گزارش دیتاست ۲ تمرین سری ۲- یادگیری ماشین

عرفان کرمی – ۹۸۲۲۲۰۷۹

در این بخش از تمرین سری دوم هدف پیاده سازی الگوریتم رگرسیون خطی و مقایسه نتیجه آموزش آن بر روی دیتاست داده شده با نتیجه مدل های پیاده سازی شده در پکیج های آماده مانند $sklean$ است.

دیتاست مجموعه ای از داده ها درباره املاک است و هدف تخمین قیمت های آنهاست. در شکل زیر اطلاعات کلی از دیتاست آمده است:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 268850 entries, 0 to 268849
Data columns (total 49 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   regio1                     268850 non-null  object
 1   serviceCharge              261941 non-null  float64
 2   heatingType                223994 non-null  object
 3   telekomTvOffer             236231 non-null  object
 4   telekomHybridUploadSpeed   45020 non-null   float64
 5   newlyConst                 268850 non-null  bool
 6   balcony                    268850 non-null  bool
 7   picturecount               268850 non-null  int64
 8   pricetrend                 267018 non-null  float64
 9   telekomUploadSpeed         235492 non-null  float64
 10  totalRent                  228333 non-null  float64
 11  yearConstructed            211805 non-null  float64
 12  scoutId                    268850 non-null  int64
 13  noParkSpaces               93052 non-null   float64
 14  firingTypes                211886 non-null  object
 15  hasKitchen                 268850 non-null  bool
 16  geo_bln                    268850 non-null  object
 17  cellar                     268850 non-null  bool
 18  yearConstructedRange       211805 non-null  float64
 19  baseRent                   268850 non-null  float64
 20  houseNumber                197832 non-null  object
 21  livingSpace                268850 non-null  float64
 22  geo_krs                    268850 non-null  object
 23  condition                  200361 non-null  object
 24  interiorQual               156185 non-null  object
 25  petsAllowed                154277 non-null  object
 26  street                     268850 non-null  object
 27  streetPlain                197837 non-null  object
 28  lift                       268850 non-null  bool
 29  baseRentRange              268850 non-null  int64
 30  typeOfFlat                 232236 non-null  object
 31  geo_plz                    268850 non-null  int64
 32  noRooms                    268850 non-null  float64
 33  thermalChar                162344 non-null  float64
 34  floor                      217541 non-null  float64
 35  numberOfFloors             171118 non-null  float64
 36  noRoomsRange               268850 non-null  int64
 37  garden                     268850 non-null  bool
 38  livingSpaceRange           268850 non-null  int64
 39  regio2                     268850 non-null  object
 40  regio3                     268850 non-null  object
 41  description                249103 non-null  object
 42  facilities                 215926 non-null  object
 43  heatingCosts               85518 non-null   float64
 44  energyEfficiencyClass      77787 non-null   object
 45  lastRefurbish              80711 non-null   float64
 46  electricityBasePrice       46846 non-null   float64
 47  electricityKwhPrice        46846 non-null   float64
 48  date                       268850 non-null  object
dtypes: bool(6), float64(18), int64(6), object(19)
memory usage: 89.7+ MB
```

در این بخش از تمارین ما تنها از سه فیچر $serviceCharge, heatingType, telekomUploadSpeed$ استفاده خواهیم کرد و هدف هم پیش بینی $totalRent$ خواهد بود.

- ابتدا باید دقت کنیم که در ستونی که میخواهیم آنرا پیشبینی کنیم مقادیر ناموجودی وجود دارد ما این مقادیر را حذف میکنیم و در نتیجه این عمل ۲۶۸۸۴۹ نمونه داده ما به ۲۲۸۳۳۳ داده تقلیل می یابند.
- در مرحله بعد برای فیچر ها مقادیر ناموجود را با استفاده از جایگزین کردن با مد داده ها مدیریت میکنیم این عملیات را با استفاده از $SimpleImputer$ که در پکیج سایکیت لرن پیاده سازی شده است انجام میدهیم.
- سپس فیچر کتگوریکالی که وجود دارد$(heatingType)$ را با استفاده از $one\ hot\ encoding$ به مقادیر عددی تبدیل میکنیم.
- داده های پرت را هم با استفاده از محاسبه $zscore$ شناسایی و از دیتاست حذف میکنیم. شرطی که برای حذف نمونه داده استفاده میکنیم به صورت زیر است:

$$if\ t \in dataset\ and\ |zscore(t)| \geq 3 \rightarrow t\ is\ outlier$$

با استفاده از این متد تعداد داده های باقی مانده به ۲۲۸۱۲۵ تقلیل می یابد.

- در مرحله بعد داده های تست و ترین را با استفاده از $train\_test\_split$ از هم جدا میکنیم
- در مرحله نهایی آماده سازی داده ها با استفاده از $StandardScaler$ بر روی دو فیچر عددی $serviceSharge, telekomUploadSpedd$ تغییر مقیاس را انجام میدهیم.

الگوریتمی که خودمان نوشته ایم را بر روی ای داده ها تست میکنیم و به نتایج زیر میرسیم:

$MSE\ on\ tran\ data$: `24334680.01762735`
$MSE\ on\ test\ data$: `3628373838.197944`

:Model trained Coefficients

