```python
import pandas as pd
import numpy as np
import random
from random import random
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
sns.set(style='darkgrid',font_scale=1.2)
plt.rcParams['font.family']='SimHei'
plt.rcParams['axes.unicode_minus']=False
```

```python
train = pd.read_csv('train.csv',index_col='Id')
train.head()
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	Pool
Id													
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	NaN
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	NaN
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	NaN
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	NaN

5 rows × 80 columns

```python
test = pd.read_csv('test.csv',index_col='Id')
test.head()
```

```css
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	Po
Id													
1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	Inside	...	120	0
1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	0
1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0
1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	0
1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	Inside	...	144	0

5 rows × 79 columns

```
train.shape
```

```
(1460, 80)
```

```
train.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	Bsmt
count	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	1460.0
mean	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	46.549
std	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	161.31
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	0.0000
25%	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	0.0000
50%	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	0.0000
75%	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	0.0000
max	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.0

8 rows × 37 columns

```
train.columns
```

```
Index(['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
       'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
       'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'PoolQC', 'Fence', 'MiscFeature', 'MiscVal',
       'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'SalePrice'],
      dtype='object')
```

Column Info

- MSSubClass: Identifies the type of dwelling involved in the sale.

```
# MSSubClass 用于分类
train['MSSubClass'] = train['MSSubClass'].astype(str)
test['MSSubClass'] = test['MSSubClass'].astype(str)
print(train['MSSubClass'].value_counts(),test['MSSubClass'].value_counts())
```

```
20     536
60     299
50     144
120     87
30      69
```

```
160     63
70      60
80      58
90      52
190     30
85      20
75      16
45      12
180     10
40       4
Name: MSSubClass, dtype: int64 20     543
60     276
50     143
120     95
30      70
70      68
160     65
80      60
90      57
190     31
85      28
180      7
75       7
45       6
40       2
150      1
Name: MSSubClass, dtype: int64
```

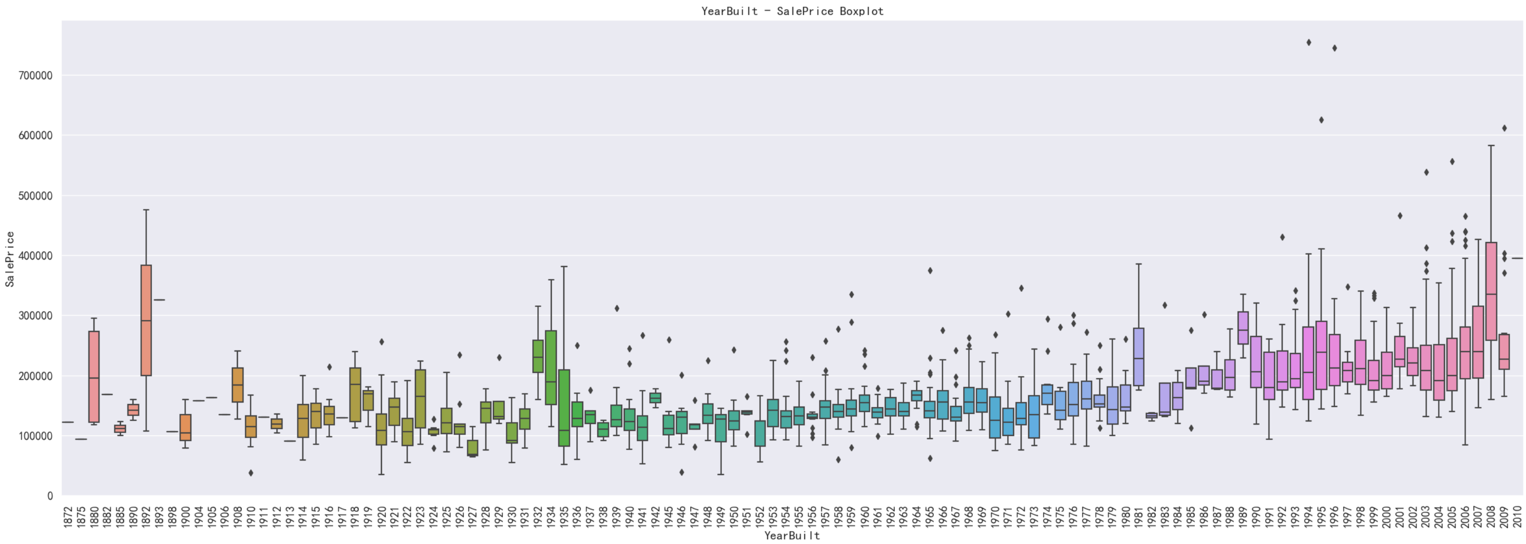
```
quant = [x for x in train.columns if train[x].dtypes != object]
quanli = [x for x in train.columns if train[x].dtypes == object]
print('quant: {}, counts: {}'.format(quant,len(quant)))
print('-----------------------------------------')
print('quanli: {}, counts: {}'.format(quanli,len(quanli)))
```

```
quant: ['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'], counts: 36
-----------------------------------------
quanli: ['MSSubClass', 'MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition'], counts: 44
```

```
total = train.isnull().sum().sort_values(ascending=False)
total
```

```
PoolQC           1453
MiscFeature      1406
Alley            1369
Fence            1179
FireplaceQu       690
                 ...
CentralAir          0
SaleCondition       0
Heating             0
TotalBsmtSF         0
MSSubClass          0
Length: 80, dtype: int64
```

