

```
In [43]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
```

```
In [44]: housing_data = pd.read_excel('AmesHousing.xls')
housing_data.head()
```

Out[44]:

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	...	Pc Ar
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	...	
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	...	
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	...	
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	...	
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	...	

5 rows × 82 columns

```
In [45]: housing_data.dtypes
```

```
Out[45]: Order           int64
          PID            int64
          MS SubClass     int64
          MS Zoning       object
          Lot Frontage    float64
          Lot Area         int64
          Street          object
          Alley           object
          Lot Shape        object
          Land Contour    object
          Utilities        object
          Lot Config       object
          Land Slope       object
          Neighborhood    object
          Condition 1     object
          Condition 2     object
          Bldg Type        object
          House Style      object
          Overall Qual    int64
          Overall Cond    int64
          Year Built       int64
          Year Remod/Add  int64
          Roof Style       object
          Roof Matl        object
          Exterior 1st     object
          Exterior 2nd     object
          Mas Vnr Type    object
          Mas Vnr Area    float64
          Exter Qual       object
          Exter Cond       object
          ...
          Bedroom AbvGr   int64
          Kitchen AbvGr   int64
          Kitchen Qual    object
          TotRms AbvGrd   int64
          Functional       object
          Fireplaces       int64
          Fireplace Qu    object
          Garage Type      object
          Garage Yr Blt   float64
          Garage Finish    object
          Garage Cars      float64
          Garage Area      float64
          Garage Qual      object
          Garage Cond      object
          Paved Drive      object
          Wood Deck SF     int64
          Open Porch SF    int64
          Enclosed Porch   int64
          3Ssn Porch       int64
          Screen Porch     int64
          Pool Area        int64
          Pool QC          object
          Fence            object
          Misc Feature     object
          Misc Val          int64
          Mo Sold          int64
```

```
Yr Sold           int64
Sale Type         object
Sale Condition   object
SalePrice        int64
Length: 82, dtype: object
```

```
In [46]: housing_data.drop(columns=[ 'Order' , 'PID' ], inplace=True)
```

In [47]: `housing_data.dtypes`

Out[47]:

MS SubClass	int64
MS Zoning	object
Lot Frontage	float64
Lot Area	int64
Street	object
Alley	object
Lot Shape	object
Land Contour	object
Utilities	object
Lot Config	object
Land Slope	object
Neighborhood	object
Condition 1	object
Condition 2	object
Bldg Type	object
House Style	object
Overall Qual	int64
Overall Cond	int64
Year Built	int64
Year Remod/Add	int64
Roof Style	object
Roof Matl	object
Exterior 1st	object
Exterior 2nd	object
Mas Vnr Type	object
Mas Vnr Area	float64
Exter Qual	object
Exter Cond	object
Foundation	object
Bsmt Qual	object
...	
Bedroom AbvGr	int64
Kitchen AbvGr	int64
Kitchen Qual	object
TotRms AbvGrd	int64
Functional	object
Fireplaces	int64
Fireplace Qu	object
Garage Type	object
Garage Yr Blt	float64
Garage Finish	object
Garage Cars	float64
Garage Area	float64
Garage Qual	object
Garage Cond	object
Paved Drive	object
Wood Deck SF	int64
Open Porch SF	int64
Enclosed Porch	int64
3Ssn Porch	int64
Screen Porch	int64
Pool Area	int64
Pool QC	object
Fence	object
Misc Feature	object
Misc Val	int64
Mo Sold	int64

```
Yr Sold           int64
Sale Type         object
Sale Condition   object
SalePrice        int64
Length: 80, dtype: object
```

```
In [48]: import warnings
warnings.filterwarnings("ignore")
housing_continuous = housing_data.select_dtypes (include = ['int64','float64'])
#housing_continuous.drop(columns = ['MS SubClass', 'Year Built', 'Year Remod/Add', 'Mo Sold', 'Yr Sold'],inplace = True)
housing_continuous.drop(columns = ['MS SubClass', 'Mo Sold'],inplace = True)
#choosing continuous columns based on d-type but dropping some of them based on looking at the description
#Continuing to keep year related columns as continuous to reduce dimensionality
```

```
In [49]: housing_continuous.shape
```

```
Out[49]: (2930, 35)
```

```
In [50]: cont_name = list(housing_continuous)
```

```
In [51]: cat_name = list(set(housing_data.columns)- set(housing_continuous.columns))
```

```
In [52]: len(cat_name)
```

```
Out[52]: 45
```

```
In [53]: cat_name
```

```
Out[53]: ['Central Air',
 'Bsmt Qual',
 'Paved Drive',
 'Garage Qual',
 'Bsmt Cond',
 'Garage Type',
 'Land Contour',
 'Street',
 'House Style',
 'Heating QC',
 'Fence',
 'Heating',
 'Mo Sold',
 'Roof Matl',
 'Condition 1',
 'Exter Cond',
 'Neighborhood',
 'Alley',
 'Mas Vnr Type',
 'MS Zoning',
 'Lot Config',
 'Misc Feature',
 'Exterior 1st',
 'BsmtFin Type 1',
 'Exterior 2nd',
 'Garage Cond',
 'Utilities',
 'Functional',
 'Roof Style',
 'Exter Qual',
 'Sale Type',
 'Fireplace Qu',
 'Sale Condition',
 'Garage Finish',
 'Condition 2',
 'Electrical',
 'Pool QC',
 'Bsmt Exposure',
 'BsmtFin Type 2',
 'Bldg Type',
 'Foundation',
 'Lot Shape',
 'MS SubClass',
 'Kitchen Qual',
 'Land Slope']
```

