Nonparametric Classification and Regression with KNN

Turner Haugen TURNER.HAUGEN712@GMAIL.COM
Machine Learning Student
Montana State University
Bozeman, MT 59718, USA

Jackson Thorn JTHORN2448@GMAIL.COM
Machine Learning Student
Montana State University
Bozeman, MT 59718, USA

Abstract

This paper aims to provide a custom implementation of three separate non-parametric algorithms on a variety of datasets, provided by the UCI Machine Learning Repository. The three algorithms we created are K-Nearest Neighbors, Edited K-Nearest Neighbors, and K-Means Clustering. These methods use distance metrics to determine outcomes, so our methods handle continuous variables normally and discrete variables through one-hot encoding. 3 of the datasets we are exploring are classification problems while the other 3 are regression problems, so our models are built to handle both situations. Accuracy is used for evaluating classification while mean square error is used for regression.

1 Introduction

K-Nearest Neighbors (KNN) is a simple and widely popular instance-based machine learning technique that has the ability to model complex data patterns without making any assumptions about the data's underlying distributions. It works under the idea that instances close to each other in the feature space are likely to share similar target values, though depending on the task these targets are determined differently. For classification, a plurality vote between the nearest neighbors predicts the target label while for regression a weighted average of neighboring points predicts the target label through the use of a Gaussian RBF kernel.

As powerful as KNN can be when evaluated on non-parametric datasets, its performance strongly depends on the choice of k and the Gaussian kernel. If k is too small, the model can over-fit to noise while if k is too large the model can over-smooth and fail to capture any structure. As for regression, the choice of sigma in the Gaussian kernel determines the weight given to neighboring points based on their distances which can also influence the model's performance.

Additionally, we explore an implementation for K-Means Clustering with the primary goal of minimizing variance within each cluster. The data is partitioned into k distinct clusters by assigning points to its nearest cluster centroid and simultaneously updating the centroid

positions. Again, the choice of k here is key: a poor choice can lead to misleading clusters, and K-Means is very sensitive to outliers.

We aim to develop custom implementations of both KNN and K-Means Clustering across six datasets: three created for regression and three for classification. To achieve the best possible accuracy, hyperparameters k and sigma will be tuned through grid-search cross-validation. Additionally, Edited KNN will be developed with the goal of improving classification accuracy through the removal of noisy data points. As for K-Means, we analyze how the choice of k clusters impacts the model's ability to find groupings within the data.

We hypothesize that the choice of k will significantly impact KNN's performance and will be different across the six datasets. For regression, the sigma parameter influences how the RBF kernel weights neighbors where increasing values provide a more generalized prediction. As an extension, Edited KNN removes noisy points with the goal of improving accuracy, especially on larger datasets. Since the goal of this method is to do K-Nearest neighbors on a cleaner dataset, it should perform around the same or better than K-nearest without editing. For K-Means Clustering, the choice of k is crucial for correctly identifying patterns in the underlying data. Due to K-Means Clustering assuming circular clusters and being fairly restrictive on regression datasets (since predictions happen with $k = 1$), datasets that involve outliers (like forestfires) may have significantly worse performance. By tuning hyperparameters through grid-search cross-validation, we intend to produce accurate target predictions for KNN and accurate clusters for K-Means clustering.

2 Design

We approached this problem by modifying the framework of the Naive Bayes Classification project. We first modified the data handler to manage the new datasets as well as the introduction of a better data imputation method and stratification with fold generation. K Nearest Neighbors was the first algorithm we tested as the other 2 methods used K Nearest Neighbors after narrowing down the number of points present. K Nearest was identified as requiring a distance function and a method of predicting outcomes. For regression, we used an RBF kernel to get a weighted average, while classification uses a basic plurality vote. Edited K Nearest neighbor needs rules to determine why a point should be removed, what defines a correct/incorrect point, and when to stop removing points. K-Means Clustering needs to assign every point to a given cluster as well as assign new centroids based on the current centroid assignments.

2.1 Data Imputation

Since some datasets had missing values, we needed to handle them. Different datasets noted what missing values were in different ways. These were handled differently per dataset. Most stored them as NaNs, but the Wisconsin Breast Cancer dataset stored them as ?. From there, data imputation was handled depending on the type of data each column held. For categorical columns, each NaN was set to the mode of the column it is in. For numeric columns, each NaN was set to the mean of the column it was in. Categorical columns that were written down as numeric columns were properly treated as categorical.

2.2 Fold Stratification

Cross-validation can sometimes result in folds that randomly have a much higher or lower proportion of a specific target value than the overall dataset. Stratification is put in place to ensure randomly generated folds have target values that match the proportion of target values in the full dataset. For regression problems, the proportion of each classification value will match their proportion in the full dataset. For regression problems, data is ordered by their target values, and fold assignments are done procedurally. This will result in folds that consist of an even number of smaller and larger target values.