```
plt.figure(figsize=(30,10),dpi=100)
sns.boxplot(train.YearBuilt,train.SalePrice)
plt.title('YearBuilt - SalePrice Boxplot')
plt.xticks(rotation=90)
plt.savefig('YearBuilt - SalePrice Boxplot.png',dpi=80)
plt.show()
```



Insights

- Outlier prices should be trimmed.
- Normal price: [around 100k, around 400k]
- TimeSerie Line plot needed to demonstrate the fluctuation in annual price.

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

```
count      1460.000000
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
```

```
train = train[train.SalePrice >= 40000]
```

```
train = train[train.SalePrice <= 500000]
```

```
train.shape
```

```
(1447, 80)
```

```
df1 = train.groupby('YearBuilt').agg({'SalePrice':'mean'})
df1
```

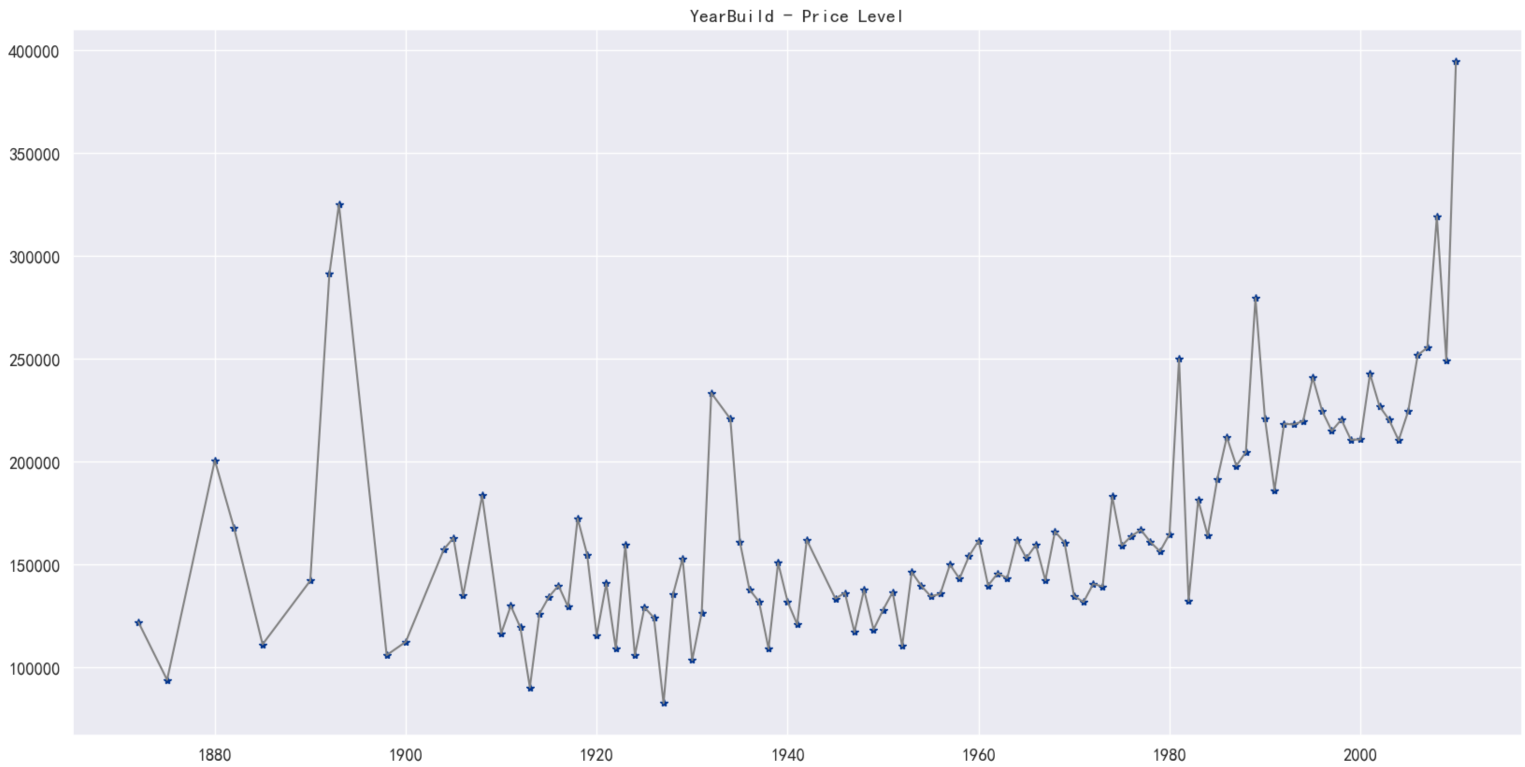
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	SalePrice
YearBuilt	
1872	122000.000000
1875	94000.000000
1880	200619.750000
1882	168000.000000
1885	111250.000000
...	...
2006	251775.447761
2007	255362.734694
2008	319188.000000
2009	249076.647059
2010	394432.000000

112 rows × 1 columns

