Project 2 Ames Housing Data & Kaggle Challenge

Chia, Dominic, Kevin, Jonathan.

Problem Statement

1) We are presented a large training data set , with 78+ features and sales price and ID column with alot of null and missing values. And a smaller test data set without sales price of the houses and also missing values.
2) We would want to **construct a regression model** using the data science process , EDA , regression models and cross validation , only selecting the **most useful features in predicting sales prices**.
3) Thus future clients can know roughly the sales prices of their houses based on the features of their houses.

What do we want to achieve?

We want to come up with the optimal regression model using the training data set by using regression techniques and machine learning to come up with the optimal model for business sale price predictions on a house based on features.

Our model should consist of :

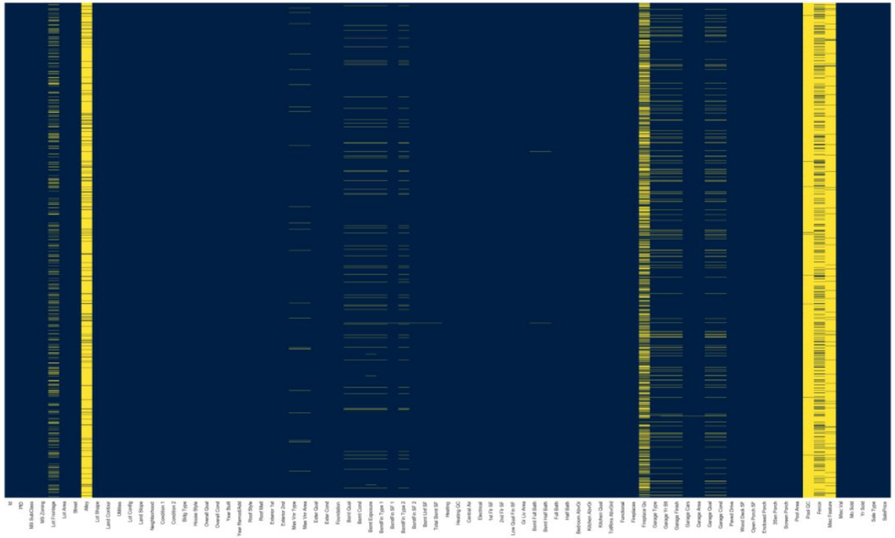
1) reduced number of features

2) a low bias and a low root mean squared error of predicting sales prices on houses on an unseen data set just based on the final best 30+ features we selected.

Data Cleaning

- Detecting Null Values in Various Columns
- Imputing the Right Values into these Null Entries
- Dropping Columns and Rows

Visualization of Null Values across Rows and Columns in Train Set



List of 26 Columns that have null values in Train Set

	Count
Pool QC	2042
Misc Feature	1986
Alley	1911
Fence	1651
Fireplace Qu	1000
Lot Frontage	330
Garage Finish	114
Garage Cond	114
Garage Qual	114
Garage Yr Blt	114
Garage Type	113
Bsmt Exposure	58
BsmtFin Type 2	56
BsmtFin Type 1	55
Bsmt Cond	55
Bsmt Qual	55
Mas Vnr Type	22
Mas Vnr Area	22
Bsmt Half Bath	2
Bsmt Full Bath	2
Garage Cars	1
Garage Area	1
Bsmt Unf SF	1
BsmtFin SF 2	1
Total Bsmt SF	1
BsmtFin SF 1	1

	Percent_Null
Pool QC	99.56%
Misc Feature	96.83%
Alley	93.17%
Fence	80.5%
Fireplace Qu	48.76%
Lot Frontage	16.09%
Garage Finish	5.56%
Garage Cond	5.56%
Garage Qual	5.56%
Garage Yr Blt	5.56%
Garage Type	5.51%
Bsmt Exposure	2.83%
BsmtFin Type 2	2.73%
BsmtFin Type 1	2.68%
Bsmt Cond	2.68%
Bsmt Qual	2.68%
Mas Vnr Type	1.07%
Mas Vnr Area	1.07%
Bsmt Half Bath	0.1%
Bsmt Full Bath	0.1%
Garage Cars	0.05%
Garage Area	0.05%
Bsmt Unf SF	0.05%
BsmtFin SF 2	0.05%
Total Bsmt SF	0.05%
BsmtFin SF 1	0.05%

5 Groups of Columns

1. Masonry Veneer Columns

- 22 Rows of Null Values
- ‘None’ values in ‘Mas Vnr Type’ columns correspond to ‘0.0’ values in ‘Mas Vnr Area’.

```
train['Mas Vnr Type'] = train['Mas Vnr Type'].fillna('None')
train['Mas Vnr Area'] = train['Mas Vnr Area'].fillna(0)
```

2. Basement Columns

	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Total Bsmt SF
1147	GLQ	1124.0	NaN	479.0	3206.0

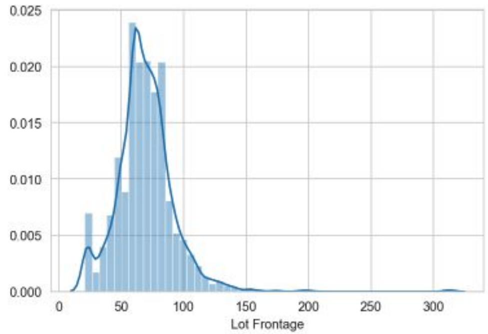
```
train['BsmtFin Type 2'][1147] = 'Rec'
```

	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF
1456	NaN	Unf	0.0	Unf	0.0	725.0	725.0
1547	NaN	Unf	0.0	Unf	0.0	1595.0	1595.0
1997	NaN	Unf	0.0	Unf	0.0	936.0	936.0

```
train['Bsmt Exposure'][1456] = 'No'
train['Bsmt Exposure'][1547] = 'No'
train['Bsmt Exposure'][1997] = 'No'
```

3. Lot Frontage Columns

- Find the median 'Lot Frontage' value of each neighborhood and impute the median value for each null value based on which neighborhood the house (row) was located in.



4. Misc Feature, Alley, Fence Columns

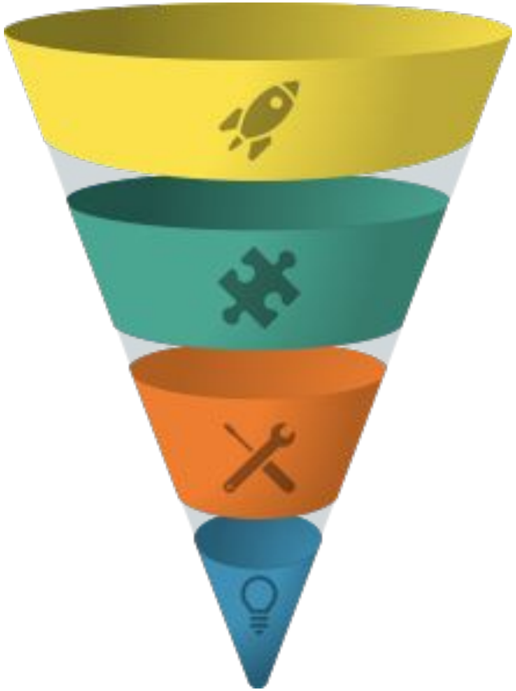
Misc Feature	96.83%
Alley	93.17%
Fence	80.5%

```
# Dropping these 3 columns
train = train.drop(['Misc Feature', 'Alley', 'Fence'], axis=1)
```

5. Garage, Fireplace, Pool Columns

Feature Selection Criteria

78 Features



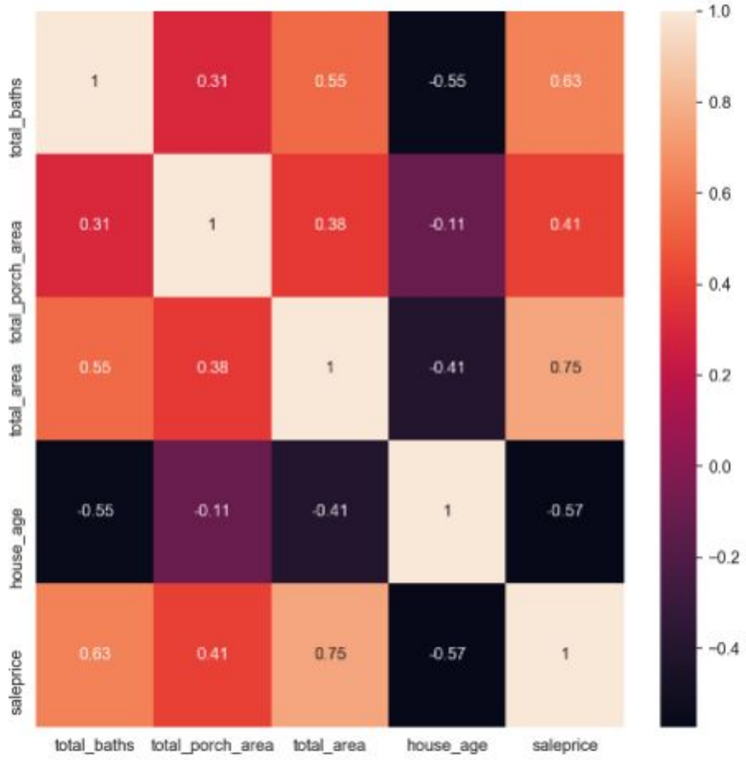
27 Features

- Feature Engineering & Mapping Ordinal Features
- 80% Threshold Test to Remove Skewed Features
- Correlation Test to Remove 15 Lowest Correlated Features
- EDA and Removing Outliers

