

گزارش پروژه دوم – مصور سازي و بررسي مدل هاي پيشگو روي داده هاي مربوط به خريد و فروش مسكن

استان خوزستان

محمد جواد ولی پور

در اين پروژه قصد داريم داده هاي مربوط به خريد و فروش مسكن در استان خوزستان را مصورسازي كنيم.

اطلاعات مربوط به كل استان ها از ۴۳۱۳۳ داده و ۱۵ متغير تشكيل شده است. كه ۷۶۸ عدد از اين داده ها اطلاعات مربوط به استان خوزستان است.

ابتدا به طور مختصر به معرفي ستون هاي داده ميپردازيم.

۱. كد قرارداد : نمونه اي از داده هاي اين ستون به شكل زير است

كد قرارداد \$: chr [1:43133] "19136381" "19140513" "19192568" "19202902" ...

اين ستون اطلاعات خاصي در اختيار ما قرار نميدهد.

۲. نوع قرارداد : نمونه اي از داده هاي اين ستون به شكل زير است

نوع قرارداد \$: chr [1:43133] "مبايعه نامه" "مبايعه نامه" "مبايعه نامه" ...

مقادير تمام داده هاي مربوط به اين ستون برابر با "مبايعه نامه" است و اطلاعات خاصي در اختيار ما قرار نميدهد.

۳. استان : نمونه اي از داده هاي اين ستون به شكل زير است

استان \$: chr [1:43133] "زنجان" "زنجان" "زنجان" ...

به داده هايي نياز داريم كه مقادير آن ها در اين ستون برابر با "خوزستان" باشد.

۴. شهرستان : نمونه اي از داده هاي اين ستون به شكل زير است

شهرستان \$: chr [1:43133] "ابهر" "ابهر" "ابهر" ...

۵. نوع ملك : نمونه اي از داده هاي اين ستون به شكل زير است

نوع ملك \$: chr [1:43133] "دستگاه آبارتمان" "دستگاه آبارتمان" "دستگاه آبارتمان" ...

مقادير تمام داده هاي مربوط به اين ستون برابر با "دستگاه آبارتمان" است و اطلاعات خاصي در اختيار ما قرار نميدهد.

۶. منطقه شهرداري : نمونه اي از داده هاي اين ستون به شكل زير است

منطقه شهرداري \$: num [1:43133] NA NA NA NA NA NA NA NA NA 4 ...

اطلاعات اين ستون فقط براي برخي شهرستان ها در برخي استان ها مقدار گرفته است، به همين دليل نميتوانيم از اين ستون نيز استفاده كنيم

۷. نوع کاربری : نمونه ای از داده های این ستون به شکل زیر است

نوع کاربری \$: chr [1:43133] "مسکونی" "مسکونی" "مسکونی" "مسکونی" ...

مقادیر تمام داده های مربوط به این ستون برابر با "مسکونی" است و اطلاعات خاصی در اختیار ما قرار نمیدهد.

۸. مساحت : مساحت ساختمان به متر مربع . نمونه ای از داده های این ستون به شکل زیر است

مساحت \$: num [1:43133] 83 90 49 80.9 80 ...

۹. درصد : درصد فروش رفته از ساختمان در معامله . نمونه ای از داده های این ستون به شکل زیر است

درصد \$: chr [1:43133] "100" "100" "100" "100" ...

۱۰. قیمت : قیمت کل ساختمان به هزار ریال. نمونه ای از داده های این ستون به شکل زیر است

قیمت \$: num [1:43133] 3000000 1080000 10000000 3240000 750000 ...

۱۱. قیمت یک متر مربع : نمونه ای از داده های این ستون به شکل زیر است

قیمت یک مترمربع \$: chr [1:43133] "36144.58" "12000.00" "204081.63" "40039.55" ...

۱۲. عمر بنا : نمونه ای از داده های این ستون به شکل زیر است

عمر بنا \$: num [1:43133] 15 9 10 9 10 0 1 6 19 0 ...

۱۳. نوع اسکلت : نمونه ای از داده های این ستون به شکل زیر است

نوع اسکلت \$: chr [1:43133] "فلزی" "فلزی" "بتونی" "فلزی" ...

۱۴. تاریخ ثبت قرارداد : نمونه ای از داده های این ستون به شکل زیر است

تاریخ ثبت قرارداد \$: chr [1:43133] "1399/04/14" "1399/04/15" "1399/04/23" "1399/04/24" ...

اطلاعات خاصی در اختیار ما قرار نمیدهد چون این اطلاعات مربوط به یک بازه یک ماهه میباشد و در این بازه کوتاه نمیتوان تاثیر زمان بر تغییرات مختلف در داده ها را به درستی بررسی کرد و نتیجه گیری های مطمئن از آن ها استخراج کرد.

۱۵. شش رقم نخست کد پستی : نمونه ای از داده های این ستون به شکل زیر است

شش رقم نخست کد پستی \$: chr [1:43133] "456179" "456615" "456173" "456194" ...

همان طور که گفته شد تعداد زیادی از این ستون ها اطلاعات خاصی به ما نمیدهند و باید قبل از کار با داده ها این اطلاعات کنار بگذاریم.

ستون هایی که در ادامه کار به آن ها نیاز داریم عبارتند از:

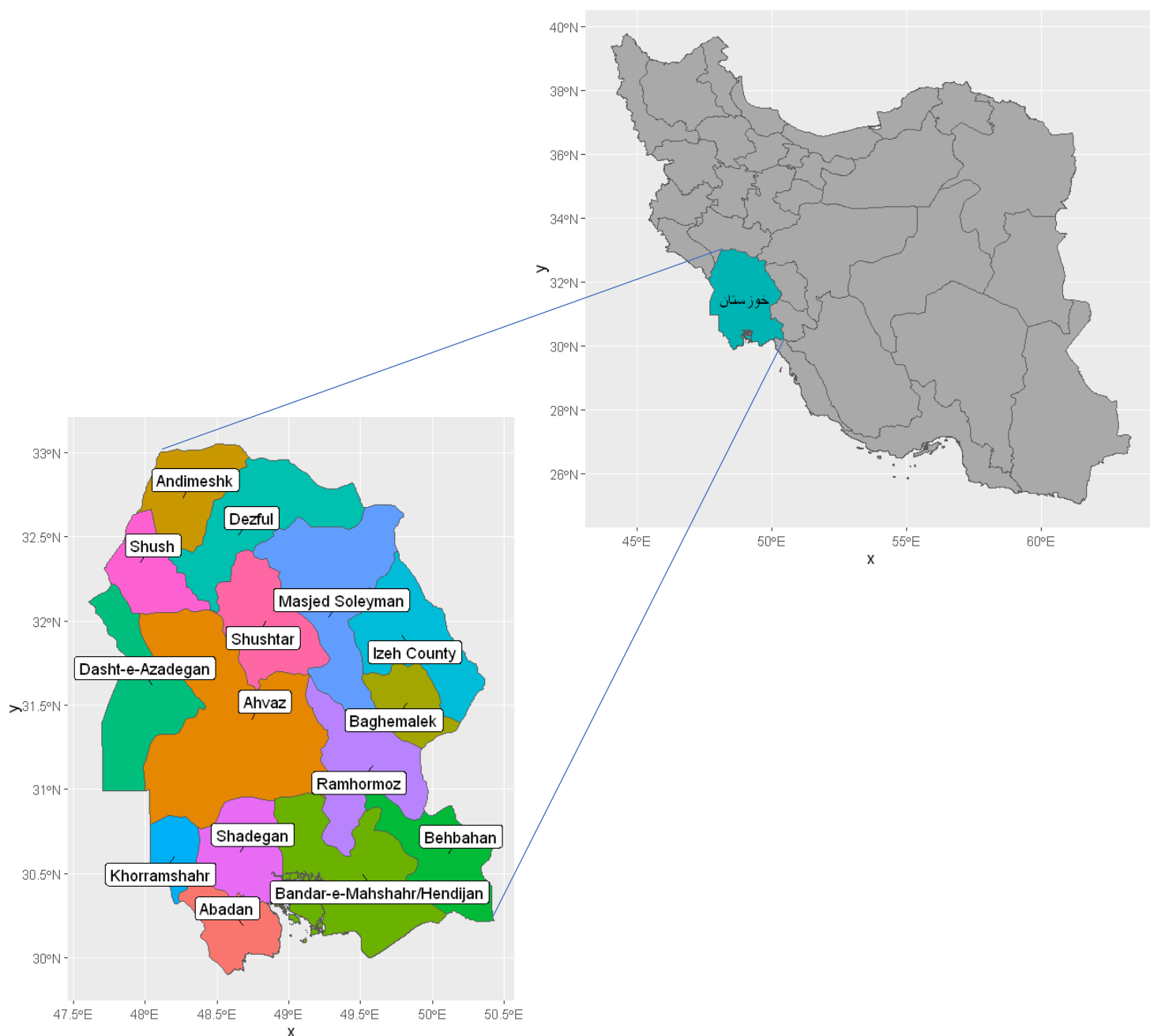
استان – شهر – مساحت – قیمت – قیمت هر متر مربع – عمر بنا – نوع اسکلت – شش رقم آخر کد پستی

که برای سادگی کار با داده ها نام این ستون ها را به ترتیب به شکل زیر در می آوریم.

State – city – area – price – price_per_square – building_age – skeleton_type – postal_code

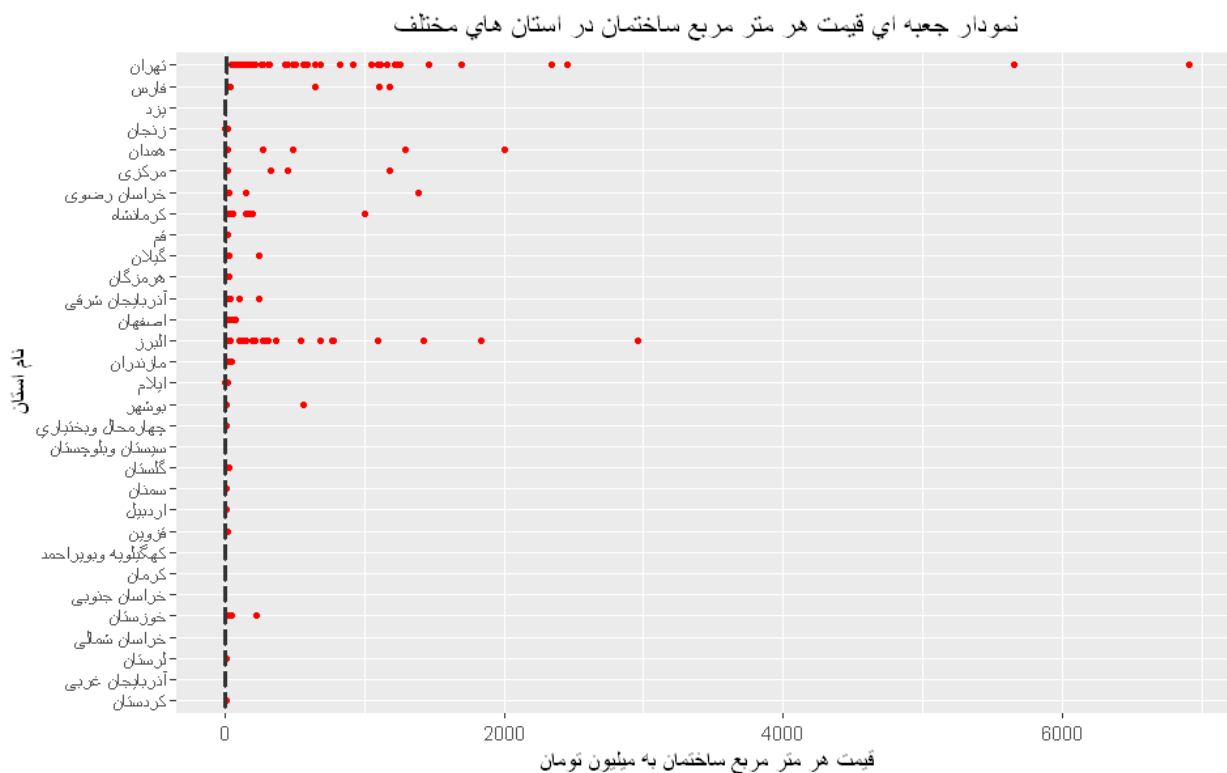
همچنین برای خوانا تر شدن اعداد موجود برای قیمت و قیمت هر متر مربع ، این اطلاعات را به شکل واحد میلیون تبدیل میکنیم.

برای شروع کار ابتدا در نقشه ایران نگاهی به موقعیت مکانی استان خوزستان و شهرستان های این استان می اندازیم.



برای بررسی وضعیت قیمت در استان خوزستان نسبت به دیگر استان ها ابتدا نمودار جعبه ای قیمت ساختمان در استان های مختلف را رسم میکنیم.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(data, aes(x=price_per_square, y=reorder(state, price_per_square, FUN=median) , fill=state)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16,outlier.size=1) +
  labs(title="نمودار جعبه ای قیمت هر متر مربع ساختمان در استان های مختلف", x="قیمت هر متر مربع ساختمان به میلیون تومان", y = "نام استان") +
  theme(legend.position="none",plot.title = element_text(hjust = 0.5))
```

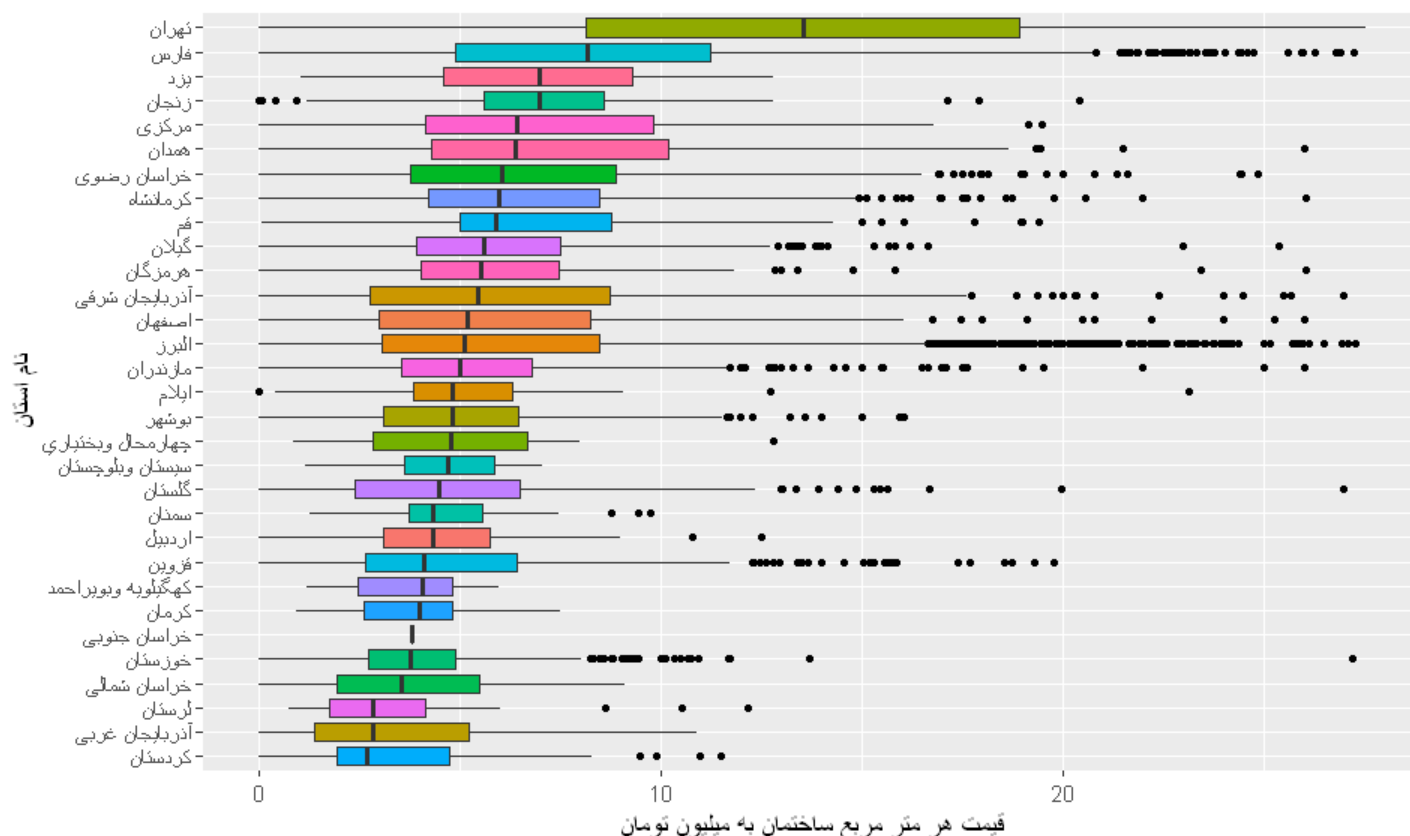


همان طور که مشاهده میکنید به دلیل وجود داده های پرت نمیتوانیم درک درستی از داده های به دست آوریم، به همین دلیل میتوانیم موقتا برخی داده های پرت را کنار بگذاریم تا بتوانیم بخش اصلی داده ها را دقیق تر بررسی کنیم، به همین منظور این نمودار را دوباره رسم میکنیم با این تفاوت که به جای رسم اطلاعات مربوط به تمام داده ها، اطلاعات داده هایی را رسم میکنیم که قیمت ساختمان در آن ها در ۹ دهک اول قیمتی وجود داشته باشد، به این ترتیب داده های بسیار بزرگی که وجود دارد را کنار میگذاریم.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(data[data$price_per_square < quantile(data$price_per_square,0.9),],
  aes(x=price_per_square, y=reorder(state, price_per_square, FUN=median) , fill=state)) +
  geom_boxplot(outlier.colour="black", outlier.shape=16,outlier.size=1) +
  labs(title="نمودار جعبه ای قیمت هر متر مربع (9 دهک اول) ساختمان در استان های مختلف", x="قیمت هر متر مربع ساختمان به میلیون تومان", y = "نام استان") +
  theme(legend.position="none",plot.title = element_text(hjust = 0.5))
```

Quantile دستوری است که برای این کار استفاده میکنیم تا مرز بین دهک نهم و دهم را پیدا کنیم و داده های را از آن جا به دو بخش تقسیم کنیم. از دستور reorder نیز استفاده میکنیم تا این نمودار های جعبه ای را به ترتیب میانه در نمودار بچینیم تا دید بهتری به ما دهد.

نمودار جعبه‌ای قیمت هر متر مربع (9 دهک اول) ساختمان‌های مختلف



همان طور که مشاهده میکنید با این کار توانستیم درک بهتری نسبت به داده‌ها پیدا کنیم و همان طور که از نمودار واضح است با توجه به میانه قیمت‌ها در استان‌های مختلف، استان خوزستان در رده پنجم استان‌های ارزان کشور است.

برای ادامه کار قصد داریم داده‌های مربوط به استان خوزستان را دقیق‌تر بررسی کنیم بنابراین داده‌هایی که مقدار آن‌ها در ستون state برابر با "خوزستان" است را جدا میکنیم.

```
khoozestan = data[data$state == 'خوزستان',]
```

در ابتدا قصد داریم همبستگی بین ستون‌ها را بررسی کنیم، برای این کار نیاز داریم تا تمام داده‌های کیفی را به کمک dummy variable‌ها به شکل داده‌های کمی دربیاوریم تا بتوانیم مقدار correlation را برای آن‌ها به دست بیاوریم. داده‌های کیفی این dataframe در ستون‌های city و building_type میباشد. برای این کار از دستور زیر استفاده میکنیم.

```
df<-khoozestan %>%
  select('city','area','building_age','skeleton_type','price','price_per_square')

df = dummy_cols(df, select_columns = c('city', 'skeleton_type'),remove_selected_columns = TRUE)
```

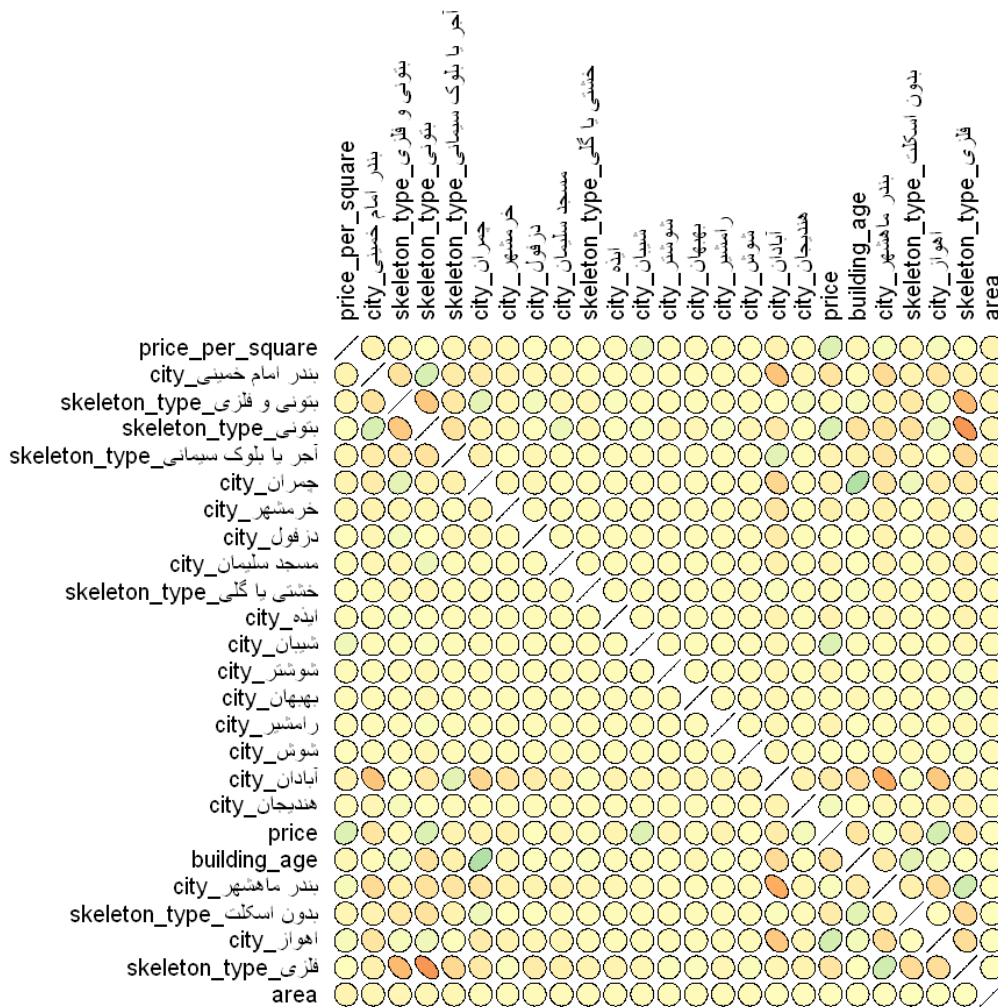
سپس با دستور `data <- cor(df)` ماتریس correlation را برای این داده‌ها به دست می‌آوریم.

