Data preparation, exploration, visualization

For this assignment, I wanted to build on top of the EDA I did for the housing data set in assignment 2. In line 67, I created a correlation matrix to once again see which variables highly correlate to another. In lines 71, 22 and 23, I converted qualitative data such as “has central air” to a binary standard and other categorical values such as “sale type” to a numeric scale. I then in line 24 created another correlation to see if the numeric conversions changed the correlation between variables. Overall, I decided to use every variable in the training set for my modeling methods and found that using all the variables would most likely create a higher accuracy.

Review research design and modeling methods

The 5 types of models I wanted to use are the linear regression, gradient boost, elastic net, ridge regression, lasso, linear regression and decision tree. In assignment 2, I explored the linear regression and lasso regression but I wanted to utilize these new ML technique to see if I could gain a higher accuracy for this predictive model. For all 5 of these models, I used a mean squared error as well as a salesprice for outcome variable. In line 35, I split the data into a training set and a test set and from lines 36-53 I conducted all five models using all the variables, I used a residual plot to evaluate the accuracy of the test set and training set score. I also used all the variables in my modeling methods, meaning that I had used 27 features for each model. I then tested the variance for each model using the mean squared error and then in line 62, I organized the techniques by accuracy.

Review results, evaluate models

I found that the most accurate result was the gradient boost model, followed by the elastic net, ridge, lasso, linear and then the decision tree. I was initially expecting that the linear regression would provide the most accurate results, however I was surprised when I saw the gradient boost was the most accurate of them all. In line 77, I exported my results to a file and submitted it onto Kaggle.

Implementation and programming as evidenced by Kaggle submission scores

My Kaggle username is Ankita Avadhani, with a score of 0.11843 and a ranking so far of 220 (getting better!)

Exposition, problem description, and management recommendations

If I were to address the management issue of what model to use to predict the sale price home based on certain features, I would recommend using a Gradient Boost model utilizing 27 of the features of the current data given. However, the most important explanatory variables still come from the quantitative aspects of the house meaning, LotSize, BasementSize and overall quality. However, I wanted to be able to include all the variables in this modeling assignment by converting the qualitative numbers to the quantitative values that we can utilize (for example, a binary classification of whether or not the house has an AC or a Pool). Once I was able to convert these explanatory variables into usable variables for my models, I believe my accuracy increased. From assignment 2, where I just analyzed the Lasso regression as well as linear regression, I only utilized 5 explanatory variables in my models. This caused my accuracy to be 70% for the linear regression and 40% for my lasso regression. I wanted to change the way I approached modeling again however, because I wanted to see how accurate I could make my models utilizing the given variables and data.

	Model	RMSE	Score
5	GradientBoost	27040.561603	87.518960
3	ElasticNet	32702.080706	81.745491
1	Ridge	33087.178311	81.313032
2	Lasso	33101.723515	81.296599
0	Linear	33101.896616	81.296403
4	Decision Tree	41136.679707	71.114636

In [63]:

```
import pandas as pd
import math
from math import sqrt
import numpy as np
import pandas_profiling
from pandas_profiling import ProfileReport as pp
import random
np.random.seed(42)

import matplotlib.pyplot as plt
import seaborn as sns
import IPython
from IPython.display import display
from scipy.stats import skew

%matplotlib inline

import sklearn
from sklearn import linear_model
from sklearn.model_selection import cross_val_score, train_test_split, cross_val_pr
edict
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV, ElasticNetCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score, make_scorer

pd.set_option('display.float_format', lambda x: '%3f' %x)


from sklearn.utils import resample
from sklearn.decomposition import PCA
from scipy import misc
from scipy import stats as st
from sklearn.model_selection import KFold, GridSearchCV
```

In [64]:

```
housing_train = pd.read_csv("train (1).csv")
housing_test = pd.read_csv("test (1).csv")
```

In [65]:

```
housing_train.SalePrice.describe()
```

Out[65]:

```
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
```

In [66]:

```
def corr_chart(df_corr):
    corr=df_corr.corr()
    #screen top half to get a triangle
    top = np.zeros_like(corr, dtype=np.bool)
    top[np.triu_indices_from(top)] = True
    fig=plt.figure()
    fig, ax = plt.subplots(figsize=(100,100))
    sns.heatmap(corr, mask=top, cmap='coolwarm',
        center = 0, square=True,
        linewidths=.5, cbar_kws={'shrink':.5},
        annot = True, annot_kws={'size': 9}, fmt = '.3f')
    plt.xticks(rotation=45) # rotate variable labels on columns (x axis)
    plt.yticks(rotation=0) # use horizontal variable labels on rows (y axis)
    plt.title('Correlation Heat Map')
```

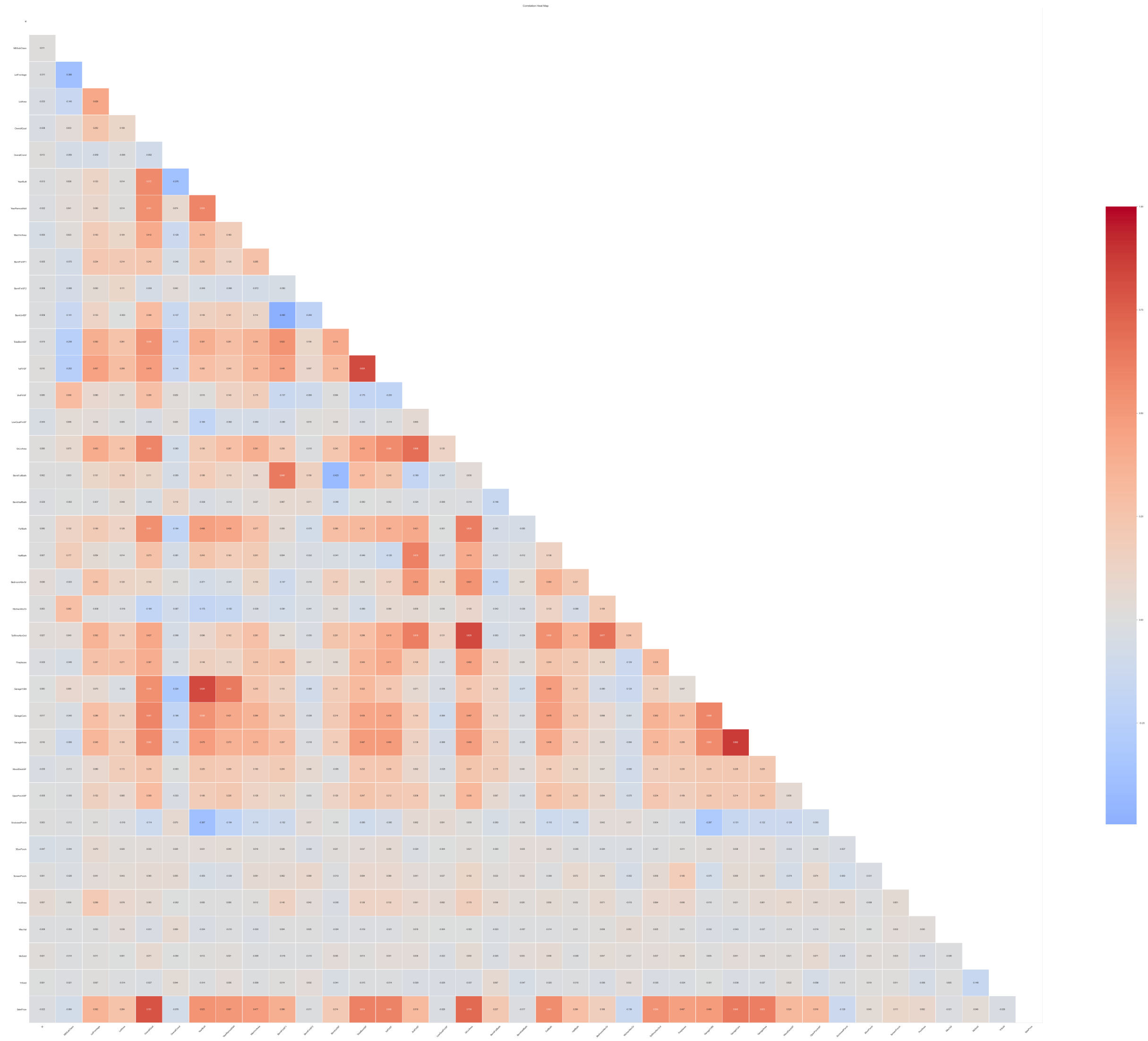
In [ ]:

```

```

In [67]:

```
corr_chart(df_corr = housing_train)
```

<Figure size 432x288 with 0 Axes>



In [68]:

```
housing_train.hist(bins=40,figsize=(30,20));
plt.show()
```

In [69]:

```
# x data and y data
delcolumns=['MSSubClass','MSZoning','Utilities','HouseStyle','Alley','OverallCond',
'Neighborhood','BsmtFinSF2','LowQualFinSF',
            'RoofStyle','RoofMatl','Exterior1st','KitchenAbvGr','GarageType','Garag
eFinish',
                                        'Exterior2nd','MasVnrType','ExterQual','Ext
erCond','Foundation',
                                        'BsmtQual','BsmtCond','BsmtFinType1',
                                        'BsmtFinType2','Electrical', 'Heating','Hea
tingQC',
                                        'BsmtHalfBath','FireplaceQu',
                                        'GarageQual','GarageCond','PavedDrive',
                                        'MiscFeature','LotFrontage','GarageYrBlt',
'MasVnrArea','PoolQC',
            'EnclosedPorch','MiscVal','YrSold','KitchenQual','Fence', 'Functional',
            'MoSold','3SsnPorch','PoolArea','Street','LandContour','LotConfig','Con
dition2','BsmtExposure','LandSlope', 'BldgType'
                                       ]
train_data=housing_train.drop(columns=delcolumns)
```

