House Prices

- Advanced Regression Techniques -

Project Report

Aisha Alryeh Mkean Alaagib

Aissatou Paye

Enoch Tetteh

Jonas Ngnawe

Yasser Salah Eddine Bouchareb

African Master of Machine Intelligence (AMMI)
Quantum Leap Africa (QLA)

African Master of Machine Intelligence
Quantum Leap Africa
https://aims-ammi.com/

Title:
House Prices

Theme:
Advanced Regression Techniques

Project Period:
26 Nov - 03 Dec 2018

Project Group:
001

Participant(s):
Aisha Alryeh Mkean Alaagib
Aissatou Paye
Enoch Tetteh
Jonas Ngnawe
Yasser Salah Eddine Bouchareb

Supervisor(s):
Marc Deisenroth

Copies: 1

Page Numbers: 33

Date of Completion:
December 7, 2018

Abstract:

Regression is one of common tasks in Machine Learning and many regression techniques more or less complex has been developed throughout the years. To have good predictive models, one need to be chosen well in order. This project is about building a predictive model to predict sales prices of residential properties. It is a currently ongoing Kaggle competition problem, where thousands of people around the world which makes it an excellent challenge. There are many approaches to build this predictive model going from using simple regression models like linear regression to more advanced like lasso or boosting algorithms, and sometimes using ensemble methods to get better results. Our approach is to use advanced regression techniques and stack them: first we apply a average a certain number of base models and boosting models, then stack the averaged model with boosting algorithms. The results obtained are quite satisfactory as they are comparable to the current top 10 submitted results to the Kaggle competition.

Contents

Preface ix

1 Introduction 1
- 1.1 Problem Statement and Objective . . . 1
- 1.2 Presentation of the Dataset . . . 1
  - 1.2.1 History of the Dataset . . . 1
  - 1.2.2 The Dataset in Details . . . 2
- 1.3 Plan . . . 4

2 Data Cleaning and Feature Engineering 5
- 2.1 Overview . . . 5
- 2.2 Data Visualization and Cleaning . . . 6
  - 2.2.1 Missing values and Outliers: . . . 7
  - 2.2.2 Exploring and Visualizing . . . 9
  - 2.2.3 Some Univariables analysis . . . 13
- 2.3 Feature Engineering . . . 19
  - 2.3.1 Skewness . . . 20
  - 2.3.2 Features Choosing: . . . 20

3 Modeling 21
- 3.1 Overview . . . 21
- 3.2 Metric . . . 21
- 3.3 Stacking . . . 22
  - 3.3.1 Base Algorithms . . . 22
  - 3.3.2 Boosting Algorithms . . . 23
- 3.4 Hyper-parameters Tuning . . . 23
- 3.5 Model Selection . . . 24
- 3.6 Scikitlearn library . . . 24

4 Results and Discussions 27
- 4.1 Hyper-parameters tuning results . . . 27
- 4.2 Training the models . . . 27

5 Conclusion 31

Bibliography 33

Todo list

Preface

This is a report on a group project for the Foundations of Machine Learning course at the African Masters of Machine Intelligence. The project itself is a Kaggle competition - "House Prices: Advances regression techniques". This project was run for two weeks from 22nd november to 8th december.
We, the undersigned, hereby declare that the work contained in this report is our original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

AIMS-AMMI, December 7, 2018

Jonas NGNAWE
<jngnawe@aimsammi.org>

Aisha ALRYEH MKEAN ALAAGIB
<aalaagib@aimsammi.org>

Aissatou PAYE
<apaye@aimsammi.org>

Yasser BOUCHAREB
<ybouchareb@aimsammi.org>

Enoch TETTEH
<etetteh@aimsammi.org>

Chapter 1

Introduction

1.1 Problem Statement and Objective

We are given a dataset of sales prices of houses (residential properties) along with different features of these houses that influences the price. The objective of the project is to build a regression model to predict the sales prizes of these residential properties. The idea is to use advanced regression techniques (beyond linear regression) to create an accurate model for prediction. For this purpose, we went through a classical process of : initial visualization, cleaning, feature engineering, visualization of cleaned data, modeling and assessment (test).

1.2 Presentation of the Dataset

The dataset we are working with is about the sale prices of residential properties in Ames, Iowa. It has 2390 observations (residential homes) with 80 explanatory variables (including the "id") which represent various features and aspects of the properties in Ames. The explanatory variables are of all types: 26 categorical(23 nominal, 23 ordinal) and 34 numerical (14 discrete, and 20 continuous).

1.2.1 History of the Dataset

Our dataset, the "Ames Housing Dataset" (AHD) [3] was built by Dean Cock for education purposes. Dean Cock is a professor of statistics at Truman State University in United States, and created this dataset for the end of semester group projects on regression. Before using the AHD, he used to work with the popular "Boston Housing Dataset"[6] which only had 500 observations and 14 variables, and moreover the prices were completely obsolete as they date from the 70's. In order to have a more substantial and actual dataset, Dean Cock started searching for a new dataset and found fortuitously the AHD at the Ames City Assessor's Office. The initial dataset given to him in form of an Excel Sheet contained 113 variables describing 3970 property sales that had occurred in Ames, Iowa between 2006 and 2010. The variables were constituted of nominal, cardinal, discrete, continuous variables plus compounded variables used by the City Assessor's Office for their need. All the sales did not concern only residential properties but also properties like stand-alone garages, condos, and storage areas; and moreover there are some properties occurring many times with their sale prices throughout the years. To get a friendlier and usable dataset he removed all the observations that does not concern residential homes, and for a property only keep the most recent price, then removing all the compounded variables and those that were specific to the City Assessor's Office requiring a special knowledge. At the end of this process, the AHD was born.

1.2.2 The Dataset in Details

The categorical variables

A categorical variable is a variable which takes a value in fixed list of values. A value is called a category or class. A categorical variable can be **nominal** in, which case there is no particular order among the categories or **ordinal**, where the order of the categories matter. AHD contains 23 ordinal and 23 nominal variables with the number of classes for each class ranging from 2 to 28.

Variable	**Description**	**Categories**
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
Alley	Type of alley access to property	– Grvl Gravel – Pave Paved – NA No alley access
CentralAir	Central air conditioning	– N No – Y Yes

Table 1.1: Example of nominal variables[7]

Variable	**Description**	**Categories**
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
PoolQC	Pool quality	– Ex Excellent – Gd Good – TA Average/Typical – Fa Fair – NA No Pool

Table 1.2: Example of ordinal variables [7]

The discrete variables

The 14 discrete variables typically quantify the number of items occurring within the house. Some discrete variables are : *Bedroom* (number of bedrooms above grade -does NOT include basement bedrooms), *Kitchen* (Kitchens above grade), *GarageCars* (Size of garage in car capacity), *TotRmsAbvGrd* (Total rooms above grade - does not include bathrooms).[7]

The continuous variables

In general the 20 continuous variables relate to various area dimensions for each observation. Some continuous variables are: *LotArea* (Lot size in square feet), *GarageArea* (Size of garage in square feet), *1stFlrSF* (first Floor square feet), *2ndFlrSF* (second floor square feet). [7]

1.3 Plan

The sequel of the document is organized as follows:

– The first chapter presents the visualization of the dataset along with the cleaning operations and then we begin our feature extraction procedure.

– The second chapter presents the predictive model we built, the algorithms used, the hyper-parameters tuning and the python library we used.

– Finally in the third chapter, we show the results of the training and the test phase.

Chapter 2

Data Cleaning and Feature Engineering

2.1 Overview

1. **Visualizing an Cleaning**
   - Show certain aspects of the data.
   - Remove features with missing values.
   - (For all features remaining) Remove data points that are outliers (eg: very high price).
   - One hot encoding
   - Visualize the data.
2. **Feature engineering:**
   - Train regression models and select the features corresponding to the best performing models
   - Correlation Matrix
   - Dimensionality Reduction using PCA

2.2 Data Visualization and Cleaning

According to majority of the Data Science courses, most of the data sets are not perfect i.e the data have some missing values and outliers. So before we start to explore and analyze our data, we need to do some cleaning and fill some missing values. Now let us first look at our data (i.e in this part we focus only on the training data set).

