House Price Prediction model with different Machine Learning Algorithms

Sai Ajith Teki
Department of computer science
Blekinge Tekniska Högskola
Karlskrona, Sweden
sate18@student.bth.se

***Abstract*—Over the past few years Machine learning plays a vital role in product recommendation,image recognition and medical diagnosis. Now a days its being helpful in enhancement of security,medical facilities and research. In this paper, we were going to discuss about different machine leaning models through which accuracy of house price prediction is determined. For the selection of the best model, we would compare the results of calculated accuracy of all the models. From this work,many real estate schemes would be get benefited by choosing the accurate algorithm as research method to predict house prices based on their accuracy. In this point, a better prediction model would be built to support a sales broker for valuation of the houses in various localities.**

***Index Terms*—Machine learning algorithms, Linear Regression, Lasso regression, Random Forest, Gradient Boosting Regression ,House price prediction.**

I. Introduction

In this globalization era, many people across the world are interested in Investment business. Although it involves a huge estimated risk through out the process, many people are interested in investing their money and time in commodities like Oil and Gas, Gold, Stocks and Properties. Since 2011, there is a huge boom in the property investment business as it always makes its investors profitable both in short and long terms.Not only for investors but also for people who want to buy a house for themselves, house price prediction plays a vital role in it.

The main aim of this work is to develop a machine leaning model with different algorithms that produce optimal prediction results.Prediction of house prices tends to be an advantage for people who are in a plan to buy a house in future so that they can estimate the price range and make financial arrangements accordingly. It is also helpful to many property investors who tends to make their business in a particular location. There are several approaches to make this house price prediction works. In this study, we will get know the best algorithm which predicts the house prices more accurately when compared to other algorithms used in this process to build a prediction model. Later on this would be helpful to choose the best algorithm among them for future evaluations and sales purposes. Out of many machine learning algorithms, Regression algorithms were proved to be functional productively in the case of prediction involving continuous variables. So in this work, we have chose Linear regression,Lasso regression ,Random Forest regression and Gradient Boosting regression.

The dataset used for this work is "kc house data" from UCI Machine learning repository. The dataset consists of 21 attributes and 21614 instances.They were:
1)id
2)date
3)price
4)bedrooms
5)bathrooms
6)sqft$_l iving$
7)$sqft_lot$
8)$floors$
9)$waterfront$
10)$view$
11)$condition$
12)$grade$
13)$sqft_above$
14)$sqft_basement$
15)yr_built
16)$yr_renovated$
17)$zipcode$
18)lat
19)$long$
20)$sqft_living15$
21)$sqft_lot15$

II. RELATED WORK

Basically, price of a house is the quantitative representation of its features. Over the past few years, many studies have been done on house prices predictions based on their features [1] [2] Based upon extensive study of house price predictions, many machine learning algorithms were used to predict prices [2].Linear Regression, Random Forest Regression and SVM were some of them [1]

This research was made to determine the price of a house by using regression analysis and particle swarm optimization . [3]

Some variables can be identified and involved in the implementation by Lasso regression which positively affects the outcome as they also deal with multi collinearity. [4]

This study proves that random forest regression acts responsibly as the target variable is a continuous variable.when compared to linear regression, random forest regression gives out the prediction more accurately as it deals mostly with the trees that it constructs. [5]

This study reveals that gradient boosting regression takes part actively in enhancing the performance in terms of accuracy. [1] [5]

III. Implementation

A. *Visualization of the data*

The data can be understood in the aspect of its relationship with various attributes and their affects on the results. Correlation among the various attributes can also be understood by visualization of the data by the means of plot graphs. [1]

```
In [5]: # Load the data from dataset
In [6]: data=pd.read_csv("kc_house_data.csv")
In [7]: # sneakpeak into the data
In [8]: data.head()
```

Out[8]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bui
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180.0	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170.0	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770.0	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050.0	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680.0	0	198

