1. Dataset

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [2]:

```
df = pd.read_csv("AmesHousing.tsv",sep = '\t')
```

In [3]:

```
df
```

Out[3]:

	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
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	NaN	0
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	Gar2	12500
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	...	0	NaN	MnPrv	NaN	0
5	6	527105030	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
6	7	527127150	120	RL	41.0	4920	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
7	8	527145080	120	RL	43.0	5005	Pave	NaN	IR1	HLS	...	0	NaN	NaN	NaN	0
8	9	527146030	120	RL	39.0	5389	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
9	10	527162130	60	RL	60.0	7500	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
10	11	527163010	60	RL	75.0	10000	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
11	12	527165230	20	RL	NaN	7980	Pave	NaN	IR1	Lvl	...	0	NaN	GdPrv	Shed	500
12	13	527166040	60	RL	63.0	8402	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
13	14	527180040	20	RL	85.0	10176	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
14	15	527182190	120	RL	NaN	6820	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
15	16	527216070	60	RL	47.0	53504	Pave	NaN	IR2	HLS	...	0	NaN	NaN	NaN	0
16	17	527225035	50	RL	152.0	12134	Pave	NaN	IR1	Bnk	...	0	NaN	NaN	NaN	0
17	18	527258010	20	RL	88.0	11394	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
18	19	527276150	20	RL	140.0	19138	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
19	20	527302110	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	NaN	0
20	21	527358140	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	...	0	NaN	MnPrv	NaN	0
21	22	527358200	85	RL	85.0	10625	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	NaN	0
22	23	527368020	60	FV	NaN	7500	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
23	24	527402200	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	Shed	700
24	25	527402250	20	RL	NaN	12537	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
25	26	527403020	20	RL	65.0	8450	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
26	27	527404120	20	RL	70.0	8400	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	NaN	0
27	28	527425090	20	RL	70.0	10500	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
28	29	527427230	120	RH	26.0	5858	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
29	30	527451180	160	RM	21.0	1680	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	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
2900	2901	916477010	20	RL	95.0	13618	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2901	2902	921205030	20	RL	88.0	11443	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2902	2903	921205050	20	RL	88.0	11577	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2903	2904	923125030	20	A (agr)	125.0	31250	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2904	2905	923202025	90	RM	78.0	7020	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2905	2906	923203090	120	RM	32.0	4500	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2906	2907	923203100	120	RM	32.0	4500	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2907	2908	923205120	20	RL	90.0	17217	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2908	2909	923225190	160	RM	41.0	2665	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2909	2910	923225240	160	RM	41.0	2665	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2910	2911	923225260	160	RM	42.0	3964	Pave	NaN	Reg	Lvl	...	0	NaN	GdPrv	NaN	0
2911	2912	923225510	20	RL	58.0	10172	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
2912	2913	923226150	90	RL	NaN	11836	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
2913	2914	923226180	180	RM	21.0	1470	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2914	2915	923226290	160	RM	21.0	1484	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2915	2916	923227100	20	RL	80.0	13384	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2916	2917	923228130	180	RM	21.0	1533	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2917	2918	923228180	160	RM	21.0	1533	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2918	2919	923228210	160	RM	21.0	1526	Pave	NaN	Reg	Lvl	...	0	NaN	GdPrv	NaN	0
2919	2920	923228260	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2920	2921	923228310	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2921	2922	923229110	90	RL	55.0	12640	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0
2922	2923	923230040	90	RL	63.0	9297	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2923	2924	923250060	20	RL	80.0	17400	Pave	NaN	Reg	Low	...	0	NaN	NaN	NaN	0
2924	2925	923251180	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2925	2926	923275080	80	RL	37.0	7937	Pave	NaN	IR1	Lvl	...	0	NaN	GdPrv	NaN	0
2926	2927	923276100	20	RL	NaN	8885	Pave	NaN	IR1	Low	...	0	NaN	MnPrv	NaN	0
2927	2928	923400125	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	Shed	700
2928	2929	924100070	20	RL	77.0	10010	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0
2929	2930	924151050	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0

