Практическая статистика и визуализация с Python

Подготовьтесь к выполнению практического задания:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,
OneHotEncoder
from sklearn.model_selection import train_test_split
import seaborn as sns
# можно и так: import seaborn as sb
from scipy.stats import norm
from scipy import stats
from pandas import DataFrame
%matplotlib inline
```

1) *Данные*

Данные (`house_train.csv`) представляют собой **набор данных о ценах на жилье**. Подготовьте информацию о датасете:

```
pd.set_option('display.max_columns', 100)
df = pd.read_csv('house_train.csv')
df.drop('Id', axis=1, inplace=True)
df.head()
```

Приведите описание датасета:
- Сколько данных в датасете?
– Сколько параметров? Выведите список всех параметров.
– Есть ли категориальные признаки? Перечислите / выведите их.
– Выведите первые пять строчек `DataFrame`.
– По умолчанию pandas, ради экономии времени, указывает приблизительные сведения об использовании памяти объектом DataFrame. Если нас интересуют точные сведения, то нужно установить параметр `memory_usage` в значение `'deep'`:

```
Пусть Имя DataFrame – df, тогда исполняем инструкцию:
df.info(memory_usage='deep')
```

2) Просмотреть основную информацию по датасету можно, выполнив инструкции

```
df.columns      #Просмотр имен столбцов функций
df.shape        #Просмотр количества строк и столбцов
df.describe()   #Просмотр основной статистики
```

Статистическая сводка для числовых данных включает в себя среднее, минимальное и максимальное значения данных, которые могут быть полезны для

определения размера некоторых переменных и оценивания того, какие переменные могут быть наиболее важными.

С помощью метода `describe()` выведите описательную статистику числовых данных:

```
df.describe()
```

А что будет, если выполнить такую инструкцию:

```
df.describe().T  ?
```

– получим статистику отдельного показателя, например, `"SalePrice"`:

```
df['SalePrice'].describe()
```

Укажите: а) чему равно среднее значение?

б) чему равно стандартное отклонение?

3) Проверьте, есть ли пропуски и повторы в данных.

Пропущенные и неопределённые значения выявляет метод `isna()`, а суммарное количество таких значений – метод `sum()`.

Вызов обоих методов можно записать в одну строку, разделив их точкой. Python сначала вызовет метод `isna()`, а затем результаты его работы передаст методу sum():

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

– Количество пустых значений в наборе данных можно сохранить. Результат сохраним в переменной `na_number`:

```
na_number=(df.isna().sum())
print(na_number)
```

3) Повторяющиеся строки – дубликаты – выявляются методом `duplicated()` и подсчитываются тем же `sum()`. Если возвращаются нули, то данные пригодны для исследования:

```
print(df.duplicated().sum())    0
```

– Количество дубликатов в наборе данных можно сохранить, например, в переменной `duplicated_number`:

```
duplicated_number=df.duplicated().sum()
print(duplicated_number)
```

4) Получите список названий столбцов, запросив атрибут `columns`.

```
print(df.columns)
```

5) **Анализ пропущенных значений и удаление**

а) Выясним, в каких параметрах отсутствует статистика (данные):

```
na_count = df.isnull().sum().sort_values(ascending=False) # Вычисляем, сколько пропущенных значений в параметрах
```

```
na_rate = na_count / len(df) # Вычисляем частоту или вероятность, с которой пропущенное значение встречается в каждом параметре. Если вероятность большая (>0.5), столбцы-параметры можно смело удалять).
# формируем массив для печати
na_data = pd.concat([na_count,
na_rate],axis=1,keys=['count','ratio'])
print(na_data)
```

Есть два способа обработать отсутствующие значения. Один – проанализировать, полезны ли параметры (признаки) с отсутствующими значениями для задачи. Бесполезные параметры удаляются.

Полезные же признаки зависят от количества отсутствующих значений. Если количество отсутствующих значений больше определенного количества (%) – удалите образцы, а если меньше определенного количества (%) – используйте среднее значение, медианное значение или моду для их восстановления.

Второй способ – проанализировать причины, по которым эти отсутствующие значения отсутствуют, и использовать определенный метод для их преобразования в тип данных (тип переменной).

Первые четыре параметра можно смело удалять.

б) Вообще говоря, если количество отсутствующих данных для определенного признака достигает более 15%, то этот признак следует удалить, и считается, что такого признака нет в наборе данных – то есть мы не будем пытаться заполнить отсутствующие значения этих признаков).

– Проследим за количеством оставшихся столбцов. Зафиксируем первоначальное количество столбцов с помощью функции:

```
df.shape      (1460, 80)
```

Удалите столбцы с максимальным количеством отсутствующих данных: 'PoolQC', 'MiscFeature' и 'Alley'. Это не должно привести к уменьшению **эффективного объема информации** в данных, поскольку буквальные значения этих признаков, похоже, не имеют ничего общего с интересующим нас признаком – цена на жилье.

```
df = df.drop(['PoolQC', 'MiscFeature', 'Alley'], axis=1)
```

или лучше создайте новый файл и выведите столбцы

```
df_new=df.drop(['PoolQC', 'MiscFeature', 'Alley'], axis=1)
```

```
print(df_new.isna().sum())
```

– Выведем отдельно количество оставшихся столбцов:

```
df_new.shape
```
(1460, 77)

В оставшихся переменных с пропущенными значениями несколько функций GarageX. 'GarageQual' и 'GarageCond' имеют одинаковое количество пропущенных значений. На основании этого делаем вывод, что они могут представлять один и тот же набор наблюдений, поэтому удалим эти функции. Ту же операцию можно выполнить для **Fence – изгородь – 1179 нулей**.

```
df_new = df_new.drop(['GarageQual', 'GarageCond', 'Fence'],
axis=1)
print(df_new.isna().sum())
```
74 столбца осталось

Что касается MasVnrArea и MasVnrType, в соответствии с их буквальным значением, мы думаем, что они не важны, и у них есть сильная корреляция (как мы увидим дальше) с YearBuilt и GeneralQual. Таким образом, мы не потеряем никакой информации, если удалим эти две функции.

```
df_new=df_new.drop(['MasVnrArea', 'MasVnrType'], axis=1)
```

```
print(df_new.isna().sum())
```
72 столбца осталось

– Выведем отдельно количество оставшихся столбцов:

```
df_new.shape # Размер данных после обработки пропущенного
значения
```

В общем, мы удалили почти все переменные с пропущенными значениями.

6. Однофакторный анализ данных

- Переименуем опять файл:

```
df = df_new
```

- Проверим размер данных:

```
df.shape
```

- Считаем заново исходный файл с данными:

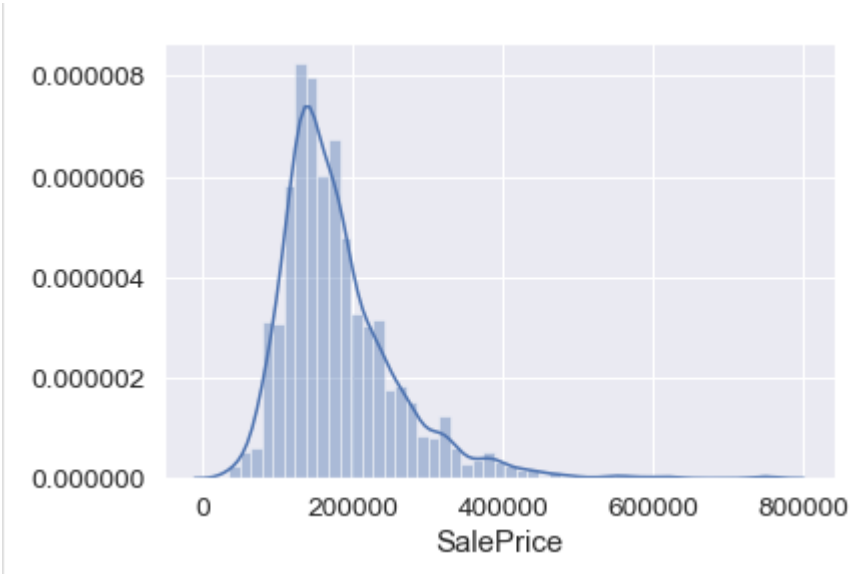
```
df = pd.read_csv('house_train.csv')
df.head()
```

6.1. Гистограмма

Построим гистограмму параметра SalePrice в библиотеке seaborn:

```
sns.distplot(df['SalePrice'])
```

Результат:



По рисунку: цена дома подчиняется нормальному распределению?

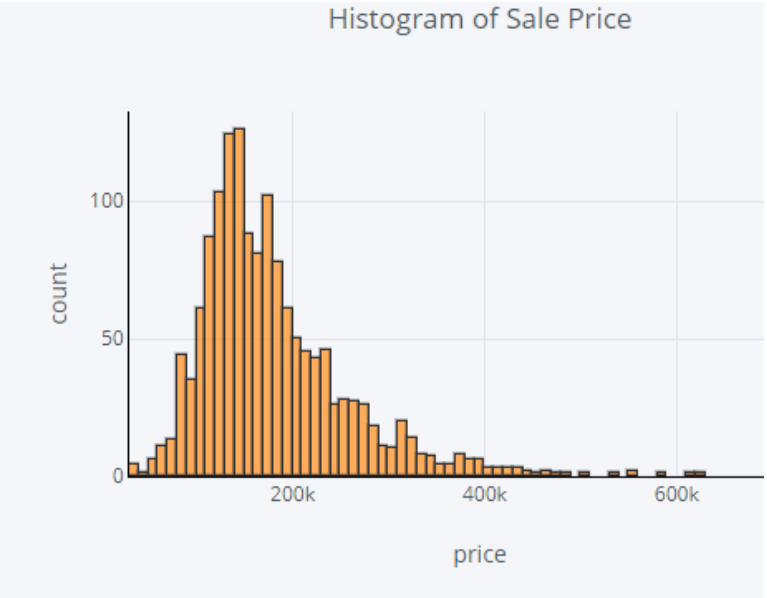
Проверьте эту гипотезу, изучив приложенные ссылки.

Можем рассчитать его асимметрию и эксцесс:

```
print("Skewness: %f" % df['SalePrice'].skew())
```

Задание: Постройте гистограмму параметра `SalePrice` всех домов с заголовком `'Histogram of Sale Price'`, заголовок оси x – `'price'`, заголовок оси y – `'count'`.

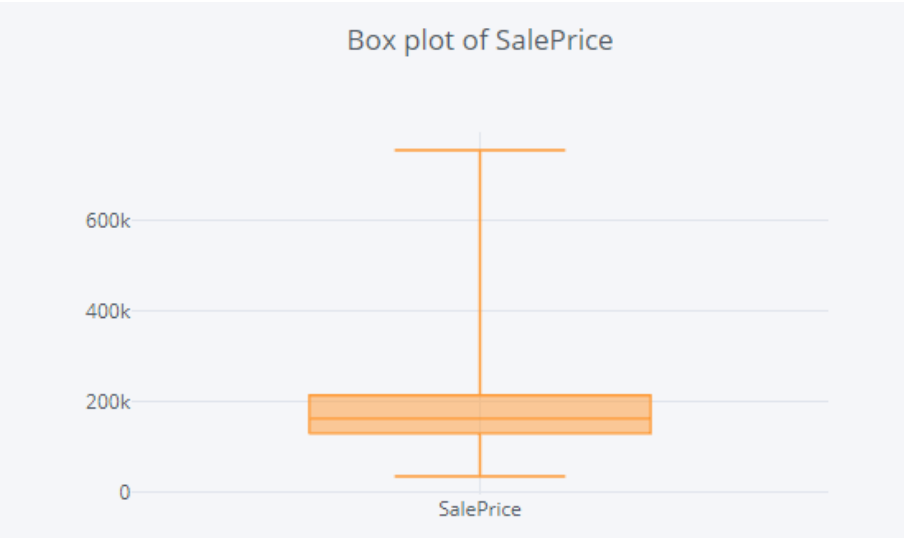
Результат:



6.2. **Boxplot**

Построить коробочную диаграмму (ящик с усами) признака `SalePrice` всех домов в данных. Боксплоты не показывают форму распределения, но они могут дать нам лучшее представление о центре и распространении распределения, а

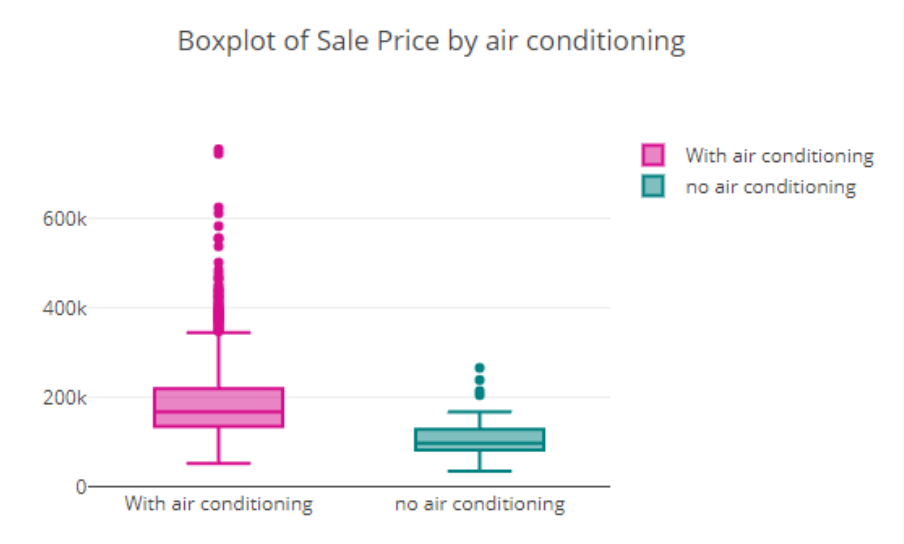
также о любых возможных выбросах, которые могут существовать. Боксплоты и гистограммы часто дополняют друг друга и помогают нам лучше понять данные. Заголовок рисунка – `title='Box plot of SalePrice'`. Результат:



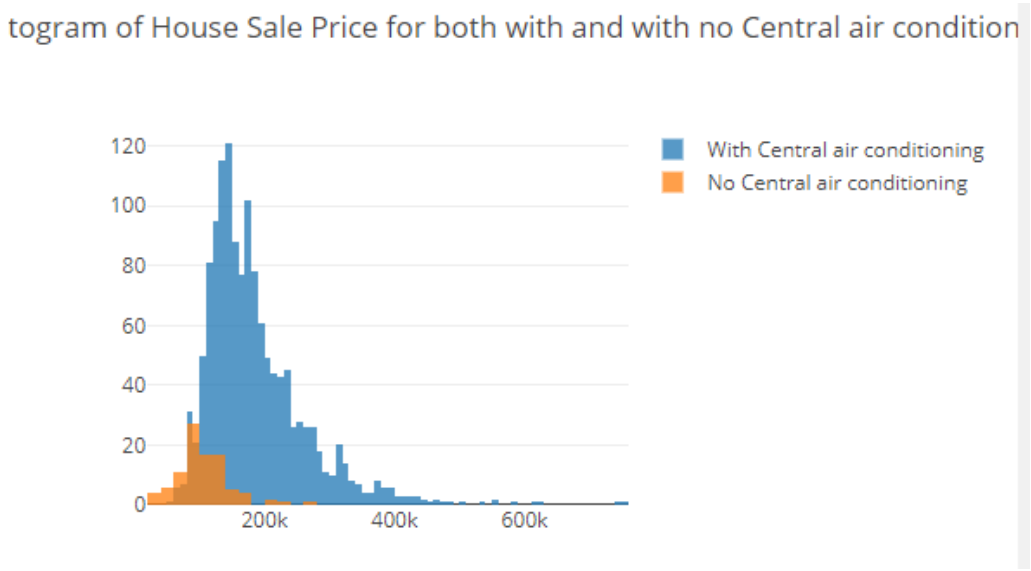
6.3. **Гистограммы и Боксплоты по группам**

На графиках по группам, мы можем видеть, как переменная меняется в ответ на изменение другой, например, как меняется стоимость дома `SalePrice` в зависимости от того, есть ли кондиционер или нет (параметр `'CentralAir'`). Или, как цена дома `SalePrice` зависит от размера гаража и т.д.

а) Для построения Boxplot и гистограмм цены дома сгруппируем данные с кондиционером (`name = 'With air conditioning'`) и без кондиционера (name = `'No air conditioning'); для` Boxplot `title = "Boxplot of Sale Price by air conditioning"`. Результат:



б) Для гистограммы – заголовок `title='Histogram of House Sale Price for both with and with no Central air conditioning'`



в) Выведем описательную статистику `'CentralAir'` и `'SalePrice'` с помощью инструкции:

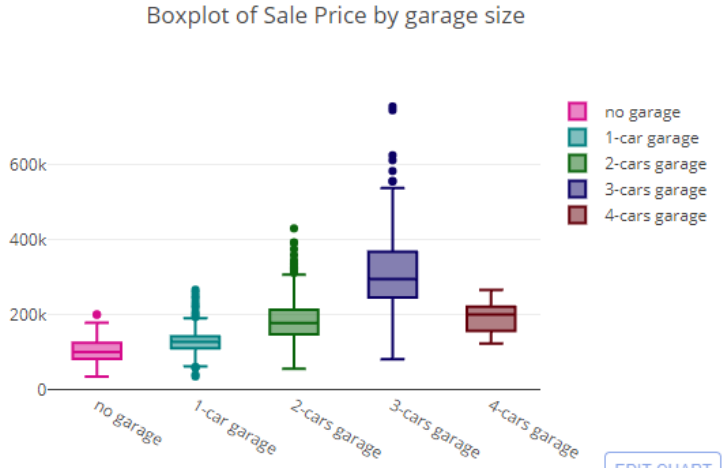
```
df.groupby('CentralAir')['SalePrice'].describe()
```

Выход: `'N'` – стоимость дома без кондиционера; `'Y'` – стоимость дома с кондиционером. Очевидно, что средняя цена продажи домов без кондиционера намного ниже, чем у домов с кондиционером.

г) Постройте Boxplot и гистограмму цены продажи домов (параметр `'SalePrice'`), сгруппированные по размеру гаража (параметр `'GarageCars'`): `title = "Boxplot of Sale Price by garage size".`

Используйте при группировке `name = 'no garage'` и `name = '1-car garage'` – если гараж для одной машины; `name = '2-car garage'` – если гараж для двух машин; `name = '3-car garage'` – если гараж для трех машин; `name = '4-car garage'` – если гараж для четырех машин.

Результат:



Судить о средней цене дома можно по черте внутри каждого блока. Прокомментируйте среднюю цену дома в зависимости от размеров гаража.

Постройте:

д) **Гистограмму цены продажи дома без гаража**