5 rows × 21 columns

```
In [12]: data.describe()
```

Out[12]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.0
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.6
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.1
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.0
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.0
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.0
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068000e+04	2.000000	0.000000	0.000000	4.000000	8.0
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.0

```
In [14]: data['bedrooms'].value_counts().plot(kind='bar')
         plt.title('Number of Bedrooms')
         plt.xlabel('bedrooms')
         plt.ylabel('Number of Houses')
         plt.show()
         sns.despine()
```

```
In [15]: plt.figure(figsize=(8,8))
         sns.jointplot(x=data.lat.values, y=data.long.values, size=8)
         plt.xlabel('Latitude')
         plt.ylabel('Longitude')
         plt.show()
         sns.despine()
```

```
In [16]: plt.scatter(data.price,data.lat)
         plt.title('Price vs Latitude')
         plt.xlabel('price')
         plt.ylabel('Latitude')
         plt.show()
         sns.despine()
```

```
In [17]: plt.scatter(data.price,(data.sqft_living+data.sqft_basement))
         plt.title('Price vs Area')
         plt.xlabel('price')
         plt.ylabel('Area')
         plt.show()
         sns.despine
```

```
In [18]: plt.scatter(data.bedrooms,data.price)
         plt.title('Price vs Number of Bedrooms')
         plt.ylabel('price')
         plt.xlabel('bedrooms')
         plt.show()
         sns.despine()
```

```
In [19]: plt.scatter(data.zipcode,data.price)
         plt.title('Location vs Price')
         plt.xlabel('zipcode')
         plt.ylabel('price')
         plt.show()
         sns.despine()
```

```
In [20]: plt.scatter(data.waterfront,data.price)
         plt.title('Waterfront vs Price')
         plt.xlabel('waterfront')
         plt.ylabel('price')
         plt.show()
         sns.despine()
```

```
In [23]: plt.scatter(data.floors,data.price)
         plt.title('Floors vs Price')
         plt.xlabel('floors')
         plt.ylabel('price')
         plt.show()
         sns.despine()
```

```
In [24]: plt.scatter(data.condition,data.price)
         plt.title('Condition vs Price')
         plt.xlabel('condition')
         plt.ylabel('price')
         plt.show()
         sns.despine()
```

```
In [26]: # check for any empty values
In [27]: df.isnull().sum()
Out[27]: id               0
         date             0
         price            0
         bedrooms         0
         bathrooms        0
         sqft_living      0
         sqft_lot         0
         floors           0
         waterfront       0
         view             0
         condition        0
         grade            0
         sqft_above       2
         sqft_basement    0
         yr_built         0
         yr_renovated     0
         zipcode          0
         lat              0
         long             0
         sqft_living15    0
         sqft_lot15       0
         dtype: int64
```

IV. Methods

Many datasets has missing values,redundant values and also even infinity values. These sort of issues should be effectively taken care of in order to make a better prediction model.


```
In [26]: # check for any empty values
In [27]: df.isnull().sum()
Out[27]: id               0
         date             0
         price            0
         bedrooms         0
         bathrooms        0
         sqft_living      0
         sqft_lot         0
         floors           0
         waterfront       0
         view             0
         condition        0
         grade            0
         sqft_above       2
         sqft_basement    0
         yr_built         0
         yr_renovated     0
         zipcode          0
         lat              0
         long             0
         sqft_living15    0
         sqft_lot15       0
         dtype: int64
```


A. *Handling null values*

In this dataset, the attribute (sqft above) has 2 null values. These null values have to replaced with numerical values so that we can have promising results.

```
: data[:] = np.nan_to_num(data)
: data.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bui
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180.0	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170.0	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770.0	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050.0	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680.0	0	198

5 rows × 21 columns

B. *Algorithm selection*

Four algorithms have been selected to make models that predicts the house price.
Linear Regression is selected because of its supervised learning nature which estimates the real continuous value output. [1] Relationship between two continuous variables can be studied and summarized by a Linear regression method. [1] Least complex nature was its advantage when compared to other algorithms.It also produces reasonable accuracy in less time.
In this study,
a) X would be regarded as the independent variable.
b) Y would be regarded as the dependent variable.
It can understand the prices of houses and can predict the price of a house with particular features.

LASSO- Least Absolute Shrinkage and Selection Operator. For a dataset having vast set of features,Lasso Regression would be used to make regularization of routines. [4] Lasso regression is chosen because it can be used to estimate the regression coefficient and perform variable selection. [4] Lasso regression would be the best to evaluate relapse models for further access to information.
a) Over fitting can be improved vastly.
b) The main motto is to reduce the prediction error for a quantitative variable.

Random Forest algorithm is selected because when compared to traditional linear regression model, it can be better at capturing nonlinear relations between the attributes. [1]
Random forest algorithm is used for both classification and regression models which builds many decision trees from data samples and gets prediction from each of the trees by mean of voting.
It also has the advantage of dealing with large data sets and can overcome the problem of Over fitting [1]. It also gives acceptable accuracy in less time.

Gradient boosting Algorithm is a machine learning method which ensembles weakly progressed models into a powerful prediction model which has high accuracy [4]. These are mostly Decision trees in which outputs powerless classifiers would combine to form a superior model.
Gradient Boosting Algorithm is used in this work because it is

a tree based model which can make good possible splits with available data at any point [4]. The obtained variable is good at predicting house prices eventually.

