

Predicting House Price on Ames Dataset

BITS ZG628T: Dissertation

by

Archit Vora

2015HT12480

Dissertation work carried out at

Adobe Systems, Bangalore

Submitted in partial fulfillment of M.Tech. Software Systems degree
programme

Under the Supervision of
Anirban Basu, Data Science Manager,
Adobe Systems, Bangalore

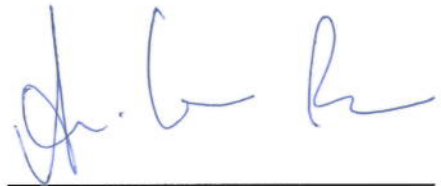


**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April, 2017

CERTIFICATE

This is to certify that the Dissertation entitled **Predicting House Price On Ames Dataset** and submitted by **Archit Vora** having ID-No. **2015HT12480** for the partial fulfillment of the requirements of M.Tech. **Software Systems** degree of BITS, embodies the bonafide work done by him under my supervision.



Signature of the Supervisor

Place : Bangalore
Date : 24th, 2017
Mark

ANIRBAN BASU (DATA SCIENCE
MANAGE, ADOBE, BANG
ALOR)

Name, Designation & Organization & Location

Birla Institute of Technology & Science, Pilani
Work-Integrated Learning Programmes Division

Second Semester 2016-2017

BITS ZG628T: Dissertation

ABSTRACT

BITS ID No. : 2015HT12480

NAME OF THE STUDENT : Archit Vora

EMAIL ADDRESS : architvora92@gmail.com

STUDENT'S EMPLOYING ORGANIZATION & LOCATION : Adobe Systems, Bangalore

SUPERVISOR'S NAME : Anirban Basu

SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION : Adobe Systems, Bangalore

SUPERVISOR'S EMAIL ADDRESS: abasu@adobe.com

DISSERTATION TITLE : Predicting House Price on Ames Dataset

ABSTRACT :

Regression Analysis has lots of application as we are marching into data driven society. It has a great role to play in time series analysis as well. In this work we apply regression analysis for predicting house price on Ames dataset. We focus in depth with mathematics for linear regression and Bayesian regression with Monte Carlo simulation. We see how this technique performs on our dataset, we provide explanation for both when things work and when it does not.

Broad Academic Area of Work: Data Mining


Key words Linear Regression, Bayesian Regression, Regularization


Signature of the Student

Name: Archit Vora

Date: 24/03/2017

Place: Bangalore


Signature of the Supervisor

Name: Anirban Basu

Date: 24/03/2017

Place: Bangalore

Acknowledgements

For me best part of my life has been my teachers and my seniors. I feel very glad that they still continue to be with me even after 3 years of graduation. Special thanks to BITS Pilani, 2 years of WILP had provided me a focused direction for data analytics learning. My mentor here at Adobe, Anirban Basu. I always here carefully what he has to say, capture the terms and Google them. Thank you Anirban for making this work more of a fundamental learnings. Adobe for wonderful work life balance environment, because of which I was able to complete this WILP course. Teachers and professors from my school and university, they have a great positive impact on me. My parents and sister !

-Archit

Contents

Abstract	i
Acknowledgements	iii
1 Introduction and Pre-Processing	1
1.1 Literature Survey	1
1.2 Data Description	2
1.3 Feature Engineering	5
1.3.1 Leveling	5
1.3.2 Filling up missing values	5
1.3.3 Box Cox Transformations	10
1.3.4 Interaction Features	12
1.4 Error Metric	12
1.5 Detailed organization of the report	13
2 Linear Regression And Regularization	14
2.1 Simple Linear Regression	14

2.1.1	Solving matrix equation	15
2.1.2	Gradient Descent	16
2.1.3	Stochastic Gradient Descent	18
2.2	Ridge Regression	18
2.2.1	Linear Algebra Solution	19
2.3	Lasso Regression	20
2.4	Shrinkage Property Of Ridge and Lasso	20
2.5	Regression with Dimensionality Reduction	23
2.5.1	Principle Component Regression	23
2.5.2	Relation of PCR with Ridge	24
2.6	Summary	25
3	Bayesian Regression	27
3.1	Introduction	27
3.2	Markov chain Monte Carlo	28
3.2.1	MCMC Steps	28
3.3	Normal and log-logistic Distribution for Target	29
3.4	Regularization in Bayesian	30
3.4.1	Lasso Regularization	30
3.5	Ridge Regularization	31
3.6	Summary	32

4	Summary	33
4.1	Summary	33
4.2	Directions For Future Work	34
A	Detailed Data Description	36
B	Missing Value Notebook	56
C	Codes	75
C.1	Gradient Descent	76
C.2	Matrix Approach for solving Ridge Regression	77
C.3	Principle Component Regression	79
C.4	Custom Distribution in pymc - log logistic	81
	C.4.1 Writing Custom Distribution	81
	C.4.2 Predictive Model	81
	Bibliography	83

List of Tables

2.1	Summary for Linear Regression And Regularization	26
3.1	Summary for Bayes	32
4.1	Accuracy for various models	35

List of Figures

1.1	Missing values for different features	5
1.2	Q-Q plot before transforming y	11
1.3	Q-Q plot after raising y with boxcox λ	11
1.4	Q-Q plot after log transformation	12
2.1	Cost function before normalization	17
2.2	Cost function after normalization	17
2.3	Tuning Lambda for Ridge Regression	19
2.4	Tuning λ parameter for matrix approach for Ridge Regression	20
2.5	Tuning Lambda for Lasso Regression	21
2.6	Shrinkage property of Ridge and Lasso	21
2.7	Coefficients for simple linear regression (Range of 0.5)	22
2.8	Coefficient for Ridge Regression lambda=15 (Range of 0.005)	22
2.9	Coefficient with Lasso Regression lambda=1 (Many of them are zero)	22
2.10	Constraint functions for different values of q	23
2.11	Variance explained by Principle Components	24

3.1	Simulation Traces	29
3.2	Laplace Distribution	31

Chapter 1

Introduction and Pre-Processing

Housing price prediction is a classic problem for regression and predictive modeling. It generally contains feature like nearby area, no of bedrooms, parking facilities etc and task is to predict the price of the house. In this work we will be exploring various regression techniques for the prediction task. Our focus will be more on mathematic for linear regression and Bayesian regression. We will also try various implementation for these models and evaluate their performance.

1.1 Literature Survey

Hujia Yu and Jaifu Wu at [8] has worked on the same problem. They have tried range of model on the same dataset for the prediction of house price including Lasso, Ridge, SVM regression and Random Forest regression. Also they have classified the price range into 7 buckets and applied classification on them. Then they discussed result of both before and after applying PCA. It seems is that there goal is to say which model gives how much accuracy. Apart from that Ames being open dataset and kaggle competition it would have been studied by a lot of people in academia.

1.2 Data Description

Ames housing dataset was designed to improve upon classical Boston dataset. It contains data for the city of Ames in the state of Iowa, United States. The dataset has 80 features (23 nominals, 23 ordinals, 14 discrete and 20 continuous). Detailed description about the dataset will be given in next section. The dataset contains missing values and features like nearby area requires feature extraction. Here is the brief description of the dataset, detailed one can be found at Appendix A.

SalePrice	the property's sale price in dollars. This is the target variable that you're trying to predict
MSSubClass	The building class
MSZoning	The general zoning classification
LotFrontage	Linear feet of street connected to property
LotArea	Lot size in square feet
Street	Type of road access
Alley	Type of alley access
LotShape	General shape of property
LandContour	Flatness of the property
Utilities	Type of utilities available
LotConfig	Lot configuration
LandSlope	Slope of property
Neighborhood	Physical locations within Ames city limits
Condition1	Proximity to main road or railroad
Condition2	Proximity to main road or railroad (if a second is present)
BldgType	Type of dwelling
HouseStyle	Style of dwelling
OverallQual	Overall material and finish quality
OverallCond	Overall condition rating
YearBuilt	Original construction date
YearRemodAdd	Remodel date

RoofStyle	Type of roof
RoofMatl	Roof material
Exterior1st	Exterior covering on house
Exterior2nd	Exterior covering on house (if more than one material)
MasVnrType	Masonry veneer type
MasVnrArea	Masonry veneer area in square feet
ExterQual	Exterior material quality
ExterCond	Present condition of the material on the exterior
Foundation	Type of foundation
BsmtQual	Height of the basement
BsmtCond	General condition of the basement
BsmtExposure	Walkout or garden level basement walls
BsmtFinType1	Quality of basement finished area
BsmtFinSF1	Type 1 finished square feet
BsmtFinType2	Quality of second finished area (if present)
BsmtFinSF2	Type 2 finished square feet
BsmtUnfSF	Unfinished square feet of basement area
TotalBsmtSF	Total square feet of basement area
Heating	Type of heating
HeatingQC	Heating quality and condition
CentralAir	Central air conditioning
Electrical	Electrical system
1stFlrSF	First Floor square feet
2ndFlrSF	Second floor square feet
LowQualFinSF	Low quality finished square feet (all floors)
GrLivArea	Above grade (ground) living area square feet
BsmtFullBath	Basement full bathrooms
BsmtHalfBath	Basement half bathrooms
FullBath	Full bathrooms above grade

HalfBath	Half baths above grade
Bedroom	Number of bedrooms above basement level
Kitchen	Number of kitchens
KitchenQual	Kitchen quality
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms)
Functional	Home functionality rating
Fireplaces	Number of fireplaces
FireplaceQu	Fireplace quality
GarageType	Garage location
GarageYrBlt	Year garage was built
GarageFinish	Interior finish of the garage
GarageCars	Size of garage in car capacity
GarageArea	Size of garage in square feet
GarageQual	Garage quality
GarageCond	Garage condition
PavedDrive	Paved driveway
WoodDeckSF	Wood deck area in square feet
OpenPorchSF	Open porch area in square feet
EnclosedPorch	Enclosed porch area in square feet
3SsnPorch	Three season porch area in square feet
ScreenPorch	Screen porch area in square feet
PoolArea	Pool area in square feet
PoolQC	Pool quality
Fence	Fence quality
MiscFeature	Miscellaneous feature not covered in other categories
MiscVal	\$Value of miscellaneous feature
MoSold	Month Sold
YrSold	Year Sold
SaleType	Type of sale

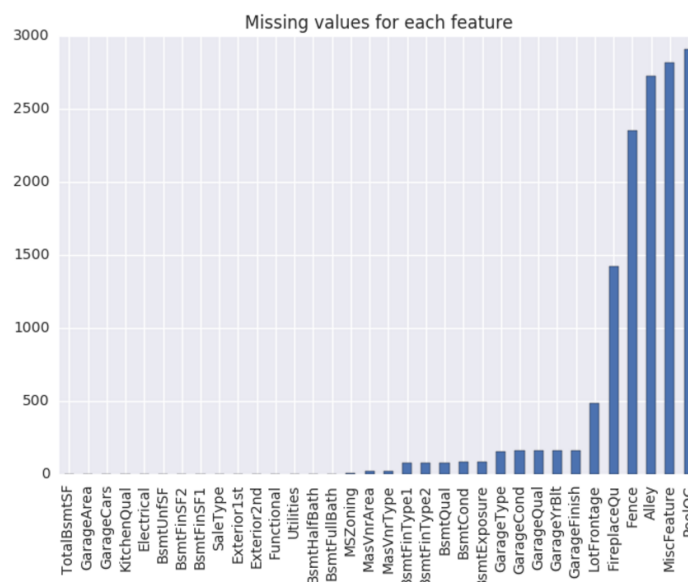


Figure 1.1: Missing values for different features

SaleCondition		Condition of sale
---------------	--	-------------------

1.3 Feature Engineering

Machine learning model have some general assumption about the underlying data. Although every model have their own requirement some feature generation are helpful for all the model. That is what we will be doing here.

1.3.1 Leveling

Categorical data needs leveling before proceeding further. So we create new column for each value of categorical variable. This has resulted into feature increase from 80 to 288.

1.3.2 Filling up missing values

The data(test + train) contained 13965 missing values. Figure 1.1 shows how it is distributed among feature. We use different strategies for filling up this value, for different feature. Some

general strategies are : For the ordinal features we replace them with integer levels and for nominal feature we create dummy variables. If there are few missing values, for discrete variables we replace them with median and for continuous variables we replace them with mean. Sometime we can infer the value for missing variable by looking at the values and distribution of related variables.

Here is the result in terms of accuracy. If we simply replace all missing value with mean and leveling for all variables, rmse error comes out to be 0.013226 (288 features) which after filling up missing values systematically was reduced to 0.013116 (263 features). We can see the no of features gets reduced to 263 from 288 upon filling up missing values. This is because some of the variables will get values assigned and leveling will not be needed.

Features can be classified as continuous and discrete. For continuous features we can replace it with mean or median based on its functional use. Discrete features can be of two types, nominal or ordinal. Ordinal variable represents some scale, like kitchen quality in our case. It has values like bad, good, excellent. We can replace them with some scale say, 0 for bad, 1 for good and 2 for excellent. We refer this process as 'Encoding' of ordinal variables. On the contrary nominal variable represents some category. One example in our case is zoning whether area is Agriculture, Commercial, Industrial, Residential etc. For such cases we generally do leveling and create dummy variable with binary values. Next we describe the strategies used for each features (Detailed code is available in Appendix B):

- PoolQC
 - PoolQC has 2000 something missing value
 - For all this 2000 cases pool area is zero
 - So we replace missing values with NA
 - This is ordinal variable
 - We will do encoding as well

- Alley

- Alley type has significant impact on SalePrice
- We are assuming null implies Alley is missing
- We will assign it NA(Data description says NA is valid value for Alley)
- Alley being nominal we will create dummy variable
- Fence
 - Fence represent Fence quality
 - It is ordinal variable
 - Data Description has NA = No Fence
 - We will apply it and do encoding.
- FireplaceQu
 - NA - no fireplace
 - Ordinal variable
 - Null = NA and encoding
- Utilities
 - Utilities has just two missing values
 - We replace with the most frequent one
 - There are two values in data and 'allPub' easily outperforms 'noswefg'
 - We will need dummy variables
- Kitchen Quality
 - Kitchen Quality has just missing value
 - Related variable is KitchenAbGr, which has no missing value
 - There seems no relation between two.
 - When Kitchen Abv Grade is zero, quality is still 3

- We will replace with the most frequent value 'TA'
- Ordinal =_i encoding
- SaleType
 - There is just one missing value
 - We will replace with most frequent
 - nominal = dummy variables
- MasVnrArea
 - According to wikipedia masonry veneer refers to the structure backed by either stone, bricks, which are durable and long lasting. We expect higher this are more should be the price
 - There are just 23 null values
 - When we plot distribution we see that most of values are zero
 - It is continuous variable and we will replace null with zero.
- MasVnrType
 - It has 24 null value
 - 23 are the cases when MasVnrArea is zero
 - We will replace it with None (Valid value as per data dictionary)
 - Also None is the most frequent value
 - So we will replace all 24 with null
 - Nominal = Dummy variable
- Various Basement Features
 - 'TotalBsmtSF', 'BsmtUnfSF', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtHalfBath', 'Bsmt-FullBath'
 - * null implies area is zero or count is zero

- * In all these cases corresponding quality is also null
- 'BsmtFinType1', 'BsmtFinType2', 'BsmtQual', 'BsmtCond', 'BsmtExposure'
 - * All has NA
 - * Apply NA when total basement area is Null (We assign null to smallest subset, BsmtFinType1 has smallest no of nulls)
- BsmtFinType2
 - * Replace With Most frequent
 - * Ordinal
 - * Encodin
- BsmtQual
 - * Replace With Most frequent
 - * Ordinal
 - * Encoding
- BsmtCond
 - * Replace With Most frequent
 - * Ordinal
 - * Encoding
- BsmtExposure
 - * Replace With Most frequent
 - * Ordinal
 - * Encoding
- MSZoning
 - Nominal variable
 - Replace with Mosty Frequent
 - Dummy Variables

- Functional
 - Description also says assume typical unless deductions are specified. We will consider this as ordinal variables and assign Type (Most Frequent) to null values.
 - Ordinal variable = Encoding
- MiscFeature
 - We will level encode without assigning anything to null values
- LotFrontage
 - We will do the regression with square root of LotArea (We found enough correlation, certainly better than filling with mean)
- Garage
 - GarageCond, GarageQual, GarageYrBlt, GarageFinish, GarageType, GarageCars, GarageArea are Garage related variables
 - We saw that for almost every null values GarageYrBlt is also null. And data dictionary had the value 'NA' for case when garage is not present.
 - So we assigned 'NA' to other variables and zero to GarageYrBlt, GarageCars and GarageArea

1.3.3 Box Cox Transformations

Many models assume data to be normally distributed. If it is not we can transform to make it normal. Some common transformations include log, square root, square etc. But how to find suitable transformation. To make the process automate the process Box-Cox transformations are used [6]. What this method does is it finds suitable parameter λ and transformation is defined for each λ as follows :

if $\lambda = 0$ then :

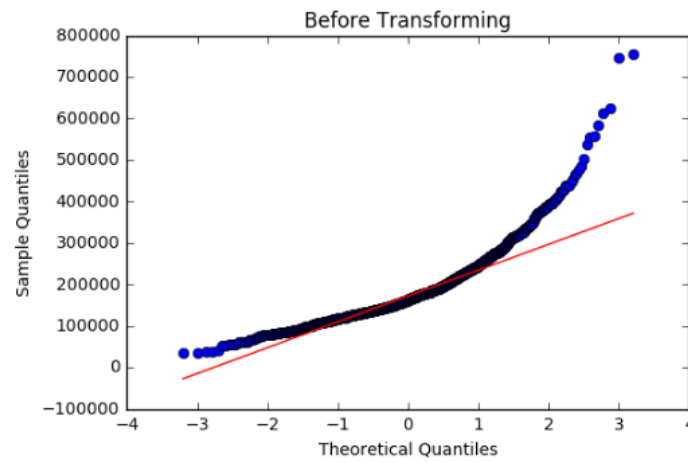
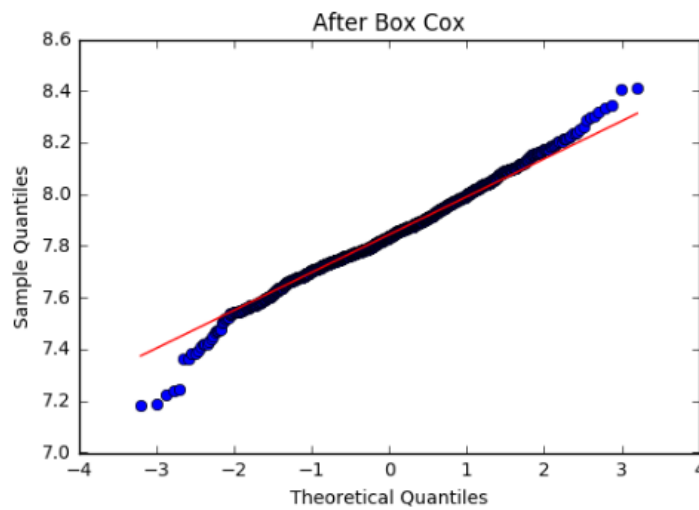


Figure 1.2: Q-Q plot before transforming y

Figure 1.3: Q-Q plot after raising y with boxcox λ

transformation = $\log(y)$

else :

transformation = y^λ

For our data box cox transformation found the value of λ to be -0.07692. Which is very near to zero. So tried both log and power transformation and here are the Q-Q plot for all three.

Although it seems that both of them are making y equally good normal, letter in model building we will find that log transformation was giving slightly better accuracy. And hence we are log transforming target variable.

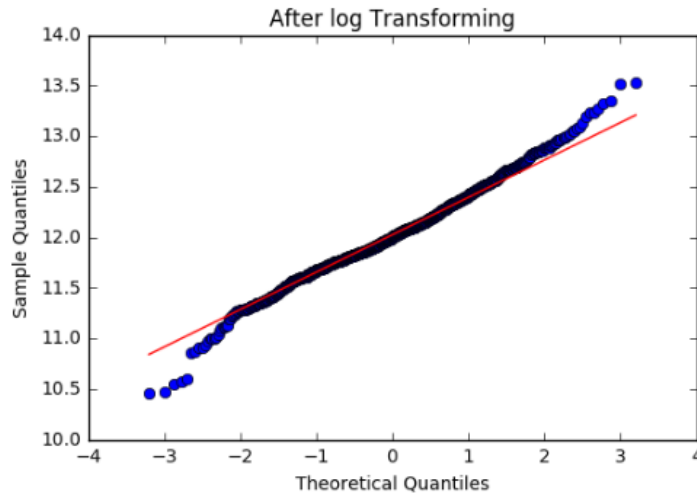


Figure 1.4: Q-Q plot after log transformation

1.3.4 Interaction Features

Many times it so happens that two or more features contribute with higher synergism. In such cases we include polynomial features in the model. Currently we are considering interaction only features, which means we are not raising single feature to a power n . We have kept degree $n=2$ which means we are generating multiplication of just two features. Doing so raised our features from 288 to 41617. And when we train them with simple ridge regression accuracy was decreased to 0.534. Another thing to note that our training set has 1460 samples against 41617 features. Feature selection become very important in such cases and we saw that Lasso with high regularization is also helping. So we are dropping out the interaction features for further analysis.

1.4 Error Metric

Through this work we will be using rmse of logarithm of house price as our error metric.

1.5 Detailed organization of the report

Chapter 2- Linear Regression We will try simple linear regression and generalized regularization techniques. We shall look the mathematic behind them all and how it fits/unfits our data. We shall also try various numerical techniques and compare them. We will also do PCA for removing collinearity.

Chapter 3- Bayesian Regression We shall try Bayesian regression with Markov chain Monte Carlo simulation. We shall again derive the mathematics behind it and see how it fits to our data. We shall see the role of prior as an expert advise and for the regularization.

Chapter 4-Summary : We will represent summary and direction of further work.

Chapter 2

Linear Regression And Regularization

2.1 Simple Linear Regression

Linear regression assumes predictor to be linear combination of independent variable. Let x_1, x_2, \dots, x_n be independent variables or features and y is predictor or target variables. Linear regression tries to fit following model :

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n + \epsilon$$

where b_0, b_1, \dots, b_n are model coefficients. ϵ is the error term which can be due to:

- We are missing some of the features
- Relationship between independent variable and target is not exactly linear

We define y' as a function of x given b .

$$y' = h_b(x)$$

$$h_b(x) = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$$

$$h_b(x) = b_0x_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n(x_0 = 1)$$

And hence

$$y = h_b(x) + \epsilon$$

We define cost function, J as the square of error terms.

$$J(b) = \sum_{i=1}^n (y_i - h_b(x))^2 \quad (2.1)$$

Our optimization objective is to minimize this function with respect to parameters b. There are several ways to do it:

- Solving matrix equation
- Gradient Descent
- Stochastic Gradient Descent

2.1.1 Solving matrix equation

If we take derivative of equation 2.1 with respect to each b_0, b_1, \dots, b_n and equate it to zero we can derive that

$$B = (X^T X)^{-1} X^T Y \quad (2.2)$$

where,

$$B = (b_0, b_1, \dots, b_n)$$

$X = (X_0, X_1, \dots, X_n)$ each X_i is column vector of each feature

$Y = (y_0, y_1, \dots, y_m)$ is target variable

We can solve it only when X is singular matrix, if not we need to use single value decomposition[5].

This also allows us to look at linear dependency between features and provides more intuition about it. We will talk about it in greater detail in section 2.5.2.

2.1.2 Gradient Descent

Gradient Descent start with some initial value of parameter $b_0, b_1, b_2, \dots, b_n$ and step parameter α . It calculated gradient with respect to each parameter b_i and moves in the negative direction of this gradient with setp size of α . Following steps keep repeating until it reached convergence for each b_i .

$$\begin{aligned}
 temp0 &= b_0 - \alpha \frac{\partial}{\partial b_0} J(b) \\
 temp1 &= b_1 - \alpha \frac{\partial}{\partial b_1} J(b) \\
 b_0 &= temp0 \\
 b_1 &= temp1
 \end{aligned} \tag{2.3}$$

(Keep repeating until convergence)

It is very import to choose α carefully. Small value of α takes more time to converge. Large value keep bounces from the optimum point. Typically we start with large value and after certain number off iteration we decrease it.

In this study we had written our own implementation of gradient descent. However it seemed diverging, we need to find out a reason for it after mid-semester. But one point to mention is Stochastic Gradient Descent (explained next) with sklearn implementation is also giving the rmse in the order of 10^{16} .

To debug the issue it always helps looking at the value of the cost function 2.1 after each iteration. We calculate value of the cost function for the each intermediate values of θ parameter and plot it as shown in Figure 2.1. As we can see it is diverging instead of converging.

What could be the reason behind such behavior. Mathematically from 2.3 we can see that step size is same for all the features irrespective of there range. So for the features whose range is in the unit of 500 will be incremented/decremented 100 time more than feature in the range of

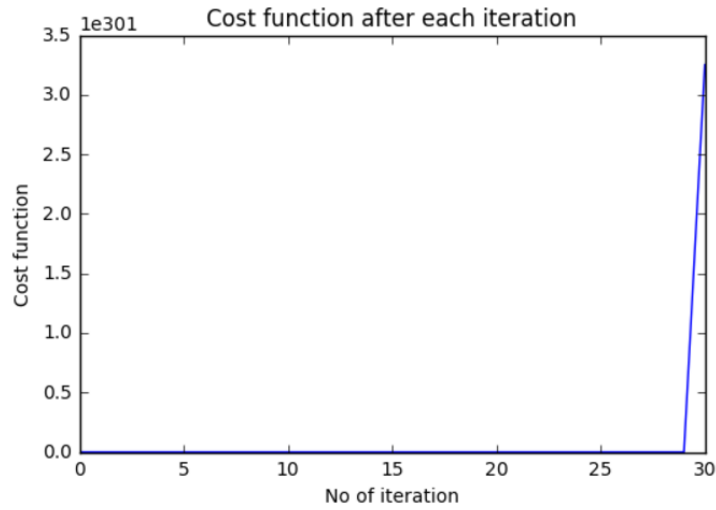


Figure 2.1: Cost function before normalization

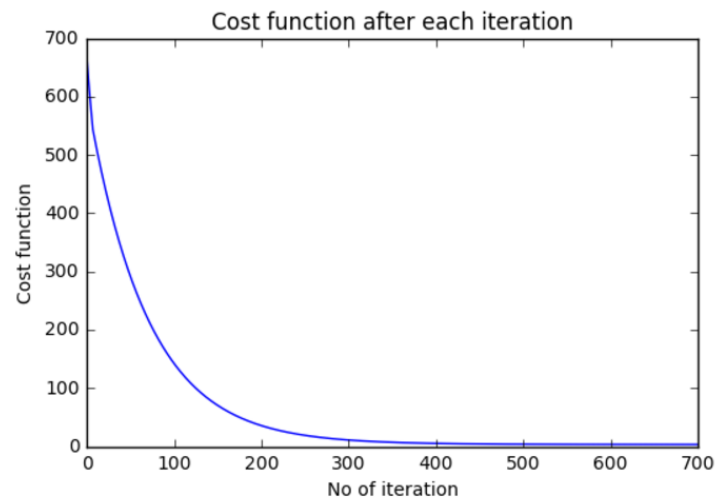


Figure 2.2: Cost function after normalization

5. To avoid this we should normalize all the features in the range of 0 to 1. We have used the min max normalization defined in 2.4

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (2.4)$$

Figure 2.2 show how cost function behaves after normalization. It clearly seems converging well. And the accuracy we found for 100000 iteration and alpha equal to 0.0005 was 0.701677.

2.1.3 Stochastic Gradient Descent

In gradient descent we considered cost function J as the summation over all the samples. We can also do it considering just one sample for calculating J and update the parameters $b_0, b_1, b_2, \dots, b_n$ based on that. This method is called stochastic gradient descent. Advantage here is that learning is fast. It is widely used method for on-line learning in real time applications. However one disadvantage is that it does not try to minimize error for the collection of sample. So this should be avoided when no of sample are less. In our case we have 1460 samples and predictors are 280 after leveling. So the error we have after stochastic gradient descent was in the order of 10^{16} .

2.2 Ridge Regression

One downside of using linear regression is that, It sometimes tends to over-fit the data. What this means that we get great accuracy on train data but it will not generalize well to test data-sets. Over-fitting is know as high variance scenario. We can reduce it by keeping coefficient value as small as possible. To achieve this Ridge regression introduces regularization parameter λ . It is also known as Tikhonov regression. So cost function now becomes :

$$J = \sum_{i=1}^n (y_i - y'_i)^2 + \lambda \sum_{i=1}^p (b_i)^2 \quad (2.5)$$

We can see that the regularization term tries to keep b values as small as possible.

The higher the value of λ , smaller will be the value of B -parameters. (b_0, b_1, \dots, b_n). In other words it tries to minimize error function, keeping B -values as small as possible.

Optimum value of λ can be obtained via cross validation. In our case we found it to be 15. Figure 2.3 shows the plot of accuracy against various values of λ .

Accuracy for $\lambda=15$ came out to be 0.139767, which is an improvement over simple linear regression value of 0.1571221.

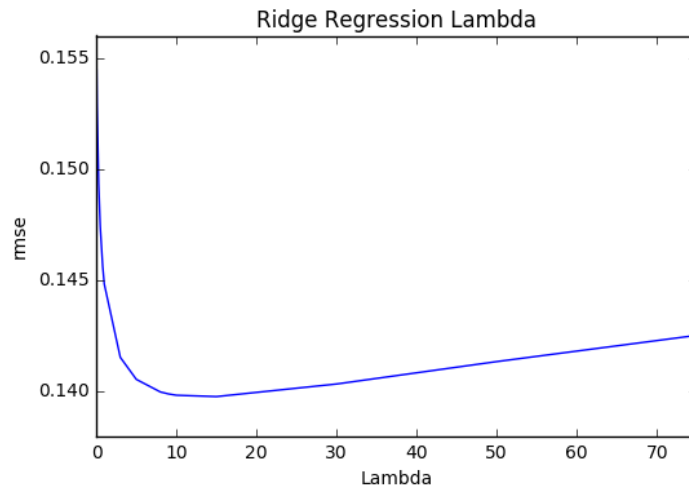


Figure 2.3: Tuning Lambda for Ridge Regression

Earlier we had said that ridge tries to make coefficient as close to zero as possible. Figure 2.7 and 2.8 shows coefficient ranges for simple linear regression and Ridge regression. The reason for this shrinkage is explained in section 2.4.

However it is important to note here is that shrinkage of coefficients implies reducing variance and increasing bias. Reduction of variance prevents model from over-fitting.

2.2.1 Linear Algebra Solution

If we take partial derivation of equation 2.5 with each parameter b_0, b_1, \dots, b_n and equate it to zero, It can be shown that

$$B = (\lambda * I + X^T X)^{-1} X^T Y$$

where I is the identity matrix. Good thing about solving above matrix inverse is that it will always be singular and normalization is not needed when we solve it via this algebraic method.

In this case also we were required to tune λ parameter, we did via cross validation and it is shown in 2.2.1. In this case it came out to be 10 while in earlier case it was 15. We have got rmse of 0.012 with this approach and it seems remarkable.

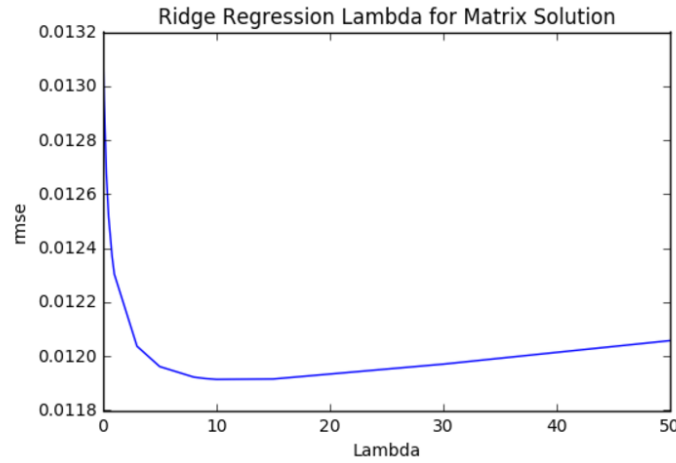


Figure 2.4: Tuning λ parameter for matrix approach for Ridge Regression

2.3 Lasso Regression

Now Ridge regression has one limitation. It moves coefficients towards zero but does not make it zero. Why would we want to make them zero? We need it when we want more interpretable model and there are very high number of features. Here is how cost function of Lasso looks :

$$J = \sum_{i=1}^n (y_i - y'_i)^2 + \lambda \sum_{i=1}^n |b_i|$$

Also Figure 2.5 shows the plot for lambda for our housing dataset. It seems that it is suggesting to avoid Lasso regression as accuracy is more when we when we have low lambda. Again it seems this will always be the case, statistics says that adding more variable will always improve accuracy, what we look for is how much gain are we getting at the expense of losing interpretability. Accuracy we got for $\lambda = 1$ was 0.197663, more than that of simple linear regression 0.157122.

2.4 Shrinkage Property Of Ridge and Lasso

How does Ridge and Lasso shrink coefficients? Figure 2.6 [3] provides intuition for that. Right side figure is the case for ridge regression. β' is the optimal least square solution and contours

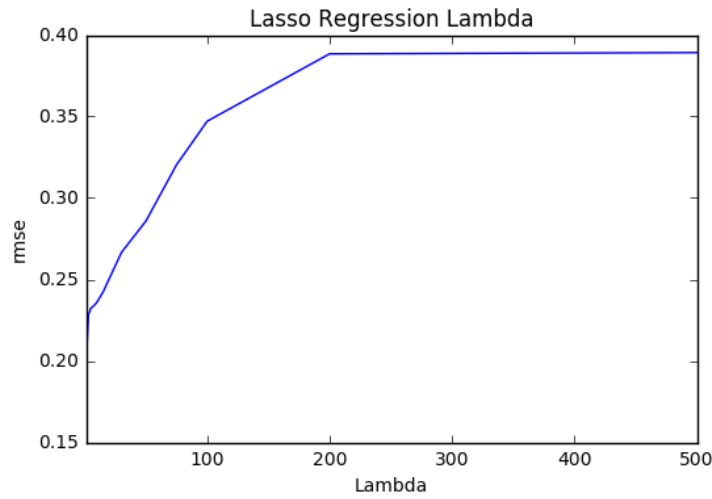


Figure 2.5: Tuning Lambda for Lasso Regression

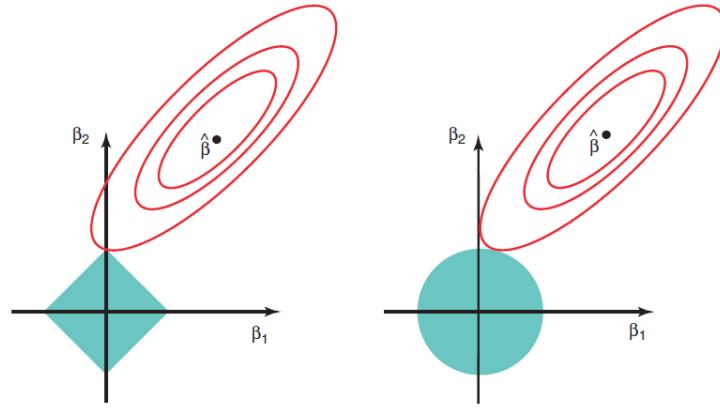


Figure 2.6: Shrinkage property of Ridge and Lasso

around it corresponds to constant error. As we can see that for Ridge, it can intersect with the circle at any point and intersecting point corresponds to coefficient.

However case is different with Lasso. Figure 2.6 on left shows constrain function for Lasoo. As we can see it intersects constrain region on axis. As axis represent each coefficient, we get a coefficient zero when contour intersects constraint function on axis. Looking at the shape of two it is clear that Lasso is more likely to make coefficient zero.

Bishop[1] suggest that we can generalize cost function by the following formula. Figure 2.10 shows its shape for various values of q .

$$J = \sum_{i=1}^n (y_i - y'_i)^2 + \lambda \sum_{i=1}^n |b_i|^q$$

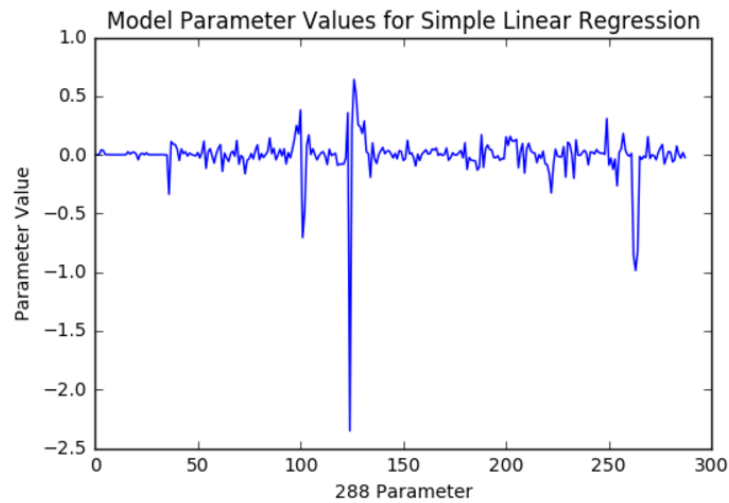


Figure 2.7: Coefficients for simple linear regression (Range of 0.5)

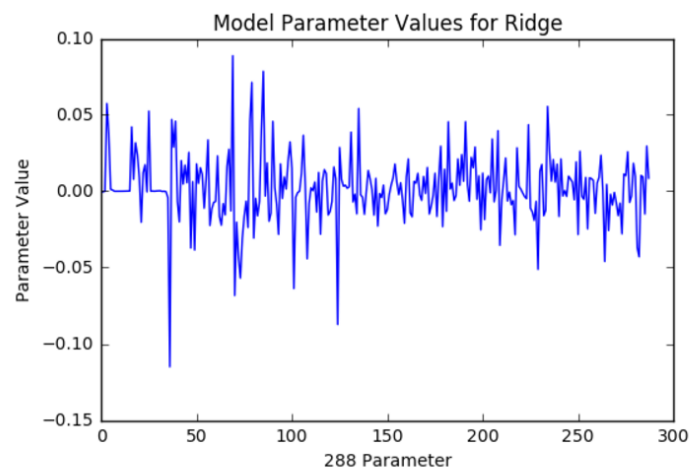


Figure 2.8: Coefficient for Ridge Regression $\lambda=15$ (Range of 0.005)

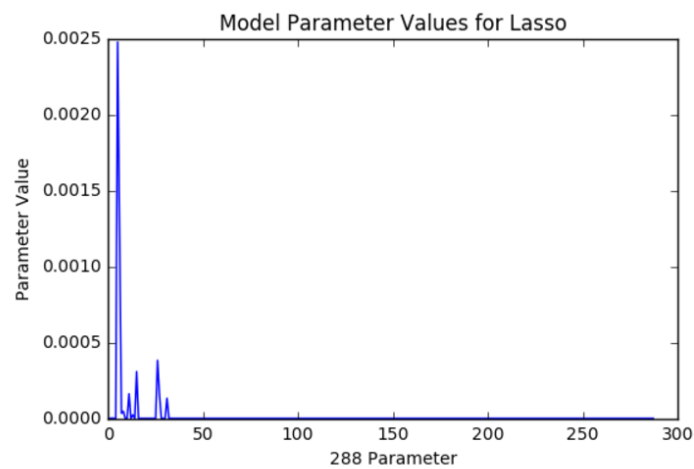
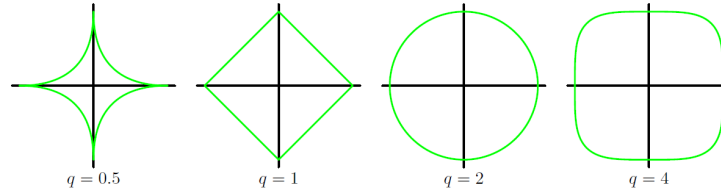


Figure 2.9: Coefficient with Lasso Regression $\lambda=1$ (Many of them are zero)

Figure 2.10: Constraint functions for different values of q

2.5 Regression with Dimensionality Reduction

2.5.1 Principle Component Regression

All the regression techniques explained so far has one potential difficulties when collinearity is present in the data. When there are correlated variable as features parameters have high variance. Having parameters with high variance has disadvantages, first we can not make concrete inference about the model and second, even for prediction it will perform poorly on test data. As we have seen in the equation 2.2 finding parameter involves finding inverse of the matrix $X^T X$. We know from linear algebra is that set of equations have multiple solutions if rank is less than number of variables. That is the exact reason why collinearity increase variance in the estimation. We now have more possible values for the same parameter.

Simple collinearity between features can be detected by scatter plot. But what if some of two or more features correlates with other features. To solve this we need some robust technique.

We solve this problem with the help of Principle Component Analysis(PCA). We do the PCA using Single Value Decomposition (SVD) and finding eigen values/eigen vector. This technique basically projects the data into n orthogonal dimensions (defined by eigen vector) where n represents number of features. SVD gives us three matrices, U , s and V . Matrix s basically is diagonal matrix and contains eigen values of X^T , which also represent importance of each orthogonal vectors.

$$X^T X = U.s.V^T$$

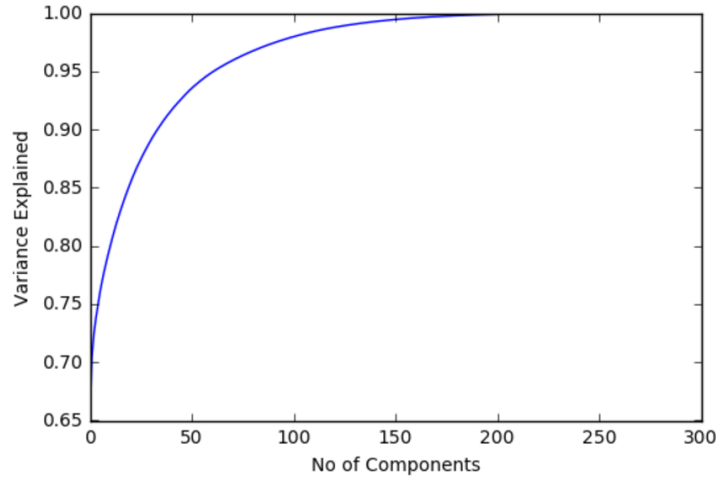


Figure 2.11: Variance explained by Principle Components

As eigen values represents variance explained by each direction, we can draw a plot as show in figure 2.11. To give the numbers we get the variance explained by first 100 component is 0.979743. And the accuracy we get is 0.14949.

One important thing we learned was to get the correct estimation of explained variance we must normalize each feature.

2.5.2 Relation of PCR with Ridge

Relationship between Ride regression and PCR has been identified at [2] If we decompose input X as $X = U.s.V^T$ following equation can be derived from simple linear regression (OLS-Ordinary Least Square) and for the ridge regression.

$$\hat{\mathbf{y}}_{\text{OLS}} = \mathbf{X}\beta_{\text{OLS}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{U}\mathbf{U}^T\mathbf{y}.$$

$$\hat{\mathbf{y}}_{\text{ridge}} = \mathbf{X}\beta_{\text{ridge}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{U} \text{diag} \left\{ \frac{s_i^2}{s_i^2 + \lambda} \right\} \mathbf{U}^T\mathbf{y}.$$

And for PCR we saw in the previous section that we are keeping some components rejecting others entirely.

$$\hat{\mathbf{y}}_{\text{PCR}} = \mathbf{X}_{\text{PCA}} \beta_{\text{PCR}} = \mathbf{U} \text{diag}\{1, \dots, 1, 0, \dots, 0\} \mathbf{U}^\top \mathbf{y},$$

Looking at the formulas of $\hat{\mathbf{y}}_{\text{ridge}}$ we notice that when regularization parameter λ penalizes smaller singular value s_i more. On the contrary in $\hat{\mathbf{y}}_{\text{PCR}}$ smaller values are removed at all (by making it zero). So we can see that ridge regression is kind of smooth version of PCR.

2.6 Summary

In this chapter we applied housing data to various regression and regularization model. We observed that regularization helps reducing coefficient, which in turn reduces the variance and makes the model more generalize. We also saw that Lasso does not make sense when number of features are small (as in our case). It would have been helpful if there were say 10k features. We also implemented various optimization techniques like solving matrix equation and gradient descent. We found that normalizing features is must for the gradient descent. Table 3.1 summarizes our effort to improve accuracy for each of them.

Table 2.1: Summary for Linear Regression And Regularization

Method	Accuracy
Simple Linear Regression (Ordinary Least Square)	0.157122
Simple Linear Regression (With Gradient Descent without normalizing data)	It was diverging
Principle Component Regression	0.14949
Simple Linear Regression (With Gradient Descent after normalizing data, step size = 0.005, no of Iterations = 5000)	0.67097
Simple Linear Regression (With Stochastic Gradient Descent)	10^{17}
Ridge Regression (svd, instead of finding inverse, $\lambda = 15$)	0.139767
Ridge Regression (Matrix Solution, $\lambda = 15$)	0.140074
Ridge Regression ($\lambda = 15$ With Stochastic Average Gradient Descent [7])	0.233114
Lasso Regression (coordinate descent, $\lambda = 1$, non-zero coefficient 11)	0.197663

Chapter 3

Bayesian Regression

When we have very small amount of data frequentest approach can not generalize the model well. However we can use Bayesian approaches with proper prior to get insight of the process. On the other end when we see hierarchy in the data again Bayesian methods helps to generalize parameter for different clusters. In this chapter we will see some simple Bayesian models for regression and regularization.

3.1 Introduction

Our goal is still the same. We want to find out parameters b_0, b_1, \dots, b_n and ϵ in the following equation.

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n + \epsilon$$

But we don't want to minimize cost function. We say that we already have some idea how parameters B's will look like, that is our prior. Next these parameters ccan be tuned further by evidence D. To tune this we will compute how likely evidence (data) is given the B parameters. This is called as likelihood. Here comes the difference, simple linear regression discussed on the previous chapter was only based on likelihood without priors.

We will be using Bayes equation to obtain these parameters :

$$p(B|D) = \frac{p(D|B)p(B)}{p(D)} \quad (3.1)$$

$$Posterior = \frac{Likelihood \times Prior}{Evidence} \quad (3.2)$$

3.2 Markov chain Monte Carlo

As Bayesian formula is complex to solve analytically, we will use the simulation. In the simulation we are trying to draw independent sample from the posterior distribution. Once we have lot of samples we are able to describer the posterior distribution.

Now onwards we will use short notation called mcmc for Markov chain Monte Carlo algorithm. And also there are many implementations of mcmc algorithm but we will explain it with the context of Metropolis-Hasting algorithm.

So mcmc algorithm combine two popular algorithms Monte Carlo and Markov chain. Traditionally Monte Carlo algorithm is used draw samples from the random processes given the mathematical formulation of that process. So why can't we use it alone? Reason is solving Bayes equation is not a simple mathematical formulation, specially the denominator. Now how does introducing Markov chain solves this problem. As we explain below it takes the ratio of posterior to sample from which cancels out the denominator term.

3.2.1 MCMC Steps

So we first specify our assumption about posterior distribution. Lets assume it to have mean μ and standard deviation of 1. Our goal is draw many sample from the posterior so we can effectively characterize μ .

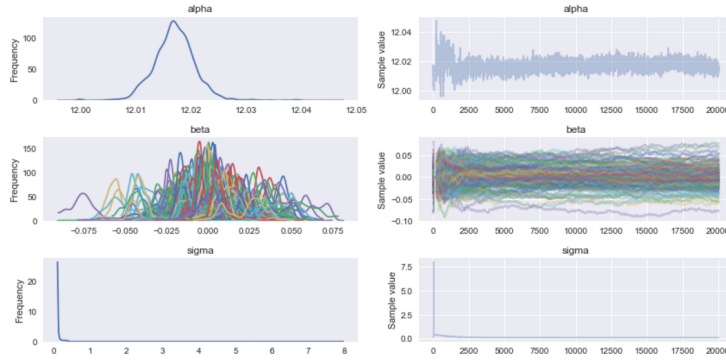


Figure 3.1: Simulation Traces

So we have some initial current values of μ . Next we sample the proposed μ for the distribution centered at current μ and standard deviation of 1.

```

muProposal = norm(muCurrent, StdDev).rvs()
likelihoodCurrent = norm(muCurrent, 1).pdf(data).prod()
likelihoodProposal = norm(muProposal, 1).pdf(data).prod()
priorCurrent = norm(muPriorMu, muPriorSd).pdf(muCurrent)
priorProposal = norm(muPriorMu, muPriorSd).pdf(muProposal)
probCurrent = likelihoodCurrent * priorCurrent
probProposal = likelihoodProposal * priorProposal
probAccept = probProposal / probCurrent
accept = np.random.rand() < probAccept
if (accept == True) then muCurrent = muProposal

```

Then we calculate likelihood and prior for each proposed μ and current μ and take the ratio. One such model is available in appendix D. Figure 3.1 shows values mcmc sample in each of 20k samples.

3.3 Normal and log-logistic Distribution for Target

As discussed above in Bayesian modeling we need to specify distribution of our unknown variables(parameters and outcomes). In our case outcome is housing price. We tried to simulate the

model with two distribution for house price, Normal distribution and log-logistic distribution.

In pymc3 if some distribution is not available in library we can specify our custom distribution by providing likelihood function. So here is how we derived likelihood function for log-logistic.

$$p(x|a, b) = \frac{(b/a)(x/a)^{b-1}}{(1 + (x/a)^b)^2}$$

$$\log p = \log(b/a) + (b-1)\log(x/a) - 2\log(1 + (x/a)^b)$$

Where a and b are parameters of the model. And mean is given by :

$$mean = \begin{cases} \frac{a*(\pi/b)}{\sin(\pi/b)} & \text{if b is greater than 1} \\ undefined & \text{otherwise} \end{cases} \quad (3.3)$$

Model using this custom distribution is available at Appendix C.

3.4 Regularization in Bayesian

3.4.1 Lasso Regularization

As we have seen earlier Lasso regularization is used for making more of the coefficients equal to zero and thus helping in feature selection. We can implement Lasso in Bayesian analysis by selecting a prior distribution which is centered at zero and quickly shrinks as we move away from zero. One such a distribution is Laplace distribution, which can be seen as two mirror exponentials.

$$f(x | \alpha, \beta) = \frac{1}{2b} \exp \left\{ -\frac{|x - \mu|}{b} \right\} \quad (3.4)$$

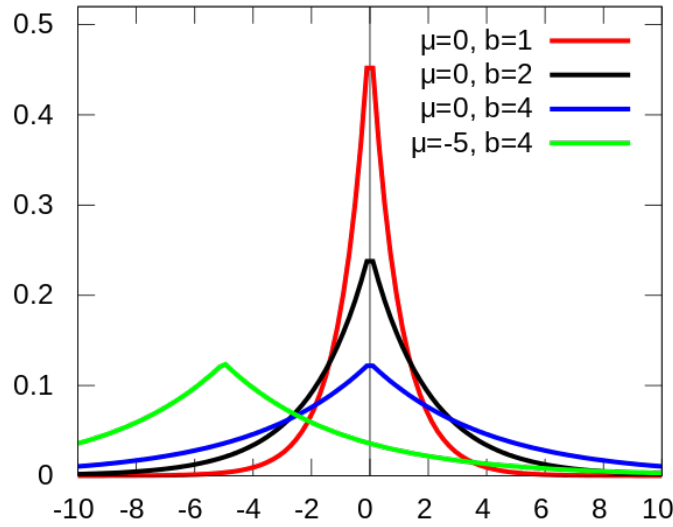


Figure 3.2: Laplace Distribution

3.5 Ridge Regularization

Earlier in this chapter we were taking Normal Distribution as a prior which actually accounts for the Ridge regularization, here is the derivation for that.

So here is the effect we saw post Lasso regularization. We count the no of parameters having absolute value less than 10^{-4} in both cases. Before lasso it was just one and after lasso it was 22. Also the rmse was reduced to 0.0083, which is the best so far.

From equation 3.2 we can say that

$$p(B|D) = \frac{p(D|B)p(B)}{p(D)} \quad (3.5)$$

$$\propto p(D|B) \times p(B) \quad (3.6)$$

$$= \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_i - B^T x_i)^2}{2\sigma^2}} \times \prod_{i=1}^p \frac{1}{\tau\sqrt{2\pi}} e^{-\frac{(B_i)^2}{2\tau^2}} \quad (3.7)$$

$$(3.8)$$

Table 3.1: Summary for Bayes

Method	Accuracy
Normal Distribution for House Price	0.2765
Log Logistic Distribution for House Price	0.311
Laplace Distribution for parameters (Lasso)	0.11773
Normal Distribution for Parameters (Ridge)	0.2765

So now we maximize likelihood considering the prior as well.

$$B = \arg \max_B e^{-\frac{1}{2\sigma^2} \sum_1^n (y_i - \hat{y}_i)^2 - \frac{1}{2\tau^2} \sum_1^p \theta^2} \quad (3.9)$$

$$= \arg \max_B -\frac{1}{2\sigma^2} \sum_1^n (y_i - \hat{y}_i)^2 - \frac{1}{2\tau^2} \sum_1^p \theta^2 \quad (3.10)$$

$$= \arg \max_B -1 \left(\sum_1^n (y_i - \hat{y}_i)^2 + \frac{\sigma^2}{\tau^2} \sum_1^p \theta^2 \right) \quad (3.11)$$

$$= \arg \min_B \sum_1^n (y_i - \hat{y}_i)^2 + \frac{\sigma^2}{\tau^2} \sum_1^p \theta^2 \quad (3.12)$$

If we consider λ as $\frac{\sigma^2}{\tau^2}$ it is very similar to equation 2.5.

3.6 Summary

We studied fundamental of Bayesian learning and how mcmc makes complex calculations simpler. We also compared performance of two distribution for our housing data. Result was that with Normal Distribution accuracy was 0.273 and with log logistic it was 0.311.

Chapter 4

Summary

4.1 Summary

We tried out linear regressions with both frequentist and Bayesian approaches. Outcomes of this effort can be summarized in following points :

- Linear regression assumes linear relationship between predictor and dependent variables. If it is not then several transformations can be applied if they are not linear.
- For simple regression solving by matrix inverse gives the best fit. But sometimes because of collinearity present in the data such inverse might not exist. In that case we can solve the equation by gradient descent.
- Normalization is important while performing gradient descent, we explained the mathematical intuition behind it saying that step size is same for each parameters.
- Also we now know how to debug such issues by plotting learning curve of cost function.
- We tried out various regularization methods. We also saw the method on constrained region to visualize how regularization keeps unknown parameter values small.
- We learned ridge regression and closed form solution for the same. We noted that such a solution is always available for Ridge.

- We applied Lasso regularization as a means of feature selection.
- We used Principle component regression to remove collinearity and how pca related to ridge regression.
- We learned how Bayesian analysis includes prior information in the analysis. We also saw that it is difficult to sample from posterior and how mcmc sampling provides solution to that problem.
- We assumed various distribution for house price and checked it for accuracy.
- We also used Laplace prior to simulated Lasso and observed the effect of zero coefficients.

Table 4.1 below contains accuracy for various approaches we have tried :

4.2 Directions For Future Work

Bayesian prior allows us to include expert advice for different coefficients differently, we may train the model considering external policy like demonetization will affect the house prices. Also control system can be developed in a sense that we have a model ready for some years say 2010. As we get the data for next year we calculate the error and adjust the parameters accordingly. One more thing can be done is hierarchical Bayesian modeling, where in we make a cluster of say houses created before 1950 and after 1950. And for this two clusters we assume parameters to come from same distribution.

On frequentist front we can try kernel PCA. Also there is a scope of more sophisticated feature engineering.

Table 4.1: Accuracy for various models

Method	Accuracy
Simple Linear Regression (With Gradient Descent without normalizing data)	It was diverging
Principle Component Regression	0.14949
Simple Linear Regression (With Gradient Descent after normalizing data, step size = 0.005, no of Iterations = 5000)	0.67097
Simple Linear Regression (With Stochastic Gradient Descent)	10^{17}
Ridge Regression (svd, instead of finding inverse, $\lambda = 15$)	0.139767
Ridge Regression (Matrix Solution, $\lambda = 15$)	0.140074
Ridge Regression ($\lambda = 15$ With Stochastic Average Gradient Descent [7])	0.233114
Lasso Regression (coordinate descent, $\lambda = 1$, non-zero coefficient 11)	0.197663
Bayesian Normal Distribution for House Price	0.2765
Bayesian Log Logistic Distribution for House Price	0.311
Bayesian Laplace Distribution for parameters (Lasso)	0.11773
Bayesian Normal Distribution for Parameters (Ridge)	0.2765

Appendix A

Detailed Data 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

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

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl Gravel

Pave Paved

Alley: Type of alley access to property

Grvl Gravel

Pave Paved

NA No alley access

LotShape: General shape of property

Reg Regular

IR1 Slightly irregular

IR2 Moderately Irregular

IR3 Irregular

LandContour: Flatness of the property

Lvl Near Flat/Level

Bnk Banked - Quick and significant rise from street grade to building

HLS Hillside - Significant slope from side to side

Low Depression

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

LotConfig: Lot configuration

Inside Inside lot

Corner Corner lot

CulDSac Cul-de-sac

FR2 Frontage on 2 sides of property

FR3 Frontage on 3 sides of property

LandSlope: Slope of property

Gtl Gentle slope

Mod Moderate Slope

Sev Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn Bloomington Heights

Blueste Bluestem

BrDale Briardale

BrkSide Brookside

ClearCr Clear Creek
CollgCr College Creek
Crawfor Crawford
Edwards Edwards
Gilbert Gilbert
IDOTRR Iowa DOT and Rail Road
MeadowV Meadow Village
Mitchel Mitchell
Names North Ames
NoRidge Northridge
NPkVill Northpark Villa
NridgHt Northridge Heights
NWAmes Northwest Ames
OldTown Old Town
SWISU South & West of Iowa State University
Sawyer Sawyer
SawyerW Sawyer West
Somerst Somerset
StoneBr Stone Brook
Timber Timberland
Veenker Veenker

Condition1: Proximity to various conditions

Artery Adjacent to arterial street
Feedr Adjacent to feeder street
Norm Normal
RRNn Within 200' of North-South Railroad
RRAn Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.

PosA Adjacent to positive off-site feature

RRNe Within 200' of East-West Railroad

RRAe Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery Adjacent to arterial street

Feedr Adjacent to feeder street

Norm Normal

RRNn Within 200' of North-South Railroad

RRAn Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.

PosA Adjacent to positive off-site feature

RRNe Within 200' of East-West Railroad

RRAe Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam Single-family Detached

2FmCon Two-family Conversion; originally built as one-family dwelling

Duplx Duplex

TwnhsE Townhouse End Unit

TwnhsI Townhouse Inside Unit

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

OverallCond: Rates the overall condition 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

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat Flat

Gable Gable

Gambrel Gabrel (Barn)

Hip Hip

Mansard Mansard

Shed Shed

RoofMatl: Roof material

ClyTile Clay or Tile

CompShg Standard (Composite) Shingle

Membran Membrane

Metal Metal

Roll Roll

Tar&Grv Gravel & Tar

WdShake Wood Shakes

WdShngl Wood Shingles

Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles
AsphShn Asphalt Shingles
BrkComm Brick Common
BrkFace Brick Face
CBlock Cinder Block
CemntBd Cement Board
HdBoard Hard Board
ImStucc Imitation Stucco
MetalSd Metal Siding
Other Other
Plywood Plywood
PreCast PreCast
Stone Stone
Stucco Stucco
VinylSd Vinyl Siding
Wd Sdng Wood Siding
WdShing Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng Asbestos Shingles
AsphShn Asphalt Shingles
BrkComm Brick Common
BrkFace Brick Face
CBlock Cinder Block
CemntBd Cement Board
HdBoard Hard Board
ImStucc Imitation Stucco

MetalSd Metal Siding
Other Other
Plywood Plywood
PreCast PreCast
Stone Stone
Stucco Stucco
VinylSd Vinyl Siding
Wd Sdng Wood Siding
WdShing Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn Brick Common
BrkFace Brick Face
CBlock Cinder Block
None None
Stone Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex Excellent
Gd Good
TA Average/Typical
Fa Fair
Po Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block

PConc Poured Concrete

Slab Slab

Stone Stone

Wood Wood

BsmtQual: Evaluates the height of the basement

Ex Excellent (100+ inches)

Gd Good (90-99 inches)

TA Typical (80-89 inches)

Fa Fair (70-79 inches)

Po Poor (<70 inches)

NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent

Gd Good

TA Typical - slight dampness allowed

Fa Fair - dampness or some cracking or settling

Po Poor - Severe cracking, settling, or wetness

NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd Good Exposure

Av Average Exposure (split levels or foyers typically score average or above)

Mn Minimum Exposure

No No Exposure

NA No Basement

BsmtFinType1: Rating of basement finished area

GLQ Good Living Quarters

ALQ Average Living Quarters

BLQ Below Average Living Quarters

Rec Average Rec Room

LwQ Low Quality

Unf Unfinished

NA No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ Good Living Quarters

ALQ Average Living Quarters

BLQ Below Average Living Quarters

Rec Average Rec Room

LwQ Low Quality

Unf Unfinished

NA No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor Floor Furnace

GasA Gas forced warm air furnace

GasW Gas hot water or steam heat

Grav Gravity furnace

OthW Hot water or steam heat other than gas

Wall Wall furnace

HeatingQC: Heating quality and condition

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

CentralAir: Central air conditioning

N No

Y Yes

Electrical: Electrical system

SBrkr Standard Circuit Breakers & Romex

FuseA Fuse Box over 60 AMP and all Romex wiring (Average)

FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair)

FuseP 60 AMP Fuse Box and mostly knob & tube wiring (poor)

Mix Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ Typical Functionality

Min1 Minor Deductions 1

Min2 Minor Deductions 2

Mod Moderate Deductions

Maj1 Major Deductions 1

Maj2 Major Deductions 2

Sev Severely Damaged

Sal Salvage only

faces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex Excellent - Exceptional Masonry Fireplace

Gd Good - Masonry Fireplace in main level

TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in

Fa Fair - Prefabricated Fireplace in basement

Po Poor - Ben Franklin Stove

NA No Fireplace

GarageType: Garage location

2Types More than one type of garage

Attchd Attached to home

Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room above garage)

CarPort Car Port

Detchd Detached from home

NA No Garage

GarageYrBltd: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

GarageCond: Garage condition

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

PavedDrive: Paved driveway

Y Paved

P Partial Pavement

N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

NA No Pool

Fence: Fence quality

GdPrv Good Privacy

MnPrv Minimum Privacy

GdWo Good Wood

MnWw Minimum Wood/Wire

NA No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev Elevator

Gar2 2nd Garage (if not described in garage section)

Othr Other

Shed Shed (over 100 SF)

TenC Tennis Court

NA None

MiscVal: \$Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD Warranty Deed - Conventional

CWD Warranty Deed - Cash

VWD Warranty Deed - VA Loan

New Home just constructed and sold

COD Court Officer Deed/Estate

Con Contract 15% Down payment regular terms

ConLw Contract Low Down payment and low interest

ConLI Contract Low Interest

ConLD Contract Low Down

Oth Other

SaleCondition: Condition of sale

Normal Normal Sale

Abnorml Abnormal Sale - trade, foreclosure, short sale

AdjLand Adjoining Land Purchase

Alloca Allocation - two linked properties with separate deeds, typically condo wi

Family Sale between family members

Partial Home was not completed when last assessed (associated with New Homes)

Appendix B

Missing Value Notebook

```
In [206]: import pandas as pd

import matplotlib.pyplot as pyplot

%pylab inline

from IPython.display import display

pd.options.display.max_columns = None

from sklearn.linear_model import Ridge, LinearRegression, LassoLarsCV, Lasso
from sklearn.model_selection import cross_val_score
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from scipy import stats
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn import preprocessing
import seaborn as sns

sns.set(color_codes=True)

from scipy.stats.stats import pearsonr

from azureml import Workspace

ws = Workspace()

ds = ws.datasets['county_facts_dictionary.csv']

frame = ds.to_dataframe()

trainDF = ws.datasets['ames_train.csv'].to_dataframe();
testDF = ws.datasets['ames_test.csv'].to_dataframe();
```

```

sampleSubmissionDF = ws.datasets['ames_sample_submission.csv'].to_dataframe();

all_data = pd.concat((trainDF.loc[:, 'MSSubClass': 'SaleCondition'],
                      testDF.loc[:, 'MSSubClass': 'SaleCondition']))

y = trainDF.SalePrice
trainDF.SalePrice = log(y)

```

Populating the interactive namespace from numpy and matplotlib

```

In [207]: def direct_rmse_error(actual, predicted):
           ans = np.sqrt(np.sum(np.square(actual-predicted))/len(actual))
           return ans

def my_cv(model, x, y, n_splits):
    kf = KFold(n_splits=n_splits)
    kf.get_n_splits(X_train)
    ary = []
    for train_index, test_index in kf.split(x):
        Xtr, Xte = x.ix[train_index, :], x.ix[test_index, :]
        Ytr, Yte = y[train_index], y[test_index]
        model.fit(Xtr, Ytr)
        Y_pred = model.predict(Xte)
        ary.append(direct_rmse_error(Yte, Y_pred))
    return np.mean(ary)

```

```

In [208]: all_data_temp = all_data.copy()
          all_data = pd.get_dummies(all_data)
          all_data = all_data.fillna(all_data.mean())
          print (all_data.shape)
          X_train = all_data[:trainDF.shape[0]] #We are using raw selector operator

```



```
X_test = all_data[trainDF.shape[0]:]
y = trainDF.SalePrice
y = log(y)
cv_score = my_cv(LinearRegression(), X_train, y, 5)
print ("rmse using least square is = ", cv_score)
all_data=all_data_temp
```

(2919, 288)

rmse using least square is = 0.0132265181196

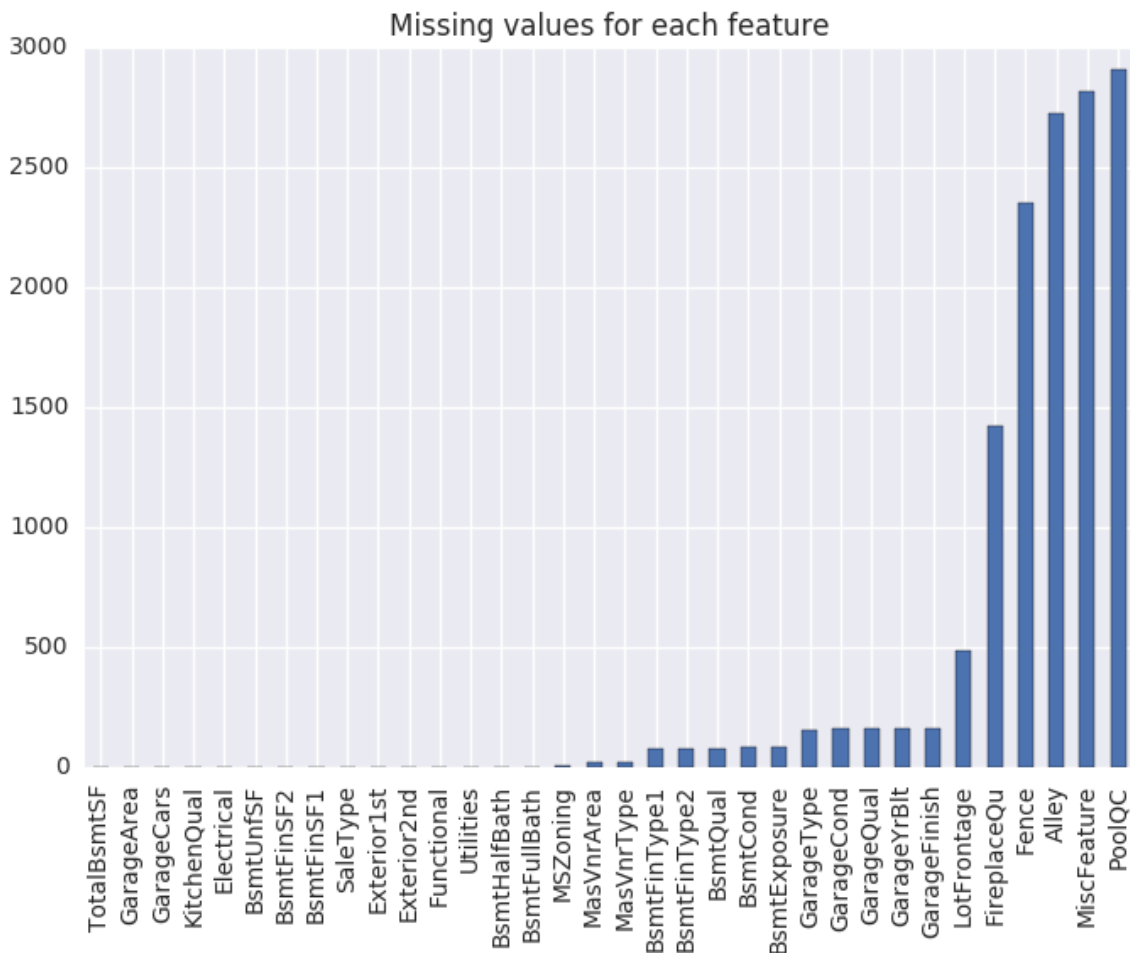
In []:

```
In [209]: missing = all_data.isnull().sum()
          missing.sum()
```

Out[209]: 13965

```
In [210]: missing = all_data.isnull().sum()
          missing = missing[missing > 0]
          missing.sort_values(inplace=True)
          missing.plot.bar(title='Missing values for each feature')
```

Out[210]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5108b2e80>



PoolQC

```
In [211]: all_data.loc[all_data.PoolQC.isnull(), 'PoolQC']='NA'
```

```
In [212]: all_data = all_data.replace({'PoolQC': {'Ex': 4,
                                                    'Gd': 3,
                                                    'TA': 2,
                                                    'Fa': 1,
                                                    'NA': 0
                                                    }
                                         })
```

Alley

```
In [213]: all_data.loc[all_data.Alley.isnull(), 'Alley']='NA'
```

Fence

```
In [214]: all_data.loc[all_data.Fence.isnull(), 'Fence']='NA'
```

```
In [215]: all_data = all_data.replace({'Fence': {'GdPrv': 4,
                                                'MnPrv': 3,
                                                'GdWo': 2,
                                                'MnWw': 1,
                                                'NA': 0
                                                }
                                       })
```

FirePlaceQu

```
In [216]: all_data.loc[all_data.FireplaceQu.isnull(), 'FireplaceQu']='NA'
```

```
In [217]: all_data = all_data.replace({'FireplaceQu': {'Ex': 5,
                                                         'Gd': 4,
                                                         'TA': 3,
                                                         'Fa': 2,
                                                         'Po': 1,
                                                         'NA': 0
                                                         }
                                       })
```

Utilities

```
In [218]: all_data.loc[all_data.Utilities.isnull(), 'Utilities']='AllPub'
```

Kitchen Quality

```
In [219]: def getCountCategorical(df, column):
            uniqueValues = pd.unique(df[column].values)
            for u in uniqueValues:
                temp=df[column]
                count = temp[df[column]==u].size
                print (u, " = ", count)
```

```
In [220]: getCountCategorical(all_data, 'KitchenQual')
```

```
Gd = 1151
```

```
TA = 1492
```

```
Ex = 205
```

```
Fa = 70
```

```
nan = 0
```

```
In [221]: all_data.loc[all_data.KitchenQual.isnull(), 'KitchenQual']='TA'
```

```
In [222]: all_data = all_data.replace({ 'KitchenQual': {'Ex': 5,
                                                         'Gd': 4,
                                                         'TA': 3,
                                                         'Fa': 2,
                                                         'Po': 1
                                                         }
                                     })
```

SaleType

```
In [223]: all_data.loc[all_data.SaleType.isnull(), 'SaleType']='WD'
```

MasVnrArea

```
In [224]: all_data.loc[all_data.MasVnrArea.isnull(), 'MasVnrArea']=0
```

MasVnrType

```
In [225]: all_data.loc[all_data.MasVnrType.isnull(), 'MasVnrType']='None'
```

Basement

```
In [226]: for c in ['TotalBsmtSF', 'BsmtUnfSF', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtHalfBat
            all_data.loc[all_data[c].isnull(), c]=0
```

```
In [227]: indexes = all_data['BsmtFinType1'].isnull()
            for c in ['BsmtFinType1', 'BsmtFinType2', 'BsmtQual', 'BsmtCond', 'BsmtExposur
            all_data.loc[indexes, c]='NA'
```

```
In [228]: indexes = all_data['BsmtFinType1'].isnull()
            for c in ['BsmtFinType1', 'BsmtFinType2', 'BsmtQual', 'BsmtCond', 'BsmtExposur
            all_data.loc[indexes, c]='NA'

            #We filtered on BsmtFinType1 because it had least no of Nulls. Now there wou
```

```
In [229]: getCountCategorical(all_data, 'BsmtCond')
```

TA = 2606

Gd = 122


```
        'NA':0
    }

    })
```

```
In [233]: getCountCategorical(all_data, 'BsmtExposure')
```

```
No    = 1904
Gd     = 276
Mn     = 239
Av     = 418
NA     = 79
nan    = 0
```

```
In [234]: all_data.loc[all_data.BsmtExposure.isnull(), 'BsmtExposure']='No'
          all_data = all_data.replace({ 'BsmtExposure': {'Gd': 4,
                                                         'Av': 3,
                                                         'Mn': 2,
                                                         'No': 1,
                                                         'NA':0
                                                         }
                                     })
```

```
In [235]: getCountCategorical(all_data, 'BsmtFinType2')
```

```
Unf    = 2493
BLQ     = 68
NA      = 79
ALQ     = 52
Rec     = 105
LwQ     = 87
```

nan = 0

MSZoning

Functional

```
In [238]: all_data.loc[all_data.Functional.isnull(), 'Functional']='Typ'

all_data = all_data.replace({ 'Functional': {'Typ': 7,
                                             'Min1': 6,
                                             'Min2': 5,
                                             'Mod': 4,
                                             'Maj1': 3,
                                             'Maj2': 2,
                                             'Sev': 1,
                                             'Sal': 0
```



```
}
```

```
})
```

```
In [239]: all_data.loc[all_data.Electrical.isnull(), 'Electrical']='SBrkr'
```

LotFrontage

```
In [240]: nullIndex = all_data.LotFrontage.isnull()
```

```
In [241]: nonNullIndex = all_data.LotFrontage.notnull()
```

```
In [242]: X_Train = np.sqrt(all_data.LotArea[nonNullIndex])
```

```
In [243]: Y_Train = all_data.LotFrontage[nonNullIndex]
```

```
In [244]: y= np.array(Y_Train)
```

```
In [245]: b=np.array(X_Train)
          x = b.reshape(len(b),1)
```

```
In [246]: model = LinearRegression()
```

```
In [247]: model.fit(x, y)
```

```
Out[247]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [248]: X_Test = np.sqrt(all_data.LotArea[nullIndex])
          b=np.array(X_Test)
          xt = b.reshape(len(b),1)
          yPred = model.predict(xt)
```

```
In [249]: all_data.loc[nullIndex, 'LotFrontage']=yPred
```

```
In [250]: all_data.LotFrontage.isnull().sum()
```

```
Out[250]: 0
```

Garage

```
In [251]: garage = all_data[['GarageCond', 'GarageQual', 'GarageYrBlt', 'GarageFinish',
```

```
In [252]: garage.head()
```

```
Out[252]:
```

	GarageCond	GarageQual	GarageYrBlt	GarageFinish	GarageType	GarageCars	\
0	TA	TA	2003.0	RFn	Attchd	2.0	
1	TA	TA	1976.0	RFn	Attchd	2.0	
2	TA	TA	2001.0	RFn	Attchd	2.0	
3	TA	TA	1998.0	Unf	Detchd	3.0	
4	TA	TA	2000.0	RFn	Attchd	3.0	

	GarageArea
0	548.0
1	460.0
2	608.0
3	642.0
4	836.0

```
In [253]: garage.GarageYrBlt.isnull().sum()
```

```
Out[253]: 159
```

```
In [254]: garage[garage.GarageYrBlt.isnull()].head()
```

```

Out [254]:   GarageCond  GarageQual  GarageYrBlt  GarageFinish  GarageType  GarageCars  \
          39         NaN         NaN         NaN         NaN         NaN         0.0
          48         NaN         NaN         NaN         NaN         NaN         0.0
          78         NaN         NaN         NaN         NaN         NaN         0.0
          88         NaN         NaN         NaN         NaN         NaN         0.0
          89         NaN         NaN         NaN         NaN         NaN         0.0

          GarageArea
          39         0.0
          48         0.0
          78         0.0
          88         0.0
          89         0.0

```

So when garage year blt is null, we can say that there is no garage. Looking at the data description it seems that GarageType, GarageFinish, GarageQuality, GarageCondition has NA values signifying No Garage. Replacing them.

```
In [255]: all_data.loc[all_data.GarageYrBlt.isnull(), 'GarageCond']='NA'
```

```
In [256]: all_data.loc[all_data.GarageYrBlt.isnull(), 'GarageFinish']='NA'
```

```
In [257]: all_data.loc[all_data.GarageYrBlt.isnull(), 'GarageQual']='NA'
```

```
In [258]: all_data.loc[all_data.GarageYrBlt.isnull(), 'GarageType']='NA'
```

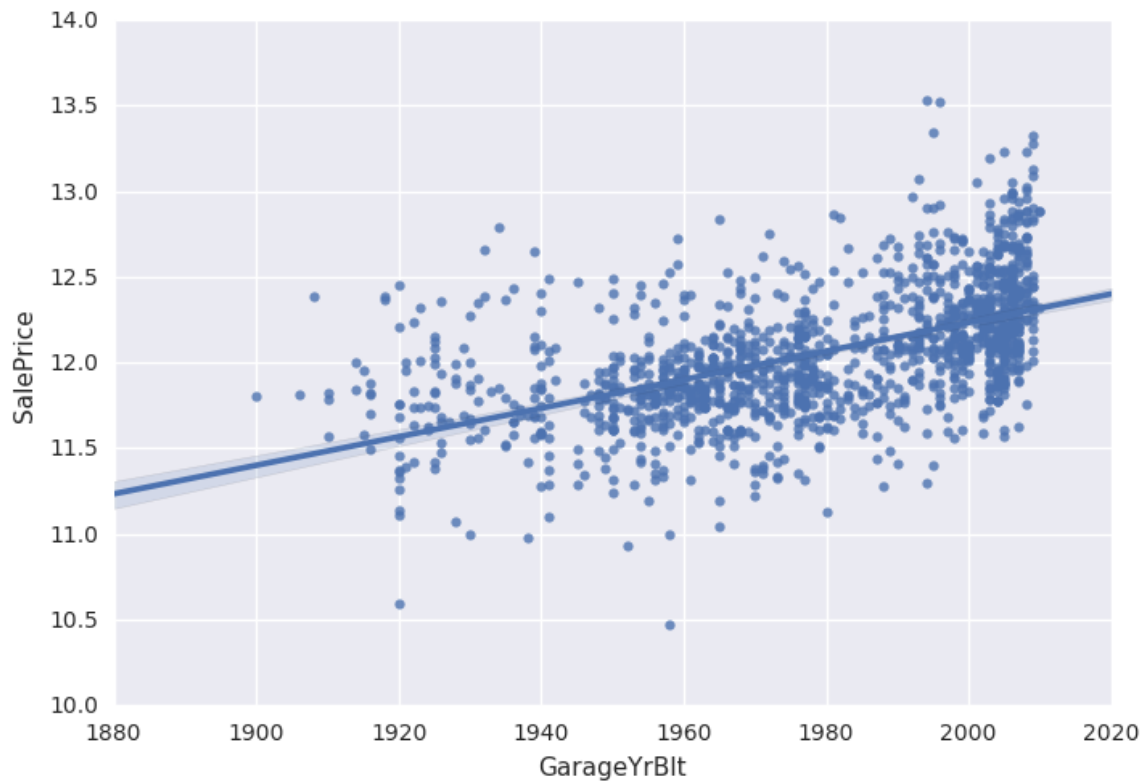
Now how to impute garage year built, when garage is not built. I think we should check the relation between Garage year build and sale price

```

In [259]: x = trainDF.GarageYrBlt[trainDF.GarageYrBlt.notnull()]
          y = trainDF.SalePrice[trainDF.GarageYrBlt.notnull()]
          sns.regplot(x, y)

```

```
Out[259]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb51089b9e8>
```

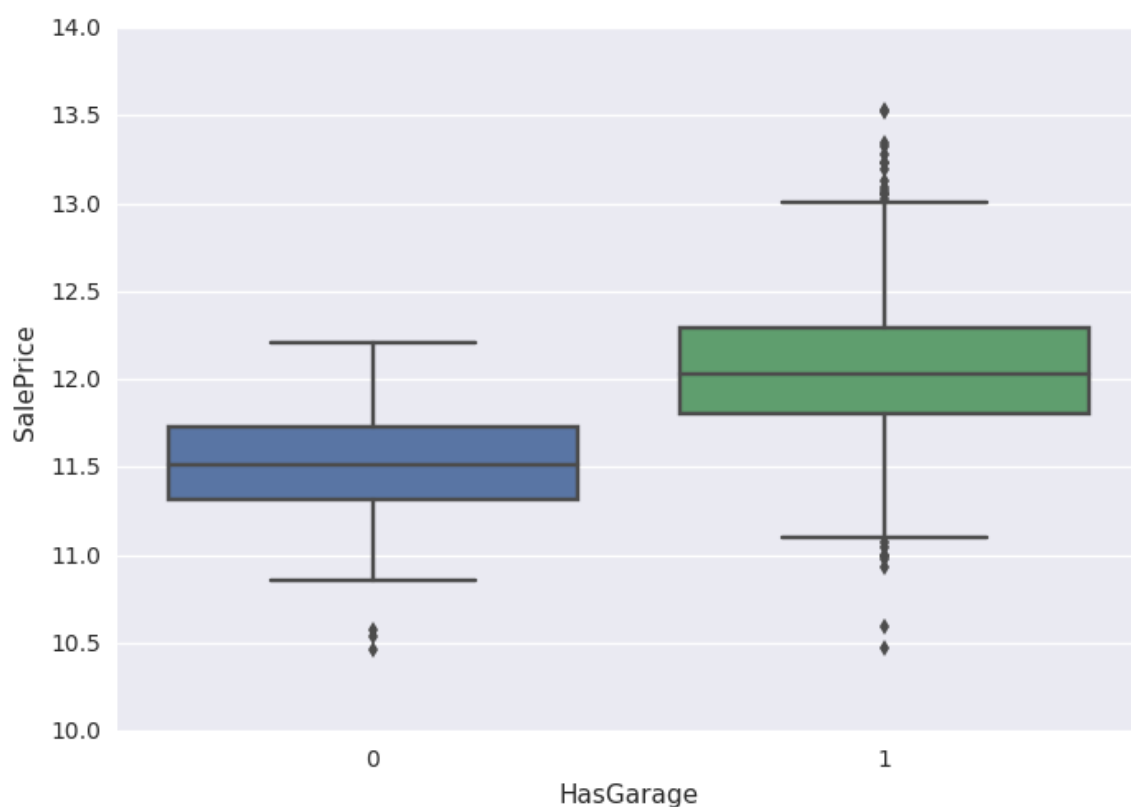


Lets create feature called has garage and draw boxplot

```
In [260]: trainDF['HasGarage']=trainDF.GarageYrBlt.apply(lambda x:0 if math.isnan(x) else 1)
```

```
In [261]: sns.boxplot(x='HasGarage', y='SalePrice', data=trainDF)
```

```
Out[261]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb510764d30>
```



Lets make it a zero and introduce a new variable called HasGarage.

```
In [262]: all_data.loc[all_data.GarageYrBlt.isnull(), 'GarageYrBlt']=0
          all_data['HasGarage']=all_data.GarageYrBlt.apply(lambda x:0 if math.isnan(x) else 1)
```

We clearly see that house that garage has more price. Now how to impute this value.

```
In [263]: garage = all_data[['GarageCond', 'GarageQual', 'GarageYrBlt', 'GarageFinish', 'GarageType', 'GarageCars']]
```

```
In [264]: garage[garage.GarageCars.isnull()]
```

```
Out[264]:   GarageCond  GarageQual  GarageYrBlt  GarageFinish  GarageType  GarageCars
1116         NA           NA           0.0           NA         NA         NaN
```

	GarageArea
1116	NaN

We will make them zero. We did not do it earlier because there were some cases when garageYrBlk was Null but Garage Cars were not null. Probably they referred to detached garage type. Lets have a look

```
In [265]: ind = garage.GarageYrBlk==0
```

```
In [266]: (garage.GarageCars[ind] !=0).sum()
```

```
Out[266]: 2
```

```
In [267]: (garage.GarageArea[ind] !=0).sum()
```

```
Out[267]: 2
```

Okay so these are just two cases, we will make them all zero.

```
In [268]: all_data.loc[all_data.GarageCars.isnull(), ('GarageCars', 'GarageArea')]=0
```

```
In [269]: missing = all_data.isnull().sum()
```

```
missing = missing[missing > 0]
```

```
missing.sort_values(inplace=True)
```

```
missing.plot.bar()
```

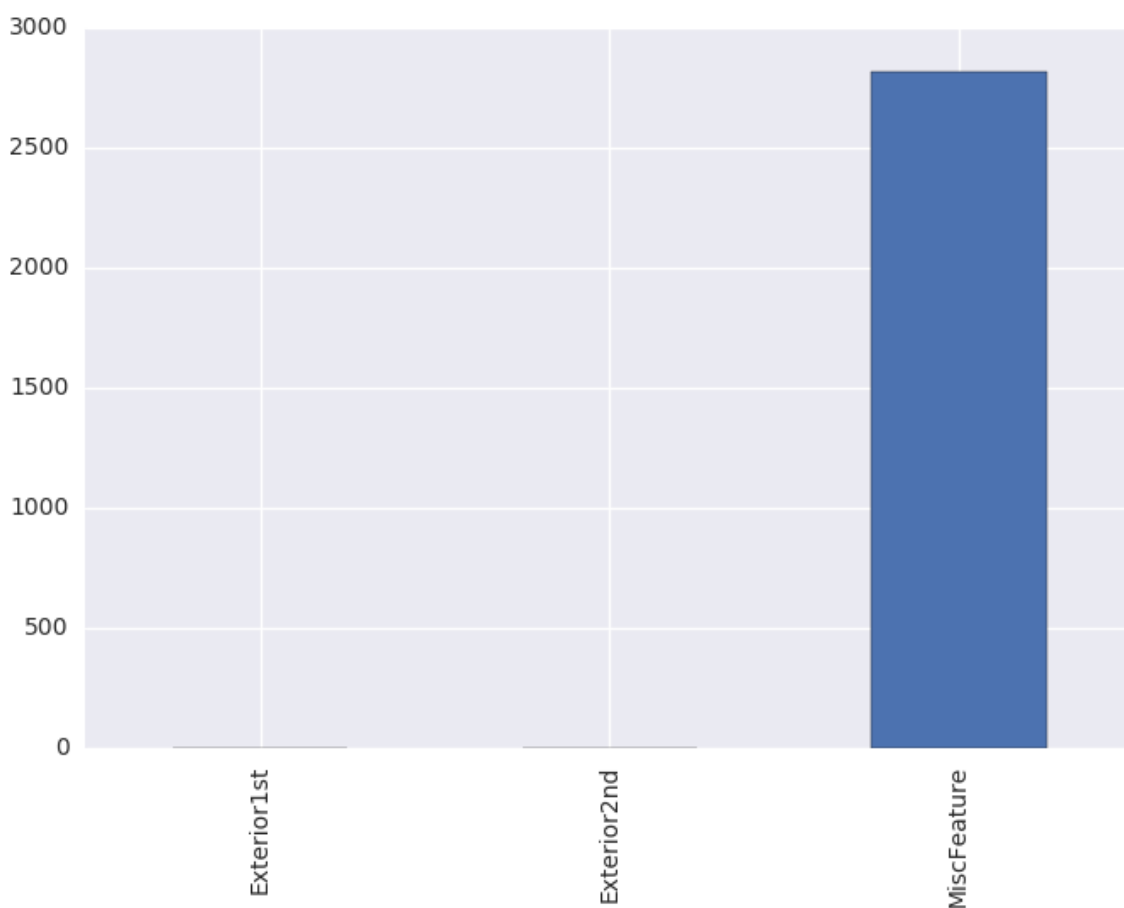
```
missing
```

```
Out[269]: Exterior1st      1
```

```
Exterior2nd      1
```

```
MiscFeature    2814
```

```
dtype: int64
```



Cool, these are the feature we had decide to use dummy variables for (along with some other feature). We are done, let's check performance.

```
In [270]: all_data = pd.get_dummies(all_data)
          print (all_data.shape)
          X_train = all_data[:trainDF.shape[0]] #We are using row selector operator
          X_test = all_data[trainDF.shape[0]:]
          y = trainDF.SalePrice
          y = log(y)
          cv_score = my_cv(LinearRegression(), X_train, y, 5)
          print ("rmse using least square is = ", cv_score)
```

(2919, 263)

rmse using least square is = 0.0131166595628

Appendix C

Codes

C.1 Gradient Descent

```
def gradientDescent(x, y, theta, alpha, m, numIterations):
    xTrans = x.transpose()
    Jcost = []
    for i in range(0, numIterations):
        hypothesis = np.dot(x, theta)
        loss = hypothesis - y
        cost = np.sum(loss ** 2) / (2 * m)
        Jcost.append(cost)
        #print("Iteration %d | Cost: %f" % (i, cost))
        gradient = np.dot(xTrans, loss) / m
        theta = theta - alpha * gradient
    return Jcost
```

C.2 Matrix Approach for solving Ridge Regression

```
def temp_train_score(Xtr, Xte, Ytr, Yte, alpha):  
    I = np.identity(Xtr.shape[1])  
    temp0 = Xtr.T.dot(Xtr) + alpha*I  
    temp1 = linalg.inv(temp0)  
    parameter = temp1.dot(Xtr.T).dot(Ytr)  
    Ypred = Xte.dot(parameter)  
    return direct_rmse_error(Yte, Ypred)  
  
def temp_cv(xin, yin, n_splits,alpha):  
    kf = KFold(n_splits=n_splits)  
    kf.get_n_splits(X_train)  
    ary = []  
    for train_index, test_index in kf.split(xin):  
        Xtr, Xte = xin[train_index], xin[test_index]  
        Ytr, Yte = yin[train_index], yin[test_index]  
        ans = temp_train_score(Xtr, Xte, Ytr, Yte, alpha)  
        ary.append(ans)  
    return np.mean(ary)  
  
alphas = [0.05, 0.1, 0.3, 0.5, 0.8, 1, 3, 5, 8, 9, 10, 15, 30, 50]  
cv_ridge = []  
cv_ridge = [temp_cv(np_X_train, np_y, 5, alpha)  
             for alpha in alphas]  
cv_ridge = pd.Series(cv_ridge, index = alphas)  
alpha = cv_ridge.idxmin()  
print ("value of alpha is = ",alpha)
```

```
print ("rmse using matrix solution is = ", cv_ridge[alpha])  
cv_ridge.plot(title = "Ridge Regression Lambda")  
plt.xlabel("Lambda")  
plt.ylabel("rmse")
```

C.3 Principle Component Regression

```
def my_cv_matrix_reduced(xin, yin, n_splits, features):  
    kf = KFold(n_splits=n_splits)  
    kf.get_n_splits(X_train)  
    ary = []  
    for train_index, test_index in kf.split(xin):  
        Xtr, Xte = xin[train_index], xin[test_index]  
        Ytr, Yte = yin[train_index], yin[test_index]  
        ans = get_accuracy_matrix_reduced(Xtr, Xte, Ytr, Yte, features)  
        ary.append(ans)  
    return np.mean(ary)  
  
def get_accuracy_matrix_reduced(Xtr, Xte, Ytr, Yte, features):  
    Ureduced = U[:, 0:features]  
    #process xtr  
    Xtr = Xtr.dot(Ureduced)  
    i = [1 for j in range(Xtr.shape[0])]   
    i = np.array(i)  
    i = np.expand_dims(i, axis=1)  
    Xtr = np.hstack((i, Xtr))  
  
    #process xte  
    Xte = Xte.dot(Ureduced)  
    i = [1 for j in range(Xte.shape[0])]   
    i = np.array(i)  
    i = np.expand_dims(i, axis=1)
```

```
Xte = np.hstack((i, Xte))

temp0 = Xtr.T.dot(Xtr)
temp1 = linalg.inv(temp0)
parameter = temp1.dot(Xtr.T).dot(Ytr)

Ypred = Xte.dot(parameter)
return direct_rmse_error(Yte, Ypred)
```

C.4 Custom Distribution in pymc - log logistic

C.4.1 Writing Custom Distribution

```
class MyLogLogit(pm.Continuous):
    def __init__(self, a, b, *args, **kwargs):
        super(MyLogLogit, self).__init__(*args, **kwargs)
        self.a = tt.as_tensor_variable(a)
        self.b = tt.as_tensor_variable(b)
        self.median = tt.as_tensor_variable(a)
    def logp(self, x):
        a, b = self.a, self.b
        one = tt.log(b/a)
        two = (b - 1)*tt.log(x/a)
        three = 1. + (x/a)**b
        four = -2.*tt.log(three)
        ans = one + two + four
        return ans
```

C.4.2 Predictive Model

```
class BayesianLogLogitRegression:
    def __init__(self):
        self.parameter = [] # Regression Coefficient
        self.alpha = None # Intercept
    def fit(self, XX, YY):
        XX = XX.values
        YY = YY.values
        with pm.Model() as reg_model:
```

```

alpha = pm.Normal('alpha', mu = 12.0, sd = 15.0)

beta = pm.Normal('beta', mu = 0.0, sd = 10.0, shape = XX.shape[1])

sigma = pm.HalfNormal('sigma', sd=10.0)

ll = [beta[q]*XX[:,q] for q in range(0, XX.shape[1])]

mu = alpha + sum(ll)

y_obs = MyLogLogit('y_obs', a=mu, b=sigma, observed=YY)

step = pm.Metropolis()

trace = pm.sample(20000, step)

trace = trace[2500:]

b=np.array(trace['beta'])

betaDF = pd.DataFrame(data=b, columns = all_data.columns)

alpha_parameter = trace[alpha].mean()

self.alpha = alpha_parameter

parameters = betaDF.mean().values

self.parameter = parameters

def predict(self, XX):

    XX = XX.values

    YY = XX.dot(self.parameter) + self.alpha

    return YY

```


Bibliography

- [1] Christopher M. Bishop Pattern Recognition and Machine Learning New York:Springer, 2006.
- [2] Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical LearningNew York:Springer, 2001
- [3] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani An Introduction to Statistical Learning with Applications in RNew York:Springer, 2013
- [4] scikit-learn : machine learning library in python. <http://scikit-learn.org/>
- [5] John Mandel. "Use of Single Value Decomposition in Regression Analysis"
The American Statistician 36(1982) : 15-25
- [6] Box, George E. P.and Cox, D. R. "An analysis of transformations"
Journal of the Royal Statistical Society 26(1964) : 211-252
- [7] N. Le Roux, M. Schmidt, and F. Bach. "A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets"
Advances in Neural Information Processing Systems 25(2012) : 2663-2671
- [8] Hujia Yu and Jaifu Wu. "Real Estate Price Prediction with Regression and Classification" CS 229 Autumn 2016 Project Final Report http://cs229.stanford.edu/proj2016/report/WuYu_HousingPrice_report.pdf

- [9] Matthew D. Hoffman and Andrew Gelman. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo" Journal of Machine Learning Research 15(2014) : 1351-1381

Checklist of items for the Final Dissertation Report
This checklist is to be attached as the last page of the report.

This checklist is to be duly completed, verified and signed by the student.

1.	Is the final report neatly formatted with all the elements required for a technical Report?	Yes / No <input checked="" type="checkbox"/> Yes
2.	Is the Cover page in proper format as given in Annexure A?	Yes / No <input checked="" type="checkbox"/> Yes
3.	Is the Title page (Inner cover page) in proper format?	Yes / No <input checked="" type="checkbox"/> Yes
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes / No <input checked="" type="checkbox"/> Yes
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes / No <input checked="" type="checkbox"/> Yes
6.	Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title	Yes / No <input checked="" type="checkbox"/> Yes
7.	Have you included the List of abbreviations / Acronyms?	Yes / No <input checked="" type="checkbox"/> Yes
8.	Does the Report contain a summary of the literature survey?	Yes / No <input checked="" type="checkbox"/> Yes
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes / No <input checked="" type="checkbox"/> Yes <input checked="" type="checkbox"/> Yes <input checked="" type="checkbox"/> Yes <input checked="" type="checkbox"/> Yes <input checked="" type="checkbox"/> Yes
10.	Is the conclusion of the Report based on discussion of the work?	Yes / No <input checked="" type="checkbox"/> Yes
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes / No <input checked="" type="checkbox"/> Yes <input checked="" type="checkbox"/> Yes
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes / No <input checked="" type="checkbox"/> Yes

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Place: Bangalore

Date: 24/03/2017


Signature of the Student

Name: Archit Vora

ID No.: 2015HT12480

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK-INTEGRATED LEARNING PROGRAMMES DIVISION
Second Semester 2016-2017

BITS ZG628T : Dissertation EC-3 Pre-Final Evaluation Sheet

Upload Softcopy of Final Dissertation Report, Pre-Final Evaluation Sheet, and Final Presentation by March 27, 2017

ID No. : 2015HT12480
NAME OF THE STUDENT : Archit Vora
EMAIL ADDRESS : architvora92@gmail.com
NAME OF THE SUPERVISOR : Anirban Basu

DISSERTATION TITLE : Predicting House Price on Ames Dataset

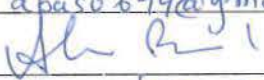
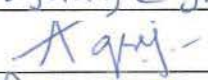
Dissertation Final Evaluation (Please put a tick (✓) mark in the appropriate box)

S No.	Evaluation Component	Excellent	Good	Fair	Poor
1.	Final Dissertation Report		✓		
2.	Final Seminar and Viva-Voce	✓			

S.No.	Evaluation Criteria	Excellent	Good	Fair	Poor
1	Technical/Professional Competence	✓			
2	Work Progress and Achievements	✓			
3	Documentation and expression		✓		
4	Initiative and Originality	✓			
5	Research & Innovation	✓			
6	Relevance to the work environment	✓			

Please **ENCIRCLE** the Recommended Final Grade: Excellent / Good / Fair / Poor

Remarks of the Supervisor: Regular progress of analysis and reports has culminated into an effective approach to solving the problem.

	Supervisor	Additional Examiner
Name	Anirban Basu	Agraj Mangal
Qualification	B.E Computer Science & Eng	B.S. M.C.A.
Designation		Computer Scientist
Employing Organization & Location	Adobe, Bangalore	Adobe, Bangalore
Phone Number	8510815735	9711367226
Mobile Number	8510815735	9711367226
Email Address	abasu69@gmail.com	agraj.mng@gmail.com
Signature		
Place & Date	Bangalore, 24th March, 2017	Bangalore, 24/3/2017