Project 2: REGRESSION

AMES Housing Data & Kaggle Challenge

Pratch Luepuwapitakkul

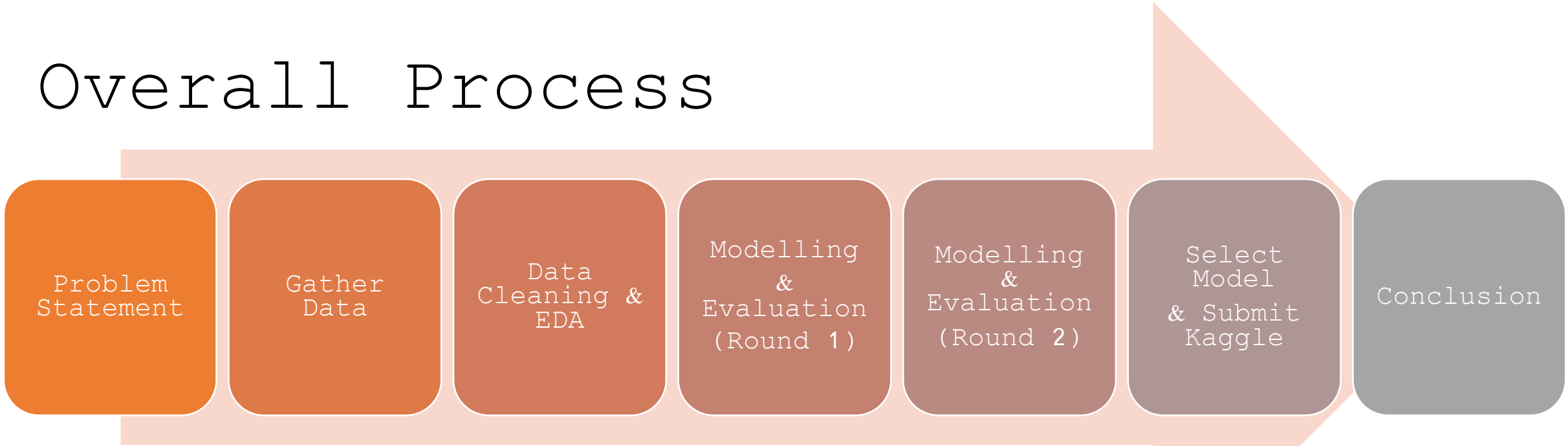
Problem Statement

The objective of this project is to create a regression model that will predict the price of a house at sale in the city of Ames, Iowa. This model will be used by real estate marketplace apps (like Property Guru) to help home sellers, buyers, or agents to estimate the sale price of a house by inputing their selection of relevant multiple features (i.e. Neighborhood, no. of rooms, etc).

Data

- **Train Dataset**
- There are **81** columns (variables)
  - The dependent variable **(Y)** in this case is the SalePrice (last column)
  - There are 80 other columns (Id + 79 features).
- There are **2,051** rows (observations)
- **Test Dataset**
- There are **80** columns (variables), same as the train dataset, except **saleprice is not included as it is the target for prediction**
- There are **879** rows

Overall Process



- Import library & read data files
- Understand Data Description

- Fix Null Values
- Check Data Types
- Map/Encode Categorical Values
- EDA
- Initial Feature Selection

Preprocessing:
- Train-Validation Split
- Scale the data

Build & Evaluate Models
- Linear Regression
- Lasso
- Ridge
- Elastic Net

Iterate on the models:
- Analyze coefficients from round 1
- Re-modelling with selected features.

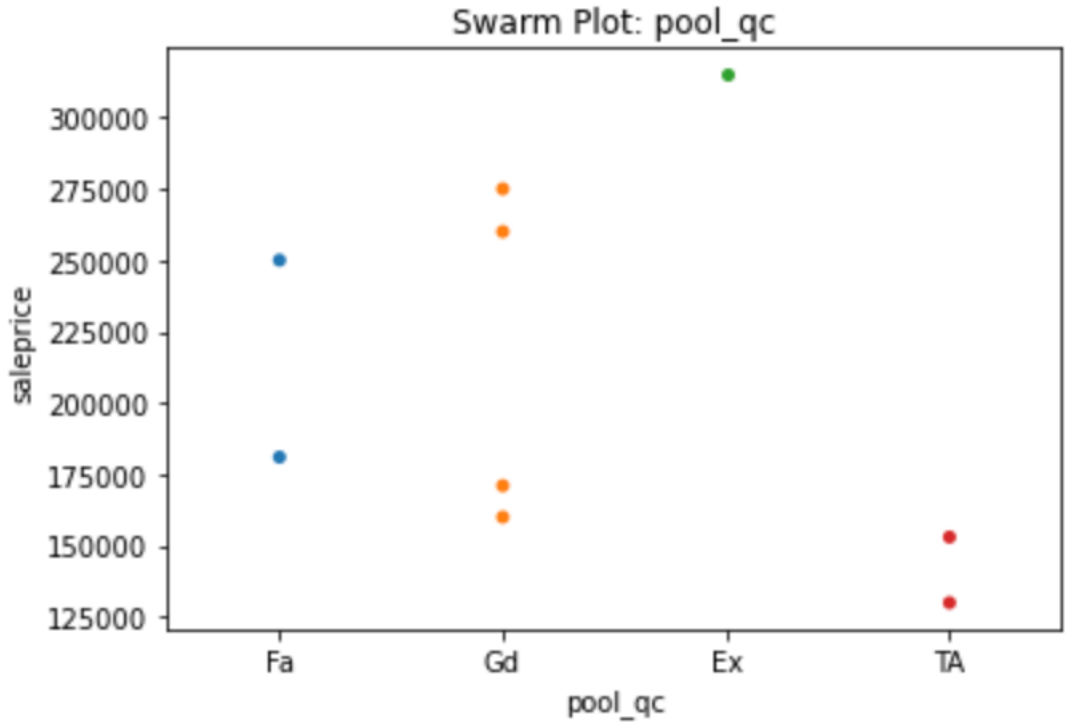
Final Predictions
- On Test dataset
- Submit Kaggle

Data Cleaning & EDA

Null Values - Drop

- More than 95% missing values = drop

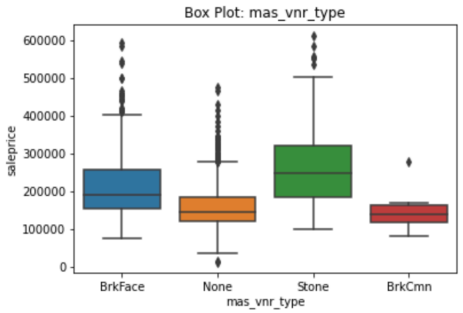
```
In [25]: plot_swarm(train,['pool_qc'], train['saleprice'])
```



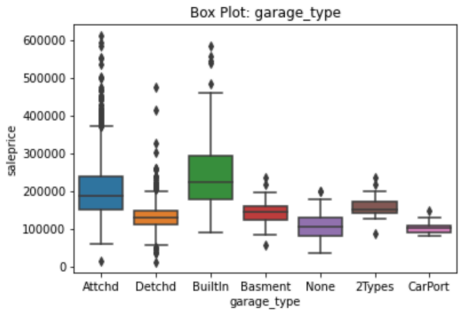
Null Values – Impute string

- For NA that indicates 'absence of feature' not a missing value.
- The boxplots below shows that filling 'None' clearly helps analyse whether not having the feature ('None') or having some variations of the features can drive saleprice.

```
In [38]: plot_box(train,['mas_vnr_type'], train['saleprice'])
```




```
In [39]: plot_box(train,['garage_type'], train['saleprice'])
```



