

타이타닉

1. 필요한 Library 호출

```
library('ggplot2')
library('caret')
library('dplyr')
library('randomForest')
library('rpart')
library('rpart.plot')
library('car')
library('e1071')
```

2. 데이터 입력

```
train.tit <- read.csv('train.csv', stringsAsFactors = F)
test.tit  <- read.csv('test.csv', stringsAsFactors = F)
test.tit$Survived <- NA

## train 와 test 데이터 결합
full_titanic <- rbind(train.tit, test.tit)

## 데이터 구조 확인
str(full_titanic)

## 'data.frame': 1309 obs. of 12 variables:
## $ PassengerId: int 1 2 3 4 5 6 7 8 9 10 ...
## $ Survived : int 0 1 1 1 0 0 0 0 1 1 ...
## $ Pclass : int 3 1 3 1 3 3 1 3 3 2 ...
## $ Name : chr "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley
(Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques H
eath (Lily May Peel)" ...
## $ Sex : chr "male" "female" "female" "female" ...
## $ Age : num 22 38 26 35 35 NA 54 2 27 14 ...
## $ SibSp : int 1 1 0 1 0 0 0 3 0 1 ...
## $ Parch : int 0 0 0 0 0 0 0 1 2 0 ...
```

```
## $ Ticket      : chr "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
## $ Fare        : num 7.25 71.28 7.92 53.1 8.05 ...
## $ Cabin       : chr "" "C85" "" "C123" ...
## $ Embarked    : chr "S" "C" "S" "S" ...
```

2. 데이터 분석 및 결측치 확인

결측치 확인

```
colSums(is.na(full_titanic))
```

```
## PassengerId   Survived    Pclass      Name      Sex      Age
##           0         418         0         0         0      263
##      SibSp     Parch     Ticket     Fare     Cabin Embarked
##           0           0           0         1         0         0
```

결측치 데이터

```
colSums(full_titanic=="")
```

```
## PassengerId   Survived    Pclass      Name      Sex      Age
##           0         NA         0         0         0      NA
##      SibSp     Parch     Ticket     Fare     Cabin Embarked
##           0           0           0         NA     1014         2
```

Age 결측치: 263, Cabin 결측치: 1014, Embarked 결측치: 2

Embarked 의 결측치를 최빈값으로 입력한다.

```
table(full_titanic$Embarked)
```

```
##
##      C      Q      S
## 2 270 123 914
```

```
full_titanic$Embarked[full_titanic$Embarked==""] = "S"
```

```
table(full_titanic$Embarked)
```

```
##
##      C      Q      S
## 270 123 916
```

Age 와 Cabin 은 결측치 많아, 분석과정 동안 결측값들을 확인 할 것이다.

Factor 로 변환 가능한 변수 확인

```
apply(full_titanic,2, function(x) length(unique(x)))
```

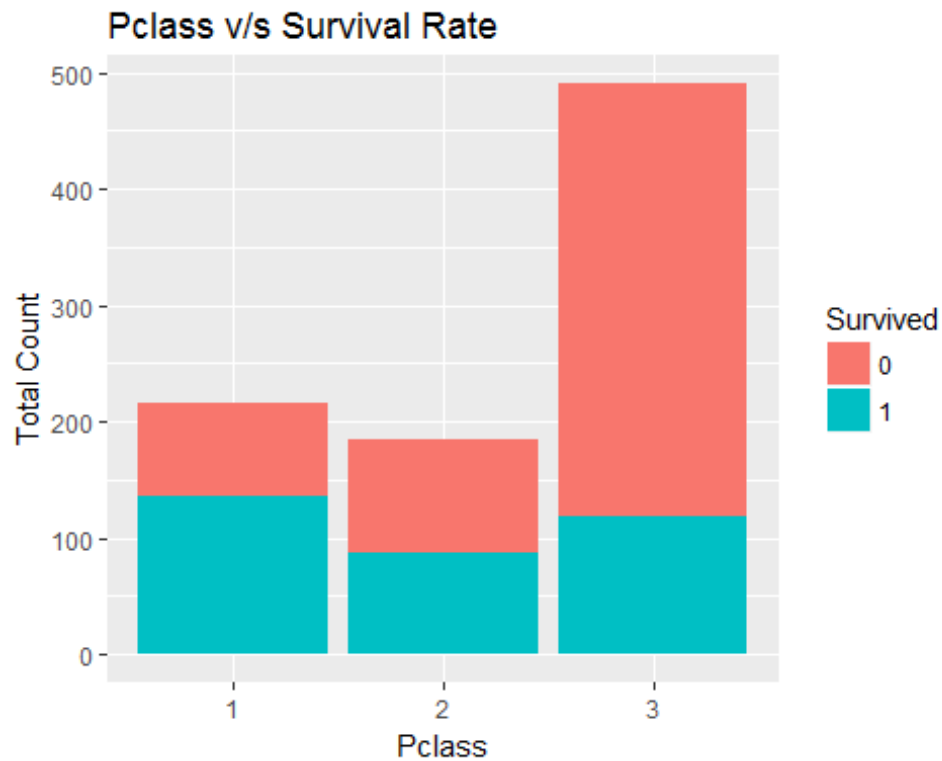
```
## PassengerId    Survived    Pclass      Name      Sex      Age
##      1309         3         3      1307      2      99
##      SibSp      Parch      Ticket    Fare      Cabin Embarked
##         7         8         929     282      187      3
```

위 변수들 중 “Survived”, “Pclass”, “Sex”, “Embarked”를 Factor 로 변환한다.

```
cols=c("Survived","Pclass","Sex","Embarked")
for (i in cols){
  full_titanic[,i]=as.factor(full_titanic[,i])
}
```

타이타닉호 에서 생존자 중 부유층이 가난한 사람들보다 더 많이 살아 남았다. 클래스별 생존자 비율을 그래프로 확인한다.

