(به نام خداوند مهربان)

پروژه دوره Data Analysis مجتمع فنی

استاد: مهندس جناب آقای ولدبیگی

نام تحلیلگر: مهدی عماری

پیش بینی قیمت مسکن در تهران

House Price Prediction

(Regression)

دیتاست : https://www.kaggle.com/datasets/mokar2001/house-price-tehran-iran

توضیحات درباره دیتاست :

این دیتاست که دیتا اصلی آن قیمت‌های خانه‌ها در شهر تهران است اطلاعاتی مانند متراژ خانه، تعداد اتاق‌ها، آسانسور، پارکینگ،موقعیت جغرافیایی و قیمت فروش است. این اطلاعات می‌تواند برای تحلیل بازار مسکن، پیش‌بینی قیمت‌ها و ارزیابی سطح معیشت در شهر تهران مورد استفاده قرار گیرد.

داده‌های این دیتاست شامل اطلاعات عددی (numerical) مانند متراژ خانه، تعداد اتاق‌ها و قیمت فروش است. همچنین شامل اطلاعات دسته‌ای (categorical) مانند وجود آسانسور، پارکینگ و موقعیت جغرافیایی نیز می‌باشد. این اطلاعات می‌توانند برای تحلیل بازار مسکن و پیش‌بینی قیمت‌ها مورد استفاده قرار بگیرند.

برای دیتاستی که شامل اطلاعات عددی و دسته‌ای است، می‌توان الگوریتم‌های متنوعی را پیاده‌سازی کرد

۱) Regression Algorithms: برای پیش‌بینی قیمت فروش می‌توان از الگوریتم‌های رگرسیون مانند Linear Regression یا Decision Tree Regression و.... استفاده کرد. (من از الگریتم های این دسته استفاده کردم چون برای پیش‌بینی یک متغیر وابسته (مانند قیمت خانه) بر اساس یک یا چند متغیر مستقل (مانند متراژ، تعداد اتاق‌ها و غیره) بسیار مفید هستند. ✓

۲) Classification Algorithms: برای پیش‌بینی وجود یا عدم وجود آسانسور، پارکینگ و موقعیت جغرافیایی می‌توان از الگوریتم‌های Classification مانند Decision Trees یا Random Forest استفاده کرد. ✖

۳) Clustering Algorithms: برای گروه‌بندی خانه‌ها بر اساس ویژگی‌هایشان می‌توان از الگوریتم‌های Clustering مانند K-Means یا DBSCAN استفاده کرد. ✖

۴) Neural Networks: از شبکه‌های عصبی برای مدل‌سازی پیچیده‌ترین روابط بین ویژگی‌های مختلف خانه‌ها و قیمت‌ها استفاده می‌شود. ✖

تحلیل روی کد :

۱) وارد کردن دیتاست (دیتاست قیمت خانه در تهران)

۲) Preprocessing یا پیش پردازش دیتا:

Preprocessing یا پیش‌پردازش داده‌ها به مرحله‌ای اطلاق می‌شود که قبل از انجام تحلیل‌های آماری یا مدل‌سازی بر روی داده‌ها، داده‌ها را تمیز می‌کند و آماده می‌کند. این شامل کارهایی مانند پر کردن مقادیر خالی (مفقود)، تبدیل داده‌های دسته‌ای به شکل عددی (مانند one-hot encoding یا label encoding)، مقیاس‌بندی داده‌ها، حذف داده‌های نامرتبط یا نویزی و ... می‌شود. هدف این مرحله ایجاد یک داده‌ی تمیز و مناسب برای انجام تحلیل‌های بعدی و مدل‌سازی است.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3479 entries, 0 to 3478
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         3479 non-null   object
 1   Room         3479 non-null   int64
 2   Parking      3479 non-null   bool
 3   Warehouse    3479 non-null   bool
 4   Elevator     3479 non-null   bool
 5   Address      3456 non-null   object
 6   Price        3479 non-null   float64
 7   Price(USD)   3479 non-null   float64
dtypes: bool(3), float64(2), int64(1), object(2)
memory usage: 146.2+ KB
```

داده های این دیتاست هم از نوع عددی(float,INT) و از نوع Boolean(true,false) هستند و objectهستند که ما در این مرحله به تبدیل مقدار آنها به نوع عدد میپردازیم

این دستورات با حذف کاماها و تبدیل مقادیر به عددی، داده‌های ستون "Area" را تمیز کرده و آماده برای استفاده در تحلیل‌های بعدی می‌کنند.

```
df['Area'] = df['Area'].replace (',' , '')
df['Area']= pd.to_numeric(df['Area'] , errors='coerce')
```

```
df = df.drop(columns = ['Price(USD)'])
boolean_features = ['Parking','Warehouse','Elevator']
df[boolean_features] = df[boolean_features].astype('int64')
```

دستور بالا اول، ستون Price(USD) را از DataFrame حذف می‌کند و دستور دوم، ستون‌های مندرج در لیست boolean_features را به نوع داده‌ای int64 تبدیل می‌کند بنابراین، این دستورات با حذف ستون Price(USD) و تبدیل ستون‌های مندرج در لیست boolean_features به نوع داده‌ایint64 ، تغییرات مورد نیاز را در DataFrame اعمال می‌کنند.

```
address_dummy = pd.get_dummies(df['Address'])

df_final = df.merge(address_dummy, left_index = True, right_index = True)

df_final.drop(columns = 'Address', inplace = True)

# df_final.head(3)

df_final
```

این کد چند کار را انجام می‌دهد اول که با استفاده از(pd.get_dummies(df['Address'] ، متغیرهای دامی برای ستون Address ایجاد می‌شود. این کار به تبدیل متغیرهای دسته‌ای به متغیرهای دامی (One-Hot Encoding) می‌پردازد. به عبارت دیگر، هر مقدار یکتا از ستون Addressبه یک ستون جدید تبدیل می‌شود که نشان دهنده حضور یا عدم حضور آن مقدار است. سپس، با استفاده از df.merge(address_dummy, left_index=True, right_index=True)، داده‌های ایجاد شده از متغیرهای دامی به DataFrame اصلی (df) اضافه می‌شود. این کار با استفاده از ایندکس‌ها left_index و right_index انجام می‌شود. در مرحله بعد، باdf_final.drop(columns='Address', inplace=True) ، ستون Addressاز DataFrame نهایی حذف می‌شود و تغییرات در محل اعمال می‌شود.

به طور خلاصه، این کد برای تبدیل متغیرهای دسته‌ای به متغیرهای دامی و اضافه کردن آن‌ها به DataFrame اصلی، سپس حذف یک ستون مشخص از DataFrame نهایی استفاده می‌شود.

