Assignment 1: Some Basic Ideas in Machine Learning

BUAD 5072 – Fall 2017

---

1. Objectives

The purpose of this assignment is to provide you with an opportunity to investigate some of the ideas regarding the assessment of model accuracy that we have been discussing in class.

2. What You Will Need

- Access to a Windows computer with R, and to the following files, which can be downloaded from the Class Schedule page of the course web site:
  - HomePrices.txt
  - LoanData.csv
  - applications.train.csv

3. What You Will Hand In

Submit your script file as Assignment1.R via Blackboard - Assignment 1.

4. Due Date

Monday October 23rd, just before midnight.

5. Note on Collaboration

This is a Category C assignment. Specifically, you may work with others or receive help from the instructor on this assignment. You must, however, turn in your own paper. You may not divide the work with others or copy another student's work. **It would be an honor code offense to do so**.

6. Preliminaries:

To get set up for the assignment, follow these steps:

1. As the first statement in your script file, enter rm(list=ls())
2. Each question in the assignment should begin with the following three comment lines, where *n* is the question number:

   ####################
   #### QUESTION *n* ####
   ####################

3. I should be able to run your script on my computer without errors or interruptions. For this to happen, you must:
   a. Avoid entering file path information…my files will be located in a different location that yours, and so your code will fail on my machine. Instead, always refer only to files in your working directory.
   b. Do not use functions like file.choose(), fix(),edit(), or q()
   c. Do not include install.packages() functions (comment them out if you like)
4. Do not create console output other than what is asked of you explicitly. For example, in your final script, remove any statements that you used to verify the contents or structure of data.
5. I suggest that you read the entire assignment before starting – there are often notes and suggestions at the end of the assignment document.

7. Assignment Tasks:

Question 1: Predicting House Prices: The Regression Setting (30%)

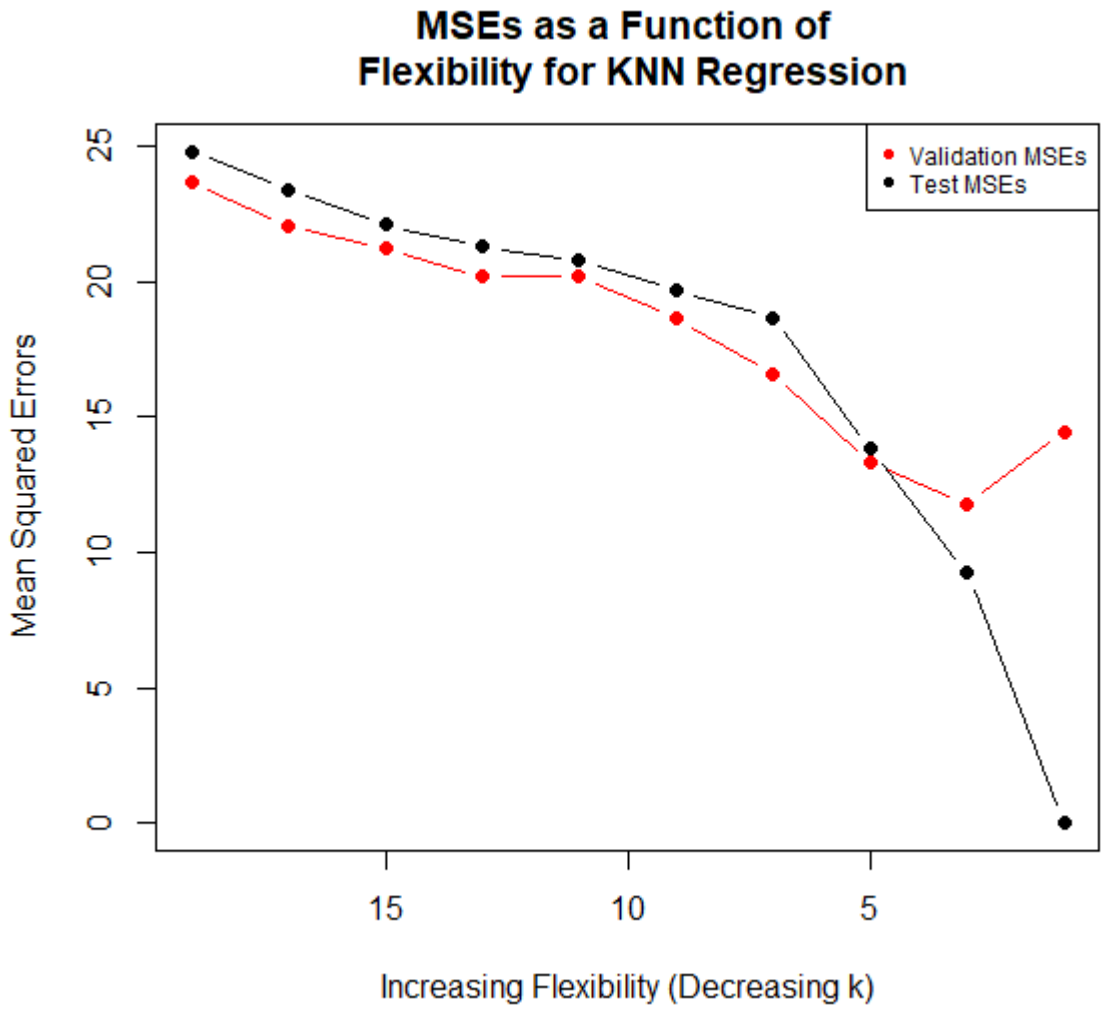
- Create a data frame from the file named HomePrices.txt. that contains median home values for 506 neighborhoods in and around Boston. In the following steps, you will use this dataset to construct a series of regression models to predict the median home value (the medv variable) using all the other variables as predictors. Do the following:
- Print the MSE that would result from always predicting the mean of medv by computing the mean of the squared distances between the values of medv and its mean.
- Print the variance of medv, assuming that the data represent a population, not a sample
  - Hint: Use the var() function and then multiply by (n-1)/n, where n is the total number of observations in the full dataset
- Because the "nearest-neighbor" models you will build are based on Euclidean distance, scale is important. Scale and center the numeric and integer predictors (but not the variable you

