Section C: Dimensionality Reduction

Dataset preparation:

I downloaded the dataset of the Communities and Crime Data Set available at the UCI Machine Learning Repository and saved as a csv type named “communities.data”, and saved the variable names as “names.data”. The next step I used R functions read.csv and read.delim, then combined the data and dataname, and finally set the data variable is ‘data’.

1) Remove the "communityname" attribute: data['communityname'] <- NULL
2) Replace missing value with column mean: iterate all the dataset by column and set every column as numeric type, then get the mean of this column and set this column value equals ‘?’ as mean value.
3) Before split data, I set the seed is 100. Then I used the function sample to get the Boolean mask of training data is 60%. Then I applied the Boolean mask on dataset as follow: TS <- data[train_ind,]; VS <- data[-train_ind,]

I. Baseline Regression Model:

1) Fitting a linear model: In this part I used the function ‘lm’ to do the linear regression in R. Before do the model training, I set the system time using proc.time() to get the current time, and used it again to get the end time, and the difference between end time and start time is the elapsed time of training model. In following part I used the same method to count the model elapsed time.
   In fitting model, I used the ‘ViolentCrimesPerPop’ as the target variable and used others as variables on TS training set. Then I saved the model as ‘fit_model’.
2) Validate the training model on validation set:
   In this part I used the function of predict.lm to validate the training model on validation dataset and saved the prediction named ‘pred’. The next step is to

calculate the the Sum of Square Errors (SSE), the Root Mean Square Error (RMSE), the Relative Square Error (RSE), and the Coeffient of Determination (R2) on validation dataset. The calculation as following:

```
SSE <- sum((VS$ViolentCrimesPerPop - pred)^2)
RMSE <- sqrt(mean((VS$ViolentCrimesPerPop - pred)^2))
RSE <- sum((VS$ViolentCrimesPerPop -
  pred)^2)/sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
SST <- sum((mean(VS$ViolentCrimesPerPop)-VS$ViolentCrimesPerPop)^2)
R_square <- 1-SSE/SST
```

In the following parts, when calculate these ratios are all using the similar equation in this part.

II. Feature Selection: Sequential Subset Selection

1) To do Sequential Subset Selection I used the function named 'step' in R stat package. step uses add1 and drop1 repeatedly; it will work for any method for which they work, and that is determined by having a valid method for extractAIC. When the additive constant can be chosen so that AIC is equal to Mallows' Cp, this is done and the tables are labelled appropriately. The set of models searched is determined by the scope argument.
2) The parameters of step function are fit_model which got from baseline model and direction is both which means using forward and backward. I saved the step model to 'model_step'. From dim(model_step$model), we can see that there exist 51 variables left after using step function.
3) In the validation part, I still used the predict function to make prediction on validation dataset VS:

   ```
   prediction <- predict(model_step, newdata = VS[,1:126])
   ```

   The calculation of validation ratios is same as above.

III. Feature Selection: Ranking Attributes

1) Multivariate Adaptive Regression Splines (MARS) models include a backwards elimination feature selection routine that looks at reductions in the generalized cross-validation (GCV) estimate of error. The varImp function tracks the changes in model statistics, such as the GCV, for each predictor and accumulates the reduction in the statistic when each predictor's feature is added to the model. Using varImp(object) tracks the reduction in the generalized cross-validation statistic as terms are added.
   The caret R package provides tools automatically report on the relevance and importance of attributes in the data and can select the most important features out.
2) Apply the function of varImp on the fit_model getting from the baseline model and ranking the overall importance of the result, and finally get the top 50 important variables.
3) Then I append all the variables as a string and apply linear regression model using lm function. The target is still “ViolentCrimesPerPop” and other variables are the string I just got and apply the model on Training set “TS”.

IV. Feature Extraction: Principal Components Analysis

1) Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them. The function used in PCA is called prcomp which comes with the default "stats" package.
2) Apply the function on the training dataset:
   pca <- prcomp(TS[,1:126], retx=TRUE, center=TRUE, scale=TRUE)

There are 126 components (because drop a column and a column is target) were constructed. The minimum number of components needed to capture at least 90% of the data variance is 28, because the cumulative proportion of PC28 is 0.90476.

3) Validate the validation set using pca components:
   Now using just the 28 principal components to explain at least 90% of the data variance to build the linear regression model. The next step is to use PCA model to transfer the validation data to pca components, pred.vs <- predict(pca, VS[,1:126]), pca is the model getting from above. The last step is to use the new linear regression model apply on transferred validation pca components.

V. Feature Extraction: Factor Analysis (FA)

1) The factor.pa( ) function in the psych package offers a number of factor analysis related functions, including principal axis factoring.
2) Build the model:
   fa_fit <- factor.pa(TS[,1:126], nfactors=30)
   I set the factor number is 30 because this amount is near to the amount of pca components captured 90% variance.
   Then I apply a linear regression model on the new 30 factors.
3) Before do the validation part, I used the same factor model applied on the validation data set and get the new factors of validation set. Then I apply the new linear model on the validation factors.

VI. Comparison of Results

	Baseline	**Step**	**Relief**	**PCA**	**FA**
Number of attributes used to construct the linear regression model	127	127	127	28	30

Number of attributes appearing in the linear regression model	127	51	50	28	30
Time taken constructing the linear regression model	0.049	112.627	0.015	0.114	0.613
Sum of Square Errors(SSE)	15.236	15.550	14.284	14.700	14.833
Root Mean Square Error(RMSE)	0.1382	0.1396	0.1338	0.1357	0.1363
Relative Square Error(RSE)	0.3811	0.3890	0.3573	0.3677	0.3710
Coeffient of Determination(R^2)	0.6189	0.6110	0.6427	0.6323	0.6289

From the table we can see that, using top 50 important variables model has the largest R_square. However, PCA only uses 28 principle components and ranks the second best R_Square, which is 0.632. It is surprise that using stepwise model get the least R_square value and model takes the longest time. The stepwise model uses 51 variables but the result of stepwise is worse than top 50 important variable model. The reason I think is the two models uses different criteria to assess the importance of the variables, and different libraries use different methods to solve the problem. Moreover, the top 50 importance variables, PCA, and FA are better than baseline model and stepwise model, and the time taken of top 50 importance variables model is the fastest model among these methods.