Map Ordinal Categorical Values

- **For ordinal values** we need to map to a numerical value.
- In this case we will use the following assumption where 3 represents average/typical/as expected:

Score	Quality/Condition	Basement Finished Area	Basement Exposure
5	Excellent		
4	Good	Good Living Quarters	Good Exposure
3	Typical/Average	Average Living Quarters/ Average Rec Room	Average Exposure
2	Fair	Below average living quarters	Minimum Exposure
1	Poor	Low Quality or Unfinished	No Exposure
0	absence of feature	No Basement	No Basement

Correct Data Type:

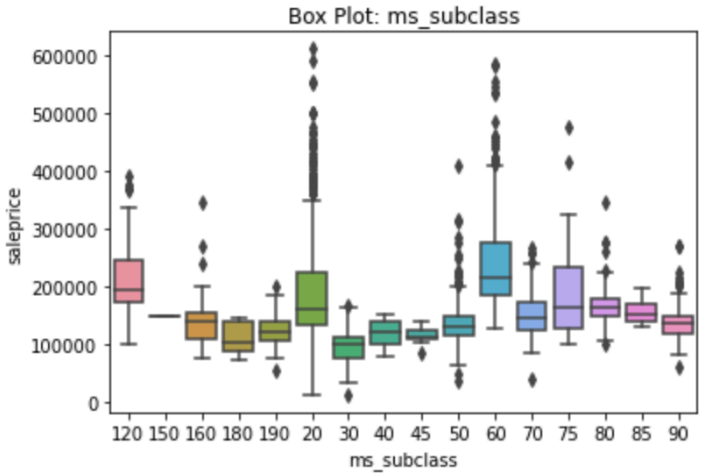
MS Subclass this is currently represented as integers with varying values (020, 030, ..., 180, 190) while the values are actually nominal (i.e. 020 is a particular class, and does not have less value than 030, etc). This should be treated as string.

In [84]:
```
# change ms_subclass from int to string
train['ms_subclass'] = train['ms_subclass'].map(lambda x : str(x))
test['ms_subclass'] = test['ms_subclass'].map(lambda x : str(x))
```

In [85]:
```
train.ms_subclass.unique()
```

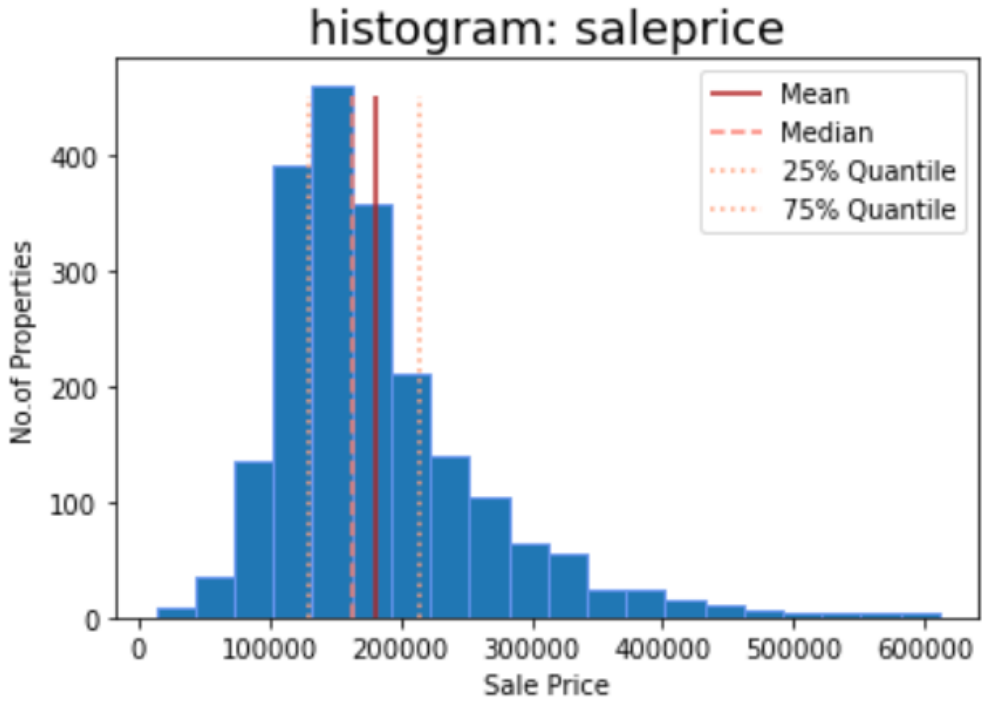
Out[85]:
```
array(['60', '20', '50', '180', '160', '70', '120', '190', '85', '30',
       '90', '80', '75', '45', '40', '150'], dtype=object)
```

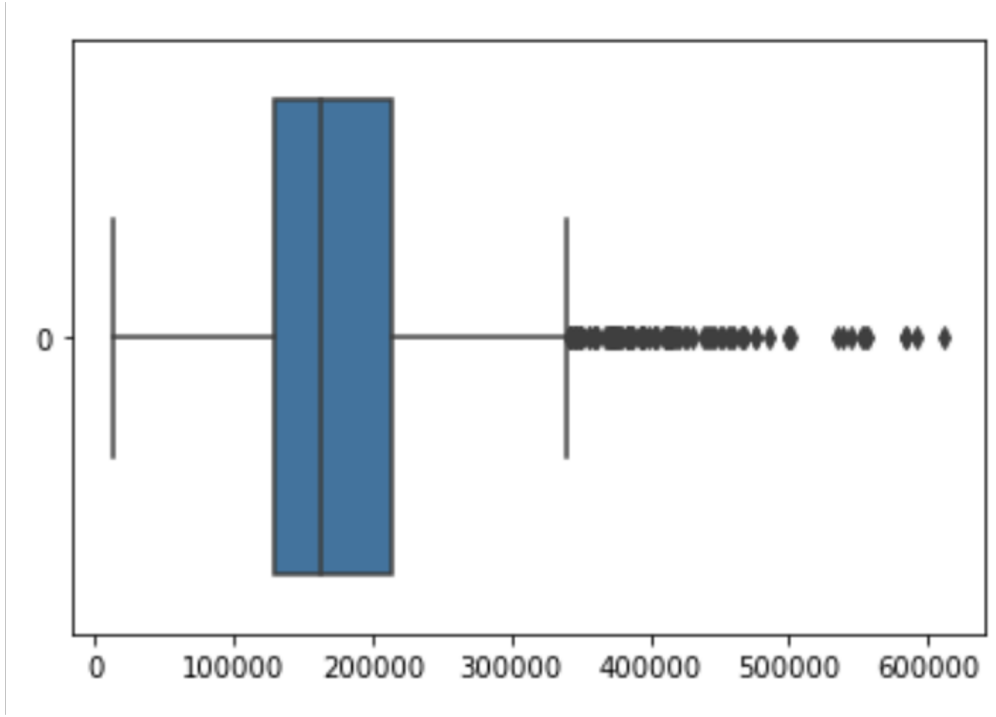
In [86]:
```
# PID and ms_subclass relation to saleprice
plot_box(train, ['ms_subclass'], train.saleprice)
```



