



دانشکده مهندسی مکانیک

گزارش تمرین دوم درس هوش مصنوعی

استاد درس:

دکتر شریعت پناهی

نام دانشجو:

علی شجاعی زاده

شماره دانشجویی:

۸۱۰۶۰۳۰۱۱

فهرست مطالب

مقدمه	۲
بخش اول: خواندن داده‌های موجود	۳
بخش دوم: مدیریت داده‌های پرت و داده‌های مفقود	۴
پرت کردن مقادیر برای داده‌های مفقود	۴
مدیریت داده‌های غیرعددی	۴
حذف داده‌های پرت	۵
بخش سوم: بررسی اطلاعات آماری دادگان	۶
بخش چهارم: ماتریس همبستگی و یافتن مؤثرترین ویژگی‌ها	۷
بخش پنجم: نمودارهای توزیع متقابل برای ویژگی‌های به‌دست آمده	۸
بخش ششم: انتخاب تعداد ویژگی‌ها با استفاده از SelectKBest	۲۳
بخش هفتم: تقسیم دادگان به بخش‌های آموزش و تست	۲۵
بخش هشتم: آموزش مدل‌ها	۲۶
بخش نهم: ارزیابی عملکرد	۲۷
خطای MSE	۲۷
خطای RMSE	۲۷
ضریب تعیین	۲۷
بخش دهم: موازنه واریانس و بایاس	۳۰
بایاس	۳۰
واریانس	۳۰
خطای غیرقابل کاهش	۳۰

امروزه با افزایش جمعیت و چالش تهیه مسکن با توجه به وضعیت اقتصادی خانوار، پیش‌بینی قیمت مسکن را به یکی از چالش‌های مهم در حوزه املاک و مستغلات تبدیل می‌کند. با استفاده از روش‌های یادگیری ماشین و تحلیل داده‌ها، می‌توان الگوهای پنهان در ویژگی‌های مختلف خانه‌ها را شناسایی و مدل‌هایی برای پیش‌بینی قیمت با دقت بالا ارائه داد. در این تمرین، با بهره‌گیری از مجموعه داده‌های مربوط به فروش خانه‌های مسکونی در آمریکا، به بررسی تأثیر ویژگی‌هایی مانند مساحت، سال ساخت، و کیفیت ساخت بر قیمت پرداخته‌ایم. با استفاده از مدل‌های رگرسیون خطی، لاسو^۱، ریدج^۲، و چندجمله‌ای پیش‌بینی را انجام داده‌ایم و هم‌چنین با ارزیابی معیارهای مختلف مانند خطای RMS^3 و ضریب تعیین (R^2)، به دنبال یافتن بهترین مدل برای این پیش‌بینی هستیم. این گزارش، مراحل پیش‌پردازش داده‌ها، انتخاب ویژگی‌های کلیدی، آموزش مدل‌ها، و تحلیل نتایج را شرح می‌دهد.

¹ Lasso Regression

² Ridge Regression

³ Root Mean Square Error

بخش اول: خواندن داده‌های موجود

در قسمت اول شروع به کار، لازم است مجموعه داده^۴ خود را که در قالب فایل CSV ذخیره شده‌است، بخوانیم تا بتوانیم عملیات‌های بعدی را بر روی آن انجام دهیم. در این تمرین، ما از کتابخانه Pandas برای مدیریت مجموعه داده خود استفاده می‌کنیم.