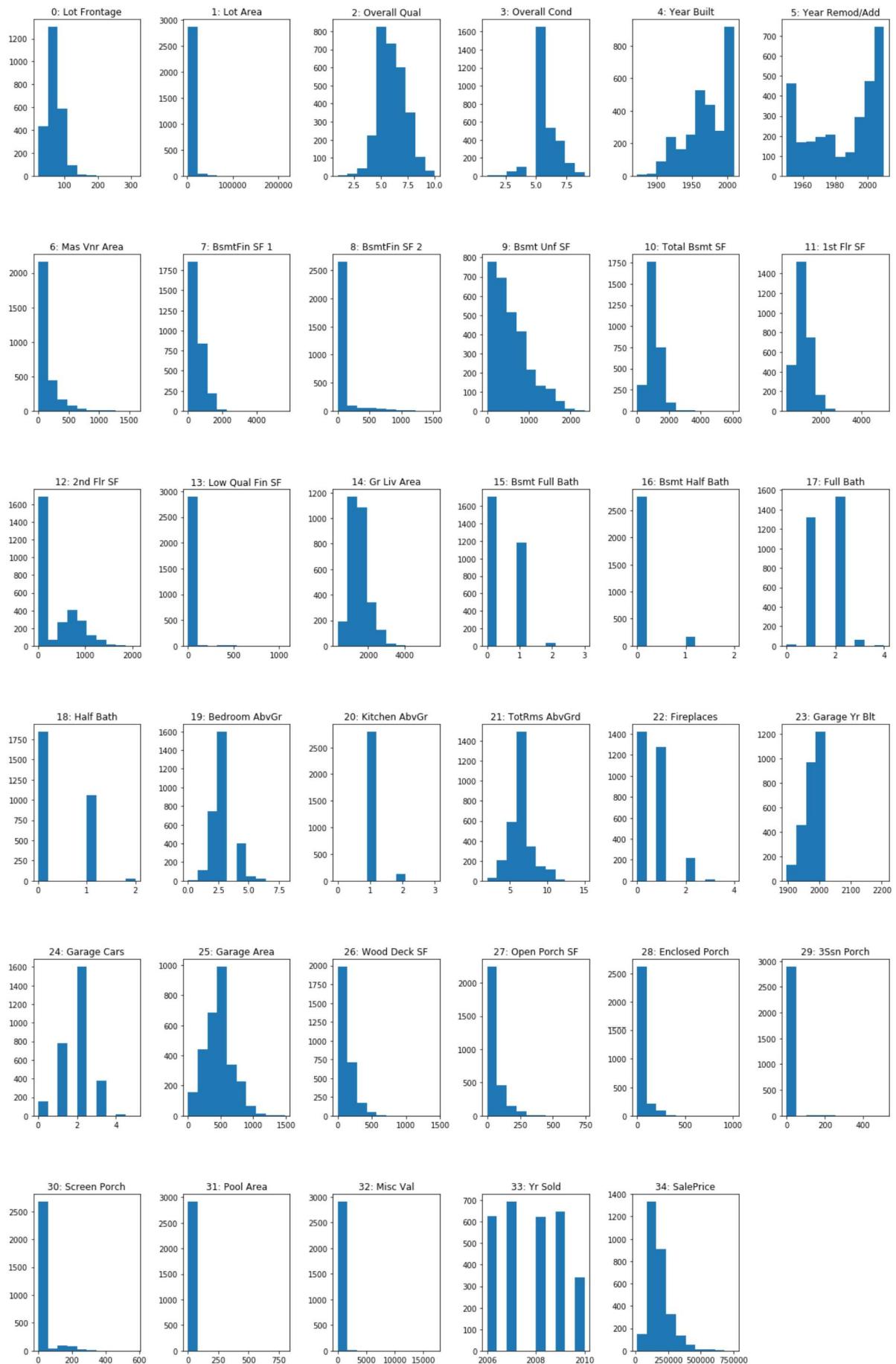
```
In [54]: #housing_categorical = housing_data.select_dtypes (include = ['object'])
#housing_categorical['MS SubClass'] = housing_data['MS SubClass']
```

1.1

```
In [55]: fig, axes = plt.subplots(6,6,figsize = (20,32))
plt.subplots_adjust(wspace=0.40,hspace=0.50)

for i, ax in enumerate(axes.ravel()):
    if i > 34:
        ax.set_visible(False)
        continue
    ax.hist(housing_continuous.iloc[:,i])
    ax.set_title("{}: {}".format(i, cont_name[i]))
```

Task 1



- We can see that all variables are not continuous - some of the histogram distribution have a lot of spaces in between them indicating that they are discrete (ex: Full bath, Garage Cars etc.)

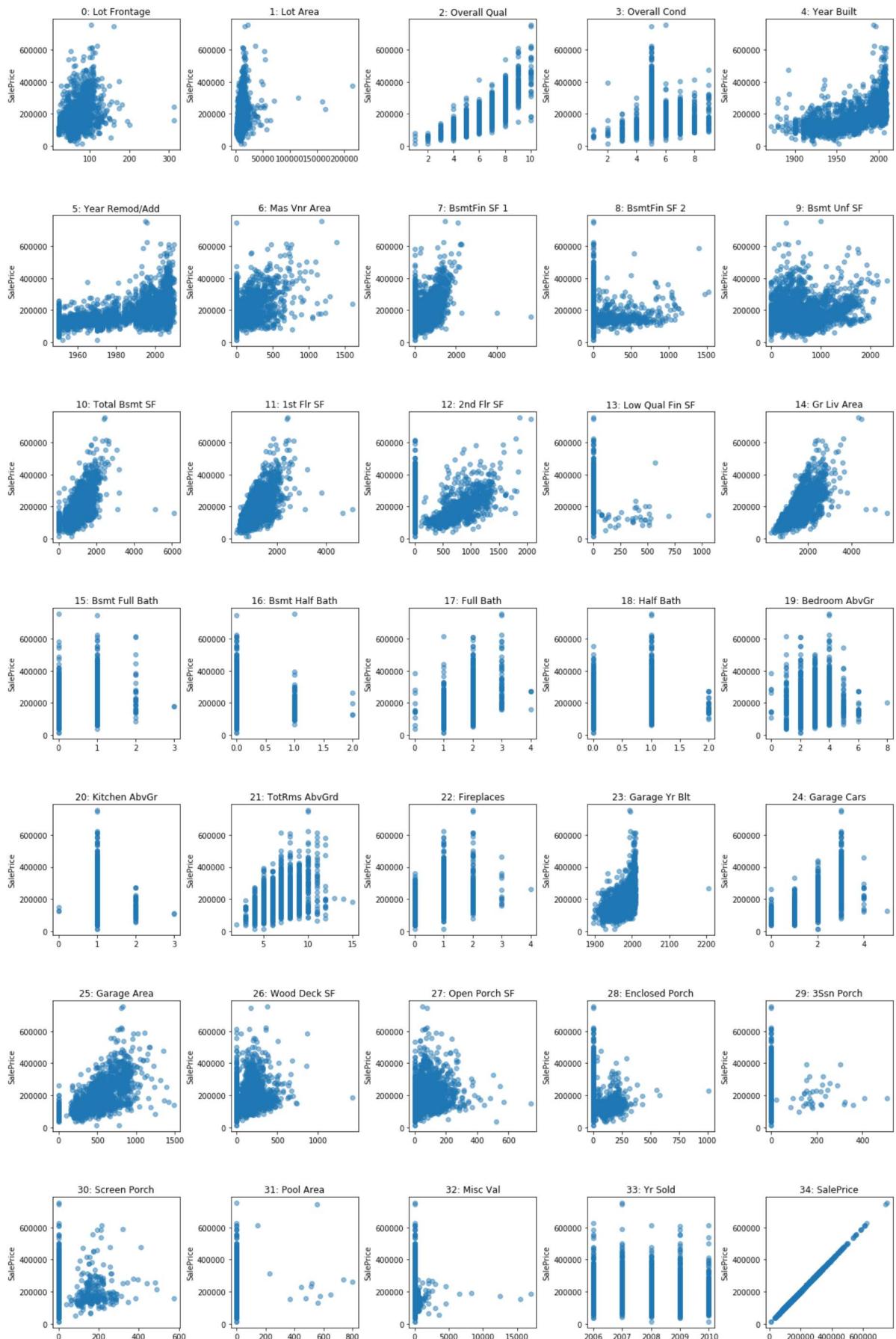
*A handful variables seem to have outliers and that is why we see that the prominent histograms are either to the left/ right along with a few smaller ones distributed across (ex: BsmtFin SF 2 etc.)