```
plt.figure(figsize=(20,10),dpi=100)
# df1.plot(color='#00338D')
plt.plot(df1,"*",color="#00338D")
plt.plot(df1,color="gray")
plt.title("YearBuild - Price Level")
# plt.savefig('YearBuild - Price Level', dpi =100)
plt.show()
```



Insights

- General annual trend of *Salesprice*: Up

Conclusion: YearBuilt correlate with SalesPrice.

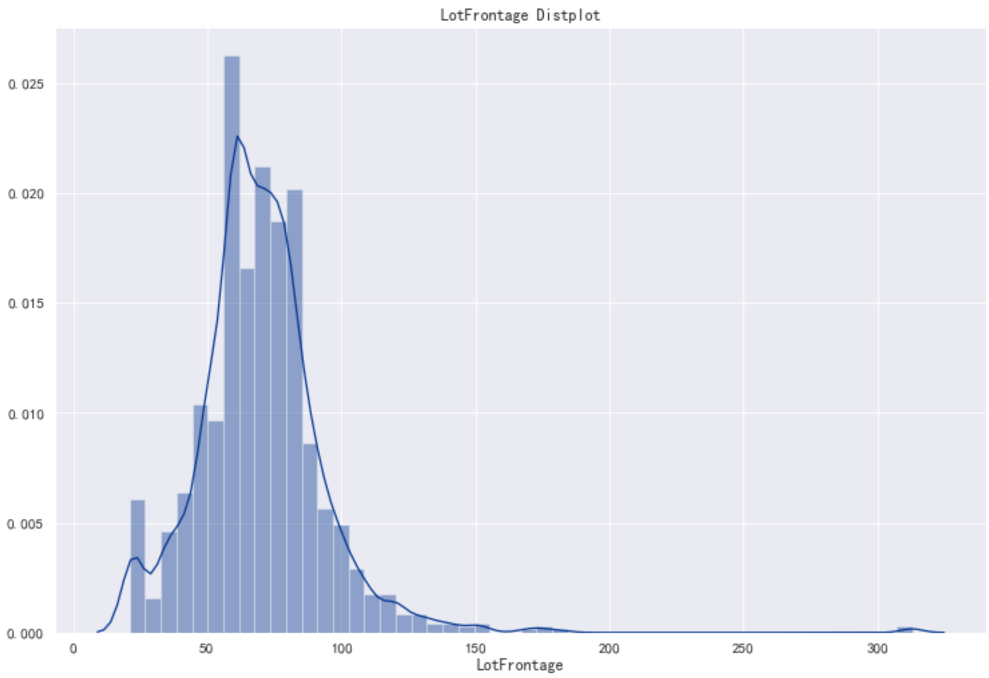
```
train.LotFrontage.describe()  #[21,313]
```

```
count    1188.000000
mean       69.891414
std        24.119385
min        21.000000
25%        59.000000
50%        69.000000
75%        80.000000
max       313.000000
Name: LotFrontage, dtype: float64
```

```
train.LotFrontage.isnull().sum()
```

```
259
```

```
plt.figure(figsize=(15,10),dpi=60)
sns.distplot(train.LotFrontage,color='#00338D')
plt.title('LotFrontage Distplot')
plt.savefig('LotFrontage Distplot.png', dpi =100)
plt.show()
```



Insights

- Lmt: [0,200]
- Skewed distribution: fillna with median value

```
train = train[train['LotFrontage']<=200]
```

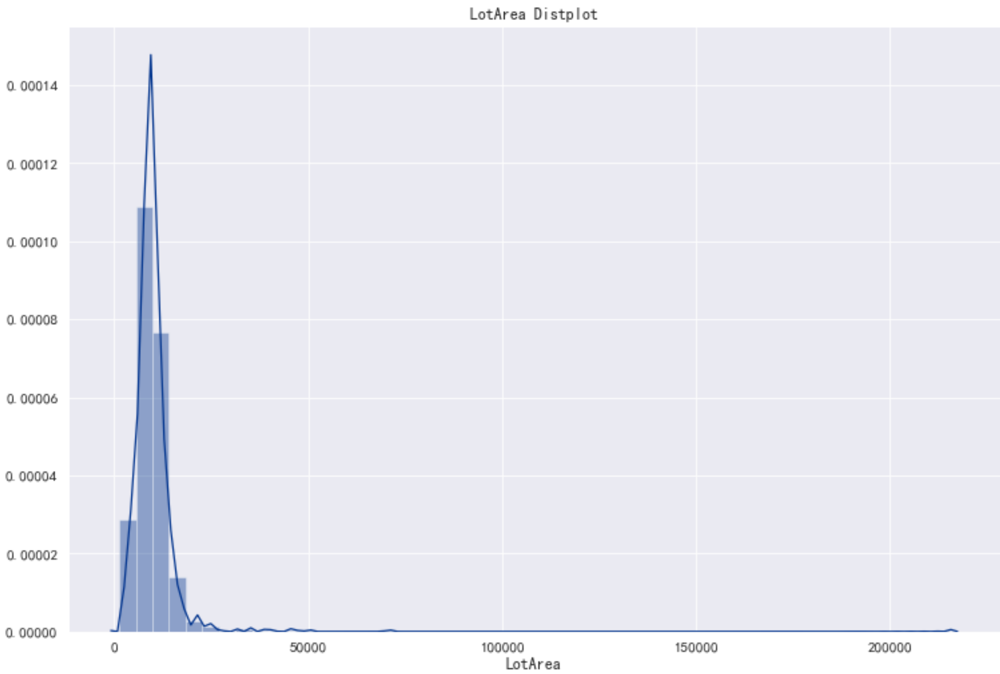
```
train.fillna({'LotFrontage':train.LotFrontage.median()},inplace=True)
train.LotFrontage.isnull().sum()
```

```
0
```

```
train.LotArea.describe() # [1.3k,215k+]
```

```
count      1186.000000
mean       9806.498314
std        7640.920262
min        1300.000000
25%        7409.000000
50%        9245.500000
75%       11205.250000
max      215245.000000
Name: LotArea, dtype: float64
```

