2019

PYTHON TERM PROJECT

COMPREHENSIVE DATA EXPLORATION FOR HOUSE PRICES

This final project aim to predict sales prices of houses by using With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa (central Iowa in America).

MEF UNIVERSITY
LEYLA YİĞİT
1/17/2019

Table of Contents

COMPREHENSIVE DATA EXPLORATION FOR HOUSE PRICES

PURPOSE AND OBJECTIVES OF THE PROJECT

This data is served for a Kaggle competition the aim is to predict sales prices of houses by using with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa (central Iowa in America)

PROJECT GUIDELINE

1. Understand the problem. We'll look at each variable and do a philosophical analysis about their meaning and importance for this problem.
2. Invariable study. We'll just focus on the dependent variable ('Sale Price') and try to know a little bit more about it.
3. Multivariate study. We'll try to understand how the dependent variable and independent variables relate.
4. Basic cleaning. We'll clean the dataset and handle the missing data, outliers and categorical variables.
5. Test assumptions. We'll check if our data meets the assumptions required by most multivariate techniques.

ABOUT DATA

File descriptions

train.csv - the training set
test.csv - the test set
Sample_submission.csv - a benchmark submission from a linear regression on year and month of sale, lot square footage, and number of bedrooms

Data fields of Sales_Price Table

Sample_submission	
Column	**Description**
Id	Unique ID
SalePrice	Selling Price of the House

Data fields of Test and Train Tables

Test and Train Tables	
Column	**Description**
Id	Unique ID

MSSubClass	The building class
MSZoning	The general zoning classification
LotFrontage	Linear feet of street connected to property
LotArea	Lot size in square feet
Street	Type of road access
Alley	Type of alley access
LotShape	General shape of property
LandContour	Flatness of the property
Utilities	Type of utilities available
LotConfig	Lot configuration
LandSlope	Slope of property
Neighborhood	Physical locations within Ames city limits
Condition1	Proximity to main road or railroad
Condition2	Proximity to main road or railroad (if a second is present)
BldgType	Type of dwelling
HouseStyle	Style of dwelling
OverallQual	Overall material and finish quality
OverallCond	Overall condition rating
YearBuilt	Original construction date
YearRemodAdd	Remodel date
RoofStyle	Type of roof
RoofMatl	Roof material
Exterior1st	Exterior covering on house
Exterior2nd	Exterior covering on house (if more than one material)
MasVnrType	Masonry veneer type
MasVnrArea	Masonry veneer area in square feet
ExterQual	Exterior material quality
ExterCond	Present condition of the material on the exterior
Foundation	Type of foundation
BsmtQual	Height of the basement
BsmtCond	General condition of the basement
BsmtExposure	Walkout or garden level basement walls
BsmtFinType1	Quality of basement finished area
BsmtFinSF1	Type 1 finished square feet
BsmtFinType2	Quality of second finished area (if present)
BsmtFinSF2	Type 2 finished square feet
BsmtUnfSF	Unfinished square feet of basement area
TotalBsmtSF	Total square feet of basement area
Heating	Type of heating
HeatingQC	Heating quality and condition
CentralAir	Central air conditioning
Electrical	Electrical system

1stFlrSF	First Floor square feet
2ndFlrSF	Second floor square feet
LowQualFinSF	Low quality finished square feet (all floors)
GrLivArea	Above grade (ground) living area square feet
BsmtFullBath	Basement full bathrooms
BsmtHalfBath	Basement half bathrooms
FullBath	Full bathrooms above grade
HalfBath	Half baths above grade
BedroomAbvGr	Number of bedrooms above basement level
KitchenAbvGr	Number of Kitchen above basement level
KitchenQual	Kitchen quality
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms)
Functional	Home functionality rating
Fireplaces	Number of fireplaces
FireplaceQu	Fireplace quality
GarageType	Garage location
GarageYrBlt	Year garage was built
GarageFinish	Interior finish of the garage
GarageCars	Size of garage in car capacity
GarageArea	Size of garage in square feet
GarageQual	Garage quality
GarageCond	Garage condition
PavedDrive	Paved driveway
WoodDeckSF	Wood deck area in square feet
OpenPorchSF	Open porch area in square feet
EnclosedPorch	Enclosed porch area in square feet
3SsnPorch	Three season porch area in square feet
ScreenPorch	Screen porch area in square feet
PoolArea	Pool area in square feet
PoolQC	Pool quality
Fence	Fence quality
MiscFeature	Miscellaneous feature not covered in other categories
MiscVal	$Value of miscellaneous feature
MoSold	Month Sold
YrSold	Year Sold
SaleType	Type of sale
SaleCondition	Condition of sale