are predicting – it is not used in distance calculations and we want to preserve its scale for comparison with the MSE computed above).
    - Recall the scale() function
    - **Important**: do this before partitioning the data in the next step.
- Display the first 6 rows of your data frame. It should look like this:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv
1	-0.4193669	0.2845483	-1.2866362	-0.2723291	-0.1440749	0.4132629	-0.1198948	0.140075	-0.9818712	-0.6659492	-1.4575580	-1.0744990	24.0
2	-0.4169267	-0.4872402	-0.5927944	-0.2723291	-0.7395304	0.1940824	0.3668034	0.556609	-0.8670245	-0.9863534	-0.3027945	-0.4919525	21.6
3	-0.4169290	-0.4872402	-0.5927944	-0.2723291	-0.7395304	1.2814456	-0.2655490	0.556609	-0.8670245	-0.9863534	-0.3027945	-1.2075324	34.7
4	-0.4163384	-0.4872402	-1.3055857	-0.2723291	-0.8344581	1.0152978	-0.8090878	1.076671	-0.7521778	-1.1050216	0.1129203	-1.3601708	33.4
5	-0.4120741	-0.4872402	-1.3055857	-0.2723291	-0.8344581	1.2273620	-0.5106743	1.076671	-0.7521778	-1.1050216	0.1129203	-1.0254866	36.2
6	-0.4166314	-0.4872402	-1.3055857	-0.2723291	-0.8344581	0.2068916	-0.3508100	1.076671	-0.7521778	-1.1050216	0.1129203	-1.0422909	28.7

- Set the random seed to 5072
- Create training, validate and test data frames from the scaled data frame – use a 75/15/10 split.
- Display the first row of each of your three data frames. The row numbers of your train, validate and test data frames should be 125, 120 and 17, respectively.
- Create the following 6 data frames:
    - train.x: the training rows from the scaled numeric predictors
    - validate.x: the validate rows from the scaled numeric predictors
    - test.x: the test rows from the scaled numeric predictors
    - train.y: the training rows from the medv column of the original data frame
    - validate.y: the validate rows from the medv column of the original data frame
    - test.y: the test rows from the medv column of the original data frame.
- Using the knn.reg() function (part of the FNN package), predict the median value of homes (the medv variable) using all the scaled variables as predictors, as follows:
    - Construct 10 models with k equal to 19, 17, 15… 1, evaluating both the validate and training MSE's for each k.
        - Note that for each k, you will need to execute the knn.reg() function twice, once to predict the validate dependent variables and once to predict the training dependent variables in order to compute the MSE's for each.
    - The knn.reg() function requires 4 input parameters and produces a named list as output. The list element named $pred contains the predictions. The input parameters are:
        - A matrix or data frame containing the predictors associated with the training data
        - A matrix or data frame containing the predictors associated with the data for which we wish to make predictions
        - A factor or character vector containing the class labels for the training observations
        - A value for k, the number of nearest neighbors to be used by the classifier.
- Plot the training and validate MSE's as a function of the k's. The x-axis should go from least flexible to most flexible – left to right.
- Print the k and associated MSE that produced the lowest training MSE. Do the same for the validate MSE.
    - My "best" validate MSE occurred with k = 3 and produced a validate MSE of 11.8069925925926