2.3 One-Hot Encoding

Calculating the Euclidean distance between 2 categorical values does not make sense. One-hot encoding is used to convert categorical values into something that a numeric distance function can understand. For every categorical column, a new column is made for every possible value that column can be. For a given point in the dataset, the value for this new column is 1 if the point is equal to that column's corresponding value from the original categorical column and 0 otherwise. The original column is removed after all new columns are made to ensure that no categorically stored columns remain.

2.4 Grid Search

All 3 algorithms we are interested in looking into contain at least one hyperparameter. Before tuning, 10% of the dataset was designated for tuning while the remaining 90% of the data was split into 10 folds. For every unique hyperparameter, we generated a range of potential values that it could be based on what seems reasonable. Grid search checks through every possible combination of each hyperparameter using 9/10 folds as the training data and the designated tuning data as the test data. This is done 10 times with each possible combination of 9 folds. The average value from the 10 different loss function values is used to compare the current set of hyperparameters with the set of hyperparameters with the current best average loss function value. The better performing set of hyperparameters is kept. Classification problems will use accuracy, so a better accuracy is a higher accuracy. Regression problems will use mean square error, so a better MSE is a lower MSE.

2.5 Determining Nearest Neighbors

To determine the nearest neighbors for each point in the testing set, the algorithm calculates the distance from each test point to each training point. Distance is calculated with the Euclidean distance, which is the root sum of all individual column-wise squared distances. These distances are sorted so that the smallest distances can be easily retrieved.

2.6 Determining Predictions

Once the k-nearest neighbors of a given test point are determined, calculating predictions is done differently depending on whether regression or classification is being used. If classification is used, then whatever target class appears the most in a test point's nearest neighbors is the prediction for that point. If regression is used, a weighted average of a test point's

nearest neighbors is used. The weights for this test point is determined by the RBF kernel, which is:

$$K(x, x') = e^{\frac{-||x-x'||^2}{2\sigma^2}}$$

where $||x - x'||^2$ is the squared Euclidean distance between 2 given points, x and x', and σ is a hyperparameter representing the variance of the Gaussian curve used to determine weights. Smaller values of σ will result in points that are further apart from each other having much smaller weights. σ is a hyperparameter that must be tuned whenever we are using regression.

2.7 Edited K-Nearest Neighbors

Edited k-nearest neighbors is a method of narrowing down a specific dataset to a core set of important points. K-nearest neighbor is used once editing is complete.

Editing runs through every point inside of the training dataset, X. A K-nearest neighbor model is fit to a dataset containing every point except for the current point, with k = 1. If the prediction on the current point is not equal to the actual value of that point, it is removed. This process is repeated until no more points are removed.

For regression data, ϵ is used to determine what makes for a "correct" observation. ϵ is the percent of the range of the response variable that is considered a "correct" observation. For example, if $\epsilon = 0.1$, then values that are 5% of the response variable's overall range on either side are considered "correct" observations. This is another hyperparameter that is tuned with grid search.

2.8 K-Means Clustering

K-means clustering is similar to edited K-nearest neighbors, as both do not generate predictions on their own and are instead used to narrow down predictions.

K-means starts by assigning the top k points as the current centroids. All points are assigned to a cluster by viewing which cluster they are the closest to, with closeness also being determined by Euclidean distance. The mean of each newly assigned cluster is taken once all cluster assignments are finished. From there, the point closest to the mean is used as the new centroid, and all cluster assignments are reassigned based on the new centroids. This process doesn't stop until cluster assignments do not change.

Once all clusters are assigned, we use k-nearest neighbors with k = 1 to determine any predictions. The predictions are based on the target value of whatever points ended up being the centroids.

3 Results

We evaluate classification data using accuracy per each cross-validation fold, which is aggregated into a final average accuracy. Regression data is evaluated based on mean-squared error. The results are on a set seed of 42 when dataset shuffling occurs. Below are the results per each dataset, per each algorithm:

K-Nearest Neighbors

Wisconsin Breast Cancer
Final average accuracy: 0.9588, best k: 3, tuning set score: 0.9971

Glass:
Final average accuracy: 0.6072, best k: 17, tuning set score: 0.7889

Soybean-small:
Final average accuracy: 1.0, best k: 1, tuning set score: 1.0

Abalone:
Final average MSE: 16.8216, best k: 1, best sigma: 0.5, tuning set score: 6.2019

Forestfires:
Final average MSE: 7406.8424, best k: 1, best sigma: 0.5, tuning set score: 11061.0255

Machine:
Final average MSE: 47797.7865, best k: 1, best sigma: 1, tuning set score: 131.0550

KNN's performance across the classification datasets varies somewhat. Aside from perfect accuracy on soybean-small, Wisconsin breast cancer performs the best, with a final average accuracy of 95.88%. The best k was found to be 3, and the high accuracy on the tuning set of 99.71% suggests the model is well-tuned. This dataset has the most data points as well. It is interesting to mention that the best k for the glass dataset was found to be as high as 17, suggesting that the glass dataset is noisy. The perfect accuracy on soybean-small comes from each fold only consisting of 4-5 observations, meaning random chance can cause individual test fold accuracies to vary from 0.75/0.8 to 1. These massive swings mean an accuracy of 1 likely does not mean much.

