regressionProject

2025-06-03 13:46

Github-Link : https://github.com/pouniq/MLR_Rasht_housing

Tags: uni , regression , ML,DS

regressionProject

دیتا :

این مدل رگرسیونی به قیمت خانه در رشت می پردازه که متغیر های مستقل و وابسته در زیر تعریف شده:

✎ متغیر های وابسته:

$Y_1 = Rent$ قیمت اجاره بها
$Y_2 = Deposit$ قیمت ودیعه

✎ متغیرهای مستقل:

$X_1 = Meterage$ متراژ خونه
$X_2 = DOB$ سال ساخت خانه
$X_3 = Area$ منطقه جغرافیایی
$X_4 = NoB$ تعداد اتاق های خانه
$X_5 = FloorNum$ طبقه چندمه؟
$X_6 = District$ ناحیه جغرافیایی
$X_7 = Elevator$ آسانسور
$X_8 = Parking$ پارکینگ
$X_9 = Storage$ انباری
$X_{10} = Balcony$ بالک

# Rent	# Deposit	# DoB	# Meterage	# FloorNum	Area	District	# NoB	# Elevator	# Parking	# Storage	# Balcony	link
25000000	450000000	1404	130	2	گلسار	1	3	1	1	1	0	https
8000000	200000000	1391	75	2	آزادگان	4	2	1	1	1	0	https
3000000	400000000	1385	80	3	استقامت	2	2	0	1	1	0	https
6000000	150000000	1395	100	1	استقامت	2	1	0	1	1	0	https
7500000	320000000	1380	116	1	اسماعیل آباد	1	2	0	1	1	0	https
6000000	250000000	1401	85	2	اسماعیل آباد	1	2	0	0	0	0	https
7000000	200000000	1385	78	4	انصاری	1	2	1	1	1	0	https
4000000	500000000	1386	90	3	انصاری	1	2	0	1	1	0	https
5000000	350000000	1390	70	1	باهنر	4	2	1	0	1	0	https
5500000	350000000	1403	100	1	باهنر	4	2	1	1	1	0	https
2000000	400000000	1400	80	3	باهنر	4	2	1	1	1	0	https
7000000	400000000	1400	97	1	باهنر	4	2	1	1	0	1	https
35000000	3000000000	1398	230	5	بلوار سمیه	2	3	1	1	1	1	https

میتونین فایل دیتا و نوت بوک ها رو از گیتهاب من دانلود کنید.

تحلیل کاوشگرانه داده‌ها (EDA):

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

اول از همه کتابخانه های موردنیاز رو وارد می کنیم

- pandas: یک کتابخانه قدرتمند برای پردازش دیتا و ایجاد تغییر در دیتافریم
- matplotlib و seaborn : دو سری از کتابخانه برای تصویرساز داده ها استفاده میشه

در اولین گام دیتایی که داریم را میخونیم و با استفاده از pandas اون رو باز می کنیم.

```
filepath = 'https://docs.google.com/spreadsheets/d/1_B0reSRNs5_Se-5FPUtUB6AmjT511Ey2e6BjzrD4nY4/export?format=csv&gid=0'
df = pd.read_csv(filepath)
```

در گام دوم میاییم و دیتامون رو به بخش های کمی و کیفی تقسیمشون میکنیم:

```
quant_cols = ['Rent', 'Deposit', 'Meterage' ]
cat_cols = ['DoB', 'Area', 'District', 'Elevator', 'Parking', 'Storage', 'Balcony' ,  'NoB' , 'FloorNum']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125 entries, 0 to 124
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Rent      125 non-null    int64 
 1   Deposit   125 non-null    int64 
 2   DoB       125 non-null    int64 
 3   Meterage  125 non-null    int64 
 4   FloorNum  125 non-null    int64 
 5   Area      125 non-null    object
 6   District  125 non-null    int64 
 7   NoB       125 non-null    int64 
 8   Elevator  125 non-null    int64 
 9   Parking   125 non-null    int64 
 10  Storage   125 non-null    int64 
 11  Balcony   125 non-null    int64 
 12  link      120 non-null    object
dtypes: int64(11), object(2)
memory usage: 12.8+ KB
```