با استفاده از تابع `read_csv` و قرار دادن فایل مربوطه در کنار کد برنامه، مجموعه داده را وارد کد می‌کنیم و با استفاده از متد `info` اطلاعات مربوط به فایل خوانده‌شده را بررسی می‌کنیم تا از صحت درست خوانده‌شدن فایل اطمینان حاصل کنیم. در شکل ۱، اطلاعات مربوط به خوانش فایل مجموعه داده نمایش داده شده‌است.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 82 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Order                 2930 non-null   int64  
1   PID                  2930 non-null   int64  
2   MS SubClass           2930 non-null   int64  
3   MS Zoning             2930 non-null   object  
4   Lot Frontage          2440 non-null   float64 
5   Lot Area             2930 non-null   int64  
6   Street               2930 non-null   object  
7   Alley               198 non-null    object  
8   Lot Shape            2930 non-null   object  
9   Land Contour         2930 non-null   object  
10  Utilities            2930 non-null   object  
11  Lot Config           2930 non-null   object  
12  Land Slope           2930 non-null   object  
13  Neighborhood         2930 non-null   object  
14  Condition 1          2930 non-null   object  
15  Condition 2          2930 non-null   object  
16  Bldg Type            2930 non-null   object  
17  House Style          2930 non-null   object  
18  Overall Qual          2930 non-null   int64  
19  Overall Cond         2930 non-null   int64  
...
80  Sale Condition       2930 non-null   object  
81  SalePrice            2930 non-null   int64  
dtypes: float64(11), int64(28), object(43)
memory usage: 1.8+ MB
```

شکل ۱- اطلاعات مربوط به خوانش فایل مجموعه داده خانه‌ها

همان‌گونه که در شکل ۱ ملاحظه می‌گردد، این مجموعه داده دربردارنده ۸۱ ویژگی^۵ می‌باشد که در آن ۲۹۳۰ سطر از داده‌ها موجود است.

^۴ Dataset

^۵ Feature

بخش دوم: مدیریت داده‌های پرت^۶ و داده‌های مفقود^۷

در این بخش، پس از خواندن مجموعه داده، عملیات پیش‌پردازش آن را در ۳ مرحله انجام می‌دهیم:

- **مرحله اول:** پر کردن مقادیر برای داده‌های مفقود
