Assessment 2
Practical Workplace–Related Data Analytics Project

House Prices analysis and prediction

Shuai Chen 12861896
Ziying Zhou 12485064
Jia Liu 12416400

Table of Contents

1. Business Understanding

House prices are always a hot topic for financial investment. The house prices even influence the economic development. Many people are interested in the house prices. Customers always care about the prices. The multiple factors could influence the house prices such as the locations, environment, areas, floor levels, etc. The good predictions could help customers to make the right decisions.

a. Business objective

The business objective is to provide the better customer services to our clients. Through the professional data analysis, the house price will be predicted. As a leading consulting organization, our accurate prediction of the future price trend could guide the investment of every client.

b. Data mining question

During the data mining process, there might be some problems. The data types might be different. The complex data combination would increase the difficulty in data processing. Moreover, the query languages of software are different as well. It requires the ability to extract data with different languages from all databases. Besides, since the noise data might affect the prediction quality, how to smooth the average numerical value is also significant.

c. Data source

Our data source is a dataset of the house prices. The data source in this report is all from Kaggle. Kaggle is the world's largest community of data scientists and machine learners. It is also the spare database of many organizations. The dataset includes 79 explanatory variables describing almost every aspect of residential homes in Ames, Iowa. The dataset is used for a competition on Kaggle. The competition asks to predict the final price of each home.

d. Project plan

The analysis will be completed by following the steps in CRISP-DM. The project plan is necessary to achieve our business objective. The business understanding leads to what kinds of data are required. The selected data will be prepared for modelling. The processed data will be evaluated. If the data is not suitable, the steps might be repeated.

During processing the data, our team will apply SAS EM to explore the appropriate dataset and apply Python to do data preparing and model building model. We will use multiple different models to train the dataset and then apply test data to predict the final results. Besides, Root-Mean-Square-Error (RMSE) will be applied to evaluate whether those results are good so that the better models could be selected.

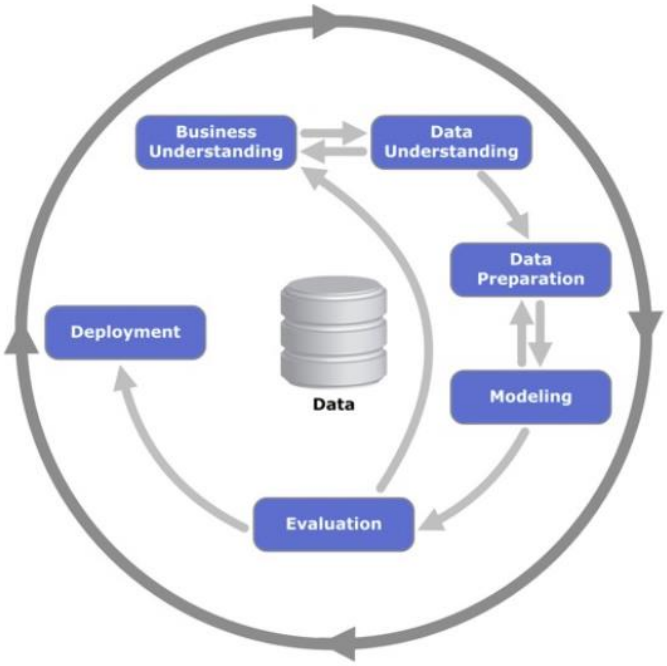


Figure 1: Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology (Vorhies, 2016)

2. Data Understanding

a. Data description

- **Train.csv**: the dataset used by training models
- **Test.csv**: the dataset that needs to use built models to predict
- **Target variable**: SalePrice; numeric target; description: the sale price of the houses in dollars;
- **Predictor variables**: a total of 80 related attributes affecting house prices including 28 interval attributes and 52 nominal attributes.
- **Description of attributes**:
    1. Properties location area: LotFrontage; LotArea; LotShape; Street; Alley; Neighbourhood, etc
    2. Properties details: OverallQual; YearBuilt; YearRemodAdd; RoofStyle; ExterQual, etc
    3. Properties equipment: BsmtQual; Heating; CentralAir; BsmtFullBath; Bedroom; Kitchen; Fireplace; GarageType; PoolArea,etc
    4. Sale details: MoSold; YrSold; SaleType; SaleCondition, etc

    The selected attributes with format and description information are listed in the Table 1 below.

Attribute	Format/Value	Description
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
Utilities: Type of utilities available	AllPub	All public Utilities (E,G,W,& S)
	NoSewr	Electricity, Gas, and Water (Septic Tank)
	NoSeWa	Electricity and Gas Only
	ELO	Electricity only
HouseStyle: Style of dwelling	1Story	One story
	1.5Fin	One and one-half story: 2nd level finished
	1.5Unf	One and one-half story: 2nd level unfinished
	2Story	Two story
	2.5Fin	Two and one-half story: 2nd level finished
	2.5Unf	Two and one-half story: 2nd level unfinished
	SFoyer	Split Foyer
	SLvl	Split Level
OverallQual: Rates the overall material and finish of the house	10	Very Excellent
	9	Excellent
	8	Very Good
	7	Good
	6	Above Average
	5	Average
	4	Below Average
	3	Fair
	2	Poor
	1	Very Poor
MSZoning: Identifies the general zoning classification of the sale.	A	Agriculture
	C	Commercial
	FV	Floating Village Residential
	I	Industrial
	RH	Residential High Density
	RL	Residential Low Density
	RP	Residential Low Density Park
	RM	Residential Medium Density
YearBuilt: Original construction date		
GrLivArea: Above grade (ground) living area square feet		

Table 1: Data Attributes

b. Data Exploration

Data exploration is the process of exploring data to discover the characteristics and distribution of the data and discover some obvious data rules. It can help us to find out which attributes are more important to the target variable and contributes to the future analysis: data preparing and building model. In this report, we describe data via statistical and visual techniques using SAS Enterprise Miner 14.

i. Descriptive statistics

Interval Variable Summary Statistics
(maximum 500 observations printed)

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
BedroomAbvGr	INPUT	2.866438	0.815778	1460	0	0	3	8	0.21179	2.230875
BsmtFinSF1	INPUT	443.6397	456.0981	1460	0	0	383	5644	1.685503	11.11824
BsmtFinSF2	INPUT	46.54932	161.3193	1460	0	0	0	1474	4.255261	20.11334
BsmtFullBath	INPUT	0.425342	0.518911	1460	0	0	0	3	0.596067	-0.8391
BsmtHalfBath	INPUT	0.057534	0.238753	1460	0	0	0	2	4.103403	16.39664
BsmtUnfSF	INPUT	567.2404	441.867	1460	0	0	476	2336	0.920268	0.474994
EnclosedPorch	INPUT	21.95411	61.11915	1460	0	0	0	552	3.089872	10.43077
Fireplaces	INPUT	0.613014	0.644666	1460	0	0	1	3	0.649565	-0.21724
FullBath	INPUT	1.565068	0.550916	1460	0	0	2	3	0.036562	-0.85704
GarageArea	INPUT	472.9801	213.8048	1460	0	0	480	1418	0.179981	0.917067
GarageCars	INPUT	1.767123	0.747315	1460	0	0	2	4	-0.34255	0.220998
GrLivArea	INPUT	1515.464	525.4804	1460	0	334	1464	5642	1.36656	4.895121
HalfBath	INPUT	0.382877	0.502885	1460	0	0	0	2	0.675897	-1.07693
KitchenAbvGr	INPUT	1.046575	0.220338	1460	0	0	1	3	4.488397	21.5324
LotArea	INPUT	10516.83	9981.265	1460	0	1300	9477	215245	12.20769	203.2433
LowQualFinSF	INPUT	5.844521	48.62308	1460	0	0	0	572	9.011341	83.23482
MSSubClass	INPUT	56.89726	42.30057	1460	0	20	50	190	1.407657	1.580188
MiscVal	INPUT	43.48904	496.123	1460	0	0	0	15500	24.47679	701.0033
MoSold	INPUT	6.321918	2.703626	1460	0	1	6	12	0.212053	-0.40411
OpenPorchSF	INPUT	46.66027	66.25603	1460	0	0	25	547	2.364342	8.490336
OverallCond	INPUT	5.575342	1.112799	1460	0	1	5	9	0.693067	1.106413
OverallQual	INPUT	6.099315	1.382997	1460	0	1	6	10	0.216944	0.096293
ScreenPorch	INPUT	15.06096	55.75742	1460	0	0	0	480	4.122214	18.43907
TotRmsAbvGrd	INPUT	6.517808	1.625393	1460	0	2	6	14	0.676341	0.880762
TotalBsmtSF	INPUT	1057.429	438.7053	1460	0	0	991	6110	1.524255	13.25048
WoodDeckSF	INPUT	94.24452	125.3388	1460	0	0	0	857	1.541376	2.992951
YearBuilt	INPUT	1971.268	30.2029	1460	0	1872	1973	2010	-0.61346	-0.43955
YearRemodAdd	INPUT	1984.866	20.64541	1460	0	1950	1994	2010	-0.50356	-1.27225
YrSold	INPUT	2007.816	1.328095	1460	0	2006	2008	2010	0.096268	-1.27186
_1stFlrSF	INPUT	1162.627	386.5877	1460	0	334	1086	4692	1.376757	5.745841
_2ndFlrSF	INPUT	346.9925	436.5284	1460	0	0	0	2065	0.81303	-0.55346
_3SsnPorch	INPUT	3.409589	29.31733	1460	0	0	0	508	10.30434	123.6624
SalePrice	TARGET	180921.2	79442.5	1460	0	34900	163000	755000	1.882876	6.536282

Figure 2: Interval Variable Summary Statistics

Figure 2 shows a summary descriptive statistic of interval variables which describes quantitatively the dataset and shows the features of each attribute. Here we calculate the mean and median to describe the central tendency; standard deviation, maximum, minimum, skewness and kurtosis to measure the variability.