- Finally, predict medv for the test set using the optimal value of k that you found for the validate set, and compute (and display) the associated MSE.
- Observe that there is such a large discrepancy between the MSE for the validate set and the MSE for the test set when using a value for k that was best for the validate set for both predictions. This will be investigated in Question 3.
- Your plot should look something like this (note that k decreases along the x-axis):



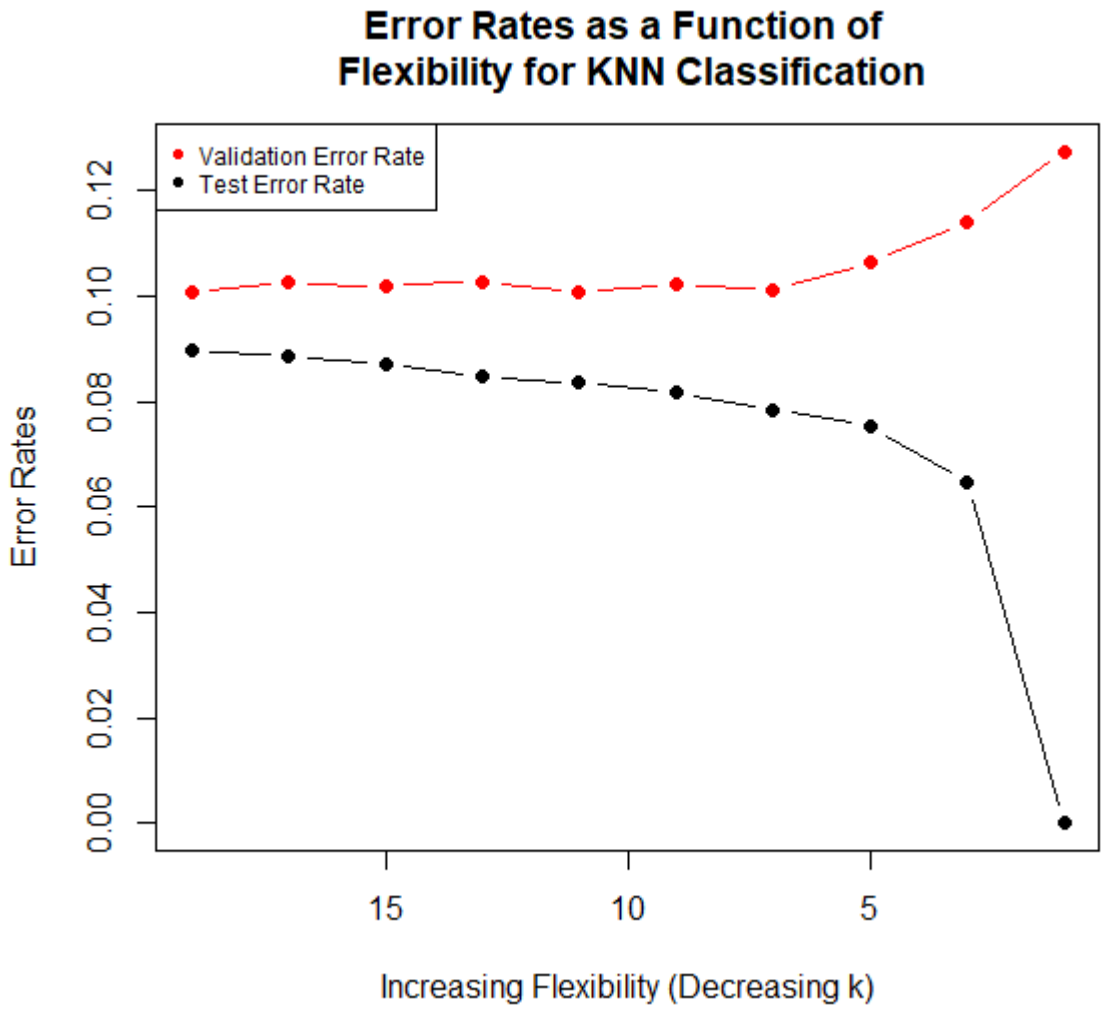
Question 2: Predicting Loan Repayment: The Classification Setting (30%)