```
title='Histogram of Sale Price of houses with no garage'
```

е) **Гистограмму цены продажи дома с гаражом на 1 машину**

```
title='Histogram of Sale Price of houses with 1-car garage'
```

ё) **Гистограмму цены продажи дома с гаражом на 2 машины**

```
title='Histogram of Sale Price of houses with 2-car garage')
```

ж) **Гистограмму цены продажи дома с гаражом на 3 машины**

```
title='Histogram of Sale Price of houses with 3-car garage'
```

з) **Гистограмму цены продажи дома с гаражом на 4 машины**

```
title='Histogram of Sale Price of houses with 4-car garage'
```

6.4. **Таблица частот**

Значение частоты говорит о том, как часто какая-то величина встречается в выборке.

Общую таблицу частот параметров можно получить, исполнив программный код:

```
x = df.OverallQual.value_counts()
x/x.sum()
```

а) Таблица частот размеров гаража

```
x = df.GarageCars.value_counts()
x/x.sum()
```

Гаражи какого размера наиболее распространены?

б) Таблица частот центрального кондиционирования

```
x = df.CentralAir.value_counts()
x/x.sum()
```

6.5. Числовые характеристики параметров

Быстрый способ получить набор числовых сводок для количественной переменной – это использовать метод `description`.

```
df.SalePrice.describe()
```

– Рассчитайте долю домов с продажной ценой между 25-м процентилем (129975) и 75-м процентилем (214000).

