۱

پروژه درس داده کاوی

طبقه بندی

استاد محترم درس

جناب آقای دکتر پاینده

دانشجو

آیدا اعلابیکی

فهرست مطالب

در ابتدا بر خود واجب می‌دانم از زحمات استاد ارجمندم جناب آقای دکتر پاینده، کمال تشکر را به جا آورم.

2 مقدمه

داده‌های این پروژه شامل سوابق پزشکی 299 بیمارِ مبتلا به نارسایی قلبی است که در آوریل و دسامبر سال 2015 جمع‌آوری شده‌اند. برای هر بیمار 13 متغیر اندازه‌گیری شده است. 12 متغیر که به نظر می‌رسد در مرگ بیمارانی که نارسایی قلبی دارند موثر هستند و متغیر دیگری با نام «وقوع مرگ» که هدف از جمع آوری این داده‌ها بررسی اثر 12 متغیر دیگر بر روی این متغیر بوده است. متغیرهایی که در مجموعه داده‌ها وجود دارند به صورت زیر هستند:

شماره	متغیر	توضیحات متغیر	نوع متغیر
1	Age	سن بیمار بر حسب سال	کمی
2	Anaemia	کم خونی (1=دارد و 0=ندارد)	اسمی دارای دو سطح
3	Phosphokinase	کراتین فسفوکیناز نشان‌دهنده سطح آنزیم CPK در خون است و برحسب میکروگرم بر لیتر اندازه‌گیری می‌شود.	کمی
4	Diabetes	دیابت (1=دارد و 0=ندارد)	اسمی دارای دو سطح
5	Ejection	درصد خون خروجی از قلب در هر انقباض (به درصد)	کمی
6	Pressure	فشار خون بالا (1=دارد و 0=ندارد)	اسمی دارای دو سطح
7	Platelets	پلاکت‌های موجود در خون (کیلوپلاکت در میلی لیتر)	کمی
8	Creatinine	سطح کراتین در خون(mg/dL)	کمی
9	Sodium	سطح سدیم در خون(mEq/L)	کمی
10	Sex	جنسیت (1=مرد و 0=زن)	اسمی دارای دو سطح
11	Smoking	سیگار کشیدن (1=بله و 0=خیر)	اسمی دارای دو سطح
12	Time	تعداد روزهایی که بیمار تحت نظر بوده است	کمی گسسته
13	DEATH	فوت (1=بله و 0=خیر)	اسمی دارای دو سطح

برای این داده‌ها متغیر پاسخ وقوع مرگ (DEATH یا در مجموعه‌ی داده‌ها DEATH_EVENT) بوده

که یک متغیر کیفی با دو سطح است.

داده‌ها از طریق لینک‌های زیر قابل دانلود هستند:

https://archive.ics.uci.edu/dataset/519/heart+failure+clinical+records
https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data

آماده سازی داده‌ها

پیش از شروع بحث کاربردی و اعمال الگوریتم‌ها و مدل‌های همسو با مجموعه داده‌ی موردنظرمان، باید مراحلی را با هدف فراهم نمودن زمینه‌ی تحلیل و بررسی شناخت حاصل شده در مرحله‌ی قبل، در جهت رسیدن به اهداف این پروژه طی نمود.

یکی از مراحل مهم در هر گونه مطالعه، به‌کارگیری و دسترسی به داده‌های اولیه خوب ومناسب است؛ که از آن به آماده‌سازی یا پیش پردازش داده‌ها یاد می شود. در واقع برای رسیدن به نتایج محتمل بایستی مقدماتی صورت گیرد؛ که مجموعه این اقدامات را آماده‌سازی داده‌ها گویند. به عبارت دیگر می‌توان گفت؛ فرایند تبدیل داده‌ها از یک قالب داده‌ی خام به ساختار موردنظرمان، با هدف ایجاد داده‌ی مناسب‌تر و با ارزش‌تر برای تحقق اهداف گفته می‌شود. با این حال، متأسفانه بسیاری اهمیت آماده‌سازی داده‌ها را فراموش کرده و یا آن را کم اهمیت می انگارند.

وظیفه اصلی آماده‌سازی داده‌ها؛ سازمان دهی داده‌ها در شکل‌های استاندارد برای داده‌کاوی یا سایر عملیات و یا فراهم کردن شرایط موردنیاز قبل از ورود به مراحل کاربردی می‌باشد. اهمیت آماده‌سازی داده‌ها به این دلیل است که اگر داده با کیفیت مناسبی نداشته باشیم، نتیجه‌ای که به دست می‌آید نیز باکیفیت نخواهد بود. همچنین اگر روش‌های پیش پردازش قبل از شروع داده‌کاوی انجام گیرد می‌تواند به میزان قابل توجهی باعث بهبود فرایند داده‌کاوی و کاهش زمان لازم برای اجرای آن شود.

شاید بطور کلی و خلاصه بتوان دو وظیفه‌ی زیر را برای مرحله‌ی آماده سازی داده‌ها در نظرگرفت:

- سازماندهی داده‌ها در یک شکل استاندارد تا برای پردازش در عمل داده کاوی مناسب باشند.

- آماده‌سازی مجموعه داده‌ها تا الگوریتم‌های داده‌کاوی بتوانند با کارایی بالایی اجرا شوند.

گاهی به تکنیک‌های آماده‌سازی داده‌ها، تکنیک‌های پیش پردازش داده‌ها نیز گفته می‌شود.

بررسی داده‌های گمشده

در بسیاری از تحقیقات با مواردی برخورد می‌کنیم که برای بعضی از متغیرها پاسخ یا جوابی وجود ندارد یا اینکه پاسخ نامعلوم بوده و قادر به استخراج آن نیستیم. به این مقادیر، مقادیر نامعلوم یا داده‌های گم‌شده می‌گوییم. درادبیات‌های آماری مترادفی برای این مفهوم وجود دارد. این اصطلاحات عبارت‌اند از مقادیر گم‌شده، داده‌های گم‌شده، داده‌های ناقص وبی‌پاسخ.

داده‌های گم‌شده مشکل‌های بسیاری را برای نتیجه‌گیری‌های صحیح پژوهش‌های آماری به وجود می‌آورند، زیرا می‌توانند قدرت استنباط آماری مطالعه را کاهش داده، باعث کاهش کارایی برآوردگرها و ایجاد اریبی شوند؛ بنابراین برای جلوگیری از مشکلاتی مانند کم شدن حجم نمونه و یا عدم حضور واحدهایی از نمونه در مجموعه ی داده‌ها باید با استفاده از روش‌های مناسب جایگذاری شوند.

اغلب در مجموعه‌ی بزرگی از داده‌ها نمونه‌هایی وجود دارند که رفتارشان با رفتار عمومی نمونه‌ها یکسان نیست. این رفتار یا کامل مختلف است و یا با دیگر نمونه‌ها ناسازگارند. به عبارتی دیگر همیشه داده‌های ما ناقص نیستند، می‌توانند وجود داشته باشند، اما با رفتاری متفاوت از اکثر نمونه‌های موجود.

وجود چنین داده‌هایی در اکثر مواقع نتیجه عملکرد نادرست کاربران است. با توجه اینکه این داده‌ها می‌توانند نتایج تحلیل را تحت تاثیر قرار دهند، می‌بایست تمامی داده‌های پرت در ابتدا شناسایی و بررسی شوند. نویز با داده‌ی پرت متفاوت است، نویز خطای یا واریانس تصادفی در داده است که باید قبل از تشخیص داده پرت حذف شود.

به‌طور خلاصه دلایل شناسایی داده‌های دور افتاده را می‌توان به صورت زیر بیان کرد :

1- مشخص کردن مشکلات پروسه به‌دست آوردن داده‌ها با تحلیل داده‌های دور افتاده امکان‌پذیر است.

2- تحلیل داده‌های دور افتاده برای غربال کردن داده‌ها برای آنالیز بیشتر مفید است.

- **دسته‌بندی داده‌ها**

داده‌های کمی را می‌توان دسته‌بندی کرد و این تکنیک هم می‌تواند به منظور تشخیص و حذف داده‌های نویز و مزاحم استفاده شوند، هم اینکه برای کاهش حجم داده‌ها نیز روش مفیدی است. فرض کنید که صفت خاصه‌ای شامل یک سری داده‌های کمی با محدوده‌ی مشخص و قابل شمارش است. در صورت کاهش تعداد داده‌ها می‌توانیم امیدوار باشیم تا روش‌های داده‌کاوی کارایی بهتری از خود نشان می‌دهند. داده‌ها می‌توانند به طرق مختلفی دسته‌بندی و پس از آن داده‌های هر دسته با یک مفهوم کلی‌تر دیگری نمایش داده شوند.

در ادامه نیز با بحث تبدیل داده‌ها در قالب عملیات‌هایی همچون استاندارسازی و تغییر یا تجمیع داده‌ها روبه‌رو خواهیم شد.

رفع مشکل افزونگی داده‌ها

منظور از افزونگی در یک پایگاه داده آن است که اطلاعات به صورت سطری یا ستونی در جدول بیهوده تکرار شوند. به این معنی که ممکن است در یک یا چند جدول، رکوردهایی وجود داشته باشند که تعداد فیلدهای تکراری زیادی داشته باشند. حتی در کل رکوردهایی از اطلاعات در جدول‌های مختلف ثبت و ذخیره شوند. این وضعیت را به نام مشکل افزونگی در پایگاه داده می‌شناسیم. به این ترتیب حجم اطلاعات بیش از حد شده و حتی جستجو در بانک اطلاعاتی نیز ممکن است دچار مشکل شود.

در این حالت با رفع مشکل افزونگی، ثبت اطلاعات تکراری به حداقل می‌رسد و به اصطلاح سیستم بهینه می‌شود.

مشاهدات مختلف داده هنگامی که گسسته از یکدیگر طراحی می‌شوند، دارای فیلدها و داده‌های یکسانی نیستند. برای این گونه مسئله‌ها روش‌های متعددی وجود دارد که برخی از آن‌ها همچون افزونگی معمول

در دادگان‌ها را با آزمون‌های مختلف آماری می‌توان حل کرد.

از آن‌جایی که داده‌ها از منابع مختلف جمع‌آوری می‌شوند، ممکن است دارای ناسازگاری‌هایی باشند مانند تفاوت در مقیاس یا صفت‌های مختلف به‌گونه‌ای با یکدیگر مرتبط باشند. در این موقعیت پیش آمده، باید سعی کنیم داده‌ها را یکپارچه سازیم که حتی الامکان دارای تفاوت کمی باشند. یکپارچه‌سازی داده‌ها در تحلیل آن‌ها بسیار حائز اهمیت است.

تغییر شکل داده‌ها

شکل مناسب داده‌ها به عنوان ورودی الگوریتم‌های داده‌کاوی نقش به سزایی در این فرایند بازی می‌کند و در مرحله‌ی آماده‌سازی داده‌ها این نقش پررنگ است. تکنیک‌های تغییر شکل داده‌ها متکی به مشکل نیستند و اغلب در اجرا نتایج بهتری از داده‌کاوی را سبب می‌شوند. استفاده از توابع تجمعی، نرمال سازی، خوشه‌بندی و رگرسیون روش‌های مرسومی هستند که برای تغییر و تبدیل شکل داده‌ها می‌توان از آن‌ها استفاده نمود. انتخاب روش مناسب به ماهیت داده‌ها، مقدار آن و اهداف داده‌کاوی بستگی دارد. استانداردسازی داده‌ها کمک می‌کند که اهمیت آن‌ها به واحد اندازه‌گیری‌شان بستگی نداشته باشد. در نتیجه در مواردی مانند داده‌کاوی و تحلیل داده‌های چند متغیره از داده‌های استاندارد شده استفاده می‌شود. قبل از انجام هر گونه تحلیلی روی داده‌ها، باید آن‌ها را استاندارد کرد. بخصوص زمانی که داده‌ها چند بُعدی باشند. استفاده از داده‌های استاندارد نشده ممکن است روی نتایج حاصل از تحلیل‌ها اثر نامناسبی داشته باشد.

در ابتدا داده‌ها را فراخوانی می‌کنیم.

```
#read data
Heart=read.csv("C:/Users/acer/Desktop/heart.csv")
head(Heart)
```

```
  age anaemia creatinine_phosphokinase diabetes ejection_fraction high_blood_pressure platelets
1  75       0                     582        0                20                    1    265000
2  55       0                    7861        0                38                    0    263358
3  65       0                     146        0                20                    0    162000
4  50       1                     111        0                20                    0    210000
5  65       1                     160        1                20                    0    327000
6  90       1                      47        0                40                    1    204000
  serum_creatinine serum_sodium sex smoking time DEATH_EVENT
1              1.9          130   1       0    4           1
2              1.1          136   1       0    6           1
3              1.3          129   1       1    7           1
4              1.9          137   1       0    7           1
5              2.7          116   0       0    8           1
6              2.1          132   1       1    8           1
```

برای شناخت بیشتر داده‌ها از کدهای زیر استفاده می‌کنیم.

```
str(Heart)
```

```
> str(Heart)
'data.frame':   299 obs. of  13 variables:
 $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
 $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
 $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
 $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
 $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
 $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
 $ platelets               : num  265000 263358 162000 210000 327000 ...
 $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
 $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
 $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
 $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
 $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
 $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

تابع بالا تعداد متغیرها و مشاهدات داده‌ی مورد نظر را نمایش می‌دهد و همچنین مشخص می‌کند متغیرهای ما از چه نوعی هستند.

برای اطلاعات راجب داده‌های گمشده و تعداد آن‌ها داریم:

```
colSums(is.na(Heart))
```

```
> colSums(is.na(Heart))
                      age                   anaemia creatinine_phosphokinase                  diabetes
                        0                         0                         0                         0
       ejection_fraction         high_blood_pressure                 platelets           serum_creatinine
                        0                         0                         0                         0
            serum_sodium                       sex                   smoking                      time
                        0                         0                         0                         0
              DEATH_EVENT
                        0
```

به‌دلیل عدم وجود مقادیر گم‌شده در مجموعه داده‌ها سراغ گام بعدی می‌رویم و برای آگاه شدن از شاخص‌های مرکزی در نرم افزار R از طریق تابع summary خلاصه‌ای از متغیرهای جامعه ارائه می‌دهیم.

```
summary(Heart)
```

خروجی به صورت زیر است:

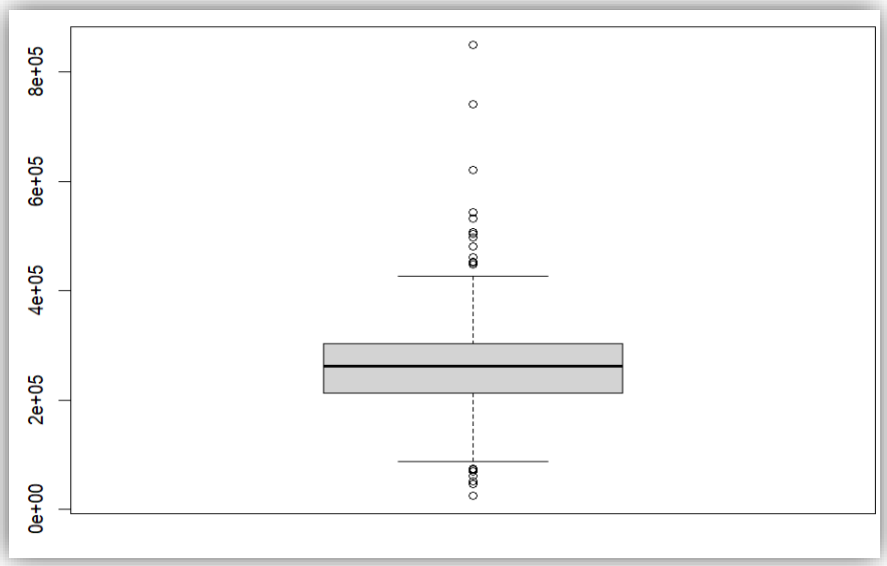
```
> summary(Heart)
      age            anaemia       creatinine_phosphokinase    diabetes      ejection_fraction
 Min.   :40.00   Min.   :0.0000   Min.   :  23.0           Min.   :0.0000   Min.   :14.00
 1st Qu.:51.00   1st Qu.:0.0000   1st Qu.: 116.5           1st Qu.:0.0000   1st Qu.:30.00
 Median :60.00   Median :0.0000   Median : 250.0           Median :0.0000   Median :38.00
 Mean   :60.83   Mean   :0.4314   Mean   : 581.8           Mean   :0.4181   Mean   :38.08
 3rd Qu.:70.00   3rd Qu.:1.0000   3rd Qu.: 582.0           3rd Qu.:1.0000   3rd Qu.:45.00
 Max.   :95.00   Max.   :1.0000   Max.   :7861.0           Max.   :1.0000   Max.   :80.00
 high_blood_pressure   platelets      serum_creatinine  serum_sodium        sex            smoking
 Min.   :0.0000      Min.   : 25100   Min.   :0.500    Min.   :113.0   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000      1st Qu.:212500   1st Qu.:0.900    1st Qu.:134.0   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000      Median :262000   Median :1.100    Median :137.0   Median :1.0000   Median :0.0000
 Mean   :0.3512      Mean   :263358   Mean   :1.394    Mean   :136.6   Mean   :0.6488   Mean   :0.3211
 3rd Qu.:1.0000      3rd Qu.:303500   3rd Qu.:1.400    3rd Qu.:140.0   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000      Max.   :850000   Max.   :9.400    Max.   :148.0   Max.   :1.0000   Max.   :1.0000
      time        DEATH_EVENT
 Min.   :  4.0   Min.   :0.0000
 1st Qu.: 73.0   1st Qu.:0.0000
 Median :115.0   Median :0.0000
 Mean   :130.3   Mean   :0.3211
 3rd Qu.:203.0   3rd Qu.:1.0000
 Max.   :285.0   Max.   :1.0000
```

با توجه به خروجی حاصل از تابع summary() مقادیر مینیمم، چارک اول، میانه، میانگین، چارک سوم و ماکزیمم برای هر متغیر موجود در جامعه، قابل مشاهده می‌باشد.

نمودار جعبه‌ای مربوط به متغیر platelets:

```
Heart%<%
  ggplot(aes( y = platelets))+
  geom_boxplot()
out=boxplot(Heart$platelets)$out
head(out)
```

```
> head(out)
[1] 454000  47000 451000 461000 497000 621000
```

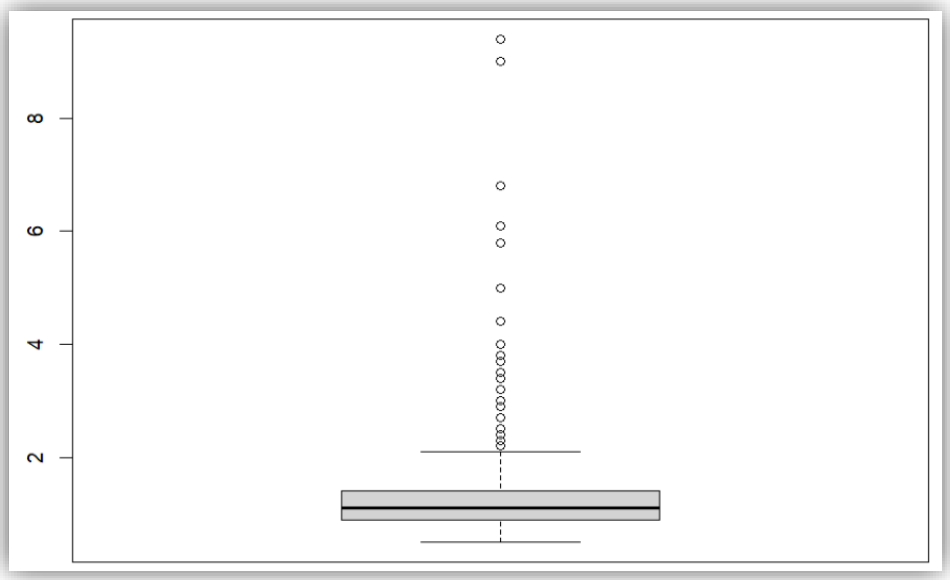


نمودار جعبه‌ای مربوط به متغیر serum_creatinine:

```
Heart%<%
  ggplot(aes( y = serum_creatinine))+
  geom_boxplot()
out=boxplot(Heart$serum_creatinine)$out
head(out)
length(out)
```

```
> head(out)
[1] 2.7 9.4 4.0 5.8 3.0 3.5
> length(out)
[1] 29
```

همانطور که از خروجی مشخص این متغیر شامل 29 مشاهده به عنوان داده پرت می‌باشد.



برای شناخت بهتر مشاهدات می‌توانیم از یکی دیگر از شاخص‌های پراکندگی، کوواریانس، و محاسبه‌ی ماتریس همبستگی متغیرهای کمی بهره ببریم.

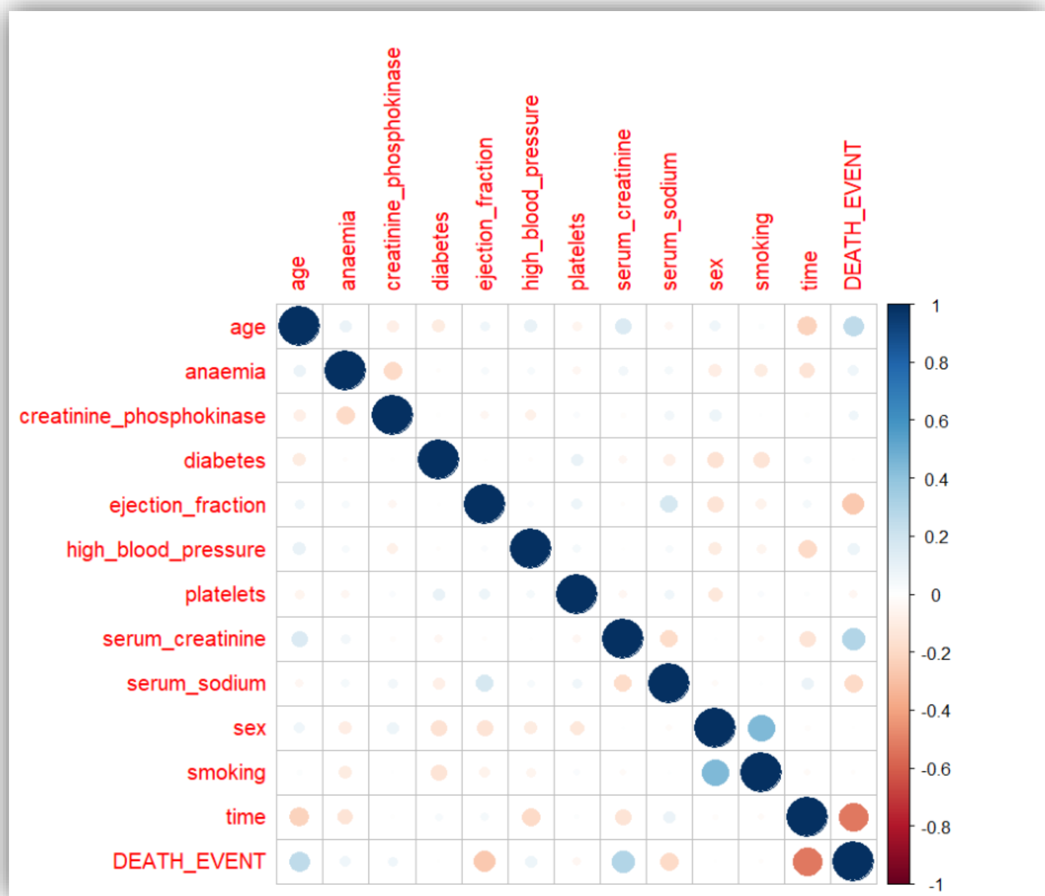
```
# Select only numeric variables for correlation
numeric_data <- Heart[, sapply(Heart, is.numeric)]

if (ncol(numeric_data) > 1) {
  cor_matrix <- cor(numeric_data, method = "pearson")
  print("Correlation Matrix:")
  print(cor_matrix)
  library(corrplot)
  corrplot(cor_matrix, method = "circle")
} else {
  print("Not enough numeric variables for a correlation matrix.")
}
```

[1] "Correlation Matrix:"

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction
age	1.00000000	0.08800644	-0.081583900	-0.101012385	0.06009836
anaemia	0.08800644	1.00000000	-0.190741030	-0.012729046	0.03155697
creatinine_phosphokinase	-0.08158390	-0.19074103	1.000000000	-0.009638514	-0.04407955
diabetes	-0.10101239	-0.01272905	-0.009638514	1.000000000	-0.00485031
ejection_fraction	0.06009836	0.03155697	-0.044079554	-0.004850310	1.00000000
high_blood_pressure	0.09328868	0.03818200	-0.070589980	-0.012732382	0.02444473
platelets	-0.05235437	-0.04378555	0.024463389	0.092192828	0.07217747
serum_creatinine	0.15918713	0.05217360	-0.016408480	-0.046975315	-0.01130247
serum_sodium	-0.04596584	0.04188161	0.059550156	-0.089550619	0.17590228
sex	0.06542952	-0.09476896	0.079790629	-0.157729504	-0.14838597
smoking	0.01866787	-0.10728984	0.002421235	-0.147173413	-0.06731457
time	-0.22406842	-0.14141398	-0.009345653	0.033725509	0.04172924
DEATH_EVENT	0.25372854	0.06627010	0.062728160	-0.001942883	-0.26860331

	dium	sex
age	5841	0.065429524
anaemia	1610	-0.094768961
creatinine_phosphokinase	0156	0.079790629
diabetes	0619	-0.157729504
ejection_fraction	2282	-0.148385965
high_blood_pressure	9470	-0.104614629
platelets	4619	-0.125120483
serum_creatinine	5210	0.006969778
serum_sodium	0000	-0.027566123
sex	5123	1.000000000
smoking	3195	0.445891712
time	0000	-0.015608220
DEATH_EVENT	3596	-0.004316376

	smoking	time	DEATH_EVENT
age	0.018667868	-0.224068420	0.253728543
anaemia	-0.107289838	-0.141413982	0.066270098
creatinine_phosphokinase	0.002421235	-0.009345653	0.062728160
diabetes	-0.147173413	0.033725509	-0.001942883
ejection_fraction	-0.067314567	0.041729235	-0.268603312
high_blood_pressure	-0.055711369	-0.196439479	0.079351058
platelets	0.028234448	0.010513909	-0.049138868
serum_creatinine	-0.027414135	-0.149315418	0.294277561
serum_sodium	0.004813195	0.087640000	-0.195203596
sex	0.445891712	-0.015608220	-0.004316376
smoking	1.000000000	-0.022838942	-0.012623153
time	-0.022838942	1.000000000	-0.526963779
DEATH_EVENT	-0.012623153	-0.526963779	1.000000000

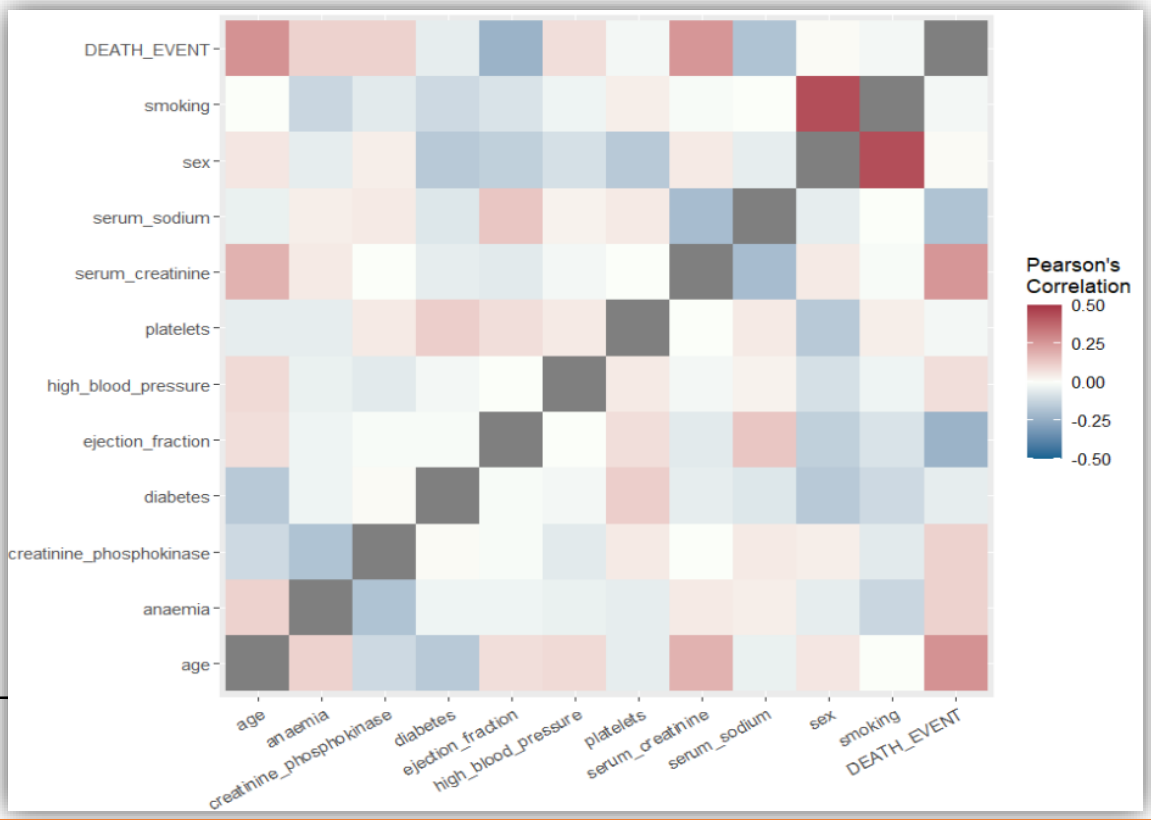


با توجه به خروجی به دست آمده توسط محاسبه‌ی همبستگی متغیرها با یکدیگر، ضریب همبستگی هر متغیر با خودش برابر با مقدار 1 است، و مقادیر منفی بیانگر وجود رابطه‌ی معکوس بین دو متغیر و مقادیر مثبت بیانگر رابطه‌ی هم‌جهت مابین آن دو می‌باشد.

هر چه این مقادیر نزدیک یا برابر مقدار 1 باشند، رابطه شدید و هم‌جهت بین دو متغیر موجود است، در این حالت می‌توان گفت که جهت تغییرات هر دو متغیر مانند یکدیگر است و بین دو متغیر رابطه مستقیم وجود دارد. بالعکس، اگر ضریب همبستگی، مقداری نزدیک یا برابر با ۱- باشد، رابطه شدید ولی در جهت عکس بین متغیرها وجود دارد. بنابراین با افزایش یکی، دیگری کاهش خواهد.

از طریق نمودار حرارتی نیز می‌توان همبستگی بین متغیرها را مشاهده نمود.

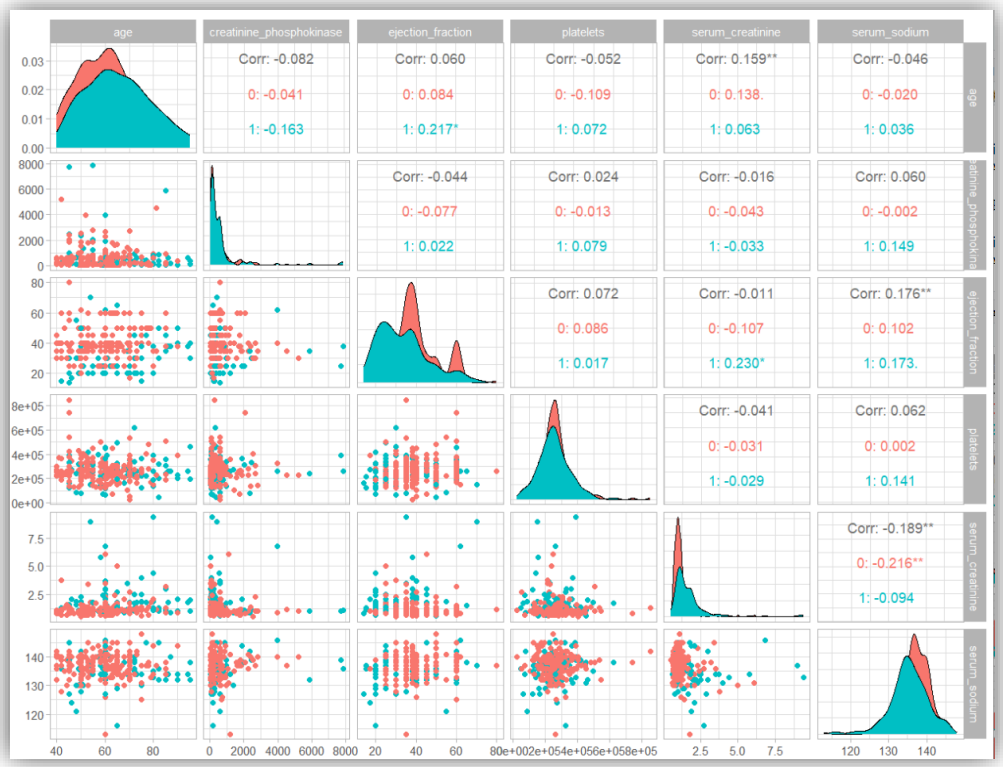
```
library(reshape2)
train_num <- data_num[train_index,]
cormat <- round(cor(train_num),2)
melted_cormat <- melt(cormat)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  labs(x = NULL, y = NULL, fill = "Pearson's\nCorrelation") +
  scale_fill_gradient2(mid="#FBFEF9",low="#0C6291",high="#A63446", limits=c(-
0.5,0.5)) +
  scale_x_discrete(guide = guide_axis(angle = 30))
```



در نمودار فوق هر چقدر از رنگ آبی به سمت رنگ قرمز مایل می‌شود، شاهد افزایش همبستگی در جهت مثبت بین متغیرها می‌شویم به بیانی دیگر می‌توان گفت دارای رابطه مستقیم می‌باشند و بالعکس رنگ آبی موجود در ماتریس حاصل دلالت بر همبستگی در جهت عکس متغیرها دارد. شایان ذکر است

