Project 1: Naive Majority Predictor Algorithm For Classification and Regression

Dongjun Cho

Programming Project 1
605.649
6 February 2022

Abstract

This project is intended to implement an algorithm called Majority Predictor for classification and regression. This algorithm was applied to discrete and continuous-valued data using six data sets obtained from the UCI Machine Learning Repository.

1. Introduction

Machine learning learns from data. It aims to build a model that can accurately predict future behavior. Data processing is a necessary task because it converts data from a given form to the desired form to obtain an accurate predictive model. Learning algorithms can use processed data to create models that make accurate assumptions to find patterns.

This project provides opportunities for students enrolled in Johns Hopkins University's Introduction to Machine Learning (EN 605.649) to build a basic machine learning pipeline and implement naive majority predictor algorithm for classification and regression.

The basic concept of the majority predictor algorithm is introduced in "Chapter 18.4:Voting" in Alpaydın (2020). For classification, it can be applied "plurality voting where the class having the maximum number of votes is the winner." So, the algorithm should return the most common label in classification tasks. For Regression, "simple averaging or median can be used to fuse the output." So, the algorithm should return the average of the output for regression tasks.

Hypothesis 1: We hypothesize that a the naive majority predictor algorithm works better for classification tasks than regression tasks. To test this hypothesis, we compared accuracy for classification and regression tasks on several data sets from the UCI Machine Learning Repository.

In Section 2, we describe how the data is processed, in section 3, we briefly describe the dataset we are going to use. In section 4, we describe the experimental approach to the algorithm, in section 5, we present the result of our experiment. We discussed the algorithm in section 6, and then we conclude the discussion with the result and possible future works in section 7.

2. Data Processing

There are 6 dataset that we obtained from UCI Machine learning repository. To process the data, there are four steps: load data, handling missing values, encoding categorical data, and discretizing real-valued features.

The first step is loading the dataset. Dataset is loaded from the folder, but some dataset doesn't have a header. After reviewing .names files, the header columns are inserted manually during loading from the folder. Also, non-feature columns in datasets are dropped.

The second step is handling missing values. For these steps, most datasets labeled missing-valued data as "?" or "NaN." Using data imputation, missing-valued data is replaced with the mean of that particular feature. Not every dataset is replaced using data imputation because some dataset has "?" labeled data on purpose.

The third step is encoding categorical data. Since the current learning algorithm can't handle non-numeric data, it had to convert the data into numerical values or tokens. To convert this dataset into the numeric, a unique function is used from Pandas library. Since the unique function is used, it returned the values in order of appearance. Since it returns in order of appearance, it can't categorize data based on their relationship. For example, if there are data features of (red, blue, green), the categorizing function uses a unique function categorized (0, 1, 2). So, red is 0, blue is 1, and green is 2. Using a unique function, it replaced non-numerical data with numerical data.

The fourth Step is discretizing real-valued features. The process of this step is to discretize a continuous-valued attribute by creating a specified number of bins Peng et al. (2009). Using discretization, it is possible to transform real-valued/continuous data into a series of discretized values. There are two methods of discretization: Equal-width, Equal-frequency. Equal-width discretization is the way to partition data by the range of feature value into equal-sized bins. To perform, it divides the data into k-intervals of equal size. Equal-frequency discretization is the way to partition data by fair distributing class in each interval. To perform, it uses one of the Pandas Library' function called "qcut." It discretize variable into equal-sized buckets based on rank or based on sample quantities.

These steps are data pre-processing steps that we applied to the dataset that we will discuss in the next section of Dataset.

3. Dataset

There are 6 data sets, each obtained from the UCI Machine Learning Repository Dua and Graff (2017): Breast Cancer, Car Evaluation, Congressional Vote, Abalone, Computer Hardware, and Forest Fires.

3.1 Wisconsin Breast Cancer Dataset

The Breast Cancer set represents a classification problem used to distinguish whether it is benign or malignant based on 10 feature attributes: sample code number, clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitosis. It has a total of 699 instances. Each attribute's instance is represented as a discrete value integer, ranging from 1 to 10.