In [70]:

```
test_data=housing_test.drop(columns=delcolumns)
```

In [71]:

```
# Convert Condition to numeric
convert_condition={'Artery':0,'Feedr':1,'Norm':3,'RRNn':4,'RRAn':5,'PosN':6,'PosA':
7,'RRNe':8,'RRAe':9}
temp_df=train_data['Condition1'].map(convert_condition)
train_data['Condition1']=temp_df

temp_df=test_data['Condition1'].map(convert_condition)
test_data['Condition1']=temp_df
```

In [72]:

```
w
convert_to_binary={'N':0,'Y':1}
temp_df=train_data['CentralAir'].map(convert_to_binary)
train_data['CentralAir']=temp_df

temp_df=test_data['CentralAir'].map(convert_to_binary)
test_data['CentralAir']=temp_df
```

In [73]:

```
convert_salecondition={'Normal':0,'Abnorml':1,'AdjLand':2,'Alloca':3,'Family':4,'Pa
rtial':5}
temp_df=train_data['SaleCondition'].map(convert_salecondition)
train_data['SaleCondition']=temp_df

temp_df=test_data['SaleCondition'].map(convert_salecondition)
test_data['SaleCondition']=temp_df
```

In [74]:

```
convert_lotshape={'Reg':0,'IR1':1,'IR2':2,'IR3':3}
temp_df=train_data['LotShape'].map(convert_lotshape)
train_data['LotShape']=temp_df

temp_df=test_data['LotShape'].map(convert_lotshape)
test_data['LotShape']=temp_df
```

In [22]:

```
# Convert SaleType to numeric
convert_saletype={'WD':0,'CWD':1,'VWD':2,'New':3,'COD':4,'Con':5,'ConLw':6,'ConLI':
7,'ConLD':8,'Oth':9}
temp_df=train_data['SaleType'].map(convert_saletype)
train_data['SaleType']=temp_df

temp_df=test_data['SaleType'].map(convert_saletype)
test_data['SaleType']=temp_df
```

In [23]:

```
train_data.fillna(0,inplace=True)
test_data.fillna(0,inplace=True)
```

In [24]:

```
corr_data=train_data.corr()
corr_data
```

Out[24]:

	Id	LotArea	LotShape	Condition1	OverallQual	YearBuilt	YearRemodA
Id	1.000000	-0.033226	-0.024071	-0.018247	-0.028365	-0.012713	-0.0219
LotArea	-0.033226	1.000000	0.315484	0.032573	0.105806	0.014228	0.0137
LotShape	-0.024071	0.315484	1.000000	0.096835	0.198994	0.229365	0.1754
Condition1	-0.018247	0.032573	0.096835	1.000000	0.109307	0.198147	0.1559
OverallQual	-0.028365	0.105806	0.198994	0.109307	1.000000	0.572323	0.5506
YearBuilt	-0.012713	0.014228	0.229365	0.198147	0.572323	1.000000	0.5928
YearRemodAdd	-0.021998	0.013788	0.175488	0.155936	0.550684	0.592855	1.0000
BsmtFinSF1	-0.005024	0.214103	0.157718	0.060744	0.239666	0.249503	0.1284
BsmtUnfSF	-0.007940	-0.002618	0.014179	0.021346	0.308159	0.149040	0.1811
TotalBsmtSF	-0.015415	0.260833	0.200469	0.085029	0.537808	0.391452	0.2910
CentralAir	0.009821	0.049755	0.099138	0.038493	0.272038	0.381831	0.2988
1stFlrSF	0.010496	0.299475	0.189251	0.096499	0.476224	0.281986	0.2403
2ndFlrSF	0.005590	0.050986	0.089380	-0.012598	0.295493	0.010308	0.1400
GrLivArea	0.008273	0.263116	0.212741	0.053928	0.593007	0.199010	0.2873
BsmtFullBath	0.002289	0.158155	0.064638	0.038696	0.111098	0.187599	0.1194
FullBath	0.005587	0.126031	0.184213	0.104931	0.550600	0.468271	0.4390
HalfBath	0.006784	0.014259	0.116576	0.069078	0.273458	0.242656	0.1833
BedroomAbvGr	0.037719	0.119690	0.060028	0.005432	0.101676	-0.070651	-0.0405
TotRmsAbvGrd	0.027239	0.190015	0.137148	0.041608	0.427452	0.095589	0.1917
Fireplaces	-0.019772	0.271364	0.202019	0.022800	0.396765	0.147716	0.1125
GarageCars	0.016570	0.154871	0.194984	0.110557	0.600671	0.537850	0.4206
GarageArea	0.017634	0.180403	0.173472	0.089003	0.562022	0.478954	0.3716
WoodDeckSF	-0.029643	0.171698	0.161717	0.059400	0.238923	0.224880	0.2057
OpenPorchSF	-0.000477	0.084774	0.093135	0.081863	0.308819	0.188686	0.2262
ScreenPorch	0.001330	0.043160	0.065182	0.018132	0.064886	-0.050364	-0.0387
SaleType	-0.028774	-0.004436	0.008355	0.025398	0.134304	0.160709	0.1090
SaleCondition	-0.021060	0.013773	0.027139	0.022991	0.269625	0.290899	0.2610
SalePrice	-0.021917	0.263843	0.267759	0.113916	0.790982	0.522897	0.5071

28 rows × 28 columns

In [34]:

```
train_x=train_data.copy()
train_x.drop(columns=['SalePrice'], inplace=True)
train_y=train_data[['SalePrice']]
```

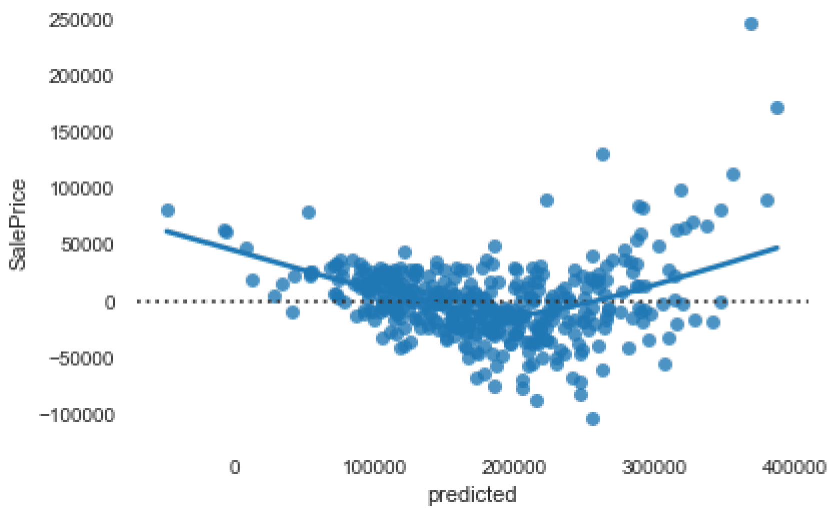
In [35]:

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
# split data using test_train_split
lin_x_train,lin_x_test,lin_y_train,lin_y_test=train_test_split(train_x,train_y,test
_size=.3,random_state=10)
```

In [36]:

```
lin_test=test_data.copy()
lin_test_res=test_data[['Id']]


#Linear Regression
lin_reg=LinearRegression(fit_intercept=True).fit(lin_x_train,lin_y_train)

lin_score=lin_reg.score(lin_x_test,lin_y_test)

print('Number of features used:',np.sum(lin_reg.coef_ !=0))
print('Training set score:', lin_reg.score(lin_x_train,lin_y_train))
print('Testing set score:',lin_reg.score(lin_x_test,lin_y_test))

lin_y_test['predicted']=lin_reg.predict(lin_x_test)

sns.residplot(lin_y_test['predicted'],lin_y_test['SalePrice'],lowess=True)
lin_np=np.sqrt(mean_squared_error(lin_y_test['SalePrice'],lin_y_test['predicted']))

lin_test_res['SalePrice']=lin_reg.predict(lin_test)
```

```
Number of features used: 27
Training set score: 0.7887849834489229
Testing set score: 0.8129640324858132

