Prediction of Real State Property Prices Using Machine Learning Algorithm

Samina Islam Eva, Souradip Dey, Md Rubayatur Bhuyian
Department of Electrical & Computer Engineering
University of Waterloo
sislamev@uwaterloo.ca, s25dey@uwaterloo.ca, mrrbhuyian@uwaterloo.ca

Abstract-

Advanced forecast of house prices is important for both potential buyers and sellers to make informed decisions. This problem is a popular machine learning problem in "Kaggle" and the data set was collected from "Kaggle" site. In this paper, an attempt has been made to predict the final price of each home located in Ames, Iowa using several machine learning algorithms such as XGBoost, Support Vector Regression (SVR), Neural Network (NN), and Random Forest Regression (RF). Later ensemble of the algorithms were done to achieve improved result. The prediction was made based on several features such as number of bedrooms, year built, location, living area etc. and the root mean square error of the price variation are selected as the performance index. Testing and validation scores for different algorithms implemented in this paper were later compared with the top scorer in "Kaggle".

***Keywords*— Machine learning; XGBoost; Support Vector Regression (SVR); Neural Network (NN); Random Forest Regression (RF)**

I. Introduction

Real estate transactions such as sales, loans, and its marketability depends on its appraisal value and can have significant impact on any transaction.

Traditionally, estimates of property prices are often determined by professional appraisers. There is a possibility that the appraiser is likely to be biased due to vested interest from the lender, mortgage broker, buyer, or seller. Therefore, an automated prediction system can serve as an independent third party source that may be less biased.

For the buyers of real estate properties, an automated price prediction system can be useful to find under/overpriced properties currently on the market. This can be useful for first time buyers with relatively little experience, and suggest purchasing offer strategies for buying properties.

II. Literature Review

There are several works done before on this type of problem specially using neural network as it performs better with large number of attributes and data points. One didactic and heuristic dataset commonly used for regression analysis of housing prices is the Boston suburban housing dataset. Previous analyses have found that the prices of houses in that dataset is most strongly dependent with its size and the geographical location. More recently, basic algorithms such as linear regression can achieve 0.113 prediction errors using both intrinsic features of the real estate properties (living area, number of rooms, etc.) and additional geographical features (socio demographical features such as average income, population density, etc.)

Yann LeCun's [7] group was able to accurately predict the temporal patterns of housing prices in the Los Angeles area by taking into account geographical data.

Machine learning algorithms like k-NN, Random Forest were implemented by Nissan Pow, Emil Janulewicz and Liu Liu [5] to accurately predict the temporal patterns of housing prices in Montreal area.

L. Li and K. H. Chu [2] varied the number and depth of the layer with different activation function for neural network to predict the real state price variation based on economic parameter.

The dataset here is from kaggle competition. Several machine learning algorithm have been tried to predict the sales price and to get close to the RMSE score of the top participants.

III. Methodology

The objective of this project is to predict the sales price for each house for each Id in the test set. To achieve it is required to train the model and select appropriate parameter. In doing so we followed the below steps:

- Data Preprocessing

Data preprocessing includes data visualization, feature engineering, data normalization, encoding, creation of dummy variables etc. Missing values were imputed and outliers were detected and handled at this stage.

- Application of different Algorithm

Several regression algorithms such as XGboost, Support Vector Regression (SVR), Random Forest (RF), Neural Network (NN), K-Nearest Neighbour (K-NN) are implemented.

- Parameter Tuning

Several parameters of each model are varied to obtain better score (RMSE).

- Result Comparison

Comparison between results have been made to find out the model with the lowest RMSE.

IV. Experimental Setup

Training data set provided by "Kaggle" has been used to train. To evaluate the model 5 fold cross-validation has been done. RMSE error has been calculated after tuning the parameter and picked the best tuning parameter which gave the lowest RMSE value.

