In [177]:

```
import pandas as pd
import numpy as np
import itertools
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.simplefilter('ignore')
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
```

In [201]:

```
train=pd.read_csv(r'C:\Users\15282\Downloads\HW2\house-prices-advanced-regression-techniques\train.c
```

In [202]:

```
test=pd.read_csv(r'C:\Users\15282\Downloads\HW2\house-prices-advanced-regression-techniques\test.csv
```

In [203]:

```
train.head()
```

Out[203]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
0	1	60	RL	65.00000	8450	Pave	NaN	Reg	Lvl	A
1	2	20	RL	80.00000	9600	Pave	NaN	Reg	Lvl	A
2	3	60	RL	68.00000	11250	Pave	NaN	IR1	Lvl	A
3	4	70	RL	60.00000	9550	Pave	NaN	IR1	Lvl	A
4	5	60	RL	84.00000	14260	Pave	NaN	IR1	Lvl	A

In [204]:

```
cols=[c for c in train.columns]
print(cols)
print(len(cols))
train[cols].dtypes.value_counts()#观察表头，获取数据类型和数量
```

```
['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley', 'LotSh
ape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Conditi
on1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBui
lt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrT
ype', 'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnf
SF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF',
'2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Function
al', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish', 'Gara
geCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'Open
PorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC', 'Fenc
e', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'Sale
Price']
81
```

Out[204]:

```
object     43
int64      35
float64     3
dtype: int64
```

In [205]:

```
train.corr()[u'SalePrice']#观察自变量与因变量的相关性
```

Out[205]:

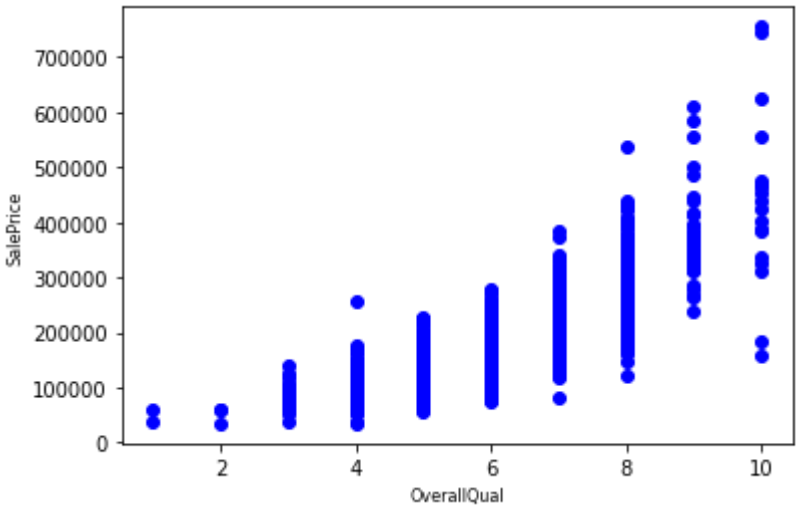
```
Id              -0.02192
MSSubClass      -0.08428
LotFrontage      0.35180
LotArea          0.26384
OverallQual      0.79098
OverallCond     -0.07786
YearBuilt        0.52290
YearRemodAdd     0.50710
MasVnrArea       0.47749
BsmtFinSF1       0.38642
BsmtFinSF2      -0.01138
BsmtUnfSF        0.21448
TotalBsmtSF      0.61358
1stFlrSF         0.60585
2ndFlrSF         0.31933
LowQualFinSF    -0.02561
GrLivArea        0.70862
BsmtFullBath     0.22712
BsmtHalfBath    -0.01684
FullBath         0.56066
HalfBath         0.28411
BedroomAbvGr     0.16821
KitchenAbvGr    -0.13591
TotRmsAbvGrd     0.53372
Fireplaces       0.46693
GarageYrBlt      0.48636
GarageCars       0.64041
GarageArea       0.62343
WoodDeckSF       0.32441
OpenPorchSF      0.31586
EnclosedPorch   -0.12858
3SsnPorch        0.04458
ScreenPorch      0.11145
PoolArea         0.09240
MiscVal         -0.02119
MoSold           0.04643
YrSold          -0.02892
SalePrice        1.00000
Name: SalePrice, dtype: float64
```

In [206]:

```
train=train.drop('Utilities',1)
test=test.drop('Utilities',1)#观察后舍去无用的变量
```

