UIC University of Illinois at Chicago

A report on

Study of advanced regression models to determine prices for houses in Ames, Iowa based on their features.

Under Guidance of

Dr. Ashkan Sharabiani

By,

Ashish Menkudale (UIN:656130575)
Sourabh Taluja (UIN:655838453)

November, 2016

Contents

List of Figures

LIST OF TABLES

INTRODUCTION

Understanding the customer needs and predicting customer's purchase intents form the core of success for any business. Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this project dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence. For the vastly diversified realty market, with prices of properties increasing exponentially, it becomes very essential to study the factors which affect directly or indirectly when a customer decides to buy a house and to predict the market trend. In general, for any purchase, a potential customer makes the decision based on the value for the money.

The problem statement was taken from the website Kaggle. We chose this specific problem because it provided us an opportunity to build a prediction model for real life problems like prediction of prices for houses in Ames, Iowa.

Also, we wanted to study the Advanced regression techniques and their implementation in machine learning language like python. Understanding the theory and implementing it into practice was a challenge for us.

DATA SET

We had a two different data sets namely train dataset and test data set. Both contained numerous variables in terms of features which were describing a house. Training data set contained 1460 observations for which sale price of the house was provided. Based on this data, a prediction model was to be built. Test data set contained 1459 observations for which the sale price was to be predicted.

In total 80 variables focus on the quality and quantity of many physical attributes of the property. Most of the variables are exactly the type of information that a typical home buyer would want to know about a potential property (e.g. When was it built? How big is the lot? How many square feet of living space is in the dwelling? Is the basement finished? How many bathrooms are there?).

Data set contained 23 nominal variables and 23 ordinal variables. Nominal Includes variables like- the weather condition and material used for construction. For the nominal and ordinal variables, the levels were in the range of 2 to 28. Total of 14 discrete variables comprise the number of kitchens, washrooms, and bedrooms. This also includes the garage capacity and construction or re-modeling dates. 20 continuous variables describe the area dimension of each observation. Lot size and total dwelling square footage are common home listing available online. Area measurements on the basement, porches and main living area are further classified into respective classes based on quality and type.

DATA CLEANING: DATA VISUALIZATION

For every data analysis, initial data cleaning is required. Otherwise outliers in the data set produces wrong prediction models and that can seriously affect the model accuracy. Initially, we decided to analyze our data through visualization. We compared the response variables (sale price) with respect to few important quantitative variables like 'Garage living area', 'Lot frontage' etc. Scatter plots were plotted and few outliers were observed right away.

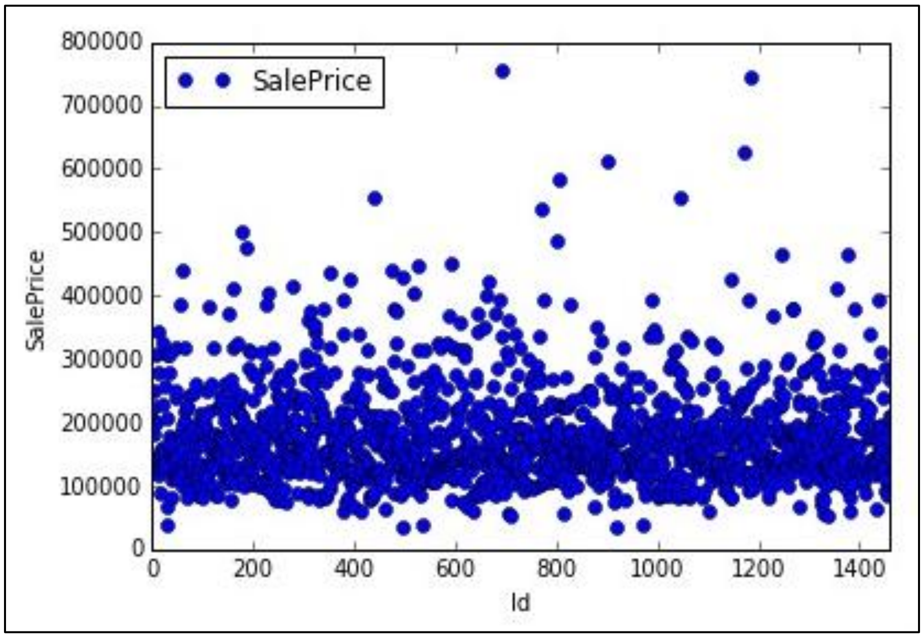


Figure 1: Scatter plot of Id versus Sale price.

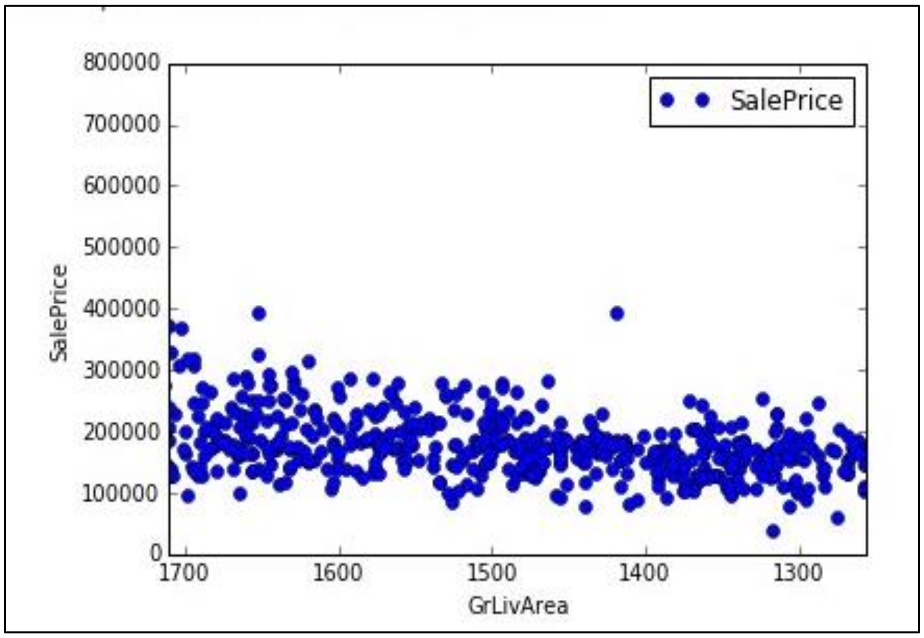


Figure 2: Scatter plot of Garage living area versus Sale price.