As this is still a running kaggle competition, the "SalesPrice" value for test set not available. To verify our result we split the train data set into 70% as training set and 30% as testing set and tried to find the "SalesPrice" for the testing set. Later RMSE value for the test set was calculated to verify our model accuracy.

Finally, model scores were compared with best kaggle's scores to get an idea about the effectiveness of each model. Several new model were tried in this project which has not been tried by kagglers, so some comparisons could not be done.

V. Data Set

The total data set is divided into two parts i.e. train set and test set. Each data set contains 80 independent features except the train data set which contains the dependent variable "SalesPrice". It represents the final price at which a house was sold. The target is to find out the "SalesPrice" for the test set. The data set contains both numerical and categorical features. Also both nominal and ordinal type of categorical value is present is the data. Train set contains 1460 samples and test set contains 1459 samples. As this is a running "Kaggle" completion "Salesprice" for the test price is not available.

VI. Data Pre-processing

A. Data Visualization

Data Visualization is very important to understand the data and to find the correlation between target and feature. Below diagram shows the correlation between "SalesPrice" and different features of the data set. There are many features in the data which are highly correlated with Sales price and some are very weekly correlated.

OverallQual, GrLivArea, GarageCars, GarageArea, TotalBsmtSF, 1stFlrSF, FullBath, TotRmsAbvGrd, YearBuilt, YearRemodAdd have more than 0.5 correlation with SalePrice.

EnclosedPorch and KitchenAbvGr have little negative correlation with target variable.

These can prove to be important features to predict SalesPrice.

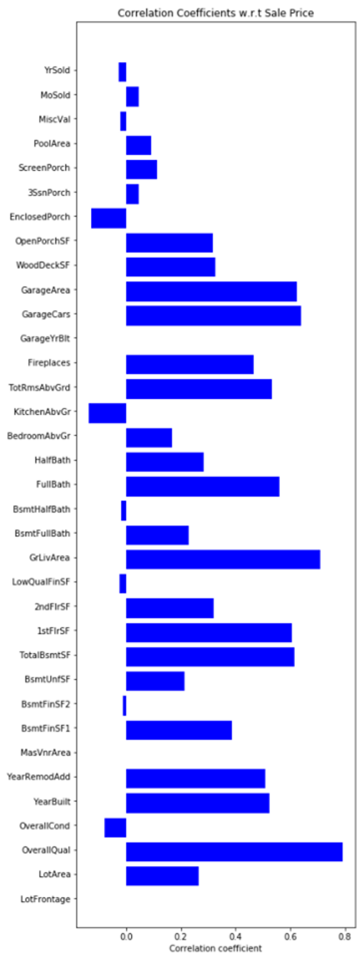
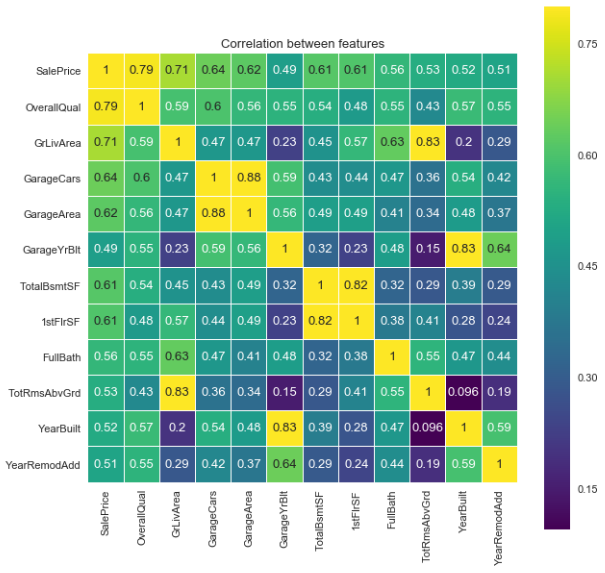