```
df.isna().sum()
```

```
[136]
...
Area            6
Room            0
Parking         0
Warehouse       0
Elevator        0
Address        23
Price           0
Price(USD)      0
dtype: int64
```

این دستور برای محاسبه تعداد مقادیر نامعتبر (NaN) Not a Number-مفقود شده در هر ستون از DataFrame به نام df استفاده می‌شود. وقتی این دستور اجرا می‌شود، خروجی نشان می‌دهد که در هر ستون چند مقدار NaN وجود دارد. این اطلاعات می‌تواند برای تشخیص دادن داده‌های ناقص و پردازش آن‌ها در مراحل بعدی مورد استفاده قرار گیرد. **۲۶ تا ادرس و ٦ تا موقعیت جغرافیای خالی هستند البته ما با دستو زیر این مشکل را رفع کردیم**

df.dropna(inplace = True)

این دستور برای حذف همه ردیف‌هایی از DataFrame به نام df استفاده می‌شود که حداقل یک مقدار NaN دارند وقتی پارامتر inplace=True استفاده می‌شود، تغییرات مستقیماً در DataFrame اعمال می‌شود و DataFrame اصلی تغییر می‌کند. به عبارت دیگر DataFrame اصلی پس از اجرای این دستور داده‌هایی که حاوی مقادیر NaN هستند را حذف می‌کند و فقط داده‌های کامل باقی می‌مانند. این کار ممکن است برای حذف داده‌های نامعتبر و پردازش داده‌ها قبل از مراحل تحلیل و مدل‌سازی مورد استفاده قرار گیرد

df2 = df.copy()

df2.loc[:, "Price"] =df["Price"].map('{:,.0f}'.format)

df2

این کد، یک کپی از داده‌ها (df) می‌سازد و سپس ستون Price را با استفاده از تابع map و فرمت‌دهی رشته، به صورت عددی با فرمت سه رقمی جداکنده با کاما تغییر می‌دهد. به این صورت که هر عدد، با استفاده از فرمت {:,.۰f} به یک رشته با فرمت سه رقمی جداکننده با کاما تبدیل می‌شود. سپس داده‌های جدید در df2 قرار می‌گیرند.(**البته این بخش دلبلخواه است و ما روی df اصلی انجام ندادیم**)

۳)اطلاعات در مورد داده ها (EDA)

EDAیا Exploratory Data Analysis به معنای تحلیل اکتشافی داده‌ها است. این فرایند شامل بررسی و تجزیه‌وتحلیل داده‌ها به منظور کشف الگوها، روابط و ویژگی‌های مهم داده‌ها می‌شود. هدف اصلیEDA ، درک بهتری از داده‌ها و تهیه زمینه‌ای برای انجام تحلیل‌های پیشرفته‌تر است برای انجام EDA بر روی داده‌ها، می‌توانید از روش‌های مختلفی مانند تجزیه و تحلیل توصیفی(Descriptive Statistics) ، نمودارها، تحلیل تفسیری و بررسی توزیع‌ها و تغییرات داده‌ها استفاده کنید. همچنین می‌توانید به دنبال داده‌های پرت، اطلاعات ناقص یا تکراری و همچنین بررسی توزیع متغیرها و روابط بین آن‌ها با یکدیگر باشید.با انجام EDA، شما می‌توانید بهترین روش‌ها برای پیش‌پردازش داده‌ها و انتخاب مدل‌های مناسب برای تحلیل داده‌ها را شناسایی کنید.

ما برای انجام Exploratory Data Analysis (EDA) بر روی دیتاستمان، اقداماتی انجام دادیم از جمله:

۱. **تجزیه و تحلیل توصیفی:** بررسی آمارهای توصیفی مانند میانگین، میانه، واریانس و کوچکترین/بزرگترین مقادیر هر ویژگی

	Room	Price	Price(USD)
count	3479	3479	3479
mean	2	5359022711	178634
std	1	8099934524	269998
min	0	3600000	120
25%	2	1418250000	47275
50%	2	2900000000	96667
75%	2	6000000000	200000
max	5	92400000000	3080000

با توجه به این جدول ،اطلاعات برخی آمارهای توصیفی می‌توان تحلیل‌های مختلفی انجام داد. برای مثال می‌توانید توزیع تعداد اتاق‌ها و قیمت‌ها را بررسی کرده و الگوها و تفاوت‌های موجود را مشاهده کنید. همچنین، می‌توانید با استفاده از این داده‌ها، رابطه بین تعداد اتاق‌ها و قیمت‌ها را بررسی کرده و احتمالاً الگوهایی مانند افزایش قیمت با

---

۲. **بررسی روابط:** بررسی روابط و ارتباطات بین ویژگی‌ها، به دنبال ویژگی‌هایی که با یکدیگر همبستگی دارند یا تاثیر یکدیگر را می‌توانند بررسی کنید.

df['Parking'].value_counts(normalize=True)*100

این دستور برای محاسبه تعداد و درصد تکرار هر گروه مختلف در ستون Parkingاز df استفاده می‌شود. از پارامتر normalize=True برای محاسبه درصد تکرار هر گروه به جای تعداد استفاده می‌شود. به عبارت دیگر این دستور نشان می‌دهد که هر گروه از مقادیر مختلف در ستون Parking چه درصد از کل داده‌ها را تشکیل می‌دهد. این اطلاعات می‌توانند برای درک بهتر توزیع داده‌ها و تحلیل متغیرها در مراحل بعدی مورد استفاده قرار گیرند.

df[df.Area <= 85]["Parking"].value_counts( )

این دستور ابتدا فیلتری را روی DataFrame به نام df اعمال می‌کند. فیلتر مربوط به مقادیر ستون Area است که از ۸۵ کمتر یا مساوی هستند سپس برای این زیرمجموعه از داده‌ها، تعداد تکرار هر مقدار مختلف در ستون "Parking" محاسبه می‌شود و به صورت یک سری (Series) از تعداد تکرار هر مقدار برگردانده می‌شود. این اطلاعات می‌تواند برای درک بهتر توزیع داده‌ها و تحلیل متغیرها در مراحل بعدی مورد استفاده قرار گیرد.

df.groupby('Room')['Price'].mean()

این دستور یک گروه‌بندی را روی df اعمال می‌کند. داده‌ها بر اساس ستون Room گروه‌بندی می‌شوند و برای هر گروه، میانگین مقادیر موجود در ستون Price محاسبه می‌شود. به عبارت دیگر، این دستور نشان می‌دهد که میانگین قیمت ملک‌ها بر اساس تعداد اتاق‌ها چقدر است. این نوع تحلیل می‌تواند به درک بهتر توزیع قیمت‌ها و ویژگی‌های مختلف ملک‌ها کمک کند.

room_parking_room_mean_df = df.groupby(['Room','Parking'])['Price'].mean().reset_index()

این دستور یک گروه‌بندی چند سطری را روی df اعمال می‌کند. داده‌ها بر اساس دو ستون Room و Parking گروه‌بندی می‌شوند و برای هر گروه، میانگین مقادیر موجود در ستون Price محاسبه می‌شود. سپس با استفاده از دستور reset_index() این گروه‌بندی به یک

DataFrame جدید با ستون‌های Room، Parking و Price تبدیل می‌شود. این کار می‌تواند برای تحلیل متغیرهای مختلف و تعامل بین آن‌ها مفید باشد، به عنوان مثال برای درک تأثیر تعداد اتاق‌ها و پارکینگ بر قیمت ملک‌ها.

	Room	Parking	Price
0	0	False	9769750000
1	0	True	223500000
2	1	False	1139677778
3	1	True	2031368545
4	2	False	1467853909
5	2	True	3588678643
6	3	False	5107865385
7	3	True	11146619318
8	4	False	6200000000
9	4	True	25881492754
10	5	False	9999000000
11	5	True	37972857143

با توجه به این داده‌ها می‌توان تحلیل‌های مختلفی انجام داد. به عنوان مثال:میانگین قیمت ملک‌ها با توجه به تعداد اتاق‌ها و وجود یا عدم وجود پارکینگ(**همون صفر و یک است چون میخواستم رابطه ام با true و false فعلا باشه برای همین از دیتاست قبل از پیش پردازش استفاده کردم**) را بررسی کردم. برای مثال، مشخص شده است که میانگین قیمت ملک‌هایی که دارای ۴ اتاق و پارکینگ هستند، حدود ۲۵ میلیارد تومان است

همچنین که میتوان توزیع قیمت ملک‌ها در هر دسته (بر اساس تعداد اتاق‌ها و وجود یا عدم وجود پارکینگ) را می‌توان بررسی کرد. برای مثال مشخص شده که قیمت ملک‌هایی که دارای ۵ اتاق و پارکینگ هستند، بین ۹ میلیارد تومان تا ۳۷ میلیارد تومان متغیر است.

همچنین می‌توان توزیع تعداد ملک‌ها را هم در هر دسته را بررسی کرد برای مثال مشخص است که تعداد ملک‌هایی که دارای ۳ اتاق و پارکینگ هستند بیشترین تعداد را دارند و می‌توان توزیع تفاوت قیمت ملک‌ها در هر دسته را بررسی کرد. برای مثال، مشخص شده است که قیمت ملک‌هایی که دارای ۰ اتاق و پارکینگ هستند، دارای بیشترین تفاوت در قیمت هستند.

۳. **بررسی داده‌های پرت:** شناسایی داده‌های پرت و اطلاعات ناقص و انجام تصمیمات مرتبط با آن‌ها.

```
def lower_upper(x):
    Q1 = np.percentile(x, 25)
    Q3 = np.percentile(x, 75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    return lower, upper


lower_area, upper_area = lower_upper(df['Area'])
lower_price, upper_price = lower_upper(df['Price'])


print(f"Lower limit for area: {lower_area:0.2f}")
print(f"Upper limit for area: {upper_area:0.2f}")
print(f"Lower limit for price: {lower_price:,}")
print(f"Upper limit for price: {upper_price:,}")
```

این کد برای محاسبه حد پایین و حد بالای داده‌های ستون‌های Area و Price در یک DataFrameبا استفاده از روش (IQR) است

تابع lower_upper ابتدا کوارتیل اول (Q1) و کوارتیل سوم (Q3) را با استفاده از تابع np.percentileمحاسبه می‌کند. سپس با استفاده از فرمولIQR = Q3 - Q1 ، IQRرا محاسبه می‌کند. در نهایت، با ضرب ۱.۵ درIQR ، حد پایین و حد بالای داده‌ها را محاسبه می‌کند.

نکته۱: IQRبه معنای "محدوده میان‌کارگری" یا به انگلیسی "Interquartile Range" است. این مقدار معیاری از پراکندگی داده‌ها است که از کوارتیل اول (Q1) تا کوارتیل سوم (Q3) محاسبه می‌شود . برای محاسبهIQR ، ابتدا باید کوارتیل اول و کوارتیل سوم را محاسبه کنیم. سپس IQR برابر با تفاوت بین این دو کوارتیل محاسبه می‌شود:

\[IQR = Q3 - Q1 \]

IQR برای محاسبه انحراف معیاری و پراکندگی داده‌ها استفاده می‌شود. همچنین برای تشخیص داده‌های پرت (outlier) و همچنین تعیین حد پایین و حد بالای مقادیر معقول استفاده می‌شود.

نکته۲: کوارتیل‌ها(چارک) چهار مقدار متمایز در مجموعه‌ای از داده‌ها هستند که به چهار بخش مساوی تقسیم می‌کنند. این چهار مقدار به ترتیب کوارتیل اول(Q1) ، میانه (کوارتیل دوم)، و کوارتیل سوم (Q3) نامیده می‌شوند برای محاسبه کوارتیل‌ها، ابتدا داده‌ها را به صورت صعودی مرتب می‌کنیم. سپس میانه داده‌ها را می‌یابیم که به عنوان کوارتیل دوم شناخته می‌شود سپس داده‌ها را به دو بخش تقسیم می‌کنیم و کوارتیل اول و سوم به ترتیب به عنوان ۲۵٪ و ۷۵٪ مقادیر مرتبط انتخاب می‌شوند کوارتیل‌ها برای توصیف و تجزیه و تحلیل داده‌ها و همچنین تشخیص داده‌های پرت (outlier) استفاده می‌شوند در خطوط بعدی، این تابع را برای ستون‌های "Area" و Price در یک DataFrame به نام df فراخوانی می‌کند و حد پایین و حد بالای داده‌ها را برای هر ستون محاسبه می‌کند

۵. **تحلیل مکانی:** اگر داده‌ها مکانی هستند، می‌توانید از نقشه‌ها و تحلیل مکانی برای بررسی الگوها و روابط مکانی استفاده کنید

۶. **تجزیه و تحلیل توزیع‌های احتمالاتی:** این بخش بررسی توزیع داده‌ها و احتمالات مربوط به آنها را شامل می‌شود، مانند توزیع نرمال، توزیع یکنواخت و توزیع دیگر.

۷. **مقایسه داده‌ها:** در این بخش، داده‌ها با یکدیگر مقایسه می‌شوند، از جمله مقایسه توزیع دو یا چند متغیر، مقایسه میانگین یا میانه دو یا چند گروه داده و مقایسه الگوهای مختلف داده‌ها.

۸. **نمودارها (Visualizaion)**: رسم نمودارهای مختلف برای نمایش توزیع و رابطه بین ویژگی‌ها است مثلاً نمودارهای توزیع فراوانی (Histogram) برای نمایش توزیع ویژگی‌ها نمودار پراکندگی (Scatter plot) برای نمایش روابط بین ویژگی‌ها و نمودار جعبه‌ای (Box plot)برای نمایش توزیع ویژگی‌ها بر اساس دسته‌بندی‌های دیگر.

```
plt.figure(figsize=(8,5))
sns.displot(df['Price'] , bins=30 , kde=True )
```

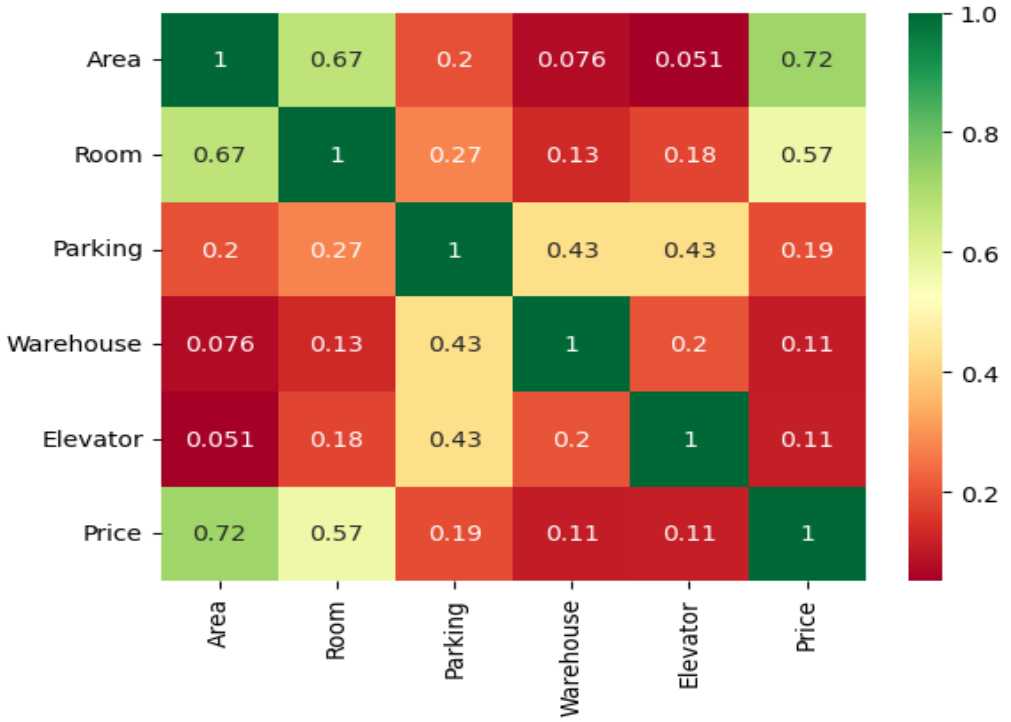
این کدی که ما نوشتیم یک نمودار توزیع فراوانی قیمت ملک‌ها را با استفاده از کتابخانه Seaborn رسم می‌کند اندازه نمودار با استفاده از figsize تنظیم کردیم برای رسم نمودار، از تابع displot استفاده می‌شود که برای توزیع‌های یک بعدی مناسب است در اینجا ستون Price از داده‌ها به عنوان ورودی به تابع داده می‌شود. با تنظیم bins به ۳۰ تعداد بازه‌هایی که برای رسم نمودار استفاده می‌شوند، تعیین می‌شود. همچنین، با تنظیم kde به True، منحنی چگالی احتمال نیز در نمودار رسم می‌شود.