```
print('The proportion of the houses with prices between
25th percentile and 75th percentile: ',
```

```
np.mean((df.SalePrice >= 129975) & (df.SalePrice <=
214000)))
```

– Рассчитайте долю домов с общей площадью в квадратных футах от 25-го процентиля (795,75) до 75-го процентиля (1298,25).

```
print('The proportion of house with total square feet of
basement area between 25th percentile and 75th percentile:
', np.mean((df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <=
1298.25)))
```

```
The proportion of the houses with prices between 25th percentile and 75th perc
entile:  0.5020547945205479
```

– Наконец, мы рассчитаем долю домов на основе любых условий. Поскольку некоторые дома соответствуют обоим критериям, приведенная ниже пропорция меньше суммы двух пропорций, рассчитанных выше.

```
a = (df.SalePrice >= 129975) & (df.SalePrice <= 214000)
b = (df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <=
1298.25)
print(np.mean(a | b))
```

```
0.7143835616438357
```

– Рассчитать цену продажи интерквартильного размаха IQR для домов без кондиционера. Англоязычная аббревиатура IQR – интерквартильный размах – число, которое показывает разброс средней половины (т.е. средние 50%) набора данных и помогает определить выбросы. IQR – это разница между третьим квартилем (Q3) и первым(Q1).

```
q75, q25 =
np.percentile(df.loc[df['CentralAir']=='N']['SalePrice'],
[75,25])
iqr = q75 - q25
print('Sale price IQR for houses with no air conditioning:
', iqr)
```

```
Sale price IQR for houses with no air conditioning:  46500.0
```

– Рассчитать цену продажи IQR для домов с кондиционером

```
q75, q25 =
np.percentile(df.loc[df['CentralAir']=='Y']['SalePrice'],
[75,25])
iqr = q75 - q25
print('Sale price IQR for houses with air conditioning: ',
iqr)
```

```
Sale price IQR for houses with air conditioning:  84410.0
```

6.6. Стратификация

Другой способ получить больше информации из набора данных – разделить его на более мелкие, более однородные подмножества и проанализировать каждое из этих подмножеств.

– Создадим новый столбец HouseAge (YearBuilt), затем разделим данные на части HouseAge и построим параллельные боксовые диаграммы цены продажи внутри каждой страты.

```
df['HouseAge'] = 2019 - df['YearBuilt']
df["AgeGrp"] = pd.cut(df.HouseAge, [9, 20, 40, 60, 80,
100, 147]) # Create age strata based on these cut points
plt.figure(figsize=(12, 5))
sns.boxplot(x="AgeGrp", y="SalePrice", data=df)
```

Вопрос: Сравните возраст дома и среднюю цену на него. Какая наблюдается закономерность?

– Ранее мы узнали, что цены на жилье, как правило, различаются по наличию или отсутствию кондиционеров в них. Из построенных по приведенному ниже коду графиков сделайте вывод: какие по возрасту дома более оснащены кондиционерами.

```
plt.figure(figsize=(12, 5))
sns.boxplot(x="YearBuilt", y="SalePrice",
hue="CentralAir", data=df)
plt.show()
```

– Теперь мы сгруппируем сначала по кондиционированию воздуха, а затем в группе кондиционирования по возрастным группам. Каждый подход подчеркивает различные аспекты данных.

```
plt.figure(figsize=(12, 5))
sns.boxplot(x="CentralAir", y="SalePrice",
hue="YearBuilt", data=df)
plt.show()
```

– Мы также можем рассортировать по возрасту дома и кондиционированию воздуха, чтобы исследовать, как тип здания зависит от обоих этих факторов одновременно.