As for regression datasets, a clear trend emerges: the best k value is 1 for each run. Additionally, the sigma values are small, suggesting that the model heavily relies on local information when making predictions. Abalone performs the best with a low MSE, though both forestfires and machine have extreme average MSE that decrease substantially on the tuning set.

Edited K-Nearest Neighbors

Wisconsin Breast Cancer:
Final average accuracy: 0.9667, best k: 1, tuning set score: 0.9855

Glass:
Final average accuracy: 0.629, best k: 1, tuning set score: 0.7056

Soybean-small:
Final average accuracy: 1.0, best k: 1, tuning set score: 1.0

Abalone:
Final average MSE: ???, best k: ???, best sigma: ???, tuning set score: ???

Forestfires:
Final average MSE: 4450.7578, best k: 13, best sigma: 2.5, best epsilon: 0.4, tuning set score: 772.3977

Machine:
Final average MSE: 38511.2721, best k: 1, best sigma: 1, tuning set score: 4933.9758

Edited KNN's performance across the classification datasets is strong for the soybean and breast cancer datasets, but not on the glass dataset. Soybean likely did well due to the very small size of each fold (5 observations) and the breast cancer dataset only has 2 possible outcomes making predictions a 50/50 split. The glass dataset has 6 different outcomes making it perform a bit worse.

As for regression datasets, due to the presence of 3 hyperparameters to tune (k for the number of points to pick after editing, σ for RBF kernel weights), and ϵ), this algorithm takes a very long amount of time to tune. We selected 19 possible k values, 5 possible σ values, and 5 possible ϵ values, resulting in 475 different combinations for grid search to explore. The abalone dataset takes especially long to tune, with each combination taking over 1 minute to run it would take more than 7 hours to run, so we did not end up running it. Additionally, there is a very large gap between the tuning set score and the final average set score, which indicates overfitting to the tuning dataset.

K-Means Clustering

Wisconsin Breast Cancer:
Final average accuracy: 0.6254, best k: 3, tuning set score: 0.9696

Glass:
Final average accuracy: 0.5428, best k: 15, tuning set score: 0.6222

Soybean-small:
Final average accuracy: 0.975, best k: 4, tuning set score: 1.0

Abalone:
Final average MSE: 11.3832, best k: 3 tuning set score: 3.6567

Forestfires:
Final average MSE: 4524.5018, best k: 5, tuning set score: 781.1439

Machine:
Final average MSE: 9900.0123, best k: 16, tuning set score: 749.1755

The results across the classification datasets vary for K-Means Clustering. For both the Wisconsin breast cancer and glass datasets, it struggled to find clusters that match classification labels. Though, the model performed very well on soybean-small with a final accuracy of 97.5%. The best k values varied across all three datasets.

As for the regression datasets, while the algorithm did find clusters within the data these clusters do not appear to always be aligned with the target values - resulting in a high MSE. There is a very large gap between the final average MSE and tuning set score that suggests overfitting during training.

4 Discussion

Our version of regular KNN provided strong results on the classification data with generally strong accuracy rates. Due to the heavy reliance on local information, KNN does not prove to be the best option to use on regression datasets.

Our version of edited KNN proved to be worse than we expected. On all 3 classification datasets, results were slightly worse on edited KNN compared to non-edited KNN. On regression datasets, results proved to be mixed. Due to the addition of ϵ as a third hyperparameter, this takes drastically longer to fit than regular KNN. Overfitting still proved to be a significant issue with regression datasets.

Our version of K-means clustering performed similarly to K-nearest neighbor, also likely because both K-means clustering relies on K-nearest neighbor on making predictions. It works pretty well on classification problems, but not as well as K-nearest neighbor. Regression datasets also show signs of overfitting, just like all of the other methods.

Exploring these nonparametric methods strengthened our understanding of expanding more basic machine learning algorithms. Being able to use algorithms that have less assumptions is really powerful on where you can use them. What limits how we can use these methods is that all 3 of these nonparametric methods perform better on classification datasets compared to regression methods. Due to the introduction of more hyperparameters and potential overfitting, these do not seem like an ideal fit for use on regression methods. The most struggling part with this project was getting the stratification and grid search working together, as we had to tweak them multiple times after finding many smaller errors. In conclusion, we found the opportunity to work with a wider variety of methods interesting, and getting an introduction to building regression methods with this project makes us look forward to learning about models that can handle regression much more effectively.

5 Appendix

Jackson wrote the following:

- Stratification in cross-validation
- Hyperparameter tuning
- Data imputing
- One-hot encoding
- K-nearest neighbors
- Primary code flow
- Introduction + results of this paper
- Video script + recording

Turner wrote the following:

- Making the model usable with all datasets
- Edited K-nearest neighbor
- K-means clustering
- Cleaning up the code
- Abstract, design, discussion, and appendix of this paper