Group Project Report
COMP4433 Data Mining and Data Warehousing

House Price Prediction
Group Ask Poe

LAI Ka Chung 22080062d
LI Chun Yin 21079842d

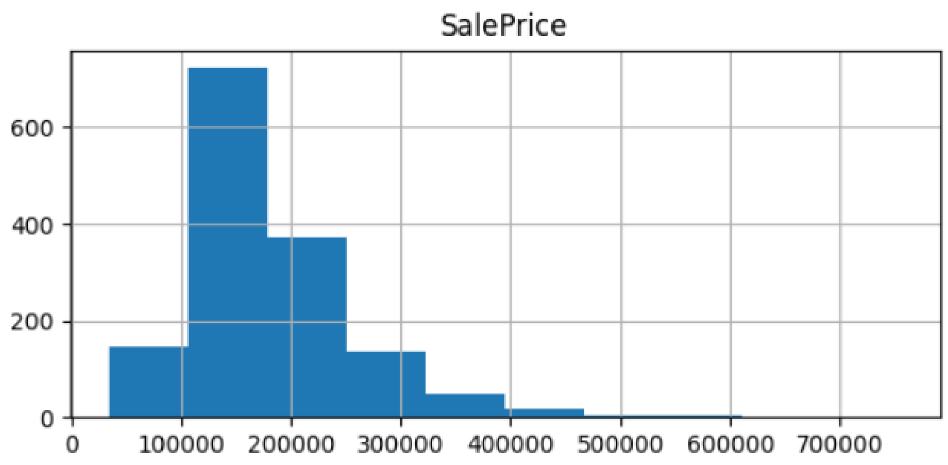
Jyupter Notebook in Kaggle

1. Preview of data

a. Data overview (distribution of data set and describe the structure of data)

The dataset provides 1460 samples, each representing a housing unit with 81 features. The 81 features are classified as 43 object attributes and 37 numerical attributes, including 3 float types and 34 integer types. In this dataset, SalePrice is considered as the target prediction variable.

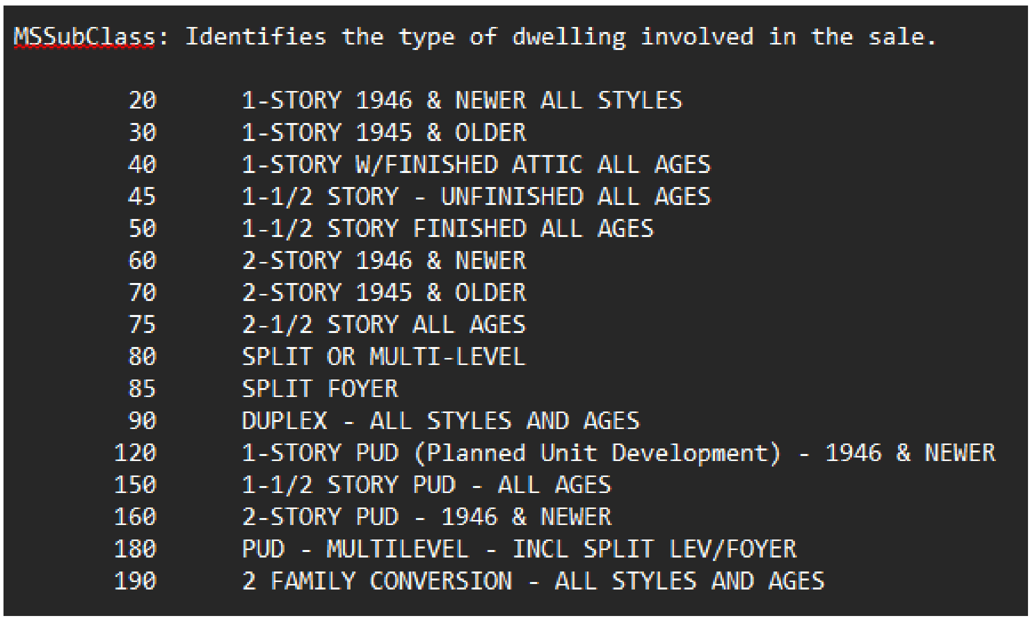
b. Review the target SalePrice



The distribution of target salePrice is heavily right-skew, with the long tail extending towards higher prices. Most houses are concentrated in low-price rangers, with few expensive outliers pulling the distribution.

c. Non-numeric predictors

There are three non-numeric predictors in the dataset. The number in the attribute doesn't mean the number directly; it is a category. Also, YrSold and MoSold don't have a numerical relationship with house prices.