Also, box plot helped us to visualize the range of house prices we were dealing with.

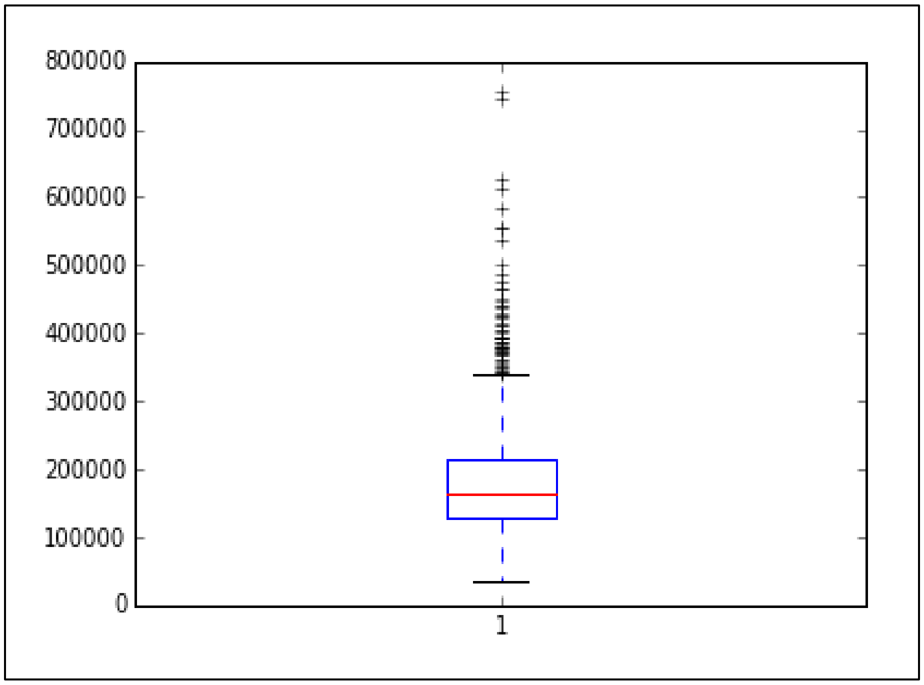


Figure 3: Box plot for Sale price.

Most the sales price are below \$500k. All the value points above \$500k (total =10) is excluded from the prediction model. Thus, outliers were removed through initial screening. To confirm the same outliers, normality check was done in next step.

DATA CLEANING: NORMALITY CHECKS

All statistical methods rely on initial assumption of data being normally distributed. Hence, before applying any statistical method, it should be confirmed whether the data is normally distributed or not.

Histogram was plotted to determine the normality trend of data.

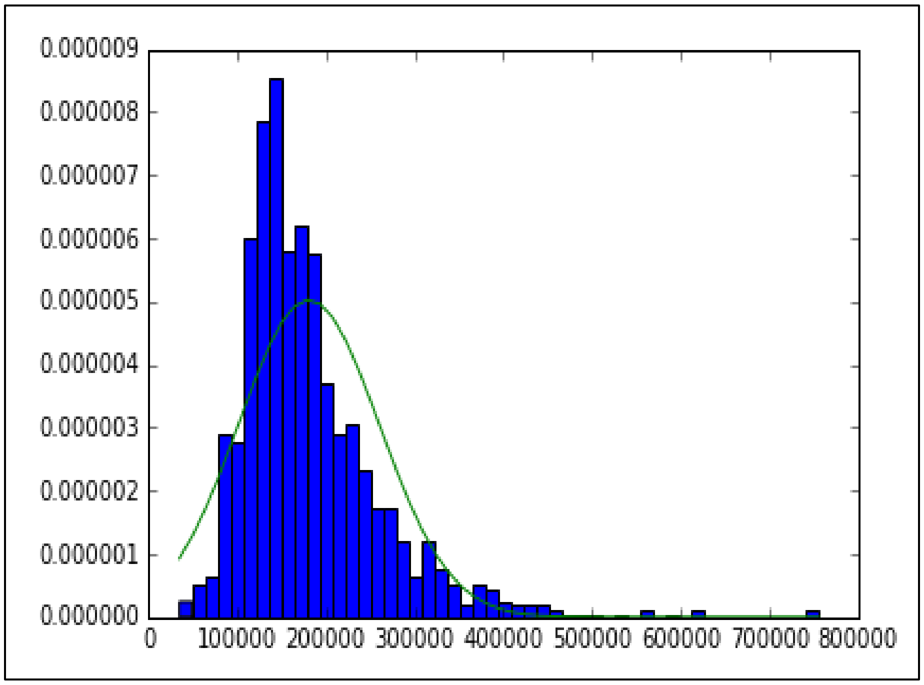


Figure 4: Distribution for Sale price.

The histogram was skewed as the outliers were present at the higher price range. Data points representing sale price more than $ 500K were excluded from the analysis.

DATA CLEANING: MISSING VALUES

Before moving further with building up regression model, we classified our data per different types. For missing values, we had to come up with different strategies as data was incomplete. We identified rows which were having missing values. Missing values are 5% of total observations. In these rows, for the categorical data, zero was substituted in place of missing values. For continuous and discrete data, the missing values are replaced by mean value of entire data of that variable.

- Nominal variables: Missing values taken as 0
- Ordinal variables: Missing values taken as 0
- Discrete variables: Missing values taken as Mean value.
- Continuous variables: Missing values taken as Mean value.

DATA CLEANING: DATA CONVERSION

In the data set, we had few parameters representing year. For these attributes, modifications were made. For the sake of simplicity, we converted this data into a numerical value by subtracting it from current year.

MODEL BUILDING: PREPARING DATA FRAME

After data cleaning, data was needed to be organized into a workable format in python. With the huge data, data frame structure was the most suitable format.

Categorical attributes were converted into dummy variables and baseline was dropped for each feature to reduce the matrix size.

