Classification Modeling Applied to Employee Attrition

Group No.09
Student Names: Yuhang Diao and Hongtu Zhang

Executive Summary: In this case study, the goal is to help the Human Resources Department of an organization discover employees with potential intention to resign and reduce attrition rate by paying more attention to those employees and taking effective measures, such as taking advice from them, learning about their practical demands and improving their working conditions. The source of data is IBM HR Analytics Attrition and Performance dataset. This case focuses on k-Nearest Neighbors, Naive Bayes, CART, and Logistic Regression methods. Based on performance analysis, the CART model performs better than others on this dataset. Classification tree model is useful for variable selection, with the most important predictors usually showing up at the top of the tree and it is also a useful classifier in cases where horizontal and vertical splitting of the predictor space adequately divides the classes. We find that, predictor variables, such as Business Travel, Marital Status, and Over Time, have the largest influence on staff attrition. The organization should pay more attention to these in order to reduce the level of attrition.

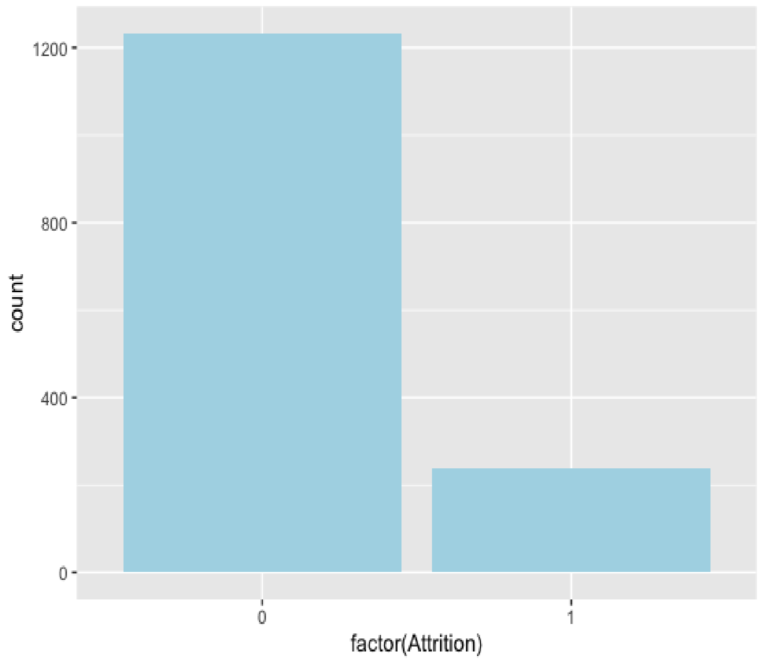
I. Background and Introduction

Nowadays, the staff attrition and turnover have become significant problems and concerns for the Human Resources Department of a company with the rapid development and increase in the number of small- and medium-sized enterprises. By definition, staff attrition refers to the loss of employees through the process, such as retirement, resignation, cancelation of a job position, personal or family issues, or other causes. With attrition, an employer will take much more effort to fill the position left by the former employee since it is costly to lose employees through the staff attrition and turnover. For example, the costs could be associated with the loss of productivity, job recruitment, interviewing some candidates, and training. Such replacement cost could be low for entry-level positions, but it will be significantly higher for professional, managemental, and technical roles. If the Human Resources Department of a company could predict whether a valuable or resourceful employee would leave the company, there are steps they can take to keep this employee rather than waiting for months to train a new staff to fill the position. The goal of this case study is to help companies discover employees with potential intention to resign and reduce attrition rate by paying more attention to those employees and taking effective measures, such as taking advice from them, learning about their practical demands and improving their working conditions.

II. Data Exploration and Visualization

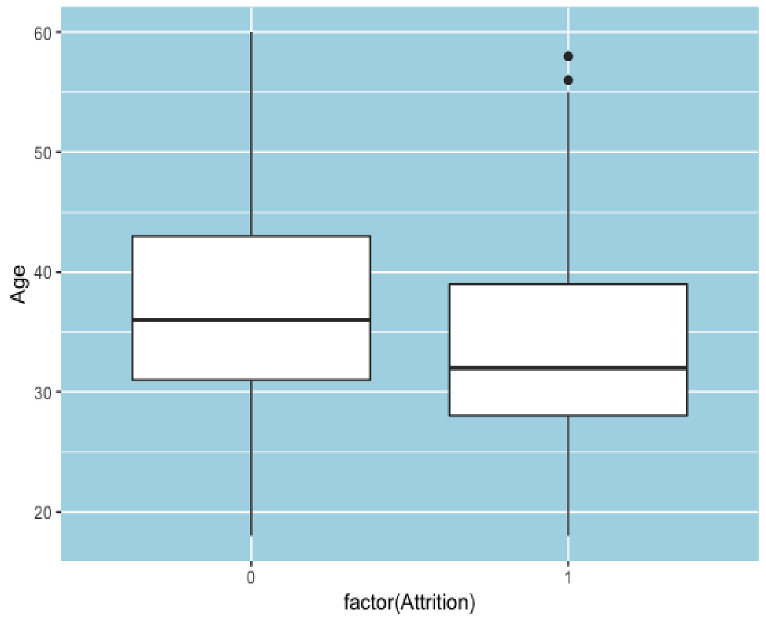
In this section, we will use some data visualization techniques to get a general overview of our dataset. First of all, we generate a bar chart to show the count of attrition and non-attrition in our data. The result is shown in Figure 2.1. We can see that we are dealing with an imbalanced data since more than 80% of employees did not leave the company while 20% did leave.

Figure 2.1



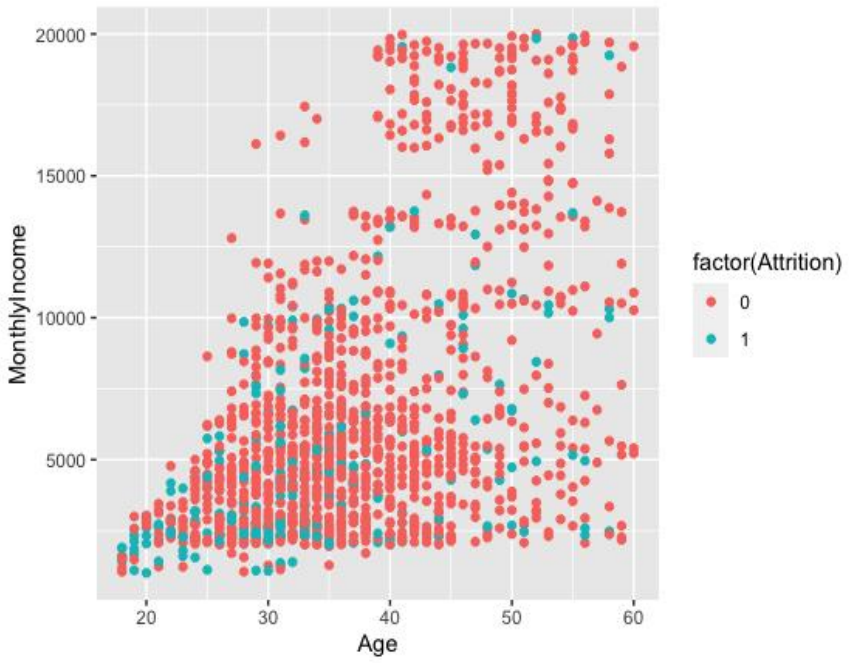
After that, we use a boxplot to get a general idea about the relationship between Age and Attrition. From Figure 2.2, we can see that the number of people between 25 and 40 years old who decide to leave the company is larger than that over 40 years old. The box plot shows that there are 2 outliers in the data.

Figure 2.2

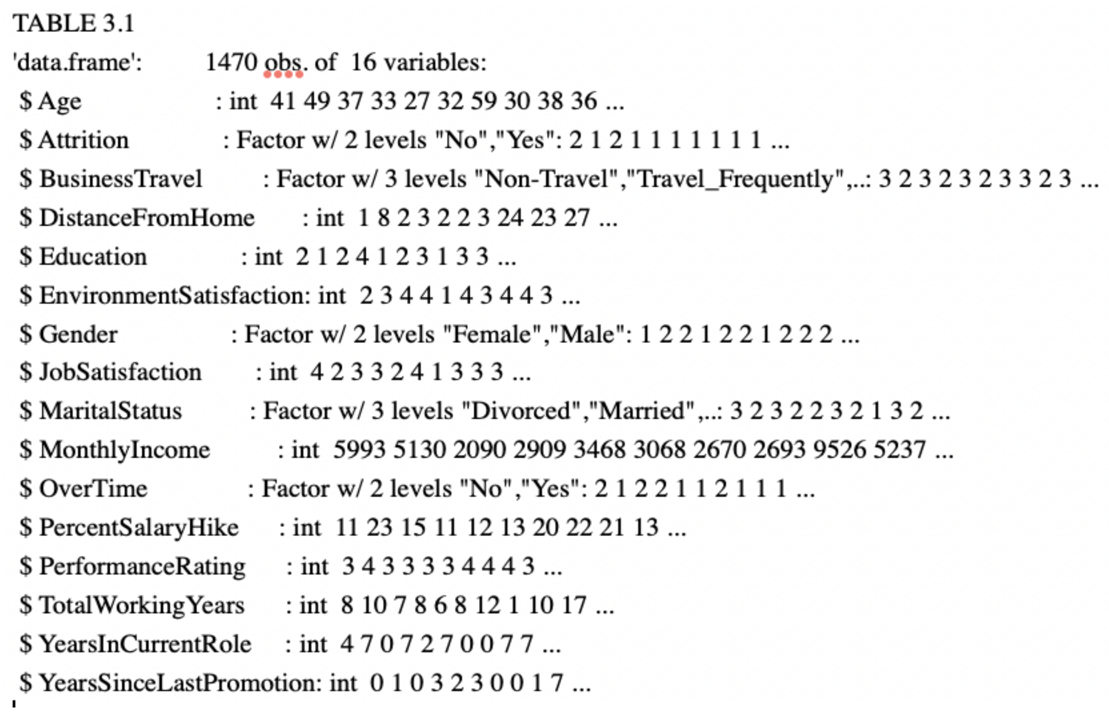


Furthermore, we generate a scatter plot to visualize the relationship between Age, Monthly Income and Attrition. As shown in Figure 2.3, we can conclude that older people with higher income are less likely to leave the company.

Figure 2.3



III. Data Preparation and Preprocessing



```
TABLE 3.1
'data.frame':       1470 obs. of  16 variables:
 $ Age                 : int  41 49 37 33 27 32 59 30 38 36 ...
 $ Attrition           : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...
 $ BusinessTravel       : Factor w/ 3 levels "Non-Travel","Travel_Frequently",..: 3 2 3 2 3 2 3 3 2 3 ...
 $ DistanceFromHome     : int  1 8 2 3 2 2 3 24 23 27 ...
 $ Education           : int  2 1 2 4 1 2 3 1 3 3 ...
 $ EnvironmentSatisfaction: int  2 3 4 4 1 4 3 4 4 3 ...
 $ Gender              : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...
 $ JobSatisfaction      : int  4 2 3 3 2 4 1 3 3 3 ...
 $ MaritalStatus        : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...
 $ MonthlyIncome        : int  5993 5130 2090 2909 3468 3068 2670 2693 9526 5237 ...
 $ OverTime            : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...
 $ PercentSalaryHike    : int  11 23 15 11 12 13 20 22 21 13 ...
 $ PerformanceRating    : int  3 4 3 3 3 3 4 4 4 3 ...
 $ TotalWorkingYears    : int  8 10 7 8 6 8 12 1 10 17 ...
 $ YearsInCurrentRole   : int  4 7 0 7 2 7 0 0 7 7 ...
 $ YearsSinceLastPromotion: int  0 1 0 3 2 3 0 0 1 7 ...
```

Based on the data summary in Table 3.1, there are 1470 observations (rows), 35 features (16 variables are kept) in our dataset, and there is no missing data. Also, there are only two data types in this dataset, which are factors and integers. Variable Attrition is the response variable, which indicates whether this employee will leave the company. Based on our domain knowledge, we could remove some irrelevant variables, such as Over 18, Standard Hour, and etc. After that, we will transform some integer variables with a large range of numerical values into smaller bins. Also, we will convert the categorical variables such as Business Travel and Marital Status to dummy variables.

IV. Data Mining Techniques and Implementation

Various data mining techniques can be used to mine the data collected from this dataset. None of those technique is universally better than another. For this case study, we will focus on k-Nearest Neighbors, Naive Bayes, CART, and Logistic Regression.

k-Nearest-Neighbors Method (k-NN)

The idea in k-nearest-neighbors method is to find the nearest k neighbors to the record to be classified. We can use a majority rule to classify the record, where the record is classified as a member of the majority class of the k neighbors. In this case study, we can use k-NN to create segments based on employee proximity to other similar employees in our data. Then, we can use these neighboring records to classify the new record as whatever the predominant class is among the nearby records.

We first normalize our data and partition the data into training data (882 observations) and validation data (588 observations). Then, we need to choose the proper value of *k*. Usually, we would choose k>1. The advantage of doing this is that higher value of k reduces the risk of overfitting due to the noise in the training dataset. If k is too low, we might have an overfitting model. However, if k is too high, the model might not capture the local structures. In order to choose the k with the best classification performance, we use the training data to classify the records in the validation dataset and compute error rates for a various selection of k. The results are shown in Table 4.1. We would choose k = 7 since it yields the highest accuracy rate in the validation set.

Table 4.1

k = 1	0	1	**k = 2**	0	1	**k = 3**	0	1	**k = 4**	0	1
0	444	62	0	443	65	0	466	75	0	470	73
1	50	32	1	51	29	1	28	19	1	24	21

k = 5	0	1	**k = 6**	0	1	**k = 7**	0	1
0	480	77	0	479	79	0	486	80
1	14	17	1	15	15	1	8	14

Once k is chosen, we can run the algorithm to generate classifications of new records.

The Naive Bayes Classifier

The basic idea behind the Naïve Bayes Classifier is to find records similar to the given record to be classified. Then, we determine what classes they all belong to and which class is more prevalent and assign that class to the new record. Before running Naïve Bayes, we first convert numerical independent variables to categorical. The table 4.2 below is the results.

Table 4.2

```
Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)
A-priori probabilities:
Y         0            1
     0.8378685    0.1621315
Conditional probabilities:
    Age
Y       2            3            4            5            6
  0  0.06359946   0.34100135   0.40054127   0.14614344   0.04871448
  1  0.16783217   0.44055944   0.20979021   0.12587413   0.05594406
   BusinessTravel
Y         0            1
  0   0.8281461    0.1718539
  1   0.7272727    0.2727273
   DistanceFromHome
Y        0            1            2            3            4            5            6
  0  0.29364005   0.24086604   0.20838972   0.07713126   0.06765900   0.07036536   0.04194858
  1  0.23776224   0.23076923   0.18881119   0.13286713   0.06993007   0.11188811   0.02797203
```

```
   Education
Y         1           2           3           4           5
  0  0.10419486  0.19756428  0.37212449  0.29093369  0.03518268
  1  0.13986014  0.19580420  0.41258741  0.23076923  0.02097902
   EnvironmentSatisfaction
Y         1          2           3           4
  0  0.1637348   0.2043302   0.3125846   0.3193505
  1  0.2937063   0.2027972   0.2587413   0.2447552
   Gender
Y         1          0
  0  0.5926928   0.4073072
  1  0.6713287   0.3286713
   JobSatisfaction
Y        1           2           3          4
  0  0.1826793   0.1867388   0.2949932  0.3355886
  1  0.2517483   0.1888112   0.3216783  0.2377622
   MaritalStatus
Y        1           2           3
  0  0.2814614   0.4695535   0.2489851
  1  0.4755245   0.3566434   0.1678322
MonthlyIncome
Y         0            1             2             3             4             5            6           7
  0  0.004059540  0.387009472  0.303112314  0.123139378  0.058186739  0.037889039
     0.069012179  0.017591340
  1  0.041958042  0.608391608  0.160839161  0.111888112  0.048951049  0.006993007
     0.013986014   0.006993007
   OverTime
Y         0            1
  0  0.7726658    0.2273342
  1  0.4545455    0.5454545
   PercentSalaryHike
Y          1           2
  0   0.5588633    0.4411367
  1   0.5384615    0.4615385
    PerformanceRating
Y          3            4
  0   0.8470907    0.1529093
  1   0.8391608    0.1608392
   TotalWorkingYears
Y         0             1             2             3             4              5          6      7      8
  0  0.050067659  0.258457375  0.347767253  0.131258457   0.098782138   0.058186739
     0.039242219  0.014884980  0.001353180
  1  0.223776224  0.300699301  0.272727273  0.076923077   0.076923077   0.034965035
     0.006993007  0.000000000  0.006993007
   YearsInCurrentRole
Y         0             1             2             3             4             5             6
  0  0.165087957  0.415426252  0.216508796  0.139377537  0.037889039  0.020297700  0.005412720
  1. 0.370629371  0.405594406  0.153846154  0.055944056  0.006993007  0.006993007  0.000000000
   YearsSinceLastPromotion
Y         0            1            2            3            4            5
  0   0.62922869   0.18538566   0.11231394   0.02706360   0.03247632   0.01353180
  1   0.66433566   0.16083916   0.10489510   0.03496503   0.02097902   0.01398601
```

The first part of the output in Table 4.2 shows the ratio of attrition and non-attrition employees in the training data. 84% of employees in the training data are in the non-attrition group and 26% are in the attrition group. The second part shows the conditional probabilities for each class, as a function of the predictor variables. We can see that the conditional probability of male attrition (0.67) is larger than female attrition (0.33). Also, the conditional probability of the attrition of employees with age between 30 and 40 is the highest, which is 0.44, and the probability of the attrition of employees with 60+ age is the lowest, which is 0.06. For a given new record, we can compute the probability of attrition by using the given characteristics based on the Naïve Bayes Assumption:

$$P\left(x_1,\cdots,x_p \mid C_i\right) = P(x_1 \mid C_i)P(x_2 \mid C_i)\cdots P(x_p \mid C_i)$$

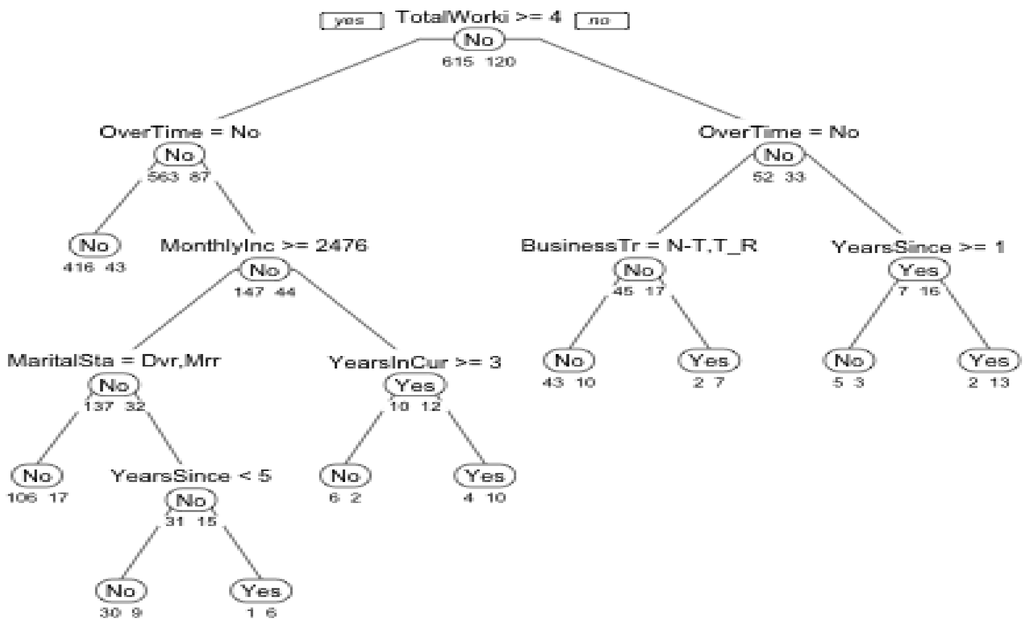
Then we can calculate the result by:

$$P\left(C_i \mid x_1,\cdots,x_p\right) = \frac{\left[P(x_1 \mid C_i)P(x_2 \mid C_i)\cdots P(x_p \mid C_i)\right]P(C_i)}{\left[P(x_1 \mid C_1)P(x_2 \mid C_1)\cdots P(x_p \mid C_1)\right]P(C_1)+\cdots+\left[P(x_1 \mid C_p)P(x_2 \mid C_p)\cdots P(x_p \mid C_p)\right]P(C_p)}$$

Classification Tree

After randomly partitioning the data into training (735 records), validation (441 records) and testing (294 records), we use the training dataset to construct a tree. A default tree with 9 splits is shown in the Figure 4.1. From the tree, we can see that, after a continuous splitting process, the eventual classification of attrition appears in the terminal nodes. Of the ten terminal nodes, four lead to classification of "attrition" and six lead to classification of "not attrition".

Figure 4.1



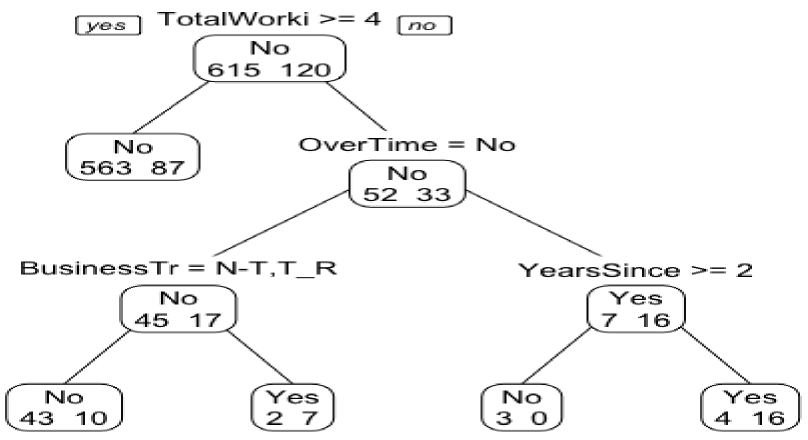
There are large quantities of splits in the full-grown tree. We can compare the performance between the training dataset and the validation dataset in terms of the default tree and the full-grown tree. When cp = 0, there are 117 splits in the full tree, the error rate is equal to 0 for the training dataset. However, we apply the model to the validation dataset, the error rate sharply increases to 21%, since the full-grown tree overfits the training dataset to perfect accuracy. Then we use a pruned tree to avoid overfitting.