Class Variables

Data Role	Variable Name	Level	CODE	Frequency Count	Type	Percent	Level Index	Role	Label	Plot
TRAIN	Alley	NA	0	1369	C	93.76712	2	INPUT	Alley	1
TRAIN	Alley	Grvl	1	50	C	3.424658	1	INPUT	Alley	1
TRAIN	Alley	Pave	2	41	C	2.808219	3	INPUT	Alley	1
TRAIN	BldgType	1Fam	0	1220	C	83.56164	1	INPUT	BldgType	1
TRAIN	BldgType	TwnhsE	3	114	C	7.808219	5	INPUT	BldgType	1
TRAIN	BldgType	Duplex	2	52	C	3.561644	3	INPUT	BldgType	1
TRAIN	BldgType	Twnhs	4	43	C	2.945205	4	INPUT	BldgType	1
TRAIN	BldgType	2fmCon	1	31	C	2.123288	2	INPUT	BldgType	1
TRAIN	BsmtCond	TA	0	1311	C	89.79452	5	INPUT	BsmtCond	1
TRAIN	BsmtCond	Gd	1	65	C	4.452055	2	INPUT	BsmtCond	1
TRAIN	BsmtCond	Fa	3	45	C	3.082192	1	INPUT	BsmtCond	1
TRAIN	BsmtCond	NA	2	37	C	2.534247	3	INPUT	BsmtCond	1
TRAIN	BsmtCond	Po	4	2	C	0.136986	4	INPUT	BsmtCond	1
TRAIN	BsmtExposure	No	0	953	C	65.27397	5	INPUT	BsmtExposure	0
TRAIN	BsmtExposure	Av	3	221	C	15.13699	1	INPUT	BsmtExposure	0
TRAIN	BsmtExposure	Gd	1	134	C	9.178082	2	INPUT	BsmtExposure	0
TRAIN	BsmtExposure	Mn	2	114	C	7.808219	3	INPUT	BsmtExposure	0
TRAIN	BsmtExposure	NA	4	38	C	2.60274	4	INPUT	BsmtExposure	0
TRAIN	BsmtFinType1	Unf	2	430	C	29.45205	7	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	GLQ	0	418	C	28.63014	3	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	ALQ	1	220	C	15.06849	1	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	BLQ	4	148	C	10.13699	2	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	Rec	3	133	C	9.109589	6	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	LwQ	6	74	C	5.068493	4	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType1	NA	5	37	C	2.534247	5	INPUT	BsmtFinType1	0
TRAIN	BsmtFinType2	Unf	0	1256	C	86.0274	7	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	Rec	4	54	C	3.69863	6	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	LwQ	5	46	C	3.150685	4	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	NA	2	38	C	2.60274	5	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	BLQ	1	33	C	2.260274	2	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	ALQ	3	19	C	1.30137	1	INPUT	BsmtFinType2	0
TRAIN	BsmtFinType2	GLQ	6	14	C	0.958904	3	INPUT	BsmtFinType2	0
TRAIN	BsmtQual	TA	1	649	C	44.45205	5	INPUT	BsmtQual	0
TRAIN	BsmtQual	Gd	0	618	C	42.32877	3	INPUT	BsmtQual	0
TRAIN	BsmtQual	Ex	2	121	C	8.287671	1	INPUT	BsmtQual	0
TRAIN	BsmtQual	NA	3	37	C	2.534247	4	INPUT	BsmtQual	0
TRAIN	BsmtQual	Fa	4	35	C	2.39726	2	INPUT	BsmtQual	0
TRAIN	CentralAir	Y	0	1365	C	93.49315	2	INPUT	CentralAir	1

Figure 3: Class Variables

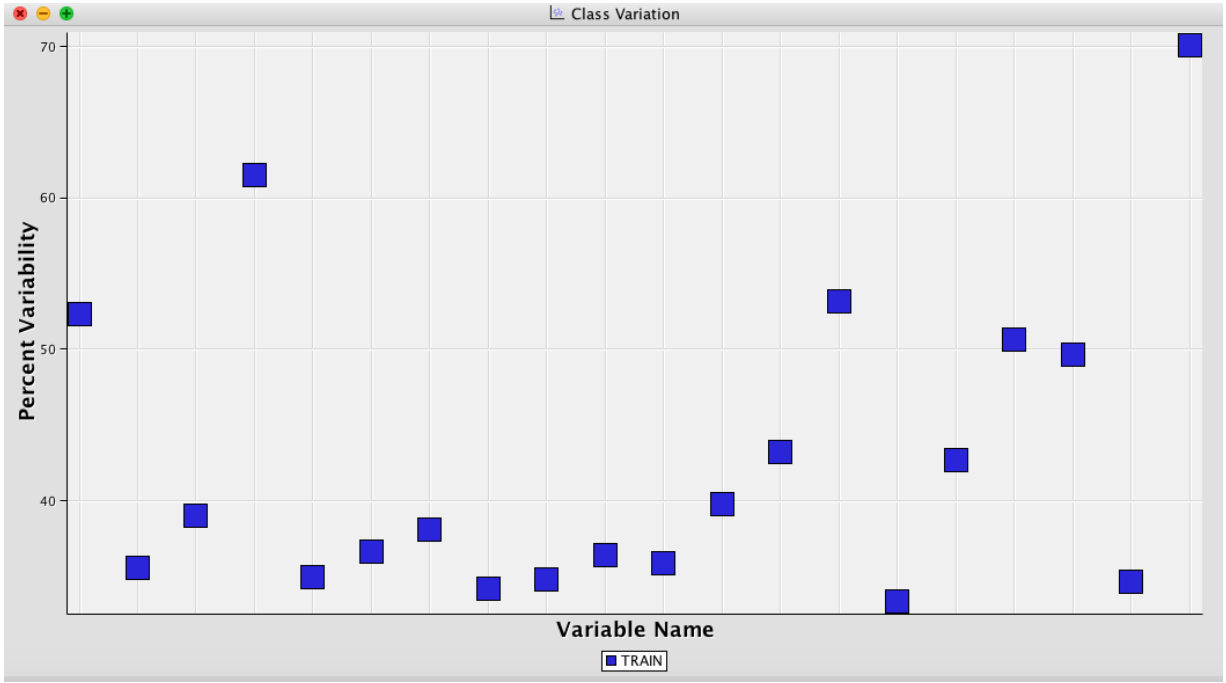


Figure 4: Percent Variability of class variables

For nominal variables, we cannot use statistical analysis with the number. Figure 3 shows how to category the nominal variables into numeric and the percentage of each value in each attribute. Figure 4 use scattered points briefly draw a percentage variability of class variation.

ii. Data visualization

A graphical method to explore dataset is easy and quick for us to find out the features of variables from the large dataset. Following we will use the histogram, line chart, pie chart to explore some interesting and important variables from the training data.

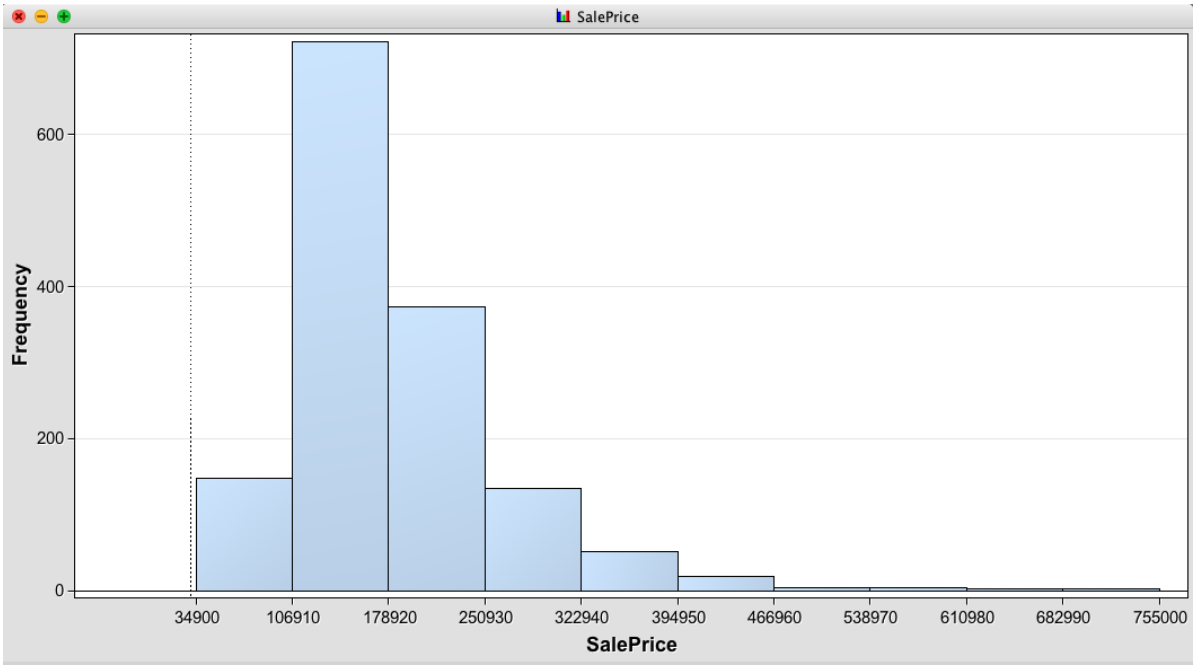


Figure 5: SalePrice Histogram

As shown in Figure 5, the distribution is right-skewed. There is largely different from maximum to minimum. The most sale price of the properties is between 106910 to 178920. The price over 4669690 is rare.

