데이터마이닝 Term Project

Churn Data

2015170378
정은영

목 차

고객이탈 데이터마이닝

Churn Data Mining

정은영

초 록

여러 분야에서 고객이탈은 매우 중요한 문제이다. 이탈하는 고객을 막기 위해 기업들은 마케팅을 진행하기도 한다.

이탈하는 고객을 막기 위해서는 이탈 가능성이 높은 고객을 찾아야하며, 이탈 가능성이 높은 고객을 찾기 위해 데이터를 통해 예측하는 것이 필요하다.

이 보고서를 통해 통신사의 고객이탈 정보를 가지고, 기계학습 전반적인 알고리즘을 사용하여 각 알고리즘의 특성을 알아보고, 각 특성들의 성능을 평가하여 하나의 알고리즘을 선택하는 과정과 함께 고객이탈여부를 예측하는 시간을 가질 것이다.

목적 변수가 있기 때문에 Supervised Learning 에 속하지만, 좀 더 다양한 시도를 위해 Clustering 을 통해 만약 고객이탈 정보가 없을 시 어떤 군집이 생성될지 살펴보며, 후에 비지도 학습에 초점을 맞추어 데이터 분석을 진행할 것이다.

R 을 통해 모델을 구현하며, 마지막에 기존 연구와 비교하며 고객이탈 데이터에 대한 분석을 마무리한다.

키워드

Clustering, Classification, R, Churn, Logistic Regression, K-means clustering, Hierarchical Clustering, DBSACN, Decision Tree, Random Forest, ANN, k-NN, SVM

Ⅰ 서론

고객이탈은 모든 분야에서 매우 중요한 이슈이다. 왜냐하면 고객이 이탈하게 되면 다시 얻는 것에 비용소모가 많이 되기 때문이다. CRM(고객 관계 관리, Customer relationship management)은 소비자들을 자신의 고객으로 만들고, 이를 장기간 유지하고자 하는 경영방식이며 기업과의 관계를 관리, 고객, 판매인, 협력자와 내부 정보를 분석하고 저장하는 데 사용하는 광대한 분야를 아우르는 방법이다.

고객을 유지하는 것이 신규고객을 획득하는 것보다는 효과적이라는 것은 CRM 전략의 핵심적인 내용 중 하나이다. 기존의 마케팅 방법으로 매년 신규고객을 획득해서 이탈하는 고객을 채우는 것이 매우 소모적이고 단기적인 전략으로 인식된다. 비용적인 관점에서 평가하면, 고객을 유지하는 것이 신규고객을 획득하는 것에 비해서 5 배가 효과적이다. 대부분 기업에서 매년 15~20% 고객을 잃어버리며, 유지율을 몇%만 증가해도 25~100%까지 수익증대를 가져올 수 있다는 결과와 많은 연구와 실제 사례에서 찾아볼 수 있다. 고객 유지의 중요성을 살펴보면 다음과 같다.

-> 고객 유지비용 대비 신규 고객 획득 비용이 평균적으로 10 배 높음

-> 평균적으로 고객이탈비용은 그 고객이 연간 연간 창출하는 가치의 5 배임

물론 신규고객과 새로운 시장의 창출은 기업의 가장 중요한 마케팅 활동이지만 기업의 관점에서 고객에게 다양한 제품과 서비스를 지속적으로 경험하여 초우량 고객화 하는 것이 보다 효과적이다.

고객이 이탈하는 원인은 무엇이고 어떤 고객이 이탈할까? 이탈징후는 다음과 같이 나타난다.

-> 최근 서비스 및 제품에 대한 활용 저하

-> 이사 등 신상의 변화

-> 온라인 활동 이력 저하

-> 생애 단계의 변화

-> 진학, 입사, 결혼, 출산 등

이러한 이탈 징후의 데이터를 정확히 이해하고 이탈 방지 활동으로 전환할 수 있어야 고객 이탈을 관리할 수 있을 것이다. 물론 이탈하는 고객에 대한 설문조사 등을 통해 직접적인 원인을 조사하는 것은 보다 우선 분석해야 할 내용이다.

어떤 고객이 다음달에 이탈할 것인가? 앞서 언급한 이탈 원인에 대한 분석을 통해 이탈에 영향을 주는 다양한 고객의 변화를 감지할 수 있어야하고, 질문에 답하기 위해서 이탈할 가능성이 있는 고객을 리스트로 추출할 수 있어야한다. 그래야 고객과 소통이 가능한 홈페이지, 이메일 등으로 이탈하지 못하도록 관리할 수 있다.

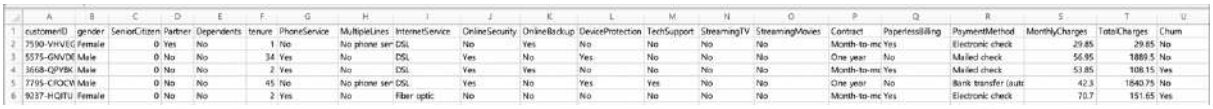
이탈할 가능성이 높은 고객을 평가해서 리스트로 추출하기 위해 고객의 다양한 정보를 통합하여 이탈예측 모형을 통해 수치화 해야하는데, 이를 위해 데이터마이닝 기법이 사용된다. 데이터마이닝 기법을 통해 IBM 의 통신사 고객이탈 데이터를 이용해 고객이탈을 예측하는 기본적 모델을 수립해보려고 한다.

II Data

IBM 사이트의 Sample Data Set 중에서
Customer Support 의 WA_Fn UseC_Telco Customer Churn.csv 데이터를 선택했다.
https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/

1. 데이터 소개

해당 데이터는 통신사 고객이탈(Customer Churn)에 대한 Target Data 와 고객 ID, 성별, 노약자정보, 결혼유무, 부양가족여부 등의 Data 를 포함하여 총 7043 개의 행과 21 개의 열로 구성된다.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
2	7590-VHVEG	Female	0	Yes	No	1	No	No phone ser	DSL	No	Yes	No	No	No	No	Month-to-mo	Yes	Electronic check	29.85	29.85	No
3	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
4	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-mo	Yes	Mailed check	53.85	108.15	Yes
5	7795-CFOCW	Male	0	No	No	45	No	No phone ser	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (autc	42.3	1840.75	No
6	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-mo	Yes	Electronic check	70.7	151.65	Yes

데이터 정보는 다음과 같이 확인해볼 수 있다.

```
## 데이터 확인
View(df) #데이터 보기
head(df)
summary(df)
dim(df) #행, 열 개수
names(df) #attribute 이름 보기
str(df) #데이터 타입 확인
```

데이터의 변수에 대한 정보를 정리해보면 다음과 같다.

통신사 고객이탈 데이터	데이터타입	WA_Fn-UseC_-Telco-Customer-Churn
CustomerID	Factor	고객아이디
Gender	Factor	성별 (male, female)
SeniorCitizen	int	고령자여부 (0,1)
Partner	Factor	결혼여부 (yes, no)
Dependents	Factor	부양가족여부(yes, no)
tenure	int	회사에 머무른 개월 수(0~72 numerical)
PhoneService	Factor	폰서비스사용여부(yes, no)
MultipleLines	Factor	여러회선사용여부(yes, no, no phone service)
InternetService	Factor	인터넷서비스제공자(DSL, Fiber optic, No)
OnlineSecurity	Factor	온라인보안여부(yes, no, no internet service)
OnlineBackup	Factor	온라인백업여부(yes, no, no internet service)
DeviceProtection	Factor	기기보호기능여부(yes, no, no internet service
TechSupport	Factor	기술지원사용여부(yes, no, no internet service)
StreamingTV	Factor	스트리밍TV보유여부(yes, no, no internet service)
StreamingMovies	Factor	영화스트리밍여부(yes, no, no internet service)
Contract	Factor	고객의 계약기간(Month-to-month, One year, Two year)
PaperlessBilling	Factor	온라인(종이없는)청구서수신여부(yes, no)
PaymentMethod	Factor	고객의 결제수단(전자(Electronic check),우편(메일)(Mailed check), 은행송금(자동)(Bank transfer(automatic), 신용카드(Credit card(automatic))
MonthlyCharges	num	매월 청구 금액(18.3~119, numerical)
TotalCharges	num	총 청구금액(18.8~8680, numerical)
Churn	Factor	target, 고객이탈여부(yes, no)

2. EDA & 전처리

다음과 같이 결측치를 확인하고 제거한다.

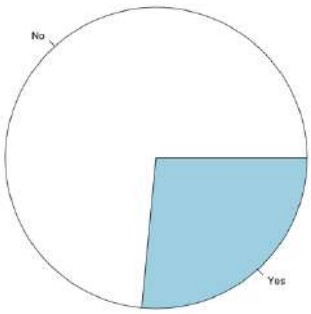
```
table(is.na(df))  # 결측치 빈도 출력, 11개의 결측치 확인
colSums(is.na(df)) # 어떤 변수에서 발생했는가 -> Total Charges
df <- na.omit(df) # 결측치 있는 행 제거
table(is.na(df)) #제거 확인
```

```
> table(is.na(df))  # 결측치 빈도 출력, 11개의 결측치 확인

 FALSE   TRUE
147892     11
> colSums(is.na(df)) # 어떤 변수에서 발생했는가 -> Total Charges
      customerID           gender    SeniorCitizen          Partner       Dependents
               0                0                0                0                0
          tenure     PhoneService    MultipleLines  InternetService   OnlineSecurity
               0                0                0                0                0
    OnlineBackup DeviceProtection      TechSupport      StreamingTV  StreamingMovies
               0                0                0                0                0
        Contract PaperlessBilling    PaymentMethod   MonthlyCharges     TotalCharges
               0                0                0                0               11
           Churn
               0
> df <- na.omit(df) # 결측치 있는 행 제거
> table(is.na(df)) #제거 확인

 FALSE
147672
```

위와 같이 Total Charges 변수에서 11 개의 결측치가 있다는 것을 알아 삭제해주었다.

```
df[1] <- NULL #고객아이디는 Churn의 변화에 영향 주지 않기 때문에 제거.
head(df)

# SeniorCitizen을 yes와 no로 바꿔줌
df <- df[complete.cases(df),]
df$SeniorCitizen <- as.factor(ifelse(df$SeniorCitizen==1, 'YES', 'NO'))
head(df)
```

다음과 같이 고객아이디는 고객이탈여부의 변화에 영향을 주지 않기 때문에 제거하고, 고령자 여부가 1,0 인 정수로 되어있기 때문에 Yes, No 인 범주형으로 변환해준다.

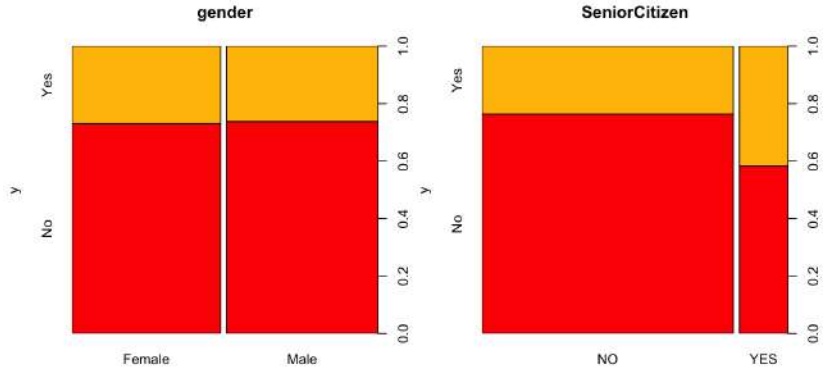


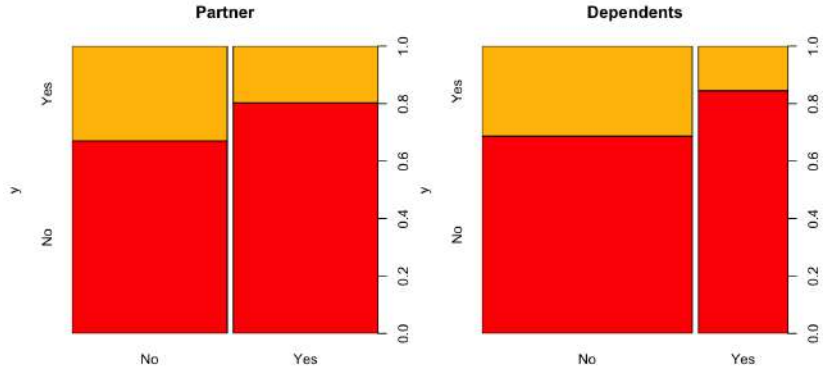
목적변수인 Churn 에 대해 pie chart 로 확인해보면 다음과 같다.

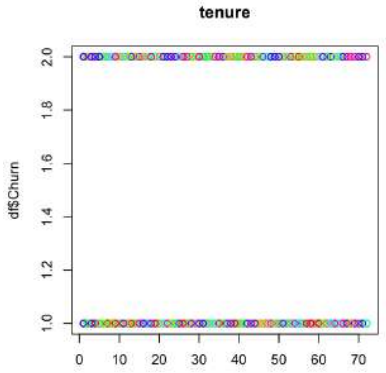
하늘색 부분이 Yes 로 이탈한 사람이 이탈하지 않은 사람 수보다 적음을 알 수 있다.

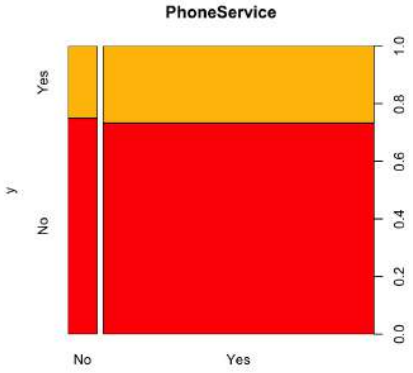
```
#그려보기
install.packages("wesanderson")
library(wesanderson)
par(mfrow = c(2,2))
for(i in 1:(length(colnames(df))-1)){
  plot(df[,i], df$Churn, main = names(df[i]), ylab = names(df$Churn), xlab =
"", col = rainbow(8))
}
dev.off()
```

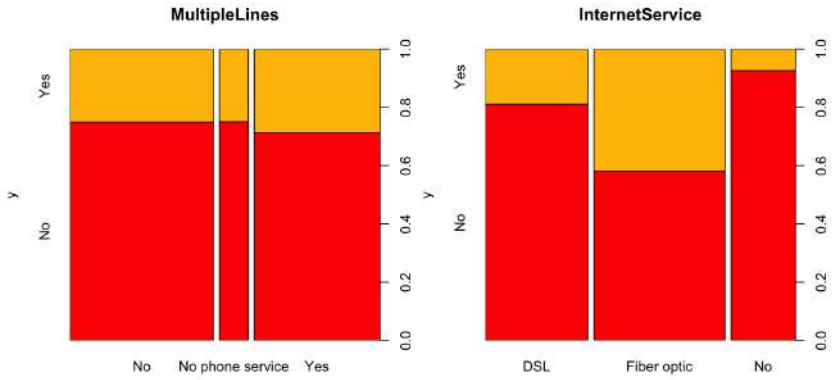
다음과 같이 x 축을 Churn 을 제외한 인풋변수로, y 축을 고객이탈여부 변수로 그래프를 그려본다.



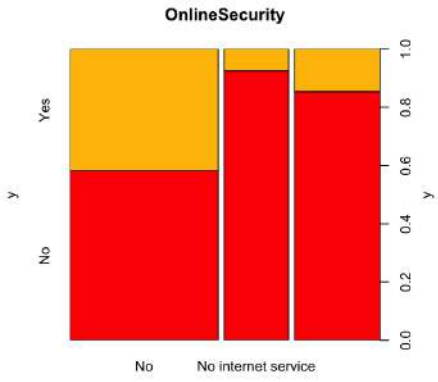


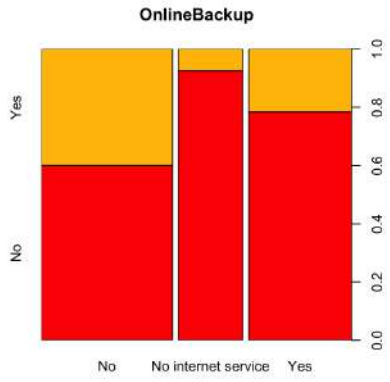


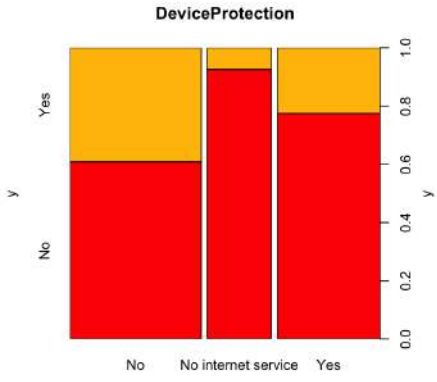


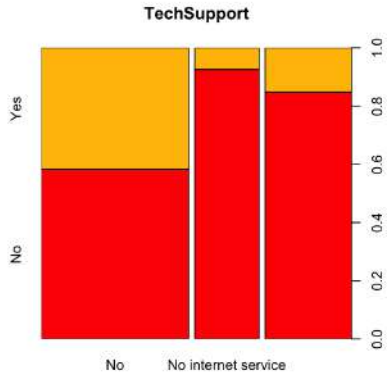

MultipleLines
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No phone service
Yes

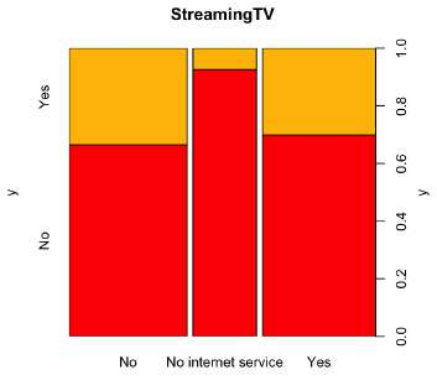
InternetService
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
DSL
Fiber optic
No

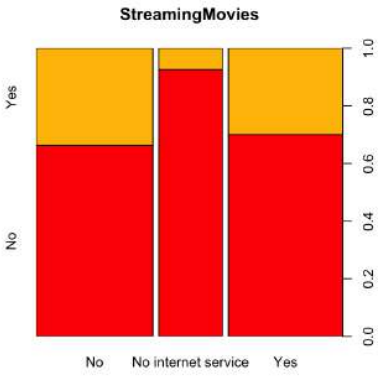

OnlineSecurity
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service

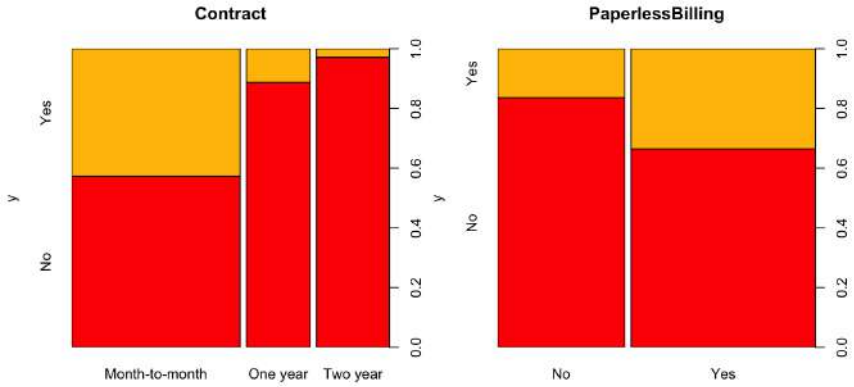

OnlineBackup
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service
Yes


DeviceProtection
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service
Yes


TechSupport
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service


StreamingTV
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service
Yes


StreamingMovies
y
Yes
No
0.0
0.2
0.4
0.6
0.8
1.0
No
No internet service
Yes

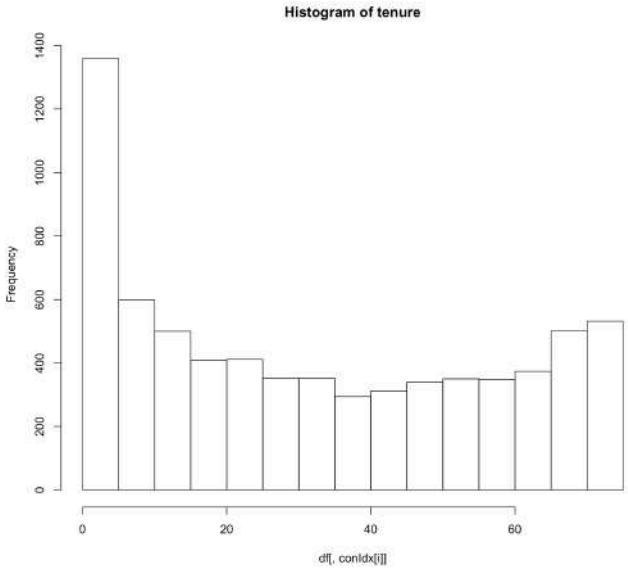


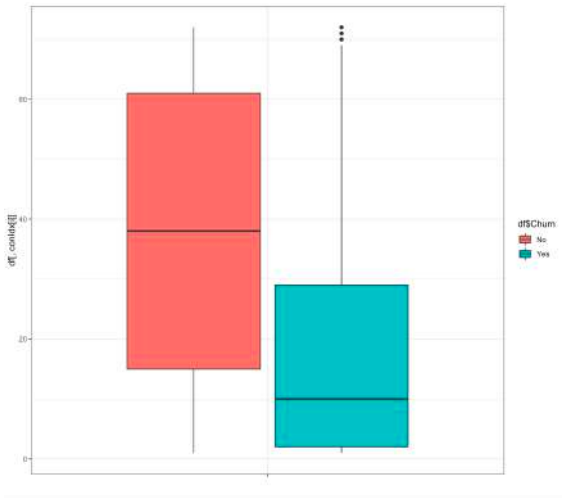
그래프를 그려보았을 때 성별, 폰서비스, 다중회선은 고객이탈여부에 영향을 주지 않는 것으로 보이며, 나머지 변수들은 차이가 있었다.