ANALYSING SALES PRICE

1. SEE AND CHECK DATA TYPES

```
import pandas as pd #Analysis
import numpy as np #Analysis
from scipy.stats import norm #Analysis
from sklearn.preprocessing import StandardScaler #Analysis
from scipy import stats #Analysis
import matplotlib.pyplot as plt #Visulization
import seaborn as sns #Visulization
import warnings
warnings.filterwarnings('ignore')
import gc

#bring in the six packs
#see data
df_train = pd.read_csv('C:/Users/Kafein/PycharmProjects/Introduction to
Python/Class/CLASS_FINAL_WORKS/FINAL/train.csv')
df_test = pd.read_csv('C:/Users/Kafein/PycharmProjects/Introduction to
Python/Class/CLASS_FINAL_WORKS/FINAL/test.csv')
df_sample_submission =
pd.read_csv('C:/Users/Kafein/PycharmProjects/Introduction to
Python/Class/CLASS_FINAL_WORKS/FINAL/sample_submission.csv')

print(df_train.tail())
print(df_test.tail())
print(df_sample_submission.tail())


 #df_train
   Id  MSSubClass MSZoning    ...      SaleType  SaleCondition SalePrice
1455  1456          60        RL    ...           WD          Normal    175000
1456  1457          20        RL    ...           WD          Normal    210000
1457  1458          70        RL    ...           WD          Normal    266500
1458  1459          20        RL    ...           WD          Normal    142125
1459  1460          20        RL    ...           WD          Normal    147500
[5 rows x 81 columns]

 #df_test

    Id  MSSubClass MSZoning      ...       YrSold  SaleType SaleCondition
1454  2915         160        RM      ...         2006         WD          Normal
1455  2916         160        RM      ...         2006         WD         Abnorml
1456  2917          20        RL      ...         2006         WD         Abnorml
1457  2918          85        RL      ...         2006         WD          Normal
1458  2919          60        RL      ...         2006         WD          Normal
[5 rows x 80 columns]

#Sample_submission
 Sales_Price                  Unnamed: 1
0      Column                 Description
1          Id                   Unique ID
2   SalePrice  Selling Price of the House

#Shape of data
print("train.csv. Shape: ",df_train.shape)
print("test.csv. Shape: ",df_test.shape)
print("sample_submission.csv. Shape: ",df_sample_submission.shape)
```

```
train.csv. Shape:  (1460, 81)
test.csv. Shape:  (1459, 80)
sample_submission.csv. Shape:  (3, 2)
```

```
#Is there any dublicate ID
A=df_train.duplicated('Id')
print(sum(i for i in A if i == True))  #check any dublicate id
0
```

- There is no dublicate Id in the data

```
#Check Data Types
```

```
print(df_train.dtypes)
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
Alley             object
LotShape          object
LandContour       object
Utilities         object
LotConfig         object
LandSlope         object
Neighborhood      object
Condition1        object
Condition2        object
BldgType          object
HouseStyle        object
OverallQual        int64
OverallCond        int64
YearBuilt          int64
YearRemodAdd       int64
RoofStyle         object
RoofMatl          object
Exterior1st       object
Exterior2nd       object
MasVnrType        object
MasVnrArea       float64
ExterQual         object
ExterCond         object
Foundation        object
                   ...
BedroomAbvGr       int64
KitchenAbvGr       int64
KitchenQual       object
TotRmsAbvGrd       int64
Functional        object
Fireplaces         int64
FireplaceQu       object
GarageType        object
GarageYrBlt      float64
GarageFinish      object
```

```
GarageCars          int64
GarageArea          int64
GarageQual         object
GarageCond         object
PavedDrive         object
WoodDeckSF          int64
OpenPorchSF         int64
EnclosedPorch       int64
3SsnPorch           int64
ScreenPorch         int64
PoolArea            int64
PoolQC             object
Fence              object
MiscFeature        object
MiscVal             int64
MoSold              int64
YrSold              int64
SaleType           object
SaleCondition      object
SalePrice           int64
Length: 81, dtype: object

print(df_sample_submission.dtypes)
Sales_Price    object
Unnamed: 1     object
dtype: object
```