To pre-process this dataset, missing values in the data were dropped and the header is inserted manually. Also, unnecessary attributes were dropped, such as sample code number. Since this dataset didn't include any non-numeric, and real-valued data, the third and fourth steps that section 2 introduced are skipped. So, it successfully processed this data into the desired form for the algorithm.

3.2 Car Evaluation

The Car Evaluation set represents a classification problem used to distinguish car acceptability based on 6 feature attributes: buying, maint, doors, persons, lug_boot, and safety. It has a total of 1728 instances. Each attribute's instance is represented as non-numeric data.

To pre-process this dataset, it checked whether the missing value exists. Missing attribute values don't exist based on dataset description (.names). Since the data in this dataset is non-numeric, the dataset is converted to numbers/tokens based on the third step in section 2. Since there is no real-valued in this dataset, step 4 in section 2 is skipped. So, it successfully processed this data into the desired form for the algorithm.

3.3 Congressional Vote

The Congressional Vote set represents a classification problem used to distinguish house party(republican/democrat) based on 16 features of attributes of legislation votes (handicapped infants, water project cost-sharing, adoption of the budget resolution, etc.) It has a total of 435 instances, 267 democrats data, 168 republicans data. Each attribute's instance is represented as "y", "n", and "?."

Based on the dataset description (.names), it shows "?" doesn't indicate missing attribute values. For this dataset, "?" means abstain. Since "?" is not part of missing values, step 2 is skipped. Since the data in the dataset is non-numeric, the dataset is converted to numbers/tokens based on the third step in section 2. Just like other classification tasks, step 4 in section 2 doesn't apply. So, it successfully processed this data into the desired form for the algorithm.

3.4 Abalone

The Abalone set represents a regression problem used to distinguish the associate age of abalone based on 8 features attributes: sex, length, diameter, height, whole weight, shucked weight, viscera weight, and shell weight. It has a total of 4177 instances. Each attribute's instance is represented as a continuous-value float.

To pre-process this dataset, it checked whether the missing value exists. Missing attribute values don't exist based on dataset description (.names). Since one column named "sex" is non-numeric, it is converted to numbers/tokens based on the third step in section 2. Since the feature attribute has real-valued data, the discretization was applied. For discretization, it used the Equal-frequency method with a bin size of 5. So, it successfully processed this data into the desired form for the algorithm.

3.5 Computer Hardware

The Computer hardware set represents a regression problem used to distinguish the CPU performance based on 10 features attributes: vendor name, Model, MYCT, MMIN, MMAX, CACH, CHMIN, and CHMAX. It has a total of 209 instances. Each attribute's instance is represented as an integer.

To pre-process this dataset, it checked whether the missing value exists. Missing attribute values didn't exist based on dataset description (.names). From attributes, the

"vendor name", "Model name", and "ERP" were dropped because they can't be used as a feature. Since the remaining features are in integer, the third step in section 2 is skipped. Even though the values are in integer, discretization was applied to handle real-valued data. So, it successfully processed this data into the desired form for the algorithm.

3.6 Forest Fires

The Forest Fires set represents a regression problem used to predict the burned area of forest fires by using 12 attributes features of meteorological data such as a month, day, FFMC, DMC, ISI, temp, etc. There is a total of 517 instances. Each attribute's instance is represented as a continuous-float variable.

To pre-process this dataset, it checked whether the missing value exists. Missing attribute values didn't exist based on dataset description (.names). There are two columns named "month", and "day" which are not numeric, so it converted to numbers/tokens based on the third step in section 2. Since the target value is very skewed toward 0.0, it used log transform. Also, discretization was applied to handle real-valued in the dataset. So, it successfully processed this data into the desired form for the algorithm.

4. Experimental Approach

This project described two algorithms: naive majority predictor for classification and regression. To test data in naive majority predictor for classification and regression, cross-validation was applied to accurately estimate the result by checking all datasets. Cross-validation is the re-sampling method that partitions training set into an equal-sized portion for validation. For cross-validation, it splits the data into the training set and testing set. Then, it partitioned the training set into k-folds. When the function partitioned the training set, it divided data into k-equal-sized partitions.

The naive majority predictor algorithm for classification, predict by checking plurality class labels in classification tasks. For this project, it used 5-fold cross-validation. So, it is partitioned by a 5-equal size. For each individual fold, it used 4-fold for training and testing for the remaining fold. It takes the most common label from 4-fold and compares it with the remaining fold. Then, it computes the accuracy for each fold.