1.2

```
In [56]: fig, axes = plt.subplots(7, 5, figsize=(20, 32))
plt.subplots_adjust(wspace=0.40, hspace=0.50)

for i, ax in enumerate(axes.ravel()):
    if i > 34:
        ax.set_visible(False)
        continue
    ax.plot(housing_continuous.iloc[:, i], housing_continuous['SalePrice'],
    'o', alpha=.5)
    ax.set_title("{}: {}".format(i, cont_name[i]))
    ax.set_ylabel("SalePrice")
```

Task 1



In [57]: `housing_data[cat_name] = housing_data[cat_name].replace({np.nan: 'NA'})`
#Taking nan values as NA, a separate category

1.3

```
In [58]: from sklearn.model_selection import train_test_split
cont_name.remove('SalePrice')
X_train, X_test, y_train, y_test = train_test_split (housing_data.loc[:, housing_data.columns != 'SalePrice'], housing_data['SalePrice'], random_state = 0)
X_train = pd.concat((X_train[cat_name], X_train[cont_name]), axis=1)
```

```
In [59]: from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
cat_score = []

for i in cat_name:
    scores = cross_val_score(LinearRegression(), pd.get_dummies(X_train[i]), y_train, cv=5)
    cat_score.append(np.mean(scores))

print ('The top 5 R^2 values are')

for i in range(0,5):

    print(cat_name[np.argsort(cat_score)[-5:][i]], cat_score[np.argsort(cat_score)[-5:][i]])
```

The top 5 R² values are
Garage Type 0.23464024676523776
Foundation 0.27072441117959734
Garage Finish 0.30005099379138944
Fireplace Qu 0.31984124628027705
Exter Qual 0.5219081154566937

In [60]: #Plotting the top 4 values from the previous section

```
import seaborn as sns
train_set = pd.concat([X_train, y_train], axis=1)
fig, ax = plt.subplots(2,2, figsize=(16,16))

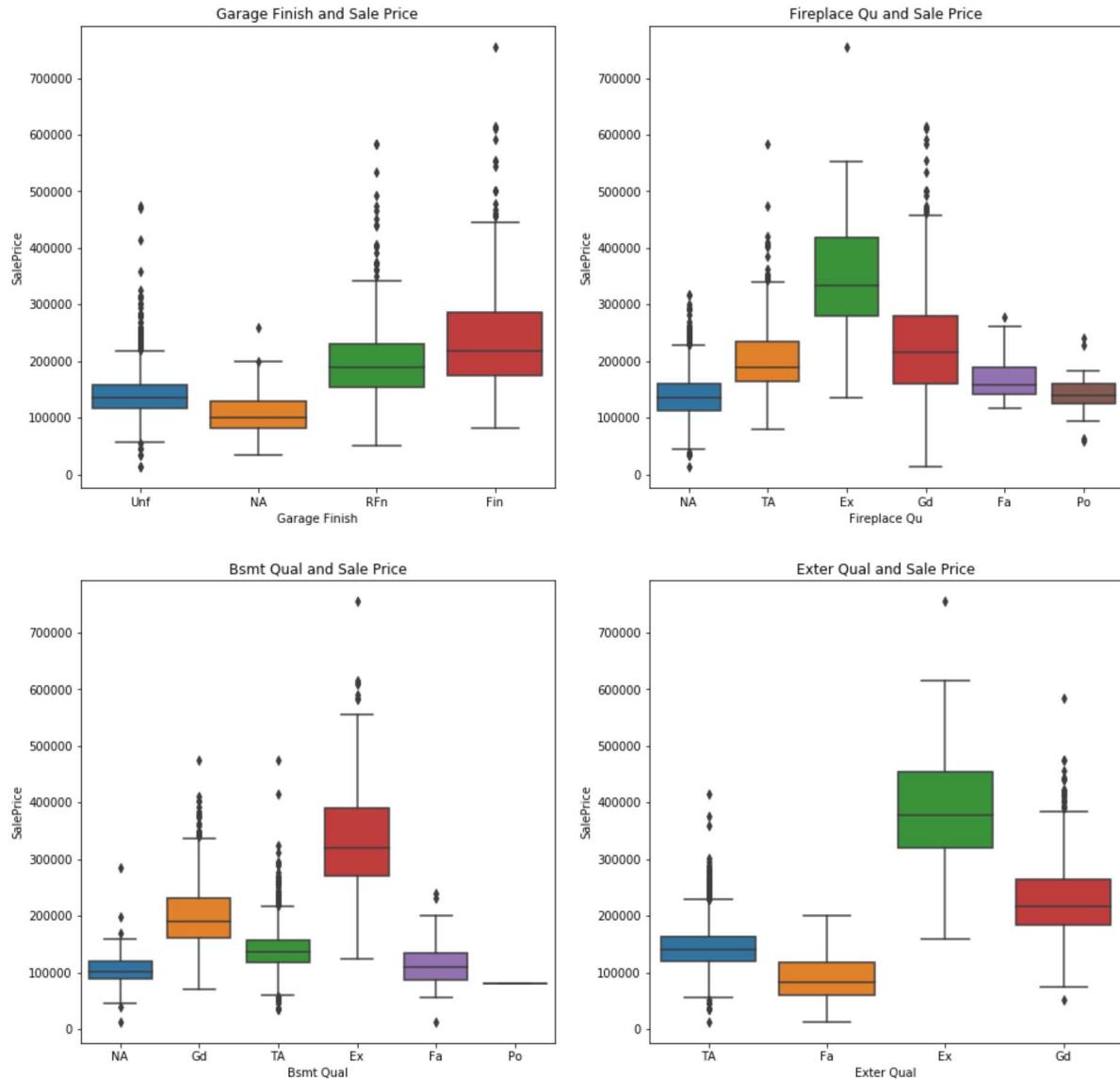
_ = sns.boxplot(x='Garage Finish', y='SalePrice', data= train_set, ax=ax[0][0])
ax[0,0].set_title('Garage Finish and Sale Price')

_ = sns.boxplot(x='Fireplace Qu', y='SalePrice', data= train_set, ax=ax[0][1])
ax[0,1].set_title('Fireplace Qu and Sale Price')

_ = sns.boxplot(x='Bsmt Qual', y='SalePrice', data= train_set, ax=ax[1][0])
ax[1,0].set_title('Bsmt Qual and Sale Price')

_ = sns.boxplot(x='Exter Qual', y='SalePrice', data= train_set, ax=ax[1][1])
ax[1,1].set_title('Exter Qual and Sale Price')
```