```
lr.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
lasso=Lasso(alpha=20,max_iter=1e5)
lasso.fit(x_train,y_train)
```

```
rf=RandomForestRegressor(n_jobs=-1)
rf.fit(x_train,y_train)
```

```
gbr=ensemble.GradientBoostingRegressor(n_estimators=400,max_depth=5,min_samples_split=2,learning_rate=0.1,loss='ls')
```

```
gbr.fit(x_train,y_train)
```

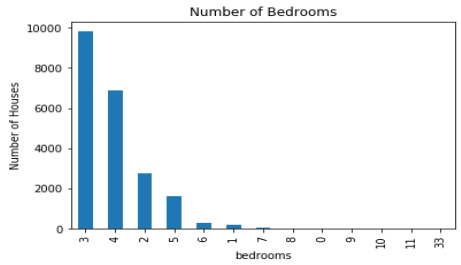
```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=400,
                          n_iter_no_change=None, presort='auto',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

C. *Training and Testing the dataset*

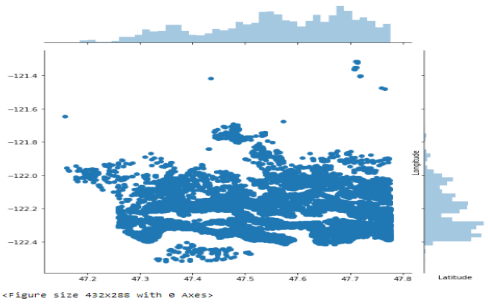
The housing dataset is divided into 90-percent training data and 10-percent test data. This is to have both training data and testing data within the main dataset to get predictions and their accuracy. [1] [3]

```
x_train,x_test,y_train,y_test=train_test_split(fnl,hhl,test_size=0.10,random_state=2)
```

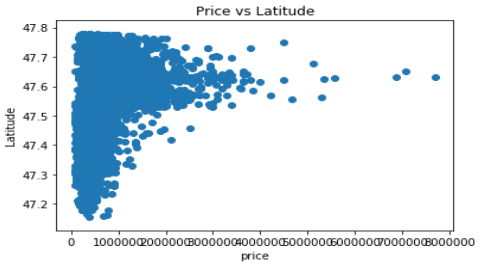
V. Execution and Outputs



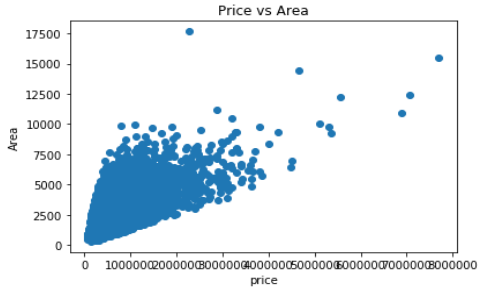

```
<Figure size 432x288 with 0 Axes>
```



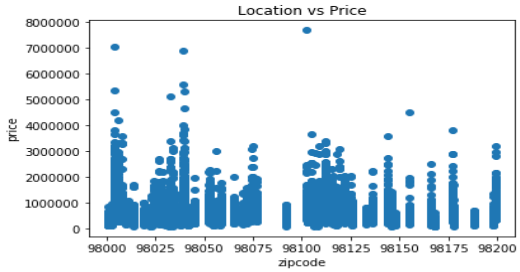
```
<Figure size 432x288 with 0 Axes>
```



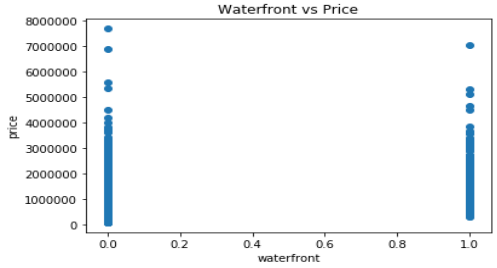

```
<Figure size 432x288 with 0 Axes>
```



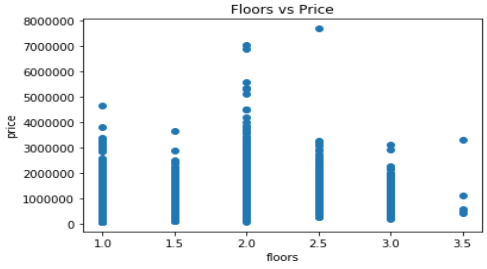
<Figure size 432x288 with 0 Axes>



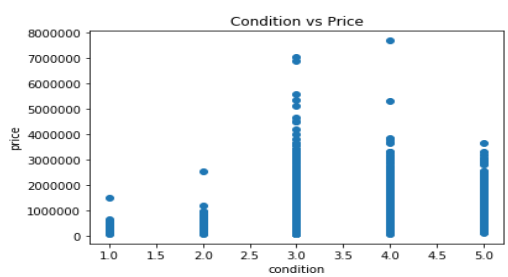
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

