Programming Project 2: K-Nearest Neighbors Algorithms for Nonparametric Classification and Regression

Ricca D. Callis

June 20, 2020

Abstract

Supervised Machine Learning Algorithms require structured (or labeled) input data in order to learn a function between inputs and outputs. Supervised Machine Learning Algorithms can be applied to parametric and nonparametric classification or regression problems. This project sought to implement the nonparametric Supervised Machine Learning Algorithm, *k*-Nearest Neighbors. Two *k*-nearest neighbor classifiers and two *k*-nearest neighbor regressors were applied to input data obtained from the UCI Machine Learning Repository.

Introduction

Machine learning is a predictive modeling technique which estimates (i.e., learns) a function (*f*) that maps input variables (X) to output variables (Y), $Y = f(X)$. Input variables are typically a vector of features or attributes, each instance of which is paired to a class or output $\{ (x_i,\ y_i) \}$. The goal of predictive modeling is to take new observations of (x_i) and predict their associated label or outcome (y_i). Thus, predictive machine learning algorithms learn their target mapping function from training data: they create subsets of data (a training subset and a testing subset), estimate $y = f(x)\ on\ X, Y$ and assumes the same $f(x)$ generalizes (i.e., works on unseen X', Y'). The structure of the input data determines the type of learning. For Supervised Machine Learning Algorithms, the input data is labeled. $(X = x_i)$ is a vector of binary, categorical, or real-valued features/attributes and $(Y = y_i)$ is either a class label or a real number. The structure of the Supervised Learning Algorithm depends on the prediction's data type: regression for real numbers, classification for categorical variables, or parse trees for complex objects.

Machine Learning Algorithms can be further delineated based on the assumptions they make regarding the mapping function. Parametric Machine Learning Algorithms assume a form for the function (e.g., Russell & Norvig, 2010). This form summarizes data with a set of parameters of fixed size, yielding an underlying distribution. Assuming the functional form greatly simplifies the learning process. The algorithm can simply estimate (or learn) the assumed underlying distribution's coefficients from the training data. Examples of parametric machine learning algorithms include: logistic regression, linear discriminant analysis, and Naïve Bayes. Although parametric machine learning algorithms may be faster, require less data, and are typically easier to understand and interpret, they are also constrained to the specified form, making them limited in complexity and may not always match the underlying mapping function.

Nonparametric Machine Learning Algorithms, on the other hand, do not make strong assumptions about the form of the mapping function. Nonparametric methods seek to best fit the training data in constructing the mapping function while still maintaining an ability to generalize to unseen data. The k-Nearest Neighbors Algorithm, for example, makes predictions based on the k-most similar training patterns for a new data instance. Rather than assuming a specified form for the mapping function, this algorithm merely assumes that close patterns are more likely to have a similar output. Thus, Nonparametric Machine Learning Algorithms offer flexibility, power, and performance in predictive modeling but are limited in that they typically require more data, are slower to train, and are at a higher risk of overfitting data (e.g., Russell & Norvig, 2010).

This project provided students enrolled in an Introduction to Machine Learning course (605.649.83.SU20), at Johns Hopkins University, the opportunity to implement two variations of a nonparametric supervised machine learning algorithm – *k*-Nearest Neighbors (KNN). Here, *KNN* was adapted for classification and regression problems. Additionally, two variants were introduced to improve cost and efficiency by reducing the set of points included in calculations.

Hypotheses

The *k*-Nearest Neighbors Algorithm (KNN) is a nonparametric density estimation which treats all features equally. As a result, it is hypothesized that predictions will have reduced accuracy in high dimensional spaces. For data sets with clearly and reasonably separated classes, the standard KNN should make highly accurate predictions. An Edited KNN will likely yield more accurate predictions for noisy data because points that lie in the space of a different class will be deleted from the prediction set and irrelevant features will be eliminated. Both Edited and Condensed KNN will likely take less time to make predictions, because they each calculate fewer cases whereas the standard KNN must calculate the distance across all points in the training set.

Algorithms and Experimental Methods

k-Nearest Neighbor Learning

KNN is an instance-based learning method (e.g., Mitchell, 2013). In instance-based learning, training examples are stored and generalization beyond those examples is postponed until a new instance must be classified. Thus, each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance.

KNN assumes that instances can be represented as points in the *n*-dimensional space $Å^n$. The nearest neighbors are defined within the standard Euclidean distance. Let an instance (x_i) be described by the feature vector $\langle X_1(x), X_2(x), \ldots, X_n(x), \rangle$, where $X_r(x)$ denotes the value of the *r*th attribute of instance (x). The distance between two instances (x_i) and (x_j) is defined as $d(x_i, x_j)$, where $d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} [X_r(x_i) - X_r(x_j)]^2}$. Thus, the Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (x_i) across all input attributes *r*. As a result, the smaller the Euclidean distance value, the more similar two instances will be. A value of 0 means that there is no difference between two instances.