برای درک بهتر این ماتریس میتوانیم از نمودار زیر استفاده کنیم.

```
options(repr.plot.width = 8, repr.plot.height = 8)
data <- cor(df)

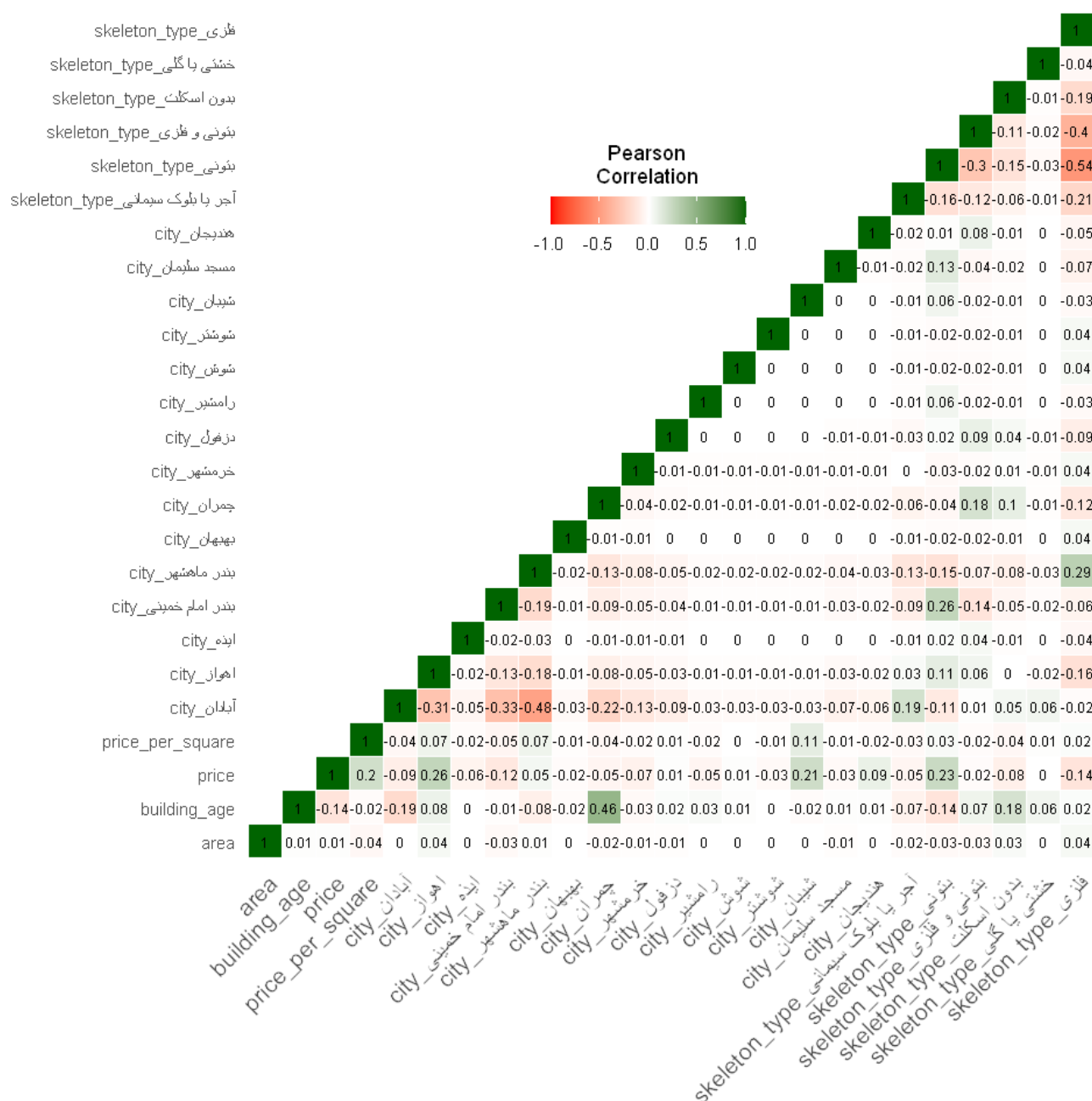
# Build a Pannel of 100 colors with Rcolor Brewer
my_colors <- brewer.pal(5, "Spectral")
my_colors <- colorRampPalette(my_colors)(100)

# Order the correlation matrix
ord <- order(data[1, ])
data_ord <- data[ord, ord]
plotcorr(data_ord, col=my_colors[data_ord*50+50], mar=c(0,0,0,0))
```



در این نمودار هر چه رنگ بیضی مربوطه به سبز نزدیک تر باشد دو متغیر رابطه مستقیم قوی تری دارند و هر چه به رنگ قرمز نزدیک تر باشد دو متغیر رابطه معکوس قوی تری دارند و بخش هایی که رنگ آن به زرد نزدیک است یعنی دو متغیر هیچ رابطه ای با یکدیگر ندارند.

برای مشاهده دقیق تر اعداد شاخص همبستگی میتوانیم از نمودار زیر نیز استفاده کنیم.

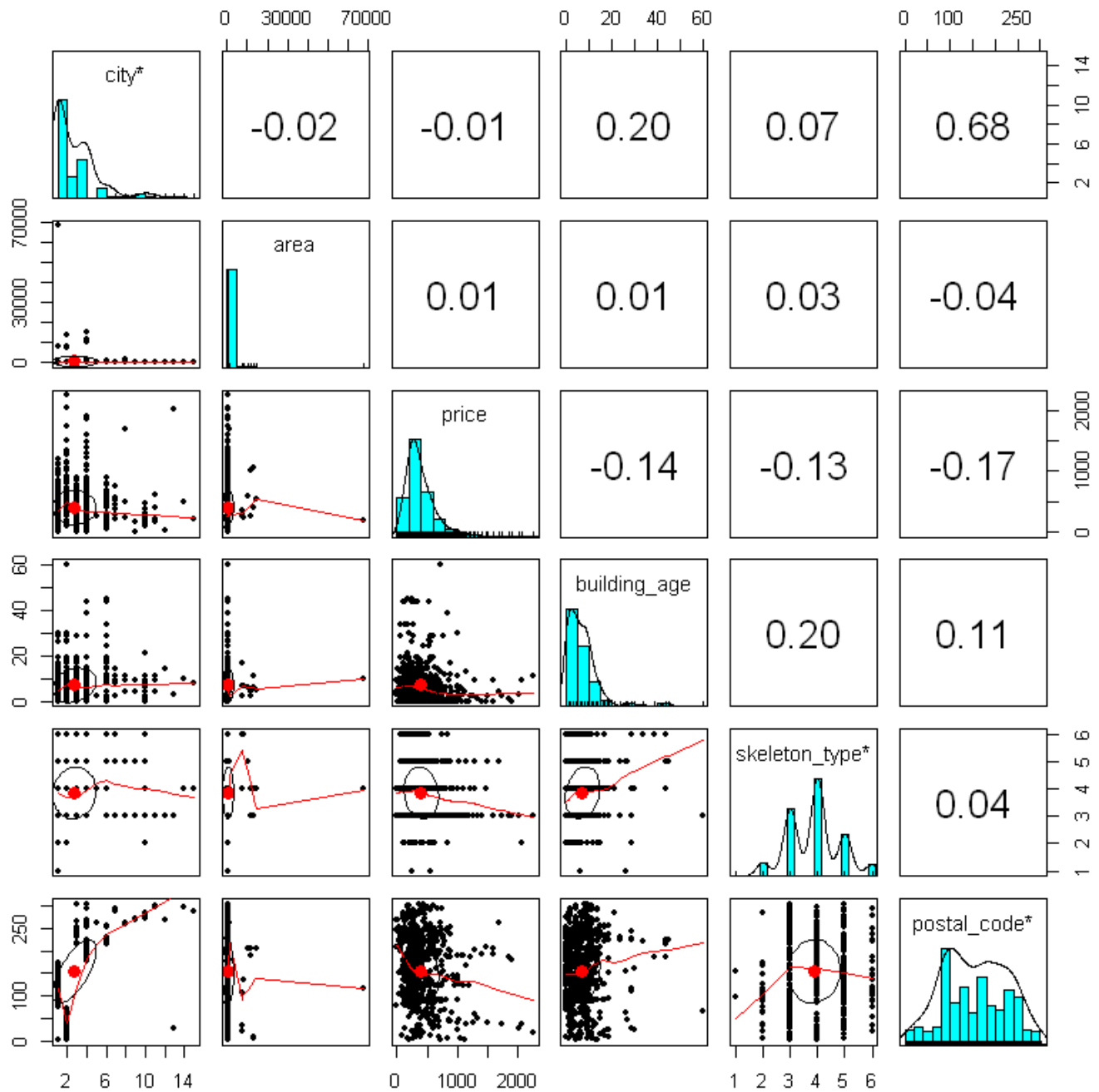


حالا نمودار جعبه ای قیمت ساختمان در شهرستان های مختلف را با دستور زیر رسم میکنیم.

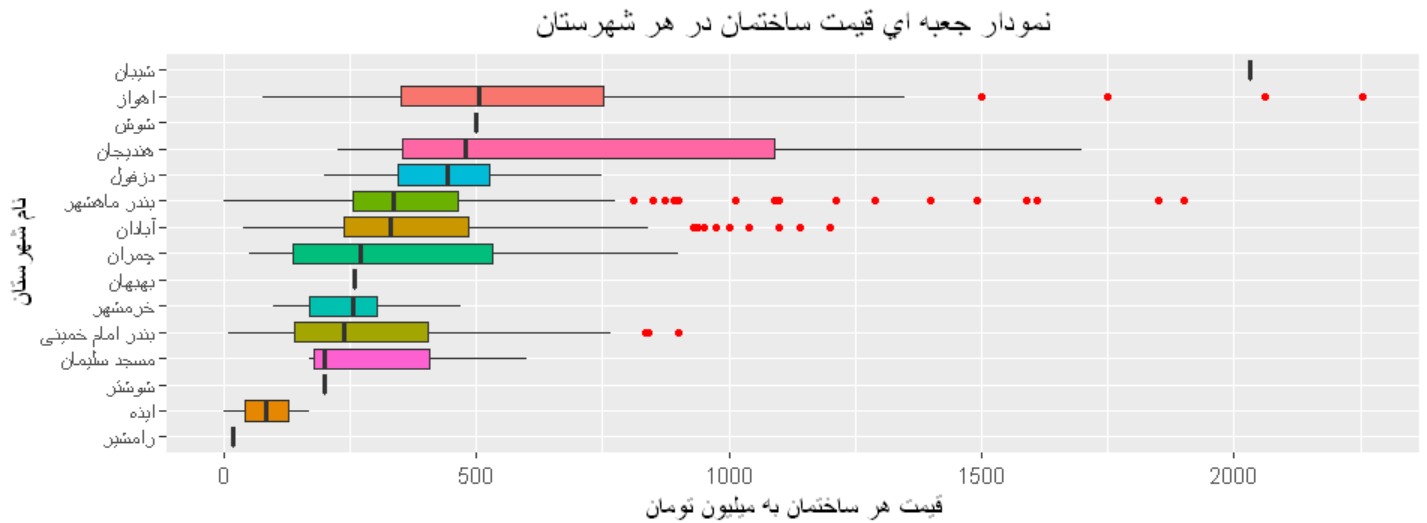
```
options(repr.plot.width = 8, repr.plot.height = 3)
ggplot(khoozestan, aes(x=price, y=reorder(city, price, FUN=median) , fill=city)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16,outlier.size=1) +
  labs(title="نمودار جعبه ای قیمت ساختمان در هر شهرستان", x="قیمت هر ساختمان به میلیون تومان", y = "نام شهرستان") +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))
```

همبستگی و نمودار پراکنش و نمودار توضیح داده های استان خوزستان قبل از تبدیل متغیر های رسته ای به dummy_variable به شکل زیر میباشد.

```
pairs.panels(khoozestan)
```

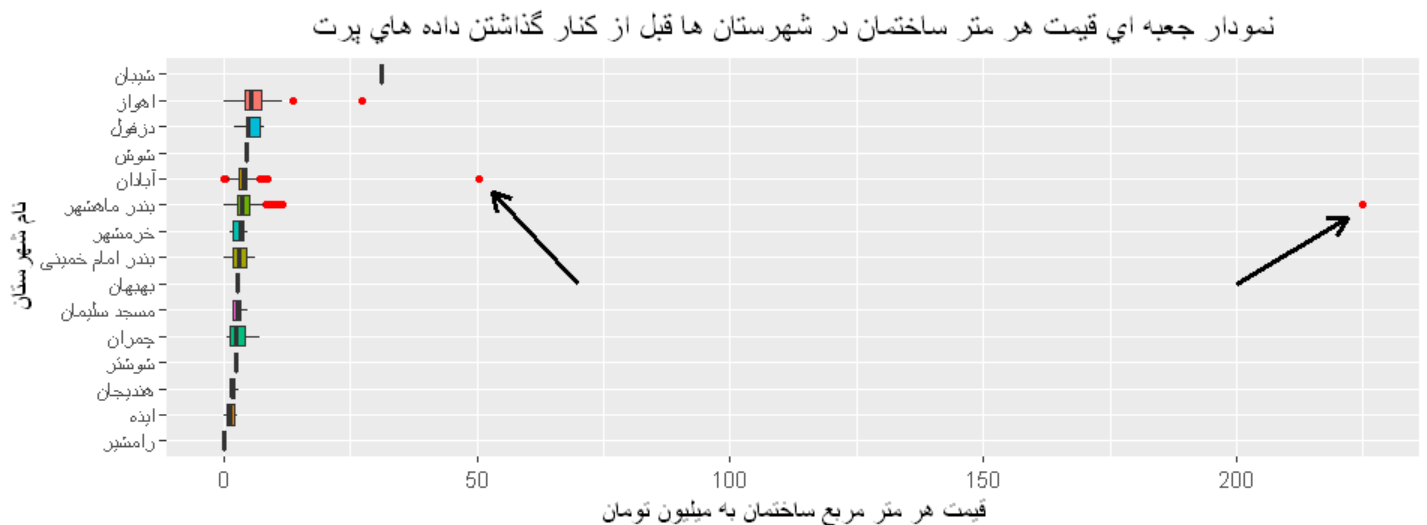


و شهرستان ها را به ترتیب میانه قیمت در نمودار نمایش میدهیم.



سپس همین کار را با قیمت هر متر مربع نیز انجام میدهیم.

```
options(repr.plot.width = 8, repr.plot.height = 3)
ggplot(khoozestan, aes(x=price_per_square, y=reorder(city, price_per_square, FUN=median) , fill=city)) +
geom_boxplot(outlier.colour="red", outlier.shape=16,outlier.size=1) +
labs(title="نمودار جعبه ای قیمت هر متر ساختمان در شهرستان ها قبل از کنار گذاشتن داده های پرت", x="قیمت هر متر مربع ساختمان به میلیون تومان", y = "نام شهرستان" ) +
geom_segment(aes(x = 200, y = 7, xend = 222, yend = 9.5),size = 1,arrow = arrow(length = unit(.3, "cm")))) +
geom_segment(aes(x = 70, y = 7, xend = 53, yend = 10.5),size = 1,arrow = arrow(length = unit(.3, "cm")))) +
theme(legend.position="none",plot.title = element_text(hjust = 0.5))
```



همان طور که مشاهده میکنید وجود این دو داده ی پرت باعث میشود نتوانیم به خوبی اطلاعات بقیه داده ها را مشاهده و بررسی کنیم به همین دلیل موقتا این دو داده را از اطلاعات کنار میگذاریم اما مشخصات مربوط به آن ها را حذف نمیکنیم تا بتوانیم در صورت نیاز به دقت این داده ها را بررسی کنیم، لازم به ذکر است که قصد ما از حذف کردن این داده ها غلط نشان دادن آن ها و یا بی اهمیت نشان دادنشان نیست و فقط برای درک بهتر بقیه داده ها این کار را انجام میدهیم.

کد قرار داد در این دو داده ۱۹۱۸۹۱۹۸ و ۱۹۲۳۶۳۰۳ بود.

مجددا این نمودار را رسم میکنیم.



به این شکل میتوانیم دقیق تر این داده ها را بررسی کنیم.

مشکلی که در این جا مواجه آن میشویم که در نمودار قبلی قطعاً نمیتوانستیم متوجه آن شویم این است که تعداد داده ها در بعضی از شهرستان ها بسیار کم است و این موضوع سطح اطمینان ما به این داده ها را کم میکند. شهرستان هایی مانند شیبان ، شوش ، شوشتر ، بهبهان و رامشیر.

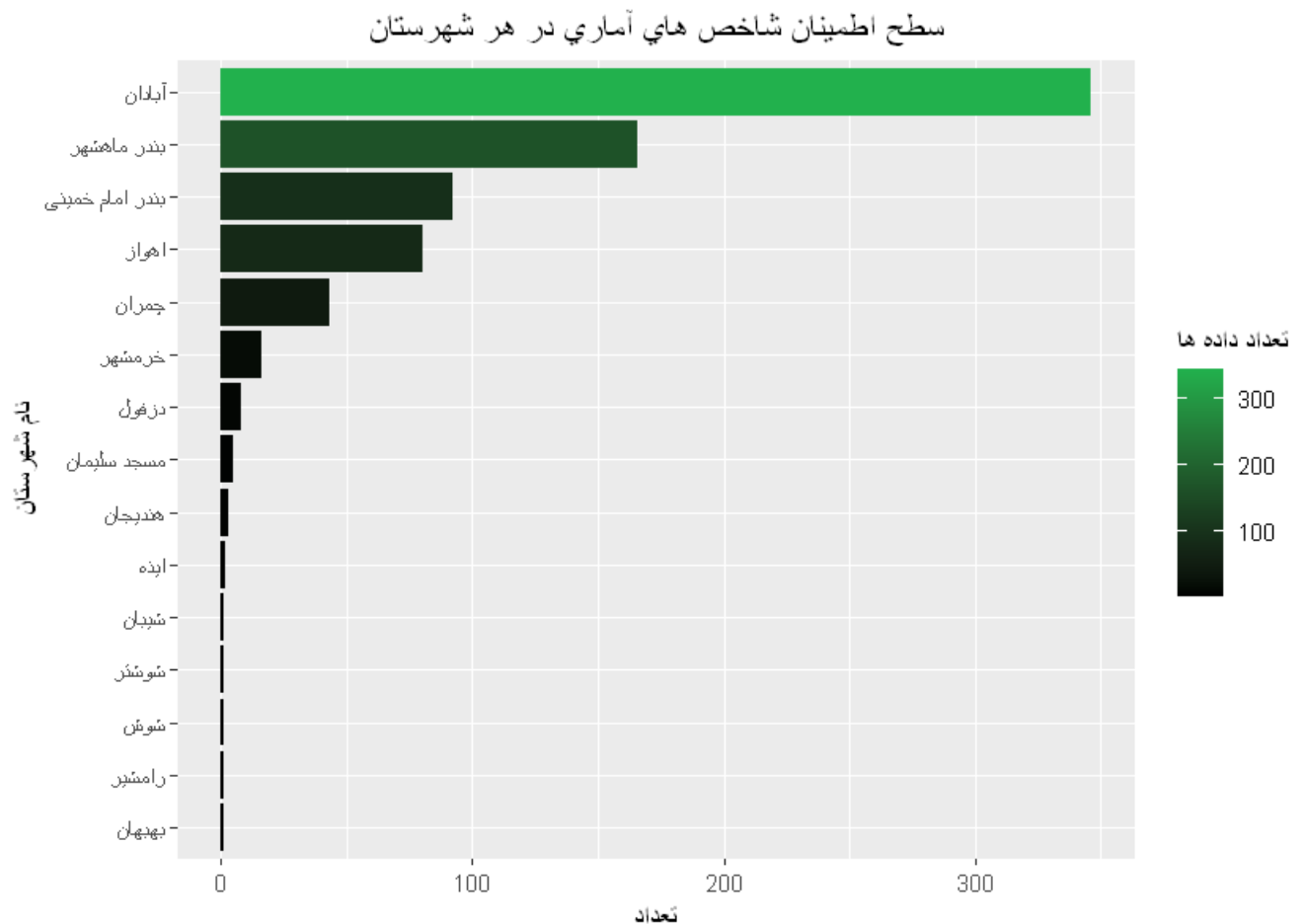
برای بررسی دقیق تر این موضوع ابتدا تعداد داده های مربوط به هر شهرستان را شمارش میکنیم.

```
confidence_level <- khoozestan %>%
  select(city) %>%
  group_by(city) %>%
  count() %>%
  arrange(n)
```

city	n
بهبهان	1
رامشیر	1
شوش	1
شوشتر	1
شیبان	1
ایذه	2
هندیجان	3
مسجد سلیمان	5
دزفول	8
خرمشهر	16
چمران	43
اهواز	80
بندر امام خمینی	92
بندر ماهشهر	166
آبادان	346

برای درک بهتر این موضوع این اعداد را در یک نمودار میله ای نمایش میدهیم. هر چه تعداد داده های مربوط به یک شهرستان بیشتر باشد، سطح اطمینان ما به شاخص های آماری مربوط به آن شهرستان نیز افزایش پیدا میکند.

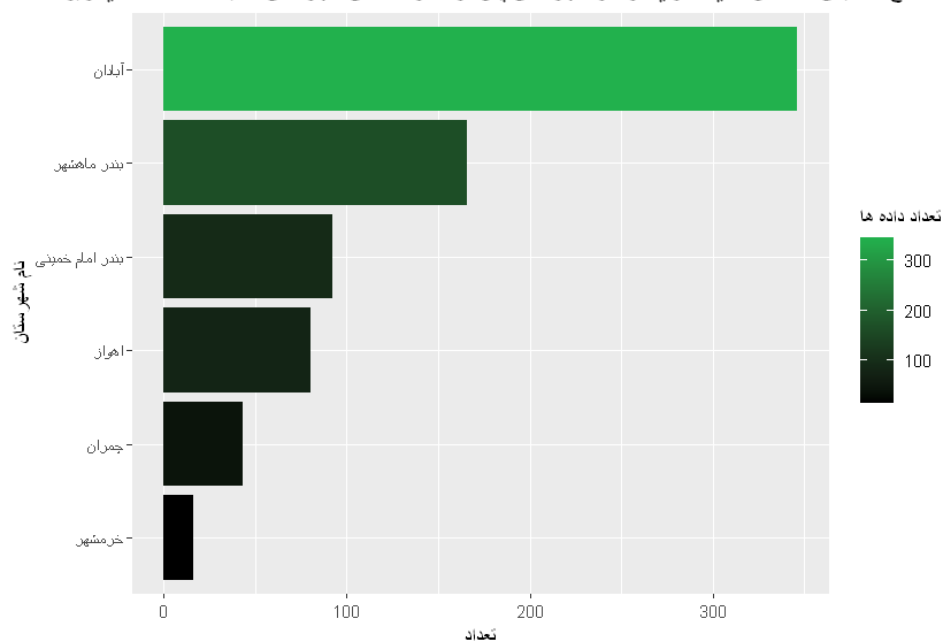
```
options(repr.plot.width = 7, repr.plot.height = 5)
ggplot(confidence_level , aes(x=n , y=reorder(city,n), fill = n)) +
  geom_bar(stat = "identity") +
  labs(title="سطح اطمینان شاخص های آماری در هر شهرستان", x="تعداد", y = "نام شهرستان") +
  scale_fill_gradient(name = "تعداد داده ها",low="black", high="#22b14d") +
  theme(plot.title = element_text(hjust = 0.5))
```



همان طور که مشاهده میکنید علاوه بر بهبان ، رامشیر ، شوش ، شوشتر و شیبلیان ، تعداد داده های شهرستان های ایذه ، هندیجان ، مسجد سلیمان و دزفول نیز زیر ۱۰ میباشد و به همین دلیل سطح اطمینان شاخص های آماری در این داده ها بسیار پایین است و به همین دلیل داده های مربوط به این شهرستان ها را کنار میگذاریم تا بتوانیم بقیه داده ها را دقیق تر بررسی کنیم.