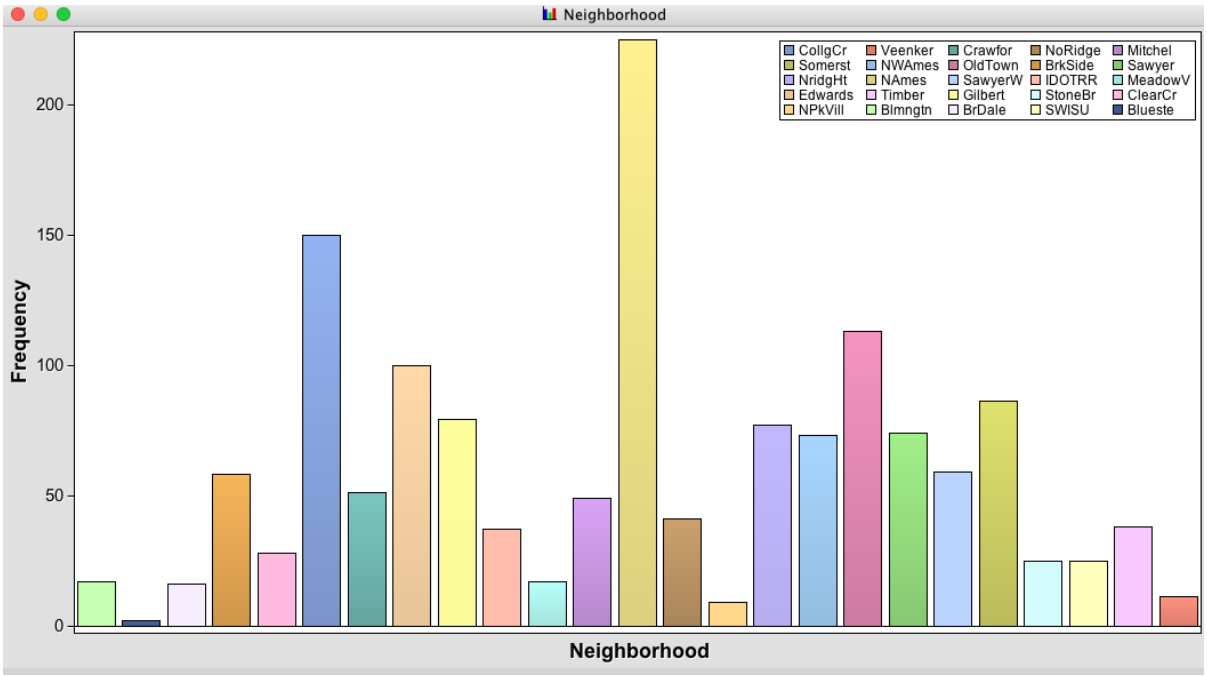


Figure 6: Neighbourhood Frequency

Figure 6 shows the categorical data including 25 different levels of values which means the Neighborhood variable has the largest distinct value in the dataset. Region named NAmes has the largest frequency in this dataset.

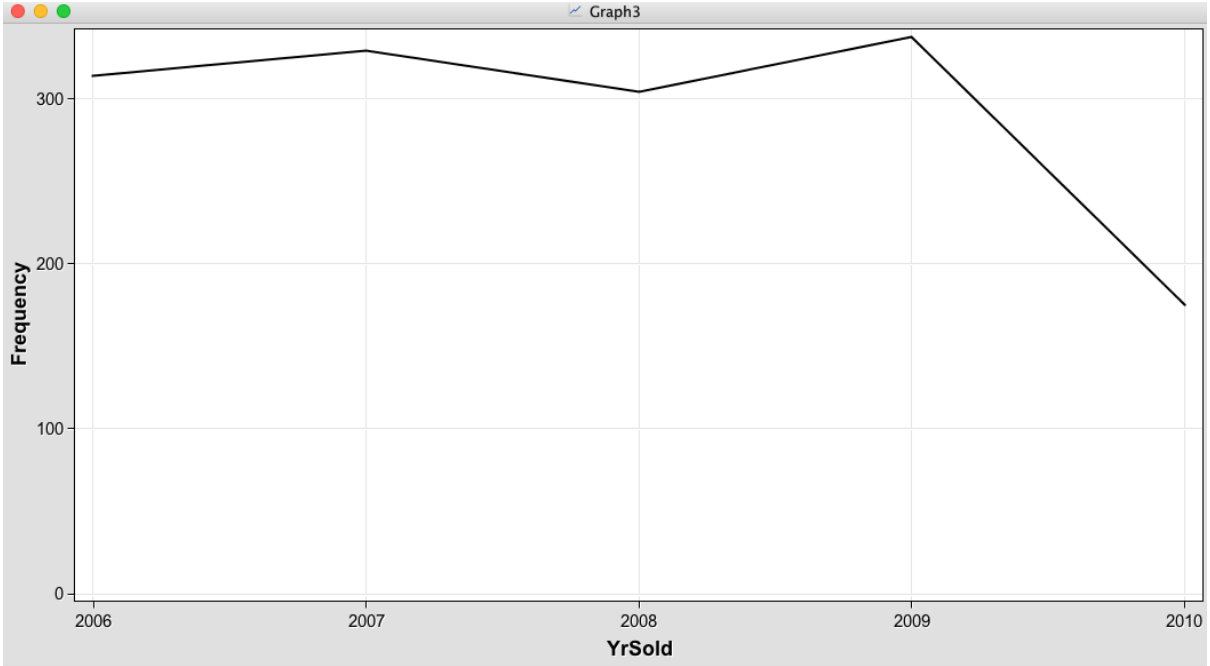


Figure 7: YrSold line chart

The line chart like Figure 7 is a very good way to show the changes in the price of properties by years. As we can see, after 2009, the number of sold has a decreased sharply.

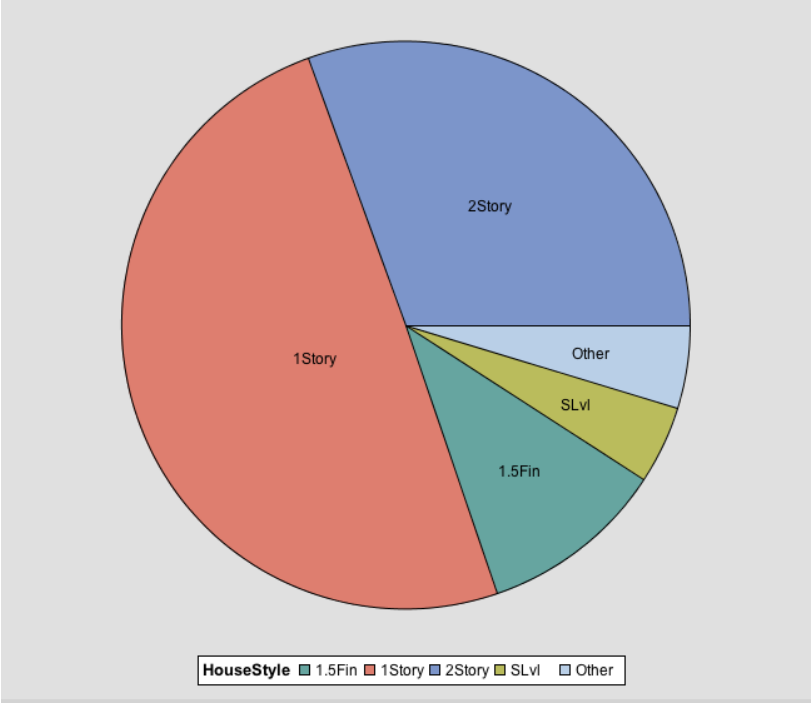


Figure 8: HouseStyle pie chart

The pie chart is not fit for large classification. As in Figure 8, the chart visually shows the proportion of each population. 1story as the house style is nearly half of the total dataset.

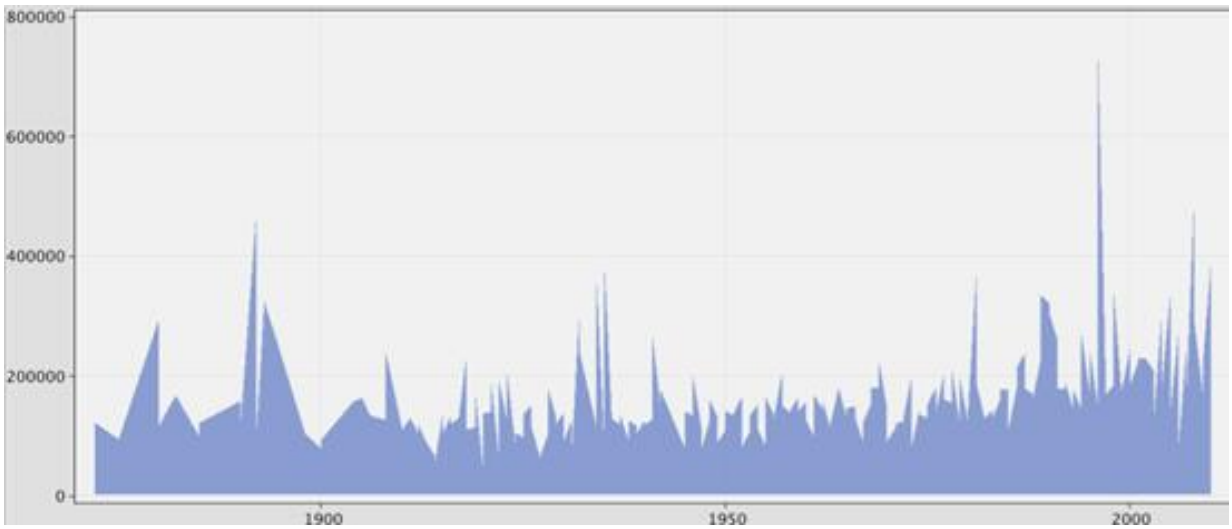


Figure 9: YearBuild vs SalePrice

In Figure 9, it shows a big peak in the around the year of 2000 and the sold of new properties which built after 2000 showing a gradual growth.