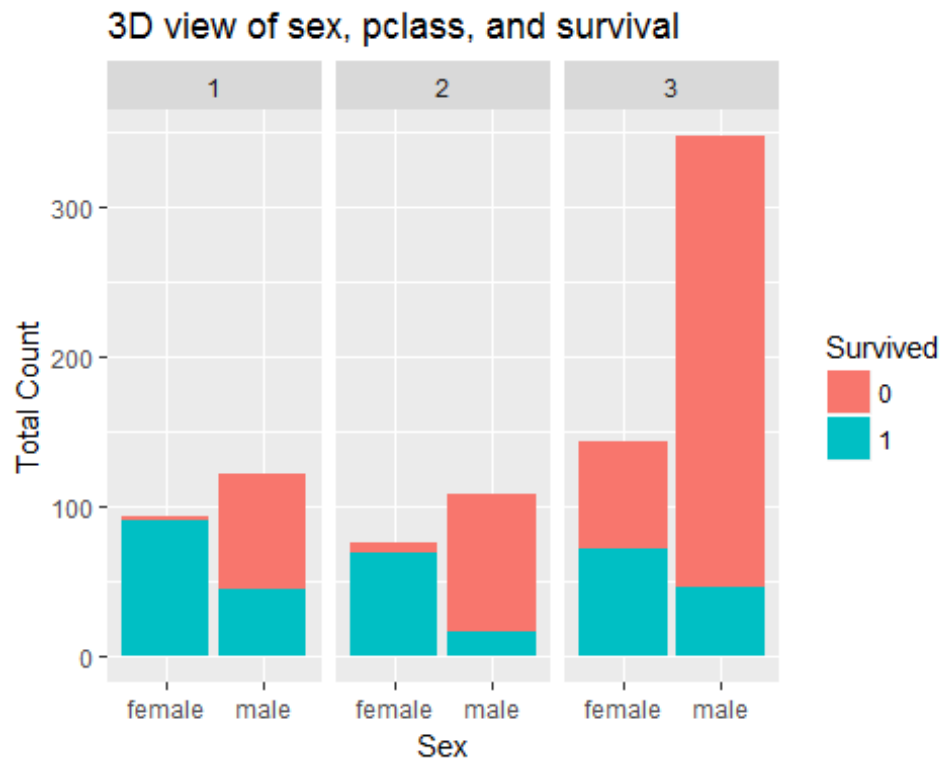
```
ggplot(full_titanic[1:891,],aes(x = Pclass,fill=factor(Survived))) +
geom_bar() +
ggtitle("Pclass v/s Survival Rate")+
xlab("Pclass") +
ylab("Total Count") +
labs(fill = "Survived")
```



그래프를 보면 1 번째 클래스에 있는 사람들이 3 번째 클래스에 있는 사람들 보다 생존율이 더 높다

Sex, Pclass, Survived 에 대한 관계를 그래프로 확인한다

```
ggplot(full_titanic[1:891,], aes(x = Sex, fill = Survived)) +  
  geom_bar() +  
  facet_wrap(~Pclass) +  
  ggtitle("3D view of sex, pclass, and survival") +  
  xlab("Sex") +  
  ylab("Total Count") +  
  labs(fill = "Survived")
```



Name 컬럼에서 생존율과의 관계를 확인한다.

```
head(full_titanic$Name)
```

```
## [1] "Braund, Mr. Owen Harris"
## [2] "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
## [3] "Heikkinen, Miss. Laina"
## [4] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
## [5] "Allen, Mr. William Henry"
## [6] "Moran, Mr. James"
```

정규식을 이용해 Name 에서 title 을 추출한다.

```
names <- full_titanic$Name
title <- gsub("^.*, (.*)\\..*$", "\\1", names)
```

```
full_titanic$title <- title
```

```
table(title)
```

```
## title
##      Capt      Col      Don      Dona      Dr
##         1         4         1         1         8
##  Jonkheer   Lady   Major   Master   Miss
##         1         1         2        61      260
##      Mlle      Mme      Mr      Mrs      Ms
```

```
##           2           1           757           197           2
##           Rev           Sir the Countess
##           8           1           1
```

MISS, Mrs, Master, Mr 의 숫자가 많다. 생존율의 오버피팅을 막기 위해 여러 개의 다른 타이틀을 큰 그룹으로 합친다.

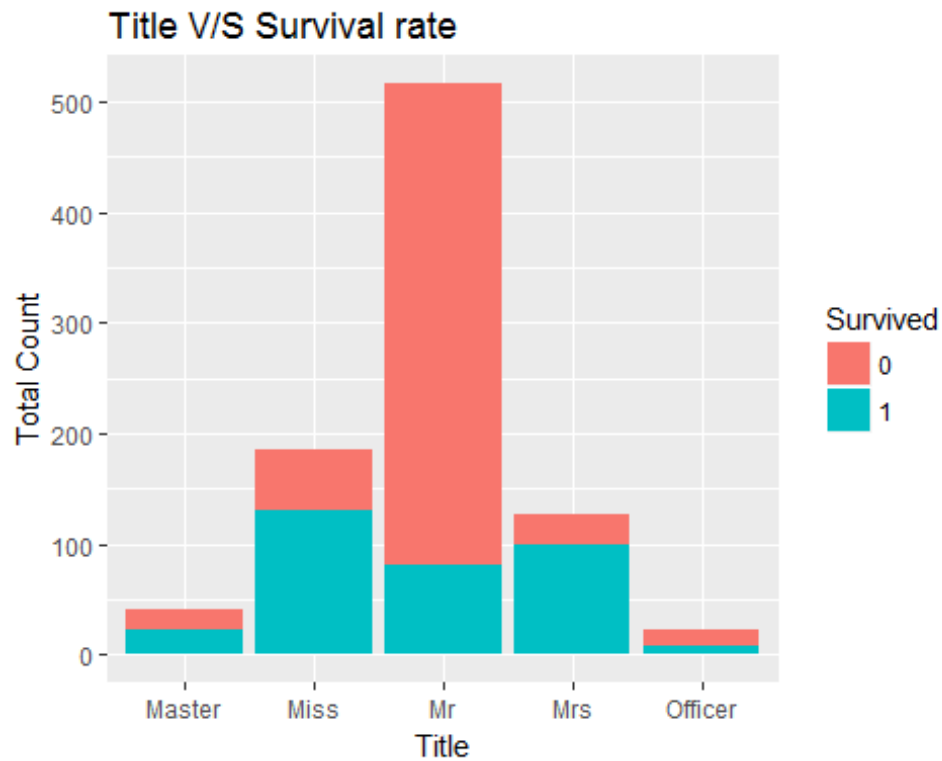
```
full_titanic$title[full_titanic$title == 'Mlle'] <- 'Miss'
full_titanic$title[full_titanic$title == 'Ms'] <- 'Miss'
full_titanic$title[full_titanic$title == 'Mme'] <- 'Mrs'
full_titanic$title[full_titanic$title == 'Lady'] <- 'Miss'
full_titanic$title[full_titanic$title == 'Dona'] <- 'Miss'
```

작은 데이터를 새로운 변수로 생성하는 것에 대한 우려가 있어, 기존의 변수에 합치는 것으로 결정한다. 군인, 의사, 일반인 등을 아래 분류로 합친다.

```
full_titanic$title[full_titanic$title == 'Capt'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Col'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Major'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Dr'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Rev'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Don'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Sir'] <- 'Officer'
full_titanic$title[full_titanic$title == 'the Countess'] <- 'Officer'
full_titanic$title[full_titanic$title == 'Jonkheer'] <- 'Officer'
```