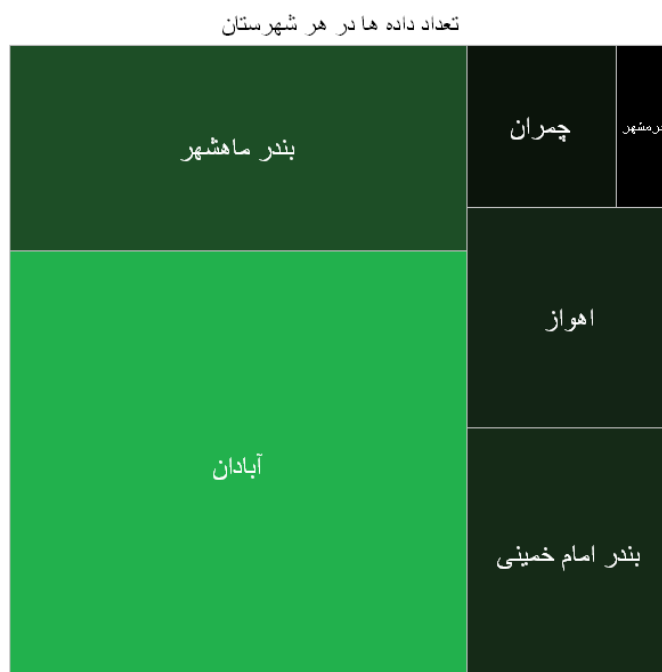
مجددا بر این نکته تاکید میکنم که تمام این داده ها برای ما مهم هستند و باید اطلاعات هر یک مورد بررسی قرار گیرد.

سطح اطمینان شاخص های آماری در هر شهرستان پس از کنار گذاشتن شهرستان ها با تعداد داده های زیر 10



همچنین برای این کار می‌توانیم از نمودار Treemap استفاده کنیم.

```
options(repr.plot.width = 5, repr.plot.height = 5)
ggplot(confidence_level, aes(area = n, fill = n, label = city)) +
  geom_treemap() +
  geom_treemap_text(colour = "white", size = 15, place = "centre", grow = FALSE) +
  labs(title = "تعداد داده ها در هر شهرستان") +
  scale_fill_gradient(name = "تعداد داده ها", low = "black", high = "#22b14d") +
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5))
```



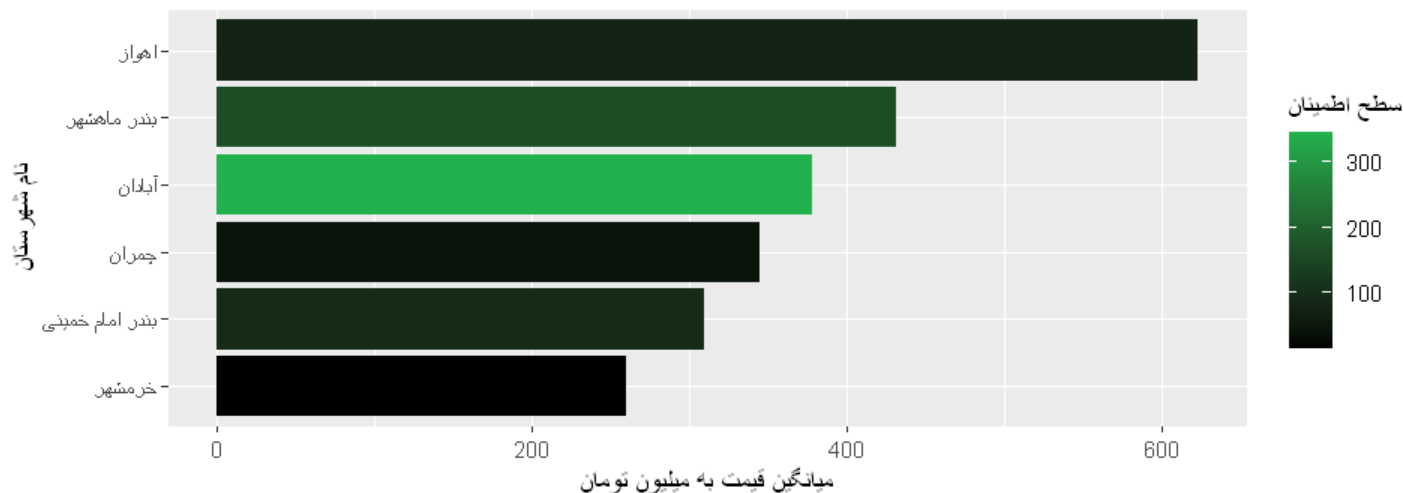
city	mean_price
اهواز	623.4250
آبادان	378.0111
بندر امام خمینی	309.6426
بندر ماهشهر	431.9117
چمران	344.5581
خرمشهر	259.9812

برای مقایسه میانگین قیمت بین این چند شهرستان از نمودار میله ای استفاده میکنیم.

```
df <- khoozestan %>%
  select(city,price) %>%
  group_by(city) %>%
  summarise(mean_price = mean(price))
df
```

```
options(repr.plot.width = 8, repr.plot.height = 3.1)
confidence = confidence_level[match(reorder(df$city , df$mean_price), confidence_level$city),]$n
ggplot(df , aes(x=mean_price , y=reorder(city,mean_price),fill = confidence)) +
  geom_bar(stat = "identity") +
  labs(title="میانگین قیمت هر شهرستان", x="میانگین قیمت به میلیون تومان", y = "نام شهرستان") +
  scale_fill_gradient(name="سطح اطمینان",low="black", high="#22b14d") +
  theme(plot.title = element_text(hjust = 0.5))
```

میانگین قیمت هر شهرستان

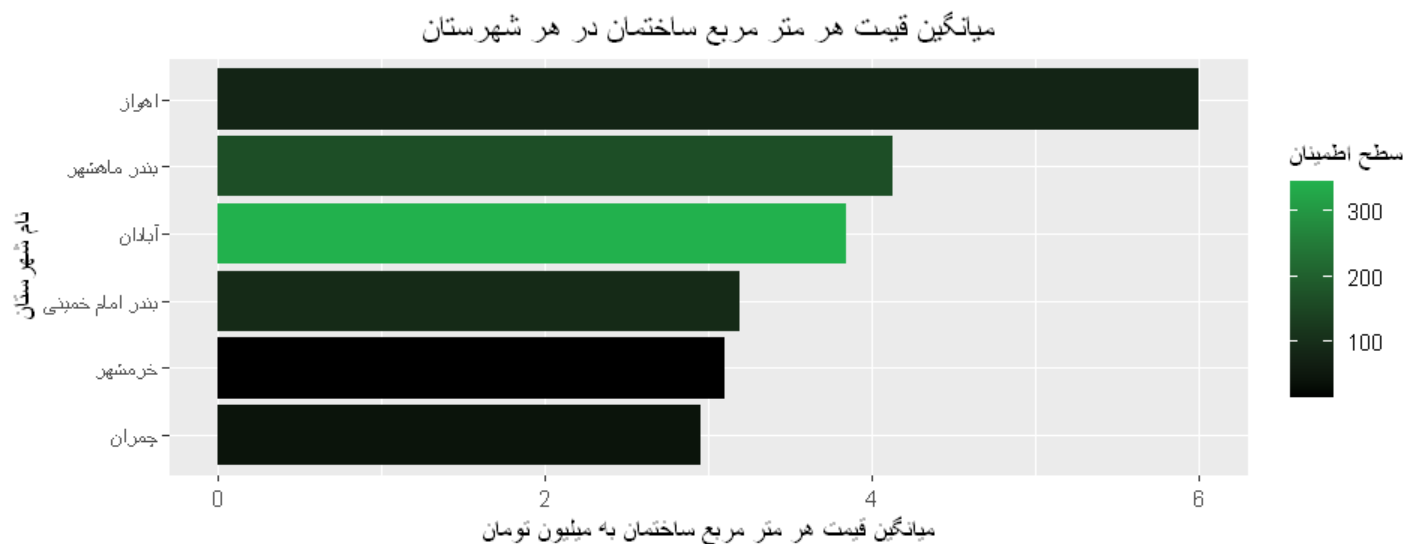


city	mean_price_per_square
اهواز	6.002169
آبادان	3.839624
بندر امام خمینی	3.195575
بندر ماهشهر	4.127006
چمران	2.949167
خرمشهر	3.095103

همین نمودار را برای اطلاعات مربوط به قیمت هر متر مربع نیز رسم میکنیم.

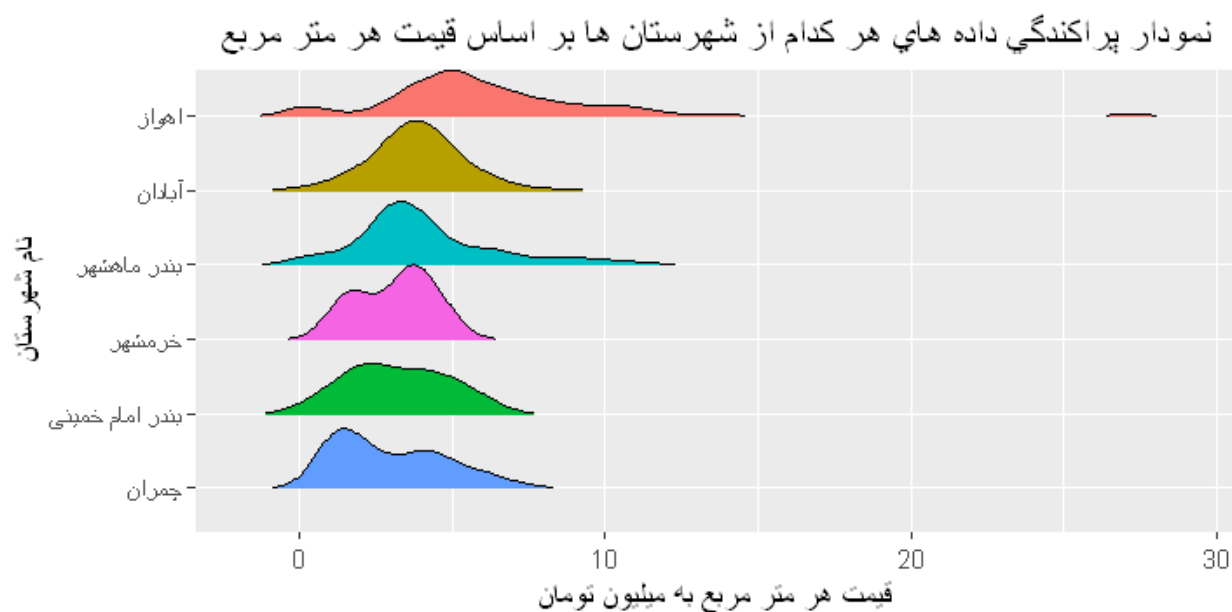
```
df <- khoozestan %>%
  select(city,price_per_square) %>%
  group_by(city) %>%
  summarise(mean_price_per_square = mean(price_per_square))
df
```

```
options(repr.plot.width = 8, repr.plot.height = 3.1)
confidence = confidence_level[match(reorder(df$city , df$mean_price_per_square), confidence_level$city),]$n
ggplot(df , aes(x=mean_price_per_square , y=reorder(city,mean_price_per_square),fill = confidence)) +
  geom_bar(stat = "identity") +
  labs(title="میانگین قیمت هر متر مربع ساختمان در هر شهرستان", x="میانگین قیمت به میلیون تومان", y = "نام شهرستان") +
  scale_fill_gradient(name="سطح اطمینان",low="black", high="#22b14d") +
  theme(plot.title = element_text(hjust = 0.5))
```



پراکندگی داده ها در هر یک شهرستان ها بر اساس قیمت هر متر مربع ساختمان به شکل زیر است.

```
options(repr.plot.width = 6, repr.plot.height = 3)
ggplot(khoozestan, aes(x = price_per_square, y = reorder(city, price_per_square, FUN=median), fill = city)) +
  geom_density_ridges_gradient(scale = 1, rel_min_height = 0.01) +
  labs(title="نمودار پراکندگی داده های هر کدام از شهرستان ها بر اساس قیمت هر متر مربع", x="قیمت هر متر مربع به میلیون تومان", y = "نام شهرستان") +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))
```



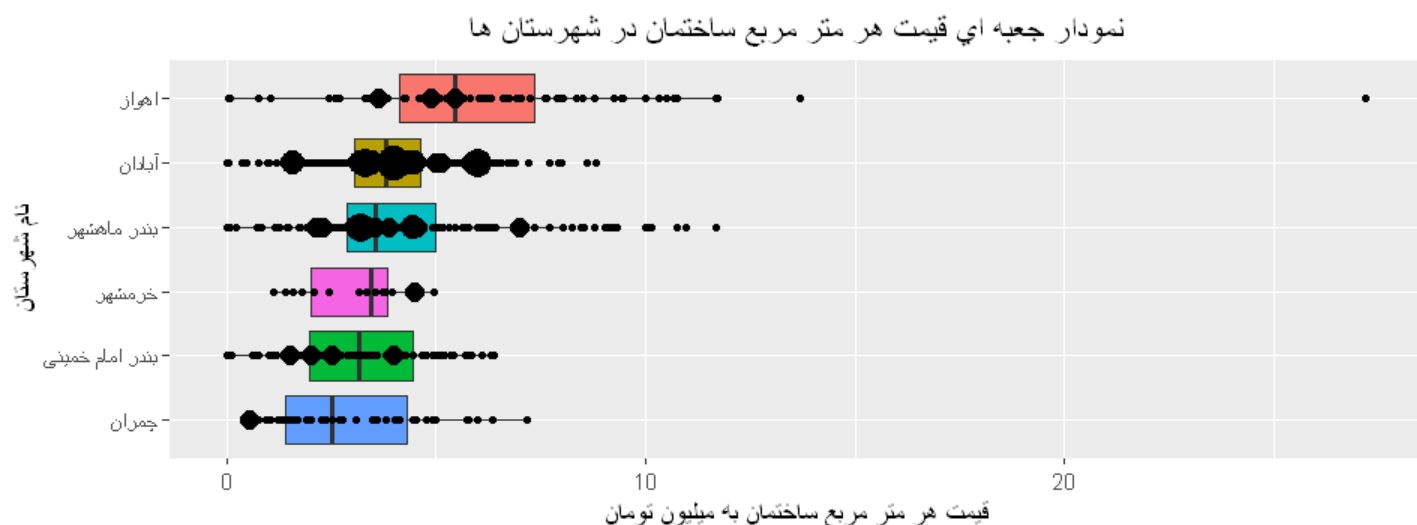
میتوان تعداد داده های موجود در هر قیمت را به کمک دسَنور geom_count روی نمودار جعبه ای نشان داد که در نمودار زیر میتوانید آن را مشاهده کنید.

```
options(repr.plot.width = 8, repr.plot.height = 3)
ggplot(khoozestan, aes(x=price, y=reorder(city, price, FUN=median), fill=city)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16, outlier.size=1) +
  geom_count(col="black", show.legend=F)+
  labs(title="نمودار جعبه ای قیمت ساختمان در هر شهرستان", x="قیمت هر ساختمان به میلیون تومان", y = "نام شهرستان") +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))
```



نمونه ی دیگری از همین نمودار برای قیمت هر متر مربع را میتوانیم به شکل زیر رسم کنیم.

```
options(repr.plot.width = 8, repr.plot.height = 3)
ggplot(khoozestan, aes(x=price_per_square, y=reorder(city, price_per_square, FUN=median) , fill=city)) +
  geom_boxplot(outlier.colour="red", outlier.shape=16, outlier.size=1) +
  geom_count(col="black", show.legend=F)+
  labs(title="نمودار جعبه ای قیمت هر متر مربع ساختمان در شهرستان ها", x="قیمت هر متر مربع ساختمان به میلیون تومان", y = "نام شهرستان") +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))
```

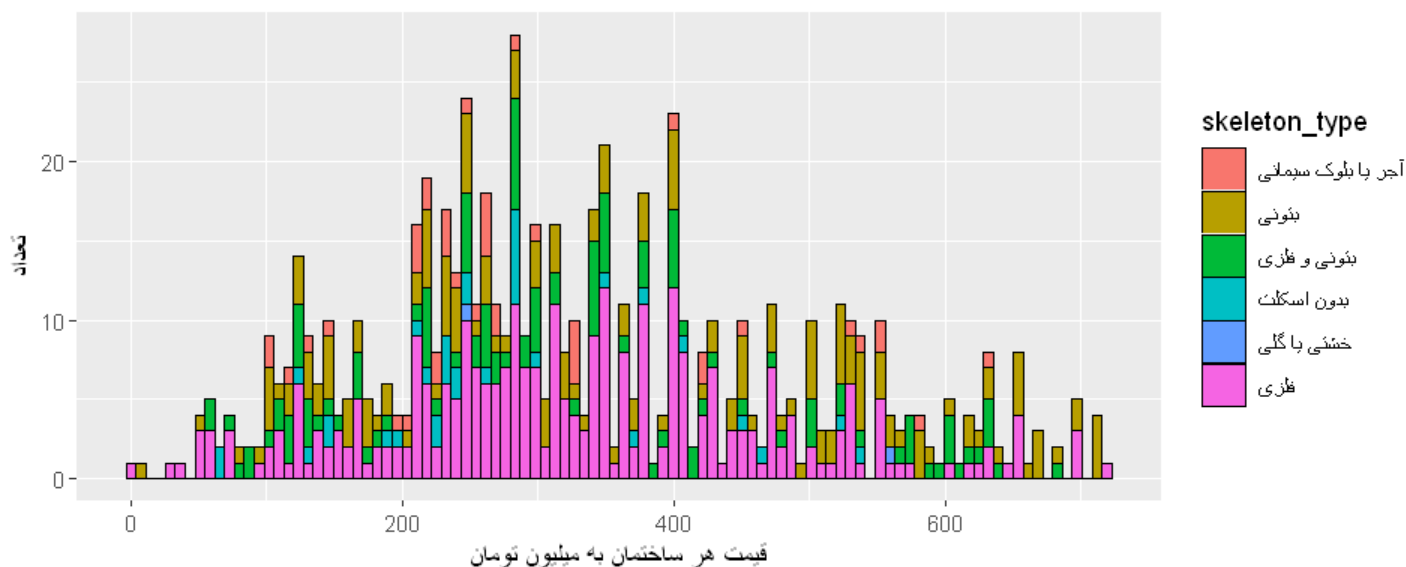


حالا قصد داریم تاثیر اسکلت ساختمان بر قیمت بنا را بررسی کنیم.

نمودار اولی که برای این منظور رسم میکنیم که نمودار پراکندگی قیمت ساختمان ها بر اساس قیمت آن ها میباشد.

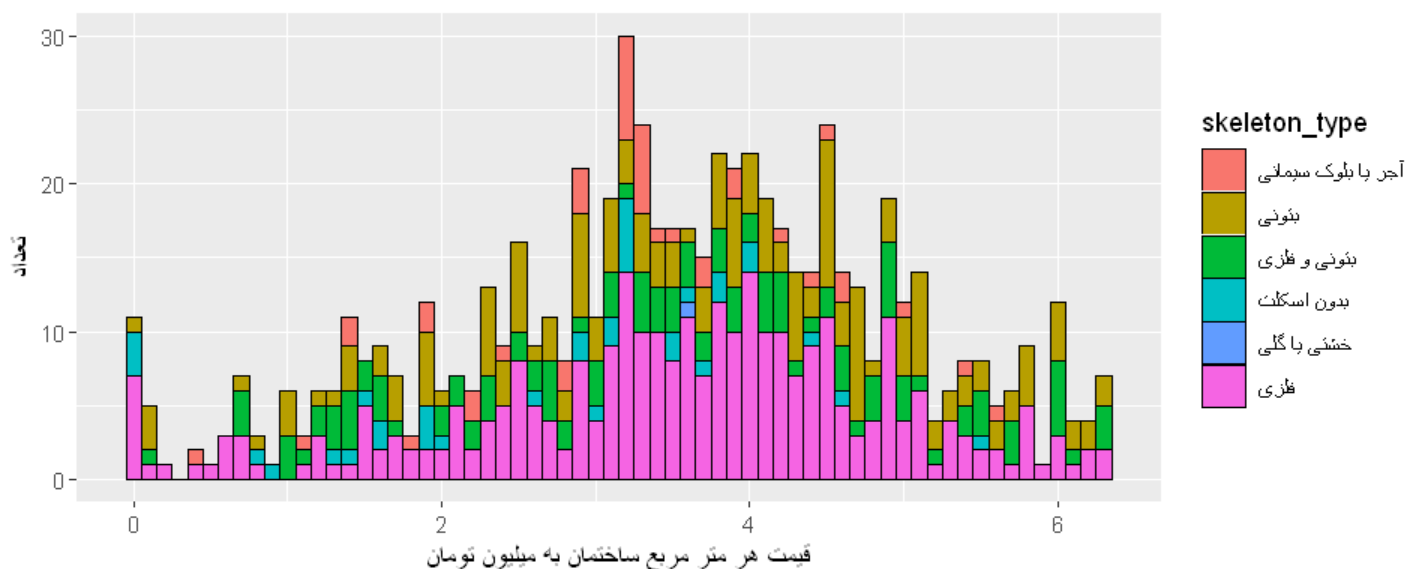
```
options(repr.plot.width = 8, repr.plot.height = 3.5)
ggplot(khoozestan[khoozestan$price < quantile(khoozestan$price,0.9)], aes(price)) +
  # geom_histogram(aes(fill=skeleton_type), binwidth = 100,col='black', size=.1) +
  geom_histogram(aes(fill=skeleton_type), bins=100, col="black", size=.1) +
  labs(title="نمودار بافت نگار قیمت (9 دهک اول) ساختمان ها قیمت به تفکیک نوع اسکلت ساختمان", x="قیمت هر ساختمان به میلیون تومان", y = "تعداد")+
  theme(plot.title = element_text(hjust = 0.5))
```

نمودار بافت نگار قیمت (9 دهک اول) ساختمان ها قیمت به تفکیک نوع اسکلت ساختمان



```
options(repr.plot.width = 8, repr.plot.height = 3.5)
ggplot(khoozestan[khoozestan$price_per_square < quantile(khoozestan$price_per_square,0.9),], aes(price_per_square)) +
  geom_histogram(aes(fill=skeleton_type), binwidth = 0.1,col='black', size=.1) +
  # geom_histogram(aes(fill=skeleton_type), bins=100, col="black", size=.1) +
  labs(title="نمودار بافت نگار قیمت هر متر مربع (9 دهک اول) ساختمان به تفکیک نوع اسکلت ساختمان", x="قیمت هر متر مربع ساختمان به میلیون تومان", y = "تعداد")+
  theme(plot.title = element_text(hjust = 0.5))
```

نمودار بافت نگار قیمت هر متر مربع (9 دهک اول) ساختمان ها قیمت به تفکیک نوع اسکلت ساختمان



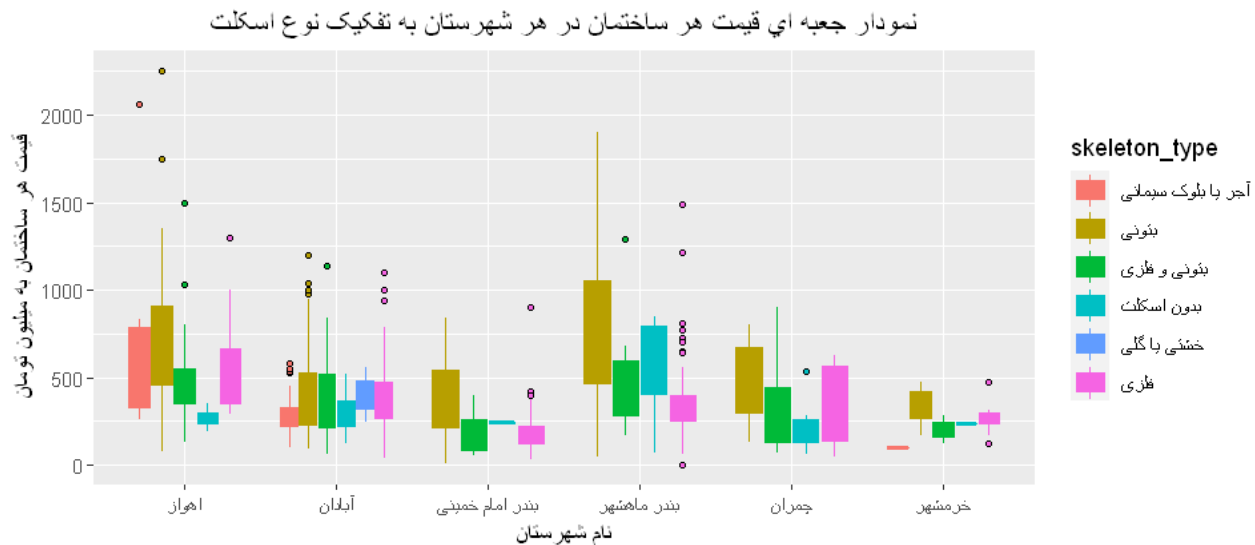
این کار را با قرار دادن مقدار bins برابر با ۱۰۰ انجام داده ایم.