iii. Correlation

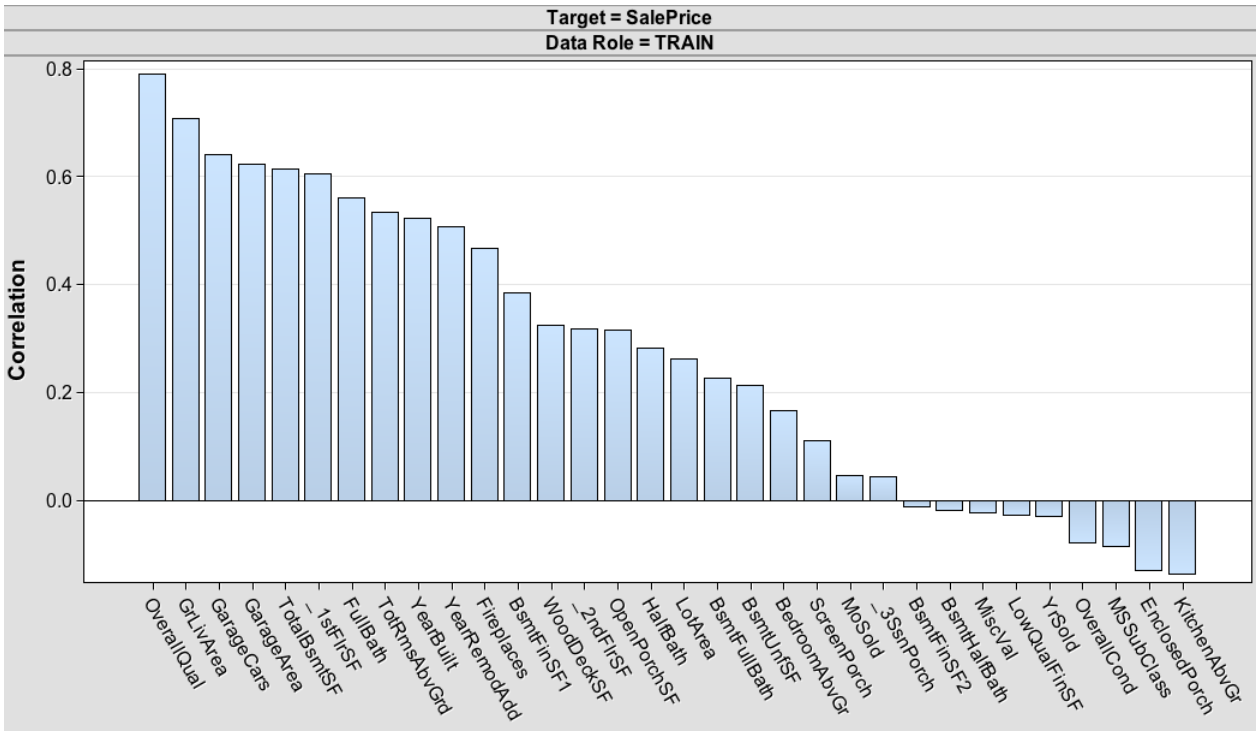


Figure 10: Correlation

To get a better accuracy of prediction, we need to select useful inputs to train the prediction model especially there are so many input variables in this project. So, the correlation between the variables and target variable is the main evaluation in data exploration.

As shown in Figure 10, we use coordinate to show the strength of the relationship between the SalePrice and other attributes. There are many values are negative like KitchenAbvGr, EnclosedPorch, MSSubClass, OverallCond, etc and some are at the low value like MoSold, _3SsnPorch and so on. We can remove these inputs directly from our model so that increases accuracy and reduce some workload.

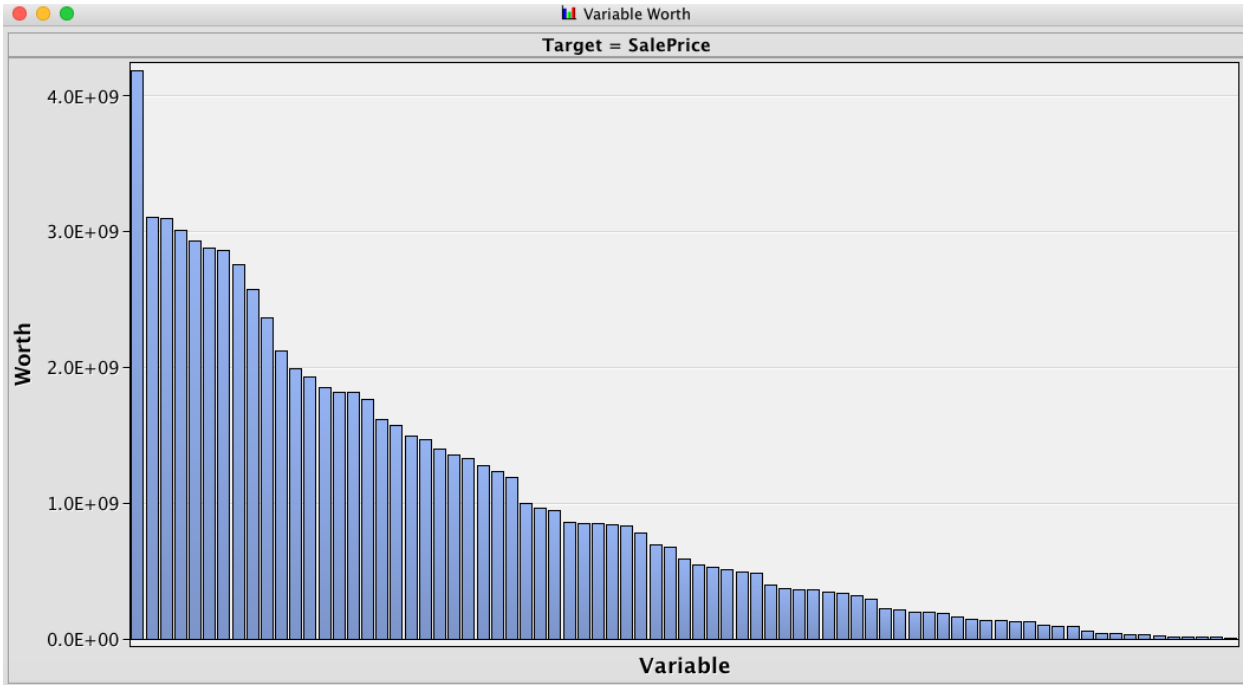


Figure 11: Variable worth

Also, the variable worth like Figure 11 can quantify the weight of importance of variables.

c. Verify data quality

i. Boxplots and outliers

To find out the anomalies in the most relevant attributes, the best way are boxplots and outliers. Extreme values do not have the strong reference so that we will delete the values in training data that we find in this part.

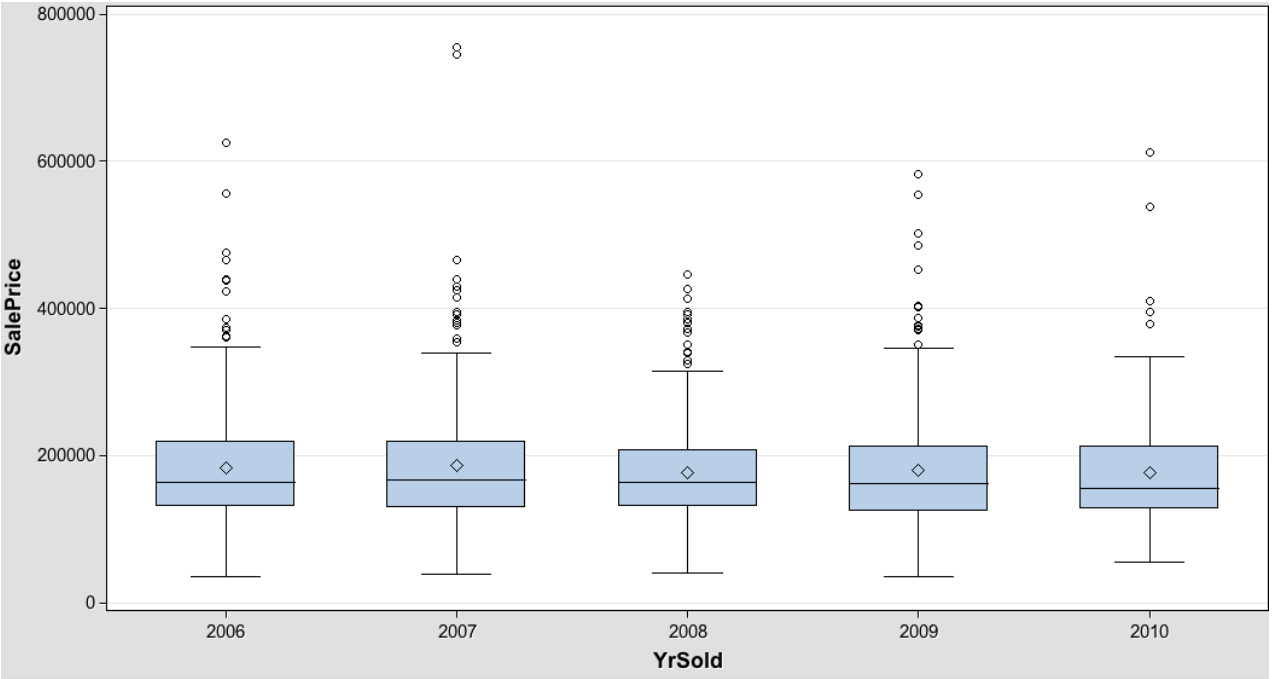


Figure 11: SalePrice vs YrSold

Figure 11 explained the sale price are similar in every year but there are still two abnormal large values need to be dealing with.