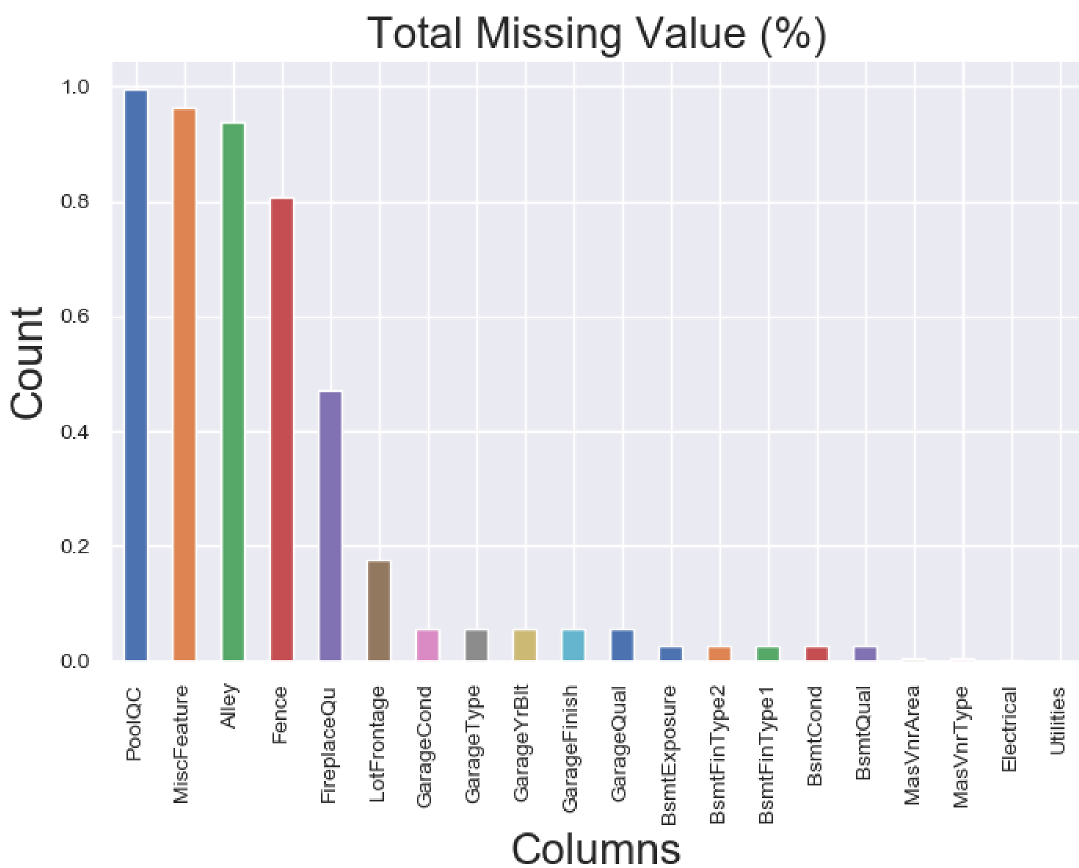
2. ANALYZE DATA

A. Finding Missing values

```
#missing data
total = df_train.isnull().sum().sort_values(ascending=False)
percent =
(df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=F
alse)
missing_data = pd.concat([total, percent], axis=1, keys=['Total',
'Percent'])
missing_data.head(20)

             Total   Percent
PoolQC        1453  0.995205
MiscFeature   1406  0.963014
Alley         1369  0.937671
Fence         1179  0.807534
FireplaceQu    690  0.472603
LotFrontage    259  0.177397
GarageCond      81  0.055479
GarageType      81  0.055479
GarageYrBlt     81  0.055479
GarageFinish    81  0.055479
GarageQual      81  0.055479
BsmtExposure    38  0.026027
BsmtFinType2    38  0.026027
BsmtFinType1    37  0.025342
BsmtCond        37  0.025342
BsmtQual        37  0.025342
MasVnrArea       8  0.005479
MasVnrType       8  0.005479
Electrical       1  0.000685
Utilities        0  0.000000
```

#missing data histogram

```python
#missing data
total = df_train.isnull().sum().sort_values(ascending=False)
percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
#histogram
#missing_data = missing_data.head(20)
percent_data = percent.head(20)
percent_data.plot(kind="bar", figsize = (8,6), fontsize = 10)
plt.xlabel("Columns", fontsize = 20)
plt.ylabel("Count", fontsize = 20)
plt.title("Total Missing Value (%)", fontsize = 20)
```

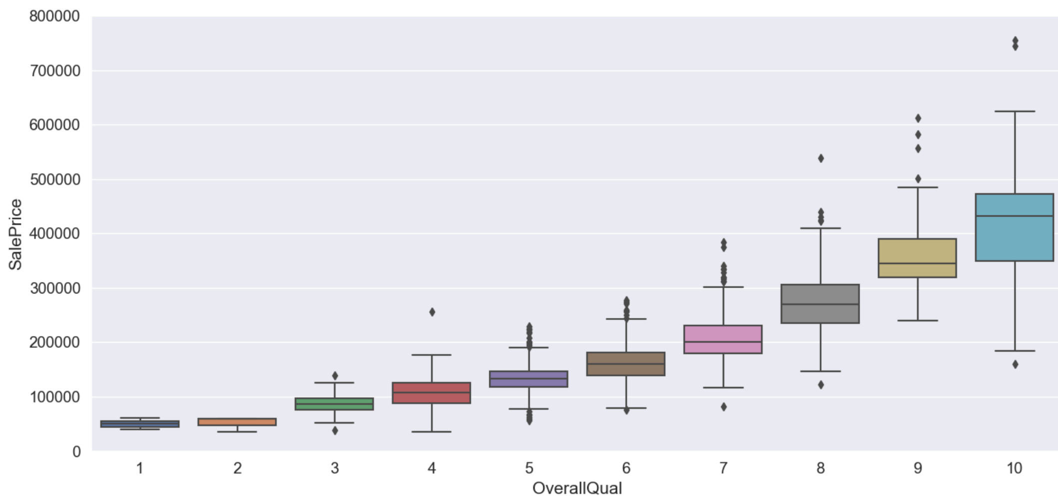


- We'll consider that when more than 15% of the data is missing, we should delete the corresponding variable and pretend it never existed. This means that we will not try any trick to fill the missing data in these cases. According to this, there is a set of variables (e.g. 'PoolQC', 'MiscFeature', 'Alley', etc.) that we should delete.

B. Outliers

```python
#outliers
data = pd.concat([df_train['SalePrice'], df_train['OverallQual']], axis=1)
f, ax = plt.subplots(figsize=(16, 10))
```

```
fig = sns.boxplot(x='OverallQual', y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```



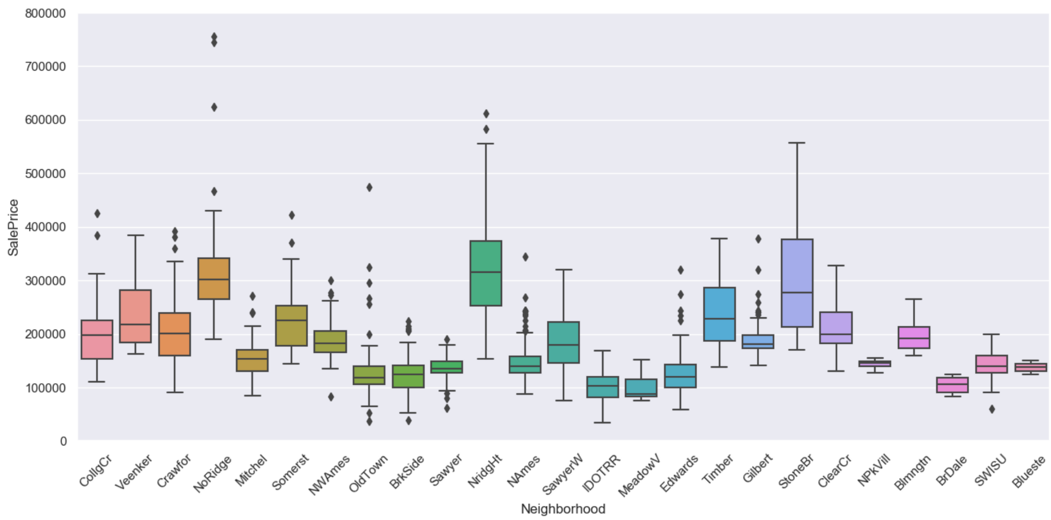
- OverallQual : 4
- OverallQual : 8
- OverallQual : 10

OUTLIER DATA

