Predicting Chemical Solubility: An Exercise in Statistical Learning

SDS385
Elisa Ferracane - ef5884
Ashutosh Singh - as79592
Mythraye Vodnala
Chakameh Zahed
Pengfei Zhang

1 Introduction

The goal of this project is to predict a response variable using a variety of techniques. As a first step, an exploratory analysis of the data is conducted. After pre-processing the data, each of the following models are trained and tested using k-fold cross-validation: linear regression, Lasso, Ridge and Principal Components Regression (PCR). The final section analyzes the model results and the important predictors.

2 Data pre-processing

The original dataset consists of 228 predictors and 1 continuous response variable. The names suggest that we are trying to build a model for predicting solubility of organic compounds using the composition, structure and geometry of the compound. $FP001$ to $FP208$ are dummy variables (Fingerprints), predictors starting with $Num*$ are count predictors and the rest ($MolWeight, HydrophilicFactor, SurfaceArea1, SurfaceArea2$) are continuous predictors. Because of the large number of predictors, a preliminary analysis was conducted to determine which predictors to drop or combine. Next, we address issues of sparsity, collinearity, heteroscedasticity, outliers and scale.

2.1 Sparsity

While all the predictors starting with $Num*$ are count data, we did not treat all of them as categorical data. We first inspected the number of levels in each factor to determine which to treat as categorical, and which to treat as continuous:

```
ncol=dim(all_data_no_factors)[2]
mp=matrix(0,2,ncol)
for (i in 1:ncol){
      mp[1,i]=colnames(all_data_no_factors)[i]
      mp[2,i]=length(unique(all_data_no_factors[,i]))
 }
```

The result of this code is listed in Figure 1. As a first pass, we decided to treat all count predictors with fewer than 20 levels as categorical, and the rest as continuous. The following code was used to convert a count variable into a categorical variable:

```
all_data$NumRings <- as.factor(all_data$NumRings)
all_data <- cbind(all_data, model.matrix( ~ NumRings - 1,
                                  data = all_data))
```

However, we found that some levels were extremely sparse (only 1 data point in a level). This issue caused the principal component regression to fail, because the variance of these sparse predictors was practically zero. As a second pass, we inspected the predictors that had a variance lower than a certain threshold, 0.001, and left these as continuous variables. The following function was created for this purpose:

```
get_unit_var_predictors <- function(data, threshold){
  var_data <- var(data)
  unit_var_predictors <- vector()
  for(i in 1:ncol(var_data)){
    if(abs(var_data[i,i])<=threshold){
      unit_var_predictors <- c(unit_var_predictors,
                colnames(var_data)[i])
    }
  }
  return(unit_var_predictors)
}
```

Variable	Num Levels	Data Type	Reason
$NumSulfer$	5	categorical	<20 levels
$NumNitrogen$	7	categorical	<20 levels
$NumRings$	8	categorical	<20 levels
$NumDblBonds$	8	continuous	level 7 has 1 point
$NumChlorine$	11	continuous	level 10 has 1 point
$NumHalogen$	11	continuous	level 10 has 1 point
$NumOxygen$	12	continuous	levels 9, 11 have 1 point
$NumRotBonds$	16	continuous	level 13 has 1 point
$NumCarbon$	28	continuous	>20 levels
$NumNonHAtoms$	36	continuous	>20 levels
$NumNonHBonds$	39	continuous	>20 levels
$NumHydrogen$	42	continuous	>20 levels
$NumAtoms$	68	continuous	>20 levels
$NumBonds$	73	continuous	>20 levels

Table 1: Interpretation of count variables as either categorical or continuous.

2.2 Collinearity

We started by checking the correlation between predictors other than the Fingerprint predictors ($FPXXX$):

```
fpVars <- names(all_data)[grepl("FP", names(all_data))]
#excluding fingerprints
all_data_no_factors <- all_data[ , !(names(all_data) %in% fpVars)]
cor_data = cor(all_data_no_factors)
corrplot(cor_data, type = "upper")
```

The resulting correlation plot in Figure 1 indicated high correlation between $SurfaceArea1$ and $SurfaceArea2$, and between $NumAromaticBonds$ and $NumMultBonds$. Since both of these pairs have predictors with similar quantities, we created new predictors by averaging them, and dropped the original predictors.

```
all_data$Avg_surfaceArea = (all_data$SurfaceArea1 +
                                    all_data$SurfaceArea2)/2
all_data$Avg_bonds = (all_data$NumAromaticBonds +
                                    all_data$NumMultBonds)/2
drops <- c("SurfaceArea1", "SurfaceArea2", "NumAromaticBonds",
                                   "NumMultBonds")
all_data <- all_data[ , !(names(all_data) %in% drops)]
```