```
plt.figure(figsize=(15,10),dpi=60)
sns.distplot(train.LotArea,color='#00338D')
plt.title('LotArea Distplot')
plt.show()
```




```
train = train[train['LotArea'] <= 50000]
```

```
train['LotArea'].isnull().sum()
```

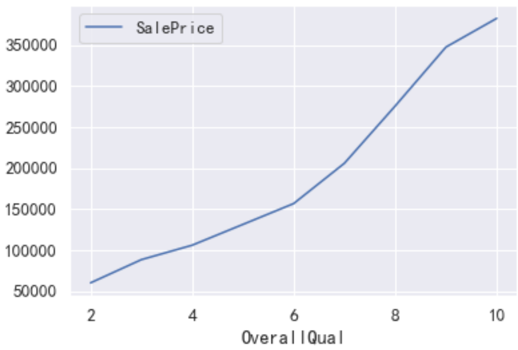
```
0
```

```
train.OverallQual.isnull().sum()
```

```
0
```

```
df2 = train.groupby('OverallQual').agg({'SalePrice':'mean'})
df2.plot()
```

```
<AxesSubplot:xlabel='OverallQual'>
```



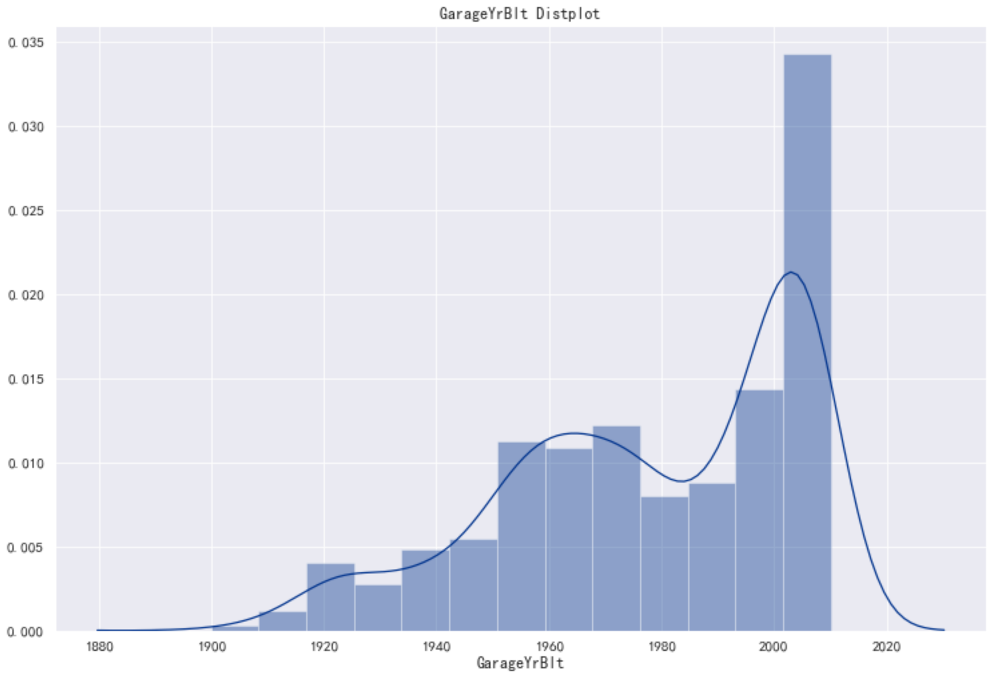
Insights

- SalePrice positively relates to overall rates of material & finish of house (AKA *OverallQual*)

```
null_var = [i for i in quant if train[i].isnull().sum() >0]
null_var
```

```
['MasVnrArea', 'GarageYrBlt']
```

```
plt.figure(figsize=(15,10),dpi=60)
sns.distplot(train.GarageYrBlt,color='#00338D')
plt.title('GarageYrBlt Distplot')
plt.show()
```




```
train.fillna({'MasVnrArea':train.MasVnrArea.median(),'GarageYrBlt':train.GarageYrBlt.median()},inplace=True)
```

```
null_var = [i for i in quant if train[i].isnull().sum() >0]
null_var
```

```
[]
```

Fillna finished.

```
train.Utilities.unique()
```

```
array(['AllPub'], dtype=object)
```

```
df3= train.groupby('Utilities').agg({'SalePrice':'mean'})
df3
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	SalePrice
Utilities	
AllPub	177552.770921

```
test.Utilities.unique()
```

```
array(['AllPub', nan], dtype=object)
```