높은 생존율을 갖는 title 에대한 점검

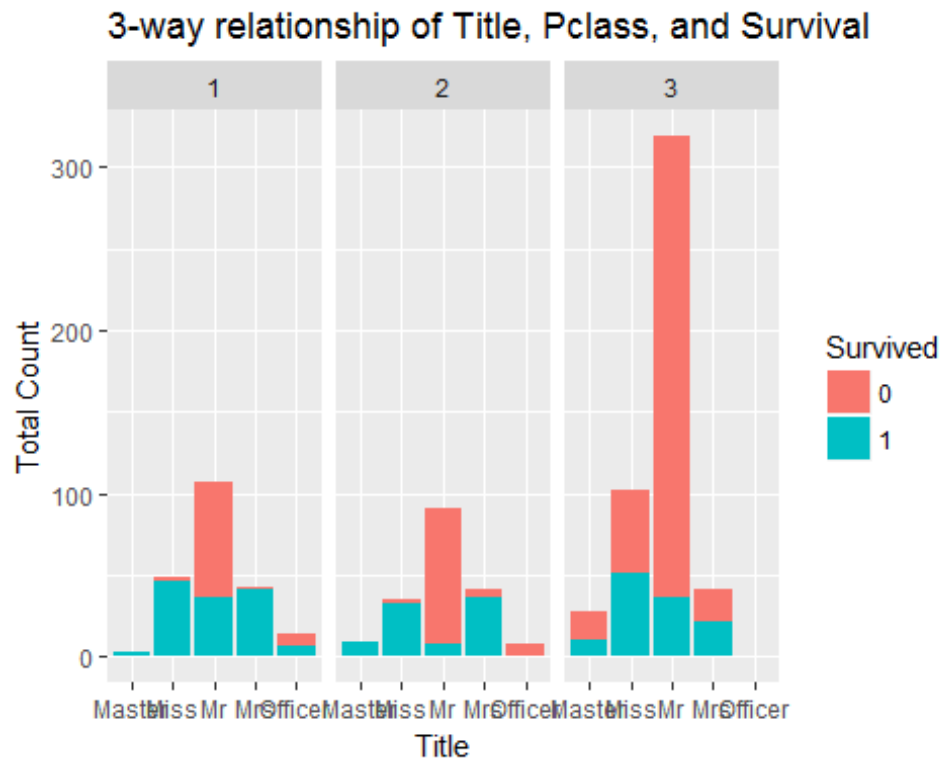
```
ggplot(full_titanic[1:891,], aes(x = title, fill=factor(Survived))) +
  geom_bar() +
  ggtitle("Title V/S Survival rate")+
  xlab("Title") +
  ylab("Total Count") +
  labs(fill = "Survived")
```



위의 그래프를 보면 Mr 는 아주 낮은 생존율을 갖는다. 반면, Miss 와 Mrs 는 Master 와 Officer 보다 높은 생존율을 갖는다.

Title, Pclass, Survival 3 변수에 대한 시각화

```
ggplot(full_titanic[1:891,], aes(x = title, fill = Survived)) +  
  geom_bar() +  
  facet_wrap(~Pclass) +  
  ggtitle("3-way relationship of Title, Pclass, and Survival") +  
  xlab("Title") +  
  ylab("Total Count") +  
  labs(fill = "Survived")
```



```

### Sibsp 와 Parch 를 이용해 Family Size 생성하기

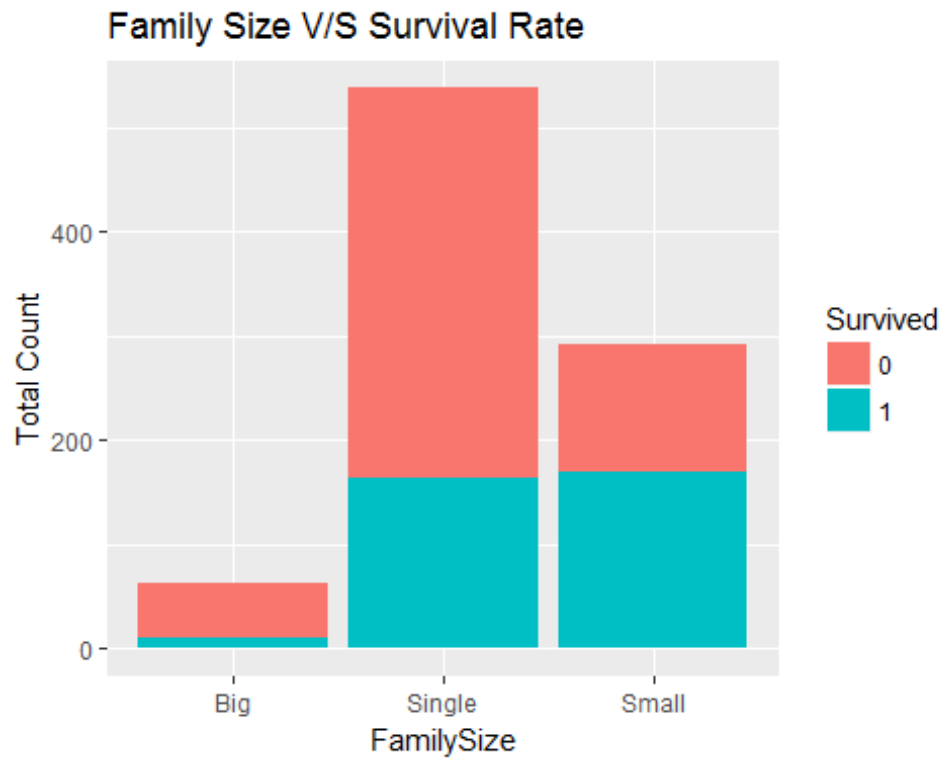
full_titanic$FamilySize <- full_titanic$SibSp + full_titanic$Parch + 1

full_titanic$FamilySized[full_titanic$FamilySize == 1] <- 'Single'
full_titanic$FamilySized[full_titanic$FamilySize < 5 & full_titanic$FamilySize >= 2] <- 'Small'
full_titanic$FamilySized[full_titanic$FamilySize >= 5] <- 'Big'

full_titanic$FamilySized=as.factor(full_titanic$FamilySized)

### Family Size 로 생존율 시각화
ggplot(full_titanic[1:891,], aes(x = FamilySized, fill=factor(Survived))) +
  geom_bar() +
  ggtitle("Family Size V/S Survival Rate") +
  xlab("FamilySize") +
  ylab("Total Count") +
  labs(fill = "Survived")

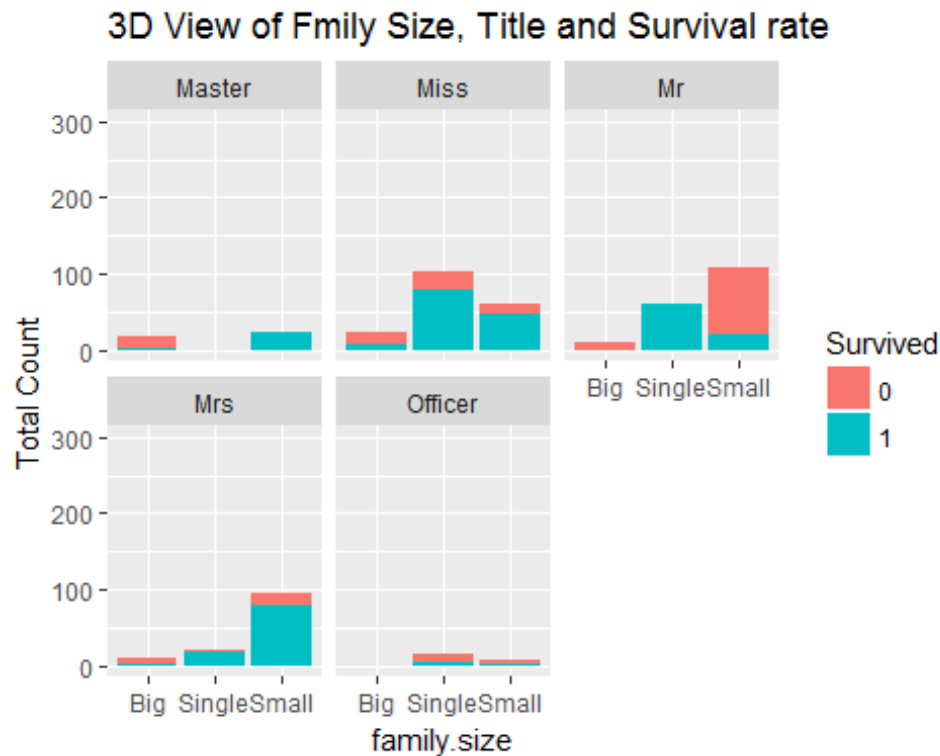
```

타이타닉에서 많은 가족수를 갖고 있는 사람들은 혼자이거나 작은 가족수를 갖는 사람들보다 생존율이 낮다.

