Journal of Machine Learning Research 24 (2024) 1-7 Submitted 8/30; Published N/A

Project 2 Design Document

Carlos Rivas f71j657@msu.montana.edu

Finnegan Davidson Davidsonfinn@gmail.com

Team: 6 Date: 8/30/2024

1 System Requirements

General Requirements

- **Design Document -** Project 2 requires the submission of a design document (the document you are reading) that outlines the requirements, tasks, architecture, and goals to be accomplished for project 2.
- **Programming -** Six separate datasets will be utilized from the UCI Machine Learning repository, 3 for classification and 3 for regression. Data pre-processing for project 2 involves 10-fold stratified cross-validation for all data, where data imputation and continuization may be necessary. Algorithms to be implemented include k-Nearest Neighbors (kNN), Edited Nearest Neighbors (ENN), and k-Means Clustering with k-Nearest Neighbors (kM-kNN). All 3 algorithms will perform classification and regression tasks.
- **Paper -** A paper entailing results from experiments will be produced with a length between 5 and 10 pages. The paper will include a problem statement, experimental approach and design, presentation of results, algorithm behavior discussion, conclusions, references, and an appendix describing workload distribution.
- **Video -** A video of 5 minutes or less will be provided demonstrating program functionality. The video will focus on program behavior and include display of inputs, data structure, and outputs. Additional demonstrations will include splitting data into 10 folds, distance function calculation, radial basis function calculation, an example being classified and regressed by kNN along with its returned neighbors, an example being removed from ENN, data being assigned to a cluster for k-Means Clustering, and the average performance of kNN, ENN, and kM-kNN for classification and regression.
- **Submission -** Final project submission will include fully documented code files with outputs after operation, the 5-10 page paper, and the demonstration video. All code files (fully commented) will be compressed into a single zip file and submitted along with a team member contribution report to the code submission dropbox. A PDF version of the paper (JMLR format) will be submitted to the TurnItIn dropbox. The demonstration video will be submitted to its respective dropbox as a text file containing a link to the video.

Dataset As previously mentioned, 6 datasets are utilized for project 2, where 3 of the datasets are meant for classification, and 3 are meant for regression. Pre-processing will involve 10-fold stratified cross-validation, where 10% of the data will be used as a hyper-parameter tuning set, and 90% will be used as a train-test set. Both the tuning set and train-test set will implement 10-fold cross-validation.

Algorithms Project 2 involves the implementation of 3 separate learning algorithms, each needing to perform classification and regression tasks. A hypothesis will be developed as a prediction for which learning algorithm will perform best for classification and regression tasks.

- **k-Nearest Neighbors -** The kNN algorithm involves storing all data that is a part of a training set in a number-of-feature (f_c) dimensional space. Examples can be considered graphed in this space based on feature values. When a prediction for a new example is required, the k-Nearest Neighbors are computed based on a given distance function. To find the nearest neighbors the distance needs to be calculated between the new example and all examples from the training set, so runtime is proportional to training dataset size. From the nearest neighbors, the most prevalent class or most probable regression value is determined.

- **Edited-Nearest Neighbors -** As per the name, ENN is similar to kNN, with tweaks to the training data. For each point in a training dataset, this point will be predicted based on all other training data. If the prediction is wrong (or far off in the case of regression) this data point is removed from the training set. The process of editing out incorrectly predicted data points can be repeated, and the number of repetitions for this process can be tuned as a hyperparameter.

- **k-Means Clustering with k-Nearest Neighbors -** The kM-kNN algorithm essentially adds an additional pre-processing step that feeds into kNN. This step initializes k_c centroids in the f_c dimensional space (denoted as k_c to differentiate between this parameter and k_n for kNN). The number of initialized centroids is a parameter that must be tuned. After initialization, centroid location is adjusted based to the average position of the data points within a region. This process is repeated until the centroids converge at a particular location. These centroid locations are then used as the training data for kNN, where there will be k_c data points in the reduced dataset. Predictions will be made based on whichever centroid is closest to the example.

Hyperparameter Tuning. There are a variety of parameters that need to be tuned for each of the 3 algorithms implemented for project 2. Tuning these parameters will encapsulate most of the "training" for these algorithms.

- **k-Nearest Neighbors -** The kNN algorithm will have 2 hyperparameters. The first parameter is k_n, where k_n represents the number of neighbors used to make a prediction for an example. The second parameter will be σ, where σ represents the bandwidth for the radial basis function (RBF) kernel. This specifically affects regression where a weight is assigned to neighbors based on their distance from an example when making a prediction for said example, and σ controls the sharpness of a weight decreasing for a neighbor as distance increases.