از طریق این کد میتونیم متوجه بشیم که آیا داده null در دیتافریم ما وجود داره یا نه و اینکه میتونیم بفهمیم که نوع دیتایی که داریم چه چیزی هست.

```
df.describe()
```

	Rent	Deposit	DoB	Meterage	FloorNum	District	NoB	Elevator	Parking	Storage	Balcony
count	1.250000e+02	1.250000e+02	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000	125.000000
mean	9.531200e+06	4.160000e+08	1393.304000	95.752000	2.000000	2.872000	2.016000	0.352000	0.744000	0.896000	0.232000
std	1.000286e+07	4.913362e+08	6.833532	39.621934	1.077632	1.367692	0.538576	0.479516	0.438178	0.306489	0.423808
min	1.000000e+06	5.000000e+07	1370.000000	40.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	4.000000e+06	2.000000e+08	1390.000000	75.000000	1.000000	2.000000	2.000000	0.000000	0.000000	1.000000	0.000000
50%	6.500000e+06	3.000000e+08	1394.000000	84.000000	2.000000	3.000000	2.000000	0.000000	1.000000	1.000000	0.000000
75%	1.200000e+07	4.000000e+08	1398.000000	105.000000	3.000000	4.000000	2.000000	1.000000	1.000000	1.000000	0.000000
max	7.000000e+07	4.000000e+09	1404.000000	300.000000	5.000000	5.000000	4.000000	1.000000	1.000000	1.000000	1.000000

این کد هم یک سری مشخصات آماری مورد نیاز به ما میده که همینطوری که مشخصه فقط برای دیتای نومریکال هستش و برای اینکه این جدول رو برای کتوگوریکال بگیریم از این کد استفاده می کنیم:

```
df.describe(include=['O'])
```

	Area	link
count	125	120
unique	48	115
top	گلسار	https://divar.ir/v/%DB%B3-%D8%AE%D9%88%D8%A7%D...
freq	18	2

```
for col in quant_cols:
    fig = plt.figure(figsize=(10,8))
    sns.histplot(x=df[col] , kde=True)
    plt.title(f'distribution for {col}')
    plt.show()
```

از این for loop هم استفاده کردم تا برای داده های کمی تک تک هیستوگرام درست کنه تا متوجه توزیع هر کدوم از این متغیرهای مستقل کمی بشم و همینکارو برای متغیرهای کیفی هم انجام دادم که به این شکل for loop طراحی میشه که countplot برای هر کدوم طراحی میکنه :

```
for col in cat_cols:
    fig = plt.figure(figsize = (12,8))
    sns.countplot(data = df , x = col)
```

و از for loop ها تو در تو استفاده شد تا متغیرهای کیفی رو در مقابل متغیرهای کمی قرار بدیم و تمام پلات های ممکنه رو رسم کنیم.

> **توجه**
>
> تمام پلات ها در فایل EDA قرار گرفته شده

پیش پردازش (PreProcessing):

از کتابخونه sci-kit learn هم یک module به نام preprocessing وجود داره که از اون MinMaxScaler رو فراخوان کردیم تا در این بخش استفاده کنیم بقیه هم قبلا کاربردشون بیان شد.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

از کد زیر استفاده کردم تا ستون هایی که نیاز نیستن رو پاک کنم از داده هام تا کار راحت تر جلو بره:
راجب ستون هایی که حذفم کردم بگم که قبل اینکه وارد این قدم به قدم بشم از قبل این رو برازش داده بودم و فهمدم که parking و storage تاثیری رو مدل ندارن:
توی این فایل تو بخش updates هم توضیح دادم و گفتم که دوباره برازشش ندم.

```
df = df.drop(['link' , 'Area' , 'Parking','Storage' ] , axis = 1)
```

یکی از کارهایی که تو preprocessing انجام دادم این بود که ستون سال تولید رو اول به چهاربخش تبدیل کردم به دهه هایی که وجود داره مثلا برای دهه ۸۰ هست ، برای دهه ۹۰ هست یا ۷۰ ، ...

```
df['DoB'] = df['DoB'].apply(lambda x: f"{int(x // 10 * 10)}s")
print(df['DoB'].value_counts())
```

```
DoB
1390s    70
1400s    28
1380s    24
1370s     3
```