Target Variable: SALEPRICE

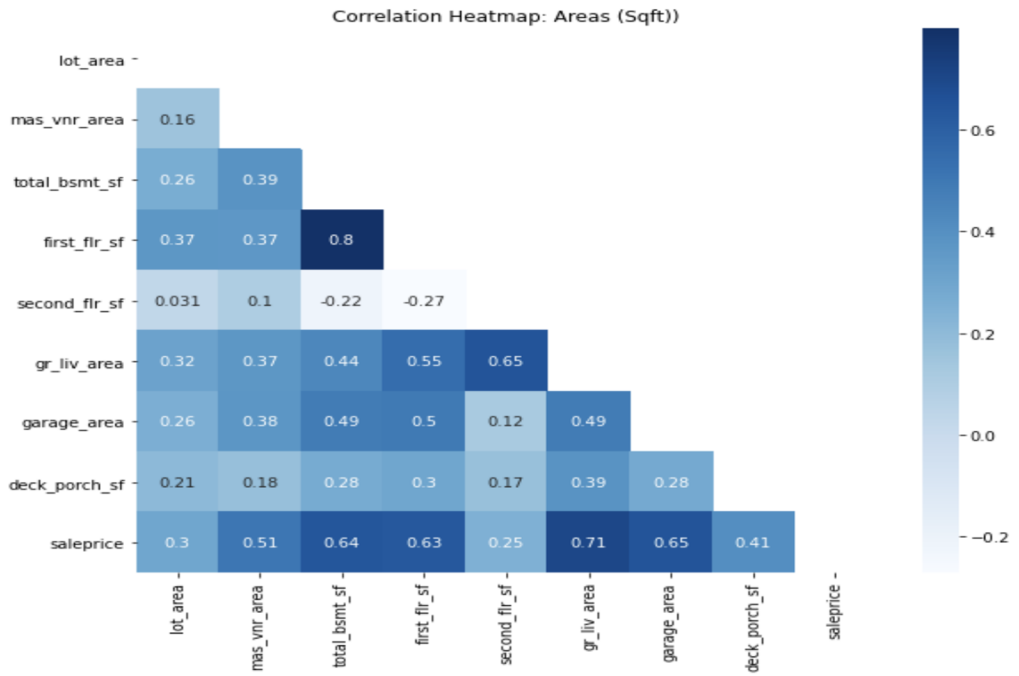
- The majority of houses are priced between **129,000 - 214,000** (Interquartile Range)
- Slightly right-skewed with many outliers (long-tail) towards higher values (max of 611,657), however, given limited data to predict house prices >350,000.

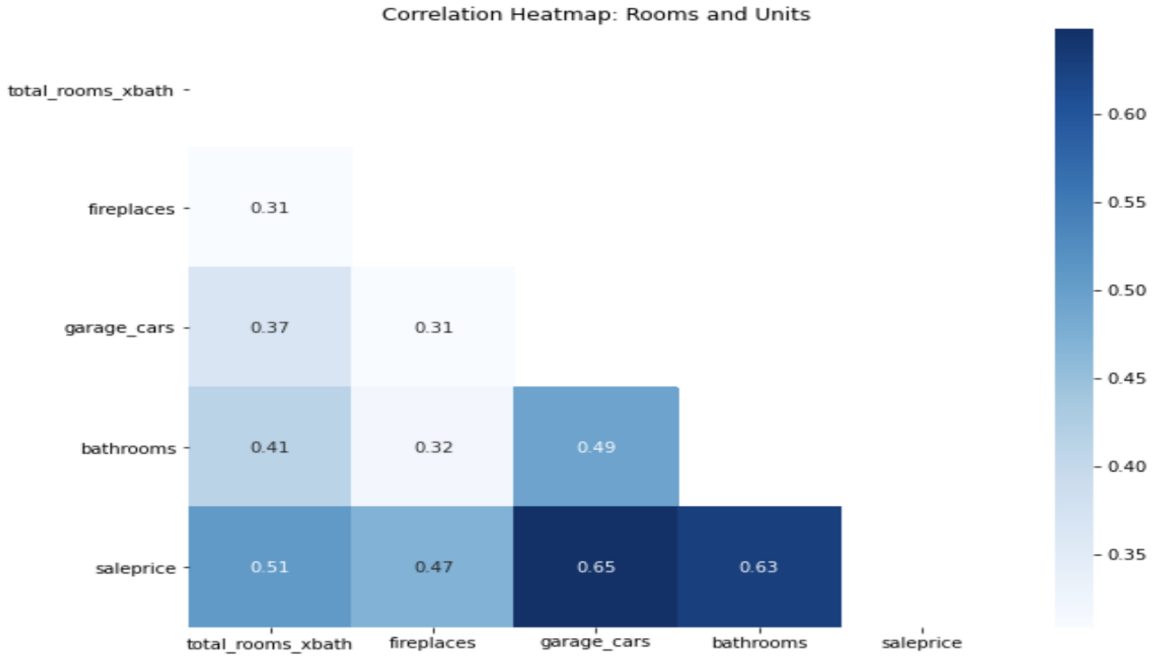




Numerical Values: Correlation Heatmaps

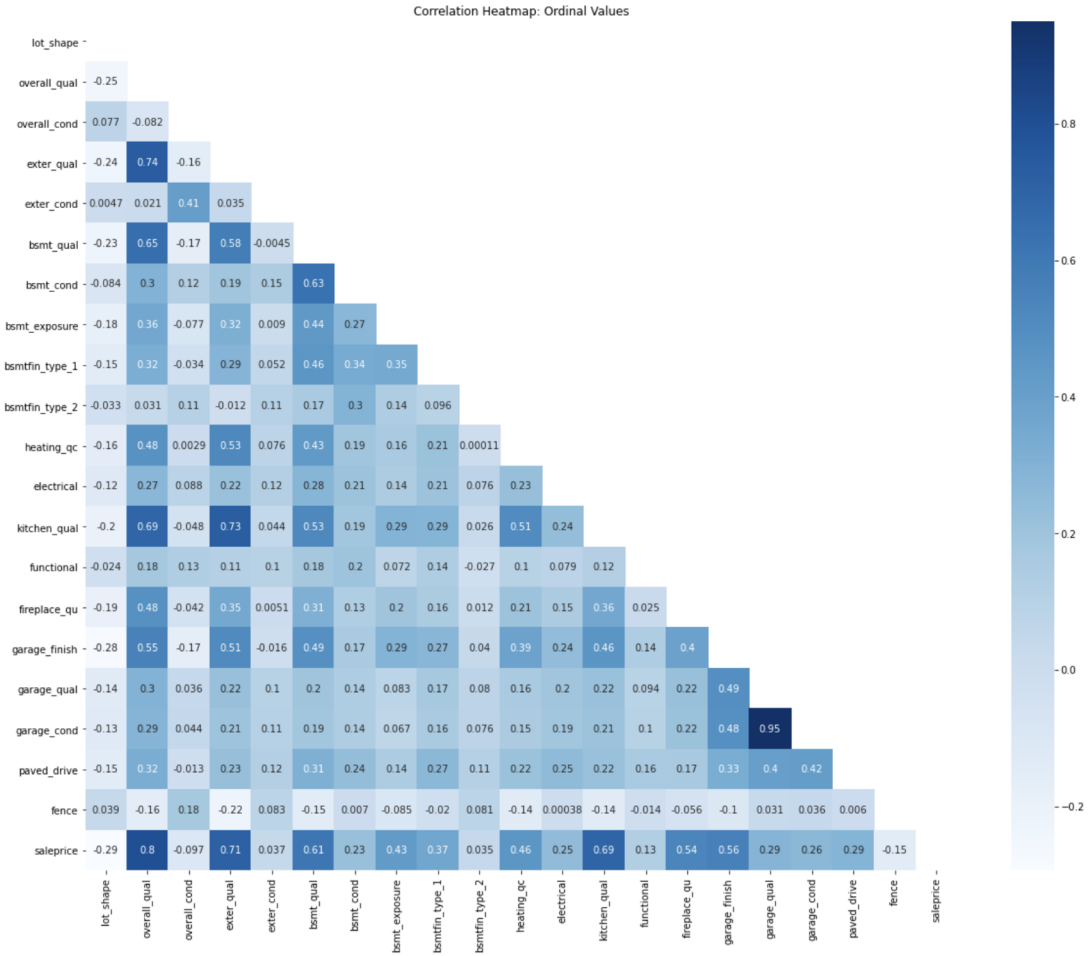
- **Separate heatmaps** with similar units of measure (area in sqft vs. discreet units):
- Drop : **low correlation to sale price** (i.e. between **-0.2** and **0.2**)
- Watch-out: **high multicolinearity between variables** (i.e. `first_flr_sf` vs. `total_bsmt_sf`) – keep at this point to be processed at modelling stage

