Let’s Do Some KNNs

Students :

Ishan Ajwani

Logan Bradley

Teacher :

Dr. Yilmaz

Abstract

The K-Nearest Neighbors (KNN) algorithm is a simple classification algorithm that is widely used, and generally performs well on most datasets. However, KNNs struggle on data that is skewed with a class imbalance, noisy, or otherwise non-ideal, severely impacting performance. Through methods such as the Synthetic Minority Oversampling Technique (SMOTE), weighted averaging, weighted distancing, and local density weighting, KNN accuracy was increased nearly 4% on a skewed and noisy dataset, and the different techniques were compared and combined.

Contents

1 Project Goal

The K-Nearest Neighbors algorithm relies on majority voting amongst the nearest neighbors of an instance to determine its classification. However, this algorithm is inherently flawed in that it favors classes with higher representation in the dataset, leading to frequent misclassification of infrequent labels. This problem is particularly prevalent in domains such as medical diagnosis, where a large majority of patients do not test positive, yet false negatives can be extremely detrimental. Our aim is to address this bias by introducing new pre-processing and weighting techniques into the model training process, improving the accuracy of KNN classifiers with respect to false negatives.

Our plan to improve upon the K-Nearest Neighbors algorithm is based on a modified pre processing and weighting approach. There are four changes we aim to test. Firstly, we will utilize the Synthetic Minority Over-sampling Technique (SMOTE) to reduce the skew in non-uniform datasets and increase the frequency of minority classifications, which should strengthen the KNN model. Chawla et. al (2011) found significant improvements by implementing SMOTE, outperforming a method of simply under-sampling the majority class. The traditional KNN algorithm simply does a majority vote of the K nearest neighbors. However, the algorithm can be enriched by considering three additional factors. Firstly, local density weighting can be used: neighbors that are in a denser region are generally stronger indicators of a classification than those in sparsely populated regions. This notion can be utilized by increasing the weightage of neighbors in dense regions to have a larger impact on the final classification. Secondly, weighted averaging can be used to account for large skews. An arbitrary inversely proportional relationship can be constructed between a neighbor's classification's frequency in the dataset and its weightage. For example, a neighbor that is classified as a label which is extremely rare in a dataset will have a higher weightage in the final outcome than one which is classified as a common label. Lastly, weighted distancing can be utilized to further sophisticate the KNN algorithm. By factoring the distance of each neighbor from the instance to be classified, more accurate predictions can be made.

To evaluate what properties make the best KNN classifier, we will use a variety of datasets and differing models. To ensure that the modifications we implement are not decreasing the potency of the classifier on regular datasets, we will test the classifier on both uniform datasets and highly skewed datasets. In addition to varying datasets, we will evaluate the performance of separate improvements by comparing a baseline KNN, a KNN using weighted averaging, a KNN using weighted distancing, a KNN using local density weighting, and a KNN using all the methods combined. Each model will also be trained using standard pre-processing techniques and SMOTE. Evaluation metrics will extend beyond accuracy, as the main focus of this project is ensuring a low false negative rate. Instead, we will utilize confusion matrices and recall scores.

2 Related Work

Imbalanced datasets pose significant challenges to accurate classification using machine learning algorithms, especially KNN. The skew present in an imbalanced dataset can lead to the classifier having a bias for the majority class, leading to frequent misclassification and low accuracy on instances of minority classes. This issue is quite important, especially in fields

where false negatives of the minority class can have significant negative effects, such as a screening or diagnosis for medical issues that could be lethal without rapid treatment. A study by Spelmen & Porkodi (2018) explored this problem and several potential solutions, including techniques such as undersampling and Random Walk Over-Sampling, and finds that there are many possibilities for algorithms that can improve the accuracy of a KNN classification model on skewed datasets.

There has been a variety of related work in this field, studying different improvements that can be made to KNNs. One notable paper is *SMOTE: Synthetic Minority Over-sampling Technique*, in which the SMOTE technique to better construct classifiers from imbalanced datasets is explained (Chawla et al., 2002). Essentially, what SMOTE does is takes a point, chooses a random K-nearest neighbor, and creates a synthetic datapoint somewhere on the line between the two points. This process is repeated a number of times until sufficient synthetic data has been created. By doing this, a classifier can "oversample" the minority class, which has the potential to improve accuracy on skewed datasets.