```
df_train[df_train['OverallQual'] == 4][df_train['SalePrice'] > 200000]
#outlier 4 , we see from the graph that outlier 4 bigger than 200000
df_train = df_train[df_train['Id'] != 458]
df_train[df_train['OverallQual'] == 8][df_train['SalePrice'] > 500000]#outlier 8
df_train[df_train['OverallQual'] == 10][df_train['SalePrice'] < 180000]#outlier 10
```

I think that I have to remove all the outliers

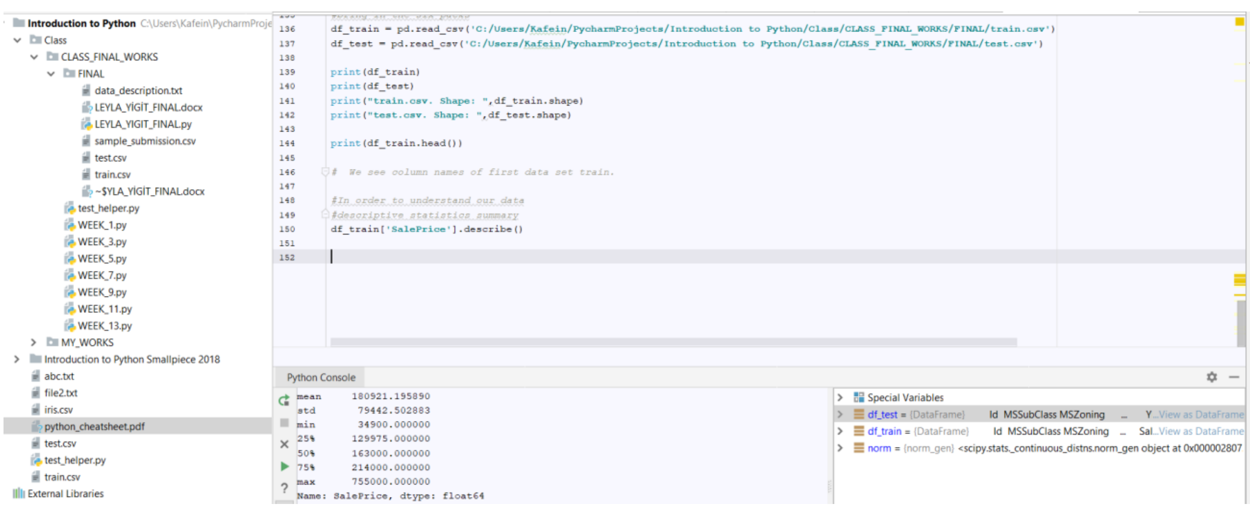
```
#remove all the outliers
df_train = df_train[df_train['Id'] != 524][df_train['Id'] != 1299]
var = 'Neighborhood'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 10))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
xt = plt.xticks(rotation=45)
```



- The fluctuation of saleprice seems to be large by neighbor.

3. STATISTICAL CONTROL

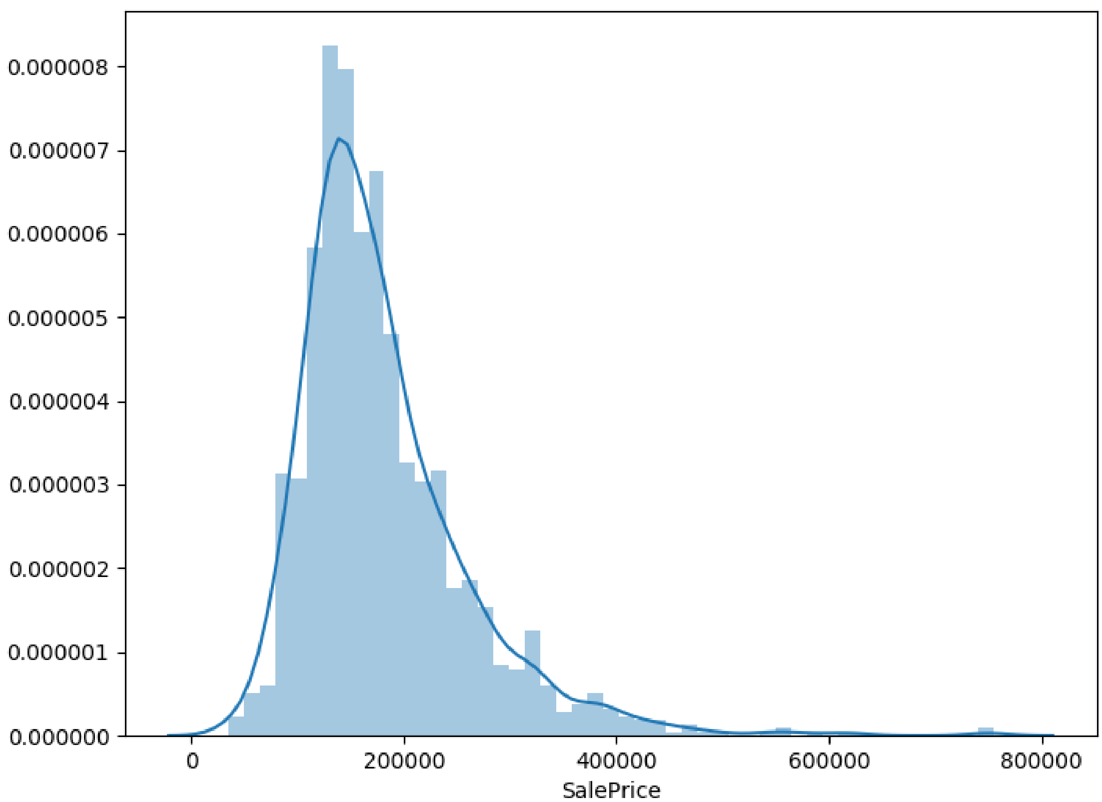
C. For Basic Statistical Control:




```
mean      180921.195890
std        79442.502883
min        34900.000000
25%       129975.000000
50%       163000.000000
75%       214000.000000
max       755000.000000
Name: SalePrice, dtype: float64 mean      180921.195890
std        79442.502883
min        34900.000000
25%       129975.000000
50%       163000.000000
75%       214000.000000
max       755000.000000
Name: SalePrice, dtype: float64
```

- The std is big.
- min is greater than 0
- There is a big difference between the minimum value and the 25th percentile.
- It's bigger than the 75th percentile and max.
- The difference between the 75th percentile and the max is greater than the 25th percentile and the max.