```
sns.heatmap(df.corr(), annot=True,cmap='RdYlGn')
```

من با استفاده از این دستور نمودار حرارتی از ماتریس همبستگی داده‌هایم را ایجاد کردم

annot=True به شما کمک می‌کند تا مقادیر واقعی همبستگی را در هر خانه از نمودار حرارتی نشان دهید cmap='RdYlGn یک رنگ‌های انتخابی برای نمودار حرارتی است در این حالت، از

رنگ‌های قرمز، زرد و سبز استفاده می‌شود که به ترتیب نشان‌دهنده مقادیر منفی، صفر و مثبت هستند.



این نمودار حرارتی ما است که ماتریس همبستگی بین متغیرهای مختلف مرتبط با ملک را نشان می دهد. متغیرهای شامل مساحت، تعداد اتاق، پارکینگ، انبار، آسانسور و قیمت هستند.

در یک ماتریس همبستگی:

مقدار ۱ (که با سبز تیره نشان دادم) در امتداد قطری نشان می دهد که هر متغیر به طور کامل با خودش همبستگی دارد.

مقادیر نزدیک به ۱ (سایه های سبز روشن) نشان دهنده همبستگی مثبت قوی است، به این معنی که هر چه یک متغیر بیشتر شود، دیگری هم بیشتر می شود.

مقادیر نزدیک به ۰ (سایه های قرمز) نشان دهنده همبستگی ضعیف است، به این معنی که رابطه کمتری بین افزایش یا کاهش دو متغیر وجود دارد

نوار رنگی در سمت راست به عنوان یک راهنما عمل می کند و یک گرادیان از قرمز (همبستگی کم) تا سبز (همبستگی زیاد) نشان می دهد

از نمودار حرارتی می توانیم بگیم که:

مساحت و قیمت دارای همبستگی مثبت نسبتاً قوی (۰.۷۲) هستند، که نشان می دهد مناطق بزرگتر به قیمت های بالاتر مرتبط هستند

پارکینگ همبستگی مثبت متوسطی با 'انبار' و 'آسانسور' دارد (هر دو ۰.۴۳)، که ممکن است نشان دهد ویژگی های بهتر پارکینگ ممکن است همچنین دسترسی به انبار و آسانسور بهتری داشته باشد یا برعکس

اتاق همبستگی مثبت متوسطی با مساحت (۰.۶۷) و قیمت (۰.۵۷) دارد، که نشان می دهد املاک با تعداد اتاق بیشتر به مساحت بزرگتر و قیمت بالاتر مرتبط هستند

همبستگی بین آسانسور و متغیرهای دیگر مانند مساحت (۰.۰۵۱) و قیمت (۰.۱۱) نسبتاً کم است، که نشان می دهد ممکن است رابطه قوی بین حضور آسانسور و اندازه یا قیمت ملک وجود نداشته باشد

این نمودار حرارتی ما یک ابزار مفید برای به سرعت تصور قوت و جهت روابط بین متغیرهای مختلف است که می تواند برای تجزیه و تحلیل داده ها، به ویژه در زمینه املاک که چنین همبستگی هایی می تواند استراتژی های سرمایه گذاری و بازاریابی را مطلع کند حیاتی باشد

---

df3 = df['Address'].value_counts().copy()

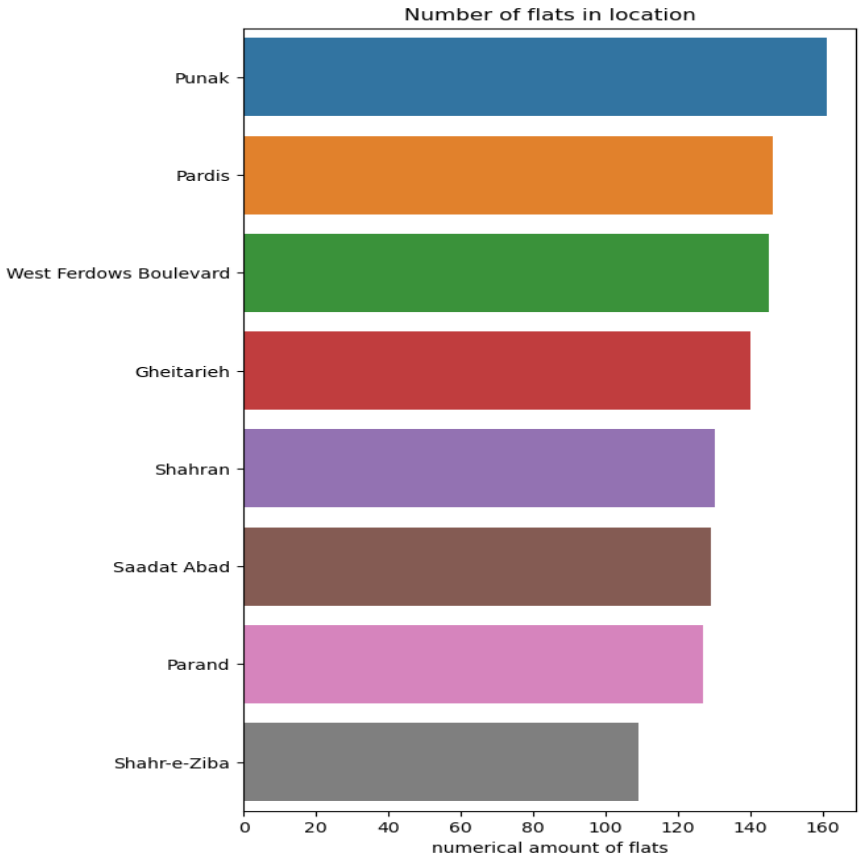
df3 = df3[:8]

این کد به این معنی است که ابتدا یک سری متغیرهای داده از ستون آدرس را انتخاب می‌کند و تعداد تکرار هر آدرس را محاسبه می‌کند. سپس با استفاده از عملگر `[:۸]`، ۸ مقدار اول (با

بیشترین تعداد تکرار) را انتخاب می‌کند. این کد به صورت مختصر اطلاعات مربوط به ۸ آدرس با بیشترین تعداد تکرار را ارائه می‌دهد

```
fig, ax = plt.subplots(figsize=(6,10))
sns.barplot(x=df3.values, y=df3.index,ax=ax)
plt.xlabel('numerical amount of flats')
plt.title('Number of flats in location')
```

من با استفاده از کد بالا یک نمودار میله‌ای از داده‌های موجود در df3 ایجاد کردم. این نمودار به شما کمک می‌کند تا تعداد آپارتمان‌ها در هر مکان را مقایسه کنید خلاصه که این کد برای ایجاد یک نمودار میله‌ای از تعداد آپارتمان‌ها در هر مکان و برچسب‌گذاری آن استفاده می‌شود



این نمودار میله‌ای افقی ما است که تعداد آپارتمان‌های موجود در مکان‌های مختلف را نشان می‌دهد. هر مکان با یک رنگ منحصر به فرد نمایش داده شده و در امتداد محور y قرار دارد، در

حالی که محور x تعداد آپارتمان‌ها را نشان می‌دهد که از ۰ تا ۱۶۰ متغیر است. میله‌ها مقدار آپارتمان‌ها را در هر مکان نشان می‌دهند.

شروع از بالا:

پونک بیشترین تعداد آپارتمان‌ها را دارد، با تعدادی بیشتر از ۱۵۵ که آن را مکانی با بیشترین تعداد آپارتمان در این نمودار می‌کند

پردیس با تعداد کمتری آپارتمان، نزدیک به ۱۴۷

بلوار غربی فردوس حدود ۱۴۵ آپارتمان دارد

قیطریه با تقریبا ۱۴۰ آپارتمان نشان داده شده است

شهران هم حدود ۱۳۵ آپارتمان دارد، کمی کمتر از قیطریه

سعادت آباد نزدیک به ۱۳۵ آپارتمان را نشان می‌دهد

پرند حدود ۱۳۰ آپارتمان دارد

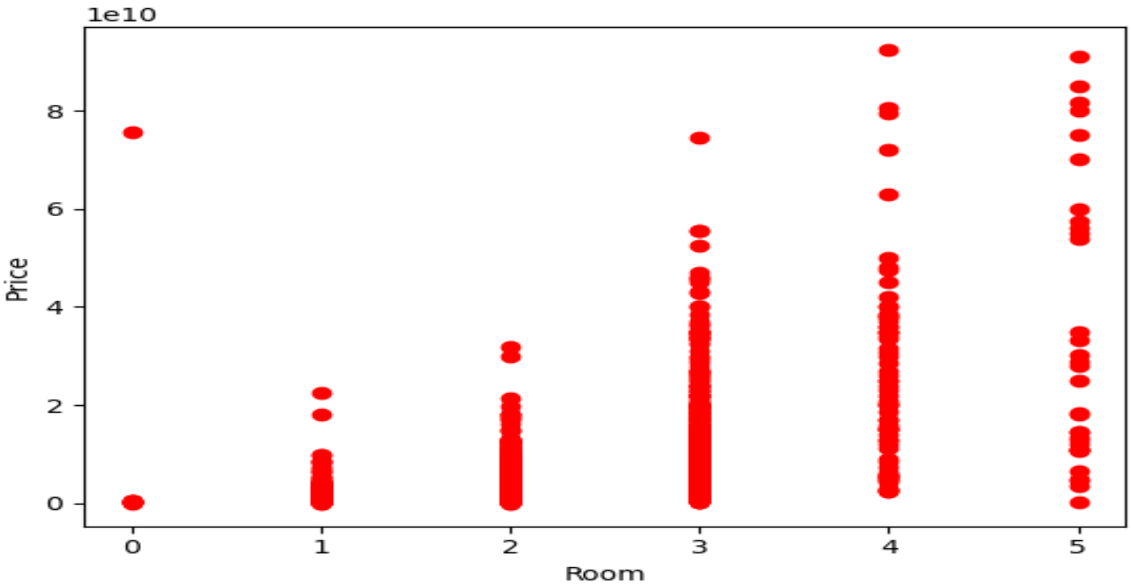
شهر زیبا کمترین تعداد آپارتمان را در این نمودار دارد، تقریبا ۱۱۰

لطفا توجه کنید که اعداد دقیق ارائه نشده‌اند بنابراین ارقام مذکور بر اساس برآوردهای بصری از طول میله‌ها است