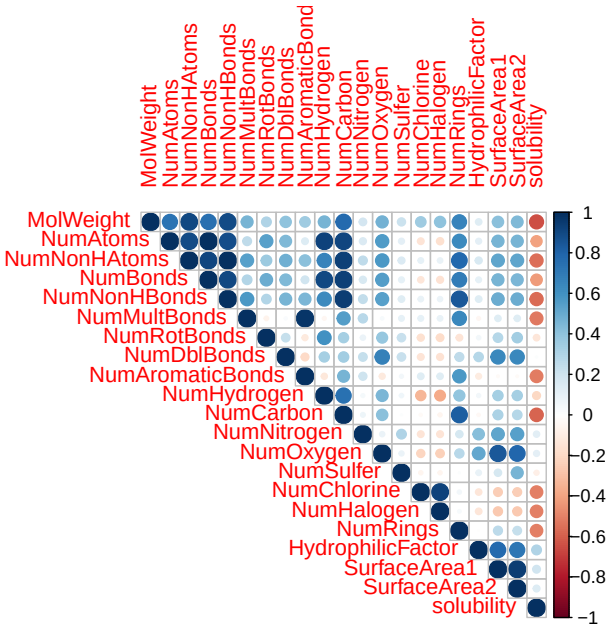


Figure 1: Correlation plot of predictors and response, excluding the fingerprints.

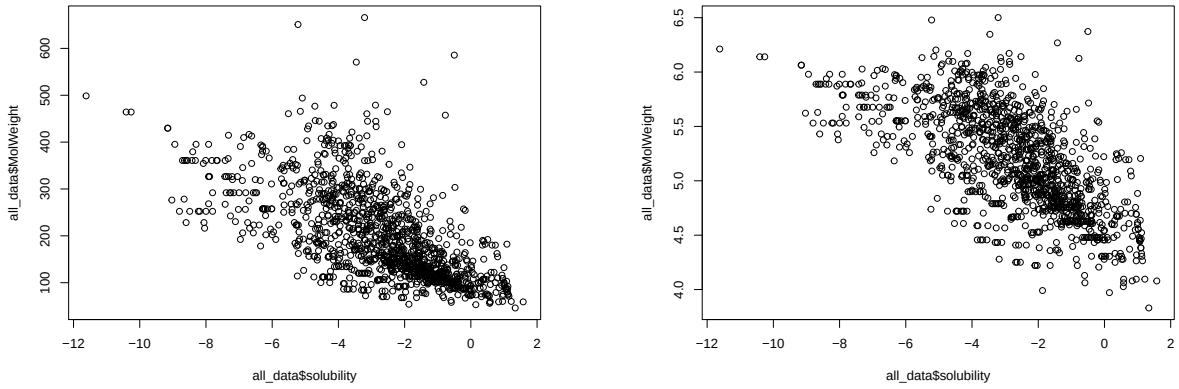
We then used a custom function to combine or drop all the highly correlated predictors (full function code in Appendix):

```
all_data <- drop_combine_correlated_predictors(all_data,
                                               response_name)
```

If the correlation factor was greater than or equal to 0.99, one of the two predictors was dropped. If the correlation was greater than 0.95 (but less than 0.99), the predictors were added together, and the two original predictors were dropped. The function was called a second time to determine whether any of the newly created variables were also highly correlated with other variables to spot three-way dependencies. Table 2 summarizes which predictors were dropped and combined in the first pass (top-half of the table) and in the second pass (bottom-half of the table).

Dropped	Combined
$FP058$	$FP001, FP163$
$FP063$	$FP013, FP014$
$FP087$	$FP025, FP180$
$FP134$	$FP032, FP033$
$FP203$	$FP032, FP183$
$NumAtoms$	$FP033, FP183$
$NumNonHAtoms$	$FP042, FP194$
	$FP046, FP166$
	$FP165, FP167$
$FP032\_FP033$	$FP032\_FP033, FP032\_FP183$
$FP032\_FP183$	

Table 2: List of variables that were either dropped or combined in a first pass, and in a second pass.



(a) Untransformed predictor

(b) Log-transformed predictor

Figure 2: Plots illustrating heteroscedasticity on the left, which is mitigated on the right by log-transforming the predictor.