```
#histogram
f, ax = plt.subplots(figsize=(8, 6))
sns.distplot(df_train['SalePrice'])
```



- Long tail formation to the right (not normal distribution)

```
#skewness and kurtosis
print("Skewness: %f" % df_train['SalePrice'].skew())
print("Kurtosis: %f" % df_train['SalePrice'].kurt())

Skewness: 1.882876
Kurtosis: 6.536282
```

- Kurtosis (kurtosis / kurtosis): If the kurtosis value (K) is close to 3, the scatter is close to the normal distribution. (K <3), the distributions can be judged to be flattened more smoothly than the normal distribution, and if the kurtosis is a positive number larger than 3 (K> 3), the distribution can be considered to be a more pointed distribution than the normal distribution

4. RELATIONSHIP WITHIN VARIABLES

SalePrice correlation matrix (zoomed heatmap style)

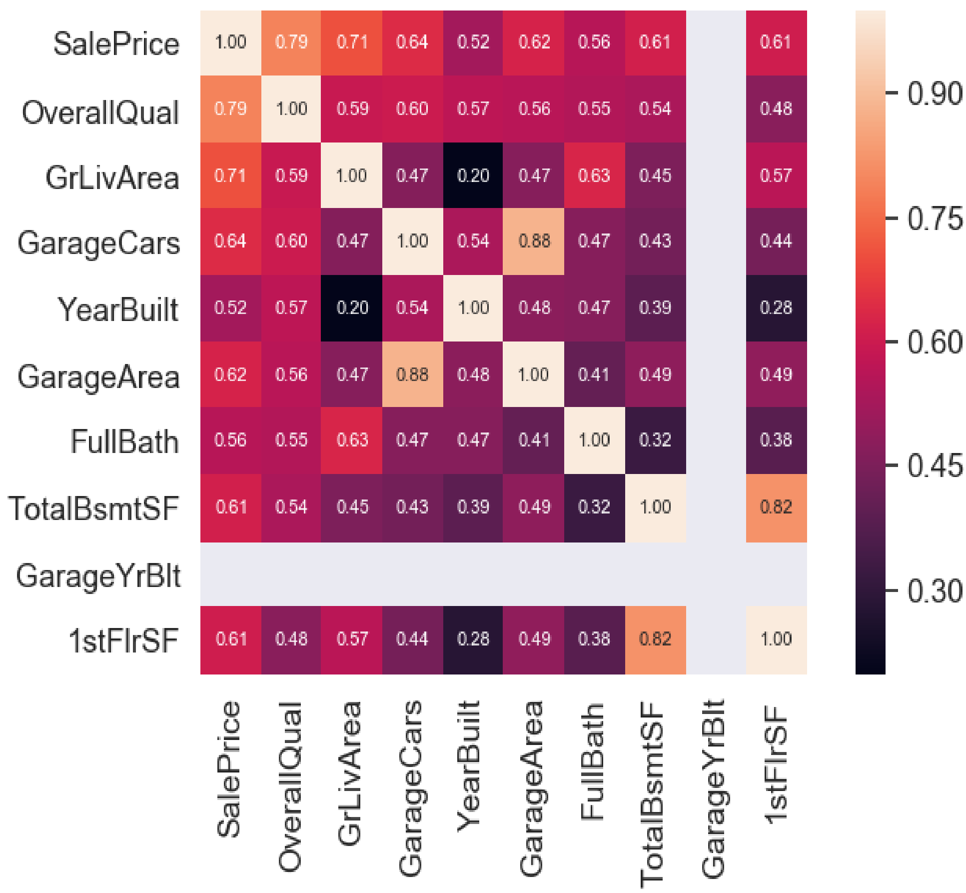
D. Pearson Product Ratio Correlation

- **Pearson correlation evaluates** the linear relationship between two metric variables. There is a linear relationship when the variation of one variable is proportional to the change of another variable.
- For example, Pearson correlation can be used to assess whether the increase in temperature in a production facility is related to changes in the thickness of the chocolate coating.

Spearman Rank Correlation

- Spearman correlation evaluates the simple relationship between two metric or sequential variables. In a simple relationship, the two variables tend to change together, but not necessarily at a constant rate. The Spearman correlation coefficient is based on the ranked value for each variable, not the raw data.
- Spearman correlation is often used to evaluate relationships containing sequential variables. For example, you can use Spearman correlation to assess whether the order in which employees complete the test exercises is related to the number of months employed.

```
#saleprice correlation matrix
k = 10 #number of variables for heatmap
corrmat = df_train.corr(method='spearman') # correlation
cols = corrmat.nlargest(k, 'SalePrice').index # nlargest : Return this many
descending sorted values
cm = np.corrcoef(df_train[cols].values.T) # correlation
sns.set(font_scale=1.25)
f, ax = plt.subplots(figsize=(8, 6))
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',
annot_kws={'size': 8}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```