Id	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Utilities		PoolQC	Fence	MiscFeature
1	RL	65.0	8450	Pave	NaN	Reg	AllPub	...	NaN	NaN	NaN
2	RL	80.0	9600	Pave	NaN	Reg	AllPub	...	NaN	NaN	NaN
3	RL	68.0	11250	Pave	NaN	IR1	AllPub	...	NaN	NaN	NaN
4	RL	60.0	9550	Pave	NaN	IR1	AllPub	...	NaN	NaN	NaN
5	RL	84.0	14260	Pave	NaN	IR1	AllPub	...	NaN	NaN	NaN

Table 2.1: Some of the Training Data set

Id	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Utilities		PoolQC	Fence	MiscFeature
1461	RH	80.0	11622	Pave	NaN	Reg	AllPub	...	120	0	NaN
1462	RL	81.0	14267	Pave	NaN	IR1	AllPub	...	0	0	NaN
1463	RL	74.0	13830	Pave	NaN	IR1	AllPub	...	0	0	NaN
1464	RL	78.0	9978	Pave	NaN	IR1	AllPub	...	0	0	NaN
1465	RL	43.0	5005	Pave	NaN	IR1	AllPub	...	144	0	NaN

Table 2.2: Some of the Test Data set

This is a subset of our training data set which contain 80 columns. The first 79 columns represent the features, which are also in the test set. The last column is our target "Sale Price", which we want to predict. The data set contains 1460 rows, and each row contains information about a single house. However, there are some columns that have fewer values and this indicates that there are **missing values**. Also, from our data set we can see that we have different data types. There are:

– Numerical (discrete and continuous)

– Categorical (nominal and ordinal)

Our first step was to combine these data sets into a single set both to account for the total missing values and to fully understand all the classes for each categorical variable. Thus, there might be missing values or different class types in the test set that are not in the training set.) and then sure we can not do numerical analysis on string so we convert all the features to one data type (Numerical). Once we do that we can start cleaning the data.

2.2.1 Missing values and Outliers:

Machine learning algorithms do not handle missing values very well, so we must obtain an understanding of the missing values in our data to determine the best way to handle them. We find that 34 (see Figure 2.4) of the features have values that are missing (i.e., "NA" and "aN"). Below we describe examples of some of the ways we treat these missing data.

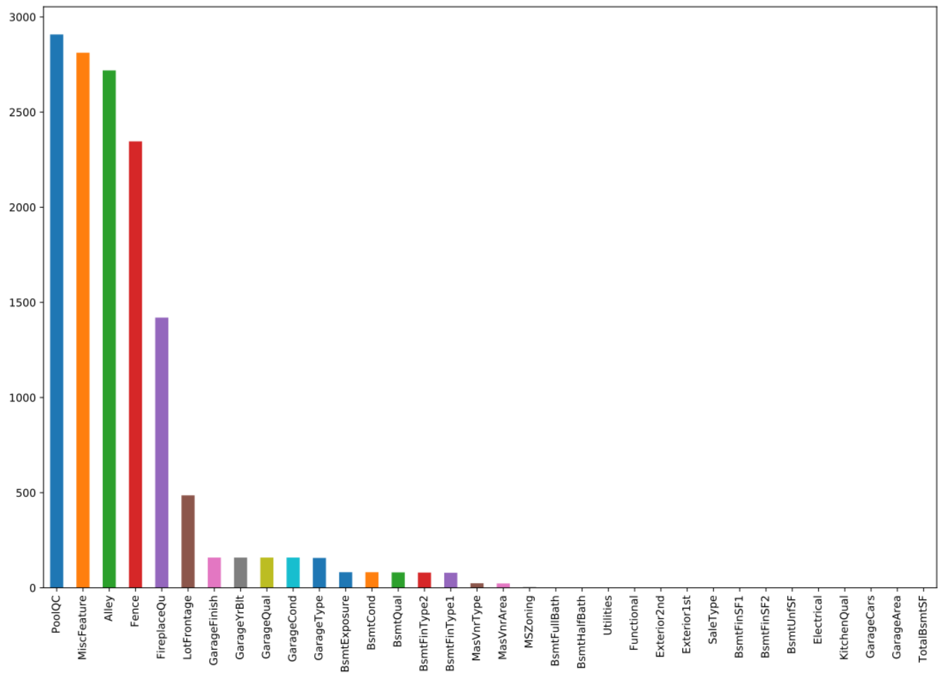

Figure 2.1: Missing Features

This data set had a lot of missing values, but the data description that came with it basically shows us how to handle most of it. For example, Pool Quality is comprised of 5 classes: Excellent, Good, Fair, Typical, and NA. The NA class describes houses that do not have a pool, but our coding languages interpret houses of NA class as a missing value instead of a class of the Pool Quality variable. So we replace it with class "None". Looking at some of the features with the highest number of missing values and the data description [7] we can understand that,

- Alley: Type of alley access to property ("Gravel", "Paved") data description says "NA" means "no alley access" so we replace it with class "None".
- Basement: ("BsmtQual", "BsmtCond", "BsmntExposure", "BsmtFinType1", "BsmtFinType2") data description says "NA" means "no basement" so we replace all them with class "None".
- FireplaceQU: Fireplace quality, data description says "NA" means "No Fireplace" so we replace it with class "None".

– Garage: ("GarageType", "GarageFinish","GaraGarageQual", "GarageCond") data description says "NA" means "No Garage" so we replace it with class "None".

Fence : Fence quality, data description says "NA" means "No Fence" so we replace it with class "None".

– MiscFeature: Miscellaneous feature not covered in other categories, data description says "NA" means "None" so we replace it with class "None".

Some variables have a moderate amount of missing values. For example, about 16.6% of the houses are missing the continuous variable, Lot Frontage, the linear feet of street connected to the property. Intuitively, attributes related to the size of a house are likely important factors regarding the price of the house. Therefore, dropping these variables seems ill-advised.

Our solution is based on the assumption that houses in the same neighborhood likely have similar features. Thus, we impute the missing Lot Frontage values based on the median Lot Frontage for the neighborhood in which the house with missing value was located.

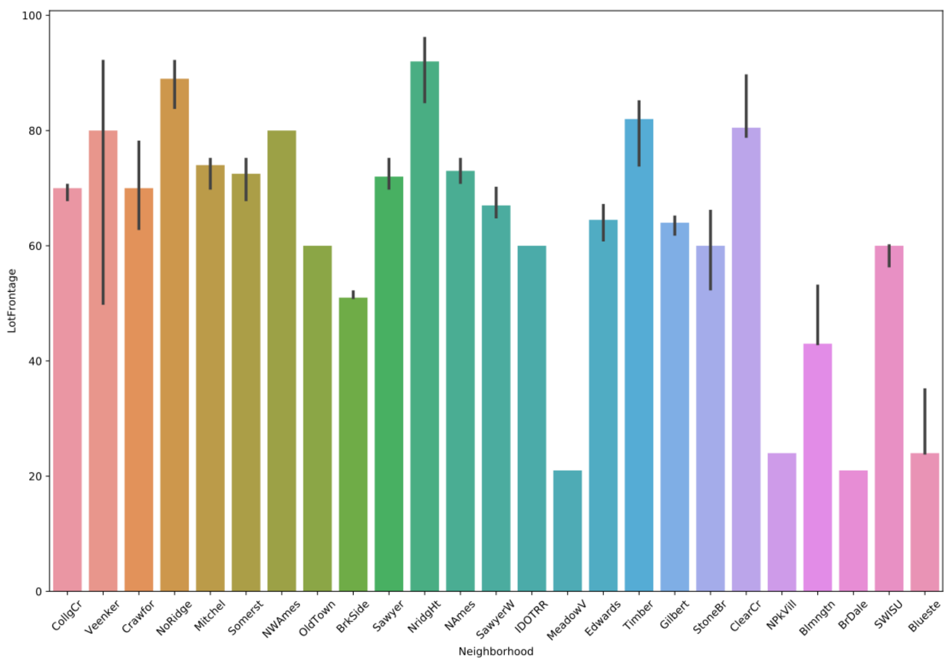

Figure 2.2: Caption

Most variables have some intuitive relationship to other variables, and imputation can be based on these related features. But some missing values are found in variables with no apparent relation to others. For example, the Electrical variable, which describes the electrical system, was missing for a single observation. Our solution was to simply find the most common class for this categorical variable and impute for this missing value.

