classification performance. Experimental evaluations on a complex loan approval dataset demonstrated 88.94% accuracy which is a 2.66% improvement compared to conventional methods, and enhancing recall for minority class predictions without sacrificing overall accuracy. As a small percentage in improvement scales, it could be worth large amounts of money for companies and individuals. This hybrid approach addresses the demands of skewed (non-uniform) datasets, making it a generalizable solution for heterogeneous real-world applications in finance and beyond.

5.1 Future Work

A current limitation of all KNN algorithms is the number of distance computations increases linearly with the value of *k*. We want to explore a faster version of our weighted mixed data type KNN algorithm to decrease the number of computations while maintaining the same performance as k increases, as inspired by other works [4]. Additionally, we want to explore a way to dynamically find the best k value for large datasets instead of a mathematical approximation.

6 References

[1] Cao Q., La, L., Liu H., & Han, S. (2018). Mixed weighted KNN for imbalanced datasets. International Journal of Performability Engineering, 14(7), 1391. https://doi.org/10.23940/ijpe.18.07.p2.13911400.

[2] Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., & Alhasanat, A. A. (2014). Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. CoRR, abs/1409.0919. http://arxiv.org/abs/1409.0919.

[3] Jainalabidin, N., Amidon, A., Ismail, N., Yusoff, Z., Tajuddin, S. N., & Taib, M. (2022). The knearest neighbor modeling by varying mahalanobis and correlation in distance metric for agarwood oil quality classification. International Journal of Advances in Applied Sciences, 11(3), 242-252. https://doi.org/10.11591/ijaas.v11.i3.pp242-252.

[4] Kim, B. (2017). A fast k-prototypes algorithm using partial distance computation. Symmetry, 9(4). https://doi.org/10.3390/sym9040058.

[5] Lo, T. W. (2024). Loan approval classification dataset. Kaggle. https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.