Journal of Machine Learning Research 24 (2024) 1-4 Submitted 8/30; Published N/A

# Project 1 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 1 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 1.
- **Programming -** Five separate datasets will be used from the UCI Machine Learning repository. Particular datasets require pre-processing to ensure that both a complete set of features is provided and continuous data is discretized. Two pre-processing options will be tested and discussed further in the dataset section. All data used for testing will be split using 10-fold cross-validation. The algorithm being implemented is the Naive Bayes classification algorithm, while 0/1 loss and F1 score will be used as loss functions.
- **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 display of the implemented discretization method, a trained sample model, the class and attribute counting process, and the classification performance along with loss function performance on a single validation fold for both versions of the pre-processed data.
- **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.

©2024 Carlos Rivas and Finnegan Davidson.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v24/21-0000.html.

**Dataset** As previously mentioned, 5 separate datasets will be used for project 1, all with different pre-processing requirements, and two separate pre-processing methods will be implemented. The first method will only perform pre-processing tasks that are strictly necessary, and the second method will be more involved. A hypothesis will be made on the expected performance of the two dataset processing methods.

**Algorithm** Project 1 involves a gentle introduction to the field of machine learning via implementation of a Naive Bayes algorithm. Training for the algorithm includes calculating the percentage of each class out of the entire dataset; i.e., a dataset of trees could be broken up into 70% coniferous and 30% deciduous. After the percentage of each class is determined, analysis will be performed for the attributes of each individual class. This will include determining the likelihood of a given attribute applying to a class, where a coniferous tree may have a 5% likelihood of having brown moss, and 95% likelihood of having green moss. The process of determining attribute likelihood for a given class will be applied to all attributes.

**Evaluation Metrics** Since 5 datasets will be utilized each with 2 separate pre-processing methods, this means 10 separate models will need to be evaluated. Model performance will be quantified using F1 Score and 0/1 loss. F1 score combines precision and recall metrics to give a holistic model performance score, while 0/1 loss simply indicates whether a classified example was identified correctly or not.

## 2 System Architecture

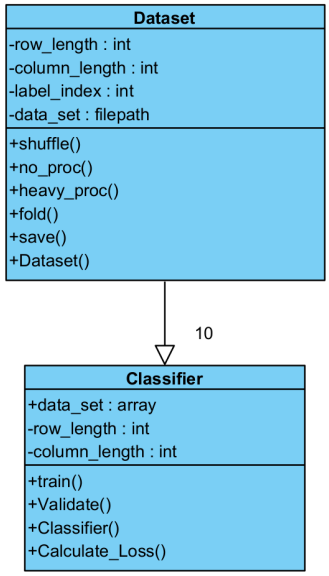


Figure 1: UML Block Diagram

**Dataset Class** The dataset class will be used to format, perform imputation, randomize, and fold the data before passing it into the algorithm class. It will contain methods to shuffle the data, complete light or heavy processing, fold the data, and save the processed data set.

**Classifier Class** The algorithm class will take in a processed data set and will be used to determine the relative frequency of each class in the training data and to calculate the conditional probability of each attribute for each class. This class will also be used to analyze performance metrics and determine loss. It will contain methods to train a model, validate a model, and analyze the performance of the models against the selected performance metrics.

## 3 System Flow

**Processing** The program starts by passing the path of the data file into the dataset class. From there, using the shuffle method, the program randomizes the examples, and uses the fold method to separate the data into 10 partitions. The data is then pre-processed using one of two methods. Method 1 will involve data imputation only where strictly necessary. Imputation will be performed by finding locations where attribute values are missing, and replacing them with a random value between the range of the maximum and minimum attribute value from other examples in the dataset. Pre-processing method 2 will be more involved, where data imputation and discretization will be implemented along with the introduction of noise into the dataset. Noise will be added by selecting 10% of the features in a given dataset, then shuffling the values across examples within a feature. Once the data is processed, it is passed to the classifier class. Once the data is in the classifier class, the train method will be called.

**Training** Within the train method, the algorithm will first determine the number of examples within each class in the dataset. This involves dividing the number of instances of each class by the total number of examples in the dataset to determine the relative frequencies of each class. These probabilities will be stored in a 2D array where the first dimension is proportional to the number of classes and the second dimension will be of length 10 due to 10-fold cross validation.

Now we determine the frequency of each attribute for a given class. Inside each class, the prevalence of each attribute is counted. Once the number of examples with a specific attribute is determined, the program divides this number by the total number of examples within the class plus the number of different attributes. Doing this for every attribute in each class gives a table of frequencies for each attribute for any class. This data will be stored in a 4D array; the first dimension will be of length 10 due to the process of 10-fold cross validation, the second dimension will be proportional to the number of classes, the third dimension will be proportional to the number of attributes, and the fourth dimension will be proportional to the number of possible attribute values. This will conclude training.

**Classification** For each example that needs to be classified, C calculations will need to be performed, where C is the number of classes in the training data. Each calculation determines the probability that the example belongs to a class in the training data, based on the attributes in the example. Using the class distribution and attribute frequencies, the example will be classified based on the highest probability that an example belongs to

a particular class. This is accomplished by multiplying the probability of a class occurring in the training data by the probability of each attribute occurring, given that the example is in that class.

## 4 Test Strategy

**Pre-Processing Verification** Pre-processing steps will be manually verified by inspecting dataset output files. This inspection will include insuring proper formatting (i.e., all labels are moved to the last indice of the row), and ensuring chosen pre-processing steps work as expected (noise was injected properly, data was imputated, etc.).

**Algorithm Verification** Equation processes will be verified by viewing output arrays and determining if the computational behavior is what we expect. Our arrays should contain a proper output to our algorithm equations, so manual calculations for a handful of steps can be used to ensure proper behavior.

**Loss Calculations** Loss calculation will be performed by using F1 score and 0/1 loss. F1 score is a holistic metric that accounts for precision and recall. This metric was chosen due to some datasets favoring a higher recall score for real-world deployment, and others favoring a higher precision score for real-world deployment, so F1 score will give a broader idea of general model performance. The second chosen metric (0/1 loss) simply counts the number correctly classified examples from the validation data.

## 5 Task Assignments and Schedule

| Task | Student | Date |
|---|---|---|
| Method 1 Pre-processing | Finn | 9/3/2024 |
| Method 2 Pre-Processing | Carlos | 9/4/2024 |
| Training Algorithm Part 1 | Finn | 9/6/2024 |
| Training Algorithm Part 2 | Carlos | 9/6/2024 |
| Classification Algorithm | Finn | Cell 9/8/2024 |
| Process Verification | Carlos | 9/9/2024 |
| Project Report | Finn/Carlos | 9/10/2024 |
| Project Video | Finn/Carlos | 9/11/2024 |

Table 1: Assignments and Dates