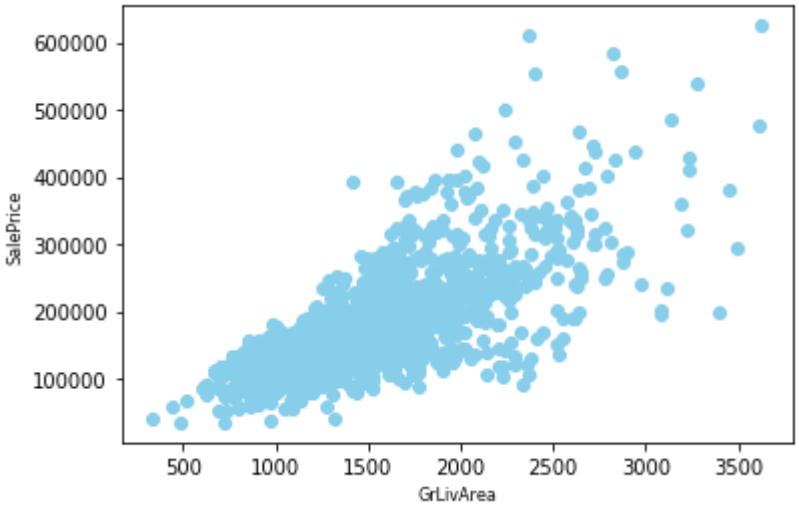
In [207]:

```
fig, ax = plt.subplots()
ax.scatter(x = train['OverallQual'], y = train['SalePrice'], c = "blue")
plt.ylabel('SalePrice', fontsize=8)
plt.xlabel('OverallQual', fontsize=8)
plt.show()
```



In [208]:

```
train.drop(train[(train['GrLivArea']>4000)&(train['GrLivArea']<30000)].index,inplace=True)
fig, ax = plt.subplots()
ax.scatter(x = train['GrLivArea'], y = train['SalePrice'], c = "skyblue")
plt.ylabel('SalePrice', fontsize=8)
plt.xlabel('GrLivArea', fontsize=8)
plt.show()
```

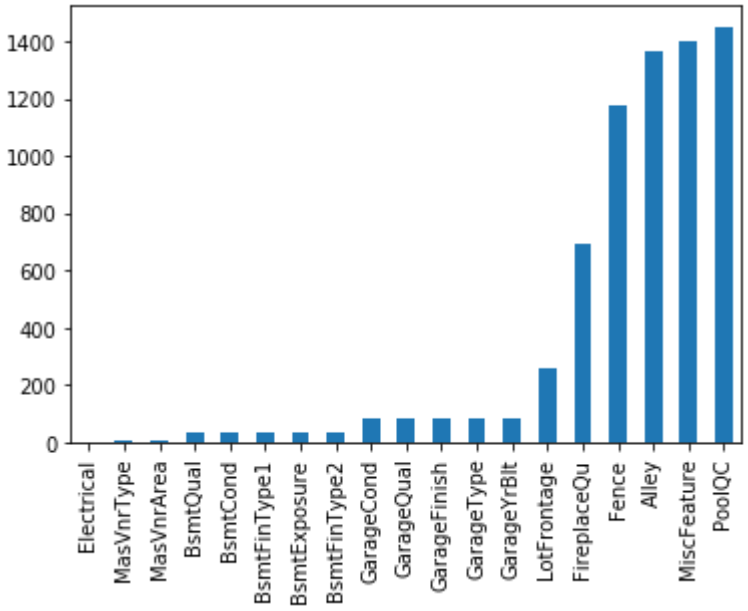


In [209]:

```
#查找train中的缺失值
missingtotal=train.isnull().sum()
missingexist=missingtotal[missingtotal>0]
missingexist.sort_values(inplace=True)
missingexist.plot.bar()
```

Out[209]:

<matplotlib.axes._subplots.AxesSubplot at 0x1e8b4c93948>



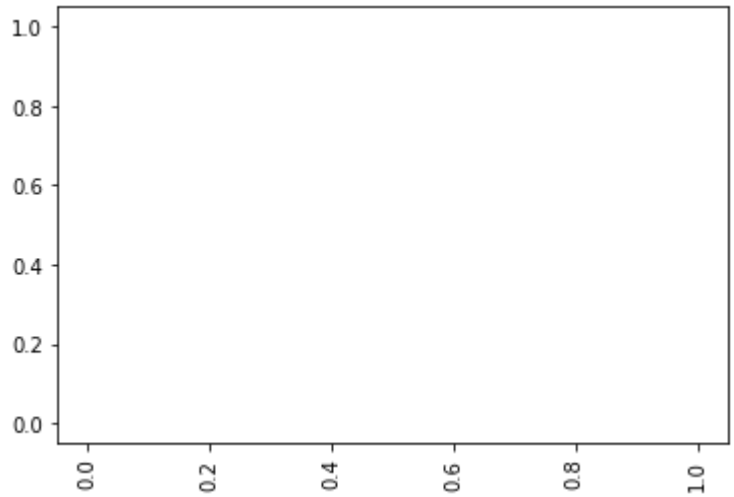
In [220]:

```
#对缺失值进行补充，通过观察train.csv中缺失值所在列的数值，判断用no或者众数进行补充
train['PoolQC'] = train['PoolQC'].fillna('NO')
test['PoolQC'] = test['PoolQC'].fillna('No')

train['MiscFeature'] = train['MiscFeature'].fillna('No')
test['MiscFeature'] = test['MiscFeature'].fillna('No')

train['Alley'] = train['Alley'].fillna('No')
test['Alley'] = test['Alley'].fillna('No')

train['Fence'] = train['Fence'].fillna('No')
test['Fence'] = test['Fence'].fillna('No')

train['FireplaceQu'] = train['FireplaceQu'].fillna('No')
test['FireplaceQu'] = test['FireplaceQu'].fillna('No')

train['LotFrontage'] = train.groupby('Neighborhood')['LotFrontage'].transform(lambda x : x.fillna(x
test['LotFrontage'] = test.groupby('Neighborhood')['LotFrontage'].transform(lambda x : x.fillna(x.m

for columns in ('GarageYrBlt','GarageType','GarageFinish', 'GarageQual','GarageCond'):
    train[columns] = train[columns].fillna('No')
    test[columns] = test[columns].fillna('No')

for columns in ('BsmtQual', 'BsmtCond', 'BsmtExposure','BsmtFinType1',  'BsmtFinType2'):
    train[columns] = train[columns].fillna('No')
    test[columns] = test[columns].fillna('No')

for columns in ('MasVnrType', 'MasVnrArea', 'Electrical'):
    train[columns] = train[columns].fillna(train[columns].mode()[0])
    test[columns] = test[columns].fillna(test[columns].mode()[0])
```

In [221]:

```
#最后预测时发现test中还有缺失值，再次进行补充
missingtotal=test.isnull().sum()
missingexist=missingtotal[missingtotal>0]
missingexist.sort_values(inplace=True)
missingexist.plot.bar()
#补充后再次检查了一次，所以图表空了
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-221-a8c7dd25ee76> in <module>
      3 missingexist=missingtotal[missingtotal>0]
      4 missingexist.sort_values(inplace=True)
----> 5 missingexist.plot.bar()
      6 #补充后再次检查所以图表空了

D:\anaconda\lib\site-packages\pandas\plotting\_core.py in bar(self, x, y,
 **kwargs)
   1001             >>> ax = df.plot.bar(x='lifespan', rot=0)
   1002         """
-> 1003         return self(kind="bar", x=x, y=y, **kwargs)
   1004
   1005     def barh(self, x=None, y=None, **kwargs):

D:\anaconda\lib\site-packages\pandas\plotting\_core.py in __call__(self, *a
rgs, **kwargs)
    845                     data.columns = label_name
    846
--> 847         return plot_backend.plot(data, kind=kind, **kwargs)
    848
    849     __call__.__doc__ = __doc__

D:\anaconda\lib\site-packages\pandas\plotting\_matplotlib\__init__.py in p
lot(data, kind, **kwargs)
     59             kwargs["ax"] = getattr(ax, "left_ax", ax)
     60     plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 61     plot_obj.generate()
     62     plot_obj.draw()
     63     return plot_obj.result

D:\anaconda\lib\site-packages\pandas\plotting\_matplotlib\core.py in genera
te(self)
    268         for ax in self.axes:
    269             self._post_plot_logic_common(ax, self.data)
--> 270             self._post_plot_logic(ax, self.data)
    271
    272     def _args_adjust(self):

D:\anaconda\lib\site-packages\pandas\plotting\_matplotlib\core.py in _post_
plot_logic(self, ax, data)
   1414         name = self._get_index_name()
   1415
-> 1416         s_edge = self.ax_pos[0] - 0.25 + self.lim_offset
   1417         e_edge = self.ax_pos[-1] + 0.25 + self.bar_width + self.lim_offset
   1418

IndexError: index 0 is out of bounds for axis 0 with size 0
```



In [222]:

```
test['BsmtFinSF1'] = test['BsmtFinSF1'].fillna(0)
test['BsmtFinSF2'] = test['BsmtFinSF2'].fillna(0)
test['BsmtUnfSF'] = test['BsmtUnfSF'].fillna(0)
test['TotalBsmtSF'] = test['TotalBsmtSF'].fillna(0)
test['GarageCars'] = test['GarageCars'].fillna(0)
test['GarageArea'] = test['GarageArea'].fillna(0)
test['BsmtFullBath'] = test['BsmtFullBath'].fillna(0)
test['BsmtHalfBath'] = test['BsmtHalfBath'].fillna(0)
```