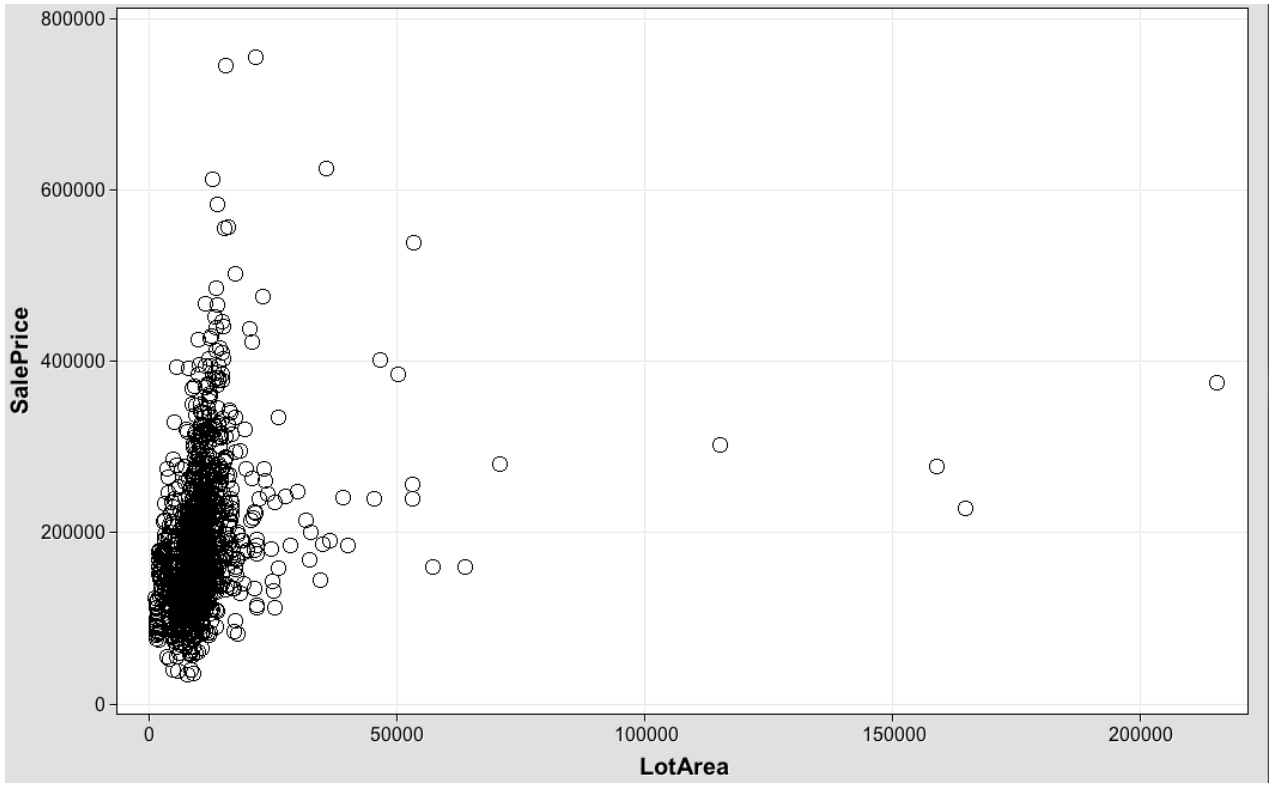


Figure 12：SalePrice vs LotArea

It is easily seen in Figure 12, there is one house with a very large area but sold very cheap. Also, two discrete points in the upper left corner show they sold huge money with the real small area.

ii. Missing & empty values

Ordered Inputs	Variable	Missing	Non Missing	Minimum
1	MiscVal	0	1460	0
2	_3SsnPorch	0	1460	0
3	LowQualFinSF	0	1460	0
4	BsmtHalfBath	0	1460	0
5	ScreenPorch	0	1460	0
6	BsmtFinSF2	0	1460	0
7	EnclosedPorch	0	1460	0
8	OpenPorchSF	0	1460	0
9	WoodDeckSF	0	1460	0
10	HalfBath	0	1460	0
11	_2ndFlrSF	0	1460	0
12	BsmtFullBath	0	1460	0
13	Fireplaces	0	1460	0
14	BsmtFinSF1	0	1460	0
15	LotArea	0	1460	1300
16	BsmtUnfSF	0	1460	0
17	MSSubClass	0	1460	20
18	GarageArea	0	1460	0
19	SalePrice	0	1460	34900
20	MoSold	0	1460	1
21	GarageCars	0	1460	0
22	TotalBsmtSF	0	1460	0
23	FullBath	0	1460	0
24	GrLivArea	0	1460	334
25	_1stFlrSF	0	1460	334
26	BedroomAbvGr	0	1460	0
27	TotRmsAbvGrd	0	1460	2
28	OverallQual	0	1460	1
29	KitchenAbvGr	0	1460	0
30	OverallCond	0	1460	1
31	YearBuilt	0	1460	1872
32	YearRemodAdd	0	1460	1950
33	YrSold	0	1460	2006

PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	MnPrv	Shed	700
0	NA	NA	NA	0
0	NA	NA	Shed	350
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	GdWo	NA	0
0	NA	GdPrv	NA	0
0	NA	NA	Shed	700
0	NA	NA	Shed	500
0	NA	NA	NA	0
0	NA	MnPrv	NA	0
0	NA	NA	NA	0
0	NA	GdPrv	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	MnPrv	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	MnPrv	NA	0
0	NA	MnPrv	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0
0	NA	NA	NA	0

Figure 13: missing values

Missing values are also very important to the process of prediction. Although there are no empty values in our dataset, we can find there are large amount of NA (not available) values which means for the missing values.

To deal with missing values, we can delete those directly or fill in with the most popular value instead. Whether which method is better for prediction result, we will try and verify in the following step.

3. Data preparation

a. Data Observations

In general, the index column of the source data is not very useful. Therefore, we can use it as an index of our pandas data frame. It is easy to retrieve it later in this way.

```
In [6]: train_df.head()
```

Out[6]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	P
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	0	
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	0	
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	0	
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	0	

5 rows × 80 columns

Starting with the selection of the attributes to be processed, we decided to analyze the relationships between “OverallQual” and “SalePrice”, and “SalePrice” and “YearBuilt”. Then we produce the box plot of those two set of attributes.

```
#box plot overallqual/saleprice
var = 'OverallQual'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```

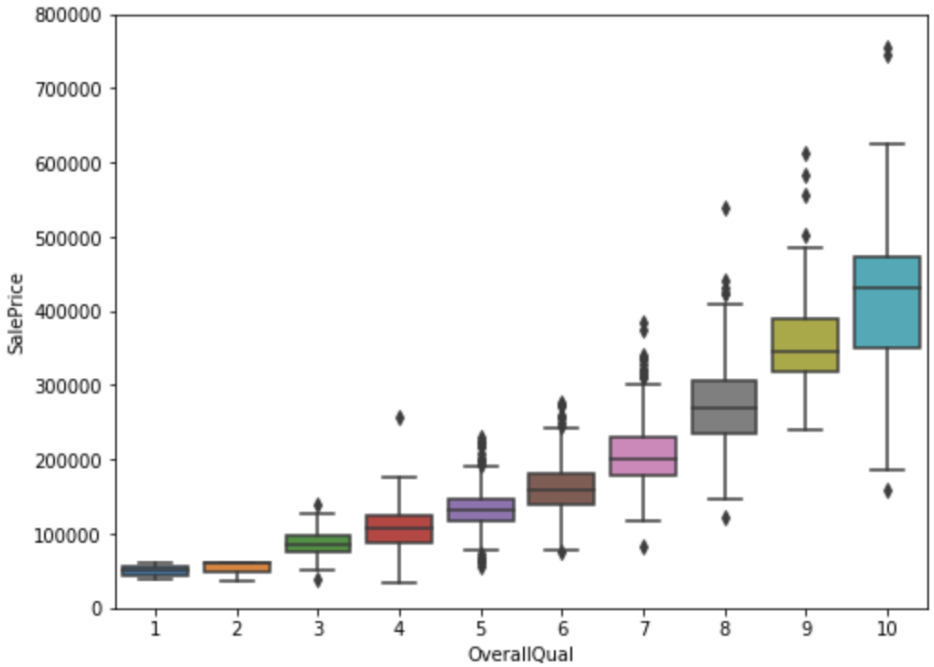


Figure 14: Box plot of OverallQual and SalePrice

The Figure 14 shows the relationship between SalePrice and OverallQual. The box plot seems positive because when the OverallQual raises, the SalePrice also raises.

```
var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
plt.xticks(rotation=90);
```

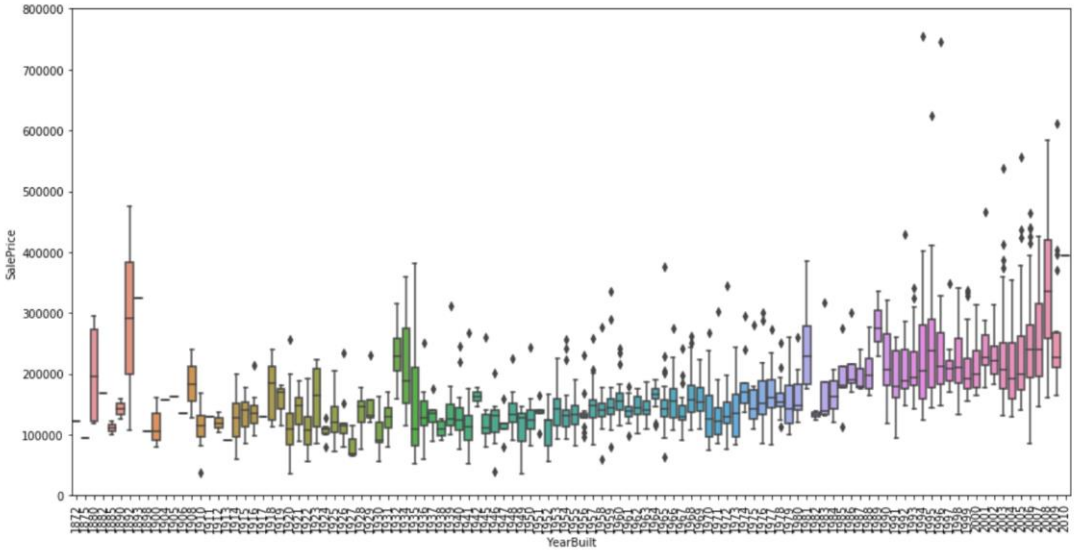


Figure 15: Box plot of YearBuilt and SalePrice

From the Figure 15, we realise that the YearBuilt and SalePrice also have relationships. However, this relationship does not look particularly stable compared with the OverallQual attribute.