```
install.packages("GGally")
install.packages("ggstats")
install.packages("hms")
library(GGally)
library(ggstats)
library(hms)
Heart$DEATH_EVENT <- factor(Heart$DEATH_EVENT)
Heart %>%
  ggpairs(columns = c("age", "creatinine_phosphokinase", "ejection_fraction", "platelets",
"serum_creatinine", "serum_sodium"),
          mapping = ggplot2::aes(color = DEATH_EVENT)) +
  ggplot2::theme_light()
}
```

می‌توان این نتیجه را از طریق رسم نمودارهای زیر نیز به‌دست آورد و پراکنش هر متغیر را مشاهده نمود.



در نمودار فوق، روی قطر اصلی نمودارهای توابع چگالی هر متغیر را شاهد هستیم.در درایه‌هایی که اندیس سطح از اندیس ستون کمتر است، همبستگی دو متغیر را نظاره‌گر هستیم.

در این مرحله به یکی از ارکان‌های مهم در علم آمار تحت عنوان مصورسازی می‌پردازیم.

ارائه یک روش موثر برای مصورسازی داده، به ما در شناخت و کشف روابط بین پدیده‌ها و داده‌های حاصل از اندازه‌گیری آن‌ها، کمک می‌کند. به این ترتیب مجموعه داده‌های پیچیده، به شکلی ساده و موثر، نمایش داده می‌شوند و امر استخراج قواعد یا تحلیل آن‌ها آسان می‌گردد. در قرن حاضر، با توجه به حجم بسیار زیاد اطلاعات و لزوم به شناخت رفتار آن‌ها در زمان کوتاه، مصورسازی داده به یک حوزه فعال تحقیق، تدریس و توسعه تبدیل شده است، بطوری که این تکنیک، تجسم علمی و اطلاعاتی را متحد کرده است.

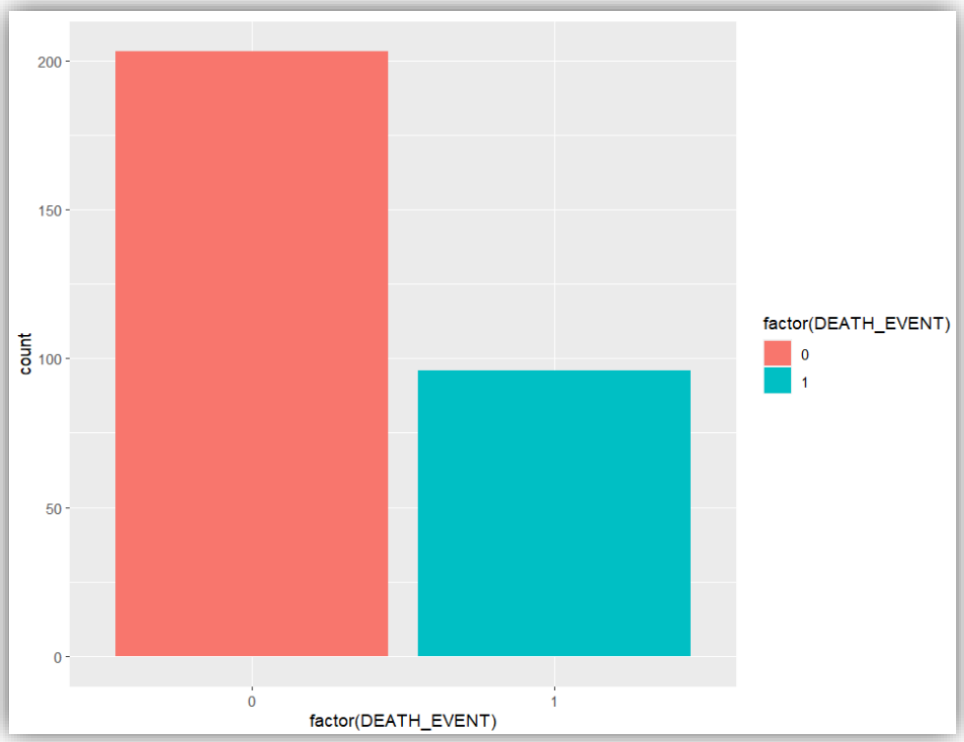
مصور سازی داده‌ها اساساً داده‌های تجزیه و تحلیل شده را در قالب تصاویری یعنی نمودارها و تصاویر قرار می‌دهد. این مصورسازی ها باعث می‌شود تا برای ما درک روند تجزیه و تحلیل از طریق تصاویر

آسان شود. از نمودارهای میله‌ای و میله‌ای تجمعی و نمودارهای جعبه‌ای در این بخش برای مصورسازی داده‌ها بهره می‌گیریم.

در ابتدا فراوانی مشاهدات در دو سطح متغیر پاسخ را بررسی می‌کنیم که دریابیم آیا تعادلی بین دو کلاس موجود است یا خیر.

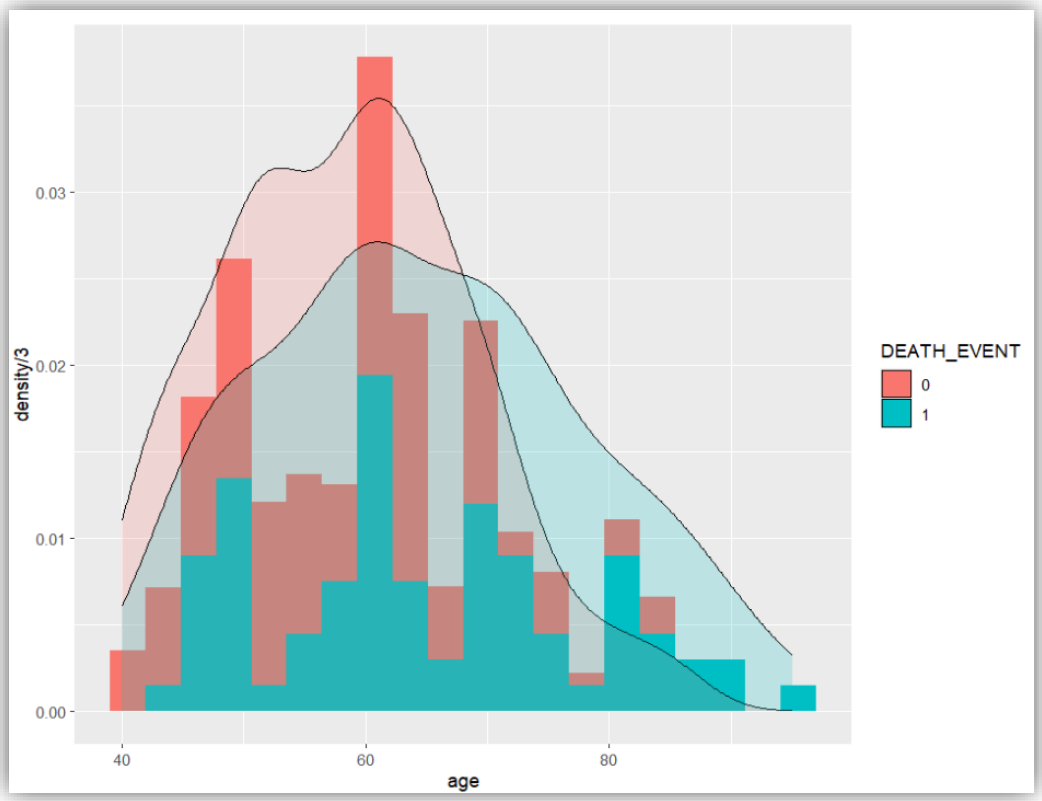
```
ggplot(Heart, aes(factor(DEATH_EVENT), fill = factor(DEATH_EVENT))) + geom_bar()
```

از نمودار پایین مشخص است که فراوانی طبقات با یکدیگر متفاوت است به‌طوری که فراوانی طبقه مربوط به سطح صفر متغیر پاسخ چند برابر فراوانی طبقه دیگر متغیر پاسخ می‌باشد.



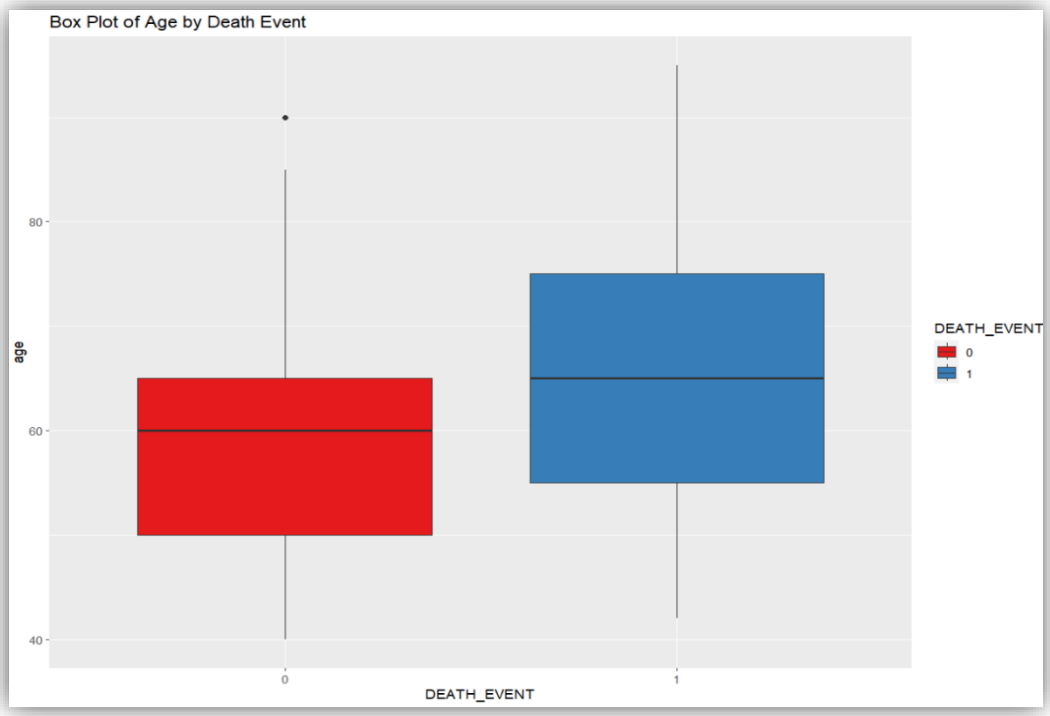
انتظار می‌رود سن بالاتر با مرگ و میر بیشتر همراه باشد. این فرض را می‌توان در هیستوگرام زیر با منحنی‌های چگالی همپوشانی تایید کرد یعنی بیماران مسن‌تر بیشتر از افراد جوان فوت می‌کردند.

```
Heart$DEATH_EVENT <- factor(Heart$DEATH_EVENT)
set.seed(2)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_not_sc <- train <- Heart[train_index,]
train_not_sc%<%

  ggplot(aes(age, fill = DEATH_EVENT))+

  theme_gray+ ()

  geom_histogram(aes(y = ..density../3), bins = 20)+

  geom_density(alpha = 0.2)
```



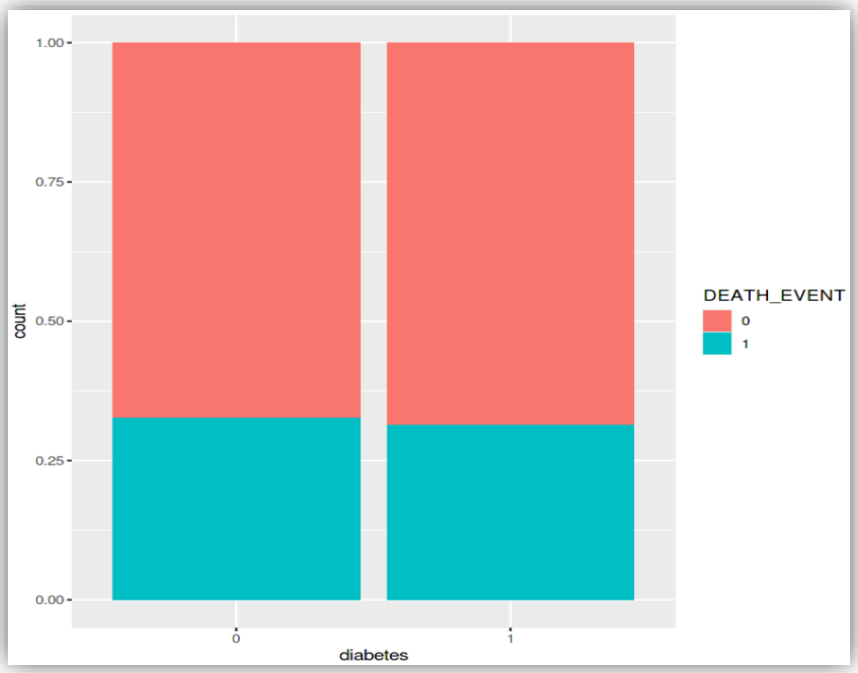
نمودارهای جعبه زیر بیشتر از این فرض حمایت می‌کنند.

```
ggplot(Heart, aes(x = DEATH_EVENT, y = age, fill = DEATH_EVENT)) +
  geom_boxplot() +
  labs(title = "Box Plot of Age by Death Event") +
  scale_fill_brewer(palette = "Set1")
```



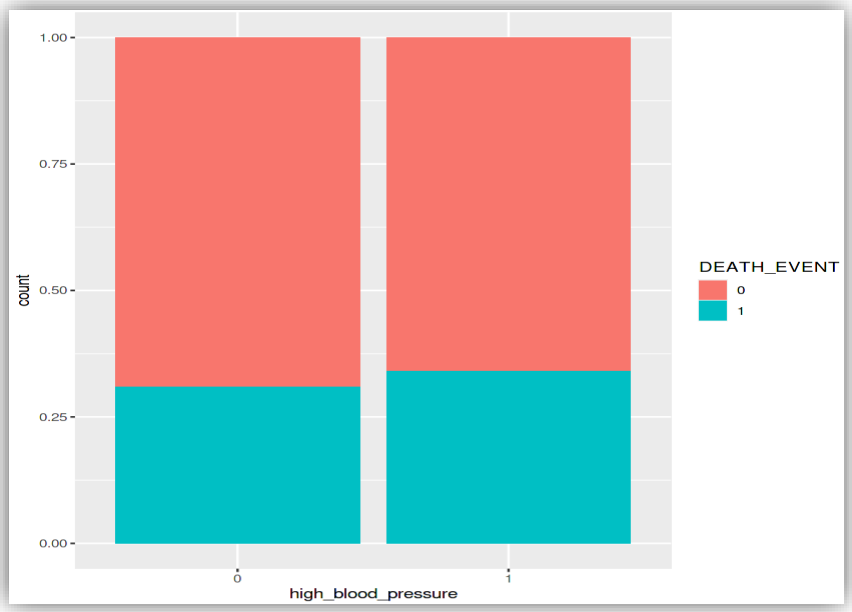
به نظر می‌رسد بیمارانی که می‌میرند بیشتر از بیمارانی که از بیمارستان مرخص می‌شوند، کم‌خونی دارند، همانطور که بر اساس نقشه حرارتی همبستگی انتظار می‌رود، تفاوت نسبتاً کوچک است.

```
train_not_sc %>%
  ggplot(aes(diabetes, fill = DEATH_EVENT)) +
  theme_gray() +
  geom_bar(position = "fill")
```

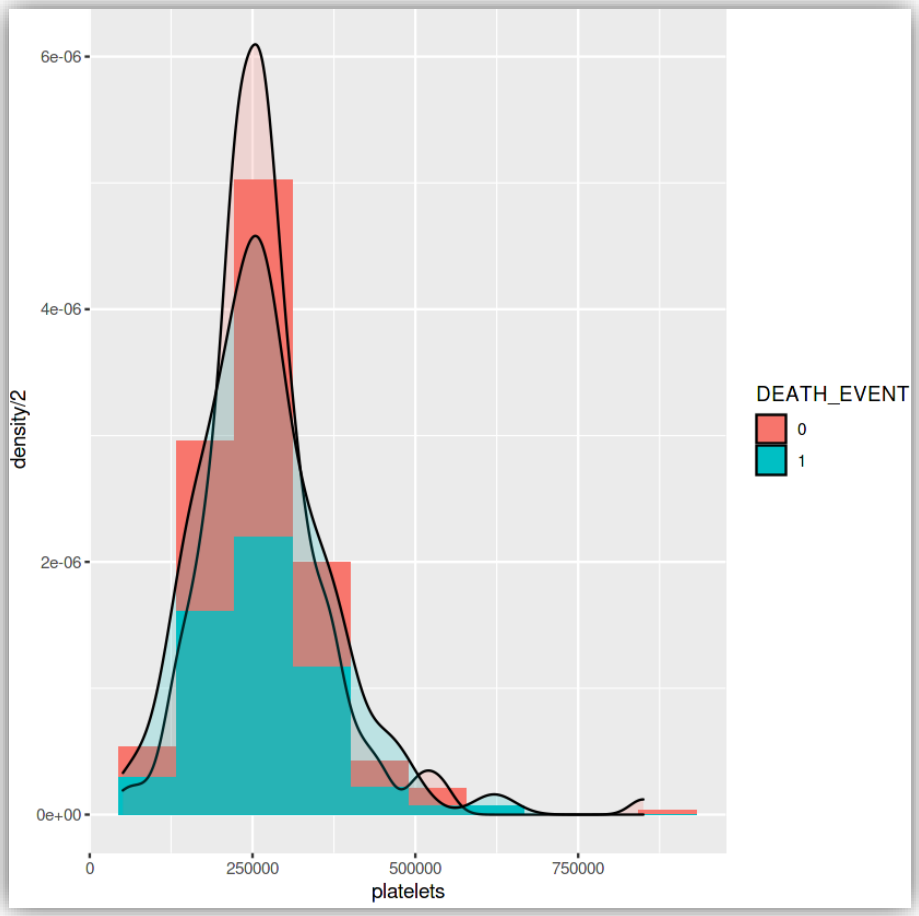


به نظر می‌رسد بیمارانی که می‌میرند بیشتر از بیمارانی که از بیمارستان مرخص می‌شوند فشار خون بالایی دارند.

```
train_not_sc %>%
  ggplot(aes(high_blood_pressure, fill = DEATH_EVENT)) +
  theme_gray() +
  geom_bar(position = "fill")
```

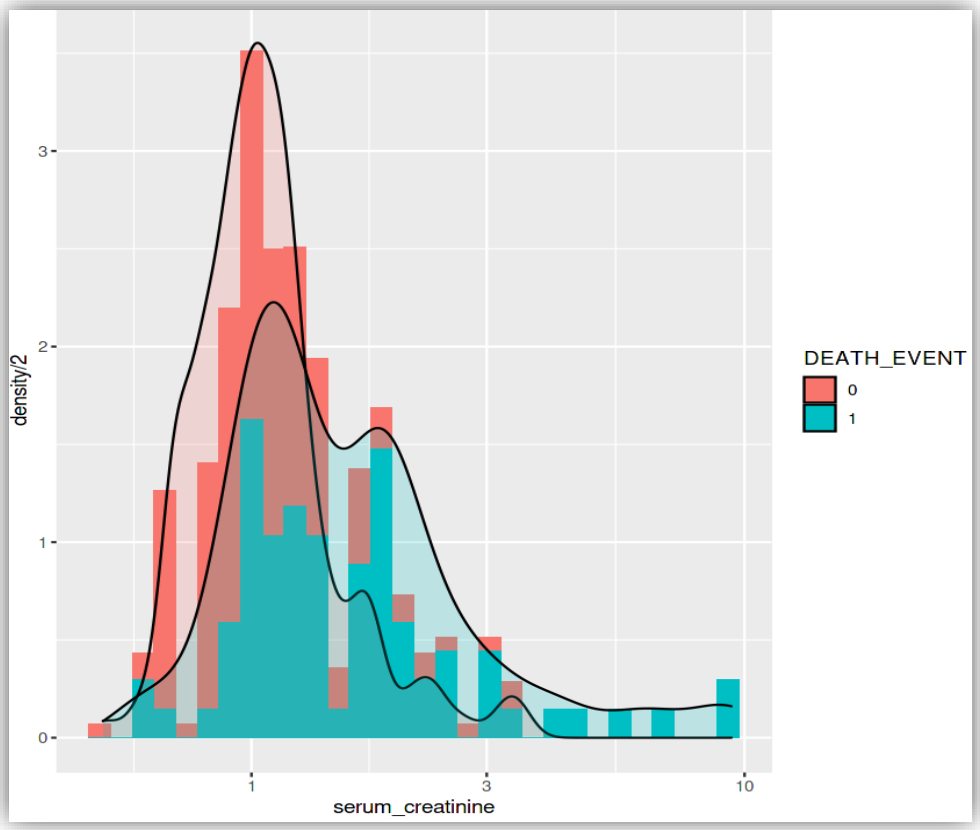


هیچ تفاوتی در سطوح پلاکتی برای هر دو گروه بیمار وجود ندارد، همانطور که در هیستوگرام زیر با منحنی‌های چگالی پوشانده نشان داده شده است.



برای کراتینین سرم و سدیم سرم همبستگی قابل توجهی در نقشه حرارتی نشان داده شد. هیستوگرام‌های زیر با منحنی‌های چگالی پوشانده شده تایید می کنند که سطوح بالاتر کراتینین در سرم و سطوح پایین‌تر سدیم در سرم با موارد بیشتر مرگ و میر بیماران مطابقت دارد.

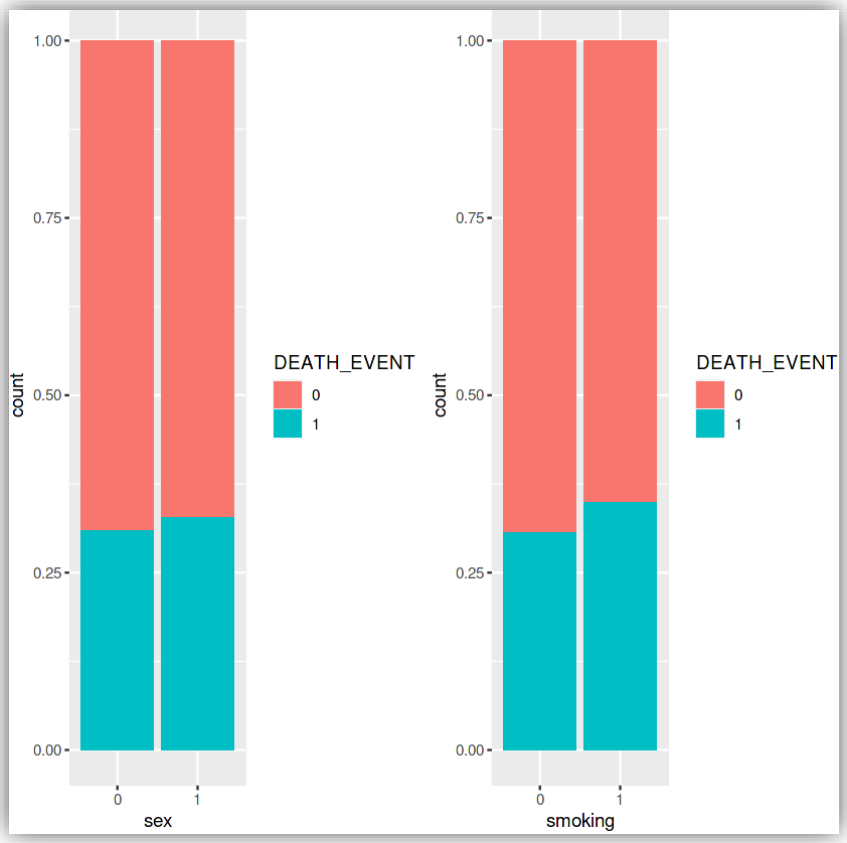
```
train_not_sc %>%
  ggplot(aes(serum_creatinine, fill = DEATH_EVENT)) +
  theme_gray() +
  geom_histogram(aes(y = ..density../2)) +
  geom_density(alpha = 0.2) +
  scale_x_log10()
```



جنسیت و سیگار کشیدن با مرگ و میر بیماران ارتباط کمی دارد. همانطور که مشاهده می شود، تفاوت معنی‌داری بین مرد و زن برای مرگ ناشی از نارسایی قلبی وجود ندارد، در حالی که در بین بیماران

سیگاری موارد مرگ و میر بیشتر از بیماران غیر سیگاری بود که به طور کلی می‌توان انتظار داشت.