Finally, one remaining feature with missing values, is Utilities. This feature within the training dataset has two unique values: "AllPub" and "NoSeWa". With "AllPub" being by far the most common. However, the test dataset has only 1 value for this column "AllPub", which means that it holds no predictive power because it is a constant for all test observations. Therefore, we droped this column.

After we handled the missing values, we need to look for outliers (observations that are abnormally far from other values). And that could be done by viewing the relationship between the features.

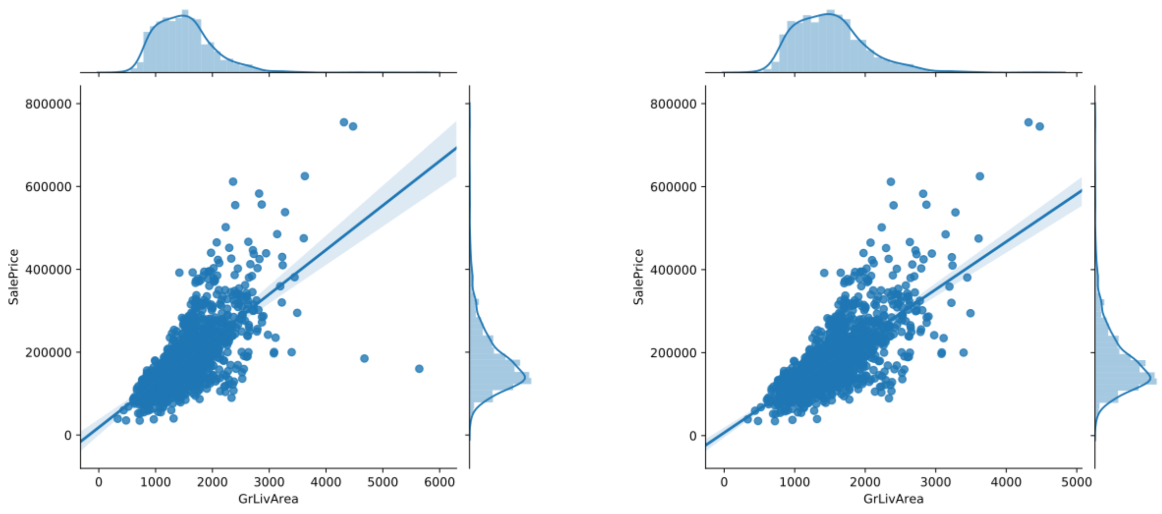

Figure 2.3: Viewing the relationship between Above Ground Living Area and Sale Price, we noticed some very large areas for very low prices.

From the plot above (total live area with the Sale Price) Figure 2.3 it makes sense that people would pay for the more living area, and we can see that there is a strong dependence relationship between them. What does not make sense are some 5 observations, and we indicate them quickly as outliers. Three of them are true outliers (Partial Sales that likely do not represent actual market values) and two of them in the bottom-right of the plot are simply unusual sales (very large houses priced relatively low). Our solution was to simply remove these outliers manually.

2.2.2 Exploring and Visualizing

After the cleaning, we can now explore our data to have more information about the features, and how they are distributed to help us choose the important features. First we see the distribution of some features and distribution of the target "Sale Price". Also, we see the relationship between the Sale Price and the features so we can know the important features and model our data.

Distribution for some features:

By using the Histogram it is clear to see the distribution for each feature and their normality

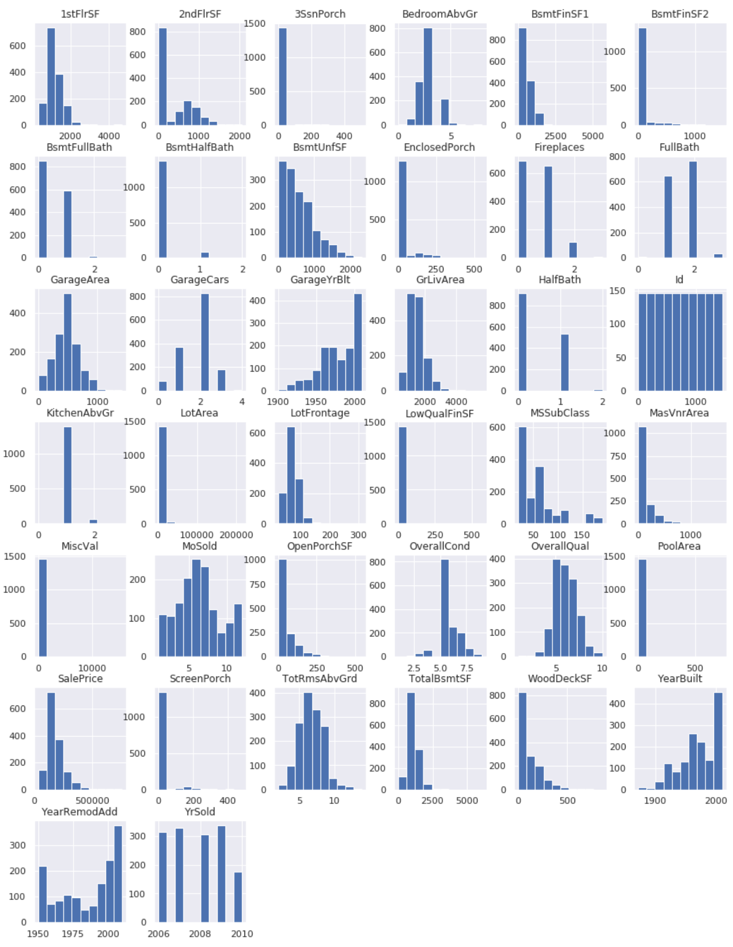

Figure 2.4: Distribution for each features

Analyze the Sale Price:

Our main objective is to calculate the Sale Price. Before we start exploring the data, let's understand the basic statistical features of our dependent variable. From the Figure2.1 below we can see that the Sale Price seems like following the Normal distribution but some data are not following the diagonal line which is representing the normal distribution. But it is easy to handle this problem by using the **Log Transformation** and we can see the result in the Figure2.2. Then we can see that the Sale Price is symmetric and we also see that Kaggle computation is evaluating based on log-Sale Price not the Sale Price.

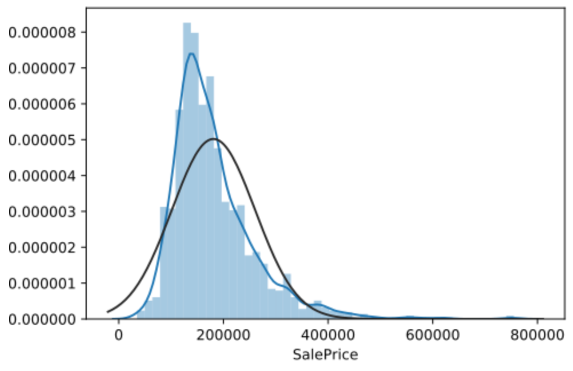

Figure 2.5: Sale Price

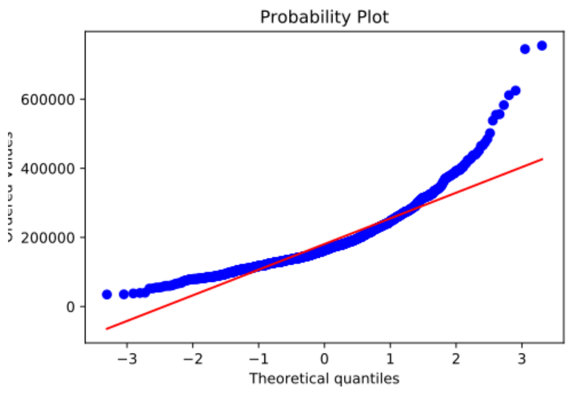

Figure 2.6: Probability plot for Sale Price

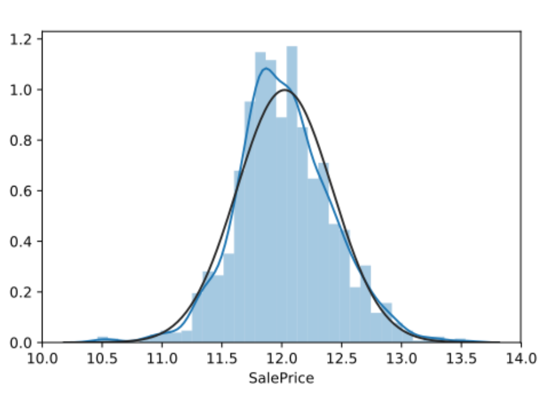

Figure 2.7: Log Sale Price distribution

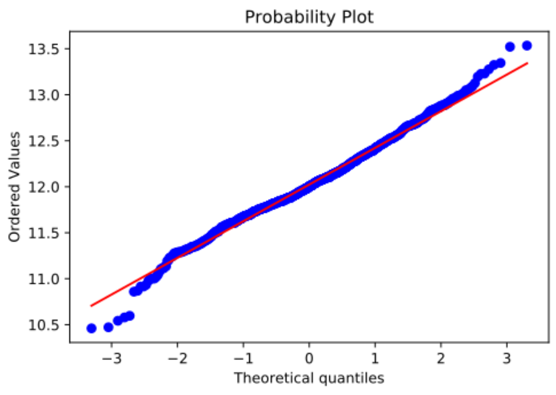

Figure 2.8: Probability plot for Log Sale Price

Now let us proceed to see the relation and correlation between the Sale Price and others features. We analysis this by build a (Correlation matrix "Heat-Map Matrix") which give us the correlation between all the features and to be more clear we zoom this matrix by picking the most correlated variables with the Sale Price.