- **مرحله دوم:** مدیریت داده‌های غیرعددی
- **مرحله سوم:** حذف داده‌های پرت

پر کردن مقادیر برای داده‌های مفقود

برای پر کردن داده‌های مفقود، دو نوع مختلف از دادگان را در نظر می‌گیریم: دادگان عددی که شامل انواع `int64` و `float64` هستند و دادگان غیرعددی که اصطلاحاً در کتابخانه `Pandas` با عنوان `Object` شناخته می‌شوند. برای پر کردن داده‌های مفقود که به صورت عددی هستند، مقدار میانه^۸ دادگان هر ویژگی را به دست می‌آوریم و آن را جایگزین مقدار مفقود مورد نظر می‌کنیم. در صورتی که با دادگان غیرعددی یا به عبارت دیگر، دادگان رشته‌ای^۹ مواجه باشیم، برای ویژگی متناظر، مُد متناظر با آن ویژگی را به دست می‌آوریم و جایگزین مقادیر مفقود می‌کنیم و به این ترتیب، مجموعه دادگانی که بعد از این مرحله در اختیار خواهیم داشت، عاری از داده‌های مفقود خواهد بود.

نهایتاً، با استفاده از عبارت `df.isnull().values.any()` باقی ماندن یا نماندن مقادیر مفقود را بررسی می‌کنیم. اگر این عبارت، مقدار غلط را به ما برگرداند، یعنی در مجموعه داده، داده مفقود وجود ندارد و در صورتی که درست باشد، یعنی در پر کردن کامل داده‌های مفقود موفق نبوده‌ایم.

مدیریت داده‌های غیرعددی

از آن‌جا که بخش قابل توجهی از دادگان موجود در مجموعه داده‌ای که در اختیار داریم، شامل دادگان رشته‌ای می‌باشد و مقادیر عددی ندارند، برای انجام این تمرین از روش `LabelEncoding` از کتابخانه `sklearn` بهره گرفته‌ایم. با این کار، به جای مقادیری که از نوع `String` هستند، با مقادیر عددی سروکار خواهیم داشت و امکان استفاده مدل‌های یادگیرنده از این ویژگی‌ها فراهم می‌شود.

⁶ Outliers

⁷ Missing Values

⁸ Median

⁹ Categorical

حذف داده‌های پرت

یکی از روش‌های مرسوم در حوزه آمار و علوم داده برای حذف داده‌های پرت، روش دامنه میان چارکی^{۱۰} می‌باشد. در این روش، چارک اول (صدک ۲۵) و چارک سوم (صدک ۷۵) را محاسبه می‌کنیم. با به‌دست آوردن این مقادیر، دامنه میان چارکی را که فاصله بین چارک اول و سوم است، حساب می‌کنیم. سپس، با استفاده از معیار ۱.۵ برابر IQR که یک قانون متداول آماری برای شناسایی داده‌های پرت است، حد بالا و پایین را به‌صورت زیر حساب می‌کنیم:

$$\text{lower band} = Q_1 - 1.5 \times IQR$$

$$\text{upper band} = Q_3 + 1.5 \times IQR$$

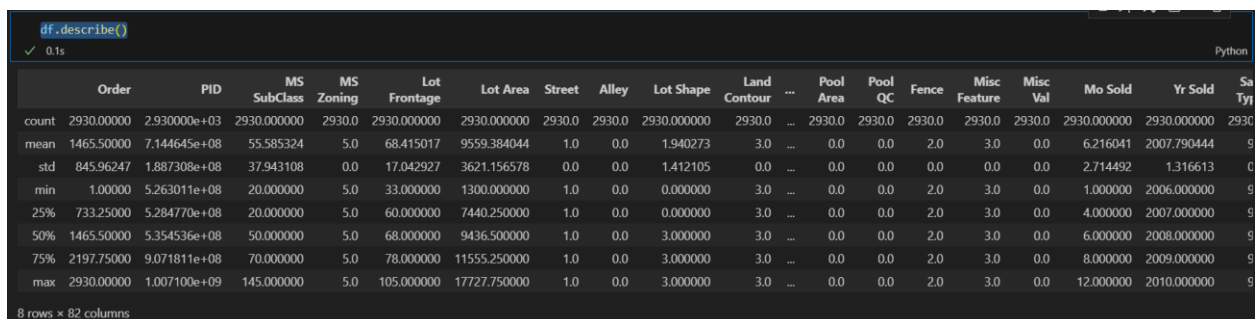