2.3 Heteroscedasticity

Because the response variable was already log-transformed, we did not inspect whether it exhibited heteroscedasticity. However, we examined the continuous predictors using the following code:

```
continous_predictors <- c("Avg_bonds", "Avg_surfaceArea",
                          "HydrophilicFactor", "MolWeight")
hetero_results <- list()
for (i in 1:length(continous_predictors))
{

  model2 = lm(
      as.formula(paste("solubility", "~",
                       paste(continous_predictors[i]),
                       sep = "" )),data=all_data)

  hetero_results[[i]] <- gvlma(model2)
}
```

This code found that $MolWeight$ exhibited heteroscedasticity. In fact, a plot of this variable in Figure 2 shows the left-hand plot has the telltale funnel shape. Taking the log of the predictor variable helped to mitigate this problem, as shown in the right-hand plot.

2.4 Outliers

Outliers were tested for by using the $outlierTest$ in the R package car [Fox et al., 2016], which yielded 1 data point listed in Table 3. Interestingly, 1 of these 2 data points also had high leverage, as illustrated by the $influencePlot$ in Figure 3. After excluding the 2 high-leverage points, the R^2 of the linear regression model increased from 0.9199 to 0.9214. Below is the code to exclude these two points:

```
#look for outliers
basic_fit <- lm(solubility ~., data=all_data)
outliers <- influencePlot(basic_fit)
n <- 3
std_res <- as.numeric(tail(row.names(outliers[order(outliers$StudRes), ]), n))
outlierRows <- all_data[which.names(names=std_res, rownames(all_data)),]
```

Studentized residual	Bonferroni-p
-4.651195	0.0047195

Table 3: Studentized residuals of outlier, with Bonferroni-adjusted p-values.

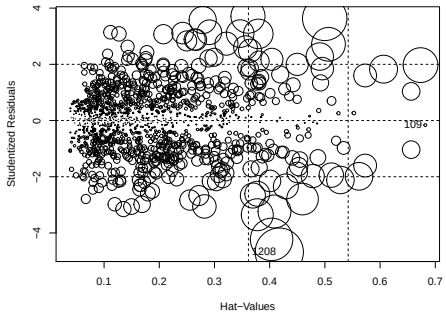
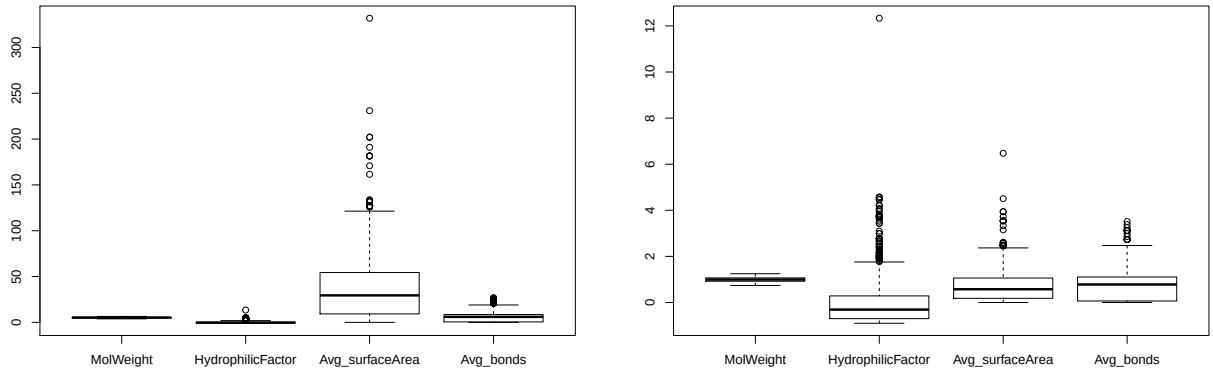


Figure 3: Bubble plot of residuals where 2 influential data points are marked.



(a) Before scaling

(b) After scaling

Figure 4: The boxplots illustrate the need for scaling the continuous predictors.

```
outTest <- outlierTest(basic_fit)

#drop high-leverage points
all_data <- all_data[-which.names(names=std_res, rownames(all_data)),]
```

2.5 Scale

Although one of the continuous predictors was log-transformed, they are still on considerably different scales, in particular $Avg_surfaceArea$, as illustrated in the left-hand panel of Figure 4. For a better estimate of the coefficients, the predictors were scaled, resulting in the right-hand panel of Figure 4.

3 Modeling

Models were created using ordinary least squares, Lasso, Ridge and PCR. The test RMSE is summarized in the Table 4. The following sections describe the results of each method in more detail.

