# PSTAT 131 - HW1

*April 15, 2020*

**Predicting Algae Blooms**

**Background** High concentrations of certain harmful algae in rivers constitute a serious ecological problem with a strong impact not only on river lifeforms, but also on water quality. Being able to monitor and perform an early forecast of algae blooms is essential to improving the quality of rivers. With the goal of addressing this prediction problem, several water samples were collected in different European rivers at different times during a period of approximately 1 year. For each water sample, different chemical properties were measured as well as the frequency of occurrence of seven harmful algae. Some other characteristics of the water collection process were also stored, such as the season of the year, the river size, and the river speed.

**Goal** We want to understand how these frequencies are related to certain chemical attributes of water samples as well as other characteristics of the samples (like season of the year, type of river, etc.)

**Data Description** The data set consists of data for 200 water samples and each observation in the available datasets is in effect an aggregation of several water samples collected from the same river over a period of 3 months, during the same season of the year. Each observation contains information on 11 variables. Three of these variables are nominal and describe the season of the year when the water samples to be aggregated were collected, as well as the size and speed of the river in question. The eight remaining variables are values of different chemical parameters measured in the water samples forming the aggregation, namely: Maximum pH value, Minimum value of $O_2$ (oxygen), Mean value of Cl (chloride), Mean value of $NO_3^-$ (nitrates), Mean value of $NH_4^+$ (ammonium), Mean of $PO_4^3$ (orthophosphate), Mean of total $PO_4$ (phosphate) and Mean of chlorophyll.

Associated with each of these parameters are seven frequency numbers of different harmful algae found in the respective water samples. No information is given regarding the names of the algae that were identified.

We can start the analysis by loading into R the data from the "algaeBloom.txt" file (the training data, i.e. the data that will be used to obtain the predictive models). To read the data from the file it is sufficient to issue the following command:

```
algae <- read_table2("algaeBloom.txt",
            col_names= c('season','size','speed','mxPH','mnO2','Cl','NO3',
                         'NH4','oPO4','PO4','Chla','a1','a2','a3','a4','a5','a6','a7'),
            na="XXXXXXX")
glimpse(algae)
```

1. *Descriptive summary statistics* Given the lack of further information on the problem domain, it is wise to investigate some of the statistical properties of the data, so as to get a better grasp of the problem. It is always a good idea to start our analysis with some kind of exploratory data analysis. A first idea of the statistical properties of the data can be obtained through a summary of its descriptive statistics.

   (a) Count the number of observations in each season using summarise() in dplyr.

   ```
   algae %>%
     group_by(season) %>%
     summarise(num_of_obs = n())
   ```

   ```
   ## # A tibble: 4 x 2
   ##   season num_of_obs
   ##   <chr>       <int>
   ## 1 autumn         40
   ## 2 spring         53
   ## 3 summer         45
   ```

```
## 4 winter         62
```

(b) Are there missing values? Calculate the mean and variance of each chemical (Ignore $a_1$ through $a_7$). What do you notice about the magnitude of the two quantities for different chemicals?

```
# Missing values check; returns a dataframe of missing value counts
algae %>%
  select(everything(), -c(a1:a7)) %>%
  summarise_all(funs(sum(is.na(.))))
```

```
## # A tibble: 1 x 11
##   season  size speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla
##    <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1      0     0     0     1     2    10     2     2     2     2    12
```

```
# Mean of each chemical
algae %>%
  select(everything(), -c(a1:a7)) %>%
  summarise_if(is.numeric, mean, na.rm = TRUE)
```

```
## # A tibble: 1 x 8
##    mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  8.01  9.12  43.6  3.28  501.  73.6  138.  14.0
```

```
# Variance of each chemical
algae %>%
  select(everything(), -c(a1:a7)) %>%
  summarise_if(is.numeric, var, na.rm = TRUE)
```

```
## # A tibble: 1 x 8
##    mxPH  mnO2    Cl   NO3      NH4  oPO4    PO4  Chla
##   <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl>  <dbl> <dbl>
## 1 0.358  5.72 2193.  14.3 3851585. 8306. 16639.  420.
```

I noticed the magnitude of the two quantities for different chemicals seem to be correlated, i.e. the chemicals with large variance values tend to have large mean values.

```
(c) Mean and Variance is one measure of central tendency and spread of data. Median and Median Absolute
```

2. Data visualization Most of the time, the information in the data set is also well captured graphically. Histogram, scatter plot, boxplot, Q-Q plot are frequently used tools for data visualization. Use ggplot for all of these visualizations.