- **Edited-Nearest Neighbors -** Since ENN inherits kNN, all kNN hyperparameters are also present in the ENN algorithm, along with 2 additional parameters. To edit out examples for the ENN algorithm, each data point is "predicted" based on the rest of the examples in the dataset, and incorrectly predicted data points are removed. After the first round of examples are removed, this process can be repeated. The number of editing repetitions, denoted as i_e can be tuned. Additionally, to determine if a regression prediction is "correct", an error value is tuned to allow slight variance in a predicted value when compared to its actual value, since the predicted value is continuous.

- **k-Means Clustering with k-Nearest Neighbors -** The kM-kNN algorithm also inherits the kNN algorithm, so all hyperparameters for kNN are present in this algorithm, with 1 additional parameter k_c. This parameter represents the number of centroids initialized for the clustering process, as k_c will control the final number of clusters in the reduced dataset.

Evaluation Metrics With 6 datasets implemented for project 2 (3 for classification, 3 for regression), this means 18 models will need to be evaluated (each algorithm will be tested on each dataset). Classification evaluation metrics will include F1 score and 0/1 loss. Regression evaluation metrics will include Mean Squared Error (MSE) and Mean Absolute Error (MAE). MAE will provide better evaluation of datasets where significant outliers are present, whereas MSE will emphasize outliers on its performance evaluation.

2 System Architecture

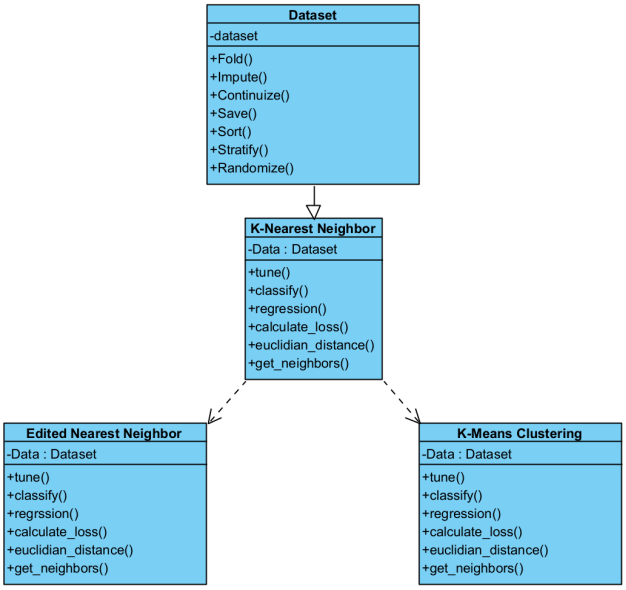

Figure 1: UML Block Diagram

Dataset Class The dataset class will take in raw data downloaded from an online database. Once the dataset has been passed into the class, it will use the randomize() method to randomize the data. It will then use the impute() and continuize() methods to clean the data and properly format it. The sort() method will be called next to organize the examples by class or response value. After this, the data is ready to be split into the two desired subsets before being sorted again. The fold() and stratify() methods will be called last to finalize the pre-processing of the data.

kNN Class The kNN class takes in a dataset object, and utilizes the tuning and testing sets from said object. The tune() method takes in a boolean to indicate whether the dataset contains classification or regression data. The tune() method utilizes calculate_loss(), get_neighbors(), and euclidean_distance(). These methods will primarily be called internally, where calculate_loss() is used to help choose the highest performing hyperparameter value, get_neighbors() takes in an argument k_n and returns the k_n nearest neighbors, and euclidean_distance() calculates the euclidean distance between two points. The classify() and regression() methods use either the hyperparameters created during tuning or default values if tuning was not called. This class then makes predictions on the testing set, where loss metrics are returned.

ENN/kM-kNN Class Both the ENN and kM-kNN class methods perform the same high level tasks described in the kNN class; the difference lies in the internals of methods. The tune() methods for these classes tune their respective hyperparameters, and the the classify/regression methods utilize the reduced datasets produced from ENN/kM-kNN when predicting the testing data.

3 System Flow

Processing The flow of data through the algorithms starts in the dataset class. Initially, the data is shuffled to remove any patterns that could be present in the data. Once the data has been shuffled, it will be imputed to ensure all examples are complete with no missing feature values. The data will then undergo continuization to transform any discrete valued features into continuous valued features. Now, the data will be in the correct format to prepare it for the algorithms. Before being passed to the algorithms, the data is sorted to help with the stratification process. For the classification datasets, the examples will be sorted by class. For the regression datasets, the data will be sorted based on the response value. Once the data is sorted, it is split into two subsets. One subset represents the testing data for the algorithms and the other represents the tuning data to determine the ideal hyperparameters to test with. The data split between these two sets will be 10% tuning data and 90% testing data. During the process of splitting the original dataset into the two subsets, stratification ensures that each subset maintains the same distribution as the original dataset. Once the tuning and testing subsets have been created and filled, the testing dataset is folded 10 times using stratification to ensure the makeup of each fold matches the overall dataset's makeup. Now that there is a folded testing dataset as well as the tuning dataset, the data is processed and ready to be passed to the algorithms.