- train 和 test Allpub 的 unique值不同，train中都是AllPub，所以对预测没有帮助，需要drop。

```
test.drop('Utilities',axis=1,inplace=True)
train.drop('Utilities',axis=1,inplace=True)
```

```
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
def leo(x):
    train[x]=LabelEncoder().fit_transform(train[x])
```

```
for i in quanli:
    leo(i)
```

```
# 将分类数据转化成整数编码
# 获取分类变量的标签值
def Labs(x):
    print(x,"--",train[x].unique())
df_obj = train[quanli]
print(list(map(Labs,df_obj)))
```

```
MSSubClass -- [ 9  4 10  8  3  7 14  0  5 12  1 11  2  6 13]
MSZoning -- [3 4 0 1 2]
Street -- [1 0]
Alley -- [2 0 1]
LotShape -- [3 0 1 2]
LandContour -- [3 0 1 2]
LotConfig -- [4 2 0 1 3]
LandSlope -- [0 1 2]
Neighborhood -- [ 5 24  6 15 11 21 17  3 19 16 20 12  9 10  7 23 22  4  8 14 13  0  2 18
  1]
Condition1 -- [2 1 0 5 8 6 4 3 7]
Condition2 -- [2 0 5 1 4 3]
BldgType -- [0 1 2 4 3]
HouseStyle -- [5 2 0 1 7 4 3 6]
RoofStyle -- [1 3 2 4 0]
RoofMatl -- [0 5 1 4 3 2]
Exterior1st -- [12  8 13  3  6 14  5  0 11  9  2  1 10  7  4]
Exterior2nd -- [13  8 15  6 14 10  5  3 12  0  2  7  1  9 11  4]
MasVnrType -- [1 2 3 0 4]
ExterQual -- [2 3 0 1]
ExterCond -- [4 1 2 3 0]
Foundation -- [2 1 0 5 3 4]
BsmtQual -- [2 3 0 4 1]
BsmtCond -- [3 1 4 0 2]
BsmtExposure -- [3 1 2 0 4]
```

```
BsmtFinType1 -- [2 0 5 4 6 3 1]
BsmtFinType2 -- [5 6 4 1 2 3 0]
Heating -- [0 1 2 4 3]
HeatingQC -- [0 2 4 1 3]
CentralAir -- [1 0]
Electrical -- [4 1 0 2 3 5]
KitchenQual -- [2 3 0 1]
Functional -- [5 2 0 3 4 1]
FireplaceQu -- [5 4 2 0 1 3]
GarageType -- [1 5 3 4 6 2 0]
GarageFinish -- [1 2 0 3]
GarageQual -- [4 1 2 5 0 3]
GarageCond -- [4 1 5 2 3 0]
PavedDrive -- [2 0 1]
PoolQC -- [3 0 1 2]
Fence -- [4 2 0 1 3]
MiscFeature -- [3 1 0 2]
SaleType -- [8 6 0 3 4 1 5 2 7]
SaleCondition -- [4 0 5 1 2 3]
[None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
```

```
corr = train.corr()
corr
```

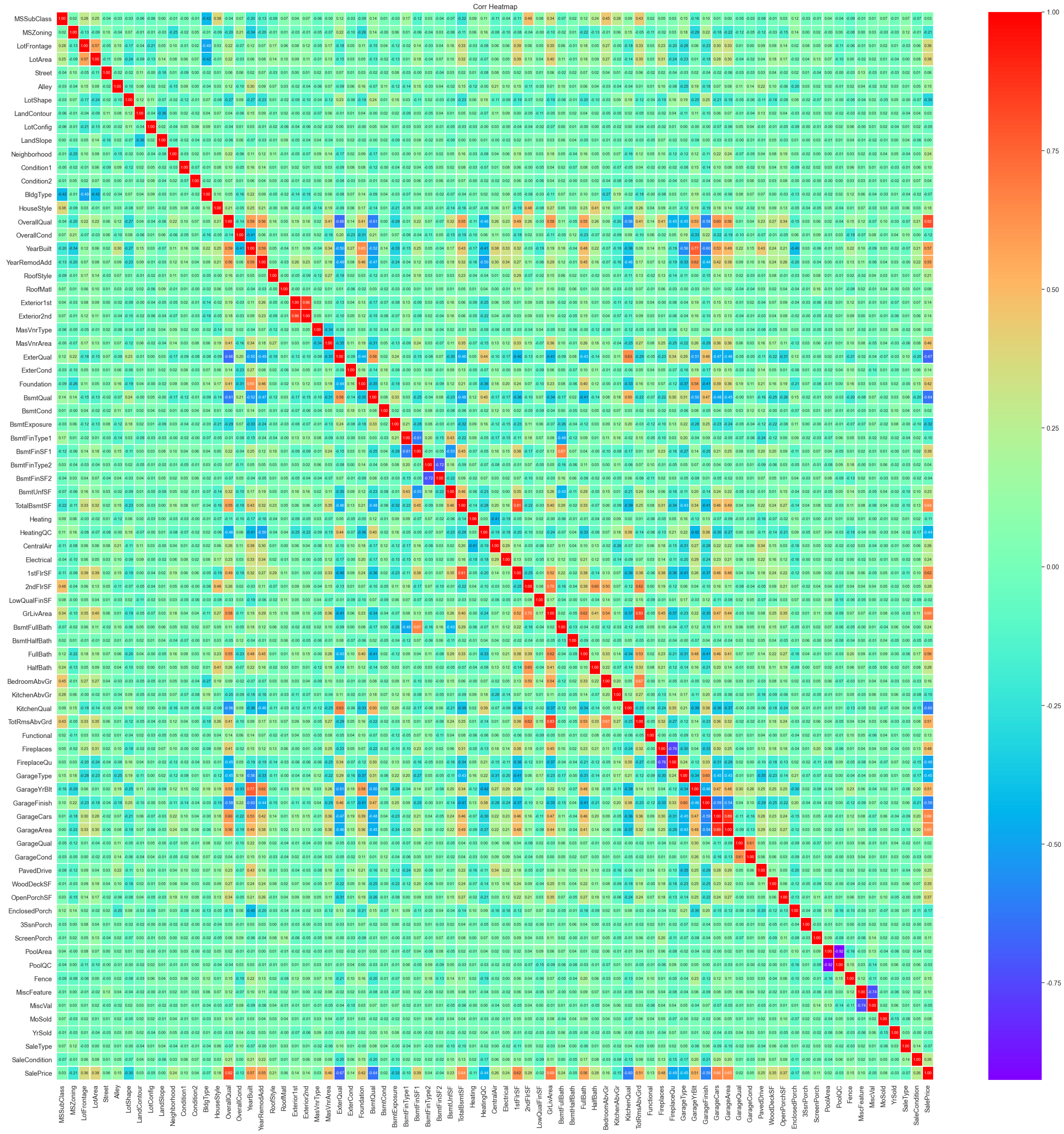
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	LotConfig	LandSlope	...
MSSubClass	1.000000	0.020940	0.279429	0.247561	-0.037199	-0.028659	-0.030395	-0.061403	-0.059057	0.002714	...
MSZoning	0.020940	1.000000	-0.129692	-0.085526	0.103255	-0.035657	0.068120	-0.013183	-0.012228	-0.029553	...
LotFrontage	0.279429	-0.129692	1.000000	0.570965	-0.047611	0.146923	-0.170013	-0.035607	-0.212730	0.049202	...
LotArea	0.247561	-0.085526	0.570965	1.000000	-0.105095	0.087676	-0.235314	-0.091517	-0.131644	0.139614	...
Street	-0.037199	0.103255	-0.047611	-0.105095	1.000000	-0.017300	-0.021099	0.105400	-0.001474	-0.164011	...
...	...	...	...	...	...	...	...	...	...	...	...
MoSold	0.067412	-0.030336	0.022697	0.005955	0.013222	0.021470	-0.049747	0.020259	0.016360	-0.008459	...
YrSold	-0.013556	-0.032690	0.005311	-0.036171	-0.029317	0.028094	0.050581	0.023113	-0.035390	0.000823	...
SaleType	0.070457	0.120846	-0.032287	0.003958	0.020851	0.010825	-0.000707	-0.036675	0.014615	0.041055	...
SaleCondition	-0.069217	-0.009377	0.061385	0.081866	0.011092	0.048470	-0.070626	0.041733	0.023852	-0.058860	...
SalePrice	0.030731	-0.208654	0.361636	0.383817	0.059211	0.151276	-0.301186	0.032256	-0.050879	0.002702	...