We can conclude from Figure 2.9:

– The higher correlated variables with the SalePrice are "OverallQual", "GrLivArea" and "TotalBsmtSF".

– Also the "GrLivArea" and "TotalBsmtSF" have the strong correlation.

– The "GarageArea" and "GarageCars" have the strong correlation between them and also with the SalePrice so we can choose one as important feature for prediction so it will be the "GarageCars" since it has higher score(0.64).

– "TotRmsAbvGrd" and "GrLivArea" are high correlated to each other so also we can choose one for prediction.

– "YearBulit" and GarageYrBlt" also are correlated to each other.

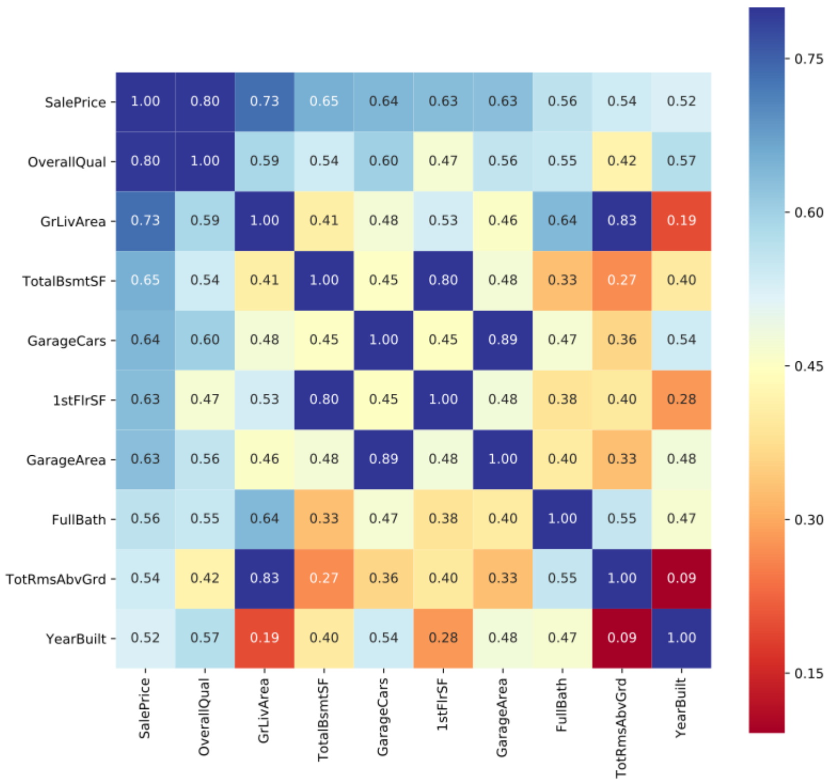

Figure 2.9: Correlation matrix show the correlation between the Sale Price and the features which help us to know the most important features for our prediction

We can see the scatter plot (Figure 2.10) give us more information about the relation between the Sale Price and the important features and show the correlation between these features.

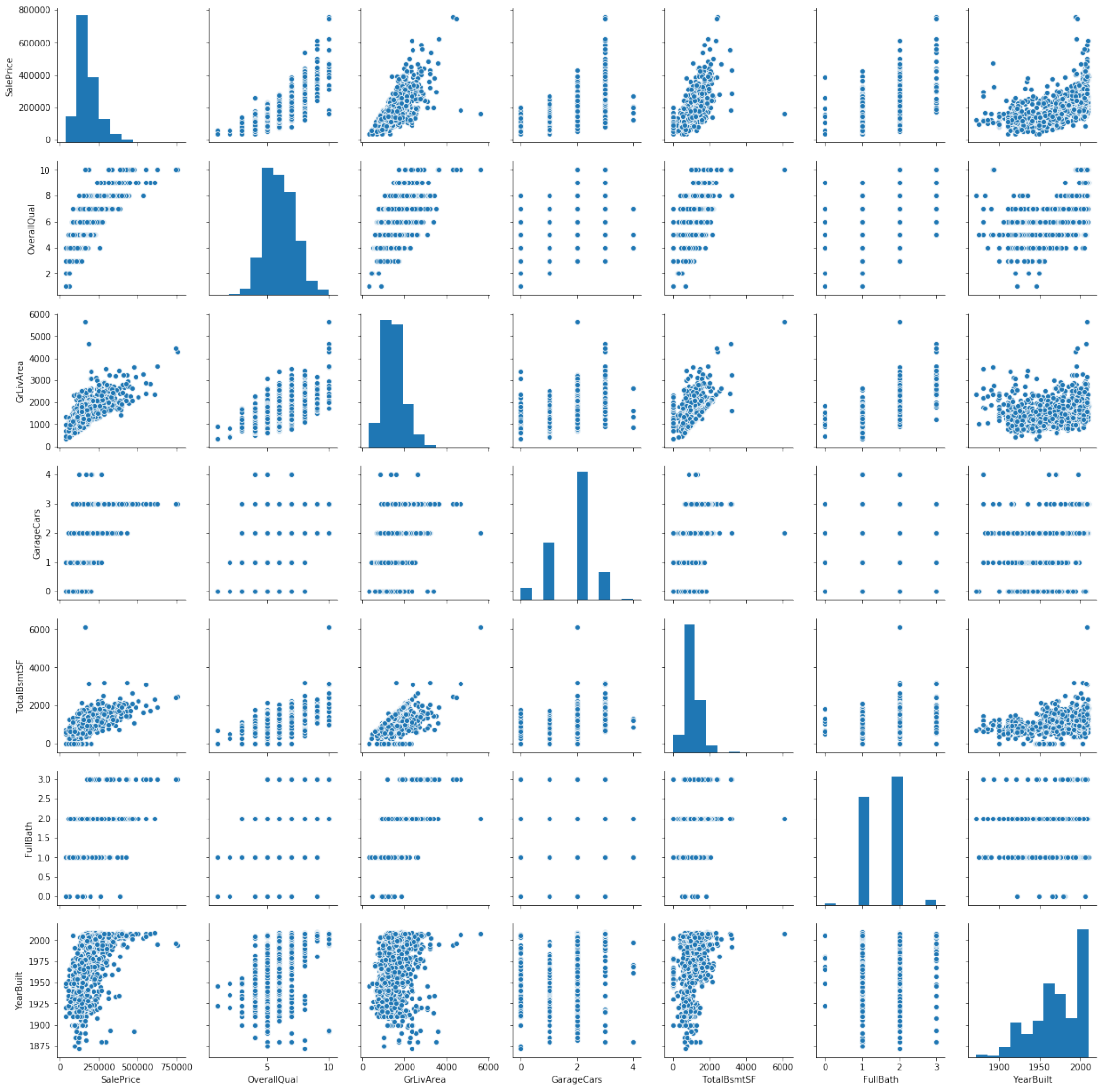

Figure 2.10: Scatter Plot showing the correlation between the features and the Sale Price and also the features with each others

2.2.3 Some Univariables analysis

Here we see the relationships between the Sale Price and some of the features.

