JHU AAP EN.605.649.83.FA21 Project 1 Report:

Data loading, processing, cleaning, and model interfacing

Mauro Chavez Mauro.antoine.chavez@gmail.com | MCHAVEZ8@JH.EDU
AAP M.S. Bioinformatics Student *(858)354-7465*
San Diego, CA

Abstract
This document summarizes the first project deliverable for Introduction to Machine Learning taken through Johns Hopkins AAP in Fall of 2021. The first project involved tasks that had students develop code for loading common machine learning datasets into friendly data structures. In addition, data grooming and cleaning occurred as well such that certain attributes were normalized / split to facilitate downstream machine learning activities. This project also required organizing code and defining how interfacing with data when training / learning in future assignments.

Video demonstration of code can be found here: https://www.youtube.com/watch?v=RFilxJM011I

1 Introduction

An important component to machine learning is interfacing with the data. In the case of this assignment, data came in the form of csv files with accompanying description text files from the UCI machine learning repository. Specifically, the assignment called for loading and cleaning the abalone, Wisconsin breast cancer, car, forest fire, house votes ('84), and machine data sets. I chose to implement my solutions in python3 following object-oriented design. A key rational behind the design of my code is that many of the actions performed on the data behave like functions where the same input should always result in the same output. As far as this assignment is concerned, excluding the data itself, there are few state variables that are important to track while performing operations. This is why my code is designed such that all my functions are static methods and organized under classes such that similar functions belong to the same class. Another note is that much of my code is demonstrated to be functional and accurate through unit tests. Please reference the unit tests when assessing the completeness of my code. The included passing_unit_tests.txt file shows the results of running all unit tests within the tests/ directory.

1.1 Problem Statement

The high-level task of this assignment included: loading data, handling missing values, handing categorical data, discretization, standardization, cross-validation, calculating evaluation metrics, and implementing a simple majority predictor. The high-level requirements stated that data had to be loaded from disk into memory in data structures that facilitated data modification, filtering, and handoff to machine learning algorithms.

1.2 Hypothesis – Implementation/Design Approach

An efficient way to implement data loading and data cleaning routines is through functions that don't require tracking the state of anything other than the data itself. Therefore, functions will take the form of static/class methods. Data will be loaded from csv files into pandas DataFrames which provide APIs that enable many of the tasks performed on the data. The majority of data related routines will in turn accept pandas DataFrames or Series as input to perform some sort of calculation or modification. The hypothesis is that this is a reasonable approach to structing a machine learning toolbox code repository. In terms of being able to test said hypothesis, the Git repo commit history over the course of this class will tell the story of how often I had to go back and tweak this initial design.

1.3 Hypothesis - Algorithm Projection

A hypothesis specific to the simple majority classification algorithm implemented in this assignment is that it will perform poorly on data sets where labels are more equally distributed across different label types. As a generalization, this model will find the most common label and apply it to all datapoints in predictions. Therefore, data sets where data points are drawn from very narrow/steep class distributions will result in better predictive power of the model. In other words, the bigger the proportion of data points belonging to a single class, the smaller the proportion of data points that will in turn be incorrectly classified as the most common class.

2 Data Pre-Processing

Loading the data from disk is accomplished by the DataLoader class. It contains a method for each of the 6 datasets used in this assignment that loads the csv into a pandas data frame and performs some general data modifications to enable machine learning activities. For the most part, the normalization that occurs is the bare minimum to present the data in a useable form. A common step of all data loading activities is checking each column for missing values and logging that information. In the case that data is missing (which did not seem to be the case from these data sets), the go to solution is to populate missing values with the mean of the values in that column. In general, while further normalization/standardization procedures could be applied to each data set, the load from disk functions were implemented to do the bare minimum modifications to the data. Any other modifications should occur on a case by case basis with respect to the model and learning objectives. Data preprocessing activities specific to each data set are described as follows…

2.1 Abalone Data

The "sex" attribute of this data set was broken out as nominal data resulting in sex_m and a sex_f columns.

2.2 Breast Cancer Data

This data was very much already good to go as the features were reported as real continuous numbers. Based on the specific application of the data, some z-score normalization or binning could be appropriate but those activities should be performed on a case by case basis.

2.3 Car Data

The car data set had some intuitive transformations. The general idea in performing these transformations is that attributes with string values that had clear relationships were normalized into ranks where 0 was the best/most desirable. This was the case with the buying attribute where a low price is preferable to a high price, the maintenance attribute where less maintenance is preferable to more maintenance, the safety attribute where safer is preferable to less safe, and the acceptability where very good is preferable to unacceptable. In respect to the doors, persons, and lug_boot categories, it was harder to generalize the rankings of these attributes, so they were normalized into nominal data columns.

2.4 Forest Fire Data

The month and day attributes were broken down into nominal data columns. The data description file for this recommended log transforming the area attribute but this step was not baked into the default steps included in loading the data off disk.

2.5 House Votes Data

Every single column was broken down into nominal data columns as there is no clear non-politicized ranking of voting behavior on each specific bill.

2.6 Machine Data

This data required few modifications, but vendor was broken down into nominal data columns.

3 Experimental Approach

To test the majority classification algorithm, a script (learning_examples/run_majority_pred.py) was written that accepts a command line parameter which specifies which data set to load and evaluate. The following table describes which label was used for training and prediction of each data set.

Data Set	Class Label Attribute Name
Abalone	shucked_weight
Breast Cancer	class
Car	acceptability
Forest Fires	area
House Votes ('84)	crime_y
Machine	PRP

Table 1: Data sets and data attribute chosen as class label for majority prediction

With the exception of the forest fires data set, all data was left as it is when returned by the DataLoader functions. In the case of the forest fires data, the fire area as used as the class label was split into 5 buckets of equal frequency to help make predictions more effective.