```
library(gridExtra)
p1 <- train_not_sc %>%
  ggplot(aes(sex, fill = DEATH_EVENT)) +
  theme_gray() +
  geom_bar(position = "fill")
p2 <- train_not_sc %>%
  ggplot(aes(smoking, fill = DEATH_EVENT)) +
  theme_gray() +
  geom_bar(position = "fill")
grid.arrange(p1, p2, nrow = 1)
```



مراحل پیاده سازی الگوریتم‌های طبقه‌بندی دارای شش بخش اصلی به صورت زیر است:

1. حجم نمونه مورد نیاز را به وسیله روابط و فرمول‌هایی که در ادامه توضیح داده خواهد شد؛

بدست می‌آوریم.

2. با استفاده از حجم نمونه تعیین شده داده‌ها را به دو دسته آزمایشی و آموزشی تقسیم می‌کنیم.
3. مدل پیشنهادی را روی داده‌های آزمایشی برازش داده و از طریق آن فرضیه‌ای برای روابط موجود بین متغیرهای ورودی و خروجی تعیین می‌شود.
4. بعد از تشخیص روابط موجود در صورتی که برخی متغیرهای ورودی در ایجاد روابط و رساندن ما به هدف مورد نظر بی تاثیر شناخته شوند، برای افزایش دقت و کارایی مدل پیشنهادی ، از مدل حذف شده و مدل اصلاح شده مجددا روی داده‌های آموزشی برازش داده می‌شود.
5. از طریق مدل اصلاح شده در گام چهار، پیش‌بینی متغیر پاسخ برای داده‌های آموزشی انجام می‌شود.

برای ارزیابی کارایی و دقت مدل پیشنهادی و پی بردن به این موضوع که آیا این مدل متناسب با داده‌های مورد مطالعه هست یا خیر مقادیر پیش بینی شده در گام پنج با مقادیر اصلی متغیر پاسخ مقایسه شده و دقت مدل محاسبه می‌شود. از طریق محاسبه این دقت و عدد حالت محاسبه شده برای مدل می‌توان تشخیص داد پیش‌بینی متغیر پاسخ برای ورودی‌های جدید با چه دقتی انجام می‌شود و این مدل چه میزان توانایی در تشخیص مقادیر صحیح متغیر پاسخ را دارا می‌باشد.

داده‌های آموزشی و آزمایشی را ایجاد می‌کنیم (داده‌هایی که برای ساخت مدل مورد استفاده قرار خواهد گرفت و سپس داده‌هایی که برای آزمون مدل استفاده می‌شوند.)

```
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
```

پس از انجام این تقسیم بندی، عمل طبقه‌بندی را به چند روش شامل روش رگرسیون لجستیک، روش k-نزدیکترین همسایه (KNN)، روش ممیزی خطی (LDA)، روش ممیزی درجه دوم (QDA) و روش ماشین بردار پیشتیبان (SVM) انجام می‌دهیم. در تمامی این روش‌ها، برازش مدل را فقط برای داده‌های آموزشی انجام می‌دهیم و سپس با استفاده از داده‌های آزمایشی کیفیت مدل برازش شده در انجام عمل طبقه‌بندی را مورد بررسی قرار می‌دهیم. در نهایت نیز این مدل‌ها را با هم مقایسه می‌کنیم.

3 طبقه‌بندی به روش رگرسیون لجستیک

برخلاف نامش، رگرسیون لجستیک یک طبقه‌بندی کننده است! برای توضیح این مطلب ابتدا فرض کنیم فقط یک متغیر پیش‌بین (x) وجود دارد. متغیر پاسخ را نیز با y نشان می‌دهیم. در رگرسیون معمولی فرض بر این است که هر دو متغیر پیش‌بین و پاسخ کمی هستند. اما به عنوان مثال اگر متغیر پاسخ دارای مقادیر کیفی (و بخصوص دارای دو سطح) باشد، در این صورت روش رگرسیون خطی معمولی پاسخ‌گو نخواهد بود و باید از رگرسیون لجستیک استفاده شود. در این روش رگرسیونی، از مفهوم نسبت بخت استفاده می‌شود.

برای آشنایی با مفهوم نسبت بخت فرض کنید یک خانواده دارای شش فرزند هستند و نسبت دخترها به پسرها برابر با $\frac{2}{3}$ است. این نسبت نشان می‌دهد که تعداد دخترها در این خانواده دو برابر تعداد پسرها است. فرض کنید یکی از فرزندان این خانواده را به صورت تصادفی انتخاب کنیم. در این صورت احتمال انتخاب یک پسر برابر با $\frac{1}{3}$ و چنین احتمالی برای یک دختر برابر با $\frac{2}{3}$ است. حال فرض کنید پیشامد A پسر بودن جنسیت فرزند انتخاب شده باشد، در این صورت بخت یک چنین پیشامدی برابر است با

$$odd(A) = \frac{\mathrm{P}(A)}{1-\mathrm{P}(A)} = \frac{1/3}{2/3} = \frac{1}{2}$$

این عدد نشان می‌دهد که بخت انتخاب یک پسر نصف بخت انتخاب یک دختر است. برعکس اگر بخت این پیشامد که جنسیت فرزند انتخاب شده دختر باشد را محاسبه کنیم، خواهیم داشت

$$odd(A') = \frac{\mathrm{P}(A')}{1-\mathrm{P}(A')} = \frac{2/3}{1/3} = 2$$

این عدد نشان می‌دهد که بخت انتخاب یک دختر دو برابر بخت انتخاب یک پسر است. همان‌طور که دیده می‌شود مقدار بخت با مقدار احتمال تفاوت دارد، زیرا به عنوان مثال در اینجا مقدار بخت بزرگتر از 1 شده است.

لوجیت یک پیشامد را برابر با لگاریتم نسبت بخت آن پیشامد تعریف می‌کنند. در مثال بالا که پیشامد

A پسر بودن جنسیت فرزند انتخاب شده بود، لوجیت A به صورت زیر محاسبه می‌شود

$$logit(\mathrm{P}(A)) = \ln[odd(A)] = \ln\left[\frac{\mathrm{P}(A)}{1-\mathrm{P}(A)}\right] = \ln\left(\frac{1}{2}\right) = -\ln 2$$

بعد از این مقدمات، مجددا به رگرسیون لجستیک برمی‌گردیم. می‌دانیم که منظور از رگرسیون خطی معمولی، ایجاد رابطه‌ای خطی برای نمایش رابطه بین متغیرهای پیش‌بین و پاسخ است. این رابطه خطی به صورت زیر در نظر گرفته می‌شود.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

این رابطه معادله یک خط است که جمله خطا (ε) به آن اضافه شده است. پارامترهای این مدل خطی، عرض از مبدا (β_0) و شیب خط (β_1) هستند. اگر میانگین را با امید ریاضی جایگزین کنیم با فرض اینکه میانگین جمله خطا صفر است، خواهیم داشت

$$E(y|x) = \beta_0 + \beta_1 x$$

که در آن $E(y|x)$ نشان‌دهنده امید ریاضی y به شرط x است. اکنون اگر مقدار متغیر پاسخ باینری (دو وضعیتی) و دارای فقط دو مقدار صفر و یک باشد، مشخص است که توزیع y برنولی است، بنابراین امید ریاضی شرطی آن به صورت زیر محاسبه می‌شود

$$E(y|x) = \mathrm{P}(y = 1|x) = \pi(x)$$

به این ترتیب اگر بتوان برای تابع $\pi(x)$ یک الگو در نظر گرفت، آنگاه می‌توان یک مدل رگرسیونی با متغیر پاسخ برنولی مشخص کرد.

از آنجایی که مقدار پیش‌بینی برای متغیر پاسخ با اتکا بر تابع احتمال $\pi(x)$ انجام می‌شود، بنابراین برای مشخص کردن رابطه بین متغیر پاسخ و پیش‌بین به جای رابطه خطی، به تابعی احتیاج داریم که در بازه صفر تا یک قرار داشته باشد. در روش رگرسیون لجستیک، از تابعی به نام تابع لجستیک استفاده می‌شود. به همین علت این روش رگرسیونی را رگرسیون لجستیک می‌نامند. در رگرسیون لجستیک، زمانی که تنها یک متغیر پیش‌بین وجود دارد، $\pi(x)$ را به صورت زیر تعریف می‌کنند

$$\pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

$\pi(x)$ برای $\beta_1 > 0$ تابعی صعودی است و داریم

$$\lim_{x\to+\infty} \pi(x) = \lim_{x\to+\infty} \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = 1$$

$$\lim_{x\to-\infty} \pi(x) = \lim_{x\to-\infty} \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = 0$$

بنابراین برای $\beta_1 > 0$، $\pi(x)$ همواره در بازه $[0,1]$ قرار می‌گیرد.

به طور مشابه، $\pi(x)$ برای $\beta_1 < 0$ تابعی نزولی است و داریم

$$\lim_{x\to+\infty} \pi(x) = \lim_{x\to+\infty} \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = 0$$

$$\lim_{x\to-\infty} \pi(x) = \lim_{x\to-\infty} \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = 1$$

و بنابراین برای $\beta_1 < 0$، $\pi(x)$ همواره در بازه $[0,1]$ قرار می‌گیرد. برای حالت $\beta_1 = 0$ نیز داریم

$$\pi(x) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$

که باز هم در بازه $[0,1]$ قرار می‌گیرد. بنابراین به طور کلی $\pi(x)$ همواره در بازه $[0,1]$ قرار می‌گیرد.

به طور مشابه، اگر تعداد متغیرهای پیش‌بین برابر p باشد، در این صورت

$$E(y|x_1,\dots,x_p) = \mathrm{P}(y=1|x_1,\dots,x_p) = \pi(x_1,\dots,x_p)$$

و $\pi(x_1,\dots,x_p)$ به صورت زیر در نظر گرفته می‌شود

$$\pi(x_1,\dots,x_p) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

بنابراین مدل رگرسیون لجستیک با متغیر پاسخ y و متغیرهای پیش‌بین $x_1,\dots,x_p$ به صورت زیر تعریف می‌شود

$$E(y|x_1,\dots,x_p) = \mathrm{P}(y=1|x_1,\dots,x_p) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

به راحتی می‌توان نشان داد که $\pi(x_1,\dots,x_p)$ داده شده در بالا همواره در بازه $[0,1]$ قرار می‌گیرد. به منظور برآورد پارامترهای این مدل، می‌توان از تبدیل لوجیت استفاده کرد. این تبدیل روی بخت $\frac{\pi(x_1,\dots,x_p)}{1-\pi(x_1,\dots,x_p)}$ انجام می‌شود. در این صورت داریم

$$g(x_1,\dots,x_p)=\ln\left[\frac{\pi(x_1,\dots,x_p)}{1-\pi(x_1,\dots,x_p)}\right]=\ln\left[\frac{\frac{e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}}{1-\frac{e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}}\right]$$

$$=\ln\left[\frac{\frac{e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}}{\frac{1}{1+e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}}}\right]=\ln\left(e^{\beta_0+\beta_1x_1+\cdots+\beta_px_p}\right)=\beta_0+\beta_1x_1+\cdots+\beta_px_p$$

در نتیجه یک مدل رگرسیون لجستیک با متغیر پاسخ y و متغیرهای پیش‌بین $x_1,\dots,x_p$ می‌تواند به صورت زیر نیز نوشته شود

$$logit(\mathrm{P}(y=1|x_1,\dots,x_p))=\ln\left[\frac{\mathrm{P}(y=1|x_1,\dots,x_p)}{1-\mathrm{P}(y=1|x_1,\dots,x_p)}\right]=\beta_0+\beta_1x_1+\cdots+\beta_px_p$$

با استفاده از تابع درستنمایی و ماکسیمم‌سازی آن می‌توان برآورد پارامترهای مدل را بدست آورد. با این کار به یک دستگاه معادلات می‌رسیم که متاسفانه برای حل آن روشی تحلیلی وجود ندارد و باید به کمک روش‌های عددی برآورد پارامترهای مدل را به دست آورد. خوشبختانه نرم‌افزارهای زیادی از جمله R قادر هستند محاسبات مربوط به رگرسیون لجستیک را به خوبی انجام دهند.

مثال: از 20 دانشجو خواسته شده که در ابتدای روز اسمی را به خاطر بسپارند. متغیر پیش‌بین زمان خواب شبانه بر حسب دقیقه است. همچنین متغیر پاسخ «موفقیت یا عدم موفقیت در یادآوری اسم در صبح فردا» است. نتایج در جدول زیر آورده شده است. موفقیت در یادآوری اسم را با یک، و عدم موفقیت در یادآوری را با صفر نشان داده‌ایم. با استفاده از رگرسیون لجستیک سعی می‌کنیم الگو یا مدلی برای میزان خواب و احتمال یادآوری موفق بیابیم.

زمان خواب (x)	30	45	60	75	90	105	105	120	135	150
یادآوری (y)	0	0	0	0	0	1	0	1	1	0
زمان خواب (x)	165	180	195	210	240	255	270	285	300	330
یادآوری (y)	1	0	1	0	1	1	1	1	1	1

با توجه به اینکه فقط یک متغیر توضیحی داریم، هدف برازش یک مدل رگرسیون لجستیک به صورت

زیر به داده‌ها است

$$\text{logit}[\text{P}(y=1|x)] = \ln\frac{\text{P}(y=1|x)}{1-\text{P}(y=1|x)} = \beta_0 + \beta_1 x$$

برآورد پارامترهای این مدل در جدول زیر داده شده است (محاسبات با کامپیوتر انجام شده).

ضریب	برآورد	خطای معیار برآورد	آماره آزمون	P-value
β_0	-4.078	1.761	-2.316	0.021
β_1	0.025	0.010	2.393	0.017

برای هر یک از پارامترها P-value نیز دیده می‌شود که مقدار آنها کمتر از 0.05 است. در نتیجه فرض صفر بودن این پارامترها در سطح خطای 5 درصد رد می‌شود، از همین رو می‌توان مدل را مناسب یافت. پس احتمال اینکه فردی پس از x ساعت خواب بتواند اسم مورد نظر را به خاطر بیاورد از طریق تابع زیر محاسبه می‌شود

$$logit[\text{P}(y=1|x)] = \ln\left[\frac{\text{P}(y=1|x)}{1-\text{P}(y=1|x)}\right] = -4.078 + 0.025x$$

مدل بالا را به صورت زیر نیز می‌توان نوشت:

$$\text{P}(y=1|x) = \frac{e^{-4.078+0.025x}}{1+e^{-4.078+0.025x}}$$

حال مثلا برای دانشجویی که $x = 200$ دقیقه خواب داشته است، احتمال یادآوری اسم به صورت زیر بدست می‌آید:

$$\text{P}(y=1|x=200) = \frac{e^{-4.078+0.025(200)}}{1+e^{-4.078+0.025(200)}} = 0.72$$

یعنی احتمال یادآوری اسم برای این دانشجو 72 درصد است.

اکنون سوالی که مطرح می‌شود این است که رگرسیون لجستیک چگونه عمل طبقه‌بندی را انجام می‌دهد. به عنوان مثال، همین دانشجویی که $x = 200$ دقیقه خواب داشته است، به کدام رده طبقه-بندی می‌شود؟ به بیان دیگر، اگر $x = 200$، پیش‌بینی مقدار y چیست؟ با توجه به اینکه y فقط دو مقدار صفر و یک را اختیار می‌کند (موفقیت در یادآوری اسم=یک، و عدم موفقیت در یادآوری=صفر)،

بنابراین در واقع با داشتن $x = 200$، می‌خواهیم این دانشجو را به یکی از دو طبقه ممکنی که وجود دارد طبقه‌بندی کنیم (این دو طبقه را طبقه 1 و طبقه 0 می‌نامیم). در رگرسیون لجستیک اگر $\mathrm{P}(y = 1|x = 200)$ دارای مقداری بزرگتر از 0.5 باشد، در این صورت مشاهده (دانشجوی) مورد نظر به طبقه 1 و در غیر این صورت به طبقه 0 تعلق می‌گیرد. در بالا دیدیم که

$$\mathrm{P}(y = 1|x = 200) = 0.72 > 0.5$$

بنابراین دانشجو با میزان خواب $x = 200$ در طبقه 1 قرار می‌گیرد. ■

3-1 طبقه‌بندی داده‌های Heart با استفاده از روش رگرسیون لجستیک

ابتدا یک مدل رگرسیون لجستیک به داده‌های آموزشی برازش می‌دهیم. کدهای زیر این عمل را انجام می‌دهند

```
library(stats)
model1=DEATH_EVENT.~
fit1=glm(DEATH_EVENT~.,data=train_data,family=binomial)
summary(fit1)
```

در خروجی این کد خلاصه‌ای از داده‌ها را می‌بینیم؛ که درآن براورد پارامترهای مدل در ستون Estimate انحراف معیار متغیرها در ستون Std. Error و مقادیر z-value و P- value مشخص شده‌اند.

```
Call:
glm(formula = DEATH_EVENT ~ ., family = binomial, data = train_data)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              9.017e+00  6.965e+00   1.295 0.195463
age                      5.392e-02  1.783e-02   3.024 0.002497 **
anaemia                 -4.500e-01  4.131e-01  -1.089 0.275950
creatinine_phosphokinase 1.521e-04  1.740e-04   0.874 0.382105
diabetes                 2.785e-02  3.949e-01   0.071 0.943769
ejection_fraction       -9.787e-02  2.045e-02  -4.787 1.70e-06 ***
high_blood_pressure      7.654e-02  4.201e-01   0.182 0.855409
platelets                3.065e-07  1.995e-06   0.154 0.877875
serum_creatinine         7.311e-01  1.943e-01   3.762 0.000168 ***
serum_sodium            -6.097e-02  4.921e-02  -1.239 0.215286
sex                     -4.382e-01  4.574e-01  -0.958 0.338054
smoking                  3.083e-02  4.533e-01   0.068 0.945771
time                    -1.790e-02  3.256e-03  -5.498 3.85e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 301.20  on 239  degrees of freedom
Residual deviance: 174.75  on 227  degrees of freedom
AIC: 200.75

Number of Fisher Scoring iterations: 6
```

در این مرحله با توجه به P- value و مقدار $\alpha=0.05$ متغیرهای معنی‌دار را نگه داشته و بقیه را حذف می‌کنیم این نتیجه‌گیری را می‌توانستیم به کمک تابع confint() و محاسبه فواصل اطمینان هر یک از متغیرها نیز به‌دست آوریم :

```
confint(fit1)
```

```
                              2.5 %        97.5 %
(Intercept)             -4.436191e+00  2.301630e+01
age                      2.012040e-02  9.041956e-02
anaemia                 -1.280468e+00  3.481379e-01
creatinine_phosphokinase -1.658030e-04  5.379808e-04
diabetes                -7.511713e-01  8.049393e-01
ejection_fraction       -1.407325e-01 -6.012028e-02
high_blood_pressure     -7.531761e-01  9.029491e-01
platelets               -3.744229e-06  4.159811e-06
serum_creatinine         3.563700e-01  1.136574e+00
serum_sodium            -1.593966e-01  3.464258e-02
sex                     -1.349486e+00  4.540768e-01
smoking                 -8.609594e-01  9.267884e-01
time                    -2.474464e-02 -1.190381e-02
```

همان‌طور که در کد قبل نیز مشاهده شد متغیرهای anaemia و creatinine_phosphokinase و diabetes و high_blood_pressure و platelets و serum_sodium و sex و smoking به دلیل قرار گرفتن در بازه‌ای شامل مقدار صفر باید از مدل حذف شوند.

اکنون دوباره با استفاده از داده‌های آموزشی مدل را برازش می‌دهیم:

```
Heartnew=subset(Heart,select =c(age
,ejection_fraction,serum_creatinine,time,DEATH_EVENT))
Heartnew$DEATH_EVENT <- factor(Heartnew$DEATH_EVENT)
set.seed(123)
train_index <- createDataPartition(Heartnew$DEATH_EVENT, p = 0.8, list =
FALSE)
train_data <- Heartnew[train_index, ]
test_data <- Heartnew[-train_index, ]
head(train_data)
fit2=glm(DEATH_EVENT~.,data=train_data,family=binomial)
summary(fit2)
```

```
Call:
glm(formula = DEATH_EVENT ~ ., family = binomial, data = train_data)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.509178   1.145300   0.445   0.6566
age               0.051078   0.016824   3.036   0.0024 **
ejection_fraction -0.096476   0.019280  -5.004 5.62e-07 ***
serum_creatinine   0.751215   0.185199   4.056 4.99e-05 ***
time              -0.017643   0.003063  -5.760 8.42e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 301.20  on 239  degrees of freedom
Residual deviance: 179.95  on 235  degrees of freedom
AIC: 189.95

Number of Fisher Scoring iterations: 5
```

حال با اجرای کد زیر احتمال پیش‌بینی متغیرپاسخ برای هر یک از داده‌های آزمایشی تعیین می‌شود:

```
DEATH_EVENT.probs=predict(fit2,newdata=test_data,type="response")
head(DEATH_EVENT.probs)
```

```
        8        12        13        16        19        20
0.1729644 0.8536848 0.6331355 0.6504525 0.8965376 0.2345915
```

در ادامه اگر این احتمال از مقدار 0/5 بیشتر باشد، متغیرپاسخ مقدار(1) و در غیر این صورت مقدار(0) را اختیار می‌کند و سپس ماتریس در هم‌ریختگی مقدار واقعی و مقدار پیش‌بینی شده به عنوان خروجی نمایش داده می‌شود :

```
glm.pred=rep(0,length(DEATH_EVENT.probs))
glm.pred[DEATH_EVENT.probs>0.5]=1
glm.pred=factor(glm.pred)
test_data$DEATH_EVENT=as.factor(test_data$DEATH_EVENT)
confusionMatrix(glm.pred,test_data$DEATH_EVENT)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 36  5
         1  4 14

               Accuracy : 0.8475
                 95% CI : (0.7301, 0.9278)
    No Information Rate : 0.678
    P-Value [Acc > NIR] : 0.002611

                  Kappa : 0.6458

 Mcnemar's Test P-Value : 1.000000

            Sensitivity : 0.9000
            Specificity : 0.7368
         Pos Pred Value : 0.8780
         Neg Pred Value : 0.7778
             Prevalence : 0.6780
         Detection Rate : 0.6102
   Detection Prevalence : 0.6949
      Balanced Accuracy : 0.8184

       'Positive' Class : 0
```

که در آن سطرها مقادیر پیش‌گویی شده و ستون‌ها مقادیر واقعی متغیرپاسخ هستند؛ البته برای تشکیل این ماتریس روش دیگری غیر از استفاده از دستور confusionMatrix وجود دارد که در زیر کد آن را مشاهده می‌کنیم