시각화를 통해 대가족의 생존율을 확인한다.

```
ggplot(full_titanic[1:891,], aes(x = FamilySize, fill = Survived)) +
  geom_bar() +
  facet_wrap(~title) +
  ggtitle("3D View of Family Size, Title and Survival rate") +
  xlab("family.size") +
  ylab("Total Count") +
  ylim(0,300) +
  labs(fill = "Survived")
```



Family Size, Title, Survived 에 대한 그래프이다. 대가족인 경우 생존율이 보다 작은 가족 구성원 보다 훨씬 낮은 것을 알 수 있다.

동일한 Ticket 을 가진 승객을 기반으로 한 분석

```
ticket.unique <- rep(0, nrow(full_titanic))
tickets <- unique(full_titanic$Ticket)

for (i in 1:length(tickets)) {
  current.ticket <- tickets[i]
  party.indexes <- which(full_titanic$Ticket == current.ticket)

  for (k in 1:length(party.indexes)) {
    ticket.unique[party.indexes[k]] <- length(party.indexes)
  }
}

full_titanic$ticket.unique <- ticket.unique

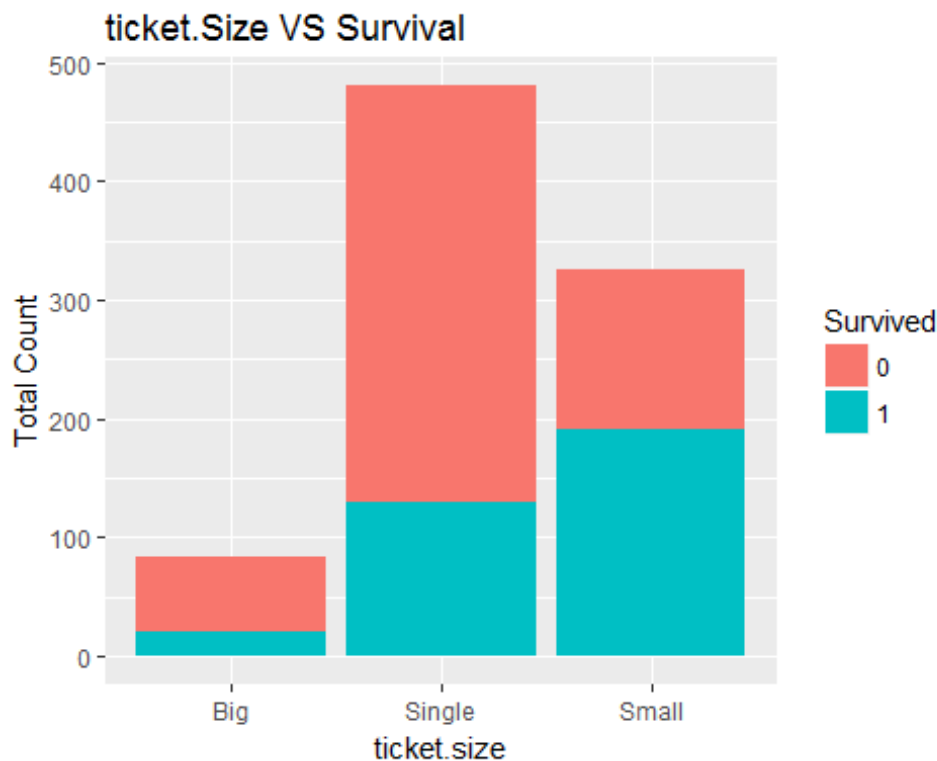
full_titanic$ticket.size[full_titanic$ticket.unique == 1] <- 'Single'
full_titanic$ticket.size[full_titanic$ticket.unique < 5 & full_titanic$ticket.
```

```

unique >= 2] <- 'Small'
full_titanic$ticket.size[full_titanic$ticket.unique >= 5] <- 'Big'

## 그래프를 통해 Ticket 사이즈를 확인한다.
ggplot(full_titanic[1:891,], aes(x = ticket.size, fill = factor(Survived))) +
  geom_bar() +
  ggtitle("ticket.Size VS Survival")+
  xlab("ticket.size") +
  ylab("Total Count") +
  labs(fill = "Survived")

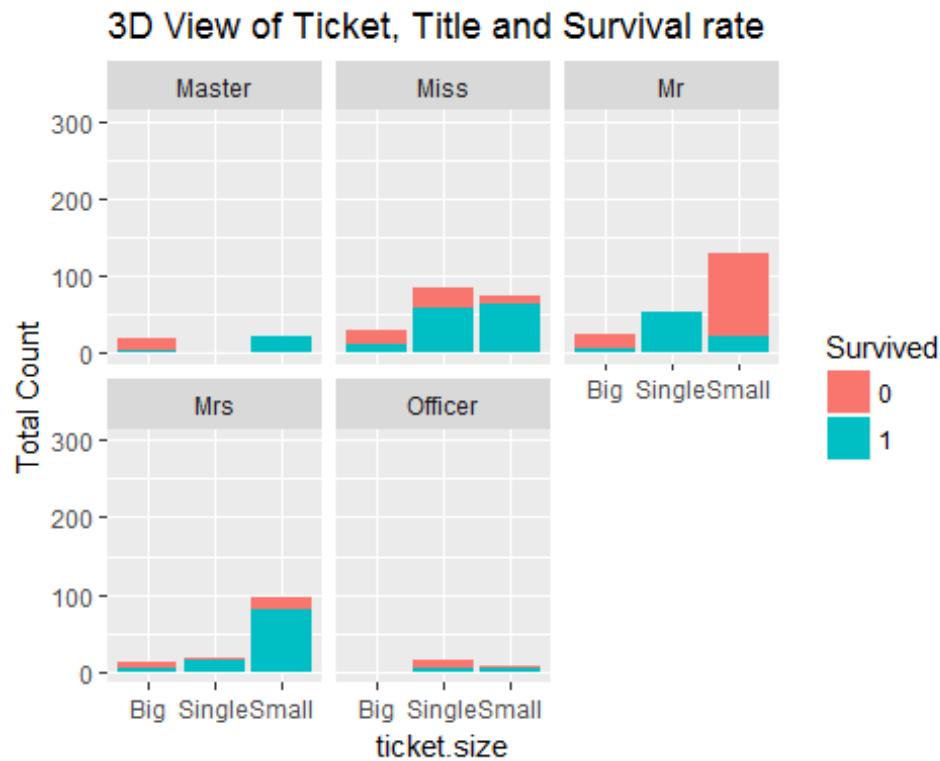
```



```

## 그래프를 통해 Ticket 와 Title 의 규모를 확인한다.
ggplot(full_titanic[1:891,], aes(x = ticket.size, fill = Survived)) +
  geom_bar() +
  facet_wrap(~title) +
  ggtitle("3D View of Ticket, Title and Survival rate") +
  xlab("ticket.size") +
  ylab("Total Count") +
  ylim(0,300) +
  labs(fill = "Survived")

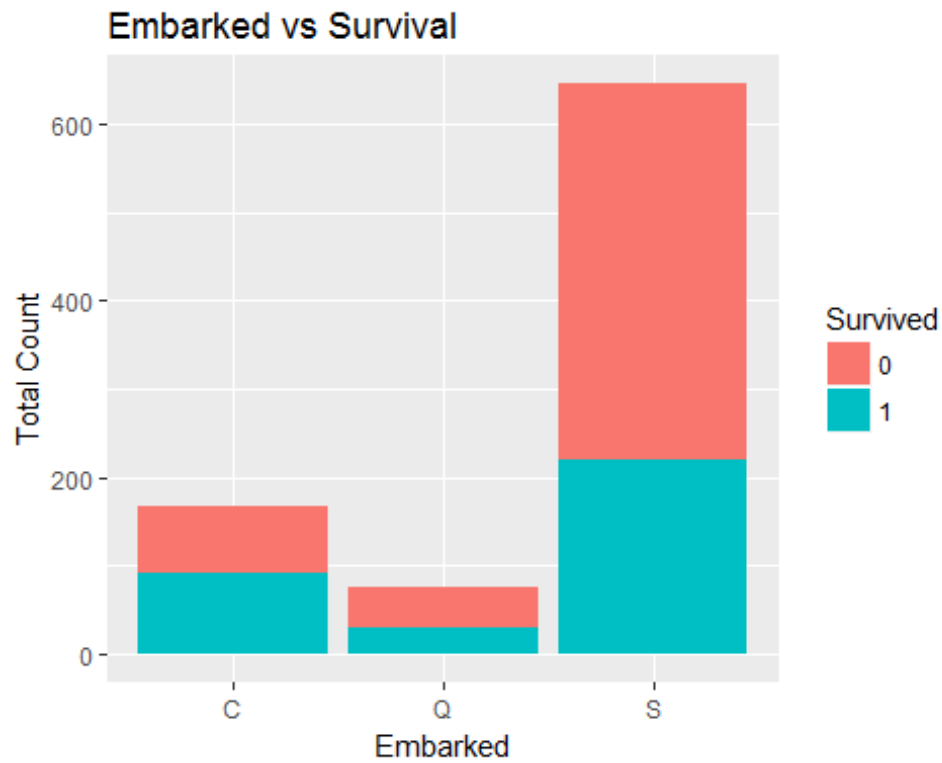
```