در نمودار دوم کل ۹ دهک اول را به ازای هر ۱۰۰ هزار تومان یک بازه جدید ایجاد کرده ایم و تعداد داده های مربوط به هر نوع از اسکلت ها را در آن نشان داده ایم.

این کار را با قرار دادن مقدار binwidth برابر با ۰.۱ انجام داده ایم چون واحد ما در این داده ها میلیون تومان است پس ۰.۱ آن برابر با ۱۰۰ هزار تومان است.

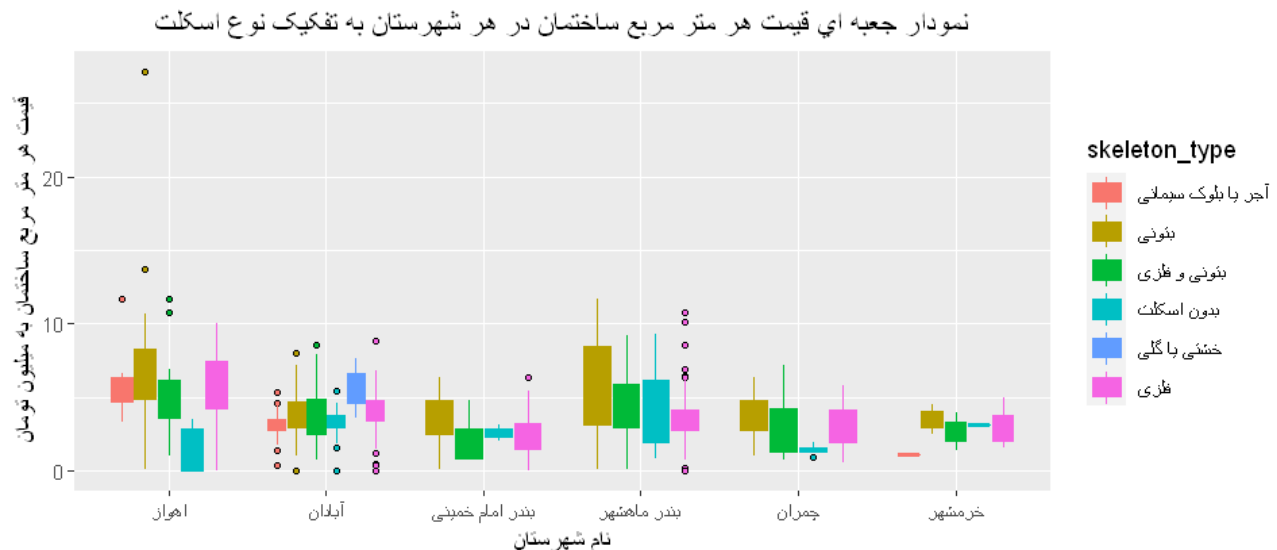
نمودار بعدی که رسم میکنیم نمودار جعبه ای قیمت شهرستان ها به تفکیک نوع اسکلت در آن ها میباشد.

```
options(repr.plot.width = 8, repr.plot.height = 3.5)
ggplot(khoozestan, aes(x=city, y=price , fill = skeleton_type)) +
  geom_boxplot(aes(colour = skeleton_type), outlier.colour="black", outlier.shape=21, outlier.size=1) +
  labs(title="نمودار جعبه ای قیمت هر ساختمان در هر شهرستان به تفکیک نوع اسکلت", y="قیمت هر ساختمان به میلیون تومان", x = "نام شهرستان") +
  theme(plot.title = element_text(hjust = 0.5))
```



و سپس این نمودار را برای قیمت هر متر مربع ساختمان نیز رسم میکنیم.

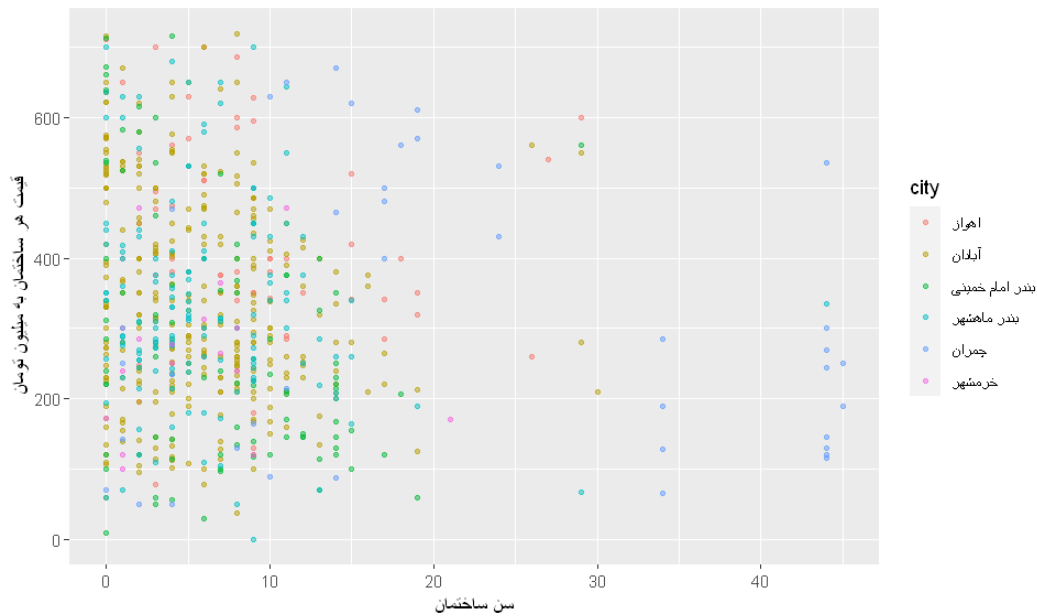
```
options(repr.plot.width = 8, repr.plot.height = 3.5)
ggplot(khoozestan, aes(x=city, y=price , fill = skeleton_type)) +
  geom_boxplot(aes(colour = skeleton_type), outlier.colour="black", outlier.shape=21, outlier.size=1) +
  labs(title="نمودار جعبه ای قیمت هر ساختمان در هر شهرستان به تفکیک نوع اسکلت", y="قیمت هر ساختمان به میلیون تومان", x = "نام شهرستان") +
  theme(plot.title = element_text(hjust = 0.5))
```



برای بررسی تاثیر سن ساختمان روی قیمت ساختمان نمودار پراکنشی داده های این دو ستون را به تفکیک شهرستان رسم میکنیم.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(khoozestan[khoozestan$price < quantile(khoozestan$price,0.9)], aes(x=building_age, y=price , color = city)) +
  geom_point(size=1,alpha=0.5) +
  labs(title="نمودار پراکنش قیمت (9 دهک اول) هر ساختمان بر اساس سن ساختمان به تفکیک شهرستان", x="سن ساختمان", y = "قیمت هر ساختمان به میلیون تومان") +
  theme(plot.title = element_text(hjust = 0.5))
```

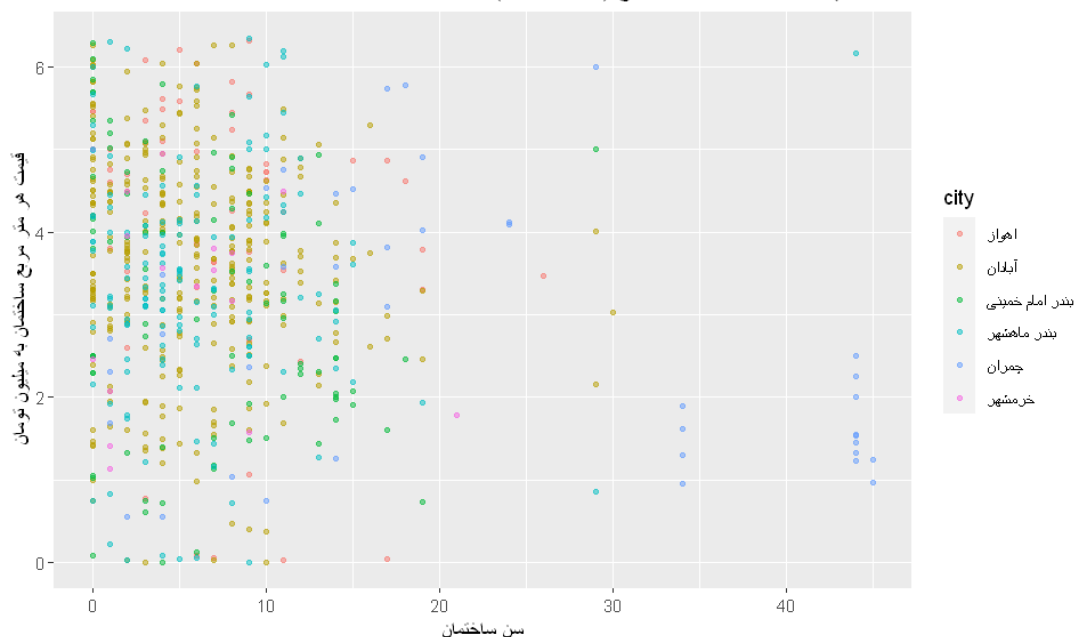
نمودار پراکنش قیمت (9 دهک اول) هر ساختمان بر اساس سن ساختمان به تفکیک شهرستان



و سپس این نمودار را برای قیمت هر متر مربع نیز رسم میکنیم.

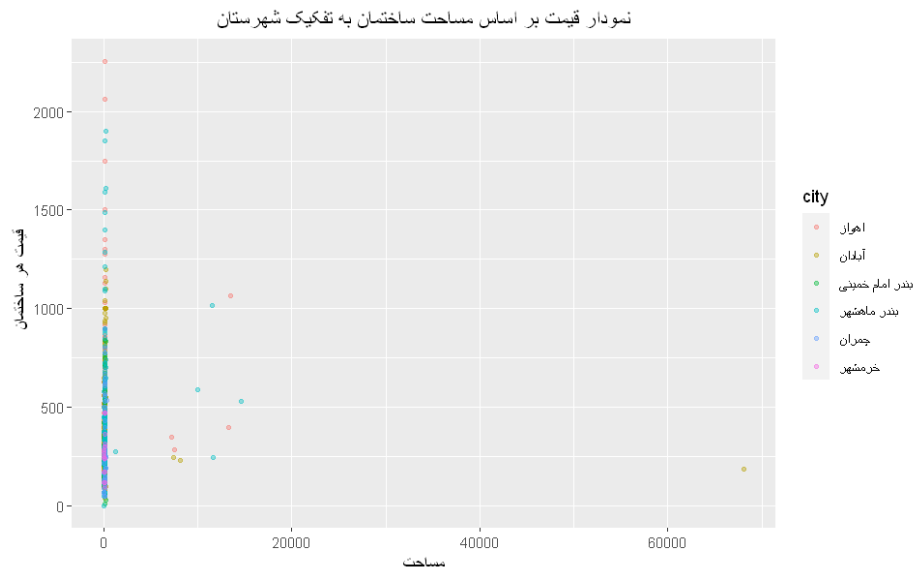
```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(khoozestan[khoozestan$price_per_square < quantile(khoozestan$price_per_square,0.9)], aes(x=building_age, y=price_per_square , color = city)) +
  geom_point(size=1,alpha=0.5) +
  labs(title="نمودار پراکنش قیمت هر متر مربع (9 دهک اول) ساختمان بر اساس سن ساختمان به تفکیک شهرستان", x="سن ساختمان", y = "قیمت هر متر مربع ساختمان به میلیون تومان") +
  theme(plot.title = element_text(hjust = 0.5))
```

نمودار پراکنش قیمت هر متر مربع (9 دهک اول) ساختمان بر اساس سن ساختمان به تفکیک شهرستان



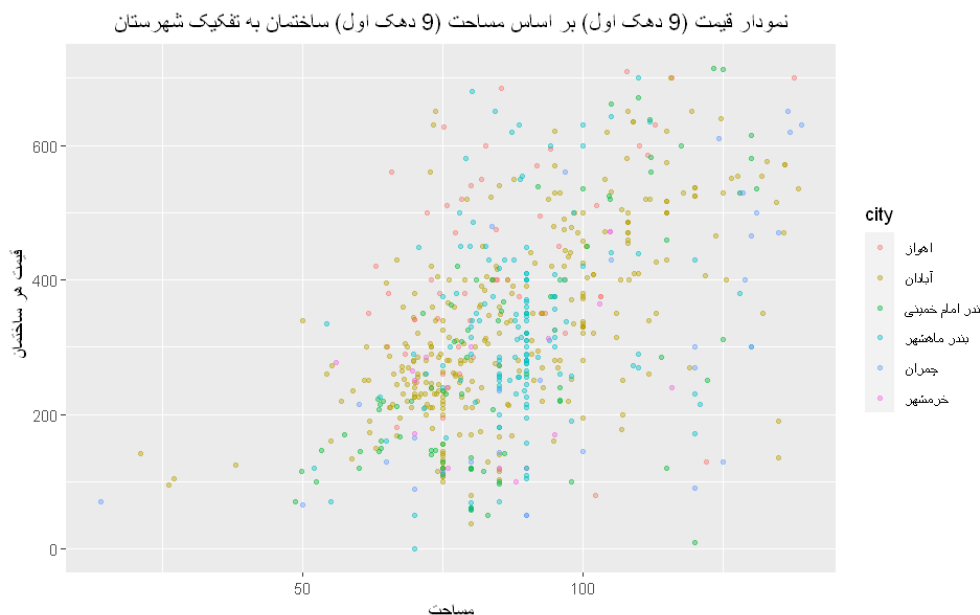
سپس برای بررسی تاثیر مساحت روی قیمت ساختمان نمودار زیر را رسم میکنیم.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(khoozestan, aes(x=area, y=price , color = city)) +
  geom_point(size=1,alpha=0.4,) +
  labs(title="نمودار قیمت بر اساس مساحت ساختمان به تفکیک شهرستان", y="قیمت هر ساختمان", x = "مساحت")+
  theme(plot.title = element_text(hjust = 0.5))
```



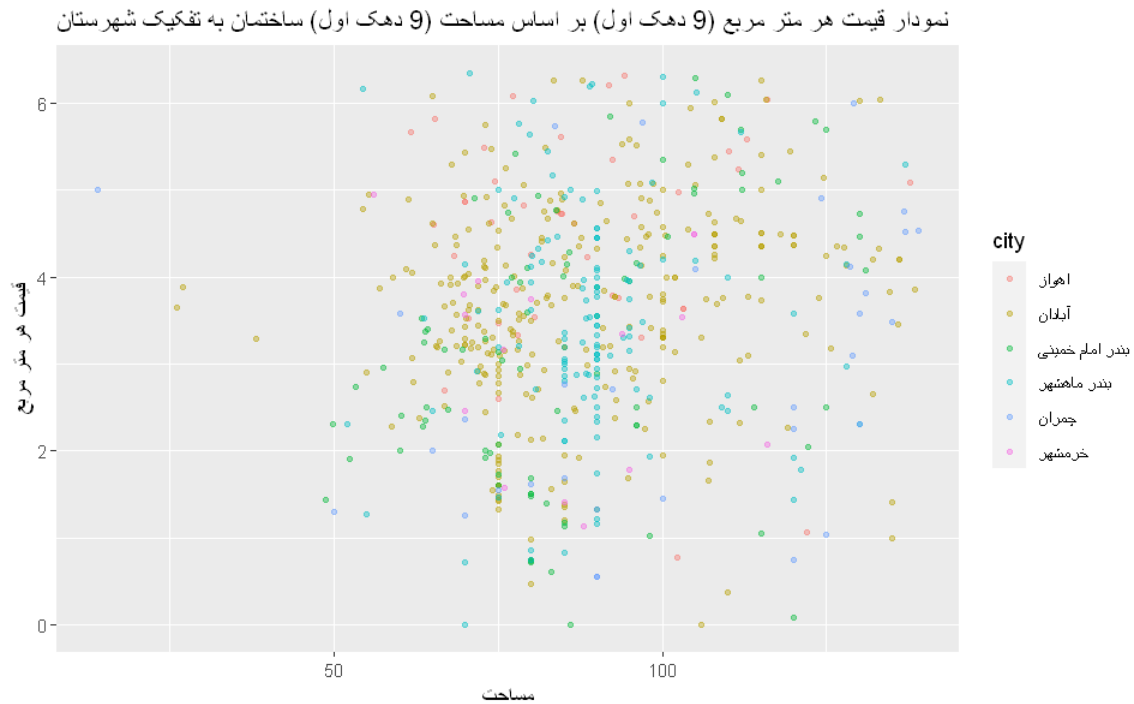
همان طور که مشاهده میکنید در این جا نیز به دلیل وجود داده های پرت و ساختمان هایی با مساحت های بسیار بالا بررسی بخش عمده ای از داده ها عملا غیر ممکن شده است بنابراین در این جا نیز تنها ۹ دهک اول داده ها را بررسی میکنم. در تمام بخش های این گزارش که تنها ۹ دهک از داده ها نمایش داده شده است به همین دلیل است که در غیر این صورت مصور سازی این داده ها نمیتوانست هیچ اطلاعاتی در اختیار ما قرار دهد.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(khoozestan[(khoozestan$area < quantile(khoozestan$area,0.90))&(khoozestan$price < quantile(khoozestan$price,0.9))],
  aes(y=price, x=area , color = city)) +
  geom_point(size=1,alpha=0.4,) +
  labs(title="نمودار قیمت (9 دهک اول) بر اساس مساحت (9 دهک اول) ساختمان به تفکیک شهرستان", y="قیمت هر ساختمان", x = "مساحت")+
  theme(plot.title = element_text(hjust = 0.5))
```



سپس همین نمودار را برای قیمت هر متر مربع نیز رسم میکنیم.

```
options(repr.plot.width = 8, repr.plot.height = 5)
ggplot(khoozestan[(khoozestan$area < quantile(khoozestan$area,0.90))&
(khoozestan$price_per_square < quantile(khoozestan$price_per_square,0.9))],
aes(y=price_per_square, x=area , color = city)) +
geom_point(size=1,alpha=0.4,) +
labs(title="نمودار قیمت هر متر مربع (9 دهک اول) بر اساس مساحت (9 دهک اول) ساختمان به تفکیک شهرستان", y="قیمت هر متر مربع", x = "مساحت")
```



بررسی مدل های پیشگو روی داده ها

۱. رگرسیون

- ۱,۱ رگرسیون با تمام متغیر های پیشگو
- ۱,۲ رگرسیون قدم به قدم پیشرو
- ۱,۳ رگرسیون قدم به قدم پسرو
- ۱,۴ رگرسیون قدم به قدم دوسویه
- ۱,۵ رگرسیون فرسا

۲. K-نزدیک ترین همسایه

۳. درخت های پیشگو

- ۳,۱ درخت پیشگو بدون تنظیم کردن پارامتر ها
- ۳,۲ عمیق ترین درخت پیشگو
- ۳,۳ عمیق ترین درخت پیشگو (هرس شده با کمترین xerror)
- ۳,۴ عمیق ترین درخت پیشگو (بهترین هرس)
- ۳,۵ جنگل تصادفی
- ۳,۶ درخت تقویت شده

۴. شبکه های عصبی مصنوعی

۱. رگرسیون

برای پیاده سازی مدل های پیشبینی ابتدا باید داده های مورد نیازمان را به درستی آماده کنیم تا بتوانیم عملکرد مدل های مختلف را به درستی روی آن بررسی کنیم چون اگر ما داده های نامناسب به هر مدل یادگیری بدهیم آن مدل نتیجه خوبی به ما نخواهد داد و یکی از مهم ترین بخش های آموزش مدل های مختلف پیش پردازش داده ها متناسب با ویژگی های آن مدل است.

در بخش تصویری سازی ستون های مختلف داده ها را معرفی کردیم و اطلاعات آن ها را نمایش دادیم.

همان‌طور که گفته شد برخی ستون‌های این داده‌ها هیچ اطلاعاتی در اختیار ما قرار نمی‌دهند یا اگر هم قرار می‌دهند اطلاعاتی است که جنبه پیشگویانه ندارد و ما نمی‌توانیم از این اطلاعات در مدل‌هایمان استفاده کنیم پس مانند بخش تصویری سازی این ستون‌ها را حذف می‌کنیم.

city	area	price	building_age	skeleton_type	postal_code
اهواز	75.00	195	2	بدون اسكلت	617563
اهواز	96.30	735	13	فلزي	191181
اهواز	111.60	585	8	بتوني	618493
اهواز	13313.00	400	11	فلزي	613668
اهواز	80.48	285	11	بتوني و فلزي	613495
اهواز	72.22	500	9	بتوني و فلزي	617763

برای اینکار از تابع `mapvalues` که در کتابخانه `plyr` موجود است استفاده میکنیم.

city before mapping

هندیجان' مسجد سلیمان' شبیان' شوشتر' شوش' رامشیر' لارقل' خرمشیر' چمران' ببیان' بندر ماهشهر' بندر امام خمینی' آبادان' ایذه' اهواز'

city after mapping

'ahvaz' 'izeh' 'abadan' 'bandare_emam_khomeini' 'bandare_mahshahr' 'behbahan' 'chamran' 'khorramshahr' 'dezfool' 'raamshir' 'shoosh'
'shooshtar' 'shiban' 'masjed_soleiman' 'handijan'

skeleton_type before mapping

‘خشتی یا گلی’ ‘آجر یا بلوک سیمانی’ ‘بتونی و فلزی’ ‘بتونی’ ‘فلزی’ ‘بدون اسکلت’

skeleton_type after mapping

'none' 'metal' 'concrete' 'metal concrete' 'brick or cement block' 'adobe or clay'

پس از این مرحله داده ما به شکل زیر در می آید.

city	area	price	building_age	skeleton_type	postal_code
ahvaz	75.00	195	2	none	617563
ahvaz	96.30	735	13	metal	191181
ahvaz	111.60	585	8	concrete	618493
ahvaz	13313.00	400	11	metal	613668
ahvaz	80.48	285	11	metal_concrete	613495
ahvaz	72.22	500	9	metal_concrete	617763

در مدل رگرسیون ما قصد داریم تا یک ترکیب خطی از متغیر های پیشگو را پیشبینی کنیم تا بوسیله آن بتوانیم متغیر برآمد داده های مورد نظرمان را از روی اطلاعاتشان پیشگویی کنیم. برای این کار نیاز است که بتوان به صورت عددی این متغیر ها را نشان داد پس باید راهی پیدا کنیم تا بتوان متغیر های رسته ای را به صورت عددی نمایش داد.

اگر متغیر رسته ای مورد نظر حالت ترتیبی داشته باشد ، مثلا متغیری که مقادیر مختلف آن "بد" "متوسط" "خوب" باشند، در این جا میتوان متغیر ها را به اعدادی ترتیبی تبدیل کرد مثلا در این جا به -۱ , ۰ , ۱ میتوان تبدیل کرد. اما اگر این متغیر ها ترتیبی نداشته باشند باید آن ها را به dummy_variable تبدیل کنیم که به آن نمایش one hot encoding نیز میگویند که به ازای هر رسته در هر متغیر رسته ای یک متغیر جدید ساخته میشود که اگر آن داده عضو آن رسته باشد مقدارش برابر ۱ و در غیر اینصورت برابر ۰ خواهد بود.

در این جا ستون city و skeleton_type متغیر های رسته ای هستند که ترتیب در آن ها معنا ندارد و باید به شکل dummy_variable تبدیل شوند. ستون کد پستی با این که یک ستون عددی میباشد اما عدد های این ستون در واقعیت نشان دهنده محله های مختلف در شهرستان های مختلف آن استان میباشد و باید همانند ستون city آن ها را نیز به dummy_variable تبدیل کنیم. نمیتوانیم بگوییم که یک کد پستی از دیگری بزرگتر است یا این که آن ها را با هم جمع کنیم و این نشان دهنده این است که این ستون از جنس عدد نیست و از جنس رسته است و باید با آن مانند یک متغیر رسته ای عمل کرد. ابعاد جدول داده ها بعد از این تبدیلات به شکل زیر میشود.