   (a) Produce a histogram of *mxPH* with the title 'Histogram of mxPH' based on algae data set. Use an appropriate argument to show the probability instead of the frequency as the vertical axis. (Hint: look at the examples in the help file for function geom_histogram()). Is the distribution skewed?

   (b) Add a density curve using geom_density() and rug plots using geom_rug() to above histogram.

   (c) Create a boxplot with the title 'A conditioned Boxplot of Algal a1' for a1 grouped by size. (Refer to help page for geom_boxplot()).

   (d) Are there any outliers for *NO3* and *NH4*? How many observations would you consider as outliers? How did you arrive at this conclusion?

   (e) Compare mean & variance vs. median & MAD for *NO3* and *NH4*. What do you notice? Can you conclude which set of measures is more robust when outliers are present?

**Predicting Algae Blooms**

Some water samples contained unknown values in several chemicals. Missing data are very common in real-world problems, and may prevent the use of certain data mining techniques that are not able to handle missing values.

In this homework, we are going to introduce various ways to deal with missing values. After all the missing values have been taken care of, we will build a model to investigate the relationship between the variable a1 and other 11 predictors (season, size, speed, mxPH, mnO2, Cl, NO3, NH4, oPO4, PO4, Chla) utilizing cross-validation in the next problem.

**Dealing with missing values**

3. (a) How many observations contain missing values? How many missing values are there in each variable?

   (b) **Removing observations with missing values**: use filter() function in dplyr package to observations with any missing value, and save the resulting dataset (without missing values) as algae.del. Report how many observations are in algae.del. Hint: complete.cases() may be useful.

   (c) **Imputing unknowns with measures of central tendency**: the simplest and fastest way of filling in (imputing) missing values is to use some measures of central tendency such as mean, median and mode. Use mutate_at() and ifelse() in dplyr to fill in missing values for each chemical with its median, and save the imputed dataset as algae.med. Report the number of observations in algae.med. Display the values of each chemical for the 48th, 62th and 199th obsevation in algae.med. This simple strategy, although extremely fast and thus appealing for large datasets, imputed values may have large bias that can influence our model fitting. An alternative for decreasing bias of imputed values is to use relationships between variables.

   (d) **Imputing unknowns using correlations**: another way to impute missing values is to use correlation with another variable. For a highly correlated pair of variables, we can fill in the unknown values by predicting one based on the other with a simple linear regression model, provided the two variables are not both unknown. Compute pairwise correlation between the continuous (chemical) variables. Then, fill in the missing value for PO4 based on oPO4 in the 28th observation. What is the value you obtain? Hint: use lm() and predict() function.

   (e) **Questioning missing data assumptions**: When might imputation using only the observed data lead you to incorrect conclusions? In a couple of sentences, describe a scenario in which the imputed values of the chemical abundances in the algae data (imputed using either the median or correlation method) might be a poor substitute for the true missing values. Hint: look at the example from lecture 2.

**Estimating the Test Error with Cross Validation (CV)**

Using algae.med dataset obtained in (3c), we will build a linear regression model to predict the levels of algae type a1 based on 11 variables (season, size, speed, mxPH, mnO2, Cl, NO3, NH4, oPO4, PO4, Chla), and test generalization of model to data that have not been used for training.

4. **Cross-validation**: in class we talked about how to use cross-validation (CV) to estimate the "test error". In $k$-fold CV, each of $k$ equally sized random~ partitions of data (chunks) are used in a heldout set (called validation set or test set). After $k$ runs, we average the held-out error as our final estimate of the validation error. For this part, we will run cross-validation on only a single model, as a way to estimate our test error for future predictions (we are not using it here for model selection since we are considering only one model). Perform 5-fold cross-validation on this model to estimate the (average) test error.