Out[60]: Text(0.5, 1.0, 'Exter Qual and Sale Price')



In [61]: X_train

Out[61]:

	Central Air	Bsmt Qual	Paved Drive	Garage Qual	Bsmt Cond	Garage Type	Land Contour	Street	House Style	Heating QC	...	Garage C
896	N	NA	N	TA	NA	Detchd	Lvl	Pave	1.5Fin	TA	...	
1901	N	NA	N	NA	NA	NA	Low	Pave	1Story	Fa	...	
390	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	TA	...	
1293	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1.5Fin	Gd	...	
1767	Y	Ex	Y	TA	TA	Attchd	Lvl	Pave	2Story	Ex	...	
251	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
2847	Y	Fa	Y	Fa	TA	Detchd	Lvl	Pave	1.5Fin	TA	...	
357	Y	Gd	Y	TA	Gd	BuiltIn	Lvl	Pave	2Story	Gd	...	
1396	Y	TA	N	TA	TA	2Types	Lvl	Pave	1.5Fin	Gd	...	
852	Y	Gd	Y	NA	TA	NA	HLS	Pave	1Story	TA	...	
841	Y	Gd	Y	TA	Gd	Detchd	Lvl	Pave	1Story	Ex	...	
1756	Y	Gd	Y	TA	TA	BuiltIn	Lvl	Pave	SLvl	Ex	...	
2472	Y	Gd	Y	NA	TA	NA	Lvl	Pave	1Story	Ex	...	
528	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
232	Y	Gd	Y	TA	TA	Attchd	Low	Pave	1Story	TA	...	
610	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	SLvl	TA	...	
1380	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
2074	Y	TA	Y	TA	Gd	Detchd	Lvl	Pave	1Story	Ex	...	
1043	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	2Story	TA	...	
1374	Y	NA	N	TA	NA	Detchd	Lvl	Pave	1.5Fin	TA	...	
653	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1Story	Ex	...	
1387	Y	TA	Y	NA	TA	NA	Lvl	Pave	1Story	TA	...	
1703	Y	Ex	Y	TA	TA	BuiltIn	Lvl	Pave	2Story	Ex	...	
676	N	NA	Y	TA	NA	Detchd	Lvl	Pave	1Story	Fa	...	
1174	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	2Story	Ex	...	
969	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Gd	...	
1042	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	2Story	TA	...	
766	N	TA	Y	TA	TA	Detchd	Lvl	Pave	2Story	TA	...	
1118	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	2Story	Ex	...	
2106	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
...
2496	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	TA	...	
1871	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Gd	...	
2046	N	TA	P	TA	TA	Detchd	Lvl	Pave	1.5Fin	Fa	...	

	Central Air	Bsmt Qual	Paved Drive	Garage Qual	Bsmt Cond	Garage Type	Land Contour	Street	House Style	Heating QC	...	Gar C
755	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1.5Fin	TA	...	
976	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
2163	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
1940	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Gd	...	
2825	Y	Gd	Y	TA	TA	Basment	Lvl	Pave	SLvl	Gd	...	
2120	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Gd	...	
2893	Y	Gd	Y	TA	TA	Attchd	Low	Pave	1Story	TA	...	
537	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
1701	Y	Ex	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
2897	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	2Story	Ex	...	
2222	Y	Gd	Y	TA	TA	Basment	Bnk	Pave	1Story	TA	...	
2135	Y	Gd	Y	TA	TA	Detchd	Lvl	Pave	SLvl	TA	...	
2599	Y	Fa	Y	Fa	Fa	Attchd	Lvl	Pave	1.5Fin	Ex	...	
705	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1.5Fin	TA	...	
2362	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1Story	TA	...	
2648	Y	TA	Y	TA	TA	Detchd	Lvl	Pave	1Story	Ex	...	
2647	Y	Fa	N	Fa	Fa	Detchd	Lvl	Pave	2Story	TA	...	
1828	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	2Story	TA	...	
1778	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	
277	N	TA	P	TA	TA	Detchd	Lvl	Pave	1.5Fin	Ex	...	
1033	Y	Gd	Y	TA	TA	Detchd	Lvl	Pave	2Story	TA	...	
1731	Y	Gd	Y	TA	TA	Attchd	Lvl	Pave	2Story	Ex	...	
763	Y	Gd	Y	NA	Gd	NA	Lvl	Pave	1Story	TA	...	
835	Y	Gd	Y	TA	TA	Detchd	Low	Pave	1Story	Gd	...	
1653	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	SLvl	TA	...	
2607	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	SLvl	TA	...	
2732	Y	TA	Y	TA	TA	Attchd	Lvl	Pave	1Story	Ex	...	