```
table(DEATH_EVENT.probs > 0.5,test_data$DEATH_EVENT)
```

```
       0  1
FALSE 36  5
TRUE   4 14
```

برای 36 تا از داده‌ها، مقدار متغیر پاسخ (DEATH_EVENT) واقعا صفر است و مدل رگرسیون لجستیک نیز این 36 داده را به درستی به رده صفر طبقه‌بندی کرده است. همچنین برای 14 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است و مدل رگرسیون لجستیک نیز این 14 داده را به درستی به رده یک طبقه‌بندی کرده است. اما برای 5 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است اما مدل رگرسیون لجستیک این 5 داده را به غلط به رده صفر طبقه‌بندی کرده است. در نهایت برای 4 تا از داده‌ها، مقدار متغیر پاسخ واقعا صفر است اما مدل رگرسیون لجستیک این 4 داده را به غلط به رده یک طبقه‌بندی کرده است.

که با توجه به آن مقادیر TP , TN , FP, FN به ترتیب: 36، 14، 5، 4 هستند.

برای به دست آوردن دقت مدل با استفاده از رابطه زیر داریم:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{36 + 14}{36 + 14 + 5 + 4} = 0.847$$

مقدار به‌دست آمده برابر با 0/85 است و این یعنی این مدل با احتمال 0/85 مقادیر صحیح متغیر پاسخ را به ما می‌دهد.

حساسیت، نرخ مثبت را نمایش می‌دهد بنابراین برای این روش داریم:

$$Sensitivity(TPR) = \frac{TP}{TP + FN} = \frac{36}{36 + 4} = 0.9$$

خصوصیت (ویژگی) نرخ منفی را نشان می‌دهد؛ برای این روش داریم:

$$Specificity(TNR) = \frac{TN}{TN + FP} = \frac{14}{14 + 5} = 0.736$$

معیار صحت این روش:

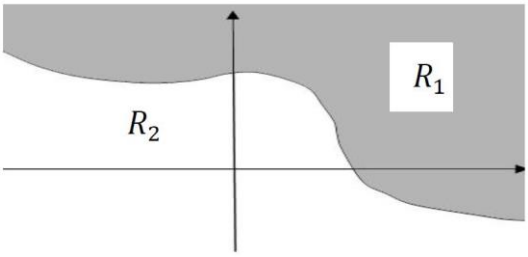
$$precision = \frac{TP}{TP + FP} = \frac{36}{36 + 5} = 0.878$$

$$F_1 = \frac{2TP}{2TP + FP + FN} = 0.88$$

4 طبقه‌بندی به روش ممیزی خطی

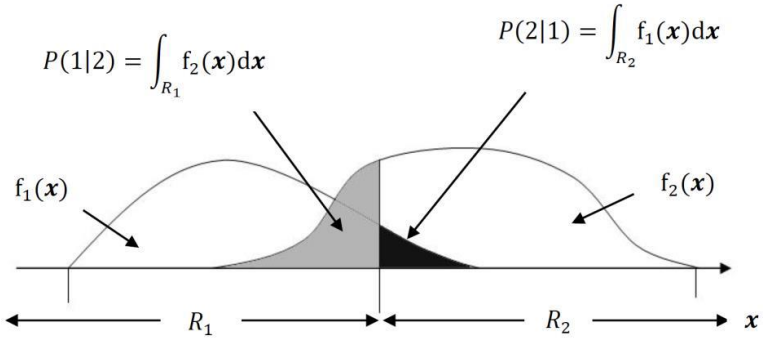
حالتی را در نظر می‌گیریم که متغیری که پاسخ فقط دارای دو مقدار (یا طبقه) است. می‌خواهیم با استفاده از طبقه‌بندی کننده ممیزی خطی تعیین کنیم که یک مشاهده جدید به کدام‌یک از این دو طبقه تعلق دارد.

در ادامه این بحث، یک بردار تصادفی p بُعدی را به صورت $\boldsymbol{X} = (X_1, \dots, X_p)$ و مقدار مشاهده شده‌ی متناظر با آن را با $\boldsymbol{x} = (x_1, \dots, x_p)$ نشان می‌دهیم. با این توضیح، فرض کنید جامعه‌ای به دو طبقه‌ی π_1, π_2 افراز شده باشد. ابتدا می‌خواهیم ناحیه‌ای تصادفی مانند R را به دو زیرناحیه R_1, R_2 به گونه‌ای افراز کنیم که اگر $\boldsymbol{x} \in R_1$ آنگاه $\boldsymbol{x}$ متعلق به طبقه π_1 و در غیر این صورت $\boldsymbol{x}$ متعلق به طبقه π_2 باشد (شکل زیر). سپس تعیین کنیم که یک مشاهده جدید به صورت $\boldsymbol{x} = (x_1, \dots, x_p)$، متعلق به کدام طبقه است و باید در کدام یک از نواحی تعریف شده قرار بگیرد. در اصطلاحات مربوط به طبقه‌بندی‌کننده ممیزی خطی، عمل اول را ممیزی و عمل دوم را طبقه‌بندی می‌گویند.



فرض کنید $f_1(\boldsymbol{x})$ و $f_2(\boldsymbol{x})$ به ترتیب نشان‌دهنده تابع چگالی بردار تصادفی $\boldsymbol{X} = (X_1, \dots, X_p)$ تحت طبقات π_1 و π_2 باشند. هدف تعیین طبقه‌ای است که مشاهده $\boldsymbol{x} = (x_1, \dots, x_p)$ به آن تعلق دارد. طبیعی است که در هر صورت احتمال طبقه‌بندی اشتباه وجود دارد. احتمال اینکه یک مشاهده مانند $\boldsymbol{x}$ را به π_2 طبقه‌بندی کنیم در حالی که در واقع متعلق به طبقه π_1 بوده است را با $P(2|1)$ و احتمال اینکه $\boldsymbol{x}$ را به π_1 طبقه‌بندی کنیم در حالی که در واقع متعلق به طبقه π_2 بوده است را با $P(1|2)$ نشان می‌دهیم. $P(2|1)$ و $P(1|2)$ احتمال‌های طبقه‌بندی اشتباه یا احتمال‌های خطا هستند. چون

تنها دو طبقه وجود دارد، بنابراین تنها همین دو خطا را داریم (شکل زیر).



با توجه به آنچه که از درس احتمال می‌دانیم، در صورتی که $\boldsymbol{X}$ یک بردار تصادفی پیوسته باشد، داریم:

$$P(2|1) = P(\boldsymbol{X} \in R_2|\pi_1) = \int_{R_2} f_1(\boldsymbol{x})d\boldsymbol{x}$$

به طور مشابه برای $P(1|2)$ داریم

$$P(1|2) = P(\boldsymbol{X} \in R_1|\pi_2) = \int_{R_1} f_2(\boldsymbol{x})d\boldsymbol{x}$$

حال فرض کنید احتمال‌های پیشین، یا احتمال اینکه از قبل بدانیم که مشاهده $\boldsymbol{x}$ متعلق به طبقه π_1 و π_2 است به صورت زیر باشند:

$$p_1 = P(\pi_1), p_2 = P(\pi_2)$$

بدیهی است که $p_1 + p_2 = 1$ (در صورتی که مقادیر p_1, p_2 را ندانیم، هر دو برابر با 0.5 در نظر گرفته می‌شوند). با این توضیحات احتمالات طبقه‌بندی زیر را داریم:

$$P\left(\text{مشاهده } \boldsymbol{x} \text{ به طور صحیح در } \pi_1 \text{ طبقه‌بندی شود}\right) = P(\boldsymbol{X} \in R_1|\pi_1)P(\pi_1) = P(1|1)p_1$$

$$P\left(\text{مشاهده } \boldsymbol{x} \text{ به اشتباه در } \pi_1 \text{ طبقه‌بندی شود}\right) = P(\boldsymbol{X} \in R_1|\pi_2)P(\pi_2) = P(1|2)p_2$$

$$P\left(\text{مشاهده } \boldsymbol{x} \text{ به طور صحیح در } \pi_2 \text{ طبقه‌بندی شود}\right) = P(\boldsymbol{X} \in R_2|\pi_2)P(\pi_2) = P(2|2)p_2$$

$$P\left(\text{مشاهده } \boldsymbol{x} \text{ به اشتباه در } \pi_2 \text{ طبقه‌بندی شود}\right) = P(\boldsymbol{X} \in R_2|\pi_1)P(\pi_1) = P(2|1)p_1$$

در صورتی که $c(i|j), i, j = 1,2$ نیز هزینه (یا زیانی) باشد که در ازای طبقه‌بندی اشتباهِ مشاهده $\boldsymbol{x}$ به طبقه π_i (در حالی که در حقیقت متعلق به طبقه π_j بوده) می‌پردازیم، در این صورت جدول زیر را

خواهیم داشت (در صورتی که بحث هزینه و زیان مطرح نباشد، در فرمول‌هایی که در ادامه ارائه می‌شوند می‌توان $c(1|2)$ و $c(2|1)$ را با هم برابر در نظر گرفت و در نتیجه با هم حذف می‌شوند).

دسته‌بندی / جمعیت	π_1	π_2
π_1	0	$c(2\|1)$
π_2	$c(1\|2)$	0

با توجه به توضیحات بالا می‌توان متوسط زیان (که به آن ریسک یا مخاطره نیز گفته می‌شود) را محاسبه کرد. متوسط زیانِ طبقه‌بندی اشتباه به صورت زیر است

$$ECM = c(2|1)P(2|1)p_1 + c(1|2)P(1|2)p_2 = P(2|1)p_1' + P(1|2)p_2'$$

که در آن

$$p_1' = c(2|1)p_1, p_2' = c(1|2)p_2$$

یک طبقه‌بندی منطقی را می‌توان با مینیمم کردن ECM به دست آورد. توجه کنید که

$$ECM = P(2|1)p_1' + P(1|2)p_2' = p_1' \int_{R_2} f_1(\boldsymbol{x})d\boldsymbol{x} + p_2' \int_{R_1} f_2(\boldsymbol{x})d\boldsymbol{x} =$$

$$= p_1' \left[1 - \int_{R_1} f_1(\boldsymbol{x})d\boldsymbol{x}\right] + p_2' \int_{R_1} f_2(\boldsymbol{x})d\boldsymbol{x}$$

$$= p_1' + p_2' \int_{R_1} [p_2' f_2(\boldsymbol{x}) - p_1' f_1(\boldsymbol{x})]d\boldsymbol{x}$$

چون مقادیر p_1', p_2' ثابت‌اند، بنابراین مینیمم کردن ECM معادل با مینیمم‌سازی انتگرال زیر است

$$\int_{R_1} [p_2' f_2(\boldsymbol{x}) - p_1' f_1(\boldsymbol{x})]d\boldsymbol{x}$$

انتگرال فوق مینیمم می‌شود اگر ناحیه R_1 شامل مقادیری از $\boldsymbol{x}$ باشد که مقدار عبارت داخل انتگرال یعنی $[p_2' f_2(\boldsymbol{x}) - p_1' f_1(\boldsymbol{x})]$ منفی باشد.پس ECM مینیمم می‌شود اگر R_1 شامل مقادیری از $\boldsymbol{x}$ باشدکه

$$p_2' f_2(\boldsymbol{x}) - p_1' f_1(\boldsymbol{x}) \leq 0 \Leftrightarrow \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \frac{c(1|2)}{c(2|1)} \frac{p_2}{p_1}$$

بنابراین نواحی طبقه‌بندی به صورت زیر مشخص می‌شوند

$$R_1 = \left\{\boldsymbol{x}: \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right\};\ R_2 = \left\{\boldsymbol{x}: \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} < \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right\}$$

احتمال کلی برای طبقه‌بندی اشتباه نیز به صورت زیر است

$$TPM = P\left(\text{طبقه‌بندی اشتباه}\right) = P(2|1)p_1 + P(1|2)p_2$$

بنابراین با توجه به قاعده فوق، اگر بخواهیم برای یک مشاهده جدید مانند $\boldsymbol{x_0}$ تصمیم‌گیری کنیم که به کدام طبقه تعلق می‌گیرد به صورت زیر عمل می‌کنیم

$\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر $\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$

و به طبقه π_2 تعلق می‌گیرد اگر $\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} < \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$

در حالتی که توزیع هر دو جامعه نرمال چندمتغیره با ماتریس کواریانس‌های برابر باشد، یعنی π_1 دارای توزیع $N_p(\boldsymbol{\mu_1}, \Sigma)$ و π_2 نیز دارای توزیع $N_p(\boldsymbol{\mu_2}, \Sigma)$ در این صورت $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

$$\Leftrightarrow \frac{(2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x_0}-\boldsymbol{\mu_1})^T\Sigma^{-1}(\boldsymbol{x_0}-\boldsymbol{\mu_1})\right]}{(2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x_0}-\boldsymbol{\mu_2})^T\Sigma^{-1}(\boldsymbol{x_0}-\boldsymbol{\mu_2})\right]} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

$$\Leftrightarrow (\boldsymbol{\mu_1}-\boldsymbol{\mu_2})^T\Sigma^{-1}\boldsymbol{x_0} - \frac{1}{2}(\boldsymbol{\mu_1}-\boldsymbol{\mu_2})^T\Sigma^{-1}(\boldsymbol{\mu_1}+\boldsymbol{\mu_2}) \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد.

در عمل معمولا $\boldsymbol{\mu_1}$، $\boldsymbol{\mu_2}$ و Σ نامعلوم هستند و لذا استفاده از موارد بالا امکان‌پذیر نیست. بنابراین از نمونه‌های آموزشی استفاده کرده و این پارامترها را برآورد می‌کنیم. $\boldsymbol{\mu_1}$ و $\boldsymbol{\mu_2}$ با استفاده از $\bar{\boldsymbol{x}}_1$ و $\bar{\boldsymbol{x}}_2$ برآورد می‌شوند که $\bar{\boldsymbol{x}}_1$ و $\bar{\boldsymbol{x}}_2$ به ترتیب میانگین نمونه‌های (آموزشی) مربوط به طبقات π_1 و π_2 هستند. ماتریس Σ را نیز با استفاده از S_p برآورد کرده و در روابط بالا قرار می‌دهیم. S_p ماتریس کواریانس آمیخته نامیده می‌شود و به صورت زیر تعریف می‌شود

$$S_p = \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2}{n_1 + n_2 - 2}$$

S_1 و S_2 نیز به ترتیب ماتریس‌های کواریانس نمونه‌های (آموزشی) مربوط به طبقات π_1 و π_2 هستند. همچنین n_1 و n_2 نیز به ترتیب حجم نمونه‌های (آموزشی) مربوط به طبقات π_1 و π_2 هستند. در نتیجه قاعده تصمیم به صورت زیر خواهد بود

$\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_p^{-1} \boldsymbol{x_0} - \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_p^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2) \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد.

تابع $L(\boldsymbol{x_0}) = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_p^{-1} \boldsymbol{x_0} - \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_p^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)$ معمولا تابع ممیزی خطی اندرسون نامیده می‌شود. دلیل اینکه به آن خطی می‌گویند این است که $L(\boldsymbol{x_0})$ یک تابع خطی نسبت به $\boldsymbol{x_0}$ است.

یک تابع ممیزی خطی معروف دیگر وجود دارد که تابع ممیزی خطی فیشر نامیده می‌شود. با استفاده از تابع ممیزی خطی فیشر، $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^T S_p^{-1} \boldsymbol{x_0} \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد.

توجه به این نکته ضروری است که تنها در صورتی می‌توان از توابع ممیزی خطی اندرسون و فیشر استفاده کرد که توزیع هر دو جامعه نرمال چندمتغیره با ماتریس کواریانس‌های برابر باشد. با این حال اگر حجم‌های نمونه‌های (آموزشی) گرفته شده از دو جامعه بزرگ باشند (که در عمل معمولا چنین است)، می‌توان از قضیه حد مرکزی (حالت چند متغیره) استفاده کرده و فرمول‌های داده شده در بالا را بدون نیاز به برقرار بودن فرض نرمال به کار گرفت.

مثال: مجددا مثال قبل را در نظر می‌گیریم. از 20 دانشجو خواسته شده که در ابتدای روز اسمی را به

خاطر بسپارند. متغیر پیش‌بین زمان خواب شبانه بر حسب دقیقه است. همچنین متغیر پاسخ «موفقیت یا عدم موفقیت در یادآوری اسم در صبح فردا» است. نتایج در جدول زیر آورده شده است. موفقیت در یادآوری اسم را با یک، و عدم موفقیت در یادآوری را با صفر نشان داده‌ایم.

زمان خواب (x)	30	45	60	75	90	105	105	120	135	150
یادآوری(y)	0	0	0	0	0	0	1	0	1	0
زمان خواب (x)	165	180	195	210	240	255	270	285	300	330
یادآوری (y)	1	0	1	0	1	1	1	1	1	1

با استفاده از این داده‌ها به عنوان داده‌های آموزشی، تابع ممیز خطی اندرسون را مشخص می‌کنیم. ابتدا محاسبات زیر را داریم

$$n_1 = n_2 = 10$$

$$\bar{x}_1 = \frac{1}{10}(30 + 45 + \cdots + 210) = 106.5$$

$$\bar{x}_2 = \frac{1}{10}(105 + 135 + \cdots + 330) = 228$$

توجه می‌کنیم که n_1 تعداد داده‌هایی است که برای آنها مقدار متغیر y برابر با صفر است و n_2 نیز تعداد داده‌هایی است که برای آنها مقدار متغیر y برابر با یک است. همچنین برای محاسبه $\bar{x}_1$ فقط از x هایی استفاده می‌کنیم که برای آنها مقدار متغیر y برابر با صفر است و برای محاسبه $\bar{x}_2$ نیز فقط از xهایی استفاده می‌کنیم که برای آنها مقدار متغیر y برابر با یک است. واریانس‌ها نیز به طور مشابه تعریف می‌شوند و محاسبه آنها نیز به صورت زیر است

$$S_1^2 = \frac{1}{9}[(30 - 106.5)^2 + (45 - 106.5)^2 + \cdots + (210 - 106.5)^2] = 3472.5$$

$$S_2^2 = \frac{1}{9}[(105 - 228)^2 + (135 - 228)^2 + \cdots + (330 - 228)^2] = 5590$$

توجه می‌کنیم که چون در این مثال تنها یک ویژگی (یک متغیر x) وجود دارد، بنابراین میانگین‌ها و واریانس‌ها به جای بردار و ماتریس، به صورت عدد هستند. ضمنا به همین دلیل واریانس‌ها را به جای

S_1 و S_2 با S_1^2 و S_2^2 نشان داده‌ایم. واریانس آمیخته به صورت زیر است

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} = 4529.3$$

بنابراین تابع ممیز خطی اندرسون به صورت زیر است (چون تنها یک متغیر x داریم لذا نمادهای ماتریسی حذف شده است)

$$L(x_0) = (\bar{x}_1 - \bar{x}_2)\frac{1}{S_p^2}x_0 - \frac{1}{2}(\bar{x}_1 - \bar{x}_2)\frac{1}{S_p^2}(\bar{x}_1 + \bar{x}_2) = -0.0268x_0 + 4.4865$$

از طرفی با فرض $c(1|2) = c(2|1)$ و $p_1 = p_2$ داریم

$$\ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right] = 0$$

بنابراین یک مشاهده مانند x_0 به طبقه π_1 (یعنی حالتی که y=0) تعلق می‌گیرد اگر

$$L(x_0) = -0.0268x_0 + 4.4865 \geq 0$$

و در غیر این صورت به طبقه π_2 (یعنی حالتی که y=1) تعلق می‌گیرد.

حال مثلا برای دانشجویی که $x_0 = 200$ دقیقه خواب داشته است داریم

$$L(200) = -0.0268(200) + 4.4865 = -0.8735 < 0$$

بنابراین دانشجو با میزان خواب $x = 200$ در طبقه π_2 (یعنی حالتی که y=1) قرار می‌گیرد.

لازم به ذکر است که در این مثال تعداد نمونه‌ها کوچک است و بنابراین قضیه حد مرکزی معتبر نیست.

بنابراین لازم است که فرض کنیم نمونه‌های مربوط به هر دو جامعه (یا هر دو طبقه) نرمال هستند. ■

4-1 طبقه‌بندی داده‌های Heart با استفاده از روش ممیزی خطی

یکی از مفروضات کلیدی تحلیل تفکیک خطی این است که هر یک از متغیرهای پیش‌بینی واریانس یکسانی دارند. یک راه آسان برای اطمینان از برآورده شدن این فرض این است که هر متغیر را به گونه‌ای مقیاس‌بندی کنیم که میانگین آن 0 و انحراف استاندارد 1 باشد. برای این کار از تابع scale استفاده می‌کنیم.

```
Heart[1:12]=scale(Heart[1:12])
```

```
           age    anaemia creatinine_phosphokinase   diabetes ejection_fraction high_blood_pressure
[1,]  1.1909487 -0.8696469              0.000165451 -0.8461608      -1.527997920           1.3569966
[2,] -0.4904571 -0.8696469              7.502062717 -0.8461608      -0.007064906          -0.7344569
[3,]  0.3502458 -0.8696469             -0.449185725 -0.8461608      -1.527997920          -0.7344569
[4,] -0.9108085  1.1460462             -0.485257493 -0.8461608      -1.527997920          -0.7344569
[5,]  0.3502458  1.1460462             -0.434757017  1.1778559      -1.527997920          -0.7344569
[6,]  2.4520030  1.1460462             -0.551217299 -0.8461608       0.161927651           1.3569966
        platelets serum_creatinine serum_sodium        sex   smoking      time
[1,]  1.678834e-02       0.48923681  -1.50151891  0.7344569 -0.686531 -1.626775
[2,]  7.523048e-09      -0.28407611  -0.14173853  0.7344569 -0.686531 -1.601007
[3,] -1.036336e+00      -0.09074788  -1.72814897  0.7344569  1.451727 -1.588122
[4,] -5.455595e-01       0.48923681   0.08489153  0.7344569 -0.686531 -1.588122
[5,]  6.507077e-01       1.26254973  -4.67433977 -1.3569966 -0.686531 -1.575238
[6,] -6.069065e-01       0.68256504  -1.04825878  0.7344569  1.451727 -1.575238
```

می‌توانیم از تابع application برای تأیید اینکه هر متغیر پیش‌بینی‌کننده اکنون میانگین 0 و انحراف استاندارد 1 دارد استفاده کنیم.

```
apply(Heart,2,mean)
apply(Heart,2,sd)
```

```
> apply(Heart1,2,mean)
                    age                 anaemia creatinine_phosphokinase                diabetes
          -2.660248e-17            5.631433e-17           -1.329598e-17            4.306313e-17
      ejection_fraction      high_blood_pressure                platelets          serum_creatinine
          -2.479052e-17           -1.810690e-17            8.978064e-17           -7.908394e-17
           serum_sodium                     sex                  smoking                    time
          -8.824202e-16            1.785307e-17            2.197268e-17           -1.798608e-16
> apply(Heart1,2,sd)
                    age                 anaemia creatinine_phosphokinase                diabetes
                      1                       1                       1                       1
      ejection_fraction      high_blood_pressure                platelets          serum_creatinine
                      1                       1                       1                       1
           serum_sodium                     sex                  smoking                    time
                      1                       1                       1                       1
```

مجموعه داده آموزشی و آزمایشی را توسط کد زیر ایجاد می‌کنیم.

```
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
```

در مرحله بعد، از تابع lda() از بسته MASS برای تطبیق مدل LDA به داده‌های خود استفاده می‌کنیم:

```
linear <- lda(DEATH_EVENT~., train_data)
```

```
Call:
lda(DEATH_EVENT ~ ., data = train_data)

Prior probabilities of groups:
        0         1
0.6791667 0.3208333

Group means:
          age    anaemia creatinine_phosphokinase    diabetes ejection_fraction high_blood_pressure
0 -0.1515971 0.02072065              -0.03730999  0.02304880         0.2028798         -0.08007576
1  0.4011979 0.04657728               0.13935295 -0.05758289        -0.4888034          0.08039510
    platelets serum_creatinine serum_sodium        sex     smoking       time
0  0.01969400       -0.2073972    0.1905595 0.02875175 0.008730807  0.3485446
1 -0.01403599        0.5036737   -0.3212766 0.05541357 0.035478197 -0.7135002

Coefficients of linear discriminants:
                                LD1
age                       0.335166776
anaemia                  -0.121122966
creatinine_phosphokinase  0.112853786
diabetes                  0.007868191
ejection_fraction        -0.628940639
high_blood_pressure       0.017710121
platelets                 0.051298169
serum_creatinine          0.403763885
serum_sodium             -0.142621133
sex                      -0.111777411
smoking                  -0.002669801
time                     -0.777928122
```

احتمال پیشین هر رده (Prior probabilities of groups):

نشان‌دهنده نسبت هر رده در مجموعه آموزشی است. 0.68 درصد در رده 0 و 0.32 درصد در رده 1 قرار دارند.

میانگین هر رده (Group means): این مقادیر میانگین را برای هر متغیر پیش‌بینی‌کننده برای هر رده نمایش می‌دهد.

ضرایب متمایز کننده‌های خطی: این ضرایب ترکیب خطی متغیرهای پیش‌بینی‌کننده را نشان می‌دهد

که برای تشکیل قانون تصمیم گیری مدل LDA استفاده می‌شود. مثلا:

LD1: 0.33*age - 0.12*anaemia +0.11*creatinine + 0.01*diabetes - 0.63ejection_fraction+0.02*high_blood_pressure+0.05*platelets+0.43*serum_creatinine-0.14*serum_sodium-0.11*sex-0.002*smoking-0.78*time

هنگامی که مدل را با استفاده از داده‌های آموزشی خود منطبق کردیم، می‌توانیم از آن برای پیش‌بینی داده‌های آزمایشی خود استفاده کنیم:

```
p <- predict(linear, test_data)
names(p)
```

```
> names(p)
[1] "class"     "posterior" "x"
```

رده(class): رده پیش‌بینی شده

احتمال پسین(posterior): احتمال پسینی که یک مشاهده متعلق به هر رده است.

متمایز کننده‌های خطی(x)

مشاهده رده پیش‌بینی شده برای شش مشاهده اول در مجموعه آزمون

```
head(p$class)
```

```
> head(p$class)
[1] 0 1 1 0 1 0
Levels: 0 1
```

مشاهده احتمالات پسین برای شش مشاهده اول در مجموعه آزمون

```
head(p$posterior)
```

```
> head(p$posterior)
           0         1
8  0.8184157 0.1815843
12 0.1700648 0.8299352
13 0.4812413 0.5187587
16 0.5024709 0.4975291
19 0.1268368 0.8731632
20 0.6057728 0.3942272
```

مشاهده متمایزهای خطی برای شش مشاهده اول در مجموعه آزمایشی

```
head(p$x)
> head(p$x)
           LD1
8  -0.08879961
12  1.60867354
13  0.77932983
16  0.73267336
19  1.79762696
20  0.50217816
```

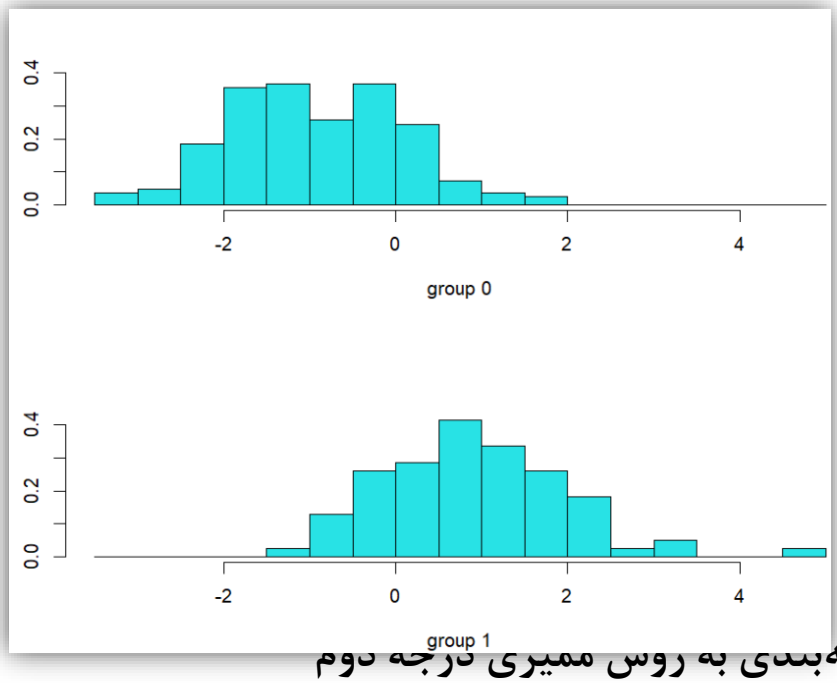
می‌توانیم از کد زیر استفاده کنیم تا ببینیم مدل LDA برای چند درصد از مشاهدات رده‌ها را به درستی پیش‌بینی کرده است:

```
mean(p$class==test_data$DEATH_EVENT)
[1] 0.7966102
```

مدل به درستی رده‌ها را برای 79٪ مشاهدات در مجموعه داده آزمایشی ما پیش‌بینی کرده است.

در نهایت، ما می‌توانیم یک نمودار LDA برای مشاهده تمایزهای خطی مدل ایجاد کنیم و تجسم کنیم که چگونه سه رده مختلف در مجموعه داده ما را از هم جدا کرده است:

```
plot(linear)
```



5 طبقه‌بندی به روش ممیزی درجه دوم

در بخش قبل دیدیم که زمانی که متغیر برچسب (y) تنها دارای دو سطح باشد (یا به عبارت دیگر تنها

دو جامعه یا دو طبقه وجود داشته باشد)، مشاهده جدید $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

و به طبقه π_2 تعلق می‌گیرد اگر

$$\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} < \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

این قاعده تصمیم کلی است (یعنی خطی یا درجه دوم نیست) و مادامی که تابع چگالی دو جامعه مشخص باشد می‌توان از آن استفاده کرد. همچنین دیدیم در حالتی که توزیع هر دو جامعه نرمال چندمتغیره با ماتریس کواریانس‌های برابر باشد، یعنی π_1 دارای توزیع $N_p(\boldsymbol{\mu_1}, \Sigma)$ و π_2 نیز دارای توزیع $N_p(\boldsymbol{\mu_2}, \Sigma)$ باشد، در این صورت $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$(\overline{\boldsymbol{x}}_{\boldsymbol{1}} - \overline{\boldsymbol{x}}_{\boldsymbol{2}})^T S_p^{-1}\boldsymbol{x_0} - \frac{1}{2}(\overline{\boldsymbol{x}}_{\boldsymbol{1}} - \overline{\boldsymbol{x}}_{\boldsymbol{2}})^T S_p^{-1}(\overline{\boldsymbol{x}}_{\boldsymbol{1}} + \overline{\boldsymbol{x}}_{\boldsymbol{2}}) \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد.

اکنون موقعیتی را در نظر بگیرید که باز هم هر دو جامعه دارای توزیع نرمال چندمتغیره هستند اما ماتریس کواریانس دو جامعه (یا دو طبقه) با هم برابر نیست. یعنی فرض کنید π_1 دارای توزیع $N_p(\boldsymbol{\mu_1}, \Sigma_1)$ و π_2 نیز دارای توزیع $N_p(\boldsymbol{\mu_2}, \Sigma_2)$ باشد. در این صورت طبق قاعده کلی که داشتیم، $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$\frac{f_1(\boldsymbol{x_0})}{f_2(\boldsymbol{x_0})} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

$$\Leftrightarrow \frac{(2\pi)^{-p/2}|\Sigma_1|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x_0}-\boldsymbol{\mu_1})^T\Sigma_1^{-1}(\boldsymbol{x_0}-\boldsymbol{\mu_1})\right]}{(2\pi)^{-p/2}|\Sigma_2|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x_0}-\boldsymbol{\mu_2})^T\Sigma_2^{-1}(\boldsymbol{x_0}-\boldsymbol{\mu_2})\right]} \geq \frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}$$

$$\Leftrightarrow -\frac{1}{2}\boldsymbol{x_0^T}(\Sigma_1^{-1} - \Sigma_2^{-1})\boldsymbol{x_0} + (\boldsymbol{\mu}_1^T\Sigma_1^{-1} - \boldsymbol{\mu}_2^T\Sigma_2^{-1})\boldsymbol{x_0}$$
$$-\frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) - \frac{1}{2}(\boldsymbol{\mu}_1^T\Sigma_1^{-1}\boldsymbol{\mu_1} - \boldsymbol{\mu}_2^T\Sigma_2^{-1}\boldsymbol{\mu_2}) \geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد.

در عمل معمولا $\boldsymbol{\mu_1}$، $\boldsymbol{\mu_2}$ و Σ_1 و Σ_2 نامعلوم هستند و لذا استفاده از موارد بالا امکان‌پذیر نیست. بنابراین از نمونه‌های آموزشی استفاده کرده و این پارامترها را برآورد می‌کنیم. $\boldsymbol{\mu_1}$ و $\boldsymbol{\mu_2}$ با استفاده از $\bar{\boldsymbol{x}}_{\boldsymbol{1}}$ و $\bar{\boldsymbol{x}}_{\boldsymbol{2}}$ برآورد می‌شوند که $\bar{\boldsymbol{x}}_{\boldsymbol{1}}$ و $\bar{\boldsymbol{x}}_{\boldsymbol{2}}$ به ترتیب میانگین نمونه‌های (آموزشی) مربوط به طبقات π_1 و π_2 هستند. ماتریس‌های Σ_1 و Σ_2 را نیز با استفاده از ماتریس‌های کواریانس نمونه‌ای یعنی S_1 و S_2 برآورد کرده و در روابط بالا قرار می‌دهیم. S_1 و S_2 به ترتیب ماتریس‌های کواریانس نمونه‌های (آموزشی) مربوط به طبقات π_1 و π_2 هستند. در نتیجه قاعده تصمیم به صورت زیر خواهد بود: $\boldsymbol{x_0}$ به طبقه π_1 تعلق می‌گیرد اگر

$$-\frac{1}{2}\boldsymbol{x_0^T}(S_1^{-1}-S_2^{-1})\boldsymbol{x_0}+(\bar{\boldsymbol{x}}_1^T S_1^{-1}-\bar{\boldsymbol{x}}_2^T S_2^{-1})\boldsymbol{x_0}-\frac{1}{2}\ln\left(\frac{|S_1|}{|S_2|}\right)-\frac{1}{2}(\bar{\boldsymbol{x}}_1^T S_1^{-1}\bar{\boldsymbol{x}}_{\boldsymbol{1}}-\bar{\boldsymbol{x}}_2^T S_2^{-1}\bar{\boldsymbol{x}}_{\boldsymbol{2}})$$
$$\geq \ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right]$$

و در غیر این صورت $\boldsymbol{x_0}$ به طبقه π_2 تعلق می‌گیرد. همان‌گونه که مشخص است، تابع

$$Q(\boldsymbol{x_0})=-\frac{1}{2}\boldsymbol{x_0^T}(S_1^{-1}-S_2^{-1})\boldsymbol{x_0}+(\bar{\boldsymbol{x}}_1^T S_1^{-1}-\bar{\boldsymbol{x}}_2^T S_2^{-1})\boldsymbol{x_0}$$
$$-\frac{1}{2}\ln\left(\frac{|S_1|}{|S_2|}\right)-\frac{1}{2}(\bar{\boldsymbol{x}}_1^T S_1^{-1}\bar{\boldsymbol{x}}_{\boldsymbol{1}}-\bar{\boldsymbol{x}}_2^T S_2^{-1}\bar{\boldsymbol{x}}_{\boldsymbol{2}})$$

یک تابع درجه دوم نسبت به $\boldsymbol{x_0}$ است، به همین دلیل به آن تابع ممیزی درجه دوم گفته می‌شود. همچنین به این روش (یعنی حالتی که دو طبقه یا دو جامعه دارای توزیع نرمال چندمتغیره با ماتریس کواریانس غیر یکسان هستند) روش ممیزی درجه دوم می‌گویند.

تابع ممیزی درجه دوم زمانی قابل استفاده است که توزیع هر دو جامعه نرمال چندمتغیره (با ماتریس کواریانس‌های نابرابر) باشد. با این حال اگر حجم‌های نمونه‌های آموزشیِ گرفته شده از دو جامعه بزرگ باشند (که در عمل معمولا چنین است)، می‌توان از قضیه حد مرکزی (حالت چند متغیره) استفاده کرده و قواعد داده شده در بالا را به کار گرفت.

مثال: مجددا مثال قبل را در نظر می‌گیریم. از 20 دانشجو خواسته شده که در ابتدای روز اسمی را به

خاطر بسپارند. متغیر پیش‌بین زمان خواب شبانه بر حسب دقیقه است. همچنین متغیر پاسخ «موفقیت یا عدم موفقیت در یادآوری اسم در صبح فردا» است. نتایج در جدول زیر آورده شده است. موفقیت در یادآوری اسم را با یک، و عدم موفقیت در یادآوری را با صفر نشان داده‌ایم.

زمان خواب (x)	30	45	60	75	90	105	105	120	135	150
یادآوری(y)	0	0	0	0	0	1	0	1	0	0
زمان خواب (x)	165	180	195	210	240	255	270	285	300	330
یادآوری (y)	1	0	1	0	1	1	1	1	1	1

با استفاده از این داده‌ها به عنوان داده‌های آموزشی، تابع ممیزی درجه دوم را مشخص می‌کنیم. ابتدا داریم

$$n_1 = n_2 = 10;\ \bar{x}_1 = 106.5, \bar{x}_2 = 228;\ S_1^2 = 3472.5, S_2^2 = 5590$$

بنابراین تابع ممیزی درجه دوم به صورت زیر است (چون تنها یک متغیر x داریم لذا نمادهای ماتریسی حذف شده است)

$$Q(x_0) = -\frac{1}{2}x_0^2\left(\frac{1}{S_1^2} - \frac{1}{S_2^2}\right) + \left(\frac{\bar{x}_1}{S_1^2} - \frac{\bar{x}_2}{S_2^2}\right)x_0 - \frac{1}{2}\ln\left(\frac{S_1^2}{S_2^2}\right) - \frac{1}{2}\left(\frac{\bar{x}_1^2}{S_1^2} - \frac{\bar{x}_2^2}{S_2^2}\right)$$
$$= -0.000055x_0^2 - 0.0101x_0 + 3.2546$$

از طرفی با فرض $c(1|2) = c(2|1)$ و $p_1 = p_2$ داریم

$$\ln\left[\frac{c(1|2)}{c(2|1)}\frac{p_2}{p_1}\right] = 0$$

بنابراین یک مشاهده مانند x_0 به طبقه π_1 (یعنی حالتی که y=0) تعلق می‌گیرد اگر

$$Q(x_0) = -0.000055x_0^2 - 0.0101x_0 + 3.2546 \geq 0$$

و در غیر این صورت به طبقه π_2 (یعنی حالتی که y=1) تعلق می‌گیرد.

مثلا برای دانشجویی که $x_0 = 200$ دقیقه خواب داشته است داریم

$$Q(x_0) = -0.000055(200)^2 - 0.0101(200) + 3.2546 = -0.95 < 0$$

بنابراین دانشجو با میزان خواب $x = 200$ در طبقه π_2 (یعنی حالتی که y=1) قرار می‌گیرد. ■

5-1 طبقه‌بندی داده‌های Heart با استفاده از روش ممیزی درجه دوم

مشابه روش ممیزی خطی، می‌توانیم از کتابخانه MASS برای برازش مدل ممیزی درجه دوم استفاده کنیم. در اینجا از تابع qda استفاده می‌کنیم. خروجی بسیار شبیه به خروجی ممیزی خطی است. زیرا شامل احتمالات پیشین و میانگین رده‌ها است اما شامل ضرایب متمایز کننده‌های خطی نیست، زیرا طبقه‌بندی کننده QDA شامل یک تابع درجه دوم و نه خطی از پیش‌بینی کننده‌ها است.