```
In [32]: #scatterplot GrLivArea/SalePrice
colors = np.random.rand(50)
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
print(type(data))
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000), c='r');
```

```
<class 'pandas.core.frame.DataFrame'>
```

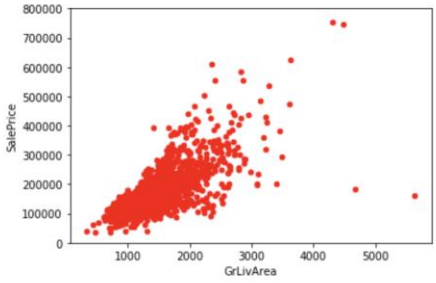


Figure 16 (a). Scatter plot

The Figure 16 (a) shows the relationship of scatter plot between SalePrice and GrLiveArea. form the figure, we can see when the GrLivArea is 4500 the SalePrice is the most. There are

two plots at the bottom right which can use very low price to get the large living area. We should remove those two plots because they are huge outliers. See Figure 16 (b)

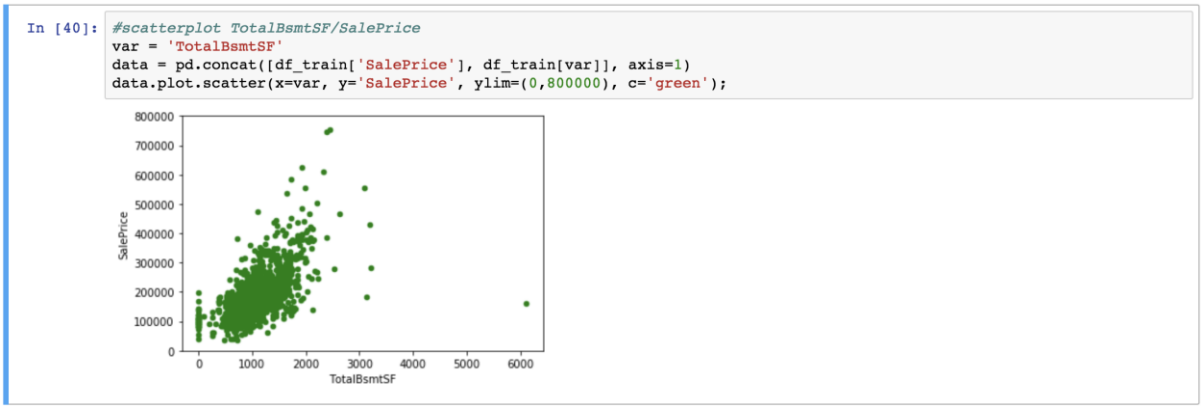


Figure 16 (b). Scatter plot

b. Data Cleaning

The original dataset contains 79 attribute vectors. However, only a few parts of the attributes are closely related to our project. Therefore, data cleaning is a very important process because it can effectively identify the null value, duplicate values, missing values or outliers.

Data missing is a common phenomenon. It has the negative impact on analysis processing. therefore, we should identify and remove those missing data to avoiding covering up the facts. The next table presents the percentage of missing values in terms of different attributes.

	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479
GarageType	81	0.055479
GarageYrBlt	81	0.055479
GarageFinish	81	0.055479
GarageQual	81	0.055479
BsmtExposure	38	0.026027
BsmtFinType2	38	0.026027
BsmtFinType1	37	0.025342
BsmtCond	37	0.025342
BsmtQual	37	0.025342
MasVnrArea	8	0.005479
MasVnrType	8	0.005479
Electrical	1	0.000685
Utilities	0	0.000000

Figure 17. Percentage of missing values

We found out that the 'PoolQC', 'MiscFeature', 'Alley', and 'FireplaceQu' take a high percentage of missing value. Therefore, they should be removed. Some attributes having a weak correlation with purchasing house also need to delete, such as GarageCond, GarageType,BsmtExposure, BsmtExposure etc.

The next code fixes the missing value problem as well as remove all the attribute with value missing.

```
In [35]: #fix the missing value
         df_train = df_train.drop((missing_data[missing_data['Total'] > 1]).index,1)
         df_train = df_train.drop(df_train.loc[df_train['Electrical'].isnull()].index)
         df_train.isnull().sum().max() #check again to see if there is still any missing value
Out[35]: 0
```

4. Data Pre-processing and Transformations

a. Consolidated data

In order to facilitate data preprocessing, we should merge the data firstly. After all the necessary pretreatment is done, we will separate them.

Firstly, the sales price attribute should be analyzed.

```
In [132]: %matplotlib inline
          prices = pd.DataFrame({"price":train_df["SalePrice"], "log(price + 1)":np.log1p(train_df["SalePrice"])})
          prices.hist(color="#008000")

Out[132]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11b91eba8>,
                  <matplotlib.axes._subplots.AxesSubplot object at 0x1a20e817f0>]], dtype=object)
```

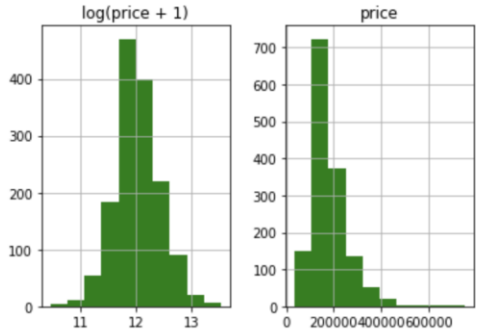


Figure 18. Analysis of SalePrice

From the Figure 18, we figure out the label itself is not smooth, so we need to deal with it (the next code).

```
In [5]: y_train = np.log1p(train_df.pop('SalePrice'))
```

And then we combine the best part.

```
In [6]: all_df = pd.concat((train_df, test_df), axis=0)
```

The result shows the consolidate data frame.

```
In [7]: all_df.shape

Out[7]: (2919, 79)
```

The y_train is the belong to the SalePrice column

```
In [8]: y_train.head()

Out[8]: Id
        1    12.247699
        2    12.109016
        3    12.317171
        4    11.849405
        5    12.429220
        Name: SalePrice, dtype: float64
```

In order to be more scientific, we need to get the correlation matrix of each feature. From this figure, we discover that the redder of pixel block indicates having a stronger correlation.

```
In [31]: #correlation matrix
corrmat = df_train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```

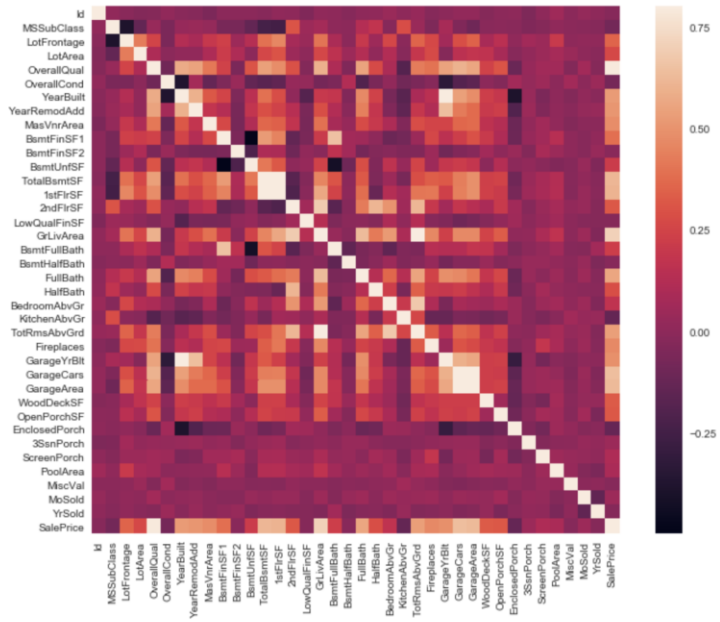


Figure 19. The correlation matrix of each feature

The strong correlation with "SalePrice" is OverallQual, YearBuilt, ToatlBsmtSF, 1stFlrSF, GrLiveArea, FullBath, TotRmsAbvGrd, GarageCars and GarageArea. therefore, we should use those attributes to create another correlation matrix for further screening.

```
In [32]: #saleprice correlation matrix
k = 10 #number of variables for heatmap
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(df_train[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values,
                 xticklabels=cols.values)
plt.show()
```

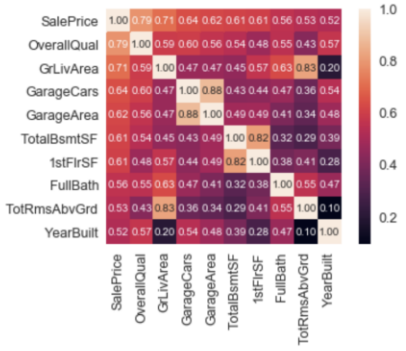


Figure 20. Correlation matrix of highly correlated features

'OverallQual' and 'GrLiveArea' are highly correlated and consistent with our expectations, so they are selected decisively.
GarageCars' is similar to 'GarageArea ', so we just need to choose one of them.
'TotalBsmtSF' and '1stFloor' also similar, so we choose 'TotalBsmtSF'.