2197 rows × 79 columns

1.4

#1.4 - a

```
In [62]: from sklearn.compose import make_column_transformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder

preprocess_imputer = ColumnTransformer([('ContImpute', SimpleImputer(strategy='median'), cont_name),
                                         ('Ohe', OneHotEncoder(handle_unknown='ignore'), cat_name)], remainder='passthrough')
```

```
In [63]: import warnings
warnings.filterwarnings("ignore")

linreg_model = make_pipeline(preprocess_imputer, LinearRegression())
linreg_scores = cross_val_score(linreg_model, X_train, y_train, cv=5)

lasso_model = make_pipeline (preprocess_imputer, Lasso())
lasso_scores = cross_val_score (lasso_model, X_train, y_train, cv=5)

ridge_model = make_pipeline (preprocess_imputer, Ridge())
ridge_scores = cross_val_score (ridge_model, X_train, y_train, cv=5)

elastic_model = make_pipeline(preprocess_imputer, ElasticNet())
elastic_scores = cross_val_score(elastic_model, X_train, y_train, cv=5)

print('Linear Regression Initial Score ', np.mean(linreg_scores))
print ('Lasso Regression Initial Score ', np.mean(lasso_scores))
print ('Ridge Regression Initial Score ', np.mean(ridge_scores))
print('Elastic Net Initial Score ', np.mean(elastic_scores))

Linear Regression Initial Score  0.8410318378923524
Lasso Regression Initial Score  0.8735966862603408
Ridge Regression Initial Score  0.6964829199408656
Elastic Net Initial Score  0.8437582497513526
```

#1.4 - b (Scaling included)

```
In [64]: from sklearn.preprocessing import StandardScaler

scaling = ColumnTransformer([('scaling', StandardScaler(), cont_name)], remainder='passthrough')

preprocess_scaling = ColumnTransformer([('ContImpute', SimpleImputer(strategy='median'), np.arange(0,len(cont_name))),
                                         ('Ohe', OneHotEncoder(handle_unknown='ignore'), np.arange(len(cont_name),len(X_train.columns)))]], remainder='passthrough')
```

```
In [65]: linreg_model_scaling = make_pipeline(scaling, preprocess_scaling, LinearRegression())
linreg_scores_scaling = cross_val_score(linreg_model_scaling, X_train, y_train, cv=5)

lasso_model_scaling = make_pipeline(scaling, preprocess_scaling, Lasso())
lasso_scores_scaling = cross_val_score(lasso_model_scaling, X_train, y_train, cv=5)

ridge_model_scaling = make_pipeline(scaling, preprocess_scaling, Ridge())
ridge_scores_scaling = cross_val_score(ridge_model_scaling, X_train, y_train, cv=5)

elastic_model_scaling = make_pipeline(preprocess_imputer, ElasticNet())
elastic_scores_scaling = cross_val_score(elastic_model, X_train, y_train, cv=5)

print('Linear Regression (Scaling) ', np.mean(linreg_scores_scaling))
print('Lasso Regression (Scaling) ', np.mean(lasso_scores_scaling))
print('Ridge Regression (Scaling) ', np.mean(ridge_scores_scaling))
print('Elastic Net (Scaling) ', np.mean(elastic_scores_scaling))
```

```
Linear Regression (Scaling)  0.8348515787662137
Lasso Regression (Scaling)  0.8736450936390707
Ridge Regression (Scaling)  0.8707794944172409
Elastic Net (Scaling)  0.8437582497513526
```

Scaling definitely helps in the case of ridge regression as we see the score move from 0.68 to 0.86 on scaling. The impact is visible but not huge for the other models but it might be due to the fact that they show a ~85% score which is high initially itself.

1.5

```
In [66]: from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
```

```
In [68]: #paragrid_linreg = {'linreg_fit_intercept':[True, False], 'linreg_normalize':[True, False], 'linreg_copy_X':[True, False]}
paragrid_lasso = {'lasso_alpha': np.logspace(-5, 3, 20)}
paragrid_ridge = {'ridge_alpha': np.logspace(-3, 3, 13)}
paragrid_elastic = {'elastic_alpha': np.logspace(-3, 3, 13), 'elastic_l1_ratio':[0.01, .1, .4, .7, .9, 0.95, 1]}

#pipeline_linreg = Pipeline(steps=[('scaling',scaling), ('preprocess', preprocess_scaling), ('linreg', LinearRegression())])
pipeline_lasso = Pipeline(steps=[('scaling',scaling), ('preprocess', preprocess_scaling), ('lasso', Lasso())])
pipeline_ridge = Pipeline(steps=[('scaling',scaling), ('preprocess', preprocess_scaling), ('ridge', Ridge())])
pipeline_elastic = Pipeline(steps=[('scaling',scaling), ('preprocess', preprocess_scaling), ('elastic', ElasticNet())])

#gcv_linreg = GridSearchCV(pipeline_linreg, param_grid=paragrid_linreg, cv=5, return_train_score=True)
#gcv_linreg.fit(X_train, y_train)

gcv_lasso = GridSearchCV(pipeline_lasso, param_grid= paragrid_lasso, cv=5, return_train_score=True)
gcv_lasso.fit(X_train, y_train)

gcv_ridge = GridSearchCV(pipeline_ridge, param_grid=paragrid_ridge, cv=5, return_train_score=True)
gcv_ridge.fit(X_train, y_train)

gcv_elastic = GridSearchCV(pipeline_elastic, param_grid=paragrid_elastic, cv=5, return_train_score=True)
gcv_elastic.fit(X_train, y_train)

#print("Linear regression best parameter is".format(gcv_linreg.best_params_))
print("Lasso best parameter is {}".format(gcv_lasso.best_params_))
print("Ridge best parameters is {}".format(gcv_ridge.best_params_))
print("Elastic Net best parameters is {}".format(gcv_elastic.best_params_))

#These values change on running through this multiple times, probably due to seed issues.
#I had these at 141, 53, <50, and 0.9) one day before submission
```

```
Lasso best parameter is {'lasso_alpha': 54.555947811685144}
Ridge best parameters is {'ridge_alpha': 31.622776601683793}
Elastic Net best parameters is {'elastic_alpha': 31.622776601683793, 'elastic_l1_ratio': 1}
```