The naive majority predictor algorithm for regression is predicted by checking the average class label in regression tasks. The regression applied similar concepts as classification tasks. For each individual fold, it used 4-fold for training and testing for the remaining fold. It takes mean values from 4-fold and compares them with the remaining fold. Then, it computes the accuracy for each fold.

For the naive majority predictor algorithm for classification and regression, it computes the accuracy by counting how many values are matched with testing. So, there are no observed values or predicted values. So, it assumes the evaluation metrics for classification and regression are averaging the prediction over all of the folds of the dataset for this project.

5. Results of Experiments

5.1 Breast Cancer Dataset

For the breast cancer dataset, the naive majority predictor algorithm for classification was applied. It demonstrated the accuracy of each fold and show total accuracy as 63.17% in the table below.

Fold	Accuracy (%)
1	52.68
2	60.71
3	50.0
4	75.0
5	77.48
Total Accuracy: 63.17%	

5.2 Car Evaluation

For car evaluation dataset, the naive majority predictor algorithm for classification was applied. It demonstrated the accuracy of each fold and show total accuracy as 73.21% in the table below.

Fold	Accuracy (%)
1	94.22
2	79.42
3	66.67
4	65.22
5	60.51
Total Accuracy: 73.21%	

5.3 Congressional Vote

For congressional vote dataset, the naive majority predictor algorithm for classification was applied. It demonstrated the accuracy of each fold and show total accuracy as 61.48% in the table below.

Fold	Accuracy (%)
1	60.0
2	61.43
3	67.14
4	57.97
5	60.87
Total Accuracy: 61.48%	

5.4 Abalone

For the abalone dataset, the naive majority predictor algorithm for regression was applied. It demonstrated the accuracy of each fold and show total accuracy as 16.25% in the table below.

Fold	Accuracy (%)
1	12.56
2	12.11
3	22.16
4	10.63
5	15.87
Total Accuracy: 14.67%	

5.5 Computer Hardware

For the computer hardware dataset, the naive majority predictor algorithm for regression was applied. It demonstrated the accuracy of each fold and show total accuracy as 18.04% in the table below.

Fold	Accuracy (%)
1	5.88
2	17.65
3	18.18
4	18.18
5	30.3
Total Accuracy: 18.04%	

5.6 Forest Fires

For the forest fires dataset, the naive majority predictor algorithm for regression was applied. It demonstrated the accuracy of each fold and show total accuracy as 13.08% in the table below.

Fold	Accuracy (%)
1	0.0
2	2.41
3	9.64
4	24.1
5	29.27
Total Accuracy: 13.08%	

6. Behavior of Algorithms

Based on the results, it shows that the naive majority predictor for classification shows better accuracy than the naive majority predictor for regression. In classification tasks, it shows good accuracy. Especially, it is very interesting that the accuracy shows almost 95% for the fold 1 in the car evaluation dataset. Based on this result for each dataset, I think the naive majority predictor for classification and regression depends on how the class data is distributed in the dataset. Based on the experiment, the result shows high accuracy if the distribution of the most common label is high.

In regression tasks, many values are continuous and real-valued. Even though discretization was applied to regression tasks, it shows lower accuracy than classification tasks.

After experimenting naive majority predictor algorithm for classification and regression, it seems this naive majority predictor algorithm works better for classification tasks.

7. Conclusion

This paper implemented the naive majority predictor algorithm for classification and regression. Since this algorithm doesn't consider any of the feature values in the decision, the naive algorithm showed low performance/accuracy. Throughout the paper, I was able to learn how to transform to the desired form to test in the algorithm. I learned how processing data is so important for the learning algorithm to make an accurate decision.

For future analysis, it should implement other common evaluation metrics, such as precision, recall, and the F1 score for classification tasks. Also, the machine learning algorithm can be improved by considering feature value in the decision.

References

Ethem Alpaydın. *Introduction to Machine Learning, fourth edition.* The MIT Press, 2020.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Liu Peng, Wang Qing, and Gu Yujia. Study on comparison of discretization methods. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, volume 4, pages 380–384, 2009. doi: 10.1109/AICI.2009.385.