Further research by Hindi & Shawar (2012) involved bayesian-based instance weighting techniques, in which weights were assigned to instances based on the conditional probability of an instance belonging to its assigned class. They also used a slightly different technique to inversely weight instances that had a high probability of belonging to other classes. These methods both aim to eliminate noise and outliers by giving less weight to unusual instances that have a high probability of being mislabeled. Their results showed small improvements on the majority of datasets with 10% or greater noise, without losing much accuracy on non-noisy datasets.

Other research on KNN weighting, like that done by Azizi (2021) involved weighting the nearest neighbor's votes by Euclidean distance instead of a simple majority vote, and this was found to outperform the traditional KNN algorithm in both classification and regression. Yet another study by Geng et al. (2018) proposed RECOME, which incorporates local density measures into KNN using a method called Relative KNN Kernel Density, or RNKD. This is a very elaborate strategy, but it essentially finds points with the highest local density, creates small clusters with those points, and then merges those clusters based on how closely they are connected. This technique, while not exactly what we are aiming to accomplish, could potentially help KNNs perform better on irregular datasets with varying densities.

3 Dataset and Features

To accurately assess the differing performances of different modifications and preprocessing, the dataset must have a number of different characteristics. For example, it must have a majority class that overshadows the others, some noise that is misclassified or just random, but still have recognizable clusters that make it possible to tell the classes apart. A KNN must be able to perform well enough on it that with modifications, it can be effective, but not well enough that modifications are unnecessary or have no meaningful effect. In order to ensure

the dataset fit all of these criteria, we decided to synthetically generate the dataset using code.

The dataset has 10 attributes and comprises 3 classes: a majority class of 1105 instances, one minority class of 409 instances, and another minority class of 386 instances. Each class is mainly a randomly generated cluster, with the majority class centered at the point (0, 0, 0, 0, 0, 0, 0, 0, 0, 0), the first minority class being centered at (3, 3, 3, 3, 3, 3, 3, 3, 3, 3), and the last class being centered at (-3, -3, -3, -3, -3, -3, -3, -3, -3, -3). The standard deviations are 3, 7, and 7, respectively, making the majority class closely packed and the minority classes looser, resulting in overlap between the classes that a regular KNN struggles to deal with.

Next, 300 instances of noise were generated on a normal distribution with a scale of 10, centered at 0, and assigned to classes randomly. They had a 40% chance of being added to the majority class, and a 30% chance for each minority class. Finally, a train-test split was performed, with 80% of the data going to the train set and 20% going to the test set.

4 Methods

Our methods build upon concepts investigated in prior research, introducing enhancements to improve the performance of KNN classifiers on non-ideal datasets through the incorporation of synthetic oversampling, advanced weighting techniques, and local density estimations.

The first step in our approach involves preprocessing; specifically, the data will be modified using the Synthetic Minority Over-sampling Technique (SMOTE), which balances class distributions by generating synthetic instances for the minority class. Because class frequency imbalance can be a major problem to KNN algorithms, being able to even out the skew can be very helpful. The SMOTE process begins by identifying the k-nearest neighbors of each sample in the minority class using a Euclidean distance metric. Once the neighbors are determined, a random neighbor is selected, and a synthetic instance is generated by linearly interpolating between the sample and its neighbor. Mathematically, this can be expressed as SyntheticInstance = x + delta*(xneighbor - x), where x is the minority-class sample, xneighbor is the chosen neighbor and delta is a random value drawn from a uniform distribution. This formula bears much similarity to the linear slope-intercept form equation. By repeating this process for all points in the train set, SMOTE can create synthetic data points that increase the minority class's presence in the data set and improve the model's ability to generalize without discarding majority-class data.

In addition to preprocessing, we propose using a weighted averaging mechanism to address the disproportionate influence of majority-class neighbors. Whereas in normal KNNs each neighbor is equally weighted, with this method we assign a custom weight to each neighbor with respect to a particular sample. For a given neighbor xi, the weight wi would be calculated as wi = 1/f(ci), where f(ci) is the frequency of the class of the given

neighbor. Instead of simply counting the classifications of each neighbor, a summation of the weighted votes would be taken instead to come to a conclusion. Specifically, the predicted class chat would be given by chat = argmaxcsumi=1→k wi*1(ci=c), where 1(ci=c) is a indicator function. By weighting minority-class neighbors over majority-class neighbors, weighted averaging could reduce bias in favor of the majority and improve the KNN model's performance.