با به‌دست آوردن حد بالا و پایین، هر مقدار کم‌تر از حد پایین و بیش‌تر از حد بالا، به‌عنوان داده پرت در نظر گرفته می‌شود. برای داده‌هایی که مقدار آن‌ها از حد پایین کم‌تر است، بایستی مقدار حد پایین جایگزین شود و برای داده‌هایی که مقدار آن‌ها از حد بالا بیش‌تر است، بایستی مقدار حد بالا جایگزین شود تا داده‌های پرت اصلاح شوند. با استفاده از یک حلقه که در بین ویژگی‌های مختلف این مجموعه داده تکرار می‌شود، برای هر ویژگی این روش را اعمال می‌کنیم تا داده‌های پرت حذف شوند.

نهایتاً، با همین معیار وجود داده پرت را مجدداً بررسی می‌کنیم و در صورتی که داده پرت موجود نباشد، می‌توانیم کار خود را با این مجموعه داده که پالایش شده‌است، ادامه دهیم.

¹⁰ Interquartile Range (IQR)

بخش سوم: بررسی اطلاعات آماری دادگان

در این بخش، با توجه به متدهایی که کتابخانه Pandas در اختیار ما گذاشته است، می‌توانیم به اطلاعات آماری مجموعه دادگان پس از انجام پیش‌پردازش‌های لازم دسترسی داشته باشیم. این کار را با استفاده از دستور `df.describe()` می‌توانیم انجام دهیم که خروجی آن در شکل ۲ آورده شده است.



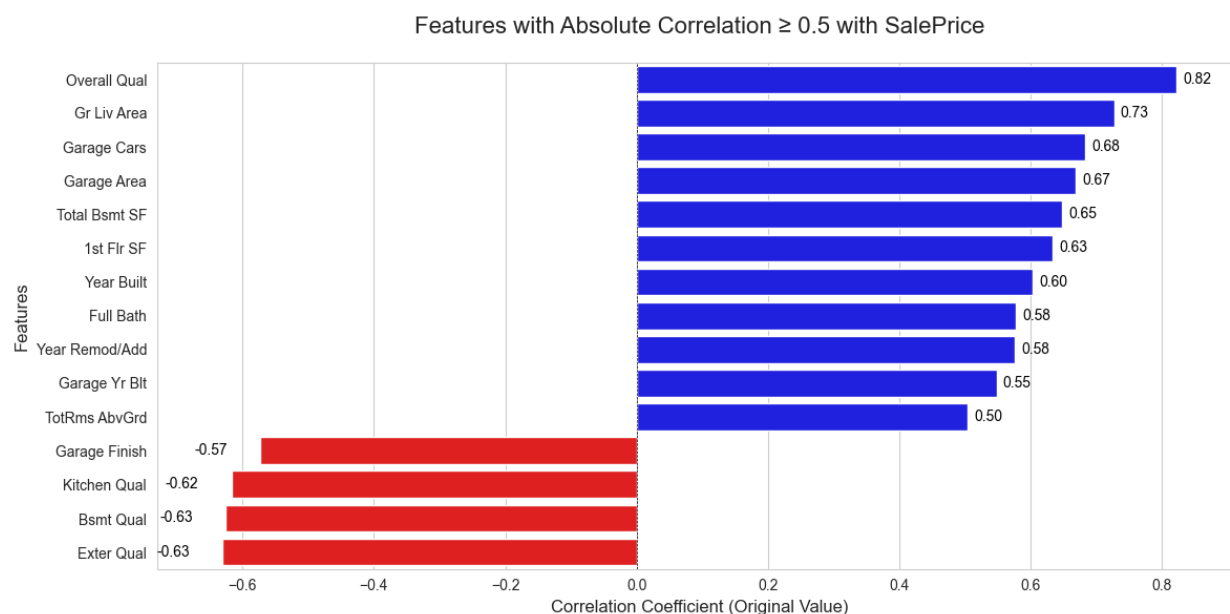
	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Pool Area	Pool QC	Fence	Misc Feature	Misc Val	Mo Sold	Yr Sold	Sale Type
count	2930.000000	2.930000e+03	2930.000000	2930.0	2930.000000	2930.000000	2930.0	2930.0	2930.000000	2930.0	2930.0	2930.0	2930.0	2930.0	2930.0	2930.000000	2930.000000	2930
mean	1465.500000	7.144645e+08	55.585324	5.0	68.415017	9559.384044	1.0	0.0	1.940273	3.0	0.0	0.0	2.0	3.0	0.0	6.216041	2007.790444	9
std	845.96247	1.887308e+08	37.943108	0.0	17.042927	3621.156578	0.0	0.0	1.412105	0.0	0.0	0.0	0.0	0.0	0.0	2.714492	1.316613	0
min	1.000000	5.263011e+08	20.000000	5.0	33.000000	1300.000000	1.0	0.0	0.000000	3.0	0.0	0.0	2.0	3.0	0.0	1.000000	2006.000000	9
25%	733.250000	5.284770e+08	20.000000	5.0	60.000000	7440.250000	1.0	0.0	0.000000	3.0	0.0	0.0	2.0	3.0	0.0	4.000000	2007.000000	9
50%	1465.500000	5.354536e+08	50.000000	5.0	68.000000	9436.500000	1.0	0.0	3.000000	3.0	0.0	0.0	2.0	3.0	0.0	6.000000	2008.000000	9