تعداد هر کدوم از دهه ها به این شکل شد و دیدم که دهه ۷۰ تو دیتای کمی که دارم خوب خودش رو نشون نداده به خاطر همین اومدم و دسته ها رو کمتر کردم

```
def Decade(x):
    if x in ['1370s' ,'1380s' ]:
        return 'Before 1390'
    else:
        return x
df['DoB'] = df['DoB'].apply(Decade)
print(df['DoB'].value_counts())
```

```
DoB
1390s          70
1400s          28
Before 1390    27
```

همینطوری که می بینیم وقتی به سه بخش تبدیل شدند می تونیم بگیم که یکم بالانس تر شدن نسبت به قبل

تابع چه میکنه؟

این تابع میاد و x رو در نظر میگیره اگه بین دهه ۷۰ تا ۸۰ بود میگه قبل دهه ۹۰ هست و چیز دیگه بود همون رو بر میگردونه

در این بخش تبدیل به نشانگر می کنیم متغیرهای کتوگوریکالی که داریم رو ،
و دستور drop_first = True هم میگه که ستون اول رو بنداز و اون ستون رو با بقیه مقادیر بسنجیم اگه همشون صفر بودن میشه اون ستون

```
df = pd.get_dummies(df , columns=['FloorNum','District','NoB' , 'Elevator'
, 'DoB' , 'Balcony'] , drop_first=True)
```

این خروجی که بهمون میداد به صورت string بود و این خوب نیست (متوجه نشدم چرا string داد) ولی برای اینکه این مشکل حل بشه از این کد استفاده کردم که میاد string ها رو تبدیل به integer میکنه:

```
dummy_cols = [ 'FloorNum_2', 'FloorNum_3',
       'FloorNum_4', 'FloorNum_5', 'District_2', 'District_3',
'District_4',
       'District_5', 'NoB_2', 'NoB_3', 'NoB_4', 'Elevator_1', 'DoB_1400s',
       'DoB_Before 1390' , 'Balcony_1']
selected_data = df[dummy_cols]
df[dummy_cols] = df[dummy_cols].astype(int)
df.head()
```

در کد بعدی از MinMaxscaler استفاده کردم تا دادهای numerical رو در یک scale مناسب قرار بگیرند در این فایل هم متدهای مختلف مثل standardScaler هم برازش داده شد که بهترین گزینه همین MMS بود.

```
scaler = MinMaxScaler()
df[['Meterage' , 'Rent','Deposit']] = scaler.fit_transform(df[[
'Meterage','Rent' , 'Deposit']])
```

در بلاک بعدی یک جورایی ستون اجاره بها و ودیعه رو یکی کردم تا یک Target variable داشته باشم به جای دو تا:

```
df['nRent'] = df['Rent'] + df['Deposit']
```

این کد هم میاد و سطرهای تکراری رو حذف میکنه و چیزی که متعجبم کرد این بود که حدود ۳ تا از سطر ها رو دوبار نوشته بودم

```
df  = df.drop_duplicates()
```

مدل رگرسیون با statsmodels

کتابخونه آماری رو وارد کردیم با مخفف sm:

```
import pandas as pd
import statsmodels.api as sm
```

متغیرهای مستقل و وابسته رو از هم جدا میکنیم :

```
Y = df['nRent']
X = df.drop(['Rent', 'Deposit' , 'nRent'], axis =1).assign(const=1)
```