Another method of assigning weightage that we want to examine is weighted distancing, which adjusts the influence of a neighbor given its proximity to the test instance. With this method, neighbors closer to a given sample will be assigned higher weights, ensuring that distant neighbors with irrelevant information exert less influence. The weight wi for a neighbor xi is calculated as wi = 1/dpi, where the weight of the neighbor will decay as distance increases. Class prediction will be performed through a summation of these weights to reduce noise from distant points and improve accuracy.

The final method that can be used to refine the classification process is a local density weighting system, which considers the density of the region surrounding each neighbor. For each neighbor xi, the local density pi is estimated by counting the number of points within a radius r: pi = E j = 1 n 1(d(xi, xj) <= r). Neighbors located in denser regions are given higher weightage, as they are more likely to represent reliable and representative patterns in the data. The final weight for a neighbor is then normalized based on the total density of all k-nearest neighbors. By integrating local density into the weighting process, this method minimizes the influence of sparse and unrepresentative outliers, further enhancing the robustness of the classification.

To create a comprehensive algorithm, we plan to combine the methods described above into a single weightage system. The composite weight wi for each neighbor will utilize frequency weighting, distance weighting, and density weighting, and can be expressed as wi = (pi)/(f(ci)*dip). The final prediction for a given sample will be determined by summing the composite weights for the K nearest neighbors. This integration will create a more adaptive model capable of handling diverse datasets, including those with large class imbalances and noise.

To ensure optimal performance, hyperparameters will need to be optimized. We will use a grid search and random search technique to tune the important parameters, such as the number of neighbors k, the distance weight decay parameter p, and the radius used in the local density estimation r. Using cross-validation, we will then identify the parameter combinations that yield the highest classification accuracy.

5 Experiments/Results/Discussion

To test the K-Nearest Neighbors model and its variations, we used two conditions: with and without Synthetic Minority Over-sampling Technique (SMOTE). The performance was evaluated through the accuracy measure and confusion matrices across the three classes (0, 1, and 2).

(% accuracy)	Non-SMOTE	SMOTE
Regular KNN	73.42%	77.11%
Weighted Averaging	76.58%	77.11%
Weighted Distancing	73.68%	77.63%
Local Density Weighting	74.47%	77.37%
All Methods Combined	75.26%	76.58%

Under *non-SMOTE* conditions, the baseline KNN model scored an accuracy of 73.42%. When observing the confusion matrix, class 0 had the highest correct predictions (203), while classes 1 and 2 were misclassified at higher rates, with 28 and 34 instances being labeled as class 0 incorrectly. Percentage wise, the majority class (0) had a percent accuracy of 91.86%, and the minority classes (1 and 2) had accuracies of 38.96% and 56.10%, respectively. The weighted averaging technique improved this accuracy to 76.58%, with large gains in correctly predicting class 1: 41 correct predictions vs. 30 in the baseline model, bringing the class 1 accuracy up to 53.25%. Weighted distancing provided, however, only a small improvement over the baseline model, with an accuracy of 73.68%, as classes 1 and 2 continued to be misclassified. Local density weighting achieved a better accuracy of 74.47%, improving upon the classification of the class 0 instances. The combined approach, using all three methods in the weighting function, resulted in an accuracy of 75.26%, which was not bad, but still outperformed by the weighted averaging technique.

Under *SMOTE conditions,* all models performed significantly better compared to the non-SMOTE dataset. Both the baseline KNN and weighted averaging achieved 77.11% accuracy, and did much better with predicting minority classes like 1 and 2, with accuracies of 54.55% on class 1 and 64.63% on class 2. The weighted distancing approach achieved the highest accuracy of the batch, with 77.63% correct, correctly predicting 43 instances of class 1, which is a class 1 accuracy of 55.84%. Local density weighting followed with an accuracy of 77.37%, having a better ability to predict minority-class instances while maintaining a low misclassification rate for the majority class. Surprisingly, however, the accuracy for the combined-metric approach dropped to 76.58%, indicating that using too many techniques used at once could potentially be detrimental to the accuracy of the algorithm.