```
[[783.18398146],
 [373.16230277],
 [  4.86713413],
 [ 34.54975101],
 [100.26810721],
 [ -6.52008139],
 [ 24.05504818],
 [261.46311662],
 [ 13.82019429],
 [158.75762303],
 [  2.5825481 ],
 [ 33.00363146],
 [ 93.96850311],
 [ 13.20496554],
 [  6.71141905],
 [ 52.4892397 ]]
```

حال الگوریتم رگرسیون خطی پیاده سازی شده در پکیج سایکیت لرن را بر روی داده ها ترین میکنیم. نتایج به صورت زیر هستند:

$MSE\ on\ tran\ data$: 24334540.52064295

$MSE\ on\ test\ data$: 3628346496.0389085

Model trained coefficients:

```
[3.72417140e+02, 4.70839506e+00, 7.58484533e+12, 7.58484533e+12,
       7.58484533e+12, 7.58484533e+12, 7.58484533e+12, 7.58484533e+12,
       7.58484533e+12, 7.58484533e+12, 7.58484533e+12, 7.58484533e+12,
       7.58484533e+12, 7.58484533e+12, 7.58484533e+12]
```

حال اگر نتایج را مقایسه کنیم میبینیم که عملکرد ها تقریبا بر روی داده های ترین و تست در هر دو مدل یکسان بوده اند، که البته مدل پکیج سایکیت لرن کمی بهتر بوده است(دلیل این امر هم این است که در کتابخانه سایکیت لرن رگرسیون خطی احتمالا با استفاده از معادله نرمال که یک روش تحلیلی است پیاده سازی شده است در نتیجه مقدار دقیق مینیمم سراسری را به دست می آورد- دقت به این امر مهم است که نرخ یادگیری را به عنوان ورودی دریافت نمیکند- درحالیکه در مدل ما با اینکه معادله نرمال هم در آن پیاده سازی شده است اما با استفاده از گرادیان کاهشی که یک روش تکرار شونده است و تابع هرینه را مینیمم میکند بر روی مدل آموزش دیده است.). البته باید دقت کنیم که با توجه به اینه در رگرسیون خطی ما یک بهینه گلوبال داریم و در صورت استفاده از متد مناسب حتما به آن خواهیم رسید. پس مقدار بالای MSE نشان دهنده کم برازش بر روی داده است و احتمالا از این ناشی میشود که یک مدل خطی نمیتواند برای فیت شدن بر روی داده ها مناسب. باشد.

البته مقایسه ضرایب آموزش دیده توسط دو مدل نشان میدهد که به جز دو ضریب اول این ضرایب بسیار متفاوت از هم بوده اند (ضریب اول در مدل ما ضریب مقدار بایاس است)و این نشان دهنده این میتواند باشد که احتمالا مهمترین فیچر ها همین دو فیچر هستند.

حال مدل های $Lasso, Ridge$ که در کتابخانه سایکیت لرن پیاده سازی شده اند را بر روی داده ها فیت میکنیم.

نتایج در صفحه بعد قابل مشاهده اند:

- Ridge:

$MSE\ on\ tran\ data$: 24334540.408087403

$MSE\ on\ test\ data$: 3628346275.6042223
Model trained coefficients:

```
[ 372.56912748,    4.7154186 ,  -58.45163401,   67.70890021,
       -101.75303253,  -36.06264722,  165.80059033,  -81.94273229,
        138.80006449, -112.55880602,  -70.28853469,   -2.97877502,
        108.30333094,  -61.31034583,   44.73362296]
```

- Lasso:

$MSE\ on\ tran\ data$: 24334560.546804726

$MSE\ on\ test\ data$: 3628352200.2974377
Model trained coefficients:

```
[372.75110254,   4.66541182, -35.13204585,  78.00316543,
       -77.51975823,  -0.        , 187.37895858, -57.54236596,
       151.65730478, -69.64761302, -42.11163546,  18.70538964,
         0.        ,  -0.        ,  41.5836656 ]
```

اگر دقت کنیم نتایج این دو مدل بر روی داده ها شبیه دو مدل قبلی هستند و این مجددا این نکته را تقویت میکنند که رگرسیون خطی و توسیع های آن مدل مناسبی برای برازش بر روی این داده ها نیستند اما نکته جالب اینجاست که ضرایب آموزش دیده توسط این دو مدل بسیارشبیه مدلی هستند که خودمان پیاده سازی کردیم، دلیل این امر هم این است که این دو الگوریتم هم از یک روند مبتنی بر مشتق برای مینیم سازی تابع هزینه استفاده میکنند(در ابتدا هم ابرپارامتر های نرخ یادگیری و تعداد تکرار ها را برای آنها باید مشخص کرد). البته برای Lasso سه تا از ضرایب کاملا صفر شده اند که این به دلیل ویژگی خاصی است که این مدل در صفر کردن اهمیت فیچر های کم اهمیت دارد.

** در این بخش خودمان الگوریتم رگرسیون خطی را پیاده سازی کردیم و آنرا بر روی تعدادی از فیچر های دیتاست املاک ترین کردیم و نتایج را با مدل های آماده مقایسه کردیم.