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



In [47]:

```
grad_test=test_data.copy()
grad_test_res=test_data[['Id']]

# split data using test_train_split
grad_x_train,grad_x_test,grad_y_train,grad_y_test=train_test_split(train_x,train_y,
test_size=.3,random_state=10)

from sklearn.ensemble import GradientBoostingRegressor
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=120, random_state=42)
gbrt.fit(grad_x_train,grad_y_train)

errors = [mean_squared_error(grad_y_test, y_pred)
          for y_pred in gbrt.staged_predict(grad_x_test)]
bst_n_estimators = np.argmin(errors)

gbrt_best = GradientBoostingRegressor(max_depth=2,n_estimators=bst_n_estimators, ra
ndom_state=42)
gbrt_best.fit(grad_x_train,grad_y_train)
```

```
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/utils/val
idation.py:73: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  return f(**kwargs)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/utils/val
idation.py:73: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  return f(**kwargs)
```

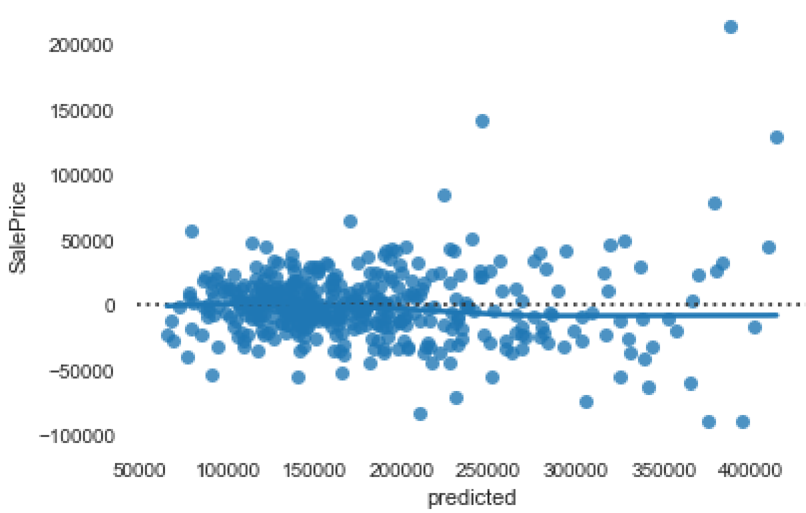
Out[47]:

```
GradientBoostingRegressor(max_depth=2, n_estimators=119, random_state=4
2)
```

In [48]:

```
grad_score=gbrt_best.score(grad_x_test,grad_y_test)
print("Training set score: {:.2f}".format(gbrt_best.score(grad_x_train,grad_y_train
)))
print("Test set score: {:.2f}".format(gbrt_best.score(grad_x_test,grad_y_test)))
#print("Number of features used: {}".format(np.sum(gbrt_best.coef_ !=0)))


grad_y_test['predicted']= gbrt_best.predict(grad_x_test)
# Plot the residuals after fitting a linear model
sns.residplot(grad_y_test['predicted'], grad_y_test['SalePrice'], lowess=True)


grad_np=np.sqrt(mean_squared_error(grad_y_test['SalePrice'],grad_y_test['predicted'
]))

grad_test_res['SalePrice']=gbrt_best.predict(grad_test)
```

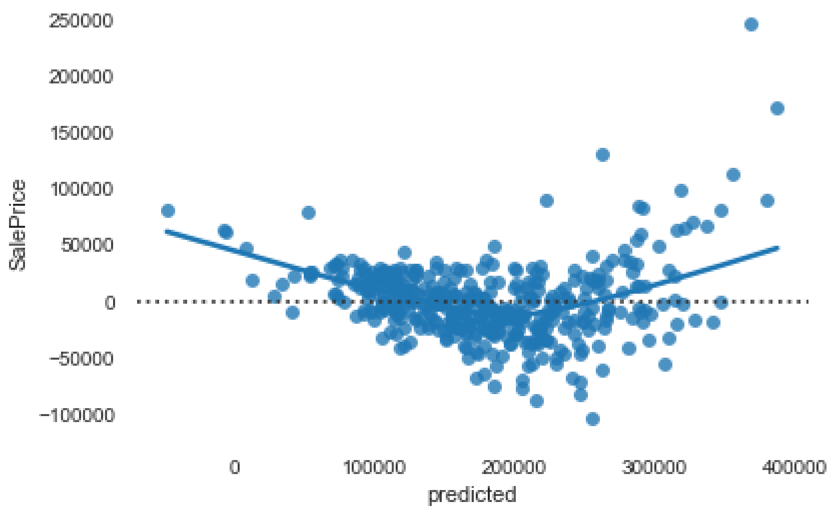
```
Training set score: 0.95
Test set score: 0.88

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



In [49]:

```
lasso_test=test_data.copy()
lasso_test_res=test_data[['Id']]

from sklearn.linear_model import Lasso
lasso_train_x, lasso_test_x,lasso_train_y,lasso_test_y=train_test_split(train_x,tra
in_y, test_size=.3,random_state=10)
lasso_model = Lasso(alpha=.1, max_iter = 10000,fit_intercept=True,tol=0.01).fit(las
so_train_x,lasso_train_y)
print("Number of features used: {}".format(np.sum(lasso_model.coef_ !=0)))
print("Training set score:" ,lasso_model.score(lasso_train_x,lasso_train_y))
lasso_score=lasso_model.score(lasso_test_x,lasso_test_y)
print("Test set score:" ,lasso_score)
lasso_test_y['predicted']=lasso_model.predict(lasso_test_x)
sns.residplot(lasso_test_y['predicted'],lasso_test_y['SalePrice'],lowess=True)

lasso_np=np.sqrt(mean_squared_error(lasso_test_y['SalePrice'],lasso_test_y['predict
ed']))

lasso_test_res['SalePrice']=lasso_model.predict(lasso_test)
```

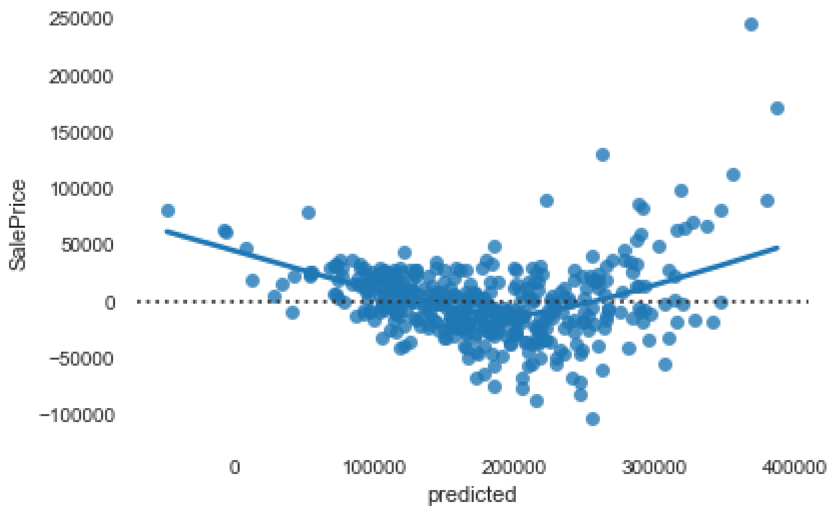
```
Number of features used: 27
Training set score: 0.7887849833262455
Test set score: 0.8129659886345015

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  app.launch_new_instance()
```



In [50]:

```
#ridge regression
ridge_test=test_data.copy()
ridge_test_res=test_data[['Id']]

from sklearn.linear_model import Ridge
ridge_train_x, ridge_test_x,ridge_train_y,ridge_test_y=train_test_split(train_x,tra
in_y, test_size=.3,random_state=10)
ridge_model = Ridge(alpha=1, solver="cholesky",fit_intercept=True).fit(ridge_train_
x,ridge_train_y)
print("Number of features used: {}".format(np.sum(ridge_model.coef_ !=0)))
print("Training set score:" ,ridge_model.score(ridge_train_x,ridge_train_y))
ridge_score=ridge_model.score(ridge_test_x,ridge_test_y)
print("Test set score:" ,ridge_score)
ridge_test_y['predicted']=ridge_model.predict(ridge_test_x)
sns.residplot(ridge_test_y['predicted'],ridge_test_y['SalePrice'],lowess=True)
ridge_np=np.sqrt(mean_squared_error(ridge_test_y['SalePrice'],ridge_test_y['predict
ed']))

ridge_test_res['SalePrice']=ridge_model.predict(ridge_test)
```

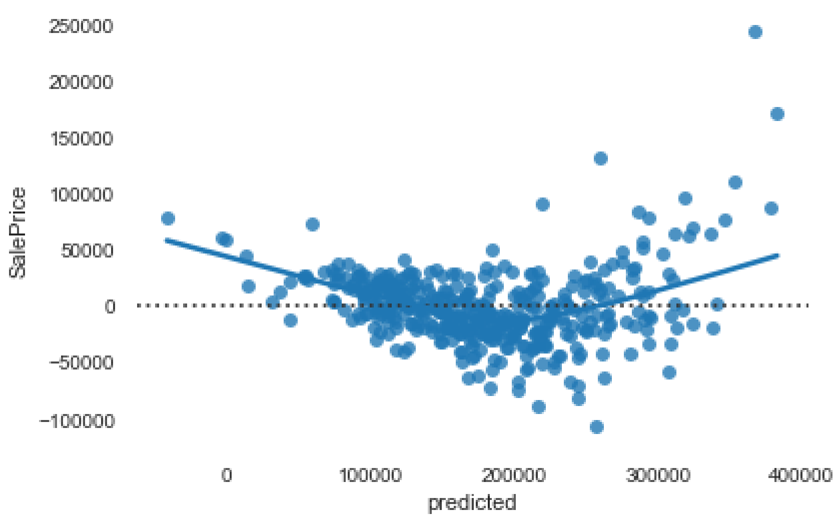
```
Number of features used: 27
Training set score: 0.788784416550699
Test set score: 0.8131303214712008