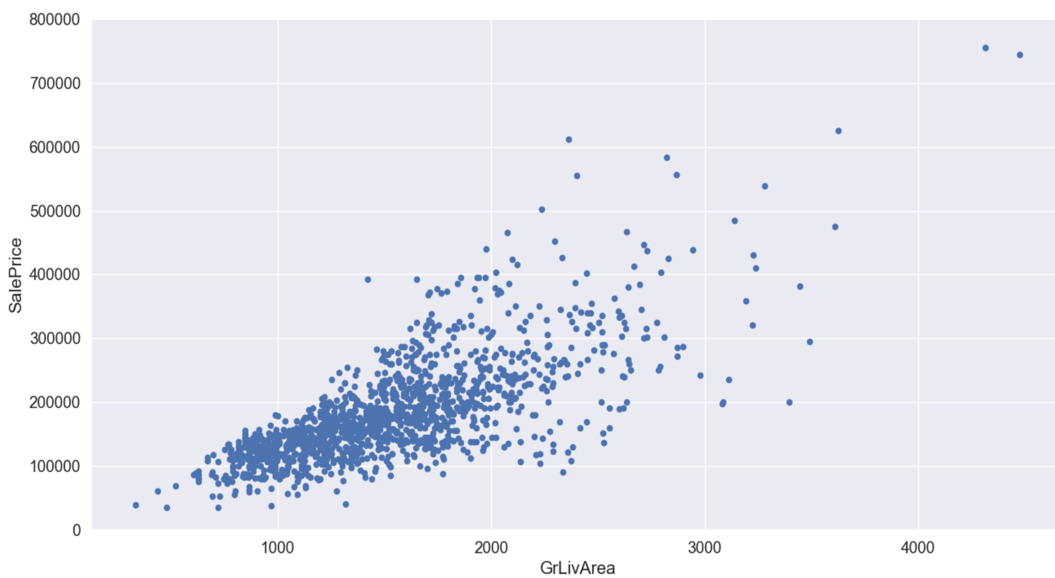


9 most relevant variables with SalePrice

- OverallQual : Overall material and finish quality
- GrLivArea : Above grade (ground : the portion of a home that is above the ground) living area square feet
- GarageCars : Size of garage in car capacity
- GarageArea : Size of garage in square feet
- TotalBsmtSF : Total square feet of basement area
- 1stFlrSF : First Floor square feet
- FullBath : Full bathrooms above grade
- TotRmsAbvGrd : Total rooms above grade (does not include bathrooms)
- YearBuilt : Original construction date

E. Relationship with numerical variables
grlivarea/saleprice

```
#scatter plot grlivarea/saleprice

"""3. The scatterplot
The scatterplot helps us visualize the relationship between two or more
variables. The code needed to execute a
scatterplot is shown below:"""
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



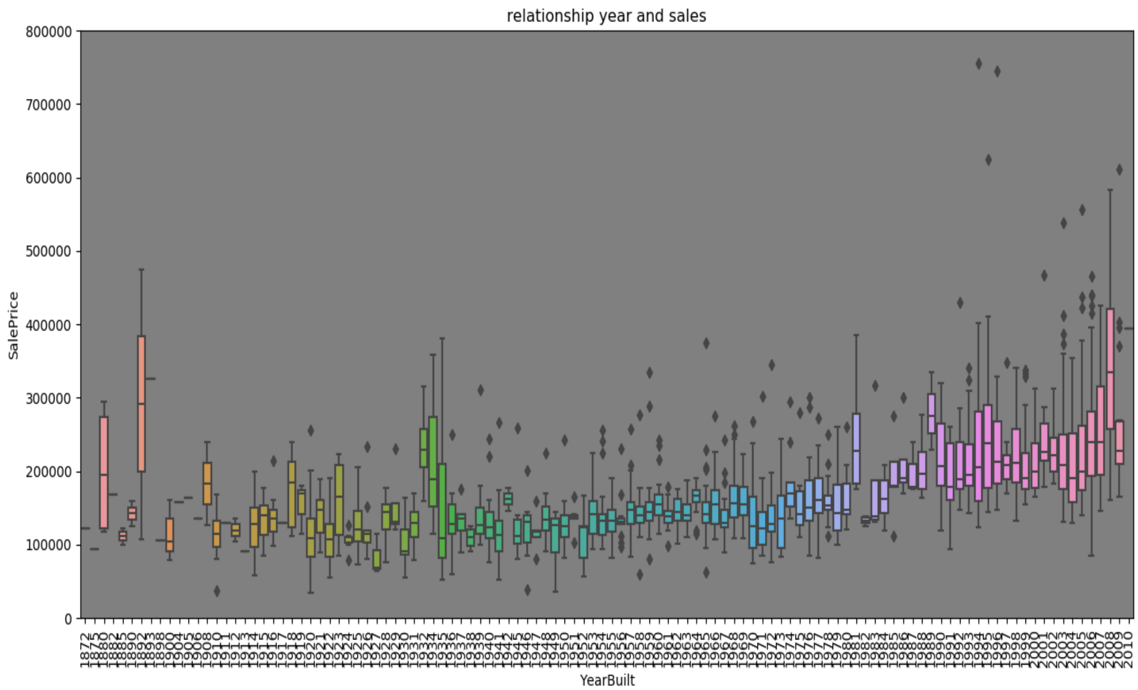
- There is a strong relationship between *'SalePrice' and 'GrLivArea' . There is a a linear relationship.*

And what about 'TotalBsmtSF'?

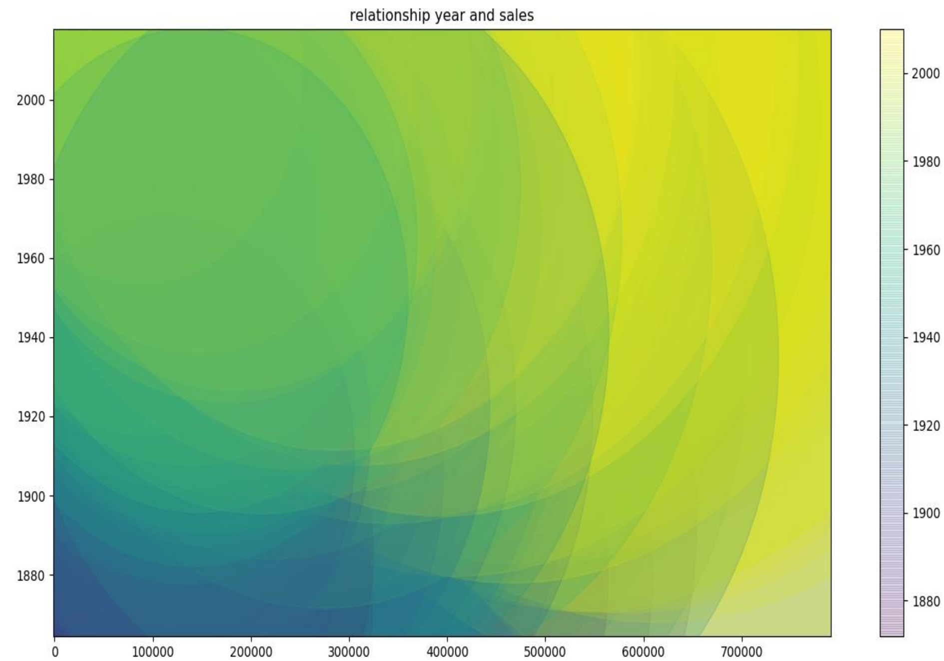
- *'TotalBsmtSF' has an impact on 'SalePrice' but this seems a much more depends on money. Everything is ok and suddenly, in a strong linear (exponential?) reaction, everything changes. Moreover, it's clear that sometimes 'TotalBsmtSF' closes in itself and gives zero credit to 'SalePrice'.*

F. Relationship with categorical features