```
plt.scatter(df.Room, df.Price,  color='red')
plt.xlabel("Room")
plt.ylabel("Price")
plt.show()
```

من با استفاده از این کد یک نمودار پراکندگی (scatter plot) از داده‌های موجود در df ایجادکردم این نمودار به من کمک کرد تا رابطه بین تعداد اتاق‌ها (Room) و قیمت (Price) آپارتمان‌ها را بررسی کنید

این نمودار به ما کمک می‌کند تا ببینید که آیا قیمت آپارتمان‌ها با افزایش تعداد اتاق‌ها افزایش یافته یا خیر، و اینکه آیا وجود هرگونه الگوی قابل توجهی در این رابطه وجود دارد یا خیر



این نمودار رابطه بین دو متغیر اتاق در محور x و قیمت در محور y را نشان می‌دهد. محور قیمت با استفاده از نمایش علمی مقیاس‌بندی شده است1e10به این معنی که قیمت‌ها در اینجا در واقع در ۱۰ به توان ۱۰ ضرب می‌شوند .این نمودار نقاط داده‌های فردی را نشان می‌دهد که نقاط مختلفی را برای دسته‌های یا تعدادات مختلف اتاق نشان می‌دهد

دسته‌های یا تعدادات اتاق از ۰ تا ۵ در محور x قرار دارند

برای هر دسته اتاق، محدوده‌ی گسترده‌ای از قیمت‌ها وجود دارد، با برخی نقاط داده که به قیمت‌های بسیار بالا در بالای نمودار می‌رسند

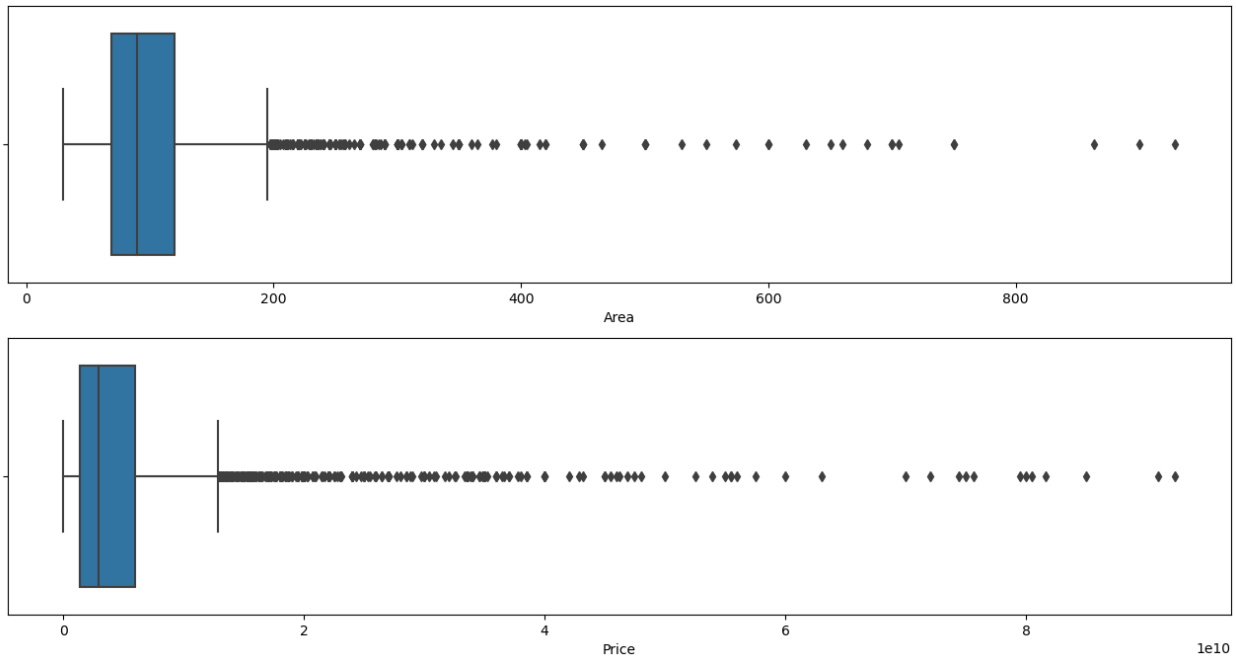
توزیع قیمت‌ها برای هر دسته اتاق به نظر می‌رسد گسسته است و شکاف‌های قابل مشاهده بین نقاط قیمت وجود دارد، که نشان می‌دهد قیمت‌ها ممکن است در دسته‌ها یا گروه‌های خاصی قرار دارند

من نمی‌توانم نتیجه بگیریم که آیا هر گونه روند یا همبستگی بین تعداد اتاق‌ها و قیمت وجود دارد یا خیر. با این حال، این نمودار می‌تواند برای شناسایی چگونگی انتشار قیمت‌ها و دیدن تنوع در هر دسته اتاق مفید باشد.

```
plt.figure(figsize = (16,8))
plt.subplot(2,1,1)
sns.boxplot(x = df['Area'])
plt.subplot(2,1,2)
sns.boxplot(x = df['Price'])
```

این کد دو نمودار باکس‌پلات را در یک شکل به ابعاد ۱۶ در ۸ اینچ نشان می‌دهد در نمودار اول داده‌های مربوط به متغیر Area از دیتافریم df را نمایش می‌دهد. نمودار باکس‌پلات نشان می‌دهد که چگونه داده‌ها در یک متغیر متمرکز شده‌اند و چگونه انحراف‌ها و توزیع‌های مختلف داده‌ها در متغیر Areaوجود دارد در نمودار دوم داده‌های مربوط به متغیر Price از دیتافریم df را نمایش می‌دهد نمودار باکس‌پلات ما نشان می‌دهد چگونه قیمت‌ها در دیتاست توزیع شده‌اند و چگونه انحراف‌ها و توزیع‌های مختلف قیمت‌ها وجود دارد.

این نمودارها برای مقایسه توزیع داده‌ها و شناسایی پتانسیل داده‌های پرت و انحراف‌های بزرگ مفید هستند.



دو نمودار باکس پلات ما داریم در اینجا که یکی بالای دیگری قرار گرفته‌اند و توزیع دو مجموعه داده مختلف را نشان می‌دهند. بر اساس برچسب‌گذاری محورها، به نظر می‌رسد که یکی از آن‌ها مربوط به مساحت و دیگری به قیمت است.

نمودار بالا، با برچسب مساحت به نظر می‌رسد که مجموعه داده‌ای با مقدار میانه حدود ۱۰۰-۱۵۰ دارد (**مقدار دقیق بدون داده‌های عددی سخته**). دامنه چهارچوب میانه (جعبه آبی) نسبتاً باریک است که نشان می‌دهد که ۵۰٪ میانی نقاط داده در یک محدوده نسبتاً کوچک تمرکز دارند. از هر دو طرف از جعبه، شاخه‌ها (whiskers) به سمت بیرون کشیده شده‌اند که نشان می‌دهد دامنه داده شامل چندین مقدار کمتر و بیشتر است. با این حال، تعداد قابل توجهی از نقاط داده خارج از شاخه‌ها قرار دارند که ممکن است به عنوان نقاط ناهنجار در نظر گرفته شوند، زیرا خارج از محدوده عادی داده‌ها قرار دارند

نمودار پایین، با برچسب قیمت مجموعه داده‌ای با میانه‌ای که به نظر می‌رسد نزدیک به ۱ است نشان می‌دهد، به طور مشابه با یک چهارچوب میانه باریک. دامنه داده، همانطور که توسط شاخه‌ها

نشان داده شده است، بسیار محدود است. تمرکز قابل توجهی از نقاط داده بین ۰ و ۲ وجود دارد، با چندین نقطه ممکن که کمی خارج از این محدوده قرار دارند

در کل هر دو نمودار باکس پلات نشان می‌دهند که بیشتر داده‌ها در یک محدوده کوچک تمرکز دارند و تعدادی نقاط ناهنجار وجود دارد. دقت مقیاس در نمودار قیمت برای تفسیر صحیح باید تأیید شود

۴)ساخت مدل (Model Building):

Model Building به معنای ساخت مدل است در علوم داده، مدل برای پیش‌بینی و توصیف رفتار داده‌ها به کار می‌رود. برای ساخت یک مدل ابتدا باید داده‌ها را بررسی کرده و تحلیل کرده، سپس به دنبال یافتن الگوهای موجود در داده‌ها هستیم به این منظور، می‌توانیم از الگوریتم‌های یادگیری ماشین استفاده کنیم که با استفاده از داده‌های آموزشی، مدلی را برای پیش‌بینی رفتار داده‌ها ایجاد می‌کنند.

در مدل سازی ابتدا باید یک هدف مشخص داشته باشیم. به عنوان مثال، هدف ما ممکن است پیش‌بینی قیمت یک محصول با توجه به ویژگی‌های آن باشد. سپس، داده‌های مورد نیاز برای ساخت مدل را جمع‌آوری و پیش‌پردازش می‌کنیم. سپس، مدل مورد نظر را با استفاده از یک الگوریتم یادگیری ماشین، مانند رگرسیون خطی یا شبکه‌های عصبی، ساخته و به آن داده‌های آموزشی را ارائه می‌دهیم. سپس، با استفاده از داده‌های آزمایشی، عملکرد مدل را ارزیابی کرده و در صورت نیاز، مدل را بهبود می‌بخشیم.

ما از مدل‌های رگرسیون در این پروژه استفاده کرده‌ایم

مدل‌های رگرسیون می‌توانند به دسته‌های مختلفی تقسیم شوند، اما چند مدل رایج رگرسیون که ما در پروژه مان استفاده کرده ایم عبارتند از:

۱. **رگرسیون خطی(linear Regression)**: این یکی از مدل‌های رگرسیون ساده‌تر است که از یک خط استفاده می‌کند تا رابطه بین متغیر وابسته و مستقل را مدل کند.

۲. **رگرسیون لاسوLasso Regression)**: رگرسیون لاسو یک روش رگرسیون است که برای انتخاب ویژگی‌ها و کاهش انحراف مدل استفاده می‌شود. این روش اغلب برای مدل‌سازی با تعداد زیادی ویژگی (متغیر مستقل) استفاده می‌شود و می‌تواند به کاهش اثرات ویژگی‌های غیرضروری و انتخاب ویژگی‌های مهم کمک کند.

۳. **رگرسیون ریدج(Ridge Regression)**: این یک مدل رگرسیون خطی با اضافه کردن جمله‌های جریمه به تابع هدف است که می‌تواند به مشکل برازش بیش از حد به داده‌ها کمک کند.

۴. **رگرسیون درخت تصمیم (Decision Tree Regression)**:

درخت تصمیم یک مدل پیش‌بینی است که برای مسائل پیش‌بینی و رگرسیون استفاده می‌شود. این مدل از یک ساختار درختی تشکیل شده است که به صورت سلسله مراتبی تصمیم‌ها را بر اساس ویژگی‌های ورودی می‌گیرد. درخت تصمیم برای پیش‌بینی یک متغیر پیوسته (مانند قیمت یک محصول یا درآمد) استفاده می‌شود و به عنوان یک روش رگرسیون استفاده می‌شود. درخت تصمیم به صورت بازگشتی و با استفاده از تقسیم‌بندی‌های مختلف روی ویژگی‌ها، داده‌ها را به گروه‌های کوچک‌تر تقسیم می‌کند و به این ترتیب یک مدل پیش‌بینی ایجاد می‌کند. هر گره درخت تصمیم یک تصمیم بر اساس یک ویژگی از داده‌ها را نشان می‌دهد و هر یال از گره‌ها به گره‌های فرزندش نشان‌دهنده‌ی این است که چه تصمیمی باید بر اساس ویژگی‌ها گرفته شود. در نهایت، هر برگ درخت تصمیم یک مقدار پیش‌بینی برای متغیر پیوسته را نشان می‌دهد.

درخت تصمیم یک مدل قابل فهم و تفسیر است و می‌تواند برای مسائل پیش‌بینی و رگرسیون در حالت‌هایی که ویژگی‌ها و تصمیم‌ها قابل فهم باشند، مفید باشد. اما در برخی موارد، ممکن است به دلیل انعطاف‌پذیری زیاد، درخت تصمیم به تاخیر انجام پیش‌بینی بیش‌برخوردار باشد.

۵. رگرسیون جنگل تصادفی (Random Forest Regression):

Random Forest Regressionیک روش پیش‌بینی و رگرسیون است که بر اساس الگوریتم موسوم به "جنگل تصادفی" کار می‌کند. این الگوریتم یک مدل پیش‌بینی مبتنی بر مجموعه از درخت‌های تصمیم است که به صورت تصادفی ایجاد می‌شود.

درRandom Forest Regression ، چندین درخت تصمیم به صورت موازی ایجاد می‌شوند و سپس پیش‌بینی‌های هر درخت با یکدیگر میانگین‌گیری می‌شوند تا پیش‌بینی نهایی بدست آید. این روش به دلیل اینکه از میانگین چندین درخت استفاده می‌کند، به طور معمول بهتر از یک درخت تصمیم معمولی عمل می‌کند و مقاومت بیشتری در برابر برازش بیش‌اندازه (overfitting) دارد

Random Forest Regressionاز ویژگی‌های تصادفی برای ساخت هر درخت تصمیم استفاده می‌کند. این شامل انتخاب تصادفی زیرمجموعه‌های از ویژگی‌ها و نمونه‌ها است که برای ساخت هر درخت استفاده می‌شود. این روش باعث می‌شود که هر درخت تصمیم به صورت مستقل از سایر درخت‌ها ایجاد شود و به این ترتیب، از تنوع بیشتری برخوردار باشد Random Forest Regression برای مسائل پیش‌بینی و رگرسیون مفید است، به خصوص زمانی که داده‌ها دارای تعداد زیادی ویژگی هستند و نیاز به مدلی با دقت بالا و مقاومت در برابر برازش بیش‌اندازه باشد. این روش معمولاً به خوبی با داده‌های پیوسته عمل می‌کند و می‌تواند به عنوان یکی از مدل‌های پیش‌بینی محبوب و قوی در مسائل رگرسیون استفاده شود.

این تنها چند مثال از مدل‌های رگرسیون هستند و بسته به مسئله‌ی مورد نظر مدل‌های دیگری نیز وجود دارند

تحلیل و توضیح کدها مربوط به مدل های رگرسیون :