Feature Engineering

- Basement Columns
- Bathroom Columns
- Porch Columns
- Area Columns
- House Year Columns
- Dropped 16 Columns
- Created 5 New Columns
- Remaining Features: 66

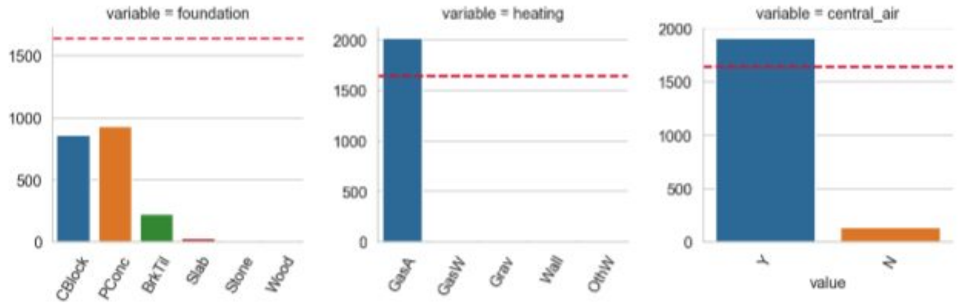
Correlation Heatmap Between New Features



Threshold Test

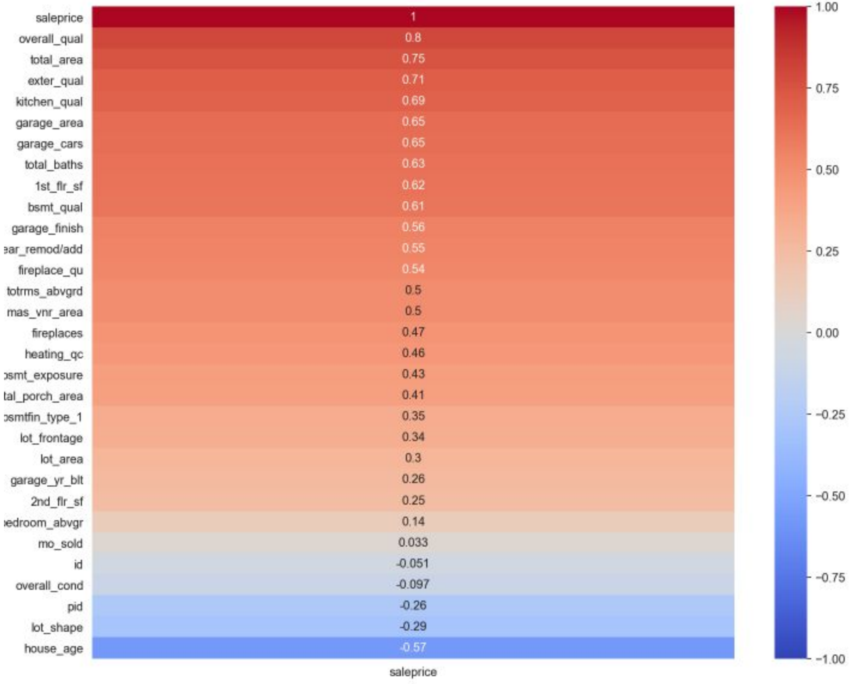
```
# Identifying columns that exceed the 80% threshold
cat_above_threshold = []
cat_within_threshold = []

for column in cat_columns:
    if train[cat_columns][column].value_counts().sort_values(ascending = False).values[0] > 0.8*train.shape[0]:
        cat_above_threshold.append(column)
    else:
        cat_within_threshold.append(column)
```



- Categorical Features: 20
  - Removed: 9
  - Kept: 11
- Numerical Features: 46
  - Removed: 15
  - Kept: 31
- Remaining Features: 42

Correlation Test




```python
# We will remove the lowest 15 variables from our train data based on absolute value of corr
corr = abs(train.corr())
abs_corr = pd.DataFrame(corr['saleprice'].sort_values(ascending = False).tail(15))
print(abs_corr.index)
len(abs_corr)
```

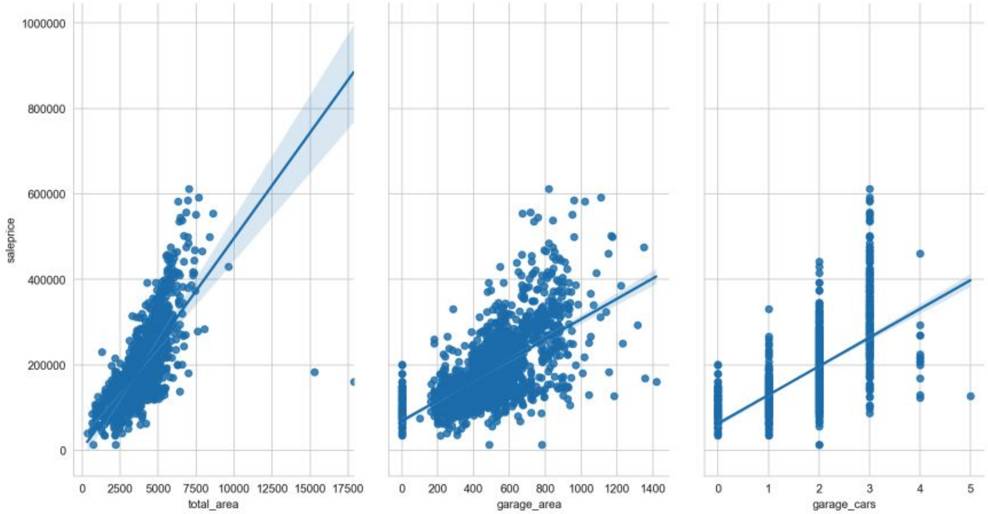
```
Index(['fireplaces', 'heating_qc', 'bsmt_exposure', 'total_porch_area',
       'bsmtfin_type_1', 'lot_frontage', 'lot_area', 'lot_shape',
       'garage_yr_blt', 'pid', '2nd_flr_sf', 'bedroom_abvgr', 'overall_cond',
       'id', 'mo_sold'],
      dtype='object')

15
```

- Dropped the lowest 15 variables from train set based on lowest absolute correlation with SalePrice
- Numerical Features Removed: 15
- Remaining Features: 27

EDA and Removing Outliers

```
# Plotting scatterplots of 'total_area', 'garage_area', 'garage_cars' against saleprice.
ax = sns.pairplot(train, x_vars = ['total_area', 'garage_area', 'garage_cars'], y_vars = 'saleprice',kind='reg');
ax.fig.set_size_inches(12,7)
```




```
# Finding these outliers
train.loc[train['total_area']>15000]
```

	ms_subclass	ms_zoning	lot_config	neighborhood	house_style	overall_qual	year_remod/add	roof_style	exterior_1st
960	60	RL	Corner	Edwards	2Story	10	2008	Hip	Stucco
1885	20	RL	Inside	Edwards	1Story	10	2009	Hip	CemntBd

- Removed 2 Rows from the train set
- Remaining Rows: 2049
- Number of Features: 27