Figure 1. Correlation Coefficients w.r.t. Sales Price



	SalePrice	OverallQual	GrLivArea	GarageCars	GarageArea	GarageYrBlt	TotalBsmtSF	1stFlrSF	FullBath	TotRmsAbvGrd	YearBuilt	YearRemodAdd
SalePrice	1	0.79	0.71	0.64	0.62	0.49	0.61	0.61	0.56	0.53	0.52	0.51
OverallQual	0.79	1	0.59	0.6	0.56	0.55	0.54	0.48	0.55	0.43	0.57	0.55
GrLivArea	0.71	0.59	1	0.47	0.47	0.23	0.45	0.57	0.63	0.83	0.2	0.29
GarageCars	0.64	0.6	0.47	1	0.88	0.59	0.43	0.44	0.47	0.36	0.54	0.42
GarageArea	0.62	0.56	0.47	0.88	1	0.56	0.49	0.49	0.41	0.34	0.48	0.37
GarageYrBlt	0.49	0.55	0.23	0.59	0.56	1	0.32	0.23	0.48	0.15	0.83	0.64
TotalBsmtSF	0.61	0.54	0.45	0.43	0.49	0.32	1	0.82	0.32	0.29	0.39	0.29
1stFlrSF	0.61	0.48	0.57	0.44	0.49	0.23	0.82	1	0.38	0.41	0.28	0.24
FullBath	0.56	0.55	0.63	0.47	0.41	0.48	0.32	0.38	1	0.55	0.47	0.44
TotRmsAbvGrd	0.53	0.43	0.83	0.36	0.34	0.15	0.29	0.41	0.55	1	0.096	0.19
YearBuilt	0.52	0.57	0.2	0.54	0.48	0.83	0.39	0.28	0.47	0.096	1	0.59
YearRemodAdd	0.51	0.55	0.29	0.42	0.37	0.64	0.29	0.24	0.44	0.19	0.59	1

Figure 2. Top-12 highly correlated features

From the above heatmap, it can be observed that some of the features show high multicollinearity. Single feature must be created from them to use them as predictors. Multicollinearity results in unstable parameter estimates which makes it very difficult to assess the effect of independent variables on dependent variables.

B. Imputing Missing Values

There were many missing values in the Dataset. So, the data set was cleaned by filling in all NA's with appropriate values. Missing values have been filled up by analyzing the relations between the variables from the description of the dataset and also using the mean of the attributes.

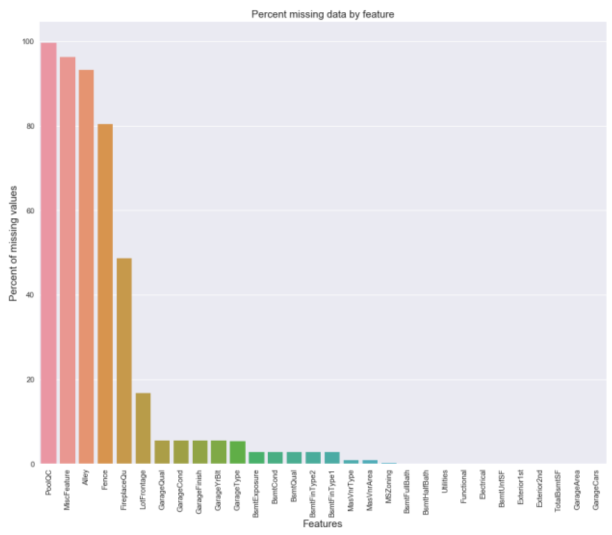


Figure 3. Percentage of missing data by feature

C. Outlier Detection

An outlier is an observation point that is distant from other observations or which does not follow the trend. An outlier may be due to variability in the measurement or it may indicate experimental error. Histogram or scatter plots are usually used to identify outliers. For the given data set outliers have been handled by putting some restrictions on the attributes.