79 rows × 79 columns

```
plt.figure(figsize=(40,40),dpi=100)
sns.set(font_scale=1.2)
ax=sns.heatmap(corr,
               xticklabels=corr.columns,
               yticklabels=corr.columns,
               linewidths=0.9,annot=True,
               cbar=True,cmap="rainbow",fmt='.2f',
               annot_kws={'size': 8})
plt.title("Corr Heatmap")
plt.savefig("Corr Heatmap.png")
plt.show()
```



1. VS SalePrice

中相关

- LotFrontage; LotArea; LotShape; MAsVnrArea; Foundation; BsmtExposure; BsmtFinSF1; HeatingQC; FullBath; TotRmsAbvGrd; Fireplaces; FireplacesQu; GarageType; GarageFinish; WoodDeckSF; OpenPorchSF;

强相关

- **OverallQual; YearBuilt; YearRemodAdd; ExterQual; BsmtQual; TotalBsmtSF; 1stFlrSF; GrLivArea; KitchQual; GarageCars; GarageArea**
- *弱相关：MSZoning; Neighborhood; HouseStyle; RoofStyle; BsmtUnfSF; CentralAir; Electical; 2ndFlrSF; BsmtFullBath; HalfBath; PavedDrive; EnclosedPorch;SaleCondition*

Insights

- 强相关显然保留
- 中相关性和弱相关性，**Deep Dive**变量之间相关性，选择性保留。

```python
strong_var = ['OverallQual','YearBuilt','YearRemodAdd','ExterQual','BsmtQual','TotalBsmtSF','1stFlrSF','GrLivArea','KitchenQual',
             'GarageCars','GarageArea']
```

```python
other_var = list(train.columns)
```

```python
for i in strong_var:
    other_var.remove(i)
```

```python
print(len(strong_var))
```

```
11
```