- Create a data frame from the file named LoanData.csv. that contains data on bank loans. Using this dataset, you will construct a series of models to predict whether or not borrowers will repay their loans (the loan.repaid variable) using all the other variables as predictors.
- Print the error rate that would result from always predicting Yes
- Because the "nearest-neighbor" models you will build are based on Euclidean distance, scale is important. Scale and center the numeric and integer predictors (but not the variable you are predicting – we will preserve its scale for comparison with the error rate computed above)
    - Recall the scale() function
    - **Important**: do this before partitioning the data in the next step.
- Display the first 6 rows of your data frame. It should look like this:

	total.credit.card.limit	avg.percentage_credit_card_limit_used_last_year	saving.amount	checking.amount	yearly.salary	age	dependent.number	loan.repaid
1	3.05506481	2.192009	1.0790733	0.4067328	1.0585076	-0.35699118	1.2465575	Yes
2	0.09525897	2.192009	-0.8989617	-1.4223100	1.3119295	-0.98186923	-0.6612572	Yes
3	-1.06752189	2.192009	-0.9634773	-0.9958875	0.3600521	-1.21619851	0.1018687	No
4	-1.17322924	2.192009	-1.1822982	-0.2308219	-1.8218488	-0.82564972	1.6281204	No
5	-0.06330205	2.192009	-0.5517747	-0.4146963	-1.8218488	-0.04455215	-0.2796942	No
6	-0.48613146	2.192009	-1.0299882	-0.4498081	-1.0121349	-1.45052778	1.6281204	No

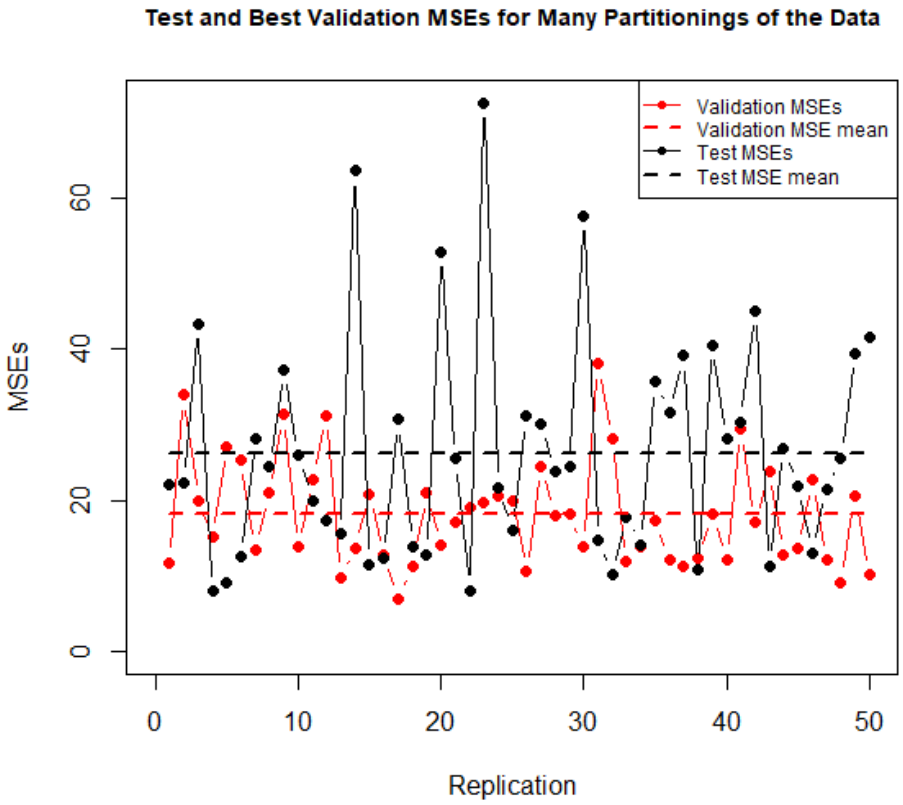
- Set the random seed to 5072
- Create training, validate and test data frames from the scaled data frame – use a 75/15/10 split.
- Display the first row of each of your three data frames. The row numbers of your train, validate and test data frames should be 6236, 13629 and 20, respectively.
- Create the following 6 objects:
    - train.x: the training rows from the scaled numeric predictors
    - validate.x: the validate rows from the scaled numeric predictors
    - test.x: the test rows from the scaled numeric predictors
    - train.y: the training rows from the loan.repaid column of the original data frame
    - validate.y: the validate rows from the loan.repaid column of the original data frame
    - test.y: the test rows from the loan.repaid column of the original data frame.
- Using the knn() function from the class package (not the knn.reg() function from the FNN package), predict the loan repayment (the loan.repaid variable) using all the other variables as predictors, as follows:
    - Construct 10 models with k equal to 19, 17, 15… 1, evaluating both the validate and training error rates for each k.
        - Note that for each k, you will need to execute the knn() function twice, once to predict the validate dependent variables and once to predict the training dependent variables in order to compute the error rates for each.
    - The knn() function requires 4 input parameters and produces a factor that contains the predictions. The input parameters (which are the same as the knn.reg() function) are:
        - A matrix or data frame containing the predictors associated with the training data