```
Heart[1:12]=scale(Heart[1:12])
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
qda.m1 <- qda(DEATH_EVENT ~ ., data = train_data)
```

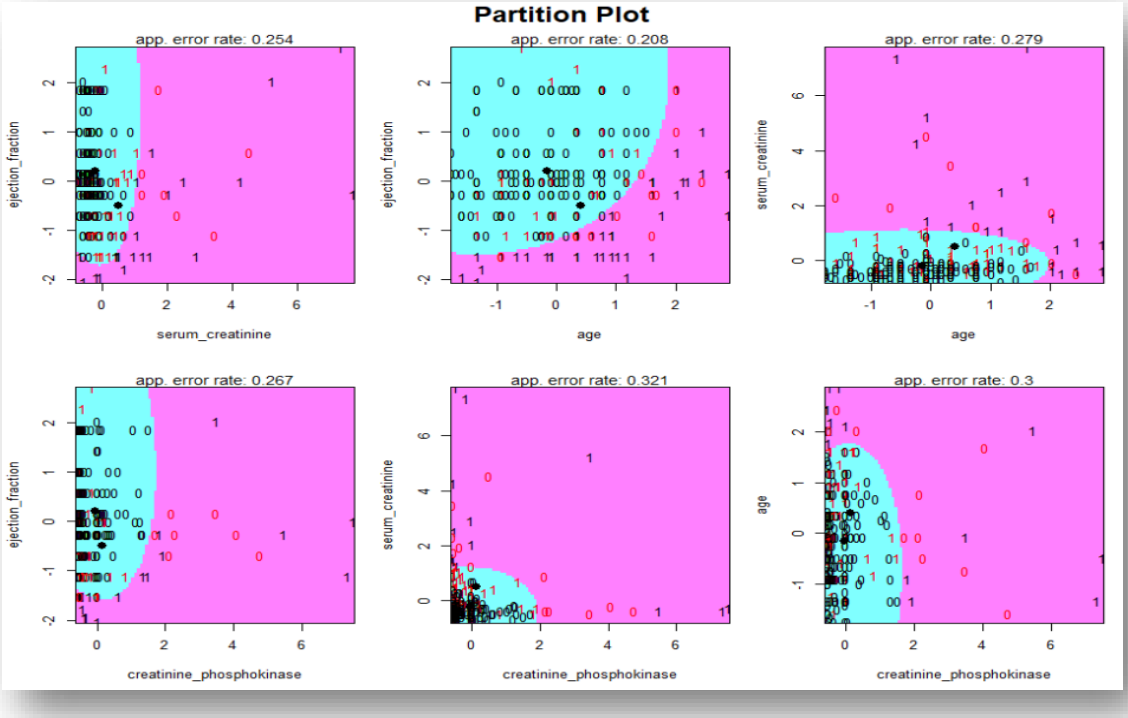
```
Call:
qda(DEATH_EVENT ~ ., data = train_data)

Prior probabilities of groups:
        0         1
0.6791667 0.3208333

Group means:
         age    anaemia creatinine_phosphokinase    diabetes ejection_fraction high_blood_pressure
0 -0.1515971 0.02072065              -0.03730999  0.02304880         0.2028798          -0.08007576
1  0.4011979 0.04657728               0.13935295 -0.05758289        -0.4888034           0.08039510
    platelets serum_creatinine serum_sodium        sex     smoking       time
0  0.01969400       -0.2073972    0.1905595 0.02875175 0.008730807  0.3485446
1 -0.01403599        0.5036737   -0.3212766 0.05541357 0.035478197 -0.7135002
```

تابع partimat راهی برای رسم توابع تفکیک درجه دوم فراهم می‌کند. این نمودارها بصری خوبی از تفاوت بین توابع خطی مورد استفاده در LDA و توابع درجه دوم استفاده شده در QDA ارائه می‌دهند. مناطق رنگی هر ناحیه طبقه بندی را مشخص می‌کنند. هر مشاهده‌ای که در یک منطقه قرار می‌گیرد، پیش‌بینی می‌شود که از یک کلاس خاص باشد. هر نمودار همچنین شامل نرخ خطای ظاهری برای آن

نمای داده است.

قابل ذکر است به دلیل اینکه تعداد متغیرها زیاد می‌باشد و رسم نمودار صورت نمی‌گیرد ما تنها متغیرهای مهم که در قسمت‌های قبل مشخص کرده بودیم را در مدل درنظر می‌گیریم.

```
partimat(DEATH_EVENT ~
        ejection_fraction +
        serum_creatinine +
        age+
       creatinine_phosphokinase, data = train_data, method="qda")
```



حال با استفاده از کد زیر می‌توان با مدل فوق، عمل پیش‌بینی را برای داده‌های آزمایشی انجام داد:

```
Heart.probs = predict(qda.m1, test_data, type = "response")
```

بخشی از خروجی کد فوق را در ادامه مشاهده می‌کنیم.

```
$class
 [1] 0 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0
[51] 0 0 0 0 0 0 0 0 0 0
Levels: 0 1

$posterior
            0           1
8   0.99573791 0.0042620866
12  0.46689252 0.5331074767
13  0.73301698 0.2669830237
16  0.75496932 0.2450306759
19  0.69722412 0.3027758844
20  0.09619172 0.9038082771
25  0.40576909 0.5942309096
31  0.26315781 0.7368421854
```

یک مشاهده به کلاس 0 تعلق می‌گیرد اگر احتمال پیش‌بینی شده مربوط به عدد 0 برای آن مشاهده از عدد 0.5 بزرگتر باشد و در غیر این صورت به کلاس 1 تعلق می‌گیرد. حال با کد زیر به طور دقیق‌تر رده پیش‌بینی شده را برای هر یک از مشاهدات به دست می‌آوریم.

```
class(Heart.probs)
class.predictions = apply(Heart.probs$posterior, 1, which.max)
class.predictions[class.predictions == 1] = levels(Heart$DEATH_EVENT)[1]
class.predictions[class.predictions == 2] = levels(Heart$DEATH_EVENT)[2]
class.predictions
```

```
> class.predictions
  8  12  13  16  19  20  25  31  38  40  43  45  59  65  78  81  82  85  93  94  99 113 116 125 129 131
  1   2   1   1   1   2   2   2   1   2   1   1   2   1   1   1   1   1   1   1   1   1   2   1   1
133 140 149 158 167 175 181 187 189 192 193 197 200 203 204 208 209 211 213 214 215 230 233 241 243 245
  1   1   1   1   1   1   1   1   1   1   1   1   2   1   2   1   1   2   1   1   1   1   1   1   1   1
248 251 256 257 264 273 298
  1   1   1   1   1   1   1
```

توجه می‌کنیم که 1 نشان‌دهنده وقوع مرگ و 0 نشان‌دهنده عدم وقوع مرگ برای یک شخص (مشاهده) است. بنابراین با مدل ممیزی درجه دوم، پیش‌بینی وقوع مرگ برای داده‌های آزمایشی به صورت بالا به

```
table(class.predictions,test_data$DEATH_EVENT)
```

```
class.predictions  0  1
                0 37 12
                1  3  7
```

بنابراین برای داده‌های آزمایشی، 37 تا از داده‌ها، مقدار متغیر پاسخ (DEATH_EVENT) واقعا صفر

است و مدل ممیزی درجه دوم نیز این 37 داده را به درستی به رده صفر طبقه‌بندی کرده است. همچنین برای 7 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است و مدل ممیزی درجه دوم نیز این 7 داده را به درستی به رده یک طبقه‌بندی کرده است. اما برای 12 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است اما مدل ممیزی درجه دوم این 12 داده را به غلط به رده صفر طبقه‌بندی کرده است. در نهایت برای 3تا از داده‌ها، مقدار متغیر پاسخ واقعا صفر است اما مدل ممیزی درجه دوم این 3 داده را به غلط به رده یک طبقه‌بندی کرده است.

قسمت دیگری از خروجی کدهای بالا آماره‌های مربوط به سنجش کیفیت طبقه‌بندی انجام شده توسط روش ممیزی درجه دوم است که در زیر آورده شده است

```
mean(class.predictions==test_data$DEATH_EVENT)*100
```

```
> mean(class.predictions==test_data$DEATH_EVENT)*100
[1] 74.57627
```

بنابراین یعنی از کل داده‌های آزمایشی، حدود 76 درصد آن‌ها به درستی طبقه‌بندی شده‌اند.

6 طبقه‌بندی به روش k-نزدیک‌ترین همسایه

جستجوی نزدیک‌ترین همسایه یک مسئله بهینه‌سازی برای پیدا کردن نزدیک‌ترین نقطه‌ها در فضاهای متریک است. مسئله بدین صورت است که در یک فضای متریک مانند M، یک مجموعه مانند S شامل تعدادی نقطه داده شده است. یک نقطه‌ی $q \in S$ را در نظر می‌گیریم. هدف پیدا کردن نزدیک‌ترین نقطه در S به این نقطه‌ی q است. در بسیاری از موارد، فضای متریک M، فضای اقلیدسی d بعدی و فاصله بین نقاط با معیار فاصله اقلیدسی، فاصله ماهالونوبیس یا فاصله منهتن سنجیده می‌شود.

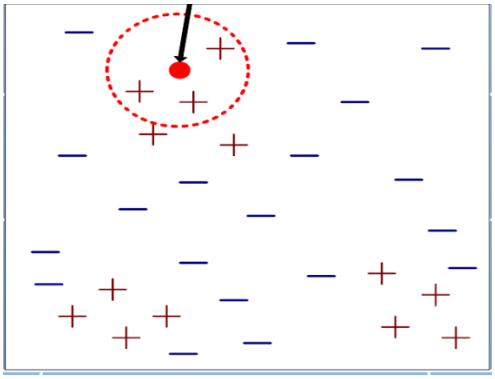
روش k نزدیک‌ترین همسایه، یک گروه شامل k داده از مجموعه داده‌های آموزشی که نزدیک‌ترین داده‌ها به داده آزمایشی هستند را انتخاب می‌کند و براساس برتری طبقه یا برچسب مربوط به آنها، در مورد طبقه‌ی داده آزمایشی مورد نظر تصمیم‌گیری می‌کند. به بیان دیگر، این روش طبقه‌ای را انتخاب

می‌کند که در همسایگی انتخاب شده، بیشترین تعداد داده‌ی منتسب به آن طبقه موجود باشد. بنابراین طبقه‌ای که در بین k نزدیک‌ترین همسایه، از همه طبقه‌ها بیشتر مشاهده شود، به عنوان طبقه‌ی داده‌ی جدید در نظر گرفته می‌شود. لازم به ذکر است که برای سنجش میزان نزدیکی، از متغیرهای مستقل (یا ویژگی‌هایی) که هم برای داده‌های آموزشی و هم برای داده آزمایشی ثبت شده استفاده می‌شود. با استفاده از مقادیر ثبت شده برای متغیرهای مستقل، مقدار فاصله (مثلا فاصله اقلیدسی) محاسبه می‌شود. برای به کارگیری الگوریتم KNN به سه مورد نیاز است: 1. یک مجموعه داده‌ی آموزشی. 2. یک متریک برای سنجش فاصله داده آزمایشی از سایر داده‌ها (یعنی داده‌های آموزشی). برای سنجش فاصله، معیارهای متعددی مانند فاصله اقلیدسی، فاصله منهتن، فاصله مینکوفسکی، فاصله ماهالونوبیس وجود دارد. 3. تعیین مقدار k.

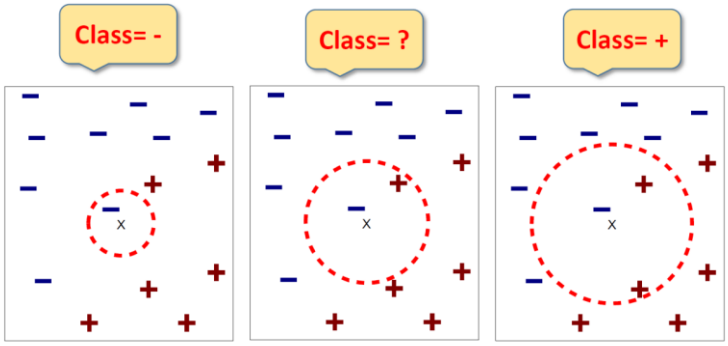
برای مسائل طبقه‌بندی دودویی، یعنی زمانی که تنها دو طبقه وجود دارد، معمولا در نظر گرفتن مقادیر فرد برای k بهتر است، زیرا امکان پیروز شدن یکی از دو طبقه را افزایش می‌دهد. برای مسائل طبقه‌بندی چند دسته‌ای، یعنی زمانی که تعداد طبقات بیشتر از دو تا است، باید عدد k را بزرگتر از تعداد طبقات و نیز متفاوت با تعداد طبقات (از نظر زوج یا فرد بودن) در نظر گرفت. یعنی اگر تعداد طبقات زوج باشد باید k را فرد گرفت و بالعکس. در روش KNN برای طبقه‌بندی کردن یک داده با طبقه‌ی نامشخص، به صورت زیر عمل می‌شود: 1. با استفاده از یک متر مشخص (که تعدادی از آنها در بالا ذکر شد)، فاصله داده جدید از همه داده‌های آموزشی محاسبه می‌شود. 2. k تا از نزدیک‌ترین همسایه‌ها مشخص می‌شوند. 3. از برچسب طبقاتِ مربوط به این نزدیک‌ترین همسایه‌ها برای پیش‌بینی طبقه‌ی این داده‌ی جدید استفاده می‌شود. به این صورت که بین k همسایه رای‌گیری شده و طبقه‌ای که بیشترین تعداد دفعات دیده شدن را در بین این k همسایه داراست، به عنوان طبقه داده‌ی جدید در نظر گرفته می‌شود. به عنوان مثال شکل زیر فرض شده که تنها دو برچسب (طبقه) + و – وجود دارد. داده با رنگ قرمز یک داده آزمایشی (جدید) است و هدف تعیین طبقه‌ی آن با استفاده از الگوریتم KNN است. سایر

داده‌ها (یعنی داده‌های آموزشی) تنها با استفاده از برچسب‌شان مشخص شده‌اند (با علامت‌های + و –).

اگر $k = 3$ در نظر بگیریم، در این صورت در الگوریتم 3NN ابتدا سه داده آموزشی که نزدیک‌ترین داده‌ها به این داده آزمایشی هستند مشخص می‌شوند. در شکل دور آنها دایره قرمزی کشیده شده است. این داده‌ها یا دارای برچسب + و یا دارای برچسب – هستند. حال برای تعیین برچسب (طبقه‌ی) مربوط به این داده‌ی آموزشی (یعنی داده‌ی جدید) در بین این سه همسایه رای‌گیری می‌شود و تعداد برچسب‌های + و – شمرده می‌شود و برچسب (طبقه‌ی) این داده جدید برابر است با بیشترین تعداد برچسبی که در این همسایه‌ها مشاهده می‌شود. ملاحظه می‌کنیم که در سه همسایه مشخص شده، 3 تا از آنها برچسب + و 0 تا از آنها برچسب – دارند. بنابراین برچسبی که به این داده جدید داده می‌شود + است.

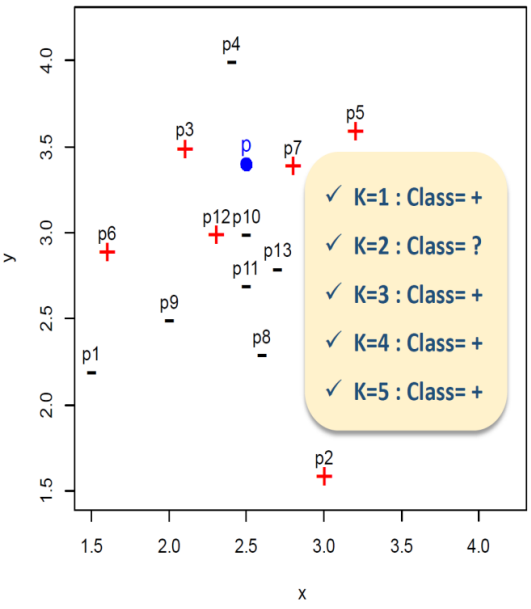


در حالت کلی با تغییر مقدار k ممکن است خروجی‌های و پیش‌بینی‌های متفاوتی مشاهده شود که به نتیجه‌گیری‌های متفاوتی منجر می‌شوند. به عنوان مثال شکل زیر را در نظر می‌گیریم. در قسمت سمت چپ این شکل از الگوریتم 1NN برای تعیین برچسب داده جدید (که با علامت × مشخص شده) استفاده شده و برچسب داده جدید – به دست آمده است. در قسمت وسط از الگوریتم 2NN استفاده شده اما در این حالت برچسب داده جدید نامشخص است زیرا تعداد رای‌های مثبت و منفی با هم برابر است. بالاخره در قسمت سمت راست از الگوریتم 3NN استفاده شده و برچسب داده جدید + به دست آمده است.



به عنوان مثالی دیگر، داده‌های جدول زیر را در نظر بگیرید. برای تعیین برچسبِ نقطه‌ی p، از روش KNN به ازای مقادیر مختلف k استفاده می‌کنیم.

Name	x	y	Class	d
p1	1.5	2.2	-	1.562
p2	3	1.6	+	1.868
p3	2.1	3.5	+	0.412
p4	2.4	4	-	0.608
p5	3.2	3.6	+	0.728
p6	1.6	2.9	+	1.03
p7	2.8	3.4	+	0.3
p8	2.6	2.3	-	1.105
p9	2	2.5	-	1.03
p10	2.5	3	-	0.4
p11	2.5	2.7	-	0.7
p12	2.3	3	+	0.447
p13	2.7	2.8	-	0.632
p	2.5	3.4	?	0



اگر $k = 1$ فاصله نقطه‌ی p با نزدیک‌ترین همسایه‌ی خود یعنی نقطه‌ی p7 برابر با 0.3 است. چون p7 دارای برچسب +است بنابراین برچسب نقطه‌ی p نیز + تعیین می‌گردد.

اگر $k = 2$ فاصله نقطه‌ی p با نزدیک‌ترین همسایه‌های خود یعنی نقاط p7 و p10 به ترتیب 0.3 و 0.4 است. این دو همسایه دارای برچسب‌های + و - هستند. با توجه به قاعده رای اکثریت، در اینجا نمی‌توان برچسب نقطه‌ی p را مشخص کرد.

اگر $k = 3$ فاصله نقطه‌ی p با نزدیک‌ترین همسایه‌های خود یعنی نقاط p7، p10 و p3 به ترتیب 0.3،

0.4 و 0.412 است که دارای برچسب‌های +، - و + هستند. با توجه به قاعده رای اکثریت، برچسب نقطه‌ی p نیز + تعیین می‌گردد.

به طور مشابه، با انتخاب مقادیر $k = 4$ و $k = 5$ برچسب نقطه‌ی p باز هم مثبت تعیین می‌شود.

به عنوان مثالی دیگر، فرض کنید شرکتی قصد دارد بر اساس 2 ویژگی (یا متغیر مستقل)، وفاداری مشتریان و خرید مجدد آنها (متغیر پاسخ) را بسنجد. ویژگی اول مدت زمانی است که مشتری از محصول آنها استفاده کرده است (برحسب سال). ویژگی دوم این است که آیا مشتری از محصولات شرکت‌های رقیب هم استفاده کرده است یا خیر. عدد صفر نشان‌دهنده خیر و عدد یک نیز نشان‌دهنده بله است.

فرض کنیم جدول داده‌های زیر در دسترس باشد (به عنوان داده‌های آموزشی):

شماره مشتری	مدت زمان استفاده (X1)	استفاده از برند دیگر (X2)	خرید مجدد از شرکت (Y)
1	5	0	بله
2	3	1	خیر
3	4	0	بله
4	1	0	بله
5	6	0	بله
6	2.5	1	خیر
7	4.5	1	خیر
8	3	0	بله

حال فرض کنید یک مشتری جدید وارد شده است. ابتدا اطلاعات (ویژگی‌ها یا متغیرهای مستقل) مربوط به این مشتری جدید ثبت می‌شود. فرض کنید مدت زمان استفاده این مشتری 3.5 سال و مشخص شود که از برند دیگری استفاده نکرده است. بنابراین مشخص می‌شود که X1=3.5 و X2=0.

حال سوال این است که آیا این مشتری باز هم از شرکت خرید می‌کند؟ یعنی می‌خواهیم مقدار Y (متغیر پاسخ) را پیش‌بینی کنیم.

برای پاسخ به این سوال باید دید که این مشتری در کدام طبقه قرار می‌گیرد. برای به کارگیری الگوریتم KNN و برای تعیین فاصله همسایه از روش منهتن استفاده می‌کنیم. در این روش باید ویژگی‌ها نظیر به نظیر مقایسه و از هم کم شوند. یعنی باید مقادیر میزان سال را از هم کم کنیم، سپس مقدار خرید از برند دیگر هم از هم کم کنیم و سپس حاصل را با هم جمع کنیم. از تمام تفاضل‌ها هم قدرمطلق گرفته می‌شود. حال اینکار را برای تمامی داده‌ها انجام می‌دهیم و فاصله مشتری جدید با همسایگان را پیدا می‌کنیم. نتایج در جدول زیر آمده است:

مشتری	X1	X2	Y	فاصله
1	5	0	بله	$\|3.5-5\|+\|0-0\|=1.5$
2	3	1	خیر	$\|3.5-3\|+\|0-1\|=1.5$
3	4	0	بله	$\|3.5-4\|+\|0-0\|=0.5$
4	1	0	بله	$\|3.5-1\|+\|0-0\|=2.5$
5	6	0	بله	$\|3.5-6\|+\|0-0\|=2.5$
6	2.5	1	خیر	$\|3.5-2.5\|+\|0-1\|=2$
7	4.5	1	خیر	$\|3.5-4.5\|+\|0-1\|=2$
8	3	0	بله	$\|3.5-3\|+\|0-0\|=0.5$

حال اگر مثلا $k=4$ در نظر بگیریم، یعنی بخواهیم با استفاده از 4 همسایه‌ی نزدیک به این مشتری، برچسب او را مشخص کنیم، به 4 مشتری با شماره‌های به ترتیب 3، 8، 1 و 2 توجه می‌کنیم (در جدول با رنگ سبز مشخص شده‌اند). فاصله این مشتری‌ها با مشتری جدید به ترتیب 0.5، 0.5، 1.5 و 1.5 است. حال مشاهده می‌کنیم که از این 4 همسایه، 1 مورد خرید مجدد نداشته و 3 مورد دیگر خرید

مجدد داشته اند. پس رای‌گیری به نفع خرید مجدد است. پس با توجه به 4 همسایه، مشتری جدید در رده‌ی مشتری‌هایی قرار می‌گیرد که خرید مجدد از این شرکت را تجربه می‌کنند و می‌توان گفت که مشتری جدید در آینده، به احتمال زیاد دوباره از شرکت خرید می‌کند (یعنی مقدار متغی پاسخ یعنی Y برابر با «بله» پیش‌بینی می‌شود).

6-1 طبقه‌بندی داده‌های Heart با استفاده از روش KNN

ابتدا داده‌ها را فراخوانی و سپس به دلیل اینکه در پیاده‌سازی الگوریتم k نزدیکترین همسایگی از مفهوم فاصله استفاده می‌شود، مشاهدات را استاندارسازی و داده‌های کیفی را نیزکدگذاری کرده تا قابل قیاس شوند.

```
Heart=read.csv("C:/Users/ASUS/Desktop/heart.csv")
str(Heart)
```

```
'data.frame':   299 obs. of  13 variables:
 $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
 $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
 $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
 $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
 $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
 $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
 $ platelets               : num  265000 263358 162000 210000 327000 ...
 $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
 $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
 $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
 $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
 $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
 $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

شکل مناسب داده‌ها به عنوان ورودی الگوریتم‌های داده‌کاوی نقش به سزایی در این فرایند بازی می‌کند و در مرحله‌ی آماده‌سازی داده‌ها این نقش پررنگ است. تکنیک‌های تغییر شکل داده‌ها متکی به مشکل نیستند و اغلب در اجرا نتایج بهتری از داده‌کاوی را سبب می‌شوند. استفاده از توابع تجمعی، نرمال سازی، خوشه‌بندی و رگرسیون روش‌های مرسومی هستند که برای تغییر و تبدیل شکل داده‌ها می‌توان

از آن‌ها استفاده نمود. انتخاب روش مناسب به ماهیت داده‌ها، مقدار آن و اهداف داده‌کاوی بستگی دارد. استانداردسازی داده‌ها کمک می‌کند که اهمیت آن‌ها به واحد اندازه‌گیری‌شان بستگی نداشته باشد. در نتیجه در مواردی مانند داده‌کاوی و تحلیل داده‌های چند متغیره از داده‌های استاندارد شده استفاده می‌شود. قبل از انجام هر گونه تحلیلی روی داده‌ها، باید آن‌ها را استاندارد کرد. بخصوص زمانی که داده‌ها چند بُعدی باشند. استفاده از داده‌های استاندارد نشده ممکن است روی نتایج حاصل از تحلیل‌ها اثر نامناسبی داشته باشد.

مقیاس‌دهی داده‌ها بسیار مهم و ضروری است، چرا که درغیراین‌صورت ممکن است یک متغیر تنها به‌خاطر مقیاسی که دارد، تأثیر زیادی بر پیش‌گویی بگذارد. استفاده از متغیرهایی که تعیین مقیاس نشده‌اند می‌تواند منجر به نتایج بی‌معنی شود.

