پیش بینی میانه قیمت مسکن

فایل BostonHousing.csv حاوی اطلاعاتی در مورد بیش از ۵۰۰ سرشماری در بوستون است که برای هر تراکت چندین متغیر ثبت شده است.

آخرین ستون (CAT.MEDV) از MEDV مشتق شده است، به طوری که اگر MEDV > 30 باشد، مقدار ۱ و در غیر این صورت ۰ به دست می آید.

با توجه به اطلاعات ۱۲ ستون اول، هدف را از پیش بینی مقدار میانه (MEDV) یک تراکت در نظر بگیرید.

داده ها را به مجموعه های آموزشی (۶۰٪) و اعتبارسنجی (۴۰٪) تقسیم کنید.

الف. یک پیش‌بینی k-NN با تمام ۱۲ پیش‌بینی‌کننده انجام دهید (ستون CAT.MEDV را نادیده بگیرید)، مقادیر k را از ۱ تا ۵ امتحان کنید. مطمئن شوید که داده‌ها را نرمال کرده‌اید و تابع knn() را از بسته class به جای بسته FNN انتخاب کنید.

برای اطمینان از اینکه R از بسته کلاس استفاده می کند (زمانی که هر دو بسته بارگذاری می شوند)، از class::knn() استفاده کنید.

بهترین k چیست؟ چه مفهومی داره؟

ب MEDV را برای یک تراکت با اطلاعات زیر با استفاده از بهترین k پیش‌بینی کنید:

ج اگر از الگوریتم k-NN فوق برای امتیاز دهی به داده های آموزشی استفاده کنیم، خطای مجموعه آموزشی چقدر خواهد بود؟

د چرا هنگام اعمال این پیش‌بینی‌کننده k-NN برای داده‌های جدید، خطای داده اعتبار در مقایسه با نرخ خطا بسیار خوش‌بینانه است؟

ه. اگر هدف پیش‌بینی MEDV برای چندین هزار قطعه جدید باشد، ضرر استفاده از پیش‌بینی k-NN چیست؟ عملیاتی را که الگوریتم برای تولید هر پیش بینی انجام می دهد لیست کنید.

داده‌ها را فراخوانی میکنیم

```
> housing.df<-read.csv("~/BostonHousing.csv",header = T)
> housing.df<-housing.df[,-14]
> str(housing.df)
'data.frame':    506 obs. of  13 variables:
 $ CRIM    : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ ZN      : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
```

```
 $ INDUS  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ CHAS   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NOX    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ RM     : num  6.58 6.42 7.18 7 7.15 ...
 $ AGE    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ DIS    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ RAD    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ TAX    : int  296 242 242 222 222 222 311 311 311 311 ...
 $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ MEDV   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

ما در اینجا برای ۵۰۶ مشاهده ۱۳ متغیر محاسبه کرده‌یم.

بنظر بعضی از متغیرها باید به **factor** تبدیل شوند مثل **RAD , CHAS** .

طبق سوال ما داده‌ها را به صورت زیر باید تقسیم کنیم :

Train = 60% Valid=40%

```
> # as.factor
> housing.df$CHAS<-as.factor(housing.df$CHAS)
> housing.df$RAD<-as.factor(housing.df$RAD)
> # 60/40 split for training and validation
> library(caret)
> trainIndex <- createDataPartition(housing.df$MEDV, p=.6,list = FALSE, times = 1)
> train.df <- housing.df[trainIndex,]
> valid.df <- housing.df[-trainIndex,]
```

نرمال سازی داده‌ها

```
> # use preProcess() from the caret package to normalize Data.
> norm.values <- preProcess(train.df, method=c("center", "scale"))
> train.norm.df <- as.data.frame(predict(norm.values, train.df))
> valid.norm.df <- as.data.frame(predict(norm.values, valid.df))
> housing.norm.df <- as.data.frame(predict(norm.values, housing.df))
```

جواب الف