```
X = df_final.drop('Price',axis=1)

y = df_final['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=101)


model= LinearRegression()

model.fit(X_train , y_train )

pd.DataFrame(model.coef_, X.columns, columns=['Coeficient'])
```

الگوریتمی که در این کد استفاده شده است الگوریتم رگرسیون خطی است. این الگوریتم برای پیش‌بینی یک متغیر پیوسته (مانند قیمت یک خانه توی پروژه ما) بر اساس ویژگی‌های دیگر داده مورد استفاده قرار می‌گیرد.

در این کد، ابتدا داده‌ها از دیتاست df_final بارگیری شده و ویژگی‌های ورودی (X) و متغیر پیش‌بینی (y) جدا می‌شوند. ویژگی‌های ورودی شامل تمامی ویژگی‌های داده‌ها می‌باشد به جز متغیر پیش‌بینی (قیمت خانه) که در متغیر y قرار می‌گیرد.

سپس داده‌های ورودی و خروجی به دو مجموعه آموزش و آزمون تقسیم می‌شوند. برای این کار از تابع train_test_split از sklearn استفاده کردیم. این تابع با توجه به مقدار test_size ، داده‌ها را به دو بخش آموزش و آزمون تقسیم می‌کند. مقدار random_state هم برای تعیین شرایط تصادفی مورد استفاده قرار می‌گیرد. سپس یک مدل رگرسیون خطی ایجاد می‌شود و یک شیء از کلاس LinearRegression ایجاد می‌شود و با استفاده از داده‌های آموزش، مدل آموزش داده می‌شود. در اینجا، متد fit برای آموزش مدل استفاده شده است.

در نهایت ضرایب مدل برای هر ویژگی ورودی در یک DataFrame نمایش داده می‌شود. این ضرایب نشان می‌دهند که هر ویژگی ورودی چقدر بر روی قیمت خانه تأثیر دارد.

این ضرایب مدل رگرسیون خطی را برای هر یک از ویژگی‌های ورودی نشان داده میشوند . ضرایب نشان دهنده این است که هر ویژگی چقدر بر روی متغیر پیش‌بینی (در اینجا قیمت خانه) تأثیر دارد.

این دستور ما سه آرگومان دارد:

model.coef_ -این آرگومان ضرایب مدل را استخراج می‌کند

X.columns -این آرگومان نام ویژگی‌های ورودی را ارائه می‌دهد

columns=['Coeficient'] -این آرگومان نام ستون‌های DataFrame را تعیین می‌کند که در اینجا Coeficient نام ستون است که مربوط به ضرایب است.

به این ترتیب، این دستور ما DataFrameرا ایجاد می‌کند که نام ویژگی‌ها را به عنوان نام سطرها و ضرایب متناظر را به عنوان مقادیر نشان می‌دهد

	Coeficient
Area	70813456
Room	345948997
Parking	-852827344
Warehouse	-142953896
Elevator	-71547100
...	...
Yousef Abad	1832475653
Zafar	2675211141
Zaferanieh	10195126348
Zargandeh	75105006
Zibadasht	-542110080

197 rows × 1 columns

به طور کلی اعداد بزرگ مثبت نشان دهنده این است که ویژگی مرتبط با آن ضریب تأثیر مثبت و معناداری بر روی قیمت خانه دارد. به عبارت دیگر، افزایش در این ویژگی منجر به افزایش قیمت خانه می‌شود. از سوی دیگر، اعداد منفی نشان دهنده تأثیر منفی و معناداری از ویژگی مرتبط بر روی قیمت خانه هستند. به عبارت دیگر، افزایش در این ویژگی منجر به کاهش قیمت خانه می‌شود

```
# from sklearn import metrics

y_pred=model.predict(X_test)

MAE= metrics.mean_absolute_error(y_test, y_pred)

MSE= metrics.mean_squared_error(y_test, y_pred)

RMSE=np.sqrt(MSE)

pd.DataFrame([MAE, MSE, RMSE], index=['MAE', 'MSE', 'RMSE'],
columns=['Metrics'])
```

این کد من برای محاسبه سه معیار عملکرد مدل استفاده می‌شود: خطای میانگین مطلق(MAE) ، خطای میانگین مربعات (MSE) و ریشه میانگین مربعات خطا (RMSE)

این معیارها به ترتیب نشان دهنده میزان خطا در پیش‌بینی‌های مدل، میزان خطا برابر با میانگین مقدار مطلق خطاها، میانگین مقدار مربع خطاها و ریشه میانگین مربعات خطاها است.

در کد من ابتدا پیش‌بینی‌های مدل بر روی داده‌های تست انجام دادم و سپس مقادیرMAE ، MSEو RMSE برای این پیش‌بینی‌ها محاسبه می‌شود. در نهایت، این مقادیر در یک DataFrameقرار داده می‌شوند.

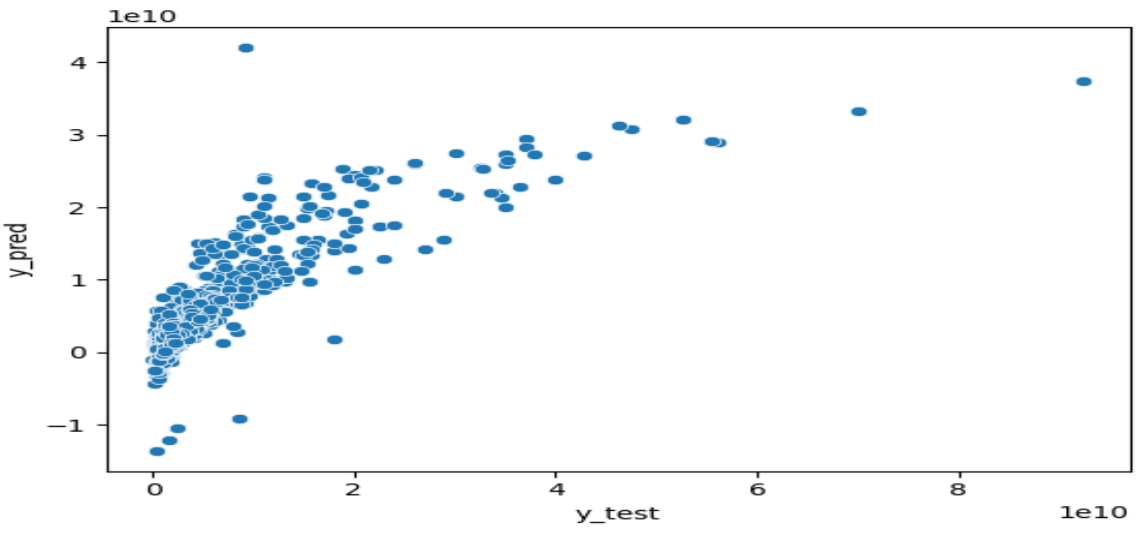
	Metrics
MAE	1972264179
MSE	16951998556154834944
RMSE	4117280481