از تکنیک‌های رایج برای مقیاس دهی داده‌ها، می‌توان به موارد زیر اشاره کرد:

- minmax normalization
- Zscore normalization
- median and MAD
- tanh estimators

تکنیک«minmax normalization» ، داده‌ها را به یک محدوده رایج تبدیل می‌کند، بنابراین تأثیر مقیاس از تمام متغیرها حذف می‌شود. برخلاف روش‌های «Zscore normalization» و «median and MAD»، روش «minmax» توزیع اصلی متغیرها را حفظ می‌کند.

عملیات استانداردسازی قبل از بسیاری از الگوریتم‌های داده‌کاوی مانند شبکه‌های عصبی، SVM، KNN بایستی انجام بگیرد تا ابعاد مختلف به صورت عادلانه توسط الگوریتم بررسی شوند و تاثیر یکی بیشتر از بقیه نباشد.

این کار را می‌توانیم به دو روش انجام دهیم:

روش اول استفاده از دستور scale است در این روش از فرمول زیر برای استاندارد سازی داده‌ها استفاده

می‌شود:

$$z_i = \frac{x_i - \mu}{\sigma}$$

روش دوم یکی دیگر از روش‌های تغییر مقیاس، که در ادامه از همین روش برای استانداردسازی داده‌ها استفاده می‌کنیم؛ استفاده از روش نرمال‌سازی Min-Max است. به این ترتیب علاوه بر یکسان سازی مقیاس داده‌ها، کران‌های تغییر آن‌ها نیز در بازه [0,1] خواهد بود.

این تبدیل به صورت زیر تعریف می‌شود:

$$x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

```
Heart[1:12]=scale(Heart[1:12])
head(Heart)
```

```
        age    anaemia creatinine_phosphokinase   diabetes ejection_fraction high_blood_pressure
1  1.1909487 -0.8696469              0.000165451 -0.8461608      -1.527997920           1.3569966
2 -0.4904571 -0.8696469              7.502062717 -0.8461608      -0.007064906          -0.7344569
3  0.3502458 -0.8696469             -0.449185725 -0.8461608      -1.527997920          -0.7344569
4 -0.9108085  1.1460462             -0.485257493 -0.8461608      -1.527997920          -0.7344569
5  0.3502458  1.1460462             -0.434757017  1.1778559      -1.527997920          -0.7344569
6  2.4520030  1.1460462             -0.551217299 -0.8461608       0.161927651           1.3569966
      platelets serum_creatinine serum_sodium        sex   smoking      time DEATH_EVENT
1  1.678834e-02       0.48923681  -1.50151891  0.7344569 -0.686531 -1.626775           1
2  7.523048e-09      -0.28407611  -0.14173853  0.7344569 -0.686531 -1.601007           1
3 -1.036336e+00      -0.09074788  -1.72814897  0.7344569  1.451727 -1.588122           1
4 -5.455595e-01       0.48923681   0.08489153  0.7344569 -0.686531 -1.588122           1
5  6.507077e-01       1.26254973  -4.67433977 -1.3569966 -0.686531 -1.575238           1
6 -6.069065e-01       0.68256504  -1.04825878  0.7344569  1.451727 -1.575238           1
```

```
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
head(train_data)
```

در گام بعدی، باید مقدار k مطلوب را مشخص کنیم، بنابراین برای مقادیر مختلف k میزان دقت را از طریق کد دستوری زیر بدست می‌آوریم.

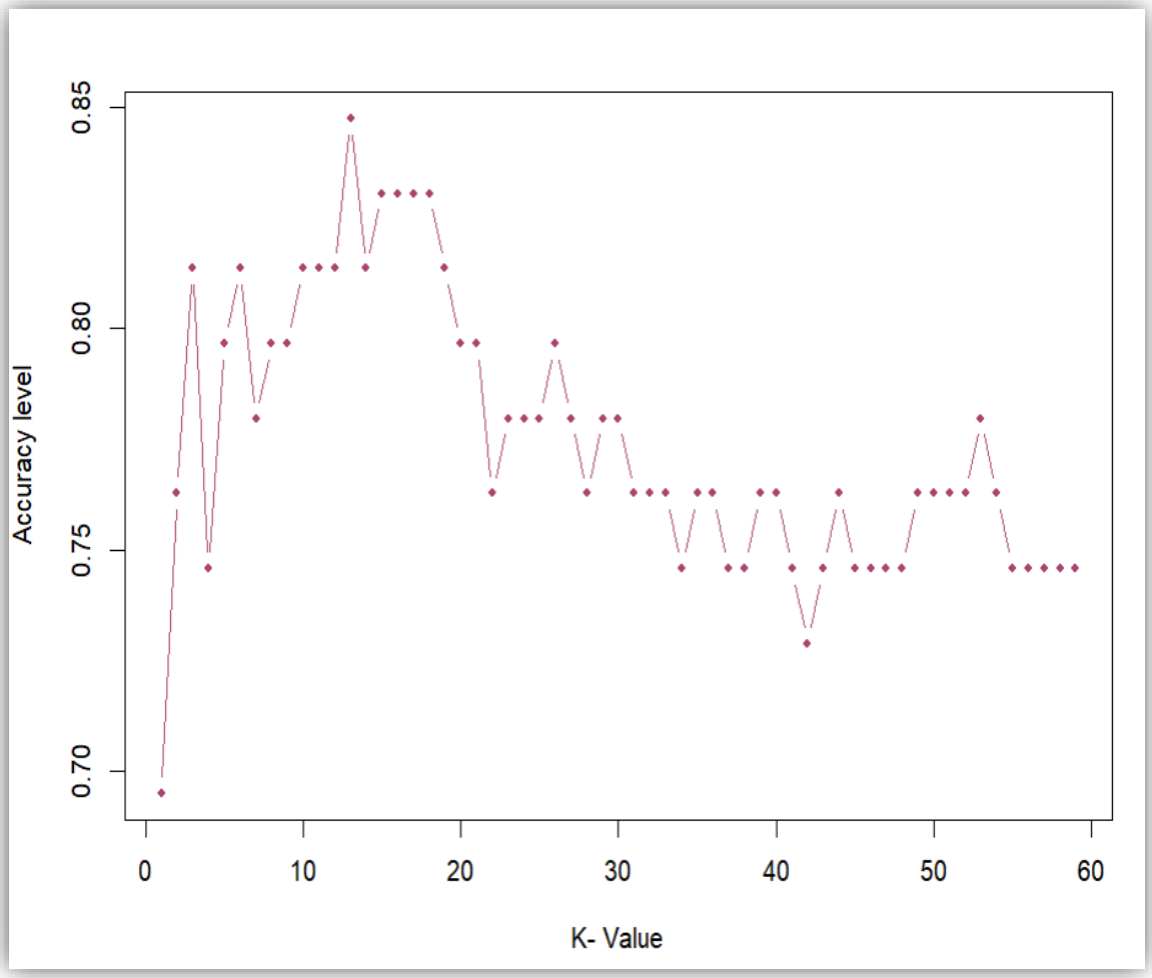
```
dim(test_data)
accuracy=c()
for(i in 2:59)}
  knn.pred=knn(train_data,test_data,cl=train_data$DEATH_EVENT,k=i)
  x=table(knn.pred,test_data$DEATH_EVENT)
  accuracy[i]=(x[1,1]+x[2,2])/dim(test_data){[1]
accuracy
```

باتوجه به خروجی، با درنظرگرفتن k از 1 تا 59 ، برای همسایگی‌های مختلف داده‌های آزمایشی موجود، دقت هر یک به دست آمده است.

```
> accuracy
 [1]        NA 0.7796610 0.8135593 0.8135593 0.7966102 0.8305085 0.7796610 0.8135593 0.7966102 0.7966102
[11] 0.8135593 0.8305085 0.8474576 0.8305085 0.8305085 0.8305085 0.8305085 0.8135593 0.8135593 0.7796610
[21] 0.7966102 0.7796610 0.7796610 0.7796610 0.7796610 0.7796610 0.7796610 0.7796610 0.7796610 0.7796610
[31] 0.7627119 0.7627119 0.7627119 0.7627119 0.7627119 0.7457627 0.7457627 0.7457627 0.7627119 0.7627119
[41] 0.7457627 0.7288136 0.7457627 0.7627119 0.7457627 0.7457627 0.7457627 0.7627119 0.7627119 0.7627119
[51] 0.7627119 0.7796610 0.7796610 0.7796610 0.7457627 0.7457627 0.7457627 0.7457627 0.7457627
```

برای درک شهودی بهتر، به رسم نمودار دقت برای همه‌ی همسایگی‌های مختلف می‌پردازیم.

```
plot(accuracy,col ="#AA4371",type="b",pch=20, xlab="K- Value",ylab="Accuracy level")
```




```
max(accuracy)
which.max(accuracy)
> max(accuracy)
[1] 0.8474576
> which.max(accuracy)
[1] 13
```

بنابراین با توجه به این که بهترین میزان دقت در k=13 رخ داده‌است با این تعداد همسایگی به پیاده‌سازی الگوریتم knn بر روی مجموعه داده‌های خود می‌پردازیم.

```
knn.pred=knn(train_data,test_data,cl=train_data$DEATH_EVENT,k=13)
head(knn.pred)
[1] 0 1 1 0 1 1
Levels: 0 1
```

در خروجی بدست آمده، مقادیر پیش‌بینی شده متغیر پاسخ برای مجموعه داده‌های آموزشی قابل مشاهده می‌باشد.

اکنون می‌توانیم ماتریس درهم‌ریختگی را به ازای k=13 تشکیل داده و در خروجی میزان دقت این الگوریتم را در پیش‌بینی کردن مقادیر پاسخ و طبقه‌بندی صحیح داده‌های آزمایشی بدست آوریم:

```
test_data$DEATH_EVENT=as.factor(test_data$DEATH_EVENT)
confusionMatrix(knn.pred,test_data$DEATH_EVENT)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 41  9
         1  0  9

               Accuracy : 0.8475
                 95% CI : (0.7301, 0.9278)
    No Information Rate : 0.6949
    P-Value [Acc > NIR] : 0.005745

                  Kappa : 0.5816

 Mcnemar's Test P-Value : 0.007661

            Sensitivity : 1.0000
            Specificity : 0.5000
         Pos Pred Value : 0.8200
         Neg Pred Value : 1.0000
             Prevalence : 0.6949
         Detection Rate : 0.6949
   Detection Prevalence : 0.8475
      Balanced Accuracy : 0.7500

       'Positive' Class : 0
```

بنابراین در روش KNN، برای 41 تا از داده‌ها، مقدار متغیر پاسخ (یعنی DEATH_EVENT) واقعا صفر است و روش KNN نیز این 41 داده را به درستی به رده صفر طبقه‌بندی کرده است. همچنین برای 9 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است و روش KNN نیز این 9 داده را به درستی به رده

یک طبقه‌بندی کرده است. اما برای 9 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است اما روش KNN این 9 داده را به غلط به رده صفر طبقه‌بندی کرده است.

باتوجه به خروجی، دقت حاصل از الگوریتم knn، 85٪ می‌باشد.

مقادیر TP ، TN ، FP و FN به ترتیب 41، 9، 9 و0 می‌باشند.

برای به دست آوردن دقت مدل که با استفاده از رابطه زیر نیز بدست می‌آید؛ داریم:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{41 + 9}{641 + 9 + 9 + 0} = 0.847$$

و این بدین معناست که این الگوریتم با احتمال 0/85 مقادیر درست متغیر پاسخ را پیش‌بینی می‌کند.

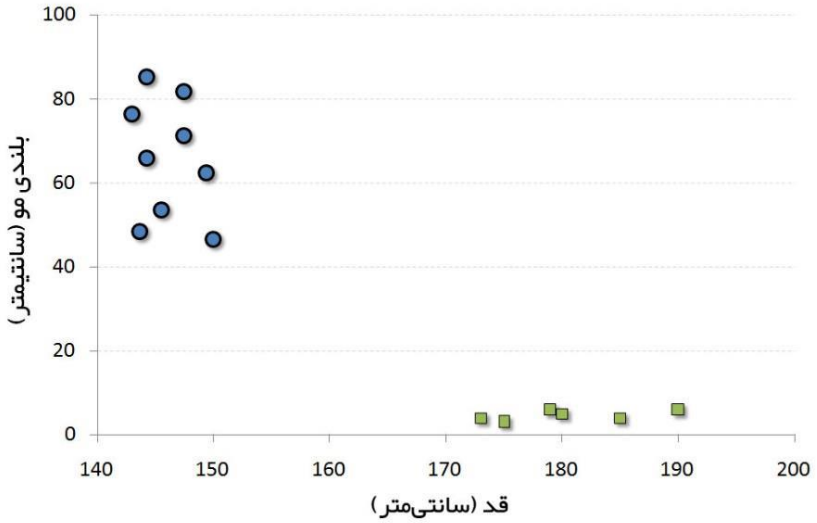
7 طبقه‌بندی به روش ماشین بردار پشتیبان (SVM)

در این بخش در مورد ماشین بردار پشتیبان[1] (SVM)، یک روش طبقه‌بندی که در دهه 1990 در علوم کامپیوتر توسعه یافت، بحث می‌کنیم. ماشین بردار پشتیبان اغلب به عنوان یکی از بهترین طبقه‌بندی‌کننده‌های «خارج از جعبه[2]» یاد می‌شود. در روش SVM هدف ایجاد بهترین خط یا مرز تصمیم‌گیری[3] است که فضای n بعدی را به رده‌هایی تفکیک می‌کند و با استفاده از این رده‌ها می‌توان به راحتی طبقه‌ی مربوط به یک مشاهده‌ی جدید را به درستی پیش‌گویی کرد.

بهترین مرز تصمیم‌گیری، ابر صفحه[4] نامیده می‌شود. SVM دورترین نقاط یا بردارهایی که برای ایجاد یک ابرصفحه کمک می‌کنند را انتخاب می‌کند. این نقاط دورافتاده بردارهای پشتیبان نامیده می‌شوند. به همین دلیل است که این الگوریتم ماشین بردار پشتیبان نامیده می‌شود. برای درک بهتر این موضوع فرض کنید جمعیتی با ترکیب 50 درصد زن و 50 درصد مرد وجود دارد و علاقه‌مندیم یک قاعده

[1] Support vector machine
[2] Out of the box
[3] Decision boundary
[4] Hyperplane

مشخص برای طبقه‌بندی این مجموعه داده بر اساس جنسیت انجام دهیم. با استفاده از الگوریتم ماشین بردار پشتیبان، قصد بر آن است که رباتی ساخته شود تا بتواند تشخیص دهد چه کسی زن و چه کسی مرد است. برای ارائه یک تصویر دوبعدی، فرض کنید ویژگی‌هایی که براساس آن‌ها طبقه‌بندی انجام می‌شود، فقط وزن و بلندی موی افراد هستند. بر این اساس، نمودار پراکنش داده‌ها در زیر رسم شده است. دایره‌های آبی موجود در این شکل نماینده زنان و مربع‌های سبز نماینده مردان هستند. برخی از برداشت‌هایی که می‌توان از این نمودار داشت عبارتند از: 1. مردان در جمعیت مثال ما، میانگین قد بلندتری دارند. 2. زنان در جمعیت مثال ارائه شده، موی بلندتری دارند.



بر این اساس اگر فردی با قد 180 سانتی‌متر و طول موی 4 سانتی‌متر در جمعیت وجود داشته باشد، بهترین حدسی که می‌توان زد آن است که فرد در طبقه مردان قرار می‌گیرد.

بردارهای پشتیبان در واقع مختصات یک مشاهده منفرد هستند. به عنوان مثال (45,150) یک بردار پشتیبان است که به یک زن اختصاص دارد. SVM مرزی است که دسته مردان و زنان را به بهترین وجه از یکدیگر جدا می‌کند. در این مثال، دو دسته وجود دارد و بنابراین جداسازی آنها به وسیله SVM آسان است.

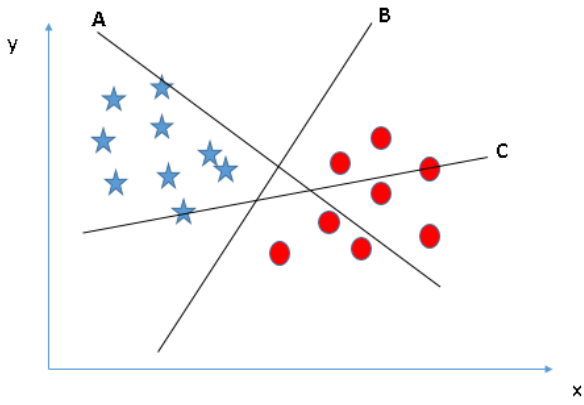
SVM در ابتدا به صورت رده‌بندی خطی دودویی مطرح شد ولی بعدها با استفاده از انواع، یکی در مقابل

یکی[1]، یکی در مقابل بقیه[2] و گراف جهت‌دار غیر مدور[3] به مسائل چندرده‌ای توسعه داده شد.

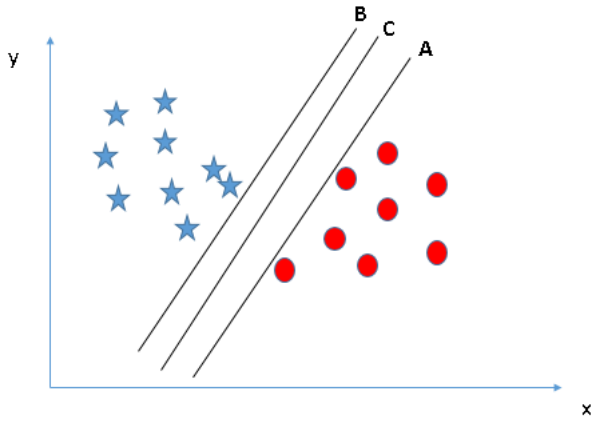
اگر داده ها به صورت خطی باشند، می‌توان آنها را با استفاده از یک خط راست جدا کرد، اما برای داده‌های غیرخطی، نمی‌توان یک خط راست ترسیم کرد، بنابراین الگوریتم‌های SVM خطی برای داده‌های غیرخطی قابل استفاده نیستند. بعدها این الگوریتم‌ها برای رده‌بندی داده‌های غیرخطی تعمیم داده شدند. برای این منظور می‌توان از آنها به عنوان الگوریتم‌های غیرخطی نیز یاد کرد. در واقع این الگوریتم‌ها قادرند در فضای ورودی به دنبال ابرسطح‌های غیرخطی باشند. برای مسائل با داده‌های غیرخطیِ تفکیک‌شدنی، SVM با استفاده از کرنل[4] توسعه یافته است.

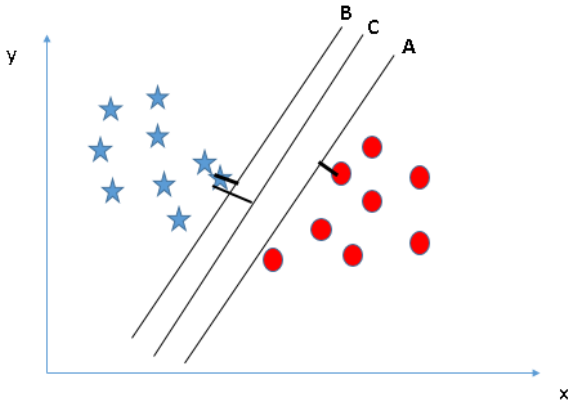
کرنل‌ها توابع ریاضی هستند که داده‌ها را از فضای ورودی به یک فضای جدید با ابعاد بالا (معروف به فضای ویژگی) تبدیل می‌کنند. در این فضای جدید می‌توان داده‌ها را با یک رویه خطی یا ابرصفحه تفکیک کرد. برای فهم بهتر این مسئله، مثال‌های شهودی زیر را در نظر می‌گیریم. در شکل زیر سه خط راست A، B و C وجود دارند. اکنون نیاز به تعیین خط راست صحیح برای طبقه‌بندی ستاره‌های آبی و دایره‌های قرمز است. در اینجا خط راستی که دو دسته را به صورت بهتری از یکدیگر تفکیک می‌کند، خطی است که باید انتخاب شود. همانگونه که در شکل مشخص است، خط B به شکل بسیار عالی هر دو دسته را از یکدیگر تفکیک می‌کند.

[1] One-versus-One
[2] One-versus-Rest
[3] Directed acyclic graph
[4] Kernel



در شکل زیر هر سه خط B، A و C دسته ستاره‌های آبی را به خوبی از دایره‌های قرمز جدا می‌کنند. در این حالت چگونه می‌توان خط راست صحیح را انتخاب کرد؟



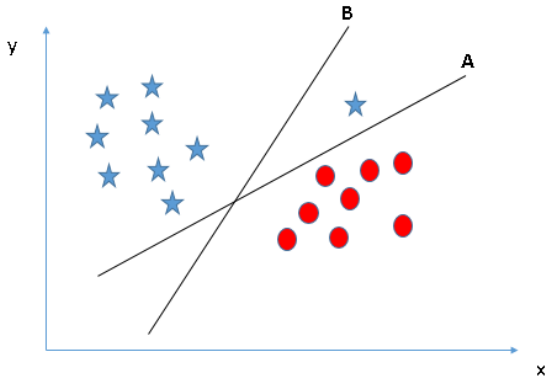


با توجه به شکل بالا، محاسبه فاصله نزدیک‌ترین داده (که می‌تواند عضو هر دسته‌ای باشد) از خط

راست می‌تواند به انتخاب خط راست صحیح کمک کند. به این فاصله حاشیه[1] گفته می‌شود.

با توجه به اینکه بزرگ‌تر بودن مقدار حاشیه بهتر است، در مثال بالا می‌توان مشاهده کرد که فاصله خط راست C در مقایسه با هر دو خط A و B از نزدیک‌ترین داده‌ی موجود در هر طبقه بیشتر است. بنابراین، خط C را به عنوان خط راست صحیح برمی‌گزینیم. دلیل دیگر برای انتخاب این خط استحکام بیشتر آن است. اگر خط راست حاشیه کمی داشته باشد، احتمال طبقه‌بندی نشدن برخی داده‌ها یا طبقه‌بندی نادرست[2] وجود دارد.

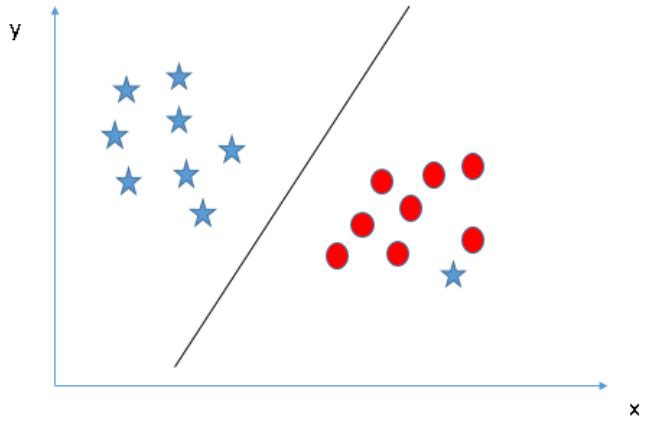
در شکل زیر برای تعیین طبقه‌ی دایره‌های قرمز و ستاره‌های آبی از قوانین تشریح شده در بخش قبل برای تعیین خط راست صحیح استفاده می‌شود. برخی ممکن است خط B را به دلیل حاشیه بیشتری که در مقایسه با A از نزدیکترین داده دارد انتخاب کنند. اما نکته مهم آن است که در اصول تعیین خط راست در الگوریتم ماشین بردار پشتیبان، خط راستی که طبقات را به درستی تقسیم کند (صحت) بر خطی که دارای حاشیه بیشتری است، اولویت دارد. بنابراین در این مثال، خط راست B یک خطای طبقه‌بندی دارد اما خط A همه داده‌ها را به درستی طبقه‌بندی کرده است. بنابراین خط راست A صحیح است.



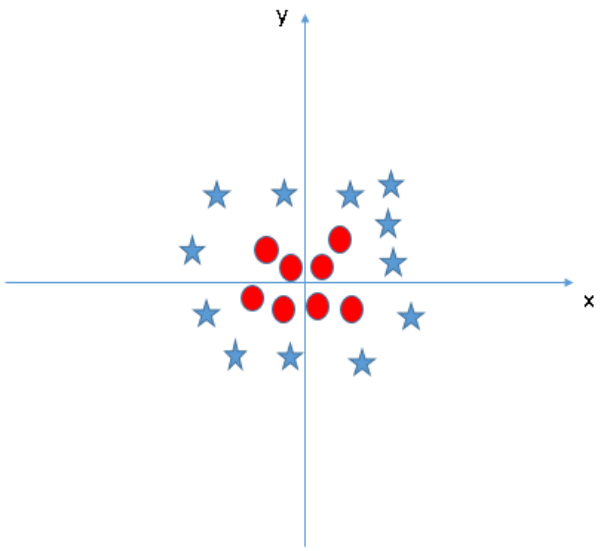
[1] Margin
[2] Miss-classification

در شکل زیر امکان طبقه‌بندی دو دسته با یک خط راست وجود ندارد، زیرا یکی از ستاره‌های آبی به صورت یک دورافتاده در قلمرو دیگر دسته یعنی دایره‌های قرمز قرار گرفته است.

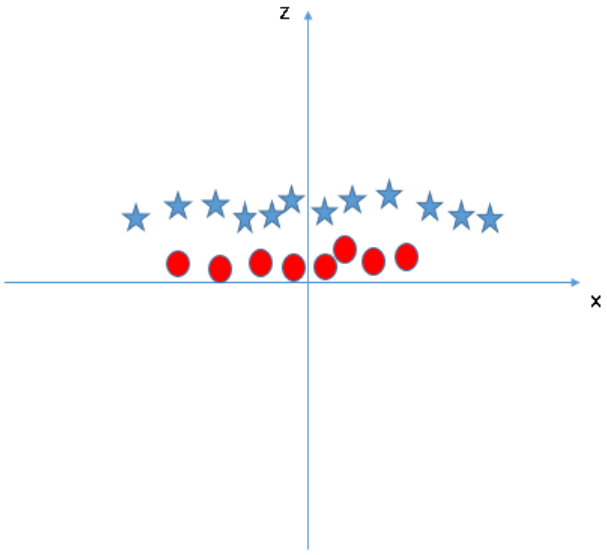
یکی از ویژگی‌های SVM آن است که دورافتاده‌ها را نادیده گرفته و تنها خط راستی را که بیشترین حاشیه را با دسته‌ها دارد انتخاب می‌کند (شکل زیر).



در شکل زیر باز هم نمی‌توان یک خط راست بین دو طبقه داشت، بنابراین این سؤال مطرح می‌شود که SVM چگونه می‌تواند این دو دسته را طبقه‌بندی کند. تاکنون و در مثال‌های پیشین تنها خطوط راست مورد بررسی قرار گرفتند.



اما در این حالت نیز ماشین بردار پشتیبان می‌تواند مسئله را به سادگی حل کند. این مسئله با افزودن یک ویژگی جدید قابل حل است. این ویژگی جدید تبدیل $z^2 = x^2 + y^2$ است که باید بر روی داده‌ها اعمال شود. اکنون می‌توان داده‌ها را روی محور x و z ترسیم کرد (شکل زیر).



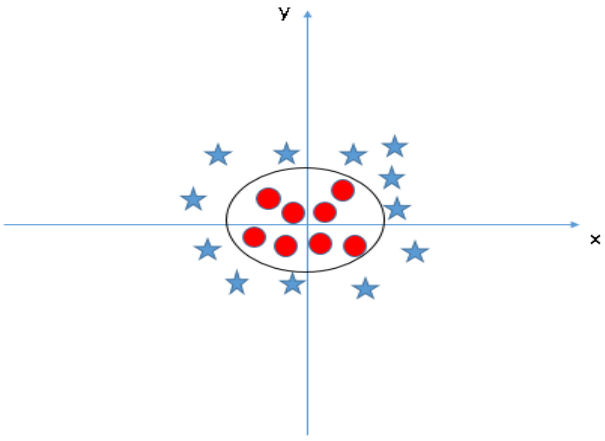
در شکل بالا نقاط داده با شرایط زیر در نظر گرفته شده‌اند:

1. مقادیر z همواره مثبت هستند زیرا z مجموع توان‌های دوم x و y است.

2. در نمودار اصلی، دایره‌های قرمز به محورهای x و y اصلی نزدیک‌ترند و این امر موجب می‌شود مقادیر

z کاهش پیدا کنند و دایره‌های قرمز در نمودار جدید به محور x نزدیک‌تر هستند و ستاره‌های آبی نسبت به دایره‌های قرمز فاصله بیشتری از محور xها دارند.

در الگوریتم SVM، داشتن یک خط راست بین این دو طبقه آسان است. اما سؤال دیگری که در این مرحله مطرح می‌شود آن است که آیا لازم است این ویژگی به صورت دستی به خط راست اضافه شود؟ پاسخ منفی است، SVM از روشی که به آن ترفند هسته[1] گفته می‌شود، استفاده می‌کند. در روش ترفند هسته، توابعی وجود دارند که فضای ورودی بُعد پایین را دریافت کرده و آن را به فضای بُعد بالاتر تبدیل می‌کنند. این تبدیل یک مسئله غیرقابل تفکیک را به مسئله‌ای قابل تفکیک مبدل می‌کند. به این توابع، توابع هسته یا کرنل گفته می‌شود. توابع کرنل بیشتر در مسائل جداسازی غیرخطی مفید هستند. این توابع برخی از داده‌های فوق‌العاده پیچیده را تبدیل می‌کنند و سپس فرآیندی را می‌یابند که با استفاده از آن بتوانند این داده‌ها را براساس برچسب‌هایی که کاربر تعریف کرده، تفکیک کنند. هنگامی که به خط جدا کننده در فضای ورودی اصلی نگاه می‌کنیم، این خط شبیه به یک دایره است (شکل زیر).