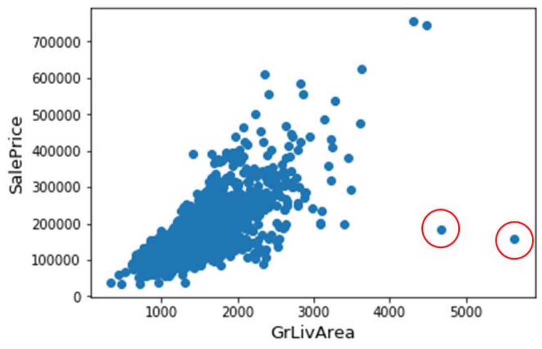


Figure 4. Samples with outlier

From the above diagram we can see that, at the bottom right corner we have two samples with extremely large GrLivArea that are of a low price. These values are huge oultliers. As a result, we have deleted those samples.

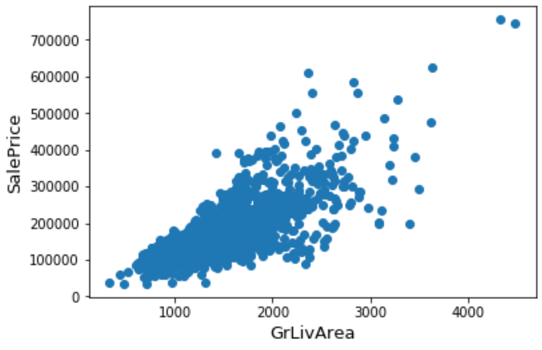


Figure 5. Samples after removing outlier

D. Log transformation

The target variable is right skewed. Linear models love normally distributed data, we transformed this variable and make it more normally distributed.

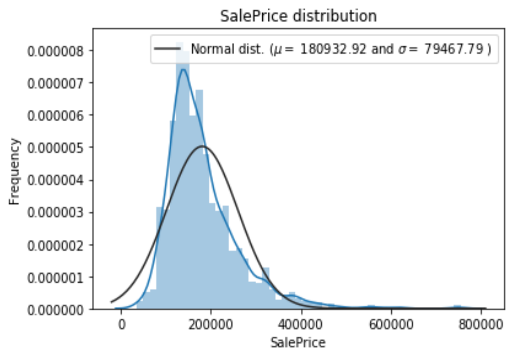


Figure 6. Distribution of "SalesPrice"

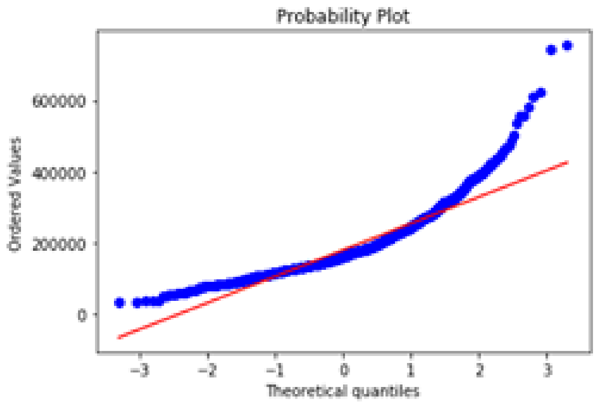


Figure 7. Q-Q plot of "SalesPrice"

Above graph shows that 'SalePrice' is not normal. It's positively skewed and does not follow the diagonal line. Therefore, we need to do a log transformation.

