Machine Learning Final Project (Group 7)
House Prices - Advanced Regression Techniques

R08222035, Yan-Cheng Wei
B07901169, Tsung-Huan Yang
R08921A16, Shih-Hsien Chuang

January 16, 2021

1 Kaggle Score

Time	Kaggle Score (RMSE)	in-class ranking	overall ranking
before 1/16 23:59	0.11232	**1**	160 (top 2.8 %)
before final presentation	0.11753	3 (or 4)	280

2 Introduction and Motivation

這次 final project 的題目為:給定一 dataset，其中有 80 個 feature ，試著以此去推測出該房子在售出時的價格，為甚麼會選擇這個題目，是因為買房子這件事情對於一般人來說是一筆不小的開銷，因此在下手購買之前必須要審慎評估其價值，隨著現在房市資料的公開普及，我們有越來越多的資訊可以去推測一個物件他所擁有的價值，因此若能夠設計出一個能準確評估一間房子他應該有的價格的 model ，不論是拿來議價或著是依此決定是否要下手購買，都會是有很大幫助的。

2.1 Dataset

training dataset包含1460個data，每個data有80個features和他們的saleprice (所以總共是81個properties)。testing dataset則包含1459的data，每個data有80個feature。我們的目標就是用這80個feature去預測他們的saleprice.

2.2 Metric

最後得到的分數是預測分數和ground truth (未公開)之間的root mean square error (RMSE)。分數愈低，排名越高。Strong baseline是0.16; simple baseline是0.18。我們分數是0.11232。

3 Data Prepossessing/Feature Engineering

做了一些資料分析後，發現data有一些問題，像是非常多的missing values (數量列在下面)。

```
PoolQC         2908   object
MiscFeature    2812   object
Alley          2719   object
Fence          2346   object
FireplaceQu    1420   object
LotFrontage     486  float64
GarageFinish    159   object
GarageQual      159   object
GarageCond      159   object
GarageYrBlt     159  float64
GarageType      157   object
BsmtExposure     82   object
BsmtCond         82   object
BsmtQual         81   object
BsmtFinType2     80   object
BsmtFinType1     79   object
MasVnrType       24   object
MasVnrArea       23  float64
MSZoning          4   object
BsmtFullBath      2  float64
BsmtHalfBath      2  float64
Functional        2   object
Utilities         2   object
GarageArea        1  float64
GarageCars        1  float64
Electrical        1   object
KitchenQual       1   object
TotalBsmtSF       1  float64
BsmtUnfSF         1  float64
BsmtFinSF2        1  float64
BsmtFinSF1        1  float64
Exterior2nd       1   object
Exterior1st       1   object
SaleType          1   object
There are 34 columns with missing values
```

除此之外，feature數目(80)相比data(1460)太多了。為了更有效率運用，避免overfitting，必須做非常sophisticated feature engineering。主要而言，我們做了以下事情：

(1) Missing values:

針對一些通用性的 feature (eg. Functional, Electrical)，我們將他的 missing value 都設定成 standard values (standard value 可以由 kaggle 的 data description 得到)，例如廚房的品質就設定為 typical / average。

若是一些不一定每間房子都會有的 feature (eg. garage、basement、poolQC、fence)，就會將他的 missing value 用 0 或是 None 來取代。例如若 garagetype 為 NA 代表該棟房子沒有車庫，其他有關garage的feature都會是NA代表不存在，此時就可以把他們都設為 0 或 None。

如果有些 feature (X) 與其他 feature (Y) 是有高度相關性的話 (eg. MSZoning 與 MSSubClass，LotFrontage 與 Neighborhood, see Fig.1 left subgraph)，那就會按 Y 分組，再用衆數去決定 missing value X 的值。其餘的數值就使用衆數去代替。之所以不直接用 原本 X 的衆數取代是因為 X 的衆數不一定是 X 最有可能的值，因為 X 會跟其他feature有相關性 [4]

(2) skewness:

由於有些 feature 的數據嚴重偏離正態分布，所以我們針對偏離太多的 feature 使用 scipy.special.boxcox1p 去做修正

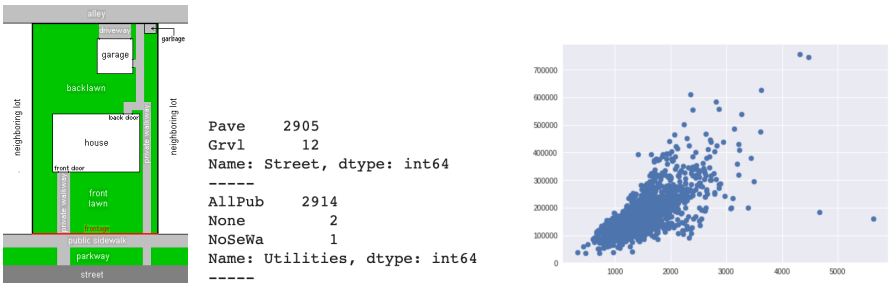

Figure 1: Left: LotFrontage 與 Neighborhood 是有高度相關性的，因為通常在鄰近地區的房子類型相似，lot frontage也應該差不多。 Lot frontage就是建築和街道的交界（紅線）。[9]; Middle: Street和Utilities的種類比例相當懸殊，對於training 沒有幫助; Right: Saleprice v.s. GrLivArea。 當GrLivArea$>$ 4000，點的趨勢不符合其餘點的行為，因此視為outliers. [4, 7]

(3) concatenating features to one features:

某些 features 性質幾乎一樣我們將他合併成一個 feature 以降低feature的數量提升model的效率，例如: 新的feature totalSF = TotalBsmtSF + 1stFlrSF + 2ndFlrSF。

(4) dropping features:

觀察 data 發現 Utilities, Street, PoolQC 3個 features 的子種類 比例相差懸殊，也就是幾乎每一個 data 都有相同的Utilities, Street, PoolQC，對於訓練 MODEL不會有幫助，所以可以刪除。請參考Fig.1 middle subgraph.

(5) dropping inappropriate data:
刪掉不合趨勢的outliers。在此份training dataset中，我們參考[7, 5]，刪掉了a.data with GrLivArea $>$ 4000 (Fig.1 right subgraph) 和 b. data with index $= [30, 88, 462, 631, 1322]$ 。另外，我們 刪掉零太多的features，避免overfitting [4]。

4 Model Description

首先，我們遵從上述的feature engineering並使用one-hot encoding。feature 數量大約是340。 之後，我們的machine learning model如下

4.1 Submodels

我們首先使用了7個獨立的submodels做training。input 是經過one-hot encoding後的features，output則是saleprice的prediction。我們train models使得預測值接近ground truth。7個獨立的submodels分別是

- Kernel ridge regression
- Lasso regression

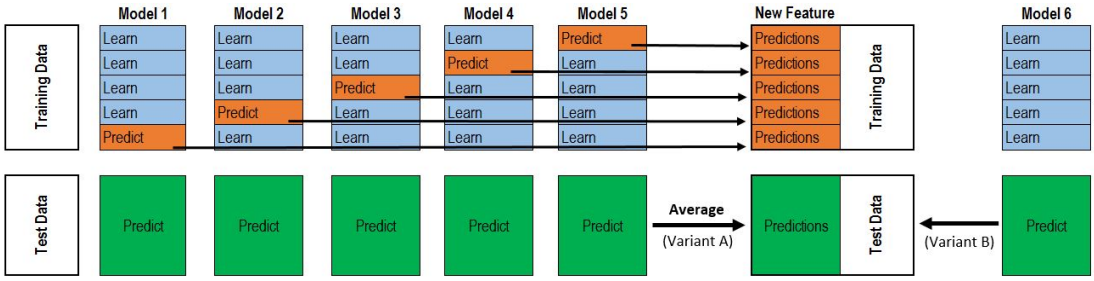

Figure 2: stack regression visualize[3]

- Elastic Net regression
- Support vector regression (SVR)
- Light Gradient Boosting Machine (Light GBM)
- gradient boosting regression
- Scalable Tree Boosting System (XGBoost)

4.2 Stacked Regression

經過這七個獨立子模型的訓練後，我們參照[6, 8]去build a stacked regression model。在training 時概念如下：

1. 首先，在training時，如果在train n 個submodels，就把data切成n份。

2. 在每一步，將其中n-1份做learning，並把其中一份做prediction。這裡有點像是n-fold cross validation.

