1-انتخاب صنعت

این داده ها مربوط به کمپین های بازاریابی مستقیم (تماس تلفنی) یک موسسه بانکی پرتغال است.

۲-درنظر گرفتن مشکلی در صنعت که برای حل ان از داده ها استفاده میکنیم

مشکل تجاری(Business Problem)

برای بانک پرتغالی کاهش درآمد حاصل شده است و آنها دوست دارند بدانند که چه اقداماتی باید انجام دهند. پس از بررسی ، متوجه شدیم که دلیل اصلی آن این است که مشتریان آنها به دفعات قبل سپرده گذاری نمی کنند. با دانستن اینکه سپرده های مدت دار به بانک ها امکان می دهد سپرده را برای مدت زمان مشخصی نگه دارند ، بنابراین بانک ها می توانند در محصولات مالی با سود بالاتر سرمایه گذاری کنند تا سود کسب کنند. علاوه بر این ، بانک ها فرصت بیشتری برای ترغیب مشتریان سپرده مدت دار برای خرید محصولات دیگر مانند وجوه یا بیمه برای افزایش بیشتر درآمد خود دارند. در نتیجه ، بانک پرتغالی مایل است مشتریان موجود را که شانس بیشتری برای اشتراک سپرده مدت دار دارند شناسایی کند و تلاش بازاریابی را بر روی چنین مشتریانی متمرکز کند.

هدف تجزیه و تحلیل

ویژگی مورد نظر "y" است - "آیا مشتری مشترک سپرده مدت دار شده است؟"

طبق مجموعه داده ، "y" دارای دو کلاس است("بله" یا "نه").بنابراین به عنوان یک مشکل طبقه بندی باینری شناخته می شود.

بله: مشتری مشترک سپرده مدت دار است.

خیر: مشتری مشترک سپرده مدت دار نشده است.

هدف ما یک رویکرد طبقه بندی (classification) به کمک الگوریتم درخت تصمیم، برای پیش بینی مشتریانی که به احتمال زیاد برای سپرده های مدت دار مشترک می شوند. (متغیر y). همچنین خوشه بندی با روش k-means را اجرا میکنیم.

۳-جمع آوری داده ها و یکپارچه سازی

دیتای ما از بانک دیتا ازسایت https://archive.ics.uci.edu/ml/datasets/Bank+Marketing با نام

Bank Marketing Data Set در فرمت csv دانلود شده است.(فایل دیتاضمیمه شده است). این دیتا دارای4521 نمونه و 17 ویژگی است.(یک ماتریس با 4521 سطر و 17 ستون)

در مرحله بعد ، ما رابطه بین هر متغیر را در مقابل y (مشتری در سپرده مدت دار مشترک شده است یا نه) از طریق نمودار جعبه ای تجسم خواهیم کرد. متوجه شدیم که فقط مدت زمان در میان سایر متغیرها دارای اهمیت است.

۴-مشخص کردن انواع داده

این دیتا ست که از انبار (repository) داده یادگیری ماشین UCI دانلود شده و دارای 17 ویژگی است که اکنون مروری بر ویژگیهای داده این داده را انجام میدهیم:

اطلاعات ویژگی(توضیح ستون های دیتا):

1. Age: سن (Integer)

2. job : نوع شغل (Factor)
3. Marital: وضعیت تاهل (Factor)
4. Education: تحصیلات (Factor)
5. default: آیا پیش فرض در اعتبار؟ (Factor)
6. Balance: وام تعادل دارد؟ (Integer)
7. housing: وام مسکن دارد؟ (Factor)
8. loan: وام شخصی دارد؟ (Factor)
9. contact: تماس با تلفن همراه یا تلفن منزل؟ (Factor)
10. day: آخرین روز تماس (Integer)
11. month آخرین ماه تماس (Factor)
12. duration: آخرین مدت زمان تماس (Integer)
13. campaign: تعداد مخاطبی که در طی این کمپین و برای این مشتری انجام شده است (عددی شامل آخرین مخاطب است) (Integer)
14. pdays: تعداد روزهایی که پس از آخرین تماس مشتری از یک کمپین قبلی گذشته است (عدد: 999 به این معنی است که مشتری قبلاً تماس نگرفته است) (Integer)
15. Previous: تعداد مخاطبی که قبل از این کمپین و برای این مشتری انجام شده است. (Integer)
16. poutcome: نتیجه بازاریابی قبلی (طبقه بندی شده: "شکست" ، "موجود" ، "موفقیت") (Factor)
17. y : آیا مشتری سپرده مدت دار مشترک شده است؟ (باینری: "بله" ، "نه"). این ستون نشان دهنده متغیر خروجی یا همان هدف مورد نظر(desired target) ما است.