```
df1 = df.groupby(["YearBuilt", "CentralAir"])["BldgType"]
df1 = df1.value_counts()
df1 = df1.unstack()
df1 = df1.apply(lambda x: x/x.sum(), axis=1)
print(df1.to_string(float_format="%.3f"))
```

Для всех домовых возрастных групп, в большинстве случаев данные о жилье представлены в столбце 1Fam. Чем старше дом, тем вероятнее, что нет кондиционера. Тем не менее, для дома 1Fam – старше 100 лет, скорее всего, есть

кондиционер, чем нет. Не было ни очень новых, ни очень старых типов дуплексов. Для дуплексного дома в 40–60 лет, скорее всего, нет кондиционера.

6.6. Отображения диаграмм

– Построим диаграммы рассеяния, отражающие взаимосвязь между переменными:

```
output,var,var1,var2 = 'SalePrice', 'GrLivArea',
'TotalBsmtSF', 'OverallQual'
fig, axes = plt.subplots(nrows=1,ncols=3,figsize=(16,5))
df.plot.scatter(x=var,y=output,ylim=(0,800000),ax=axes[0])
df.plot.scatter(x=var1,y=output,ylim=(0,800000),ax=axes[1]
)
df.plot.scatter(x=var2,y=output,ylim=(0,800000),ax=axes[2]
)
```

Наблюдается ли взаимосвязь между переменными?

– Отображение прямоугольной диаграммой

```
fig, ax = plt.subplots(figsize=(8,6))
sns.boxplot(x=var2,y=output,data=df)
ax.set_ylim(0,800000)
plt.show()
```

Между какими параметрами построен `boxplot`? Есть ли выбросы в данных?

– Рассмотрите эффект, нарисованный seaborn:

```
var3 = 'YearBuilt'
fig, ax = plt.subplots(figsize=(16,8))
sns.boxplot(x=var3,y=output,data=df)
ax.set_ylim(0,800000)
plt.xticks(rotation=90)
plt.show()
```

С помощью диаграмм рассеяния и коробчатых диаграмм мы можем обнаружить, что признаки `GrLivArea` и `TotalBsmtSF` имеют линейную связь с `SalePrice`; общий признак `OverallQual` и `YearBulit` также связаны с `SalePrice`. Проанализируем другие переменные. Тогда нам нужно подобрать определенные функции и методы, чтобы не определять отдельно отношения между ними.

6.7. Отображение тепловой карты характеристической ковариационной матрицы

– Получите ковариационную матрицу для всех данных DataFrame и используйте анализ тепловой карты:

```
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap (corrmat, vmax = .8, square = True, ax = ax) # Параметр square гарантирует, что когда corrmat - неквадратная матрица, общий вывод графика по-прежнему будет квадратным
plt.show()
```

– Давайте выберем 10 параметров с наибольшей корреляцией с `SalePrice`, чтобы проанализировать корреляцию между ними.