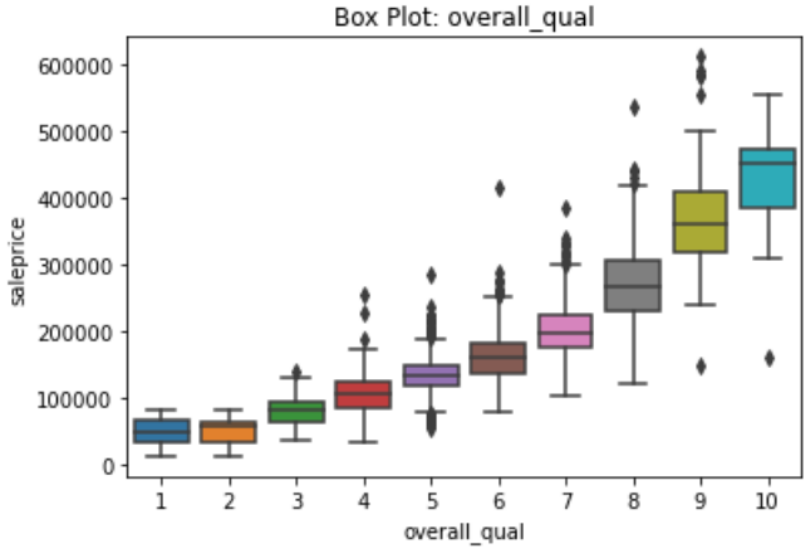




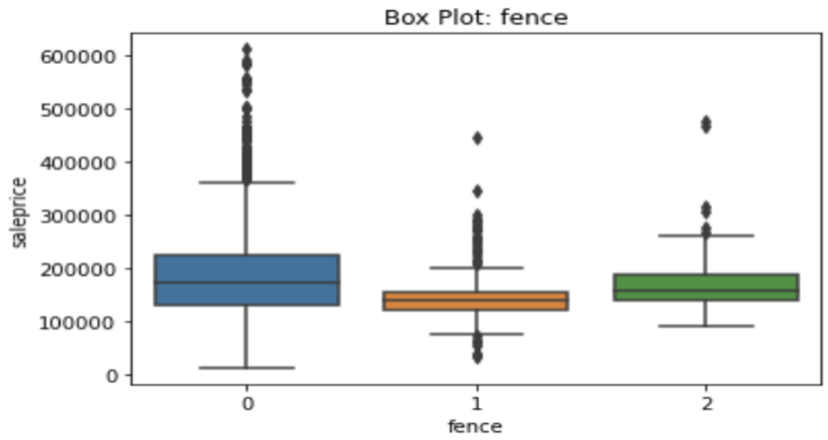
Ordinal Values: Correlation Heatmaps & Boxplots

- Plot correlation heatmaps & confirm with boxplots.
- Drop : low correlation to sale price (i.e. between -0.2 and 0.2)