```
var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
plt.title("relationship year and sales")
fig.patch.set_facecolor('grey') #change colors
plt.xticks(rotation=90);
```

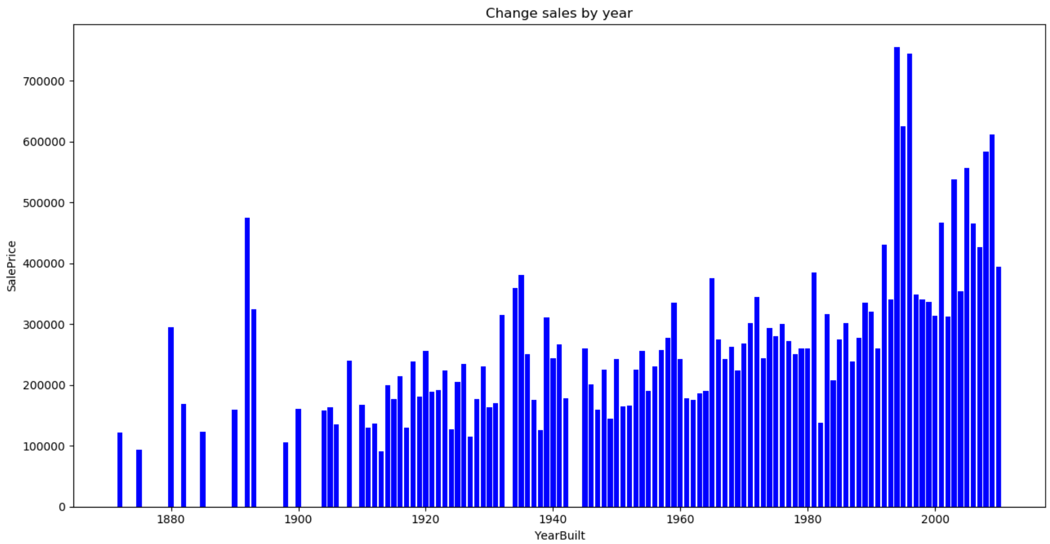



```
#TRY DİFFERENT GRAPH
var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)

x = df_train['SalePrice']
y = df_train['YearBuilt']
colors =df_train['YearBuilt']
sizes = df_train['SalePrice']

plt.scatter(x, y, c=colors, s=sizes, alpha=0.2,
            cmap='viridis')
plt.colorbar();  # show color scale
```



- Although it's not a strong tendency, I'd say that 'SalePrice' is more prone to spend more money in new stuff than in old relics.
- *Note: we don't know if 'SalePrice' is in constant prices. Constant prices try to remove the effect of inflation. If 'SalePrice' is not in constant prices, it should be, so than prices are comparable over the years.*

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

var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
x = df_train['YearBuilt']
y = df_train['SalePrice']
# plt.plot(x,  color='blue')
plt.bar(x, y,color='Navy')
plt.xlabel('YearBuilt')
plt.ylabel('SalePrice')
plt.title('Change sales by year')
plt.show()
```



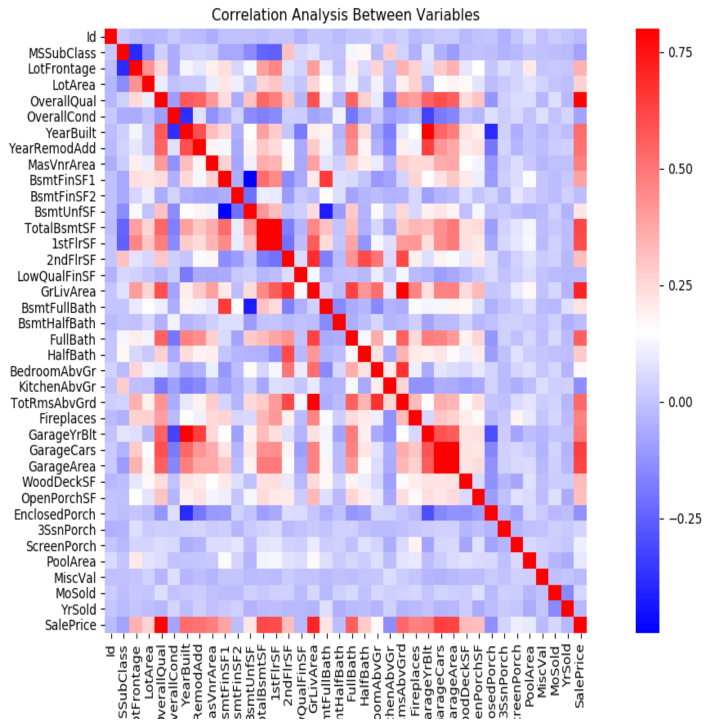
SUMMARY OF RELATIONSHIP BETWEEN VARIABLES