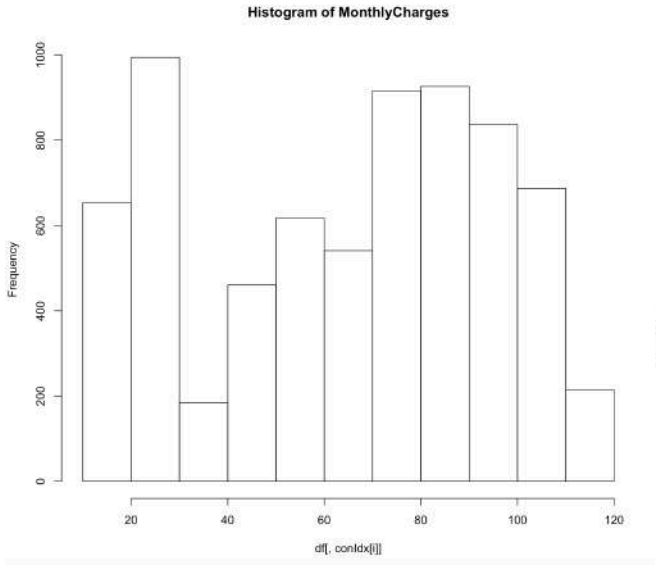
```
# 데이터프레임을 참고하고 그래프를 그려본 결과 tenure, Monthly Charges,
TotalCharges 를 제외한 모든 변수들이 categorical 이라고 생각됨. (5,18,19)
conIdx <- c(5,18,19)
cateIdx = c(1:(length(colnames(df))-1))[!(c(1:(length(colnames(df))-1)) %in%
conIdx)] #categorical index

# mean, standard deviation, skewness, kurtosis 포함 matrix 정의
nMat <- matrix(c(1:length(conIdx)*4),nrow=length(conIdx),ncol=4)
colnames(nMat)<- c("mean", "std","skewness","kurtosis")
rownames(nMat)<- colnames(df[conIdx])

#Continuous value 에 대해 histogram그리고, matrix 채우기.
for(i in 1:length(conIdx)){
  hist(df[,conIdx[i]], main = paste("Histogram of" , colnames(df[conIdx[i]]
)))
  box <- ggplot(df, aes(y= df[,conIdx[i]], x = "", fill = df$Churn)) +
geom_boxplot()+ theme_bw()+ xlab(" ")
  print(box)
  nMat[i,1] <- mean(unlist(df[,conIdx[i]]))
  nMat[i,2] <- sqrt(var(df[,conIdx[i]]))
  nMat[i,3] <- skewness(df[,conIdx[i]])
  nMat[i,4] <- kurtosis(df[,conIdx[i]])
}
dev.off()
View(nMat)
```

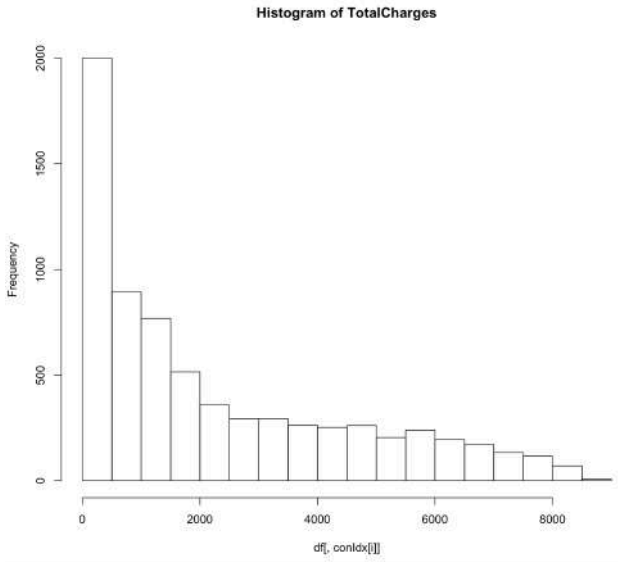
데이터를 보고 그림을 그려본 결과, tenure, Monthly Charges, Total Charges 가 연속형 변수고 나머지는 범주형 변수라고 생각되었다.

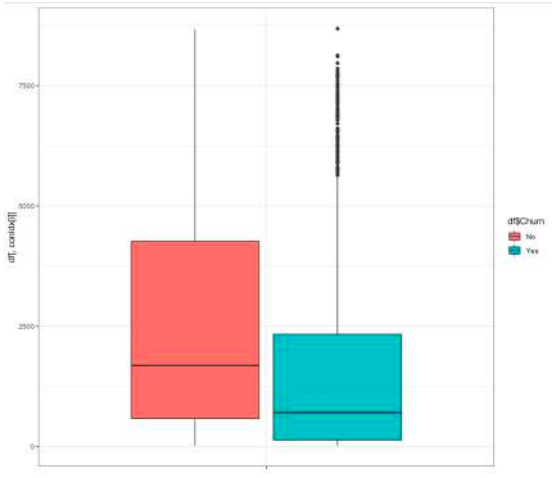
각각 변수에 대해 mean, SD, skewness, kurtosis 를 포함하는 행렬을 작성하고, 각각에 대해 histogram 과 box-plot 을 그려보았다.











각각의 히스토그램을 봤을 때 정규분포의 형태를 보이지 않았다.

각 변수가 고객이탈여부에 영향을 줄 것이라는 것을 boxplot 을 통해 확인할 수 있다.

	mean	std	skewness	kurtosis
tenure	32.42179	24.54526	0.2376801	1.612311
MonthlyCharges	64.79821	30.08597	-0.2220555	1.743883
TotalCharges	2283.30044	2266.77136	0.9614374	2.767513

또한 mean, standard deviation, skewness, kurtosis 를 확인해보았을 때, skewness 가 0 과 가깝지 않고, kurtosis 도 3 이랑 멀기 때문에 정규분포가 아니라는 것을 확인할 수 있다.

다음은 연속형 변수에 대해 outlier 를 제거하는 과정이다.

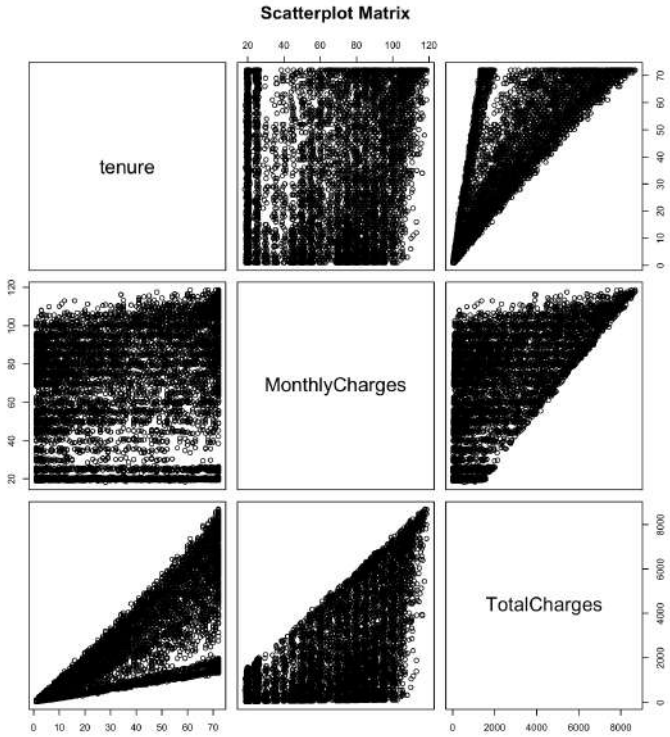
```
#outlier 제거
for(i in conIdx){
  print(colnames(df[i]))
  summary(df[,i])
  q1 <- quantile(df[,i], c(0.25))
  q3 <- quantile(df[,i], c(0.75))
  IQR <- q3 - q1
  upper_range <- q3 + 1.5*IQR
  lower_range <- q1 - 1.5*IQR
  print(nrow(df[df[,i] > upper_range,])+nrow(df[df[,i] < lower_range,]))
#outliers 개수
  df[,i]<- ifelse(df[,i] > upper_range, NA, df[,i]) #outlier 제거(윗부분)
  df[,i]<- ifelse(df[,i] > upper_range, NA, df[,i]) #outlier 제거(윗부분)
  df<- na.omit(df) #outlier 해당 데이터 데이터셋에서 제거
} #outlier 없음.
```

```
"tenure"
0
"MonthlyCharges"
0
"TotalCharges"
0
```

왼쪽과 같은 결과가 나온 것으로 보아 outlier 가 없음을 알 수 있다.

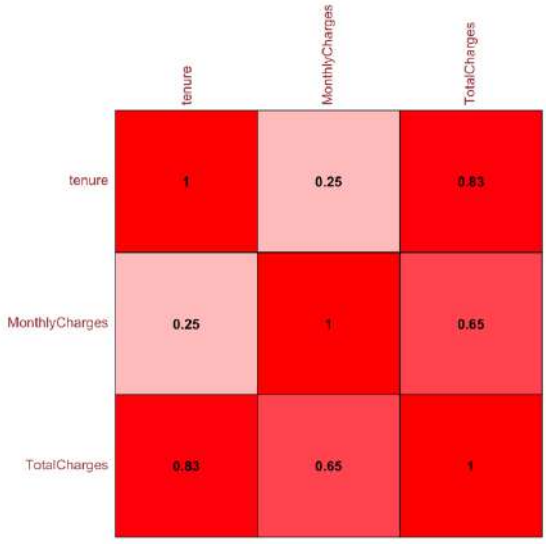
연속형 변수에 대해 Scatter Plot 도 그려본다.

```
#Scatter Plot
pairs(~.,data=df[,conIdx],
      main="Scatterplot Matrix")
dev.off()
```



Monthly Charges 랑 Total Charges 랑 상관관계가 있을 것이라는 것을 예상할 수 있으며, Tenure 즉 고객이 회사에 머무른 개월 수와 총 비용 역시 양의 상관관계를 가질 것을 예상할 수 있다. 그것을 확인하기 위해 Correlation Plot 도 그려본다

```
# correlation plot
corr <- cor(df[,conIdx])
corrplot(corr, method = "color", outline = T, cl.pos = 'n', rect.col = "black",
tl.col = "indianred4", addCoef.col = "black", number.digits = 2, number.cex = 1,
tl.cex = 1, cl.cex = 1, col = colorRampPalette(c("green4","white","red"))(100))
```



다음과 같이 예상한 결과가 나왔다.

Tenure 와 Total Charges 는 매우 높은 양의 상관관계(0.83)를 가지며, 총비용과 월비용도 꽤 높은 상관관계(0.65)를 가짐을 확인할 수 있다.

다음은 범주형 변수에 대한 기본 정보이다

```
#Categorical Value에 대해 Summary 확인
for(i in 1:length(cateIdx)){
  print(colnames(df[cateIdx[i]]))
  print(summary(factor(df[,cateIdx[i]])))
}
```

```
[1] "gender"
Female   Male
  3483   3549
[1] "SeniorCitizen"
  NO  YES
5890 1142
[1] "Partner"
  No  Yes
3639 3393
[1] "Dependents"
  No  Yes
4933 2099
[1] "PhoneService"
  No  Yes
 680 6352
[1] "MultipleLines"
             No No phone service             Yes
           3385             680            2967
[1] "InternetService"
        DSL Fiber optic          No
       2416        3096        1520
[1] "OnlineSecurity"
                No No internet service                Yes
              3497                1520               2015
[1] "OnlineBackup"
                No No internet service                Yes
              3087                1520               2425
[1] "DeviceProtection"
                No No internet service                Yes
              3094                1520               2418
[1] "TechSupport"
                No No internet service                Yes
              3472                1520               2040
[1] "StreamingTV"
                No No internet service                Yes
              2809                1520               2703
[1] "StreamingMovies"
                No No internet service                Yes
              2781                1520               2731
[1] "Contract"
Month-to-month       One year       Two year
          3875           1472           1685
[1] "PaperlessBilling"
  No  Yes
2864 4168
[1] "PaymentMethod"
Bank transfer (automatic)   Credit card (automatic)         Electronic check             Mailed check
                     1542                      1521                     2365                     1604
```

summary 를 확인한 결과, no internet service, no phone service 는 No 를 뜻하는 것을 알았기 때문에 no 로 바꿔 준다.

```
# Cleaning Categorical Feature
# no internet service, no phone service 가 no라는 것을 알기 때문에 바꿔준다.
df <- data.frame(lapply(df, function(x) {
  gsub("No internet service", "No", x)}))
df <- data.frame(lapply(df, function(x) {
  gsub("No phone service", "No", x)}))
```

다음과 같이 EDA & 전처리를 마쳤다. 알고리즘에 따른 전처리는 따로 이루어진다.

III Clustering

비지도 학습은 반응변수 Y 가 없으며 예측에는 관심이 없다. 대신 변수들의 측정에 대해 흥미로운 것들을 발견하고자 하는 것이 목적이다. 예를 들어 온라인 쇼핑 사이트는 유사한 브라우징 및 구매이력을 가진 구매자들의 그룹들을 구변하고, 각 그룹내에서 구매자들이 특별히 관심을 가지는 항목들도 식별하고자 할 수 있다.

이 데이터의 경우 반응변수 Y 가 있으나 반응변수가 없다고 가정하여 진행하도록 하겠다.

클러스터링은 각 그룹내의 관측치들이 상당히 유사하도록, 다른 그룹의 관측치들과는 다르게 나누고자 하는 방법이다. 구체적으로 하려면 두개 이상의 관측치들이 유사하다 또는 다르다는 것을 정의해야 하며 보통 데이터에 대한 지식을 기반으로 결정되는 도메인 특정사항이다.

1. K-means Clustering

관측치들을 미리 지정된 클러스터들로 나누려고 하는 방법이다. K 개의 겹치지 않는 클러스터로 분할하는 단순한 방법이다. Global optimal 이 아닌 Local optimal 을 찾기 때문에 초기의 클러스터 할당에 따라 다르다. 이러한 이유로 초기의 랜덤할당을 다르게 하여 알고리즘을 여러번 실행하는 것이 중요하다.

```
# 데이터 전처리
cChurn <- df[,21] #고객 이탈여부 열
cX <- df[,c(6,19,20)] #clustering X=cX, 연속형 변수만 뽑음.
head(cX)
cX_scaled <- as.matrix(scale(cX, center = TRUE, scale = TRUE)) #각 변수들의 range가 모두
다르므로 군집화에 영향을 줄 수 있어 평균 0, 표준편차 1로 scaling
```

각 변수들의 Range 가 다르기 때문에 연속형 변수에 대해 Scaling 을 한다. 군집화에 영향을 줄 수 있기 때문이다.
또한 K-means 클러스터링의 경우 군집 개수를 정하고 시작해야하므로 최적 군집 수를 찾아본다.

```
#최적 군집 수 찾기
cX_clValid <- clValid(cX_scaled, 2:6, clMethods="kmeans",
                      validation=c("internal","stability"))
summary(cX_clValid) #Silhouette 을 통해 cluster 2개로 설정하겠다.
```

```
Clustering Methods:
 kmeans

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                            2        3        4        5        6

kmeans APN             0.1666   0.2202   0.2515   0.2944   0.3837
       AD              1.6307   1.3577   1.2107   1.0842   1.0633
       ADM             0.4912   0.5227   0.5982   0.5426   0.6609
       FOM             0.7810   0.6996   0.6453   0.6043   0.5827
       Connectivity   83.4127 185.9373 202.0218 276.2083 262.8619
       Dunn            0.0034   0.0043   0.0060   0.0035   0.0055
       Silhouette      0.4796   0.4516   0.4724   0.4440   0.4390

Optimal Scores:

             Score   Method Clusters
APN           0.1666 kmeans 2
AD            1.0633 kmeans 6
ADM           0.4912 kmeans 2
FOM           0.5827 kmeans 6
Connectivity 83.4127 kmeans 2
Dunn          0.0060 kmeans 4
Silhouette    0.4796 kmeans 2
```

Silhouette 에 대해 설명하자면, S=1-a/b 로 a 는 한점에서 그 점이 속한 클러스터안의 점들의 거리평균이며, b 는 한점에서 다른 cluster 의 점들과의 거리 평균으로 1 에 가까울수록 의미있다. 즉 클수록 의미있다.

이 경우 2 부터 6 까지 클러스터 수를 늘려가면서 구해본 결과, 클러스터 개수가 2 일 때 Silhoutte 이 0.4796 으로 가장 크기 때문에 최적 개수로 설정되었다. 다른 지표들에 대한 개수도 확인할 수 있다.

또한 K-means 클러스터링의 경우 Global optimal 이 아닌 Local optimal 을 찾기 때문에 초기의 클러스터 할당에 따라 다르다. 이러한 이유로 초기의 랜덤할당을 다르게 하여 알고리즘을 여러번 실행하는 것이 중요하기 때문에 다음과 같이 함수를 설정하여 10 번 반복해준다.

```r
#n은 반복수, k는 군집수로 하는 kmeans 정보 출력 함수 작성
Kmeans_Cluster_Info <- function(k,n){
  for(i in 1:n){
    cX_kmc <- kmeans(cX_scaled,k)
    print(paste("n=",i))
    print(cX_kmc$centers)
    print(cX_kmc$size)
  }
}
# Perform K-Means Clustering with the best K determined by Silhouette
Kmeans_Cluster_Info(2,10) #Silhouette
Kmeans_Cluster_Info(4,10) #Dunn
```

n= 1			n= 6		
tenure	MonthlyChar	TotalCharges	tenure	MonthlyChar	TotalCharges
-0.5029294	-0.4188174	-0.624161	-0.5029294	-0.4188174	-0.624161
0.9962651	0.8296455	1.236416	0.9962651	0.8296455	1.236416
4673	2359		4673	2359	
n= 2			n= 7		
tenure	MonthlyChar	TotalCharges	tenure	MonthlyChar	TotalCharges
0.9962651	0.8296455	1.236416	0.9962651	0.8296455	1.236416
-0.5029294	-0.4188174	-0.624161	-0.5029294	-0.4188174	-0.624161
2359	4673		2359	4673	
n= 3			n= 8		
tenure	MonthlyChar	TotalCharges	tenure	MonthlyChar	TotalCharges
-0.5029294	-0.4188174	-0.624161	-0.5029294	-0.4188174	-0.624161
0.9962651	0.8296455	1.236416	0.9962651	0.8296455	1.236416
4673	2359		4673	2359	
n= 4			n= 9		
tenure	MonthlyChar	TotalCharges	tenure	MonthlyChar	TotalCharges
-0.5029294	-0.4188174	-0.624161	0.9962651	0.8296455	1.236416
0.9962651	0.8296455	1.236416	-0.5029294	-0.4188174	-0.624161
4673	2359		2359	4673	
n= 5			n= 10		
tenure	MonthlyChar	TotalCharges	tenure	MonthlyChar	TotalCharges
0.9962651	0.8296455	1.236416	0.9962651	0.8296455	1.236416
-0.5029294	-0.4188174	-0.624161	-0.5029294	-0.4188174	-0.624161
2359	4673		2359	4673	

다음은 엑셀로 결과를 정리하였으며 클러스터의 순서만 달랐지 모두 같은 클러스터가 생성되었음을 볼 수 있다.