The next figure presents the scatter plots between 'SalePrice' and correlated attributes

```
In [33]: #scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.pairplot(df_train[cols], size = 2.5)
plt.show();
```

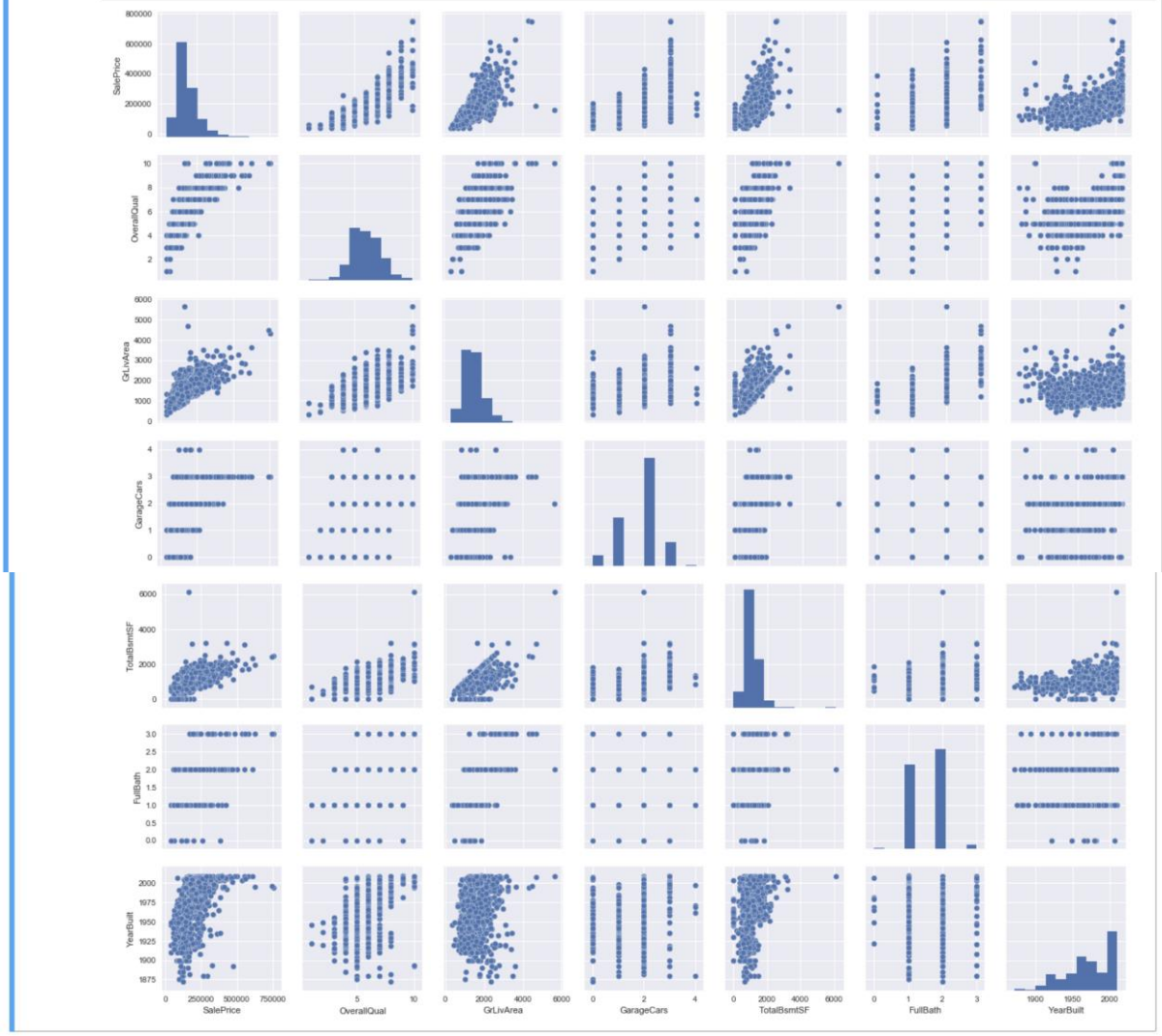


Figure 21. Scatter plots between 'SalePrice' and correlated attributes

b. Variable conversion

Variable conversion is a unity which can deal with the inconvenient or inconsistent data. First, we should transform the variable correctly. the values of MSSubClass would be a category. When we use DF, this type of number symbol will be remembered as a numeric by default. This is very misleading, and we need to change it back to the string.

```
In [9]: all_df['MSSubClass'].dtypes
Out[9]: dtype('int64')

In [10]: all_df['MSSubClass'] = all_df['MSSubClass'].astype(str)
```

```
In [11]: all_df['MSSubClass'].value_counts()

Out[11]: 20     1079
         60      575
         50      287
         120     182
         30      139
         160     128
         70      128
         80      118
         90      109
         190      61
         85       48
         75       23
         45       18
         180      17
         40        6
         150       1
         Name: MSSubClass, dtype: int64
```

When we use numerical to express categorical, we should pay attention to that the number itself has the meaning of size, so using numeric indiscriminately will cause trouble for model learning later. We can use the one-hot method to express category based on pandas.

```
In [12]: pd.get_dummies(all_df['MSSubClass'], prefix='MSSubClass').head()
```

Out[12]:

Id	MSSubClass_120	MSSubClass_150	MSSubClass_160	MSSubClass_180	MSSubClass_190	MSSubClass_20	MSSubClass_30	MSSubClass_40	MSSubClass_45
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0

At this point, MSSubClass is divided into 12 columns, each representing a category.
And similarly, we're going to handle all data of category by one-hot process.

```
In [13]: all_dummy_df = pd.get_dummies(all_df)
         all_dummy_df.head()
```

Out[13]:

Id	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	...	SaleType_ConLw	SaleTy
1	65.0	8450	7	5	2003	2003	196.0	706.0	0.0	150.0	...	0	
2	80.0	9600	6	8	1976	1976	0.0	978.0	0.0	284.0	...	0	
3	68.0	11250	7	5	2001	2002	162.0	486.0	0.0	434.0	...	0	
4	60.0	9550	7	5	1915	1970	0.0	216.0	0.0	540.0	...	0	
5	84.0	14260	8	5	2000	2000	350.0	655.0	0.0	490.0	...	0	

5 rows × 303 columns

There are still existing some problems after dealing with the numerical variables, such as data missing.

```
In [14]: all_dummy_df.isnull().sum().sort_values(ascending=False).head(10)

Out[14]: LotFrontage     486
         GarageYrBlt     159
         MasVnrArea       23
         BsmtHalfBath      2
         BsmtFullBath      2
         BsmtFinSF2        1
         GarageCars        1
         TotalBsmtSF       1
         BsmtUnfSF         1
         GarageArea        1
         dtype: int64
```

```
In [15]: mean_cols = all_dummy_df.mean()
         mean_cols.head(10)

Out[15]: LotFrontage        69.305795
         LotArea         10168.114080
         OverallQual         6.089072
         OverallCond         5.564577
         YearBuilt        1971.312778
         YearRemodAdd     1984.264474
         MasVnrArea        102.201312
         BsmtFinSF1        441.423235
         BsmtFinSF2         49.582248
         BsmtUnfSF         560.772104
         dtype: float64
```

From this set of code, we found that the most missing column is LotFrontage. Then, we will fill in the gaps with averages.

```
In [16]: all_dummy_df = all_dummy_df.fillna(mean_cols)

In [17]: all_dummy_df.isnull().sum().sum()

Out[17]: 0
```

In order to make the data points smoother and easier to calculate, we will standardize the numerical data.

```
In [18]: numeric_cols = all_df.columns[all_df.dtypes != 'object']
         numeric_cols

Out[18]: Index(['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
                'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
                'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
                'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
                'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt',
                'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
                'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
                'MoSold', 'YrSold'],
               dtype='object')

In [19]: numeric_col_means = all_dummy_df.loc[:, numeric_cols].mean()
         numeric_col_std = all_dummy_df.loc[:, numeric_cols].std()
         all_dummy_df.loc[:, numeric_cols] = (all_dummy_df.loc[:, numeric_cols] - numeric_col_means) / numeric_col_std
```

5. Build the models

In the modelling part, to get a better prediction, we tried different models from the python libraries to compare the different results. The following part will explain the modelling details by steps. We used python language to do the modelling analysis because the python has many very good libraries for data analytics. For example, the Numpy, the Sklearn, the Matplotlib, the Seaborn, etc.

The first step, we use the dummy to get the data frame for train and test. The shape function of dummy checked the total columns and rows of the train.csv and test.csv file as shown in the below figure.

```
In [152]: dummy_train_df = all_dummy_df.loc[train_df.index]
          dummy_test_df = all_dummy_df.loc[test_df.index]

In [153]: dummy_train_df.shape, dummy_test_df.shape

Out[153]: ((1460, 303), (1459, 303))
```

a. Linear Ridge Regression

In the second step, we tried the ridge regression model. Because the dataset has multiple attributes as factors, so choosing ridge regression is suitable for dealing with the dataset like

this. As shown in the below code, the ridge linear model is imported from the Sklearn library. The cross-validation function is also imported for the later use.

```
In [154]: from sklearn.linear_model import Ridge
          from sklearn.model_selection import cross_val_score
```

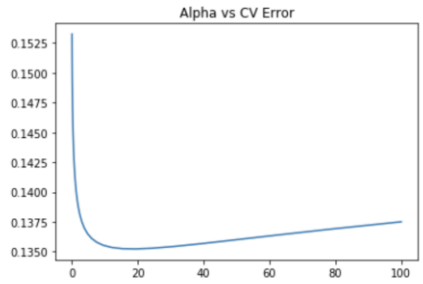
The values of the data frame are transformed into Numpy arrays. This step is to make the dataset more suitable to be used by the Sklearn library.

```
In [155]: X_train = dummy_train_df.values
          X_test = dummy_test_df.values
```

Here we use the cross-validation function of the Sklearn library to test the model. Then save the Cross-Validation(CV) error value to compare with the Alpha value for choosing a better value. As we can see in the output graph when the alpha in the range between 10 to 20, the test_scores has the lowest value of approximately 0.135.

```
In [156]: alphas = np.logspace(-3, 2, 50)
          test_scores = []
          for alpha in alphas:
              clf = Ridge(alpha)
              test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=10, scoring='neg_mean_squared_error'))
              test_scores.append(np.mean(test_score))
```

```
In [157]: import matplotlib.pyplot as plt
          %matplotlib inline
          plt.plot(alphas, test_scores)
          plt.title("Alpha vs CV Error");
```



b. Random Forest

The second model we tried is the random forest. Random forest is an ensemble learning algorithm for classifications and regressions. It constructs a multitude of decision trees. Random forest could solve the overfitting problem for the decision trees' habit. Random forest is one of the commonly used algorithms by the data analyst.