In [69]: #Visualising the performance of Lasso and Ridge

```
fig, ax = plt.subplots(1,2, figsize=(20,10))

lasso_res = pd.DataFrame(gcv_lasso.cv_results_)
ridge_res = pd.DataFrame(gcv_ridge.cv_results_)

ax[0].plot(lasso_res['param_lasso_alpha'], lasso_res['mean_train_score'], label = 'Mean Train Score')
ax[0].plot(lasso_res['param_lasso_alpha'], lasso_res['mean_test_score'], label = 'Mean Validation Score')

ax[1].plot(ridge_res['param_ridge_alpha'], ridge_res['mean_train_score'], label = 'Mean Train Score')
ax[1].plot(ridge_res['param_ridge_alpha'], ridge_res['mean_test_score'], label = 'Mean Validation Score')

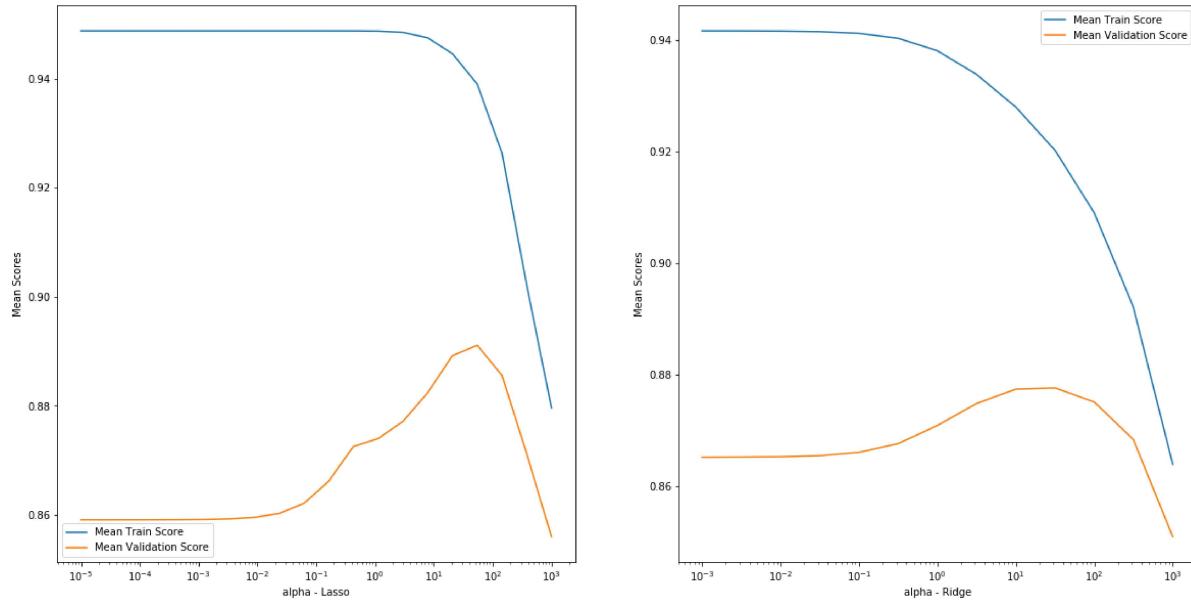
ax[0].set_xscale('log')
ax[1].set_xscale('log')

ax[0].legend()
ax[1].legend()

ax[0].set_xlabel('alpha - Lasso')
ax[0].set_ylabel('Mean Scores')

ax[1].set_xlabel('alpha - Ridge')
ax[1].set_ylabel('Mean Scores')
```

Out[69]: Text(0, 0.5, 'Mean Scores')

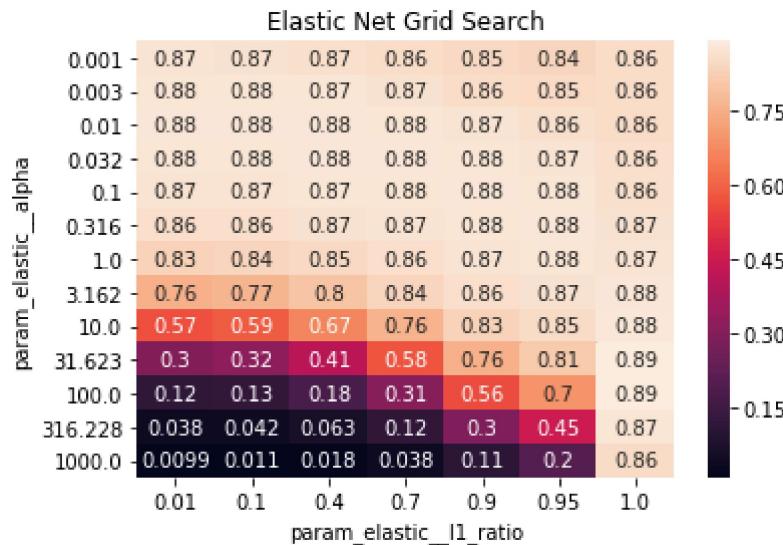


```
In [70]: #Visualising the performance of ElasticNet
import seaborn as sns

res = pd.pivot_table(pd.DataFrame(gcv_elastic.cv_results_), values = 'mean_test_score', index = 'param_elastic_alpha',
                     columns = 'param_elastic_l1_ratio')

res = res.set_index(res.index.values.round(3))
ax = sns.heatmap(res, annot = True)
plt.title('Elastic Net Grid Search')
plt.ylabel ('param_elastic_alpha')
```

Out[70]: Text(33.0, 0.5, 'param_elastic_alpha')