One Hot Encoding

- For Categorical Variables, we will create further dummy variables using the one hot encoding method and add them into our train and test sets.
- This will make our datasets more interesting and can potentially lead to better results and predictions in our modeling later on.

```
# Creating dummy variables in train set
train = pd.get_dummies(train, columns = cat_within_threshold, drop_first = True)
train.columns
```

```
Index(['overall_qual', 'year_remod/add', 'mas_vnr_area', 'exter_qual',
       'bsmt_qual', '1st_flr_sf', 'kitchen_qual', 'totrms_abvgrd',
       'fireplace_qu', 'garage_finish',
       ...
       'foundation_PConc', 'foundation_Slab', 'foundation_Stone',
       'foundation_Wood', 'garage_type_Attchd', 'garage_type_Basment',
       'garage_type_BuiltIn', 'garage_type_CarPort', 'garage_type_Detchd',
       'garage_type_NA'],
      dtype='object', length=122)
```

```
# Drop these columns in the train set excluding saleprice column
train = train.drop(['ms_subclass_150', 'ms_zoning_C (all)', 'neighborhood_GrnHill', 'neighborhood_Landmrk',
                    'exterior_1st_CBlock', 'exterior_1st_ImStucc', 'exterior_1st_Stone', 'exterior_2nd_Stone'],
                   axis = 1)
train.shape
```

```
(2049, 114)
```

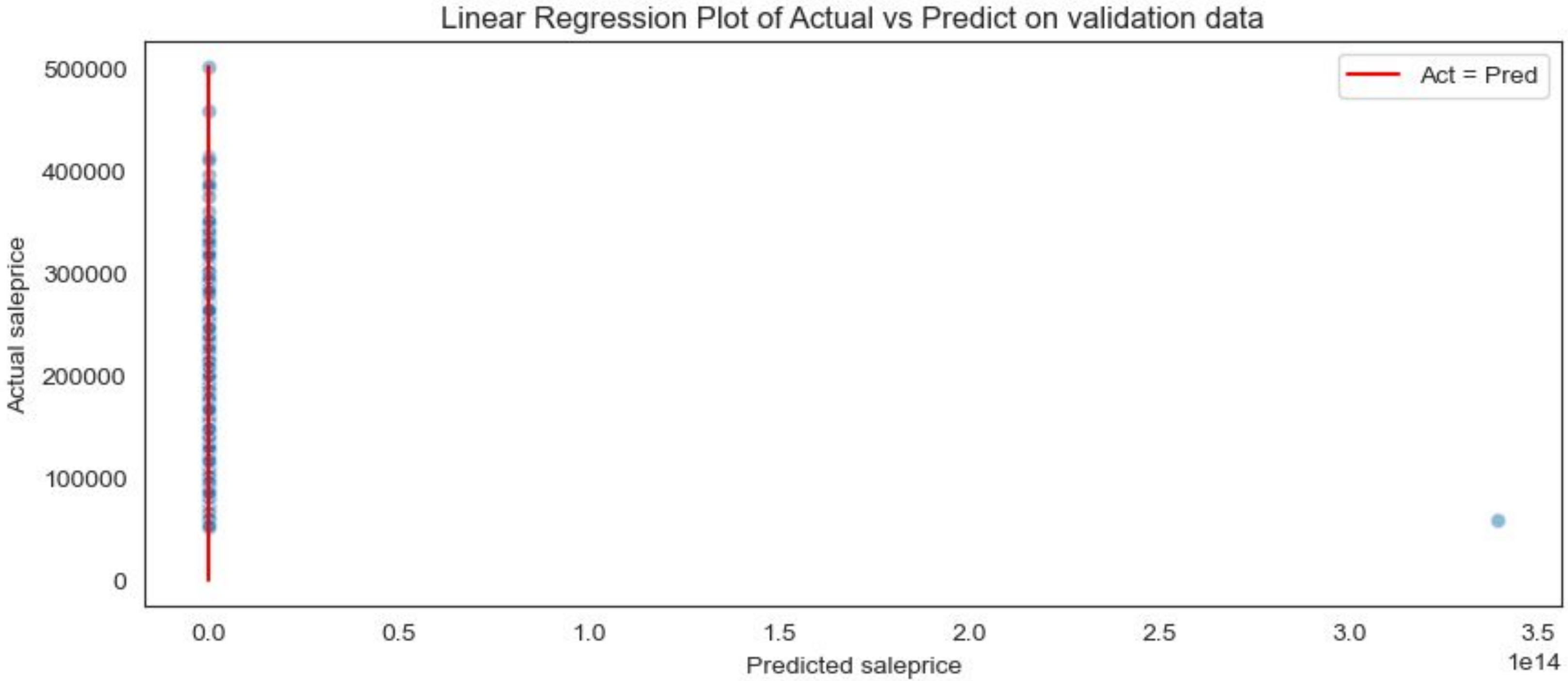
Number of Remaining Features: **114**

Modeling of Variables

- Contains 158 features
- Use of LR, Ridge, Lasso & Elastic Net
- Use of Cross Validation to identify features & parameters
- Polynomial Features to find interaction terms.

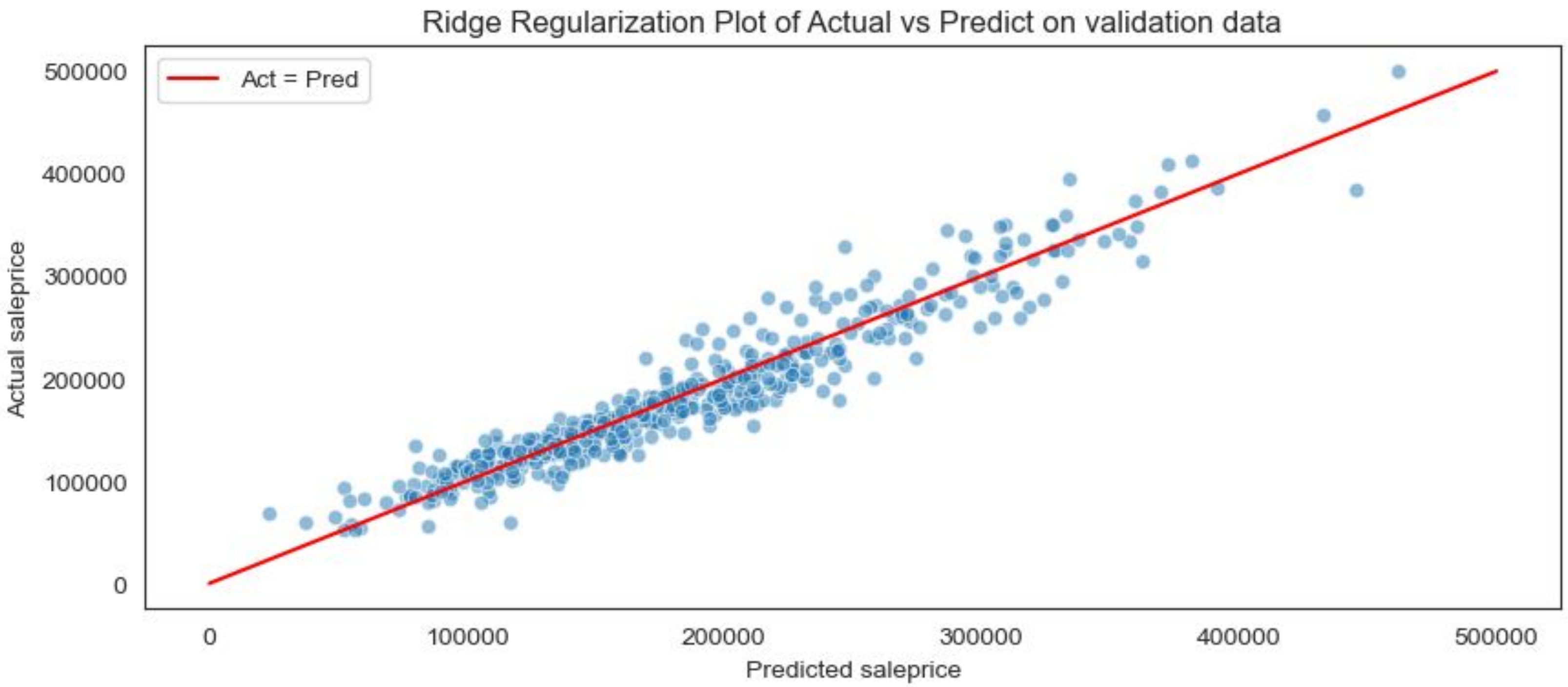
LR Plot with all variables

LR	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89987	N.A	24,099.34600
Train	0.91667	0.90713	21,982.27952
Test	0.91761	0.88092	20,484.14569