Silhouette 으로 결정된 2 개로 결정하고 결정된 클러스터가 혹시 고객이탈여부를 뜻하는 군집이 될지 확인해보는 과정이다. 이는 목적변수가 있기 때문에 가능한 부분인데 대부분의 경우 이렇지 않다.

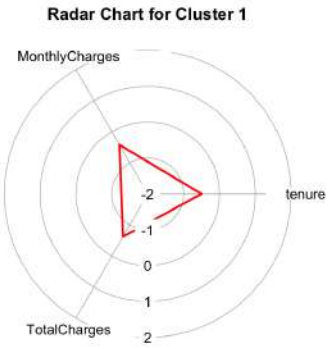
```
cX_kmc <- kmeans(cX_scaled,2) #Silhouette으로 설정.

# Compare the cluster info. and class labels
real_churn <- cChurn #고객이탈여부를 실제 결과라 하자, 원래는 clustering에 target 없음.
kmc_cluster <- cX_kmc$cluster
table(real_churn,kmc_cluster) #안 맞음..
```

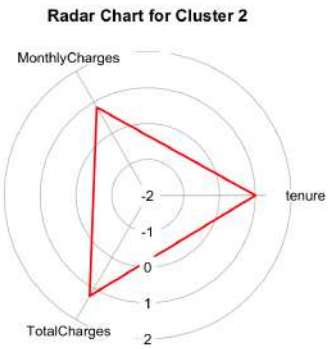
```
          kmc_cluster
real_churn    1    2
       No  3202 1961
       Yes 1471  398
```

다음과 같이 Real 과 클러스터의 결과가 대각선으로 모여지는 것이 아닌 한쪽으로 쏠리기 때문에 이 클러스터는 고객이탈여부를 뜻한다고 볼 수 없다. 또한 범주형 변수를 제외한 연속형 변수로만 시행하였기 때문에 그럴 수도 있다. 따라서 이 집단의 특성을 알기 위해서는 전문적인 지식이 필요하다.

다음은 두개의 클러스터에 대해 비교하는 과정이다.
Rader Chart 로 대략적인 추이를 보며, t-test 함수를 만들어 수행하도록 한다.

```r
# Compare each cluster for KMC
cluster_kmc <- data.frame(cX_scaled, clusterID = as.factor(cX_kmc$cluster))
kmc_summary <- data.frame()

for (i in 1:(ncol(cluster_kmc)-1)){
  kmc_summary = rbind(kmc_summary,
                      tapply(cluster_kmc[,i], cluster_kmc$clusterID, mean))
}

colnames(kmc_summary) <- paste("cluster", c(1:2))
rownames(kmc_summary) <- colnames(cX)

# Rader Chart 그리기
par(mfrow = c(2,1))
for (i in 1:2){
  plot_title <- paste("Radar Chart for Cluster", i, sep=" ")
  radial.plot(kmc_summary[,i], labels = rownames(kmc_summary),
              radial.lim=c(-2,2), rp.type = "p", main = plot_title,
              line.col = "red", lwd = 2.5, show.grid.labels=1)
}
dev.off()
```





다음과 같이 Rader Chart 가 그려지며, 각 변수에 대해 큰 값을 가지면 클러스터 2 를, 아니면 1 을 뜻한다. 그림으로 봤을 때 차이가 있어 보인다.

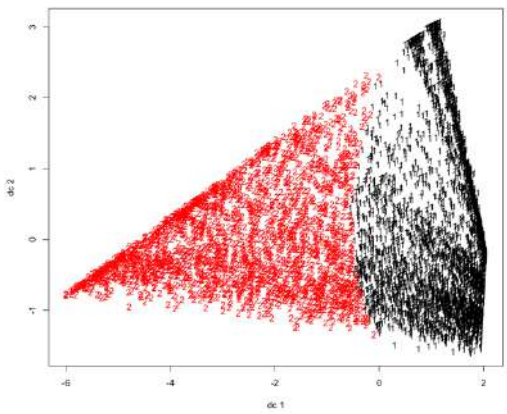
다음은 클러스터 2 개에 대해 차이가 진짜 유의미한지 t-test 를 진행한 과정이다.

```r
# Cluster 2개에 대해 t-test 함수 만들어 수행.
Cluster_tTest <- function(m,n1,n2){

  if(m=="kmc"){
    cluster1 <- cX[cX_kmc$cluster == n1,]
    cluster2 <- cX[cX_kmc$cluster == n2,]
    t_result <- data.frame()
  }else if(m=="hc"){
    cluster1 <- cX[cluster_hc$cluster == n1,]
    cluster2 <- cX[cluster_hc$cluster == n2,]
    t_result <- data.frame()
  }

  for (i in 1:ncol(cX)) {

    t_result[i,1] <- t.test(cluster1[,i], cluster2[,i],
                            alternative = "two.sided")$p.value

    t_result[i,2] <- t.test(cluster1[,i], cluster2[,i],
                            alternative = "greater")$p.value

    t_result[i,3] <- t.test(cluster1[,i], cluster2[,i],
                            alternative = "less")$p.value
  }
  colnames(t_result) <- c("two.sided","greater","less")
  rownames(t_result) <- colnames(cX)
  print(t_result)
}

# Cluster_tTest(method,ClusterIndex(i),ClusterIndex(j))
# Cluster 1 vs. Cluster 2
Cluster_tTest("kmc",1,2)
```

```
                two.sided greater less
tenure                  0       1    0
MonthlyCharges          0       1    0
TotalCharges            0       1    0
```

two.sided 는 두 C(클러스터)가 같은가에 대한, 두번째 열은 C1 이 C2 보다 큰가에 대한, 세번째는 작은가에 대한 가설 검정이다. 각각의 변수에 대해 다음과 같은 결과가 나왔다.

p-value 가 0.05 보다 작을 때 귀무가설을 기각하기 때문에 첫번째 열을 봤을 때 0 에 가까워 C1 과 C2 가 같지 않다는 것을 의미하며, 두번째 열, 세번째 열을 보면 C1 이 C2 보다 각 변수에서 작은 값을 가진다는 것을 의미한다.



왼쪽은 fpc 라이브러리의 plotcluster 함수를 사용하여 그린 결과로, 검은색은 C1, 빨간색은 C2 를 뜻한다.

2. Hierarchical Clustering

계층적 군집화는 K 값의 지정이 필요하지 않는 기법이다. K-means 클러스터링에 비해 또 다른 장점은 관측지들을 덴드로그램이라고 하는 트리기반으로 표현한다는 것이다.

상향식(botton-up) 또는 (agglomerative) 클러스터링은 가장 보편적인 유형의 계층적 클러스터링으로 잎에서 시작하여 줄기까지 클러스터를 결합하여 만들어진다.

처음에는 K-means 와 마찬가지로 최적 개수를 찾아본다.

```
# 최적 개수 찾기
cX_clValid <- clValid(cX_scaled, 2:6, clMethods="hierarchical",
                      validation=c("internal","stability"))
summary(cX_clValid)
```

```
Clustering Methods:
 hierarchical

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2        3        4        5        6

hierarchical APN            0.1302   0.2701   0.3379   0.2628   0.2852
             AD             1.6307   1.4770   1.3576   1.1845   1.1081
             ADM            0.4193   0.6118   0.7263   0.6819   0.6535
             FOM            0.7207   0.7178   0.6707   0.6410   0.6141
             Connectivity  53.6417  71.7183 120.2115 134.8163 153.7563
             Dunn           0.0102   0.0100   0.0118   0.0132   0.0156
             Silhouette     0.4635   0.4154   0.3501   0.3979   0.3950

Optimal Scores:

             Score   Method       Clusters
APN           0.1302 hierarchical 2
AD            1.1081 hierarchical 6
ADM           0.4193 hierarchical 2
FOM           0.6141 hierarchical 6
Connectivity 53.6417 hierarchical 2
Dunn          0.0156 hierarchical 6
Silhouette    0.4635 hierarchical 2
```

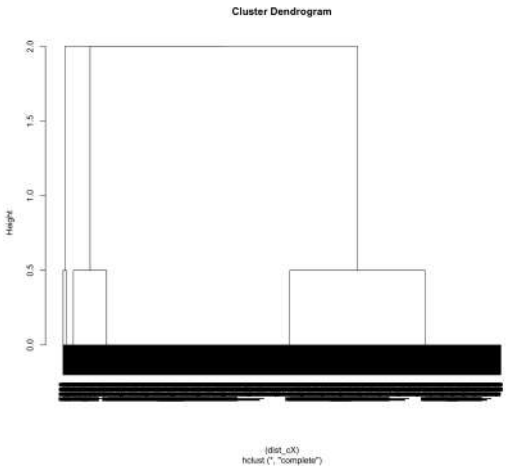
다음과 같이 Silhouette 기준으로 2 개가 나왔다.

융합되는 과정(bottom-up)에서 융합이 발생될 때의 기준은 가까움(proximity)인데 가까움의 기준은 여러가지가 있다. 그 중 수업시간에 배운 Complete, average, single, centroid, ward.D 방법을 통해 수행해보았다.

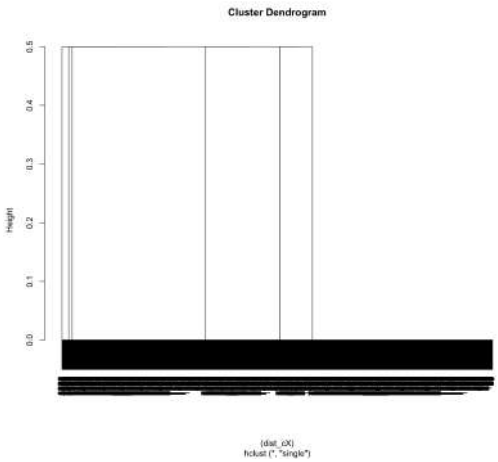
Single, 단일연결은 최소 클러스터간 비유사성으로 클러스터 A 내의 관측치들과 클러스터 B 내의 관측치들 사이의 모든 쌍별 비유사성을 계산하여 비유사성이 가장 작은 것을 기록한다. 이는 관측치들이 한번에 하나씩 융합되는 확장된 trailing 클러스터들을 초래할 수 있다.

Complete 연결은 최대 클러스터간 비유사성, Average 는 평균 클러스터간 비유사성, Centroid 는 무게중심간의 비유사성이다.

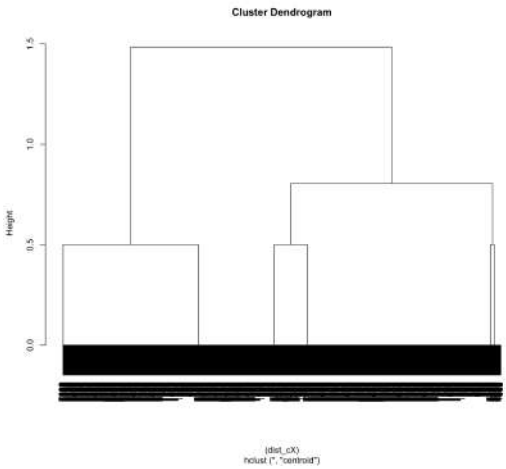
```
## Perform hierarchical clustering
hcMethod <- c("complete","average","single","mcquitty","median","centroid"
,"ward.D","ward.D2")
for (item in hcMethod){
  hr <- hclust((dist_cX), method = item, members=NULL)
  plot(hr)
}
dev.off()
```



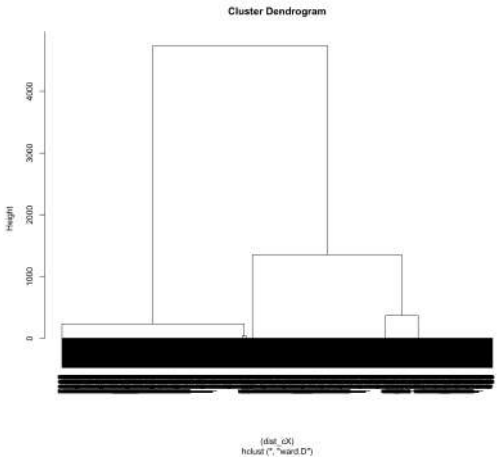
<Complete>



<Single>



<Complete>

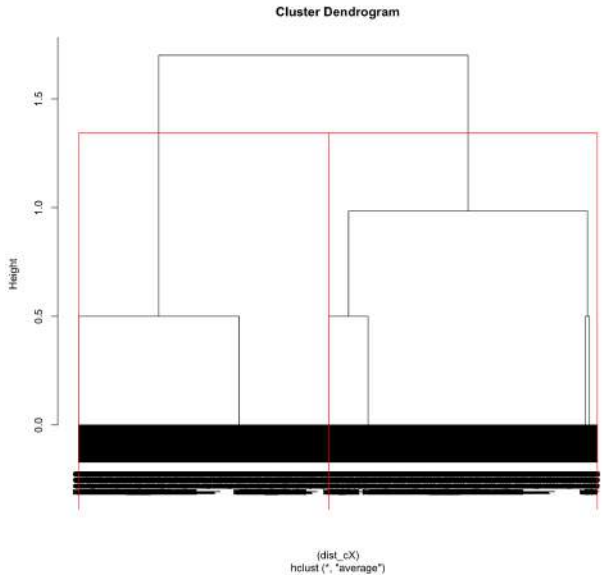


<Ward.D>

Complete 와 Ward.D 가 아주 약간 유사한 모양을 띠며 다른 것들은 달랐다.

Average 방법으로 설정해 2 개로 나눈 그림을 살펴본다.

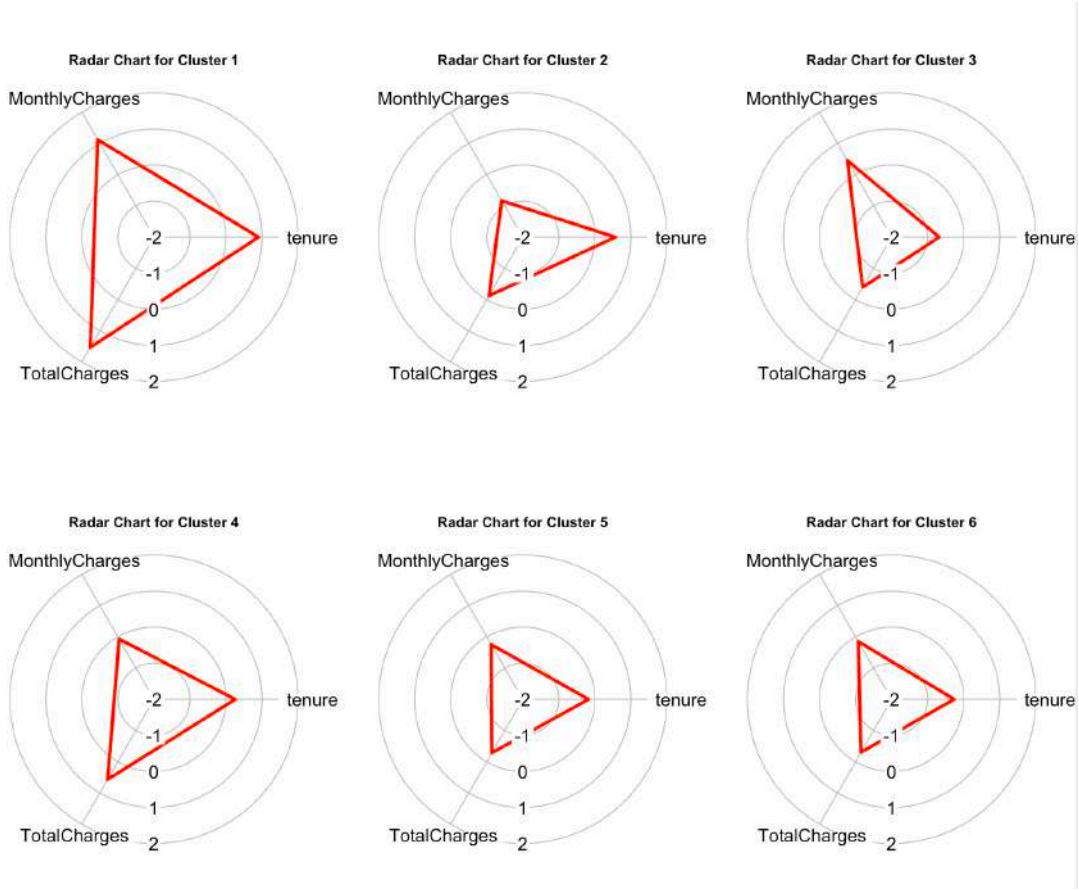
```r
# Find the clusters
hr <- hclust((dist_cX), method = "average", members=NULL)
mycl <- cutree(hr, k=2)
mycl

plot(hr)
rect.hclust(hr, k=2, border="red")
dev.off()
```



계층적 군집화의 장점은 다음과 같이 원하는 클러스터 개수를 설정하여 자를 수 있다는 것이다.
Dunn 으로 결정했을 때 6 개의 클러스터의 차이를 보도록 하겠다.

```r
# Dunn으로 6개로 결정했을 때
hr <- hclust((dist_cX), method = "average", members=NULL)
mycl <- cutree(hr, k=6)
mycl
# Compare each cluster for HC
cluster_hc <- data.frame(cX_scaled, clusterID = as.factor(mycl))
hc_summary <- data.frame()

for (i in 1:(ncol(cluster_hc)-1)){
  hc_summary = rbind(hc_summary,
                     tapply(cluster_hc[,i], cluster_hc$clusterID, mean))
}

colnames(hc_summary) <- paste("cluster", c(1:6))
rownames(hc_summary) <- colnames(cX)
hc_summary

# Radar chart
par(mfrow = c(2,3))
for (i in 1:10){
  plot_title <- paste("Radar Chart for Cluster", i, sep=" ")
  radial.plot(hc_summary[,i], labels = rownames(hc_summary),
              radial.lim=c(-2,2), rp.type = "p", main = plot_title,
              line.col = "red", lwd = 3, show.grid.labels=1)
} #1과 나머지로 분류됨(2-5까지 비슷)
dev.off()
```

Dunn index 기준으로 6 개의 클러스터로 설정하고 rader Chart 를 그려본다.