از assign(const =1) استفاده کردم تا R^2 , تمرکز یافته بشه

و روی داده های preprocess شده میایم مدل می کنیم:

```
model = sm.OLS(Y , X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   nRent   R-squared:                       0.771
Model:                             OLS   Adj. R-squared:                  0.737
Method:                  Least Squares   F-statistic:                     22.15
Date:                 Sun, 01 Jun 2025   Prob (F-statistic):           1.36e-26
Time:                         22:26:55   Log-Likelihood:                 85.982
No. Observations:                  122   AIC:                            -138.0
Df Residuals:                      105   BIC:                            -90.30
Df Model:                           16
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Meterage       1.3746      0.123     11.184      0.000       1.131       1.618
FloorNum_2     0.0326      0.035      0.927      0.356      -0.037       0.102
FloorNum_3     0.0206      0.037      0.560      0.577      -0.052       0.093
FloorNum_4     0.0616      0.047      1.311      0.193      -0.032       0.155
FloorNum_5     0.3248      0.147      2.206      0.030       0.033       0.617
District_2    -0.0117      0.039     -0.299      0.765      -0.089       0.066
District_3    -0.0191      0.039     -0.491      0.624      -0.096       0.058
District_4    -0.0441      0.036     -1.210      0.229      -0.116       0.028
District_5    -0.0175      0.045     -0.386      0.700      -0.107       0.072
NoB_2         -0.1006      0.045     -2.243      0.027      -0.190      -0.012
NoB_3         -0.1271      0.068     -1.867      0.065      -0.262       0.008
NoB_4          0.0277      0.160      0.173      0.863      -0.290       0.346
Elevator_1     0.0513      0.032      1.595      0.114      -0.012       0.115
DoB_1400s     -0.0184      0.035     -0.523      0.602      -0.088       0.051
DoB_Before 1390 -0.0546    0.033     -1.649      0.102      -0.120       0.011
Balcony_1     -0.0111      0.032     -0.343      0.732      -0.076       0.053
const          0.0089      0.045      0.200      0.842      -0.080       0.098
==============================================================================
Omnibus:                       67.962   Durbin-Watson:                   1.915
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1717.373
Skew:                           1.205   Prob(JB):                         0.00
Kurtosis:                      21.222   Cond. No.                         23.7
------------------------------------------------------------------------------
```

که R^2 به ۷۷ رسید.

و همینطوری که میبینیم متراژ تاثیر خیلی زیادی رو اجاره بهای جدید ما داره

مدل رگرسیون با Sklearn:

مثل قبل کتابخونه های مورد نیاز رو وارد کردیم:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score , mean_squared_error ,
mean_absolute_error
```

بعد از این train_test_split انجام دادم تا داده هام overfit نشن و یک فضای امنی برای مدل ایجاد کنم:

```
X_train , X_test , y_train , y_test = train_test_split(X , Y , test_size =
0.2 , random_state=42)
```

و به این شمل مدل رو برازش میدیم :

```
model = LinearRegression(fit_intercept=True)
model.fit(X_train , y_train)
```

و برای سنجش r2 ، از این دستور استفاده می کنم:

```
r2 = r2_score(y_test,y_hat)
```

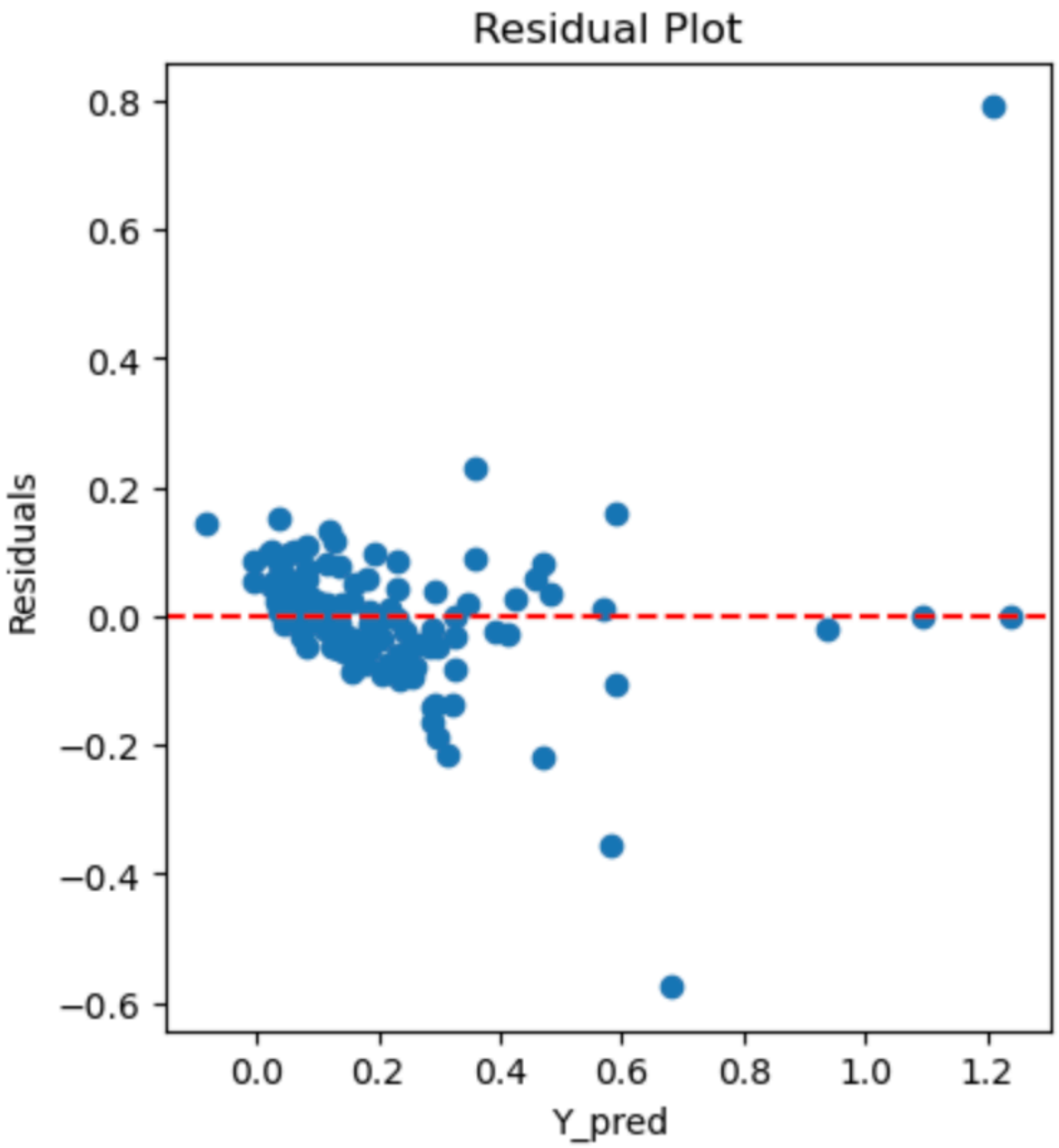
```
0.5966783933903692
```

این حدود ۲۰ درصد از برازشی که با کتابخونه قبلی انجام دادیم کمتر شد ، به نظر من دلیلش اینکه توی اینجا train_test_split انجام شده و از اونجایی که دیتای نسبتا کمی دارم مدل نتونسته اونجوری که باید دیتای من رو بشناسه

تحلیل باقی مانده ها (Residuals Analysis):

باقی مانده ها رو از این طریق بدست آوردن یعنی اون دیتای اصلی که داشتم رو منهای داده پیش بینی شده کردم و باقی مانده بدست اومد.

```
residuals = df['nRent'] - df['y_pred']
```



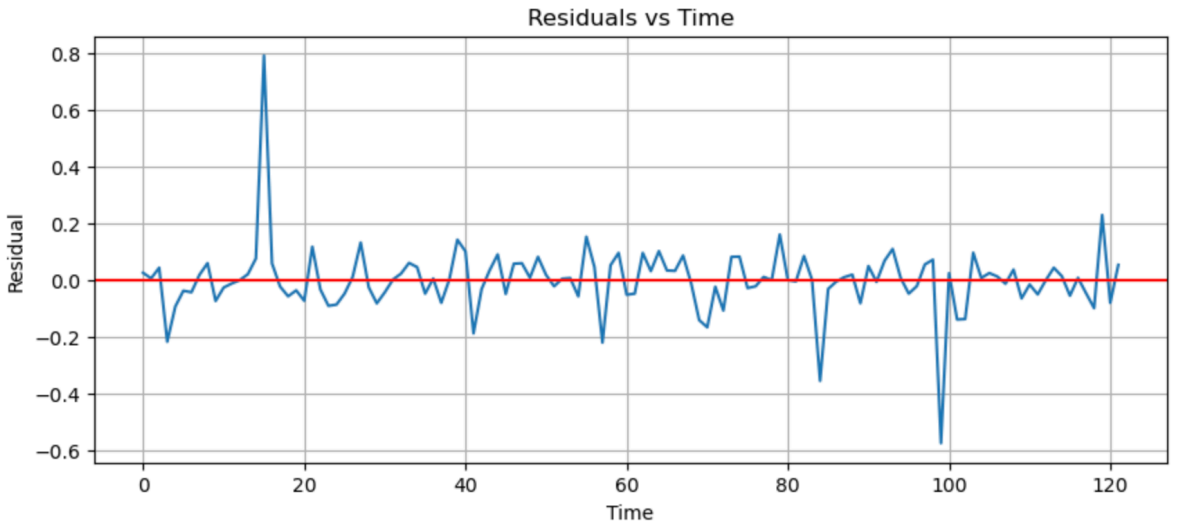
وقتی باقی مانده ها رو نسبت به $\hat{Y}$ پلات کردم نسبتا پلات پراکنده شده ای رو به نشون میده.