2930 rows × 82 columns

2. First Model

2.1 one numerical features

In [4]:

```
def transform_features(df):
    return df
def select_features(df):
    return df[["Gr Liv Area", "SalePrice"]]
def train_and_test(df):
    train = df[:1460]
    test = df[1460:]
    numeric_train = train.select_dtypes(include = ['float','integer'])
    numeric_test = test.select_dtypes(include = ['float','integer'])
    features = numeric_train.columns.drop('SalePrice')

    # train the model
    lr = LinearRegression()
    lr.fit(train[features],train['SalePrice'])
    pre = lr.predict(test[features])
    mse = mean_squared_error(pre,test['SalePrice'])
    rmse = mse**0.5
    return rmse
```

In [5]:

```
df = pd.read_csv("AmesHousing.tsv", delimiter="\t")
transform_df = transform_features(df)
filtered_df = select_features(transform_df)
rmse = train_and_test(filtered_df)

rmse
```

Out[5]:

```
57088.25161263909
```

3. Cleaning data

3.1 drop columns with more than 10% missing values

In [6]:

```
df = pd.read_csv("AmesHousing.tsv", delimiter="\t")
cols_missing = df.isnull().sum()
drop_cols = cols_missing[(cols_missing > len(df)/10)]
df = df.drop(drop_cols.index, axis=1)
```

3.2 drop text columns with missing valuse

In [7]:

```
text_missing = df.select_dtypes(include=['object']).isnull().sum()
drop_cols = text_missing[(text_missing > 0)]
df = df.drop(drop_cols.index, axis=1)
```

3.3 drop useless columns

In [8]:

```
df = df.drop(["PID", "Order", "Mo Sold", "Sale Condition", "Sale Type"], axis=1)
```

3.4 new feature by learning domain knowledge

In [9]:

```
df['Year before Sale'] = df['Yr Sold'] - df['Year Built']
df['Year since remod'] = df['Yr Sold'] - df['Year Remod/Add']
df = df[df['Year before Sale'] >= 0]
df = df[df['Year since remod'] >= 0]
df
```

Out[9]:

	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	...	Open Porch SF	Enclosed Porch	3Ssn Porch	Screen Porch
0	20	RL	31770	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	...	62	0	0	0
1	20	RH	11622	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	...	0	0	0	120
2	20	RL	14267	Pave	IR1	Lvl	AllPub	Corner	Gtl	NAmes	...	36	0	0	0
3	20	RL	11160	Pave	Reg	Lvl	AllPub	Corner	Gtl	NAmes	...	0	0	0	0
4	60	RL	13830	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	...	34	0	0	0
5	60	RL	9978	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	...	36	0	0	0
6	120	RL	4920	Pave	Reg	Lvl	AllPub	Inside	Gtl	StoneBr	...	0	170	0	0