```
dim(dummy_khoozestan)
768 327
```

از ۰.۸ داده ها برای آموزش مدل و از ۰.۲ دیگر برای اعتبار سنجی استفاده میکنیم.

```
set.seed(831)

train_index = sample.int(nrow(dummy_khoozestan), 0.8*nrow(dummy_khoozestan), replace=F)

train = dummy_khoozestan[train_index,]
validation = dummy_khoozestan[-train_index,-c(17)]
y_validation = dummy_khoozestan[-train_index,c(17)]
```

```
dim(train)
614 327

dim(validation)
154 326
```

۱.۱ رگرسیون با تمام متغیر های پیشگو

در این بخش ابتدا یک مدل رگرسیون با تمام متغیر های پیشگو میسازیم.

```
all.reg <- lm(price ~ . ,data = train)
```

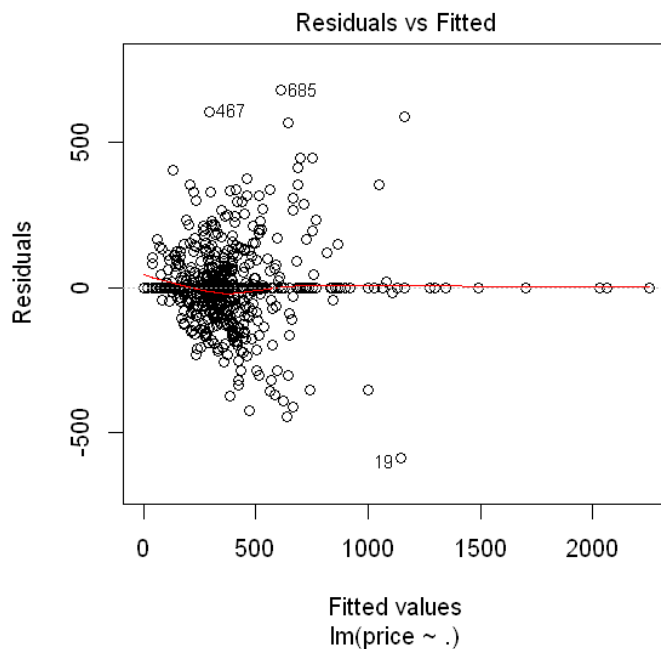
به دلیل بزرگ بودن جدول summary اطلاعات متغیر ها را در این جا نمیتوان نشان داد ، برای بررسی آن ها به کد مراجعه شود.

```
Residual standard error: 184.9 on 339 degrees of freedom
Multiple R-squared: 0.7593, Adjusted R-squared: 0.5647
F-statistic: 3.903 on 274 and 339 DF, p-value: < 0.0000000000000022
```

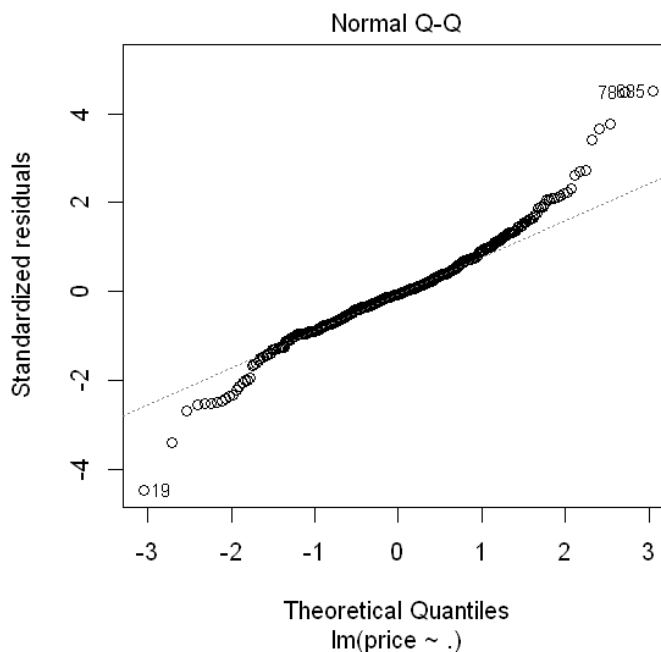
بعد از fit کردن مدل میتوان با دستور plot چند نمودار برای بررسی عملکرد مدل رسم کرد که این نمودار ها را برای تمام مدل ها دیگر نیز رسم میکنیم.

یک بار مختصرا هر کدام را توضیح میدهیم.

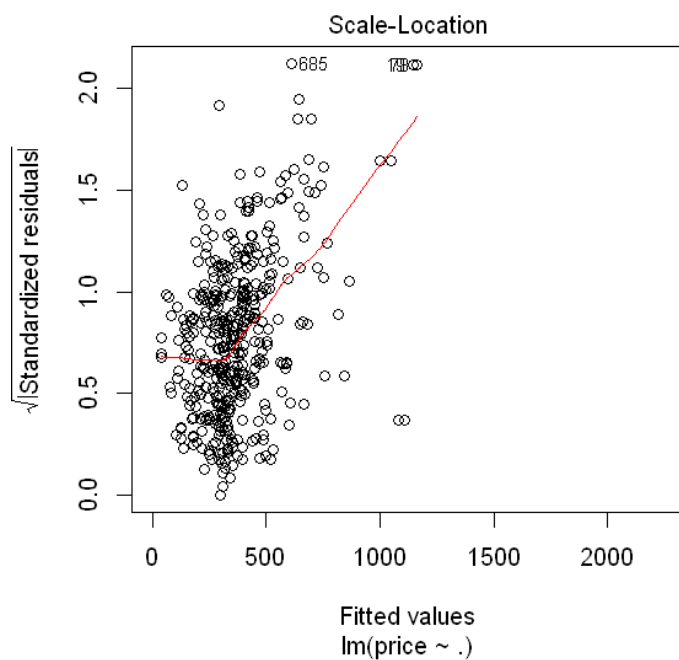
نمودار اول نمودار خطای پیشبینی بر اساس مقدار پیش بینی شده است ، هرچقدر نقاط به خط افقی نزدیک تر باشند یعنی مدل ما خطای کمتری دارد و خط قرمز وسط صفحه نیز نشان میدهد که میانگین خطا در قیمت های مختلف چقدر است که بهترین حالت آن درشرایطی است که این خط کاملا صاف باشد. اگر این خط صاف نباشد به این معنی است که شرط **linearity** در مدل رعایت نشده است و باید تبدیل هایی در متغیر های پیشگو یا برآمد به وجود آید تا مشکل حل شود و همچنین داده های پرت را نیز به ما نشان میدهد. بهترین حالت زمانی است که تراکم داده ها در طول محور X ثابت باشد.



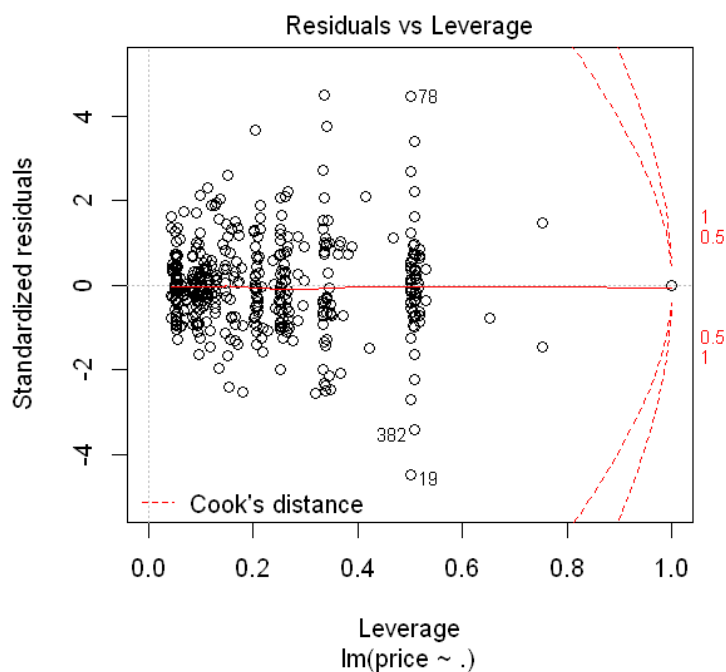
نمودار دوم بررسی میکند که آیا خطا ها از توزیع نرمال پیروی میکنند یا خیر که بهترین حالت این است که تمام نقاط روی خط قرار بگیرند.



نمودار سوم واریانس خطا را بررسی میکند که در آن بهترین حالت این است که خط قرمز کاملاً افقی باشد و توضیح داده‌ها دور خط قرمز در تمام X ها به یک شکل باشد.



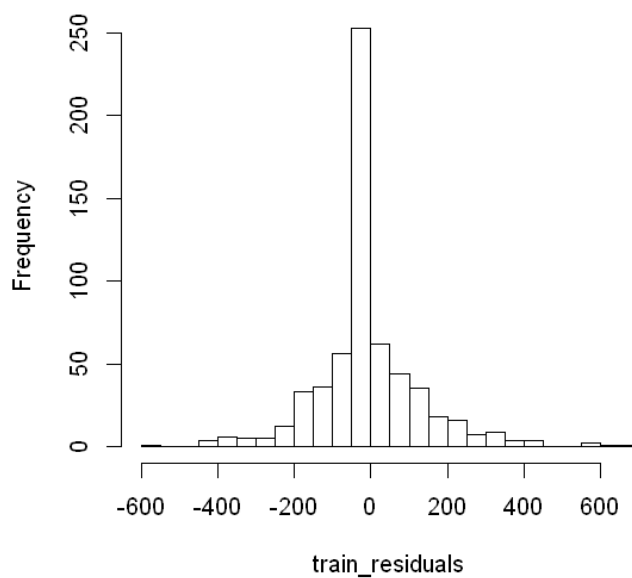
نمودار چهارم نیز خطای استاندارد شده نقاط را بر اساس قدرت آن‌ها در جابجایی خط رگرسیون نمایش میدهد که از این طریق میتوان داده‌های پرت را نیز بر اساس معیار فاصله کوک مشخص کرد که حذف آن‌ها باعث عملکرد بهتر مدل روی بقیه داده‌ها میشود. اما ما در این جا قصد حذف داده‌ها را نداریم.



دقت و توزیع خطاها روی داده های آموزشی و اعتبارسنجی به شکل زیر است.

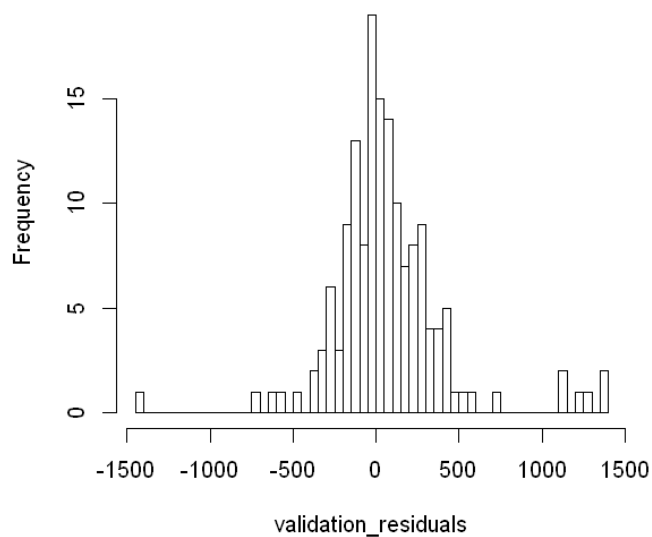
	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.000000000000004703078	137.406	85.20685	-2683.718	2700.118	0.4397862

Histogram of train_residuals



	ME	RMSE	MAE	MPE	MAPE
Test set	71.29738	353.6341	225.9979	-9446.7	9494.791

Histogram of validation_residuals



۱،۲ رگرسیون قدم به قدم پیشرو

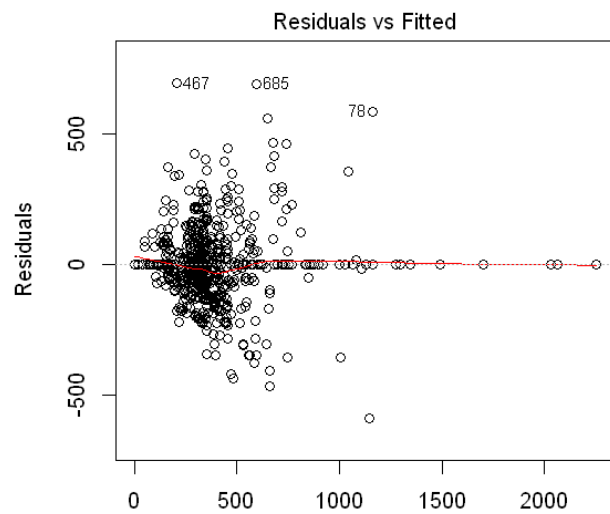
ابتدا یک فرمول از تمامی متغیر هایی که میتواند وارد مدل شود میسازیم و این فرمول را به عنوان scope به تابع step میدهیم و از یک مدل که تنها یک intercept دارد شروع میکنیم و مدل را گسترش میدهیم.

```
formula.all.variables = formula(lm(price~.,train))
```

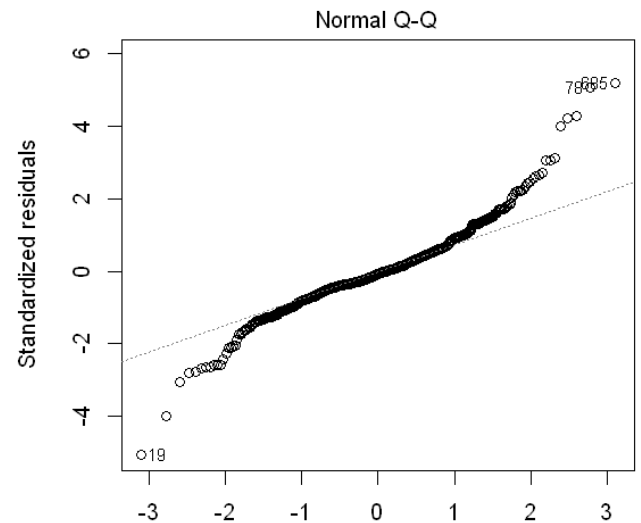
```
forward.reg = step(lm(price ~ 1 ,data = train),direction='forward',scope=formula.all.variables)
```

```
Residual standard error: 163.9 on 508 degrees of freedom  
Multiple R-squared: 0.7166, Adjusted R-squared: 0.658  
F-statistic: 12.23 on 105 and 508 DF, p-value: < 0.0000000000000022
```

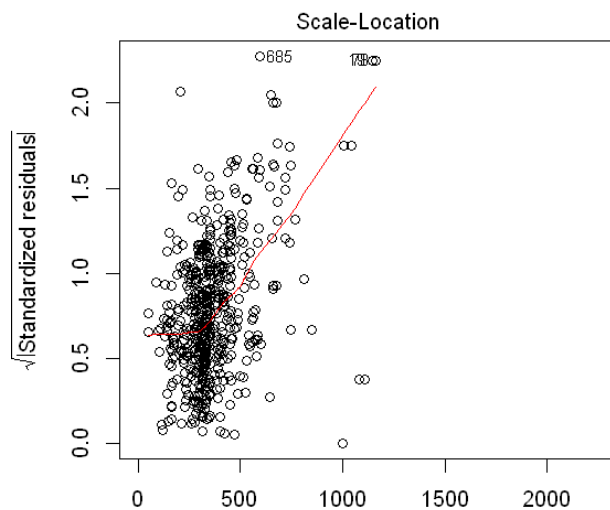
اطلاعات کامل تر از مدل درون کد موجود است.



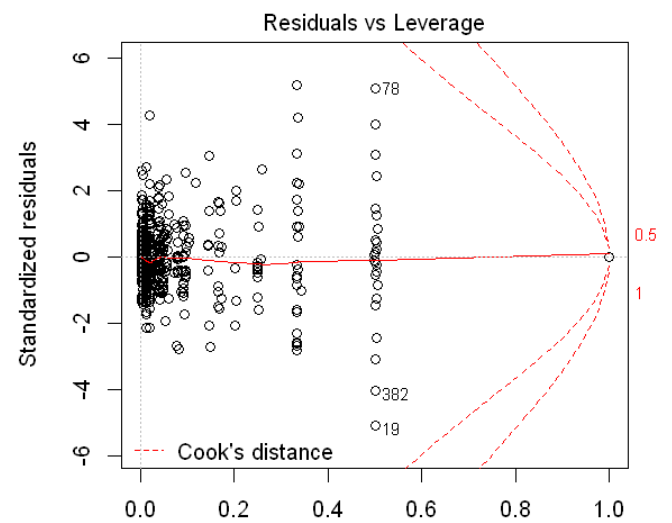
Fitted values
rice ~ cityahvaz + cityshiban + postal_code613865 + postal_code61!



Theoretical Quantiles
rice ~ cityahvaz + cityshiban + postal_code613865 + postal_code61!



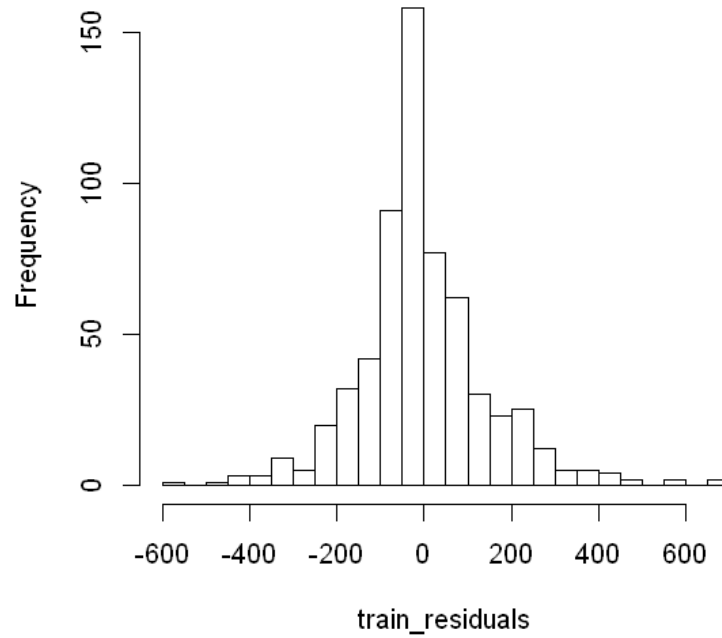
Fitted values
rice ~ cityahvaz + cityshiban + postal_code613865 + postal_code61



Leverage
rice ~ cityahvaz + cityshiban + postal_code613865 + postal_code61!

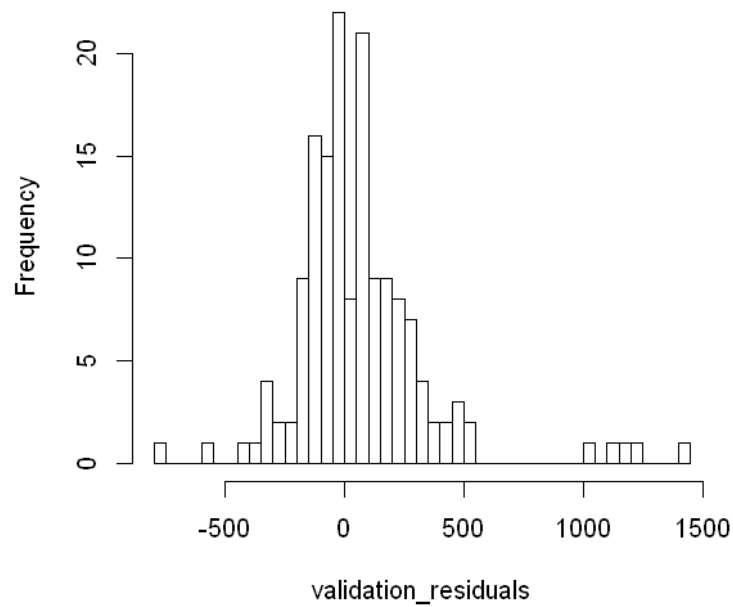
	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	0.000000000000002153634	149.0952	102.6793	-2555.901	2575.02	0.5299685

Histogram of train_residuals



	ME	RMSE	MAE	MPE	MAPE
Test set	63.64744	296.6739	189.6729	-10821.63	10856.77

Histogram of validation_residuals



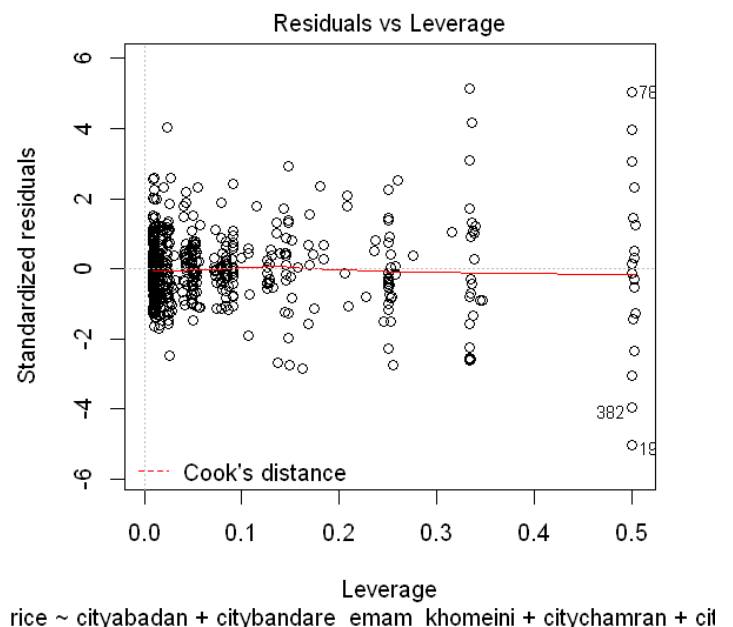
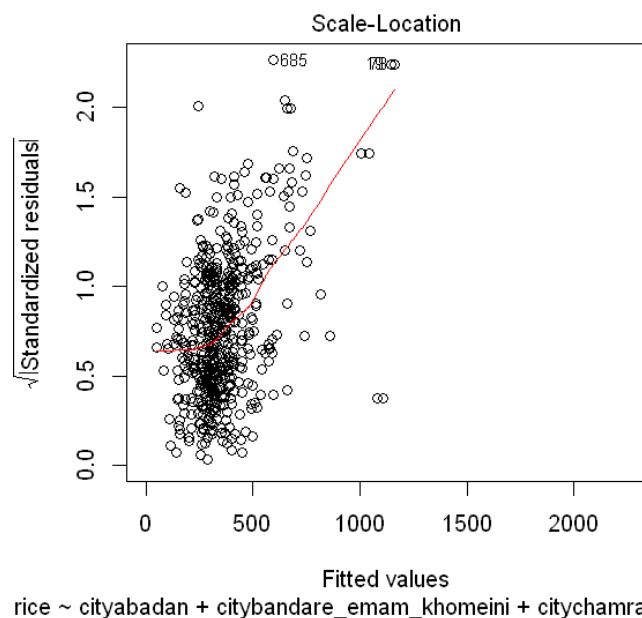
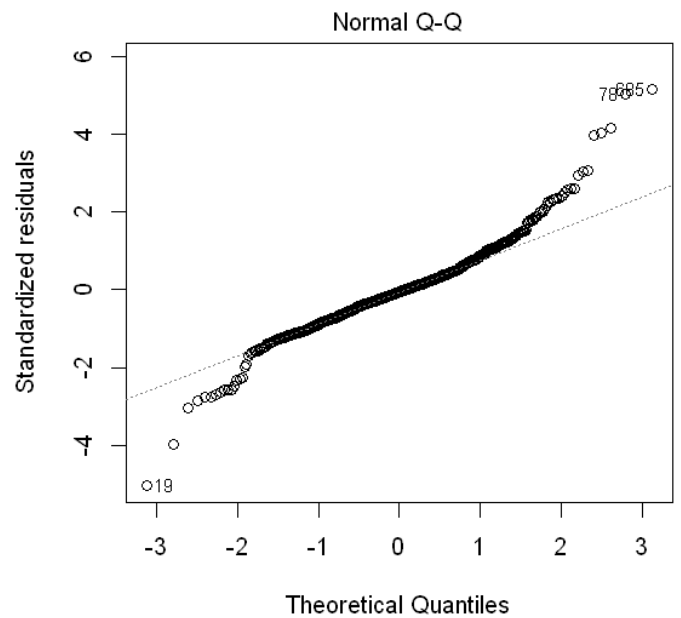
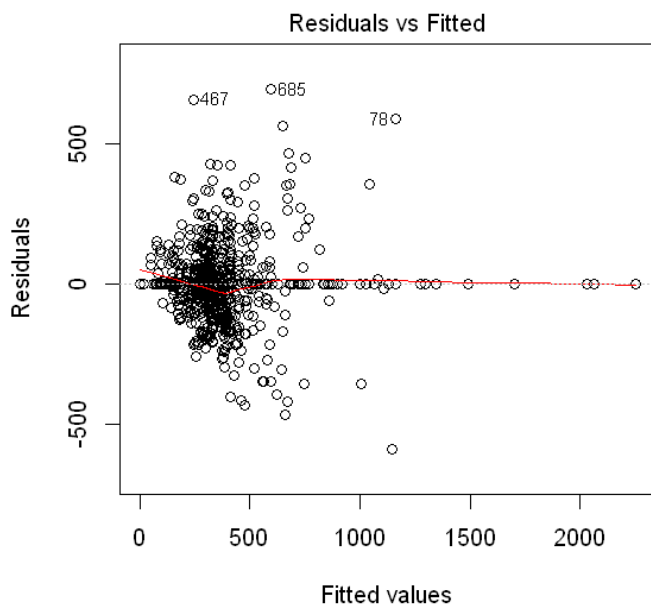
۱,۳ رگرسیون قدم به قدم پسرو

ابتدا یک مدل با استفاده از تمام پیشگو ها میسازیم و سپس در هر مرحله یکی از پیشگو های اضافی را حذف میکنیم.