   (a) First randomly partition data into 5 equal sized chunks. Hint: a simple way to randomly assign each observation to a chunk is to do the following. First, use cut(. . . , label=FALSE) to divide observation ids (1, 2, . . . ) into equal numbers of chunk ids. Then, randomize output of cut()by using sample().

(b) Perform 5-fold cross-validation with training error and validation errors of each chunk determined from (4a). Since same computation is repeated 5 times, we can define the following function for simplicity

```
do.chunk <- function(chunkid, chunkdef, dat){ # function argument
train = (chunkdef != chunkid)
Xtr = dat[train,1:11] # get training set
Ytr = dat[train,12] # get true response values in training set
Xvl = dat[!train,1:11] # get validation set
Yvl = dat[!train,12] # get true response values in validation set
lm.a1 <- lm(a1~., data = dat[train,1:12])
predYtr = predict(lm.a1) # predict training values
predYvl = predict(lm.a1,Xvl) # predict validation values
data.frame(fold = chunkid,
           train.error = mean((predYtr - Ytr)^2), # compute and store training error
           val.error = mean((predYvl - Yvl)^2)) # compute and store test error
}
```

First argument chunkid indicates which chunk to use as validation set (one of 1:5). Second argument chunkdef is chunk assignments from (4a). Third argument dat will be algae.med dataset. In order to repeatedly call do.chunk() for each value of chunkid, use functions lapply() or ldply(). Note that chunkdef and dat should be passed in as optional arguments (refer to help pages). Write the code and print out the train.error and val.error five times (e.g. for each chunk).

5. Test error on additional data: now imagine that you actually get new data that wasn't available when you first fit the model.

   (a) Additional data can be found in the file algaeTest.txt.

```
algae.Test <- read_table2('algaeTest.txt',
                col_names=c('season','size','speed','mxPH','mnO2','Cl','NO3',
                    'NH4','oPO4','PO4','Chla','a1'),
                na=c('XXXXXXX'))
```

This data was not used to train the model and was not (e.g. wasn't used in the CV procedure to estimate the test error). We can get a more accurate measure of true test error by evaluating the model fit on this held out set of data. Using the same linear regression model from part 4 (fit to all of the training data), calculate the "true" test error of your predictions based on the newly collected measurements in algaeTest.txt. Is this roughly what you expected based on the CV estimated test error from part 4?

**Cross Validation (CV) for Model Selection**

In this problem, we will be exploring a dataset of wages from a group of 3000 workers. The goal in this part is to identify a relationship between wages and age.

6. First, install the ISLR package, which includes many of the datasets used in the ISLR textbook. Look at the variables defined in the Wage dataset. We will be using the wage and age variables for this problem.

   (a) Plot wages as a function of age using ggplot. Your plot should include the datapoints (geom_point()) as well as a smooth fit to the data (geom_smooth()). Based on your visualization, what is the general pattern of wages as a function of age? Does this match what you expect?

   (b) In this part of the problem, we will find a polynomial function of age that best fits the wage data. For each polynomial function between $p = 0, 1, 2, \ldots 10$:

      i. Fit a linear regression to predict wages as a function of $age$, $age^2$, . . . $age^p$ (you should include an intercept as well). Note that $p = 0$ model is an "intercept-only" model.

ii. Use 5-fold cross validation to estimate the test error for this model. Save both the test error and the training error.

(c) Plot both the test error and training error (on the same plot) for each of the models estimated above as a function of $p$. What do you observe about the training error as $p$ increases? What about the test error? Based on your results, which model should you select and why?

Note: poly(age, degree=p, raw=TRUE) will return a matrix with $p$ columns, where the $p$-th column is $age^p$ . For the predictors in your regression use poly(age, degree=p, raw=FALSE). The raw=FALSE option returns predictors which are numerically more stable (it returns a matrix of "orthogonal polynomials"). Numerical stability can be an issue because $age^p$ can be very very large if $p$ is large. The orthogonal polynomials returned when raw=FALSE are rescaled to account for this so please use the raw=FALSE option. Hint: A function similar to do.chunk from problem 4 will be helpful here as well.