```
from scipy import stats
shapiro , p_value = stats.shapiro(residuals)
print(f"Shapiro-Wilk Test {shapiro:.4f}")
print(f"P-value: {p_value:}")
```

```
Shapiro-Wilk Test 0.7565
P-value: 6.046840545642 64e-13
```

مفهموم shapiro-wilk اینکه هر چی نزدیک تر باشه این عدد به ۱ یعنی داده های ما به توزیع نرمال نزدیک شدن و p_value به این کوچیکی نشون میده که H_0 که نرمال بود توزیع داده ها رو آزمون می کرد به شدت رد شده و این یک

جای پیشرفت برای این مدل هست که ممکنه اتفاقات مختلفی رخ داده باشه یا استفاده از یک مدل دیگه عاقلانه تر باشه.



این هم پلات باقی مانده نسبت به زمان هست که در یک جا یک اوج و سقوط خیلی زیاد گرفته که ممکنه نشان دهنده outlier داشتن داده ها باشه.

راه های بهتر کردن این مدل:

- بررسی outliers ها (این رو بررسی کردم و R2 اومد پایین تر وقتی که outlier ها رو حذف کردم)
- شاید باید از یک مدل دیگه استفاده کنم؟

انتخاب بهترین متغیرها (Feature Selection):

مهم ترین کدی که در این سطح وجود داره این بلاک از کد هست :
که توضیح میدم تمام پارامتر های اون رو:

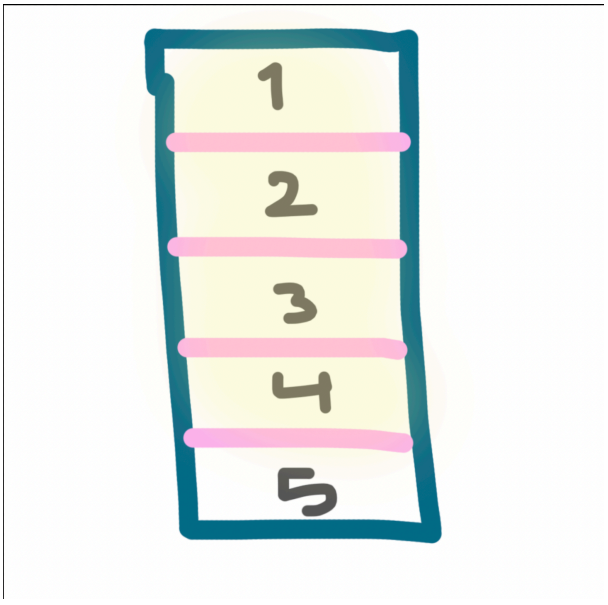
```
model = LinearRegression()
sfs = SequentialFeatureSelector(model,
                                k_features='best',
                                forward=True,
                                floating=False,
                                scoring='neg_mean_squared_error',
                                cv=5)
sfs.fit(X, y)
print('Selected features:', sfs.k_feature_idx_)
```

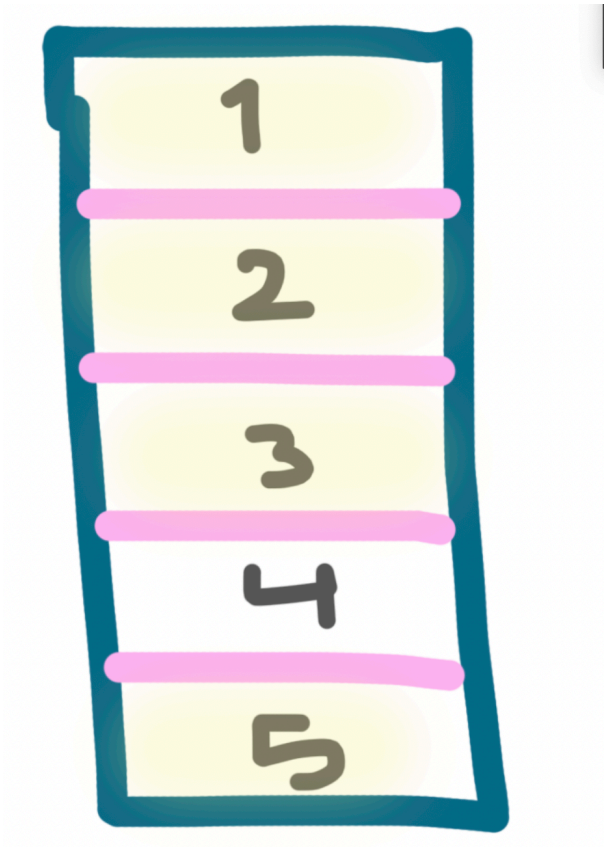
برای انتخاب بهترین متغیرها از این تابع از sklearn می گیریم .

1. مدلی هست که میخوایم روش این رو اعمال کنیم
2. به ما میگه که چند تا از متغیر های مستقل رو انتخاب کنه برای ما یا اگه رو best باشه به طور خودکار تعدادش رو مشخص می کنه
3. نشون میده که چه نوع selection میخوایم اعمال بشه که در اینجا forward انجام میشه و اگه False بزاریم به صورت backward اعمال میشه

4. این پارامتر یک جورایی stepwise رو برای ما اعمال که در اینجا False قرار داده شده
5. روش ارزیابی مشخص میکنیم با این پارامتر که منفی MSE استفاده شده
6. این پارامتر هم خیلی برام جالب بود به نام Cross-validation هست که باعث میشه از overfit شدن جلوگیری میکنه