```
backward.reg = stepAIC(lm(price ~ . ,data = train),direction='backward')
```

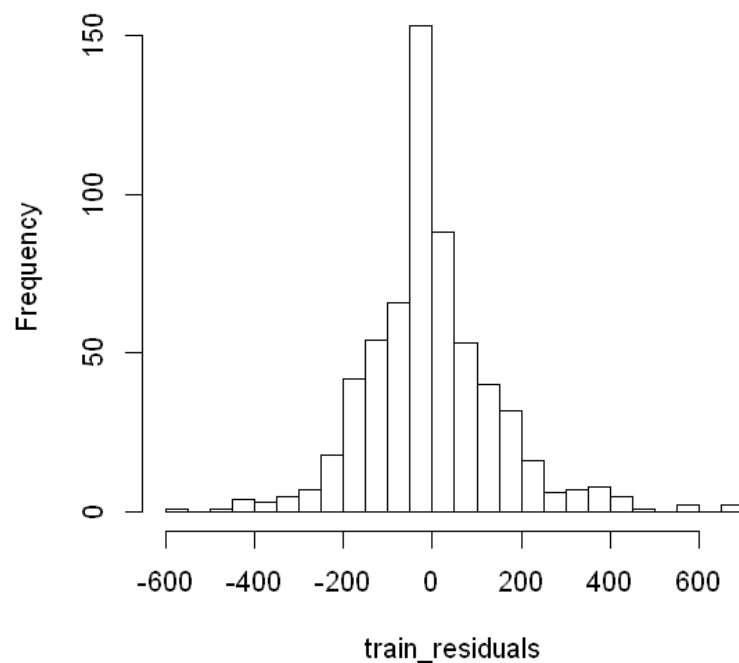
```
Residual standard error: 165.1 on 505 degrees of freedom  
Multiple R-squared: 0.714, Adjusted R-squared: 0.6528  
F-statistic: 11.67 on 108 and 505 DF, p-value: < 0.0000000000000022
```

اطلاعات کامل تر از مدل درون کد موجود است.



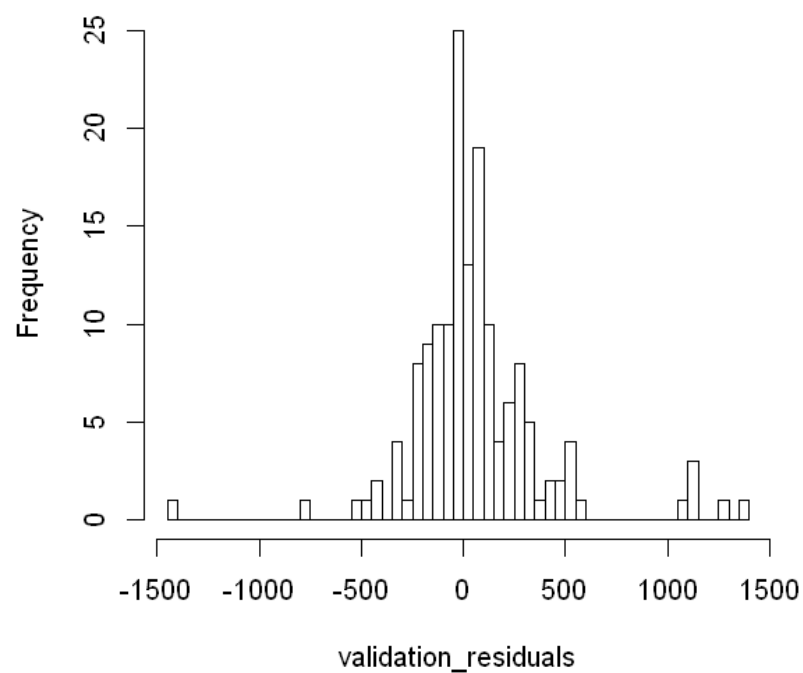
	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.00000000000003810937	149.7751	104.3567	-2993.646	3013.738	0.5386261

Histogram of train_residuals



	ME	RMSE	MAE	MPE	MAPE
Test set	59.59292	336.5469	210.0531	-11009.88	11050.56

Histogram of validation_residuals



۱,۴ رگرسیون قدم به قدم دوسویه

رگرسیون قدم به قدم مانند رگرسیون پسرو میباشد و از یک مدل با تمام متغیرها شروع میشود با این تفاوت که در هر مرحله پس از حذف کردن یکی از متغیرها بررسی میکند که آیا اضافه کردن یکی دیگر از متغیرها میتواند AIC را پایین بیاورد یا خیر اگر چنین تغییری پیدا کرد آن را به مدل اضافه میکند.

```
stepwise.reg = step(lm(price ~ . ,data = train),direction='both')
```

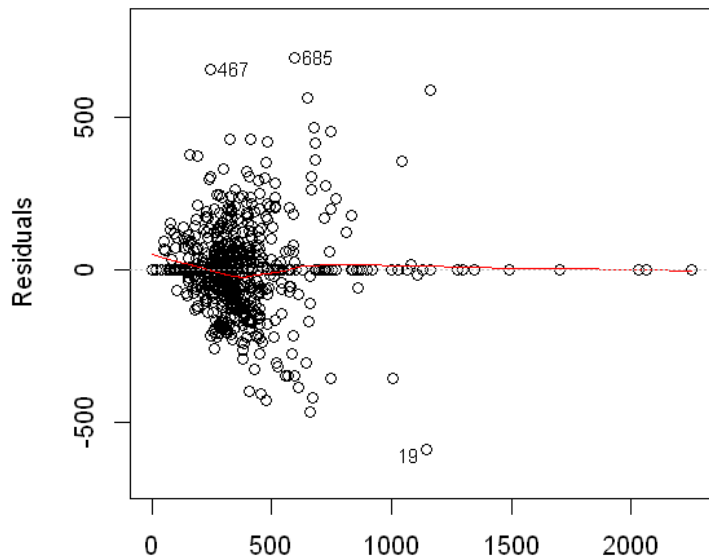
Residual standard error: 163.6 on 500 degrees of freedom

Multiple R-squared: 0.7221, Adjusted R-squared: 0.6593

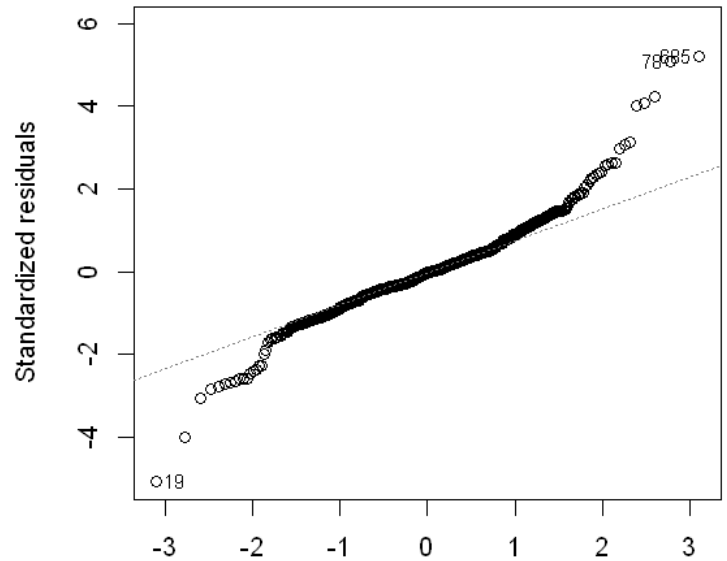
F-statistic: 11.5 on 113 and 500 DF, p-value: < 0.00000000000000022

اطلاعات کامل تر از مدل درون کد موجود است.

Residuals vs Fitted



Normal Q-Q



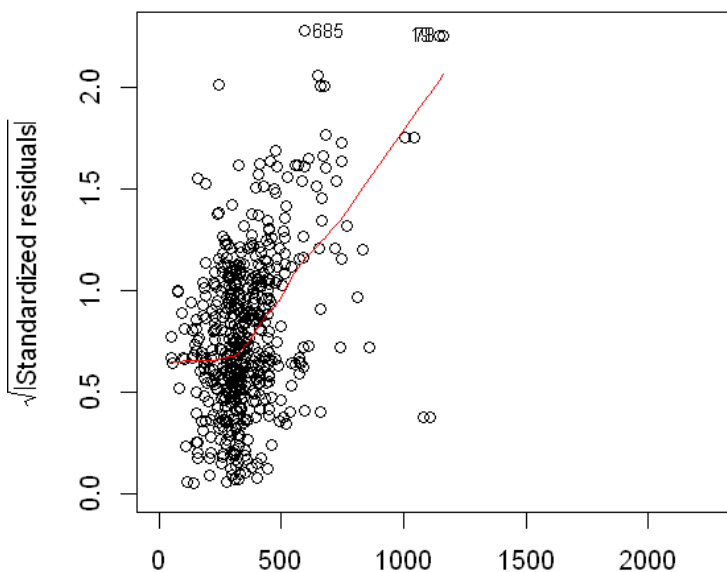
Fitted values

rice ~ cityabadan + citybandare_emam_khomeini + citychamran + cit

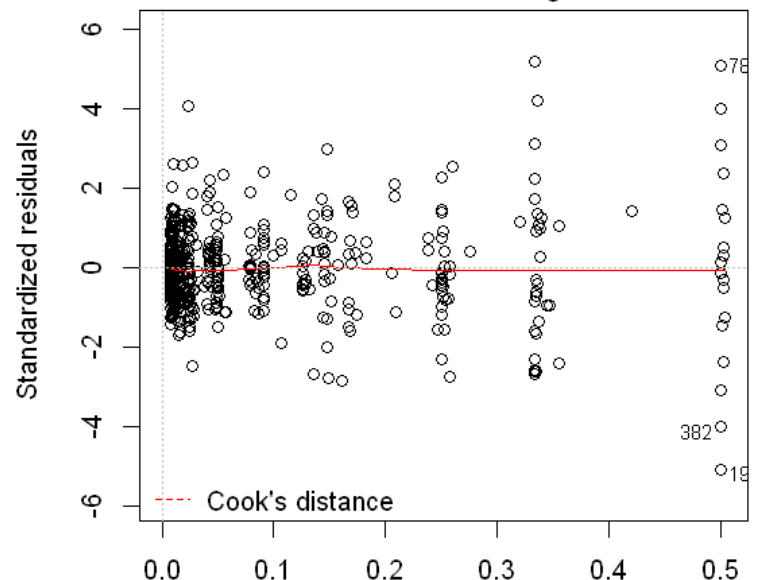
Theoretical Quantiles

rice ~ cityabadan + citybandare_emam_khomeini + citychamran + cit

Scale-Location



Residuals vs Leverage



Fitted values

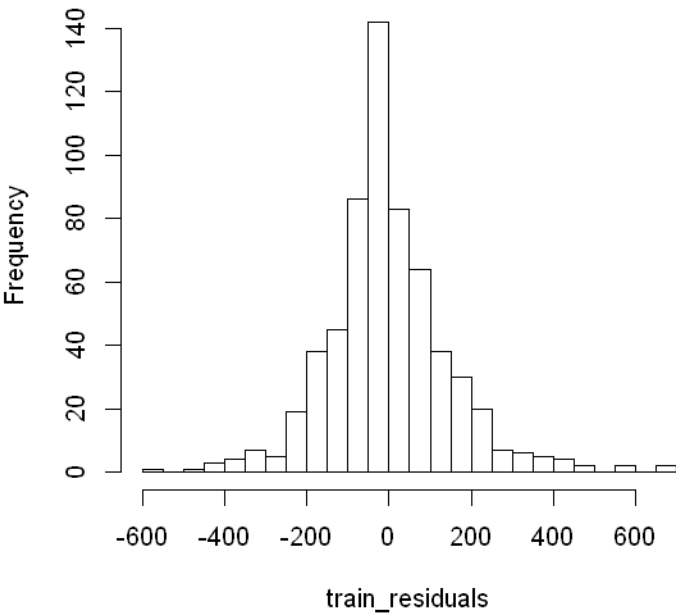
rice ~ cityabadan + citybandare_emam_khomeini + citychamran

Leverage

rice ~ cityabadan + citybandare_emam_khomeini + citychamran + cit

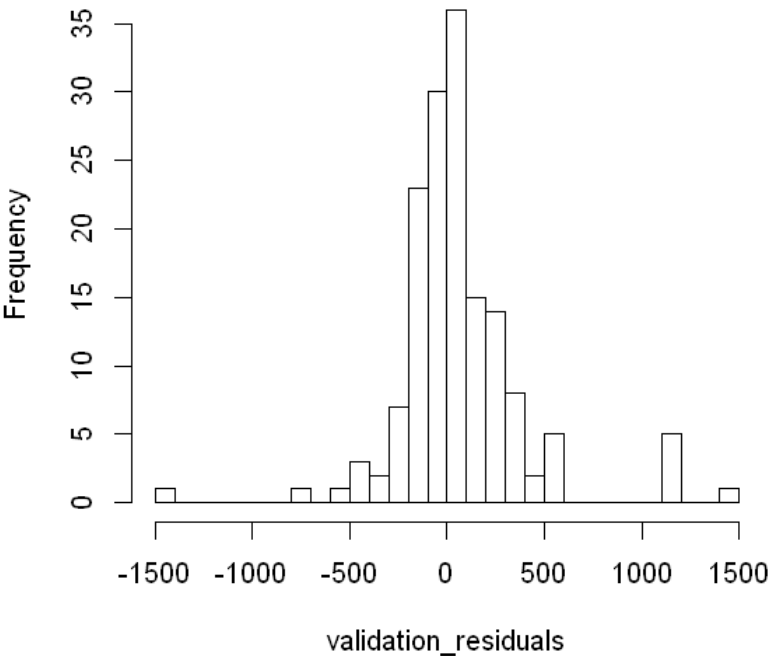
	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	-0.00000000000000429018	147.634	102.7167	-2993.936	3013.852	0.5301617

Histogram of train_residuals



	ME	RMSE	MAE	MPE	MAPE
Test set	66.7623	336.73	209.9644	-10761.34	10803.69

Histogram of validation_residuals



۱,۵ رگرسیون فرسا

این مدل با بررسی کردن تمام ترکیب های مختلف از متغیر های پیشگو و بررسی عملکرد آن ها بهترین مدل را به شما معرفی میکند.

مزیت این مدل این است که قطعا بهترین مدل را پیدا میکند اما ضعف این مدل این است که زمانی که نیاز است تا این مدل پیدا شود رابطه نمایی با تعداد متغیر ها دارد و برای مثال در این داده ها که تعداد متغیر های پیشگو ۳۲۶ میباشد مدل باید ۲^{۳۲۶} رگرسیون عادی را امتحان کند و نتایج را با یکدیگر مقایسه کند.

سعی کردم این مدل را اجرا کنم اما بعد از ۱۸ ساعت مدل موفق به حل این مسئله نشد به همین دلیل فعالیت آن را قطع کردم.

```
exhaustive.reg = regsubsets(price ~ . ,data = train, nbest=1, nvmax = dim(train)[2], method = 'exhaustive',really.big=T)
```

بررسی :

آزمایشی (Test)					آموزشی (Train)					با تمام متغیر ها قدم به قدم پیشرو قدم به قدم پسرو قدم به قدم دوسویه فرسا
MAPE	MPE	MAE	RMSE	ME	MAPE	MPE	MAE	RMSE	ME	
9494.79	-9446.7	225.99	353.63	71.29	2700.11	-2683.71	85.2	137.4	0	
10856.77	-10821.6	189.67	296.67	63.64	2572.02	-2555.9	102.67	149.09	0	
11050.56	-11009.9	210.05	336.54	59.59	3013.73	-2993.64	104.35	149.77	0	
10803.69	-10761.3	209.96	336.73	66.76	3013.85	-2993.93	102.71	147.63	0	
0	0	0	0	0	0	0	0	0	0	

بهترین مدل رگرسیون ما مدل رگرسیون قدم به قدم پیشرو بود که توانست **Adjusted** را نیز به ۰.۶۸ برساند و همچنین خطای **MAE** و **RMSE** این مدل روی داده های دیده نشده نسبت به بقیه مدل ها کمتر بود.

۲.K-نزدیک ترین همسایه

در این بخش علاوه بر تمام پیش پردازش هایی که در مدل رگرسیون انجام دادیم لازم است تا داده ها را نرمال نیز بکنیم، زیرا در مدل knn ما با استفاده از پیدا کردن نزدیک ترین همسایه ها به نقطه مورد نظرمان که قصد پیشگویی برای آن را داریم و میانگین گیری از مقادیر متغیر برآمد آن ها پیشگویی را انجام میدهیم و محاسبه این فاصله از فاصله اقلیدسی بین نقاط استفاده میکنیم که فرمول آن به شکل زیر است.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_i - q_i)^2 + \dots + (p_n - q_n)^2}.$$

که در این جا اعداد ۱ تا n مشخص کننده ویژگی های مختلف نقاط p و q هستند.

در این جا برای این اگر یکی از متغیر ها شامل اعداد بزرگتری نسبت به دیگر متغیر ها باشد ، در محاسبه فاصله بین این نقاط تاثیر سایر متغیر ها بسیار کم میشود و عملا تغییری که فاصله بین نقاط را مشخص میکند تغییری است که شامل اعداد بزرگتری میباشد و فاصله این دو نقطه در بقیه متغیر ها در برابر این متغیر قابل چشم پوشی است و این امر باعث میشود که این مدل عملکرد ضعیفی داشته باشد به همین دلیل برای از بین بردن تاثیر مقیاس متغیر های مختلف در این فرمول از نرمال کردن استفاده میکنیم که باعث میشود دامنه تغییر تمام متغیر ها به بازه [0,1] محدود شود.

برای این کار از تابع زیر استفاده میکنیم.

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

و این تابع را روی تمام متغیر ها به جز متغیر برآمد اجرا میکنیم.

و سپس کل داده ها را به دو بخش train و test تقسیم میکنیم.