- *'GrLivArea' and 'TotalBsmtSF' seem to be linearly related with 'SalePrice'. Both relationships are positive, which means that as one variable increases, the other also increases. In the case of 'TotalBsmtSF', we can see that the slope of the linear relationship is particularly high.*
- *'OverallQual' and 'YearBuilt' also seem to be related with 'SalePrice'. The relationship seems to be stronger in the case of 'OverallQual', where the box plot shows how sales prices increase with the overall quality.*

5. CORRELATION MATRIX

G. correlation matrix

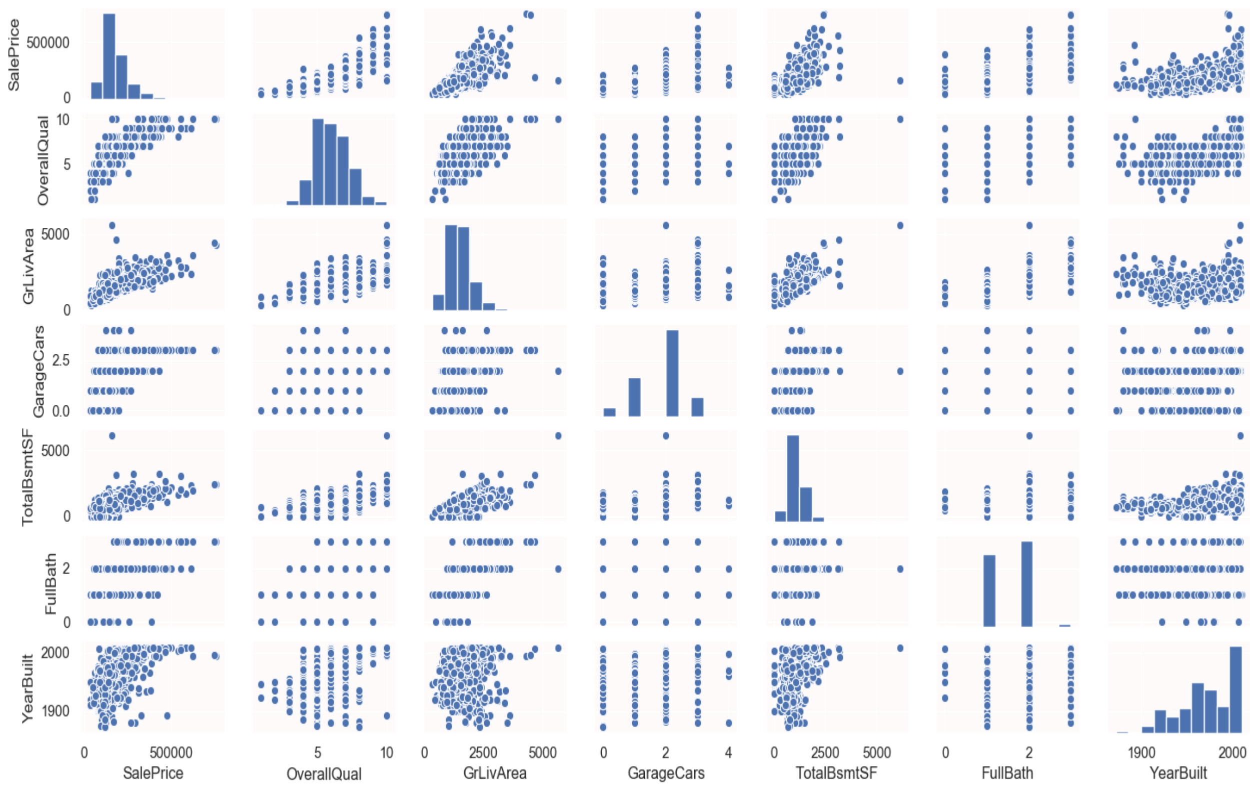
```
#correlation matrix
corrmat = df_train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True,cmap='bwr')
plt.title("Correlation Analysis Between Variables");
```



- *'TotalBsmtSF' and '1stFlrSF' variables are important variables, and the second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. Actually, this correlation is so strong that it can indicate a situation of multicollinearity. If we think about these variables, we can conclude that they give almost the same information so multicollinearity really occurs. Heatmaps are great to detect this kind of situations and in problems dominated by feature selection, like ours, they are an essential tool.*
- *Another thing that got my attention was the 'SalePrice' correlations. We can see our well-known 'GrLivArea', 'TotalBsmtSF', and 'OverallQual' saying they are strongly correlated', but we can also see many other variables that should be taken into account.*

H. Scatter plots between 'SalePrice' and correlated variables

```
#scatterplot
import seaborn as sns; sns.set(style="ticks", color_codes=True)
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars',
'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.pairplot(df_train[cols], size = 2.5)
# plt.title("Correlation Analysis Between Variables");
plt.show();
```



RESULTS:

- *One of the figures we may find interesting is the one between 'TotalBsmtSF' and 'GrLiveArea'. In this figure we can see the dots drawing a linear line, which almost acts like a border. It totally makes sense that the majority of the dots stay below that line. Basement areas can be equal to the above ground living area, but it is not expected a basement area bigger than the above ground living area (unless you're trying to buy a bunker).*

- *The plot concerning 'SalePrice' and 'YearBuilt' can also make us think. In the bottom of the 'dots cloud', we see what almost appears to be a shy exponential function (be creative). We can also see this same tendency in the upper limit of the 'dots cloud' (be even more creative). Also, notice how the set of dots regarding the last years tend to stay above this limit (I just wanted to say that prices are increasing faster now).*

REFERENCES

Dataset: http://jse.amstat.org/v19n3/decock.pdf(Dataset was compiled by Dean De Cock for use in data science education)

Data From: https://www.kaggle.com/

For Colors: https://jakevdp.github.io/PythonDataScienceHandbook/04.02-simple-scatter-plots.html
http://www.discoveryplayground.com/computer-programming-for-kids/rgb-colors/

Plot: https://pythonspot.com/matplotlib-bar-chart/

Numpy: https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.ndarray.shape.html