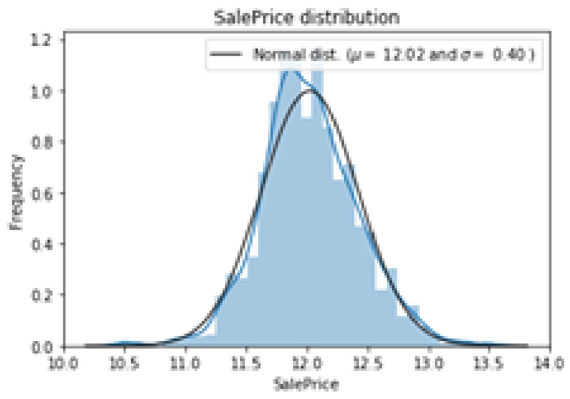


Figure 8. Distribution of "SalesPrice" after log transformation

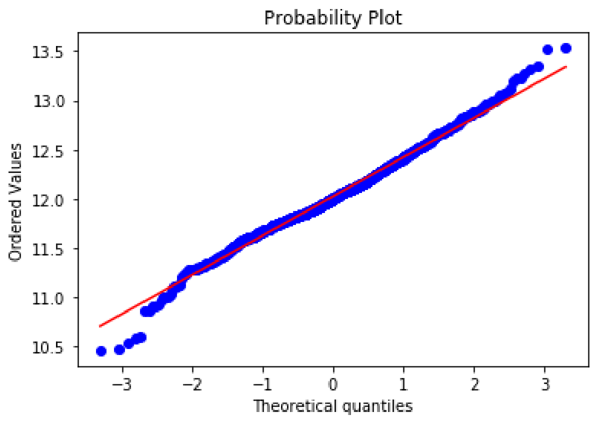


Figure 9. Q-Q plot of transformed "SalesPrice"

E. Label encoding

There are several categorical features with ordinal values. So label encoding was done on those features to transform them to numerical labels.

F. One hot encoding

One hot encoding is performed to transform the categorical features into binary features. Later it is included as feature to train the model. Usually this helps ML algorithms to do a better job in prediction.

G. Normalization

Normalizing means transforming the vector so that it has unit norm. For this particular case Min Max normalization has been selected.

VII. Algorithms

A. Gradient Boosting Regression (XGBoost)

Boosting is essentially a principled way of carefully controlling the variance of a model when attempting to build a complex predictive function. A weak hypothesis or weak learner is defined as one whose performance is at least slightly better than random chance. Gradient boosting involves a loss function which needs to be optimize, a weak learner to make prediction and an additive model to add weak learners to minimize the loss function. Prediction is desired such that loss function (MSE) is minimum, gradient descent is used and updating our predictions based on a learning rate, we can find the values where MSE is minimum.

The main idea is that we should build the predictive function very slowly, and constantly check our work to see if we should stop building. This is why using a small learning rate and weak individual learners is so important to using boosting effectively. These choices allow us to layer on complexity very slowly, and apply a lot of care to constructing out predictive function. It allows us many places to stop, by monitoring the MSE at each stage in the construction.

B. Support Vector Regression (SVR)

Support vector machine algorithm is one of the most popular used method for classification type problems. As the target variable is continuous support vector regression is used instead of svm. These two are quite similar except the use of slack variable. SVM classification involves assigning one slack variable for each observations where as there are two slack variable used for each training point in SVR. The optimization function for SVR is

$$C = \sum_{n=1}^{N} (\varepsilon_n - \widehat{\varepsilon_n})$$

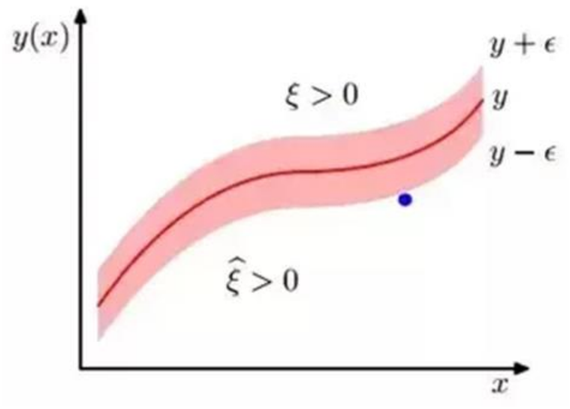


Figure 10. Support Vector Regression

In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. The main idea is always the same which is to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation. Most common used kernels are polynomial and Gaussian radial basis function (RBF).