```
train_index = sample.int(nrow(dummy_khoozestan),0.8*nrow(dummy_khoozestan),replace=F)  
  
X = as.data.frame(lapply(dummy_khoozestan[, -c(17)],normalize))  
Y = dummy_khoozestan['price']  
  
x_train = X[train_index,]  
y_train = Y[train_index,]  
train = cbind(x_train,y_train)  
  
x_test = X[-train_index,]  
y_test = Y[-train_index,]  
test = cbind(x_test,y_test)
```

در این جا برای انتخاب بهترین مدل و همچنین بررسی درست عملکرد مدل قصد داریم که مدل را به روش cross validation روی داده train آموزش دهیم و از روی آن بهترین مقدار k را انتخاب کنیم و سپس عملکرد آن را روی داده test که تا به حال ندیده است بررسی کنیم.

برای این کار ابتدا تابع رو به رو را تعریف میکنیم که وظیفه

آن تولید times بار نمونه تصادفی از آرایه y است به طوری

که در هر نمونه فقط p*(length(y)) عضو دارد و p عددی

بین ۰ و ۱ است.

```
createRandomDataPartition <- function(y, times, p) {  
  vec <- 1:length(y)  
  n_samples <- round(p * length(y))  
  
  result <- list()  
  for(t in 1:times){  
    indices <- sample(vec, n_samples, replace = FALSE)  
    result[[t]] <- indices  
    #names(result)[t] <- paste0("Resample", t)  
  }  
  names(result) <- prettySeq(result)  
  result  
}
```

که نمونه عملکرد آن را در اینجا مشاهده میکنید.

```
createRandomDataPartition(y=1:10 , time=5 , p=0.6)
```

\$Resample1

6 9 10 4 3 8

\$Resample2

3 9 8 5 1 4

\$Resample3

7 8 3 9 5 10

\$Resample4

5 10 4 9 8 7

\$Resample5

5 8 4 1 2 9

۵ نمونه از داده های (1,2,3,4,5,6,7,8,9,10) به طوری که

طول هر نمونه برابر $10 * 0.6 = 6$ میباشد.

قصد داریم عملکرد مدل ما به این شکل باشد که مدل را برای k های مختلف به تعداد نمونه ابتدا روی اندیس هایی که در نمونه وجود دارد آموزش دهیم و سپس از اندیس هایی که در نمونه وجود ندارد برای اعتبار سنجی داده ها استفاده کنیم و در نهایت در هر k شاخص های دقت را با میانگین گیری از شاخص های دقت مربوط به نمونه های آن k مشخص کنیم تا بتوانیم به درستی این k ها را با هم مقایسه کنیم و در نهایت عملکرد بهترین k را روی داده های $test$ بررسی کنیم.

در این جا باید یک `trainControl` بسازیم که عملکرد یادگیری را مدیریت کند و پارامتر `index` را برابر با اندیس نمونه های تصادفی از داده های آموزشی در نظر میگیریم و به این ترتیب اندیس هایی که در آن نمونه وجود نداشته باشد به عنوان داده های اعتبارسنجی بررسی میشوند.

```
set.seed(1234)
n_repeats <- 20
train_fraction <- 0.8

parts <- createRandomDataPartition(1:nrow(train), times = n_repeats, p = train_fraction)

ctrl <- trainControl(method = "repeatedcv", ## The method doesn't matter
  index= parts,
  savePredictions = TRUE
)
```

سپس فرایند یادگیری را به وسیله تابع `train` از کتابخانه `caret` به این صورت انجام میدهیم.

```
m1 <- train(y_train ~. ,
  tuneGrid = expand.grid(k=1:30),
  data = train,
  metric = 'MAE',
  method = 'knn',
  trControl = ctrl)
```

همان طور که مشاهده میکنید در این جا به جای استفاده از ۶۱۴ نمونه موجود در `train` تنها از ۴۹۱ عدد از این نمونه ها به صورت `fold`-۱۰ استفاده شده است و شاخص های دقتی که در صفحه بعد مشاهده میکنید میانگین بین این ۱۰ نمونه به ازای k های مختلف میباشدند.

k-Nearest Neighbors

614 samples
326 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 491, 491, 491, 491, 491, ...
Resampling results across tuning parameters:

k	RMSE	Rsquared	MAE
1	295.9929	0.1371818	185.2124
2	264.8218	0.1703314	174.5649
3	249.6771	0.2182988	165.7721
4	249.4402	0.2115939	165.4578
5	246.9486	0.2100887	163.8605
6	245.3673	0.2166829	163.3118
7	246.0461	0.2079222	163.9413
8	246.7112	0.2002395	164.6784
9	247.0609	0.1976539	165.8000
10	247.8849	0.1907857	167.0615
11	249.0790	0.1828342	167.8993
12	249.4082	0.1805639	168.6234
13	250.0844	0.1768484	169.8711
14	250.0874	0.1788288	170.4537
15	249.9776	0.1782771	170.6001
16	251.6690	0.1685201	171.7831
17	251.2107	0.1692580	171.8539
18	251.8960	0.1663435	172.4794
19	251.6729	0.1642199	172.5186
20	251.9242	0.1586018	172.6902
21	252.0149	0.1585415	172.3350
22	251.8685	0.1590744	172.0873
23	252.1537	0.1607392	172.5145
24	252.3013	0.1601493	172.7714
25	252.5642	0.1586088	173.1863
26	252.8201	0.1569045	173.5999
27	253.3949	0.1531235	174.0665
28	253.4523	0.1524731	173.9409
29	253.8992	0.1493840	174.3817
30	253.9711	0.1498748	174.5646

MAE was used to select the optimal model using the smallest value.
The final value used for the model was **k = 6**.

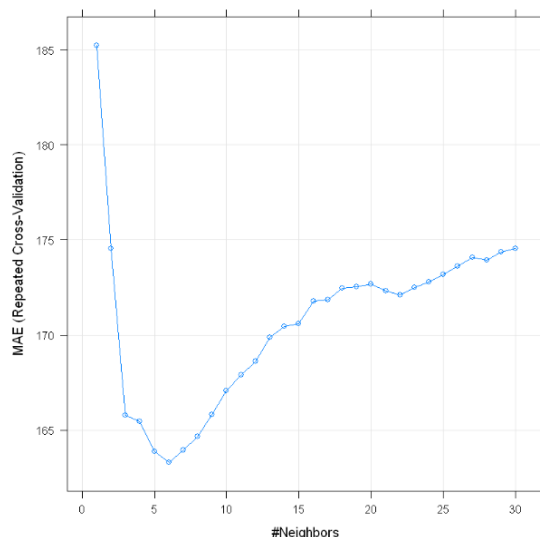
و چون در این جا معیار ما برای انتخاب بهترین k کم کردن MAE بود

k=6 انتخاب شده است که همان طور که میبینید کمترین RMSE

را نیز دارد و Rsquared آن نیز بعد از k=3 ، بالاترین مقدار است و

در کل k=6 عدد خوبی برای این مدل و این داده ها میباشد.

در این جا میتوانید نمودار مقدار MAE بر حسب k را نیز مشاهده کنید.



حال زمان بررسی عملکرد مدل روی داده جدید است که مدل تا به این جا ندیده است.

برای این کار از داده test استفاده میکنیم و عملکرد آن به شکل زیر است.

```
accuracy(predict(m1,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	5.843856	277.2888	182.0896	-2107.498	2131.768

برای بهتر کردن این مدل میتوان به جای استفاده از میانگین ساده از میانگین وزن دار استفاده کرد و تاثیر فاصله نقاط را نیز در پیشبینی در نظر گرفت.

موفق به پیدا کردن پکیجی که در R این کار را به خوبی انجام داده باشد نشدم (چند مورد وجود داشت که نتوانستم نتیجه مطلوب بگیرم)

```
weighted_average.knn <- function(x_train,x_test,y_train,k){
  df = FNN::knnx.index(data = x_train , query=x_test , k=k)
  values = data.frame(1:nrow(x_test))
  for(var in 1:k)
  {
    values = cbind(values,y_train[df[,var]])
  }
  values = values[-c(1)]
  dist = FNN::knnx.dist(data = x_train , query=x_test , k=k)
  weights = 1/dist
  print(as.data.frame(rowSums(weights)))
  return (rowSums(values * weights)) / as.data.frame(rowSums(weights))
}
```

به همین دلیل خودم سعی کردم پیاده سازی آن را انجام دهم اما

این پیاده سازی نیز برای بعضی از نقاط مقدار بینهایت را پیشبینی

میکرد که نتوانستم متوجه شوم چه تغییری برای حل این مشکل

باید روی این تابع اعمال کنم.

۳. درخت های پیشگو

تمام پیش پردازش ها دقیقا مانند رگرسیون میباشند.

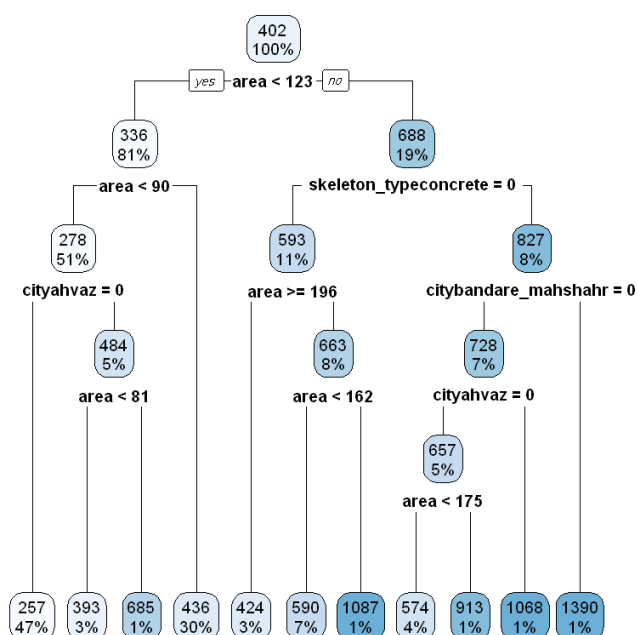
برای ساخت درخت از تابع `rpart` استفاده میکنیم و چون قصد پیشبینی برای این داده ها را داریم و نه رده بندی به جای اینکه پارامتر `method` آن را برابر با `class` قرار دهیم آن را برابر با `anova` قرار میدهیم.

۳,۱ درخت پیشگو بدون تنظیم کردن پارامتر ها

در مرحله اول یک مدل از درخت را بدون تنظیم کردن هیچکدام از پارامتر ها آموزش میدهیم و عملکرد آن را بررسی میکنیم.

```
reg.tree = rpart(formula = y_train~. , data = train, method = "anova")
```

درختی که در نتیجه این دستور به وجود می آید و قوانین آن را در زیر مشاهده میکنید



y_train	cover
257 when area < 90 & cityahvaz is 0	47%
393 when area < 81 & cityahvaz is 1	3%
424 when area >= 196 & skeleton_typeconcrete is 0	3%
436 when area is 90 to 123	30%
574 when area is 123 to 175 & skeleton_typeconcrete is 1 & cityahvaz is 0 & citybandare_mahshahr is 0	4%
590 when area is 123 to 162 & skeleton_typeconcrete is 0	7%
685 when area is 81 to 90 & cityahvaz is 1	1%
913 when area >= 175 & skeleton_typeconcrete is 1 & cityahvaz is 0 & citybandare_mahshahr is 0	1%
1068 when area >= 123 & skeleton_typeconcrete is 1 & cityahvaz is 1 & citybandare_mahshahr is 0	1%
1087 when area is 162 to 196 & skeleton_typeconcrete is 0	1%
1390 when area >= 123 & skeleton_typeconcrete is 1 & citybandare_mahshahr is 1	1%

دقت مدل روی داده های آموزشی و آزمایشی نیز به شکل زیر میباشد.

```
accuracy(predict(reg.tree,x_train),y_train)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	-0.00000000000004001206	196.6246	133.6232	-5980.619	5981.634

```
accuracy(predict(reg.tree,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	45.25621	257.2425	159.983	-1699.248	1729.426

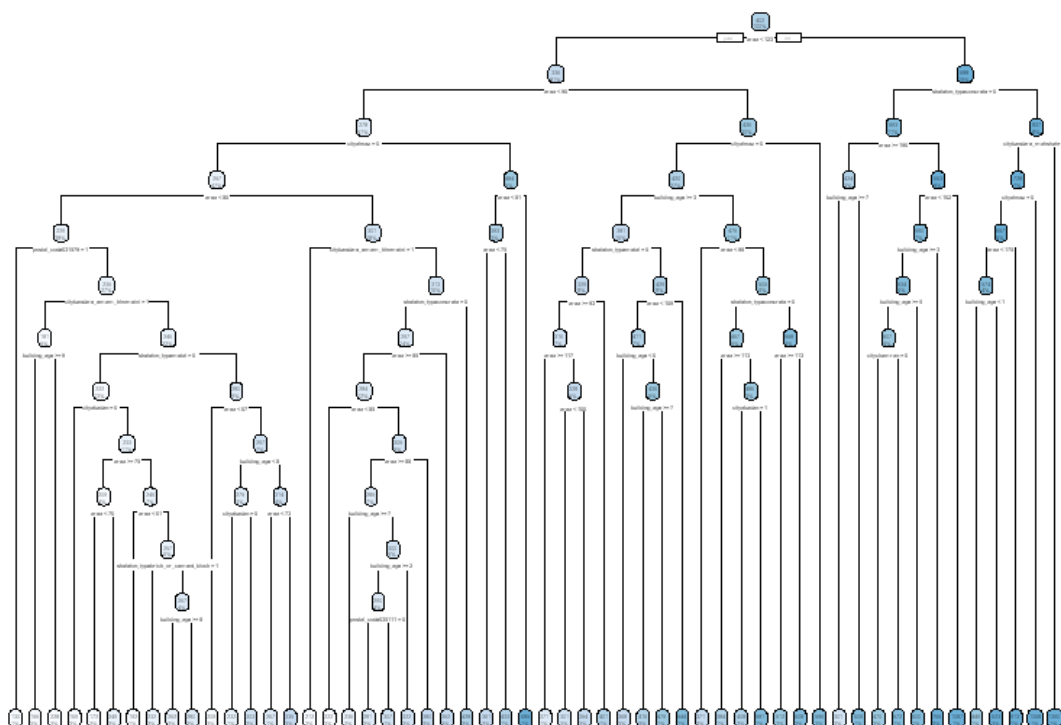
برای بهبود عملکرد این مدل ابتدا قصد داریم یک درخت با بیش ترین شاخه ممکن را بسازیم و سپس با هرس کردن این درخت **generalization** را در آن بالا ببریم تا روی داده های جدید هم عملکرد خوبی داشته باشد.

۳،۲ عمیق ترین درخت پیشگو

برای ساخت این درخت تابع **rpart** را با **cp** برابر با ۰ اجرا میکنیم که باعث میشود درخت برای اضافه کردن شاخه جدید جریمه نشود و این کار باعث میشود که عمیق ترین درخت ممکن ساخته شود.

```
deep.reg.tree = rpart(formula = y_train~. , data = train, method = "anova", cp=0)
```

درخت خروجی به شکل زیر است.



همان طور که مشاهده میکنید این درخت قواعد تصمیم بسیار زیادی دارد که باعث میشود بیش از حد روی داده آموزشی fit شود و از generalization کافی برای داشتن عملکرد خوب روی داده آزمایشی برخوردار نباشد.

```
accuracy(predict(deep.reg.tree,x_train),y_train)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	-0.000000000000002256542	176.3462	113.7177	-4872.698	4891.876

```
accuracy(predict(deep.reg.tree,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	42.81598	259.4333	151.7825	-1755.838	1785.949

حالا برای بالا بردن generalization و بیشتر کردن خطا روی داده های آموزشی میخواهیم درخت را هرس کنیم و برخی از این شاخه های اضافی را حذف کنیم. برای این کار به جدول cp های مختلف موجود برای این درخت مراجعه میکنیم که برخی از مقادیر آن را در زیر مشاهده میکنید.

در این جا یکبار مقدار cp را انتخاب میکنیم که کمترین

xerror را داشته باشد که در این جا در سطر ۱۴ قرار

دارد و از آن جا که خود این مقدار xerror ممکن است

دارای خطا باشد که مقدار آن در سطر xstd نوشته شده

است دامنه جابجایی xerror را برای این مقدار حساب

میکنیم و از آن جا که هدف ما generalization هست

میخواهیم تا حد امکان در این جدول به سمت بالا حرکت

کنیم و تعداد شاخه های موجود در درخت را کاهش دهیم

پس xerror + xstd را محاسبه میکنیم xerror ای را

پیدا میکنیم که از این مقدار کمتر باشد و از بقیه xerror

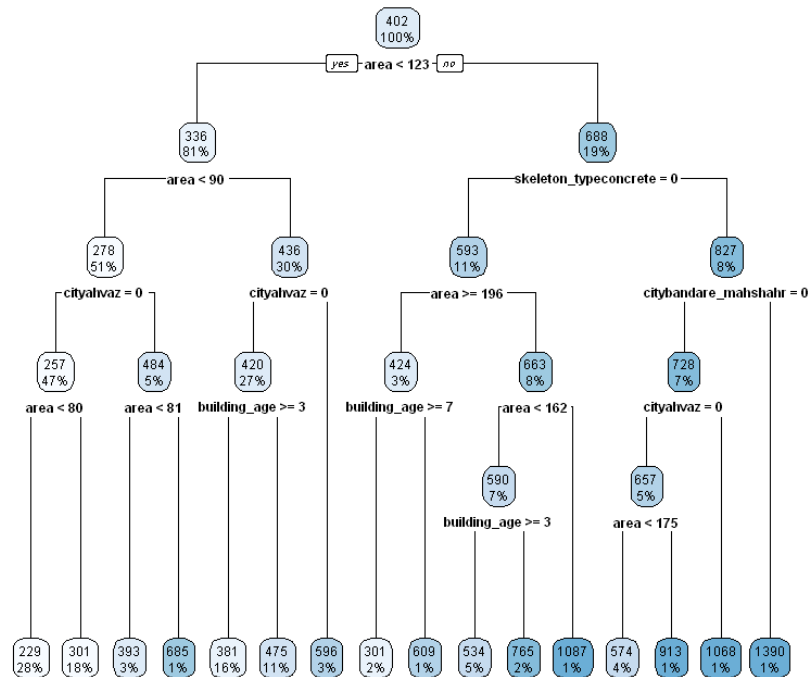
های کمتر از این مقدار بزرگتر باشد و cp مربوط به این

سطر را به عنوان cp در درخت با بهترین هرس انتخاب میکنیم.

	CP	nsplit	rel error	xerror	xstd
1	0.240793018	0	1.00000	1.00425	0.13557
2	0.059847642	1	0.75921	0.78844	0.11014
3	0.042885368	2	0.69936	0.73839	0.11272
4	0.027998852	4	0.61359	0.75817	0.11512
5	0.023601447	5	0.58559	0.74939	0.11563
6	0.020250407	7	0.53839	0.75634	0.11575
7	0.014443176	8	0.51814	0.76050	0.11585
8	0.010966263	9	0.50369	0.75430	0.11458
9	0.009908470	10	0.49273	0.74085	0.11506
10	0.009469504	11	0.48282	0.73929	0.11506
11	0.008377270	12	0.47335	0.73854	0.10821
12	0.007361642	13	0.46497	0.74018	0.10872
13	0.007339947	14	0.45761	0.73618	0.10872
14	0.005140450	15	0.45027	0.72909	0.10872
15	0.004330327	16	0.44513	0.73782	0.11025
16	0.003150177	17	0.44080	0.73954	0.11029
17	0.003088965	18	0.43765	0.73763	0.10937
18	0.002805694	20	0.43147	0.73763	0.10937
19	0.002200956	21	0.42867	0.73532	0.10907
20	0.002152135	22	0.42646	0.74394	0.11085
21	0.002138819	24	0.42216	0.74735	0.11103
22	0.001631756	27	0.41574	0.74260	0.10713
23	0.001438326	30	0.41084	0.74256	0.10695
24	0.001183281	31	0.40940	0.74725	0.10695
25	0.001140911	32	0.40822	0.74503	0.10691
26	0.001116409	33	0.40708	0.74533	0.10691
27	0.000977614	34	0.40596	0.74958	0.10695
28	0.000964405	35	0.40499	0.74801	0.10696
29	0.000883467	36	0.40402	0.74817	0.10696
30	0.000725336	37	0.40314	0.74653	0.10693
31	0.000650813	38	0.40241	0.74742	0.10695
32	0.000559276	39	0.40176	0.74772	0.10695
33	0.000524747	41	0.40064	0.74608	0.10659
34	0.000510447	42	0.40012	0.74606	0.10659
35	0.000508986	43	0.39961	0.74606	0.10659

۳,۳ عمیق ترین درخت پیشگو (هرس شده با کمترین xerror)

```
pruned.deep.reg.tree = prune(deep.reg.tree,cp=deep.reg.tree$cp[which.min(deep.reg.tree$cp[,'xerror']),'CP'])
```



y_train	.1	.2	.3	.4	.5	.6	.7	.81	.2	.3	.4	.5	.6	.7	.8	cover
229	when area < 80																	28%
301	when area is 80 to 90																	18%
301	when area >= 196								...	0	& building_age >= 7							2%
381	when area is 90 to 123										& cityahvaz is ...							16%
393	when area < 81										& cityahvaz is ...							3%
475	when area is 90 to 123										& cityahvaz is ...							11%
534	when area is 123 to 162									...	0	& building_age >= 3						5%
574	when area is 123 to 175																	4%
596	when area is 90 to 123																	3%
609	when area >= 196									...	0	& building_age < 7						1%
685	when area is 81 to 90																	1%
765	when area is 123 to 162																	2%
913	when area >= 175																	1%
1068	when area >= 123																	1%
1087	when area is 162 to 196									...	0							1%
1390	when area >= 123																	1%

```
accuracy(predict(pruned.deep.reg.tree,x_train),y_train)
```

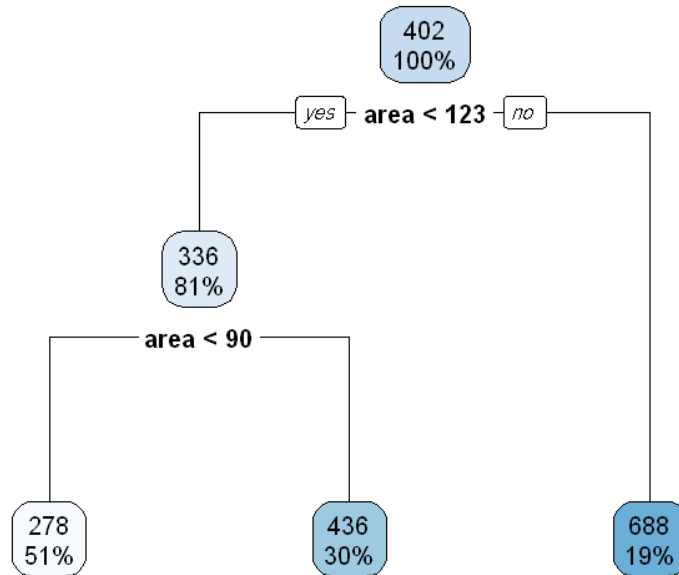
	ME	RMSE	MAE	MPE	MAPE
Test set	0.000000000000000792294	187.9625	126.7318	-6084.495	6105.274

```
accuracy(predict(pruned.deep.reg.tree,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	35.27346	252.3743	153.564	-1519.289	1547.558

۳,۴ عمیق ترین درخت پیشگو (بهترین هرس)

```
best.pruned.deep.reg.tree = prune(deep.reg.tree,cp=0.059847642)
```



y_train	cover
278 when area < 90	51%
436 when area is 90 to 123	30%
688 when area >= 123	19%

```
accuracy(predict(best.pruned.deep.reg.tree,x_train),y_train)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	-0.0000000000000001883166	234.2528	152.3075	-6204.969	6226.402

```
accuracy(predict(best.pruned.deep.reg.tree,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	49.27933	270.3546	167.2234	-1842.7	1872.237

همان طور که مشاهده میکنید این کار باعث بالا رفتن خطا روی داده train شد و وابستگی مدل را به داده train کم کرد و generalization مدل را بالا برد.

۳،۵ جنگل تصادفی

در این جا برای ساختن جنگل از ۱۰۰ درخت استفاده میکنیم.