Ridge Plot with all variables

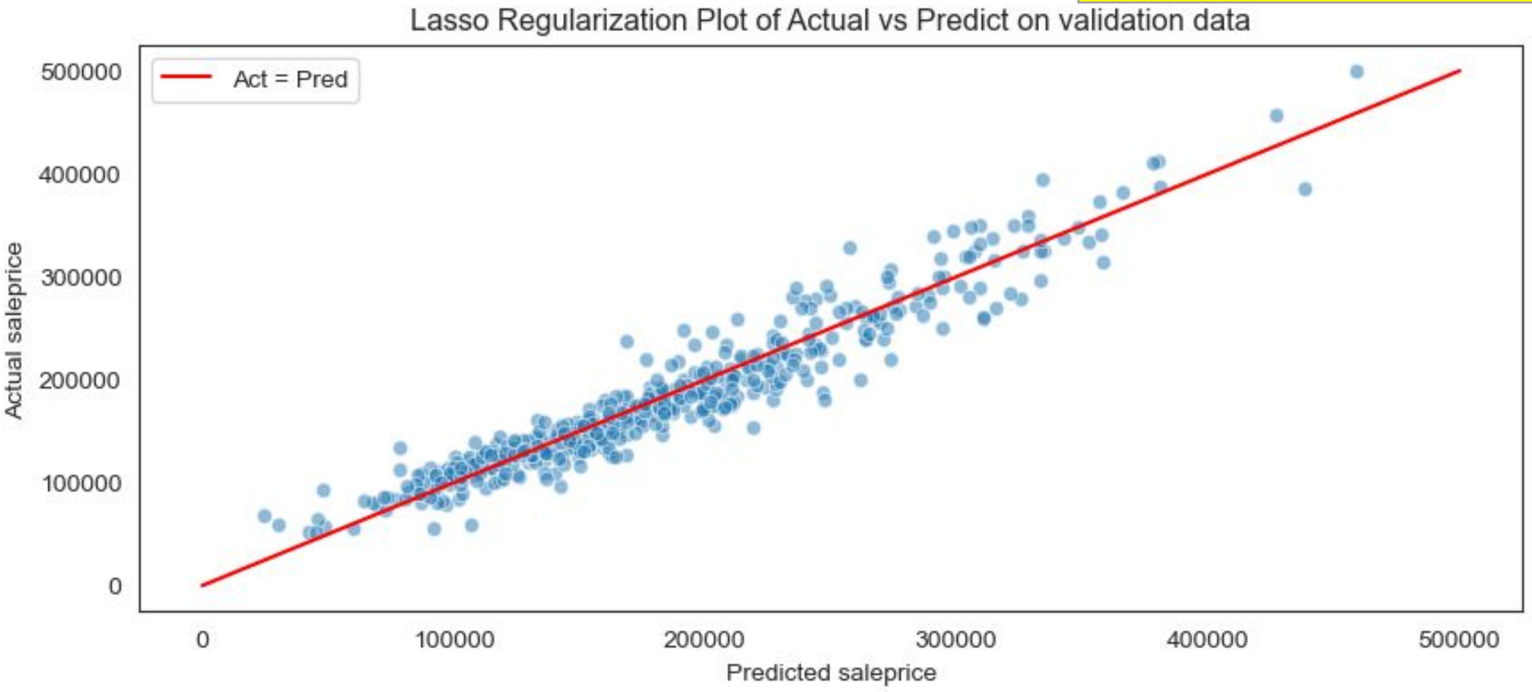
Lasso	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89349	N.A	24,859.17121
Train	0.92018	0.91104	21,514.47429
Test	0.91606	0.87868	20,675.94979



Lasso Plot with all variables

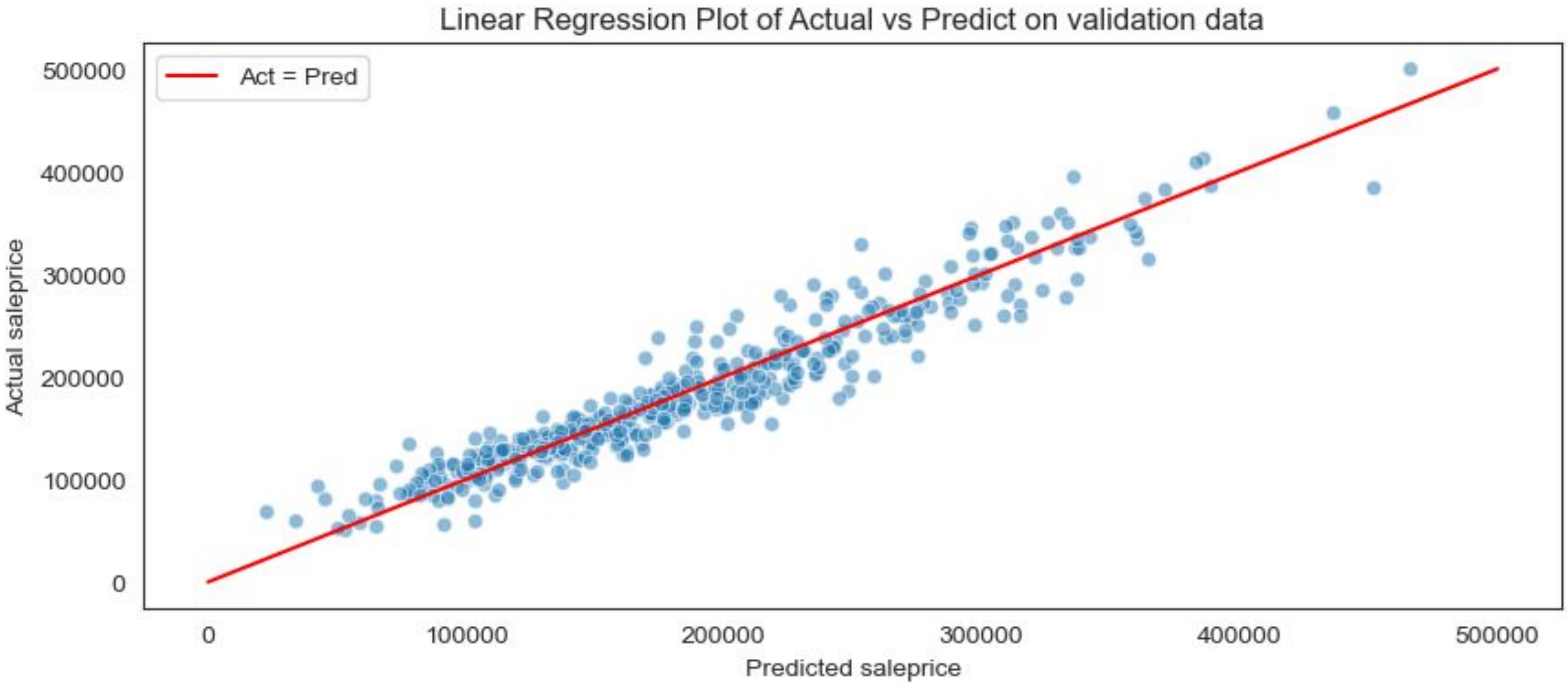
Lasso	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89987	N.A	24,099.34600
Train	0.91667	0.90713	21,982.27952
Test	0.91761	0.88092	20,484.14569

Number of redundant features to drop: 75



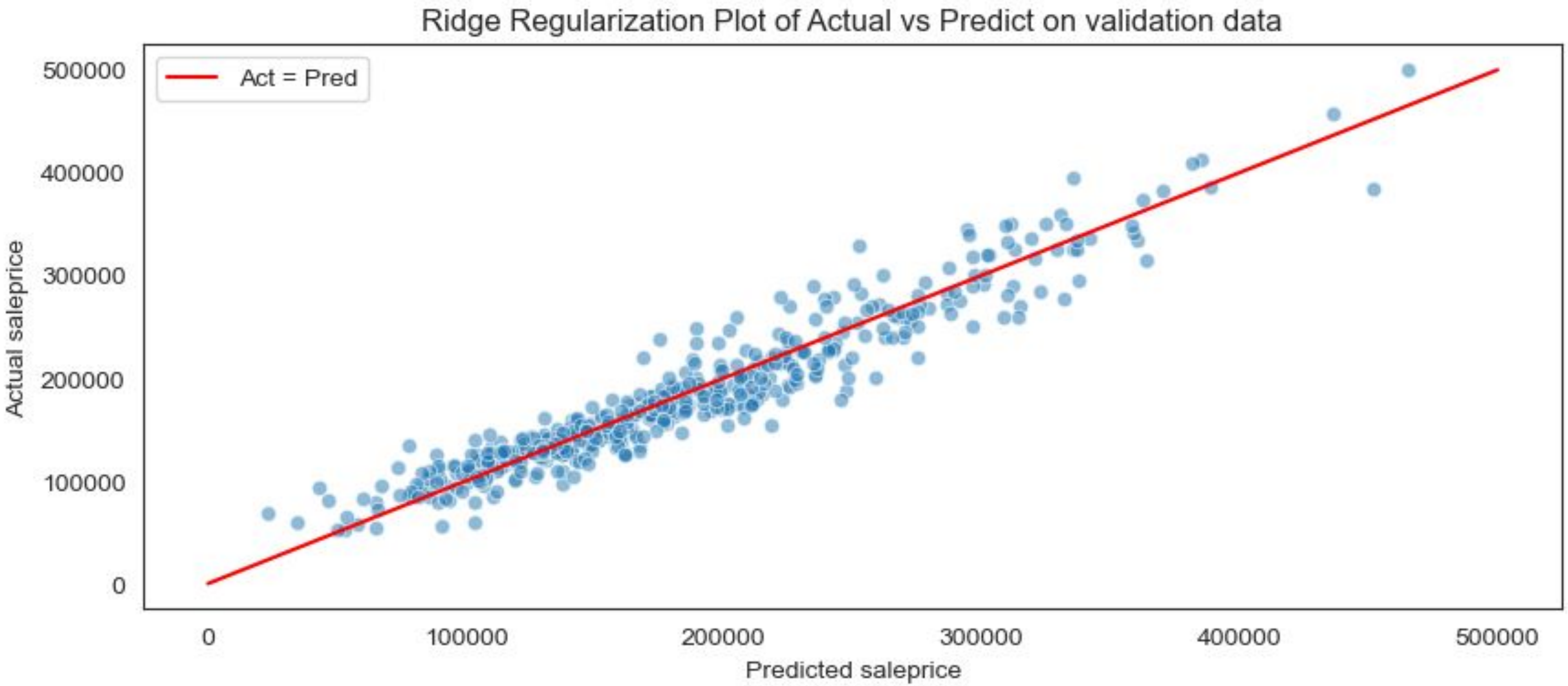
Model plot after removal of redundant feature