```
df['Price'].mean()

test_residuals=y_test-y_pred

sns.scatterplot(x=y_test, y=y_pred)

plt.xlabel(' y_test ')

plt.ylabel(' y_pred ')
```

نمودار پراکنش از واقعیت و پیش‌بینی مدل برای قیمت‌ها استفاده می‌کند همچنین با استفاده از توابع plt.xlabel وplt.ylabel برچسب‌های محور x و y تنظیم می‌شوند

در این نمودار، محور x نشان دهنده قیمت‌های واقعی و محور y نشان دهنده پیش‌بینی‌های مدل است. هر نقطه در این نمودار نشان دهنده یک نمونه از داده‌های تست است. اگر پیش‌بینی مدل دقیقاً با قیمت واقعی مطابقت داشته باشد، نقاط باید به صورت خطی با شیب یک نسبت به خط مرجع(خط y=x)قرار گیرند.

ما میانگین قیمت‌ها را با df['Price'].mean() محاسبه کردیم حالا می‌توانید مقایسه‌ای از میانگین قیمت‌ها با پیش‌بینی‌های مدل داشته باشیم. این کد دستوری که ما نوشتیم می‌تواند به من کمک کند تا ببینیم که مدلم چقدر به طور میانگین از قیمت‌های واقعی فاصله دارد.



این نمودار پراکندگی است که یک نوع نمودار آماری است که برای نمایش مقادیر معمولاً دو متغیر برای یک مجموعه داده استفاده می‌شود

محور افقی با عنوان "y_test" و محور عمودی با عنوان "y_pred" مشخص کردیم ، که نشان می‌دهد که این نمودار مقادیر پیش‌بینی شده (y_pred) را با مقادیر واقعی، مشاهده شده یا آزمایشی (y_test) مقایسه می‌کند. نشانه 1e10روی هر محور نشان می‌دهد که مقادیر داده‌ها در مرتبه ۱۰^۱۰ (یا میلیارد) هستند، به این معنی که مقادیر عددی روی محورها از ۰ تا پتانسیل ۱۰ میلیارد ممکن است

نمودار پراکندگی یک روند نشان می‌دهد که با افزایش مقادیرy_test ، مقادیر y_pred نیز افزایش می‌یابد، که نشان می‌دهد که بین دو متغیر درجه‌ای از همبستگی وجود دارد. با این حال، رابطه کاملاً مطابقت ندارد، زیرا نقاط همه بر روی یک خط راست قرار نمی‌گیرند. نقاط بیشتر در نزدیکی مبدا تراکم دارند و به نظر می‌رسد که هنگامی که از مبدا دورتر می‌روند، پراکندگی دارند

این نوع نمودار معمولاً برای ارزیابی عملکرد یک مدل پیش‌بینی مانند یک مدل رگرسیون استفاده می‌شود. هر چه نقاط نزدیک‌تر به یک خط ۴۵ درجه(جایی که y_pred برابر y_test است) باشند پیش‌بینی‌های مدل بهتر خواهد بود. در این مورد، پیش‌بینی‌ها برای مقادیر کمتر منطقی به نظر می‌رسند اما برای مقادیر بالاتر، واریانس بیشتری نشان می‌دهند. برخی از نقاط به عنوان نقاط ناهنجاری نیز ظاهر می‌شوند، به ویژه آن‌هایی که دورتر از خوشه اصلی نقاط هستند.

به طور خلاصه من میتوانم بگویم این نمودار نمایش تصویری از همبستگی بین مقادیر پیش‌بینی و واقعی در یک مجموعه داده است و نشان می‌دهد که مدل مبنایی عملکرد پیش‌بینی مناسبی دارد اما کاملاً مطابقت ندارد.

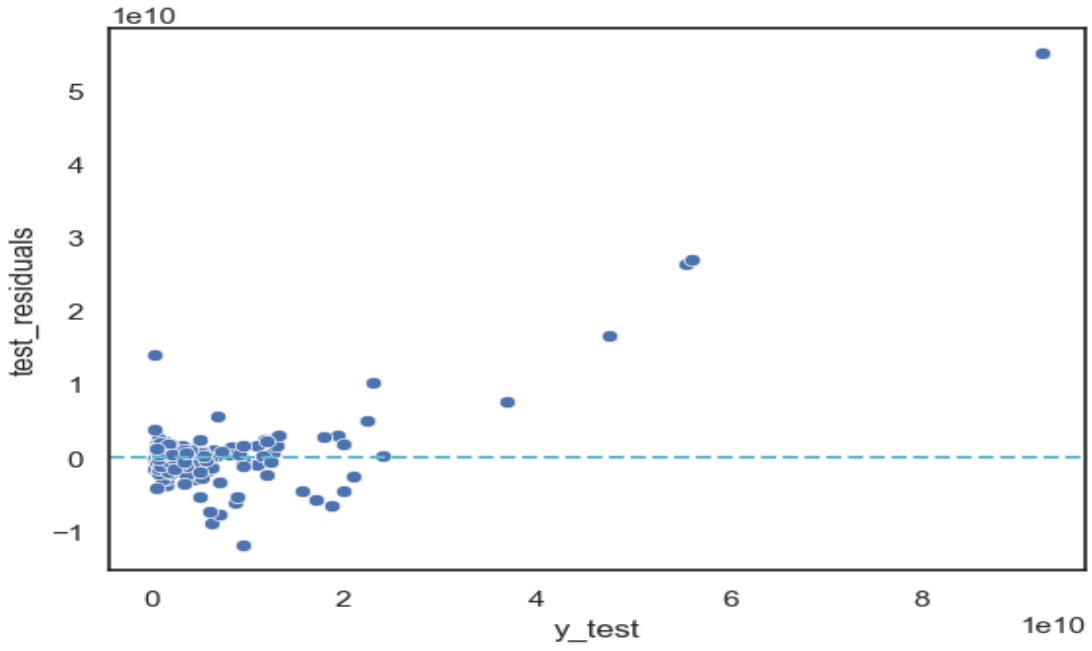
```
sns.scatterplot(x=y_test, y=test_residuals)
plt.axhline(y=0, color='c', ls='--')
plt.xlabel('y_test')
plt.ylabel('test_residuals')
```

در این کد من از تابع scatterplot برای رسم نمودار پراکندگی استفاده کردم . متغیر y_test به عنوان محور افقی و متغیر test_residuals به عنوان محور عمودی برای نمودار انتخاب شده است

همچنین با استفاده از تابعplt.axhline ، یک خط افقی به رنگ cyan (c) و با الگوی خط چین (ls='--')رسم می‌شود که در ارتفاع صفر قرار دارد. این خط معمولاً برای نشان دادن مقدار صفر (یا معادل صفر) برای مقادیر باقیمانده (residuals) استفاده می‌شود

در نهایت، با استفاده از توابع xlabel وylabel برچسب‌های مناسب برای محورهای x و y تنظیم کردم

با اجرای این کد، من یک نمودار پراکندگی با محورهای مناسب و یک خط افقی در ارتفاع صفر برای مقادیر باقیمانده رسم خواهد شد.



این نمودار پراکندگی ما نشان دهنده رابطه بین دو متغیر است، که یکی به عنوان y_test در محور افقی و دیگری به عنوان test_residuals در محور عمودی نشان داده شده است هر دو محور با استفاده از نمایش علمی مقیاس دهی شده اند به عنوان مثال 1e10نشان دهنده1 times 10^10 است که برابر با ۱۰ میلیارد است **نکته:Times =همون ضرب هستش**

از این نمودار زیر من نتیجه میگیرم که :

۱ .مقادیر y_test از تقریباً ۰ تا1 times 10^10 متغیر است و اکثر نقاط داده در نزدیکی ۰ خوشه‌بندی شده‌اند

۲ test_residualsنشان دهنده تفاوت‌های بین مقادیر مشاهده شده و مقادیر پیش‌بینی شده از یک مدل است. در صورتی که مدل دقیق باشد، انتظار دارید که مابقی باقیمانده به صورت تصادفی در اطراف خط افقی در ۰ (یعنی بدون خطای سیستماتیک) پخش شوند

۳ .بیشتر باقیمانده‌ها در اطراف ۰ خوشه‌بندی شده‌اند که نشان می‌دهد برای مقادیر y_test نزدیک به ۰، پیش‌بینی‌های مدل بسیار دقیق هستند

۴ یک الگوی قابل توجه وجود دارد که نشان می‌دهد با افزایش مقادیرy_test باقیمانده‌ها بیشتر مثبت می‌شوند. این نشان می‌دهد که مدل تمایل دارد که به طور نسبی مقادیر واقعی را کمتر از حد واقعی تخمین بزند همچنین چند باقیمانده با مقادیر قابل توجه بزرگتر وجود دارد که نشان دهنده نقاط پرت یا مواردی است که پیش‌بینی‌های مدل به طور قابل توجهی از مقادیر واقعی انحراف کرده‌اند

۵ .پخش باقیمانده‌ها به نظر می‌رسد با افزایش مقادیر `y_test` افزایش می‌یابد این می‌تواند به معنای هتروسکداسیته(اینو براساس تحقیقی که کردم گفتم) باشد به این معنی که واریانس باقیمانده‌ها در سراسر دامنه y_test ثابت نیس

این نوع نمودار معمولاً در تجزیه و تحلیل رگرسیون برای ارزیابی مدل و بررسی الگوهای باقیمانده‌ها که ممکن است مشکلاتی مانند غیرخطی بودن، نقاط پرت یا مسائل واریانس را نشان دهد استفاده می‌شود.

```
msk = np.random.rand (len (df)) < 0.8

train = df[msk]

test = df[~msk]

regr = linear_model.LinearRegression()

x = np.asanyarray(train[['Area' , 'Room' , 'Parking' , 'Warehouse' , 'Elevator']])

y = np.asanyarray(train[['Price']])

regr.fit (x, y)

print ('Coefficients: ', regr.coef_)

print ('Intercept: ', regr.intercept_)
```

این قطعه کد ما یک مدل رگرسیون خطی را برای پیش‌بینی قیمت خانه‌ها بر اساس ویژگی‌های مختلف مانند مساحت، تعداد اتاق‌ها، تعداد پارکینگ، وجود انبار و وجود آسانسور ایجاد می‌کند. ابتدا داده‌ها به دو بخش آموزش و آزمون تقسیم می‌شوند، سپس یک مدل رگرسیون خطی روی داده‌های آموزشی آموزش داده می‌شود

در این قطعه کد `np.random.rand(len(df)) < 0.8` برای ایجاد یک ماسک برای جدا کردن داده‌ها به صورت تصادفی به کاربرده ایم سپس داده‌ها به دو بخش آموزشی و آزمون تقسیم کرده ایم. سپس یک مدل رگرسیون خطی ایجاد می‌شود و برازش داده‌های آموزشی انجام می‌شود.

ضرایب و عرض نازل مدل با استفاده از `` `regr.coef_` `` و `` `regr.intercept_` `` چاپ می‌شوند

در صورتی که اگر بخواهیم پیش‌بینی‌های مدل را بر روی داده‌های آزمون انجام دهیم باید مراحل پیش‌بینی را انجام دهید و سپس معیارهای ارزیابی مانند میانگین مربعات خطا یا R-squared را بررسی کنیم

Coefficients: [[6.84985699e+07 1.73422861e+09 -6.15555284e+07 1.37586654e+09

9.14458545e+08]]

Intercept: [-7.51949352e+09]

نتیجه کد صفحه قبل

این ضرایب و عرض نازل به دست آمده از مدل رگرسیون خطی است که بر روی داده‌های آموزشی برازش داده شده است. این ضرایب نشان دهنده تأثیر هر ویژگی بر متغیر پاسخ (در اینجا قیمت خانه‌ها) است. هر عدد در ضرایب نشان دهنده این است که با افزایش یک واحد در ویژگی مرتبط، چقدر متغیر پاسخ تغییر می‌کند.

Times=همون ضرب است

برای مثال، ضریب مربوط به "ویژگی ۱" حدود (6.85 times 10^7) است. این بدان معناست که با افزایش یک واحد در "ویژگی ۱"، متغیر پاسخ حدود (6.85 times 10^7) واحد تغییر می‌کند عرض نازل (Intercept) حدود (-7.52 times 10^9) است. این مقدار نشان دهنده مقدار متغیر پاسخ (در اینجا قیمت خانه) در صورتی است که تمام ویژگی‌ها صفر باشند