After comparing data from training dataset and test dataset, we observed that some attributes with some levels were not mutually inclusive in both sets. We identified these attributes and removed those. After all this preparation, the final data has 521 columns and 1460 rows

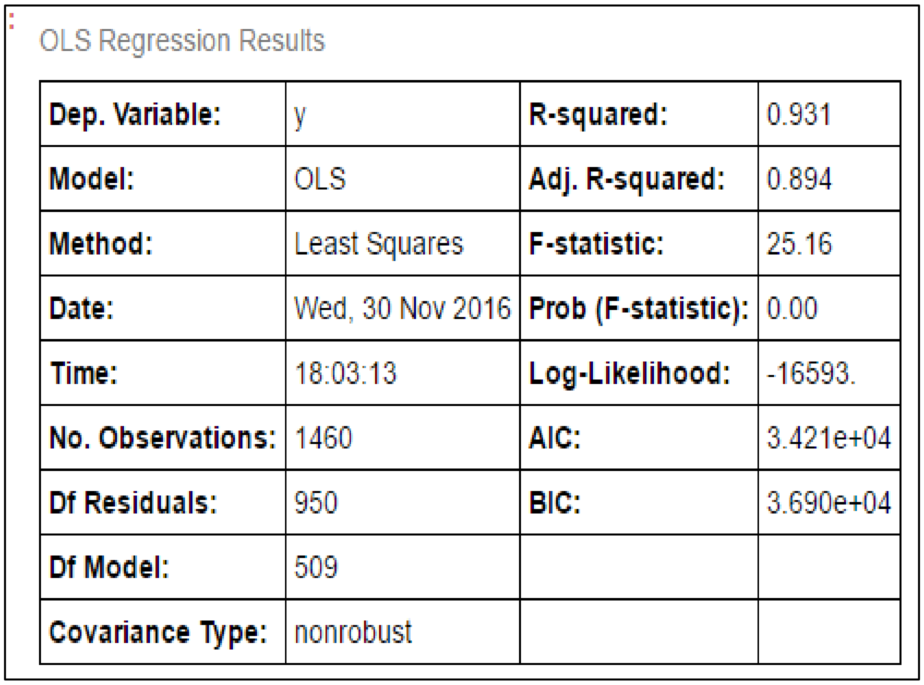
REGRESSION ANALYSIS: ORDINARY LEAST SQUARES (OLS)

In statistics, ordinary least squares (OLS) or linear least squares is a method for estimating the unknown parameters in a linear regression model, with the goal of minimizing the sum of the squares of the differences between the observed responses in the given dataset and those predicted by a linear function of a set of explanatory variables (visually this is seen as the sum of the vertical distances between each data point in the set and the corresponding point on the regression line – the smaller the differences, the better the model fits the data).

The resulting estimator can be expressed by a simple formula, especially in the case of a single regressor. The OLS estimator is consistent when the regressors are exogenous, and optimal in the class of linear unbiased estimators when the errors are homoscedastic and serially uncorrelated. Under these conditions, the method of OLS provides minimum-variance mean-unbiased estimation when the errors have finite variances.

Under the additional assumption that the errors be normally distributed, OLS is the maximum likelihood estimator.

Initially, we performed regression analysis with ordinary least square method. The summary of analysis is shown further.


OLS Regression Results

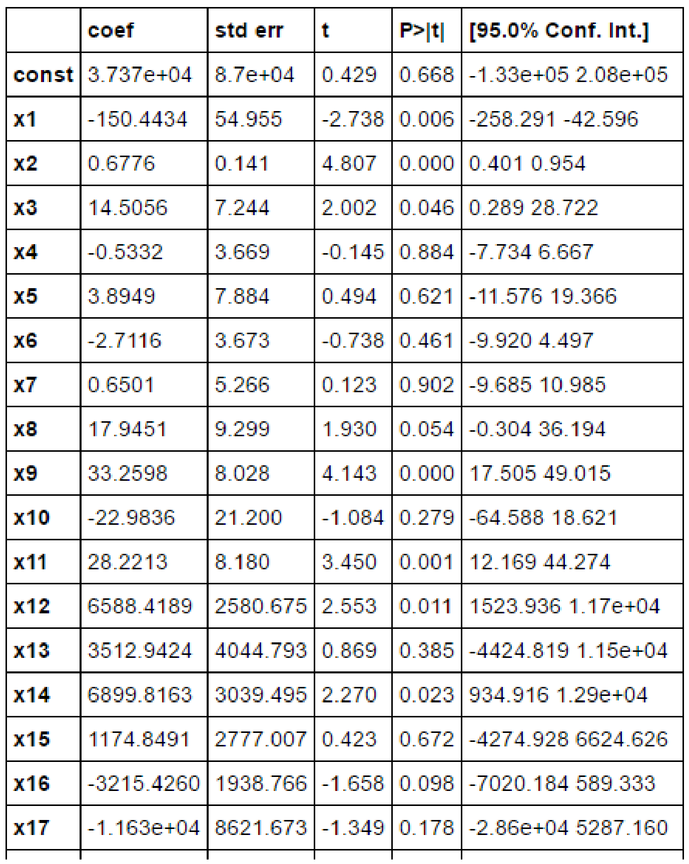
Dep. Variable:	y	**R-squared:**	0.931
Model:	OLS	**Adj. R-squared:**	0.894
Method:	Least Squares	**F-statistic:**	25.16
Date:	Wed, 30 Nov 2016	**Prob (F-statistic):**	0.00
Time:	18:03:13	**Log-Likelihood:**	-16593.
No. Observations:	1460	**AIC:**	3.421e+04
Df Residuals:	950	**BIC:**	3.690e+04
Df Model:	509		
Covariance Type:	nonrobust		

Table 1: Result Summary of Ordinary least square regression analysis.

Observing the result summary, we concluded that, the model has pretty good fit over data as R square value is 93%. R square shows the measure of variation in the response variable due to variation in input variables. Although, the r square value has an increasing trend over increase in input variables.