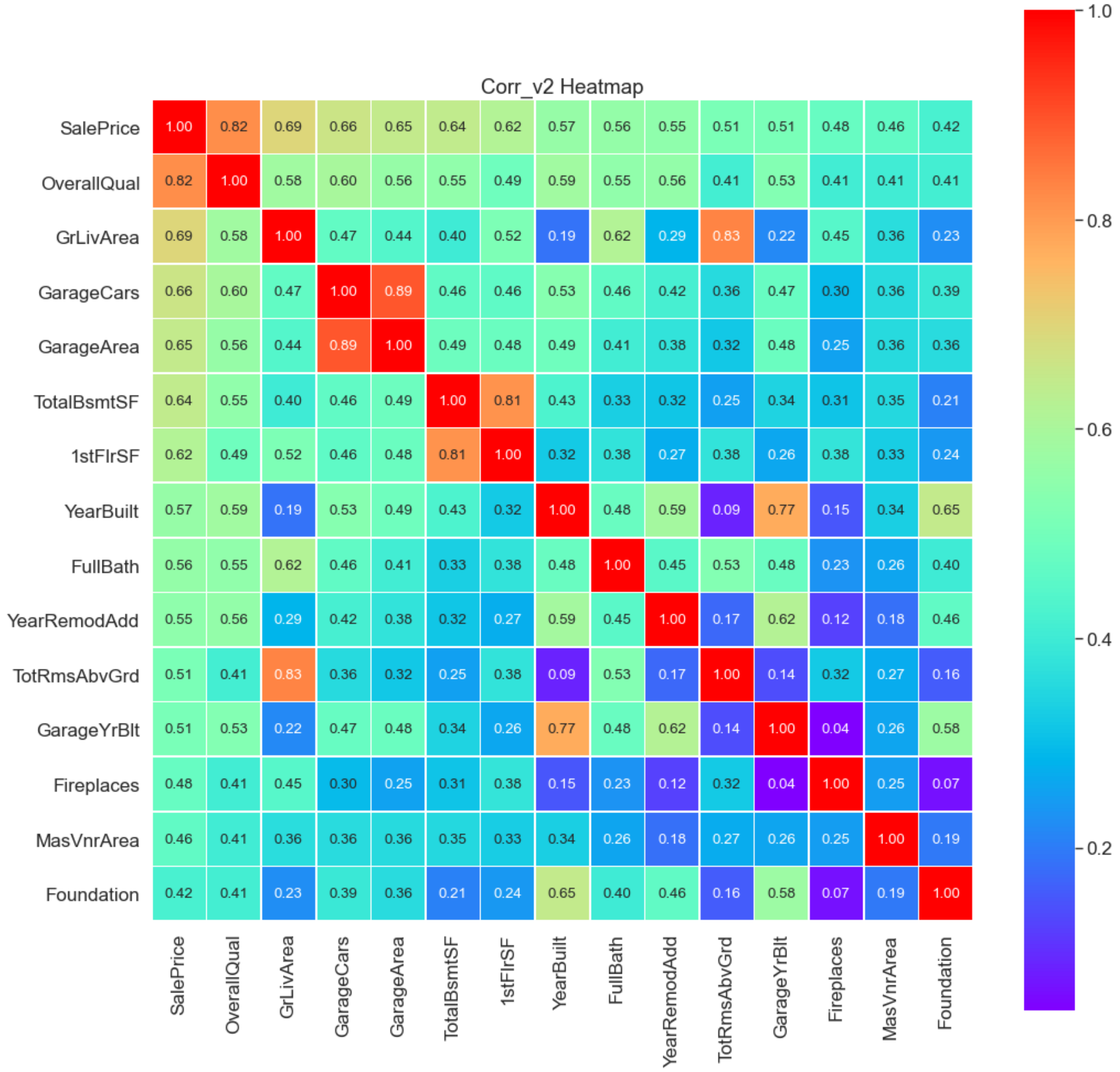
```python
k = 15
var_ = corr.nlargest(k, 'SalePrice')['SalePrice'].index
```

- strong_var
- corr.nlarest(k,'SalePrice')

```python
cm = np.corrcoef(train[var_].values.T)
cm
```

```
array([[1.        , 0.81836496, 0.69405958, 0.66058917, 0.6462927 ,
        0.64310743, 0.61768451, 0.56574773, 0.56303443, 0.55088068,
        0.51319332, 0.51063903, 0.47549164, 0.46349279, 0.4213752 ],
       [0.81836496, 1.        , 0.58450966, 0.59923389, 0.56356986,
        0.55246083, 0.489251  , 0.58554145, 0.55424945, 0.55683769,
        0.41286386, 0.5271104 , 0.40821558, 0.40520085, 0.41226335],
       [0.69405958, 0.58450966, 1.        , 0.4670706 , 0.4428922 ,
        0.40127177, 0.51535057, 0.188675  , 0.61736605, 0.28689558,
        0.83232925, 0.21961226, 0.44688989, 0.36195117, 0.22613848],
       [0.66058917, 0.59923389, 0.4670706 , 1.        , 0.89365001,
        0.4630363 , 0.45940744, 0.53472301, 0.46250507, 0.41582948,
        0.35733104, 0.47382228, 0.29591308, 0.3612021 , 0.39468072],
       [0.6462927 , 0.56356986, 0.4428922 , 0.89365001, 1.        ,
        0.48688889, 0.48195849, 0.48984406, 0.40908054, 0.37740308,
        0.31963221, 0.48175325, 0.2545989 , 0.35714997, 0.35877619],
       [0.64310743, 0.55246083, 0.40127177, 0.4630363 , 0.48688889,
        1.        , 0.81032781, 0.42749083, 0.33351637, 0.31553456,
        0.25034587, 0.34009332, 0.31201215, 0.35269621, 0.20686582],
       [0.61768451, 0.489251  , 0.51535057, 0.45940744, 0.48195849,
        0.81032781, 1.        , 0.32287916, 0.37844895, 0.27261388,
        0.37774798, 0.26200353, 0.38144482, 0.33066204, 0.23862028],
       [0.56574773, 0.58554145, 0.188675  , 0.53472301, 0.48984406,
        0.42749083, 0.32287916, 1.        , 0.48007967, 0.59203061,
        0.08783456, 0.7741855 , 0.15036689, 0.33522184, 0.64560153],
       [0.56303443, 0.55424945, 0.61736605, 0.46250507, 0.40908054,
        0.33351637, 0.37844895, 0.48007967, 1.        , 0.45160637,
        0.52924064, 0.48039193, 0.23192388, 0.25997238, 0.39983469],
       [0.55088068, 0.55683769, 0.28689558, 0.41582948, 0.37740308,
        0.31553456, 0.27261388, 0.59203061, 0.45160637, 1.        ,
        0.17465557, 0.61959647, 0.12399901, 0.1828303 , 0.46318838],
       [0.51319332, 0.41286386, 0.83232925, 0.35733104, 0.31963221,
        0.25034587, 0.37774798, 0.08783456, 0.52924064, 0.17465557,
        1.        , 0.13779946, 0.32415371, 0.26946452, 0.1599862 ],
       [0.51063903, 0.5271104 , 0.21961226, 0.47382228, 0.48175325,
        0.34009332, 0.26200353, 0.7741855 , 0.48039193, 0.61959647,
        0.13779946, 1.        , 0.04441026, 0.26031938, 0.57677663],
       [0.47549164, 0.40821558, 0.44688989, 0.29591308, 0.2545989 ,
        0.31201215, 0.38144482, 0.15036689, 0.23192388, 0.12399901,
        0.32415371, 0.04441026, 1.        , 0.24782868, 0.06650263],
       [0.46349279, 0.40520085, 0.36195117, 0.3612021 , 0.35714997,
        0.35269621, 0.33066204, 0.33522184, 0.25997238, 0.1828303 ,
        0.26946452, 0.26031938, 0.24782868, 1.        , 0.19462898],
       [0.4213752 , 0.41226335, 0.22613848, 0.39468072, 0.35877619,
        0.20686582, 0.23862028, 0.64560153, 0.39983469, 0.46318838,
        0.1599862 , 0.57677663, 0.06650263, 0.19462898, 1.        ]])
```

```python
plt.figure(figsize=(15,15),dpi=80)
sns.set(font_scale=1.4)
hm = sns.heatmap(cm, cbar=True, linewidths=0.5,
                 annot=True, square=True,
                 fmt='.2f',annot_kws={'size': 12}, cmap="rainbow",
                 yticklabels=var_.values, xticklabels=var_.values)
plt.title("Corr_v2 Heatmap")
plt.savefig("Corr_v2 Heatmap.png")
plt.show()
```