5-الگوریتم درخت تصمیم برای classification

C5.0 Decision Tree Algorithm for classification and feature selection

یکی از شناخته شده ترین الگوریتم های پیاده سازی درخت تصمیم، C5.0 است. درختان تصمیم تحت الگوریتمC5.0 ، در مقایسه با مدل های پیشرفته یادگیری ماشین (به عنوان مثال شبکه های عصبی و ماشین های بردار پشتیبان) معمولاً تقریباً عملکرد خوبی دارند و به راحتی قابل استفاده و استقرار هستند.

- انتخاب بهترین تقسیم (Best Split)

اولین چالش اصلی که درخت تصمیم با آن روبرو می شود ، شناسایی ویژگی مورد تقسیم است .اگر تفکیک داده ها فقط یک کلاس واحد داشته باشد ، داده خالص (pure) در نظر گرفته می شود. C5.0 از مفهوم آنتروپی (entropy)برای اندازه گیری خلوص استفاده می کند .آنتروپی نمونه ای از داده ها نشان می دهد که مقادیر کلاس چقدر مخلوط شده اند . مقدار 0 نشانگر این است که نمونه کاملاً همگن است ، در حالی که 1 بیشترین میزان اختلال را نشان می دهد. فرمول آنتروپی بصورت زیر است:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

برای بخش مشخصی از داده ها (S) ، اصطلاح c به تعداد سطح کلاس های مختلف اشاره دارد و pi به نسبت مشاهداتی که در هر کلاس قرار می گیرد اشاره دارد. برای 2 کلاس، آنتروپی از 0 تا 1 است. برای n کلاس ، آنتروپی از 0 تا log2(n) است .حداقل آنتروپی مربوط به داده هایی است که کاملاً همگن هستند (کاملاً قطعی / قابل پیش بینی) و حداکثر آنتروپی نشان دهنده داده های کاملاً بی نظم (تصادفی یا بسیار پر سر و صدا). شاید از خود بپرسید که استفاده از آنتروپی چه فایده ای دارد؟ پاسخ این است که هرچه آنتروپی کوچکتر باشد ، اطلاعات بیشتری در تقسیم گره تصمیم مربوطه وجود دارد بنابراین سر هر گره درخت، ویژگی با حداقل آنتروپی برای تقسیم انتخاب میشود.

-هرس کردن درخت تصمیم(Pruning the Decision Tree)

یک درخت تصمیم ممکن است داده های ما را بیش از حد طبقه بندی کند. بنابراین چگونه می توان اندازه درخت تصمیم را کنترل کرد؟ یک راه حل ممکن این است که تعداد تصمیماتی که یک درخت تصمیم می گیرد را قطع کنیم .به طور مشابه ، ما می توانیم حداقل تعداد نقاط را در هر بخش کنترل کنیم .به این روش درخت تصمیم گیری ، توقف زودرس یا قبل از هرس گفته می شود .با این حال ، این ممکن است قبل از برخی از پارتیشن های مهم، روند تصمیم گیری را خاتمه دهد. الگوریتم از روش پس از هرس استفاده میکند. در روش پس از هرس، ما با رشد یک درخت تصمیم بزرگ شروع کنیم و متعاقبا شاخه ها را بر اساس نرخ خطا با مجازات در گره ها کاهش دهیم. این اغلب موثرتر از محلول قبل از هرس است.

الگوریتم C5.0 از روش پس از هرس (post-pruning method) برای کنترل اندازه درخت تصمیم استفاده می کند. در ابتدا یک درخت بزرگ شامل تمام احتمالات و پارتیشن بندی ها رشد می کند. سپس ، گره ها و شاخه ها را قطع می کند.

پیاده سازی مدل در R

ابتدا با استفاده از تابع read.csv() مجموعه داده بارگیری شده و به نام "bank" ذخیره می شود.. شرح کوتاه هر کد به همراه کد نوشته شده است(بصورت کامنت با علامت #مشخص شده است)

```
# #Import data to R

bank <- read.csv("C:/Users/fahime/Desktop/bank.csv", sep = ";", header = TRUE, stringsAsFactors = FALSE)
```

نکته: دقت در فراخوانی داده بجای آدرس C:/Users/fahime/Desktop/ دقیقا آدرس جایی را قرار دهید که فایل bank-full.csv در آن قرار دارد.