```
y_hat= regr.predict(test[['Area' , 'Room' , 'Parking' , 'Warehouse' , 'Elevator']])
x = np.asanyarray(test[['Area' , 'Room' , 'Parking' , 'Warehouse' , 'Elevator']])
y = np.asanyarray(test[['Price']])
print("Residual sum of squares: %.2f"
    % np.mean((y_hat - y) ** 2))
print('Variance score: %.2f' % regr.score(x, y))
```

اول که مقادیر پیش‌بینی شده توسط مدل بر روی داده‌های تست محاسبه می‌شود. سپس، مقادیر واقعی از داده‌های تست گرفته می‌شود. سپس مجذور میانگین باقی‌مانده (residual sum of squares)محاسبه می‌شود و به عنوان یک معیار برای اندازه‌گیری دقت مدل استفاده می‌شود.

همچنین، امتیاز واریانس (variance score) نیز محاسبه می‌شود که نشان دهنده میزان توضیح داده شده توسط مدل است. امتیاز واریانس بین ۰ و ۱ است، که ۱ نشان دهنده پیش‌بینی کامل مدل و ۰ نشان دهنده عدم توانایی مدل در توضیح دادن داده‌ها است.

Residual sum of squares: 325900270851559628800.00

Variance score: 0.56

نتیجه کد صفحه قبل

```
X = df_final.drop(columns = 'Price')

y = df_final['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

print(f"shape of x train: {X_train.shape}")

print(f"shape of y train: {y_train.shape}")

print(f"shape of x test: {X_test.shape}")

print(f"shape of y train: {y_test.shape}")
```

این کد ما یک مدل رگرسیون را برای پیش‌بینی قیمت مسکونی از داده‌های موجود ساخته و آموزش می‌دهد. ابتدا، داده‌ها به دو بخش آموزش و آزمون تقسیم می‌شوند. سپس شکل (اندازه) داده‌های آموزش و آزمون چاپ می‌شود تا اطمینان حاصل شود که تقسیم داده‌ها به درستی انجام شده است.

به عنوان مثال، اندازه داده‌های آموزش (X_train) چاپ می‌شود تا مشخص شود که چند ردیف و چند ستون داده در این بخش وجود دارد همچنین، اندازه برچسب‌های آموزش (y_train) و داده‌ها و برچسب‌های آزمون (X_test و y_test) نیز چاپ می‌شود. این اطلاعات برای اطمینان حاصل کردن از صحت تقسیم داده‌ها استفاده می‌شود.

```
lr = LinearRegression(n_jobs = -1)
parameters = {}
start = time.time()
grid = GridSearchCV(lr,
                    param_grid = parameters,
                    refit = True,
                    cv = KFold(shuffle = True, random_state = 1),
                    n_jobs = -1)
grid_fit = grid.fit(X_train, y_train)
y_train_pred = grid_fit.predict(X_train)
y_pred = grid_fit.predict(X_test)
lr_train_score = grid_fit.score(X_train, y_train)
lr_test_score = grid_fit.score(X_test, y_test)
lr_RMSE = np.sqrt(mean_squared_error(y_test, y_pred))
model_name = str(lr).split('(')[0]
end = time.time()
print(f"The best parameters for {model_name} model is:
{grid_fit.best_params_}")
print("--" * 10)
print(f"(R2 score) in the training set is {lr_train_score:0.2%} for
{model_name} model.")
print(f"(R2 score) in the testing set is {lr_test_score:0.2%} for
{model_name} model.")
```

print(f"RMSE is {lr_RMSE:,} for {model_name} model.")

print("--" * 10)

print(f"Runtime of the program is: {end - start:0.2f}")

ما در اینجا از الگوریتم GridSearchCV برای جستجوی بهترین پارامترها برای یک مدل استفاده کردیم این الگوریتم یک روش جستجوی متقابل گرید است که تمامی ترکیب‌های ممکن از مقادیر پارامترهای مدل را بررسی می‌کند و بهترین مقادیر را بر اساس یک معیار عملکرد مشخص انتخاب می‌کند.

در کد ارائه کردم یک مدل رگرسیون خطی (LinearRegression) تعریف شده و سپس یک دیکشنری خالی به عنوان پارامترها برای جستجو تعیین می‌شود. سپس الگوریتم GridSearchCVبا استفاده از این مدل و پارامترها، بهترین مدل را بر اساس معیار عملکردی که تعیین شده است مثلاً (R2 score) انتخاب می‌کند.

سپس مدل بهترین پارامترها را با داده‌های آموزش (X_train و y_train) آموزش می‌دهد. سپس این مدل بر روی داده‌های آموزش و آزمون ارزیابی می‌شود و معیارهای عملکرد مدل مانند R2 score و RMSE محاسبه می‌شود. در نهایت، این معیارها همراه با بهترین پارامترها و زمان اجرای برنامه چاپ می‌شوند. **این اطلاعات برای ارزیابی عملکرد مدل و انتخاب بهترین مدل برای پیش‌بینی استفاده می‌شوند.**

در اینجا lr مدل LinearRegression است که قبلاً تعریف شده است param_grid . پارامترهایی است که برای جستجو در نظر گرفته می‌شود refit=True .بدان معنی است که پس از یافتن بهترین پارامترها، مدل با این پارامترها بر روی کل داده‌های آموزش مجدداً آموزش داده شود cv یک شیء KFold است که برای انجام اعتبارسنجی متقابل (cross-validation) استفاده می‌شود. در اینجا از KFold با shuffle=True و random_state=1 برای اعتبارسنجی متقابل استفاده می‌شود n_jobs=-1 .به GridSearchCV می‌گوید که تمام منابع موجود را برای اجرای موازی استفاده کند.

این شیء GridSearchCV سپس برای جستجوی بهترین پارامترها و انجام اعتبارسنجی متقابل بر روی داده‌ها استفاده می‌شود

The best parameters for LinearRegression model is: {}

(R2 score) in the training set is 73.89% for LinearRegression model.

(R2 score) in the testing set is 75.59% for LinearRegression model.

RMSE is 4,136,170,584.251215 for LinearRegression model.

Runtime of the program is: 7.16

نتیجه کد۲ صفحه قبل

این خروجی نشان می‌دهد که بهترین پارامترها برای مدل LinearRegression پیدا شده‌اند. امتیاز R2 برای داده‌های آموزش و آزمون نیز گزارش شده است که به ترتیب برابر با ۷۳.۸۹٪ و ۷۵.۵۹٪ است. همچنین مقدار RMSE نیز برای مدل LinearRegression گزارش شده است که برابر با ۴,۱۳۶,۱۷۰,۵۸۴.۲۵۱۲۱۵ است.

در نهایت، زمان اجرای برنامه نیز گزارش شده است که برای این برنامه ۷.۱۶ ثانیه بوده است. این اطلاعات به ما کمک می‌کنند تا عملکرد مدل را در داده‌های آموزش و آزمون ارزیابی کنیم و بر اساس این ارزیابی‌ها، مدل LinearRegression را به عنوان بهترین مدل برای پیش‌بینی انتخاب کنیم

---

ridge = Ridge(random_state = 1)

param_ridge = {'alpha': [0.001, 0.01, 0.1, 1, 10]}

```
start = time.time()


grid = GridSearchCV(ridge,
                    param_grid = param_ridge,
                    refit = True,
                    cv = KFold(shuffle = True, random_state = 1),
                   n_jobs = -1)
```

این کد یک مدل Ridge را با استفاده از جستجوی خطی بهترین پارامترها (GridSearchCV) ایجاد می‌کند. پس از آموزش مدل با بهترین پارامترها، عملکرد مدل بر روی داده‌های آموزش و آزمون ارزیابی شده و نتایج گزارش می‌شود. همچنین زمان اجرای برنامه نیز گزارش می‌شود.

نتیجه

```
The best parameters for Ridge model is: {'alpha': 1}
----------
(R2 score) in the training set is 73.09% for Ridge model.
(R2 score) in the testing set is 75.94% for Ridge model.
RMSE is 4,106,340,011.964132 for Ridge model.
----------
Runtime of the program is: 1.04
```

نتایج ارائه شده نشان می‌دهد که مدل Ridge با پارامتر بهینه‌ی {'alpha': 1} بر روی داده‌های آموزش و آزمون ارزیابی شده است. امتیاز R2 برای مدل بر روی داده‌های آموزش ۷۳.۰۹٪ و بر روی داده‌های آزمون ۷۵.۹۴٪ است. همچنین مقدار RMSE برابر با ۴,۱۰۶,۳۴۰,۰۱۱.۹۶۴۱۳۲ گزارش شده است. زمان اجرای برنامه نیز ۱.۰۴ ثانیه بوده است.

بر اساس این نتایج، می‌توانیم بگوییم که مدل Ridge با پارامتر بهینه {'alpha': 1} نسبت به مدل‌های دیگر، عملکرد بهتری داشته است. امتیاز R2 بر روی داده‌های آموزش و آزمون نسبتاً نزدیک به یک است که نشان می‌دهد مدل خوبی براحتی می‌تواند الگوهای داده را یاد بگیرد. با این حال مقدار بسیار بزرگ RMSE نشان می‌دهد که مدل هنوز نتوانسته است به خوبی داده‌ها را پیش‌بینی کند و نیاز به بهبود دارد.

براساس همین کدها با پارمترهای متفاوت برای زدن مدل های مختلف مثل Lasso وDecisionTreeRegressorوRandomForestRegressor را انجام دادم

این کار به من کمک کرد تا مدل‌های Lasso، DecisionTreeRegressor و RandomForestRegressor را با استفاده از بهترین پارامترها ایجاد کرده و عملکرد آن‌ها را بر روی داده‌های آموزش و آزمون ارزیابی کنید. این اقدام می‌تواند به شما کمک کند تا بهترین مدل برای مسئله خود را انتخاب کنم.

و سپس.....