    - A matrix or data frame containing the predictors associated with the data for which we wish to make predictions
    - A factor or character vector containing the class labels for the training observations
    - A value for k, the number of nearest neighbors to be used by the classifier.
- Plot the training and validate error rates as a function of the k's. The x-axis should go from least flexible to most flexible – left to right.
- Print the k and associated error rate that produced the lowest training error rate. Do the same for the validate error rate.
    - My "best" validate error rate occurred with k = 11 and produced a validate error rate of 0.100895679662803
- Finally, predict loan.repaid for the test set using the optimal value of k that you found for the validate set, and compute (and display) the associated error rate.
- Your plot should look something like this (note that k again decreases along the x-axis)::



Question 3: Investigating the Variance of the knn.reg model on the home pricing dataset used in Part 1 (30%):

- Recall that there is such a large discrepancy between the MSE for the validate set and the MSE for the test set when using a value for k that was best for the validate set for both predictions.
- Recall also that the (informal) definition of variance of a model is the amount by which predictions change across different samples.
- In Question 1, we "tuned" the model by changing the level of flexibility (the k's) and evaluating those changes using the validation set (since the training set would have always indicated that more flexibility was better). We then evaluated the tuned model on a new sample – the test set - and observed a very large difference. In other words, the model with k=3 displayed a high level of variance.
- This begs the following question: Was this large change due to the fact that the test set was radically different from the evaluation set? That is, did we just get unlucky with the random split of the original data into training, validate and test sets?
- To answer this question, set the random seed to 5072 (once at the beginning of your Question 3 code) and repeat Part 1 with 50 different random 75/15/10 partitions of the data into train, validate and test sets, each time using the validate set to choose the best k (lowest validate MSE) and then using that k to evaluate the test MSE.
    - You don't need to compute and track the training MSE for this question, just the validate MSE for the best k and the test MSE for each of the 50 replications.
- Produce the following outputs:
    - Print the means and the (sample) standard deviations of both the "best" validate MSE's and the test MSE's (the "best" validate MSE's are those associated with the best k's for each random split).
    - Plot the validate and test MSE's. Add horizontal lines to the plot that show both the validate MSE mean and the test MSE mean (see below).
- Were we just unlucky in Question 1? Add a brief comment to your script explaining your answer.
- My numerical results were as follows:
    - mean.validate.errors: 18.1472
    - sd.validate.errors: 7.042044
    - mean.test.errors: 26.32294
    - sd.test.errors 14.69504
- My plot is on the following page:



Question 4: Predicting College Applications (up to 10%):

- The data set named applications.train.csv contains records from 600 colleges and universities about various aspects of their operations. The features are as follows:
  - Applications: Number of applications received
  - Accepted: Number of applications accepted
  - Enrolled: Number of new students enrolled
  - Top10.Percent: Pct. new students from top 10% of H.S. class
  - Top25.Percent: Pct. new students from top 25% of H.S. class
  - FT.Undergrads: Number of fulltime undergraduates
  - PT.Undergrads: Number of parttime undergraduates
  - OutOfState: Out-of-state tuition
  - RoomAndBoard: Room and board costs
  - Book.Costs: Estimated book costs
  - Personal.Costs: Estimated personal spending
  - PhD.Students: Pct. of faculty with Ph.D.'s
  - Terminal.Degrees: Pct. of faculty with terminal degree
  - Student.Faculty.Ratio: Student/faculty ratio
  - Percent.Donating.Alumni: Pct. alumni who donate
  - Spending.On.Students: Instructional expenditure per student
  - Graduation.Rate: Graduation rate

- Use the knn.reg() function to build a knn regression model to predict the Applications variable. The predictors have already been scaled. You may choose to use all or only some of these scaled predictors in the data set, but you may not change them.
    - Your answer will simply be the value of k that you think best determines the level of flexibility for the situation. Just provide it as a comment for your answer to this question.
    - If you choose to use only a subset of the variables as predictors, please also provide the R statement you used to omit the variables.
- I have withheld a test set of 177 observations. I will use your model to predict Applications for these unseen 177 observations, and grades for this question will be assigned as follows:
    - MSE's in the top quartile: 10%
    - MSE's in the next quartile: 8%
    - MSE's in the bottom half: 6% (provided your model works)