برای تأیید منطبق بودن نوع ویژگی با توضیحات شرح داده شده در قسمت **مشخص کردن انواع داده** ، از تابع str () استفاده می شود.

```
#Check the class

str(bank)
```

نتیجه بصورت زیراست:

```
'data.frame':   45211 obs. of  17 variables:
 $ age     : int  58 44 33 47 33 35 28 42 58 43...
 $ job     : chr  "management" "technician" "entrepreneur" "blue-collar... "
 $ marital  : chr  "married" "single" "married" "married... "
 $ education: chr  "tertiary" "secondary" "secondary" "unknown... "
 $ default  : chr  "no" "no" "no" "no... "
 $ balance  : int  2143 29 2 1506 1 231 447 2 121 593...
 $ housing  : chr  "yes" "yes" "yes" "yes... "
 $ loan     : chr  "no" "no" "yes" "no... "
 $ contact  : chr  "unknown" "unknown" "unknown" "unknown... "
 $ day     : int  5 5 5 5 5 5 5 5 5 5...
 $ month    : chr  "may" "may" "may" "may... "
 $ duration : int  261 151 76 92 198 139 217 380 50 55...
 $ campaign : int  1 1 1 1 1 1 1 1 1 1...
 $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1...
 $ previous : int  0 0 0 0 0 0 0 0 0 0...
 $ poutcome : chr  "unknown" "unknown" "unknown" "unknown... "
 $ y       : chr  "no" "no" "no" "no" ...
```

اسکن برای مقادیر از دست رفته (NA)

این مجموعه داده با استفاده از تابع is.na() برای مقادیر از دست رفته اسکن می شود.هیچ مقداری از دست رفته یافت نمی شود. اکنون می توان فرض کرد که مجموعه داده هیچ مقداری از دست رفته را نداشته باشد.

```
#Scanning for NAs
colSums(is.na(bank))
```

نتیجه:

```
age       job   marital education   default   balance   housing      loan
      0         0         0         0         0         0         0         0
 duration  campaign     pdays  previous  poutcome         y
```

0 0 0 0 0 0

برای پیاده سازی مدلها ابتدا باید دیتا را به دو قسمت train و test تقسیم کنیم. 80درصد داده رو بصورت تصادفی برای train و 20درصد باقیمانده را بعنوان test درنظر میگیریم.

```
####Model 1 - Build Model by fitting Decision Tree Classification

#Split the dataset into traning and testing datase

set.seed(254)

train_ind <- sample(1:nrow(data), 0.8 * (nrow(data)))

training_set <- data[train_ind, ]

testing_set <-data[-train_ind,]
```

ما برای آموزش مدل درخت تصمیم گیری خود از الگوریتم C5.0 در پکیج C50 استفاده خواهیم کرد. C5.0 یک مدل درخت طبقه بندی یا مدل های مبتنی بر الگوریتم Quinlan’s C5.0 است.

```
#fit a simple classification tree model

#Fit classification tree models or rule-based models using Quinlan's C5.0 algorithm

install.packages("C50")

library(C50)

cmodel <- C5.0(x = training_set[, -17], y = as.factor(training_set$y))

#Results of train model

summary(cmodel)
```

نتیجه:

```
Evaluation on training data (3616 cases):

        Decision Tree

      ----------------

      Size      Errors

        40     268( 7.4%)   <<
```

(a) (b) <-classified as

---- ----

3142 54 (a): class no

214 206 (b): class yes

Attribute usage:

100.00% duration

100.00% poutcome

30.97% month

19.39% contact

13.91% loan

7.88% default

3.48% age

1.80% housing

1.30% day

1.30% campaign

1.02% marital

0.53% education

0.36% previous

0.22% balance

خروجی خطاها اشاره می کند که مدل به درستی همه موارد را به جز 268 مورد از 3616 مورد آموزش را برای نرخ خطای 7.4 درصد طبقه بندی کرده است. درمجموع 54 نمونه واقعی به طور اشتباه به عنوان خیر (مثبت نادرست) طبقه بندی شده اند ، در حالی که 214 نمونه بله در طبقه بندی غلط خیر (منفی کاذب) طبقه بندی شده اند.