در اینجا هر کدوم داده های ما به ۵ بخش مختلف تقسیم میشن که به هر کدوم از این بخش ها fold گفته میشه , که چهار تای اونها به عنوان training استفاده میشن و یکی از اون ها به عنوان test به کار رفته میشه
این فرآیند با تمام حالت هاش اجرا میشه و بعد یک خروجی به ما میده





و ستون های DoB_Before 1390 ، District_2 ، FloorNum_5 ، Meterage به عنوان متغیرهای بهتر برای مدل رگرسیونی ما انتخاب شدن. که وقتی با این ستون ها مدل رو برازش کردم:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   nRent   R-squared:                       0.738
Model:                             OLS   Adj. R-squared:                  0.730
Method:                  Least Squares   F-statistic:                     84.65
Date:                 Mon, 02 Jun 2025   Prob (F-statistic):           5.26e-34
Time:                         14:19:19   Log-Likelihood:                 81.003
No. Observations:                  125   AIC:                            -152.0
Df Residuals:                      120   BIC:                            -137.9
Df Model:                            4
Covariance Type:             nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Meterage           1.3191      0.080     16.446      0.000       1.160       1.478
FloorNum_5         0.3112      0.139      2.232      0.027       0.035       0.587
District_2         0.0235      0.031      0.771      0.442      -0.037       0.084
DoB_Before 1390   -0.0662      0.028     -2.352      0.020      -0.122      -0.010
const             -0.0591      0.022     -2.660      0.009      -0.103      -0.015
==============================================================================
Omnibus:                       42.070   Durbin-Watson:                   1.786
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              904.390
Skew:                           0.313   Prob(JB):                     4.11e-197
Kurtosis:                      16.163   Cond. No.                         13.2
==============================================================================
```

که r2 برابر ۷۳ درصد شد ولی یک چیزی که از قبل پیشرفت کرد p_value های هرکدوم از متغیرهای مستقل بود.

> **Tip**
>
> خیلی جا برای کار داره ولی به همینجا بسنده می کنم فعلا

References:

می تونین در آخر این فایل بررسیش کنید .