Model	Test RMSE
OLS	0.722622
Lasso	0.669559
Ridge	0.679769
PCR	0.694552

Table 4: Test RMSE with 5-fold cross-validation.

3.1 Ordinary least squares

When first creating the linear regression with the lm command in R, the code generated a warning stating that the matrix was rank-deficient and automatically dropped the following 5 predictors: $NumHydrogen$, $NumRings7$, $NumSulfer4$, $NumNitrogen1$, $NumNitrogen6$. 5-fold cross-validation was performed on the training data, which yielded 0.722622. Notably, OLS performs the worst of the 4 models.

3.1.1 Lasso

Lasso regression was carried out using $cv.glmnet$ in the R $glmnet$ package with $alpha = 1$ [Friedman et al., 2009]. Interestingly, the lasso model shrinks 112 predictors to zeros, and achieves the best test RMSE of 0.669559.

3.1.2 Ridge

Ridge regression was carried out using $cv.glmnet$ in the R $glmnet$ package with $alpha = 0$. This model shrinks 12 predictors to almost zero, achieving a test RMSE of 0.679769.

3.1.3 PCR

Principal component regression was carried out using pcr in the R pls package. The fold with the lowest test RMSE of 0.694552 was achieved using 197 principal components. This point is illustrated with the red horizontal line in Figure 5.

4 Discussion

TODO: Talk about RMSE results and why Lasso is best, and how to improve model by combining sparse levels of the categorical variable, and transforming continuous predictors that had 0 values.

We determined the top 5 most important predictors by examining the loadings on the first principal component of PCR. The predictors with the highest absolute value loadings are listed in Table 5. The

Predictor
$Avg\_surfaceArea$
$FP067$
$NumNonHBonds$
$FP068$
$FP091$

Table 5: Top 5 important predictors.

most important predictors are consistent with the theory, as solubility of organic compounds is highly determined by the strength of hydrogen bonding which in turn is related to the number of hydrogen and oxygen atoms. The custom function(drop_combine_correlated_predictors) to combine and remove correlated predictors drops $NumAtoms$ and $NumNonHAtoms$, which is quite intuitive as the number of atoms equals the sum of number of each type of atoms, and the number of non hydrogen atoms equals the sum of number of each type of atom minus the number of hydrogen atom.

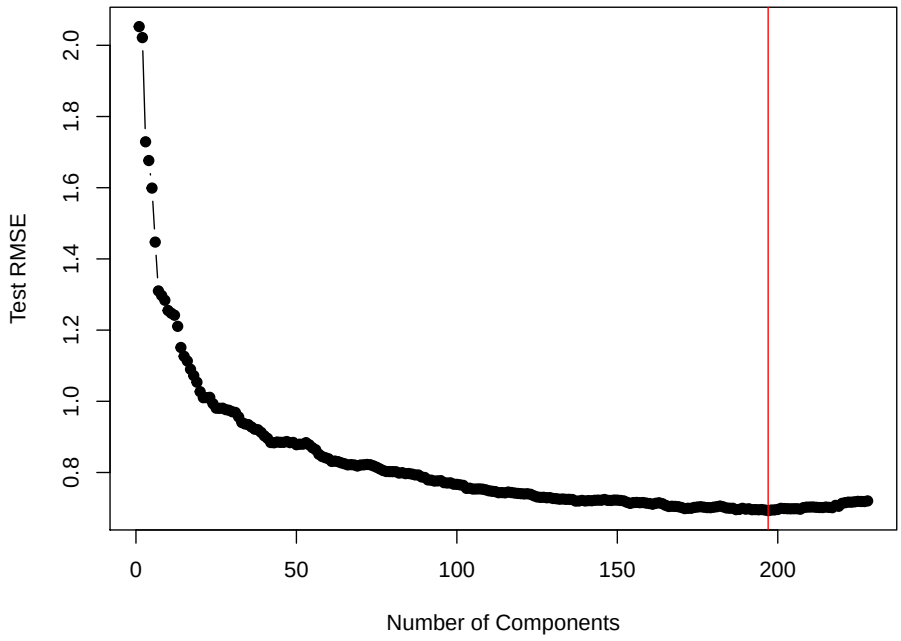


Figure 5: Test RMSE as the number of principal components is varied, with the red horizontal line indicating the best test RMSE at 197.

References

John Fox, Sanford Weisberg, Daniel Adler, Douglas Bates, Gabriel Baud-Bovy, Steve Ellison, David Firth, Michael Friendly, Gregor Gorjanc, Spencer Graves, et al. Package 'car', 2016.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1(4), 2009.