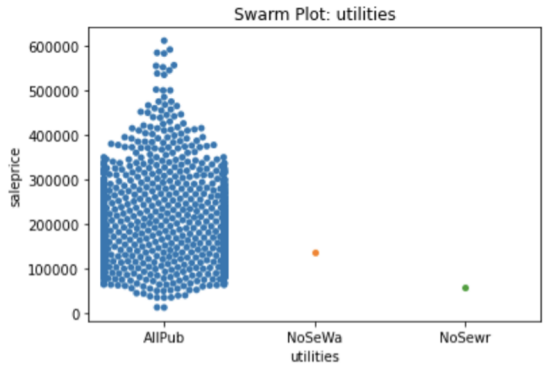
High Effect

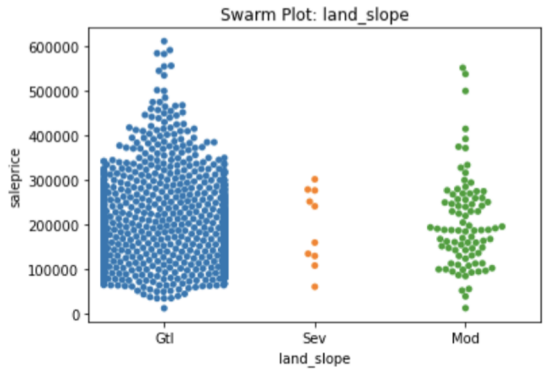


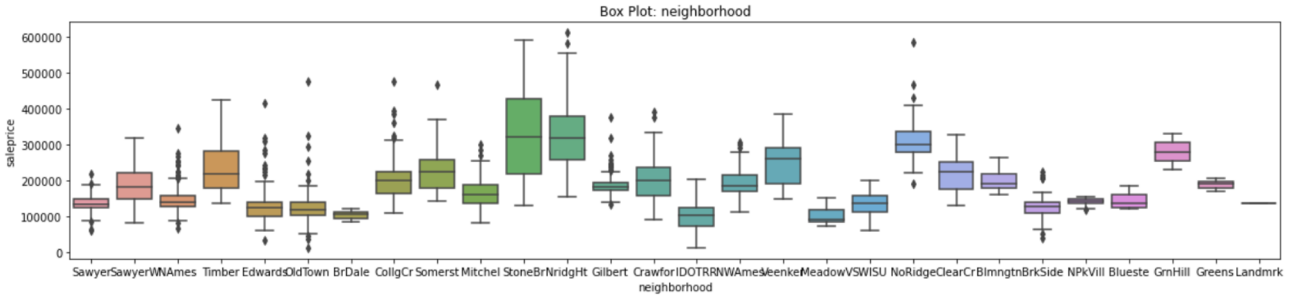
Low Effect

Categorical Values: Swarmplots & Boxplots

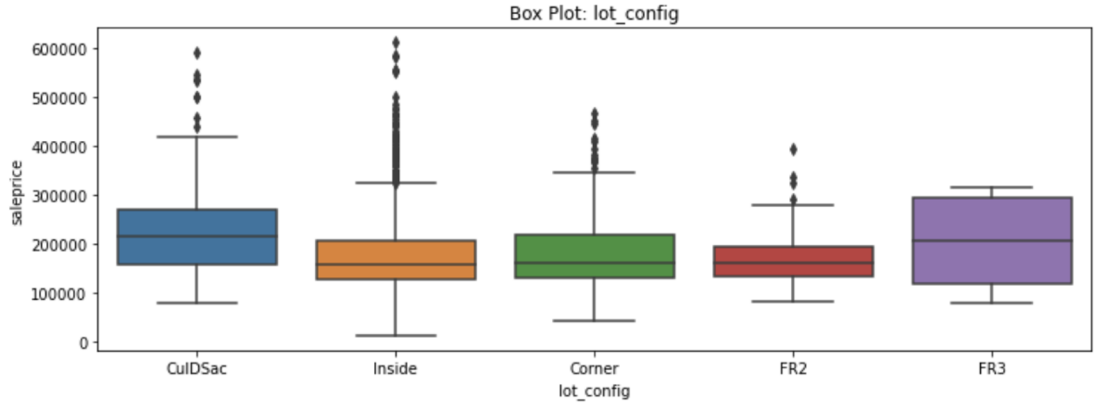
- **Swarmplots** to identify features with dominant values on one category (i.e. >90%)
- **Boxplots** to identify relationship with sale price (similar to oridnal values)
- **One-Hot Encoding** to convert categorical values into dummies (0,1) before modelling.







High Effect

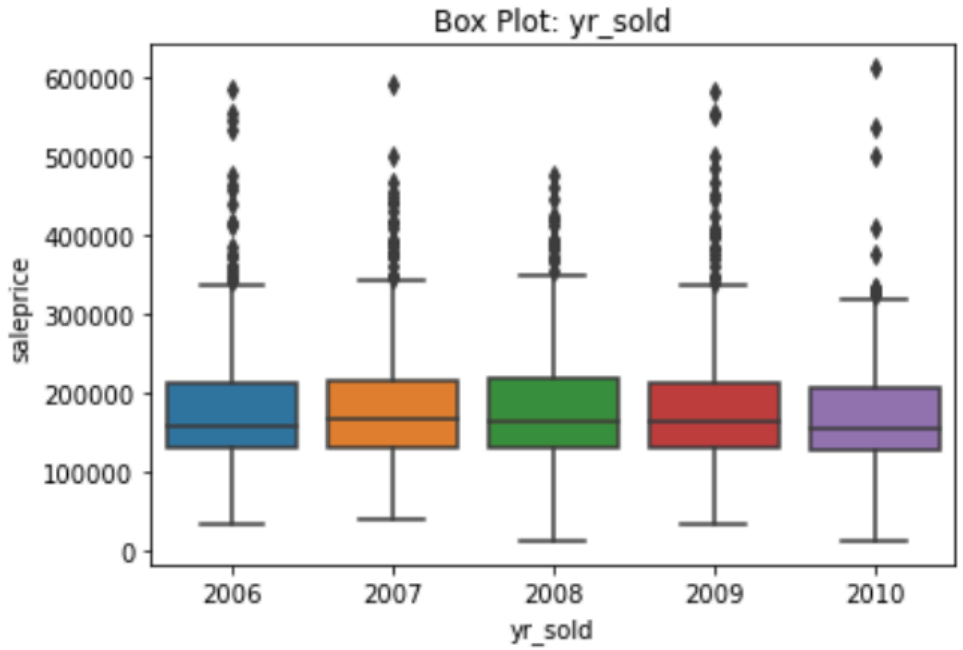


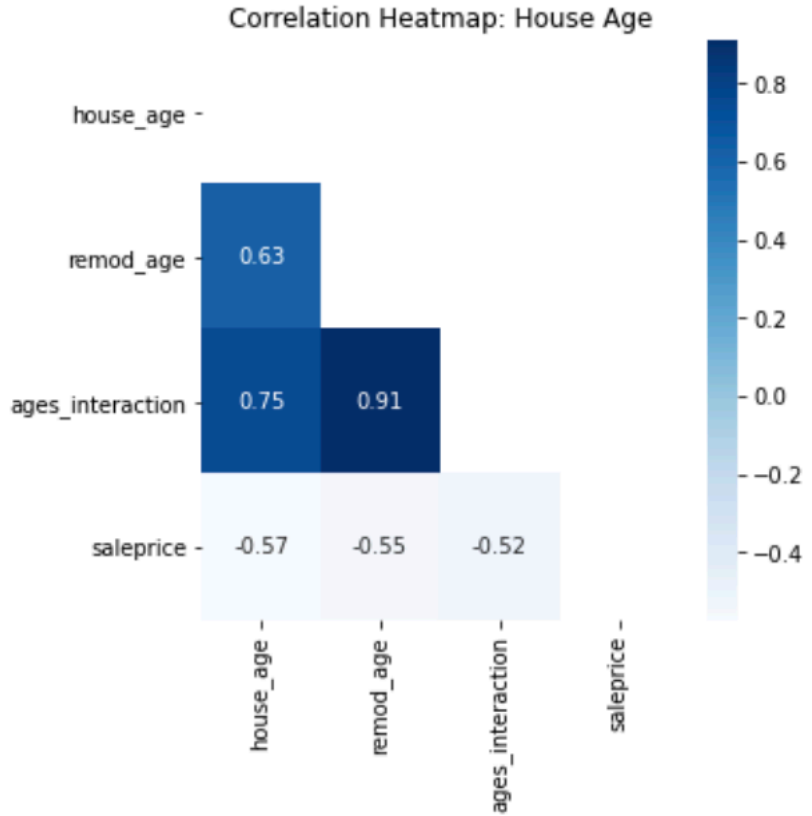
Low Effect

Feature Engineering

Ages instead of years

- Years sold does not have much relationship to price from the boxplot
- Converting years (years_built and years_remodelled/add) into ages (house_age, remod_age) would be more meaningful. Note, they have negative relationship to price (older house = lower price)





Number of bathrooms

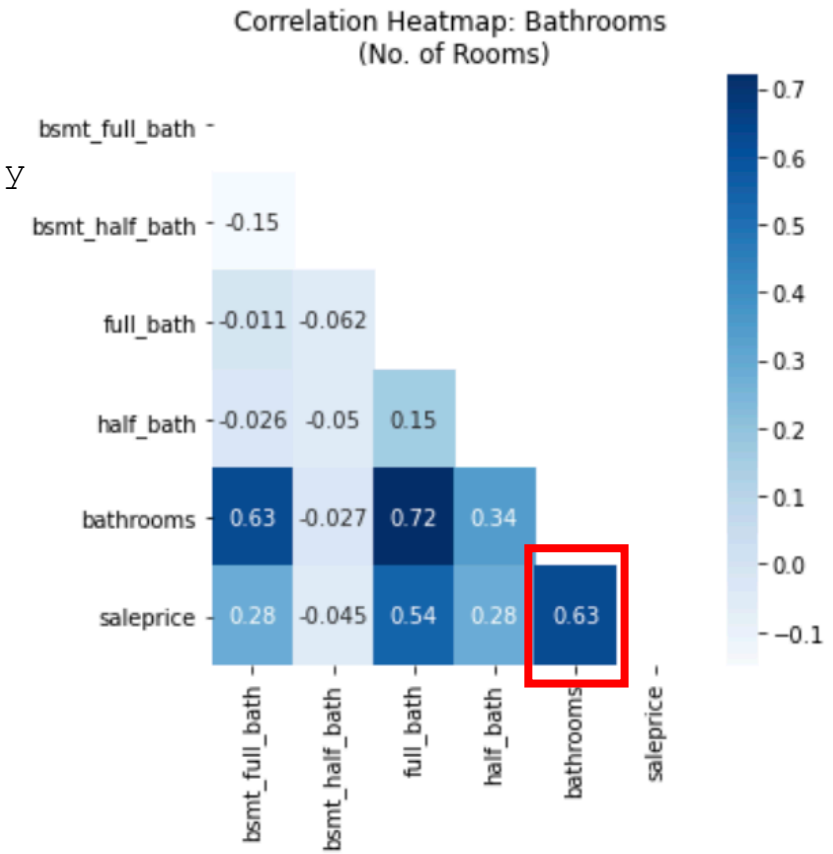
- **Bathrooms** Summarize all bathrooms into 1 feature with the following assumption
  - Full bath = 1
  - Half bath = 0.4 instead of 0.5 because half bath is usually without shower or bathtub, hence having 2 half baths does not equal 1 bathroom with shower
  - The combined total bathrooms has the highest correlation to saleprice, hece will keep this and drop the other bathrooms

```
In [108]: # add all bathrooms to 'bathrooms', half_bath will have the value of 0.4

train['bathrooms'] = train['bsmt_full_bath'] + (train['bsmt_half_bath']*0.4
                    ) + train['full_bath'] + (train['half_bath']*0.4)

test['bathrooms'] = test['bsmt_full_bath'] + (test['bsmt_half_bath']*0.4
                    ) + test['full_bath'] + (test['half_bath']*0.4)
```

```
In [109]: train[['bsmt_full_bath', 'bsmt_half_bath', 'full_bath', 'half_bath','bathrooms']].head()
```

Out[109]:

	bsmt_full_bath	bsmt_half_bath	full_bath	half_bath	bathrooms
0	0.0	0.0	2	1	2.4
1	1.0	0.0	2	1	3.4
2	1.0	0.0	1	0	2.0
3	0.0	0.0	2	1	2.4
4	0.0	0.0	2	0	2.0



Modelling

Data Preprocessing

(Shape of DataFrame)	Train	Test
Original	(2051, 81)	(879, 80)
After Data Cleaning & EDA (including map/encode categorical values)	(2049, 162)	(879, 153)

	Train (subset)	Validation	Test
Train-test-split	X = (1536, 161) Y = (1536,)	(513, 161) (513,)	(879, 153)

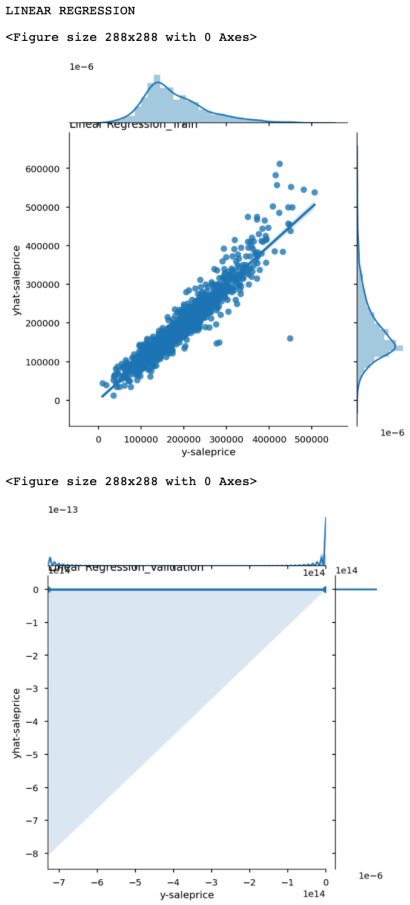
Regression Models

01	Linear Regression	• Most basic form of regression • Finds the correlation between dependent and independent variable • Tends to overfit
02	Ridge Regression (L2)	• Essentially Linear Regression + Error Penalty • Coefficients can tend to 0, but will never be 0 • Useful to keep all features
03	Lasso Regression (L1)	• Similar to Ridge Regression, but with a different penalty term • Coefficients can shrink to 0 • Useful for feature elimination
04	Elastic Net Regression	• Combines Ridge and Lasso • Ratio of Ridge and Lasso is controlled by L1 hyperparameter

Modelling Round 1:

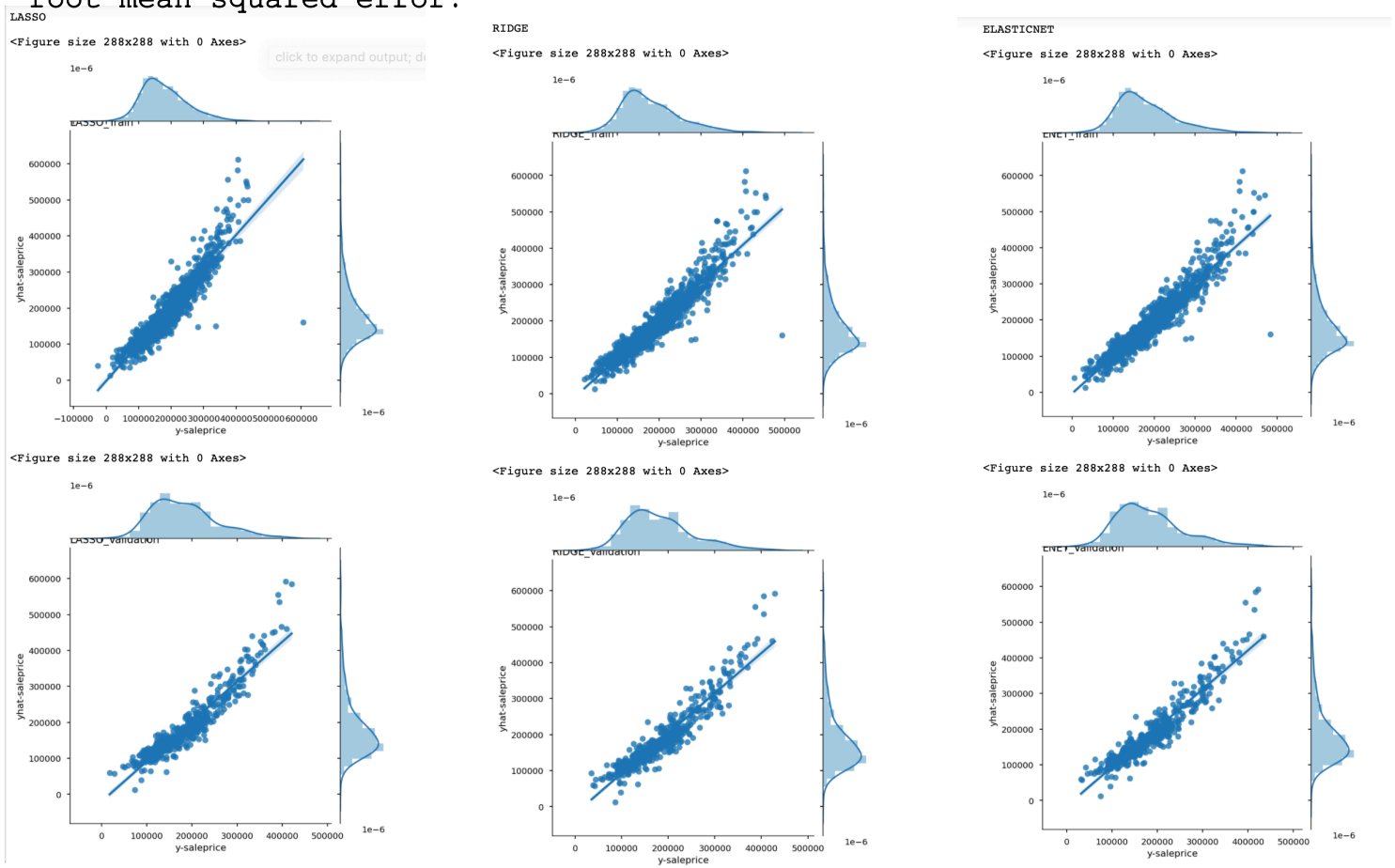
Linear Regression

- OVERFIT
  - Performed well on training data, but does not generalize well on unseen data



Lasso, Ridge, ElasticNet

- All performed better than Linear Regression in terms of performance on unseen data
- Of the 3, Lasso is closest at predicting validation data both in R2 and root mean squared error.



Modelling Round 1:

	model	dataset	mean_sq_error	rt_mean_sq_error	mean_abs_error	r2	r2_adj
0	linear_1	train-subset	576586451.12740	24012.21462	11634.99343	0.90399	0.89274
1	linear_1	validation	10368375472881922063781068 80.00000	32199961914390.39844	15423.94337	-146054856924904480.00000	-213048680186755232.00000
2	lasso_1	train-subset	821030001.51396	28653.62109	13564.80696	0.86328	0.84727
3	lasso_1	validation	904960206.33814	30082.55651	15153.59557	0.87252	0.81405
4	ridge_1	train-subset	644253322.29345	25382.14574	11085.60851	0.89272	0.88015
5	ridge_1	validation	896254862.60394	29937.51597	14398.89607	0.87375	0.81584
6	enet_1	train-subset	625460904.66588	25009.21639	11162.13354	0.89585	0.88365
7	enet_1	validation	844226545.75782	29055.57684	15035.79970	0.88108	0.82653

Overfit!

Best at generalizing on unseen data

Review Lasso Coefficients & selected only 34.

- Zeroed-out 126 features
- Selected 34 features (drop 1 'id')

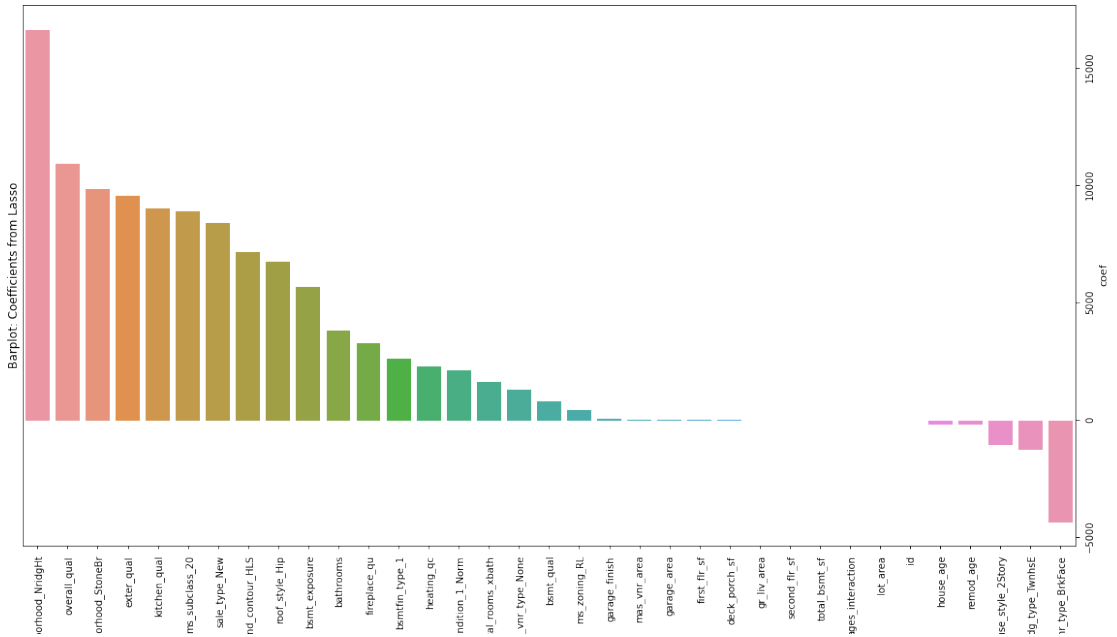
1. Features "zeroed out" = 126

In [58]:
```
# There are 126 features that has been zer

lasso_coefs[(lasso_coefs.abs_coef == 0)]
```

Out[58]:

	variable	coef	abs_coef
105	roof_style_Gable	-0.00000	0.00000
114	exterior_1st_CemntBd	0.00000	0.00000
122	exterior_1st_Wd Sdng	-0.00000	0.00000
121	exterior_1st_VinylSd	0.00000	0.00000
120	exterior_1st_Stucco	-0.00000	0.00000
...	...	...	...
63	neighborhood_CollgCr	-0.00000	0.00000
62	neighborhood_ClearCr	0.00000	0.00000
61	neighborhood_BrkSide	0.00000	0.00000
60	neighborhood_BrDale	-0.00000	0.00000
160	sale_type_WD	-0.00000	0.00000

126 rows × 3 columns

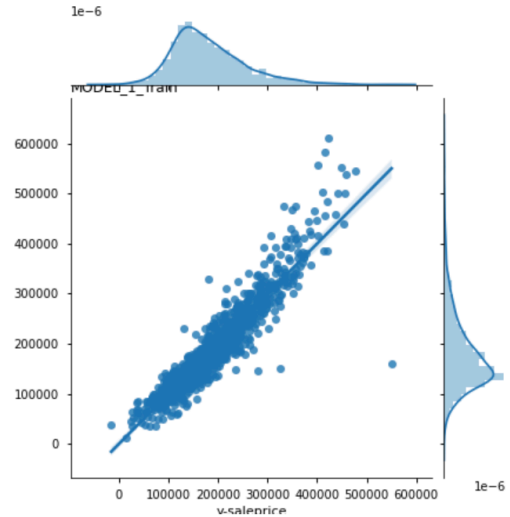


Final Model & Conclusions:

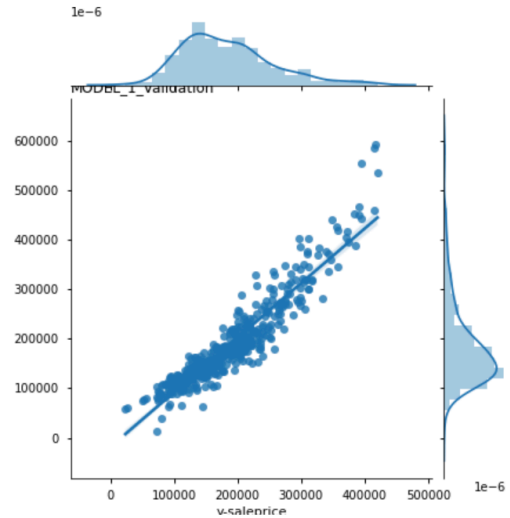
- Based on the problem statement, we have identified the regression model that can predict 'saleprice' of a house in Ames Iowa. The model is based on 34 the following features.

- This model performs well agaist both training & validation datasets, hence can generalize predictions on unseen data.

- Performs well on both train & validation data
  - **Model 1 Training Date**



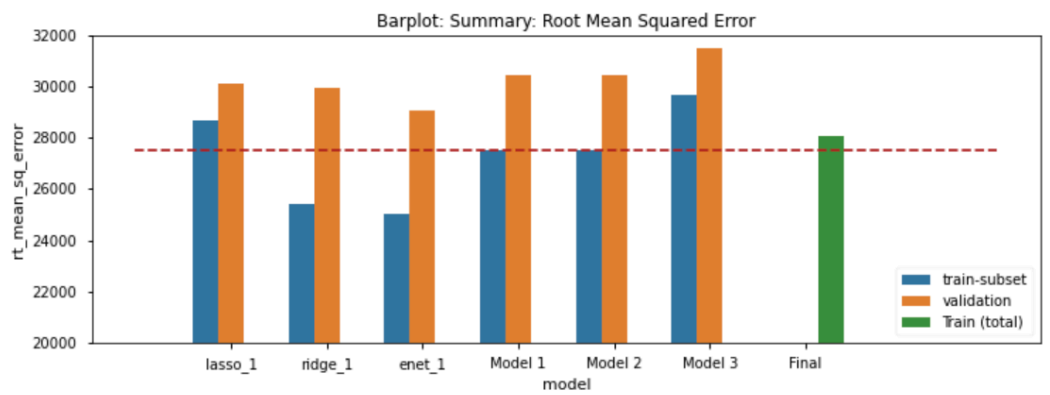
  - **Model 1 Validation Data**



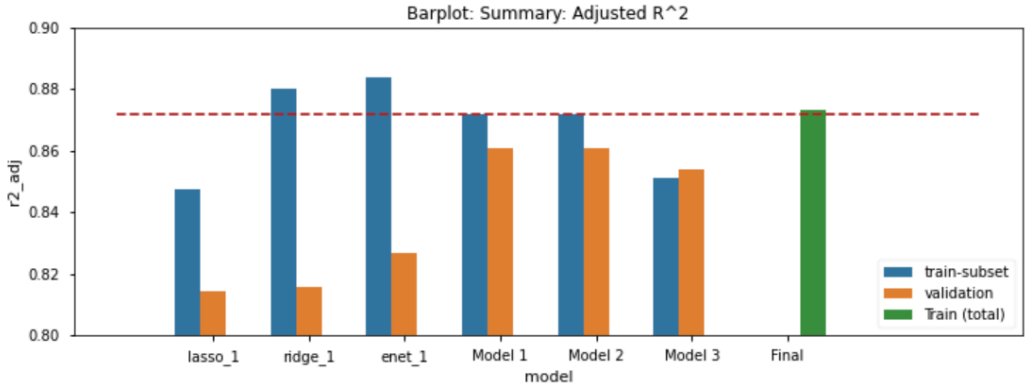
Final Model & Conclusions:

Most generalised scores on both accuracy (R2) and error (Root mean square error)

- Low Error (Root Mean-Squared Error)
  - **Model 1** has the best bias-variance trade-off. Although Rigde & ElasticNet has lower error in training, but higher in validation

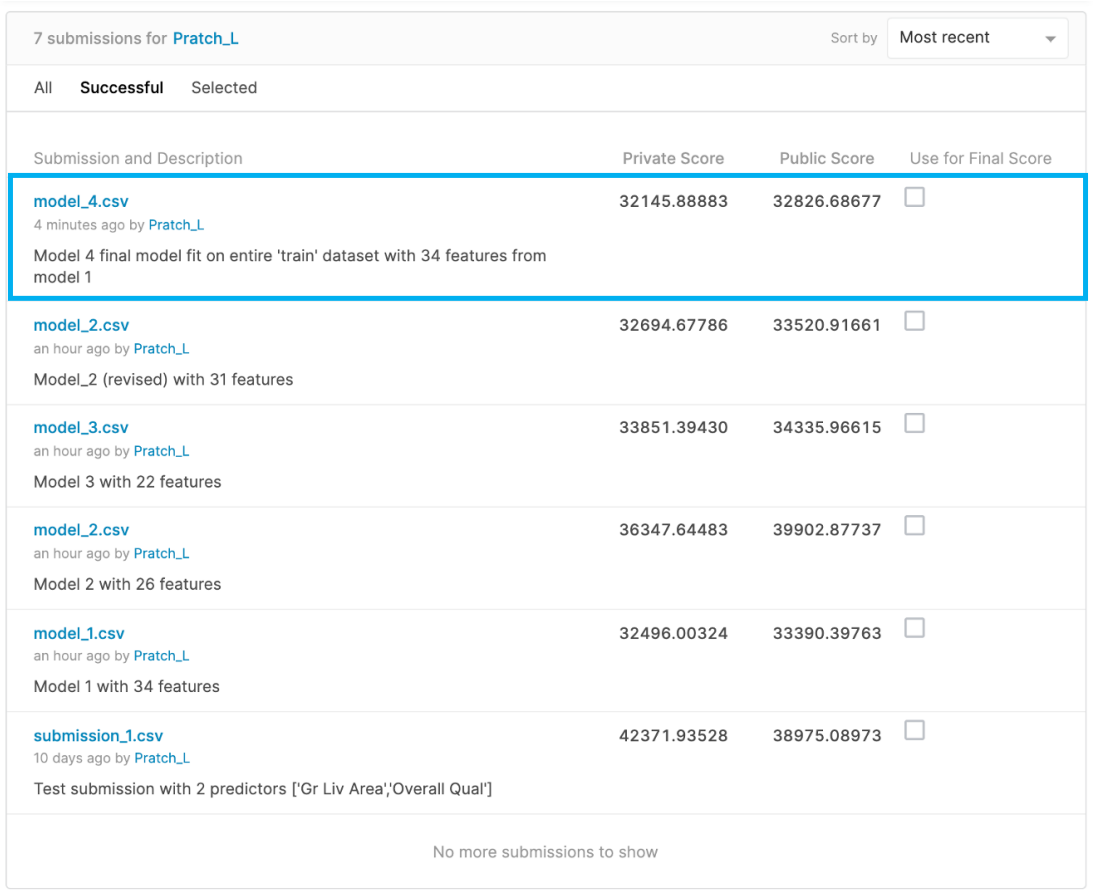


- High Accuracy (Adjusted R-square)
  - **Model 1** has the best bias-variance trade-off. Although Rigde & ElasticNet has higher R2 in training, but lower in validation



Kaggle Submission

- Based on Model 4 : Fit final features from Model 1 on the entire train dataset, then predict test dataset.



7 submissions for Pratch_L — Sort by: Most recent

All | Successful | Selected

Submission and Description	Private Score	Public Score	Use for Final Score
model_4.csv 4 minutes ago by Pratch_L Model 4 final model fit on entire 'train' dataset with 34 features from model 1	32145.88883	32826.68677	☐
model_2.csv an hour ago by Pratch_L Model_2 (revised) with 31 features	32694.67786	33520.91661	☐
model_3.csv an hour ago by Pratch_L Model 3 with 22 features	33851.39430	34335.96615	☐
model_2.csv an hour ago by Pratch_L Model 2 with 26 features	36347.64483	39902.87737	☐
model_1.csv an hour ago by Pratch_L Model 1 with 34 features	32496.00324	33390.39763	☐
submission_1.csv 10 days ago by Pratch_L Test submission with 2 predictors ['Gr Liv Area','Overall Qual']	42371.93528	38975.08973	☐

No more submissions to show