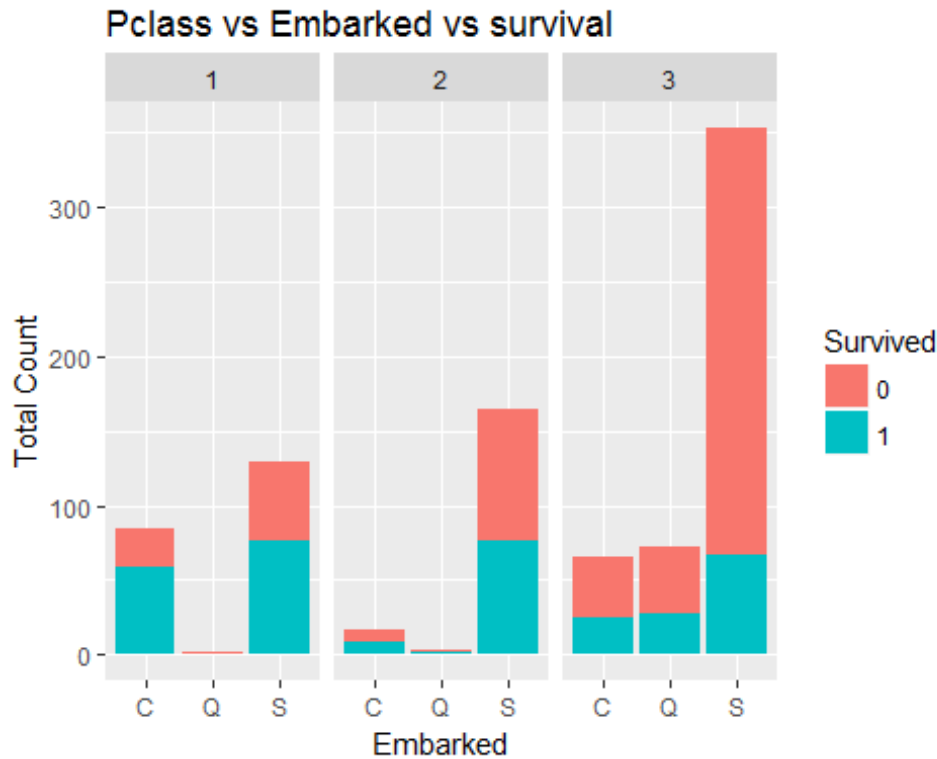
Family Size 와 Ticket Size 에 대한 큰 차이점은 없다.

승선 위치에 따른 생존율에 관계를 확인한다.

```
ggplot(full_titanic[1:891,], aes(x = Embarked, fill=factor(Survived))) +
  geom_bar() +
  ggtitle("Embarked vs Survival") +
  xlab("Embarked") +
  ylab("Total Count") +
  labs(fill = "Survived")
```



```
## Pclass 별 Embarked 위치를 나눈 시각화
ggplot(full_titanic[1:891,], aes(x = Embarked, fill = Survived)) +
  geom_bar() +
  facet_wrap(~Pclass) +
  ggtitle("Pclass vs Embarked vs survival") +
  xlab("Embarked") +
  ylab("Total Count") +
  labs(fill = "Survived")
```



Embarked 와 Survived 와의 관계는 별로 없어 보인다.

Cabin 에는 결측값들이 많아 사용하기 좋지 않아 보인다.

Age 를 대신해 Title 을 사용한다.