Hence, due to the large number of input variables, we were more concerned about the adjusted R square. Adjusted R square penalizes the model for the consideration of number of input variables. In spite of considering such a large number of input variables, it was observed that, Adjusted R square being 89% was representing better model.

After that, we studied our analysis results in more detail. We observed the P values for main effects and interaction terms.



	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	3.737e+04	8.7e+04	0.429	0.668	-1.33e+05 2.08e+05
x1	-150.4434	54.955	-2.738	0.006	-258.291 -42.596
x2	0.6776	0.141	4.807	0.000	0.401 0.954
x3	14.5056	7.244	2.002	0.046	0.289 28.722
x4	-0.5332	3.669	-0.145	0.884	-7.734 6.667
x5	3.8949	7.884	0.494	0.621	-11.576 19.366
x6	-2.7116	3.673	-0.738	0.461	-9.920 4.497
x7	0.6501	5.266	0.123	0.902	-9.685 10.985
x8	17.9451	9.299	1.930	0.054	-0.304 36.194
x9	33.2598	8.028	4.143	0.000	17.505 49.015
x10	-22.9836	21.200	-1.084	0.279	-64.588 18.621
x11	28.2213	8.180	3.450	0.001	12.169 44.274
x12	6588.4189	2580.675	2.553	0.011	1523.936 1.17e+04
x13	3512.9424	4044.793	0.869	0.385	-4424.819 1.15e+04
x14	6899.8163	3039.495	2.270	0.023	934.916 1.29e+04
x15	1174.8491	2777.007	0.423	0.672	-4274.928 6624.626
x16	-3215.4260	1938.766	-1.658	0.098	-7020.184 589.333
x17	-1.163e+04	8621.673	-1.349	0.178	-2.86e+04 5287.160

Table 2: Result Summary of Ordinary least square regression analysis: Significance of input variables.

We observed the P values and also figured out few significant attributes. We performed regression analysis considering only few of them just to get a generalized idea of significance.

While performing such few trials with subset of input variables, we found that the regression analysis considering attributes ‘GrLivArea’ and ‘Neighborhood’ only, the R square was 80%. So, to conclude, rest variables were adding overall accuracy of 13% to the prediction model in regression analysis using ordinary least square.

We plotted few graphs to visualize the results. Firstly, we plotted the graph of residual vs. fitted values to see the trend of residuals.

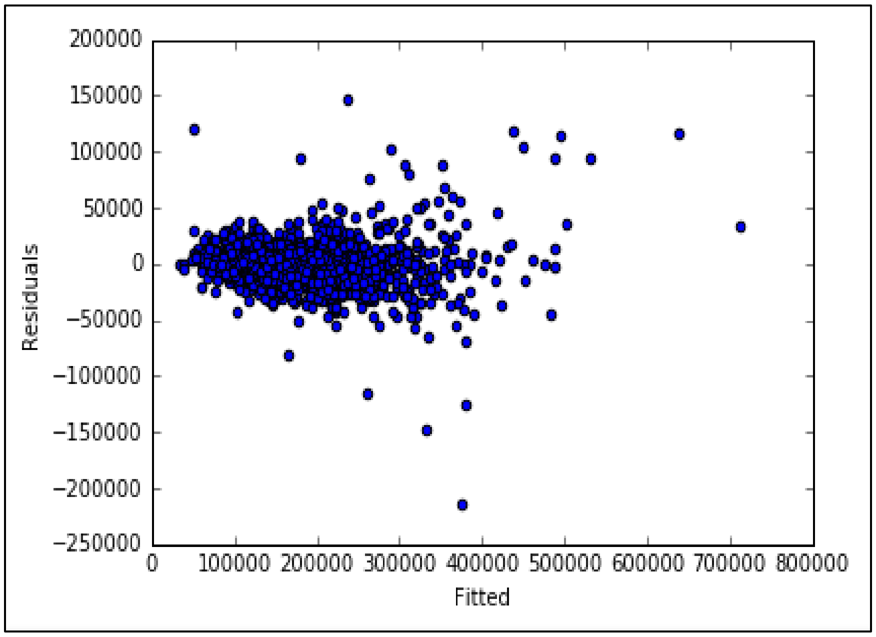


Figure 5: Graph of Fitted values against residuals.

Regression analysis: Advanced regression analysis

The major shortcoming of OLS regression analysis is that, it does not eliminate the effect of errors due to correlated attributes. With 79 attributes describing a house, there was a large possibility that, the prediction model had enhanced error due to correlation. Thus, there was a need to eliminate the error to improve accuracy.

Ridge and Lasso techniques were used as advanced regression analysis techniques which overcomes this disadvantage.

In Ridge regression, a constraint is put over betas (weights for attributes) such that, those betas cannot exceed more than a certain constant. The constraint is put over second degree of betas.