حال 5 ویژگی برتر را شناسایی میکنیم:

Attribute usage:

100.00%	duration
100.00%	poutcome
30.97%	month
19.39%	contact
13.91%	loan

اکنون عملکرد مدل را ارزیابی میکنیم.برای اعمال درخت تصمیم بر روی مجموعه داده های آزمایشی(testing_set) ، از تابع predict استفاده می کنیم ، همانطور که در کد زیر نشان داده شده است:

```
###Evaluate model performance
cmodel_pred <- predict(cmodel, testing_set)

###Cross table validation
install.packages("gmodels")
library(gmodels)
CrossTable(testing_set$y, cmodel_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
          dnn = c('actual default', 'predicted default'))
```

نتیجه:

Total Observations in Table: 905

	predicted default		
actual default	no	yes	Row Total
no	780	24	804
	0.862	0.027	

yes	69	32	101
	0.076	0.035	
Column Total	849	56	905

نتایج برای 905 نمونه تست نشان میدهد که مدل ما 780 مورد(no) و 32 مورد(yes) را درست پیش بینی کرده است. در نتیجه ، دقت مدل 89.7 درصد(0.862+0.35=0.897) و میزان خطای مدل 10.3 درصد (0.027+0.076=0.103) است.

نتایج نشان می دهد که مدل با خطای کم (7.4٪) برای ارزیابی داده های آموزش و دقت 89.7٪ در مجموعه آزمون ، ارزیابی شده است.

6- روش clustering با k-means

یکی از معایب عمده این روش این است که فقط با متغیرهای عددی کار می کند. از آنجاییکه هدف ما تقسیم بندی مشتریانمان است، بنابراین ما تمام ستون های کاراکتری را باید به عدد تبدیل کنیم. ابتدا متغیرهایی که دارای دوحالت yes و no هستند را به یک و صفر تبدیل میکنیم:

```
###### Model 2 - Build Model by fitting K-means clustering ####

###Converting to numerial factors

# with two levels are transformed into a binary numerical variable

bank$y <- recode(bank$y, "'no'=0; 'yes'=1", as.factor=FALSE)

bank$default <- recode(bank$default, "'no'=0; 'yes'=1", as.factor=FALSE)

bank$housing <- recode(bank$housing, "'no'=0; 'yes'=1", as.factor=FALSE)

bank$loan <- recode(bank$loan, "'no'=0; 'yes'=1", as.factor=FALSE)
```

سپس مابقی متغیرهای کاراکتری را به عدد تبدیل میکنیم:

```
# This code of chunk make the character data into numeic format

bank$job <- as.numeric(as.factor(bank$job))

bank$marital <- as.numeric(as.factor(bank$marital))

bank$education <- as.numeric(as.factor(bank$education))

bank$month <- as.numeric(as.factor(bank$month))

bank$contact <- as.numeric(as.factor(bank$contact))
```

```
bank$poutcome <- as.numeric(as.factor(bank$poutcome))
```

چک میکنیم که تمام متغیرها فرم عددی داشته باشند:

```
str(bank)
```

نتیجه بصورت زیر در کنسول قابل مشاهده هست و میبینم دیگر کاراکتری وجود ندارد:

```
'data.frame':   4521 obs. of  17 variables:
 $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
 $ job      : num  11 8 5 5 2 5 7 10 3 8 ...
 $ marital  : num  2 2 3 2 2 3 2 2 2 2 ...
 $ education: num  1 2 3 3 2 3 3 2 3 1 ...
 $ default  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
 $ housing  : num  0 1 1 1 1 0 1 1 1 1 ...
 $ loan     : num  0 1 0 1 0 0 0 0 0 1 ...
 $ contact  : num  1 1 1 3 3 1 1 1 3 1 ...
 $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
 $ month    : num  11 9 1 7 9 4 9 9 9 1 ...
 $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
 $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
 $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
 $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
 $ poutcome : num  4 1 1 4 4 1 2 4 4 1 ...
 $ y        : num  0 0 0 0 0 0 0 0 0 0 ...
```

همچنین نرمالیزه کردن متغیرهایی که در k-means استفاده می کنیم، رایج است. تابعی با نام normalize تعریف میکنیم:

```
# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

به کمک تابع تعریف شده دیتای bank را نرمال سازی میکنیم:

```
# normalize the data to get rid of outliers if present in the data set

bank_normalize <- as.data.frame(lapply(bank, normalize))
```

یکی دیگر از معایب این است که کاربر باید به k-means تعداد خوشه هایی که الگوریتم تعریف می کند بگوید. یک راه برای تصمیم گیری این است که چندین اعداد را امتحان کنیم و رفتار مجموع مجذورهای درون خوشه ای را مطالعه کنید. ما از 2 تا 7 خوشه را امتحان خواهیم کرد.

این پارامتر با دستور kmeans برگردانده می شود و ما آن را در متغیری به نام totwinss ذخیره خواهیم کرد:

```
totwinss=c()

for (i in 2:7){

  k_cl=kmeans(bank_normalize,i)

  totwinss[i] <- k_cl$tot.withinss

}
```

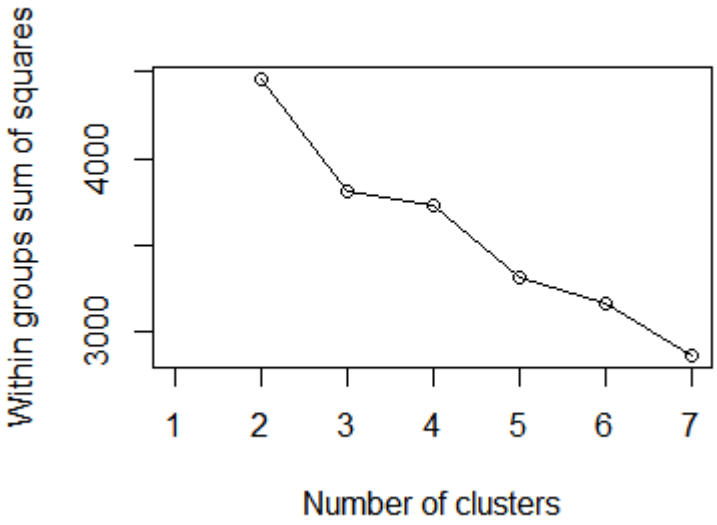
اکنون متغیر totwinss را رسم میکنیم:

```
plot(1:7, totwinss,

    xlab = "Number of clusters",

    ylab = "Total ithin ss")

lines(1:7, totwinss)
```



چیزی که ما دنبال آن می گردیم یک به اصطلاح آرنج (elbow) است، جاییکه مجموع مربع های درون خوشه ای (within-cluster sum of squares) به طور ناگهانی کاهش می یابد و سپس کم و بیش مسطح میشود. در اینجا نامزدهای احتمالی 3 یا 5 خوشه (محور x) هستند..

ابتدا 5 را خوشه امتحان کنیم:

```
#we have elbow in 5 clusters

k_cl=kmeans(bank_normalize,5)
```

میخواهیم دقت مدل را بصورت دستی محاسبه کنیم. اول متغیر هدف را به فاکتور سپس به عدد تبدیل میکنیم:

```
#Calculate the accuracy of the model

#chang y variable to numeric format

directions.factor<-factor(data$y)

y<-as.numeric(directions.factor)
```

متغیر k_cl حاوی لیستی است خوشه ای که هر نمونه dat_scaled به آن تعلق دارد. این لیست توسط k_cl$cluster قابل دسترسی است و در متغیر kmean_result ذخیره میکنیم:

```
kmean_result<-cbind(k_cl$cluster,y)

n_true_kmean<-kmean_result[which(kmean_result[,1] == kmean_result[,2]), ]
```

با کمک فرمول زیر دقت مدل را ارزیابی میکنیم:

```
accuracy_kmean<- (nrow(n_true_kmean)/4521)*100
```

accuracy_kmean

12.09909

دقت مدل 12 درصد هست و نشانه تعداد خوشه نامناسب است. پس 3 خوشه را امتحان میکنیم:

```
#we have elbow in 3 clusters
k_cl2=kmeans(bank_normalize,3)

#Calculate the accuracy of the model
kmean_result2<-cbind(k_cl2$cluster,y)
n_true_kmean2<-kmean_result2[which(kmean_result2[,1] == kmean_result2[,2]), ]
accuracy_kmean2<- (nrow(n_true_kmean2)/4521)*100
accuracy_kmean2
```

40.58837

دقت مدل افزایش چشمگیری داشت. این نشان دهنده این است که انتخاب تعداد خوشه نامناسب میتواند ضربه بدی به مدل بزند.

چون در دیتای اصلی دو دسته برای متغیر y داشتیم پس دو کلاستر را هم بررسی میکنیم:

```
#we have elbow in 2 clusters
k_cl3=kmeans(bank_normalize,2)

#Calculate the accuracy of the model
kmean_result3<-cbind(k_cl3$cluster,y)
n_true_kmean3<-kmean_result3[which(kmean_result3[,1] == kmean_result3[,2]), ]
accuracy_kmean3<- (nrow(n_true_kmean3)/4521)*100
accuracy_kmean3
```

61.09268

دقت خوشه بندی در حالت دو خوشه بیشترین مقدار شد.