For each data set, k-fold cross validation was performed using 5 partitions. Partitions were made such that the distribution of labels in each was comparable to the distribution of labels in the complete data set. For each learning task, a prediction score was reported that is the percent of correctly predicted data points as a decimal. These are reported for each k-fold cross validation step and averaged. Additionally, when a numerical value was predicted, the mean squared error for each k-fold cross validation step is reported along with the average. To see the complete output of this experiment, please check out the majority_pred_output.txt file.

4 Experimental Results

The following table describes the results of this experiment…

Data Set	Count Unique Labels	Classification Scores	Average Classification Score	Mean Squared Errors	Average Mean Squared Error

Abalone	1515	0.0018450184501845018 0.001771479185119575 0.002677376171352075 0.004524886877828055 0.008583690987124463	0.003880490334321734	0.12655131626691266 0.06668570017714792 0.051488415327978584 0.043044923642533935 0.03937746673819743	0.0654295644305541
Breast Cancer	2	0.6524822695035462 0.6571428571428571 0.6571428571428571 0.6546762589928058 0.6546762589928058	0.6552241003549744	1.3900709219858156 1.3714285714285714 1.3714285714285714 1.381294964028777 1.381294964028777	1.3791035985801026
Car	4	0.6994219653179191 0.6994219653179191 0.6994219653179191 0.6994219653179191 0.7034883720930233	0.7002352466729399	N/A	N/A
Forest Fire	5	0.5961538461538461 0.5961538461538461 0.5961538461538461 0.6019417475728155 0.6078431372549019	0.5996492846578512	N/A	N/A
House Votes	2	0.5681818181818182 0.5681818181818182 0.5747126436781609 0.5697674418604651 0.5697674418604651	0.5701222327525455	N/A	N/A
Machine	116	0.016666666666666666 0.044444444444444446 0.05 0.0625 0.125	0.05972222222222222	49258.075 13160.711111111112 255.8 234.3125 124.125	12606.604722222222

Table 2: results of k-fold cross validation of majority prediction algorithm. When relevant, mean squared error is reported from learning process. Results from the analysis of each k-fold cross validation step are reported along with the average as captured in the log messages provided by running /learning_examples/run_majority_pred.py [DATASET_NAME]. Classification scores report the percentage of correctly classified data points as a decimal.

Interestingly, the house votes data set in predicting voting yes on the crime bill was less accurate than the breast cancer class prediction. This is probably a consequence of the majorities in each data set. In other words, the majority of data points in the breast cancer set are more descriptive of the data as a whole than the majority of data points in the house votes set of that data. The predictive power of the majority predictor seems to degrade exponentially as the number of unique labels in the data set increases. For example, breast cancer and car sets have ~70% of samples classified correctly with 2 and 4 unique classes respectively. This goes town to ~6% in the machine data set with 116 classes and 0.3% in the abalone data set with 1515 class labels.

5 Discussion

The table from section 4 tells a pretty clear story, when there are less labels/class types in the data set, majority predictor is more effective. In the cases of the breast cancer, car, forest fire, and house votes data sets, classification scores are in the range where slightly more than half of all data points are classified correctly. However, as the number of labels approaches 2, a random predictor would on average classify half of all data points correctly. This means that the majority predictor is only occasionally better than a random predictor. Furthermore, the accuracy of the Forest Fire data set is only on account of the binning of class labels. Without binning the class labels into discrete buckets, majority predictor would have effectiveness more comparable to the machine or abalone data sets. Another thing to note is that majority predictor falls apart when attempting to classify labels that exist on a continuous range of values. Breast cancer and car data sets performed the best not only because they had relatively few labels, but also because all labels fell within 2 or 4 for breast cancer or 0, 1, 2, 3 for car data points. The main takeaway though is that majority predictor, given the simplest data sets where it is expected to perform at its best, is only slightly better than a random predictor.

The mean squared error across each data set is completely dependent on the range of real numbers that each class label can occupy. Discrete ranges perform much better than continuous ranges. For example, the range of potential machine labels is much wider than the range of breast cancer labels. That being said, the breast cancer labels could only occupy values 2 or 4. Given that the mean squared error is ~1.4 for the breast cancer data set, on average the error of each prediction puts the actual value in the middle of 2 or 4. This effectively makes this prediction worthless for practical applications.

If this were a more serious algorithm, I would have been interested in mapping the distribution of label types for each data set. This way we could see how the width of the distribution hurts the prediction score. As noted earlier, the majority predictor performed only slightly better than a random predictor so it's really not worth reading into. The goal of this application was to practice the workflow of loading data, cleaning it, feeding it into a model, and assessing the effectiveness of said model.

6 Conclusion

I found my object-oriented design approach to this assignment very effective when time came to string together the machine learning workflow with the majority classifier. Most of my time was spent implementing the individual functions and writing unit tests to validate their functionality on simple examples that could be solved and traced by hand. I encourage you to review my unit tests to validate the correctness of my methods. If this is not a good way to demonstrate code functionality please let me know and I'll refrain from taking that approach in future assignments.

On another note, I am hoping that the way in which I structured my code base facilitates implementing the solutions to future assignments. Any extra helper functions will go in the data_utils/ directory. Models learned in future units will then go in the learning_algorithms/ directory and will be accompanied by scripts in the learning_examples/ directory that exposes a command line interface to running said model on the 6 data sets studied in project 1. I would have liked to incorporate some data visualization in the form of graphs and plots but ran out of time. I didn't write any code to easily draw these up so I might take some time to add that functionality prior to the next assignment. To conclude, my primary take away is to start these assignments early and that majority classifiers shouldn't be used outside of learning the data analysis process.