다음과 같이 그려지며, C1 을 제외한 나머지는 나름 비슷한 형태를 보였다. 즉 2 개의 클러스터로 구성해도 될 것 같다.
가장 비슷한 집단과 가장 다른 집단에 대해 t-test 를 진행하였다.

```
# Compare the cluster
# Hc_tTest(method,ClusterIndex(i),ClusterIndex(j))
# 차이가 가장 비슷한 집단 Cluster 5  vs. Cluster 6
Cluster_tTest("hc", 5, 6)

# 차이가 가장 극명한 집단 Cluster 1 vs. Cluster 5
Cluster_tTest("hc", 1, 5)
```

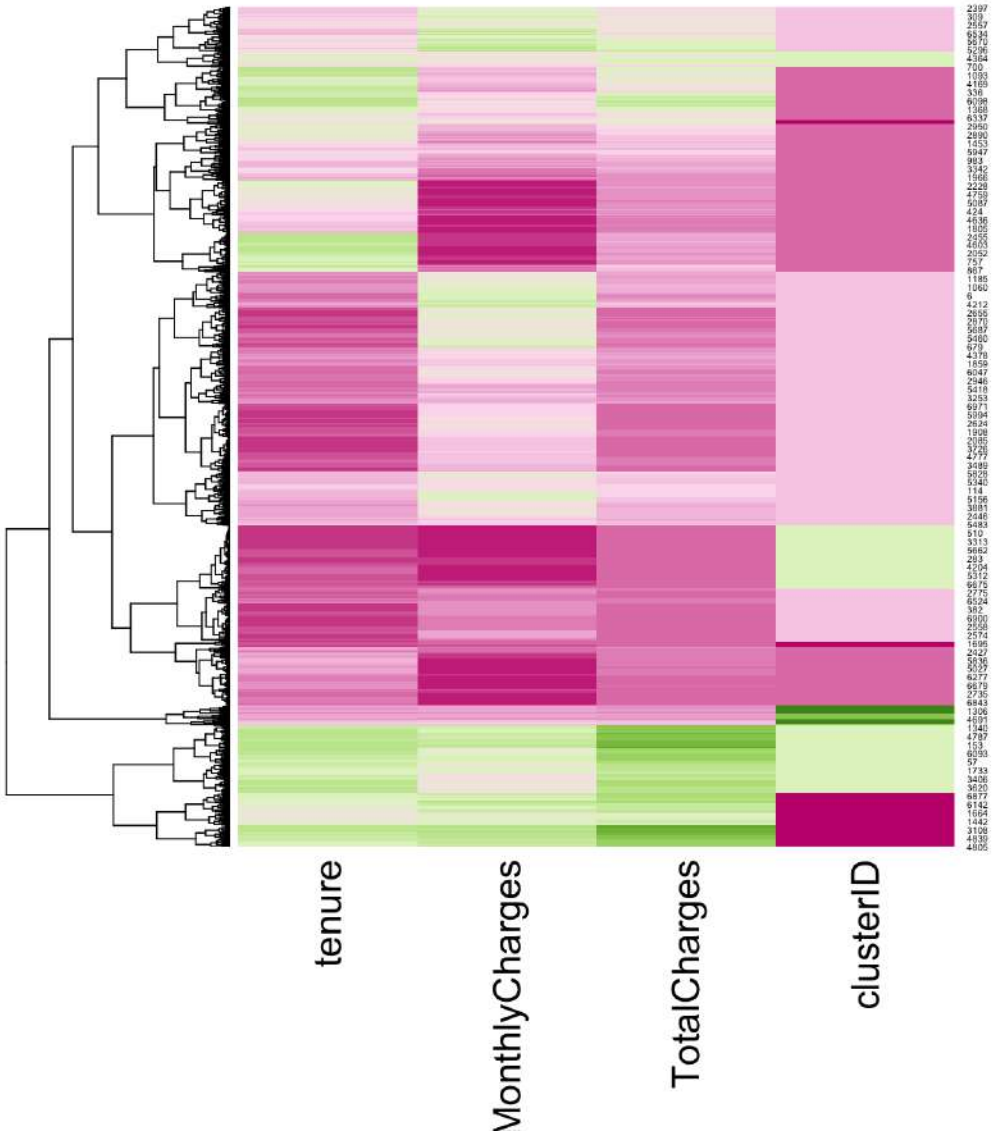
```
> # 차이가 가장 비슷한 집단 Cluster 5  vs. Cluster 6
> Cluster_tTest("hc", 5, 6)
                two.sided    greater       less
tenure         0.02019636 0.01009818 0.989901820
MonthlyCharges 0.01243286 0.99378357 0.006216432
TotalCharges   0.66418155 0.33209077 0.667909227
> # 차이가 가장 극명한 집단 Cluster 1 vs. Cluster 5
> Cluster_tTest("hc", 1, 5)
                   two.sided       greater less
tenure          1.834595e-65  9.172974e-66    1
MonthlyCharges  9.774557e-87  4.887279e-87    1
TotalCharges   5.362444e-120 2.681222e-120    1
```

C5, C6 을 t-test 했을 시 tenure(회사에 머무른 개월 수)와 월 비용은 두 집단을 비교하는데 유의미한 차이를 보이지만 총 비용은 그렇지 못하다.

또한 1,5 를 비교했을 때 모든 변수가 두 집단을 비교하는데 유의미하다고 볼 수 있다.

계층적 군집화에 대한 히트맵을 그려보았다.

```
#heatmap
library(RColorBrewer)
coul = colorRampPalette(brewer.pal(8, "PiYG"))(25)
heatmap(data.matrix(cluster_hc),Colv=NA, col=coul, scale="column",
margin=c(5,10))
```



히트맵을 그려본 결과 뚜렷하게 분리되지는 않음을 알 수 있다.

3. DBSCAN

밀도 기반 클러스터링은 noise 에 robust 한 기법으로, core, border, noise point 로 결정된다. 대표적인 2 개의 parameter 가 필요한데, Eps, MinPts 이다. 즉 일정 거리에 들어오는 점의 수를 만족시키면 core 포인트, 일정 거리에 들어오지만, 점의 수를 만족시키지 못하면 border, 그 이외는 Noise point 로 결정된다.

따라서 이 parameter 의 차이에 의해 군집수가 결정된다. 따라서 여러가지 경우를 확인해보았다.

```
# DBSCAN & Visualization
nMat <- matrix(c(1:100),nrow=4,ncol=4)
colnames(nMat)<- c("eps", "minPts","군집 수","Noise 수")

eps <- c(0.5,0.6)
minPts <- c(350,360)
for (i in 1:2){
  for(j in 1:2){
    nMat[(2*i-2)+j,1] <- eps[i]
    nMat[(2*i-2)+j,2] <- minPts[j]
    nMat[(2*i-2)+j,3] <- length(unique(dbscan(cX_scaled, eps = eps[i],
MinPts = minPts[j])$cluster))-1
    nMat[(2*i-2)+j,4] <- length(which((dbscan(cX_scaled, eps = eps[i],
MinPts = minPts[j])$cluster)==0))
  }
}
db_table<-as.data.frame(nMat)
View(db_table)
```

다음과 같이 Eps, MinPts 를 바꾸어가며 군집수와 노이즈 수를 확인해본다. 결과이다.

	eps	minPts	군집 수	Noise 수
1	0.5	350	1	698
2	0.5	360	3	757
3	0.6	350	1	54
4	0.6	360	1	69

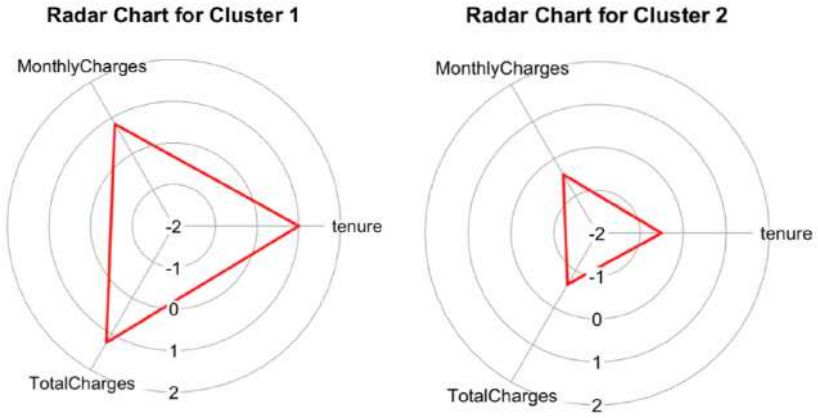
다음과 같이 군집수와 노이즈가 생성되었다. 군집수 1 개는 의미 없기 때문에 2 개를 만들기 위해 MinPts 를 350 에서 360 사이의 값인 352 로 지정해주었다.

```
# eps=0.5, MinPts = 350-360 사이가 군집수 2개가 될 것이라 생각해 결정.
DBSCAN_multishapes <- dbscan(cX_scaled, eps = 0.5, minPts = 352)
DBSCAN_multishapes #0이 노이즈, 1,2가 각각 클러스터

cluster_dbs <- data.frame(cX_scaled, clusterID = as.factor(DBSCAN_multishape
s$cluster))
dbs_summary <- data.frame()

for (i in 1:(ncol(cluster_dbs)-1)){
  dbs_summary = rbind(dbs_summary,
                      tapply(cluster_dbs[,i], cluster_dbs$clusterID, mean))
}
colnames(dbs_summary) <- paste("cluster", c(0:2))
dbs_summary #DBSCAN 의 클러스터 center 정보 확인

dbs_summary <- dbs_summary[,-1] #cluster0은 노이즈이기 때문에 제거한다.
```

```
DBSCAN clustering for 7032 objects.
Parameters: eps = 0.5, minPts = 352
The clustering contains 2 cluster(s) and 708 noise points.

   0    1    2
 708 4866 1458

Available fields: cluster, eps, minPts
```

결과를 보았을 때 0 이 뜻하는 바는 noise 이며, 1,2 는 클러스터이다. 노이즈는 708 개이다.
노이즈 열을 제거하고, Rader Chart 를 그려보았다.

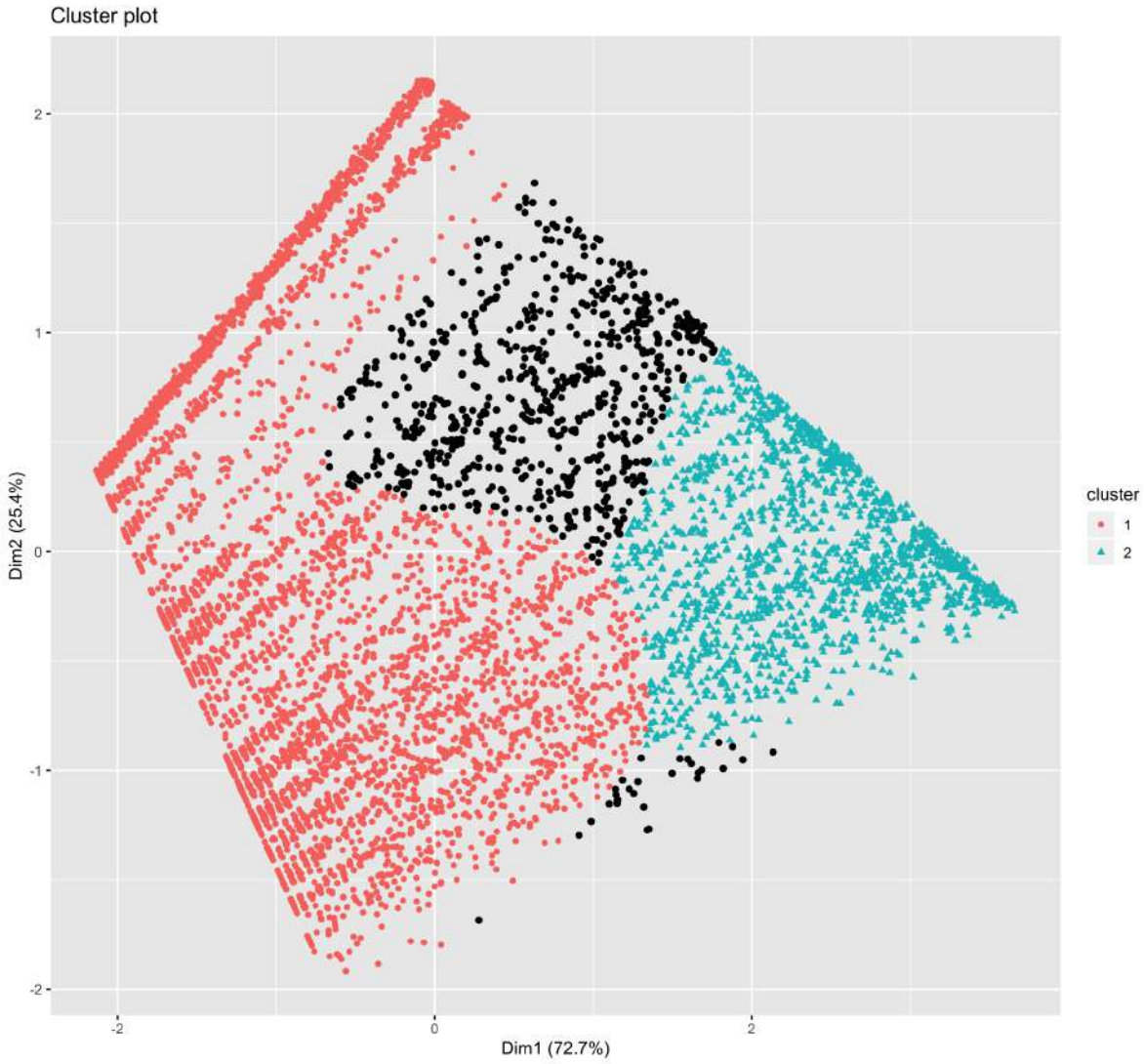
```
#DBSCAN Rader Chart
colnames(dbs_summary) <- paste("cluster", c(1:2))
rownames(dbs_summary) <- colnames(cX)

par(mfrow = c(2,1))
for (i in 1:2){
  plot_title <- paste("Radar Chart for Cluster", i, sep=" ")
  radial.plot(kmc_summary[,i], labels = rownames(kmc_summary),
              radial.lim=c(-2,2), rp.type = "p", main = plot_title,
              line.col = "red", lwd = 2.5, show.grid.labels=1)
}
dev.off()
```



차이를 보인다.

마지막으로 PCA 로 차원을 2 차원으로 축소하여 그래프를 그려보았다.

```
#PCA로 2차원으로 축소하여 그리기
fviz_cluster(DBSCAN_multishapes, cX_scaled, ellipse = FALSE, geom = "point",
show.clust.cent = FALSE)
```



노이즈가 검정색, 빨간색이 C1, 파란색이 C2 이다.
월비용, 총비용, 회사에 머무른 개월수가 크면 클러스터 1, 아니면 2 이다. 이 변수들이 크면 어떤 집단이고, 아니면 어떤 집단인지에 대해서는 전문가들의 의견이 필요한 부분이라고 생각된다.

Ⅳ Classification

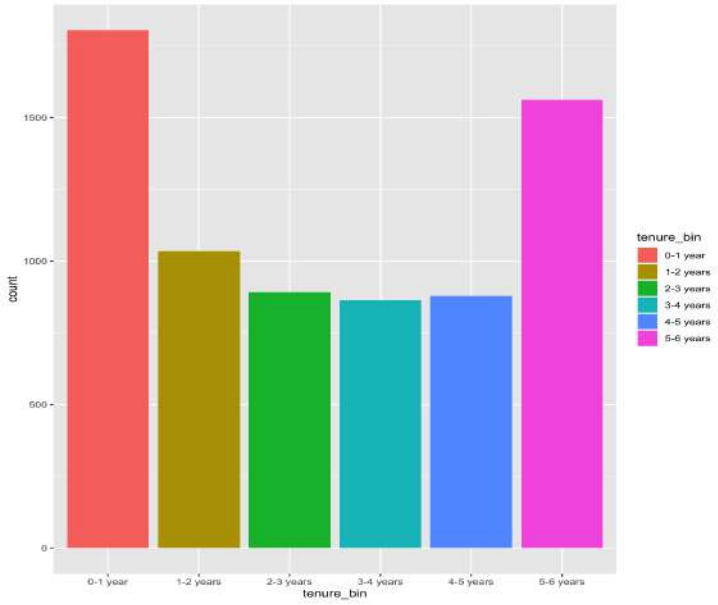
1. Logistic Regression

반응변수가 두개의 범주 중 하나에 속하는 자료를 고려할 때, 로지스틱 회귀는 반응변수 Y 를 직접 모델링하지 않고 Y 가 특정범주에 속하는 확률을 모델링한다. 로지스틱 함수로 0 과 1 로 바꾸어준다.

```
# Standardising Continuous features
# 연속형 변수에 대해 scale 차이가 있어 결과에 영향을 줄 수 있으므로
스케일을 맞춰준다.
df[,conIdx] <- sapply(df[,conIdx], as.numeric)
df_int <- df[,conIdx]
df_int <- data.frame(scale(df_int))
```

스케일을 맞춰주고 회사에 머무른 개월수가 월 단위이기 때문에 연 단위로 바꾸어 주고 factor 형 변수로 바꾸어준다.

```
#tenure이 월단위이기 때문에 연 단위로 바꾸어준다.
df <- mutate(df, tenure_bin = tenure)

df$tenure_bin[df$tenure_bin >=0 & df$tenure_bin <= 12] <- '0-1 year'
df$tenure_bin[df$tenure_bin > 12 & df$tenure_bin <= 24] <- '1-2 years'
df$tenure_bin[df$tenure_bin > 24 & df$tenure_bin <= 36] <- '2-3 years'
df$tenure_bin[df$tenure_bin > 36 & df$tenure_bin <= 48] <- '3-4 years'
df$tenure_bin[df$tenure_bin > 48 & df$tenure_bin <= 60] <- '4-5 years'
df$tenure_bin[df$tenure_bin > 60 & df$tenure_bin <= 72] <- '5-6 years'

df$tenure_bin <- as.factor(df$tenure_bin)

# 그려보기
ggplot(df, aes(tenure_bin, fill = tenure_bin)) + geom_bar()
```



범주형 변수에 대해 dummy 변수로 만들어주며, 스케일 된 연속형 변수와 합쳐 데이터 준비를 마친다. 이 데이터를 70%의 training, 30%의 validation sets 로 샘플을 뽑아 임의로 나눠준다.

```
## Creating Dummy Variables
df_cat <- df[,-c(5,18,19)]
dummy <- data.frame(sapply(df_cat,function(x) data.frame(model.matrix(~x-1,data =
df_cat))[,-1]))
head(dummy)

# Combining the data
df_final <- cbind(df_int,dummy)
head(df_final)

# Split the data into the training/validation sets
set.seed(12345)
trn_idx <- sample(1:nrow(df_final), round(0.7*nrow(df_final)))
df_trn <- df_final[trn_idx,]
df_tst <- df_final[-trn_idx,]
```

다음은 준비된 데이터를 가지고 Logistic Regression 의 full model(모든 변수 사용한 모델) 학습하는 과정이다.

```
# Logistic Regression
# Build the first model using all variables
full_lr <- glm(Churn ~ ., family=binomial, df_trn) #GLM_generalize linear
model
summary(full_lr) #유의수준. 이상
one_lr <- glm(Churn ~1, family=binomial, df_trn)
```

```
glm(formula = Churn ~ ., family = binomial, data = df_trn)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8588  -0.6870  -0.3072   0.7107   3.1704

Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -0.75156    0.36311  -2.070 0.038472 *
tenure                                -0.46899    0.25131  -1.866 0.062016 .
MonthlyCharges                        -0.06552    0.04489  -1.460 0.144386
TotalCharges                           0.05421    0.04451   1.218 0.223247
gender                                -0.08852    0.07694  -1.151 0.249889
SeniorCitizen                          0.05468    0.09987   0.547 0.584051
Partner                               -0.13586    0.09088  -1.495 0.134929
Dependents                            -0.16497    0.10609  -1.555 0.119958
PhoneService                          -0.35342    0.15821  -2.234 0.025495 *
MultipleLines                          0.20338    0.09076   2.241 0.025035 *
InternetService.xFiber.optic           0.80437    0.11337   7.095 1.29e-12 ***
InternetService.xNo                   -1.08409    0.17606  -6.158 7.39e-10 ***
OnlineSecurity                        -0.46042    0.09933  -4.635 3.56e-06 ***
OnlineBackup                          -0.35306    0.08881  -3.976 7.02e-05 ***
DeviceProtection                      -0.20327    0.09247  -2.198 0.027938 *
TechSupport                           -0.43503    0.10391  -4.187 2.83e-05 ***
StreamingTV                            0.21774    0.09481   2.297 0.021646 *
StreamingMovies                        0.21789    0.09396   2.319 0.020396 *
Contract.xOne.year                    -1.03786    0.12271  -8.458  < 2e-16 ***
Contract.xTwo.year                    -1.85848    0.19666  -9.450  < 2e-16 ***
PaperlessBilling                       0.35317    0.08901   3.968 7.25e-05 ***
PaymentMethod.xCredit.card..automatic. -0.07310    0.13399  -0.546 0.585368
PaymentMethod.xElectronic.check        0.37689    0.11256   3.348 0.000813 ***
PaymentMethod.xMailed.check            0.14265    0.13662   1.044 0.296429
tenure_bin.x1.2.years                 -0.39543    0.18522  -2.135 0.032770 *
tenure_bin.x2.3.years                 -0.10744    0.29890  -0.359 0.719246
tenure_bin.x3.4.years                  0.03060    0.42121   0.073 0.942088
tenure_bin.x4.5.years                 -0.08511    0.54631  -0.156 0.876192
tenure_bin.x5.6.years                  0.43361    0.67775   0.640 0.522314
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5652.2  on 4921  degrees of freedom
Residual deviance: 4158.5  on 4893  degrees of freedom
AIC: 4216.5

Number of Fisher Scoring iterations: 6
```

*이나 . 이 없는 변수는 무의미하다.

하지만 여러 모델을 비교한 후 나중에 변수선택을 최종적으로 할 것이기 때문에 일단은 넘어간다.

회귀 모형의 성능 비교 시 AIC(Akaike Information Criterion)을 사용한다. AIC 는 한 모델이 새로운 데이터를 예측할 수 있는 능력은 이 모델이 기존의 데이터를 얼마나 잘 설명하는지, 그리고 그 모델이 얼마나 간단한지의 두 요소에 의해 결정된다는 것을 말해준다. AIC 가 작을수록 올바른 모형에 가깝다.