	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	...	Open Porch SF	Enclosed Porch	3Ssn Porch	Screen Porch
7	120	RL	5005	Pave	IR1	HLS	AllPub	Inside	Gtl	StoneBr	...	82	0	0	144
8	120	RL	5389	Pave	IR1	Lvl	AllPub	Inside	Gtl	StoneBr	...	152	0	0	0
9	60	RL	7500	Pave	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	...	60	0	0	0
10	60	RL	10000	Pave	IR1	Lvl	AllPub	Corner	Gtl	Gilbert	...	84	0	0	0
11	20	RL	7980	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	...	21	0	0	0
12	60	RL	8402	Pave	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	...	75	0	0	0
13	20	RL	10176	Pave	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	...	0	0	0	0
14	120	RL	6820	Pave	IR1	Lvl	AllPub	Corner	Gtl	StoneBr	...	54	0	0	140
15	60	RL	53504	Pave	IR2	HLS	AllPub	CulDSac	Mod	StoneBr	...	36	0	0	210
16	50	RL	12134	Pave	IR1	Bnk	AllPub	Inside	Mod	Gilbert	...	12	0	0	0
17	20	RL	11394	Pave	Reg	Lvl	AllPub	Corner	Gtl	StoneBr	...	0	0	0	0
18	20	RL	19138	Pave	Reg	Lvl	AllPub	Corner	Gtl	Gilbert	...	0	0	0	0
19	20	RL	13175	Pave	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	...	0	0	0	0
20	20	RL	11751	Pave	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	...	122	0	0	0
21	85	RL	10625	Pave	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	...	120	0	0	0
22	60	FV	7500	Pave	Reg	Lvl	AllPub	Inside	Gtl	Somerst	...	96	0	0	0
23	20	RL	11241	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	...	0	0	0	0
24	20	RL	12537	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	...	0	0	0	0
25	20	RL	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	...	85	184	0	0
26	20	RL	8400	Pave	Reg	Lvl	AllPub	Corner	Gtl	NAmes	...	0	0	0	0
27	20	RL	10500	Pave	Reg	Lvl	AllPub	FR2	Gtl	NAmes	...	0	0	0	0
28	120	RH	5858	Pave	IR1	Lvl	AllPub	FR2	Gtl	NAmes	...	68	0	0	0
29	160	RM	1680	Pave	Reg	Lvl	AllPub	Inside	Gtl	BrDale	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2900	20	RL	13618	Pave	Reg	Lvl	AllPub	Corner	Gtl	Timber	...	38	0	0	0
2901	20	RL	11443	Pave	Reg	Lvl	AllPub	Inside	Gtl	Timber	...	66	0	0	0
2902	20	RL	11577	Pave	Reg	Lvl	AllPub	Inside	Gtl	Timber	...	225	0	0	0
2903	20	A (agr)	31250	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	0	135	0	0
2904	90	RM	7020	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	48	0	0	0
2905	120	RM	4500	Pave	Reg	Lvl	AllPub	FR2	Gtl	Mitchel	...	125	0	0	0
2906	120	RM	4500	Pave	Reg	Lvl	AllPub	FR2	Gtl	Mitchel	...	199	0	0	0
2907	20	RL	17217	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	56	0	0	0
2908	160	RM	2665	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2909	160	RM	2665	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	26	0	0	0
2910	160	RM	3964	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	20	0	0	0
2911	20	RL	10172	Pave	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	...	120	0	0	0
2912	90	RL	11836	Pave	IR1	Lvl	AllPub	Corner	Gtl	Mitchel	...	0	0	0	0
2913	180	RM	1470	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2914	160	RM	1484	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2915	20	RL	13384	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	...	0	0	0	0
2916	180	RM	1533	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2917	160	RM	1533	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2918	160	RM	1526	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	34	0	0	0
2919	160	RM	1936	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	0	0	0	0
2920	160	RM	1894	Pave	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	...	24	0	0	0
2921	90	RL	12640	Pave	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	...	0	0	0	0
2922	90	RL	9297	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	0	0	0	0
2923	20	RL	17400	Pave	Reg	Low	AllPub	Inside	Mod	Mitchel	...	41	0	0	0
2924	20	RL	20000	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	0	0	0	0
2925	80	RL	7937	Pave	IR1	Lvl	AllPub	CulDSac	Gtl	Mitchel	...	0	0	0	0

	MS SubClass	MS Zoning	Lot Area	Street	Lot Shape	Land Contour	Utilities	Lot Config	Land Slope	Neighborhood	...	Open Porch SF	Enclosed Porch	3Ssn Porch	Screen Porch
2926	20	RL	8885	Pave	IR1	Low	AllPub	Inside	Mod	Mitchel	...	0	0	0	0
2927	85	RL	10441	Pave	Reg	Lvl	AllPub	Inside	Gtl	Mitchel	...	32	0	0	0
2928	20	RL	10010	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	...	38	0	0	0
2929	60	RL	9627	Pave	Reg	Lvl	AllPub	Inside	Mod	Mitchel	...	48	0	0	0

2927 rows × 62 columns

3.5 Three ways to deal with numerical columns (mean/median/mode)

In [10]:

```
num_missing = df.select_dtypes(include=['int','float']).isnull().sum()
fill_cols = num_missing[(num_missing > 0)]
fill_dictionary_mode = df[fill_cols.index].mode().to_dict('records')[0]
fill_dictionary_mean = df[fill_cols.index].mean().to_dict()
fill_dictionary_median = df[fill_cols.index].median().to_dict()
```

In [11]:

```
df_mode = df.fillna(fill_dictionary_mode)
df_mean = df.fillna(fill_dictionary_mean)
df_median = df.fillna(fill_dictionary_median)
```