از نظر ریاضی، کرنل یک تابع است که دو آرگومان دارد، یک نگاشت (انتقال) روی آرگومان‌ها اعمال می‌کند و سپس مقدار حاصل‌ضرب نقطه‌ای آنها را برمی‌گرداند. به عنوان مثال فرض کنید $\boldsymbol{x}_1$ و $\boldsymbol{x}_2$ دو نقطه داده‌ی مثلا p بعدی، $\boldsymbol{\phi}(.)$ یک نگاشت و $K(.,.)$ تابع کرنل باشد. در این صورت کرنل به صورت

[1] Kernel

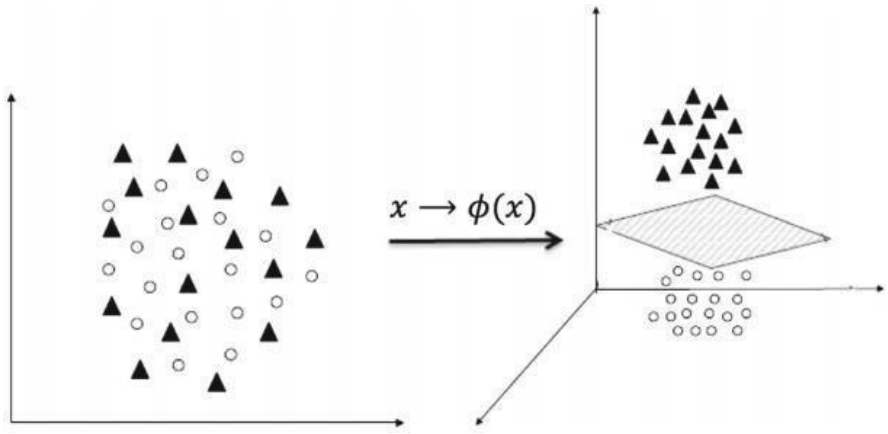
زیر تعریف می‌شود

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}(\boldsymbol{x}_1).\boldsymbol{\phi}(\boldsymbol{x}_2) = \boldsymbol{\phi}(\boldsymbol{x}_1)^T\boldsymbol{\phi}(\boldsymbol{x}_2)$$

برخی از کرنل‌های مهم و معروف در جدول زیر داده شده‌اند.

نام	کرنل
چندجمله‌ای (همگن)	$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1.\boldsymbol{x}_2)^d$
چندجمله‌ای (ناهمگن)	$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = (1 + \boldsymbol{x}_1.\boldsymbol{x}_2)^d$
گوسی	$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|\right)$
تانژانت هذلولوی	$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \tanh(\kappa\boldsymbol{x}_1.\boldsymbol{x}_2 + c)$

شکل زیر نمایش هندسی اعمال یک کرنل فرضی بر روی داده‌ها را نشان می‌دهد.



از مزایای الگوریتم SVM می‌توان به موارد زیر اشاره کرد:

1. هنگامی که اطلاعات زیادی درباره داده‌ها وجود ندارد SVM روش بسیار مفیدی است و نیازی به اطلاعات توزیعی در مورد داده‌ها ندارد.

2. برای داده‌ها با ابعاد بالا تقریباً خوب جواب می‌دهد.

3. در مسائل طبقه‌بندی خطی با استفاده از SVM، یک فرض قوی وجود دارد که داده‌ها به طور خطی

تفکیک‌پذیر باشند. در حالی که با استفاده از کرنل‌ها، داده‌های ورودی را می‌توان بدون نیاز به این فرض به داده‌های با ابعاد بالا، برای طبقه‌بندی تبدیل کرد.

4. SVM به طور کلی از شرایط بیش برازش رنج نمی‌برد و اگر طبقات واقعا تفکیک‌پذیر باشند، دارای عملکرد خوبی است.

5. ماشین بردار پشتیبان در یافتن ابرصفحه جدا کننده مفید است، یافتن ابرصفحه می‌تواند برای طبقه‌بندی صحیح داده‌ها بین گروه‌های مختلف مفید باشد.

از معایب SVM نیز می‌توان به موارد زیر اشاره کرد:

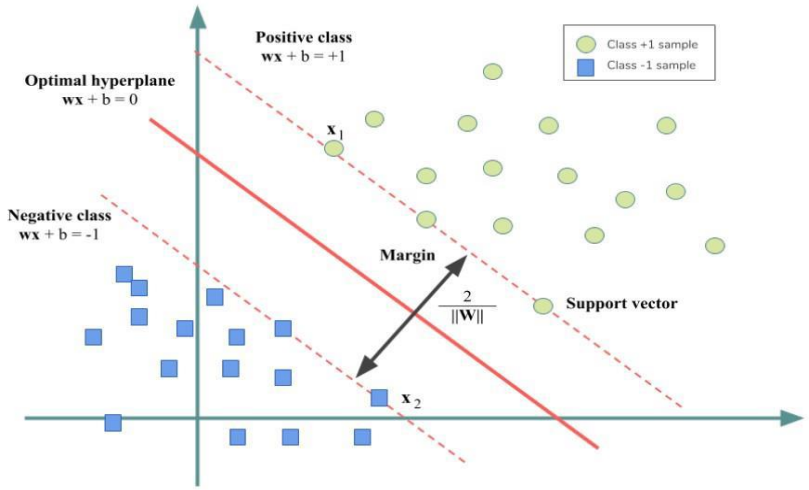
1. نیازمند زمان آموزش بسیار زیادی است، بنابراین هنگامی که مجموعه داده‌ها بسیار بزرگ است عملکرد خوبی ندارد.

2. هنگامی که مجموعه داده‌ها نویز (نوفه) زیادی داشته باشد، SVM عملکرد خوبی ندارد و طبقات هدف دچار همپوشانی می‌شوند.

حال سوالی که مطرح می‌شود این است که در SVM، چطور می‌توان مسئله موردنظر را به صورت ریاضی فرمول‌بندی کرد و به دنبال حل چه مسئله‌ای هستیم. در حقیقت مسئله طبقه‌بندی SVM یک مسئله بهینه سازی بوده که نمایش ریاضی آن شامل یک تابع هدف و یک سری محدودیت است. به این گونه موارد، بهینه سازی مقید گفته می‌شود که معمولا برای حل آن‌ها باید صورت دوگان[1] آن را نوشت.

برای ورود به فرمول بندی ریاضی ماشین بردار پشتیبان، شکل زیر را در نظر بگیرید.

[1] Dual



در این شکل یک طبقه‌بندی کننده SVM دودویی را مشاهده می‌کنیم که تنها برای داده‌های تفکیک‌پذیر خطی کاربرد دارد. از آنجایی که حضور هرگونه نویز یا نقطه دورافتاده به شدت روی حاشیه تأثیر می‌گذارد و آن را جا به جا می‌کند، لذا فرصت هیچ‌گونه خطای طبقه‌بندی اشتباه وجود ندارد و با یک حاشیه سخت‌گیرانه روبرو هستیم. به همین دلیل به عنوان SVM حاشیه سخت شناخته می‌شود.

حال فرض کنید هر نقطه داده در فضای p-بعدی، به صورت $\boldsymbol{x} = (x_1, \dots, x_p) \in \mathbb{R}^p$ باشد که نشان‌دهنده p ویژگی (متغیر) است که طبیعتا هر ویژگی را می‌توان به عنوان یک محور مختصات در دستگاه مختصات دکارتی در نظر گرفت. بدیهی است در صورت تفکیک‌پذیری، با ترسیم یک ابرصفحه p-بعدی می‌توان داده‌های مختلف و متمایز از یکدیگر را طبقه‌بندی کرد. بنابراین می‌توانیم مجموعه داده‌های آزمایشی را با D نشان داده و به صورت زیر تعریف کنیم:

$$D = \{(x_i, y_i): x_i \in \mathbb{R}^p, y_i \in \{-1,1\}, i = 1,2, \dots, n\}$$

که در آن متغیر y برای برچسب گذاری در یکی از دو رده استفاده می‌شود و بنابراین می‌تواند دو مقدار اختیار کند که در این جا فرض می‌کنیم این مقادیر برابر با +1 و -1 هستند. هدف پیدا کردن ابرصفحه جداکننده با بیشترین فاصله از نقاط حاشیه‌ای است که نقاط با $y_i = 1$ را از نقاط با $y_i = -1$ جدا کند. در حالت کلی معادله ابرصفحه در فضای p-بعدی به صورت زیر است

$$\boldsymbol{w}^T \boldsymbol{x} + b = c$$

که در آن، باتوجه به شکل بالا و برچسب تعریف شده $c \in \{0, -1,1\}$. همچنین ضریب $\boldsymbol{w}$ بردار نرمال ابرصفحه بوده که بر آن عمود است. علاقه‌مندیم بردار $\boldsymbol{w}$ و ثابت اریبی b را طوری انتخاب کنیم که بیشترین فاصله بین ابرصفحه‌های موازی که داده‌ها را از هم جدا می‌کنند، ایجاد شود. توجه کنید در این جا، حاشیه با دو ابرصفحه موازی به صورت $\boldsymbol{w}^T\boldsymbol{x} + b = 1$ و $\boldsymbol{w}^T\boldsymbol{x} + b = -1$ تعریف می‌شود. اگر فرض عدم وجود خطای طبقه‌بندی نادرست را در تعیین حاشیه‌های سخت در نظر بگیریم (معنی حاشیه سخت‌گیرانه را به یاد بیاورید)، به راحتی می‌توان از فرمول فاصله یک نقطه از صفحه برای تعیین پارامترهای مورد نیاز جهت بیشینه کردن فاصله بین دو ابرصفحه استفاده کرد. با توجه به شکل بالا فاصله بردارهای پشتیبان (نقاط سبز و آبی مرزی) از ابرصفحه جداکننده (خط قرمز) به ترتیب برابر $\frac{|1-b|}{\|\boldsymbol{w}\|}$ و $\frac{|-1-b|}{\|\boldsymbol{w}\|}$ است. در نتیجه، طول حاشیه کل برابر $\frac{2}{\|\boldsymbol{w}\|}$ به دست می‌آید. با توجه به این که به دنبال ماکسیمم کردن طول حاشیه‌ی کل هستیم، باید عبارت $\frac{2}{\|\boldsymbol{w}\|}$ را ماکسیمم یا به طور معادل عبارت $\frac{\|\boldsymbol{w}\|}{2}$ مینیمم شود. لذا مسئله زیرا را حل می‌کنیم:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 = \min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

این بهینه‌سازی با محدودیت‌هایی همراه است. با فرض برچسب‌های طبقات $y \in \{-1,1\}$، محدودیت $\boldsymbol{w}^T\boldsymbol{x} + b \geq 1$ برای نقاط داده متعلق به رده بیشتر از +1 و محدودیت $\boldsymbol{w}^T\boldsymbol{x} + b \leq -1$ برای نقاط داده متعلق به رده کمتر از -1 در مسئله بهینه‌سازی در نظر گرفته می‌شود. در حقیقت با یک مسئله بهینه‌سازی مقید (البته از نوع غیرخطی) روبرو هستیم. به عبارتی برای اینکه از ورود نقاط به حاشیه‌ها جلوگیری کنیم، برای هر i داریم: $y_i = 1$ اگر $\boldsymbol{w}^T\boldsymbol{x}_i + b \geq 1$ و $y_i = -1$ اگر $\boldsymbol{w}^T\boldsymbol{x}_i + b \leq -1$. می‌توان این دو محدودیت را ترکیب کرده و آنها را به صورت $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$ بیان کرد. بنابراین مسئله بهینه‌سازی مقید به صورت زیر تعریف می‌شود:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \quad s.t \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$

این مسئله بهینه‌سازی مقید اولیه بوده که دارای مینیمم سراسری است. با استفاده از ضرایب لاگرانژ α_i

می‌توان مسئله فوق را به صورت زیر بازنویسی کرد

$$\min_{\boldsymbol{w},b} L(\boldsymbol{w},b,\alpha_i) = \min_{\boldsymbol{w},b}\left[\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n}\alpha_i\,(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1)\right]$$

این تابع لاگرانژ SVM نامیده می‌شود که با مشتق‌گیری نسبت به پارامترهای مورد علاقه و مساوی صفر قرار دادن داریم

$$\nabla_{\boldsymbol{w}}\, L(\boldsymbol{w},b,\boldsymbol{\alpha}) = \boldsymbol{0} \Longrightarrow \boldsymbol{w} = \sum_{i=1}^{n}\alpha_i\, y_i\boldsymbol{x}_i$$

$$\nabla_b\, L(\boldsymbol{w},b,\boldsymbol{\alpha}) = 0 \Longrightarrow \sum_{i=1}^{n}\alpha_i\, y_i = 0$$

با جایگزینی روابط فوق مسئله دوگان SVM به صورت زیر حاصل می‌شود

$$\max_{\boldsymbol{\alpha}}\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j\, y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_{i=1}^{n}\alpha_i\right] \quad s.t \quad \sum_{i=1}^{n}\alpha_i\, y_i = 0$$

حل مسئله دوگان آسان‌تر است زیرا فقط ضرب در ضرایب لاگرانژ دارد.

بدیهی است مسائل بهینه‌سازی فوق را می‌توان با استفاده از کرنل بازنویسی کرد که در همه موارد فوق کرنل مورد نظر به صورت $K(\boldsymbol{x}_1,\boldsymbol{x}_2) = \boldsymbol{x}_i^T\boldsymbol{x}_j$ (یعنی کرنل چندجمله‌ای از درجه اول) است. در صورتی که لازم باشد از تبدیلات کرنل دیگری استفاده کنیم، می‌توانیم عبارت $\boldsymbol{x}_i^T\boldsymbol{x}_j$ در مسئله بهینه‌سازی را تغییر بدهیم. بنابراین کرنل با تابع انتقال $\phi(.)$ بردار ضرایب $\sum_{i=1}^{n}\alpha_i\, y_i\boldsymbol{\phi}(\boldsymbol{x}_i)$ را در فضای انتقال یافته نتیجه می‌دهد. در نهایت برای برآورد پارامتر اریبی یعنی b از میانگین مشاهدات استفاده می‌کنیم.

7-1 طبقه‌بندی داده‌های Heart با استفاده از روش ماشین بردار پشتیبان

در این بخش به پیاده‌سازی الگوریتم ماشین بردار پشتیبان بر روی مجموعه داده‌ها می‌پردازیم.

در گام اول قبل از انجام هر گونه تحلیلی روی داده‌ها، باید آن‌ها را استاندارد کرد. بخصوص زمانی که داده‌ها چند بُعدی باشند. استفاده از داده‌های استاندارد نشده ممکن است روی نتایج حاصل از تحلیل‌ها اثر نامناسبی داشته باشد و سپس در وهله‌ی بعد، مانند قسمت‌های پیشین داده‌های آموزشی و آزمایشی را تشکیل می‌دهیم.

```
Heart=read.csv("C:/Users/ASUS/Desktop/heart.csv")
str(Heart)
Heart[1:12]=scale(Heart[1:12])
head(Heart)
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list =
FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
head(train_data)
```

از طریق داده‌های آموزشی به تشکیل مدل به روش الگوریتم ماشین بردار پشتیبان و همچنین به واسطه‌ی داده‌های آزمایشی در مراحل بعد، به ارزیابی مدل ایجاد شده توسط داده‌های آموزشی خواهیم پرداخت.

بدین منظور توسط کد زیر بسته‌ی مورد نیاز را فراخوانی و الگوریتم را برمبنای داده‌های آموزشی و با در نظرگرفتن متغیر مربوط به ویژگی فوت به عنوان متغیرپاسخِ مدل خود، تشکیل می‌دهیم.

```
install.packages('e1071')
library(e1071)
```

```
formula=DEATH_EVENT.~

svm_Linear=svm(formula,data=train_data,type="C-classification"
,kernel="linear")
summary(svm_Linear)
```

```
Call:
svm(formula = formula, data = train_data, type = "C-classification", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  104

 ( 51 53 )


Number of Classes:  2

Levels:
 0 1
```

با توجه به خروجی به دست آمده از الگوریتم svm، توسط این روش مجموعه داده‌های ما در دو طبقه متمایز برمبنای دسته‌بندی خطی الگوریتم ماشین بردار پشتیبان با تعداد 104 بردار پشتیبان تقسیم شده‌اند.

اکنون به ارزیابی مدل تشکیل شده از طریق پیش‌بینی کردن طبقه‌ی داده‌های آزمایشی‌ای که در ایجاد مدل دخیل نبوده‌اند، می‌پردازیم.

```
svm.pred=predict(svm_Linear,test_data)
head(svm.pred,200)
```

```
  8  12  13  16  19  20  25  31  38  40  43  45  59  65  78  81  82  85  93  94  99 113 116 125 129 131
  0   1   0   0   1   0   1   1   0   1   0   0   1   0   0   1   0   1   0   1   1   0   0   1   0   0
133 140 149 158 167 175 181 187 189 192 193 197 200 203 204 208 209 211 213 214 215 230 233 241 243 245
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0
248 251 256 257 264 273 298
  0   0   0   0   0   0   0
Levels: 0 1
```

توسط کد دستوری زیر از طریق تشکیل ماتریس درهم‌ریختگی به مقدار دقت و میزان کارایی الگوریتم

دست می‌یابیم.

```
confusionMatrix(table(svm.pred,test_data$DEATH_EVENT))
```

```
Confusion Matrix and Statistics


svm.pred  0  1
       0 36  9
       1  4 10

               Accuracy : 0.7797
                 95% CI : (0.6527, 0.8771)
    No Information Rate : 0.678
    P-Value [Acc > NIR] : 0.05933

                  Kappa : 0.458

 Mcnemar's Test P-Value : 0.26726

            Sensitivity : 0.9000
            Specificity : 0.5263
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.7143
             Prevalence : 0.6780
         Detection Rate : 0.6102
   Detection Prevalence : 0.7627
      Balanced Accuracy : 0.7132

       'Positive' Class : 0
```

باتوجه به خروجی به دست آمده؛ درمی‌یابیم این الگوریتم با احتمال 0/78 به درستی مقادیر متغیر پاسخ مربوط به مجموعه داده‌ی ما را پیش‌بینی می‌کند.

همچنین مقادیر TP، TN، FP و FN به ترتیب 36 ، 10 ، 9 و 4 می‌باشند.

دقت مدل از رابطه زیر نیز قابل محاسبه است :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{36 + 10}{36 + 10 + 9 + 4} = 0.779$$

بنابراین از 59 داده آزمایشی، برای 36 تا از داده‌ها، مقدار متغیر پاسخ (DEATH_EVENT) واقعا صفر

است و مدل SVM نیز این 36 داده را به درستی به رده صفر طبقه‌بندی کرده است. همچنین برای 10 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است و مدل SVM نیز این 10 داده را به درستی به رده یک طبقه‌بندی کرده است. اما برای 9 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است اما مدل SVM این 9 داده را به غلط به رده صفر طبقه‌بندی کرده است. در نهایت برای 4 تا از داده‌ها، مقدار متغیر پاسخ واقعا صفر است اما مدل SVM این 4 داده را به غلط به رده یک طبقه‌بندی کرده است.

8 روش بیز ساده

در حوزه یادگیری ماشین، روش طبقه‌بندی بیز ساده (نایو بیز) با به کارگیری قضیه بیز و در نظر گرفتن فرض استقلال بین متغیرهای پیش‌بین (یا ویژگی‌ها)، یک طبقه‌بندی کننده‌ی مبتنی بر احتمال محسوب می‌شود. در این روش، شیوه یادگیری از نوع یادگیری نظارت شده است.

از روش طبقه‌بندی بیز ساده، اغلب به عنوان یک راهکار ساده برای طبقه‌بندی و تعیین روشی برای تشخیص برچسب اشیا یا نقاط استفاده می‌شود. امروزه از این روش برای حل مسائلی مانند تشخیص ایمیل‌ها یا پیام‌های هرزنامه (spam) استفاده می‌کنند. برای انجام این کار از برآورد تابع احتمال و از طریق فراوانی یا فراوانی نسبی کلمات در اسناد متنی استفاده می‌شود.

برای توضیح تئوری این روش فرض کنید $\boldsymbol{X} = (X_1, \dots, X_p)$ یک بردار تصادفی و $\boldsymbol{x} = (x_1, \dots, x_p)$ مقدار مشاهده شده‌ی این بردار تصادفی باشد. بردار $\boldsymbol{X} = (X_1, \dots, X_p)$ نشان‌دهنده بردار ویژگی‌ا (یا متغیرهای پیش‌بین) است. به این روش، بیز ساده گفته می‌شود زیرا فرض بر این است که مولفه‌های بردار تصادفی $\boldsymbol{X} = (X_1, \dots, X_p)$ به عنوان متغیرهای تصادفی از یکدیگر مستقل هستند (البته به شرط متغیری که نشان‌دهنده برچسب است). همچنین فرض کنید Y نیز یک متغیر تصادفی (گسسته) با مقادیر (یا برچسب‌های) $\{c_1, \dots, c_m\}$ باشد. در این صورت بنابر قضیه بیز می‌توان نوشت

$$P(Y = c_j | \boldsymbol{X} = \boldsymbol{x}) = \frac{P(Y = c_j, \boldsymbol{X} = \boldsymbol{x})}{P(\boldsymbol{X} = \boldsymbol{x})} = \frac{P(Y = c_j) P(\boldsymbol{X} = \boldsymbol{x} | Y = c_j)}{P(\boldsymbol{X} = \boldsymbol{x})}$$

که در آن $P(Y = c_j | \boldsymbol{X} = \boldsymbol{x})$ احتمال پسین برای طبقه با برچسب c_j، $P(Y = c_j)$ احتمال پیشین مربوط به طبقه با برچسب c_j، $P(\boldsymbol{X} = \boldsymbol{x} | Y = c_j)$ تابع درستنمایی و $P(\boldsymbol{X} = \boldsymbol{x})$ نیز احتمال پیشین مربوط به ویژگی‌ها است. با استفاده از قاعده «انتخاب تصمیم محتمل‌تر»، پس از مشاهده مقدار ویژگی‌ها یعنی بردار $\boldsymbol{x} = (x_1, \dots, x_p)$، از بین تصمیمات مختلف، آن تصمیمی را اتخاذ می‌کنیم که بیشترین احتمال رخداد را دارد. به بیان دیگر، مشاهده $\boldsymbol{x} = (x_1, \dots, x_p)$ به طبقه‌ای مانند $c \in \{c_1, \dots, c_m\}$ تعلق می‌گیرد اگر

$$c = \hat{y} = \arg\max_{y \in \{c_1, \dots, c_m\}} P(Y = y | \boldsymbol{X} = \boldsymbol{x})$$

به عبارت دیگر به دنبال طبقه‌ای مانند $c \in \{c_1, \dots, c_m\}$ هستیم که در رابطه زیر صدق کند

$$P(Y = y | \boldsymbol{X} = \boldsymbol{x}) \leq P(Y = c | \boldsymbol{X} = \boldsymbol{x});\ \forall y \in \{c_1, \dots, c_m\}$$

چون ویژگی‌ها به شرط معلوم بودن طبقه از هم مستقل هستند داریم

$$P(\boldsymbol{X} = \boldsymbol{x} | Y = y) = P\big(X_1 = x_1, X_2 = x_2, \dots, X_p = x_p | Y = y\big) = \prod_{i=1}^{p} P(X_i = x_i | Y = y)$$

حال با توجه به قضیه بیز داریم

$$P(Y = y | \boldsymbol{X} = \boldsymbol{x}) = \frac{P(Y = y, \boldsymbol{X} = \boldsymbol{x})}{P(\boldsymbol{X} = \boldsymbol{x})} = \frac{P(Y = y) P(\boldsymbol{X} = \boldsymbol{x} | Y = y)}{P(\boldsymbol{X} = \boldsymbol{x})}$$

$$= \frac{P(Y = y) \prod_{i=1}^{p} P(X_i = x_i | Y = y)}{P(\boldsymbol{X} = \boldsymbol{x})} \propto P(y) \prod_{i=1}^{p} P(X_i = x_i | Y = y)$$

بنابراین مقدار y را طوری پیدا می‌کنیم که عبارت $P(y) \prod_{i=1}^{p} P(X_i = x_i | Y = y)$ ماکسیمم شود.

یعنی برچسب طبقه مربوط به مشاهده $\boldsymbol{x} = (x_1, \dots, x_p)$ به وسیله رابطه زیر تعیین می‌شود

$$c = \hat{y} = \arg\max_{y \in \{c_1, \dots, c_m\}} P(Y = y) \prod_{i=1}^{p} P(X_i = x_i | Y = y)$$

بدین منظور ابتدا می‌بایست به ازای همه مقادیر y، مقدار $P(X_i = x_i | Y = y)$ را توسط داده‌ها برآورد کرد. به عنوان مثال با استفاده از روش برآورد احتمال توسط فراوانی نسبی داریم

$$\hat{P}(X_i = x_i | Y = y) = \frac{\#\{X_i = x_i, Y = y\}}{\#\{Y = y\}}$$

که نماد # نشان‌دهنده فراوانی (تعداد) در مجموعه داده‌های آموزشی است.

در صورتی که ویژگی X_i به صورت کمی باشد، با در نظر گرفتن توزیع مناسب برای $X_i = x_i | Y = y$ و جایگزینی $P(X_i = x_i | Y = y)$ با تابع چگالی احتمال شرطی $f_{X_i | Y = y}(x_i)$ می‌توان پارامترهای مجهول را محاسبه کرده و به جای $\hat{P}(X_i = x_i | Y = y)$ از $\hat{f}_{X_i | Y = y}(x_i)$ استفاده کرد.

اگر ویژگی‌ها از نوع پیوسته باشند، می‌توان توزیع نرمال را برای آنها در نظر گرفت که در این حالت به

این روش، روش طبقه‌بندی بیز ساده‌ی نرمال (یا گاوسی) می‌گویند. در این حالت ویژگی‌ها در هر طبقه دارای توزیع نرمال هستند. به این ترتیب اگر m طبقه به صورت $\{c_1, \dots, c_m\}$ داشته باشیم، می‌توانیم برای هر طبقه میانگین و واریانس داده‌های آموزشی را محاسبه کرده و پارامترهای توزیع نرمال را برآورد کنیم. اگر $\mu_i^{(y)}$ و $\sigma_i^{2(y)}$ به ترتیب نشان‌هنده میانگین و واریانس ویژگی X_i، زمانی که $Y = y$ است، باشند، در این صورت داریم

$$X_i|Y = y \sim N\left(\mu_i^{(y)}, \sigma_i^{2(y)}\right)$$

اگر $\mu_i^{(y)}$ و $\sigma_i^{2(y)}$ را با استفاده از داده‌های آموزشی به صورت

$$\hat{\mu}_i^{(y)} = \bar{x}_i^{(y)}, \hat{\sigma}_i^{2(y)} = s_i^{2(y)}$$

برآورد کنیم، در این صورت داریم

$$X_i|Y = y \sim N\left(\bar{x}_i^{(y)}, s_i^{2(y)}\right)$$

و بنابراین توزیع $X_i|Y = y$ برآورد می‌شود و لذا برای محاسبه $P(X_i = x_i|Y = y)$ می‌توان از آن استفاده کرد. در ادامه روش بیز ساده و نحوه به کارگیری فرمول‌های بالا در یادگیری ماشین با مثال شرح داده می‌شود.

مثال: از 20 دانشجو خواسته شده که در ابتدای روز اسمی را به خاطر بسپارند. متغیر پیش‌بین زمان خواب شبانه بر حسب دقیقه است. همچنین متغیر پاسخ «موفقیت یا عدم موفقیت در یادآوری اسم در صبح فردا» است (که دارای m=2 سطح است). نتایج در جدول زیر آورده شده است. موفقیت در یادآوری اسم را با یک، و عدم موفقیت را با صفر نشان داده‌ایم.