LR	R^2	Adj-R^2	RMSE
Mean (Cv)	0.90341	N.A	23,676.05747
Train	0.91988	0.91539	21,555.02788
Test	0.91128	0.89451	21,256.37796



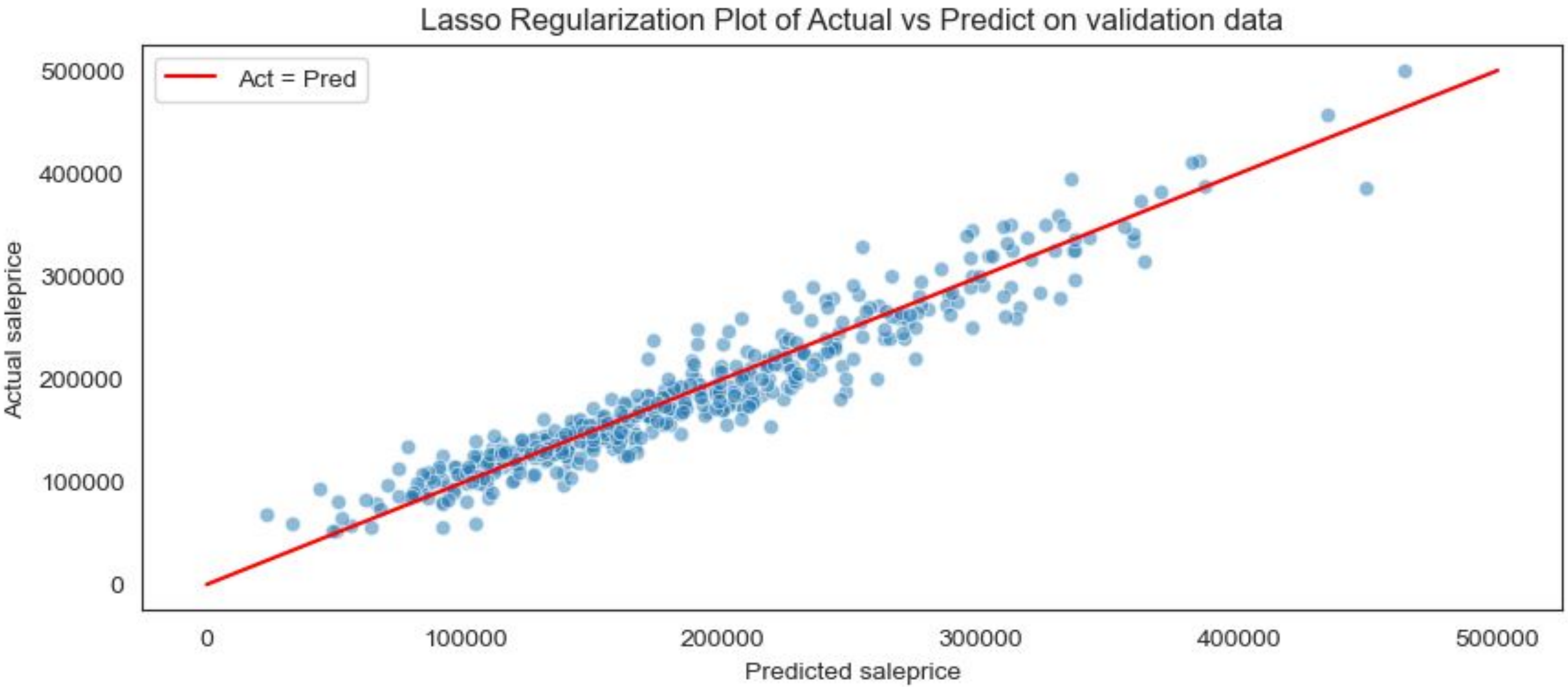
Model plot after removal of redundant feature

LR	R^2	Adj-R^2	RMSE
Mean (Cv)	0.90393	N.A	23,618.55509
Train	0.91985	0.91536	21,558.70488
Test	0.91173	0.89504	21,202.87923



Model plot after removal of redundant feature

LR	R^2	Adj-R^2	RMSE
Mean (Cv)	0.90376	N.A	23,631.94110
Train	0.91969	0.91519	21,581.42846
Test	0.91377	0.89747	20,955.96026



Evaluation of 1st model fitting

- All model have reduced RMSE significantly (~21K) compared to base model (75K)
  - Best model are Ridge & Lasso model which penalise high variance.

- Main focus was on lasso - feature elimination attributes to remove redundant features - 75 variables

Features Engineering - Part 1

- After removal of 75 features, 83 out of 158 features remains

- How can we reduce 83 variables to our acceptable range of 20-30 variables?
  - Most variables have low correlation with price
  - Most of the remaining variables are dummy variables which are hard to explain to stakeholder & do not have strong correlation to be retained (>0.5)

- Remove features with correlation below 0.2 (vs saleprice) => Retained 30
- Remove features that are dummy variables => Retained 20

Feature Engineering - Part 2

- Group common variable together to find interaction terms.
- 3 group - Lot group, Basement group & Garage group
- Avoid already highly correlated variable that can be apply across all variables eg. overall qual, exter_qual
- Created 32 interaction terms

List of retained features

'lot_frontage',
'lot_area',
'overall_qual',
'mas_vnr_area',
'exter_qual',
'bsmt_qual',
'bsmt_cond',
'bsmt_exposure',
'bsmtfin_type_1',
'total_bsmt_sf',
'heating_qc',
'gr_liv_area',
'kitchen_qual',
'fireplaces',
'garage_yr_blt',
'garage_finish',
'garage_area',
'basmt_total_bath',
'total_bath',
'year_build_remod'