75%	2197.750000	9.071811e+08	70.000000	5.0	78.000000	11555.250000	1.0	0.0	3.000000	3.0	0.0	0.0	2.0	3.0	0.0	8.000000	2009.000000	9
max	2930.000000	1.007100e+09	145.000000	5.0	105.000000	17727.750000	1.0	0.0	3.000000	3.0	0.0	0.0	2.0	3.0	0.0	12.000000	2010.000000	9

شکل ۲- توصیف اطلاعات آماری برای هریک از ویژگی‌های موجود در مجموعه داده

بخش چهارم: ماتریس همبستگی و یافتن مؤثرترین ویژگی‌ها

در این بخش از تمرین، قصد داریم تا مؤثرترین ویژگی‌ها را در بین ویژگی‌های موجود در مجموعه داده بر روی قیمت که متغیر هدف ماست، بیابیم. برای این کار، از ماتریس همبستگی استفاده می‌کنیم. از آنجا که ماتریس همبستگی تنها قابلیت پیاده‌سازی بر روی داده‌های عددی را دارد، ابتدا شرایط را بررسی می‌کنیم و در صورتی که شرایط مساعد باشد، می‌توانیم با استفاده از متد `CORR` در کتابخانه `Pandas`، ماتریس همبستگی را در اختیار داشته باشیم. از آنجا که مقدار همبستگی از $+1$ (تأثیر بسیار زیاد با رابطه مستقیم) تا -1 (تأثیر بسیار زیاد با رابطه معکوس) می‌تواند تغییر کند، در تعیین همبستگی و انتخاب ویژگی‌های مؤثر، بایستی قدرمطلق همبستگی را در نظر بگیریم؛ نه صرفاً میزان همبستگی را.

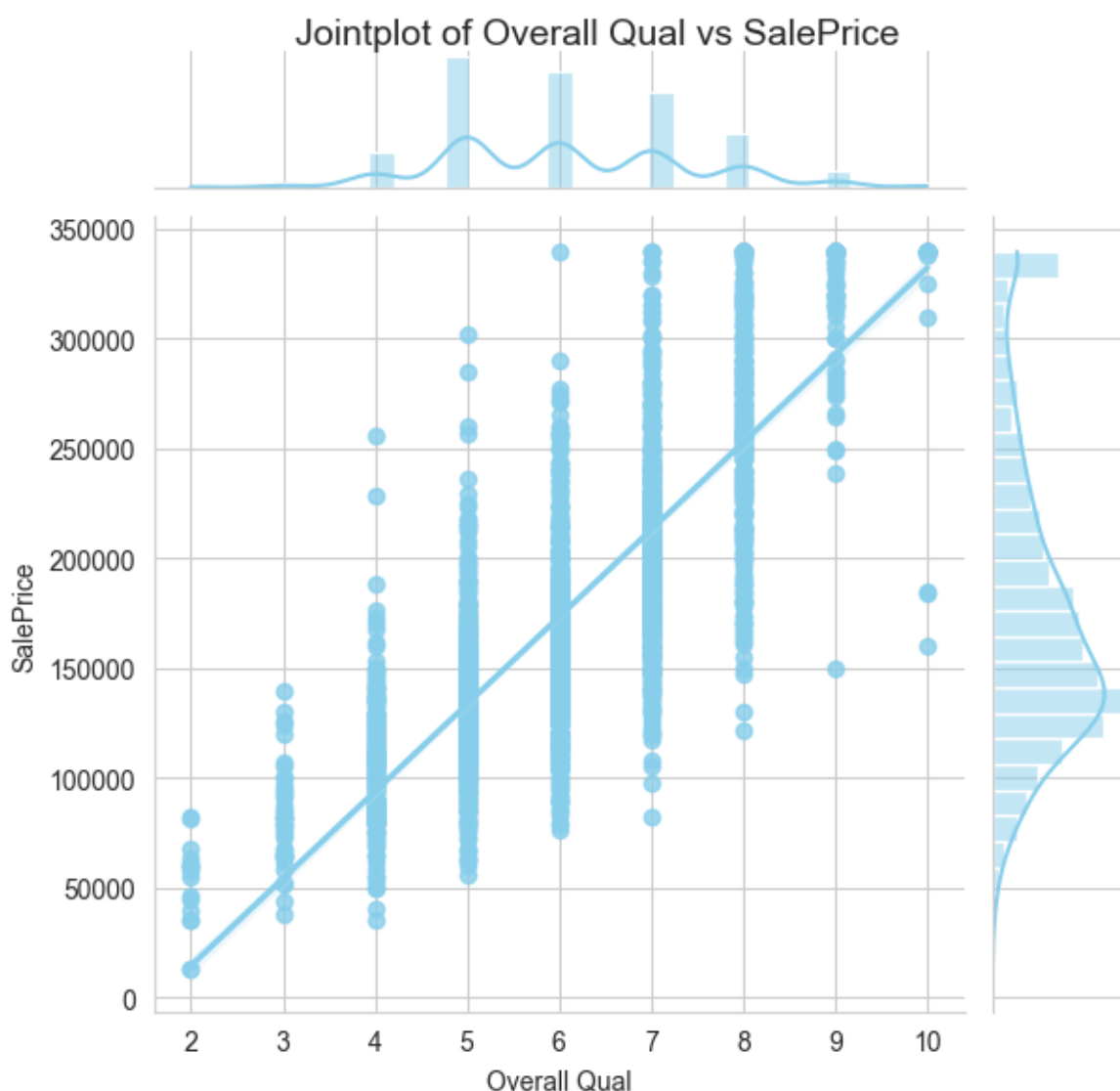
در ادامه، ویژگی‌هایی که همبستگی آن‌ها با متغیر هدف که قیمت خانه می‌باشد را به دست آورده‌ایم و معیار ما این بوده است که قدرمطلق همبستگی بیش‌تر از ۵۰ درصد باشد. از این حیث، ویژگی‌های مؤثر مطابق شکل ۳ به دست می‌آیند.



شکل ۳- ویژگی‌های مؤثر بر قیمت خانه‌ها با اندازه ضریب همبستگی بیش از ۵۰ درصد

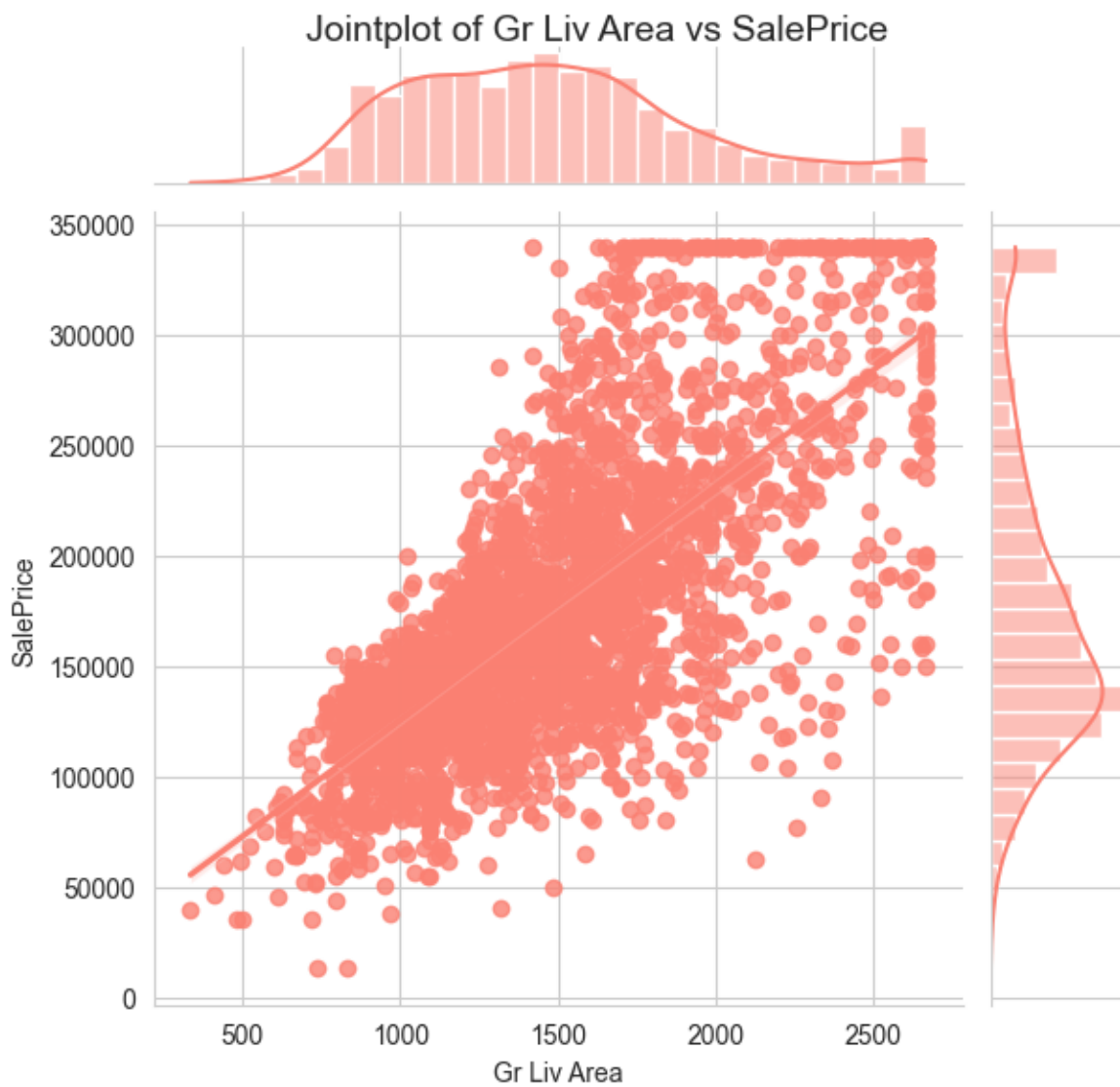
بخش پنجم: نمودارهای توزیع متقابل^{۱۱} برای ویژگی‌های به‌دست آمده

در این بخش، با استفاده از کتابخانه Seaborn و دستور jointplot نمودارهای توزیع متقابل را برای ویژگی‌هایی که در بخش چهارم به‌دست آورده‌ایم، ترسیم می‌کنیم. با توجه به این که در بخش چهارم، با معیار انتخاب شده ۱۵ ویژگی به‌عنوان ویژگی‌های مؤثر انتخاب شده‌اند، در این بخش نمودارهای توزیع متقابل برای هریک از این ۱۵ ویژگی را در شکل‌های ۴ الی ۱۸ نمایش داده‌ایم.

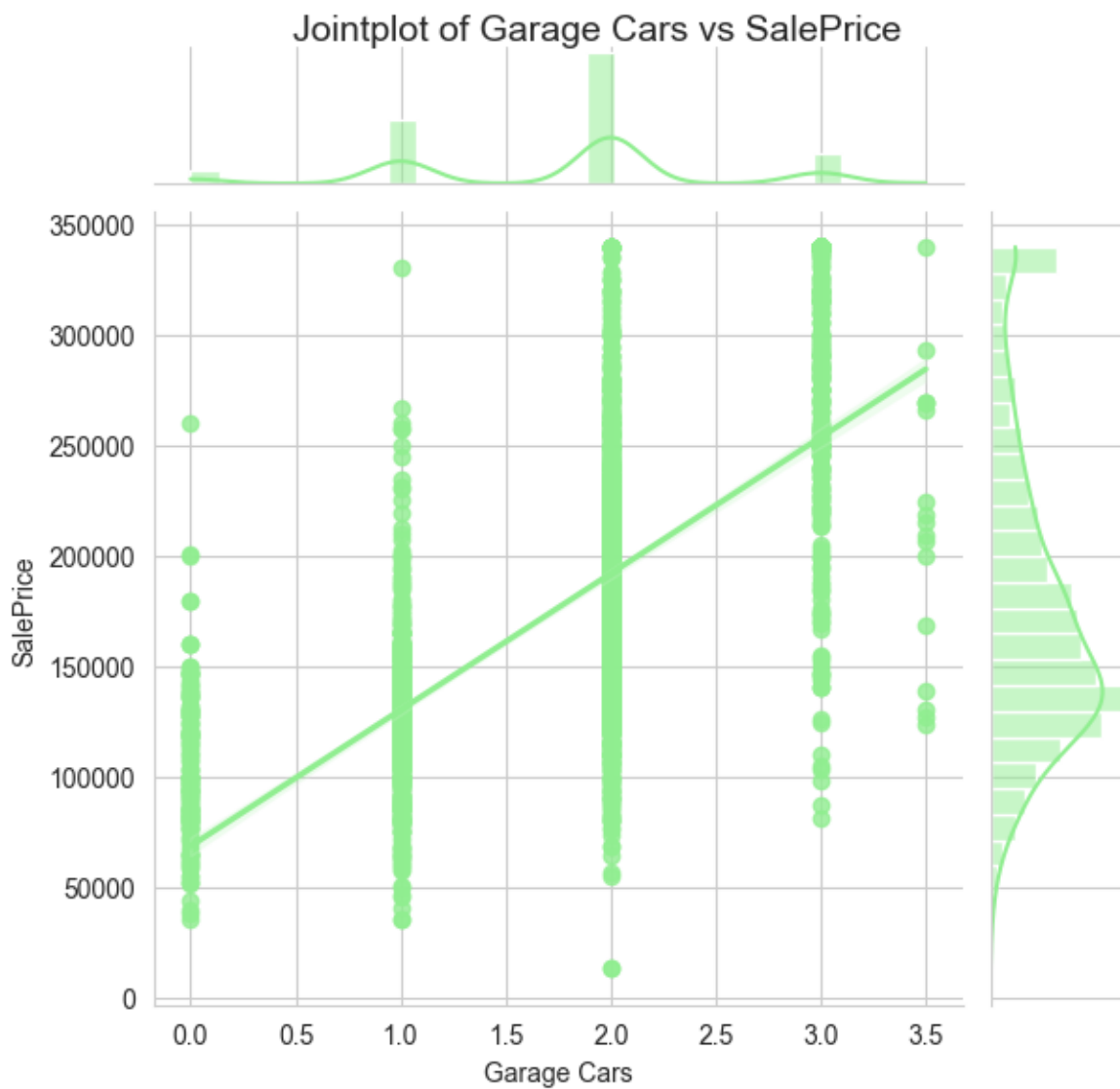


شکل ۴- نمودار توزیع متقابل کیفیت کلی و قیمت فروش خانه

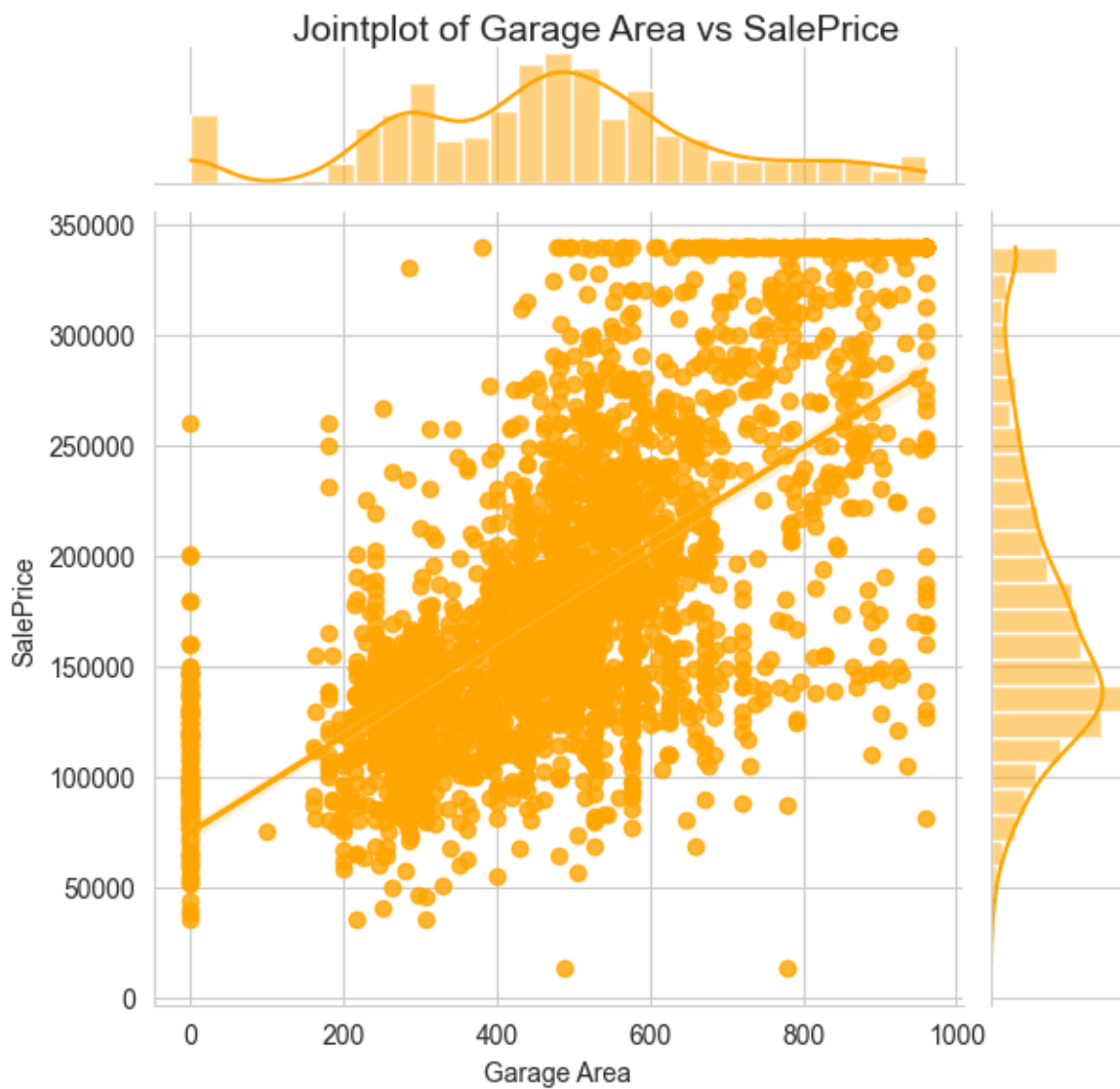
^{۱۱} Joint Plot