앞으로 선택될 모델, 즉 변수 선택에 따른 AIC 를 정리하기 위해 다음과 같이 행렬을 만들어준다.

```
#AIC matrix 생성.
AIC_mat <- matrix(0, nrow = 5, ncol = 3)
# Initialize a performance summary
colnames(AIC_mat) <- c("Formula","AIC","Time")
rownames(AIC_mat) <- c("full","forward","back","stepwise","GA")
AIC_mat[1,1] <- as.character(summary(full_lr)$call)[2]
AIC_mat[1,2] <- summary(full_lr)$aic
AIC_mat[1,3] <- NA
AIC_mat
```

(1) Dimensionality Reduction

최적의 모델을 제공하는 변수의 서브 세트를 얻기 위해 변수를 추가하거나 제거하는 반복적인 프로세스인 변수 선택을 위해 stepAIC 을 사용할 것이다. 다음과 같이 Local 방법에 대해 먼저 수행한다.

```
#local (forward, backward, stepwise)
start_time <- proc.time()
model_forward <- stepAIC(one_lr, direction="forward",scope=list(upper
=full_lr,lower = one_lr))
end_time <- proc.time()
AIC_mat[2,3] <- (end_time - start_time)[3]
summary(model_forward)
AIC_mat[2,1] <- as.character(summary(model_forward)$call)[2]
AIC_mat[2,2] <- summary(model_forward)$aic

start_time <- proc.time()
model_back <- stepAIC(full_lr, direction = "backward")
end_time <- proc.time()
AIC_mat[3,3] <- (end_time - start_time)[3]
summary(model_back)
AIC_mat[3,1] <- as.character(summary(model_back)$call)[2]
AIC_mat[3,2] <- summary(model_back)$aic

start_time <- proc.time()
model_both <- stepAIC(one_lr, direction="both",scope=list(upper=full_lr
,lower = one_lr))
end_time <- proc.time()
AIC_mat[4,3] <- (end_time - start_time)[3]
summary(model_both)
AIC_mat[4,1] <- as.character(summary(model_both)$call)[2]
AIC_mat[4,2] <- summary(model_both)$aic
View(AIC_mat) #forward, backward, stepwise 가 AIC가 같아 stepwise 선택
```

Forward Selection: 선택 안된 모델에서 출발하여 변수를 하나씩 추가하고 다시 제거할 수 없다.
Backward Elimination: Full model 에서 변수를 제거한다. 다시 추가할 수 없다.
Stepwise selection: 아무것도 없는 모델에서 출발하여 Forward 와 Backward 를 반복한다.
각각 fit 지표가 향상되지 않을 때까지 반복한다. 여기서는 AIC 를 사용한다.

각각에 대한 시간과 AIC, 선택된 변수에 대해서는 GA 까지 진행한 후 보기로 했다.

GA

GA 는 Genetic Algorithm 의 약자로, 유전 알고리즘 (Genetic Algorithm) 이란 자연계에 있어서 생물의 유전 (Genetics) 과 진화 (Evolution) 의 메카니즘을 공학적으로 모델화하는 것에 의해 생물이 갖는 환경에서의 적응능력을 취급하는 것이다.

Genetic algorithm 은 풀고자하는 문제에 대한 가능한 해들을 정해진 형태의 자료구조로 표현한 다음, 이들을 점차적으로 변형함으로써 점점 더 좋은 해들을 생성한다. 즉 풀고자 하는 문제에 대한 가능한 해들을 염색체로 표현한 다음 이들을 점차적으로 변형함으로써 점점 더 좋은 해들을 생성한다. 각각의 가능한 해를 하나의 유기체 (Organism) 또는 개체 (Individual) 로 보며 이들의 집합을 개체군 (Population) 이라 한다. 하나의 개체는 보통 한 개 또는 여러 개의 염색체로 구성되며 염색체를 변형하는 연산자들을 유전연산자 (Genetic Operator) 라 한다. 기본적인 연산자는 다음의 3 가지가 있다. 선택 (Selection, 집단 중에서 적응도의 분포에 따라서 다음의 단계로 교배를 행하는 개체의 생존 분포를 결정한다. 적응도의 분포에 기초하고 있기 때문에 적응도가 높은 개체일수록 보다 많은 자손을 남기기 쉽게 된다), 교배 (Crossover, 2 개의 염색체 사이에서 유전자를 바꾸어 넣어 새로운 개체를 발생시킨다), 돌연변이 (Mutation, 유전자의 어떤 부분의 값을 강제적으로 변화시킨다)

```
fit_AIC <- function(string){
  sel_var_idx <- which(string == 1) #사용하는 변수의 index 항에 집어넣기.
  # Use variables whose gene value is 1
  sel_x <- x[, sel_var_idx] #사용하라고 지정된 것만 사용
  xy <- data.frame(sel_x, y)
  # Training the model
  GA_lr <- glm(y ~ .,family=binomial, xy)
  return(summary(GA_lr)$aic)
}

x <- as.matrix(df_trn[,-24])
y <- df_trn[,24]
```

다음과 같이 유전을 AIC 를 증가시키는 방향으로 하기 위해 fitness 함수를 작성한다.

```
start_time <- proc.time()
GA_AIC <- ga(type = "binary", fitness = fit_AIC, nBits = ncol(x),
             names = colnames(x), popSize = 100, pcrossover = 0.5,
             pmutation = 0.01, maxiter = 50, elitism = 0, seed = 123)
end_time <- proc.time()
AIC_mat[5,3] <- (end_time - start_time)[3]
```

원래 maxiter 를 100 이상으로 설정해야지 좋지만 시간이 너무 오래걸려 50 으로 설정했다. type = binary 는 염색체가 1,0 으로 구성되어 있다는 듯, fitness = 앞서 정의한 fitness(AIC), nBits= 변수개수, popsize = 한세대의 염색체 최대 몇개 가져갈 것인가, pcrossover = 난수생성으로 기준으로 바꾸는건데 cutoff 0.5 , pmutation 은 돌연변이 비율. maxiter 세대 수, elitism 정말 뛰어난 부모이면 계속 살아간다는것.. 자식세대가 부모를 뛰어넘는다는 보장 없어서 계속 살아남아 있는 수이다.

iter = 1	Mean = 4547.751	Best = 5197.677	iter = 26	Mean = 4540.054	Best = 5118.664
iter = 2	Mean = 4527.679	Best = 5197.677	iter = 27	Mean = 4547.972	Best = 5118.664
iter = 3	Mean = 4535.727	Best = 5197.677	iter = 28	Mean = 4546.820	Best = 5048.363
iter = 4	Mean = 4504.567	Best = 4840.875	iter = 29	Mean = 4522.705	Best = 4864.661
iter = 5	Mean = 4480.345	Best = 4840.875	iter = 30	Mean = 4537.471	Best = 4864.661
iter = 6	Mean = 4466.515	Best = 4814.460	iter = 31	Mean = 4517.248	Best = 4797.023
iter = 7	Mean = 4446.151	Best = 4814.460	iter = 32	Mean = 4541.762	Best = 4918.804
iter = 8	Mean = 4462.815	Best = 4814.460	iter = 33	Mean = 4545.774	Best = 4882.803
iter = 9	Mean = 4452.275	Best = 4831.376	iter = 34	Mean = 4555.839	Best = 4987.794
iter = 10	Mean = 4433.185	Best = 4831.376	iter = 35	Mean = 4569.652	Best = 4883.263
iter = 11	Mean = 4447.473	Best = 4823.551	iter = 36	Mean = 4575.319	Best = 5048.224
iter = 12	Mean = 4429.885	Best = 4859.070	iter = 37	Mean = 4596.083	Best = 5105.825
iter = 13	Mean = 4444.873	Best = 4831.376	iter = 38	Mean = 4606.791	Best = 5105.825
iter = 14	Mean = 4443.092	Best = 4812.123	iter = 39	Mean = 4594.619	Best = 5105.825
iter = 15	Mean = 4456.766	Best = 4864.565	iter = 40	Mean = 4579.918	Best = 5168.701
iter = 16	Mean = 4457.368	Best = 4864.565	iter = 41	Mean = 4577.091	Best = 5168.701
iter = 17	Mean = 4449.445	Best = 4969.184	iter = 42	Mean = 4592.545	Best = 5169.956
iter = 18	Mean = 4457.958	Best = 4969.184	iter = 43	Mean = 4574.120	Best = 5169.956
iter = 19	Mean = 4479.780	Best = 4969.184	iter = 44	Mean = 4607.672	Best = 5090.952
iter = 20	Mean = 4501.948	Best = 4971.225	iter = 45	Mean = 4601.083	Best = 5090.952
iter = 21	Mean = 4518.551	Best = 5118.664	iter = 46	Mean = 4613.642	Best = 5070.947
iter = 22	Mean = 4520.94	Best = 5184.23	iter = 47	Mean = 4605.789	Best = 5002.801
iter = 23	Mean = 4534.303	Best = 5118.664	iter = 48	Mean = 4613.771	Best = 5151.756
iter = 24	Mean = 4541.592	Best = 5118.664	iter = 49	Mean = 4595.181	Best = 5151.756
iter = 25	Mean = 4546.655	Best = 5118.664	iter = 50	Mean = 4614.282	Best = 5151.756

다음은 결과를 표로 정리한 것이다.

다음은 모든 Dimensionality Reduction 방법에 대해 AIC 와 시간을 비교한 표이다.

	Formula	AIC	Time
full	Churn ~ .	4216.51429	*NA*
forward	Churn ~ Contract.xTwo.year + Contract.xOne.year + InternetService.xFiber.optic + tenure + PaymentMethod.xElectronic.check + InternetService.xNo + OnlineSecurity + TechSupport + PaperlessBilling + OnlineBackup + tenure_bin.x1.2.years + tenure_bin.x5.6.years + StreamingMovies + Dependents + StreamingTV + DeviceProtection + MonthlyCharges + PhoneService + MultipleLines + PaymentMethod.xMailed.check + Partner	4207.08913	8.047
back	Churn ~ tenure + MonthlyCharges + Partner + Dependents + PhoneService + MultipleLines + InternetService.xFiber.optic + InternetService.xNo + OnlineSecurity + OnlineBackup + DeviceProtection + TechSupport + StreamingTV + StreamingMovies + Contract.xOne.year + Contract.xTwo.year + PaperlessBilling + PaymentMethod.xElectronic.check + PaymentMethod.xMailed.check + tenure_bin.x1.2.years + tenure_bin.x5.6.years	4207.08913	8.165
stepwise	Churn ~ Contract.xTwo.year + Contract.xOne.year + InternetService.xFiber.optic + tenure + PaymentMethod.xElectronic.check + InternetService.xNo + OnlineSecurity + TechSupport + PaperlessBilling + OnlineBackup + tenure_bin.x1.2.years + tenure_bin.x5.6.years + StreamingMovies + Dependents + StreamingTV + DeviceProtection + MonthlyCharges + PhoneService + MultipleLines + PaymentMethod.xMailed.check + Partner	4207.08913	13.408
GA	Churn ~ MonthlyCharges + gender + Partner + Dependents + OnlineSecurity + PaymentMethod.xCredit.card..automatic. + Churn + tenure_bin.x1.2.years	5149.75852	95.835

forward, backward 의 시간은 비슷하게 걸렸으며, Stepwise 는 더 걸렸다. GA 는 시간이 많이 걸렸다. GA 의 경우 AIC 가 크게 결정되었으며, 3 가지 방법은 모두 같게 나왔다. Forward, Backward, Stepwise 의 AIC 가 같고 작기 때문에 사용하면 좋겠다. 여기서 forward 와 stepwise 의 결과가 같기 때문에 이 변수들을 사용한다.

```
# forward Model
model_AIC <- glm(Churn ~ Contract.xTwo.year + Contract.xOne.year + InternetServic
e.xFiber.optic + tenure + PaymentMethod.xElectronic.check + InternetService.xNo +
OnlineSecurity + TechSupport + PaperlessBilling + OnlineBackup + tenure_bin.x1.2
.years + tenure_bin.x5.6.years + StreamingMovies + Dependents + StreamingTV +
DeviceProtection + MonthlyCharges + PhoneService + MultipleLines + PaymentMethod
.xMailed.check + Partner, family=binomial, df_trn)
summary(model_AIC)

#VIF(다중공산성) 계산
install.packages("car")
library("car")
vif(model_AIC) #모두 5보다 작아 제거할 필요없음.
```

결정된 모델에 대해 Summary 를 확인해보고 VIF(다중공산성)도 계산해본다.

```
Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                    -0.87491    0.16780  -5.214 1.85e-07 ***
Contract.xTwo.year             -1.85221    0.19594  -9.453  < 2e-16 ***
Contract.xOne.year             -1.03168    0.12187  -8.466  < 2e-16 ***
InternetService.xFiber.optic    0.82080    0.11213   7.320 2.48e-13 ***
tenure                         -0.48967    0.06164  -7.944 1.95e-15 ***
PaymentMethod.xElectronic.check 0.41615    0.09028   4.610 4.03e-06 ***
InternetService.xNo            -1.10995    0.17489  -6.347 2.20e-10 ***
OnlineSecurity                 -0.46060    0.09911  -4.647 3.36e-06 ***
TechSupport                    -0.44055    0.10350  -4.257 2.08e-05 ***
PaperlessBilling                0.35658    0.08872   4.019 5.84e-05 ***
OnlineBackup                   -0.34921    0.08859  -3.942 8.09e-05 ***
tenure_bin.x1.2.years          -0.40460    0.10578  -3.825 0.000131 ***
tenure_bin.x5.6.years           0.53731    0.15771   3.407 0.000657 ***
StreamingMovies                 0.21963    0.09379   2.342 0.019198 *
Dependents                     -0.17688    0.10374  -1.705 0.088185 .
StreamingTV                     0.21435    0.09464   2.265 0.023523 *
DeviceProtection               -0.20621    0.09231  -2.234 0.025496 *
MonthlyCharges                 -0.07494    0.04457  -1.681 0.092701 .
PhoneService                   -0.34163    0.15718  -2.173 0.029746 *
MultipleLines                   0.19938    0.09030   2.208 0.027238 *
PaymentMethod.xMailed.check     0.17772    0.11848   1.500 0.133622
Partner                        -0.13326    0.09010  -1.479 0.139144
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5652.2  on 4921  degrees of freedom
Residual deviance: 4163.1  on 4900  degrees of freedom
AIC: 4207.1

Number of Fisher Scoring iterations: 6
```

.이나 없는것을 제거한다. 즉 0.05 기준으로 큰 것은 제거한다고 생각하면 된다. 무의미하기 때문이다.

```
> vif(model_AIC) #모두 5보다 작아 제거할 필요없음.
             Contract.xTwo.year              Contract.xOne.year
                       1.333467                        1.226704
   InternetService.xFiber.optic                          tenure
                       2.041295                        2.363180
PaymentMethod.xElectronic.check             InternetService.xNo
                       1.381743                        1.831786
                OnlineSecurity                     TechSupport
                       1.119291                        1.185454
               PaperlessBilling                   OnlineBackup
                       1.123502                        1.154421
         tenure_bin.x1.2.years           tenure_bin.x5.6.years
                       1.056023                        2.041262
              StreamingMovies                      Dependents
                       1.452350                        1.236567
                  StreamingTV                DeviceProtection
                       1.468527                        1.244779
               MonthlyCharges                     PhoneService
                       1.416682                        1.466691
                MultipleLines       PaymentMethod.xMailed.check
                       1.367628                        1.504184
                      Partner
                       1.326279
```

VIF 를 계산한 결과 다 5 보다 작기 때문에 다중공산성이 없다고 볼 수 있다.

이탈 여부를 결정하는 데에 유의미하지 않은 변수(Dependents, Monthly, Charges)들을 제거한 결과를 확인한다.

```
# select model //0.1 기준으로 .이나 없는거 뺌(Dependents, Monthly Charges,
전자우편으로 지불, 파트너 제거)
model_select <-glm(Churn ~ Contract.xTwo.year + Contract.xOne.year + InternetServ
ice.xFiber.optic + tenure + PaymentMethod.xElectronic.check + InternetService.xNo
+ OnlineSecurity + TechSupport + PaperlessBilling + OnlineBackup + tenure_bin.x1
.2.years + tenure_bin.x5.6.years + StreamingMovies + StreamingTV + DeviceProtecti
on +  PhoneService + MultipleLines, family=binomial, df_trn)
summary(model_select)
vif(model_select) #5보다 작으므로 제거 안함.

final_model <- model_select
```

```
> vif(model_select) #5보다 작으므로 제거 안함.
             Contract.xTwo.year              Contract.xOne.year
                       1.320994                        1.208609
   InternetService.xFiber.optic                          tenure
                       1.905792                        2.320525
PaymentMethod.xElectronic.check             InternetService.xNo
                       1.128848                        1.616132
                OnlineSecurity                     TechSupport
                       1.115656                        1.176314
               PaperlessBilling                   OnlineBackup
                       1.115722                        1.140232
         tenure_bin.x1.2.years           tenure_bin.x5.6.years
                       1.054476                        2.036911
              StreamingMovies                      StreamingTV
                       1.429300                        1.442003
               DeviceProtection                     PhoneService
                       1.226721                        1.419793
                MultipleLines
                       1.346062
```

```
Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -0.85706    0.15623  -5.486 4.11e-08 ***
Contract.xTwo.year              -1.92474    0.19461  -9.890  < 2e-16 ***
Contract.xOne.year              -1.06471    0.12061  -8.828  < 2e-16 ***
InternetService.xFiber.optic     0.76748    0.10806   7.102 1.23e-12 ***
tenure                          -0.51478    0.06100  -8.439  < 2e-16 ***
PaymentMethod.xElectronic.check  0.36238    0.08148   4.448 8.68e-06 ***
InternetService.xNo             -1.00451    0.16407  -6.122 9.22e-10 ***
OnlineSecurity                  -0.47626    0.09875  -4.823 1.41e-06 ***
TechSupport                     -0.42986    0.10285  -4.180 2.92e-05 ***
PaperlessBilling                 0.35357    0.08826   4.006 6.17e-05 ***
OnlineBackup                    -0.35662    0.08783  -4.060 4.90e-05 ***
tenure_bin.x1.2.years           -0.42137    0.10549  -3.994 6.49e-05 ***
tenure_bin.x5.6.years            0.57884    0.15705   3.686 0.000228 ***
StreamingMovies                  0.22265    0.09294   2.396 0.016592 *
StreamingTV                      0.22221    0.09370   2.372 0.017713 *
DeviceProtection                -0.20430    0.09150  -2.233 0.025559 *
PhoneService                    -0.38562    0.15384  -2.507 0.012191 *
MultipleLines                    0.19035    0.08944   2.128 0.033325 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5652.2  on 4921  degrees of freedom
Residual deviance: 4177.6  on 4904  degrees of freedom
AIC: 4213.6

Number of Fisher Scoring iterations: 6
```

모델이 유의미한 변수들로 구성되었으며 다중공산성도 없어 최종 모델로 결정하였다.