```
random.forest = randomForest(x=x_train, y=y_train, ntree=100, importance=T, nodesize=4)
```

```
accuracy(predict(random.forest,x_train),y_train)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	5.204199	86.16244	55.99443	-2299.6	2308.987

```
accuracy(predict(random.forest,x_test),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	44.75623	250.0397	148.2198	-1708.236	1735.545

در زیر نیز تاثیر هر یک از متغیر ها را در این جنگل مشاهده میکنید.

random.forest

area
postal_code613865
building_age
cityahvaz
postal_code615568
postal_code635115
skeleton_typeconcrete
postal_code635160
postal_code635917
postal_code615573
citybandare_mahshahr
postal_code635171
cityhandijan
postal_code635165
postal_code635162
skeleton_typedmetal
postal_code635174
postal_code631591
postal_code615583
citybandare_emam_khomeini
citychamran
skeleton_typedmetal_concrete
skeleton_typebrick_or_cement_block
postal_code635164
postal_code631683
cityabadan
skeleton_typednone
postal_code616464
postal_code631579
postal_code631481



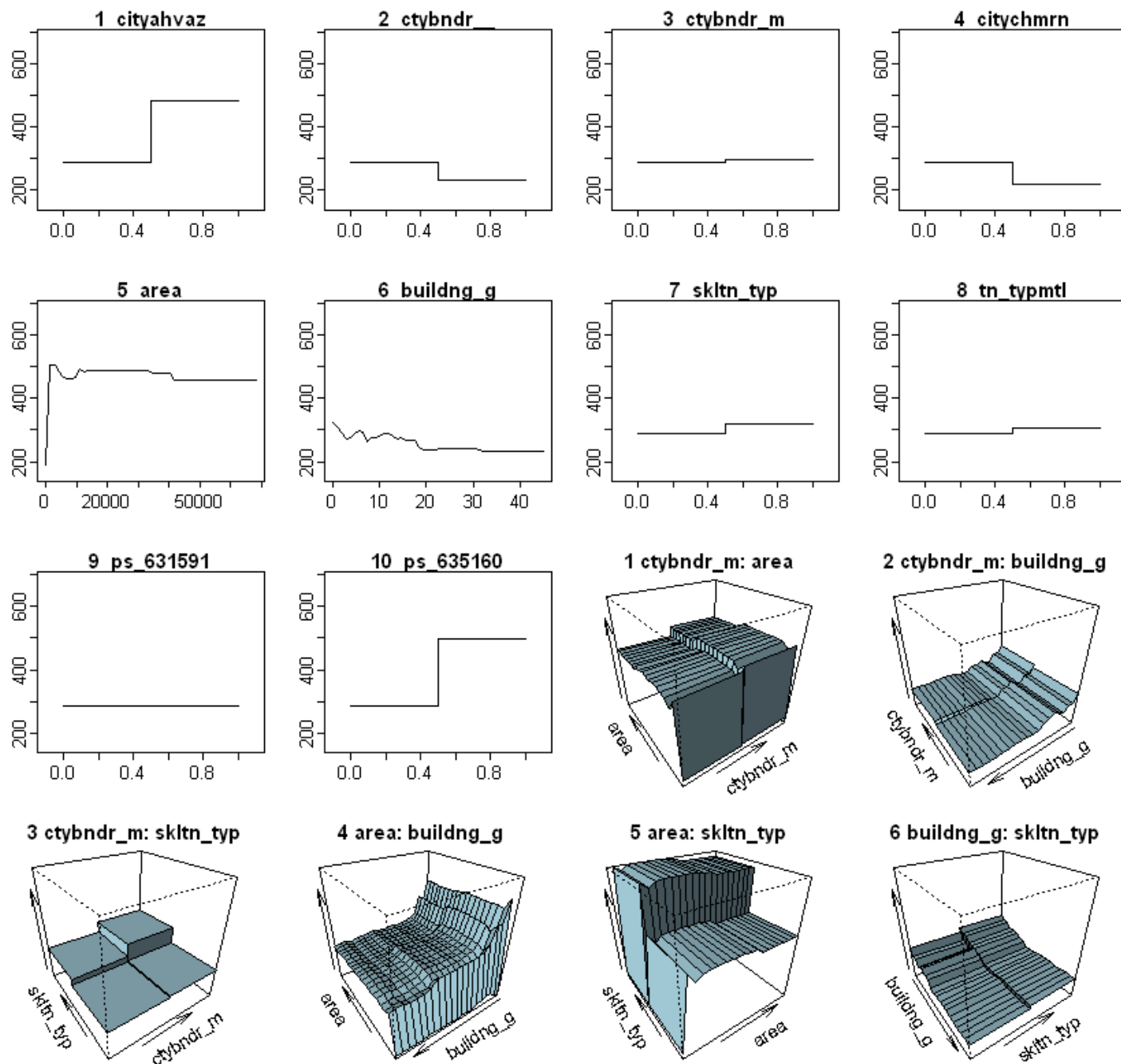
random.forest

area
postal_code635514
skeleton_typeconcrete
postal_code631591
postal_code635113
postal_code631579
cityahvaz
citybandare_mahshahr
citybandare_emam_khomeini
postal_code631794
building_age
postal_code631491
postal_code631686
postal_code631773
citychamran
postal_code631481
skeleton_typedmetal
skeleton_typednone
citydezfool
postal_code635160
postal_code631473
cityabadan
postal_code631784
postal_code635198
postal_code635165
postal_code635171
postal_code635412
postal_code631875
postal_code635513
postal_code641665



در نمودار صفحه بعد نیز میتوان به شکل دقیق تر تصمیم های این جنگل را بررسی کرد.

```
randomForest(x=x_train, y=y_train, ntree=100, nodesize=4, importance=T)
```



۳,۶ درخت تقویت شده

برای ساخت درخت تقویت شده نیز از کتابخانه gbm و روش ۱۰-fold cross validation روی داده آموزشی استفاده میکنیم و در نهایت عملکرد آن را روی داده آزمایشی بررسی میکنیم.

```
set.seed(101)
boosted.tree = gbm(y_train~. , data=train, cv.folds = 10)
```

در این جا نیز برخی از مهم ترین ویژگی های تاثیر گذار در این مدل و سطح اهمیتشان را مشاهده میکنید.

	var	rel.inf
area	area	72.9519090
cityahvaz	cityahvaz	13.1132179
skeleton_typeconcrete	skeleton_typeconcrete	5.2239214
building_age	building_age	4.2112154
citybandare_mahshahr	citybandare_mahshahr	2.5276543
citychamran	citychamran	1.2785502
cityabadan	cityabadan	0.2850227
citybandare_emam_khomeini	citybandare_emam_khomeini	0.2734410
skeleton_typenone	skeleton_typenone	0.1350681
citybehbahan	citybehbahan	0.0000000

عملکرد آن نیز به شکل زیر میباشد

```
accuracy(predict(boosted.tree,x_train),y_train)
```

Using 100 trees...

	ME	RMSE	MAE	MPE	MAPE
Test set	2.448334	205.4627	134.6677	-6076.502	6097.818

```
accuracy(predict(boosted.tree,x_test),y_test)
```

Using 100 trees...

	ME	RMSE	MAE	MPE	MAPE
Test set	39.19129	244.9458	148.4321	-1824.321	1850.706

بررسی :

آزمایشی (Test)				
MAPE	MPE	MAE	RMSE	ME
1729.42	-1699.24	159.98	257.24	45.25
1785.94	-1755.83	151.78	259.43	42.81
1547.55	-1519.28	153.56	252.37	35.27
1872.237	-1842.7	167.22	270.35	49.27
1735.54	-1708.23	148.21	250.03	44.75
1850.7	-1824.32	148.43	244.94	39.19

آموزشی (Train)				
MAPE	MPE	MAE	RMSE	ME
5981.63	-5960.61	133.62	196.62	0
4891.87	-4872.69	113.71	176.34	0
6105.27	-6084.49	126.73	187.96	0
6226.4	-6204.96	152.3	234.25	0
2308.98	-2299.6	55.99	86.16	5.2
6097.81	-6076.5	134.66	205.46	2.44

درخت بدون تنظیم cp
عمیق ترین درخت
هوس بر اساس xerror
بهترین هوس
جنگل تصادفی
درخت تقویت شده

در بین مدل های مبتنی بر درخت دو مدل جنگل تصادفی و درخت تقویت شده مشترکاً توانستند بهترین عملکرد را روی داده های جدید از خود نشان بدهند و به عنوان بهترین مدل های این بخش انتخاب میشوند.

۴. شبکه های عصبی مصنوعی

شبکه های عصبی مصنوعی مدل دیگری است که قصد داریم عملکرد آن را روی داده هایمان بررسی کنیم.

از آن جا آموزش شبکه عصبی با تعداد متغیر های پیشگو نیار به یک شبکه عصبی پیچیده با تعداد نرون های زیاد دارد و این عامل باعث میشود که سرعت یادگیری در شبکه به شدت افت کند قصد داریم که تعداد متغیر های پیشگو را کاهش دهیم و تنها از تعداد انگشت شماری از متغیر ها برای پیشگویی استفاده کنیم اما از آن جا که انتخاب کردن این متغیر ها از داده های اصلی ممکن است باعث میشود بخش زیادی از اطلاعات موجود از بین برود و واریانس کمی از داده ها در متغیر های انتخاب شده وجود داشته باشد، ابتدا قصد داریم که با استفاده از روش PCA مهم ترین ویژگی های داده را به شکل ترکیبی خطی از ویژگی های اولیه داده به دست بیاوریم ، به این شکل با انتخاب کردن تعداد انگشت شماری از این ویژگی ها میتوانیم بخش بزرگی از واریانس داده ها را شناسایی کنیم و از آن برای پیشگویی در مدل استفاده کنیم.

برای این کار لازم است که PCA را روی داده های آموزشی اجرا کنیم و مهم ترین بردار های ویژگی را به دست بیاوریم و مدل ها را روی این ویژگی های جدید آموزش دهیم و سپس برای آزمایش مدل داده های آزمایشی را با استفاده از ترکیب های خطی بهینه یافته شده از داده های آموزشی تبدیل کنیم و سپس از آن برای آزمایش داده ها استفاده کنیم.

برای انجام PCA روی داده های آموزشی به شکل زیر عمل میکنیم.

```
pcs = prcomp(x_train)
```

با اجرای دستور summary روی pcs میتوانیم تاسیر هر یک از بردار ها جدید را بررسی کنیم. بخشی از خروجی این دستور را در زیر مشاهده میکنید.

```
summary(pcs)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	0.6277	0.5811	0.47103	0.37583	0.3262	0.28818	0.27791
Proportion of Variance	0.1622	0.1390	0.09133	0.05814	0.0438	0.03419	0.03179
Cumulative Proportion	0.1622	0.3011	0.39247	0.45061	0.4944	0.52859	0.56039
	PC8	PC9	PC10	PC11	PC12	PC13	PC14
Standard deviation	0.24415	0.20925	0.18287	0.16914	0.16448	0.16150	0.14714
Proportion of Variance	0.02454	0.01802	0.01377	0.01178	0.01114	0.01074	0.00891
Cumulative Proportion	0.58492	0.60295	0.61671	0.62849	0.63962	0.65036	0.65927
	PC15	PC16	PC17	PC18	PC19	PC20	PC21
Standard deviation	0.13895	0.1314	0.12283	0.12124	0.1187	0.11829	0.11755
Proportion of Variance	0.00795	0.0071	0.00621	0.00605	0.0058	0.00576	0.00569
Cumulative Proportion	0.66722	0.6743	0.68053	0.68658	0.6924	0.69814	0.70383
	PC22	PC23	PC24	PC25	PC26	PC27	PC28
Standard deviation	0.10725	0.10652	0.1057	0.10496	0.10447	0.10086	0.09893
Proportion of Variance	0.00474	0.00467	0.0046	0.00453	0.00449	0.00419	0.00403
Cumulative Proportion	0.70857	0.71324	0.7178	0.72237	0.72686	0.73105	0.73508
	PC29	PC30	PC31	PC32	PC33	PC34	PC35
Standard deviation	0.09878	0.09864	0.09794	0.09673	0.09549	0.09450	0.09230
Proportion of Variance	0.00402	0.00401	0.00395	0.00385	0.00375	0.00368	0.00351
Cumulative Proportion	0.73910	0.74310	0.74705	0.75090	0.75466	0.75833	0.76184
	PC36	PC37	PC38	PC39	PC40	PC41	PC42
Standard deviation	0.09031	0.09016	0.0895	0.08871	0.08845	0.08593	0.08309
Proportion of Variance	0.00336	0.00335	0.0033	0.00324	0.00322	0.00304	0.00284
Cumulative Proportion	0.76520	0.76854	0.7718	0.77508	0.77830	0.78134	0.78418

در این جا واریانس پوشش داده شده توسط هر بردار و واریانس تجمعی پوشش داده شده تا آن بردار را مشاهده میکنید.

همان طور که مشاهده میکنیم بردار اول به تنهایی ۱۶ درصد از کل واریانس داده ها را پوشش میدهد.

در این بخش من چندین مدل مختلف با تعداد متغیر های مختلف را بررسی کردم که این توضیح را در کنار هر مدل ذکر خواهم کرد.

مدلی که با ۱۱۲ متغیر ساخته و ۱۰۰ نوروں ساختم که تقریباً ۹۰ درصد واریانس داده ها را پوشش میداد بعد از ۱۵ ساعت موفق به همگرا شدن نشد.

مدل های دیگری نیز با تعداد کمتری متغیر ساختم که آن ها نیز همگرا نشدند.

در نهایت دو دسته مدل ساختم که برخی از آن ها از ۹ بردار ویژگی استفاده کردند و برخی دیگر از ۴ بردار ویژگی استفاده کردند.

در این جا روش تبدیل داده های آموزشی اولیه به بردار های جدید را نشان میدهیم.

```
x_train.pca = pcs$x[,1:4]
```

```
head(x_train.pca)
```

	PC1	PC2	PC3	PC4
649	-1.0523432	-0.3145527	0.01584797	-0.32758941
639	-1.0557971	-0.3136770	0.02383819	-0.32541412
241	0.4912139	0.5772892	-0.74522645	-0.07406887
662	0.1159690	-0.9360292	0.19398902	-0.81532686
727	0.2333827	-0.2921811	-1.21826290	0.40104081
276	0.3985958	0.5518737	-0.06038210	-0.08441918

```
train.pca = as.data.frame(cbind(y_train,x_train.pca))
```

و سپس ستون هدف را به داده های آموزشی وصل میکنیم.

```
dim(train.pca)
```

```
614 5
```

```
names(train.pca)
```

```
'y_train' 'PC1' 'PC2' 'PC3' 'PC4'
```

حالا زمان آن رسیده که داده های آزمایشی را نیز به همین فضای حالت ببریم تا بتوانیم آن ها را برای ارزیابی مدل آماده کنیم.

برای این کار ابتدا باید center را در این بردار ها یکی کنیم و سپس به وسیله چرخشی که برای ساخت هر کدام از بردار ها لازم است آن ها را به بردار های جدید

تبدیل کنیم.

```
x_test.centered = scale(x_test, center= pcs$center)
x_test.pca = x_test.centered %%% pcs$rotation
x_test.pca = as.data.frame(x_test.pca[,1:4])
```

```
head(x_test.pca)
```

	PC1	PC2	PC3	PC4
1	0.4466654	-0.6028422	-0.71302714	0.7662643
7	-1.1273885	-0.3270660	0.04705929	1.1931403
9	1.2744104	-1.6243480	0.54857499	0.1351745
17	0.5806250	-0.3371365	-0.69636422	0.6781684
20	1.1375791	-1.7468060	0.02827934	0.8414800
23	1.2800244	-1.6435276	0.46315910	0.1964228

```
dim(x_test.pca)
```

```
154 4
```

از تابع فعالسازی logistic و tanh در این مدل ها استفاده میکنیم

و از بهینه ساز resilient backpropagation چون سرعت آن از backpropagation کلاسیک بهتر است و میتوان learning rate را بر صورت پویا و با توجه به مسئله تغییر داد و در شرایطی مختلف learning rate را کم و زیاد میکند.

و برای محاسبه خطا نیز از جمع مربعات خطاها SSE استفاده میکنیم.

مدل ۱:۹ ویژگی - ۱ لایه مخفی با ۹ نرون

```
nn = neuralnet(formula = y_train ~ ., data = train.pca, hidden = c(9), threshold = 10,
  stepmax = 1e+08 ,algorithm = "rprop+",err.fct = "sse", learningrate.factor = list(minus = .5 ,plus = 1.5),
  act.fct = "logistic", linear.output = TRUE, lifesign = 'full', lifesign.step=100)
```

```
accuracy(as.numeric(compute(nn,train.pca[-c(1)])$net.result),train.pca[,1])
```

	ME	RMSE	MAE	MPE	MAPE
Test set	0.0001031294	218.77	144.8914	-5176.347	5197.467

```
accuracy(as.numeric(compute(nn,x_test.pca)$net.result),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	372.071	696.6781	538.4502	1794.504	1879.446

بیش برآزش روی داده های آموزشی داشته است پس تعداد نرون ها را کمتر میکنیم.

مدل ۲:۹ ویژگی - ۱ لایه مخفی با ۷ نرون

```
nn2 = neuralnet(formula = y_train ~ ., data = train.pca, hidden = c(7), threshold = 1,
  stepmax = 1e+04 ,algorithm = "rprop+",err.fct = "sse", learningrate.factor = list(minus = .5 ,plus = 1.5),
  act.fct = "logistic", linear.output = TRUE, lifesign = 'full', lifesign.step=100)
```

```
accuracy(as.numeric(compute(nn2,train.pca[-c(1)])$net.result),train.pca[,1])
```

	ME	RMSE	MAE	MPE	MAPE
Test set	-0.0004841371	277.0954	190.7664	-5182.401	5205.557

```
accuracy(as.numeric(compute(nn2,x_test.pca)$net.result),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	76.31488	319.4234	216.3862	-2695.067	2737.178

بیش برآزش برطرف شد حالا بررسی میکنیم که آیا با تعداد متغیر کمتر هم میتوانیم پیش بینی خوبی روی داده ها داشته باشیم یا خیر

مدل ۳: ۴ ویژگی - ۱ لایه مخفی با ۷ نرون

```
nn2 = neuralnet(formula = y_train ~ ., data = train.pca, hidden = c(7), threshold = 0.1,
  stepmax = 1e+08 ,algorithm = "rprop+",err.fct = "sse", learningrate.factor = list(minus = .7 ,plus = 1.4),
  act.fct = "logistic", linear.output = TRUE, lifesign = 'full', lifesign.step=100)
```

```
accuracy(as.numeric(compute(nn2,train.pca[-c(1)])$net.result),train.pca[,1])
```

	ME	RMSE	MAE	MPE	MAPE
Test set	-0.000003528016	275.0204	188.7649	-5324.152	5347.422

```
accuracy(as.numeric(compute(nn2,x_test.pca)$net.result),y_test)
```

	ME	RMSE	MAE	MPE	MAPE
Test set	79.53797	319.2327	216.0056	-1366.788	1410.741

عملکرد مدل بهبود یافت.

میتوان این کار را با تعداد ویژگی های مختلف و تعداد نرون های مختلف در لایه های پنهان مجددا انجام داد و نتایج مدل را بهبود داد ، اما به دلیل زیاد شدن محاسبات این آموزش ها نیاز به سیستم های قدرتمند تری دارد و برخی از مدل های بنده بعد از گذشت زمان طولانی همگرا نشدند و به همین دلیل آن ها را در این گزارش ذکر نکردم.

بررسی :

آزمایشی (Test)				
MAPE	MPE	MAE	RMSE	ME
1876.44	1794.5	538.45	696.67	372.07
2737.17	-2695.06	216.38	319.42	76.31
1410.74	-1366.78	216	319.23	79.53

آموزشی (Train)				
MAPE	MPE	MAE	RMSE	ME
5197.46	-5176.34	144.89	218.77	0
5205.55	-5182.4	190.76	277.09	0
5347.42	-5324.15	188.76	275.02	0

۹ ویژگی ۹ نرون
۹ ویژگی ۷ نرون
۴ ویژگی ۷ نرون

بهترین مدل شبکه عصبی ما مدل شماره ۳ میباشد چون تعداد ویژگی های کمتری نیاز دارد و عملکرد آن نیز نسبت به بقیه مدل های شبکه عصبی روی داده جدید بهتر بود.

مقایسه عملکرد برترین مدل های هر دسته با یکدیگر

در جدول زیر شاخص های مختلف دقت را برای بهترین مدل های هر کدام از این ۴ دسته مشاهده میکنید.

آزمایشی (Test)					آموزشی (Train)					رگرسیون قدم به قدم پیشرو 6- نزدیک ترین همسایه جنگل تصادفی درخت تقویت شده مدل سوم شبکه عصبی
MAPE	MPE	MAE	RMSE	ME	MAPE	MPE	MAE	RMSE	ME	
10856.77	-10821.6	189.67	296.67	63.64	2572.02	-2555.9	102.67	149.09	0	
4840.35	-4819.53	141.81	215.86	3.57	2131.76	-2107.49	182.08	277.28	5.84	
1735.54	-1708.23	148.21	250.03	44.75	2308.98	-2299.6	55.99	86.16	5.2	
1850.7	-1824.32	148.43	244.94	39.19	6097.81	-6076.5	134.66	205.46	2.44	
1410.74	-1366.78	216	319.23	79.53	5347.42	-5324.15	188.76	275.02	0	

به نظر من مهم ترین این شاخص ها MAE و RMSE میباشد و به همین دلیل مدل ۶-نزدیک ترین همسایه را به عنوان بهترین مدل انتخاب میکنم.

در این گزارش ابتدا سعی کردیم تا داده های مربوط به خرید و فروش مسکن در استان خوزستان را بررسی کنیم ، به عوامل تاثیر گذار روی قیمت مسکن پرداختیم و سعی کردیم تاثیر هر یک را بر شهرستان های مختلف استان خوزستان آشکار کنیم تا برای شخصی که قصد دارد برای اولین بار با این داده ها کار کند دید کافی را ایجاد کنیم تا بتواند دقیق تر این داده ها را مورد بررسی قرار دهد ، قطعاً نکات مهم دیگری نیز در این داده ها وجود دارد که بنده نتوانستم به خوبی آن ها را نمایش دهم.

سپس در بخش دوم با سعی کردیم با استفاده از مدل های مختلف قیمت خانه ها را با توجه به سایر ویژگی هایشان پیشبینی کنیم که نتایج را مشاهده کردید.

این داده ها مشکلاتی نیز داشتند که میتوان به کم بودن تعداد داده ها در شهرستان های مختلف و کم بودن پراکندگی داده ها در برخی از ستون ها اشاره کرد که باعث شد اطلاعات خیلی زیادی برای نمایش و بررسی موجود نباشد.