```
models_score = pd.DataFrame({'Training score': [lr_train_score, ridge_train_score, lasso_train_score, dtr_train_score, rfr_train_score],
                             'Testing score': [lr_test_score, ridge_test_score, lasso_test_score, dtr_test_score, rfr_test_score],
                             'RMSE': [lr_RMSE, ridge_RMSE, lasso_RMSE, dtr_RMSE, rfr_RMSE]},
                             index = ['LinearRegression', 'Ridge', 'Lasso', 'DecisionTreeRegressor', 'RandomForestRegressor'])
models_score
```

این کد من یک DataFrame با نام models_score ایجاد می‌کند که شامل سه ستون است : 'Training score'، 'Testing score'و .'RMSE' هر ستون دارای مقادیر مربوط به امتیاز آموزش، امتیاز آزمون و RMSE برای پنج مدل مختلف استLinear Regression :، Ridge، Lasso، Decision Tree Regressorو.Random Forest Regressor

با استفاده از دیکشنری اطلاعات مربوط به امتیازها و RMSE برای هر مدل و همچنین اندیس‌های مربوط به نام مدل‌ها ساخته شده است. سپس با فراخوانی models_score، اطلاعات مربوط به امتیازها و RMSE برای هر مدل نمایش داده می‌شود.


[103] ✓ 0.0s

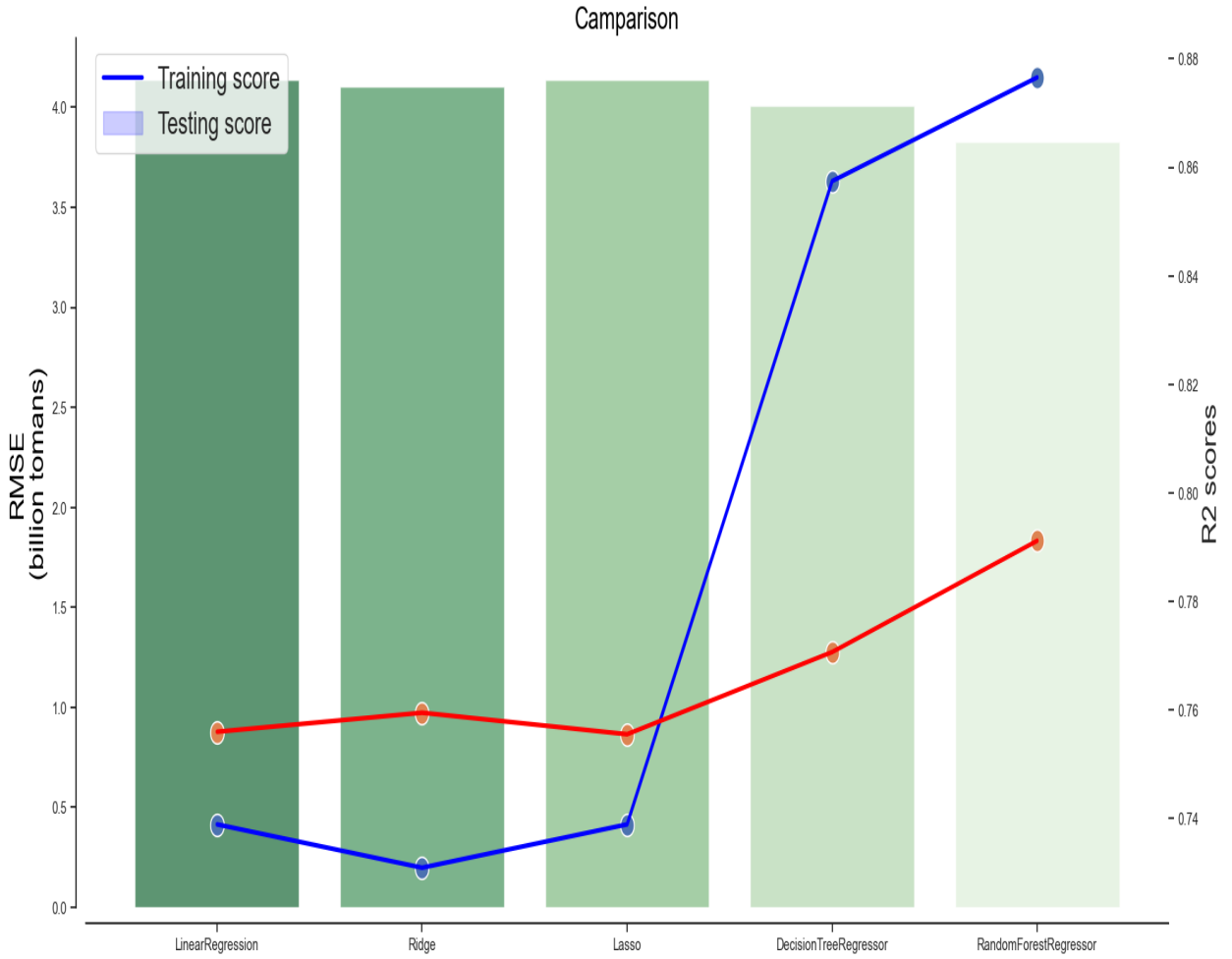
	Training score	Testing score	RMSE
LinearRegression	1	1	4136170584
Ridge	1	1	4106340012
Lasso	1	1	4140043072
DecisionTreeRegressor	1	1	4009679949
RandomForestRegressor	1	1	3826250288

مقادیر ۱ برای امتیازهای آموزش و آزمون نشان می‌دهد که مدل‌ها برای داده‌های آموزش و آزمون خود به خوبی عمل کرده‌اند و داده‌های آموزش را به خوبی یادگرفته‌اند. اما برای مقدار RMSE، مدل RandomForestRegressor با مقدار ۳۸۲۶۲۵۰۲۸۸ بهترین عملکرد را داشته است. مقدار RMSE برای هر مدل نشان دهنده میزان خطا در پیش‌بینی مقادیر واقعی است، بنابراین مدلی که مقدار RMSE کمتری داشته باشد، بهترین عملکرد را دارد. در اینجا، مدل RandomForestRegressor با مقدار RMSE کمتر از سایر مدل‌ها برای پیش‌بینی بهتری از داده‌ها استفاده شده است.

در نتیجه با توجه به امتیازهای آموزش و آزمون و مقادیر RMSE، مدل RandomForestRegressor بهترین عملکرد را دارد.

```
fig, ax = plt.subplots(figsize=(20,10))

sns.set(style='white')

ax.set_title("Camparison", fontsize = 20)

ax = sns.barplot(x = list(models_score.index), y =
models_score['RMSE']/1000000000, alpha = 0.7, palette='Greens_r')

ax.set_ylabel("RMSE\n(billion tomans)", fontsize = 20)

sec_ax = ax.twinx()

sec_ax = sns.lineplot(x = list(models_score.index), y =
models_score['Training score'], linewidth = 3, color = 'blue')

sec_ax = sns.scatterplot(x = list(models_score.index), y =
models_score['Training score'], s = 200)

sec_ax = sns.lineplot(x = list(models_score.index), y =
models_score['Testing score'], linewidth = 3, color = 'red')

sec_ax = sns.scatterplot(x = list(models_score.index), y =
models_score['Testing score'], s = 200)

sec_ax.set_ylabel("R2 scores", fontsize = 20)

sec_ax.legend(labels = ['Training score', 'Testing score'], fontsize = 20)

sns.despine(offset = 10)

plt.show()
```

این کد به صورت گرافیکی یک مقایسه بین مدل‌های مختلف را ارائه می‌دهد. ابتدا یک نمودار نواری از RMSE برای هر مدل رسم می‌شود که به واحد میلیارد تومان است. سپس دو نمودار خطی و نقطه‌ای برای امتیازهای R2 برای آموزش و آزمون نیز رسم می‌شود. این نمودارها به صورت موازی دو مقیاس را نشان می‌دهند، به طوری که مقادیر RMSE را بر روی محور چپ و امتیازهای R2 را بر روی محور راست نمایش می‌دهند.



یک نمودار دو محوری است که عملکرد مدل‌های یادگیری ماشین مختلف که ما روشون کار کردیم را بر اساس دو معیار، یعنی RMSE (خطای میانگین مربعات متغیر) و امتیازR^2 ، مقایسه می‌کند. این نمودار با عنوان "مقایسه" است و یک تصویرسازی ارائه می‌دهد برای ارزیابی عملکرد پنج مدل رگرسیون:LinearRegression ، Ridge، Lasso، DecisionTreeRegressorو RandomForestRegressor. که ما روی انها کار کردیم

در محور y چپ، مقادیر RMSE به میلیاردها (تا یک رقم اعشار) نشان داده شده‌اند و در محور y راست، ما امتیازهای R^2 را داریم. مقادیر RMSE توسط نوارها نشان داده شده‌اند، در حالی که امتیازهای R^2 توسط خطوط نمایش داده شده‌اند. هر مدل یک نوار و دو نقطه برای امتیازهای آموزش (خط آبی) و آزمون (خط قرمز) دارد.

توضیحاتی در مورد این نمودار:

۱ .مدل‌هایLinearRegression ، Ridgeو Lasso عملکرد کم‌خطا را در هر دو امتیاز آموزش و آزمون نشان می‌دهند، که نشان می‌دهد که این مدل‌ها به طور مشابه عمل می‌کنند و دقت پیش‌بینی بهتری نسبت به دو مدل دیگر دارند. به نظر می‌رسد که مدل Lasso مقداری بیشتری از RMSE نسبت به مدل‌های LinearRegression و Ridge دارد.

۲ .مدل‌های DecisionTreeRegressor و RandomForestRegressor مقادیر RMSE بسیار بالاتری دارند که نشان دهنده عملکرد بدتر در معیار داده شده است.

۳ .از نظر امتیازهای R^2 که نشان‌دهنده نسبت واریانس متغیر وابسته است که قابل پیش‌بینی از متغیرهای مستقل است، مشخص است که RandomForestRegressor بالاترین امتیازها را برای هر دو آموزش و آزمون دارد که نشان می‌دهد که این مدل قادر است تا واریانس متغیر هدف را بهتر از مدل‌های دیگر توضیح دهد، البته علی‌رغم مقادیر بالاتر RMSE آن.

۴ .مدل‌های LinearRegression و Ridge امتیازهای مشابهی برای هر دو آموزش و آزمون دارند، با امتیازهایی کمی بالاتر از مدلLasso

۵ .به نظر می‌رسد که مدل DecisionTreeRegressor امتیاز بالایی برای داده‌های آموزش دارد اما امتیاز کمتری برای داده‌های آزمون دارد، که ممکن است نشانه بیش‌برازش بودن باشد.

به طور کلی، در حالی که RandomForestRegressor بهترین امتیازهای R^2 را دارد، مقادیر RMSE آن بالاترین است، که به تضاد در عملکرد مدل بین این دو معیار اشاره می‌کند. مدل‌های LinearRegression و Ridge عملکرد متوازنی با RMSE کم و امتیازهای مناسب R^2 ارائه می‌دهند. ممکن است مدل DecisionTreeRegressor مشکلات بیش‌برازش داشته باشد که نشان‌دهنده امتیازهای بالای آموزش و امتیازهای کمتر آزمون است، و مدل Lasso امتیازهای معیارها را کمی پایین‌تر از مدل‌های LinearRegression و Ridge دارد.

در نتیجه این نمودار هم باز بهترین مدل به نظر می‌رسد RandomForestRegressor باشد. این مدل برای هر دو مجموعه داده آموزش و آزمون بالاترین امتیاز R^2 را دارد که نشان می‌دهد که درصد بیشتری از واریانس متغیر هدف را توضیح می‌دهد. علاوه بیشتر RMSE (خطای میانگین مربعات متغیر) آن کمترین است میان مدل‌ها، که نشان می‌دهد که دقت پیش‌بینی آن با کمترین مقدار خطا است.

متشکرکه تا اینجا من را همرایم کردید 🙏

پایان