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  app.launch_new_instance()
```



In [53]:

```
es_test=test_data.copy()
es_test_res=test_data[['Id']]

from sklearn.linear_model import ElasticNet
es_train_x, es_test_x,es_train_y,es_test_y=train_test_split(train_x,train_y, test_s
ize=.3,random_state=10)
es_model = ElasticNet(random_state=1, alpha=0.1,fit_intercept=True,tol=0.01).fit(es
_train_x,es_train_y)
print("Number of features used: {}".format(np.sum(es_model.coef_ !=0)))
print("Training set score:" ,es_model.score(es_train_x,es_train_y))
es_score=es_model.score(es_test_x,es_test_y)
print("Test set score:" ,es_score)
es_test_y['predicted']=es_model.predict(es_test_x)
sns.residplot(es_test_y['predicted'],es_test_y['SalePrice'],lowess=True)
es_np=np.sqrt(mean_squared_error(es_test_y['SalePrice'],es_test_y['predicted']))

es_test_res['SalePrice']=es_model.predict(es_test)
```

```
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear_mo
del/_coordinate_descent.py:531: ConvergenceWarning: Objective did not c
onverge. You might want to increase the number of iterations. Duality g
ap: 671258869385.2974, tolerance: 66374050967.45818
  positive)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  from ipykernel import kernelapp as app

Number of features used: 27
Training set score: 0.7880790115300065
Test set score: 0.8174549139372843
```



In [56]:

```
des_tree=test_data.copy()
des_tree_res=test_data[['Id']]

# split data using test_train_split
des_x_train,des_x_test,des_y_train,des_y_test=train_test_split(train_x,train_y,test
_size=.3,random_state=10)

#Decision tree
from sklearn.tree import DecisionTreeRegressor  # machine learning tree

tree_model_maker = DecisionTreeRegressor(random_state = 9999, max_depth = 5)


# fit regression tree using model 1 training/test split
tree_model_fit = tree_model_maker.fit(des_x_train, des_y_train)

#print("Number of features used: {}".format(np.sum(tree_model_maker.coef_ !=0)))

tree_score=tree_model_maker.score(des_x_test,des_y_test)

des_y_train['predicted'] = tree_model_fit.predict(des_x_train)
full_tree_train_result = round(np.power(des_y_train['SalePrice'].corr(des_y_train[
'predicted']),2),3)
print('\nFull Tree Proportion of Training Set Variance Accounted for: ',full_tree_t
rain_result)

# compute the proportion of response variance for test data

des_y_test['predicted'] = tree_model_fit.predict(des_x_test)
full_tree_train_result = round(np.power(des_y_test['SalePrice'].corr(des_y_test['pr
edicted']),2),3)
print('\nFull Tree Proportion of Training Set Variance Accounted for: ',full_tree_t
rain_result)

tree_np=np.sqrt(mean_squared_error(des_y_test['SalePrice'],des_y_test['predicted'
]))


des_tree_res['SalePrice']=tree_model_maker.predict(des_tree)
```

```
Full Tree Proportion of Training Set Variance Accounted for:  0.873

Full Tree Proportion of Training Set Variance Accounted for:  0.713

/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/Users/avadhani/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:33: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [62]:

```
data={'Model':['Linear','Ridge','Lasso','ElasticNet','Decision Tree','GradientBoos
t'],
      'RMSE':[lin_np,ridge_np,lasso_np,es_np,tree_np,grad_np],
      'Score':[round(lin_score*100,30),round(ridge_score*100,30),
               round(lasso_score*100,30),round(es_score*100,30),
               round(tree_score*100,30),
               round(grad_score*100,30)
              ]
     }
result=pd.DataFrame(data,columns=['Model','RMSE','Score'])
result.sort_values(by='RMSE')
```

Out[62]:

	Model	RMSE	Score
5	GradientBoost	27040.561603	87.518960
3	ElasticNet	32702.080706	81.745491
1	Ridge	33087.178311	81.313032
2	Lasso	33101.723515	81.296599
0	Linear	33101.896616	81.296403
4	Decision Tree	41136.679707	71.114636

In [77]:

```
result.to_csv("submission.csv")
```

In [ ]:

```

```

submission

Id	SalePrice
1461	120836.64818814676
1462	160942.77326224043
1463	184873.13385510867
1464	194758.0024079365
1465	193284.76680408217
1466	172112.49789589652
1467	176406.69460978877
1468	162456.52475293228
1469	182490.81017712774
1470	122818.12608462932
1471	196528.59613622967
1472	94169.55381847931
1473	94258.51487861751
1474	147670.28366322152
1475	112607.72328148005
1476	381447.6943090885
1477	251927.0426714853
1478	285128.8545021899
1479	280870.91167147073
1480	518688.9539986728
1481	316499.22428498295
1482	204981.7067835306
1483	178630.18949332973
1484	164210.72767673328
1485	182144.3530275173
1486	195121.15256743127
1487	345249.8938972267
1488	231244.41394081106
1489	199342.45159414748

1490	236974.31739543067
1491	194457.7881052933
1492	91655.11782999629
1493	174494.1513090954
1494	294947.9582547871
1495	284042.4246968649
1496	239012.54540375536
1497	178614.6867253717
1498	164348.62117456656
1499	158819.58054959914
1500	155694.4611816735
1501	176559.50472496313
1502	142104.89055519336
1503	294628.9079861114
1504	240206.06324593
1505	228171.7498333699
1506	185991.24306553518
1507	255567.36266964642
1508	196230.27348264135
1509	158679.03842631617
1510	144849.35060152764
1511	151798.17502245464
1512	171035.0117112518
1513	143381.98701848838
1514	157023.82408884808
1515	203284.3324512461
1516	149967.4925592457
1517	161372.6611499341
1518	129063.07333491408
1519	217328.67507461802
1520	132499.02912354565
1521	136117.47872963717

1522	172806.005554176
1523	112931.11300154263
1524	125656.461373078
1525	122691.88952110097
1526	112580.011208591
1527	102018.25337583879
1528	132764.2470536191
1529	149038.52176680643
1530	173780.6940118463
1531	103494.444826792
1532	95705.47811664142
1533	147532.49464035616
1534	124282.20416264288
1535	150577.90216550467
1536	107016.40182378245
1537	57191.61634576515
1538	147814.9125245537
1539	197203.82794514744
1540	94375.04627220715
1541	142560.569530372
1542	143588.47597438697
1543	197799.0109074269
1544	82500.92337354738
1545	109730.26103896994
1546	133336.02024545742
1547	135604.09359696467
1548	137274.44651577438
1549	115906.14198258953
1550	135764.06880454437
1551	109621.30584111776
1552	139043.82324854896
1553	154553.14818785936

1554	113204.26226151263
1555	163570.56697123472
1556	85585.35606765846
1557	110174.33715043533
1558	99995.48741631283
1559	67592.65314289894
1560	130086.89808220865
1561	137640.24072296018
1562	130136.24384873349
1563	119880.18008262648
1564	161491.78920061045
1565	147936.79165959702
1566	233497.90855360826
1567	74685.05014612534
1568	232075.85591156944
1569	132185.2685169939
1570	133894.95365244232
1571	120103.56909307072
1572	145790.44061083178
1573	246761.70623305245
1574	115570.78441060711
1575	221502.81526649487
1576	248987.31226058205
1577	185731.49493435194
1578	142386.9909435073
1579	146065.27931105258
1580	192405.3121087636
1581	152715.40914172615
1582	124366.48831744274
1583	308383.0815132928
1584	223806.15474038376
1585	139399.9177770101