----분류 모델의 성능을 평가하기 위한 함수를 작성한다(Confusion Matrix 기반)

```r
#perference evaluation funtion
perf_eval <- function(cm){
  # True positive rate: TPR (Recall)
  TPR <- cm[2,2]/sum(cm[2,]) #cm=>confusion matrix,R-script는 오름차순이어서
TPR이 2행.
  # True negative rate: TNR
  TNR <- cm[1,1]/sum(cm[1,])
  # False Positive Rate: FPR
  FPR <- cm[1,2]/sum(cm[1,])
  # False Negative Rate: FNR
  FNR <- cm[2,1]/sum(cm[2,])

  # Precision
  PRE <- cm[2,2]/sum(cm[,2])
  # Simple Accuracy
  ACC <- (cm[1,1]+cm[2,2])/sum(cm)
  # Balanced Correction Rate
  BCR <- sqrt(TPR*TNR)
  # F1-Measure
  F1 <- 2*TPR*PRE/(TPR+PRE)
  return(c(TPR, TNR, FPR, FNR, PRE, ACC, BCR, F1))
}
```

```
# Initialize the performance matrix
perf_mat <- matrix(0, 7, 8) #cut-off들에 대해
cutoff <- seq(from =0.2, to=0.8, by=0.1) #cutoff 0.2부터 0.6까지 올려가며
colnames(perf_mat) <- c("TPR (Recall)", "TNR","FPR","FNR","Precision", "ACC",
"BCR", "F1")
rownames(perf_mat) <- paste("cluster", cutoff)

all_perf_mat <- matrix(0, 7, 8) #로지스틱, DT, RF비교함.
colnames(all_perf_mat) <- c("TPR (Recall)", "TNR","FPR","FNR","Precision", "ACC",
"BCR", "F1")
rownames(all_perf_mat) <- c("Logistic", "Decision Tree","RandomForest","ANN","K
-NN","SVM","Naive Bayes")
```

성능 평가 행렬에 대해 초기화한다. perf_mat 는 로지스틱에서 cutoff 에 따라 비교하기 위함이고, 아래의 all_perf_mat 는 로지스틱, DF, Random Forest, ANN, k-NN, SVM, Naive Bayes 방법을 비교하기 위한 행렬이다.

```
# Model Evaluation using Validation Data
GLMpred <- predict(final_model, type = "response", newdata = df_tst[,-24])

#결과
for(i in 1:length(cutoff)){
  print(cutoff[i])
  pred_churn <- factor(ifelse(GLMpred >= cutoff[i], "Yes", "No"))
  actual_churn <- factor(ifelse(df_tst$Churn==1,"Yes","No"))
  cm_full <- table(actual_churn,pred_churn)
  #cm_full <- matrix(0, nrow = 2, ncol = 2)
  #cm_full[1,1] <- length(which(actual_churn == "No" & pred_churn == "No"))
  #cm_full[1,2] <- length(which(actual_churn == "No" & pred_churn == "Yes"))
  #cm_full[2,1] <- length(which(actual_churn == "Yes" & pred_churn == "No"))
  #cm_full[2,2] <- length(which(actual_churn == "Yes" & pred_churn == "Yes"))
  print(cm_full)
  perf_mat[i,] <- perf_eval(cm_full)
  if(i==5){
    all_perf_mat[1,] <- pref_eval(cm_full) #Decision tree 랑 비교하기 위해
default값인 0.5를 넣어줌
  }
}
```

다음과 같이 Final model 에 대해 예측을 해주고, cutoff 를 다르게 하여 성능을 비교해본다.

	TPR (Recall)	TNR	FPR	FNR	Precision	ACC	BCR	F1
cluster 0.2	0.86643836	0.6461337	0.35386632	0.1335616	0.4837476	0.7071090	0.7482212	0.62085890
cluster 0.3	0.77910959	0.7463958	0.25360419	0.2208904	0.5403800	0.7554502	0.7625773	0.63814867
cluster 0.4	0.63527397	0.8197903	0.18020970	0.3647260	0.5743034	0.7687204	0.7216588	0.60325203
cluster 0.5	0.47260274	0.8938401	0.10615990	0.5273973	0.6301370	0.7772512	0.6499471	0.54011742
cluster 0.6	0.29794521	0.9606815	0.03931848	0.7020548	0.7435897	0.7772512	0.5350051	0.42542787
cluster 0.7	0.12842466	0.9882045	0.01179554	0.8715753	0.8064516	0.7502370	0.3562440	0.22156573
cluster 0.8	0.02054795	0.9967235	0.00327654	0.9794521	0.7058824	0.7265403	0.1431105	0.03993344

결과를 보아 0.3 일때의 ACC, FI 가 높으므로 결정한다. 각 성능 지표가 뜻하는 것은 다음과 같다.

#TPR: 실제로 '예'일 때, 얼마나 자주 '예'라고 예측하는가?
#TNR: 실제로 '아니오'일 때, 얼마나 자주 '아니오'를 예측하는가?

#FPR: 실제로 '아니오'일 때, 얼마나 자주 '예'라고 예측하는가?
#FNR: 실제로 '예'일 때, 얼마나 자주 '아니오'를 예측하는가?

#Precision: '예'라고 예측했을 때, 얼마나 자주 정확한가?
#Accuracy: 전반적으로 얼마나 자주 분류가 정확한가?

#BCR(balanced correction rate): TPR, TNR 의 곱을 제곱근 취한 것으로 하나가 0 이면 0 이 되어 하나만 커도 과대평가되는 것을 막아준다
#FI-Measure: 정밀도(Precision: 찾아야할 것이라고 예측한 것 중 진짜 그런 것)과 재현율(TPR: 찾아야 할 것 중 실제로 찾은 비율)의 평균으로 성능을 평가할 때 자주 쓰인다.

따라서 FI 가 높은 것을 선택하였다. 하지만 이탈 안한 경우를 예측하는 것이 중요하지 않을 수도 있어 TPR 이 가장 큰 0.2 를 선택해도 좋다.

```
# Logistic Result
pred_churn <- factor(ifelse(GLMpred >= 0.3, "Yes", "No"))
actual_churn <- factor(ifelse(df_tst$Churn==1,"Yes","No"))
cm_full <- table(actual_churn,pred_churn)
print(cm_full)
```

로지스틱 결과는 다음과 같다.

```
             pred_churn
actual_churn   No  Yes
         No  1139  387
         Yes  129  455
```

뒤에 나올 Decision Tree 는 0.5 기반으로 분류하기 때문에 성능행렬에는 0.5 의 cutoff 일때의 값을 넣는다.

2. Decision Tree

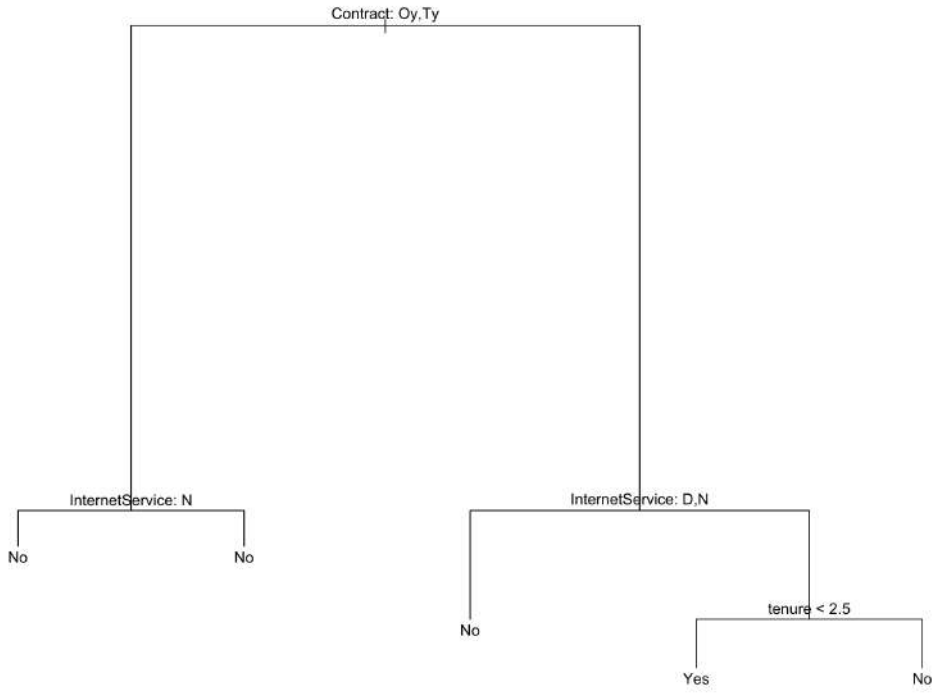
의사결정 트리는 설명변수 공간을 다수의 영역으로 계층화 또는 분할한다. 분할 규칙들은 트리로 요약될 수 있어 의사결정트리 방법으로 알려져 있다. 해석하기 쉽고 유용하지만 예측 정확도가 떨어진다는 단점이 있다.

```
#Preference Matrix
DT_perf_mat <- matrix(0, nrow = 4, ncol = 8)
rownames(DT_perf_mat) <- c("tree","rpart","RWeka","party")
colnames(DT_perf_mat) <- c("TPR (Recall)", "TNR","FPR","FNR","Precision", "ACC", "BCR",
"F1")
```

결정나무 결정하는 패키지는 여러 패키지가 있어 여러가지를 사용하여 best tree 를 설정하도록 하겠다.

Tree Package

```
# Training the tree
CART.model <- tree(Churn ~ ., df_trn, method = "class")
summary(CART.model)

# Plot the tree
plot(CART.model)
text(CART.model, pretty = 1)

# Prediction
CART.prey <- predict(CART.model, df_tst[,-24], type = "class")
CART.cfm <- table(df_tst$Churn, CART.prey)
CART.cfm

# Find the best tree
set.seed(12345)
CART.model.cv <- cv.tree(CART.model, FUN = prune.misclass) #cross validation으로
misclass 작아지도록 pruning

# Plot the pruning result
plot(CART.model.cv$size, CART.model.cv$dev, type = "b")
CART.model.cv #best 찾기, 5개일때 dev 가장 작으므로 원래 그린거랑 똑같음.
```



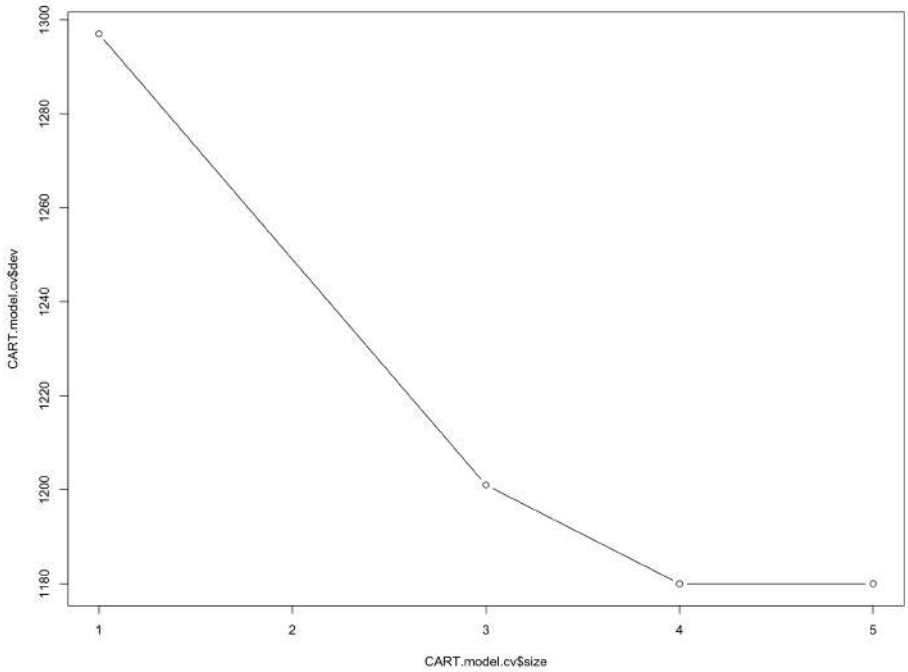
다음과 같이 계약기간이 1,2 년이 아니면 즉 월단위일때, 인터넷 서비스를 Fiber Optic 을 사용하며 회사에 머무른 개월수가 2.5 개월보다 작을 때 이탈한다고 볼 수 있다.

```
     CART.prey
        No  Yes
No   1518    8
Yes   515   69
```

결과는 위와 같으며 실제 이탈 했을 때 예측한 것이 어긋난다. 이는 매우 큰 손실이다. 왜냐하면 이탈하는 것을 잘 측정해야하기 때문이다.

```
# Find the best tree
set.seed(12345)
CART.model.cv <- cv.tree(CART.model, FUN = prune.misclass) #cross validation으로 misclass 작아지도록 pruning

# Plot the pruning result
plot(CART.model.cv$size, CART.model.cv$dev, type = "b")
CART.model.cv #best 찾기, 5개일때 dev 가장 작으므로 원래 그린거랑 똑같음.
```

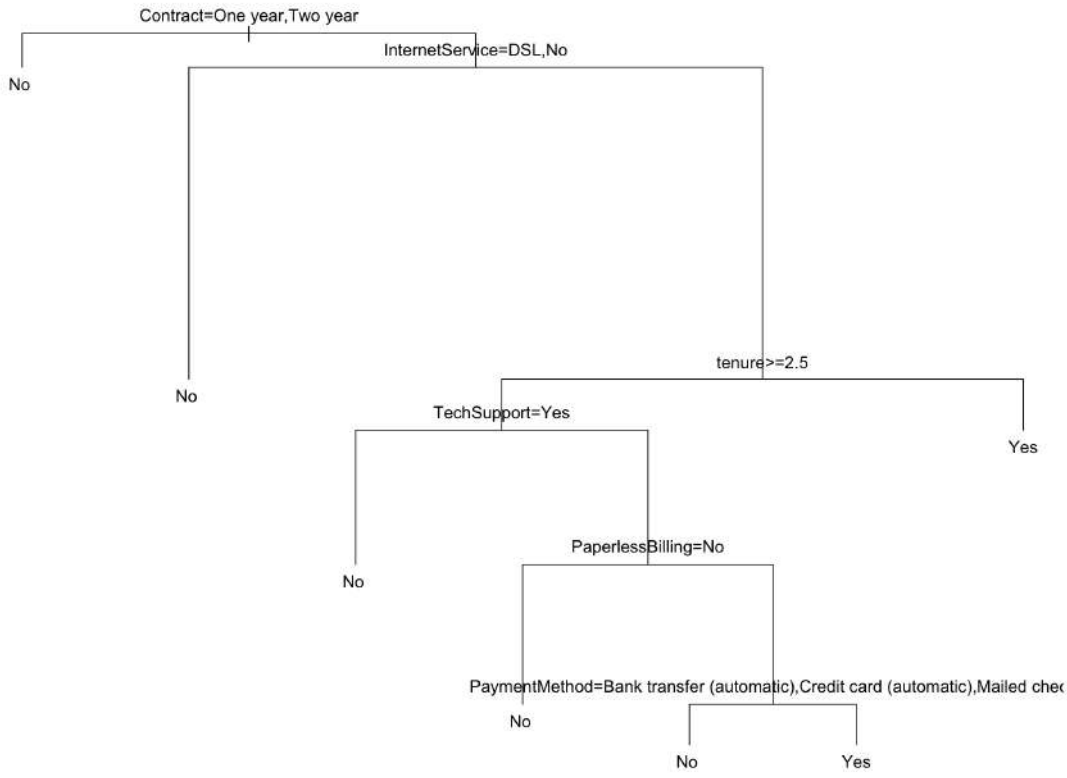
best tree 를 찾기 위해 pruning 하는 과정이다.



다음과 같이 dev 가 5 일때 가장 적게 나타나 위에서 그린 트리를 그래도 사용해도 좋다는 것을 알았다.

rpart

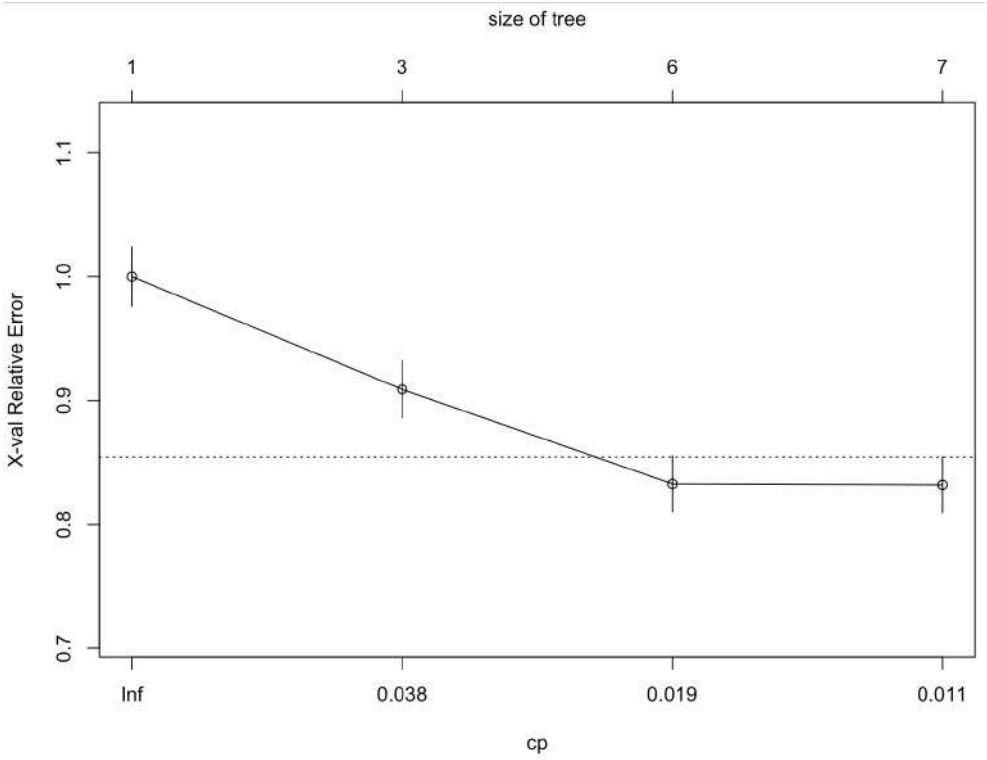
```
#rpart
rpart.model <- rpart(Churn ~ ., data = df_trn, method = "class")
plot(rpart.model)
text(rpart.model, pretty = 1)
```



다음 트리는 계약이 월단위이고, 인터넷서비스를 Fiber Optic 을 사용하며 머무른 개월수가 2.5 개월이 넘으며 기술 지원을 받지 않고 종이명세서를 받으며 자동신용카드 사용하지 않을 시 이탈한다고 보며, 계약이 월단위이고, 인터넷서비스를 Fiber Optic 을 사용하며 머무른 개월수가 2.5 개월이 넘지 않을 때 이탈한다고 본다.

```
#pruning rpart
printcp(rpart.model)
plotcp(rpart.model)

#rpart pruning 결과
rpart.model.pruned <- prune(rpart.model, cp= rpart.model$cptable[which.min(rpart
.model$cptable[,"xerror"]),"CP"])
plot(rpart.model.pruned)
text(rpart.model.pruned)

#rpart prediction & evaluation
rpart.prey <- predict(rpart.model.pruned, df_tst[,-24], type = 'class')
rpart.cfm <- table(df_tst$Churn, rpart.prey)
rpart.cfm
```

다음과 같이 pruning 을 진행한다.



error 가 가장 작을 때 size 는 6 이나 7 인데 leaf node 수가 가지치기 전에도 7 이기 때문에 그대로 진행한다.
결과는 다음과 같다.

```
    rpart.prey
       No  Yes
No   1432   94
Yes   385  199
```

여전히 진짜 이탈했을 때 이탈예측은 잘못된 것이 더 많다.

RWeka

```
##RWeka
RWeka.model <- J48(Churn ~ ., data = df_trn)

#option
C = c(0.4, 0.25, 0.05) #C: <pruning confidence>, Set confidence threshold for pruning.
(default 0.25)
M = c(2,10,20) # M: <minimum number of instances> Set minimum number of instances per
leaf. (default 2)

rWeka_pref_mat <- matrix(0, nrow = 9, ncol = 8)
rNames <- c()
colnames(rWeka_pref_mat) <- c("TPR (Recall)", "TNR","FPR","FNR","Precision", "ACC",
"BCR", "F1")
```

RWeka 의 경우 parameter 설정이 필요하며 C, M 이 있다. C 는 가지치기 confidence 의 threshold 를 의미하며, M 은 leaf 마다 최소 instance 개수이다.

```
for(i in 1:3){
  for(j in 1:3){
    rNames <- c(rNames, paste("C:",C[i],",M:",M[j]))
    RWeka.model.pruned <- J48(Churn ~ ., data = df_trn, control = Weka_control(C = C[i],
M = M[j]))
    #if(require("party",quietly = TRUE)){plot(RWeka.model.pruned)}
    RWeka.prey <- predict(RWeka.model.pruned, df_tst[,-24])
    RWeka.cfm <- table(df_tst$Churn, RWeka.prey)
    RWeka.cfm
    rWeka_pref_mat[(3*(i-1)+j),] <- perf_eval(RWeka.cfm)
  }
}
rownames(rWeka_pref_mat) <- rNames
View(rWeka_pref_mat)
#C: 0.4, M:2 일 때 TPR 가장 높음.
```

다음과 같이 for 문을 돌리면 각각의 경우에 대해 performance 가 입력된다. 결과는 다음과 같다.