1.6

```
In [71]: #Checking number of co-efficient after Lasso
len (gcv_lasso.best_estimator_.named_steps['lasso'].coef_)
#Checking number of co-efficients after onehotencoding
#len(cont_name) = 34
x = scaling.fit_transform(X_train)
categorical_ohe_lasso = preprocess_scaling.fit(x).named_transformers_[ 'Ohe' ].get_feature_names()
len(preprocess_scaling.fit(x).named_transformers_[ 'Ohe' ].get_feature_names())
#verifying that lengths are the same
```

Out[71]: 307

```
In [72]: len(np.append (cont_name, preprocess_scaling.fit(x).named_transformers_[ 'Ohe' ].get_feature_names()))
#verifying that lengths are the same
```

Out[72]: 341

```
In [73]: lasso_features = pd.DataFrame(gcv_lasso.best_estimator_.named_steps['lasso'].c
oef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_['Ohe'
].get_feature_names())).reset_index()
lasso_features.columns = ['lasso_feature', 'lasso_feature_value']
lasso_features['lasso_feature_value'] = lasso_features['lasso_feature_value'].abs()
lasso_features.sort_values(by=['lasso_feature_value'], ascending = False, inplace = True)
lasso_features.head()
```

Out[73]:

	lasso_feature	lasso_feature_value
106	x13_ClyTile	477604.104900
112	x13_WdShngl	36166.251006
144	x16_NoRidge	35413.949849
151	x16_StoneBr	33760.245291
241	x29_Ex	31804.211235

```
In [74]: ridge_features = pd.DataFrame(gcv_ridge.best_estimator_.named_steps['ridge'].c
oef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_['Ohe'
].get_feature_names())).reset_index()
ridge_features.columns = ['ridge_feature', 'ridge_feature_value']
ridge_features['ridge_feature_value'] = ridge_features['ridge_feature_value'].abs()
ridge_features.sort_values(by=['ridge_feature_value'], ascending = False, inplace = True)
```

```
In [75]: elastic_features = pd.DataFrame(gcv_elastic.best_estimator_.named_steps['elast
ic'].coef_,np.append (cont_name, preprocess_scaling.fit(x).named_transformers_
['Ohe'].get_feature_names())).reset_index()
elastic_features.columns = ['elastic_feature', 'elastic_feature_value']
elastic_features['elastic_feature_value'] = elastic_features['elastic_feature_
value'].abs()
elastic_features.sort_values(by=['elastic_feature_value'], ascending = False,
inplace = True)
```

```
In [76]: lasso_features['lasso_feature'][:15]
```

```
Out[76]: 106      x13_ClyTile
112      x13_WdShngl
144      x16_NoRidge
151      x16_StoneBr
241          x29_Ex
145      x16_NridgHt
333          x43_Ex
302      x39_1Fam
183      x22_BrkFace
133      x16_Crawfor
274          x34_PosA
12       2nd Flr SF
150      x16_Somerst
2        Overall Qual
291          x37_Gd
Name: lasso_feature, dtype: object
```