Polynomial

$$k(x_i, x_j) = (x_i, x_j)^d$$

Gaussian Radial Basis Function

$$k(x_i, x_j) = exp(-\frac{||(x_i - x_j||^2}{2\sigma^2})$$

C. Neural Network (NN)

Neural network maps one set of continuous inputs to another set of continuous outputs. Attributes will be fed into each node of the last hidden layer, and each will be multiplied by a corresponding weight. The sum of those products is added to a bias and fed into an activation function. Activation function can both be linear, sigmoid or RELU (rectified linear unit), RELU is commonly used and highly useful because it doesn't saturate on shallow gradients as sigmoid activation functions do. For each hidden node the activation function outputs an activation and the activations are summed going into the output node, which simply passes the activations sum through. So a neural network performing regression will have one output node, and that node will just multiply the sum of the previous layer's activations by 1. The result will be the network's estimate, the dependent variable that all attributes map to. To perform back propagation and make the network learn it simply compares the estimation to the observation and adjust the weights and biases of the network until error is minimized. Here the loss function is RMSE (root mean squared error).

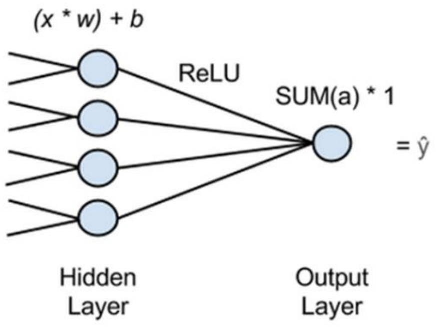


Figure 11. Linear Regression with Neural Network

D. K-Nearest Neighbor (K-NN)

k-nearest neighbors algorithm is a non parametric method used for both classification and regression.in k-NN regression output is the property value for the object. This value is the average of the values of its k-nearest neighbor. The algorithm begins by storing all the input feature vectors and outputs from our training set. For each unlabeled input feature vector, we find the k nearest neighbors from our training set. It uses s Euclidean distance in the m-dimensional feature space. The distance between two vector x and w is

$$d(x,w) = \sqrt{\sum_{i=1}^{m}(x_i - w_i)^2}$$

Here 5-fold cross validation used to cross validate our prediction and Gaussian kernel has been used to weight the output of our neighbor.

$$e^{\frac{-d^2}{2\sigma^2}}$$

Where d is the geographical shortest distance between the two points (in km) and σ is a hyper-parameter to be optimized. By making σ smaller, predictions are weighted more on the nearest neighbours that are also close in a geographic sense. This is very intuitive since houses with similar features that are also close geographically should have similar values. The same concept would be used by a seller or buyer to estimate the price of a house, i.e. looking at the neighbourhood houses with similar features.

E. Random Forest Regression (RF)

Random forest is a supervised learning algorithm. It creates a forest and makes it random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. A decision tree is built top down from a root node and involves portioning the data into subset that contain instances with similar values. On the other hand in a regression tree since the target variable is a real valued number it fits a regression model to the target variable using each of the independent variables. Then for each independent variable the data is split at several split points. Sum squared error is being calculated at each split point between the predicted value and the actual values. The variable resulting in minimum SSE is selected for the node.

$$S = \sum_{c \in leaves(T)} \sum_{i \in C} y_i - m_c$$

Where $m_c = \frac{1}{n_c} \sum_{i \in c} y_i$ the prediction for leaf c.

VIII. Optimization and Parameter Tuning

A. Gradient Boosting Regression (XGBoost)

For XGBoost maximum depth of the tree, no of tree and learning rate were varied to come up with the optimum parameter.

Table 1. Parameter Tuning for XGBoost (n_estimator & Max Depth)

		n_estimator	
		10000	30000
Max Depth	3	0.115374	0.115766
	4	0.115434	0.115993
	8	0.115826	0.116173
	10	0.115968	0.116205

For Max Depth= 3, Learning rate=0.01 & n_estimator=10000 we got lowest RMSE.