In [223]:

```
#把数值表现的分类变量转换为类别变量
train['MSSubClass'] =train['MSSubClass'].astype(str)
test['MSSubClass'] =test['MSSubClass'].astype(str)

train['OverallQual'] =train['OverallQual'].astype(str)
test['OverallQual'] =test['OverallQual'].astype(str)

train['OverallCond'] =train['OverallCond'].astype(str)
test['OverallCond'] =test['OverallCond'].astype(str)

train['MoSold'] =train['MoSold'].astype(str)
test['MoSold'] =test['MoSold'].astype(str)

train['YrSold'] =train['YrSold'].astype(str)
test['YrSold'] =test['YrSold'].astype(str)
```

In [224]:

```
#把类别用数值代替
from sklearn.preprocessing import LabelEncoder
cols=( 'MSSubClass', 'MSZoning',  'Street', 'Alley',
      'LotShape', 'LandContour',  'LotConfig', 'LandSlope',
      'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
      'HouseStyle', 'OverallQual', 'OverallCond', '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',  'MoSold', 'YrSold', 'SaleType', 'SaleCondition','GarageYrBlt')

for c in cols:
    lbl=LabelEncoder()
    lbl.fit(list(train[c].values))
    train[c]=lbl.transform(list(train[c].values))

for x in cols:
    lbl=LabelEncoder()
    lbl.fit(list(test[x].values))
    test[x]=lbl.transform(list(test[x].values))
```

In [225]:

```
#把自变量与因变量分开，并随机生成训练集和测试集
X = train.iloc[:,np.r_[0:79]]
y = train['SalePrice']


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=None)
print(X_test)#用于检查是否随机
```

```
1155         280          1437         1          4           1           4
960          162           858         1          2           1           4

      1stFlrSF  2ndFlrSF  LowQualFinSF  GrLivArea  BsmtFullBath  BsmtHalfBath  \
1423      1575       626             0       2201             0             0
1232      1224         0             0       1224             0             0
448        780       596             0       1376             0             0
676       1095       679             0       1774             1             0
454       1728         0             0       1728             2             0
...        ...       ...           ...        ...           ...           ...
641       1057       872             0       1929             1             0
375        904         0             0        904             1             0
875       1184      1426             0       2610             0             0
1155      1437         0             0       1437             1             0
960        858         0             0        858             1             0

      FullBath  HalfBath  BedroomAbvGr  KitchenAbvGr  KitchenQual  \
1423         2         0             4             1            2
1232         2         0             2             2            3
```

In [226]:

```
from sklearn import linear_model
from sklearn.model_selection import cross_val_score
```

In [229]:

```
#比较了ridge回归和lasso回归，感觉数值上差不多，最终选择了ridge
clf=linear_model.Ridge(alpha=5).fit(X_train,y_train)
clf.fit(X_train,y_train)

r2_score=clf.score(X_test, y_test )
cv_score=cross_val_score(clf,X_train,y_train,cv=5)
print(np.mean(cv_score))
print(r2_score)
```

```
0.8685974621960465
0.8935462436457087
```

In [228]:

```
clf=linear_model.Lasso(alpha=1)
clf.fit(X_train,y_train)

r2_score=clf.score(X_test, y_test )
cv_score=cross_val_score(clf,X_train,y_train,cv=5)
print(np.mean(cv_score))
print(r2_score)
```

```
0.868553953936574
0.8933253791886909
```

In [135]:

```
pd.set_option('max_columns',1000)
pd.set_option('max_row',300)
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

In [230]:

```
#对test.csv中的数据进行预测
y_id=test['Id']
y_final =clf.predict(test)
df1=pd.DataFrame(y_id)
df2=pd.DataFrame(y_final)
data=pd.concat([df1,df2],axis=1)
data.columns=['Id','SalePrice']

#data= {"Id" : [y_id], "SalePrice": [y_final]}
print(data)
```

```
        Id    SalePrice
0     1461 110561.64329
1     1462 133877.27431
2     1463 175071.90211
3     1464 177363.07799
4     1465 165383.66344
...    ...          ...
1454  2915  75727.40178
1455  2916  55204.05389
1456  2917 148452.80499
1457  2918 122495.29202
1458  2919 211215.67944

[1459 rows x 2 columns]
```

In [231]:

```
#保存导出
data.to_csv('result.csv',index=False,sep=',')
```