In [12]:

```
df_mode.isnull().sum().value_counts()
```

Out[12]:

```
0    62
dtype: int64
```

In [13]:

```
df_mean.isnull().sum().value_counts()
```

Out[13]:

```
0    62
dtype: int64
```

In [14]:

```
df_median.isnull().sum().value_counts()
```

Out[14]:

```
0    62
dtype: int64
```

3.6 Outputs

In [15]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
def transform_features(df):
    cols_missing = df.isnull().sum()
    drop_cols = cols_missing[(cols_missing > len(df)/10)]
    df = df.drop(drop_cols.index, axis=1)

    text_missing = df.select_dtypes(include=['object']).isnull().sum()
    drop_cols = text_missing[(text_missing > 0)]
```

```
    drop_cols = text_missing[(text_missing > 0)]
    df = df.drop(drop_cols.index, axis=1)

    num_missing = df.select_dtypes(include=['int','float']).isnull().sum()
    fill_cols = num_missing[(num_missing < len(df)/10) & (num_missing > 0)]
    fill_dictionary_mode = df[fill_cols.index].mean().to_dict
    df = df.fillna(fill_dictionary_mode)

    df['Year before Sale'] = df['Yr Sold'] - df['Year Built']
    df['Year since remod'] = df['Yr Sold'] - df['Year Remod/Add']
    df = df[df['Year before Sale'] >= 0]
    df = df[df['Year since remod'] >= 0]
    df = df.drop(["Year Built", "Year Remod/Add", "Yr Sold"], axis = 1)
    df = df.drop(["PID", "Order", "Mo Sold", "Sale Condition", "Sale Type"], axis=1)


    return df

def select_features(df):
    return df[["Gr Liv Area", "SalePrice"]]

def train_and_test(df):
    train = df[:1460]
    test = df[1460:]
    numeric_train = train.select_dtypes(include = ['float','integer'])
    numeric_test = test.select_dtypes(include = ['float','integer'])
    features = numeric_train.columns.drop('SalePrice')

    # train the model
    lr = LinearRegression()
    lr.fit(train[features],train['SalePrice'])
    pre = lr.predict(test[features])
    mse = mean_squared_error(pre,test['SalePrice'])
    rmse = mse**0.5
    return rmse

df = pd.read_csv("AmesHousing.tsv", delimiter="\t")
transform_df = transform_features(df)
filtered_df = select_features(transform_df)
rmse = train_and_test(filtered_df)

rmse
```

Out[15]:

```
55275.367312413066
```

4 Feature selection

4.1 correlation

In [16]:

```
corr = abs(df.corr())
corr = corr['SalePrice']
corr = corr[corr > 0.5]
#corr.sort_values()
corr = corr.index
corr
```

Out[16]:

```
Index(['Overall Qual', 'Year Built', 'Year Remod/Add', 'Mas Vnr Area',
       'Total Bsmt SF', '1st Flr SF', 'Gr Liv Area', 'Full Bath',
       'Garage Yr Blt', 'Garage Cars', 'Garage Area', 'SalePrice'],
      dtype='object')
```

In [17]:

```
matrix = df[corr].corr()
```

In [18]:

```
matrix
```

Out[18]:

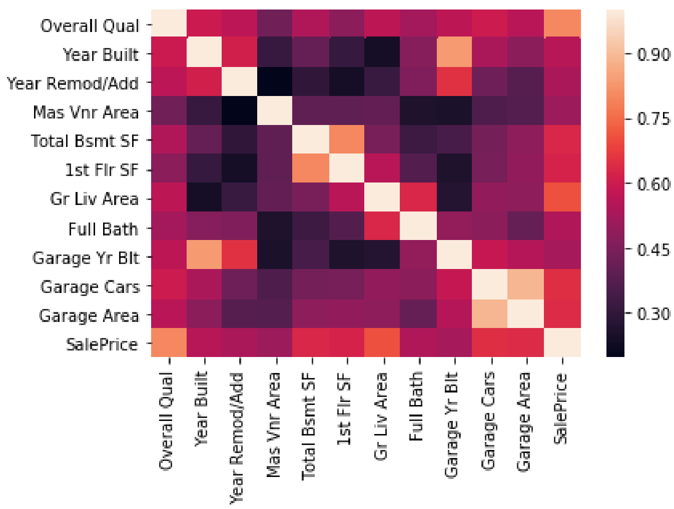
	Overall Qual	Year Built	Year Remod/Add	Mas Vnr Area	Total Bsmt SF	1st Flr SF	Gr Liv Area	Full Bath	Garage Yr Blt	Garage Cars	Garage Area	SalePric
Overall Qual	1.000000	0.597027	0.569609	0.429418	0.547294	0.477837	0.570556	0.522263	0.570569	0.599545	0.563503	0.79926
Year Built	0.597027	1.000000	0.612095	0.313292	0.407526	0.310463	0.241726	0.469406	0.834849	0.537443	0.480131	0.55842
Year Remod/Add	0.569609	0.612095	1.000000	0.196928	0.297481	0.242108	0.316855	0.457266	0.652310	0.425403	0.376438	0.53297
Mas Vnr Area	0.429418	0.313292	0.196928	1.000000	0.397040	0.395736	0.403611	0.260153	0.254784	0.360159	0.373458	0.50828
Total Bsmt SF	0.547294	0.407526	0.297481	0.397040	1.000000	0.800720	0.444675	0.324973	0.347571	0.437608	0.485504	0.63228
1st Flr SF	0.477837	0.310463	0.242108	0.395736	0.800720	1.000000	0.562166	0.371584	0.260170	0.439458	0.491223	0.62167
Gr Liv Area	0.570556	0.241726	0.316855	0.403611	0.444675	0.562166	1.000000	0.630321	0.272848	0.488829	0.484892	0.70678
Full Bath	0.522263	0.469406	0.457266	0.260153	0.324973	0.371584	0.630321	1.000000	0.494397	0.478182	0.407464	0.54560
Garage Yr Blt	0.570569	0.834849	0.652310	0.254784	0.347571	0.260170	0.272848	0.494397	1.000000	0.586731	0.555019	0.52696
Garage Cars	0.599545	0.537443	0.425403	0.360159	0.437608	0.439458	0.488829	0.478182	0.586731	1.000000	0.889676	0.64787
Garage Area	0.563503	0.480131	0.376438	0.373458	0.485504	0.491223	0.484892	0.407464	0.555019	0.889676	1.000000	0.64040
SalePrice	0.799262	0.558426	0.532974	0.508285	0.632280	0.621676	0.706780	0.545604	0.526965	0.647877	0.640401	1.00000

In [19]:

```
import seaborn as sns
%matplotlib inline
sns.heatmap(matrix)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x2c565deada0>



4.2 Nominal

In [20]:

```
df = pd.read_csv("AmesHousing.tsv", delimiter="\t")
nominal = ['PID','MS SubClass','MS Zoning','Street','Alley','Land Contour','Lot Config','Neighborhood',
          'Condition 1','Condition 2','Bldg Type','House Style','Roof Style','Roof Matl',
        'Mas Vnr Type','Foundation','Heating','Central Air','Garage Type','Misc Feature',
           'Sale Type','Sale Condition']
```

```
cols =[]
for col in nominal:
    haha = df[col].value_counts()
    if len(haha) < 10:
        cols.append(col)
if col in cols:
    if col in corr:
        print(col)
```

In [21]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
def transform_features(df):
    cols_missing = df.isnull().sum()
    drop_cols = cols_missing[(cols_missing > len(df)/10)]
    df = df.drop(drop_cols.index, axis=1)

    text_missing = df.select_dtypes(include=['object']).isnull().sum()
    drop_cols = text_missing[(text_missing > 0)]
    df = df.drop(drop_cols.index, axis=1)

    num_missing = df.select_dtypes(include=['int','float']).isnull().sum()
    fill_cols = num_missing[(num_missing < len(df)/10) & (num_missing > 0)]
    fill_dictionary_mode = df[fill_cols.index].mode().to_dict(orient='records')[0]
    df = df.fillna(fill_dictionary_mode)

    df['Year before Sale'] = df['Yr Sold'] - df['Year Built']
    df['Year since remod'] = df['Yr Sold'] - df['Year Remod/Add']
    df = df[df['Year before Sale'] >= 0]
    df = df[df['Year since remod'] >= 0]
    df = df.drop(["Year Built", "Year Remod/Add", "Yr Sold"], axis = 1)
    df = df.drop(["PID", "Order", "Mo Sold", "Sale Condition", "Sale Type"], axis=1)
    return df
def select_features(df,correlation_threshold=0.4,unique_threshold=10):
    cols =[]
    # find columns whose correlation with SalePrice more than correlation_threshold
    df_corr = df.corr()
    df_saleprice = df_corr['SalePrice']
    df_saleprice_th_cols = df_saleprice[df_saleprice < correlation_threshold].index
    df = df.drop(df_saleprice_th_cols, axis=1)

    # for better model, using uniqueness to delete some useless columns
    nominal = ['PID','MS SubClass','MS Zoning','Street','Alley','Land Contour','Lot Config','Neighb
orhood',
          'Condition 1','Condition 2','Bldg Type','House Style','Roof Style','Roof Matl',
        'Mas Vnr Type','Foundation','Heating','Central Air','Garage Type','Misc Feature',
           'Sale Type','Sale Condition']
    cols = []
    for col in nominal:
        if col in df.columns:
            if len(df[col].value_counts()) > 10:
                cols.append(col)
    df = df.drop(cols, axis=1)

    text_cols = df.select_dtypes(include=['object'])
    for col in text_cols:
        df[col] = df[col].astype('category')
    df = pd.concat([df, pd.get_dummies(df.select_dtypes(include=['category']))], axis=1)

    return df
```

4.3 cross validation

In [22]:

```
def train_and_test(df,k=0):
    numeric_df = df.select_dtypes(include=['integer', 'float'])
    features = numeric_df.columns.drop("SalePrice")
```

```
    if k == 0:
        train = df[:1460]
        test = df[1460:]
        numeric_train = train.select_dtypes(include = ['float','integer'])
        numeric_test = test.select_dtypes(include = ['float','integer'])


        # train the model
        lr = LinearRegression()
        lr.fit(train[features],train['SalePrice'])
        pre = lr.predict(test[features])
        mse = mean_squared_error(pre,test['SalePrice'])
        rmse = mse**0.5
        return rmse

    if k == 1:
        shuffled_df = df.sample(frac=1, )
        fold_one = df[:1460]
        fold_two = df[1460:]

        numeric_one = fold_one.select_dtype(include=['float','integer'])
        numeric_two = fold_two.select_dtype(include=['float','integer'])



        # train the model on fold one
        lg = LinearRegression()
        lg.fit(numeric_one[features],numeric_one['SalePrice'])
        pre = lg.predict(numeric_two[features])
        mse = mean_squared_error(pre,numeric_two['SalePrice'])
        rmse1 = mse**0.5

        # train the model on fold two
        lg = LinearRegression()
        lg.fit(numeric_two[features],numeric_two['SalePrice'])
        pre = lg.predict(numeric_one[features])
        mse = mean_squared_error(pre,numeric_one['SalePrice'])
        rmse2 = mse**0.5

        rmse = (rmse1+rmse2)/2
        return rmse

    else:
        kf = KFold(n_splits=k, shuffle=True)
        rmse = []
        for train_idx, test_idx in kf.split(df):
            train = df.iloc[train_idx]
            test = df.iloc[test_idx]
            lr = LinearRegression()
            lr.fit(train[features],train['SalePrice'])
            pre = lr.predict(test[features])
            mse = mean_squared_error(pre,test['SalePrice'])
            rmse0 = mse**0.5
            rmse.append(rmse0)
            print(rmse0)
        avg_rmse = sum(rmse)/len(rmse)
        return avg_rmse
```

In [23]:

```
df = pd.read_csv("AmesHousing.tsv", delimiter="\t")
transform_df = transform_features(df)
filtered_df = select_features(transform_df)
rmse = train_and_test(filtered_df, k=4)

rmse
```

```
36040.36253591515
25731.338218060046
25284.384304479623
28937.0482923508
```

Out[23]:

```
28998.283337701403
```