	TPR (Recall)	TNR	FPR	FNR	Precision	ACC	BCR	F1
C: 0.4 ,M: 2	0.4948630	0.8466579	0.15334207	0.5051370	0.5525813	0.7492891	0.6472864	0.5221319
C: 0.4 ,M: 10	0.4777397	0.8761468	0.12385321	0.5222603	0.5961538	0.7658768	0.6469700	0.5304183
C: 0.4 ,M: 20	0.4126712	0.9003932	0.09960682	0.5873288	0.6132316	0.7654028	0.6095624	0.4933470
C: 0.25 ,M: 2	0.5000000	0.8689384	0.13106160	0.5000000	0.5934959	0.7668246	0.6591428	0.5427509
C: 0.25 ,M: 10	0.4623288	0.8866317	0.11336828	0.5376712	0.6094808	0.7691943	0.6402463	0.5258033
C: 0.25 ,M: 20	0.3921233	0.9193971	0.08060288	0.6078767	0.6505682	0.7734597	0.6004307	0.4893162
C: 0.05 ,M: 2	0.4743151	0.8879423	0.11205767	0.5256849	0.6183036	0.7734597	0.6489718	0.5368217
C: 0.05 ,M: 10	0.4777397	0.8866317	0.11336828	0.5222603	0.6172566	0.7734597	0.6508296	0.5386100
C: 0.05 ,M: 20	0.4880137	0.8912189	0.10878113	0.5119863	0.6319290	0.7796209	0.6594900	0.5507246

이번에 TPR 이 가장 높은 것을 결정하도록 하겠다 즉 C:0.4, M:2 이다.

```
#best model
RWeka.model.pruned <- J48(Churn ~ ., data = df_trn, control = Weka_control(C = 0.4, M =
2))
#plot
#if(require("party",quietly = TRUE)){plot(RWeka.model.pruned)}
# 플롯 그릴시 R이 shutdown 되어 주석처리한다.
RWeka.prey <- predict(RWeka.model.pruned, df_tst[,-24])
RWeka.cfm <- table(df_tst$Churn, RWeka.prey)
RWeka.cfm
```

best model 을 설정한 후 결과를 보면 다음과 같다. 여전히 비슷한 결과이다.

```
    RWeka.prey
       No  Yes
No   1292  234
Yes   295  289
```

Party

```
##party
party.model <- ctree(Churn ~ ., data = df_trn)
# party패키지는 가지치기를 significance를 사용해서 하기 때문에 별도의
pruning 과정이 필요 없다.
#[Q3-4] plot
plot(party.model)

#party prediction & evaluation
party.prey <- predict(party.model, df_tst[,-24])
party.cfm <- table(df_tst$Churn, party.prey)
party.cfm

#분류성능
DT_perf_mat[4,] <- perf_eval(party.cfm)
View(DT_perf_mat)
```

Party Package 의 경우도 비슷하게 나타났다.

```
    party.prey
       No  Yes
No   1398  128
Yes   347  237
```

다음은 모든 패키지를 비교한 결과이다.

	TPR (Recall)	TNR	FPR	FNR	Precision	ACC	BCR	F1
tree	0.1181507	0.9947575	0.005242464	0.8818493	0.8961039	0.7521327	0.3428284	0.2087746
rpart	0.3407534	0.9384010	0.061598952	0.6592466	0.6791809	0.7729858	0.5654762	0.4538198
RWeka	0.4948630	0.8466579	0.153342071	0.5051370	0.5525813	0.7492891	0.6472864	0.5221319
party	0.4058219	0.9161206	0.083879423	0.5941781	0.6493151	0.7748815	0.6097391	0.4994731

다음은 모든 tree package 를 비교한 결과이다. 우리는 TPR 이 높은 것이 필요하기 때문에 RWeka 패키지를 선택하여 모델을 결정한다.

3. Ensemble: Random Forest

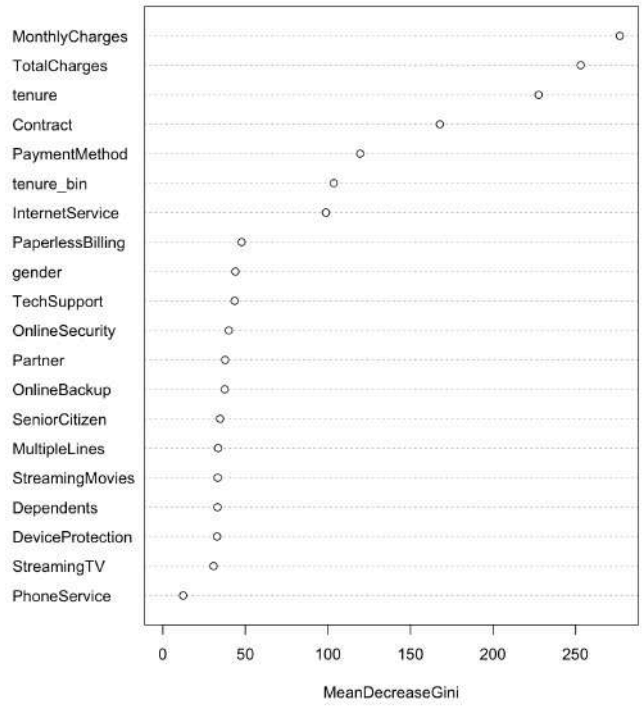
랜덤포레스트는 트리들의 상관성을 제거하는 간단한 방법으로 배깅된 트리들보다 더 나은 성능을 제공한다. 의사결정 트리를 만들 때 트리 내에서 분할이 고려될 때마다 p 개의 설명변수들의 전체 집합에서 m 개 설명변수로 구성된 랜덤 표본이 분할 후보로 선택된다. 분할은 이들 m 개 설명변수 중 하나만을 사용하도록 허용된다. 각 분할에서 m 개 설명변수들의 새로운 표본이 선택되며 보통 m = p^0.5 가 사용된다.

```
#Training the RandomForest Model
model.rf <- randomForest(Churn ~ ., data=df_trn, proximity=FALSE,importance = FALSE,
                         ntree=500,mtry=4, do.trace=FALSE)
model.rf
```

다음과 같이 모델을 생성하며 결과는 다음과 같다

```
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 20.7%
Confusion matrix:
      No Yes class.error
No  3299 338  0.09293374
Yes  681 604  0.52996109
```

Random forest 도 마찬가지로 TPR 이 좋지 않을 것이 예상된다.
아래는 변수 중요도 플롯으로 지니의 평균 감소로 인해 가장 중요한 속성이 감소하는 순서로 표시됩니다. Mean 감소 Gini 는 노드가 트리의 끝에 얼마나 순수한지를 측정한다. 지니 인덱스가 높을수록 동질성이 우수하다. varImpPlot(model.rf) 를 사용하여 보면, 분류에서 총 비용 변수가 중요하다는 것을 알 수 있다.



4. ANN: Artificial Neural Network

ANN 은 뇌의 뉴런을 본뜬 perceptron 여러개로 이루어진 모델이다. 학습시간이 오래 걸린다는 단점이 있지만 flexible 한 경계를 생성할 수 있다. 이것은 hidden node 의 개수에 따라 결정된다. 다음과 같이 설정한다.

```
# Artificial Neural Network
# Train ANN
ann_trn_input <- df_trn[,-24]
ann_trn_target <- class.ind(df_trn[,24])

# Find the best number of hidden nodes in terms of BCR
# Candidate hidden nodes
nH <- seq(from=5, to=30, by=5) #히든노드의 설정개수
```

Cross Validation 을 실시한다. Cross Validation 은 원래 데이터가 Training, Validation, Test data 세가지로 분류되어야 하는데 데이터가 적을 시 Training 데이터를 쪼개 하나씩 validation 데이터로 설정하여 반복하는 것이다.
ANN 은 데이터가 많아야 좋기 때문에 5-fold cross validation 을 수행하는 코드를 직접 적어 실행하였다.

```
# 5-fold cross validation index
val_idx <- sample(c(1:5), dim(ann_trn_input)[1], replace = TRUE, prob = rep(0.2,5))
# 1-5까지 랜덤하게 trn 데이터 수만큼 뽑기, 0.2의 확률로
val_perf <- matrix(0, length(nH), 3)

for (i in 1:length(nH)) {

  cat("Training ANN: the number of hidden nodes:", nH[i], "\n") #중간 print하기
  eval_fold <- c() #y, yhat matrix 만드려구..

  for (j in c(1:5)) {

    # Training with the data in (k-1) folds
    tmp_trn_input <- ann_trn_input[which(val_idx != j),]
    tmp_trn_target <- ann_trn_target[which(val_idx != j),]
    tmp_nnet <- nnet(tmp_trn_input, tmp_trn_target, size = nH[i], decay = 5e-4, maxit = 50)
    #decay => wij < 5*10^(-04) 이면 0으로 바꿔라. 실제 뉴런의 연결에서 일어나는 현상
    # maxit 최대 에폭의수 iteration 더 크게 바꿔야함.

    # Evaluate the model withe the remaining 1 fold
    tmp_val_input <- ann_trn_input[which(val_idx == j),]
    tmp_val_target <- ann_trn_target[which(val_idx == j),]

    eval_fold <- rbind(eval_fold, cbind(max.col(tmp_val_target),
                                        max.col(predict(tmp_nnet, tmp_val_input))))
  }

  # Confusion matrix
  cfm <- table(eval_fold[,1], eval_fold[,2])

  # nH
  val_perf[i,1] <-nH[i]
  # Record the validation performance
  val_perf[i,2:3] <- perf_eval_multi(cfm)
}
```

hidden	5			hidden	10		
#	weights	157		#	weights	312	
initial	value	2499.06979		initial	value	1750.10012	
iter	10	value	2059.79044	iter	10	value	1090.22994
iter	20	value	2058.12615	iter	20	value	1005.27062
iter	30	value	1596.00329	iter	30	value	960.117875
iter	40	value	1554.15203	iter	40	value	903.649442
iter	50	value	1530.69471	iter	50	value	860.572762
final	value	1530.69471		final	value	860.572762	
stopped	after	50	iterations	stopped	after	50	iterations
#	weights	157		#	weights	312	
initial	value	1975.60095		initial	value	1815.61253	
iter	10	value	1589.76818	iter	10	value	1085.0733
iter	20	value	1566.21279	iter	20	value	1028.43136
iter	30	value	1558.11978	iter	30	value	988.873152
iter	40	value	1554.09261	iter	40	value	930.842057
iter	50	value	1552.57655	iter	50	value	879.669076
final	value	1552.57655		final	value	879.669076	
stopped	after	50	iterations	stopped	after	50	iterations
#	weights	157		#	weights	312	
initial	value	2550.94466		initial	value	1658.78268	
iter	10	value	2063.58976	iter	10	value	1113.3689
iter	20	value	2062.39762	iter	20	value	1050.08461
iter	30	value	2062.29418	iter	30	value	1007.26052
iter	40	value	1757.19838	iter	40	value	965.267906
iter	50	value	1582.19943	iter	50	value	926.084009
final	value	1582.19943		final	value	926.084009	
stopped	after	50	iterations	stopped	after	50	iterations
#	weights	157		#	weights	312	
initial	value	1802.31458		initial	value	2139.96114	
iter	10	value	1447.64251	iter	10	value	1324.06109
iter	20	value	1126.79909	iter	20	value	1088.13135
iter	30	value	1072.25342	iter	30	value	1026.172
iter	40	value	1050.62294	iter	40	value	965.737222
iter	50	value	1039.12211	iter	50	value	924.637496
final	value	1039.12211		final	value	924.637496	
stopped	after	50	iterations	stopped	after	50	iterations
#	weights	157		#	weights	312	
initial	value	2056.56416		initial	value	2380.86363	
iter	10	value	1136.22158	iter	10	value	2094.72789
iter	20	value	1098.68367	iter	20	value	1835.40311
iter	30	value	1084.65531	iter	30	value	1620.12014
iter	40	value	1070.54134	iter	40	value	1592.10344
iter	50	value	1050.56991	iter	50	value	1567.15269
final	value	1050.56991		final	value	1567.15269	
stopped	after	50	iterations	stopped	after	50	iterations

다음은 hidden 노드 개수가 5,10 일 때의 결과이다.

```
ordered_val_perf <- val_perf[order(val_perf[,3], decreasing = TRUE),] #BCR 기준으로
내림차순으로 정렬
colnames(ordered_val_perf) <- c("nH", "ACC", "BCR")
ordered_val_perf #PERFORMANCE가 어느지점에서 꺾이는지 더 범위 확장해주어야함. MAX가 가장
좋은 성능일 때,

# Find the best number of hidden node
best_nH <- ordered_val_perf[1,1]

# Test the ANN
ann_tst_input = df_tst[,-24]
ann_tst_target = class.ind(df_tst[,24])

ctgs_nnet <- nnet(ann_trn_input, ann_trn_target, size = best_nH, decay = 5e-4, maxit =
50)
```

다음과 같이 BCR 기준으로 내림차순으로 정렬하면 다음과 같다.

```
nH       ACC       BCR
15 0.7629013 0.6440629
10 0.7606664 0.5876194
30 0.7431938 0.4977297
 5 0.7602601 0.4432447
20 0.7541650 0.4358804
25 0.7468509 0.4208707
```

즉 hidden node 가 15 일 때 BCR 이 가장 높아 선택한다. 그걸 기반으로 다시 학습시킨다.

5. k-NN: k-Nearest Neighbor

k-NN 은 instanced based Classifier 로 비모수방식이다. noise 에 영향을 많이 받으며 결정 바운더리가 매우 유연하다. k-NN 은 새로 들어온 "★은 ■ 그룹의 데이터와 가장 가까우니 ★은 ■ 그룹이다." 라고 분류하는 알고리즘이다. 여기서 k 의 역할은 몇 번째로 가까운 데이터까지 살펴볼 것인가를 정한 숫자이다.

```
control <- trainControl(
  method = "cv",
  number = 10,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  verboseIter = FALSE
)
knn_model <- train(Churn ~ ., data = df_trn, method = "knn", trControl = control,
preProcess = c("center","scale"), tuneLength = 50)
knn_model

KNNpred <- predict(knn_model, df_tst[,-24])
knn_cm <- table(df_tst$Churn,KNNpred)
knn_cm
```

cross validation 함수를 설정해준다. 10-fold cross validation 이다. 또한 caret 패키지의 train 함수를 이용한다.

```
k-Nearest Neighbors

4922 samples
  28 predictor
   2 classes: 'No', 'Yes'

Pre-processing: centered (28), scaled (28)
Resampling: Cross-Validated (10 fold)
```

k	ROC	Sens	Spec	k	ROC	Sens	Spec
5	0.7768297	0.838317	0.5212694	55	0.8195072	0.875996	0.5251999
7	0.7899735	0.8509649	0.5212997	57	0.8198874	0.8770956	0.5267684
9	0.7955079	0.8559191	0.5205305	59	0.8197614	0.8759967	0.5244489
11	0.8018249	0.8581199	0.5081274	61	0.8198973	0.8781991	0.5291303
13	0.8030483	0.8531665	0.5159036	63	0.8204521	0.8765485	0.5330245
15	0.8059458	0.8614234	0.5166546	65	0.8204827	0.876827	0.5376938
17	0.8083043	0.8636235	0.5244489	67	0.8209396	0.8771002	0.5361374
19	0.811121	0.8652749	0.5283733	69	0.8210098	0.875725	0.5299176
21	0.8123269	0.8608793	0.5229046	71	0.8212252	0.8751756	0.5267987
23	0.8124307	0.8636265	0.5252301	73	0.821348	0.8771009	0.5275678
25	0.8132029	0.8625261	0.5221294	75	0.8219662	0.8787485	0.5298995
27	0.8140958	0.8655519	0.5166424	77	0.8220077	0.8806739	0.5314559
29	0.815924	0.8655511	0.5166546	79	0.8214881	0.8826007	0.5275618
31	0.8171804	0.8658258	0.5135659	81	0.8216692	0.8815003	0.5252241
33	0.8173053	0.8694018	0.5073401	83	0.8218852	0.8815003	0.521336
35	0.8173062	0.8688516	0.5174843	85	0.8220859	0.881775	0.5275799
37	0.8177662	0.8726978	0.5205729	87	0.8219822	0.8831517	0.5267805
39	0.8184074	0.8713272	0.5228985	89	0.8219406	0.8831509	0.5244247
41	0.8182986	0.8705	0.5182292	91	0.8218737	0.8820536	0.5306504
43	0.8184387	0.8729732	0.5190165	93	0.8218765	0.8828785	0.5298813
45	0.8184306	0.869952	0.5252483	95	0.82166	0.8837049	0.5283188
47	0.8183923	0.8718736	0.5314559	97	0.8214482	0.885628	0.5267502
49	0.8191864	0.8718713	0.5283309	99	0.8212732	0.8842529	0.5205366
51	0.8190937	0.8710471	0.5314438	101	0.8211058	0.8839781	0.5244065
53	0.8188625	0.8759975	0.5244247	103	0.8211914	0.8839774	0.5205184

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 85.

결과는 다음과 같으며, 즉 K=85 로 설정한다. 따라서 그것으로 예측하면 다음과 같다.

```
    KNNpred
       No  Yes
No   1336  190
Yes   286  298
```

KNN 은 이탈을 이탈으로 예측하는 개수가 더 많이 나왔음을 알 수 있다. (의사결정나무와 비교하여)

6. SVM: Support Vector Machine

서포트 벡터 머신은 최대마진분류기라고 불리는 가장 단순하고 직관적인 분류기를 일반화한 것이다. 두개의 이진 분류 설정을 위한 것이다. 초평면을 정의하고 최적의 분류 초평면을 선택한다. 일반적으로 초평면을 사용하여 데이터가 완벽하게 분류될 수 있는 무한개의 초평면이 존재한다. 아주 약간 위아래로 조정될 수 있기 때문이다. 분리 초평면을 기반으로 분류기를 구성하기 위해 무한개의 가능한 분리 초평면 중 어느 것을 사용할 지 결정하는 합리적인 방법이 필요하다.

이는 훈련관측치로부터 가장 멀리 떨어진 분리 초평면인 최대마진 초평면을 선택하는 것이 바람직하다.

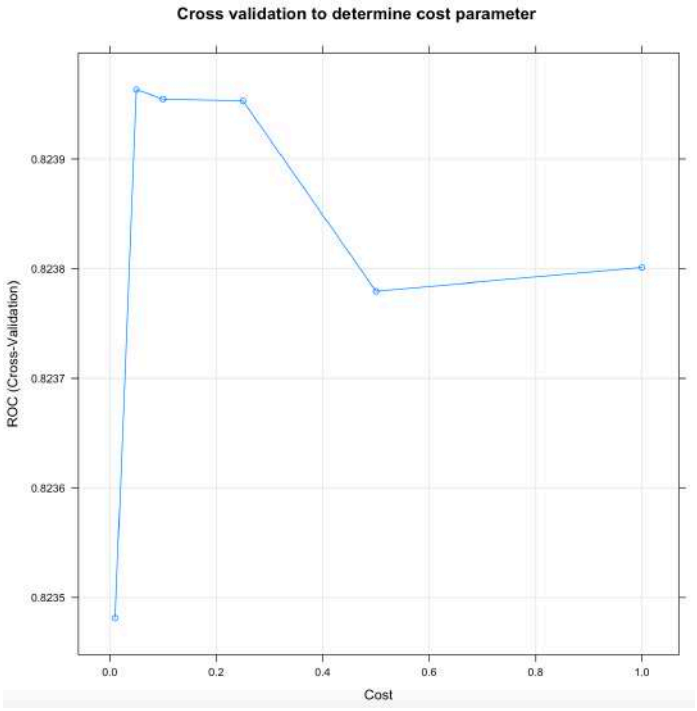
```
# 6-fold validation
grid <- expand.grid(C = c(0.01, 0.05, 0.1, 0.25, 0.5, 1)) #cost parameter
svm_linear_model <- train(Churn ~., data = df_trn, method = "svmLinear", trControl=
control, preProcess = c("center", "scale"),tuneLength = 6,tuneGrid = grid)
svm_linear_model
plot(svm_linear_model, main = "Cross validation to determine cost parameter")

SVMpred <- predict(svm_linear_model, newdata = df_tst[,-24])
svmcm <- table(df_tst[,24],SVMpred)
svmcm
```

이 경우 6-fold Validation 을 진행하였다.

```
Support Vector Machines with Linear Kernel

4922 samples
  28 predictor
   2 classes: 'No', 'Yes'

Pre-processing: centered (28), scaled (28)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4430, 4430, 4429, 4429, 4429, 4429, ...
Resampling results across tuning parameters:

  C     ROC        Sens       Spec
  0.01  0.8234813  0.9087261  0.4723232
  0.05  0.8239632  0.9081729  0.4660974
  0.10  0.8239549  0.9098228  0.4684472
  0.25  0.8239532  0.9098220  0.4707788
  0.50  0.8237794  0.9092725  0.4684351
  1.00  0.8238011  0.9098235  0.4692103

ROC was used to select the optimal model using the largest value.
The final value used for the model was C = 0.05.
```

여기서 C 는 조율 파라메터이다. M 이 마진의 폭일 때 가능한 값을 크게 하는데, 슬랙변수가 입실론이라고 했을 때 입실론이 1 보다 크면 관측치는 초평면의 옳지 않은 쪽에 있다. 이 경우 입실론의 합을 한정하여 마진(그리고 초평면)에 대한 허용될 위반의 수와 그 정도를 결정하는 것이 C 이다. C 가 0 이면 마진을 위반할 예산이 없으며 입실론이 모두 0 이어야한다. C 가 증가함에 따라 마진 위반에 대한 허용정도가 더 크게 되어 마진의 폭이 넓어진다.