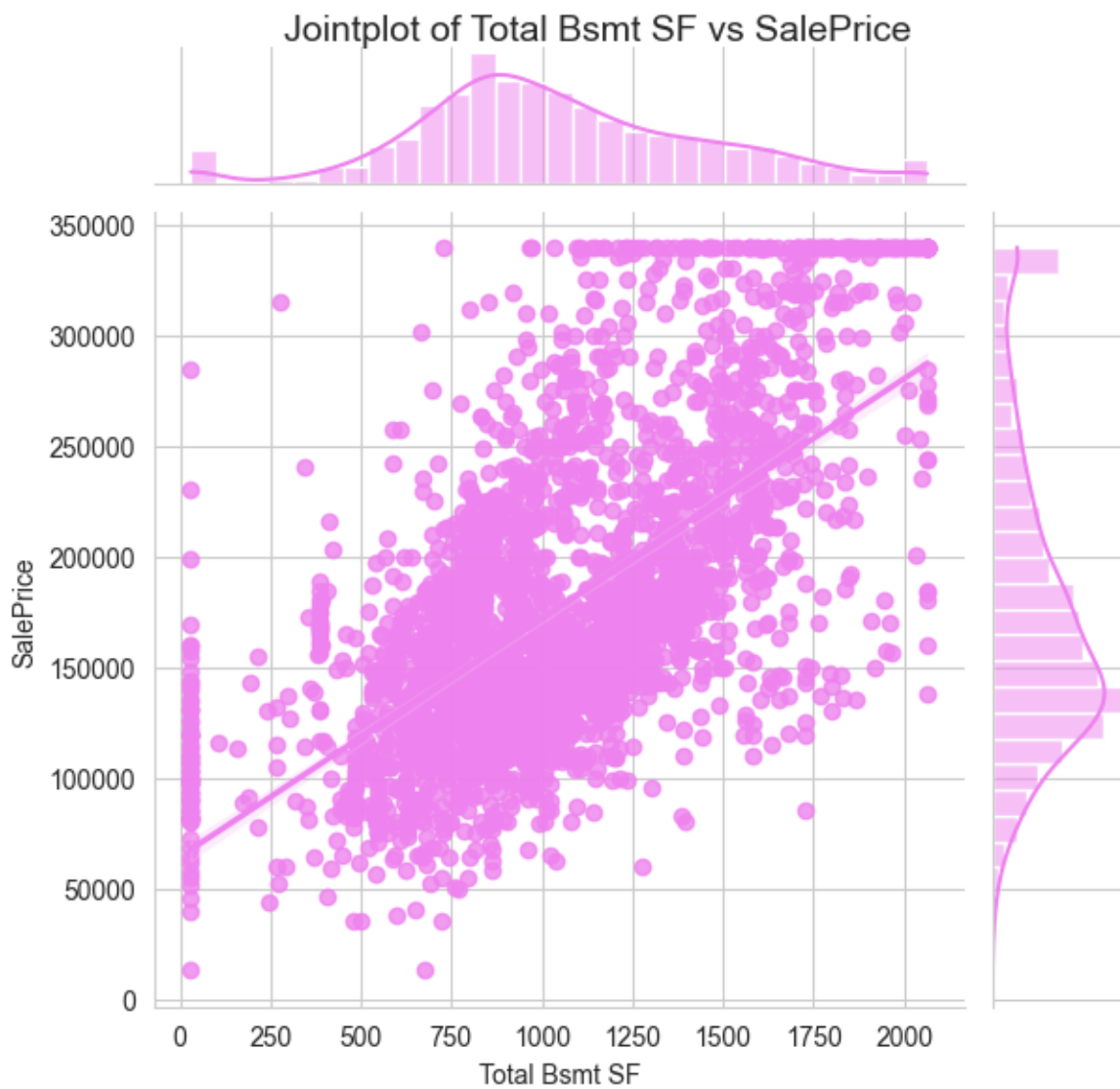
شکل ۵- نمودار توزیع متقابل مساحت ناخالص و قیمت فروش خانه



شکل ۶- نمودار توزیع متقابل گاراژ خودروها و قیمت فروش خانه



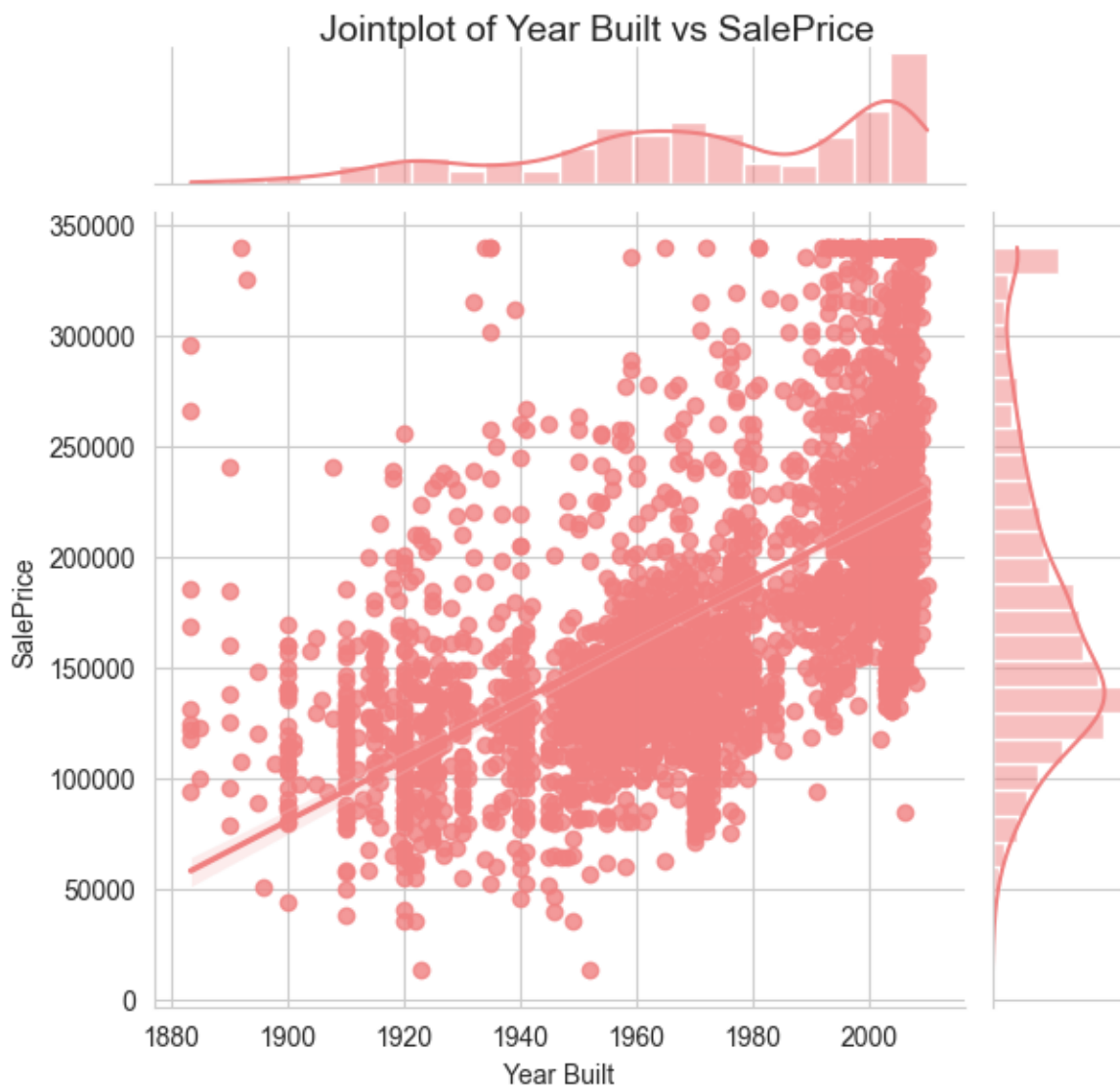
شکل ۷- نمودار توزیع متقابل مساحت گاراژ و قیمت فروش خانه



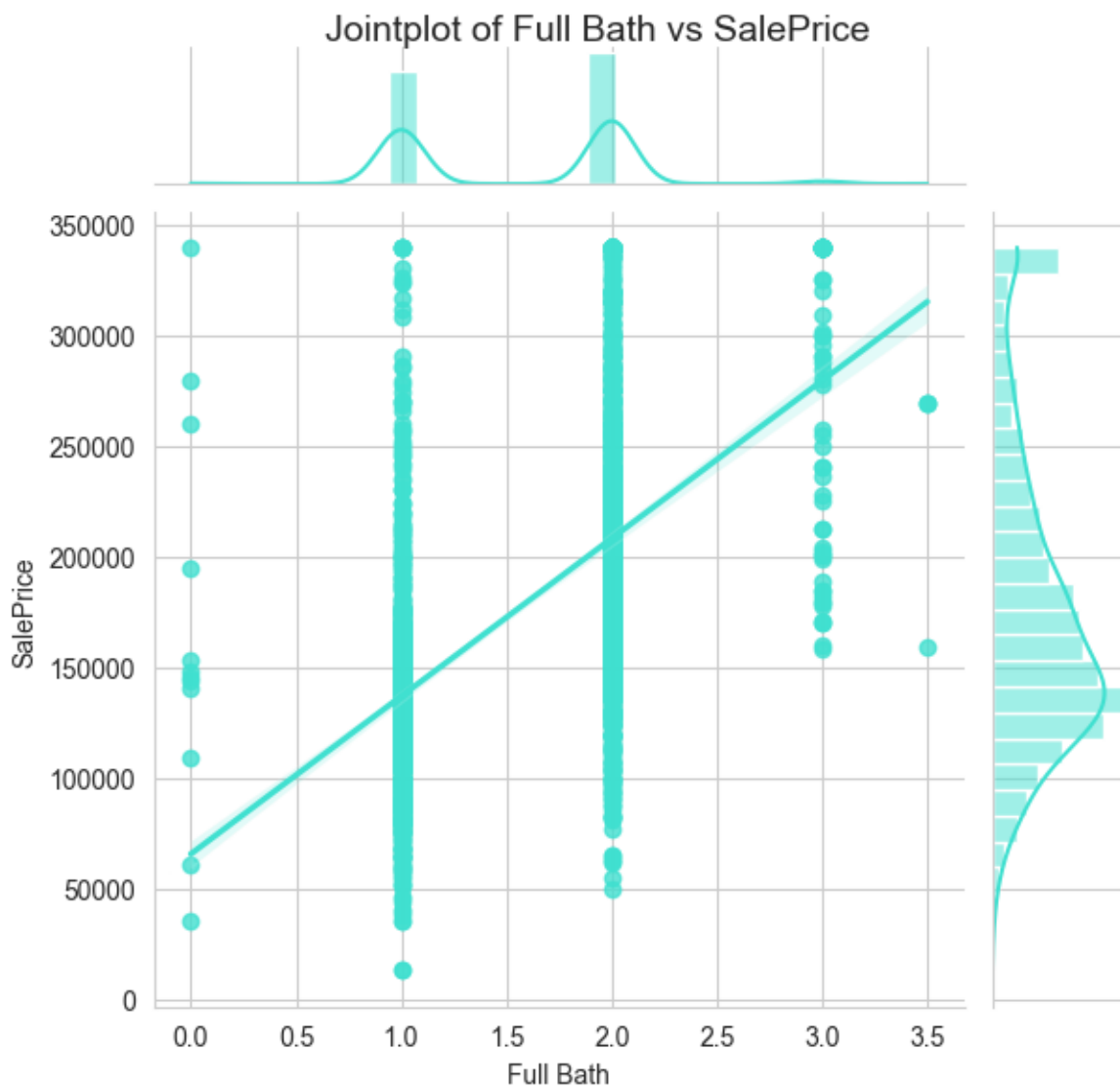
شکل ۸- نمودار توزیع متقابل مساحت زیرزمین به فوت مربع و قیمت فروش خانه



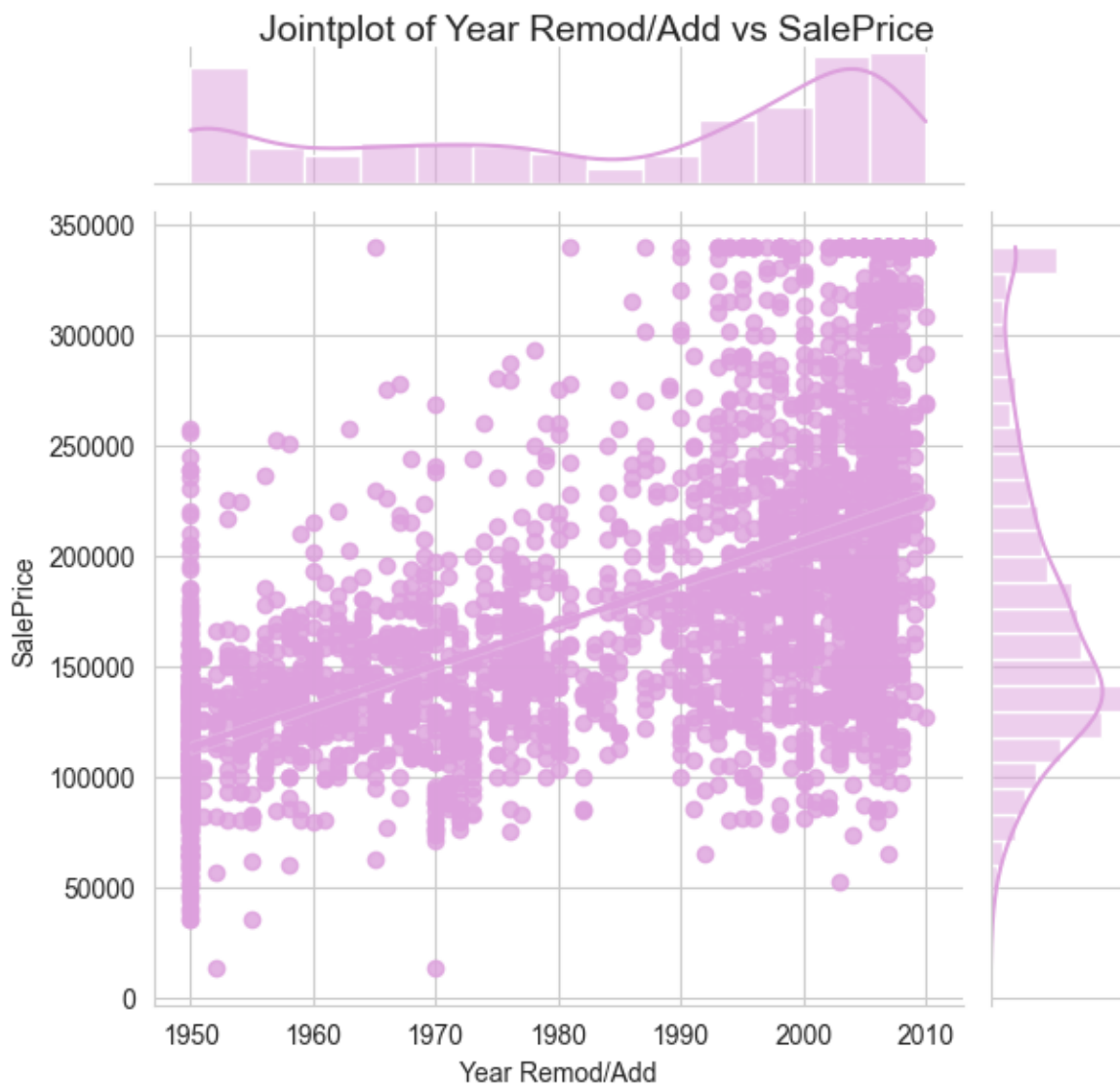
شکل ۹- نمودار توزیع متقابل مساحت طبقه اول و قیمت فروش خانه



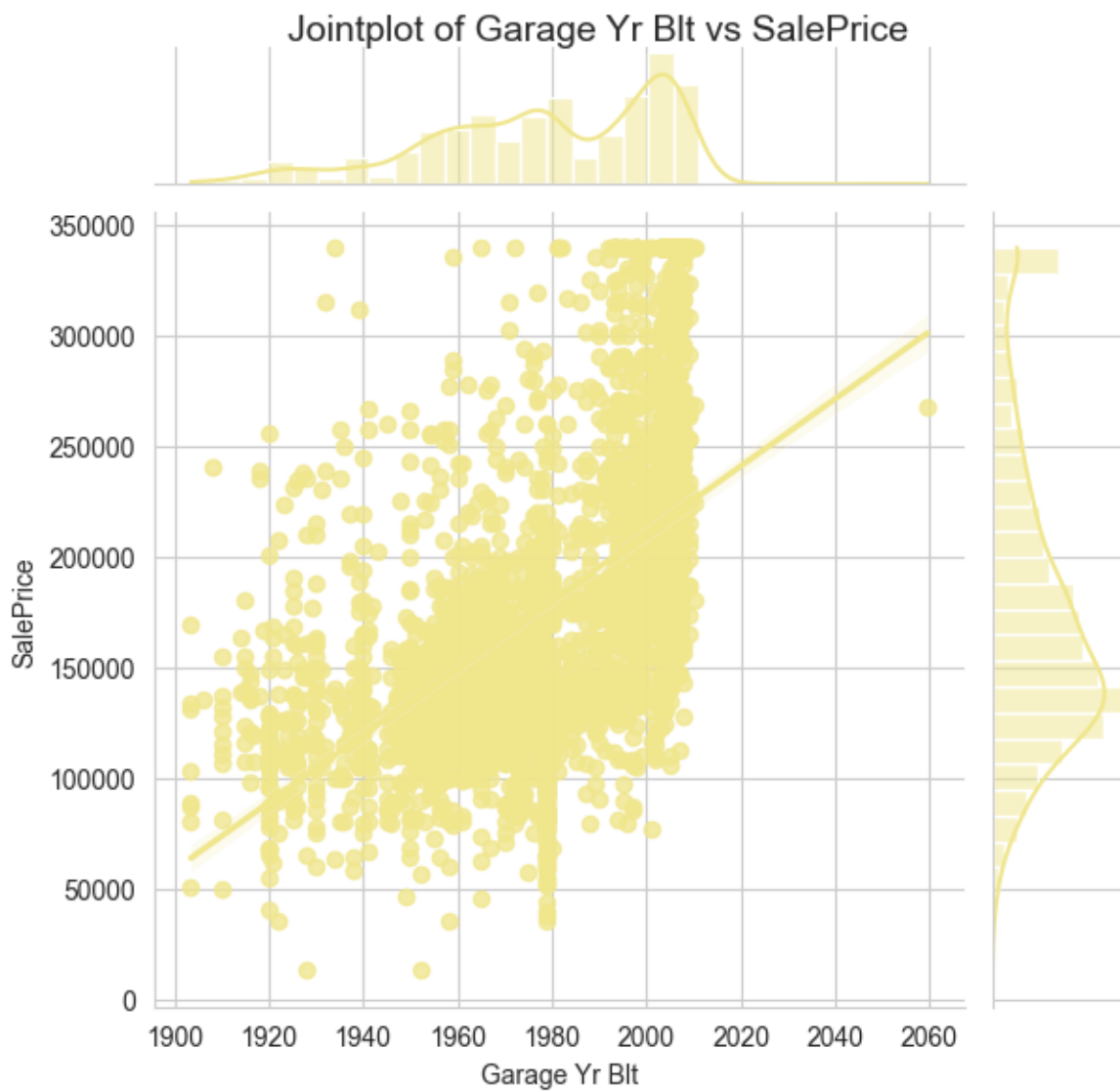
شکل ۱۰- نمودار توزیع متقابل سال ساخت خانه و قیمت فروش خانه



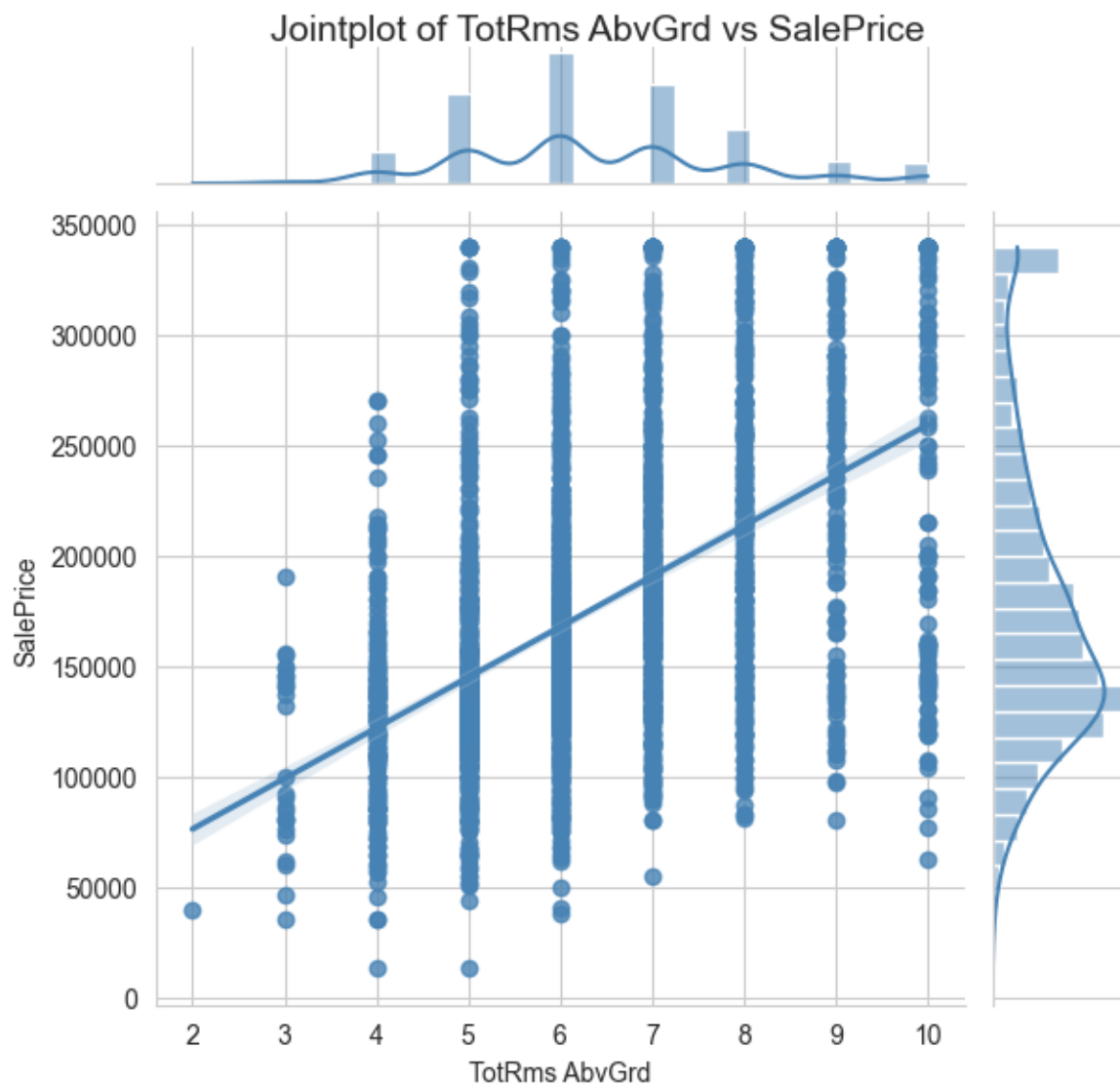
شکل ۱۱- نمودار توزیع متقابل تعداد حمام‌ها و قیمت فروش خانه



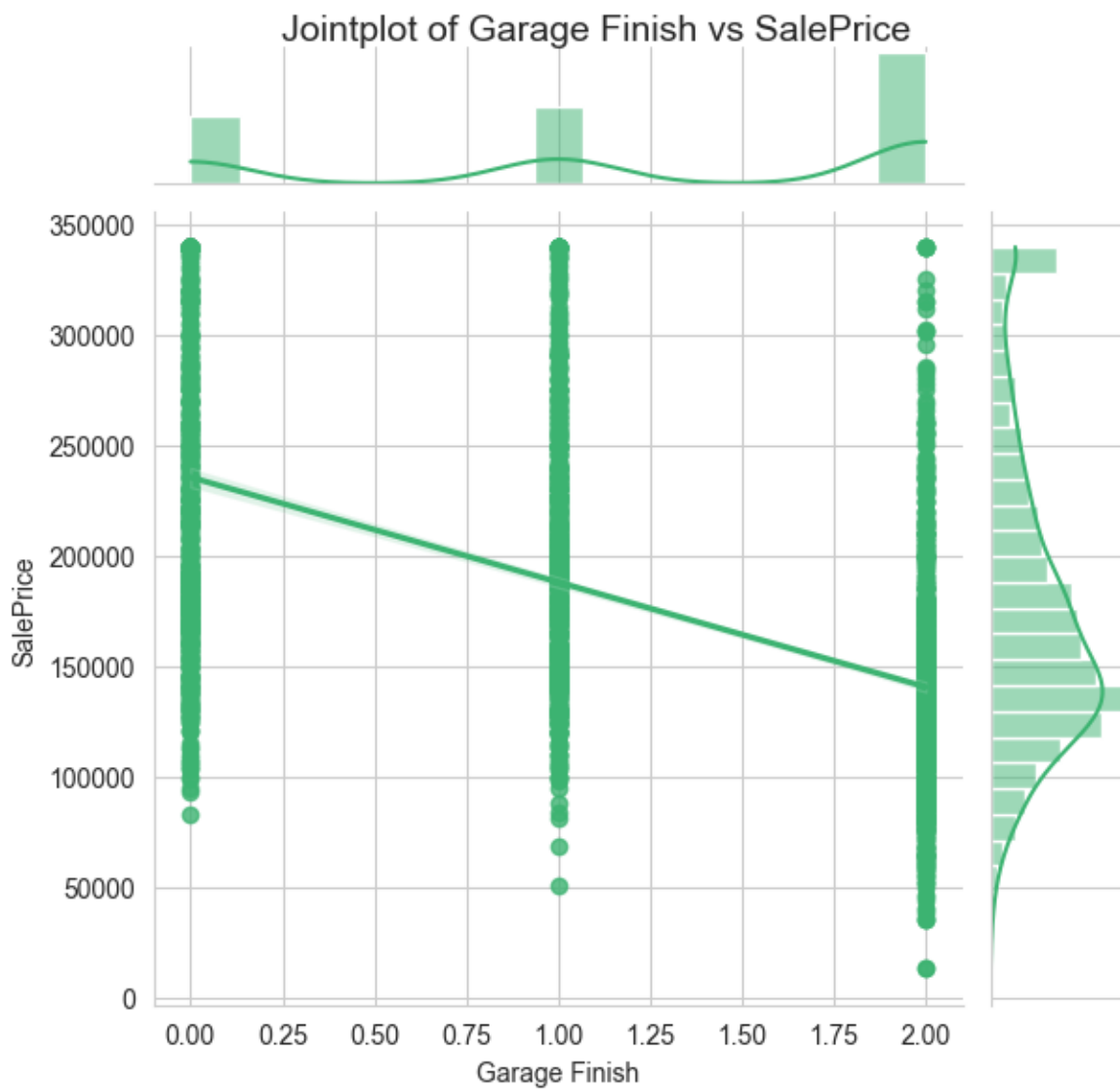
شکل ۱۲- نمودار توزیع متقابل سال بازسازی و قیمت فروش خانه



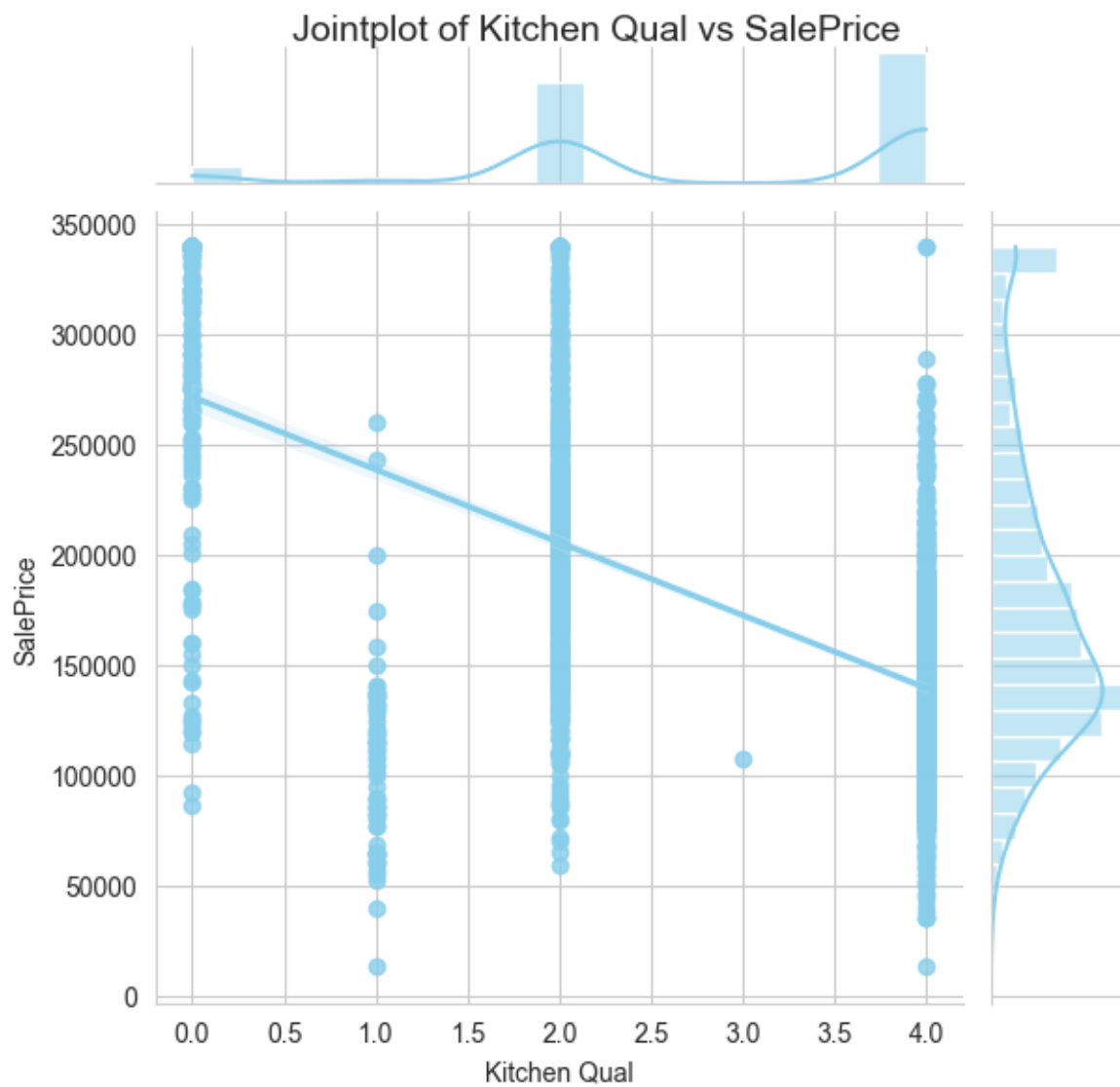
شکل ۱۳- نمودار توزیع متقابل سال ساخت گاراژ و قیمت فروش خانه



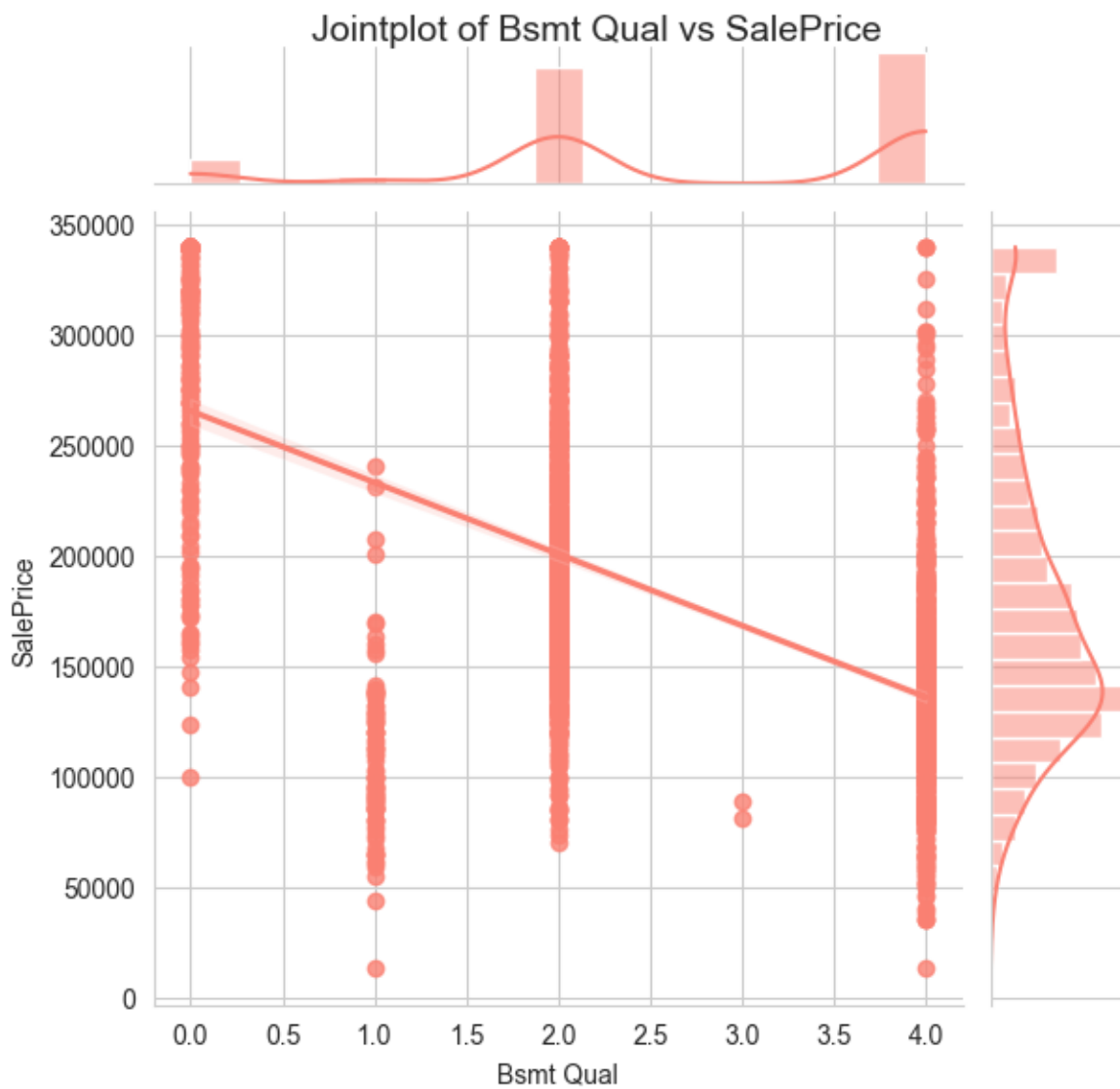
شکل ۱۴- نمودار توزیع متقابل تعداد کل اتاق‌ها و قیمت فروش خانه



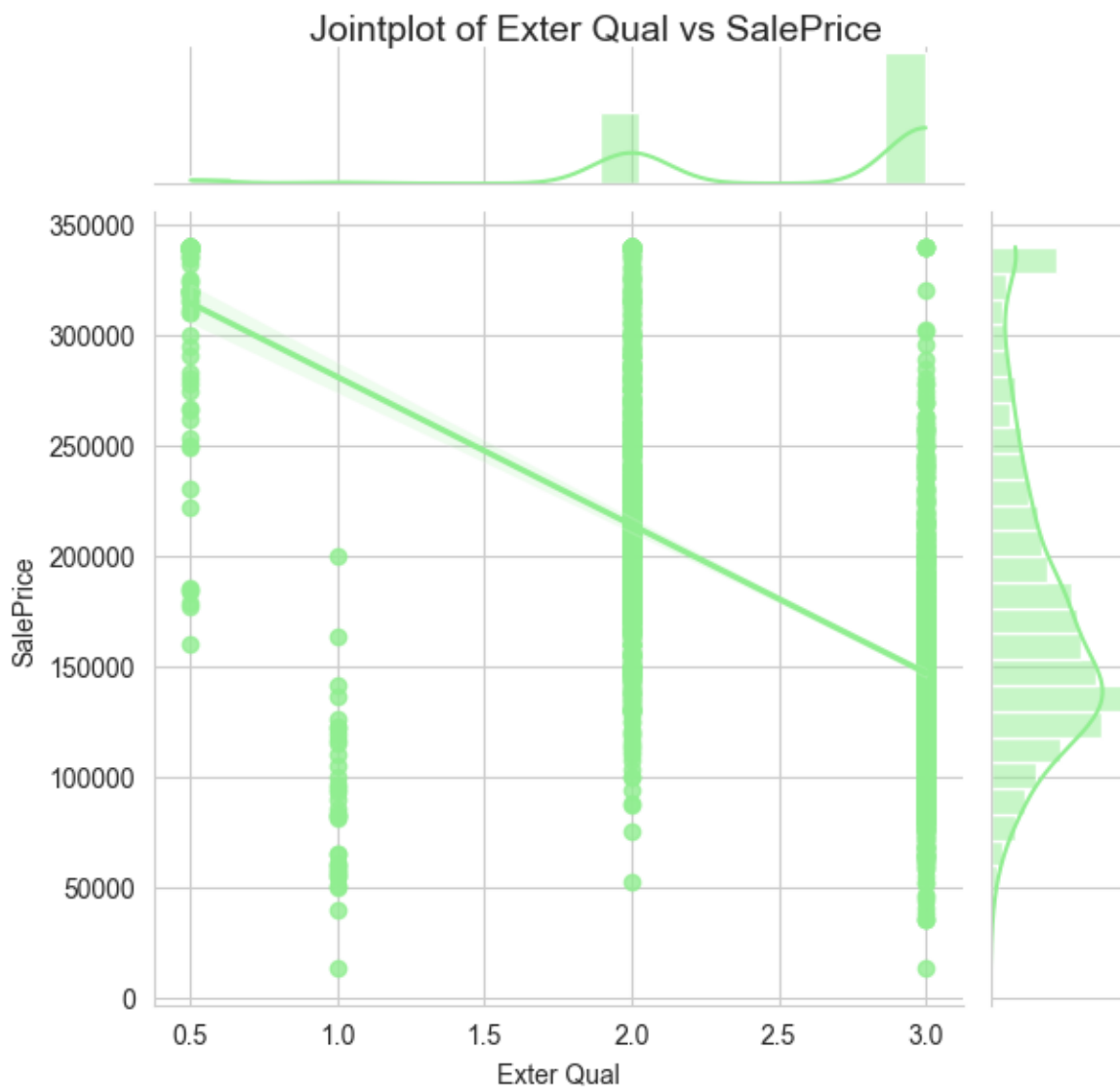
شکل ۱۵- نمودار توزیع متقابل پایان کار گاراژ و قیمت فروش خانه



شکل ۱۶- نمودار توزیع متقابل کیفیت آشپزخانه و قیمت فروش خانه



شکل ۱۷- نمودار توزیع متقابل کیفیت زیرزمین و قیمت فروش خانه



شکل ۱۸- نمودار توزیع متقابل کیفیت محیط بیرونی و قیمت فروش خانه

بخش ششم: انتخاب تعداد ویژگی‌ها با استفاده از SelectKBest

در این قسمت، قصد داریم تا با استفاده از دستور SelectKBest از کتابخانه sklearn، تعداد ویژگی‌های مؤثر برای آموزش مدل‌های را انتخاب کنیم. برای این کار، می‌توانیم از امتیاز اعتبار سنجی متقابل^{۱۲} استفاده کنیم؛ بدین معنا که به ازای همه k های مختلف که از ۱ الی تعداد کل ویژگی‌ها می‌تواند متغیر باشد، این امتیاز را محاسبه کنیم و ببینیم در کدام حالت به بیشینه مقدار می‌رسیم. با استفاده از تکه کد نشان داده‌شده در شکل ۱۹، این امتیاز را برای همه k های موجود از ۱ الی ۸۱ می‌توانیم به‌دست آوریم و بین آن‌ها، مشاهده می‌شود که مقدار $k=50$ بیش‌ترین امتیاز را به‌دست می‌آورد و در نتیجه، ما ۵۰ ویژگی با بیش‌ترین تأثیر را در آموزش مدل‌ها استفاده خواهیم کرد.