그리고, Fare 와 Pclass 와는 관계가 높음으로 분석대상으로 선정한다.

```
full_titanic$ticket.size <- as.factor(full_titanic$ticket.size)
full_titanic$title <- as.factor(full_titanic$title)
```

3. 변수 선정

지금까지 분석한 내용을 바탕으로 모델에 사용할 변수를 정리한다.

```
## "Pclass", "title", "Sex", "Embarked", "FamilySize", "ticket.size"
```

중복된 변수들은 삭제하고 적당한 형식으로 데이터를 변경한다.

```
feature1 <- full_titanic[1:891, c("Pclass", "title", "Sex", "Embarked", "FamilySize", "ticket.size")]
```

```

response <- as.factor(train.tit$Survived)
feauter1$Survived=as.factor(train.tit$Survived)

### 데이터 검증을 위해 원본 데이터에서 20%를 사용하지 않고 남겨둔다.

set.seed(500)
ind=createDataPartition(feauter1$Survived,times=1,p=0.8,list=FALSE)
train_val=feauter1[ind,]
test_val=feauter1[-ind,]

#### 원본 training 데이터, train 와 test 데이터의 생존율 비율을 확인한다.

round(prop.table(table(train.tit$Survived)*100),digits = 1)

##
##    0    1
## 0.6 0.4

round(prop.table(table(train_val$Survived)*100),digits = 1)

##
##    0    1
## 0.6 0.4

round(prop.table(table(test_val$Survived)*100),digits = 1)

##
##    0    1
## 0.6 0.4

```

4. 모델생성 및 예측

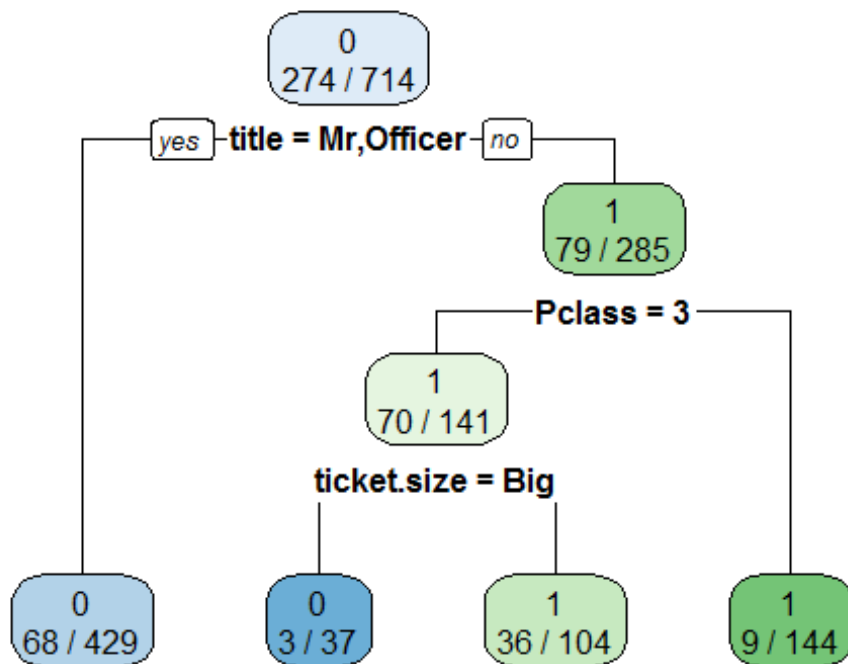
의사결정 트리 (Decision tree)

```

set.seed(1234)
Model_DT=rpart(Survived~.,data=train_val,method="class")

rpart.plot(Model_DT,extra = 3,fallen.leaves = T)

```



Plot 내용을 보면, Single tee 모델에서 Title, Pclass, Ticket.size 변수를 사용하고 있다. 그외 변수들은 사용하지 않는다.

train 데이터를 Predict 하고 single tree 의 정확성을 확인한다.

```
PRE_TDT=predict(Model_DT,data=train_val,type="class")
confusionMatrix(PRE_TDT,train_val$Survived)
```

Confusion Matrix and Statistics

##

Reference

Prediction 0 1

0 395 71

1 45 203

##

Accuracy : 0.8375

95% CI : (0.8084, 0.8639)

No Information Rate : 0.6162

P-Value [Acc > NIR] : < 2e-16

##

Kappa : 0.6502

McNemar's Test P-Value : 0.02028

##

Sensitivity : 0.8977

Specificity : 0.7409


```
##          Pos Pred Value : 0.8476
##          Neg Pred Value : 0.8185
##          Prevalence : 0.6162
##          Detection Rate : 0.5532
##          Detection Prevalence : 0.6527
##          Balanced Accuracy : 0.8193
##
##          'Positive' Class : 0
##
```

Accuracy 가 0.8375 이다.

그리 나쁜 결과는 아니다. 3 개의 변수만 사용했다.

Single tree 는 Overfitting 이 발생할 수 있어, '10 fold 알고리즘'을 사용해 검증한다.

```
set.seed(1234)
cv.10 <- createMultiFolds(train_val$Survived, k = 10, times = 10)

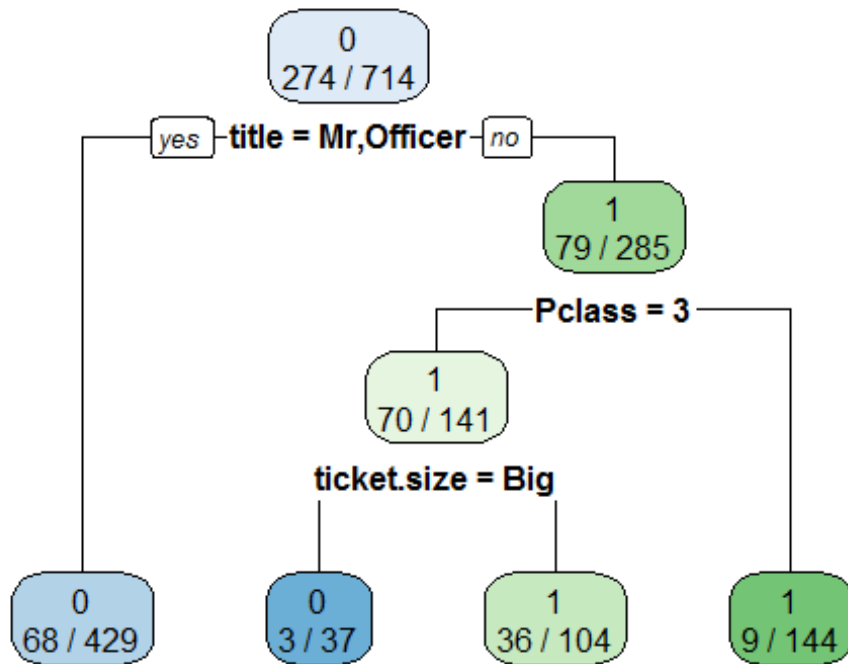
# 조절
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
                     index = cv.10)

## 모델생성
Model_CDT <- train(x = train_val[,-7], y = train_val[,7], method = "rpart", tuneLength = 30, trControl = ctrl)
```

Accuracy 가 0.8139 이다. 앞의 Single tree 에서 Overfitting 된 것으로 보인다.

