Predict which employees had attrition in their company. Using KNN Classifier
Which employees have attrition for the company?

Project Objective
We have a dataset of employees from IBM and the data about them.
Objective: predict whether employees have attrition or not. Attrition in our case is question “Do employees want to quit the job or not?”. We will use KNN classifier.

Step1: Load the dataset

```
#Importing the dataset
data = read.csv('Employee-Attrition.csv')
```

Step2: Get familiar with it

```
#Getting familiar with the dataset
head(data)
str(data)
```

When I saw the head and structure of the dataset, it became clear that there is 35 columns that is too much. Because if we use 35 variables for just one prediction, there will be too much noise in our results.

```
'data.frame':   1470 obs. of  35 variables:
```

So the first thing to do is to clean the dataset.

Step3: Data Cleaning
I created new dataset that contains only necessary columns.

```
#selecting only relevant columns
data1 <- data[,c(1,2,12,17,19,23,29,31)]
head(data1)
str(data1)
```

The necessary columns are the columns that could explain the attrition level in people:

```
> str(data1)
'data.frame':   1470 obs. of  8 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 ...
 $ 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 ...
 $ 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 ...
 $ TotalWorkingYears: int  8 10 7 8 6 8 12 1 10 17 ...
 $ WorkLifeBalance  : int  1 3 3 3 3 2 2 3 3 2 ...
```

Here we see that the 8 columns can describe the variable “Attrition” with ‘Yes’ or ‘No’ levels. But the problem is that for kNN we have to use only numeric values in order to make it work as best as it can.

Step4: Converting values from factor to numeric values

```
#For kNN we need to convert all values into numeric
#converting factor values into numeric
data1$Attrition = as.integer(data1$Attrition) #No = 1, Yes = 2
data1$Gender = as.integer(data1$Gender) # Female = 1, Male = 2
data1$OverTime = as.integer(data1$OverTime) # No = 1, Yes = 2
```

As we converted, cleaned and made our dataset look nice, let’s check it.

```
> head(data1)
  Age Attrition Gender JobSatisfaction MonthlyIncome OverTime TotalWorkingYears
1  41         2      1               4          5993        2                 8
2  49         1      2               2          5130        1                10
3  37         2      2               3          2090        2                 7
4  33         1      1               3          2909        2                 8
5  27         1      2               2          3468        1                 6
6  32         1      2               4          3068        1                 8
  WorkLifeBalance
1               1
2               3
3               3
4               3
5               3
6               2
```

We see that our values are numeric and data is clean. But now we need to normalize it before applying kNN because we see that some numbers are too big and others are too small.

Step5: Scaling or Normalization

```
#Normalization for kNN
#using all columns except for target(Attrition)
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
  }
data1.n <- as.data.frame(lapply(data1[,c(1,3:8)], normalize))
```

I applied a function to normalize our data. So now our data is normalized. Now we can work with it.

```
> head(data1.n)
        Age Gender JobSatisfaction MonthlyIncome OverTime TotalWorkingYears
1 0.5476190      0       1.0000000     0.2624539        1             0.200
2 0.7380952      1       0.3333333     0.2170090        0             0.250
3 0.4523810      1       0.6666667     0.0569247        1             0.175
4 0.3571429      0       0.6666667     0.1000527        1             0.200
5 0.2142857      1       0.3333333     0.1294892        0             0.150
6 0.3333333      1       1.0000000     0.1084255        0             0.200
  WorkLifeBalance
1       0.0000000
2       0.6666667
3       0.6666667
4       0.6666667
5       0.6666667
6       0.3333333
```

Step6: Split data into training and test sets.

```
#Partitioning our dataset
set.seed(123)
tr.ind <- sample(1:nrow(data1.n),
                 size = nrow(data1.n) * 0.7,
                 replace = FALSE)

train.data1 <- data1[tr.ind,]
test.data1 <- data1[-tr.ind,]

#Creating separate dataframe for "Attrition" that is our target
train.labels <- data1[tr.ind, 2]
test.labels <- data1[-tr.ind, 2]
```

We split the data into 70% of training set and 30% of test set.

Step7: Fit KNN classifier with initial k value and make predictions
Firstly, lets randomly pick **k=5** and fit our model into the dataset. Let's see the code and confusion matrix below to see the results. **1=No, 2=Yes** for Attrition.
We see that model could correctly predict that 358 of total(people without attrition) had NO attrition and that 5 out of total(people with attrition). We can see that model could predict the test set, but we need to check the accuracy level.

```
# choosing k
#try k=5 first
knn.5 <- knn(train=train.data1,
             test=test.data1,
             cl=train.labels,
             k=5)
#confusion matrix
cm <- table(knn.5,test.labels)
cm
```

```
> cm
     test.labels
knn.5   1   2
    1 358  72
    2   6   5
```

Step8: Check the accuracy with 1st k=5
I created a function for accuracy and applied it to our cm(confusion matrix).

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(cm)
```

As a result our accuracy was **82.31293.** It is pretty good accuracy for such model. But there's definitely better **k** that will have higher accuracy. lets find it.

Step9: Finding optimal k value

```
# optimal k selection
accuracy_k <- NULL

nnum<-nrow(train.data1)/2
nnum

for(kk in c(1:nnum))
{
  set.seed(1234)
  knn_k<-knn(train=train.data1,test=test.data1,cl=train.labels,k=kk)
  accuracy_k<-c(accuracy_k,sum(knn_k==test.labels)/length(test.labels))
}

# plot for k=(1 to n/2) and accuracy
test_k<-data.frame(k=c(1:nnum), accuracy=accuracy_k[c(1:nnum)])
```

Here the code consists of several small steps that in total create test for optimal **k** value.

Now our task is to find a **k** value that has highest accuracy, but lowest possible number.

```
# the lowest k with the highest accuracy
min(test_k[test_k$accuracy %in% max(accuracy_k),"k"])
```

The result is **k=18.** The accuracy of result is **0.8321995 = 83.2%** accuracy. It is the highest accuracy. Remember **k=5** had the accuracy of **82.31293 = 83.3%.**

Step10: Make predictions and check the accuracy with 2nd optimal k.

```
#k=18 knn
knn.18 <- knn(train=train.data1,
              test=test.data1,
              cl=train.labels,
              k=18)
#confusion matrix
cm1 <- table(knn.18,test.labels)
cm1
#Checking accuracy of the test for k = 18
accuracy(cm1)
```

```
      test.labels
knn.18   1   2
     1 364  74
     2   0   3
```

The confusion matrix shows that now the test performed better than k=5 because it has higher accuracy of 83.21995. Also we see that for prediction **2=yes** that means people with attrition, 3 out of 3 predictions were predicted correctly.

Step11: Conclusion

Now we are ready to predict and apply the kNN model for our model. And we know that k=18 is the best k for our model.