Unfinished square feet of basement area

This feature has a significant positive correlation with Sale Price, with a small proportion of data points having a value of 0. This tells us that most houses will have some amount of square feet unfinished within the basement, and this actually positively contributes towards Sale Price. The amount of unfinished square feet also varies widely based on location and style. Whereas the average unfinished square feet within the basement is fairly consistent across the different lot shapes.

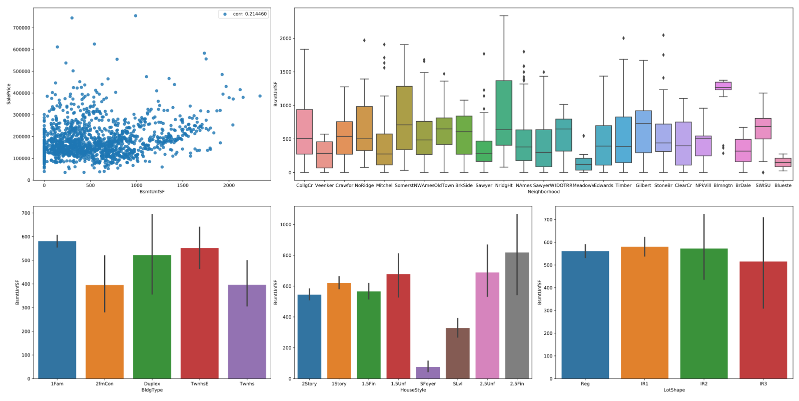

Figure 2.11: Unfinished square feet of basement area and Sale Price

Total square feet of 1st floor area

Clearly this shows a very high positive correlation with Sale Price, this will be an important feature during modeling. Once again, this feature varies greatly across neighborhoods and the size of this feature varies across building types and styles. This feature does not vary so much across the lot size.

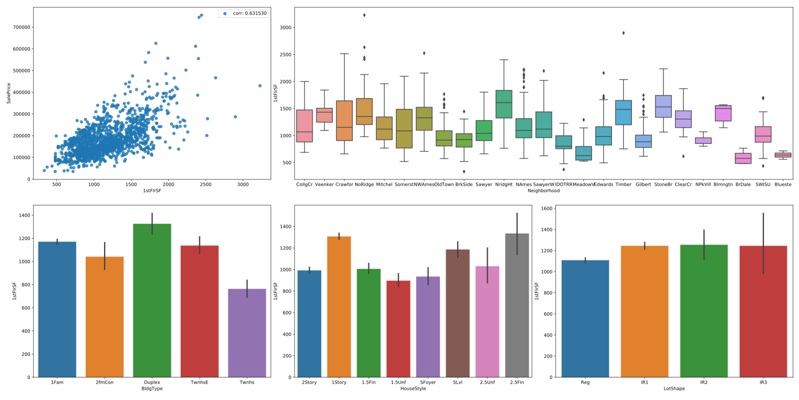

Figure 2.12: Total square feet of 1st floor area and Sale Price

Total square feet of 2nd floor area

Interestingly, we see a highly, positively correlated relationship with Sale Price, however we also see a significant number of houses with value = 0. This is explained with the other visuals, showing that some styles of houses perhaps do not have a second floor, hence cannot have a value for this feature - such as "1Story" houses. We also see a high dependence and variation between neighborhoods, building types and lot sizes. It is evident that all the variables related to "space" are important in this analysis.

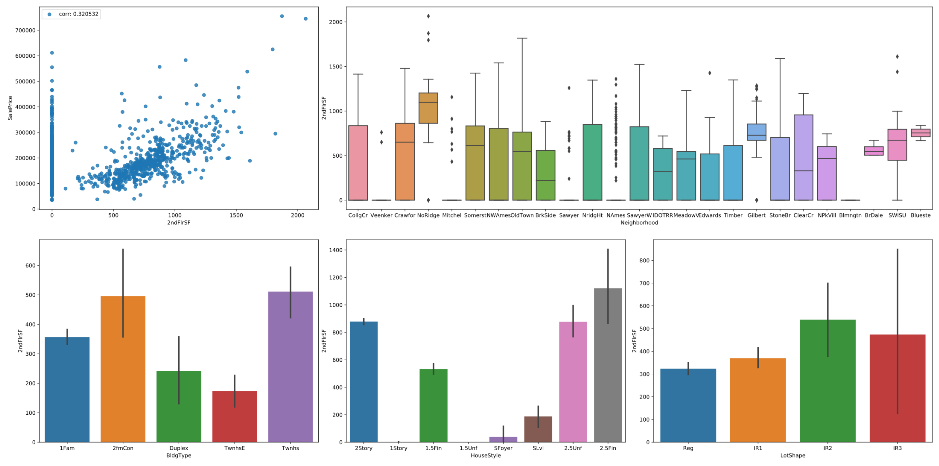

Figure 2.13: Total square feet of 2nd floor area and Sale Price

Over all Quality

This feature, although being numeric is actually categorical and ordinal to be more specific. As the value increases so does the Sale Price. Hence, we will keep it as a numeric feature. We see here a nice positive correlation with the increase in Over all Quality and the Sale Price, as you'd expect.

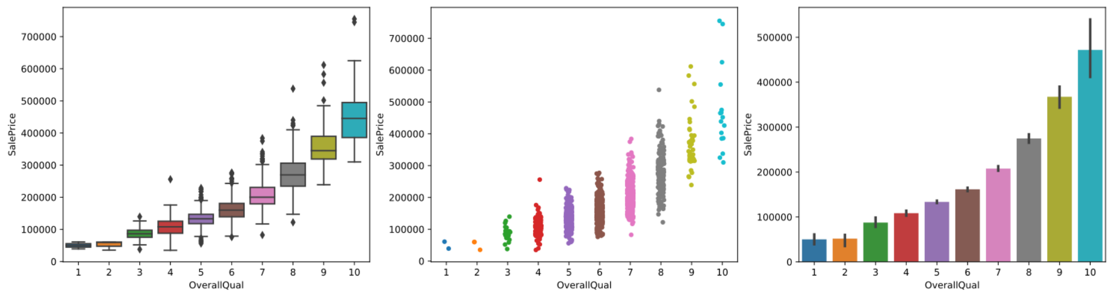

Figure 2.14: Over all Quality and Sale Price

Over all Condition

Interestingly, we see here that it does follow a positive correlation with Sale Price, however we see a peak at a value of 5, along with a high number of observations at this value. The highest average Sale Price actually comes from a value of 5 as opposed to 10, which may be a reasonable assumption

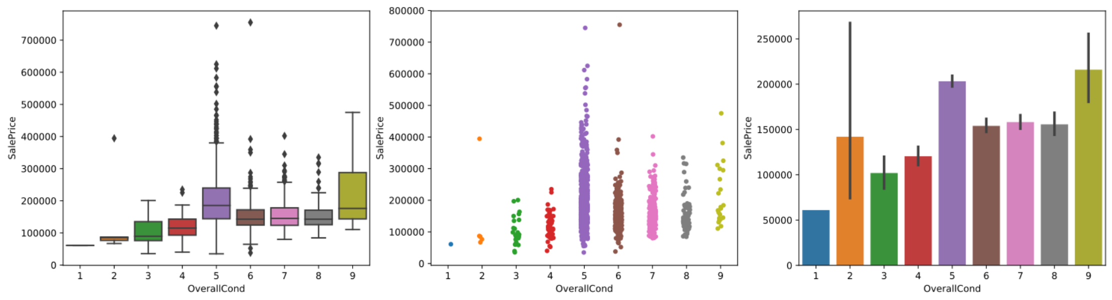

Figure 2.15: Over all Condition and Sale Price

Exterior covering on house

Looking at these 2 features together, we can see that they exhibit very similar behaviours against Sale Price. This tells us that they are very closely related.