중요 변수를 확인한다. Single tree 와 동일한 결과인가?

```
rpart.plot(Model_CDT$finalModel,extra = 3,fallen.leaves = T)
```



사용된 변수에는 차이가 없다.

test 데이터로 정확성을 확인한다.

```
PRE_VDTS=predict(Model_CDT$finalModel,newdata=test_val,type="class")
```

```
confusionMatrix(PRE_VDTS,test_val$Survived)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  0  1
```

```
##           0 97 20
```

```
##           1 12 48
```

```
##
```

```
##           Accuracy : 0.8192
```

```
##           95% CI : (0.7545, 0.8729)
```

```
##           No Information Rate : 0.6158
```

```
##           P-Value [Acc > NIR] : 3.784e-09
```

```
##
```

```
##           Kappa : 0.6093
```

```
##           Mcnemar's Test P-Value : 0.2159
```

```
##
```

```
##           Sensitivity : 0.8899
```

```
##           Specificity : 0.7059
```

```
##           Pos Pred Value : 0.8291
```

```
##           Neg Pred Value : 0.8000
```

```
##           Prevalence : 0.6158
##           Detection Rate : 0.5480
##      Detection Prevalence : 0.6610
##           Balanced Accuracy : 0.7979
##
##           'Positive' Class : 0
##
```

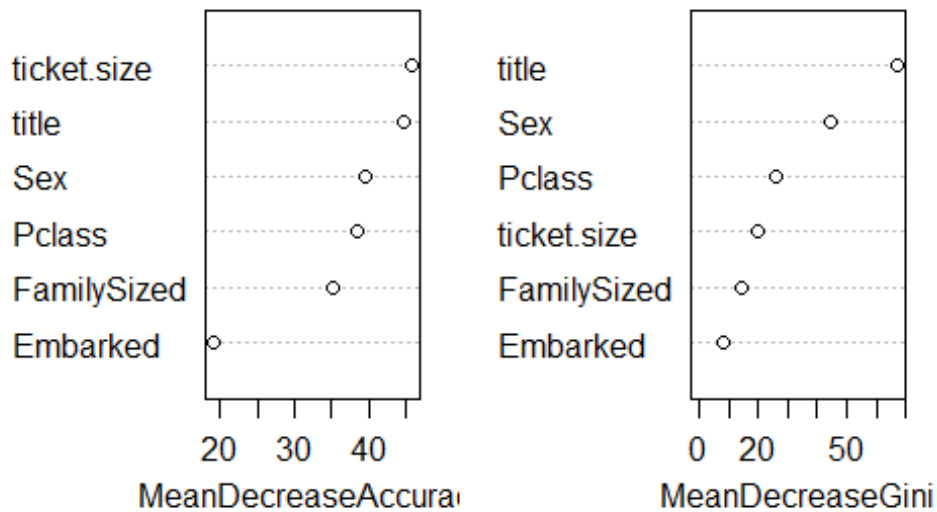
Accuracy 가 0.8192 이다.

랜덤포레스트 (Random Forest)

```
#install.packages("randomForest")
library(randomForest)
set.seed(1234)
rf.1 <- randomForest(x = train_val[,-7],y=train_val[,7], importance = TRUE, n
tree = 1000)
rf.1

##
## Call:
## randomForest(x = train_val[, -7], y = train_val[, 7], ntree = 1000,
importance = TRUE)
##           Type of random forest: classification
##           Number of trees: 1000
## No. of variables tried at each split: 2
##
##           OOB estimate of  error rate: 17.09%
## Confusion matrix:
##      0   1 class.error
## 0 395  45  0.1022727
## 1  77 197  0.2810219
varImpPlot(rf.1)
```

rf.1



랜덤포레스트 accuracy rate 는 82.91 이다. 의사결정 트리 보다 1% 더 낫다.

중복 변수 2 개를 삭제하고 모델을 재생성한다.

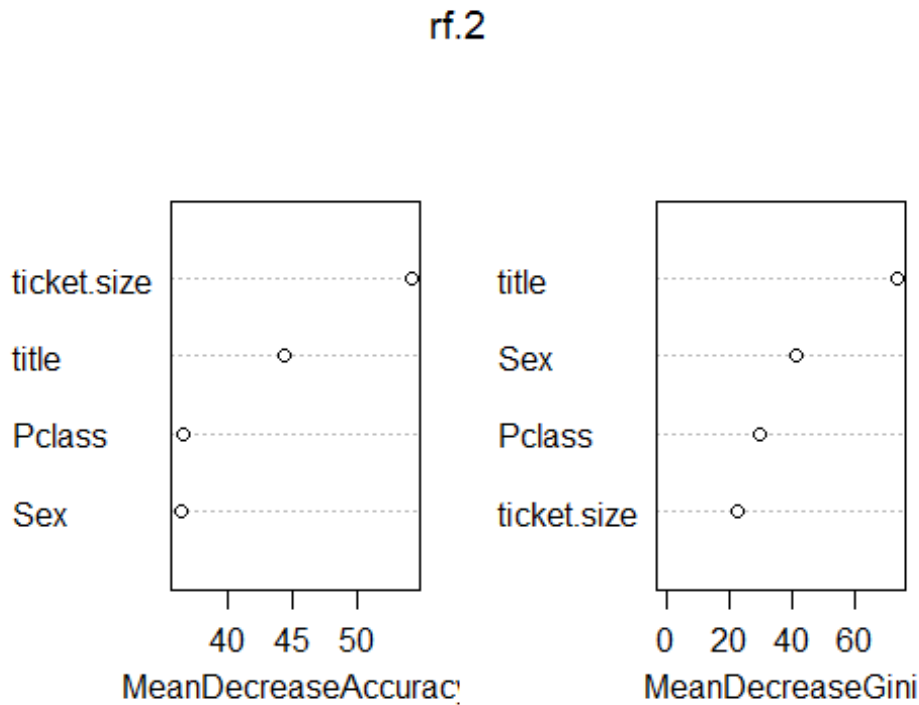
```
train_val1=train_val[,-4:-5]
test_val1=test_val[,-4:-5]

set.seed(1234)
rf.2 <- randomForest(x = train_val1[,-5],y=train_val1[,5], importance = TRUE,
  ntree = 1000)
rf.2

##
## Call:
## randomForest(x = train_val1[, -5], y = train_val1[, 5], ntree = 1000,
  importance = TRUE)
##
##           Type of random forest: classification
##           Number of trees: 1000
## No. of variables tried at each split: 2
##
##           OOB estimate of  error rate: 15.97%
## Confusion matrix:
##      0    1 class.error
```

```
## 0 395 45 0.1022727
## 1 69 205 0.2518248
```

```
varImpPlot(rf.2)
```



2 변수 삭제로 인해, accuracy 가 84.03 으로 증가했다.

교차 검증을 한 후 모델을 선정한다.

```
library(caret)
set.seed(2348)
cv10_1 <- createMultiFolds(train_val1[,5], k = 10, times = 10)

# caret 의 trainControl 를 구성한다.
ctrl_1 <- trainControl(method = "repeatedcv", number = 10, repeats = 10,
                        index = cv10_1)

set.seed(1234)
rf.5<- train(x = train_val1[,-5], y = train_val1[,5], method = "rf", tuneLength = 3,
             ntree = 1000, trControl =ctrl_1)

rf.5
```

```
## Random Forest
##
## 714 samples
## 4 predictor
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 642, 643, 642, 643, 643, 643, ...
## Resampling results across tuning parameters:
##
## mtry Accuracy Kappa
## 2 0.8392390 0.6538299
## 3 0.8382551 0.6518851
## 4 0.8368466 0.6489636
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

교차 검증결과 accuracy rate 는 0.8393 이다.

Test data 를 예측한다.