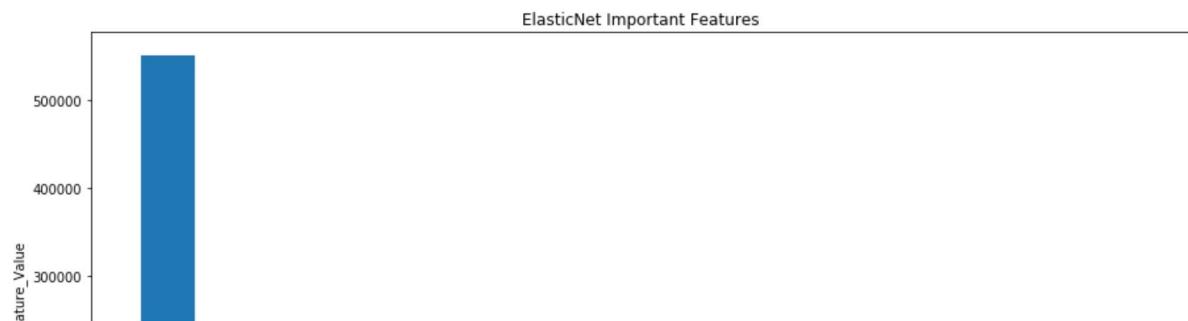
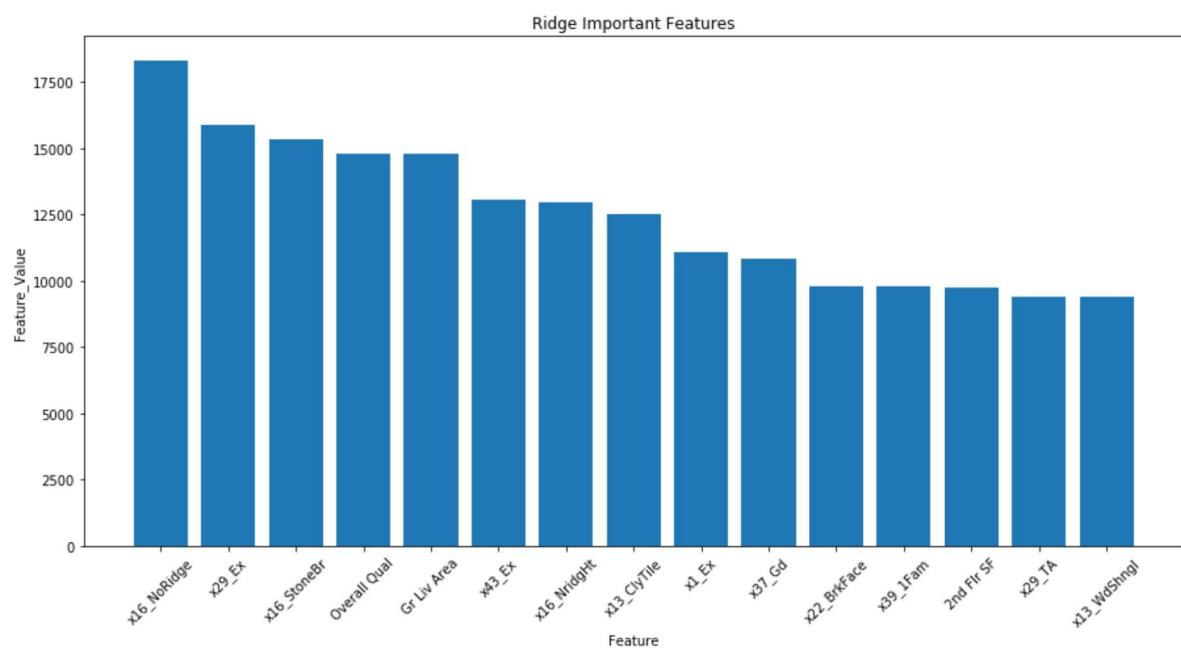
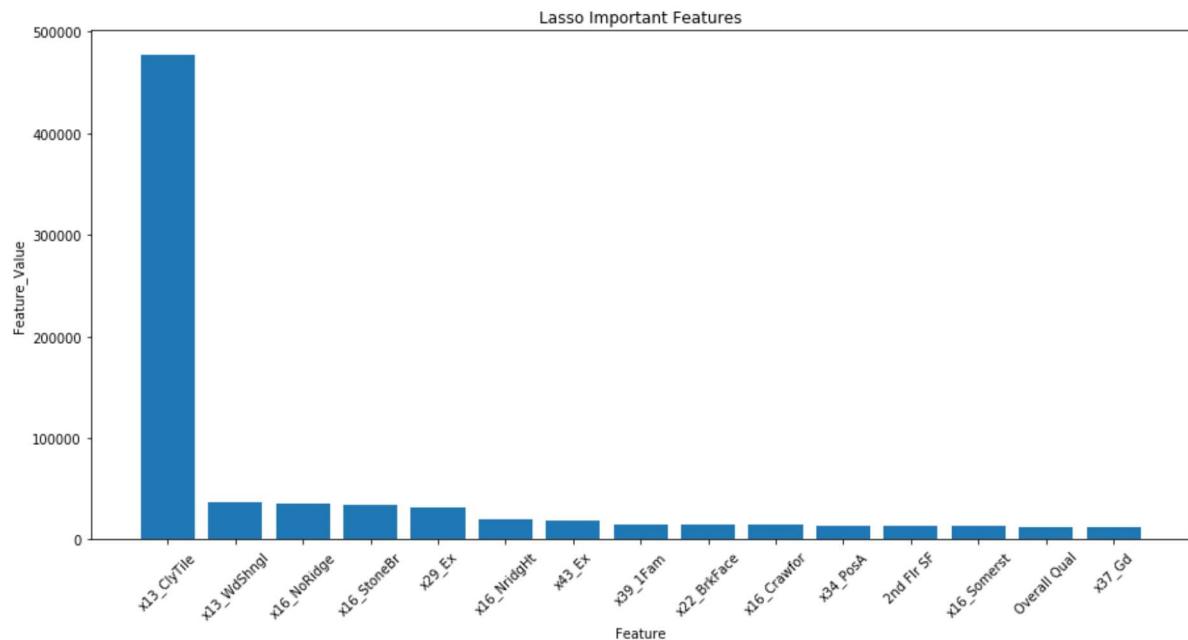
```
In [77]: fig, ax = plt.subplots(3,1, figsize=(15,30))
plt.subplots_adjust(wspace=0.02, hspace=0.60)

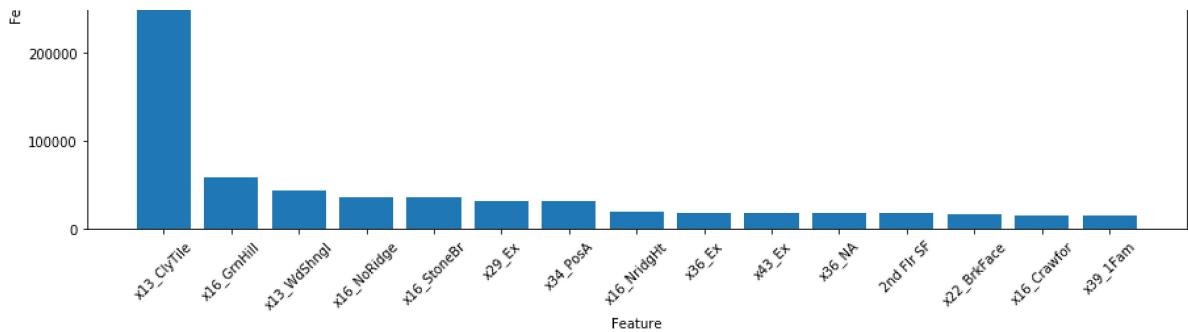
ax[0].bar(lasso_features['lasso_feature'][:15], lasso_features['lasso_feature_value'][:15])
ax[0].set_xticklabels(lasso_features['lasso_feature'][:15], rotation=45)
ax[0].set_title('Lasso Important Features')
ax[0].set_xlabel('Feature')
ax[0].set_ylabel('Feature_Value')

ax[1].bar(ridge_features['ridge_feature'][:15], ridge_features['ridge_feature_value'][:15])
ax[1].set_xticklabels(ridge_features['ridge_feature'][:15], rotation=45)
ax[1].set_title('Ridge Important Features')
ax[1].set_xlabel('Feature')
ax[1].set_ylabel('Feature_Value')

ax[2].bar(elastic_features['elastic_feature'][:15], elastic_features['elastic_feature_value'][:15])
ax[2].set_xticklabels(elastic_features['elastic_feature'][:15], rotation=45)
ax[2].set_title('ElasticNet Important Features')
ax[2].set_xlabel('Feature')
ax[2].set_ylabel('Feature_Value')
```

Out[77]: Text(0, 0.5, 'Feature_Value')





```
In [78]: a = list (lasso_features['lasso_feature'][:15])
b= list (ridge_features['ridge_feature'][:15])
c= list (elastic_features['elastic_feature'][:15])

set(a).intersection(b, c)
```

```
Out[78]: {'2nd Flr SF',
 'x13_ClyTile',
 'x13_WdShngl',
 'x16_NoRidge',
 'x16_NridgHt',
 'x16_StoneBr',
 'x22_BrkFace',
 'x29_Ex',
 'x39_1Fam',
 'x43_Ex'}
```

On running this before submission, 10 out of 15 features are common across these models - from looking at variables, we can further find that certain categorical variables repeat (one-hot encoded forms)

```
In [ ]:
```