Algorithms Once the data is processed, it will be used to tune the hyperparameters necessary for the algorithm under investigation. After ideal hyperparameter values are

found, the 10 folds in the testing set will be used for classification or regression for each of the following algorithms.

- **k-Nearest Neighbor -** Within kNN, the examples from 9 of the 10 folds of data will be used as a model. The algorithm will iterate through each example in the holdout fold and calculate the distance between it and every example in the model. It will then use the k_n nearest neighbors to predict the location of the example relative to the other data points. Once all of the examples in the holdout folds have a corresponding prediction, the results are saved and the experiment is repeated using a different fold as the hold out set. Once each fold in the testing set is used as the hold out fold, the results from each model are saved and loss can be calculated.

- **Edited-Nearest Neighbor -** The data flow for ENN is similar to the data flow for kNN. First, the kNN process is completed. After the kNN process, each prediction that the model makes will be checked to see if any predictions don't align with the expected response for the example (class for classification, error for regression). Then, if the prediction doesn't match, the data point will be removed from the dataset, effectively removing a noisy data point. This editing process can be repeated as many times as desired. Once the dataset has been modified to reduce noise, kNN is run again with the smaller dataset. The results from the final kNN process are then saved and loss can be calculated.

- **k-Means Clustering with k-Nearest Neighbor -** The process for kM-kNN shrinks the dataset and then uses kNN to get classification or regression results. To shrink the dataset, the algorithm will initialize k_c centroids randomly into the multidimensional feature space. Each example in the testing set will then be assigned to the centroid it is closest to, using euclidean distance. For each centroid, the algorithm will calculate the average position of all the examples assigned to it. These average positions will be used as new centroids. This process will be repeated until the centroid's position stops moving by a significant amount on each iteration. Once the centroids converge, the locations of each centroid will be saved. These positions become the modified testing dataset. This testing set is then passed into the kNN process to determine the results of the clustering on the algorithm. The results of this kNN process are then saved and loss can be calculated.

4 Test Strategy

Pre-Processing Verification Pre-processing steps for project 2 will be verified using both visual inspection in a save file and Jupyter Notebook print statements to determine if the processing steps are occurring correctly. We will specifically be looking for a proper split between the tuning and testing data, stratification and division into 10 partitions for both of those datasets, and the imputation + continuization of examples where necessary.

Tuning Verification Parameter tuning will be verified by observing program behavior via print statements. We will be looking for final chosen hyperparameters to have the best performance metrics out of all that were tested. This will involve observing the performance

of a range of hyperparameter values, and ensuring the highest performing value is the one picked.

k-Nearest Neighbor Verification To verify the kNN algorithm, we will utilize print statements to show that each step in the process is being executed in the correct order. We will also manually verify the predictions by printing the k_n nearest neighbors, as well as the corresponding prediction. By ensuring that the actual prediction matches the expected prediction we can verify that the algorithm is functioning properly.

Edited-Nearest Neighbor Verification We will utilize print statements within ENN to ensure that only points that were classified incorrectly are being removed from the dataset. To do this, we will print the predicted value from the algorithm, the true value from the raw dataset, and whether or not the point was removed. This process allows an observer to visually verify the ENN editing performance. All other necessary verification is handled by verifying proper operation of the kNN algorithm.

k-Means Clustering with k-Nearest Neighbor Verification Once kNN has been verified, verifying the kM-kNN algorithm can be done by completing clustering on the same dataset twice. Since the centroids are initially placed randomly, if the result of both runs is roughly the same, the clustering is finding the true centroids in the data. This will be accomplished by storing the centroid locations after the initial run and comparing them to the centroid positions after the second run.

Loss Calculations Since both classification and regression tasks need to be evaluated for project 2, separate evaluation metrics need to be used for each learning type. Classification metrics will include F1 score and 0/1 loss. F1 score provides a holistic evaluation to model performance, where it accounts for precision and recall, and 0/1 loss indicates the percentage of examples that were classified incorrectly. Regression performance will be evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE). MSE places a higher emphasis on outliers in the regression performance, where an outlier will have a more negative affect on score. MAE places a lower emphasis on outliers which can provide a more accurate depiction of performance depending on the data under test. An example of this would be predicting the prices of homes based on attributes, where most homes in the testing data (the model) have lower prices, but that doesn't mean more expensive homes don't exist.

5 Task Assignments and Schedule

Task	Student	Date
Dataset Class	Finn	9/26/2024
kNN Class	Carlos	9/29/2024
ENN Class	Finn	10/1/2024
kM-kNN Class	Carlos	10/3/2024
Loss Calculations and Data Presentation	Finn	10/4/2024
Process Verification	Carlos	10/4/2024
Project Report	Finn/Carlos	10/6/2024
Project Video	Finn/Carlos	10/8/2024

Table 1: Assignments and Dates