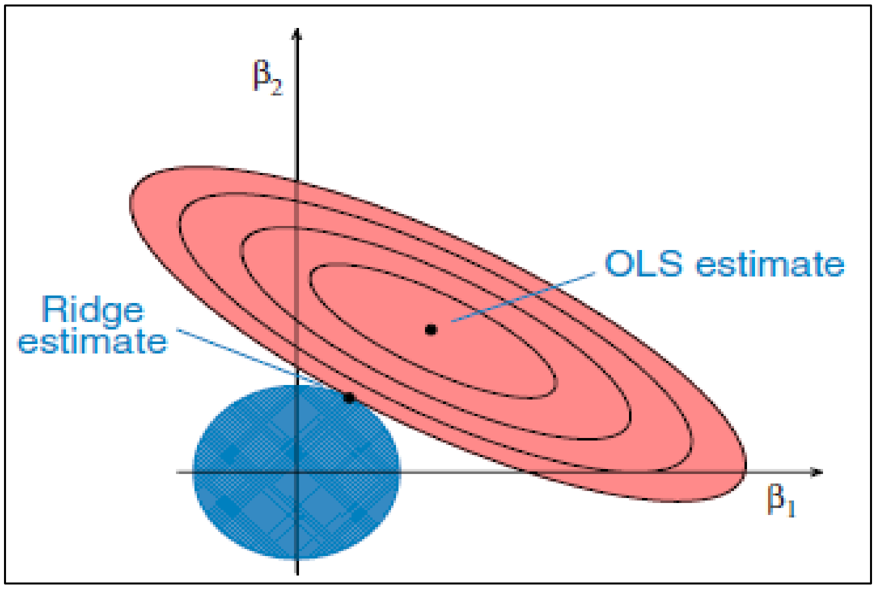
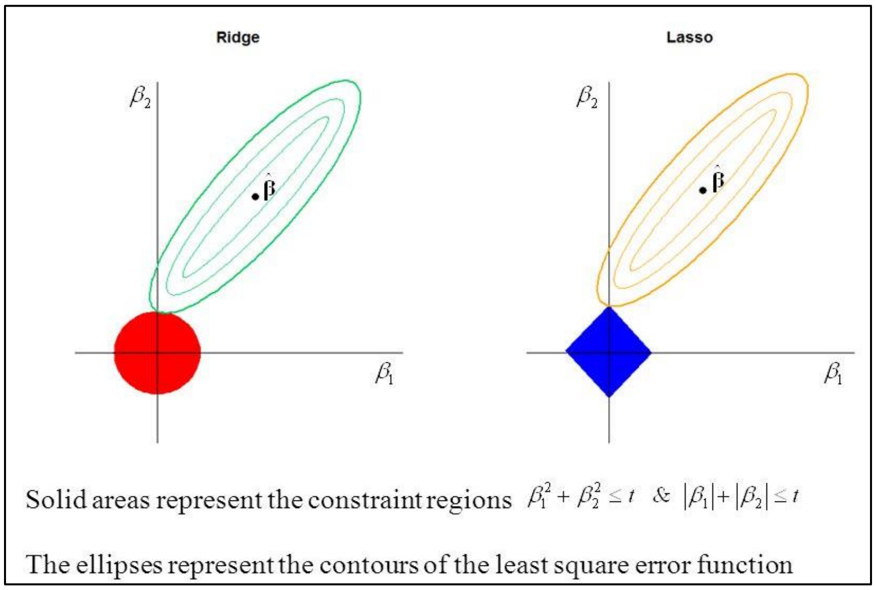


Figure 6: Advantage of Ridge regression over OLS.

In terms of root mean square error value, Ridge regression model had significantly lesser error than OLS regression. Moreover, we used Lasso regression. In this method, the constrain is on single degree of betas. Due to this, the function estimate intersects the volume of betas at some axis. At this point, the values for some of betas might be zero. Hence, these betas are dropped.

In simple words, these attributes are eliminated in the regression analysis. Hence, Lasso provides best subset selections while the model is penalized for this subset selection up to some extent in terms of increase in root mean square error.


Ridge
Lasso
β_2
$\hat{\beta}$
β_1
Solid areas represent the constraint regions $\beta_1^2 + \beta_2^2 \leq t$ & $|\beta_1| + |\beta_2| \leq t$
The ellipses represent the contours of the least square error function
Figure 7: Advantage of Lasso regression over Ridge regression.

Overall, the result summary for RMSE values for different analyses is shown further.

Result summary for RMSE			
Regression analysis	Baseline regression	Ridge regression	Lasso Regression
RMSE	21835074	20140	22784

Table 3: RMSE result summary.

Looking at the spread of house sale prices, we concluded that the error value obtained from the ridge analysis is around 10%. Considering the scope of model, the error is not that significant.

TWO SAMPLE Z TEST

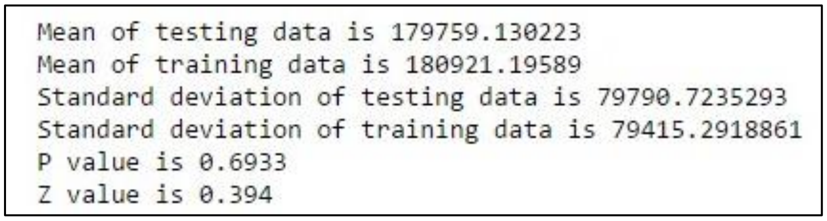
We have one set of house prices from training data which were observed. After building regression equation, we passed the test data through this model for prediction analysis. In results, we got another sample of predicted values.

To compare these two samples over population data, two tail z test was performed to determine whether there was some trend in the both samples and whether their means were different.

Samples,
Sample1: Sale price (output variable) in training data set
Sample2: Predicted sale price (output variable) in test data set

Formulation of hypotheses,
H_0 : Null hypothesis : $\mu_{sample1} = \mu_{sample2}$
H_a : Alternate hypothesis : $\mu_{sample1}$!= $\mu_{sample2}$

The Z test was performed in python. Result summary is shown further.



```
Mean of testing data is 179759.130223
Mean of training data is 180921.19589
Standard deviation of testing data is 79790.7235293
Standard deviation of training data is 79415.2918861
P value is 0.6933
Z value is 0.394
```

Table 4: Two sample Z test result summary.

P value obtained is 0.69 which is more than 0.05. Hence, we concluded that, we fail to reject the null hypothesis. In simple words, there’s no difference in the mean of two samples.

RESULTS: VISUALIZATION

It was observed that there's no specific trend in the residuals. Which confirms the absence of constant variance in error.

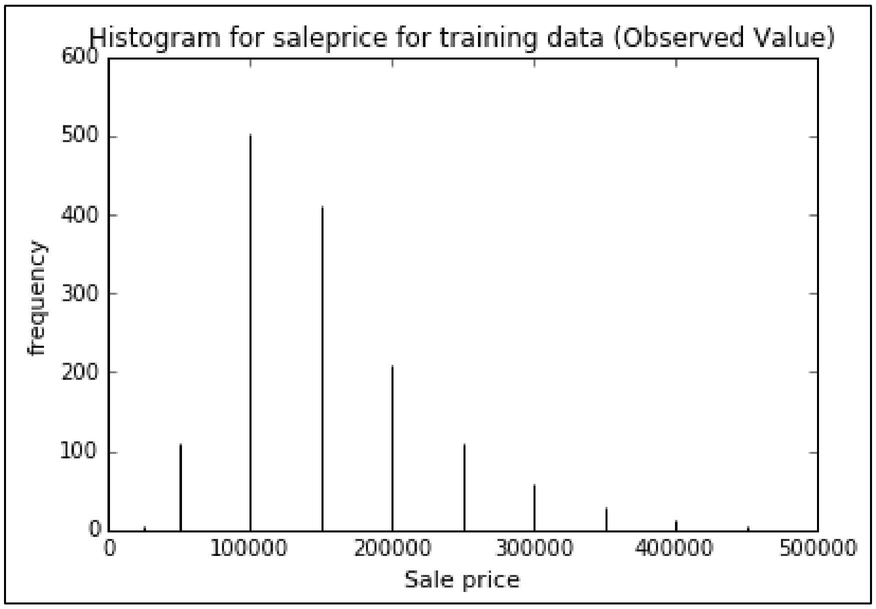


Figure 8: Histogram for sale price given in training data.

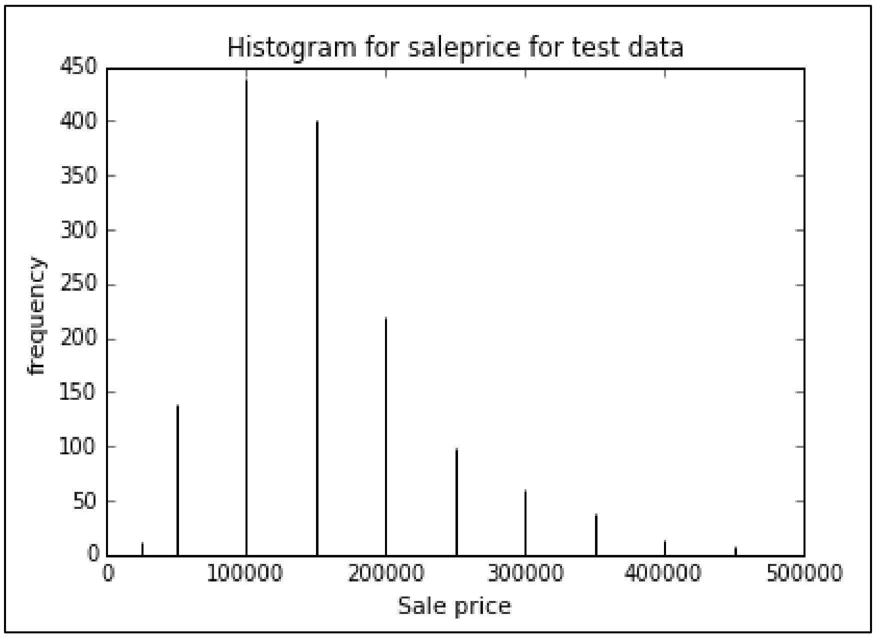


Figure 9: Histogram for sale price obtained from prediction analysis.

Also, histograms for two samples used in Z test shows similar trend in sale price of the houses.

CONCLUSION

We conluded our experiment with the prediction model having 93% control over the response variable due to chainge in input variables.

Also, advanced regression techniques like Ridge and Lasso resulted into singificant reduction in root mean square error. The error was around 10% after iterating the regression analysis upto maximum limit i.e. total number of datapoint.

With lasso regression, subset of input variables was used instead of entire range of input variables. That resulted into slight increase in RMSE value. But the computataion time was reduced.

FUTURE SCOPE

In spite of prediction error of 93%, there is a lot of scope for improvising the model. Reduction in RMSE value can be done in future with better subset selection. Also, from Lasso regression, the improvised subset of input variables can be identified which has relatively more effect on the response variable than other input variables. After deterring such variables, non-linear analysis can be performed by tweaking these selective variables to get better fit.

On the website, from where the dataset was taken, the use of regression trees was recommended. Due to large number of input variables, the complexity of regression tree could have been increased significantly. However, depending on the prediction error, the effectiveness of regression trees can be determined.

REFERENCE

- http://nbviewer.jupyter.org/github/JWarmenhoven/ISL-python/blob/master/Notebooks/Chapter%203.ipynb
- https://www.youtube.com/watch?v=ZyTO4SwhSeE&list=PLQVvvaa0QuDfefDfXb9Yf0la1fPDKluPF&index=3
- https://ww2.amstat.org/publications/jse/v16n2/datasets.pardoe.html
- https://ww2.amstat.org/publications/jse/v16n3/datasets.woodard.html
- http://support.sas.com/resources/papers/proceedings12/333-2012.pdf

CODE

```
# In[1]:
#Importing necessary modules
import pandas as pd
import numpy as np
import scipy as sp
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
get_ipython().magic(u'matplotlib inline')
import matplotlib.mlab as mlab
import os


# In[2]:
#changing directory
os.chdir('E:\Fall_2016\Data Science 1\Project')


# In[3]:
#Reading scv file into python and preparing dataframe
a = pd.read_csv('train.csv')
ax = a.plot(x= 'Id', y="SalePrice", style="o")
ax.set_ylabel("SalePrice")


# In[4]:
#For initial data claening, data visualization is used. Few plots were plotted to see the scatter.
ax = a.plot(x= 'GrLivArea', y="SalePrice", style="o")
ax.set_ylabel("SalePrice")


# In[5]:
#Box plot was plotted to visualize the range of sale price.
fig = plt.figure()
plt.boxplot(a.SalePrice)


# In[16]:
```

```
#Data normality checks. Plotting histogram.
n, bins, patches = plt.hist(a.SalePrice, 50, normed=1)
mu = np.mean(a.SalePrice)
sigma = np.std(a.SalePrice)
plt.plot(bins, mlab.normpdf(bins, mu, sigma))
plt.label("Distribution of saleprice")
plt.xlabel("SalePrice")


# In[17]:
# Columns were dropped for which data conversion was done.
a = a.drop('YearBuilt', 1)
a = a.drop('YearRemodAdd', 1)
a = a.drop('GarageYrBlt', 1)
a = a.drop('YrSold', 1)
a = a.drop('Id', 1)
#a.head()


# In[18]:
#list of columns was prepared which incuded all the attributes having missing values
#along with the count of missing values and later list was sorted.
cols_having_missing = a.isnull().sum(axis = 0)
cols_having_missing.sort(axis=0, ascending=False, kind='quicksort', na_position='last',
inplace=True)
cols_having_missing


# In[19]:
# missing values for categorical attributes were replaced by zero.
# missing value for quantitative variables were replacecd by mean of that column.
#a['poolQC'].fillna((a['poolQC'].mean()), inplace=True)
a['PoolQC'].fillna(value = 0, inplace=True)
a['MiscFeature'].fillna(value = 0, inplace=True)
#a[some_list].fillna(value = 0, inplace=True)
a['Alley'].fillna(value = 0, inplace=True)
a['Fence'].fillna(value = 0, inplace=True)
a['FireplaceQu'].fillna(value = 0, inplace=True)
a['LotFrontage'].fillna((a['LotFrontage'].mean()), inplace=True)
a['GarageType'].fillna(value = 0, inplace=True)
a['GarageCond'].fillna(value = 0, inplace=True)
```

```
a['GarageQual'].fillna(value = 0, inplace=True)
#a['GarageYrBlt'].fillna((a['GarageYrBlt'].mean()), inplace=True)
a['GarageFinish'].fillna(value = 0, inplace=True)
a['BsmtFinType2'].fillna(value = 0, inplace=True)
a['BsmtExposure'].fillna(value = 0, inplace=True)
a['BsmtFinType1'].fillna(value = 0, inplace=True)
a['BsmtQual'].fillna(value = 0, inplace=True)
a['BsmtCond'].fillna(value = 0, inplace=True)
a['MasVnrType'].fillna(value = 0, inplace=True)
a['MasVnrArea'].fillna(value = 0, inplace=True)
a['Electrical'].fillna(value = 0, inplace=True)
a


# In[20]:
#A list was prepared in which all the categorical variables were included and later this list was passed to prepare dummy variables.
#A dataframe for training data set was finalized was was to be used for furtherr regression analysis.
i1 = ['MSSubClass', 'MSZoning', 'Street', 'Alley', 'LandContour', 'LotConfig', 'Neighborhood', 'Condition1',
     'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
     'Foundation', 'Heating', 'CentralAir', 'GarageType', 'MiscFeature', 'SaleType', 'SaleCondition']

i2 = ['LotShape', 'Utilities', 'LandSlope', 'OverallQual', 'OverallCond', 'ExterQual', 'ExterCond', 'BsmtQual', 'BsmtCond',
    'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'HeatingQC', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu',
    'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence']

i3 = ['YearBuilt_1','YearRemodAdd_1','GarageYrBlt_1','YrSold_1']

i = i1 + i2 + i3
train = pd.get_dummies(a, columns = i, drop_first = True) #train matrix full
train.head()


# In[21]:
# Columns were dropped for which data conversion was done.
t = pd.read_csv('test.csv')
```

```
t = t.drop('Id', 1)
t = t.drop('YearBuilt', 1)
t = t.drop('YearRemodAdd', 1)
t = t.drop('GarageYrBlt', 1)
t = t.drop('YrSold', 1)
t.head()


# In[22]:
#list of columns was prepared which incuded all the attributes having missing values
#along with the count of missing values and later list was sorted.
cols_having_missing_test = t.isnull().sum(axis = 0)
cols_having_missing_test.sort(axis=0,    ascending=False,    kind='quicksort',    na_position='last',
inplace=True)
print cols_having_missing_test[:40]


# In[23]:
# missing values for categorical attributes were replaced by zero.
# missing value for quantitative variables were replacecd by mean of that column.
t['PoolQC'].fillna(value = 0, inplace=True)
t['MiscFeature'].fillna(value = 0, inplace=True)
t['Alley'].fillna(value = 0, inplace=True)
t['Fence'].fillna(value = 0, inplace=True)
t['FireplaceQu'].fillna(value = 0, inplace=True)
t['LotFrontage'].fillna((a['LotFrontage'].mean()), inplace=True)
t['GarageType'].fillna(value = 0, inplace=True)
t['GarageCond'].fillna(value = 0, inplace=True)
t['GarageQual'].fillna(value = 0, inplace=True)
#t['GarageYrBlt'].fillna(value = 0, inplace=True)
t['GarageFinish'].fillna(value = 0, inplace=True)
t['BsmtFinType2'].fillna(value = 0, inplace=True)
t['BsmtExposure'].fillna(value = 0, inplace=True)
t['BsmtFinType1'].fillna(value = 0, inplace=True)
t['BsmtFinType2'].fillna(value = 0, inplace=True)
t['BsmtQual'].fillna(value = 0, inplace=True)
t['BsmtCond'].fillna(value = 0, inplace=True)
t['MasVnrType'].fillna(value = 0, inplace=True)
t['MasVnrArea'].fillna(value = 0, inplace=True)
t['Electrical'].fillna(value = 0, inplace=True)
t['MSZoning'].fillna(value = 0, inplace=True)
```

```
t['BsmtHalfBath'].fillna(value = 0, inplace=True)
t['Utilities'].fillna(value = 0, inplace=True)
t['Functional'].fillna(value = 0, inplace=True)
t['BsmtFullBath'].fillna(value = 0, inplace=True)
t['BsmtFinSF2'].fillna(value = 0, inplace=True)
t['BsmtFinSF1'].fillna(value = 0, inplace=True)
t['BsmtUnfSF'].fillna(value = 0, inplace=True)
t['TotalBsmtSF'].fillna(value = 0, inplace=True)
t['Exterior2nd'].fillna(value = 0, inplace=True)
t['SaleType'].fillna(value = 0, inplace=True)
t['Exterior1st'].fillna(value = 0, inplace=True)
t['KitchenQual'].fillna(value = 0, inplace=True)
t['GarageArea'].fillna(value = 0, inplace=True)
t['GarageCars'].fillna(value = 0, inplace=True)
t


# In[24]:
#Earlier list of categorical variables was used to prepare dummy variables.
#A dataframe for training data set was finalized was was to be used for furtherr regression analysis.
test = pd.get_dummies(t, columns = i, drop_first = True)
test.head()


# In[25]:
#The columns in training dataset and test dataset should be aligned before passing values.
#The regression analysis should be performed on mutually inclusive attributes.
#Hence datarames were to me maded identical before proceeding further.
listtrain_1 = [col for col in train.columns if col!= 'SalePrice']
#listtrain_1
listtest_1 = [col for col in test.columns if col!= 'SalePrice']
#listtest_1

diff = list(set(listtest_1) - set(listtrain_1))
diff1 = list(set(listtrain_1) - set(listtest_1))
difference = diff + diff1

#Mutually exclusive ariables were dropped from  both dataframes.
train.drop(diff1,inplace=True,axis=1)
#b
```

```
test.drop(diff,inplace=True,axis=1)
#t1


# In[30]:
#Basic regression analysis was performed.
#output was obained in the array of intercept and coeficients.
collist = [col for col in train.columns if col != 'SalePrice']
X = train[collist].values
y = train[['SalePrice']]
reg2 = LinearRegression()
reg2.fit(X, y)
(reg2.intercept_, reg2.coef_)


# In[32]:
#Trend for residuals against fitted values was observed to understand the effect of corelated input variables.
def residuals_vs_fitted(fitted, residuals, xlabel, ylabel):
    plt.subplot(111)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.scatter(fitted, residuals)
    polyline = np.poly1d(np.polyfit(fitted, residuals, 2))    # model non-linearity with quadratic
    xs = range(int(np.min(fitted)), int(np.max(fitted)))
    plt.plot(xs, polyline(xs), color='r', linewidth=2.5)

def qq_plot(residuals):
    sm.qqplot(residuals)

def standardize(xs):
    xmean = np.mean(xs)
    xstd = np.std(xs)
    return (xs - xmean) / xstd

fitted = reg2.predict(X)
residuals = y - fitted
std_residuals = standardize(residuals)

residuals_vs_fitted(fitted, residuals, "Fitted", "Residuals")
```

```
# In[33]:
#Ordinary least sqaure regression analysis was performed.
#collist = [col for col in train.columns if col!= 'SalePrice']
X = train[collist].values
X = sm.add_constant(X)  # add the intercept term
y = train["SalePrice"].values
ols = sm.OLS(y, X).fit()
ols.summary()


# In[53]:
#Test data was passed into regression model to prredict the sale pricce of houses for test data.
#output was obtained in the format of array which included intercept (beto 0) and coeficients.
SalePrice_test = reg2.predict(test)
SalePrice_test


# In[ ]:
#Samples were formed for performing two tailed two sample Z test to understand the difference
in  means.
train_new = train
#train_new.shape

train_new.drop('1460', 0)
#train_new.shape


# In[68]:
# histogram for saleprice for test data was plotted.
bins = [0, 25000, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000,
500000]
plt.title("Histogram for saleprice for test data")
plt.xlabel('Sale price', fontsize=11)
plt.ylabel('frequency', fontsize=11)
plt.hist(SalePrice_test, bins, histtype = 'bar', width = 0.8)


# In[69]:
#Histogram for saleprice for training data (Observed Value) was plotted.
```

```
bins = [0, 25000, 50000, 100000, 150000, 200000, 250000, 300000, 350000, 400000, 450000,
500000]
plt.title("Histogram for saleprice for training data (Observed Value)")
plt.xlabel('Sale price', fontsize=11)
plt.ylabel('frequency', fontsize=11)
plt.hist(train.SalePrice, bins, histtype = 'bar', width = 0.8)


# In[ ]:
#modules for statistical analyses were imported.
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.metrics import mean_squared_error
from sklearn.cross_validation import KFold


# In[ ]:
#Advanced regression techniques were performed and RMSE vavlue was obtained.
#RMSE values for baseline regression, Ridge regression and Lasso regression was obtained.
def cross_validate(X, y, nfolds, reg_name):
    rmses = []
    kfold = KFold(X.shape[0], n_folds=nfolds)
    for train, test in kfold:
        Xtrain, ytrain, Xtest, ytest = X[train], y[train], X[test], y[test]
        reg = None
        if reg_name == "ridge":
            reg = Ridge()
        elif reg_name == "lasso":
            reg = Lasso()
        else:
            reg = LinearRegression()
        reg.fit(Xtrain, ytrain)
        ypred = reg.predict(Xtest)
        rmses.append(np.sqrt(mean_squared_error(ytest, ypred)))
    return np.mean(rmses)

collist = [col for col in train.columns if col != "SalePrice"]
X = train[collist].values
y = train["SalePrice"].values
```

```
rmse_baseline = cross_validate(X, y, 1458, "baseline")
rmse_ridge = cross_validate(X, y, 1458, "ridge")
rmse_lasso = cross_validate(X, y, 1458, "lasso")
(rmse_baseline, rmse_ridge, rmse_lasso)

# In[44]:
# CSV file including results was read for performing two sample Z test.
ztest = pd.read_csv('Z_test.csv')
#ztest.head(1)

# In[46]:
# two sample Z test was peprformedd.
train_mean = ztest.SalePrice.mean()
test_mean = ztest.Prediction.mean()
train_std = np.std(ztest.SalePrice)
test_std = np.std(ztest.Prediction)
train_mean = ztest.SalePrice.mean()
test_mean = ztest.Prediction.mean()
train_std = np.std(ztest.SalePrice)
test_std = np.std(ztest.Prediction)
train_sample_size = 1460
test_sample_size = 1460
pop_mean_diff = 0

def twoSampZ(X1, X2, mudiff, sd1, sd2, n1, n2):
   from numpy import sqrt, abs, round
   from scipy.stats import norm
   pooledSE = sqrt(sd1**2/n1 + sd2**2/n2)
   z = ((X1 - X2) - mudiff)/pooledSE
   pval = 2*(1 - norm.cdf(abs(z)))
   return round(z, 3), round(pval, 4)

z, p = twoSampZ(train_mean, test_mean, pop_mean_diff, train_std, test_std, train_sample_size,
test_sample_size)

print "Mean of testing data is",test_mean
print "Mean of training data is",train_mean
print "Standard deviation of testing data is",test_std
print "Standard deviation of training data is",train_std
print "P value is",p
print "Z value is",z
```