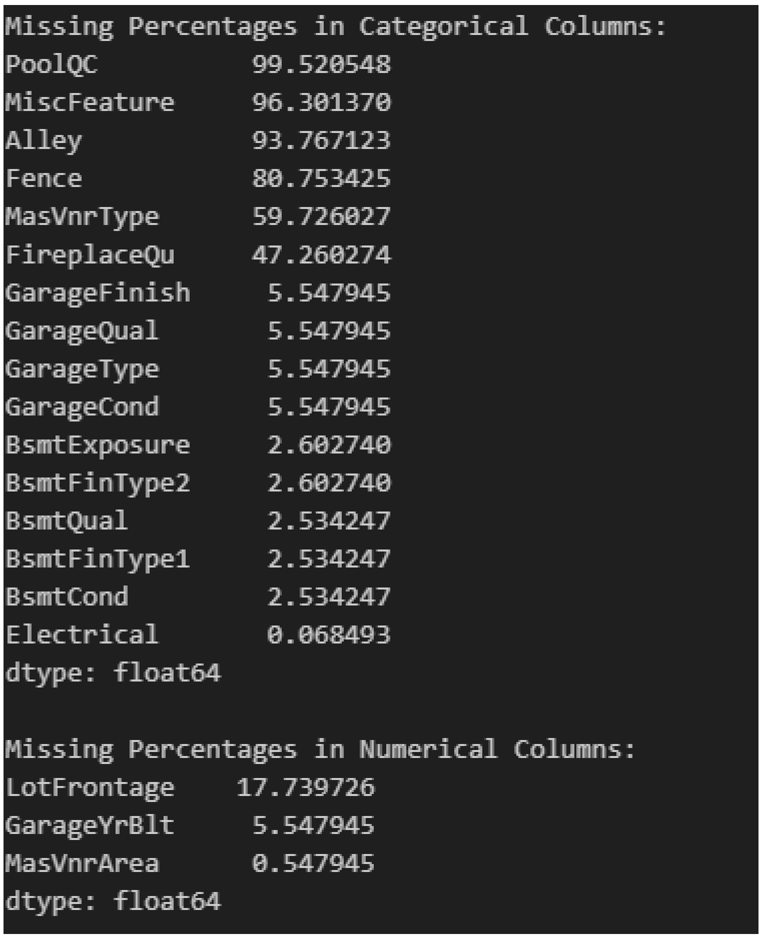


```
MSSubClass: Identifies the type of dwelling involved in the sale.

        20      1-STORY 1946 & NEWER ALL STYLES
        30      1-STORY 1945 & OLDER
        40      1-STORY W/FINISHED ATTIC ALL AGES
        45      1-1/2 STORY - UNFINISHED ALL AGES
        50      1-1/2 STORY FINISHED ALL AGES
        60      2-STORY 1946 & NEWER
        70      2-STORY 1945 & OLDER
        75      2-1/2 STORY ALL AGES
        80      SPLIT OR MULTI-LEVEL
        85      SPLIT FOYER
        90      DUPLEX - ALL STYLES AND AGES
       120      1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150      1-1/2 STORY PUD - ALL AGES
       160      2-STORY PUD - 1946 & NEWER
       180      PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190      2 FAMILY CONVERSION - ALL STYLES AND AGES
```

d. Missing data analysis

Among these attributes, we noticed that there are two types of missing value categories: structural missing values and Information missing values. First, a structural missing value is missing data that represents the absence of features. For example, GarageArea = NA (the house has no garage), PoolArea = NA (the house has no pool). Second, information missing values are data that should exist but is missing. For numerical columns, use LotFrontage and MasVnrArea—categorical columns, like Electrical, MasVnrType.



```
Missing Percentages in Categorical Columns:
PoolQC          99.520548
MiscFeature     96.301370
Alley           93.767123
Fence           80.753425
MasVnrType      59.726027
FireplaceQu     47.260274
GarageFinish     5.547945
GarageQual       5.547945
GarageType       5.547945
GarageCond       5.547945
BsmtExposure     2.602740
BsmtFinType2     2.602740
BsmtQual         2.534247
BsmtFinType1     2.534247
BsmtCond         2.534247
Electrical       0.068493
dtype: float64

Missing Percentages in Numerical Columns:
LotFrontage    17.739726
GarageYrBlt     5.547945
MasVnrArea      0.547945
dtype: float64
```

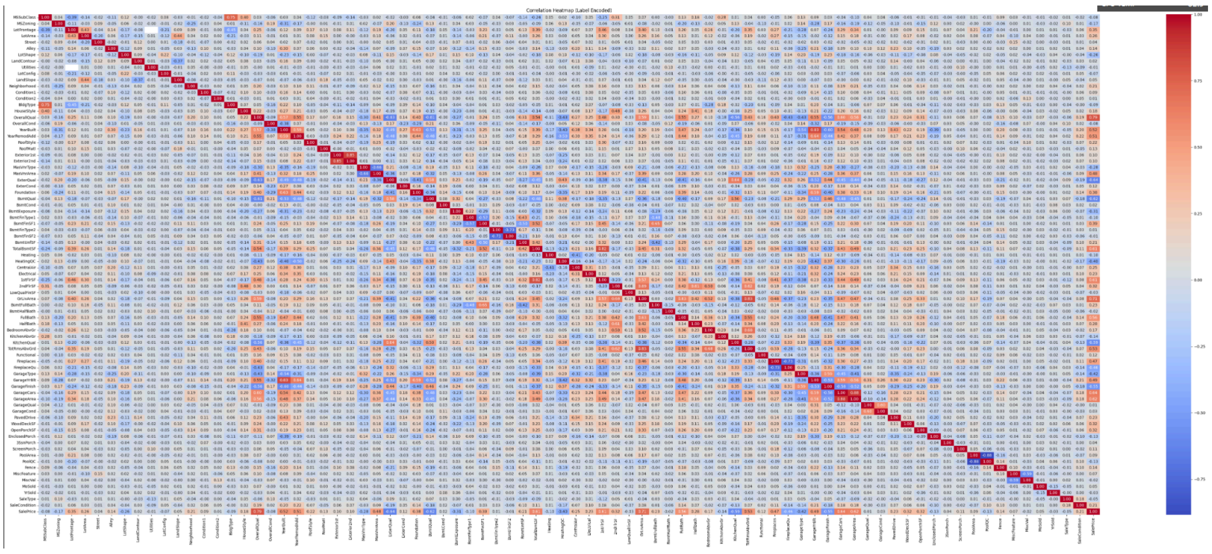
Based on our result from Python, 16 of the attributes got NA in their value. The meaning of NA is listed below:

PoolQC	NA= No Pool
MiscFeature	NA= None
Alley	NA = No alley access
Fence	NA =No Fence
MasVnrType	NA =None
FireplaceQu	NA =No Fireplace
GarageFinish	NA =No Garage
GarageQual	NA =No Garage
GarageType	NA =No Garage
GarageCond	NA =No Garage
BsmtExposure	NA =No Basement

BsmtFinType2	NA	=No Basement
BsmtQual	NA	=No Basement
BsmtFinType1	NA	=No Basement
BsmtCond	NA	=No Basement
Electrical	NA	= missing data

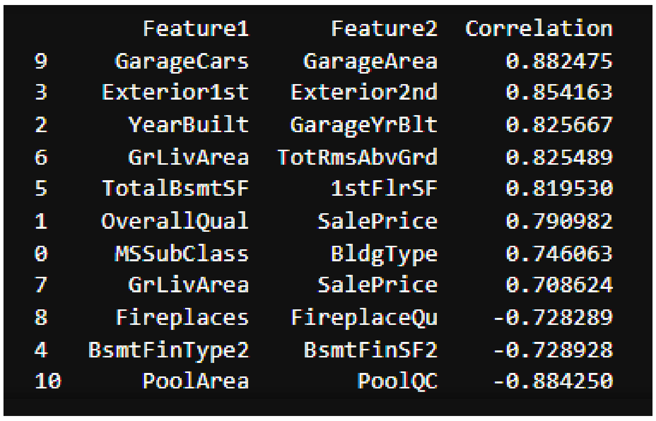
Six of them, Alley, PoolQC, MiscFeature, Fence, MasVnrType, and FireplaceQu, had more than 40 % missing percentages in the Categorical Column.

e. Correlation between variables



The heat map indicates the correlation between attributes and the correlation between the sale price and different attributes. We use this heat map to understand the relationship between variables.

Since the heatmap is large and it is difficult to find all the high correlations manually, we simplified the process of finding high correlation pairs with Python. Here are pairs of features with a high correlation higher than 0.7, which can help analyse redundant attributes.



```
        Feature1      Feature2  Correlation
9     GarageCars    GarageArea     0.882475
3    Exterior1st   Exterior2nd     0.854163
2      YearBuilt   GarageYrBlt     0.825667
6      GrLivArea  TotRmsAbvGrd     0.825489
5    TotalBsmtSF      1stFlrSF     0.819530
1    OverallQual     SalePrice     0.790982
0     MSSubClass      BldgType     0.746063
7      GrLivArea     SalePrice     0.708624
8     Fireplaces   FireplaceQu    -0.728289
4   BsmtFinType2    BsmtFinSF2    -0.728928
10      PoolArea        PoolQC    -0.884250
```

f. Observations

- **Dominated attribute value**
  Based on our observation, the pool area, pool quality, street, and utility are not informative; only 5 samples have a pool, and among 1460 samples, only 2 of them have different labels on "street."
- **Redundant attributes**
  Based on the heat map, some features highly correlate with each other but not with our target attribute, SalePrice, except for GrLivArea and OverallQual. With this result, we hypothesise that some of the attributes may overlap information. The result tidied by Python proved our hypothesis: GarageCars and GarageArea can represent each other due to the similar information represented. Moreover, OverallQual can maintain a high correlation with SalePrice since it represents the overall information of other features.

2. Feature engineering (based on part 1)

a. what have we done about it

- **Filling missing data**
  To facilitate our prediction model, we use 0 for numerical structural missing value to maintain data accuracy since it reflects reality, for example, GarageYrBlt.

  We are using None for the categorical missing value. For example, GarageFinish, PoolQC, Alley.
  To reduce the impact of outliers, use the median instead of the mean for numerical features such as MasVnrArea and LotFrontage.
  We are using mode for categorical features common category distribution. For example, MasVnrType and Electrical.

- **Convert non-numeric predictors into strings.**
  To prevent algorithm bias, treating YrSold and MoSold as numbers might make the model think 2020 is better than 2010 since 2020 is larger than 2010 in numerical terms. Also, Years/months don't have a numerical relationship with SalePrice. To remove the implied numeral ordering, convert YrSold and MoSold into strings.

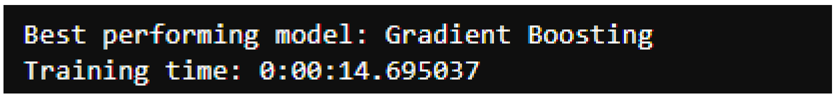
  In MSSubClass, the value shown in Excel is a number that the model will misidentify. The value stands for a category. For example, 20 = "1-STORY 1946 & NEWER ALL STYLES," 30 = "1-Story 1945 & Older." These numbers represent categories, not actual quantities. They are considered to be string data rather than numerical data. To prevent models from treating them as numerical values, they are converted to a string.

  These can provide better feature representation, prevent false numerical relationships, and improve model interpretation.

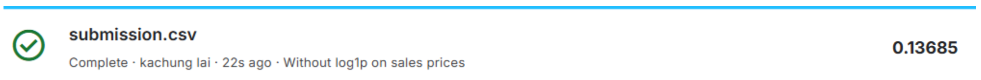
- **Apply label encoding**
  Change Categorical data into numerical values from 0 to n-1. If an attribute has five string values as attribute values, it may turn into 0,1,2,3 and 4. For example, ExterQual, Ex =4, Gd =3, TA =2, Fa =1,Po = 0.

  Converting categorical text data into numbers is essential for the machine learning requirement since machine learning can only process numerical data.

  It also provides better memory efficiency, as numbers take less memory than strings, which helps with faster computation compared to text data.

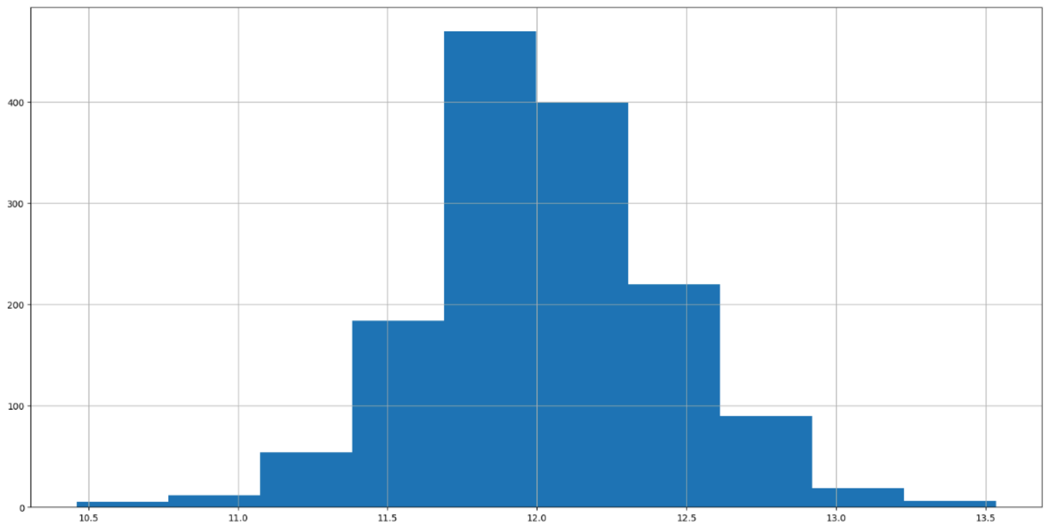


```
Best performing model: Gradient Boosting
Training time: 0:00:14.695037
```



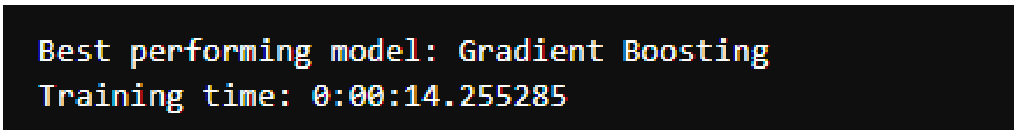
submission.csv
Complete · kachung lai · 22s ago · Without log1p on sales prices
0.13685

- **Log1p normalization for Sales Price**
  To solve the problem of right skewness, the distribution should be made more normal after using the log1p function on the sales price.

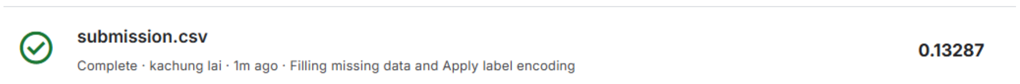


(Sales Price after normalized)

This helps reduce the impact of outliers and improves the model's performance.

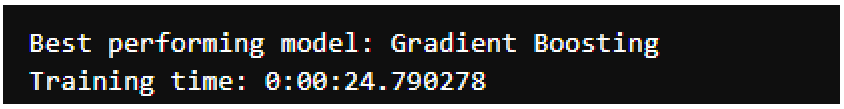


```
Best performing model: Gradient Boosting
Training time: 0:00:14.255285
```

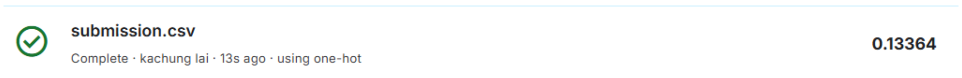


submission.csv
Complete · kachung lai · 1m ago · Filling missing data and Apply label encoding
0.13287

■ **One-hot encoding**



```
Best performing model: Gradient Boosting
Training time: 0:00:24.790278
```

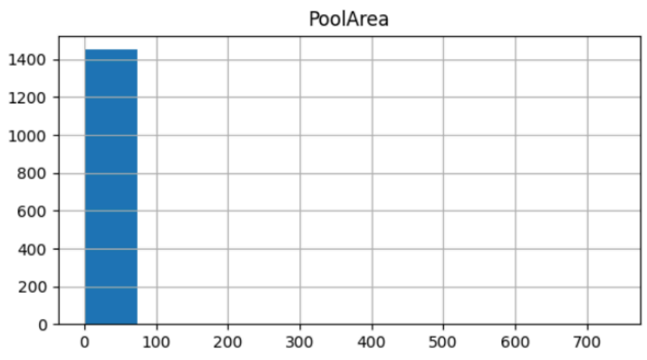


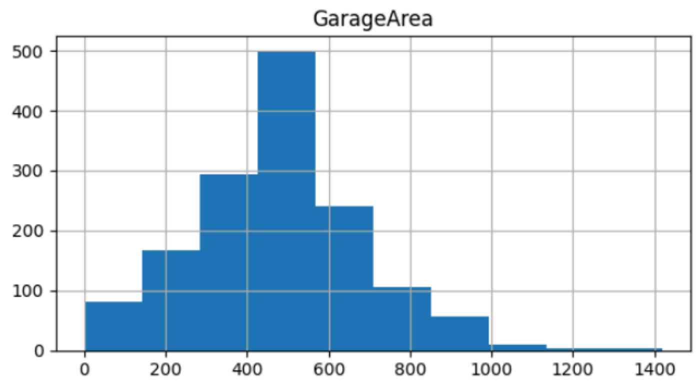
submission.csv
Complete · kachung lai · 13s ago · using one-hot
0.13364

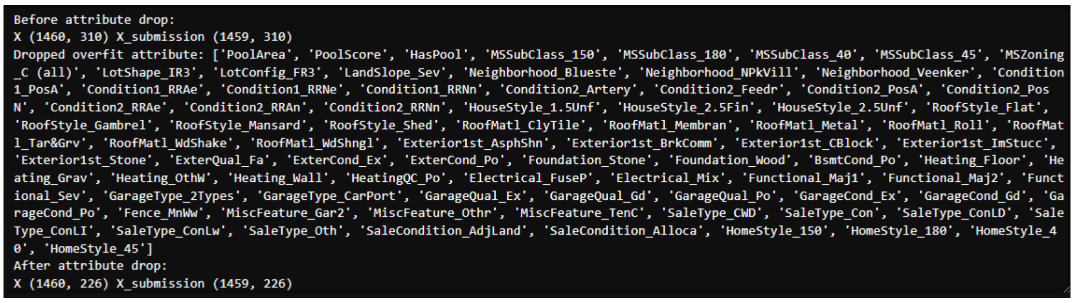
Python can hardly use string data to analyze their relationship against target variables. We turn them to values of 1 and 0 to facilitate data analysis. For example, the attribute ‘Street’ is separated by 2 values, Gravel and Paved. Through one-hot encoding, ‘Street’ will be transformed into two attributes, ‘Gravel’ and ‘Paved’. Both of them have attribute values of 1 and 0. This method sacrificed the dimensionality of the data set to facilitate our analysis on SalePrice.

■ **Dropping attributes (increase accuracy)**

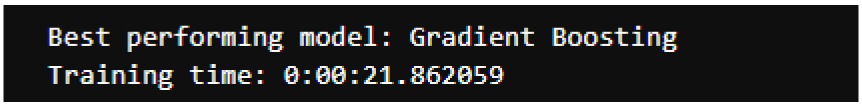
Attributes dominated by one attribute value provide minimal information for our regression prediction. To explain, the attribute value remains unchanged regardless of different sale prices. To reduce the complexity of the data, attributes with values dominating 99% of the sample will be dropped. Our group expressed these attributes clearly with bar charts, setting the y-axis as the number of samples and the x-axis as the attribute value.



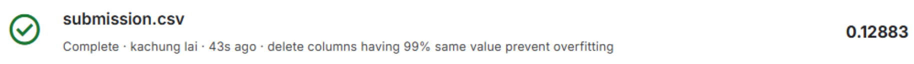




```
Before attribute drop:
X (1460, 310) X_submission (1459, 310)
Dropped overfit attribute: ['PoolArea', 'PoolScore', 'HasPool', 'MSSubClass_150', 'MSSubClass_180', 'MSSubClass_40', 'MSSubClass_45', 'MSZoning
_C (all)', 'LotShape_IR3', 'LotConfig_FR3', 'LandSlope_Sev', 'Neighborhood_Blueste', 'Neighborhood_NPkVill', 'Neighborhood_Veenker', 'Condition
1_PosA', 'Condition1_RRAe', 'Condition1_RRNe', 'Condition1_RRNn', 'Condition2_Artery', 'Condition2_Feedr', 'Condition2_PosA', 'Condition2_Pos
N', 'Condition2_RRAe', 'Condition2_RRAn', 'Condition2_RRNn', 'HouseStyle_1.5Unf', 'HouseStyle_2.5Fin', 'HouseStyle_2.5Unf', 'RoofStyle_Flat',
'RoofStyle_Gambrel', 'RoofStyle_Mansard', 'RoofStyle_Shed', 'RoofMatl_ClyTile', 'RoofMatl_Membran', 'RoofMatl_Metal', 'RoofMatl_Roll', 'RoofMat
l_Tar&Grv', 'RoofMatl_WdShake', 'RoofMatl_WdShngl', 'Exterior1st_AsphShn', 'Exterior1st_BrkComm', 'Exterior1st_CBlock', 'Exterior1st_ImStucc',
'Exterior1st_Stone', 'ExterQual_Fa', 'ExterCond_Ex', 'ExterCond_Po', 'Foundation_Stone', 'Foundation_Wood', 'BsmtCond_Po', 'Heating_Floor', 'He
ating_Grav', 'Heating_OthW', 'Heating_Wall', 'HeatingQC_Po', 'Electrical_FuseP', 'Electrical_Mix', 'Functional_Maj1', 'Functional_Maj2', 'Funct
ional_Sev', 'GarageType_2Types', 'GarageType_CarPort', 'GarageQual_Ex', 'GarageQual_Gd', 'GarageQual_Po', 'GarageCond_Ex', 'GarageCond_Gd', 'Ga
rageCond_Po', 'Fence_MnWw', 'MiscFeature_Gar2', 'MiscFeature_Othr', 'MiscFeature_TenC', 'SaleType_CWD', 'SaleType_Con', 'SaleType_ConLD', 'Sale
Type_ConLI', 'SaleType_ConLw', 'SaleType_Oth', 'SaleCondition_AdjLand', 'SaleCondition_Alloca', 'HomeStyle_150', 'HomeStyle_180', 'HomeStyle_4
0', 'HomeStyle_45']
After attribute drop:
X (1460, 226) X_submission (1459, 226)
```



```
Best performing model: Gradient Boosting
Training time: 0:00:21.862059
```



submission.csv
Complete · kachung lai · 43s ago · delete columns having 99% same value prevent overfitting

0.12883

■ Combining features

Combining features helps capture complex relationships.
For example:
features['BasementRatio'] = features['TotalBsmtSF'] / features['1stFlrSF']
It shows the basement ratio from the total square feet of the basement area to the first floor square feet.
Combining features enhances predictive power, which reveals hidden patterns and captures interaction effects, helps improve model accuracy.

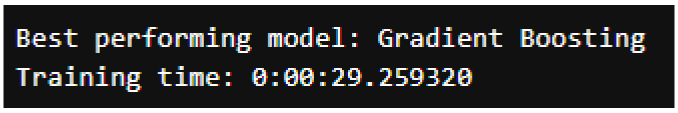
In our implementation:

- YrBltAndRemod: Combines year built and remodeling year
- TotalSF: Total square footage (basement + 1st floor + 2nd floor)
- Total_Bathrooms: Weighted sum of all bathrooms (full bath = 1, half bath = 0.5)
- Total_porch_sf: Combined square footage of all porch types
- GarageAge: Age of garage relative to house (GarageYrBlt - YearBuilt)
- RoomDensity: Rooms per living area ratio
- BasementRatio: Basement to first floor size ratio
- FireplaceQu: Maps fireplace quality to numeric values (Ex=5 to Po=1)
- BsmtFinType2: Maps basement finish type to numeric values (GLQ=6 to Unf=1)
- PoolQC: Maps pool quality to numeric values (similar to fireplace)
- FireplaceScore: Fireplaces count × FireplaceQu
- BsmtFinScore: BsmtFinSF2 × BsmtFinType2
- PoolScore: PoolArea × PoolQC

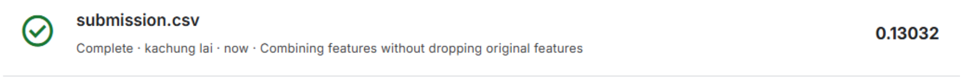
It can also integrate domain knowledge into the model, which reflects real estate expertise, to create industry-standard metrics that better represent property value factors.

However, it has some limitations; after combining features, the number of features increases, which increases the model's memory usage and processing time. Also, some potential information may get lost.

1. **without dropping original features**
   Some individual features might have a unique significance, which helps the model benefit from both combined and original data.



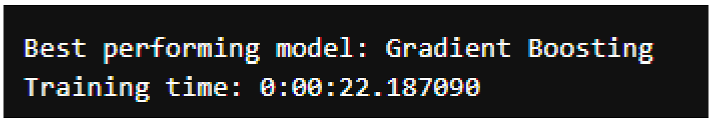
```
Best performing model: Gradient Boosting
Training time: 0:00:29.259320
```



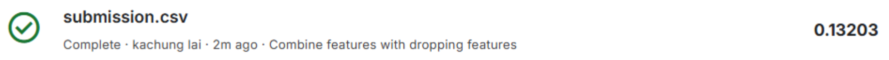
submission.csv
Complete · kachung lai · now · Combining features without dropping original features
0.13032

   When not dropping original features, the accuracy of the prediction decreases, and the training time of the model increases, which means the total performance of the model decreases.

2. **with dropping original features**
   The combined feature fully captures the relationship from the original features, and the originals have a high correlation.



```
Best performing model: Gradient Boosting
Training time: 0:00:22.187090
```

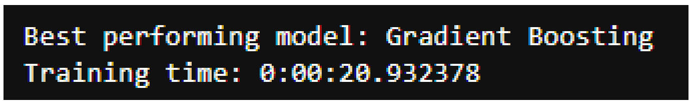


submission.csv
Complete · kachung lai · 2m ago · Combine features with dropping features
0.13203

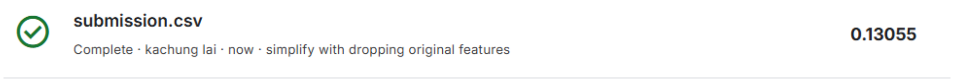
   When the original features are dropped, the prediction's accuracy still decreases. However, the training time can remain nearly the same.

- **Simplify Features**
  With dropping the original features from combined features, Multiple condition variables can be simplified, and similar categories can be grouped to reduce the complexity of the data.
  For example, The Neighborhood has over 20+ categories that can be divided into 3 different groups: high-end, medium-end, and Standard. This reduces complexity and gives better price range representation in the mode.

  In our implementation:
  HomeStyle: Simplifies MSSubClass into descriptive categories
  NeighborhoodClass: Groups neighbourhoods into High/Medium/Standard
  Binary Features: HasGarage, HasFireplace, HasPool, HasBasement



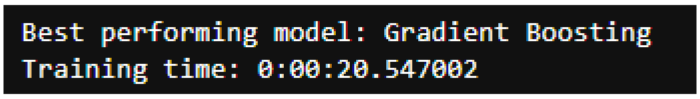
```
Best performing model: Gradient Boosting
Training time: 0:00:20.932378
```



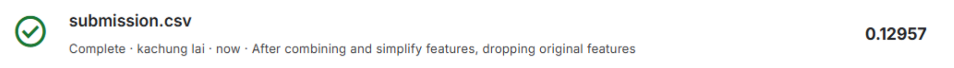
submission.csv
Complete · kachung lai · now · simplify with dropping original features
0.13055

- **Dropping high correlation attribute**
  A high correlation means high repetition between those two attributes. The attribute with a threshold larger than 0.7 is dropped due to a high correlation to remove the redundancy of the features. The way of selection is keeping one feature from highly correlated pairs, like GarageCars and GarageArea. Only keep GarageCars and drop GarageArea.

  These have improved the model to prevent overfitting and reduce multicollinearity. Also, dropping attributes with a smaller feature set helps with faster training and less memory usage.



```
Best performing model: Gradient Boosting
Training time: 0:00:20.547002
```



submission.csv
Complete · kachung lai · now · After combining and simplify features, dropping original features
0.12957

b. Compare with result

- **Result with Apply label encoding and applying one-hot encoding**
  The accuracy and also training time of one-hot encoding are worse than label encoding, which is due to the increased dimensionality of the dataset. However, label encoding is suitable for all of the data which don't have a clear ranking, one-hot is better for nominal categories without inherent order. If every attribute is labeled encoding, it might lead mode to assume numerical relationships between categories. One-hot encoding can prevent the model from assuming a numerical relationship, and no flas ordinal relationships are introduced, which makes more sense and gives the right relationship to the model for training.

- **Result with Combining features with dropping or not dropping**
  1. the accuracy decreased after we dropped the combine attribute. We deduce the accuracy decrease happened because of the different requirements of customers. For example,

features['TotalSF']=features['BsmtFinSF1'] + features['BsmtFinSF2']+ features ['BsmtUnfSF']+ features['1stFlrSF'] + features['2ndFlrSF']

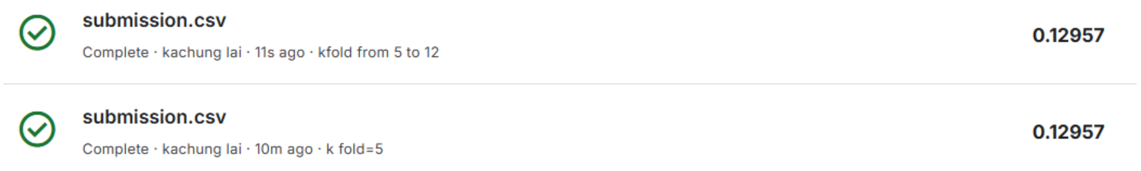
It is true that the features ['TotalSF'] contain all information about its composition. However, some consumers maybe more concerned with features['BsmtFinSF1'] than the features ['TotalSF']. If this is the case, features['TotalSF'] will not be enough to represent all its composition since underlying information is behind rather than just numerical information.

3. Model training

The model training aims to achieve higher precision on house prices and high performance in the balance. We have applied a few different models and strategies to compare.

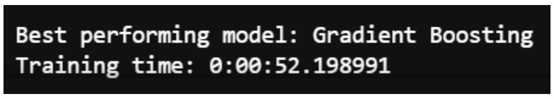
a. Cross-Validation Strategy

This strategy assesses model performance and generalizability. Given limited data, this strategy helps reduce overfitting and produces more reliable performance estimates on house pricing prediction.

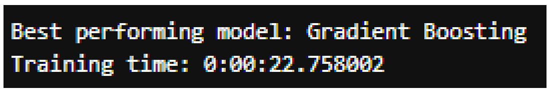

submission.csv
Complete · kachung lai · 11s ago · kfold from 5 to 12
0.12957

submission.csv
Complete · kachung lai · 10m ago · k fold=5
0.12957

The modeling approach begins with a robust cross-validation strategy using K-Fold (k=5) to ensure reliable model evaluation. We have tried increasing the fold (k=12)of the model training part, but the accuracy remains the same. However, the run time is around double that of the fold having less (k=5). Not a higher fold means better performance.


```
Best performing model: Gradient Boosting
Training time: 0:00:52.198991
```

(k=12)


```
Best performing model: Gradient Boosting
Training time: 0:00:22.758002
```

(k=5)

b. Data Preprocessing:

- **StandardScaler:**
  The data undergoes standardization, using StandardScaler to normalize all features to the same scale, which improves model convergence and prevents feature dominance.

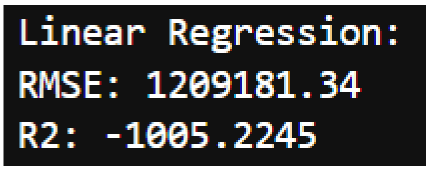
- **Normailise salesPrice (log1p):**

Using the log1p function transforms the target variable (salesPrice) to handle skewness and stabilize the variance. Not using normalization will cause model impact issues since most linear models assume a normal distribution.

c. Classification rule applied

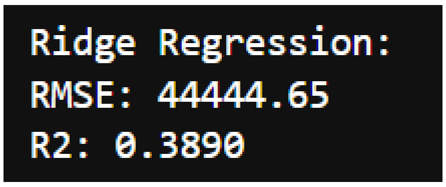
Six algorithms are applied to train the model with the data independently. The results of each model are compared to choose the best model.

- linear regression
  Basic linear relationship modelling interpretable results and capturing linear relationships. However, this model assumes a linear relationship in house prices, so it doesn't work well in this dataset.



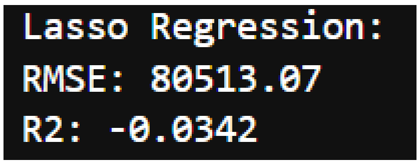
```
Linear Regression:
RMSE: 1209181.34
R2: -1005.2245
```

- Ridge Regression
  Linear regression with L2 regularisation to handle multicollinearity and prevent overfitting by penalizing large coefficients.



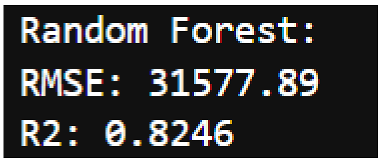
```
Ridge Regression:
RMSE: 44444.65
R2: 0.3890
```

- Lasso Regression
  Linear regression with L1 regularisation for feature selection and model simplification by potentially reducing coefficients to zero.



```
Lasso Regression:
RMSE: 80513.07
R2: -0.0342
```

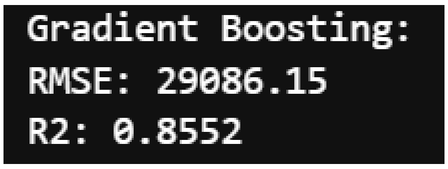
- Random Forest
  Ensemble of decision trees that capture non-linear relationships and provide feature-importance insights.



```
Random Forest:
RMSE: 31577.89
R2: 0.8246
```

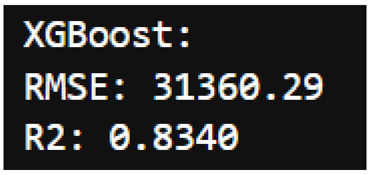
- Gradient Boosting
  Ensemble learning is a technique for building a strong predictive model by combining weak learners (typically decision trees)

sequentially. Each new model attempts to correct the errors made by previous models.



```
Gradient Boosting:
RMSE: 29086.15
R2: 0.8552
```

- XGboost
  Gradient boosting implementation utilises gradient boosting, offering high performance through sequential tree building and advanced regularization techniques.



```
XGBoost:
RMSE: 31360.29
R2: 0.8340
```

d. Training and Scoring

- RMSE (Root Mean Square Error)
  Root Mean Square Error measures prediction accuracy in the target variable's original scale, heavily penalizing large errors.
- R2
  R-squared quantifies the proportion of variance explained by the model, providing a scale-independent measure of fit.

e. Prediction and submission

- Sale Price Prediction
  The best model with the lowest RMSE means having the best accuracy for the target variable. In our submission, it always was the Gradient Boosting algorithm's model.
- Inverse Transformation
  Since the sales prices have been transformed by the log1p function, the np.expm1 function is needed to transform them back to the original price.

4. Obstacle

- **Failure to drop the attribute**
  We try to delete all data with a correlation below absolute (0.1), but it seems inefficient. As a reference, Attribute Pool Quality was not dropped. Pool Quality exists for housing units with pools only, while only 5 of the samples have pools. Therefore, we also consider it an attribute with minimal information. However, its correlation against Sale Revenue is 0.13. Therefore, we change to the current method to remove attributes.

  Moreover, before reading the data, we plan to remove every sample with any NA attribute value that we expect to be missing data that may create noise. Applying this to house price data, 90% of the sample will be removed since some attribute values got 90% of NA data. The NA percentage of attributes can be referred to in Part 1.

- **Understanding the attribute**
  In order to combine features, we first have to understand them. For example, we got features ['Total_porch_sf '] = (features ['OpenPorchSF '] + features ['3SsnPorch '] + features ['EnclosedPorch '] + features ['ScreenPorch '] + features ['WoodDeckSF ']) in our analysis. Each of them is considered a type of porch, while they are terms not similar to Hong Kong. It takes time for us to read the description and group these attributes.

5. Conclusion

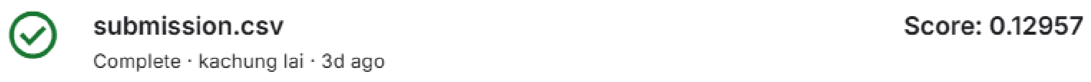
a. Which feature engineering effort is most useful in your competition work? Any reason behind?

Dropping dominated attributes proved to be highly beneficial for our competitive analysis. According to the results from part 2, our testing accuracy improved from 0.13364 to 0.12883 after removing these attributes. The dominant attributes exhibited similar values regardless of the SalePrice, indicating that they did not contribute meaningful information for predicting SalePrice. Instead, they can be classified as noise attributes, which can obscure valuable insights since they provide information as a part of data. By eliminating this noise, we were able to enhance the accuracy of our predictions.

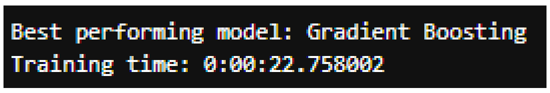
b. If you have submitted more than 1 entry, describe the case your team has obtained the best improvement (not absolute performance).

Our team achieved significant improvement through comprehensive data preprocessing and feature engineering. The data process began with filling missing data with median, zero or mode, depending on the data type and converting non-numeric predictors into a string format to prevent numerical relationships on these predictors—also, log1p normalisation to the target variable, which is SalesPrice, addresses skewness and ensures normal distribution. Moreover, in the feature engineering part, using one-hot encoding for categorical variables, strategic removal of outlier attributes, and the reaction of new combined features while dropping their original components to reduce dimensionality and improve performance and further reined our dataset by simplifying features and eliminating highly correlated attributes to reduce multicollinearity. In the model training part, model validation has been implemented with a robust 5-fold cross-validation strategy, with StandardScaler to normalise all features to the same scale. Finally, training our model with a gradient boost and predict the final result.

This result is not our best score, but it has the best performance in training time and complexity, with just a few losses in accuracy despite improving a lot in training time.



submission.csv
Complete · kachung lai · 3d ago
Score: 0.12957



```
Best performing model: Gradient Boosting
Training time: 0:00:22.758002
```

(Training 6 model total time with k=5)