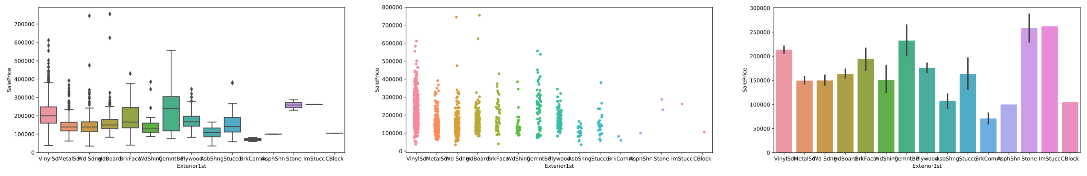

Figure 2.16: Exterior covering on house and Sale Price

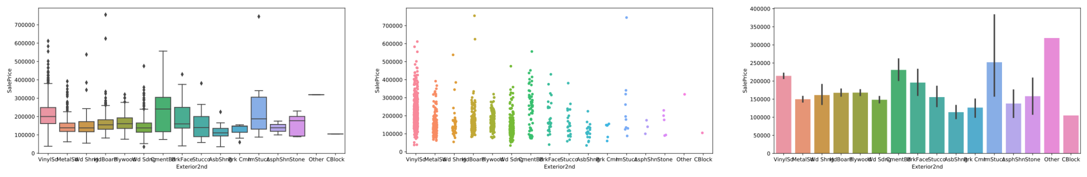

Figure 2.17: Exterior covering on house (if more than one material) and Sale Price

Masonry veneer area in square feet

From this we can see that this feature has negligible correlation with Sale Price, and the values for this feature vary widely based on house type, style and size. Since this feature is insignificant in regards to Sale Price, and it also correlates highly with "Masonry veneer type" (if "MasVnrType = "None" then it has to be equal to 0).

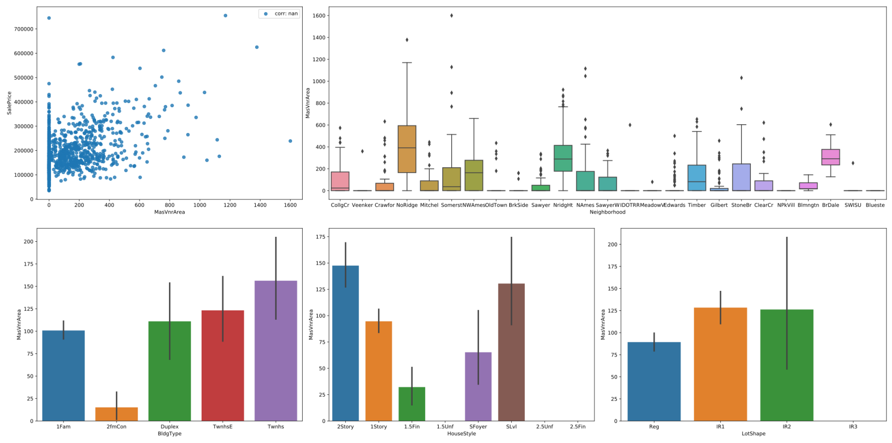

Figure 2.18: Masonry veneer area in square feet and Sale Price

External Quality

We can see here that this feature shows a clear order and has a positive correlation with Sale Price. As the quality increases, so does the Sale Price. We see the largest number of observations within the two middle classes, and the lowest observations within the lowest class.

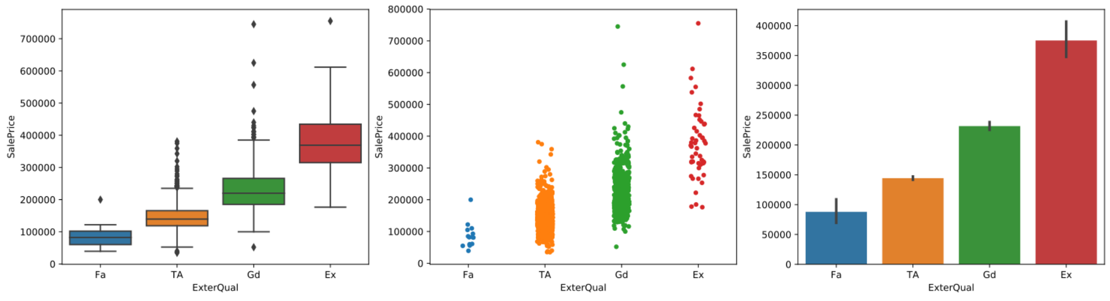

Figure 2.19: External Quality and Sale Price

Size of garage in car capacity

We generally see a positive correlation with an increasing garage car capacity. However, we see a slight dip for 4 cars so we believe due to the low frequency of houses with a 4 car garage.

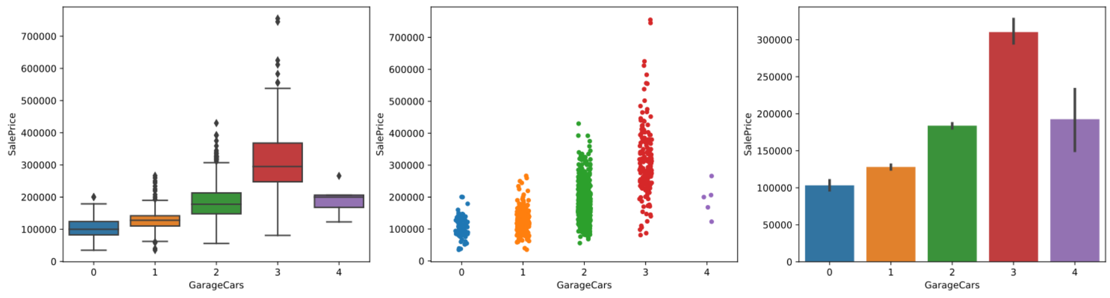

Figure 2.20: Size of garage in car capacity and Sale Price

Size of garage in square feet

This has an extremely high positive correlation with Sale Price, and it is highly dependant on Neighborhood, building type and style of the house. This could be an important feature in the analysis

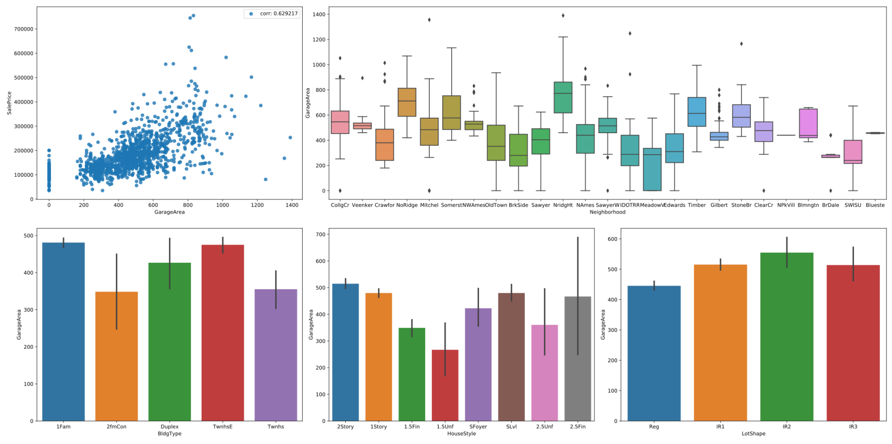

Figure 2.21: Size of garage in square feet and Sale Price

Total Porch area in square feet

Open porch area, enclosed porch, three season porch and screen porch are four features describe the porch area in square feet, we will sum these features together to create a total porch in square feet feature. We can see a high number of data points having a value of 0 here once again.

Apart from this, we see a high positive correlation with Sale Price showing that this may be an influential factor for analysis. Finally, we see that this value ranges widely based on location, building type, style and lot.

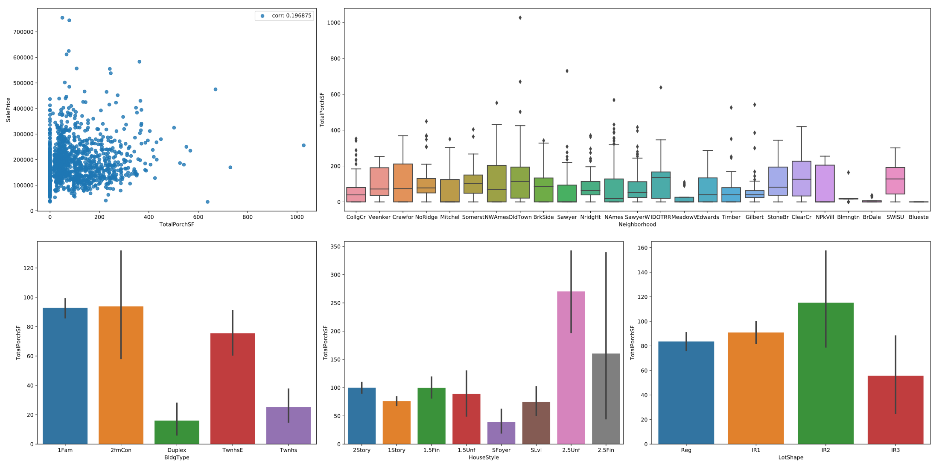

Figure 2.22: Total Porch area in square feet and Sale Price

Heating quality and condition

Here we see a positive correlation with Sale Price as the heating quality increases. With "Ex" bringing the highest average Sale Price.