```
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

X = df.drop('SalePrice', axis=1)
y = df['SalePrice']

# Try different values of k and evaluate model performance
for k in range(1, X.shape[1] + 1):
    selector = SelectKBest(score_func=f_regression, k=k)
    X_new = selector.fit_transform(X, y)

    model = LinearRegression()
    scores = cross_val_score(model, X_new, y, cv=5, scoring='r2')

    print(f"k = {k}, R2 score = {np.mean(scores):.4f}")
```

شکل ۱۹- انتخاب k مناسب برای آموزش مدل‌ها

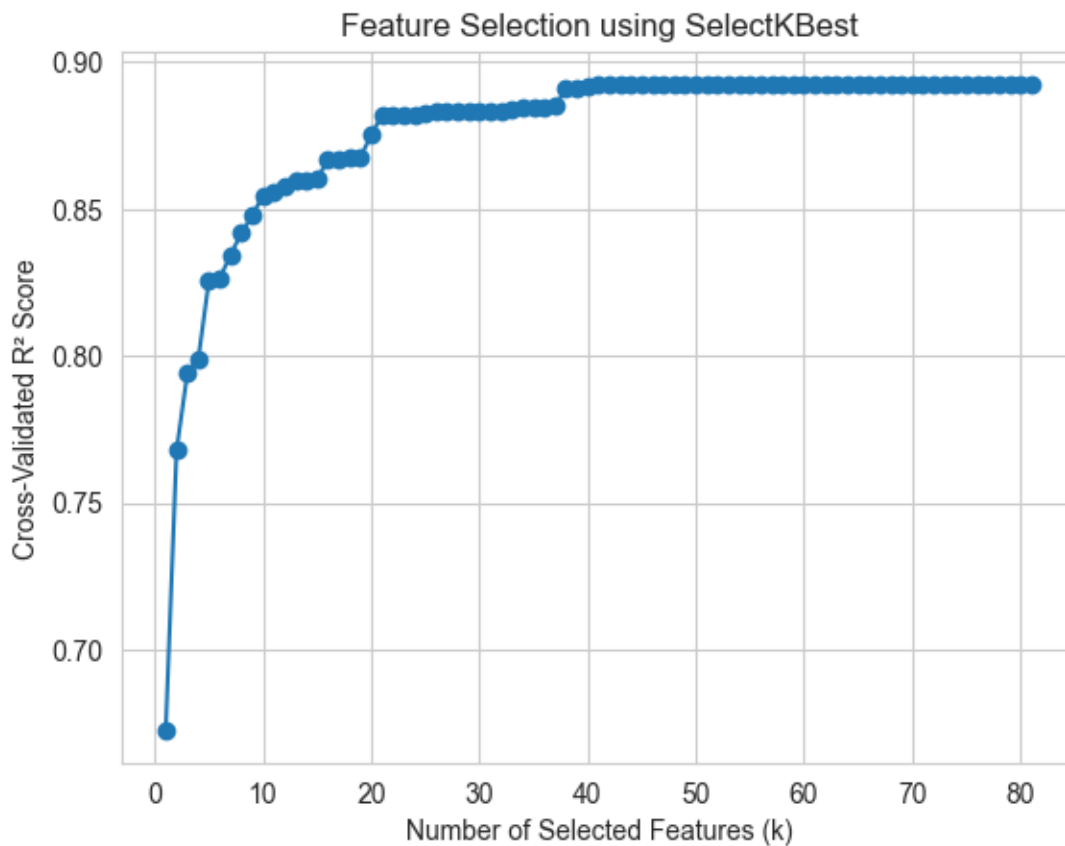
لازم به ذکر است که در دستور زیر:

```
scores = cross_val_score(model, X_new, y, cv=5, scoring='r2')
```

مقدار cv به این معناست که داده‌ها به ۵ بخش مساوی تقسیم می‌شوند؛ در هر مرحله، ۴ بخش برای آموزش و ۱ بخش برای آزمون استفاده می‌شود. این فرآیند ۵ بار تکرار می‌شود تا هر بخش دقیقاً یک بار به عنوان داده آزمون استفاده شود. هم‌چنین برای ارزیابی از معیار ضریب تعیین یا R^2 Score استفاده شده‌است.

¹² Cross Validation Score

کار دیگری که می‌توان انجام داد، این است که نمودار ضریب تعیین را به ازای k های مختلف ترسیم کنیم و ببینیم شدت تغییرات در چه مقداری از k کم یا زیاد می‌شود و برحسب آن تصمیم بگیریم. شکل ۲۰، این نمودار را به ازای k های مختلف نمایش می‌دهد.



شکل ۲۰- نمودار ضریب تعیین برحسب تعداد ویژگی‌ها (k)

در ادامه، برای آموزش مدل‌های مختلف از $k=50$ استفاده خواهیم کرد.

بخش هفتم: تقسیم دادگان به بخش‌های آموزش و تست

در این قسمت، دادگان آموزشی خود را به دو قسمت آموزش و تست تقسیم می‌کنیم. دادگان آموزش را برابر با ۷۵٪ از کل داده‌ها و دادگان تست را برابر با ۲۵٪ از کل داده‌ها در نظر می‌گیریم. این کار در تکه کدی که در شکل ۲۱ نشان داده شده، انجام گردیده‌است.

```
from sklearn.model_selection import train_test_split

# Assuming X and y are already defined and X has all features
# Apply SelectKBest with k=50
selector = SelectKBest(score_func=f_regression, k=50)
X_selected = selector.fit_transform(X, y)

# Split the data: 75% training, 25% testing
X_train, X_test, y_train, y_test = train_test_split(
    X_selected, y, test_size=0.25, random_state=42
)

# Optional: print shapes to verify
print("Training set shape:", X_train.shape)
print("Test set shape:", X_test.shape)

✓ 0.0s

Training set shape: (2197, 50)
Test set shape: (733, 50)
```

شکل ۲۱- نحوه تقسیم‌بندی داده‌های آموزش و داده‌های تست

با توجه به شکل ۲۱، تعداد ۲۱۹۷ داده برای آموزش و تعداد ۷۳۳ داده برای تست مورد استفاده قرار خواهند گرفت.

بخش هشتم: آموزش مدل‌ها

در این قسمت، با مشخص شدن داده‌های آموزش و داده‌های تست، می‌توانیم آموزش مدل‌ها را آغاز کنیم. این کار در تکه کد نشان داده‌شده در شکل ۲۲ انجام شده‌است.