1586	67154.5752918953
1587	99852.81532068273
1588	148773.5851411884
1589	100748.92969431632
1590	135122.4614569348
1591	94385.29500707067
1592	118702.81828903248
1593	126004.00608916325
1594	130108.69598930067
1595	107789.5571181866
1596	234892.3542841479
1597	194705.7351880877
1598	216297.68926935387
1599	163676.99912822995
1600	163364.34488229535
1601	62976.688699262464
1602	114787.99069889865
1603	76692.85964096163
1604	261591.90298338025
1605	240675.07453202913
1606	169459.05104549925
1607	170423.71979568826
1608	215362.89869792067
1609	185561.0680405894
1610	150956.9409309456
1611	139867.0946046081
1612	169282.71219105204
1613	166515.00520630123
1614	122271.87344975665
1615	87351.02595141699
1616	74441.16561855526
1617	92130.97794976964

1618	127419.45442833762
1619	139454.2493155578
1620	151135.40048837272
1621	140401.5626239054
1622	141856.31888598402
1623	260339.13247485537
1624	214299.6379042953
1625	119565.19994944813
1626	163352.39973019378
1627	192981.20729533606
1628	270984.0065973604
1629	176333.3160811252
1630	367512.43441668525
1631	221969.01903641177
1632	240024.62398113823
1633	168240.8126941375
1634	187191.47369696238
1635	175543.12862517993
1636	154235.57913730794
1637	201648.58955300495
1638	180824.4569744159
1639	183564.2775046416
1640	253590.12195444579
1641	176944.20822987854
1642	235493.6078895029
1643	216434.95980806812
1644	231799.88872747179
1645	202565.34202727623
1646	158315.9670644639
1647	156338.23706136507
1648	131817.69502295347
1649	140347.13834405039

1650	112630.29626729521
1651	121685.75687306505
1652	95944.66503129344
1653	97964.87508069725
1654	145221.34607838577
1655	140007.02605512776
1656	142185.71995387992
1657	149285.70328056172
1658	142865.86906468132
1659	118593.35465592484
1660	158884.44815628638
1661	459563.6287461878
1662	396768.6495041878
1663	393125.64746865875
1664	519710.9398469794
1665	308915.4725325517
1666	323489.0995796826
1667	385650.7714964621
1668	339915.3029906076
1669	297814.51252013823
1670	352251.71884313616
1671	260813.80920615344
1672	468020.8775898179
1673	288128.84174544853
1674	241550.54251223593
1675	195891.80086071033
1676	194998.45206811832
1677	221267.2880053418
1678	493688.9622833114
1679	388178.311992224
1680	353289.36581017217
1681	247966.25428127887

1682	299654.96327959094
1683	182009.95828128862
1684	171562.16491012037
1685	172030.30817194394
1686	166483.99748392473
1687	167377.81441235528
1688	189639.2682543897
1689	194235.48567851828
1690	194437.29520352595
1691	180689.45724207786
1692	261422.49007007806
1693	166174.86342839067
1694	181145.25866986162
1695	162432.065029031
1696	270457.904836825
1697	167228.24813308287
1698	357435.2464106193
1699	318621.2512570062
1700	254710.08644349725
1701	263760.4580554084
1702	247769.36479179063
1703	243891.37262199103
1704	277009.15983690374
1705	230989.92337403435
1706	438802.35371586605
1707	215909.31742399733
1708	206967.05711591482
1709	266468.78733956366
1710	216289.97261158732
1711	266666.8175245235
1712	259907.80759303298
1713	275931.1865170959

1714	223117.2937100082
1715	206804.18315041377
1716	182891.63937257655
1717	170290.03997851224
1718	134630.87137642535
1719	199501.83408456732
1720	236758.21263985313
1721	165052.46789749467
1722	125263.46696899708
1723	154488.53258528208
1724	210136.5269206205
1725	240010.1867703883
1726	184875.13010686397
1727	159094.71177442351
1728	175302.17158739304
1729	168327.80377470568
1730	155886.0197130488
1731	121396.22487211281
1732	127344.6378109873
1733	118037.33690305584
1734	122121.49166348536
1735	127659.17323309886
1736	111529.17499723252
1737	280662.6715878352
1738	237525.14675685868
1739	266171.0759884806
1740	227445.94015377603
1741	195085.05115324186
1742	174519.09995770227
1743	175440.5636487453
1744	310679.36549172725
1745	222384.412218421

1746	177294.7389255638
1747	212169.96362060437
1748	225615.57910260666
1749	148062.62789585078
1750	126038.7464350779
1751	249643.2177891669
1752	118532.8948398944
1753	149532.7510883439
1754	185953.31448310893
1755	167638.7162492225
1756	127434.04725423292
1757	122201.0597524411
1758	143026.80411099998
1759	163257.5322747114
1760	166212.2674853775
1761	144008.07740812065
1762	179489.88584798685
1763	172120.0144800203
1764	114538.8668231526
1765	168396.24872360472
1766	182098.27866464536
1767	224152.74499446154
1768	144670.9067628761
1769	170548.72363867296
1770	148617.88940814804
1771	124310.26363548054
1772	133810.36254816744
1773	126849.93408507603
1774	139422.87562601766
1775	139059.73329593992
1776	125111.51271590899
1777	115666.0755651742

1778	139378.5067348977
1779	117382.18938495655
1780	173363.86578683485
1781	130313.87833719165
1782	88590.30480468023
1783	143199.65803942122
1784	103922.5384223949
1785	123713.86102507742
1786	132009.45714961994
1787	162843.03097283078
1788	55658.96594338182
1789	94156.39914388905
1790	75670.33726094814
1791	180930.0994481076
1792	162774.17867558997
1793	125630.47025227323
1794	152032.21198232385
1795	133192.78454864683
1796	118317.73257744574
1797	113401.2024892309
1798	122494.06741117351
1799	109608.10735980734
1800	132004.23002465576
1801	129357.31190596995
1802	129546.05344929095
1803	146156.47898100596
1804	142693.35852185523
1805	135225.32747238406
1806	122442.80369279563
1807	127904.71850592457
1808	119152.31483890356
1809	123889.20901594633

1810	140474.51664092872
1811	100605.80950631136
1812	100290.16601073713
1813	122751.74195363995
1814	98267.11024376856
1815	52903.07514862192
1816	99530.98453745224
1817	109621.51068253964
1818	148721.49358147325
1819	120668.09164667418
1820	54537.09380220737
1821	112894.69885446671
1822	150463.74983733436
1823	46900.14821406295
1824	131161.77650298516
1825	131044.4041967179
1826	104458.24950392685
1827	100826.05315961399
1828	134707.13421783043
1829	116983.28350638616
1830	143282.4558427664
1831	142791.79335950644
1832	72909.3682996447
1833	141863.38852469024
1834	118206.25299126784
1835	121589.31131871534
1836	126341.70289298176
1837	89952.2206317079
1838	122118.83571756343
1839	94252.38778507466
1840	148474.27081215286
1841	137155.45448222585

1842	86428.7478107373
1843	127755.8924841136
1844	134086.271311681
1845	138651.52114762465
1846	156139.34799734468
1847	162707.78300653643
1848	53117.68402086589
1849	115650.98962210823
1850	117618.97599491317
1851	157284.07820462767
1852	123883.38679194805
1853	128973.11480631685
1854	171720.607880611
1855	150959.62863328602
1856	221879.85401167514
1857	155622.4867521339
1858	138232.6643679453
1859	117763.88550052776
1860	149845.15959519707
1861	117282.71694642368
1862	304385.3558357046
1863	282696.6871788736
1864	282777.52026391897
1865	333639.4650659029
1866	333519.90469433664
1867	229624.35753536143
1868	301381.5978507059
1869	204393.01236774106
1870	227039.56997246726
1871	247004.7593585877
1872	172370.71636468126
1873	238953.49991963909

1874	148087.56243804065
1875	199992.75185500557
1876	197037.77966827955
1877	210576.87224141447
1878	206553.4089277552
1879	128958.55306083483
1880	132872.49350921944
1881	246917.96159677443
1882	248863.0331601465
1883	188236.61191452865
1884	207568.98470222118
1885	234957.96041675576
1886	289592.26359628
1887	210943.55632959862
1888	271499.9360546469
1889	165468.5448836326
1890	119946.60498450897
1891	130073.95521112252
1892	98786.017970137
1893	132340.2111293729
1894	126584.88645345214
1895	133522.78309950288
1896	126484.12379621298
1897	111681.6856607406
1898	112215.7653770949
1899	156700.9404009163
1900	145070.03355132384
1901	173656.3473983059
1902	152467.81635556786
1903	210132.7866572285
1904	138917.36385767505
1905	196905.54293690668

1906	160288.61951429007
1907	205193.01263731078
1908	109455.04540106177
1909	132909.37806177585
1910	118752.26563093175
1911	221251.8327481307
1912	332498.8465177402
1913	145409.8015193522
1914	59383.61996620319
1915	321464.93403765105
1916	64598.84536123933
1917	247314.72743305212
1918	139155.02911654103
1919	174515.28262701828
1920	152652.3910777053
1921	414649.32794893684
1922	320089.7848219646
1923	229816.6106132259
1924	227020.50269108094
1925	195464.81974995014
1926	389002.12522668054
1927	134597.18338167822
1928	158288.90810851057
1929	125186.54978044091
1930	131057.44487362748
1931	145781.2683849099
1932	142348.2728310146
1933	180427.37542176028
1934	180116.85866819313
1935	170606.78572793512
1936	190916.88555574848
1937	182945.38635511167