Table 4.3

	CP	nsplit	rel error	xerror	xstd
1	0.0375000	0	1.000000	1.00000	0.083503
2	0.0250000	3	0.883333	0.98333	0.082939
3	0.0166667	4	0.858333	0.96667	0.082366
4	0.0138889	12	0.708333	1.05000	0.085147
5	0.0125000	18	0.625000	1.10000	0.086720
6	0.0111111	24	0.550000	1.10000	0.086720
7	0.0083333	30	0.483333	1.11667	0.087230
8	0.0059524	58	0.250000	1.17500	0.088957
9	0.0044872	69	0.183333	1.20833	0.089905
10	0.0041667	82	0.125000	1.27500	0.091724
11	0.0033333	98	0.058333	1.27500	0.091724
12	0.0027778	108	0.025000	1.30833	0.092595
13	0.0000100	117	0.000000	1.30833	0.092595

Pruning the tree with the validation data solves the problem of overfitting, but it does not address the problem of instability. We repeatedly use cross-validation to avoid relying on just one partition of the data into training and validation. We could build the best-pruned tree by using the estimated standard error of the cross-validation error (xstd) to prune the tree even further; we can also add one standard error to the minimum xerror. In this case, the tree in row 3 has the lowest xstd; besides, the sum of the xerror and standard error is also the lowest. This tree is pruned back from the largest tree using the complexity parameter value CP = 0.0166667 as shown in Table 4.3. The best-pruned tree is shown in Figure 4.2.

Figure 4.2



Logistic Regression
Logistic regression is used to predict classes and focuses on binary classification. After estimating the probabilities of belonging to each class, we use a cutoff value on the probabilities to classify each case in one of the classes. We partition the data into training data (882 records) and validation data (588 records). By building a general linear model with family = "binomial" to fit the logistic regression of the training data, we can get the coefficients in Table 4.4.

Table 4.4

```
Coefficients:
                          Estimate      Std. Error    z value    Pr(>|z|)
(Intercept)               2.07526190    1.25343476    1.656      0.097791 .
Age                      -0.03376546    0.01607638   -2.100      0.035701 *
BusinessTravel            0.76092138    0.25062317    3.036      0.002396 **
DistanceFromHome          0.04766945    0.01277172    3.732      0.000190 ***
Education                 0.04064876    0.10369287    0.392      0.695050
EnvironmentSatisfaction  -0.34572030    0.09741937   -3.549      0.000387 ***
Gender                    0.57793488    0.22581180    2.559      0.010486 *
JobSatisfaction          -0.39354067    0.09580910   -4.108      0.00003998780 ***
MaritalStatus            -0.84396470    0.15911297   -5.304      0.00000011318 ***
MonthlyIncome            -0.00009957    0.00004528   -2.199      0.027876 *
OverTime                  1.38255108    0.22413535    6.168      0.00000000069 ***
PercentSalaryHike        -0.00706868    0.04662631   -0.152      0.879500
PerformanceRating         0.11180288    0.46975742    0.238      0.811880
TotalWorkingYears        -0.01859933    0.03059161   -0.608      0.543196
YearsInCurrentRole       -0.18776905    0.04763324   -3.942      0.00008081328 ***
YearsSinceLastPromotion   0.18176228    0.04795916    3.790      0.000151 ***
```

Then we get the first five predicted probabilities of the validation data as Table 4.5 based on the logistic response function:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q)}}$$

We set a default cutoff value 0.5, if the predicted probability is larger than 0.5, the predicted result of the response variable is 1; otherwise, it’s 0.

Table 4.5

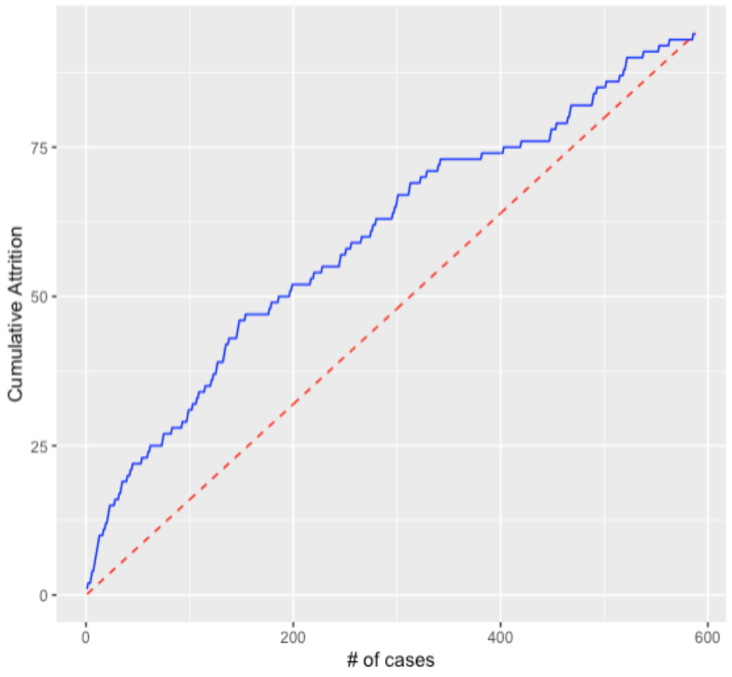
	Actual Class	Predicted Probability
2	0	0.03005665
4	0	0.21561445
5	0	0.28817735
6	0	0.10372021
10	0	0.17274979

V. Performance Evaluation

k-Nearest-Neighbors Method (k-NN)

To evaluate the performance of our k-NN algorithm, we can plot a lift chart. The lift chart is shown in Figure 5.1. This curve is compared to assigning a naive prediction to each record and accumulating these average values, which results in a diagonal line. The farther away the lift curve is from the diagonal benchmark line, the better the model is doing in separating records with high-value outcomes from those with low-value outcomes.

Figure 5.1



As we can see from Figure 5.1, the lift curve of our k-NN method is doing well compared to the baseline model. However, when the number of cases increases to a very large value, the predictive performance is not so good compared to that of lower number of cases. The error rate of the k-NN model is 15.3% derived from Table 5.1.

Table 5.1

	Predicted Class	
Actual Class	1	0
1	12	5
0	85	486

The Naive Bayes Classifier

In order to evaluate the performance of Naïve Bayes Classifier, we can generate a confusion matrix to see its accuracy. Table 5.2 shows the confusion matrix for the training set, and Table 5.3 shows the confusion matrix for the validation set. We can see that the overall error rate is about 15% for both sets.

Table 5.2
```
Confusion Matrix and Statistics          Accuracy: 0.8424
pred.class1    0    1                    95% CI: (0.8167, 0.8658)
       0     687   87                    P-Value [Acc > NIR]: 0.378109
       1      52   56                    Kappa: 0.3564
                                         Mcnemar's Test P-Value: 0.003929

Sensitivity: 0.9296                      Specificity: 0.3916
Pos Pred Value: 0.8876                   Neg Pred Value: 0.5185
Prevalence: 0.8379                       Detection Rate: 0.7789
Detection Prevalence: 0.8776             Balanced Accuracy: 0.6606
```

Table 5.3
```
Confusion Matrix and Statistics          Accuracy: 0.8469
           Reference                     95% CI: (0.8153, 0.8751)
Prediction   0    1                      P-Value [Acc > NIR]: 0.35087
      0    458   54                      Kappa: 0.3823
      1     36   40                      Mcnemar's Test P-Value: 0.07314

Sensitivity: 0.9271                      Specificity: 0.4255
Pos Pred Value: 0.8945                   Neg Pred Value: 0.5263
Prevalence: 0.8401                       Detection Rate: 0.7789
Detection Prevalence: 0.8707             Balanced Accuracy: 0.6763
```

Classification Tree

The best-pruned tree for the attrition has four splits, and the error rate of the testing data (Table 5.4) is 14.7%, lower than that of the full tree model based on the classification performance of the validation data (Table 5.5).

Table 5.4

	Predicted Class	
Actual Class	1	0
1	50	3
0	47	241

Table 5.5

	Predicted Class	
Actual Class	1	0
1	27	53
0	40	321

Logistic Regression
The overall error rate of the logistic regression model is 15.14% as shown in Table 5.6.

Table 5.6

	Predicted Class	
Actual Class	1	0
1	21	74
0	15	478

We plot a lift chart to evaluate the performance of our logistic regression model. From Figure 5.2, we can see that the model is doing well compared to the baseline. However, when the number of cases increases to a very large value, the predictive performance will not get improved significantly.

Figure 5.2

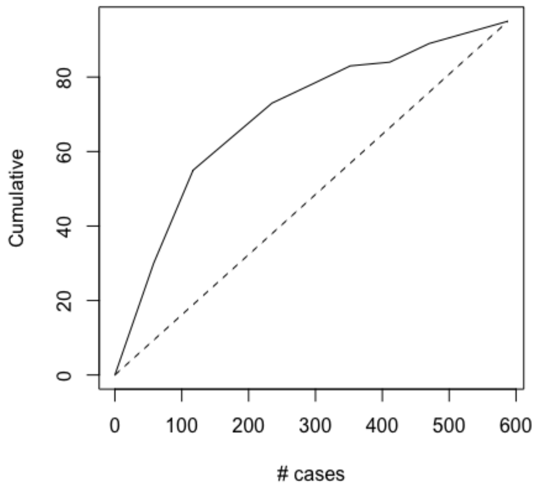


Table 5.7

Method	k-NN	Naive Bayes Classifier	Classification Tree	Logistic Regression
Error rate	15.3%	15.3%	14.7%	15.14%

According to the performance analysis shown in Table 5.7, the Classification Tree has the lowest error rate, so this method is the best one to classify the employee attrition on this dataset.

VI. Discussion and Recommendation

k-NN is a simple and nonparametric method. It requires no assumption about normal distribution. It is effective at capturing complex interactions among different variables in a dataset without having to define a statistical model. In a large enough dataset, the k-NN method performs fairly well, especially when each class is determined by multiple combinations of predictor values.

However, there are difficulties with the practical application of k-NN method. Firstly, it requires that the size of the training set increases exponentially with the number of predictors. Furthermore, it takes a long time to find distances to all the neighbors and then identify the nearest one. Also, in order to eliminate the “curse of dimensionality”, we need to limit the number of predictors using dimension reduction methods.

Naïve Bayes Classifier is a simple and computationally efficient method. It handles purely categorical data well and works well with very large data sets. However, it requires a large number of records. Furthermore, it could cause problems when a predictor category is missing in the training set since it will be assigned 0 probability, ignoring other predictors. A good performance could be obtained when the goal is to classification or ranking probability, but we could get biased results when the goal is to estimate the actual probability.

Tree models are good off-the-shelf classifiers and predictors. They are also useful for variable selection, with the most important predictors usually showing up at the top of the tree. Trees require relatively little effort from users and are also intrinsically robust to outliers. Classification trees are useful classifiers in cases where horizontal and vertical splitting of the predictor space adequately divides the classes. Classification trees require a large dataset in order to construct a good classifier. Since the splits are done on one predictor at a time, rather than on combinations of predictors, the tree is likely to miss relationships between predictors and can be relatively expensive to grow. Pruning the data using the validation sets adds further computation time.

Logistic regression model is more useful to predict the class with binary outcomes. However, if one predictor is a linear combination of other predictors, the model will fail.

Based on performance analysis, we recommend the organization use the Classification Tree to predict the staff attrition due to its higher accuracy rate, as this method provides easily understandable classification rules, and it doesn’t require much effort. Since classification trees require a large dataset in order to construct a good classifier, the organization needs to frequently collect more information from its staff and make sure the completeness of this data.

VII. Summary

In this case study, we use k-Nearest Neighbors, Naive Bayes, CART, and Logistic Regression methods to predict staff attrition. Based on our test results, we find that predictor variables, such as Business Travel, Marital Status, and Over Time, have the largest influence on staff attrition. As expected, for employees who frequently have business travel plans, it is more likely for them to leave the company since they might want to stay at home with their family after work rather than leaving home for business. It could also be expected that single employees are more likely to leave the company and they could leave for better opportunities with less hesitation. Unsurprisingly, employees who have more overtime work would tend to leave the company because overtime work means less leisure time. Knowing the most possible reasons why employees leave the company can help the Human Resources Department take action and reduce the level of Attrition inside the organization, eventually contributing to reduce the hiring cost.

Appendix: R Code for use case study

```
library(class)
library(caret)
library(ggplot2)
library(e1071)
library(gmodels)
library(forecast)
library(adabag)
library(rpart)
library(caret)

attrition <- read.csv("EmployeeAttrition.csv")
attrition <- attrition[,c(1:3,6,7,11,12,17:19,23:25,29,33,34)]

ggplot(attrition, aes(x = factor(Attrition))) +
  geom_bar(fill = 'lightblue')

ggplot(attrition, aes(x = factor(Attrition), y = Age)) +
  geom_boxplot() +
  scale_fill_brewer(palette='Dark2') +
  theme(panel.background = element_rect(fill='lightblue', colour='lightblue'))

ggplot(attrition, aes(x = factor(Attrition), y = MonthlyIncome)) +
  geom_point()

ggplot(attrition, aes(x = Age, y = MonthlyIncome, col = factor(Attrition))) +
  geom_point()

# rewrite predictors
attrition$BusinessTravel <- ifelse(attrition$BusinessTravel == "Travel_Frequently", 1, 0)

attrition$Gender <- ifelse(attrition$Gender == "Male", 1, 0)

attrition$MaritalStatus <-ifelse(attrition$MaritalStatus == "Single", 1,
                                 ifelse(attrition$MaritalStatus == "Married", 2, 3))

attrition$Attrition <- ifelse(attrition$Attrition == "Yes", 1, 0)

attrition$OverTime <- ifelse(attrition$OverTime == "Yes", 1, 0)

nomalized <- scale(attrition[,-2])
attrition.norm <- cbind(nomalized,attrition$Attrition)
colnames(attrition.norm)[16] <- "Attrition"

set.seed(1)
```

```
train.index <- sample(c(1:dim(attrition.norm)[1]), dim(attrition.norm)[1]*0.6)
train.df <- attrition.norm[train.index, ]
train.df <- as.data.frame(train.df)
valid.df <- attrition.norm[-train.index, ]
valid.df <- as.data.frame(valid.df)

knn1 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 1, prob = FALSE)
knn2 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 2, prob = FALSE)
knn3 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 3, prob = FALSE)
knn4 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 4, prob = FALSE)
knn5 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 5, prob = FALSE)
knn6 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 6, prob = FALSE)
knn7 <- knn(train.df[,1:15], valid.df[,1:15], cl = train.df[,16] ,
        k= 7, prob = FALSE)

table(knn1, valid.df[,16])
table(knn2, valid.df[,16])
table(knn3, valid.df[,16])
table(knn4, valid.df[,16])
table(knn5, valid.df[,16])
table(knn6, valid.df[,16])
table(knn7, valid.df[,16])

# choose k = 7

attrition.knn <- knnreg(Attrition~.,train.df, k=7)
attrition.knn.pred <- predict(attrition.knn, valid.df)

attrition.lift <- data.frame(case = c(1:dim(valid.df)[1]),
                   knnreg = cumsum(valid.df$Attrition[order(attrition.knn.pred, decreasing = T)]),
                   baseline = c(1:dim(valid.df)[1])*mean(valid.df$Attrition))

ggplot(attrition.lift, aes(x = case)) +
  geom_line(aes(y = knnreg), color = "blue") +
  geom_line(aes(y=baseline), color = "red", linetype = "dashed") +
  labs(x = "# of cases", y = "Cumulative Expenses")

# naive bayes
attrition1 <- attrition
# transform to factors
```

```
attrition1$Age <- factor(round(attrition1$Age/10))
attrition1$DistanceFromHome <- factor(round(attrition1$DistanceFromHome/5))
attrition1$MonthlyIncome <- factor(round(attrition1$MonthlyIncome/3000))
attrition1$PercentSalaryHike <- factor(round(attrition1$PercentSalaryHike/10))
attrition1$TotalWorkingYears <- factor(round(attrition1$TotalWorkingYears/5))
attrition1$YearsInCurrentRole <- factor(round(attrition1$YearsInCurrentRole/3))
attrition1$YearsSinceLastPromotion <- factor(round(attrition1$YearsSinceLastPromotion/3))
attrition1$Education <- as.factor(attrition1$Education)
attrition1$EnvironmentSatisfaction <- as.factor(attrition1$EnvironmentSatisfaction)
attrition1$Attrition <- as.factor(attrition1$Attrition)
attrition1$BusinessTravel <- as.factor(attrition1$BusinessTravel)
attrition1$Gender <- as.factor(attrition1$Gender)
attrition1$MaritalStatus <- as.factor(attrition1$MaritalStatus)
attrition1$JobSatisfaction <- as.factor(attrition1$JobSatisfaction)
attrition1$OverTime <- as.factor(attrition1$OverTime)
attrition1$PerformanceRating <- as.factor(attrition1$PerformanceRating)

train.df1 <- attrition1[train.index, ]
train.df1 <- as.data.frame(train.df1)
valid.df1 <- attrition1[-train.index, ]
valid.df1 <- as.data.frame(valid.df1)

attrition.nb <- naiveBayes(Attrition ~ ., data = train.df1)
attrition.nb

# predict prob
pred.prob <- predict(attrition.nb, newdata = valid.df1, type = "raw")
# predict class in validation
pred.class.v <- predict(attrition.nb, newdata = valid.df1)
# predict class in validation
pred.class.t <- predict(attrition.nb, newdata = train.df1)

CrossTable(x = valid.df1$Attrition, y = pred.class.v)

df <- data.frame(actual = valid.df1$Attrition,
                 predicted = pred.class.v, pred.prob)

# confusion matrix
table(pred.class.t, train.df1$Attrition)
table(pred.class.v, valid.df1$Attrition)

# CART
library(rpart)
library(rpart.plot)
attrition <- read.csv("EmployeeAttrition.csv")
attrition <- attrition[,c(1:3,6,7,11,12,17:19,23:25,29,33,34)]
```

```
set.seed(1)

train.index2 <- sample(c(1:dim(attrition)[1]),
                       dim(attrition)[1]*0.5)
train.df2 <- attrition[train.index2, ]
train.df2 <- as.data.frame(train.df2)
except.train2 <- attrition[-train.index2, ]
valid.index2 <- sample(c(1:dim(except.train2)[1]),
                       dim(except.train2)[1]*0.6)
valid.df2 <- except.train2[valid.index2, ]
valid.df2 <- as.data.frame(valid.df2)
test.df2 <- except.train2[-valid.index2, ]
test.df2 <- as.data.frame(test.df2)

# default tree
attrition.ct <- rpart(Attrition ~., data = train.df2, method = "class")
prp(attrition.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10)

# full grown tree
deeper.ct <- rpart(Attrition ~., data = train.df2,
                   method = "class", cp = 0.00001, minsplit = 1)
# count number of leaves
length(deeper.ct$frame$var[deeper.ct$frame$var == "<leaf>"])
# plot tree
prp(deeper.ct, type = 1, extra = 1, under = TRUE,
    split.font = 1, varlen = -10,
    box.col=ifelse(deeper.ct$frame$var == "<leaf>", 'gray', 'white'))

# use printcp() to print the table.
printcp(deeper.ct)

deeper.pred <- predict(deeper.ct, valid.df2, type = 'class')
table(deeper.pred, valid.df2$Attrition)

# prune by lower cp
pruned.ct <- prune(deeper.ct, cp = 0.0166667)
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])
prp(pruned.ct, type = 1, extra = 1, split.font = 1, varlen = -10)
pruned.pred <- predict(pruned.ct, test.df2, type = 'class')
table(pruned.pred, test.df2$Attrition)

# randomforest
library(randomForest)
rf <- randomForest(as.factor(Attrition) ~ ., data = train.df2,
                   ntree = 500, mtry = 4,
                    nodesize = 5, importance = TRUE)

# variable importance plot
varImpPlot(rf, type = 1)

# confusion matrix
rf.pred <- predict(rf, valid.df2, type = "class")
table(rf.pred, valid.df2$Attrition)

# boosting
set.seed(1)
train.df2$Attrition <- as.factor(train.df2$Attrition)
boost <- boosting(Attrition ~ ., data = train.df2)
pred <- predict(boost, valid.df2)
table(pred$class, valid.df2$Attrition)

# logistic
# use glm() (general linear model) with family = "binomial" to fit a logistic regression
set.seed(1)
train.index3 <- sample(c(1:dim(attrition)[1]),
                       dim(attrition)[1]*0.6)
train.df3 <- attrition[train.index3, ]
train.df3 <- as.data.frame(train.df3)
valid.df3 <- attrition[-train.index3, ]
valid.df3 <- as.data.frame(valid.df3)

logit.reg <- glm(Attrition ~ ., data = train.df3, family = "binomial")
options(scipen=999)
summary(logit.reg)
# use predict() with type = "response" to compute predicted probabilities.
logit.reg.pred <- predict(logit.reg, valid.df3[,-2], type = "response")

# first 5 actual and predicted records
data.frame(actual = valid.df3$Attrition[1:5],
           predicted = logit.reg.pred[1:5])

logit.reg.pred1 <- ifelse(logit.reg.pred > 0.5, 1, 0)
table(valid.df3$Attrition, logit.reg.pred1)


library(gains)
gain <- gains(as.numeric(valid.df3$Attrition), logit.reg.pred, groups=10)

# plot lift chart
plot(c(0,gain$cume.pct.of.total*sum(as.numeric(valid.df3$Attrition)))~c(0,gain$cume.obs),
     xlab="# cases", ylab="Cumulative", main="", type="l")
```

```
lines(c(0,sum(as.numeric(valid.df3$Attrition)))~c(0, dim(valid.df3)[1]),
      lty=2)
```