Neighbors for each new data instance are the k-closest instances, as defined by the Euclidean distance. In practical application, the KNN algorithm will first calculate the Euclidean distance between each record in the dataset to the new data instance. These distances will be stored in a list, which will then be sorted in ascending order. The top k-instances will be selected as neighbors.

KNN can learn both discrete-valued target functions or real-valued target functions (e.g., Mitchell, 2013).

KNN Learning for Discrete-Valued Target Functions: Classification

Mitchell (2013) explains that KNN Algorithms used for classification problems have discrete-valued target functions in the form of $f\colon \mathring{A}^n \rightarrow V$, where V is the finite set $\{v_1, \dots, v_s\}$. During training, each training example $\langle x, f(x) \rangle$ is added to a list of training examples. During classification, given a query instance x_q, the algorithm calculates the distances of each instance from x_q. We let $x_1, \dots, x_k$ denote the k instances from the list of training examples that are nearest to x_q and the algorithm returns

$$\hat{f}(x_q) \leftarrow \begin{matrix} argmax \\ v \in V \end{matrix} \sum_{i=1}^{k} \delta\big(v, f(x_i)\big)$$

where

$$\delta(a,b) = \begin{cases} 1 & if\ a = b \\ 0 & otherwise \end{cases}$$

KNN Learning for Continuous-Valued Target Functions: Regression

KNN Algorithms used for classification problems have discrete-valued target functions in the form of $f\colon \mathring{A}^n \rightarrow \mathring{A}$. During training, each training example $\langle x, f(x) \rangle$ is added to a list of training examples. During classification, given a query instance x_q, the algorithm calculates the distances of each instance from x_q. We let $x_1, \dots, x_k$ denote the k instances from the list of training examples that are nearest to x_q and the algorithm returns

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Rather than calculating the most common value k (as was the case in the discrete-valued target function), this algorithm calculates the mean value of the k-nearest training examples (e.g., Mitchell, 2013).

Edited KNN

Recall that for a standard KNN, at the time of a new query, n-distance calculations must be made. For large data sets, this is time consuming. To reduce error rate and computational cost, Edited KNN creates a smaller set of training points by eliminating training instances that are not near their class centers (e.g., Bhatia et al, 2010). This process is similar to a backward elimination procedure. At the time of query, each training instance (x_i) is iteratively classified using all other data points. If the prediction is not equal to the actual target value (i.e.,$if\ \widehat{y_i} \neq y_i$), then (x_i) is dropped from the training set. This iterative process continues until no further changes are made or until performance on the validation set decreases.

Condensed KNN

Similar to Edited KNN, Condensed KNN reduces computational cost and improves performance by constructing a subset of examples which are able to correctly classify the original data (e.g., Bhatia et al., 2010). Thus, at prediction time, the class of the single nearest neighbor is used (i.e., k = 1).

Data Sets

This analysis was conducted on 4 data sets, each obtained from the UCI Machine Learning Repository:

(1) Ecoli Data Set

(2) Image Segmentation Data Set

(3) Computer Hardware Data Set

(4) Forest Fires Data Set

For each data set, descriptive statistics were calculated for all features and for each feature grouped by class label. To ensure proper scale, each dataset was standardized such that all columns were centered at zero. For the first two data sets (i.e., Ecoli and Image Segmentation), a KNN-classifier was utilized. For the last two data sets (i.e., Computer Hardware and Forest Fires), a KNN-regressor was utilized. All discrete-valued inputs were one-hot encoded. For each data set, 5-fold cross-validation was performed. GridSearch was utilized to find the value of *k* which yielded the best performance, as defined by

$$k^* = \frac{argmax\ f\overline{(k)}}{1 \leq k \leq 5}$$

For discrete-valued target functions (i.e., classification problems):

$$f(k) = accuracy\ (i.e., porportion\ of\ correctly\ labeled\ instances)$$

For continuous-valued target functions (i.e., regression problems):

$$f(k) = -RMSE\ (i.e., negative\ square\ root\ mean\ squared\ error)$$

Finally, all regression problems were solved by taking the mean across the target values of the *k*-neighbors.

Ecoli Data Set

Data Description: Classifies the localization site of a protein, based on 8 feature attributes: Sequence Name, mcg, gvh, lip, chg, aac, alm1, and alm2 (Nakai, 1996). The class is the localization site and included the following labels: cp (cytoplasm), im (inner membrane without signal sequence), pp (perisplasm), imU (inner membrane), om (outer membrane), omL (outer membrane lipoprotein), imL (inner membrane lipoprotein), and imS (inner membrane). This is a multivariate data set with 336 instances, where each attribute's instance is represented as a continuous-value float and each class label is represented as a discrete-value integer using one-hot encoding.