We also see a high number of houses with this heating quality too, which means most houses had very good heating!

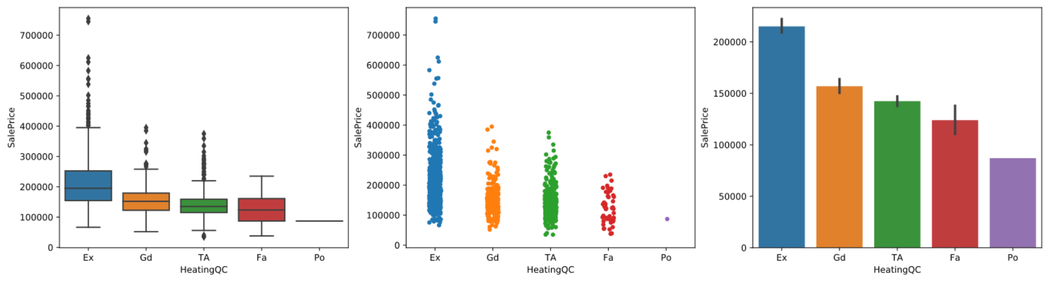

Figure 2.23: Heating quality and condition and Sale Price

2.3 Feature Engineering

Feature engineering is one of the most important steps in model creation. Often there is valuable information "hidden" in the predictors that is only revealed when manipulating these features in some way. Bellow are some transformation and additional feature we have done.

– Transforming year and month into categorical features.

- Transforming overall condition and overall quality into categorical features.
- We created Over all area in square feet a feature that sum the variable that describe the area of different section in a house TotalBsmtSF, 1stFlrSF and 2ndFlrSF.
- Add Remodeled Difference a feature that capture the difference between the year the house was built and the year it was remodeled.

Next, concerning the categorical features we differentiate two types,

- Ordinal Categories where there is an inherent order to the classes (e.g., Excellent is greater than Good, which is greater than Fair), we mention 'OverallQual', 'ExterQual', 'PoolQC', etc...
- Nominal Categories where there is no obvious order (e.g.,'CollgCr', 'Veenker', and'Crawfor'), we mention 'GarageType', 'SaleCondition', 'HouseStyle', etc...

The problem is that categorical variables must be treated as continuous in the model. So, our solution for ordinal variables was to simply assign the classes a number corresponding to their relative ranks. For example, Kitchen Quality has five classes: Excellent, Good, Typical, Fair, and Poor, which we encoded to the numbers 5, 4, 3, 2, and 1, respectively.

On the other hand the ranking of nominal categories is not appropriate as there is no actual rank among the classes. Our solution was to one-hot encode these variables, which creates a new variable for each class with values of zero (not present) or one (present).

2.3.1 Skewness

While there are few assumptions regarding the independent variables of regression models, often transforming skewed variables to a normal distribution can improve model performance. Our solution was to log transform several of the features.

2.3.2 Features Choosing:

PCA is the one of the techniques that it use to reduce the dimensionality of the data that contain features which has correlation between each other. So the transformed features are the one we used for training the model. To do PCA we use scikit-learn library which does first Singular Value Decomposition (SVD). In the PCA we go through some steps. First, we normalize our data and compute the covariance matrix after and get the eigenvectors and eigenvalues, after we choose the principal components. At the end we transform to get the original data. By PCA in the feature engineering we get 23 variables from 225 [9].

Chapter 3

Modeling

3.1 Overview

How does one choose a particular model or groups of models for a machine learning task? On what criteria is the choice based on, and how about the hyper-parameters to consider?

Given that the purpose of this project is to explore advanced regression techniques, we considered the advantages and disadvantages of several regressors based on domain knowledge. We, then looked at the best performing algorithms on competitions like the famous Kaggle Competition and selected nine of these algorithms.

To know how these algorithms will perform on our data set, we perform a cross validation on our data set with these algorithms, and compute their errors. Based on these results we select the best performing ones for the modelling.

In order to build a more robust model and improve its generalization, we combine the outputs or predictions of several algorithms over a single algorithm. Our goal is to build a model that overcomes the weaknesses of individual models.

3.2 Metric

For this regression problem, we use the Root Mean Square Error (RMSE) as the metric to measure the performance of our model. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{N}(\sum(Y_i - Ypred_i)^2} \tag{3.1}$$

N : number of data
Y: target
Y_{pred} : prediction of model

3.3 Stacking

In modeling, the objective is to have a good generalization capacity (low generalization error) for the predictive model. Techniques that aim to achieve this forms a family called "model ensembling"[8]. In Stacking or meta ensembling ,we have two types of learners: Base learners and Meta learners . This method combine the results of many predictive models to create a (better) new model.[5]. Our strategy will be first averaging over models then stacking the averaged model with other boosting algorithms and make predictions.

3.3.1 Base Algorithms

Ridge Regression

The first of our base models is the popular ridge regression. This model is credited for handling data that suffers from multicollinearity (i.e independent variables are highly correlated). The problem of multicollinearity is solved by "Penalizing" the parameter to be estimated. The effort of this is to end up with a parameter with low variance and the prediction error is thereby reduced. Ridge regression uses L2 regularization.
Objective function:
$L(w) = \|y - w^T x\|_2^2 - \alpha\|w\|_2^2$

Lasso Regression (least absolute shrinkage and selection operator)

Also spelled lasso or LASSO, is a regression method that do both feature selection and (l1) regularization. As the name implies, it shrinks other features and selects the right feature in order to reduce variability. Given a highly correlated features, lasso selects only one and shrinks the others to zero. It is similar to ridge regression, in that it places a "penalty" on the parameters, but uses L1 regularization.
Objective function:
$L(w) = \frac{1}{N}\|y - w^T x\|_2^2 + \alpha\|w\|_1$

Elastic Net

Is a regression technique that combine l1 and l2 regularization. Elastic Net is one of the "powerful " regression algorithm because it combines two useful techniques, the ridge regression and Lasso regression techniques. This enables to handle multi correlated features, as it trades-off between the two techniques.
Objective function: $L(w) = \|y - w^T x\|_2^2 + \alpha_1\|w\|_2^2 + \alpha_2\|w\|_1$

Bayesian Ridge

This technique is similar to the classic rigide but estimate a probabilistic model of regression problem as following:

$$p(w\|\lambda) = N(w\|0, \lambda^{-1} I_p) \tag{3.2}$$

the parameters λ and w are estimated jointly during fitting of model.

Kernel Ridge

This technique kernelizes a ridge regression. That is to say, it combines Ridge regression with the kernel trick. It is similar to support vector regression (SVR), but the only difference is the type of loss function used.

3.3.2 Boosting Algorithms

Boosting is an ensemble technique in which the predictors are sequentially,by combining weak learner after weak learner, the final model is able to account for a lot of the error from the original model and reduces the error.
Boosting algorithms produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. For our modelling , we will the following:

Gradient Boosting

Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. it allows the optimization of differentiable loss function. In each stage a regression tree is fit on the negative gradient of the given loss function. Gradient boosting can be used for all machine learning problem that the gradient of objective functions can be compute.

XGBoost(eXtreme Gradient Boosting)

XGBoost is a scalable and accurate version of gradient boosting algorithm on decision tree (GBDTs) . This model push the limits of computational speed and model accuracy.

CatBoost

This technique is one of the fastest of gradient boosting decision tree . The algorithm uses specific kind of trees ,symmetric trees.Catboost can also handle categorical features of given dataset but if catfeatures has no argument, CatBoost will treat all variables of the given dataset as numerical variables.

LightGBM

Data is increasing again and again and it is becoming difficult for traditional model to give faster results. Light GBM is prefixed as 'Light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run with good accuracy. Similar to CatBoost ,LightGBM successfully handles categorical by using a specific algorithm to find the value of categorical features. LightGBM is sensitive to overfitting ,it is not recommended to use on small datasets.