1938	174873.84607600223
1939	246187.45245257596
1940	192109.30414790183
1941	170192.3804690078
1942	185984.73070713622
1943	207068.55550000246
1944	359767.03494004335
1945	399667.8942306078
1946	139879.93184023746
1947	312778.54770389415
1948	172229.86534782976
1949	250849.0269620963
1950	182344.27356109465
1951	250655.93874719655
1952	207931.18132732075
1953	166109.89584612334
1954	181353.283990128
1955	133205.14420084737
1956	306542.7793458395
1957	154942.00507076675
1958	306435.5356314783
1959	142643.92206151126
1960	116784.30202708486
1961	120178.27145902476
1962	96534.3469389212
1963	106049.46798217655
1964	107472.64908047767
1965	143352.1758504812
1966	138687.73181919265
1967	294347.7684274767
1968	427505.407880991
1969	388706.56201462686

1970	417008.8271480262
1971	449909.6638425434
1972	378946.70538660884
1973	264465.09596815536
1974	330444.96512819064
1975	550495.9448067334
1976	268173.1338852346
1977	353585.78364420944
1978	351550.73095021554
1979	336879.86483394145
1980	201670.01724089537
1981	356617.2353830648
1982	216445.24709581578
1983	199643.21297964532
1984	168349.37045909342
1985	215527.58305221394
1986	211975.32103744248
1987	174046.56741577887
1988	176155.07786692452
1989	192535.1499234921
1990	210472.7083402244
1991	236419.22147258825
1992	212788.1039832972
1993	166889.85458333007
1994	230379.1244873412
1995	182546.0322274487
1996	252874.7172398297
1997	314918.6068706732
1998	337132.0266422077
1999	273714.9711584081
2000	321491.9104659803
2001	281745.19628662465

2002	242898.86745711727
2003	254610.55304505257
2004	279931.2348130214
2005	218394.94604826433
2006	220046.5126011893
2007	248443.80955795723
2008	215604.18199285993
2009	195671.99564974048
2010	194212.29843808166
2011	138492.81619136478
2012	173108.82357420574
2013	179914.70355585864
2014	184620.87242511276
2015	204136.3362820081
2016	193687.69221483544
2017	194983.34642307297
2018	118092.53598095337
2019	134417.01210688538
2020	103715.9800168298
2021	103620.44239756739
2022	181877.66529637852
2023	146919.97560621472
2024	267742.36658262456
2025	352529.95921046747
2026	172048.19061573694
2027	156349.07079841488
2028	150103.73102556134
2029	166540.0293035034
2030	267817.17396726063
2031	230259.38041340117
2032	250605.21768354287
2033	256237.60151297064

2034	170600.44393809722
2035	214867.57656880893
2036	197230.4676137406
2037	206114.67877607158
2038	305871.0355170848
2039	223645.02739156407
2040	279820.80083745753
2041	300763.6460824328
2042	206938.33238209883
2043	175579.4831071274
2044	171179.96591160895
2045	209153.10191340686
2046	145151.35054616784
2047	146637.79684757022
2048	138073.54868134452
2049	140293.1014995623
2050	176812.85166980955
2051	105082.16596983508
2052	124494.53937498535
2053	148545.25404032247
2054	88446.96304621751
2055	159064.94573572342
2056	141830.2219384109
2057	114746.61489755657
2058	221769.6256902655
2059	130571.26507561287
2060	173417.73367455107
2061	172930.43774495792
2062	130702.62087929185
2063	115439.96158408042
2064	139825.25943538803
2065	118650.82357640064

2066	159109.349586661
2067	110853.99673808321
2068	143073.15987257627
2069	87534.75841966864
2070	115837.3052843392
2071	92132.171521137
2072	129555.82965508982
2073	134443.3443531381
2074	188329.8193693507
2075	143192.06867994173
2076	120968.08151324844
2077	156616.45064906287
2078	126768.66696270704
2079	131779.30836231806
2080	117888.61028035745
2081	126495.29391263149
2082	123074.23442010838
2083	145281.95133188574
2084	108687.58119519548
2085	119955.44698494364
2086	107650.92146612918
2087	120930.71550947547
2088	94561.99453439449
2089	78001.16146237405
2090	126363.45157718056
2091	112960.64411295479
2092	126474.40979067422
2093	117616.7973233023
2094	107104.15211529838
2095	148874.2419991118
2096	82357.78262693611
2097	96260.88019716609

2098	146987.31305905894
2099	53066.01784285969
2100	69853.73187184392
2101	118376.84327063714
2102	123589.62808243508
2103	102378.31773966405
2104	134030.30849501022
2105	126541.63245795718
2106	58030.35358968751
2107	197214.31212119295
2108	116112.92812799891
2109	111535.08339847217
2110	134352.08259136506
2111	137503.52323746175
2112	139317.23801188148
2113	122890.61297309902
2114	117341.14808126382
2115	164276.35558029683
2116	118960.34666585058
2117	150457.84724959478
2118	125316.28953026925
2119	114759.7801680415
2120	111850.23140260506
2121	89633.93550626494
2122	117941.43806707938
2123	87203.67436120377
2124	157949.92186506436
2125	124136.65433551474
2126	150584.55396586773
2127	156775.26761809355
2128	126178.18691234579
2129	92632.18453060077

2130	137446.4457376035
2131	136358.41733949524
2132	115571.89156651356
2133	124506.73898894219
2134	120013.49932483473
2135	93219.71688842415
2136	77208.73796603094
2137	108273.27420667082
2138	132690.18514459176
2139	153314.7424217518
2140	141741.5607322057
2141	147874.89321433118
2142	127796.94708663269
2143	150836.09712531712
2144	119536.46017051776
2145	137047.4063606317
2146	161209.68840442126
2147	146514.67016492388
2148	130281.2503514195
2149	150751.12366070208
2150	220277.82806028347
2151	110994.57437955265
2152	183251.07003055594
2153	161089.6297599999
2154	110909.16754949123
2155	140611.55013459976
2156	273525.2118314756
2157	230924.1808856448
2158	238838.29323060327
2159	210638.876548638
2160	175081.1282191331
2161	239968.64457000146

2162	381162.68726390344
2163	357731.8096800037
2164	243117.47529202036
2165	194295.44295064057
2166	154717.9180267879
2167	213469.14632267284
2168	197479.94605332377
2169	196776.37985689542
2170	217175.70475754153
2171	155287.7866516374
2172	131837.47342260473
2173	165849.75361470762
2174	224813.4288906602
2175	285620.5957438849
2176	318002.0942964553
2177	241338.34068157495
2178	208621.83997482012
2179	137929.2184999648
2180	222723.676025561
2181	191061.5111290261
2182	222757.33854629914
2183	189149.74764767714
2184	121567.60899481011
2185	122531.14833284388
2186	146639.4558104984
2187	153567.95053679787
2188	156675.2572870483
2189	297954.9116237495
2190	80226.01224232535
2191	81950.2785520556
2192	91584.47145989489
2193	107444.03680743198

2194	102676.09987925668
2195	109343.17593450757
2196	98613.21669999434
2197	118248.70671384796
2198	155549.60436213174
2199	179328.09892440445
2200	151139.04084060926
2201	141622.753944881
2202	216512.37632037146
2203	136137.06932989572
2204	184989.4743757192
2205	117746.09760520815
2206	139821.2571845432
2207	215493.77310801967
2208	263391.29248238634
2209	247269.36294438312
2210	119012.52217836691
2211	113564.75286788918
2212	119073.3572397058
2213	108105.39335207304
2214	140299.93276777666
2215	97925.176454283
2216	145029.04584061052
2217	63288.572055188
2218	71479.68798590662
2219	66011.80817941415
2220	69715.03687846563
2221	323480.94655391254
2222	279877.119334917
2223	295687.55583896354
2224	209303.55150862283
2225	120156.55090802153

2226	189102.61372878047
2227	200934.87472127323
2228	264359.83446622797
2229	253573.56791814754
2230	152457.77972142797
2231	216354.5169214681
2232	181497.07641354608
2233	180226.51581568483
2234	240984.5286214071
2235	225538.79036076675
2236	265396.68362122844
2237	333056.29759886296
2238	192280.21053935162
2239	110600.12828591303
2240	156231.45501203876
2241	157656.96564556423
2242	126133.58281845212
2243	127136.52422518577
2244	101998.93065365608
2245	96107.43482117404
2246	144668.5056219468
2247	110741.64851908549
2248	123066.15734660474
2249	119622.65875524661
2250	129515.48013347678
2251	98094.80453282053
2252	174496.3553630284
2253	153163.40987665174
2254	178527.39688205667
2255	188141.54582039543
2256	178066.59295421018
2257	208568.2126968066

2258	161151.85132087593
2259	175274.19943332742
2260	155162.57585889692
2261	180568.43614696083
2262	205595.991681782
2263	389443.26933445164
2264	499031.80814187095
2265	174494.95505877535
2266	293831.57355398283
2267	378327.6381049642
2268	420276.37267609104
2269	146924.15056323845
2270	196114.52899280295
2271	214385.58691444443
2272	197453.81212611066
2273	164290.3513957156
2274	179322.52774543874
2275	163048.6261686984
2276	191007.73820933033
2277	181524.00323405158
2278	152198.52718408997
2279	123431.56377432683
2280	119140.31092434301
2281	171244.3135968594
2282	189544.32479242
2283	105109.23298569757
2284	111581.91981028272
2285	139564.79206175008
2286	124435.83425590764
2287	350058.0368387328
2288	271598.3176136865
2289	377200.0863142683

2290	450578.6210786617
2291	332632.551193667
2292	444225.83458299306
2293	511331.48894364893
2294	402909.0660799722
2295	492323.82243992435
2296	287234.5827682698
2297	325412.28649289237
2298	332734.0873371655
2299	374884.45161580667
2300	335167.5372749887
2301	303291.4406565667
2302	247991.82935922474
2303	241716.50473280213
2304	244896.39238384098
2305	195807.6969025321
2306	191285.34808550344
2307	200347.33294285325
2308	224586.65187788283
2309	283299.7450296121
2310	208437.8446952845
2311	198915.5372738291
2312	174523.90120386245
2313	172742.31859010918
2314	170089.34189835566
2315	178256.54588511048
2316	193000.1748656128
2317	188544.96268210377
2318	179689.66304402184
2319	180128.61370682536
2320	181844.57976317249
2321	229845.49728196103

2322	176357.45927755148
2323	186668.44397529436
2324	172365.98353627173
2325	214459.2190684477
2326	170449.52503308028
2327	205617.45584467775
2328	223133.42272703536
2329	189327.52349170166
2330	180308.41132231933
2331	357497.2749048962
2332	396234.29106979514
2333	322459.3026516138
2334	260187.09954749563
2335	278968.2465063509
2336	320971.68996639596
2337	200842.7304786394
2338	266760.7344220367
2339	216904.56313872075
2340	409724.53374638595
2341	214766.78285755258
2342	230672.74444774166
2343	225655.59888036244
2344	221263.38221252742
2345	234556.523821938
2346	210509.8350732884
2347	196527.07006407416
2348	243588.3065862579
2349	177694.849951581
2350	306972.8696919537
2351	261726.74314215028
2352	258711.50256126004
2353	252755.50810406732

2354	136653.40566702056
2355	144450.80546247773
2356	154082.35238182027
2357	188657.12475253554
2358	199253.57789127823
2359	131354.44498799878
2360	116797.57143258712
2361	143245.90745813044
2362	274010.6426747854
2363	138033.292699882
2364	158249.75417864101
2365	216010.48153966095
2366	187514.09380502847
2367	223335.74355548059
2368	214091.39795957215
2369	216372.37094577728
2370	168596.34954614704
2371	169624.72052330745
2372	194236.02197066366
2373	283248.8205444707
2374	316576.7134008888
2375	248444.04617526266
2376	282026.9067129346
2377	348348.1372405767
2378	144134.81834888598
2379	196177.53218181705
2380	149112.62613435645
2381	166892.3328418349
2382	201243.71817042338
2383	205662.2378528366
2384	248520.2326914032
2385	159510.16540963564

2386	133776.5361943018
2387	136048.26288192926
2388	100921.34187198993
2389	120503.21781084074
2390	148353.6163138746
2391	146366.52154333566
2392	113913.95372115095
2393	163330.71016046955
2394	139965.7188356997
2395	212829.7150150603
2396	142689.26585561174
2397	224974.16541755316
2398	131869.66714601842
2399	62634.22762375745
2400	57474.80653640703
2401	116352.04997655946
2402	138235.4321079489
2403	139290.87954910047
2404	149080.0668257807
2405	162804.30871507814
2406	142141.45103139497
2407	126109.9576547526
2408	144214.52859978075
2409	120106.93299213819
2410	167587.00282483286
2411	116743.31646149034
2412	158333.34824768425
2413	129974.29405278942
2414	148215.03410349129
2415	130580.40973876021
2416	124987.38585751494
2417	132252.2296585875

2418	130241.99590294292
2419	129259.47701628869
2420	123965.301318371
2421	150696.6102078281
2422	106447.14512739459
2423	125148.1177076738
2424	154170.57822385212
2425	205901.40074131108
2426	136165.73919747584
2427	136155.5355030867
2428	172872.65137954277
2429	112651.28831885055
2430	133099.70449288381
2431	109186.14665152157
2432	145917.9768469977
2433	147696.6552229186
2434	139475.63751653596
2435	153593.44652255508
2436	105850.30468240204
2437	107077.2056730302
2438	122515.96734404302
2439	106360.34017516796
2440	120437.73309555434
2441	94820.29160955358
2442	95053.5731170511
2443	125860.51016094761
2444	129175.69817981185
2445	90466.68911792361
2446	135659.28167955673
2447	178643.6007048312
2448	129752.05223097876
2449	112934.14168602375

2450	151790.96597524008
2451	121392.7563025332
2452	194604.48432376375
2453	92974.63388195919
2454	125687.33142713137
2455	123515.60696575564
2456	141621.56134226546
2457	135774.52225612532
2458	127409.4484407412
2459	112258.14996259483
2460	137251.59333124445
2461	123463.2247778297
2462	137907.02933719935
2463	126513.49099436327
2464	178703.96581229675
2465	144828.80336045494
2466	119607.35453562358
2467	129114.88199123171
2468	78989.24013402694
2469	77843.39171507324
2470	190624.10667163495
2471	212048.85401814993
2472	159345.8499360666
2473	116481.18217754255
2474	77206.16834553558
2475	216061.34497340373
2476	115569.86500393058
2477	128841.7822160906
2478	154687.98849544997
2479	103272.19943543703
2480	154826.44815049903
2481	121775.98966769928

2482	127106.00082580849
2483	110108.64944558626
2484	124418.99283234334
2485	124994.69504084194
2486	155424.96488577058
2487	181103.47684895122
2488	157493.23324633826
2489	151520.54955940298
2490	146939.21558890172
2491	93952.61646530978
2492	190338.5685296473
2493	151217.56581272514
2494	153390.14218680098
2495	91566.9289744308
2496	247138.01489782726
2497	154669.42425283534
2498	108529.55160816187
2499	90401.6002096101
2500	125415.35740362301
2501	140624.98341943108
2502	149108.77424214265
2503	95992.8149824097
2504	189082.3991648595
2505	226331.5841347258
2506	254085.0389303907
2507	289181.40129507054
2508	254921.78649170254
2509	215558.29130443343
2510	216520.81482070428
2511	175270.02031409796
2512	205791.44236486207
2513	220608.19322155183

2514	263348.0788404307
2515	151498.7076571131
2516	174226.41974464757
2517	144641.11763208284
2518	153133.4087486668
2519	224999.81342796492
2520	215082.50279351103
2521	189240.9985848729
2522	225132.76707487938
2523	121907.49489557793
2524	139088.75807044425
2525	151484.55811717492
2526	135118.27183897
2527	114486.18940621799
2528	129898.52641674383
2529	129748.60795731205
2530	120640.61940202891
2531	251738.7470363337
2532	225360.5655905741
2533	195932.8698865202
2534	223111.75454249786
2535	297434.0305646339
2536	236395.31910447677
2537	230434.36822492807
2538	180969.34981249383
2539	187116.44479693865
2540	176692.74930113947
2541	180304.07242069693
2542	166780.6476595597
2543	124981.01118279433
2544	124309.66561863257
2545	134859.96131811355

2546	128673.67446828159
2547	139731.1528172606
2548	151694.9823936455
2549	159087.22196839112
2550	698606.0321839529
2551	137252.25052913223
2552	132718.8001770365
2553	76564.2544921579
2554	99380.54398179478
2555	116938.87332674887
2556	99942.8774523212
2557	105250.48948579669
2558	151355.89679667816
2559	140478.2037407655
2560	148249.97276976603
2561	138711.56721715102
2562	135637.1193633096
2563	149243.07397838164
2564	202891.911116314
2565	141781.75156141171
2566	159736.34888815664
2567	133910.60869952204
2568	198400.7114672772
2569	225782.7521487361
2570	116793.21286608998
2571	188919.48052531644
2572	147527.79349648545
2573	224059.0793615114
2574	289027.0294773341
2575	131154.03234865144
2576	122587.24287321858
2577	150609.90916145116

2578	84823.92576605603
2579	62139.65621664139
2580	107679.53007998425
2581	109039.85246447414
2582	107862.02182473321
2583	282441.5380572454
2584	174663.645125571
2585	188881.32068348167
2586	224781.1979277313
2587	195387.4056236206
2588	140827.37445541617
2589	152702.25938177016
2590	204089.14623371235
2591	219205.33349306835
2592	214448.24684307625
2593	255332.55902671593
2594	180711.4492129416
2595	201760.05411732962
2596	297891.53155732766
2597	187973.2896073205
2598	276382.8966762186
2599	329276.4695311454
2600	150596.31104407072
2601	152305.86395956838
2602	82944.48928104388
2603	93146.3770415716
2604	88063.63005002546
2605	77962.25670046394
2606	151014.73179450867
2607	195334.15583638754
2608	210195.632673903
2609	169429.11803220012

2610	112007.29945987399
2611	120935.1495376422
2612	158129.00421090832
2613	130126.47639090584
2614	123513.79018741957
2615	154356.63817584026
2616	145700.9899661392
2617	184064.60204332042
2618	202858.21909437567
2619	194255.1304071029
2620	182613.63305409744
2621	175746.3715910611
2622	183493.60480764174
2623	244723.286603599
2624	308418.6789643431
2625	301089.3871787067
2626	171730.82239096874
2627	165166.5598368732
2628	502673.0223724861
2629	537975.4875517547
2630	360188.36806582107
2631	516891.18558109144
2632	439474.00291186513
2633	302576.46197838796
2634	432841.684241424
2635	160127.61253241202
2636	173549.32078930276
2637	176686.83369709807
2638	264075.65876487875
2639	188173.41368914908
2640	150634.1967095219
2641	102098.36596747124

2642	195372.57044489367
2643	100425.336500853
2644	122426.38472141593
2645	107299.30565813542
2646	93875.57046628284
2647	101234.04189778876
2648	143662.96419881718
2649	150920.35434563697
2650	136298.28535100684
2651	138943.2215391899
2652	408687.25183825195
2653	247514.95733808726
2654	256412.86280615936
2655	402650.06945174775
2656	317977.4785405834
2657	350034.33329115284
2658	302202.6821460173
2659	294810.4356021989
2660	367551.19511091395
2661	342826.7470995404
2662	365941.7783256748
2663	281191.20058057766
2664	274071.09315087175
2665	321318.53800324525
2666	285533.0778139804
2667	168219.0299691817
2668	178009.70529938792
2669	183281.86026774446
2670	272450.50299710827
2671	184755.09802766403
2672	192630.29988399934
2673	202431.35428363303

2674	196616.10618939812
2675	164835.849445914
2676	191076.20668537743
2677	196436.5168519605
2678	253463.65793203158
2679	274970.8689216876
2680	283518.8428803413
2681	410038.18617870484
2682	325783.69901251304
2683	542371.4044495545
2684	313788.1376317515
2685	331611.2413936247
2686	252707.295273361
2687	327736.6888946597
2688	211443.02643780378
2689	203905.43688118717
2690	468048.11675731547
2691	191307.26392703975
2692	136651.53221381165
2693	201244.3639910182
2694	136988.96846181471
2695	193330.12135202455
2696	184151.12692442664
2697	189217.3425506804
2698	193462.5627917882
2699	176016.98600550884
2700	157737.35922884935
2701	152056.5654311397
2702	110285.44300462812
2703	124802.14813562084
2704	141433.0052936063
2705	118566.79783773632

2706	113046.56096076187
2707	124295.65924973317
2708	133654.9937050043
2709	107238.13798730177
2710	126108.70739804277
2711	282489.4474468561
2712	384138.61433257425
2713	170205.0505758892
2714	150900.59151951785
2715	172943.86473474026
2716	140284.8379884236
2717	186342.12491810345
2718	220528.5055162126
2719	157696.57619196328
2720	169273.24279228333
2721	133548.30925487942
2722	160714.48911233412
2723	143674.5173603066
2724	126251.9250180629
2725	131329.19715794484
2726	152528.99237966796
2727	176573.26661355078
2728	177397.31644074284
2729	151551.6916974652
2730	149731.84607211803
2731	131752.45137084316
2732	134184.46416673524
2733	161766.7249828537
2734	144755.22953663388
2735	137740.64896027226
2736	140014.94235098152
2737	123270.81541397437

2738	134668.00543577533
2739	162278.71696505864
2740	135507.26630329542
2741	149290.02744622857
2742	151341.42154774582
2743	155067.74680880576
2744	159990.8550092539
2745	139555.26224814806
2746	136763.2499083937
2747	140166.16955058448
2748	127004.0287059079
2749	130726.0000141353
2750	134372.81183991468
2751	135560.35340919183
2752	194606.4563138682
2753	146325.41660936197
2754	206855.0049977292
2755	131205.42084018447
2756	92384.06757243701
2757	66369.29970415353
2758	88141.13175707411
2759	160234.19328157784
2760	125536.76796210132
2761	145000.34384386695
2762	143234.91214062294
2763	183371.6723935558
2764	142021.4121351444
2765	273014.9326890318
2766	135182.5476050181
2767	89182.25023991
2768	132231.66729472973
2769	133546.0400449248

2770	134946.78001534424
2771	108781.15880090816
2772	112346.97238637254
2773	173036.01235182528
2774	145269.17636991423
2775	117471.58743487054
2776	138859.70934834454
2777	140213.01664083017
2778	98001.33411030336
2779	127419.5012452356
2780	94667.52367126032
2781	95917.30414173279
2782	93226.89390196407
2783	97860.88445732245
2784	120727.83200247986
2785	140610.74409047514
2786	66310.44174243578
2787	127106.2233578293
2788	75422.6868981501
2789	169986.83488739934
2790	93852.8233400351
2791	110017.64041402627
2792	61873.644707908636
2793	165484.18941059077
2794	98615.81101823469
2795	125454.10746262231
2796	98770.86422119243
2797	201984.28367658926
2798	110355.63534342265
2799	118064.1805852887
2800	63399.552163906264
2801	110260.57152572757

2802	130490.89542721538
2803	166811.7071621993
2804	132585.9438662991
2805	101427.1996027277
2806	83456.2795645479
2807	155219.8688320256
2808	149909.19633266662
2809	136140.78217790593
2810	129060.69010445154
2811	162126.59578383664
2812	159540.80672520134
2813	155110.27118887307
2814	152852.5481794933
2815	104397.22492051257
2816	243199.71200090123
2817	156029.02091666896
2818	132179.68694793925
2819	161739.15750195086
2820	138354.82152739435
2821	110599.07547681902
2822	199101.59558567
2823	323676.1499995346
2824	183441.60104619828
2825	149226.98749036776
2826	132514.18556085485
2827	139361.0226073579
2828	221974.4901630918
2829	219326.13330355415
2830	229705.253495364
2831	175690.32148854923
2832	261726.98859486263
2833	297155.34174272005

2834	212573.4291967905
2835	214783.99401676055
2836	191699.05798484996
2837	153238.52123832013
2838	149695.59290170064
2839	176082.6964894755
2840	199599.12992960413
2841	208144.55778221635
2842	233334.0831131152
2843	149947.27054194416
2844	166868.5552572579
2845	124174.909641497
2846	212002.90165124586
2847	206372.85855872667
2848	218077.771078615
2849	198276.9781001503
2850	280607.2663649125
2851	219458.987498267
2852	234266.01536461333
2853	240485.46311713528
2854	140719.43213981192
2855	204028.45139768493
2856	199285.51887044575
2857	187185.39875130504
2858	203089.63257898664
2859	116098.74571337362
2860	126942.3263096752
2861	124452.6278051522
2862	194152.15590131684
2863	126518.18961578263
2864	251697.2885182413
2865	136993.61177203967

2866	138576.8143974102
2867	99581.92741388606
2868	96960.70844536272
2869	106447.85315140687
2870	121811.37311660184
2871	80128.85752588013
2872	50152.75596088961
2873	95452.82840503476
2874	147101.97298343998
2875	115599.95939298178
2876	161971.65217648412
2877	145797.76937814846
2878	161699.15893298472
2879	143588.07721748148
2880	102862.69058747755
2881	147606.78312567418
2882	177424.3310125217
2883	194458.81548115012
2884	195129.76391440566
2885	190928.76261118287
2886	232379.75430361176
2887	95204.20208476721
2888	139847.00017507345
2889	56093.54586475422
2890	81040.3677118451
2891	134371.0154835643
2892	49940.58478384673
2893	81858.7097540065
2894	57563.15197749663
2895	340493.2638502061
2896	282124.7970521053
2897	204632.65734364474

2898	145493.70261053453
2899	214048.6708342525
2900	160231.38112930593
2901	210447.66727496966
2902	191119.53327536143
2903	338559.2079332213
2904	353268.54828489525
2905	88781.87775913162
2906	193524.3172887688
2907	110210.12062874567
2908	130210.42827240986
2909	147814.53244141935
2910	78296.06038975506
2911	80224.43916186158
2912	145193.42772731764
2913	83756.09181029258
2914	76838.48248660585
2915	85042.9772302422
2916	83496.30083336026
2917	166785.94041732245
2918	118384.09837015715
2919	221444.86998688208