Programming Project #1

Chris Wells

# Introduction

The purpose of this project is to build a suite of tools that will pre-process the datasets used in this course and verify their functionality by constructing a very basic machine learning pipeline that uses a “Null Model” predictor and 5-fold cross validation.

I will be treating four of the datasets: abalone, breast-cancer-wisconsin, car, and house-votes-84 as classification problems. The target values of these datasets belong to two or more categories and the Null Model predictor will always predict the mode, or most frequently occurring class in the training data. The score will be calculated as the fraction of correctly predicted classes.

With this simple model and given that the training data will be stratified (see Section 2), I expect the accuracy of the predictions for these datasets to be approximately equal to the frequency of the mode in each dataset: ~16% for abalone (mode: “9”), ~65% for breast-cancer-wisconsin (mode: “2”), ~70% for car (mode: “unacc”), and ~61% for house-votes-84 (mode: “democrat”). Since all the training and testing datasets will be stratified I expect very low standard deviation for the scores.

I will be treating the remaining two datasets: forestfires and machine, as regression problems with numerical target values. The Null Model predictor will always predict the average of the training data. The score will be calculated as the Mean Squared Error (MSE), the average of the square of the difference between the predicted value and the true value.

With this simple model I expect the MSE to be quite large since I expect a high proportion of errors and the score squares them. Because the data will not be stratified I also expect there to be high variations in the score of different folds since they may contain ranges of data closer or father from the average. Therefore, I also expect to see a high standard deviation for the scores.

# Pre-Processing

I will be storing all my datasets as Pandas DataFrame objects, reading them from the data files using the pd.read\_csv() function. In all the datasets except forestfires, I pass in the column names as a list using the “names” attribute, but in the case of forestfires I specify the first row as containing column names.

## Abalone

After reading the data file into a new Pandas DataFrame, I one-hot encode the nominal feature ‘Sex’, creating three columns “Sex\_M”, “Sex\_F”, and “Sex\_I” and dropping the original “Sex” column.

## Breast-Cancer-Wisconsin

After reading the data file into a new Pandas DataFrame, I replace all instances of the string “?” (denoting a missing value in the “Bare Nuclei” column) with a NaN value and cast all the columns as having integer values. With this done I can use the pandas fillna function to replace all the missing “Bare Nuclei” values with the feature average.

## Car

After reading the data file into a new Pandas DataFrame, I encode all the feature columns, which are ordinal values, to an integer value.

For “buying” and “maint”, I set “low”, “med”, “high”, and “vhigh” values to the integers 0 through 4 respectively.

For “doors” I set “2”, “3”, “4”, and “5more” to integers 0 through 3 respectively.

For “persons” I set “2”, “4”, and “more” to integers 0 through 2 respectively.

For “lug\_boot” I set “small”, “med”, and “big” to integers 0 through 2 respectively.

For “safety” I set “low”, “med”, and “high” to integers 0 through 2 respectively.

## ForestFires

After reading the data file into a new Pandas DataFrame, I encode the “month” and “day” feature columns, which are ordinal values, to an integer value.

For “month” I set “jan” to 0, “feb” to 1, and so on through “dec” to 11.

For “day” I set “sun” to 0, “mon” to 1, and so on through “sat” to 6.

## House-Votes-84

After reading the data file into a new Pandas DataFrame, I one-hot encode all the feature columns, which are nominal variables. So each feature “<vote>” becomes 3: “<vote>\_yea”, “<vote>\_nay”, and “<vote>\_?”. The original “<vote>” feature is dropped.

## Machine

After reading the data file into a new Pandas DataFrame, I one-hot encode the nominal feature “vendor”, creating several new columns representing “vendor\_<vendor\_name>” and dropping the original “vendor” column.

# Experiment

After pre-processing is completed each dataset is represented as a Pandas DataFrame object. The experiment was then conducted for each dataset as follows.

## Training-Validation Split

## The original dataframe is split into two separate dataframes. The first, containing 80% of the entries, is the training set. The second dataframe contains the validation set used in Hyperparameter tuning. For classification datasets these datasets are stratified.

## Training Set Partition

## The training set is partitioned into 5 approximately equal dataframes (or “folds”) and returned as a list of dataframe objects. For classification datasets these datasets are stratified.

## Hyperparameter Tuning

## Five Null Model Predictors are created and each is trained on 4 of the 5 folds, with each predictor leaving out a different fold. The models are then tested on the validation set and the model with the best accuracy score is chosen to execute 5-fold cross validation (see below).

## Because of the lack of parameters for Null Model Predictors, the winner of the hyperparameter tuning will be the model that gets the “best” draw of the folds. That is, the fold with the most values equal to the mean or the closest values to the average.

## 5-Fold Cross Validation

## Using the same parameter values on the “winning” model above, 5 Null Model Predictors are created. Each is trained on 4 of the 5 folds, with each predictor leaving out a different fold. Each predictor is then tested on its non-trained fold. Each of the 5 predictors’ accuracy scores are added to an array of scores.

## Calculating Results

## The array of scores are averaged using the np.mean() function and their standard deviation is found using the np.std() function. These values are then displayed to the user.

# Results

Abalone: Avg classification score= 16.477923%, Std Dev= 0.000824

Breast Cancer: Avg classification score= 65.474582%, Std Dev= 0.003752

Car: Avg classification score= 70.043426%, Std Dev= 0.001324

Forest Fires: Avg MSE= 3745.538892, Std Dev= 5792.846685

House Votes: Avg classification score= 61.495798%, Std Dev= 0.001345

Machine: Avg MSE= 20900.766552, Std Dev= 11904.794808

As hypothesized, for the classification problems the score was approximately equal to the frequency of the most common class and the standard deviation was extremely low for the classification sets. If the datasets were not stratified during the experiment, I would expect a much higher variance.

To check the MSE values I first calculated the MSE for the entire dataset in MS Excel for comparison (files included in the project code: “forestfires.xlsx” and “machine.xlsx”):

Forestfires MSE 4,044.226

Machine MSE 25,742.76

For the regression problems the MSE was extremely high, though close to the MSE values for the entire dataset as expected (they are not identical since the model is only trained on a part of the dataset). Given that the datasets were not stratified the variance was extremely high as hypothesized.

# Conclusion

In this project the tools necessary for a machine learning pipeline have been successfully created, including functions to read, pre-process, train, test, and evaluate each of the six datasets that will be used in this course.

This pipeline has been tested using a basic “Null Model” predictor and returned results consistent with what was expected from this model. Different models may be plugged in to this pipeline or different parameters used during execution as required (such as a different training-validation split or a different number of k-folds) for future projects.