```
> train.knn.predictors<-train.norm.df[,-13]
> train.knn.target<-train.df[,13]
>
> valid.knn.predictors<-valid.norm.df[,-13]
> valid.knn.target<-valid.df[,13]
>
```

```
>
> #initialize a data frame with two columns: k, and accuracy
> accuracy.df <- data.frame(k = seq(1, 5, 1), RMSE = rep(0, 5))
>
> # compute knn for different k on validation.
> for(i in 1:5){
+   knn.pred<-class::knn(train = train.knn.predictors,
+                        test = valid.knn.predictors,
+                        cl =train.knn.target, k = i)
+   accuracy.df[i,2]<-RMSE(as.numeric(as.character(knn.pred)),valid.knn.target)
+ }
>
> accuracy.df
  k     RMSE
1 1 3.781706
2 2 4.560000
3 3 5.398217
4 4 5.344909
5 5 5.122177
```

مشخص است که **k=1** دارای کمترین میزان **RMSE** است یعنی بالاترین میزان دقت در بین **K=1:5** دارد ولی ممکن است استفاده از **k=1** مسئله **overfit** اتفاق بیوفتد و مشکل ساز شود به همین خاطر از **k=2** استفاده میکنیم که بعد از **k=1** بالاترین دقت را دارد.

جواب ب

```
> #create new dataframe with table values
> new.df<-data.frame(CRIM=0.2, ZN=0, INDUS=7 , CHAS=0, NOX=0.538, RM=6, AGE=62, DIS=4
.7, RAD=4, TAX=307, PTRATIO=21, LSTAT=10)
>
> #norm your new data
> new.norm.values <- preProcess(new.df, method=c("center", "scale"))
> new.norm.df <- predict(new.norm.values, newdata = new.df)
>
> #predict the MEDV
> new.knn.pred <- class::knn(train = train.knn.predictors,
+                            test = new.norm.df,
+                            cl = train.df$MEDV, k = 2)
> new.knn.pred
[1] 20.6
175 Levels: 7 7.2 7.4 8.1 8.4 8.5 8.7 9.5 9.6 9.7 10.2 10.8 10.9 11.3 11.5 11.7 11.8
12 12.1 12.3 12.5 12.7 13 13.1 13.2 13.3 13.4 13.5 ... 50
```

یعنی برای مشاهده‌ای با متغیرهای داده شده طبق دو مشاهده (رکورد) نزدیک **MEDV=21.6** پیش بینی میشود .

جواب ج

```
> new.accuracy.df<-RMSE(as.numeric(as.character(new.knn.pred)),valid.df[,13])
> new.accuracy.df
[1] 9.395313
```

جواب د

از آنجایی که داده‌های Valid ما بخشی از همان مجموعه‌ای بود که داده‌های Train ما از آن استخراج شده‌اند، یعنی داده‌های Train , Valid ما از یک مجموعه هستند که این باعث میشه ما مدلمون رو برای یک مجموعه داده خاص آموزش بدیم که این خیلی خوشبینانه است چون مدل ما برای داده ای خارج از اون مجموعه ممکنه به مشکل بخوره و دقتش پایین بیاید .

جواب ه

پیش بینی برای هزاران قطعه جدید زمان بسیار زیادی نیاز دارد چون باید برای هر قطعه که میخواهیم پیش بینی کنیم فاصله قطعه جدید را با تمام قطعه‌های موجود در نمونه محاسبه کند سپس برای پیش بینی MEDV باید از تمام همسایه‌های نزدیک میانگین بگیره

الگوریتم برای تولید هر پیش‌بینی باید عملیات زیر را انجام دهد:

محاسبه فاصله مشاهده جدید از تمام مشاهدات موجود در مجموعه داده.

مرتب کردن فاصله‌ها بصورت صعودی (یعنی اول مشاهده‌ای که نزدیکتر از همه است و...)

یک اعتبارسنجی بر اساس RMSE برای بدست آوردن بهترین عدد k .

محاسبه میانگین وزنی متغیر بر اساس K همسایه نزدیک .