زمان خواب (x)	30	45	60	75	90	105	105	120	135	150
یادآوری(y)	0	0	0	0	0	0	1	0	1	0
زمان خواب (x)	165	180	195	210	240	255	270	285	300	330
یادآوری (y)	1	0	1	0	1	1	1	1	1	1

با استفاده از این داده‌ها به عنوان داده‌های آموزشی، محاسبات مربوط به روش بیز ساده را انجام می‌دهیم. با توجه به اینکه تنها یک ویژگی (x) داریم که کمی و پیوسته است، بنابراین فرض می‌کنیم توزیع داده‌های آن هم به ازای y=0 و هم به ازای y=1 نرمال هستند. یعنی

$$X|Y=0 \sim N\left(\mu^{(0)}, \sigma^{2(0)}\right)$$
$$X|Y=1 \sim N\left(\mu^{(1)}, \sigma^{2(1)}\right)$$

ابتدا با استفاده از داده‌ها، چهار پارامتر $\mu^{(0)}, \sigma^{2(0)}$ و $\mu^{(1)}, \sigma^{2(1)}$ را به صورت زیر برآورد می‌کنیم.

$$n_0 = n_1 = 10$$

$$\hat{\mu}^{(0)} = \bar{x}^{(0)} = \frac{1}{n_0} \sum_{\{i:y_i=0\}} x_i = \frac{1}{10}(30 + 45 + \cdots + 210) = 106.5$$

$$\hat{\mu}^{(1)} = \bar{x}^{(1)} = \frac{1}{n_1} \sum_{\{i:y_i=1\}} x_i = \frac{1}{10}(105 + 135 + \cdots + 330) = 228$$

توجه می‌کنیم که n_0 تعداد داده‌هایی است که برای آنها مقدار متغیر y برابر با صفر است و n_1 نیز تعداد داده‌هایی است که برای آنها مقدار متغیر y برابر با یک است. همچنین برای محاسبه $\bar{x}^{(0)}$ فقط از x هایی استفاده می‌کنیم که برای آنها مقدار متغیر y برابر با صفر است و برای محاسبه $\bar{x}^{(1)}$ نیز فقط از x هایی استفاده می‌کنیم که برای آنها مقدار متغیر y برابر با یک است. واریانس‌ها نیز به طور مشابه تعریف می‌شوند. پس داریم:

$$\begin{aligned}\hat{\sigma}^{2(0)} = s^{2(0)} &= \frac{1}{n_0 - 1} \sum_{\{i:y_i=0\}} \left(x_i - \bar{x}^{(0)}\right)^2 \\ &= \frac{1}{9}[(30 - 106.5)^2 + (45 - 106.5)^2 + \cdots + (210 - 106.5)^2] \\ &= 3472.5\end{aligned}$$

$$\hat{\sigma}^{2(1)} = s^{2(1)} = \frac{1}{n_1 - 1} \sum_{\{i:y_i=1\}} \left(x_i - \bar{x}^{(1)}\right)^2$$

$$= \frac{1}{9}[(105-228)^2 + (135-228)^2 + \cdots + (330-228)^2] = 5590$$

بنابراین داریم

$$X|Y = 0 \sim N(106.5, 3472.5)$$

$$X|Y = 1 \sim N(228, 5590)$$

در نتیجه قاعده تصمیم به صورت زیر است: مشاهده x به طبقه صفر (یعنی حالتی که y=0) تعلق می‌گیرد اگر

$$P(Y=0)P(X_i = x_i|Y=0) \geq P(Y=1)P(X_i = x_i|Y=1)$$

و در غیر این صورت به طبقه یک (یعنی حالتی که y=1) تعلق می‌گیرد.

در یادگیری ماشین، به دست آوردن رابطه بالا را اصطلاحا یادگیری می‌گویند. در واقع داده‌های داده شده در بالا به عنوان داده آموزشی در نظر گرفته می‌شوند و با استفاده از آنها به ماشین آموزش داده می‌شود تا این روش را یاد بگیرد. پس از آموزش و یادگیری مدل بالا، ماشین از این مدل برای پیش‌بینی مقدار متغیر برچسب‌دار (که در اینجا y است) استفاده می‌کند.

در اینجا اگر فرض کنیم توزیع پیشینِ Y یکنواخت است، یعنی

$$P(Y=0) = P(Y=1) = \frac{1}{2}$$

در این صورت مثلا برای دانشجویی که $x = 200$ دقیقه خواب داشته است داریم

$$P(Y=0)P(X=200|Y=0) = P(Y=0)f(200|Y=0)$$

$$= \frac{1}{2}\frac{1}{\sqrt{2\pi(3472.5)}}\exp\left[-\frac{1}{2(3472.5)}(200-106.5)^2\right] = 0.00096$$

و

$$P(Y=0)P(X=200|Y=0) = P(Y=1)f(200|Y=0)$$

$$= \frac{1}{2}\frac{1}{\sqrt{2\pi(5590)}}\exp\left[-\frac{1}{2(5590)}(200-228)^2\right] = 0.00249$$

مشاهده می‌کنیم که

$$P(Y=0)P(X=200|Y=0) < P(Y=0)P(X=200|Y=0)$$

بنابراین دانشجو با میزان خواب $x=200$ در طبقه یک (یعنی حالتی که y=1 است) قرار می‌گیرد. ■

نکته‌ای که در مورد الگوریتم طبقه‌بندی کننده بیز ساده می‌توان گفت این است که این الگوریتم یک الگوریتم برای پیاده‌سازی سریع است. همچنین این روش نسبت به سایر روش‌های طبقه‌بندی به داده آموزشی کمتری نیاز دارد.

نکته دیگری که در مورد این الگوریتم می‌توان گفت این است که در عمل ممکن است برخی از ویژگی‌ها به جای توزیع نرمال، دارای توزیع دیگری باشند، در این صورت بایستی $P(X_i = x_i|Y=y)$ را متناسب با همان توزیع محاسبه یا برآورد کرد. در برخی موارد می‌توان با استفاده از تبدیلات مناسب (مثلا تبدیل باکس-کاکس)، توزیع برخی ویژگی‌ها را به توزیع نرمال تبدیل کرد. در این حالت می‌توان از فرمول‌های ذکر شده در بالا (برای حالت نرمال) را روی این متغیرهای تبدیل یافته اعمال کرد.

مزایا و معایب روش دسته‌بند بیز ساده

دسته‌بندی کردن داده‌های آزمایشی آسان و سریع است. همچنین زمانی که تعداد دسته‌ها از دو بیشتر باشد نیز عملکرد خوبی از خودش نشان می‌دهد.

- تا زمانی که شرط مستقل بودن برقرار باشد، یک دسته‌بندی‌کننده بیز ساده عملکرد بهتری نسبت به مدل‌های دیگر مانند رگرسیون لوژستیک دارد و به حجم آموزش کمی نیاز دارد.

- در حالتی که ورودی‌هایمان دسته‌بندی شده باشند این روش عملکرد بهتری نسبت به حالتی دارد که ورودی‌هایمان عدد باشند. برای حالتی که ورودی عدد باشد معمولاً فرض می‌شود که از توزیع نرمال پیروی می‌کنند. (که فرض قوی‌ای است)

علاوه بر مزایایی که این روش دسته‌بندی دارد معایبی نیز دارد، از جمله:

- یکی از مهم‌ترین ایرادات طبقه‌بندی بیز ساده، استقلال قوی ویژگی‌های آن است، زیرا در زندگی

واقعی تقریبا غیرممکن است که یک مجموعه از ویژگی‌هایی داشت که کاملا مستقل از یکدیگر باشند.

- در صورتی که ورودی‌مان دسته‌بندی شده باشد و در مرحله یادگیری دسته‌ای وجود داشته باشد که دسته‌بندی بیز هیچ داده‌ای از آن دسته را مشاهده نکرده باشد، دسته‌بندی بیز احتمالی برابر صفر برای آن دسته در نظر می‌گیرد و قادر به دسته‌بندی کردن نخواهد بود. برای حل این مشکل می‌توان از تکنیک‌های هموارسازی مانند تخمین‌گر لاپلاس استفاده کرد.

گام‌های رده بندی بیزی

- **گام اول** اگر $X= (x_1 , x_2, \dots, x_n)$ برداری از n ویژگی را بیان می‌کند که به صورت متغیرهای مستقل می‌باشند. همانطور که از قبل می‌دانیم احتمال رخداد y_k (بیانگر یکی از حالت‌های کلاس رخدادهای مختلف به ازای kهای متفاوت) به شرط متغیرهای مستقل یا همان ویژگی‌ها، بنابر قضیه بیز، به صورت زیر می‌باشد:

$$P(y_k| X) = \frac{P(X| y_k)\, P(y_k)}{P(X)}$$

- **گام دوم** حال اگر فرض کنیم هر متغیری نسبت به متغیرهای دیگر به شرط دسته y_k مستقل است یعنی $P(x_i| x_{i+1} , \dots, x_n, y_k) = p(x_i | y_k)$ به نتیجه پایین می‌رسیم:

$$\propto P(y_k) \prod_{i=1}^{n} p(x_i | y_k) \;\; P(y_k| X)$$

- **گام سوم** هدف پیدا کردن محتمل‌ترین دسته است که با استفاده از فرمول زیر به این منظور دست پیدا می‌کنیم:

$$\hat{y} = \text{argmax} \;\; P(y_k) \prod_{i=1}^{n} P(x_i | y_k)$$

- **گام چهارم** در نهایت با توجه به توزیع‌های مختلفی که ممکن است نمونه تصادفی داشته باشد یعنی $P(x_i \mid y_k)$ ، می‌توان پارامترها را محاسبه و یا برآورد کرد.

8-1 طبقه‌بندی داده‌های Heart با استفاده از روش بیز ساده

مانند قسمت‌های پیشین ابتدا داده‌ها را فراخوانی و اقدامات لازم را برای آماده‌سازی بستر دسته‌بندی بیزی انجام می‌دهیم

```
Heart=read.csv("C:/Users/ASUS/Desktop/heart.csv")
Heart$DEATH_EVENT <- factor(Heart$DEATH_EVENT)
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
head(train_data)
```

توسط کد زیر الگوریتم دسته‌بند بیزی را برمبنای داده‌های آموزشی و با در نظرگرفتن متغیرپاسخ، تشکیل می‌دهیم.

```
nv=naiveBayes(DEATH_EVENT~.,train_data)
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.6791667 0.3208333

Conditional probabilities:
   age
Y       [,1]     [,2]
  0 59.03067 10.49187
  1 65.60606 13.21258

   anaemia
Y        [,1]      [,2]
  0 0.4417178 0.4981219
  1 0.4545455 0.5011947
```

```
   creatinine_phosphokinase
Y       [,1]      [,2]
  0 545.6380  775.5595
  1 717.0519 1463.1186

   diabetes
Y       [,1]      [,2]
  0 0.4294479 0.4965228
  1 0.3896104 0.4908597

   ejection_fraction
Y       [,1]     [,2]
  0 40.48466 10.25800
  1 32.29870 12.23354

   high_blood_pressure
Y       [,1]      [,2]
  0 0.3128834 0.4650962
  1 0.3896104 0.4908597

   platelets
Y       [,1]      [,2]
  0 265284.2 102484.15
  1 261985.2  99554.98

   serum_creatinine
Y       [,1]      [,2]
  0 1.179325 0.6829277
  1 1.914935 1.5900071

   serum_sodium
Y       [,1]     [,2]
  0 137.4663 3.839821
  1 135.2078 4.918648
```

```
   sex
Y         [,1]      [,2]
  0 0.6625767 0.4742878
  1 0.6753247 0.4713240

   smoking
Y         [,1]      [,2]
  0 0.3251534 0.4698757
  1 0.3376623 0.4760139

   time
Y         [,1]     [,2]
  0 157.31288 70.23334
  1  74.88312 63.80706
```

احتمالات شرطی به طور مجزا برای هر متغیر ورودی آورده شده‌است

در مرحله‌ی بعدی به ارزیابی مدل تشکیل شده می‌پردازیم. بدین منظور ابتدا توسط کد دستوری زیر به پیش‌بینی طبقه‌ی داده‌های آزمایشی که در تشکیل مدل دخیل نبوده‌اند، اقدام می‌کنیم

```
nvp=predict(nv,newdata=test_data)
head(nvp,300)
```

```
 [1] 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
[51] 0 0 0 0 0 0 0 0 0 0
Levels: 0 1
```

در آخر دقت مدل به‌دست آمده را ارزیابی می‌کنیم.

```
confusionMatrix(table(nvp,test_data$DEATH_EVENT))
```

```
Confusion Matrix and Statistics


nvp  0  1
  0 38 13
  1  2  6

               Accuracy : 0.7458
                 95% CI : (0.6156, 0.8502)
    No Information Rate : 0.678
    P-Value [Acc > NIR] : 0.164890

                  Kappa : 0.3134

 Mcnemar's Test P-Value : 0.009823

            Sensitivity : 0.9500
            Specificity : 0.3158
         Pos Pred Value : 0.7451
         Neg Pred Value : 0.7500
             Prevalence : 0.6780
         Detection Rate : 0.6441
   Detection Prevalence : 0.8644
      Balanced Accuracy : 0.6329

       'Positive' Class : 0
```

بنابراین از 59 داده آزمایشی، برای 38 تا از داده‌ها، مقدار متغیر پاسخ (DEATH_EVENT) واقعا صفر است و مدل بیز ساده نیز این 38 داده را به درستی به رده صفر طبقه‌بندی کرده است. همچنین برای 6 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است و مدل بیز ساده نیز این 6 داده را به درستی به رده یک طبقه‌بندی کرده است. اما برای 13 تا از داده‌ها، مقدار متغیر پاسخ واقعا یک است اما مدل بیز ساده

این 13 داده را به غلط به رده صفر طبقه‌بندی کرده است. در نهایت برای 2 تا از داده‌ها، مقدار متغیر پاسخ واقعا صفر است اما مدل بیز ساده این 2 داده را به غلط به رده یک طبقه‌بندی کرده است.

باتوجه به خروجی به دست آمده؛ درمی‌یابیم این الگوریتم با احتمال 0/74 به درستی مقادیر متغیر پاسخ مربوط به مجموعه داده‌ی ما را پیش‌بینی می‌کند.

همچنین مقادیر TP، TN، FP و FN به ترتیب 38 ، 6 ، 13 و 2 می‌باشند.

برای به دست آوردن دقت مدل با استفاده از رابطه زیر داریم:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{38 + 6}{38 + 6 + 13 + 2} = 0.745$$

مقدار به‌دست آمده برابر با 0/75 است و این یعنی این مدل با احتمال 0/75 مقادیر صحیح متغیر پاسخ را به ما می‌دهد

9 شبکه‌های عصبی

شبکه‌های عصبی مصنوعی[1] (ANN) یا به زبان ساده‌تر شبکه‌های عصبی، سیستم‌ها و روش‌های محاسباتی نوین برای یادگیری ماشین، نمایش دانش و در انتها اعمال دانش به دست آمده در جهت بیش‌بینی پاسخ‌های خروجی در سامانه‌های پیچیده هستند. ایده اصلی این گونه شبکه‌ها تا حدودی الهام گرفته از شیوه کارکرد سیستم عصبی زیستی برای پردازش داده‌ها و اطلاعات به منظور یادگیری و ایجاد دانش می‌باشد. عنصر کلیدی این ایده، ایجاد ساختارهایی جدید برای سامانه پردازش اطلاعات است.

این سیستم از شمار زیادی عناصر پردازشی فوق‌العاده بهم‌پیوسته با نام نورون تشکیل شده‌اند که برای حل یک مساله با هم هماهنگ عمل می‌کنند و توسط سیناپس‌ها (ارتباطات الکترومغناطیسی) اطلاعات را منتقل می‌کنند. در این شبکه‌ها اگر یک سلول آسیب ببیند، بقیه سلول‌ها می‌توانند نبود آن را جبران کرده، و نیز در بازسازی آن سهیم باشند. این شبکه‌ها قادر به یادگیری‌اند؛ مثلاً با اعمال سوزش به سلول‌های عصبی لامسه، سلول‌ها یاد می‌گیرند که به طرف جسم داغ نروند و با این الگوریتم سیستم می‌آموزد که خطای خود را اصلاح کند. یادگیری در این سیستم‌ها به صورت تطبیقی صورت می‌گیرد، یعنی با استفاده از مثال‌ها، وزن سیناپس‌ها به گونه‌ای تغییر می‌کند که در صورت دادن ورودی‌های جدید، سیستم پاسخ درستی تولید کند.

[1] Artificial Neural Networks

9-1 طبقه‌بندی داده‌های Heart با استفاده از شبکه‌های عصبی

اکنون می‌خواهیم مدلی از یک شبکه عصبی را، بر روی مجموعه داده‌ی مورد مطالعه‌مان پیاده‌سازی کنیم. مقیاس‌دهی داده‌ها بسیار مهم و ضروری است، چرا که درغیراین‌صورت ممکن است یک متغیر تنها به‌خاطر مقیاسی که دارد، تأثیر زیادی بر پیش‌گویی بگذارد. استفاده از متغیرهایی که تعیین مقیاس نشده‌اند می‌تواند منجر به نتایج بی‌معنی شود. ما از«minmax normalization» ، برای مقیاس دهی داده‌ها استفاده می‌کنیم.

```
Heart=read.csv("C:/Users/ASUS/Desktop/heart.csv")
Heart$DEATH_EVENT <- factor(Heart$DEATH_EVENT)
normalize <- function(x){}
  return ((x - min(x)) / (max(x) - min(x))){
Heart[1:12]=normalize(Heart[1:12])
```

داده‌ها را به مانند قبل به دو مجموعه آموزش و آزمایش تقسیم می‌کنیم. از مجموعه آموزش، برای پیدا کردن رابطه بین متغیرهای وابسته و مستقل استفاده می‌شود، درحالی‌که مجموعه آزمایش، کارایی مدل را ارزیابی می‌کند.

```
set.seed(123)
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list =
FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
head(train_data)
```

حالا یک شبکه عصبی را بر روی داده‌هایمان پیاده‌سازی می‌کنیم. از «کتابخانه عصبی» برای تجزیه‌وتحلیل استفاده می‌کنیم.

```
library(nnet)
library(caret)
neu=nnet(DEATH_EVENT~.,data=train_data,size=3,maxit=10000,decay=0.001,r
ang=0.05)
```

```
# weights:  43
initial  value 169.022762
iter  10 value 150.576573
iter  20 value 150.571996
iter  30 value 150.569876
iter  40 value 150.552101
iter  50 value 150.540729
iter  60 value 150.531092
iter  70 value 150.514969
iter  80 value 150.441282
iter  90 value 150.140575
iter 100 value 149.542098
iter 110 value 149.429902
iter 120 value 149.365169
iter 130 value 149.340548
iter 140 value 149.330818
iter 150 value 149.327803
iter 160 value 149.324257
iter 170 value 149.322272
iter 180 value 149.322024
iter 190 value 149.321626
iter 200 value 149.321090
iter 210 value 149.320531
iter 220 value 149.320008
iter 230 value 149.319734
final  value 149.319553
converged
```

همان‌طور که مشاهده می‌شود، تعداد وزن‌هایی که می‌بایست توسط الگوریتم براورد شود 43 تا می‌باشد. همچنین مقادیر مختلف وزن‌ها در تکرارهای مختلف الگوریتم مشاهده می‌شود و در نهایت در تکرار 230 همگرایی توسط الگوریتم تشخیص داده شده و بهترین برآورد برای پارامتر وزن 149.32 می‌باشد. در مرحله‌ی بعدی به ارزیابی مدل تشکیل شده می‌پردازیم. بدین منظور ابتدا توسط کد دستوری زیر به پیش‌بینی طبقه‌ی داده‌های آزمایشی که در تشکیل مدل دخیل نبوده‌اند، اقدام می‌کنیم.

```
neupred=predict(neu,newdata=test_data ,type="class")
neupred=factor(neupred)
```

توسط کد دستوری زیر از طریق تشکیل ماتریس درهم‌ریختگی به مقدار دقت و میزان کارایی الگوریتم دست می‌یابیم.

```
tableneu=table(test_data $DEATH_EVENT,neupred,dnn=c("actual","predicted"))
accuracyneu=sum(diag(tableneu)/sum(tableneu))
paste(floor(accuracyneu*100),"%")
confusionMatrix(neupred, test_data $DEATH_EVENT)
```

```
> paste(floor(accuracyneu*100),"%")
[1] "67 %"
```

یعنی از کل داده آزمایشی، حدود 68 درصد آنها به درستی طبقه‌بندی شده‌اند.

10 مقایسه روش‌های طبقه‌بندی

در این بخش روش‌هایی که برای طبقه‌بندی داده‌های Heart مورد استفاده قرار گرفت را با هم مقایسه می‌کنیم. معیاری که برای مقایسه روش‌ها مورد استفاده قرار می‌دهیم، درصد پیش‌گویی صحیح برای پیش‌بینی طبقه مربوط به داده‌های آزمایشی است. در جدول زیر درصد پیش‌بینی درست برای هر تمام روش‌ها آورده شده است. لازم به ذکر است که نتایج درخت تصمیم و جنگل تصادفی را که در پروژه قبلی انجام شد به این جدول اضافه کرده‌ایم.

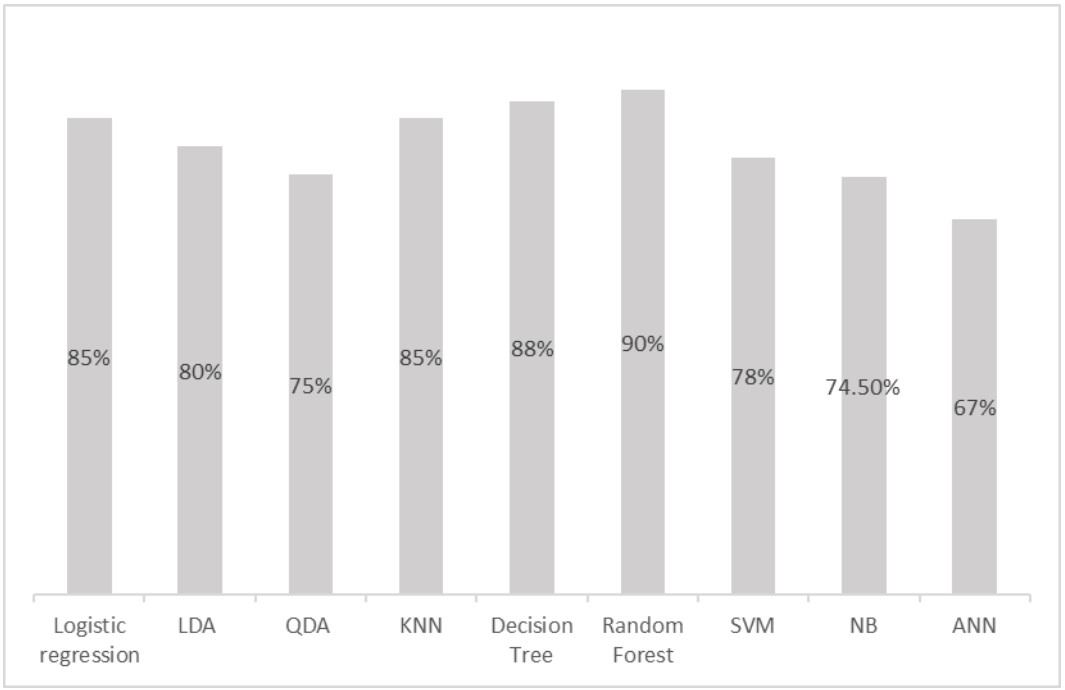
روش	درصد پیش‌بینی درست
رگرسون لجستیک	% 85
ممیزی خطی	% 80
ممیزی درجه دوم	% 75
KNN	% 85
درخت تصمیم	% 88
جنگل تصادفی	% 90
SVM	% 78
بیز ساده	% 74.5
شبکه‌های عصبی	% 67

با توجه به جدول فوق مشخص می‌شود که روش جنگل تصادفی داری بالاترین درصد پیش‌بینی درست در مجموعه داده‌های آزمایشی است و بنابراین برای این مجموعه داده‌ها، روش جنگل تصادفی به عنوان بهترین روش برای طبقه‌بندی انتخاب می‌شود.

11 نتیجه‌گیری

در این مرحله باید ارزیابی شود که مدل‌های پیشنهاد شده تا چه حد می‌توانند به اهداف مسئله و تحقیق ما کمک کنند. اگر زمان‌بندی و بودجه پروژه اجازه دهد بهتر است مدل در دنیای واقعی آزمایش شود. نتایج آزمایش کمک می‌کند تا مدل‌های ارائه شده ارزیابی شوند و شاید اطلاعات جدیدتری به دست آید که به کامل‌تر شدن مدل‌ها کمک کنند.

این مرحله بسیار مهم و چالش‌برانگیز است. در این مرحله تیم پروژه باید نشان دهد که دانش به‌دست‌آمده از مدل می‌تواند الگوها و روابط جدیدی را به تصمیم‌گیر نشان دهد که با استفاده از آن ارزش جدیدی برای کسب‌وکار خلق می‌شود. این مانند حل کردن یک معما است. آنچه از فرآیند داده‌کاوی به دست می‌آید تنها بخشی از یک کل است. مدیران و تحلیل‌گران باید نتایج را در فضای کلی آن کسب‌وکار مورد ارزیابی قرار دهند. در اینجا دانش کسب‌وکار کمک بسیاری به بررسی خروجی‌های مدل می‌کند.



در این پروژه مجموعه داده‌های قلب (Heart) را در نظر گرفته و طبقه‌بندی مشاهدات را به روش‌های مختلف رگرسیون لجستیک، درخت تصمیم، ممیزی خطی، ممیزی درجه دوم، KNN، SVM، جنگل تصادفی، بیزساده و شبکه‌های عصبی انجام دادیم. سپس برای مقایسه این روش‌ها از معیار درصد پیش-بینی درست استفاده کردیم و مشخص شد که روش جنگل تصادفی با 90 درصد پیش‌بینی درست، داری بالاترین درصد پیش‌بینی درست و شبکه‌های عصبی دارای کمترین میزان دقت در مجموعه داده‌های آزمایشی است و بنابراین برای این مجموعه داده‌ها، روش جنگل تصادفی به عنوان بهترین روش انتخاب شد.

همچنین برای بررسی عوامل تاثیرگذار روی متغیر پاسخ که همان فوت بیماران می‌باشد؛ باتوجه به بررسی‌هایی که در قسمت‌های قبل انجام شد می‌توانیم بگوییم **بیشترین متغیر تاثیرگذار** بر روی متغیرپاسخ از بین متغیرهای ورودی **متغیر مدت زمانی است که بیمار در بیمارستان بستری** بوده است.

پیوست

کدهای ار استفاده شده در این پروژه

```
library(tidyr)
library(dplyr)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(Boruta)#
library(ROCR)
library(gbm)
library(corrplot)
library(stringr)
##################################
Heart=read.csv("C:/Users/acer/Desktop/heart.csv")
colSums(is.na(Heart))
Removing missing values #
clean_data <- na.omit(Heart)
sum(duplicated(Heart))
Heart$DEATH_EVENT <- factor(Heart$DEATH_EVENT)
glimpse(Heart)
summary(Heart)
numeric_data <- Heart[, sapply(Heart, is.numeric)]
if (ncol(numeric_data) > 1) { # check if we have at least two numeric variables
cor_matrix <- cor(numeric_data, method = "pearson")
print("Correlation Matrix:")
print(cor_matrix)
library(corrplot)
corrplot(cor_matrix, method = "circle")
} else {
```

```
print("Not enough numeric variables for a correlation matrix.")
{


library(GGally)
%<>% Heart
ggpairs(columns = c("age", "creatinine_phosphokinase", "ejection_fraction",
,("platelets", "serum_creatinine", "serum_sodium
+ (mapping = ggplot2::aes(color = DEATH_EVENT)
()ggplot2::theme_light
#####################################
%<>% Heart
+ ggplot(aes(x = age))
,geom_histogram(binwidth = 5
,"color = "white
+ (alpha = 0.5
+ labs(title = "Age Distribution")
scale_x_continuous(breaks = seq(40,100,10))
#####################################
(123)set.seed
train_index <- createDataPartition(Heart$DEATH_EVENT, p = 0.8, list = FALSE)
train_data <- Heart[train_index, ]
test_data <- Heart[-train_index, ]
head(train_data)
tree_model <- rpart(DEATH_EVENT ~ ., data = train_data, method = "class")
rpart.plot(tree_model, main = "Decision Tree for Heart Failure", box.palette = "RdBu")
```

```
pred <- predict(tree_model, newdata = test_data, type = "class")
confusionMatrix(pred, test_data$DEATH_EVENT)
pred <- predict(tree_model, newdata = test_data, type = "prob")
pred_obj <- prediction(pred[, 2], test_data$DEATH_EVENT)
perf_obj <- ROCR::performance(pred_obj, measure = "tpr", x.measure = "fpr")
,"plot(perf_obj, main = "ROC Curve for Logistic Regression Model
(xlab = "False Positive Rate", ylab = "True Positive Rate", colorize=T, lwd = 2
abline(a = 0, b = 1, lwd = 1.5, col = "gray")
auc <- ROCR::performance(pred_obj, "auc")
[[1]]auc <- auc@y.values
legend("bottomright", paste("AUC = ", round(auc, 3)), bty = "n", cex = 0.8)
########################################
p1<-ggplot(Heart, aes(x=age))+geom_bar(fill="lightblue")+ labs(x="Age Group")+
theme_minimal(base_size=10)
p2<-ggplot(Heart, aes(x=sex))+geom_bar(fill="indianred3")+ labs(x="Sex")+
theme_minimal(base_size=10)
p3<-ggplot(Heart, aes(x=smoking))+geom_bar(fill="seagreen2")+ labs(x="Smoking")+
theme_minimal(base_size=10)
+p4<-ggplot(Heart, aes(x=diabetes))+geom_bar(fill="orange2")
labs(x="Diabetes Status")+ theme_minimal(base_size=10)
library(ggplot2)
library(patchwork)
devtools::install_github("thomasp85/patchwork")
(p1+p2+p3 +p4)+plot_annotation(title="Demographic and Histology Distribution")
####################################
+ ggplot(Heart, aes(x=sex, fill=sex))
+ ()geom_bar
+ geom_text(stat='count', aes(label=..count..), vjust=2, colour = "white")
```

```
+ labs(title = "Heart Failure by Gender")

theme(plot.title = element_text(size=14, color = "red"))
```