```
pr.rf=predict(rf.5,newdata = test_val1)

confusionMatrix(pr.rf,test_val1$Survived)

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 97 20
##           1 12 48
##
##              Accuracy : 0.8192
##              95% CI : (0.7545, 0.8729)
##      No Information Rate : 0.6158
##      P-Value [Acc > NIR] : 3.784e-09
##
##              Kappa : 0.6093
##  Mcnemar's Test P-Value : 0.2159
##
##              Sensitivity : 0.8899
##              Specificity : 0.7059
##      Pos Pred Value : 0.8291
##      Neg Pred Value : 0.8000
##              Prevalence : 0.6158
##      Detection Rate : 0.5480
```

```
##      Detection Prevalence : 0.6610
##      Balanced Accuracy : 0.7979
##
##      'Positive' Class : 0
##
```

accuracy rate 는 0.8192 로 기대 보다 낮았다.

서포트 벡터 머신(Support Vector Machine)

```
library(e1071)
set.seed(1274)
liner.tune=tune.svm(Survived~.,data=train_val1,kernel="linear",cost=c(0.01,0.
1,0.2,0.5,0.7,1,2,3,5,10,15,20,50,100))
```

```
liner.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##     3
##
## - best performance: 0.1736502
```

Cost=3 일때, 최고의 성능을 낸다. Accuracy rate 가 82.7 이다.

```
### 최상의 liner 모델을 생성한다.
best.linear=liner.tune$best.model
```

```
## test data 를 이용해 Survival rate 을 예측한다.
```

```
best.test=predict(best.linear,newdata=test_val1,type="class")
confusionMatrix(best.test,test_val1$Survived)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  0  1
```

```
##          0  97  21
##          1  12  47
##
##              Accuracy : 0.8136
##              95% CI : (0.7483, 0.8681)
##      No Information Rate : 0.6158
##      P-Value [Acc > NIR] : 1.058e-08
##
##              Kappa : 0.5959
##  McNemar's Test P-Value : 0.1637
##
##              Sensitivity : 0.8899
##              Specificity : 0.6912
##      Pos Pred Value : 0.8220
##      Neg Pred Value : 0.7966
##      Prevalence : 0.6158
##      Detection Rate : 0.5480
##      Detection Prevalence : 0.6667
##      Balanced Accuracy : 0.7905
##
##      'Positive' Class : 0
##
```

Linear 모델 accuracy 는 0.8136 이다.

Radial Support vector Machine

```
set.seed(1274)

rd.poly=tune.svm(Survived~.,data=train_val1,kernel="radial",gamma=seq(0.1,5))

summary(rd.poly)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   gamma
##     2.1
##
## - best performance: 0.166608
##
## - Detailed performance results:
```



```
##      gamma      error dispersion
## 1    0.1 0.1680164 0.04245604
## 2    1.1 0.1680164 0.03983673
## 3    2.1 0.1666080 0.04166448
## 4    3.1 0.1666080 0.04166448
## 5    4.1 0.1666080 0.04166448
```

```
best.rd=rd.poly$best.model
```

Non Linear Kerenel 은 높은 accuracy 를 제공한다.

```
## test data 예측하기
pre.rd=predict(best.rd,newdata = test_val1)
```

```
confusionMatrix(pre.rd,test_val1$Survived)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 97 20
##           1 12 48
##
##              Accuracy : 0.8192
##              95% CI : (0.7545, 0.8729)
##      No Information Rate : 0.6158
##      P-Value [Acc > NIR] : 3.784e-09
##
##              Kappa : 0.6093
##  Mcnemar's Test P-Value : 0.2159
##
##              Sensitivity : 0.8899
##              Specificity : 0.7059
##              Pos Pred Value : 0.8291
##              Neg Pred Value : 0.8000
##              Prevalence : 0.6158
##              Detection Rate : 0.5480
##      Detection Prevalence : 0.6610
##              Balanced Accuracy : 0.7979
##
##              'Positive' Class : 0
##
```

Non Liner 모델은 사용한 test data 의 accuracy 는 0.81 이다.

로지스틱 회귀모델 (Logistic Regression)

```
contrasts(train_val1$Sex)
```

```
##           male
## female      0
## male        1
```

```
contrasts(train_val1$Pclass)
```

```
##    2 3
## 1 0 0
## 2 1 0
## 3 0 1
```

변수가 어떻게 구성되어 있는지 확인한다.

로지스틱 회귀 모델을 실행한다.

```
log.mod <- glm(Survived ~ ., family = binomial(link=logit),
               data = train_val1)
```

summary 확인

```
summary(log.mod)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = logit),
##      data = train_val1)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4187  -0.5944  -0.3937   0.5805   3.0414
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    16.8752    624.0921   0.027  0.978428
## Pclass2         -1.1968     0.3129  -3.824  0.000131 ***
## Pclass3         -2.1324     0.2721  -7.838  4.58e-15 ***
## titleMiss      -16.1021    624.0921  -0.026  0.979416
## titleMr         -3.7422     0.5216  -7.175  7.24e-13 ***
## titleMrs       -16.0186    624.0921  -0.026  0.979523
## titleOfficer    -4.3752     0.8595  -5.090  3.58e-07 ***
## Sexmale        -15.6157    624.0919  -0.025  0.980038
## ticket.sizeSingle  2.0968     0.4082   5.137  2.79e-07 ***
## ticket.sizeSmall  2.0356     0.3870   5.260  1.44e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[illegible]

[illegible]

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##              2.5 %      97.5 %  
## (Intercept)  -80.363813      NA  
## Pclass2      -1.821261 -0.5924824  
## Pclass3      -2.676712 -1.6082641  
## titleMiss      NA 81.1580568  
## titleMr       -4.806899 -2.7544131  
## titleMrs      NA 81.2072009  
## titleOfficer  -6.200669 -2.7777761  
## Sexmale      NA 81.9299127  
## ticket.sizeSingle 1.318753 2.9224495  
## ticket.sizeSmall 1.294852 2.8160324
```

```
###train data 예측하기
```

```
train.probs <- predict(log.mod, data=train_val1,type = "response")  
table(train_val1$Survived,train.probs>0.5)
```

```
##  
##      FALSE TRUE  
##  0    395    45  
##  1     70   204
```

```
(395+204)/(395+204+70+45)
```

```
## [1] 0.8389356
```

로지스틱 회귀모델은 train data 에서 accuracy rate 을 0.83 으로 예측했다.

```
test.probs <- predict(log.mod, newdata=test_val1,type = "response")  
table(test_val1$Survived,test.probs>0.5)
```

```
##  
##      FALSE TRUE  
##  0     97    12  
##  1     21    47
```

```
(97+47)/(97+12+21+47)
```

```
## [1] 0.8135593
```

teat data 의 accuracy rate 는 0.8135 이다.

여러 모델을 실행한 결과 대부분 결과가 비슷하게 나온 것을 확인할 수 있었다. 모델 선정도 중요하지만 데이터를 이해하고 변수 분석을 통해 좀더 예측 결과에 효과를 줄 수 있도록 결측치 처리 와 기존 변수를 통한 파생변수를 생성이 더 중요함을 느꼈다.