3. 最後則用每個submodel的prediction做inputs，ground truth為目標output，去訓練新的一個model，在此稱為new model。

在testing時，在1. 2.時則是把所有data都拿去做prediction。在 3. 時則把所有submodel的prediction做平均，最後在餵進new model裡面，並用new model的output當作整個model的最終output。請參考示意圖 Fig.2。

這稱為stacked regression。 因為他利用到了每個submodel，他有可能表現得更好。[2]

4.3 Ensemble

我們參考[6, 8]，將得到stacked regression model和前起個submodel做線性疊加，得到一個ensemble，形成新的model。

4.4 Blending

參考到[7, 5, 1]，我們可以將好幾個都不錯的model一起做blending，也就是在做線性疊，並得到更好的model。舉例來說，假設我們由前一步得到k個ensemble

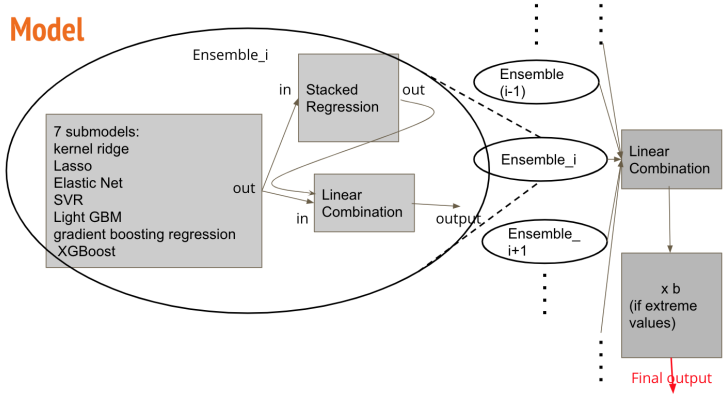

Figure 3: Overall Model Visualization

model, E_i, 我們可以

$$E_{blending}(x) = \sum_{i=1}^{l} c_i E_i(x) \tag{1}$$

當然在constraint

$$\sum_{i=1}^{k} c_k = 1 \tag{2}$$

下去最佳化。

4.5 Extreme values

因為之前遇到outliers問題，如果是training則把它去除。但是如果是在testing dataset，這樣的極端值會對RMSE造成很大的影響。因此，我們在最後針對預測出來saleprices> quantile (0.9999) 的data乘上一個小於一的factor (b)，並把b當作最佳化的參數之一，並output最終的prediction。

所有的步驟可以參考示意圖Fig.3。詳細的實驗佐證請見下一個章節。

4.6 Different Models

除了上述的Model (default) 之外，我們也嘗試做了別的model（總共五個）。舉例來說，我們現用PCA降維度在做後續的machine learning; 以及少使用一個submodel去做stacked regression。詳細資訊請見Table.3和Sec.5.3。

5 Experiment and Discussion

在這部分，我們根據一些實驗做討論。除了實驗之外，我們在8.2附上了所有data的分佈圖以供參考。

Case	Kaggle Score (RMSE)
With all feature engineering	0.11786
Without fixing skewness	0.1182
Without concatenating some features to a new feature	0.11866
Without dropping unbalanced data	0.11890
Without dropping outliers	0.11743
Without dropping the case with most of zeros (avoid overfitting)	0.12248

Table 1: Kaggle scores with different feature engineering cases

5.1 Feature Engineering

首先，我們對於feature engineering的效果做分析。我們陸續把feature engineering 的method省略掉，並觀察跑出來的Kaggle score。 Table.1 列了實驗結果。

基本上，做所有的feature engineering才能得到比較好的結果，說明大部分的methods都有效果。舉例來說，with all feature engineering 和 without fixing skewness相比，有做skewness的調整才會有比較好的成績，因此推得skewness對於training model有幫助。不少Kaggle競爭者也有得到類似的結論[7, 4]

然而，Without dropping outliers 竟然得到比 with all feature engineering 還要好一點點的分數。我們認為是這只是誤差。如果看training時去做cross validation的誤差(Table 2)，uncertainty就差不多$0.01 \sim 0.02$。在這個case，Without dropping outliers和with all feature engineering不超過0.005，因此我們認為這是源於training初始值random不同而造成非常小的差異。（在此我們用cross validation的uncertainty討論的原因是，Kaggle score的uncertainty需要耗費很多次繳交次數才能獲得，但是這麼做不切實際，會浪費很多上傳機會。）