3.4 Hyper-parameters Tuning

Assuming one has a clean data to work with, the crucial and most important next step to take is choosing the hyper-parameters of the algorithm. Hyper-parameters tuning involves choosing the best parameters for each single algorithm that will give us the best accuracy for the given model. There are many approaches to hyper-parameters tuning like:

- **manual tuning**
- **grid search**
- **random search**
- **Bayesian optimization**
- **black magic etc.**

.

Each of these methods above comes with their own shortfalls. Grid search is computationally expensive since one must search through a sufficiently large space of parameters. The random choice might work but if we do not "guess" close to optimal values it might give very bad results. The manual tuning requires experience that we do not have in this case. Since we are not fans of "Black magic" and none of us is a wizard (at least not yet), we do not opt for that.
Therefore, we chose to do **Bayesian Optimization** which has the main advantage that we will find the parameters that average all possible values for the respective parameters, thus giving more suitable results.
We want to find some hyper-parameters θ of our algorithms such as to minimize an objective function $f(\theta)$: find $\theta_{opt} = \arg\min_{\theta} f(\theta)$. The idea of the Bayesian optimization [4] is choose an optimizing parameter among N evaluations of the objective function. These evaluations are obtained recurrently at points that optimizes an acquisition function which is itself function of the posterior of a Gaussian Process (GP) that approximates the true objective. In order to use Bayesian optimization, one needs to choose the acquisition function and the kernel of the GP. In this work, we use the three following acquisition functions[1]:

– The Expected Improvement : $-EI(\theta) = -\mathbb{E}(f(\theta) - f(\theta_{best}))$

– The Lower Confidence Bound: $LCB(\theta) = \mu_{GP}(\theta) + \kappa\sigma_{GP}(\theta)$, where κ is the knobs to balance the exploitation versus exploration.

– The Expected Improvement: : $-PI(\theta) = \mathbb{P}(f(\theta) \geq f(\theta_{best}) + \kappa$

There are two minimization methods to choose from *skopt* (the scikitlearn module for optimization): *forest_minimize*, *GP_minimize*. For each of them we chose the acquisition function:

– *forest_minimize*: we use the Expected Improvement function for this method because it performed better.

– *GP_minimize*: we use *GP_hedge* which chooses automatically the most appropriate acquisition function at each iteration.

3.5 Model Selection

Model selection is an essential step in the process of developing a predictive model.These are some of the techniques for used :

– **Akaike Information criterion (AIC)**

– **Bayesian Information Criterion (BIC)**

– **stepwise regression ect.**

.

Cross validation is the most used methods to evaluate predictive performances of a model. Given a dataset for a predictive modelling problem, the training data is split into training set and validation set. We first train the model on the training set, test its performers on the validation set, and the models with best performance are selected. For this project, we use **Cross- validation** method to select the model used for stacking.

3.6 Scikitlearn library

`Scikitlearn`https://scikit-learn.org is an opensource Machine Learning library in python. It offers a wide range of tools for doing classification, regression, clustering, dimensionality reduction, model selection and

preprocessing.[2].
Basically, almost all the algorithms implemented in this project are from the scikit learn library including the module used for the Bayesian optimization.

Chapter 4

Results and Discussions

Once we have our cleaned set of datasets (train data and test data) and design our predictive model along with our hyper-parameters tuning method, we train our model and test it. In the following, we present the results of the hyper-parameters tuning, then those of training process, finally we show the generalization performance of our predictive model on the validation dataset.

4.1 Hyper-parameters tuning results

The results of the Bayesian optimization are given in below tree table .

4.2 Training the models

In this part, we build the relationship between SalePrice (dependent variable) and the explanatory variable (independent variable). we have used nine algorithms and make and averaging models of them for prediction. To control the learning algorithms, we define RMSE which gives the performances of models . Cross validation of the RMSE for each model gives us the best score of them and then we can select models. The following table is a summary of score of our model:

Models	Kernel Rigide	Elastic Net	Lasso
Initial parameters	alpha= 0.9826, kernel= 'polynomial', degree= 2, coef0= 3.5	alpha=0.0002835 , l1_ratio=0.9999905 , random_state=0	max_iter=3000, alpha= 0.000281, random_state= 5

Table 4.1: Hyperparameters 1

Models	Bayesian Rigide	XGBoost	Light GBM

Initial parameters	n_iter= 100,
	alpha_1= 6e-05,
	alpha_2= 0.01,
	lambda_1= 0.01,
	lambda_2= 6e-05,
	tol= 1e-05
		learning_rate= 0.0225,
		colsample_bytree= 0.1,
		n_estimators= 2500,
		max_depth= 3,
		subsample= 0.6982,
		min_child_weight=1,
		)
			learning_rate= 0.0166,
			objective= ’regression’,
			num_leaves= 30,
			num_iterations= 1500,
			feature_fraction = 0.2175,
			bagging_fraction= .9870,
			min_data_in_leaf= 9,
			max_depth= 68

Table 4.2: Hyperparameters 2

Models	Rigide	Gradient Boosting	CatBoost

Initial parameters	alpha= 8.8435,
	solver= ’cholesky’
		n_estimators= 3000,
		max_depth= 4,
		min_samples_leaf= 2,
		max_features= ’sqrt’,
		loss=’huber’,
		random_state= 0,
		learning_rate= 0.0223,
		min_samples_split= 100
			iterations= 1929,
			depth= 2,
			learning_rate= 0.0229,
			loss_function= ’RMSE’,
			l2_leaf_reg=0,
			rsm= 1.0,
			one_hot_max_size=194

Table 4.3: Hyperparameters 3

Model	Score
Lasso	0.113
Elastic Net	0.113
Rigide Regression	0.1151
Bayesian rigide	0.1152
Gradient Boosting	0.1097
XGBoost	0.1095
LightGBM	0.1175
CatBoost	0.1187
Kernel Rigide	0.1191
Average Models	0.1094

Table 4.4: Performance of models

Our averaging model is build with Elastic Net , lasso , Gradient Boosting and Rigide Regression which gives best performance.

Chapter 5

Conclusion

The objective of this work was to model a regression model for house prizes using the Ames Housing Dataset(AHD). The AHD is a particularly nice and rich dataset, in the sense that it has a diverse type of comprehensive explanatory variables (nominal, ordinal , discrete and continuous) that describe almost every aspect of a residential home in Ames, Iowa. In order to build a good predictive model, we started first by visualizing and cleaning the dataset (removing and filling missing value, removing outliers , encoding the categorical variables using one hot encoding) which was pretty not complicated as the dataset is already well set, then did feature extraction to keep only important features and have a reasonable dimensionality: for this we used lasso regression and PCA. After this, we built we choosed stacking as our predictive model, i.e averaging some base models with boosting models. Our base models are: ridge regression, lasso, elastic net, kernel ridge; and the boosting models used are: xgboosting, gradient boosting, Cat boosting and LightGBM. To tune our hyper-parameters, we used Baysesian Optimization.
We trained our models and stacked them and the best score (rmse: root mean squared error) we got was 0.1157 which is comparable to the top 10 of the current submissions of Kaggle. We certainly want to improve this score therefore we will do a better feature extraction or try other models or use another stacking policy different from the simple averaging like taking a weighted mean of the stacked models where the weights would be function of the individual scores.

Bibliography

[1] David Cournapeau. *Bayesian optimization with skopt*. `https://scikit-optimize.github.io/notebooks/bayesian-optimization.html`. Blog.

[2] David Cournapeau. *Scikitlearn website homepage*. `https://scikit-learn.org/stable/index.html`. Blog.

[3] Dean De Cock. "Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project". In: *Journal of Statistics Education* 19.3 (2011).

[4] Peter I Frazier. "A Tutorial on Bayesian Optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[5] Ben Gorman. *A Kaggler's Guide to Model Stacking in Practice*. `http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/`. Blog. 2016.

[6] David Harrison Jr and Daniel L Rubinfeld. "Hedonic housing prices and the demand for clean air". In: *Journal of environmental economics and management* 5.1 (1978), pp. 81–102.

[7] Kaggle. *House Prices: Advanced Regression Techniques*. `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data`. Blog.

[8] Jovan Sardinha. *An introduction to model ensembling*. `https://medium.com/weightsandbiases/an-introduction-to-model-ensembling-63effc2ca4b3`. Blog. 2017.

[9] Jiao Yang Wu. "Housing Price prediction Using Support Vector Regression". In: (2017).