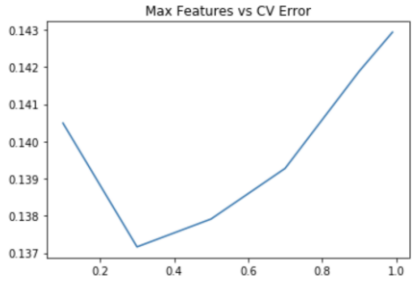
```
In [158]: from sklearn.ensemble import RandomForestRegressor
```

The same step as in the previous model, importing the random forest regressor from the ensemble Sklearn library.

```
In [159]: max_features = [.1, .3, .5, .7, .9, .99]
          test_scores = []
          for max_feat in max_features:
              clf = RandomForestRegressor(n_estimators=200, max_features=max_feat)
              test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=5, scoring='neg_mean_squared_error'))
              test_scores.append(np.mean(test_score))
```

Then we applied the random forest model. The estimators are set to 200. The result of the max features and the test scores are outputted as a graph showing below. In the results of the Random Forest, the best score is around 0.137.

```
In [160]: plt.plot(max_features, test_scores)
          plt.title("Max Features vs CV Error");
```



c. Ensemble

In machine learning, the ensemble methods use multiple learning algorithms to obtain the better predictive performance. Stacking two or more different models can obtain better results than the single learning algorithm model. The combined model may benefit from the advantages of the different model. In order to get a better model, the best parameters should be selected.

```
In [161]: ridge = Ridge(alpha=15)
          rf = RandomForestRegressor(n_estimators=500, max_features=.3)
```

```
In [162]: ridge.fit(X_train, y_train)
          rf.fit(X_train, y_train)

Out[162]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features=0.3, max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Since we have already imported the necessary libraries and models at the beginning of the modelling section, we do not need to import any other library here. Since the single classifier's effects are limited, we prefer to combine multiple classifiers to get a good result. And the Ridge (Alpha = 15) gives a better result than others, so Ridge (15) is selected. Then the selected parameters are applied as shown above.

Because we applied the log (price +1) to smooth the data in the previous data preprocessing section, so we need to transform the data back at this stage. Here we use the expm1 function to reverse the data back. After this step, we could have the final y value which is our predicted SalePrice value now.

```
In [163]: y_ridge = np.expm1(ridge.predict(X_test))
          y_rf = np.expm1(rf.predict(X_test))
```

```
In [164]: y_final = (y_ridge + y_rf) / 2
```

d. Bootstrap Aggregating (Bagging)

Bootstrap aggregating is one of the ensemble methods. The difference of bagging distinguished from the other ensemble models is that it improved the stability and accuracy of the machine learning algorithm in regression and statistical classifications. It also reduces the variance and helps the overfitting problem. Bagging aggregated many small classifiers and combine the results by voting. Here we tried bagging with the bagging regressor from the Sklearn library.

```
In [24]: from sklearn.ensemble import BaggingRegressor
         from sklearn.model_selection import cross_val_score
```

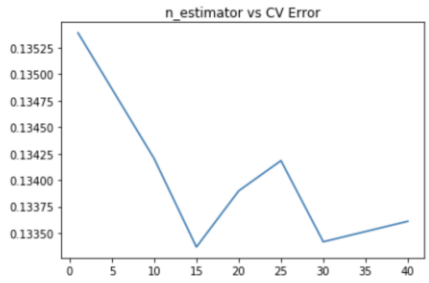
The Bagging Regressor is imported first. The Cross-Validation is imported for testing the influences of the numbers of different classifiers.

The thing needs to pay attention is that the base estimator needs to be filled with the ridge classifier.

```
In [25]: params = [1, 10, 15, 20, 25, 30, 40]
         test_scores = []
         for param in params:
             clf = BaggingRegressor(n_estimators=param, base_estimator=ridge)
             test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=10, scoring='neg_mean_squared_error'))
             test_scores.append(np.mean(test_score))
```

The result is shown in the figure below. As we can see that the lowest score is about 0.132 now which is better than the previous ensemble model.

```
In [26]: import matplotlib.pyplot as plt
         %matplotlib inline
         plt.plot(params, test_scores)
         plt.title("n_estimator vs CV Error");
```



e. Boosting

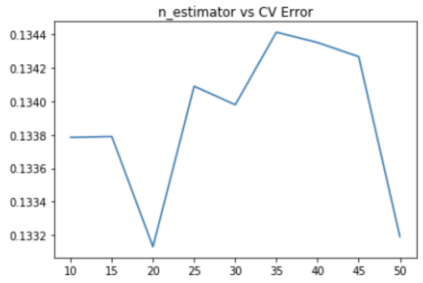
Boosting is another ensemble method which is more advanced than the bagging model. Boosting could adjust the weight of the different classifiers in the learning process.

```
In [29]: from sklearn.ensemble import AdaBoostRegressor
```

The same step applied for importing the necessary model. The AdaBoost model is used in this trial. As we can see in the result below, the combination of AdaBoost and Ridge reached the score of 0.132 which is a good score as well.

```
In [30]: params = [10, 15, 20, 25, 30, 35, 40, 45, 50]
         test_scores = []
         for param in params:
             clf = BaggingRegressor(n_estimators=param, base_estimator=ridge)
             test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=10, scoring='neg_mean_squared_error'))
             test_scores.append(np.mean(test_score))
```

```
In [31]: plt.plot(params, test_scores)
         plt.title("n_estimator vs CV Error");
```



f. EXtreme Gradient Boost(XGBoost)

XGBoost is the shortage name of the extreme gradient boost. XGBoost is one of the boosting models. However, the XGBoost made some improvements compared with the AdaBoost.

The XGBoost algorithm is developed in recent years. The model is adopted by a lot of Kaggle competitors. It is a very popular model in the Kaggle competitions. We also tried this popular model for our house prices predictions.

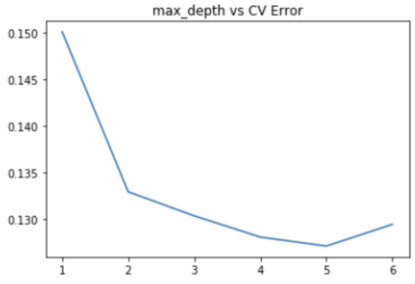
Firstly, import the model as usual. Then we use the Cross-Validation function to test the model.

```
In [34]: from xgboost import XGBRegressor
```

```
In [35]: params = [1,2,3,4,5,6]
         test_scores = []
         for param in params:
             clf = XGBRegressor(max_depth=param)
             test_score = np.sqrt(-cross_val_score(clf, X_train, y_train, cv=10, scoring='neg_mean_squared_error'))
             test_scores.append(np.mean(test_score))
```

The cross-validation score is printed out as below. When the max depth equals to 5, the lowest CV Error score reaches to 0.127 which is the best score for now.

```
In [36]: import matplotlib.pyplot as plt
         %matplotlib inline
         plt.plot(params, test_scores)
         plt.title("max_depth vs CV Error");
```



g. Prediction result

The last step is getting the result of our data analytics model to the prediction. The following image below shows the prediction output result. The first 10 prediction results of the SalePrice are listed here.

```
In [34]: prediction_df = pd.DataFrame(data= {'Id' : test_df.index, 'SalePrice': y_final})
```

```
In [35]: prediction_df.head(10)
```

Out[35]:

	Id	SalePrice
0	1461	120118.843363
1	1462	150534.034444
2	1463	174949.171993
3	1464	190033.219358
4	1465	195403.971822
5	1466	176120.940101
6	1467	177339.908296
7	1468	169171.953945
8	1469	184306.081042
9	1470	123191.998070

6. Evaluation

Finally, we picked the Extreme Gradient Boost model as our suitable model to predict the house prices. The reason for choosing the XGBoost is because of its lowest text scores in our training process. The things that we think we may do in the future is to try more attributes for analysis in the later works. The dataset is a huge dataset with very complicated attributes. Some other attributes that we did not try in this report may have influences to the predictions. We do not have enough time to go through other possible models. And the weight of the different factors that influences the house prices may be different. In the real world cases, some single factor could play more important weight to the house prices. And the house prices is not only influenced by the house related attributes, it may be influenced by other political or environmental factors. However, our prediction model will provide great helps for the customer to analysis the value of the houses. The model can be used by companies and customers to predict the house prices as a suggestion to help them make the decisions. That is the meaning of the data analytics. Make the data analytics benefits for the business strategy and decisions it the good effect that data analytics can do to our real life.

7. References

Vorhies, W. 2016, *CRISP-DM – a Standard Methodology to Ensure a Good Outcome*, Datasciencecentral.com, viewed 24 September 2018, <https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome>