Introduction to Data Science Tools & Techniques

Dr. Kashif Zafar

PROJECT REPORT

Discovering Optimal Models for Real Estate Prices Prediction using Linear and Non-Linear Regression Techniques

Samama Imtiaz Butt
Dept. of Computer Science
National University of Computer & Emerging Sciences
Lahore, Pakistan
l181882@lhr.nu.edu.pk

Areeb Waseem
Dept. of Computer Science
National University of Computer & Emerging Sciences
Lahore, Pakistan
l181823@lhr.nu.edu.pk

Rehman Shahid
Dept. of Computer Science
National University of Computer & Emerging Sciences
Lahore, Pakistan
l181890@lhr.nu.edu.pk

***Abstract*— this paper is intended to explore a comprehensive study of the important factors that play a vital role in determining the Real Estate prices. We know that real estate prices follow specific trends that depends on various characteristics. This project aims at applying advanced regression techniques to predict real estate prices given a set of features. The dataset is a combination of several features such as interior, exterior, area, location, heating, number and condition of rooms etc. Next, we will then train our models using the machine learning approaches that have never applied before depending on these features and the respective sale prices. In the end we will run our models to predict real estate prices and test the accuracy of our results.**

Keywords— comprehensive, trends, regression, features

I. INTRODUCTION

Buying a real estate property is a nerve-necking problem for the individuals involved. Buyers are usually not aware of the deterministic features that impacts the real estate prices. Therefore, a prediction model is required to overcome the information gap and improve the efficiency of real estate market. The goal of this project is to predict real estate prices by finding the set of features and conditions that influence the sale prices. In order to accomplish our goal we must need to know first what actually the real estate is.

A. Real Estate & its types

Real Estate is defined as the property, land, buildings and air rights above land and underground rights below land. There are four common types of real estate's given below:

- **Residential Real Estate** includes both newly constructed and resale homes.
- **Industrial Real Estate** covers factories, warehouses etc.
- **Commercial Real Estate** includes shopping centers, malls, hotels and offices.
- **Land** involves vacant land, working farms and ranches.

We are solely focusing on the **Residential** real estate prices in this analysis. Real estate prices follow certain tendencies of better condition leading to better selling price and vice versa. We have studied various research papers and came to know about different popular optimization techniques and machine learning algorithms already used in predicting residential real estate prices such as Lasso Regression, Hedonic Model, Artificial Neural Network, Linear and Multivariate Regression etc. In this experiment, we have implemented ANN, Ensembled Linear Regressor, raw Multivariate Linear Regressor, Kernel Ridge Regressor and Random Forest using a different approach which increases the accuracy drastically. In addition, we used several optimization techniques that involves raw Batch Gradient Descent and raw Stochastic Gradient Descent. We have chosen the dataset from Kaggle consisting of different characteristics of a house in the form of a csv file with a well-defined structure. We will first preprocess the dataset as per the standards/techniques discussed in the class. Then we will train our models that will evaluate the specific set of features of each house in accordance with its price. Lastly, we will run this model for predicting the prices and check the accuracy of the results.

II. PROBLEM STATEMENT

Most of the real estate properties are brought through agents. Occasionally, buyers directly buy from sellers since there are a lot of legal terminologies involved and they are unaware of them. Hence, real estate agents contributes as a communication medium between seller and buyer as well as they help in the legal aspects. Involving a middleman increases the cost of the property. In a result of which houses are overpriced and actual buyer should need to know the actual price of the property. Therefore, a prediction model is required to overcome the information gap and improve the efficiency of real estate market. Since we will be predicting price in this experiment which is a continuous variable so we will treat this as a regression problem.

III. LITERATURE REVIEW

Over the past year, a lot of studies have examined the relationship between real estate prices and its features. For instance, Azme Bin Khamis and Nur Khalidah Khalilah Binti Kamarudin[6] implemented Multilinear Regression and ANN on house prices prediction dataset. Kr´ol [1] investigated the relationship between the price of an apartment and its significant features based on the results of hedonic analysis in Poland. The work of [2] discussed which house features have negative or positive effects upon the value of the house in Turkey. Kryvobokov and Wilhelmsson [3] derived the weights of the relative importance of location features that influence the market values of

apartments in Donetsk, Ukraine. Ottensmann et al. [4] compared measures of location using both distances and travel time, to the CBD, and to multiple employment centers to understand how residence location relative to employment location affects house price in Indianapolis, Indiana, USA. Ozalp and Akinci [30] determined the housing and environmental features that were effective on residential real estate sale prices in Artvin, Turkey.

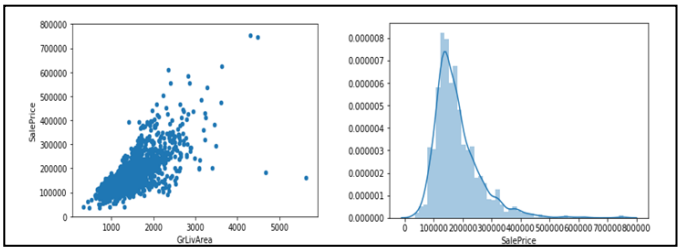
IV. PROPOSED METHODOLOGY

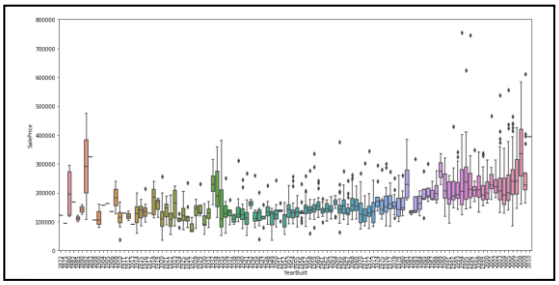
A. *Analyzing the dataset*

The dataset consists of several features of each house and the respective prices. Having a bird's eye view of the data shows that the data is strongly and linearly correlated which forms the basis for our machine learning models.

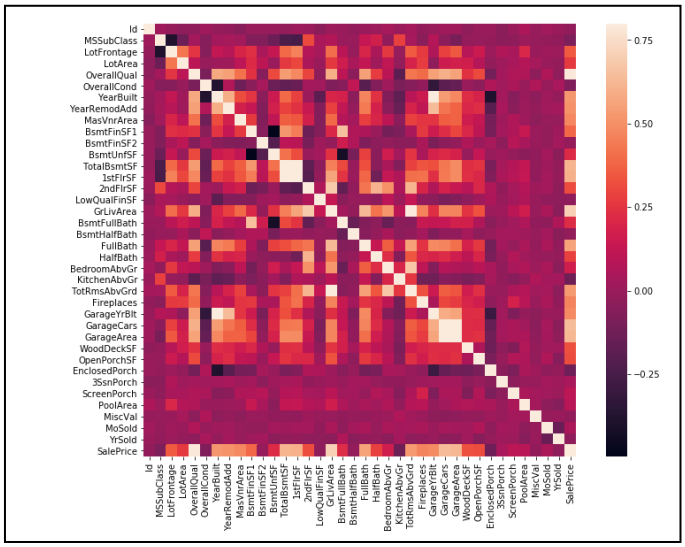
The sales price shows the following trends:

- Deviate from the normal distribution
- Have appreciable positive skewness
- Show peakedness
- Sales Price and GrLivArea have a linear relationship
- Sale prices increase per year





1) *Taking Correlation and using heatmap*



- OverallQual, GrLivArea and TotalBsmtSF are strongly correlated with SalePrice.
- GarageCars and GarageArea are also some of the most strongly correlated variables.
- TotalBsmtSF and 1stFloor are also strongly correlated variables.
- There are two red colored squares:
- The first one refers to TotalBsmtSF and 1stflrSF variables and the second one refers to the GarageX variables.
- The correlation is so strong that it can indicate a situation of multicollinearity.
- TotRmsAbvGrd and GrLivArea are also correlated.
- TotalBsmtSF and 1st Floor also show co-relationship.

B. *Data Preprocessing*

First we have to check for missing values. The following results were found after summing the missing values for each column.

	Train	Test
Alley	1369	1352.0
BsmtCond	37	45.0
BsmtExposure	38	44.0
BsmtFinSF1	0	1.0
BsmtFinSF2	0	1.0
BsmtFinType1	37	42.0
BsmtFinType2	38	42.0
BsmtFullBath	0	2.0
BsmtHalfBath	0	2.0
BsmtQual	37	44.0
BsmtUnfSF	0	1.0
Electrical	1	0.0
Exterior1st	0	1.0
Exterior2nd	0	1.0
Fence	1179	1169.0
FireplaceQu	690	730.0
Functional	0	2.0
GarageArea	0	1.0
GarageCars	0	1.0
GarageCond	81	78.0
GarageFinish	81	78.0
GarageQual	81	78.0
GarageType	81	76.0
GarageYrBlt	81	78.0
KitchenQual	0	1.0
LotFrontage	259	227.0
MSZoning	0	4.0
MasVnrArea	8	15.0
MasVnrType	8	16.0
MiscFeature	1406	1408.0
PoolQC	1453	1456.0
SaleType	0	1.0
TotalBsmtSF	0	1.0
Utilities	0	2.0

Looking at the results it is quite evident that there are features which have a large set of missing values and in turn gives us the motivation to preprocess the data and standardize it.

After analyzing the data we decided to get rid of features which had more than half of the missing information or didn't correlate to sales price.

The following features were dropped:

Utilities, RoofMatl, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, Heating, LowQualFinSF, BsmtFullBath, BsmtHalfBath, Functional, GarageYrBlt, GarageArea, GarageCond, WoodDeckSF, OpenPorchSF,

EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, PoolQC, Fence, MiscFeature, MiscVal.

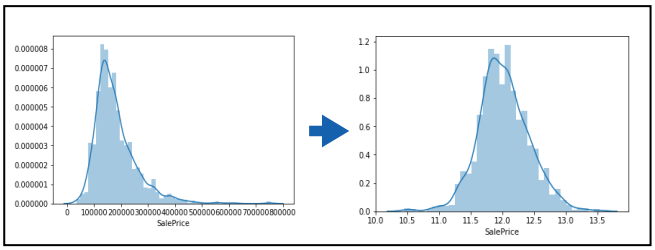
The next task was to fill Nan values and convert types of features.

TABLE I. FEATURE CONVERSION

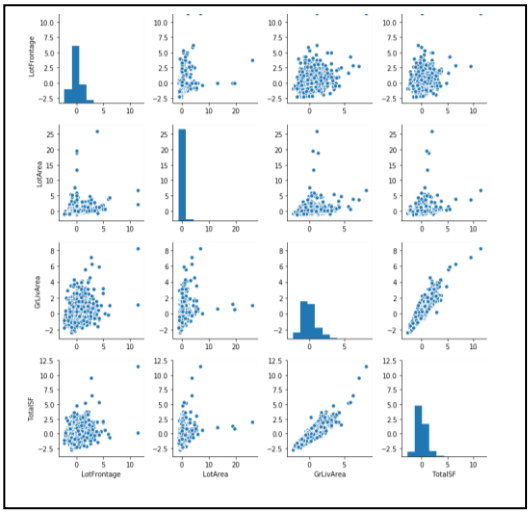
Feature Name	*Conversion*
MsSubClass	toStr
MsZoning	Fill Na with most common feature (mode)
LotFrontage	Fill Na with Mean
Alley	Fill Na with No Access
OverAllCond	toStr
MasVnrType	Mode
Bsmt	Fill Na with No Basement
TotalBsmtSF	Fill Na 0
Electrical	Mode
KitchenAbvGr	toStr
KitchenQual	Mode
FireplaceQu	Fill Na with NoFp
Garage	Fill Na with NoGRG
SaleType	Mode
YrSold	toStr
MoSold	toStr

The Sale price was skewed right and to make it symmetric we took log transform.

The resultant sales price looks much better and symmetric around a sale price of 12.0.



Now we standardized the numeric data by taking the mean of each feature, subtracting it from each entry and dividing it by standard deviation. The resultant plot is shown and it can be clearly seen that all of the values are now densified in the range 0-10, which will form the basis for our machine learning models.



Now that the data was standardized, we decided to split the data in training and test sets into 90% and 10% respectively using random selection

TABLE II. SPLITTING DATASET

Data	*Matrix Rows*	*Matrix Columns*
Training Data	1314	262
Test Data	146	262

C. *Machine Learning Models*

After analyzing and preprocessing it was quite clear that we needed to implement and compare multiple regression models. Following are the regression models we used in this experiment:

1) *Regression through Normal Equation*

The concept behind normal equation is that given a labeled dataset we can find a weight matrix that can eventually be applied on test data for regression purposes. For example if we have a

Matrix X containing an individual elements with each element normalized to a size of s resulting in an nXs matrix, we can multiply it with a weight matrix W which will give us the resultant matrix

Y.

As X and Y are already available in the training data we can find the weight matrix W by using

The equation:

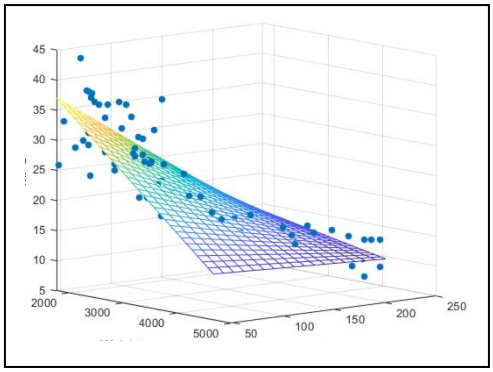
$$Y = W.X$$

We can find W through the formula:

$$W = (XT.X)\text{-}1.XT.Y$$

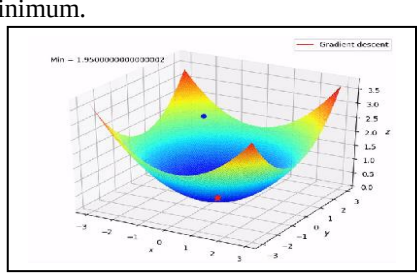
Where XT is the transpose of the matrix X.

Once we have the matrix W we can take a test object X and multiplying them gives us the Resultant matrix Y.



2) *Gradient Descent*

Gradient Descent is the most popular optimizing algorithm in machine learning and can be easily combined with any other algorithm. It is used to train a model by iteratively updating a function to reduce the cost function to its local minimum.



Cost function is used to measure the performance of a machine learning model. It calculates the loss or error between the actual and expected value and represents it in a form of single real number. It returns two types of values either it returns the smallest number i.e. error, loss, cost etc. or returns the maximum number i.e. reward. There are many types of cost function but I have used mean squared error in my algorithm which can be written as following:

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y_i]^2$$

Predicted Value — True Value

By taking the derivative of this cost function, we can compute the gradient of our linear model.

$$\begin{aligned}\frac{\partial}{\partial\Theta}J_\Theta &= \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y) \\ &= \frac{1}{m}(h_\Theta(x_i) - y)x_i\end{aligned}$$

"A gradient measures how much the output of a function changes if you change the inputs a little bit."— Lex Fridman (MIT)

Amount of step that gradient should take into the direction of local minimum is called learning rate. If this step is too large then we will not be able to find the minimum. If we take it to small then it will going to increase the number of iterations to find the minimum. So, we can say that it plays an important role in computing gradient descent.

$$\Theta_j = \Theta_j - \alpha\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)$$

Learning Rate

Therefore,

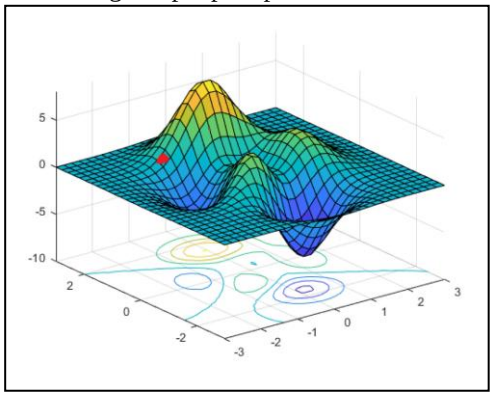
$$\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]$$

a) ***Batch Gradient Descent***

Uses whole training data per epoch.

b) ***Stochastic Gradient Descent***

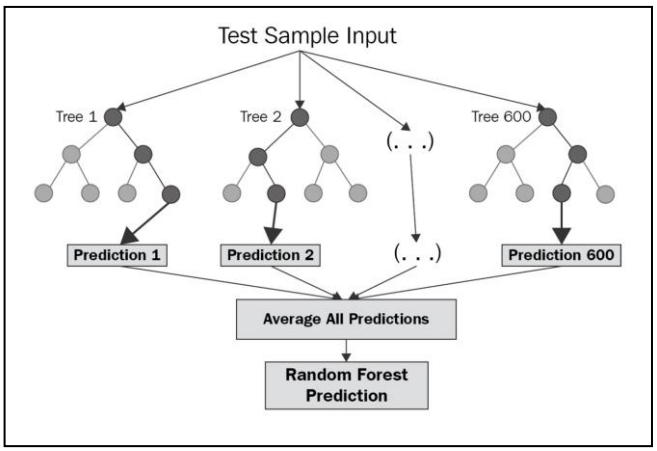
Uses one training sample per epoch.



3) ***Random Forest***

Random forests can be used for regression analysis and are in fact called Regression Forests. They are an ensemble of different regression trees and are used for nonlinear multiple regression. Each leaf contains a distribution for the continuous output variable/s.

The model was applied to fit to our training set and then the accuracy was measured through our test data.



4) ***Arificial Neural Network***

Artificial Neural Network or ANN is a clone of human brain learning process. It follows the neuron information processing where a neuron accepts signals from dendrites and passes electoral signals down to the cell body.

In 1957, Frank Rosenblatt started investigating the Artificial Neural Network at the Cornell Aeronautical Laboratory which he later called Perceptron. These Perceptron's were then transformed into Multilayer Perceptron's which we call today Artificial Neural Network. It is a supervised learning algorithm which means we will provide training data to it for learning process. Initially, ANN will just do random predictions, then it compares the predictions with the target output and compute error. It computes error using a cost function which we discussed in a Gradient Descent section that is actually the difference between the target and actual value. Training a neural network is basically minimizing the cost function. ANN runs in two phases i.e. Feed Forward and Backpropagation. Let's discuss both of them one by one:

a) ***Feed Forward***

In Feed Forward phase of ANN, predictions are made based on the values we provide to the input layer and the weights. Weights are just the strings that we have to adjust in order to predict the correct output. We perform the below steps in this phase:

- Calculate the dot product between inputs and weights and add a bias.
- Pass the summation of dot product through an activation function.

b) ***Backpropagation***

In the beginning, ANN has made the random predictions which are obviously incorrect. Next, we have to compare these predictions with the target output and adjust the weights and bias in such a way that our actual predicted

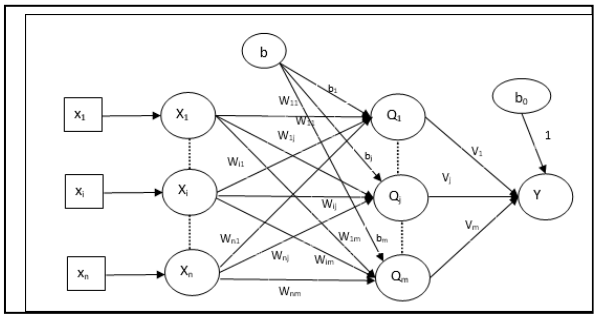
value comes closer to the target value. It consist of the following steps:

- Calculate the cost using some cost function
- Minimizing the cost using Gradient Descent

There are many types of ANN now available but three of them are really popular i.e.

- Recurrent Neural Network
- Convolutional Neural Network
- Reinforcement Learning

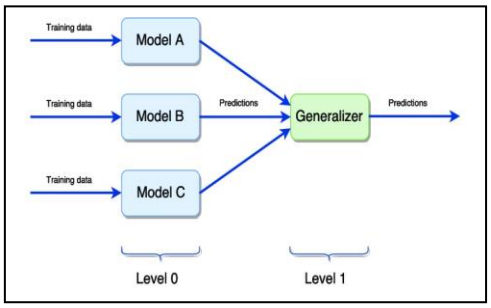
Following diagram explains how an ANN works:



*5) **Ensembed Regressor***

The idea is to combine different machine learning models in order to improve the accuracy. It can be divided into three phase's i.e.

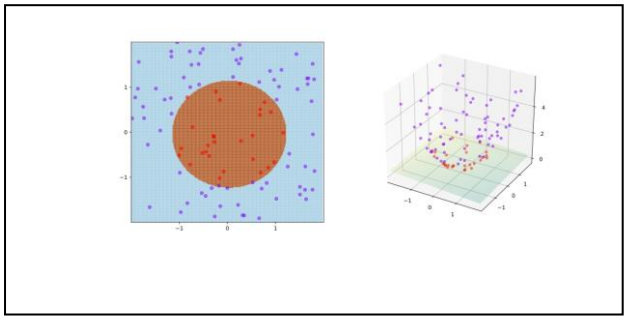
- Generation phase where the set of candidate models are induced.
- Pruning phase to select the subset of those models.
- Integration phase where we combine the output of different models to generate a prediction.



*6) **Kernel Ridge Regressor***

Kernel Ridge Regression combines the ridge regressor with kernel trick to learn a nonlinear function by mapping our data to a higher order domain.

The kernel trick helps us avoid the expensive tasks of mapping our data by taking inner product and raising it to the order of the domain.



V. EXPERIMENTAL SETUP

Dataset has been preprocessed. Next, we need to select and train our different machine learning models. After selecting the models we have to choose suitable parameters for each model.

A. Dataset

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset. This data set describing the sale of individual residential property in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. We have already reduced the features in the previous section now we will just set the parameters of the machine learning models and apply it on the dataset.

B. Parameter Settings

*1) **Regression through Normal Equation***

The key idea here was to set the matrix X as the standardized features of our training example and vector Y was set to the respective sale prices of those houses.

The resultant weight vector W gave us the weights that will ultimately be used to predict house sale prices.

*2) **Gradient Descent***

The key idea here was to set the matrix X as the standardized features of our training example and vector Y was set to the respective sale prices of those houses.

The resultant weights were learned by first initializing them to 0.1 and then applying gradient descent with a learning rate of 0.0000000001 and 0.0000001 for batch and stochastic gradient descent respectively.

*3) **Random Forest***

We have used 100 trees in our algorithm and random is 42.

*4) **ANN***

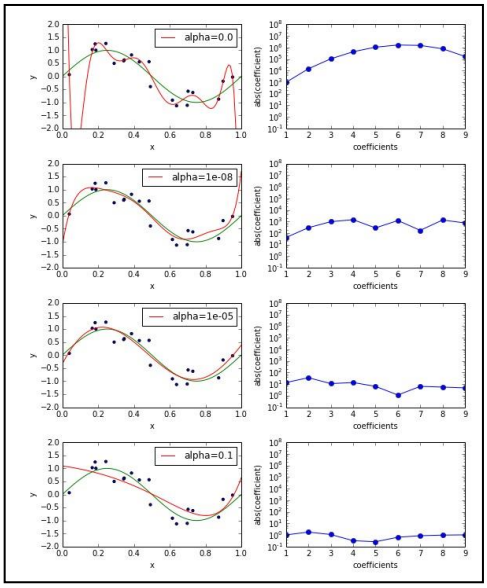
We have used 32 hidden layers with sequence of 15 10 5 2.

*5) **Ensembled Regressor***

Combined three machine learning models and used there prediction average which are Normal Equation, Batch Gradient Descent and Stochastic Gradient.

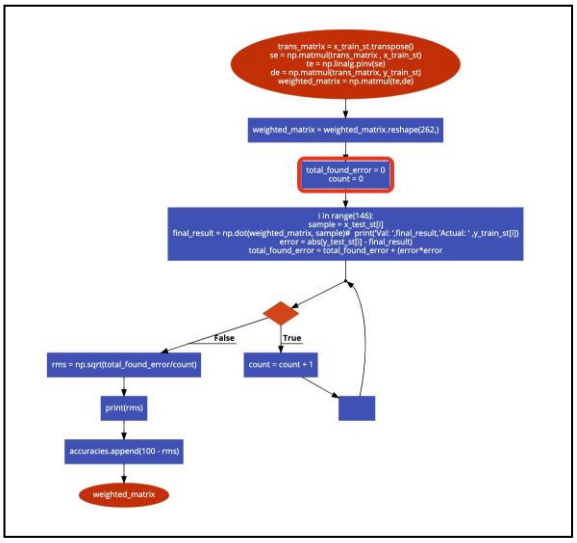
*6) **Kernel Ridge Regressor***

Used regularization parameter of **1.0** in our algorithm. Following diagram shows the impact of setting regularization parameter:



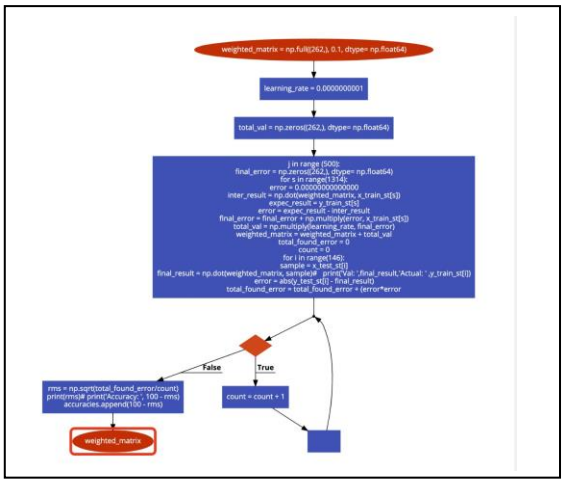
C. *Experiment 1 (Regression through Normal Equation)*

Following flow chart shows the implementation of multiple regression through Normal Equation:



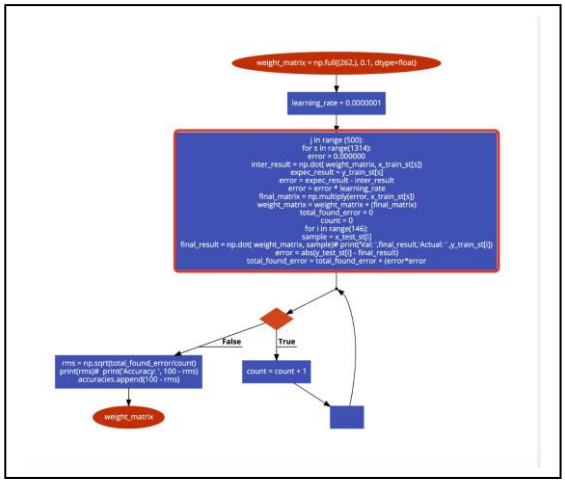
D. *Experiment 2 (Batch Gradient Descent)*

Following flow chart expresses the implementation of Batch Gradient Descent:



E. *Experiment 3 (Stochastic Gradient Descent)*

Following flow chart expresses the implementation of Stochastic Gradient Descent:



F. *Experiment 4 (ANN)*

We used the library of Scikit Learn to implement the ANN. Below is the pseudo code of the algorithm:

PSEUDO CODE:

```
ann_regressor = MLPRegressor(hidden_layer_sizes=(15,10,5,2),
random_state = 1,learning_rate_init= 0.01)
   ann_regressor.fit(x_train_st, y_train_st)
   predictions = ann_regressor.predict(x_test_st)

   errors = abs(predictions - y_test_st)
```

G. *Experiment 5 (Ensembled Regressor)*

We have combined the outputs of Normal Equation, Batch Gradient Descent and Stochastic Gradient Descent to predict the real estate prices. Following is pseudo code of the algorithm:

PSEUDO CODE:

```
mat1 =  get_normal_regressor()
   mat2 =  get_gradient_regressor()
   mat3 =  get_stochastic_gradient_regressor()
final_result = float((float(np.dot( mat1, sample)) +
float(np.dot( mat2, sample)) + float(np.dot( mat3,
sample)))/3)
```

H. *Experiment 6 (Kernel Ridge Regressor)*

We used the library of Scikit Learn to implement the KRR. Below is the pseudo code of the algorithm:

PSEUDO CODE:

```
kernel_ridge_regressor = KernelRidge(alpha=1.0)

   kernel_ridge_regressor.fit(x_train_st, y_train_st)

   predictions = kernel_ridge_regressor.predict(x_test_st)
```

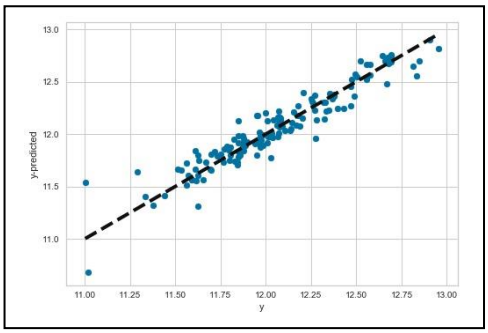
VI. Results and Analysis

A. *Prediction Error Plots*

Below are the prediction error plots of all the models we have implemented:

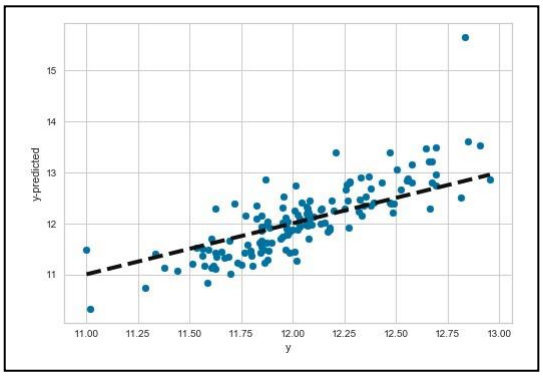
1) *Normal Equation*

The results are reasonable. Optimal Linear Decision Boundary learned but cannot be compared to non-linear model. RMSE value is 0.1206.



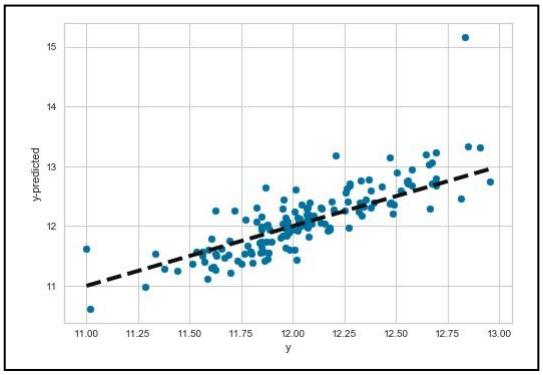
2) ***Batch Gradient Descent***

The model performed worst as compared to others. RMSE value was 0.4502. Batch processes are more prone to settling at local minima rather than global minima.



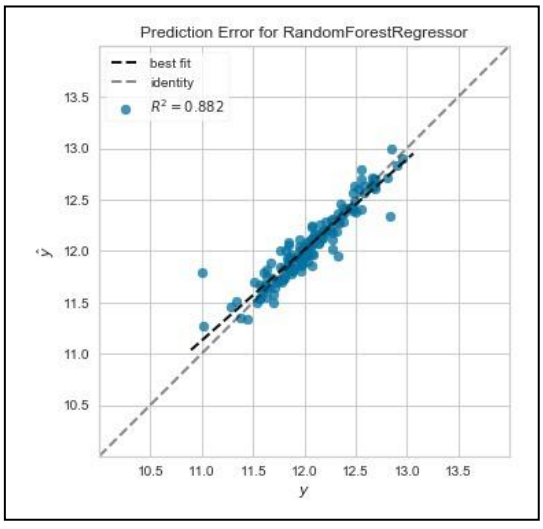
3) ***Stochastic Gradient Descent***

The model performed better than Batch Gradient Descent. RMSE value is 0.3451. Stochastic processes iteratively modify weights giving them a high chance of finding global minima.



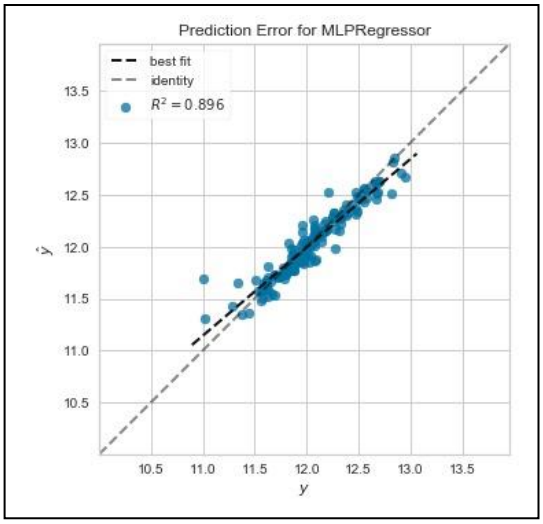
4) ***Random Forest***

Random Forest combines multiple decision trees to produce an accurate averaged non-linearly regressed output. There is a sudden jump in accuracy as we shift to non-linear models. RMSE value is 0.1256.



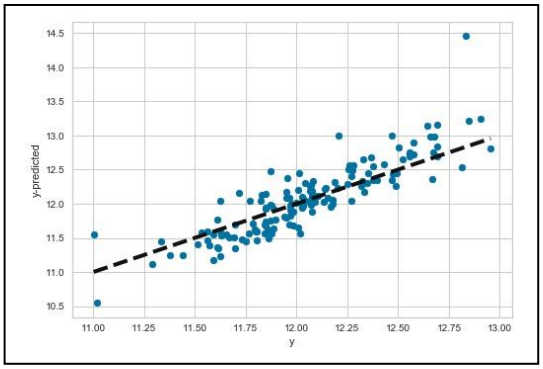
5) ***ANN***

A combination of multi-layered perceptron's including hidden layers enable the model to learn a non-linear function. RMSE value is 0.1195. Better than all of the models except Kernel Ridge Regression.



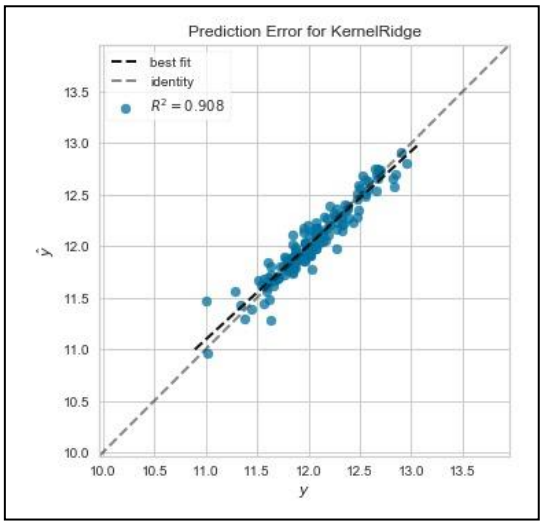
6) ***Ensembled Regressor***

Ensembling shows that different models can be combined to produce better results. The results were in between of 3 of our linear models. RMSE value is 0.2695.



7) ***Kernel Ridge Regressor***

The model greatly enhances the optimal linear multi-variate regression by mapping the data points to a higher order domain and learning a linear function which eventually results in a non-linear function in a lower order domain without any increased computation expense. RMSE value is 0.1126.



CONCLUSION

It was seen that non linear methods like ANN, kernel ridge and random forest perform better than linear models like Multivariate and gradient descent regression.

The results back our conclusion as the rms error fell to 0.1126 while batch gradient descent had 0.45 which is a big difference. The reason is that kernel ridge gradient enhances the normal ridge regression by mapping data points to higher order and a linear hyperplane in a higher domain is actually a nonlinear function in lower order domain.

We can conclude that for standardized variables with reasonable sized datasets, kernel ridge performs the best. In cases where datasets are very large the Kernel Perceptron will inevitably be the best possible solution as the space complexity of Kernel Ridge Regressor will be very high.

REFERENCES

[1] A. Kr´ol, “Application of hedonic methods in modelling real estateprices in poland,” inData Science, Learning by Latent Structures, andKnowledge Discovery, 2013, pp. 501–511.

[2] R. YAYAR and D. DEM ˙IR, “Hedonic estimation of housing marketprices in turkey,”Erciyes ¨Universitesi ˙Iktisadi ve ˙Idari Bilimler Fak ¨ultesiDergisi, no. 43, pp. 67–82, 2014.

[3] M. Kryvobokov and M. Wilhelmsson, “Analysing location attributes witha hedonic model for apartment prices in donetsk, ukraine,”InternationalJournal of Strategic Property Management, vol. 11, no. 3, pp. 157–178,2007.

[4] J. R. Ottensmann, S. Payton, and J. Man, “Urban location and housingprices within a hedonic model,”Journal of Regional Analysis&Policy,vol. 38, no. 1, pp. 19–35, 2008.

[5] J. R. Ottensmann, S. Payton, and J. Man, “Urban location and housingprices within a hedonic model,”Journal of Regional Analysis&Policy,vol. 38, no. 1, pp. 19–35, 2008.

[6] Azme Bin Khamis and Nur Khalidah Khalilah Binti Kamarudin, “Comparative Study On Estimate House Price Using Statistical And Neural Network Model”, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 3, ISSUE 12, December 2014.