여기서는 C=0.05 로 형성되었다.



다음과 같이 그래프를 그려서 확인할 수도 있다. C=0.05 일 때 ROC(AUC) 가 작다.

	SVMpred	
	No	Yes
No	1392	134
Yes	325	259

다음과 같이 결과가 나왔다. 이도 결정나무와 마찬가지로 TPR 이 작을 것으로 예상된다.

7. Naive Bayesian Classifier

기계 학습분야에서, '나이브 베이즈 분류(Naïve Bayes Classification)는 특성들 사이의 독립을 가정하는 베이즈 정리를 적용한 확률 분류기의 일종이다.

나이브 베이즈는 분류기를 만들 수 있는 간단한 기술로써 단일 알고리즘을 통한 훈련이 아닌 일반적인 원칙에 근거한 여러 알고리즘들을 이용하여 훈련된다. 모든 나이브 베이즈 분류기는 공통적으로 모든 특성 값은 서로 독립임을 가정한다. 예를 들어, 특정 과일을 사과로 분류 가능하게 하는 특성들 (둥글다, 빨갛다, 지름 10cm)은 나이브 베이즈 분류기에서 특성들 사이에서 발생할 수 있는 연관성이 없음을 가정하고 각각의 특성들이 특정 과일이 사과일 확률에 독립적으로 기여 하는 것으로 간주한다.

나이브 베이즈의 장점은 다음과 같다. 첫째, 일부의 확률 모델에서 나이브 베이즈 분류는 지도 학습 (Supervised Learning) 환경에서 매우 효율적으로 훈련 될 수 있다. 많은 실제 응용에서, 나이브 베이즈 모델의 파라미터 추정은 최대우도방법 (Maximum Likelihood Estimation (MLE))을 사용하며, 베이즈 확률론이나 베이지안 방법들은 이용하지 않고도 훈련이 가능하다. 둘째, 분류에 필요한 파라미터를 추정하기 위한 트레이닝 데이터의 양이 매우 적다는 것이다. 셋째, 간단한 디자인과 단순한 가정에도 불구하고, 나이브 베이즈 분류는 많은 복잡한 실제 상황에서 잘 작동한다. 2004 년의 한 분석[3]은 나이브 베이즈 분류의 이러한 능력에 명확한 이론적인 이유가 있음을 보여 주었다. 또한

2006 년에는 다른 분류 알고리즘과의 포괄적인 비교를 통하여 베이지안 분류는 부스트 트리 또는 랜덤 포레스트와 같은 다른 접근 방식을 넘어섰다는 것이 밝혀졌다.

```
# Training the Naive Bayesian Classifier
NB.model <- naiveBayes(Churn ~ ., data = df_trn)
NB.model

# Predict the new input data based on Naive Bayesian Classifier
NB.posterior = predict(NB.model, df_tst[-24], type = "raw")
NBpred = predict(NB.model, df_tst[-24], type ="class")

NB.cfm <- table(df_tst[,24], NBpred)
NB.cfm
```

결과는 다음과 같다.

A-priori	probabilities				DeviceProtection		
Y				Y			
	No	Yes			No	0.3637613	0.4811473
	0.7389273	0.2610727			Yes	0.2848249	0.4515067
Conditional	probabilities				TechSupport		
	tenure			Y			
Y					No	0.3332417	0.4714369
	No	0.1627543	0.976186		Yes	0.1571984	0.3641295
	Yes	-0.4324471	0.9427314				
					StreamingTV		
	MonthlyCharges			Y			
Y					No	0.3623866	0.4807558
	No	-0.1018728	0.9708759		Yes	0.4287938	0.4950964
	Yes	0.2915999	1.0239979				
					StreamingMovies		
	TotalCharges			Y			
Y					No	0.3698103	0.4828196
	No	-0.0139868	0.9857568		Yes	0.4412451	0.4967292
	Yes	0.01103952	1.0219317				
					Contract.xOne.year		
	gender			Y			
Y					No	0.2559802	0.4364708
	No	0.5166346	0.4997919		Yes	0.09105058	0.287793
	Yes	0.4996109	0.5001945				
					Contract.xTwo.year		
	SeniorCitizen			Y			
Y					No	0.31784438	0.4657027
	No	0.1308771	0.3373123		Yes	0.03035019	0.1716158
	Yes	0.2536965	0.4352954				
					PaperlessBilling		
	Partner			Y			
Y					No	0.5320319	0.4990415
	No	0.531207	0.4990938		Yes	0.7548638	0.4303354
	Yes	0.3634241	0.4811727				
					PaymentMethod.xCredit.card..automatic.		
	Dependents			Y			
Y					No	0.251306	0.4338238
	No	0.3508386	0.4772981		Yes	0.1322957	0.3389439
	Yes	0.1789883	0.3834918				
					PaymentMethod.xElectronic.check		

	PhoneService			Y			
Y					No	0.2524058	0.4344526
	No	0.9021171	0.2971971		Yes	0.566537	0.495746
	Yes	0.9143969	0.279886				
					PaymentMethod.xMailed.check		
	MultipleLines			Y			
Y					No	0.251306	0.4338238
	No	0.4069288	0.4913289		Yes	0.1642023	0.3706033
	Yes	0.4723735	0.4994306				
					tenure_bin.x1.2.years		
	InternetService.xFiber.optic			Y			
Y					No	0.1462744	0.3534297
	No	0.3497388	0.476953		Yes	0.1540856	0.3611713
	Yes	0.6996109	0.4586057				
					tenure_bin.x2.3.years		
	InternetService.xNo			Y			
Y					No	0.1182293	0.3229238
	No	0.27385208	0.4459953		Yes	0.12607	0.3320575
	Yes	0.06070039	0.2388729				
					tenure_bin.x3.4.years		
	OnlineSecurity			Y			
Y					No	0.1352763	0.3420655
	No	0.3354413	0.4722094		Yes	0.1027237	0.3037159
	Yes	0.1634241	0.3698961				
					tenure_bin.x4.5.years		
	OnlineBackup			Y			
Y					No	0.14737421	0.3545273
	No	0.3753093	0.4842693		Yes	0.07003891	0.2553119
	Yes	0.2754864	0.446933				
					tenure_bin.x5.6.years		
				Y			
					No	0.2623041	0.4399476
					Yes	0.1128405	0.3165209

각각이 독립이라고 가정하여 확률을 구한 것이다.

```
    NBpred
       No  Yes
No   1126  400
Yes   140  444
```

결과는 다음과 같이 나왔다. 제대로 분류된 비율이 지금까지의 방법들보다 높아보인다.

8. Performance 비교 & 모델선정

지금까지의 모든 분류 모델들의 performance 를 비교해보겠다.

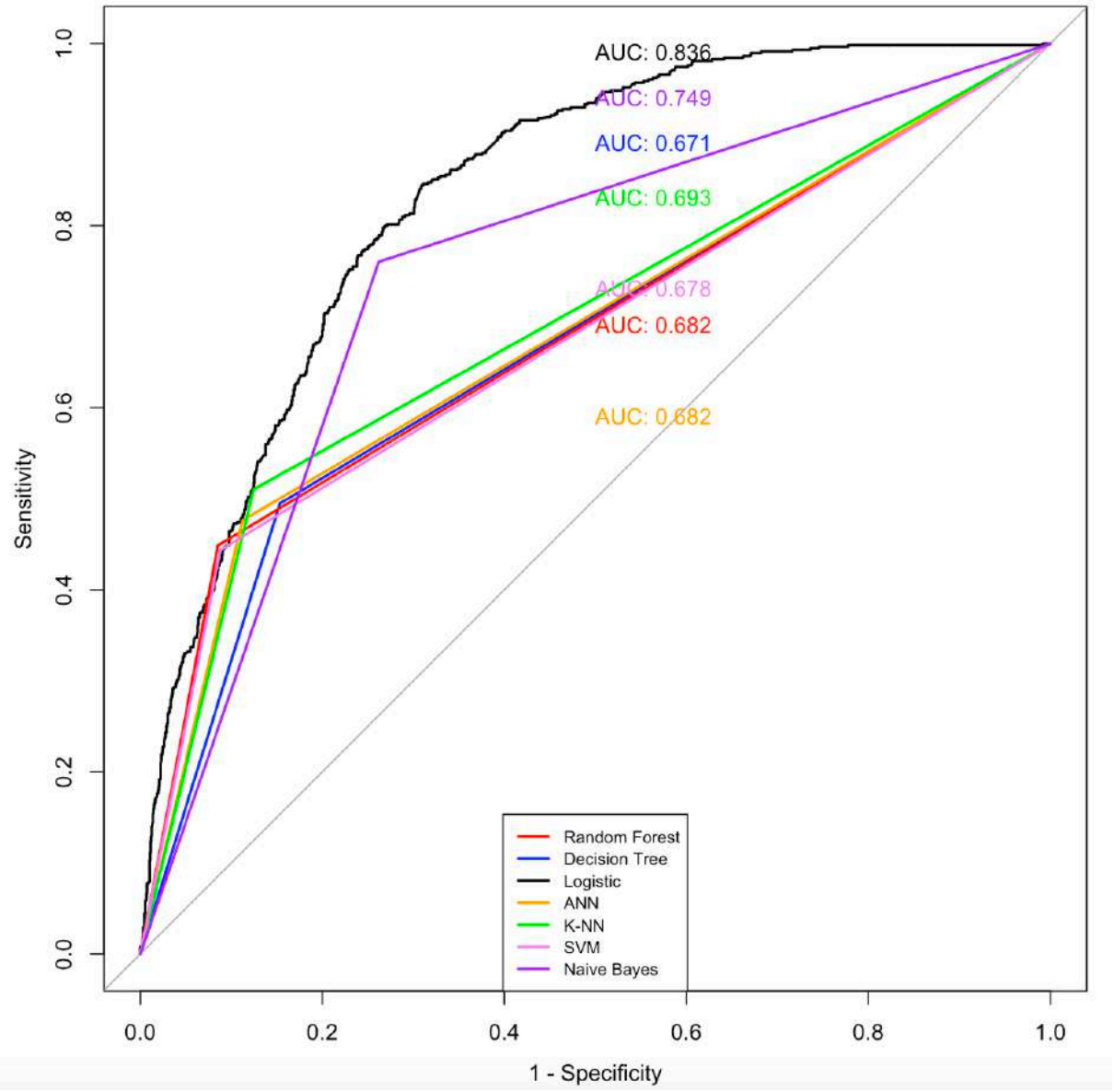
먼저 성능 평가 지표의 값들이 담긴 all_perf_mat 를 본다.

	TPR (Recall)	TNR	FPR	FNR	Precision	ACC	BCR	F1
Logistic	0.4726027	0.8938401	0.10615990	0.5273973	0.6301370	0.7772512	0.6499471	0.5401174
Decision Tree	0.4948630	0.8466579	0.15334207	0.5051370	0.5525813	0.7492891	0.6472864	0.5221319
RandomForest	0.4486301	0.9148100	0.08519004	0.5513699	0.6683673	0.7857820	0.6406335	0.5368852
ANN	0.4760274	0.8879423	0.11205767	0.5239726	0.6191537	0.7739336	0.6501422	0.5382381
K-NN	0.5102740	0.8754915	0.12450852	0.4897260	0.6106557	0.7744076	0.6683865	0.5559701
SVM	0.4434932	0.9121887	0.08781127	0.5565068	0.6590331	0.7824645	0.6360420	0.5301945
Naive Bayes	0.7602740	0.7378768	0.26212320	0.2397260	0.5260664	0.7440758	0.7489917	0.6218487

naive Bayes 의 경우 TPR 이 월등히 높게 나타났다. Accuracy 는 SVM 이 가장 높았다. 우리가 중요하게 생각하는 것이 TPR 이기 때문에 이 결과를 보면 Naive Bayes 를 사용할 것이다.

다른 평가 지표로 AUROC 를 구해본다.

```
library(pROC)
library(cowplot)

glm.roc <- roc(response = df_tst$Churn, predictor = as.numeric(GLMpred))
dt.roc <- roc(response = df_tst$Churn, predictor = as.numeric(DTpred))
rf.roc <- roc(response = df_tst$Churn, predictor = as.numeric(RFpred))
ann.roc <- roc(response = df_tst$Churn, predictor = as.numeric(ANNpred))
knn.roc <- roc(response = df_tst$Churn, predictor = as.numeric(KNNpred))
svm.roc <- roc(response = df_tst$Churn, predictor = as.numeric(SVMpred))
nb.roc <- roc(response = df_tst$Churn, predictor = as.numeric(NBpred))

plot(glm.roc, legacy.axes = TRUE, print.auc.y = 1.0, print.auc = TRUE)
plot(dt.roc, col = "blue", add = TRUE, print.auc.y = 0.9, print.auc = TRUE)
plot(rf.roc, col = "red" , add = TRUE, print.auc.y = 0.7, print.auc = TRUE)
plot(ann.roc, col = "orange" , add = TRUE, print.auc.y = 0.6, print.auc = TRUE)
plot(knn.roc, col = "green" , add = TRUE, print.auc.y = 0.84, print.auc = TRUE)
plot(svm.roc, col = "violet" , add = TRUE, print.auc.y = 0.74, print.auc = TRUE)
plot(nb.roc, col = "purple" , add = TRUE, print.auc.y = 0.95, print.auc = TRUE)

legend("bottom", c("Random Forest", "Decision Tree", "Logistic","ANN","K-NN","SVM"
,"Naive Bayes"),
       lty = c(1,1), lwd = c(2, 2), col = c("red", "blue", "black","orange","green"
,"violet","purple"), cex = 0.75)
```



다음과 같은 Curve 가 그려졌고, AUC 가 1 에 가까울수록 좋다. Logistic 이 가장 높게 나왔고 그 다음은 Naive Bayes 가 높게 나왔다.

하지만 우리가 관심있는 TPR 은 Naive 가 높기 때문에 최종적으로도 Naive Bayes Classifier 모델으로 결정할 것이다.

기존연구 & 비교

<Applying data mining to telecom churn management Shin-Yuan Hung a, David C. Yen b,*, Hsiu-Yu Wang c>

>정리

대만의 무선통신 회사의 데이터에 대한 연구이다.

대만의 무선 통신 회사는 고객 관련 데이터를 제공한다. 고객의 개인 정보를 보호하기 위해 데이터 소스에는 2001 년 7 월부터 2002 년 6 월까지 전화 번호를 기준으로 무작위로 선택된 14,000 개의 churners 를 포함하여 약 160,000 명의 가입자가 포함된다.

이 논문에서는 의사결정트리(C5.0)과 역전파신경망(BPN) 기술로 구현된 모델이 어떻게 수행되는지 평가하기 위해 K-means clustering 으로 청구금액, 회사 머무른 기간, 지불방법에 따라 각 클러스터에 따라 의사결정 트리 모델을 생성하였다. 모델 성능을 평가하기 위해 LIFT 와 Hit ratio 을 사용하였다. Hit ratio is defined as A/(A+B), instead of (A+D)/(A+B+C+D). LIFT 는 상위 몇%의 비율을 살펴보는 것이다.

loyalty, contribution, and usage, we selected bill amount, tenure, MOU (outbound call usage), MTU (inbound call usage), and payment rate 을 변수로 선택하고 K-Means 를 사용하여 고객을 5 개 클러스터로 모델링 했다.

EDA 에서 C5.0 의사 결정 트리 모델링을 위해 약 40 개의 변수를 선택했다. LIFT (10 %)를 기반으로 한 모델 성과, 즉 10 % 가입자의 LIFT 를 최고 변동폭 점수와 비교했다.
BPN: 43 개의 입력과 하나의 출력 만 포함했다. 숨겨진 레이어의 학습 속도 나 뉴런 수와 같은 주요 모델링 매개 변수에 대한 공개 정보가 없기 때문에 다양한 조합을 시도한다. 히든레이어에서 18 개의 노드을 사용하는 결과를 보여준다.

일반적으로 NN 이 DT 보다 성능이 좋았다.

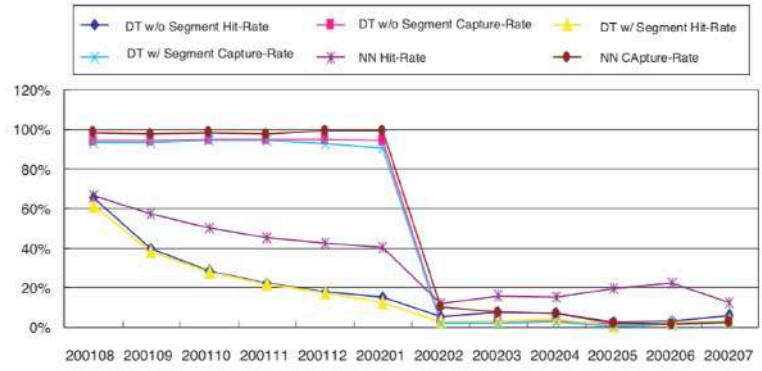


Fig. 5. Hit ratio and capture rate of different models.

>비교

이 연구와 비교했을 때, 유사점과 차이점의 두 부분으로 나눌 수 있다. 먼저 이 논문에서는 적합도로 Hit ratio is defined as A/(A+B), instead of (A+D)/(A+B+C+D) 를 사용했는데, 이는 내가 분석하면서 진행했을 때 Accuracy 를 중요하게 생각하는 것이 아닌 TPR 을 중점적으로 보았던 것과 유사하다.

차이를 말하자면 이 논문의 경우 신경망의 경우 의사결정나무보다 뛰어난 성능을 보였으나, 내 경우 TPR 만을 보았을 때 의사결정나무가 더 훌륭한 것을 알 수 있었다. 이는 데이터 셋의 차이 때문일 수도 있다고 보인다. 또한 Bayes Classifer 의 경우 예전에는 잘 사용되지 않았는데 최근 GPU 의 성능이 좋아지며 각광받고 있다. 독립을 가정한 Naive Bayes 를 사용하여 높은 성능을 얻을 수 있었다.

V 결론

이동 통신 시장 자유화에 있어 예측 및 관리가 중요하다. 시장에서 경쟁력을 갖추려면 이동 통신 서비스 제공 업체가 가능한 고객을 예측하고 소중한 고객을 보유하기 위해 사전 대책을 강구해야 한다.
보고서를 작성하면서 텔레콤 변동 예측을 위한 예측 모델을 구축하는 다양한 기법을 제안했다. 모델 구축에 부적절한 데이터의 영향을 조사하였다. 이 과정을 통해 Naive Bayes Classifier 가 데이터에 적합한 모델이라는 것을 알았고, 우리가 무엇보다 관심있는 이탈고객에 대한 예측을 잘 할 수 있어 선호하였다.

효과적인 이탈 예측 모델은 회사가 어느 고객이 떠나려고 하는지를 알 수 있도록 지원한다. 모바일 서비스 제공 업체는 고객 만족을 위한 좋은 회원 유지 프로그램을 개발해야 한다. 또한 고객 이탈률 점수를 고객 세그먼트와 통합하고 고객 가치를 적용하는 것은 이동 통신 서비스 제공 업체가 귀중한 고객을 보유 할 수 있는 올바른 전략을 수립하는 데 도움이 될 것이다.

데이터 마이닝 기술은 신용 카드 사기 탐지, 신용 점수, churners 및 보존 프로그램 간의 유사성, 응답 모델링 및 고객 구매 결정 모델링과 같은 많은 CRM 필드에 적용될 수 있다. 보고서를 통해 CRM 필드에 데이터마이닝 기법이 적용된다는 것을 알았으며, 후에 고객 이탈에 어떤 변수들이 영향을 줄 수 있고 각각이 무엇을 의미하는지 더 깊게 공부하고 싶은 생각이 드는 시간이었다.