VI. Results

The accuracy of different models and their comparison were the results of this project.

```
lr_acc=lr.score(x_test,y_test)
print("Accuracy of linear regression:",lr_acc)
```

```
Accuracy of linear regression: 0.7319844127683144
```

```
lasso_acc=lasso.score(x_test,y_test)
print("Accuracy of Lasso Regression:",lasso_acc)
```

```
Accuracy of Lasso Regression: 0.7320400385186092
```

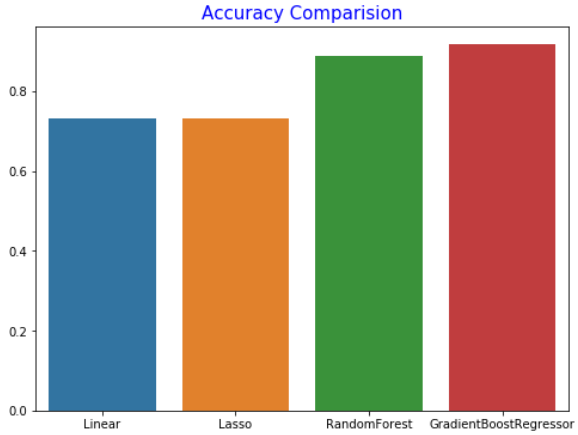
```
rf_acc=rf.score(x_test,y_test)
print("Accuracy of Random Forest Regressor:",rf_acc)
```

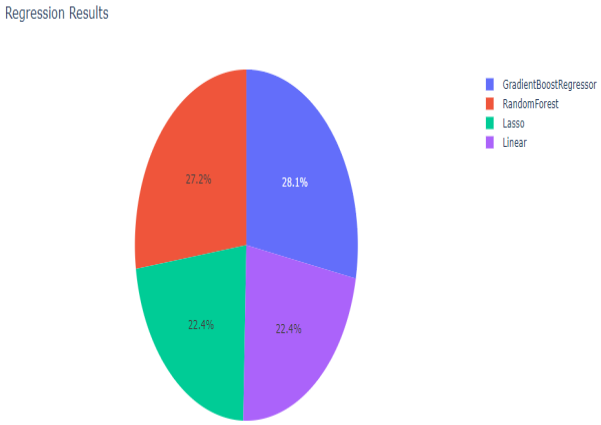
```
C:\Users\saiaj\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning

The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
```

```
Accuracy of Random Forest Regressor: 0.8879689348801579
```

```
gbr_acc=gbr.score(x_test,y_test)
print("Accuracy of Gradient Boosting Regressor:",gbr_acc)
```

```
Accuracy of Gradient Boosting Regressor: 0.9170298061178987
```





VII. Conclusion

We have managed to compare and select the best model based on accuracy that gives novel approach on house price predictions.

Both Linear Regression and Lasso Regression has almost equal accuracy. Random forest has a bit more accuracy when compared to Linear and lasso regressions as depicted in the output logs.

As the time permits, we have used Gradient Boosting algorithm in our prediction system which has combined the poorly accurate prediction models all together to produce the highest accuracy among them.

Over all the machine learning techniques like Linear Regression, Lasso Regression, Random Forest and Gradient Boosting algorithm, Gradient Boosting Algorithm was proved to be the best machine learning technique to predict house prices as it ensembles weakly progressed models into a powerful prediction model. [1]

References

[1] P. Flach, *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[2] A. Varma, A. Sarma, S. Doshi, and R. Nair, "House price prediction using machine learning and neural networks," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 1936–1939, IEEE, 2018.

[3] S. Raghavan, "Create a model to predict house prices using python," 2017.

[4] S. J. Xin and K. Khalid, "Modelling house price using ridge regression and lasso regression," *International Journal of Engineering & Technology*, vol. 7, no. 4.30, pp. 498–501, 2018.

[5] A. J. Bency, S. Rallapalli, R. K. Ganti, M. Srivatsa, and B. Manjunath, "Beyond spatial auto-regressive models: Predicting housing prices with satellite imagery," in *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 320–329, IEEE, 2017.