Data Cleaning & Transformation: There was no missing data to handle. The attribute id was also dropped from the data set, as it represented a unique identifier that would not serve to teach class attributes.

Exploratory Data Analysis: There were 8 class labels with the following assignment frequencies: 143 cp, 77 im, 52 pp, 35imU, 20 om, 5 omL, 2 imL, 2 imL (see Figure 1). Each feature was also described by class (descriptive statistics included mean, standard deviation, minimum, maximum, range, 1st quartile, median, and 3rd quartile) and were plotted using a box-plot (See for example Figure 2). The means for each feature by class can be observed in Table 1.

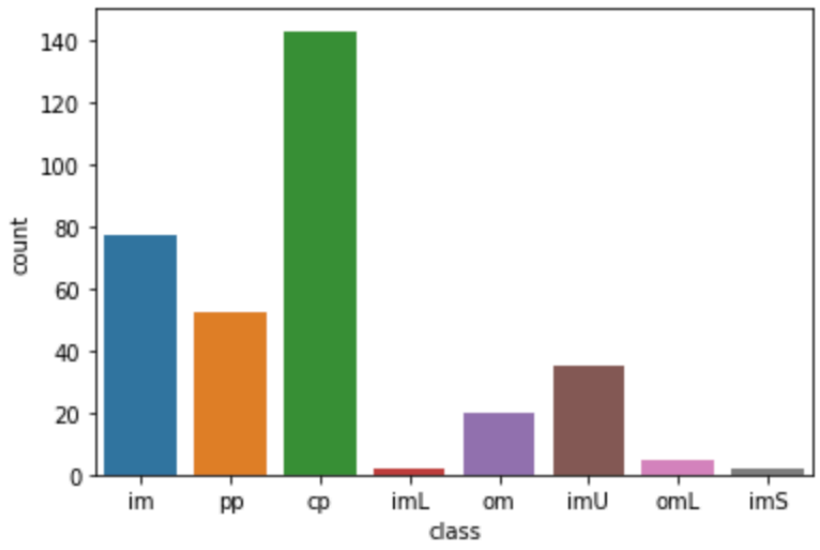

FIGURE 1. Ecoli data set plot showing the actual raw count values for Ecoli localization site classification.

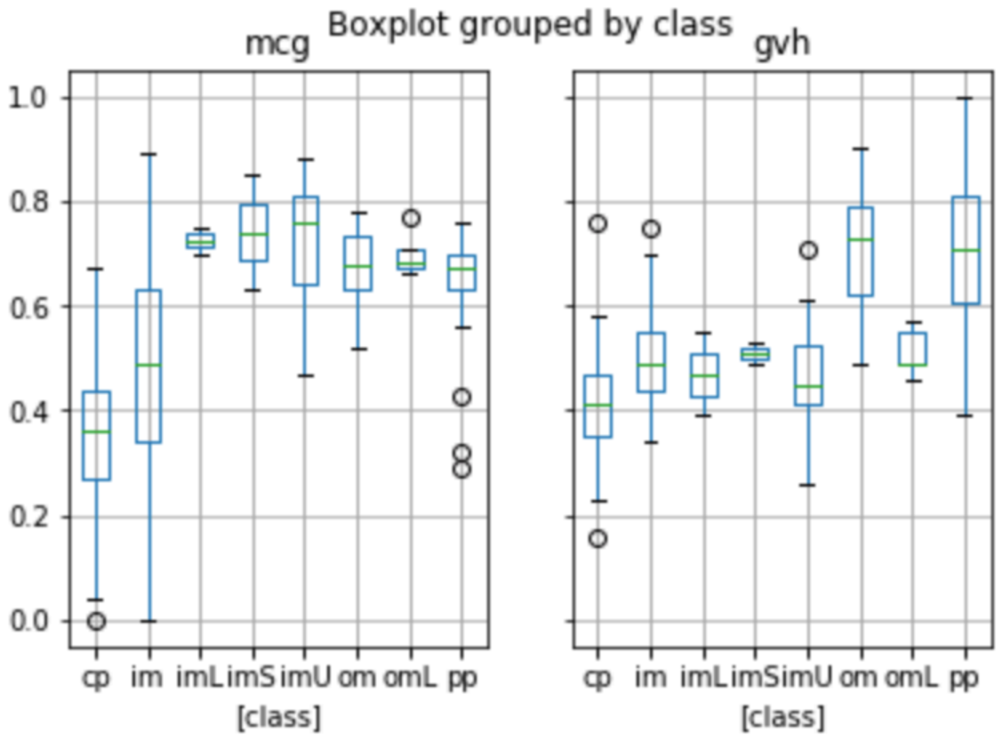

FIGURE 2. Ecoli data set boxplot showing attribute grouped by class. Above example depicts the features mcg and gvh by class label assignment.

Class Label	**Attributes/Features**						
	mcg	**gvh**	**lip**	**chg**	**aac**	**alm1**	**alm2**
cp	0.363566	0.409720	0.480000	0.50	0.454476	0.312657	0.395245
im	0.478442	0.496623	0.486753	0.50	0.536104	0.757532	0.730390
imL	0.725000	0.470000	1.000000	0.75	0.455000	0.645000	0.570000
imS	0.740000	0.510000	0.480000	0.50	0.535000	0.640000	0.570000
imU	0.726000	0.458857	0.494857	0.50	0.558857	0.744000	0.748000
om	0.672500	0.710000	0.506000	0.50	0.739000	0.462500	0.3055
omL	0.698000	0.512000	1.000000	0.50	0.542000	0.564000	0.222000
pp	0.652115	0.699808	0.480000	0.50	0.436731	0.468077	0.374423

TABLE 1. Mean values of each feature by Ecoli localization site in the Ecoli Data Set.

Image Segmentation Set

Data Description: Classifies part of an image based on a given pixel, based on 19 feature attributes of pixels: region-centroid-col, region-centroid-row, region-pixel-count, short-line-density-5, short-line-density-2, vedge-mean, vedge-sd, hedge-mean, hedge-sd, intensity-mean, rawred-mean, rawblue-mean, rawgreen-mean, exred-mean, exblue-mean, exgreen-mean, value-mean, saturation-mean, and hue-mean (Vision Group, 1990). The class attribute has 7 labels (representing different pictures), each with 30 instances. This is a multivariate data set with 210 instances, where each attribute's instance is represented as a continuous value float and each class label is represented as a discrete-value integer using one-hot encoding.

Data Cleaning & Transformation: There were no missing data. The attribute region-pixel-count was also dropped from the data set, as it had no variance.

Exploratory Data Analysis: There were 30 instances for each class label (see Figure 3). Each feature was also described by class (descriptive statistics included mean, standard deviation, minimum, maximum, range, 1st quartile, median, and 3rd quartile) and were plotted using a box-plot (See for example Figure 4). The means for a few of the feature by class can be observed in Table 2.

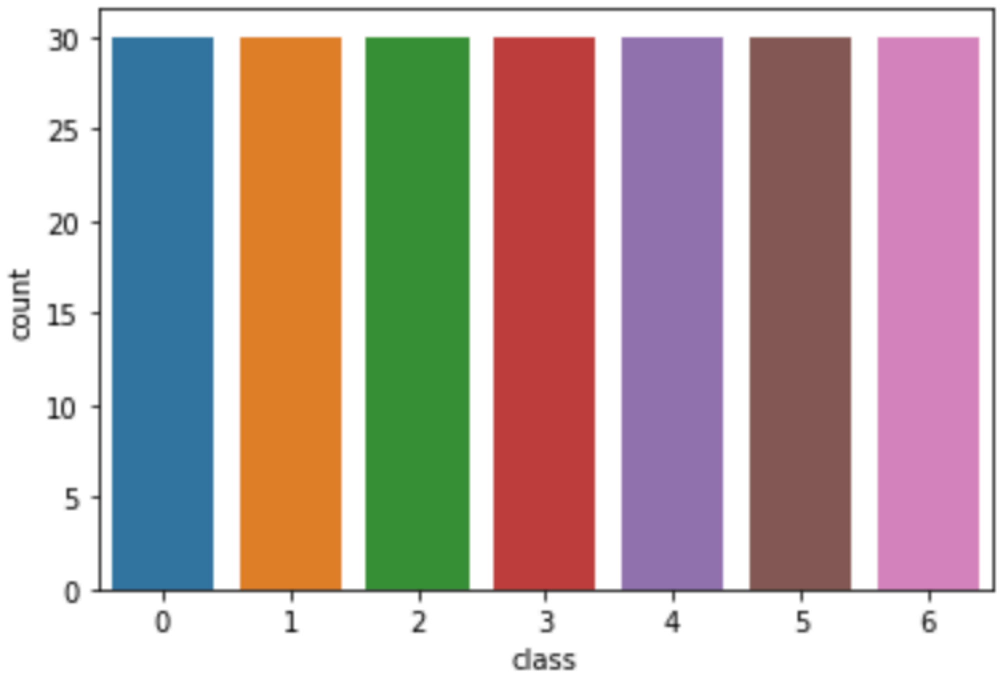

FIGURE 3. Image Segmentation data set plot showing the actual raw count values for image classification (where each classification represents a different picture).

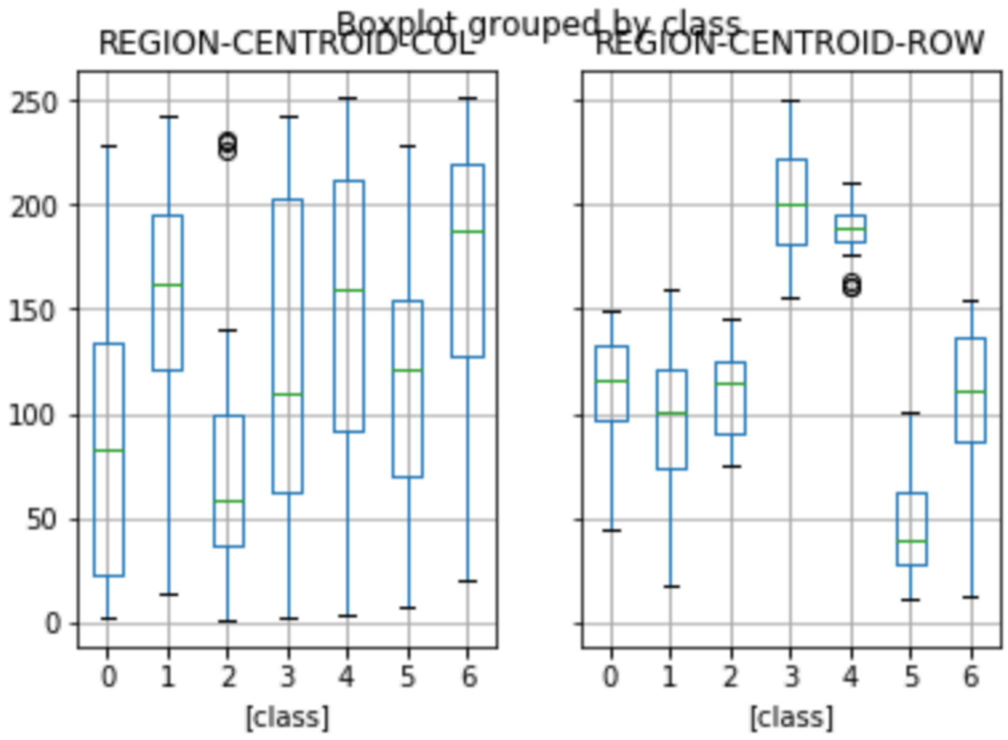

FIGURE 4. Image Segmentation data set boxplot showing attribute grouped by class. Above example depicts the features region-centroi-col and region-centroid-row by class (where each class label represents a different image/picture).

Class Labels	**Attributes/Features**								
	Region-Centroid-Col	**Region-Centroid-Row**	**Short-Line-Density5**	**Short-Line-Density-2**	**Vedge-Mean**	**Vedge-SD**	**Hedge-Mean**	**Hedge-SD**	**Intensity-Mean**
0	83.4000	109.333	0.0037	0.0000	1.03703	1.03086	1.33703	0.85160	13.16543
1	150.5000	97.16666	0.00740	0.00740	2.95185	2.43526	2.56481	3.79208	43.54691
2	76.50000	111.4000	0.00370	0.01481	3.82777	30.9057	5.29074	58.9660	10.99259
3	130.7000	203.5000	0.02592	0.00000	1.50740	1.97301	2.14259	2.06424	14.97777
4	150.1666	187.2333	0.01111	0.02222	2.40000	2.11818	4.62037	10.0159	49.49135
5	116.4000	45.86666	0.00740	0.00000	0.83148	0.58065	1.13703	0.79859	119.069
6	164.866667	104.80000	0.00000	0.00000	0.92037	0.9929	1.1370	4.9801	8.39382

TABLE 2. Mean values of each feature by image classification in the Image Segmentation Data Set.

Computer Hardware Data Set

Data Description: Predicts the performance of a given CPU, based on 9 feature attributes: vendor_name, model_name, myct, mmin, mmax, cash, chmin, chmax, prp, and erp (Ein-Dor & Feldmesser, 1987). The class attribute has 29 labels: adviser, amdahl,apollo, basf, bti, burroughs, c.r.d, cambex, cdc, dec, dg, formation, four-phase, gould, honeywell, hp, ibm, ipl, magnuson, microdata, nas, ncr, nixdorf, perkin-elmer, prime, siemens, sperry, sratus, and wang. This is a multivariate data set with 209 instances, where each attribute's instance is represented as a continuous value float.

Exploratory Data Analysis: There were 32 instances of imb, 19 instances of nas, 13 instances of sperry, 13 instances of Honeywell, 13 instances of ncr, 12 instances of siemens, 9 instances of cdc, 9 instances of amdahl, 8 instances of burroughs, 7 instances of hp, 7 instances of dg, 7 instances of harris, 6 instances of dec, 6 instances of c.r.d., 6 instances of magnuson, 6 instances of ipl, 5 instances of formation, 5 instances of cambex, 5 instances of prime, 3 instances of gould, 3 instances of perkin-elmer, 3 instances of Nixdorf, 2 instances of basf, 2 instances of wang, 2 instances of Apollo, 2 instances of bti, 1 instance of sratus, 1 instance of four-phase, 1 instance of microdata, and 1 instance of adviser (see Figure 5). Each feature was also described by area (descriptive statistics included mean, standard deviation, minimum, maximum, range, 1st quartile, median, and 3rd quartile) and were plotted using a box-plot (See for example Figure 6). A sample of the means for a select few of the attributes and class labels can be observed in Table 3.

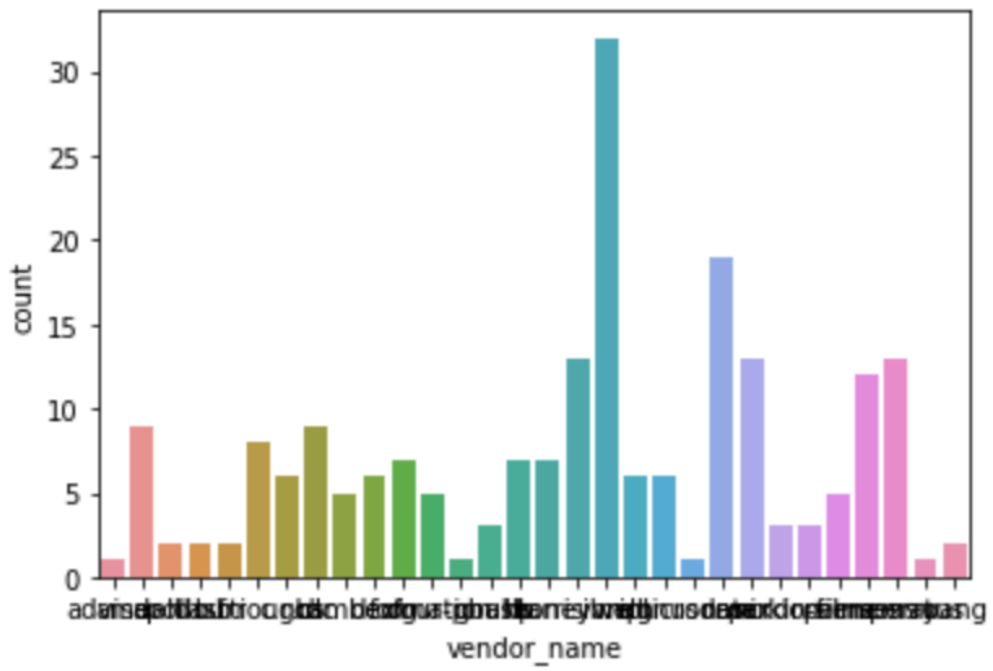

FIGURE 5. Computer Hardware data set plot showing the actual raw count values for vendor name classification.

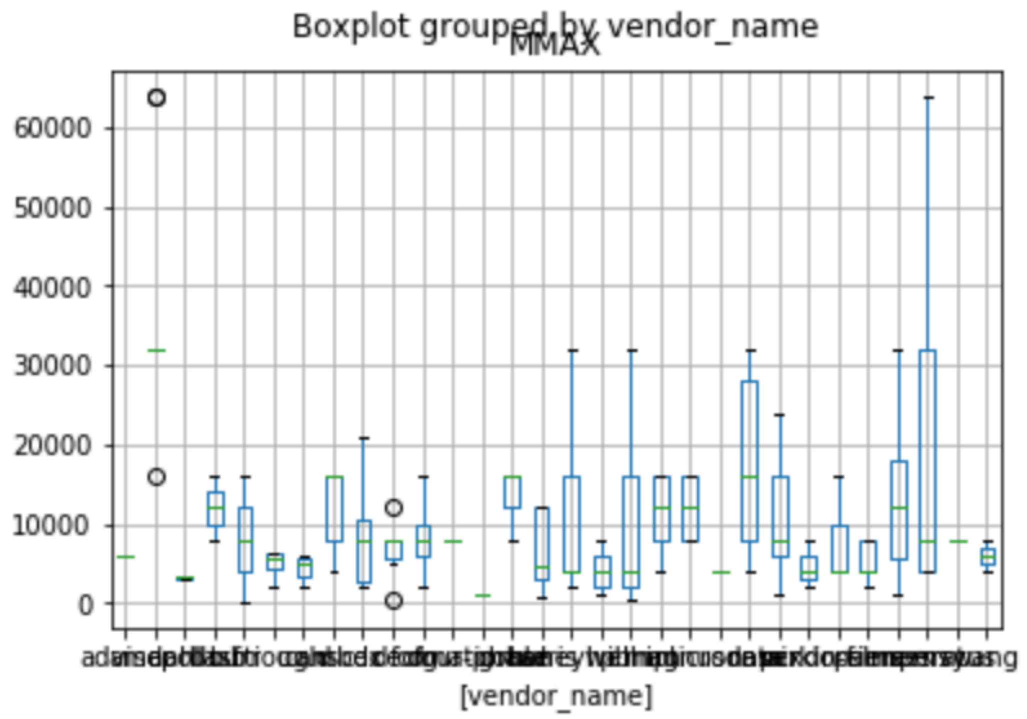

FIGURE 6. Computer Hardware data set boxplot showing attribute grouped by class. Above example depicts the feature MMAX by class (vendor namne).

Class Labels	**Attributes/Features**			
	MYCT	MMIN	MMAX	CASH
Adviser	125.000000	256.000000	6000.000000	256.000000
Amdal	26.000000	13333.333333	37333.333333	56.888889
Apollo	400.000000	756.000000	3250.000000	2.000000

TABLE 3. Mean values of each a few features by a few vendor names (class labels) in the Computer Hardware Data Set.

Forest Fires Data Set

Data Description: Predicts a forest fire's burn area based on 10 feature attribute: month, day, ffmc, dmc, dc, isi, temp, rh, wind, and rain (Cortez & Morais, 2008). The target-value is a continuous-float variable.

Exploratory Data Analysis: Each feature was described by area (descriptive statistics included mean, standard deviation, minimum, maximum, range, 1st quartile, median, and 3rd quartile).

Results

Ecoli Data Set

kNN Classifier on Ecoli Data Set: The Ecoli data set represents a classification problem used to classify localization sites of proteins in Ecoli cells. Three variations of KNN were applied to this data set: Standard KNN, Edited KNN, and Condensed KNN and results are shown in Table 4. A stratified 5-fold cross-validation was utilized to create proportional splits, GridSearch was utilized to tune *k*, and the performance metric Accuracy (i.e., proportion of correctly labeled instances) was measured. As mentioned above, *k* was set to 1 for the condensed KNN algorithm.

Algorithm	***k***	**Accuracy**
Standard KNN	5	83.38%
Edited KNN	4	79.85%
Condensed KNN	1	75.76%

TABLE 4. KNN classification results on Ecoli Data Set. Three KNN classification algorithms were implement (standard KNN, Edited KNN, Condensed KNN). *k* values indicate the optimal number of nearest neighbors for each algorithm. Accuracy was calculated as the proportion of correctly labeled instances.

Overall Conclusions: Results show accuracy at 83.38% for Standard KNN (where $k = 5$), 79.85% for Edited KNN (where $k = 4$), and 75.76% for Condensed KNN (where $k = 1$). As there was little noise in the data set, this result confirms the hypothesis that the Standard KNN Algorithm would outperform both Edited and Condensed KNN Algorithms.

Image Segmentation Data Set

kNN Classifier on Image Segmentation Data Set: The Image Segmentation data set represents a classification problem used to classify images based on pixel attributes. Three variations of KNN were applied to this data set: Standard KNN, Edited KNN, and Condensed KNN and results are shown in Table 5. A stratified 5-fold cross-validation was utilized to create proportional splits, GridSearch was utilized to tune *k*, and the performance metric Accuracy (i.e., proportion of correctly labeled instances) was measured. As mentioned above, *k* was set to 1 for the condensed KNN algorithm.

Algorithm	***k***	**Accuracy**
Standard KNN	1	84.28%
Edited KNN	1	79.52%
Condensed KNN	1	83.33%

TABLE 5. KNN classification results on Image Segmentation Data Set. Three KNN classification algorithms were implement (standard KNN, Edited KNN, Condensed KNN). *k* values indicate the optimal number of nearest neighbors for each algorithm. Accuracy was calculated as the proportion of correctly labeled instances.

Overall Conclusions: Results show accuracy at 84.28% for Standard KNN (where $k = 5$), 79.52% for Edited KNN (where $k = 1$), and 83.33% for Condensed KNN (where $k = 1$). As there was little noise in the data set, this result confirms the hypothesis that the Standard KNN Algorithm would outperform both Edited and Condensed KNN Algorithms. Though, the disparity in hyper-parameter tuning between Standard KNN and Edited KNN was unexpected. It is unclear why Standard KNN had $k = 1$.

Computer Hardware Data Set

kNN Regressor on Computer Hardware Data Set: The Computer Hardware data set represents a regression problem used to predict the relative performance given a set of CPU characteristics. Due to the fact that both the Edited and Condensed KNN Algorithms rely on class labels, only the Standard KNN Algorithm was applied to this data set and the result is shown in Table 6. A stratified 5-fold cross-validation was utilized to create proportional splits, GridSearch was utilized to tune *k*, and the performance metric -RMSE (i.e., negative square root mean error) was measured.

Algorithm	***k***	**RMSE**
Standard KNN	1	69.88%

TABLE 6. KNN Regression results on Computer Hardware Data Set. Note that only the Standard KNN Algorithm was implemented. *k* values indicate the optimal number of nearest neighbors for each algorithm. Accuracy was calculated as -RMSE.

Overall Conclusions: Results show Standard KNN performed at 69.88 RMSE, where *k* =1. Interestingly, this is lower than the performance observed by Ein-Dor and Feldmesser (1987) who used a linear regression model for predictions. This may represent the limitation of nonparametric models when applied to models which can be solved using parametric methods.

Forest Fires Data Set

kNN Regressor on Forest Fires Data Set: The Forest Fires data set represents a regression problem used to predict the burn area of a forest fire given a set of feature characteristics. Due to the fact that both the Edited and Condensed KNN Algorithms rely on class labels, only the Standard KNN Algorithm was applied to this data set and the result is shown in Table 7. Month and Day attributes were one-hot encoded. A stratified 5-fold cross-validation was utilized to create proportional splits, GridSearch was utilized to tune *k*, and the performance metric -RMSE (i.e., negative square root mean error) was measured. This experiment was conducted using two different models. The first model included all inputs from the original data set and the second model included only temperature, relative humidity, wind, and rain (as recommended by Cortez & Morais, 2008).

Algorithm	***Features***	***k***	**RMSE**
Standard KNN	All	1	89.48
Standard KNN	Temperature, Relative Humidity, Wind, & Rain	1	87.21

TABLE 7. KNN regression results on Forest Fire Data Set. Only standard KNN was implemented. Two variations were run: a full model (all attributes) and a partial model (with only temperature, relative humidity, wind, and rain as feature attributes). *k* values indicate the optimal number of nearest neighbors. Accuracy was calculated as the proportion of correctly labeled instances.

Overall Conclusions: Results show Standard KNN performed at 89.48 RMSE, when all features where included in the model and where $k = 4$. Not surprisingly, the partial model had better performance. Standard KNN had 87.21 RMSE (i.e., reduced error) (when only temperature, relative humidity, wind, and rain were included as features and $k = 1$. It is likely that these results demonstrate the issues faced by KNN algorithms when dealing with high dimensionality,

Conclusions

This paper implemented the k-Nearest Neighbors (KNN) Algorithm as an instance-based, Supervised Machine Learning algorithm for nonparametric predictive modeling. Three variations of KNN were utilized for classification problems (i.e., discrete-valued target functions): Standard KNN, Edited KNN, and Condensed KNN. While each algorithm assumes points close in feature space share predictive properties, Edited and Condensed KNN Algorithms modify the training procedure in order to improve cost and performance efficiency. Standard KNN was also implanted for regression problems (i.e., continuous-valued target functions), but is limited in it's ability to handle high-dimensional data sets or known parametric distributional data.

References

Alpaydın, E. (2020). *Introduction to Machine Learning*. Cambridge, MA: MIT Press.

Bhatia, et al. (2010). Survey of Nearest Neighbor Techniques. International Journal of Computer Science and Information Security, 8(2).

Cortez, P., & Morais, A. (2008, February 29). Forest Fires Data Set. Retrieved June 22, 2020 from https://archive.ics.uci.edu/ml/datasets/Forest+Fires

Ein-Dor, P., & Feldmesser, J. (1987, October 01). Computer Hardware Data Set. Retrieved June 22, 2020 from https://archive.ics.uci.edu/ml/datasets/Computer+Hardware

Mitchell, T. M. (2013). *Machine learning*. New York: McGraw-Hill.

Nakai, K. (1996, September 01). Ecoli Data Set. Retrieved June 22, 2020, from https://archive.ics.uci.edu/ml/datasets/Ecoli

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence a modern approach*. Upper Saddle River (N.J.): Pearson.

Vision Group (1990, September 01). Image Segmentation Data Set. Retrieved June 22, 2020 from https://archive.ics.uci.edu/ml/datasets/Image+Segmentation