Overall, the confusion matrices consistently showed challenges with predicting classes 1 and 2, the minority classes, and significantly less issue with class 0, the majority class. However, SMOTE effectively reduced the bias against minority classes by generating synthetic data, which led to an overall increase in accuracy and reduced false negatives.

6 Conclusion/Future Work

The results of this study shows that KNNs certainly have issues with noisy or skewed data, and that modifications can be very effective in mitigating the negative effects of these issues. Using SMOTE was the single most effective technique, likely due to its complexity rather than just a simple change to weighting. In fact, after using SMOTE, weighting techniques only had a marginal effect on the accuracy of the model, suggesting that SMOTE renders them unnecessary.

Even with all of our techniques, the best percent accuracy achieved was only 77.63%, and we did not achieve any minority class accuracies over 65%. For future work, it may be fruitful to investigate more advanced techniques to improve the accuracy further. Alternatively, a more exhaustive search of K and R values could result in better accuracy, though would require significantly more time or compute resources. Finally, testing each weighting and preprocessing method on a variety of datasets, each with different skew and noise, could result in a more comprehensive picture of the strengths and weaknesses of each method.

7 Contributions

Building proposal: Mostly Ishan

Creating the dataset: Logan & Ishan

Writing Code: Mostly Logan

Interpreting Code: Logan & Ishan

Results and Analysis: Mostly Ishan

Building Final Report: Logan & Ishan

Presentation: Logan & Ishan

8 References/Bibliography

Azizi, S. (2021). Improving the KNN algorithm by using weighted Euclidean distance. *Journal of Software Engineering & Intelligent Systems*, *6*(1), 47-52. https://d1wqtxts1xzle7.cloudfront.net/68370304/V6N1_5-libre.pdf?1627549763=&response-content-disposition=inline%3B+filename%3DIMPROVING_THE_KNN_ALGORITHM_BY_USING_WEI.pdf&Expires=1736913038&Signature=Yk5Mr~AlZnpI0~XRdXX2~KMr

6Cuk8jV6mLqrLnSdYx8NhWQDdyl5Y6g99HNQTcpdV4jWYgYvSlopm97ZzYV7pr9r79i0RtVa2fyO8rIvamusN4pi2ud-KmHJTmw2eF9RdoiMF6eg91QQOttairJ1rAdOQ0S3ABI3engavrqlHAjdJiDQkdPbCSyvQWu1aDeJQaeQ7uEdZvzZu-9CdNoDUCOzidQ~eDzFVBy-hAff1K1Q9xmSi6upfC8GaEjwEtYtxq9JzuAbJcN2Sr~sPaRnUL2Ci-JEj59xfwfbKsjnOqWqyfsnzQU6R~0w27DY032VRPZUimR9MIQpnGieAmeGYQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002, June). *SMOTE: Synthetic minority over-sampling technique*. arXiv. https://doi.org/10.48550/arXiv.1106.1813

Geng, Y., Li, Q., Zheng, R., Zhuang, F., He, R., & Xiong, N. (2018, April). *RECOME: A new density-based clustering algorithm using relative KNN kernel density*. ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0020025518300215

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hindi, K. E., & Shawar, B. A. (2012, February). *Bayesian-Based Instance Weighting Techniques for Instance-Based Learners*. ResearchGate. https://www.researchgate.net/profile/Bayan-Shawar/publication/262165219_Bayesian-Based_Instance_Weighting_Techniques_for_Instance-Based_Learners/links/56b88a2b08ae5ad3605f3c05/Bayesian-Based-Instance-Weighting-Techniques-for-Instance-Based-Learners.pdf

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research, 18(17)*, 1–5. http://jmlr.org/papers/v18/16-365.html

McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56). https://doi.org/10.25080/Majora-92bf1922-00a

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*(Oct), 2825-2830.

Spelmen, V. S., & Porkodi, R. (2018). *A Review on Handling Imbalanced Data*. ResearchGate. https://www.researchgate.net/profile/Porkodi-Drr/publication/329584489_A_Review_on_Handling_Imbalanced_Data/links/5c10a53ba6fdcc494feda0d0/A-Review-on-Handling-Imbalanced-Data.pdf