- 变量之间相关性强的需要drop

```python
vars_ =list(var_)
```

```python
vars_.remove('TotRmsAbvGrd')
```

```python
vars_.remove('GarageArea')
```

```python
vars_.remove('1stFlrSF')
```

```python
vars_.remove('GarageYrBlt')
```

```python
print('Finally var counts: {} {}var: {}'.format(len(vars_),'\n',vars_))
```

```
Finally var counts: 11
var: ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'YearBuilt', 'FullBath', 'YearRemodAdd', 'Fireplaces', 'MasVnrArea', 'Foundation']
```

```
vars_.remove('SalePrice')
x_train_fnl = train[vars_]
y_train_fnl = train['SalePrice']
```

```
x_test_fnl = test[vars_]
```

```
def leo2(x):
    x_test_fnl[x]=LabelEncoder().fit_transform(x_test_fnl[x])
```

```
for i in vars_:
    leo2(i)
```

```
x_test_fnl.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	OverallQual	GrLivArea	GarageCars	TotalBsmtSF	YearBuilt	FullBath	YearRemodAdd	Fireplaces	MasVnrArea	Foundation
Id										
1461	4	64	1	202	56	1	11	0	0	1
1462	5	313	1	477	53	1	8	0	58	1
1463	4	518	2	234	92	2	48	1	0	2
1464	5	500	2	233	93	2	48	1	6	2
1465	7	282	2	455	87	2	42	0	0	2

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(copy=False)
scaler.fit_transform(x_train_fnl)
train2 = scaler.transform(x_train_fnl)
train2
```

```
array([[ 0.65288347,  0.43354519,  0.31409347, ..., -0.90822251,
         0.55530153,  0.82762684],
       [-0.07438691, -0.47798807,  0.31409347, ...,  0.70745753,
        -0.56020872, -0.52097172],
       [ 0.65288347,  0.58818029,  0.31409347, ...,  0.70745753,
         0.36179465,  0.82762684],
       ...,
       [ 0.65288347,  1.71538883, -0.98511134, ...,  2.32313758,
        -0.56020872,  3.52482395],
       [-0.80165729, -0.8523678 , -0.98511134, ..., -0.90822251,
        -0.56020872, -0.52097172],
       [-0.80165729, -0.49019611, -0.98511134, ..., -0.90822251,
        -0.56020872, -0.52097172]])
```

```
from sklearn.linear_model import LinearRegression
```

```
scaler2 = StandardScaler(copy=False)
scaler2.fit_transform(x_test_fnl)
x_test = scaler2.transform(x_test_fnl)
lr =LinearRegression()
```

```python
from sklearn.model_selection import train_test_split #数据集训练集划分
# from sklearn import metrics
from sklearn.metrics import adjusted_rand_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,precision_score,recall_score,f1_score,r2_score #分类报告
x_train,x_test,y_train,y_test=train_test_split(x_train_fnl,y_train_fnl,test_size=.2,random_state=22)
lr.fit(x_train,y_train)
pred = lr.predict(x_test)
```

```python
y_test = pd.DataFrame(y_test)
pred = pd.DataFrame(pred)
print('r2_score:',r2_score(pred,y_test))
```

```
r2_score: 0.8504507414086491
```

```python
# print('f1_score:',precision_score(pred,y_test))
xx = np.arange(len(y_test))
```

```python
plt.figure(figsize = (10,10), dpi=80)
plt.scatter(xx, y_test, color="#00338D",s= 20, marker='*')
plt.scatter(xx,pred,color="red", s= 20,marker = 'o')
```

```
<matplotlib.collections.PathCollection at 0x1129b838850>
```


```python
lr.fit(train2,y_train_fnl.values)
y_train_pred = lr.predict(x_test)
y_train_pred
```

```
array([ 96036.72599752, 154142.89713714, 181295.75480668, ...,
       165202.97140824, 112613.91975802, 244607.5935728 ])
```

```python
ypred = pd.DataFrame(y_train_pred)
# ypred.to_csv('Bostion ypred v1.csv')
```