5.2 Performances of different submodels (for single ensemble)

在這個部分，我們討論在一個ensemble裡面，不同submodel的效果。Table.2列了我們使用不同model的corss validation RMSE 以及 Kaggle score。

單論一個submodel的話，效果差不了多少（index $0 \sim 6$）。我們參考[6, 8, 7, 5]並使用stacked regression的結構。我們將submodel $0 \sim 6$ 當作子模型並訓練 Stacked regression，得到顯著的提升(Table.2的index 7)。

最後，參考[8]，將最後train出來的submodels和stacked regression做linera combination，Kaggle上得成績稍為變好了一點(Table.2的index 8)。

5.3 Performances of different ensembles and the final blending model

根據上面的章節，我們能夠組合出一個ensemble，我們稱這是default 的 ensemble (Ensemble E in Table.3)。之後，我們想了不同的想法，例如說利

index	model	cross validation RMSE when training	Kaggle Score (RMSE)
0	kernel ridge	0.1041 (0.0149)	0.12318
1	Lasso	0.1040 (0.0144)	0.12353
2	Elastic Net	0.1035 (0.0139)	0.12950
3	Support vector regression	0.1041 (0.0145)	0.12352
4	Light GBM	0.1076 (0.0183)	0.12349
5	gradient boosting regression	0.1092 (0.0153)	0.12785
6	XGBoost	0.1079 (0.0173)	0.12431
7	Stacked regression (use $0 \sim 6$)	X	0.11882
8	linearly combine submodels $0 \sim 7$	X	0.11786

Table 2: Kaggle scores and cross validation RMSE with different submodels

index	model description	Blending Ratio of the final ensemble model	Kaggle Score (RMSE)
Ensemble A	Simpler feature engineering (not define new features, not fix the skewness, not drop features, not delete the outliers)	0.25	0.11990
Ensemble B	Stacked regression does not include SVM, slightly different light gbm settings	0.05	0.11784
Ensemble C	Totally different feature engineering methods (See Appendix.8.1)	0.15	0.11898
Ensemble D	Apply PCA, reduce from dimension = 331 to 200, and then do the machine learning training	0.05	0.11798
Ensemble E	Default	0.50	0.11786

Table 3: Kaggle scores of different ensembles

index	model	Kaggle Score (RMSE)
0	$\text{Model}_{\text{Blending}}$	0.11736
1	$\text{Model}_{\text{Blending}}$, addressing the data with quantile > 0.9999	0.11232

Table 4: Kaggle score of blending models with and without addressing the extreme values

用principle component analysis (PCA)降低維度在training (Ensemble D)或是採用不同feature engineering (Appendix.8.1)，但Kaggle score都沒有明顯的提升(Table.3)

因此，我們參考第一此作業使用的blending model，也就是做出很多個model，再根據的他們的Kaggle score做線性疊加，組合出更好的model [1]。另外，[7, 5]也使用一樣的方法。

於是我利用Kaggle scorec算出他們的線性比例。不過考量到有些model彼此比較相近，所以組合出來的效果也不會比較好，所以最後也是手動調這些參數使得Kaggle score最高。最後結果的參數如Table.3所顯示，

$$\begin{aligned}\text{Model}_{\text{Blending}} = 0.25 \times \text{Ensemble}_A + 0.05 \times \text{Ensemble}_B + \\ 0.15 \times \text{Ensemble}_C + 0.05 \times \text{Ensemble}_D + 0.50 \times \text{Ensemble}_E\end{aligned} \tag{3}$$

得到稍微好一點點的Kaggle score (Table.4, index 0)。

最後，我們參考[7]的最終步驟，他們將data的quantile$> 0.99 (< 0.0042)$的點都乘以0.77 (1.1)。他們沒有明列原因，下面的討論串也有人對此感到困惑。我猜測的原因如下。因為在做training時有把outliers 去 除，所以對於極端值本身就預測的不準。而這篇文章的作者應該是利用這個方式把極端data暴力找出來並作調整。因此，我們也在final project的最後一步做這件事情。我們最佳化得到的結果是：將quantile> 0.9999的saleprices乘以0.25，並且得到傑出地提升(Table.4 index1)

6 Discussion

在這次的 final project 中，查了很多網路上許多的文章，也從中學習到了很多，最後所做出來的結果也不錯，在 presentation 中老師問了一個問題，就是會不會拿這個來當作在台灣買房子的依據，這件事情其實滿值得討論的，因為在這次所使用的資料並不是台灣自己的資料，我們認為若在台灣要做房價預測其實還需要考慮更多不同的方面，例如學區、附近是否有捷運等等的，而且這次的 dataset 中其實有滿大一部分是獨棟的房子，但在台灣相對來說獨棟的房子其實是比較少的，更多的是公寓或著是社區大樓，而若是住在大樓也有更多考慮的面向，是不是在頂樓、公設佔比等等的，那這些資訊可能就需要更深入的 feature engineering，所以我們在這次所做出來的 model 可能並不能完美的應用在台灣的情況。

另外要考慮的是題目的metric。因為Kaggle分數是RMSE，意思是說如果價格非常高的住宅，如果預測稍微有點失誤(error rate)，就會受到很大的penalty。

反過來說，對於價格很低的住宅，即便預測值超過ground truth的50%, 100%，很可能也受到不高的懲罰。因此，此model適不適用也取決於我們關心的價格：如果我要買低價住宅，那這個model很可能表現很差，即便Kaggle score (RMSE)非常好。

7 Conclusion

首先，我們利用feature engineering將data做更有效率的運用。大部分的feature engineering methods 都有作用(Table.1)。另外，我們參考非常多篇文章，build出我們的model。概念就是結合stacked regression和blending model，並且在最後處理極端值。model裡的每個步驟都有利用實驗證明其效果。我們也利用這個model拿到0.11232 (# 1 in-class ranking)的分數。

References

[1] @ASZWRvp7SjOEdYLqF3JYdg. Hw1 - handwritten assignment. Online. https://hackmd.io/@ASZWRvp7SjOEdYLqF3JYdg/Bk6mgvOHv.

[2] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, July 1996.

[3] Faron. Getting started. Online. https://www.kaggle.com/getting-started/1815#post103381.

[4] hemingwei. Top 2% from laurenstc on house price prediction. Online. https://www.kaggle.com/hemingwei/top-2-from-laurenstc-on-house-price-prediction.

[5] Alex Lekov. Stacking lr&gb = top1 [0.10649] {House Prices} v44. Online. https://www.kaggle.com/itslek/blend-stack-lr-gb-0-10649-house-prices-v57/data?scriptVersionId=11189608.

[6] Amal Nair. Guide to implement stackingcvregressor in python with machinehack's predicting restaurant food cost hackathon. Online. https://analyticsindiamag.com/stackingcvregressor-in-python/.

[7] Nanashi. #1 house prices solution [top 1%]. Online. https://www.kaggle.com/jesucristo/1-house-prices-solution-top-1.

[8] Sergine. Stacked regressions : Top 4% on leaderboard. Online. https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard.

[9] Wikipedia. Land lot. Online. https://en.wikipedia.org/wiki/Land_lot.

8 Appendix

8.1 Alternative way for feature engineering

我們參考[4]。 他的步驟是

1. 去除outliers，fix the skewness of saleprices

2. concatenate new features

3. 找出PoolArea > 0和 PoolQC是NA，用一些domain knowledge 推出PoolQC應該的數值。

4. 對於Basement系列

$$(basement_columns = [' BsmtQual',' BsmtCond',' BsmtExposure', \\ ' BsmtFinType1',' BsmtFinType2',' BsmtFinSF1',' BsmtFinSF2',' BsmtUnfSF',' Tota \tag{4}$$

，做一系列分析。像是drop掉超過一半都是NA的data，並用domain knowledge推出應該的數值。

5. 發現其中有一項data是

$$features['GarageYrBlt'] == 2207 \tag{5}$$

明顯出錯，車庫建造年份不可能等於2207。所以直接改成 2007 6. fix 所有data的skewness

7. drop unbalanced features.

8. delete outliers, 刪掉零太多的features，避免overfitting。

因此，和在本文講到的feature engineering大部份相同。主要的差別在於多做了3. 4. 5. ，也就是用了常識和domain knowledge做一些探討。

8.2 Data distribution visualization

圖中(Fig.4)可以看到，大致上testing data抓住training data的趨勢。

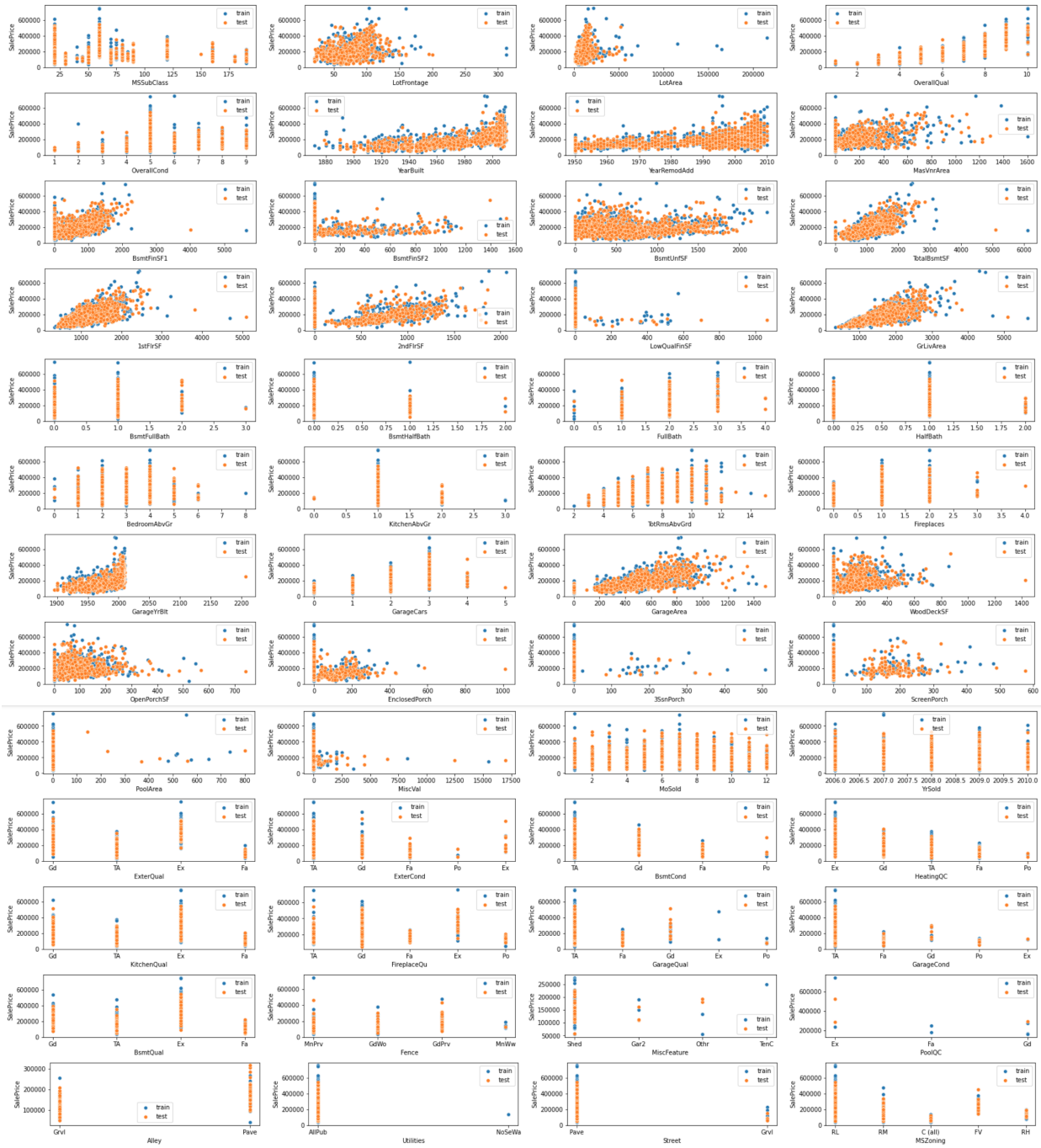

Figure 4: Data Visualization, including training and testing datasets.