```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# 1. Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# 2. Lasso Regression
lasso_model = Lasso(alpha=1.0)
lasso_model.fit(X_train, y_train)

# 3. Ridge Regression
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)

# 4. Polynomial Regression (using degree 2)
poly_model = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
poly_model.fit(X_train, y_train)
```

شکل ۲۲- آموزش مدل‌های خواسته شده

در آموزش مدل‌ها، پارامتر α وجود دارد که برای هر دو مدل Lasso و Ridge مقدار آن را برابر با یک که همان پیش‌فرض آن‌هاست، در نظر گرفته‌ایم. این پارامتر برای تنظیم^{۱۳} ضرایب مورد استفاده قرار می‌گیرد و باعث می‌شود که ضرایب رگرسیون تقریباً در یک مرتبه قرار گیرند. این امر سبب می‌شود که تعمیم‌پذیری حداکثر شود و مانع از بیش‌برازش داده‌های موجود شود.

هم‌چنین، برای مدل چندجمله‌ای حداقل درجه ممکن بعد از رگرسیون خطی که از درجه ۱ حساب می‌شود، یعنی درجه ۲ استفاده گردیده‌است.

¹³ Regularization

بخش نهم: ارزیابی عملکرد

در این بخش، ابتدا به معرفی معیارهای ارزیابی مدل و نحوه محاسبه آن می‌پردازیم؛ سپس، نتایج به‌دست آمده از ارزیابی مدل‌های بخش قبل را بررسی می‌کنیم.

خطای MSE

میانگین مربعات خطای یکی از پرکاربردترین معیارها است که هم در آموزش مدل‌ها و هم در مقایسه مدل‌ها استفاده می‌شود. این معیار دارای بُعد است، به همین دلیل به عنوان گزارش نهایی در مسائل رگرسیون مناسب نیست. محاسبه میانگین مربعات خطا، به‌صورت رابطه زیر انجام می‌پذیرد:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - f_i)^2 = \frac{1}{n} \sum_{i=1}^n e_i^2$$

رفتار معیار میانگین مربعات خطا نسبت به خطای هر داده به‌صورت سهمی است؛ لذا به خطاهای بزرگ‌تر، وزن بیشتری اختصاص می‌یابد. همچنین، این معیار مشتق‌پذیر بوده و با فرض یک مدل خطی، می‌توان از آن مشتق نیز گرفت.

خطای RMSE

جذر میانگین مربعات خطا با استفاده از رابطه زیر و از روی میانگین مربعات خطا محاسبه می‌شود:

$$RMSE = \sqrt{MSE}$$

دلیل استفاده از جذر میانگین مربعات خطا، یکسان بودن بُعد و مقیاس آن با ویژگی هدف است. برای مثال، این معیار اغلب در گزارش نتایج استفاده می‌شود و به عنوان تابع هزینه استفاده نمی‌شود.

ضریب تعیین

ضریب تعیین، بر اساس نسبت تغییرات کل نتایج پیش‌بینی شده توسط مدل، تخمینی از میزان مطلوبیت داده پیش‌بینی شده که خروجی مدل است، ارائه می‌دهد. ضریب تعیین با کم کردن نسبت مجموع مربعات مانده‌ها (SS_{res}) به مجموع کل مربع‌ها (SS_{total}) از عدد یک به‌دست می‌آید. روابط زیر نحوه به‌دست آوردن ضریب تعیین را شرح می‌دهند:

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$$

$$SS_{res} = \sum_i (y_i - f_i)^2$$

$$SS_{total} = \sum_i (y_i - \bar{y})^2$$

که در آن y_i مقدار اندازه‌گیری شده، f_i مقدار پیش‌بینی شده توسط شبکه عصبی و \bar{y} میانگین مقادیر اندازه‌گیری شده می‌باشد. ضریب تعیین می‌تواند مقادیری بین ۰ تا ۱ داشته باشد و مقادیر بالاتر نشان‌دهنده تناسب بهتر مدل با داده‌هاست. ضریب تعیین ۱ به این معنی است که مدل به طور کامل متغیر وابسته را پیش‌بینی می‌کند، در حالی که ضریب تعیین ۰ به این معنی است که مدل فاقد توانایی پیش‌بینی و تعمیم‌دهی است.

پس از ارائه توضیحات در خصوص معیارهای ارزیابی، با بهره‌گیری از معیار خطای RMS و ضریب تعیین، مدل‌های مذکور را مورد ارزیابی قرار می‌دهیم. این ارزیابی در شکل ۲۳ انجام شده‌است و نتایج آن نیز قابل مشاهده می‌باشد.

```
# Linear Regression
y_pred_linear = linear_model.predict(X_test)
r2_linear = r2_score(y_test, y_pred_linear)
rmse_linear = np.sqrt(mean_squared_error(y_test, y_pred_linear))

# Lasso Regression
y_pred_lasso = lasso_model.predict(X_test)
r2_lasso = r2_score(y_test, y_pred_lasso)
rmse_lasso = np.sqrt(mean_squared_error(y_test, y_pred_lasso))

# Ridge Regression
y_pred_ridge = ridge_model.predict(X_test)
r2_ridge = r2_score(y_test, y_pred_ridge)
rmse_ridge = np.sqrt(mean_squared_error(y_test, y_pred_ridge))

# Polynomial Regression
y_pred_poly = poly_model.predict(X_test)
r2_poly = r2_score(y_test, y_pred_poly)
rmse_poly = np.sqrt(mean_squared_error(y_test, y_pred_poly))

# Display the results
print("Model Performance on Test Set:")
print(f"Linear Regression    -> R²: {r2_linear:.6f}, RMSE: {rmse_linear:.4f}")
print(f"Lasso Regression     -> R²: {r2_lasso:.6f}, RMSE: {rmse_lasso:.4f}")
print(f"Ridge Regression      -> R²: {r2_ridge:.6f}, RMSE: {rmse_ridge:.4f}")
print(f"Polynomial Regression -> R²: {r2_poly:.6f}, RMSE: {rmse_poly:.4f}")
```

✓ 0.0s

```
Model Performance on Test Set:
Linear Regression    -> R²: 0.905180, RMSE: 21940.4118
Lasso Regression     -> R²: 0.905187, RMSE: 21939.6706
Ridge Regression      -> R²: 0.905192, RMSE: 21939.0279
Polynomial Regression -> R²: 0.900646, RMSE: 22458.8507
```

شکل ۲۳- نحوه ارزیابی مدل‌ها و به‌دست آوردن ضریب تعیین و خطای RMS

همان‌طور که در شکل ۲۳ مشاهده می‌شود، کلیه مدل‌ها دارای ضریب تعیین بالاتر ۰.۹ می‌باشند که این امر حاکی از تعمیم‌پذیری مناسب مدل‌های آموزش دیده می‌باشد تا بتوانند داده‌های نادیده را با دقت خوبی تخمین بزنند. همچنین، خطای RMS برای کلیه مدل‌ها تقریباً در یک محدوده می‌باشد که از این حیث نیز می‌توان می‌توان عملکرد کلی را مناسب ارزیابی کرد و در عمل هرچهار مدل تقریباً به‌طور یکسان داده‌های نادیده را پیش‌بینی می‌کنند.

در حالت کلی، در حوزه یادگیری ماشین، خطا را در ۳ دسته کلی می‌توان جای داد:

بایاس

بایاس را می‌توان خطای ناشی از **فرضیات ساده‌سازی شده مدل** در یادگیری دانست. مدل‌های با بایاس بالا معمولاً **بیش‌ازحد ساده** هستند و پیچیدگی لازم برای درک داده‌های غیرخطی و پیچیده را ندارند؛ مثلاً اگر بخواهیم از رگرسیون خطی روی داده‌های پیچیده استفاده کنیم، قطعاً به مشکل بایاس برخورد می‌کنیم. بایاس موجب می‌شود که مدل نتواند الگوهای واقعی داده را خوب یاد بگیرد و عملاً سبب کم‌برازش^{۱۴} خواهد شد.

واریانس

واریانس را در عمل، می‌توان حساسیت مدل به **نوسانات داده آموزشی** دانست. مدل‌هایی با واریانس بالا معمولاً **بیش‌ازحد پیچیده** هستند و به داده آموزشی کاملاً منطبق می‌شوند. این مشکل در زمانی که داده‌ها دارای نویز باشند، سبب می‌شود که مدل نتواند تعمیم‌پذیری مناسبی داشته‌باشد و صرفاً نویز را یاد می‌گیرد.

باید در نظر داشته‌باشیم که وقتی پیچیدگی مدل را افزایش می‌دهیم، بایاس کم و واریانس زیاد می‌شود. در نتیجه بایستی بین این دو تعادل برقرار کنیم تا خطای کل کمینه شود.

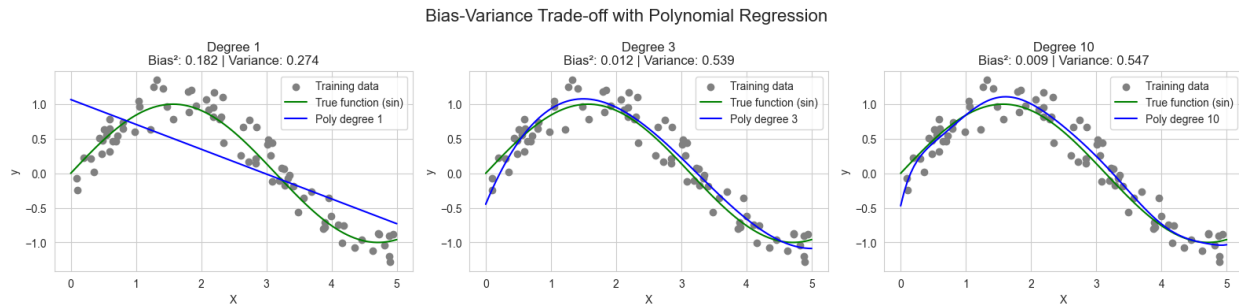
خطای غیرقابل کاهش

این دسته از خطاها که محل بحث ما نیستند را می‌توان نویز ذاتی در داده‌ها دانست که هیچ مدلی توانایی پیش‌بینی آن را ندارد.

بعد از توضیح موارد، به ارائه یک مثال می‌پردازیم:

فرض می‌کنیم که می‌خواهیم با استفاده از توابع چندجمله‌ای، تعدادی داده تولید کنیم که از یک تابع سینوسی به‌دست آمده‌اند. می‌خواهیم با درجات ۱، ۳ و ۱۰ این نقاط را با رگرسیون چند جمله‌ای تقریب بزنیم. نتیجه به‌صورت زیر در شکل ۲۴ در خواهد آمد:

¹⁴ Underfitting



شکل ۲۴- تخمین با استفاده از چند جمله‌ای درجه ۱، ۳ و ۱۰

همان‌طور که در شکل ۲۴ مشاهده می‌شود، مدل درجه ۱ (مدل ساده) بایاس زیاد و واریانس کم دارد که موجب کم‌برازش شده‌است. مدل درجه ۱۰ (مدل پیچیده) دارای واریانس زیاد و بایاس کم است که موجب بیش‌برازش شده‌است. نهایتاً، مدل درجه ۳ که مدل متوازن محسوب می‌شود، دارای قابلیت تعمیم‌پذیری مناسب و بایاس و واریانس متعادل است.