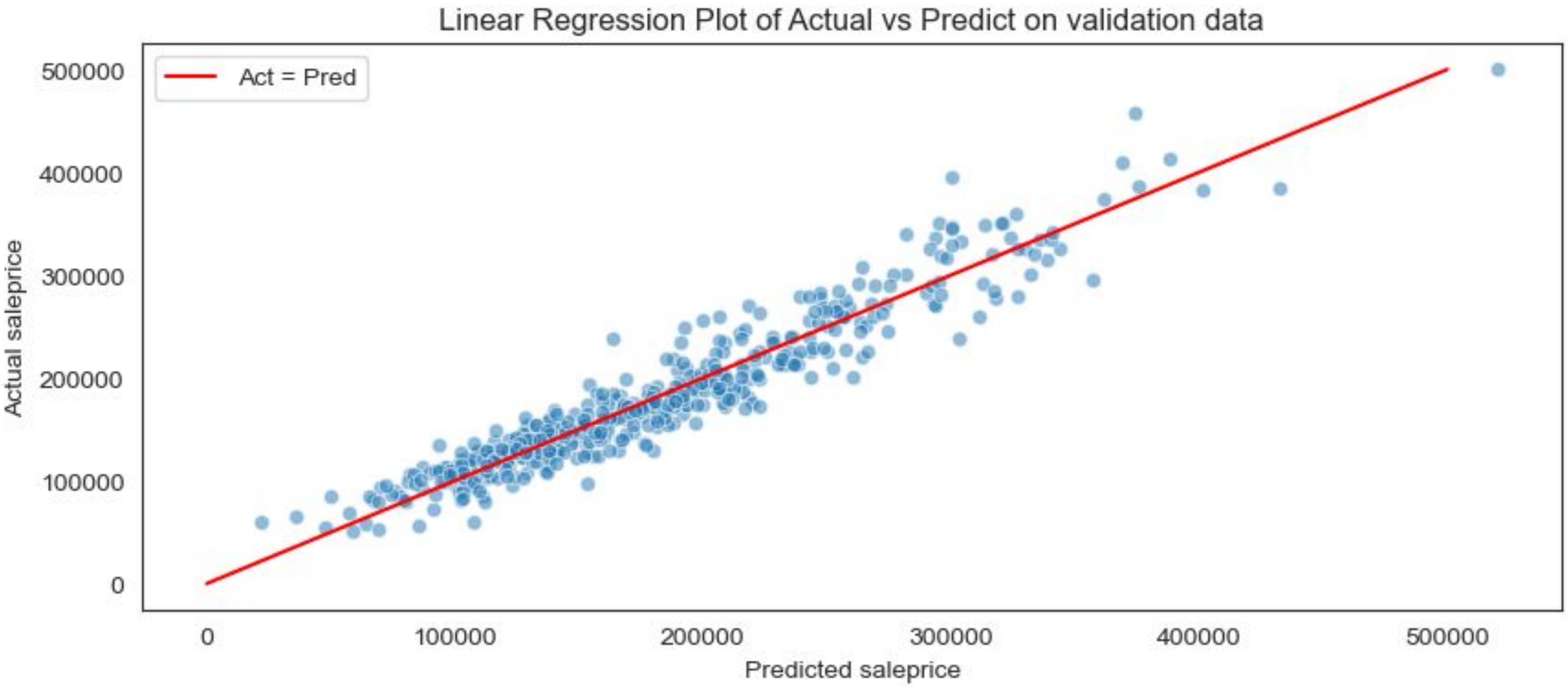
2nd Set of Model Fitting

- Features elimination process:
  - Due to inclusion of interaction terms, to remove features that are redundant
  - To reduce our range of features to as close to acceptable range of around 30

Lasso	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89189	N.A	25,023.26776
Train	0.90004	0.89654	24,076.29822
Test	0.91319	0.90339	21,027.11678
Number of redundant features to drop: 18			

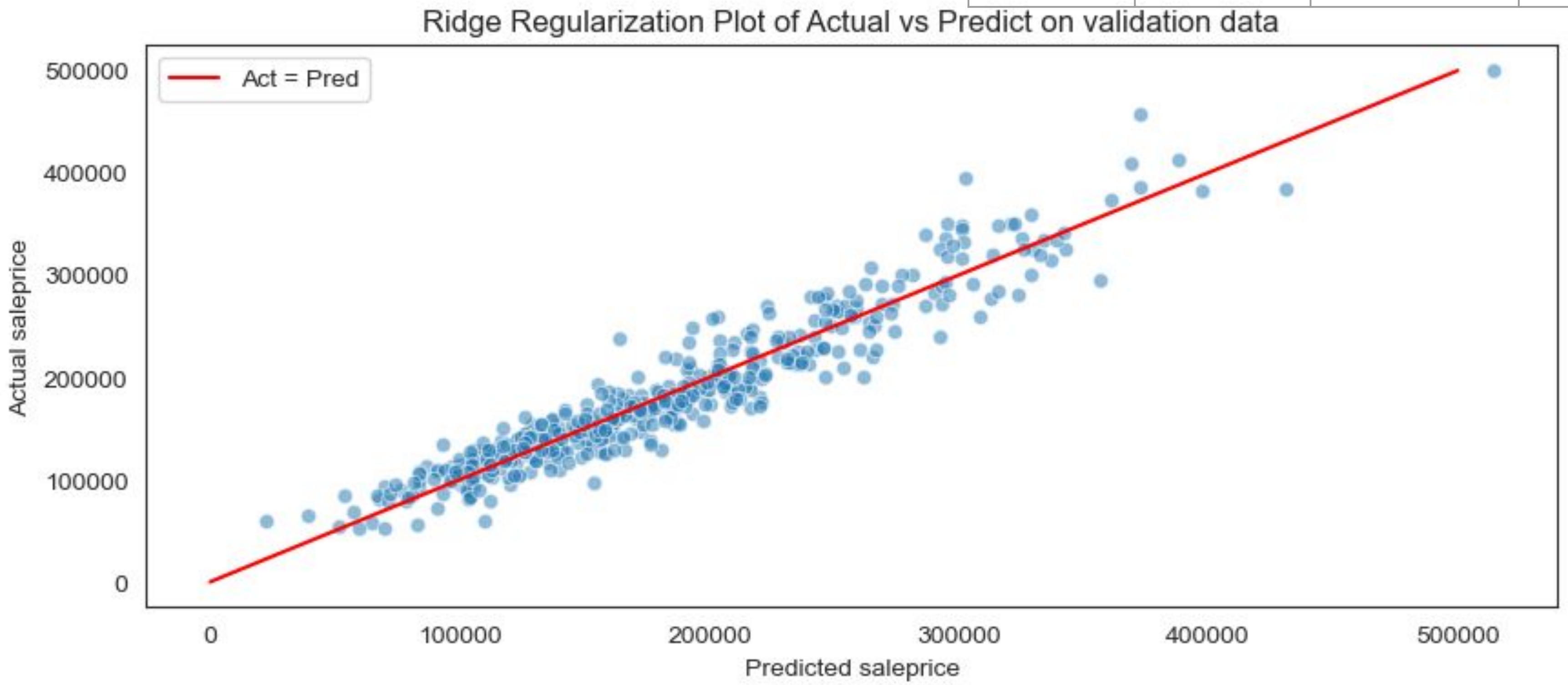
2nd Model - LR

LR	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89240	N.A	24,965.24038
Train	0.90061	0.89839	24,007.40924
Test	0.91199	0.90581	21,171.22912



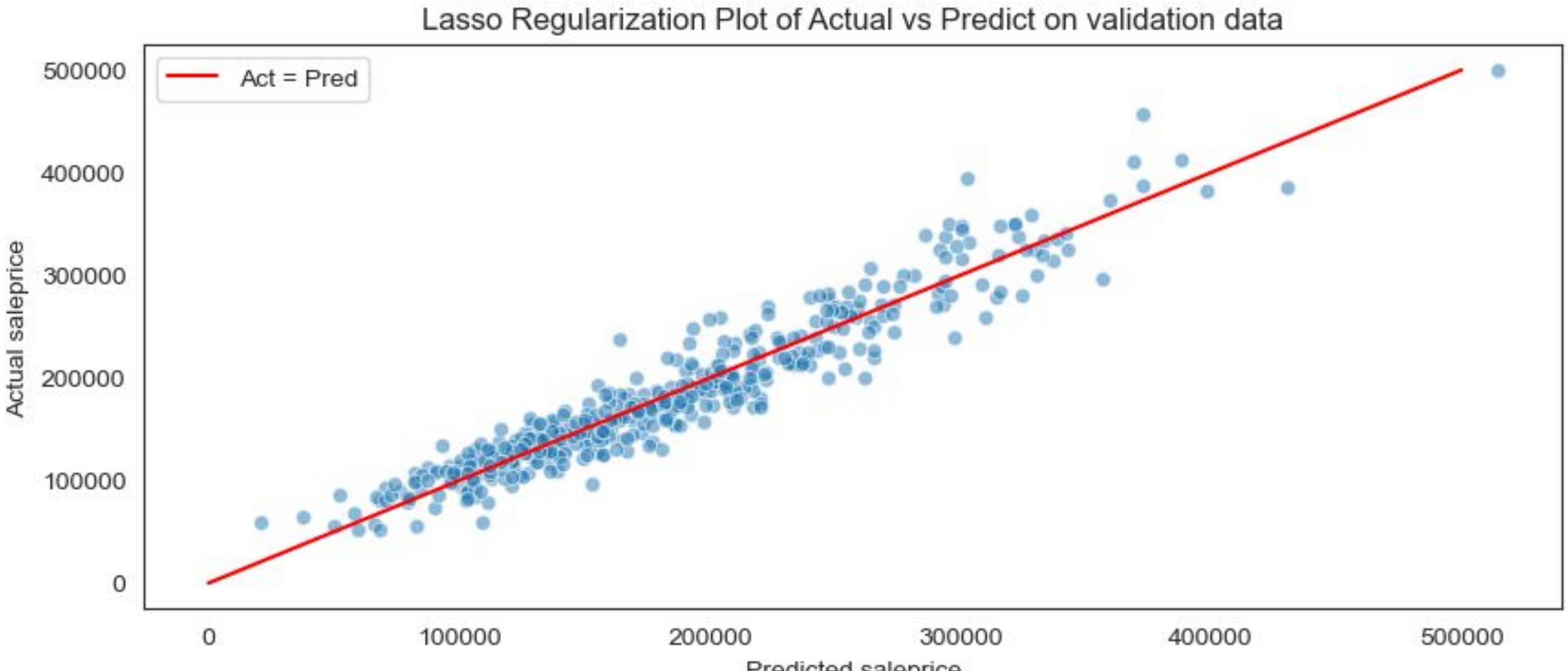
2nd Model - Ridge

Ridge	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89194	N.A	25,015.70735
Train	0.89980	0.89756	24,105.18305
Test	0.91362	0.90756	20,974.25300

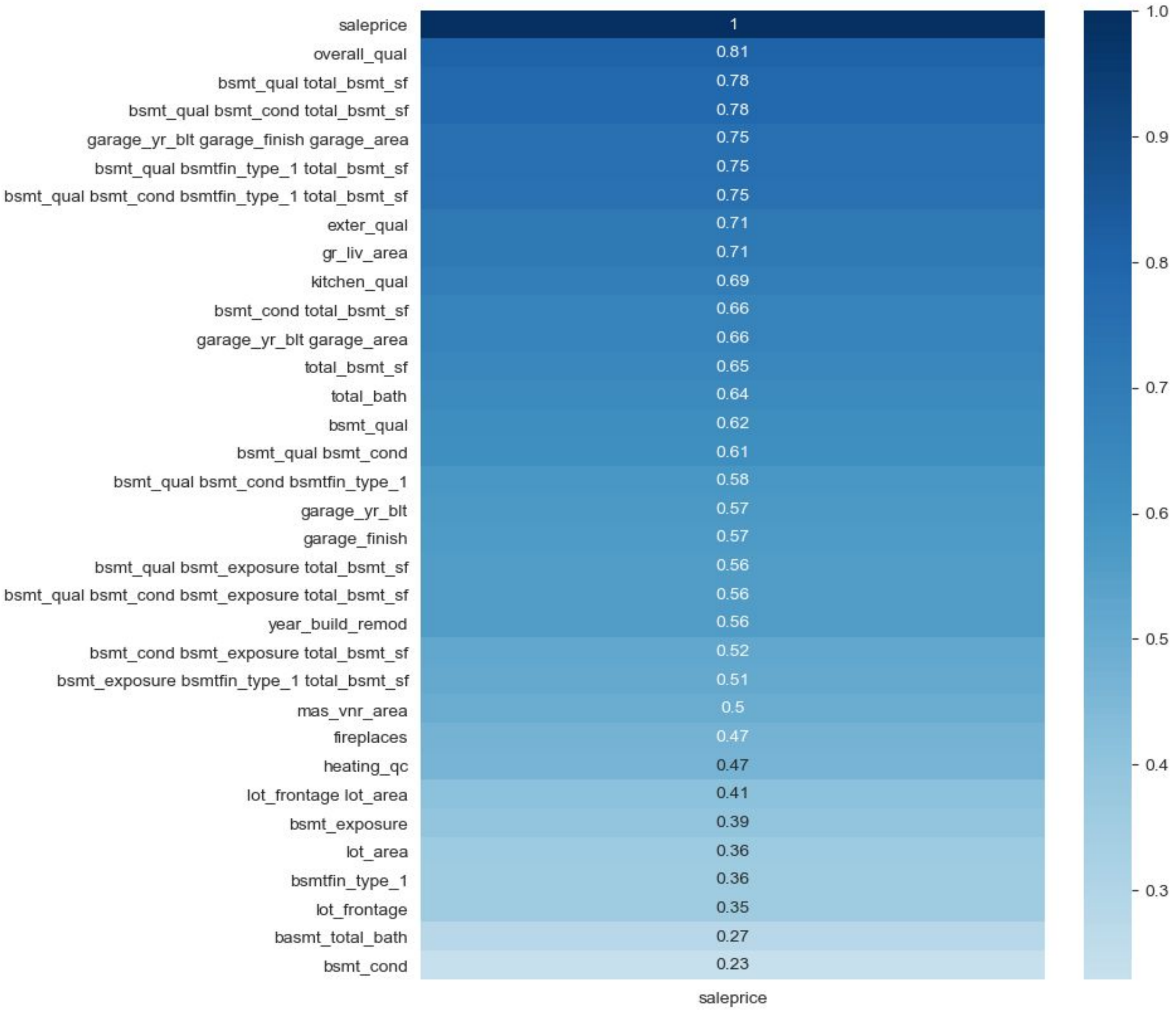


2nd Model - Lasso

Lasso	R^2	Adj-R^2	RMSE
Mean (Cv)	0.89229	N.A	24,975.03955
Train	0.90004	0.89781	24,076.3728
Test	0.91319	0.90709	21,026.85411



2nd Model Variable correlation with price



Fitting Final Features for Kaggle Test

- Choose 4 type of models - LR, Ridge, Lasso & Elastic Net.
- Find the model with the best params

```
# Instantiate models
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
lr_cv_scores = cross_val_score(lr, X_train_scaled,y_train, cv=5)

lasso_cv = LassoCV(n_alphas=100, cv=5, random_state=42,max_iter=10000)
lasso_cv.fit(X_train_scaled, y_train)

ridge_cv = RidgeCV(cv=5)
ridge_cv.fit(X_train_scaled, y_train)

en_cv = ElasticNetCV(cv=5, alphas=np.linspace(0.1,1,100), max_iter=10000, random_state=42)
en_cv.fit(X_train_scaled, y_train)
```

Result of model Validation Scoring

- Lasso provided the best Adj-R^2 based on cross-validation on train-test-split

Model	R^2	Adj-R^2	RMSE
Base			71,365.37
LR (CV)	0.912	0.906	21,671.23
Ridge (CV)	0.912	0.906	21,112.08
Lasso (CV)	0.913	0.907	21,026.85
Elastic Net (CV)	0.913	0.907	21,055.42

Fitting Kaggle Test into Trained Lasso Model

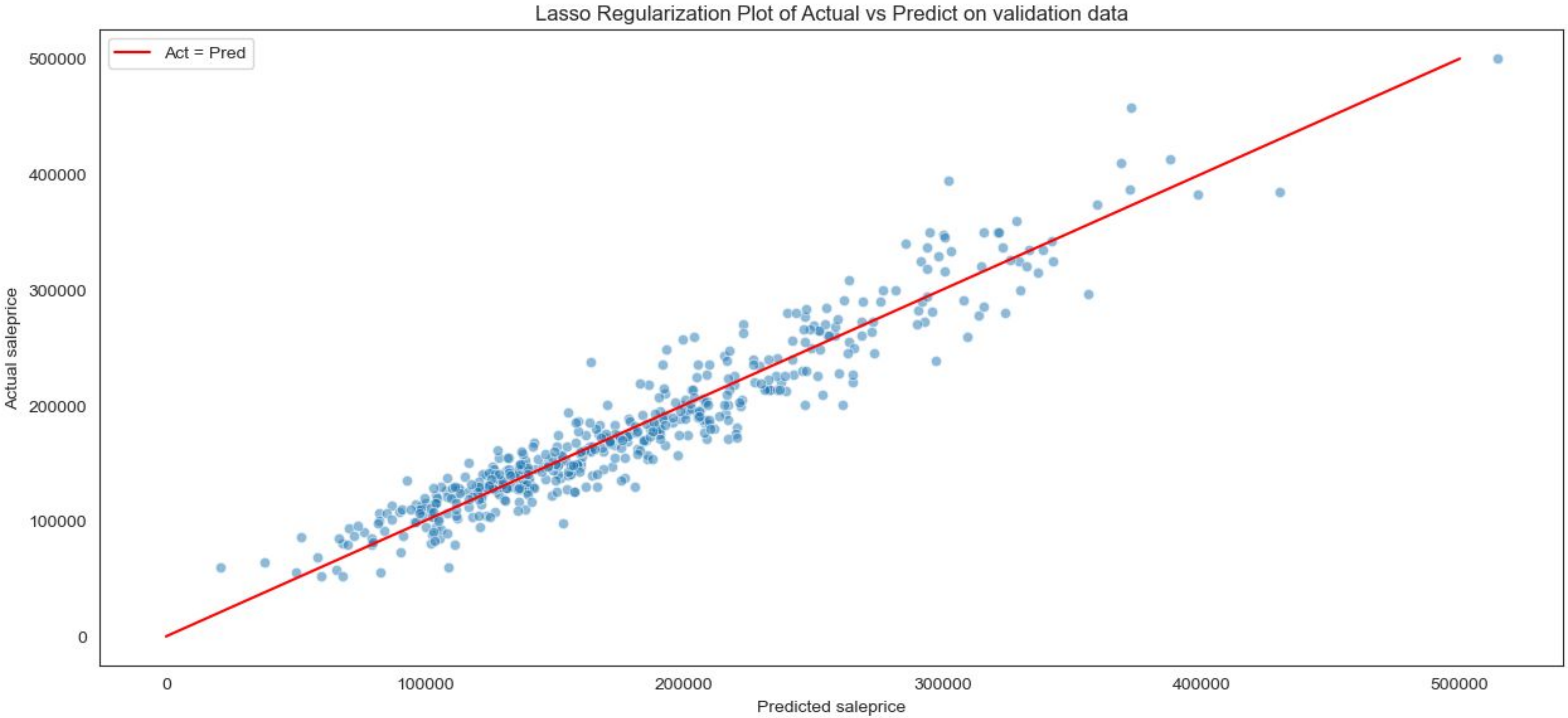
In [48]:
```
lasso_cv.alpha_
```
Out[48]:
```
61.7304488630612
```

In [49]:
```
lasso = Lasso(alpha=lasso_cv.alpha_, max_iter=10000)
lasso.fit(X_train_scaled,y_train)
```
Out[49]:
```
Lasso(alpha=61.7304488630612, max_iter=10000)
```

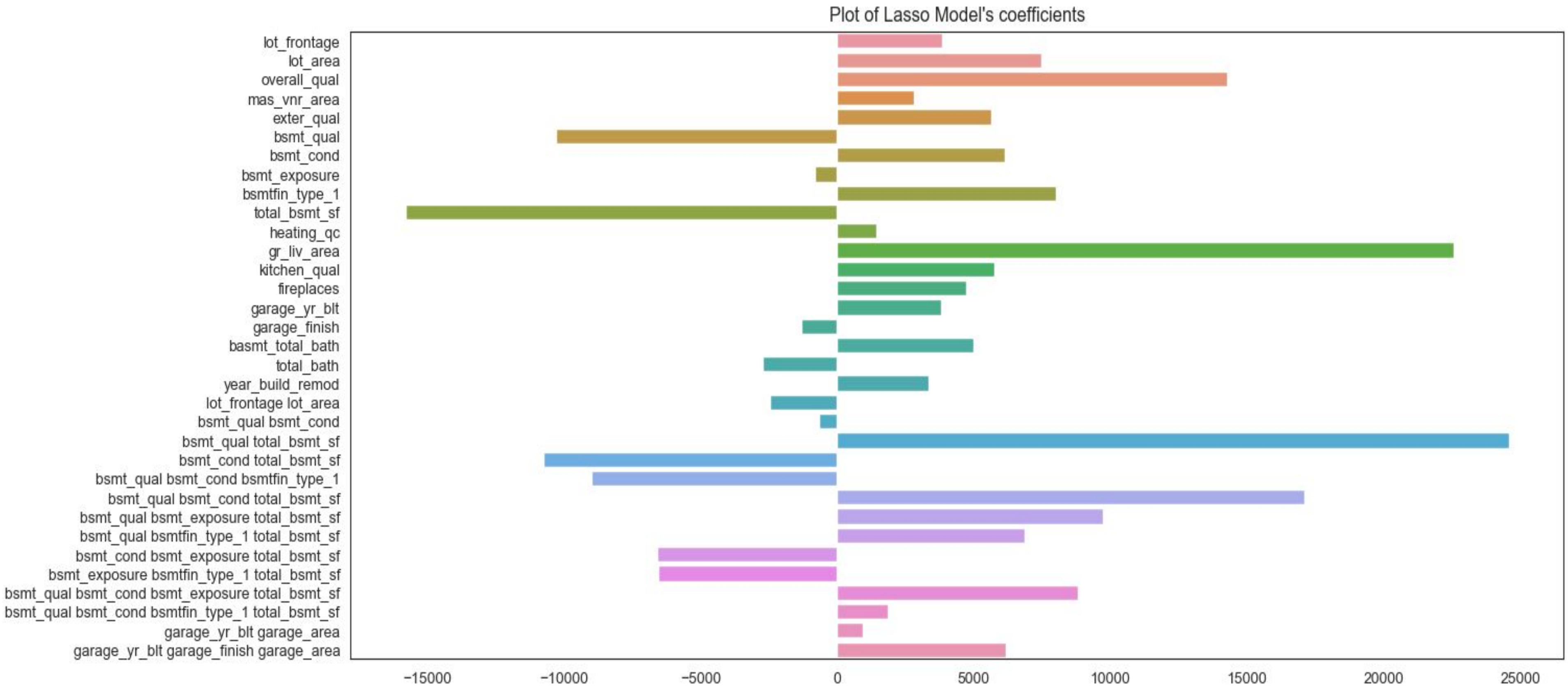
In [50]:
```
print(f'R^2 of Ridge(train):{lasso.score(X_train_scaled,y_train)}')
print(f'RMSE of Ridge(train): {np.sqrt(metrics.mean_squared_error(y_train, lasso.predict(X_train_scaled))): ,.5f}')
print('-'*60)
print(f'R^2 of Ridge(test):{lasso.score(X_test_scaled,y_test)}')
print(f'Adj-R^2 of Ridge(test):{r2_adj(y_test, lasso.predict(X_test_scaled), len(X_train.columns))}')
print(f'RMSE of Ridge(test): {np.sqrt(metrics.mean_squared_error(y_test, lasso.predict(X_test_scaled))): ,.5f}')
```
```
R^2 of Ridge(train):0.9000425825959958
RMSE of Ridge(train):  24,076.37282
------------------------------------------------------------
R^2 of Ridge(test):0.9131892454207905
Adj-R^2 of Ridge(test):0.907094022226931
RMSE of Ridge(test):  21,026.85411
```

Base Model based on validation data

In [51]:
```
# Root Mean Squared Error of y_test
kt_test = pd.DataFrame(y_test, columns = ['saleprice'])
kt_test['saleprice_mean'] = kt_test['saleprice'].mean()
print_mse(kt_test, 'saleprice', 'saleprice_mean')
```
```
Shape of dataframe: (504, 2)
Base Mean Squared Error: 5,093,016,366.180
Base Root Mean Squared Error: 71,365.372
```


Lasso Regularization Plot of Actual vs Predict on validation data
Act = Pred
Actual saleprice
Predicted saleprice
0
100000
200000
300000
400000
500000

Coefficient of Model



Conclusion

- With the **33** selected features and using cross validation on validation data, the **Lasso model is able to explain 91% of total variances**. (Adj-R^2 of 0.91) and have the lowest RMSE score compared to other models when doing model selection using cross validation.

- In addition compared to base model which has a **RMSE of 71K**. Our model only have a **RMSE of 21K**. This prove that our model work significantly well compared in predicting sale prices for houses versus using the baseline model of not dropping any features.

- Based on these, I can conclude that the **Lasso model provided us with the best prediction of sales prices** for Iowa's real estate.

Analysis of coefficients

- Looking at the 33 variables, it can be concluded that variables relating to basement are produce significant drag on sale price.

Will the model work in other cities?

We would need to look at the unique features in the other cities , and run our model on it to predict on the sales price and use the same data science process to find the optimal regression model.

Our model is currently limited to Iowa due to the features and sales price data set.

Business recommendations for sales price

- **Developers**

1. May want to consider houses that may not have basement, this can boost sale price and also lower their cost as more labour efforts are need to create the basement.

Should the developers design their basements , they should take into consideration on the quality, condition and size of basement as this combined feature ,rather than as individual components, has a very significant positive impact on sale price.

2. Lot frontage surprisingly has a positive impact to sale price, it could be due to resident preferring a good social distance/privacy or a sense of owning gardens that set their houses a distance away from connecting streets.

- **Home Owners:**

1) How to improve their sales prices of their houses , given that they cant increase the square feet of their house.
2) They can improve their overall quality of their homes These include : fireplace, kitchen quality, basement condition & quality (if they have basement)

Thank you!