```
k = 10
top10_attr = corrmat.nlargest(k, output).index
top10_mat = corrmat.loc[top10_attr, top10_attr]
fig,ax = plt.subplots(figsize=(8,6))
sns.set(font_scale=1.25)
sns.heatmap(top10_mat, annot=True, annot_kws={'size':12}, square=True)
# Установите аннотацию для отображения чисел в маленьких ячейках и annot_kws для настройки числового формата
plt.show()
```

Что обнаружено:

– `OverallQual`, `TotalBsmtSF` имеют сильную корреляцию с `SalePrice`;

– `GarageCars` и **`GarageArea`**, практически, тоже одно и то же. Они имеют сильную корреляцию с `SalePrice` (0,88). Эти два признака связаны, поэтому достаточно сохранить признак с более высоким коэффициентом корреляции – `GarageCars` (0,64).

–Точно так же оставим `TotalBsmtSF`, отбросим **`1stFloor`** и **`TotRmsAbvGrd`** связаны с `GrLivArea`, поэтому отброшены;

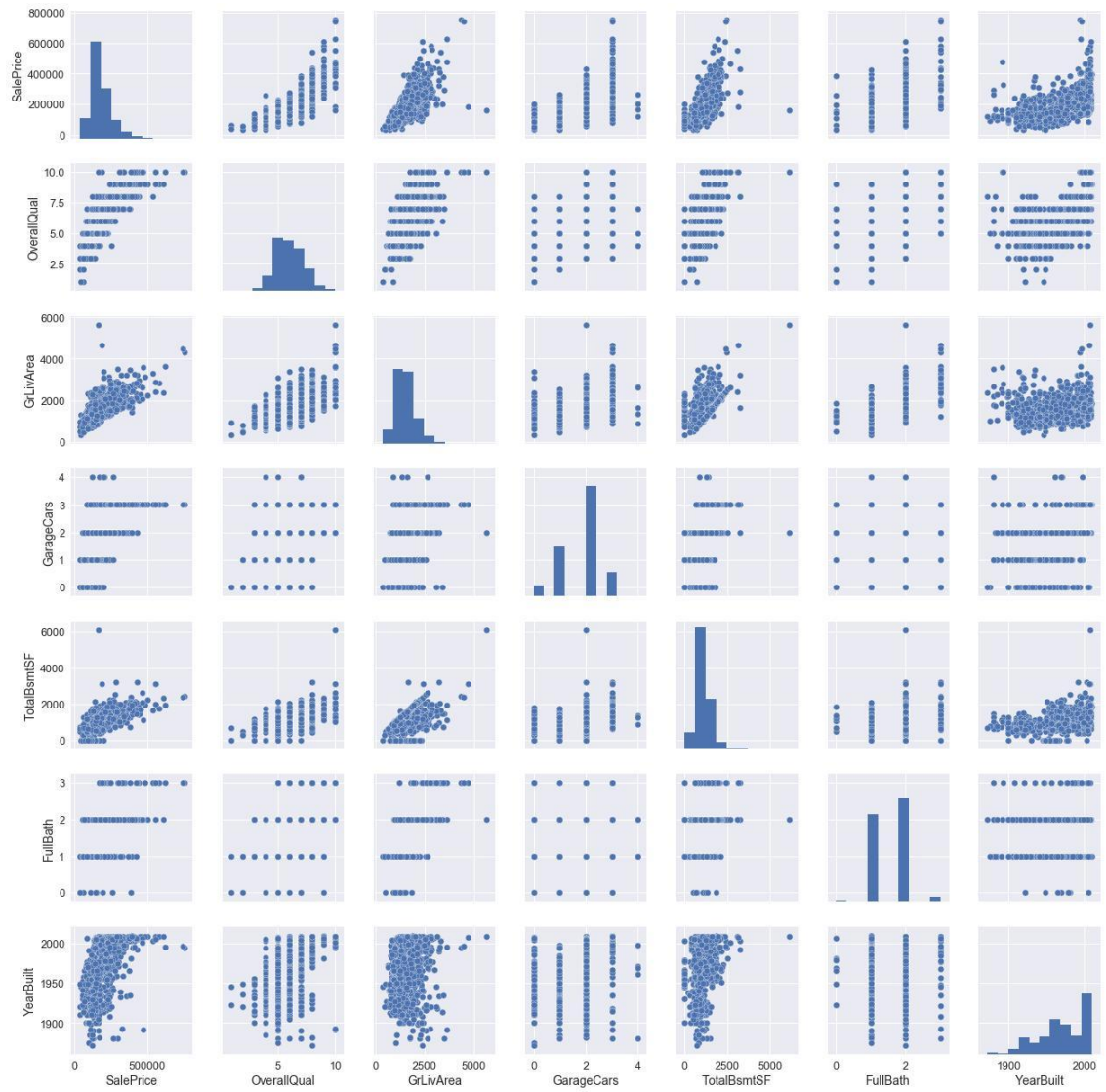
– Оставим `FullBath` и `YearBuilt`.

6.7. Графическая матрица, объединяющая точечные и столбчатые диаграммы

Seaborn может интегрировать информацию о диаграммах рассеяния и гистограммах нескольких объектов. В результате получим матрицу графиков, образованную комбинациями между каждыми двумя параметрами.

Постройте выбранные параметры следующим образом:

```
var_set = ['SalePrice', 'OverallQual', 'GrLivArea',
'GarageCars', 'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.set (font_scale = 1.25) # Устанавливаем размер шрифта
по горизонтальной и вертикальной оси
sns.pairplot (df[var_set]) # 7 * 7 графическая матрица
# Различные типы отображения могут быть установлены в
параметрах kind и diag_kind, вот диаграммы разброса и
гистограммы, и вы также можете установить разные типы
отображения на каждом графике
plt.show()
```



==========