Table 2. Parameter Tuning for XGBoost (n_estimator & Learning rate)

		Learning rate				
		0.01	0.05	0.1	0.6	1
n_estimator	500	0.1558	0.1186	0.1187	0.1415	0.1774
	1000	0.1218	0.1181	0.1190	0.1415	0.1774
	2000	0.1185	0.1182	0.1191	0.1415	0.1774
	5000	0.1167	0.1188	0.1191	0.1415	0.1774
	30000	0.1173	0.1188	0.1191	0.1415	0.1774

B. Support Vector Regression (SVR)

For SVR penalty parameter C of the error term, Kernel coefficient "gamma" and kernel type were varied. "rbf" and "ploynomial" kernel has been tested to figure out which kernel performs better for the given dataset.

Table 3. Parameter Tuning for SVR for "rbf" kernel (Gamma & C)

kernel="rbf"								
		gamma						
		0.01	0.02	0.005	0.002	0.001	0.0001	0.00002
C	20	0.181731	0.223746	0.154855	0.136421	0.126711	0.1237	0.14395
	50	0.181731	0.223746	0.154855	0.138249	0.131742	0.118043	0.131321
	70	0.181731	0.223746	0.154855	0.138285	0.132886	0.116934	0.127435
	100	0.181731	0.223746	0.154855	0.138285	0.133849	0.116266	0.123461
	200	0.181731	0.223746	0.154855	0.138285	0.136094	0.118398	0.118868
	500	0.181731	0.223746	0.154855	0.138285	0.136109	0.122658	0.11678
	1000	0.181731	0.223746	0.154855	0.138285	0.136109	0.126933	0.117148

Table 4. Parameter Tuning for SVR for "polynomial" kernel (Gamma & C)

kernel="poly"								
		gamma						
		0.01	0.02	0.005	0.002	0.001	0.0001	0.00002
C	20	0.140267	0.140267	0.140267	0.140275	0.135125	0.161959	0.35073
	50	0.140267	0.140267	0.140267	0.140275	0.139719	0.146237	0.29912
	70	0.140267	0.140267	0.140267	0.140275	0.140275	0.141452	0.279021
	100	0.140267	0.140267	0.140267	0.140275	0.140275	0.137311	0.257823
	200	0.140267	0.140267	0.140267	0.140275	0.140275	0.128715	0.228101
	500	0.140267	0.140267	0.140267	0.140275	0.140275	0.120561	0.201255
	1000	0.140267	0.140267	0.140267	0.140275	0.140275	0.119308	0.183091

From the result it can be seen that "rbf" kernel performs better for the given data set for C=100 & gamma=0.0001.

C. Neural Network (NN)

For neural network number of neurons were varied in different layer to get the lowest RMSE. Also different kernels such as "linear", "relu" has been tried to find the best kernel for the data set provided.

Table 5. Parameter Tuning for NN for "linear" kernel (No. of hidden layers)

"linear"						
No. of Neurons in Hidden Layer		layer 2				
		100	300	400	500	600
layer 1	100	0.343299	0.326485	0.31788	0.325177	0.375135
	300	0.32446	0.355752	0.323224	0.307124	0.322349
	400	0.333497	0.318135	0.349568	0.379026	0.307811
	500	0.282058	0.363063	0.345107	0.409122	0.388828
	600	0.361296	0.358239	0.370395	0.402252	0.331565

Table 6. Parameter Tuning for NN for "relu" kernel (No. of hidden layers)

"relu"						
No. of Neurons in Hidden Layer		layer 2				
		100	300	400	500	600
layer 1	100	0.362541	0.339223	0.353876	0.352505	0.328511
	300	0.316693	0.319169	0.303731	0.300932	0.30308
	400	0.310117	0.287188	0.295033	0.297105	0.336789
	500	0.289877	0.300661	0.310624	0.309844	0.283093
	600	0.29615	0.302093	0.300299	0.288638	0.250434

From the above result it can be observed that "relu" performed better than "linear" kernel. Also higher no of neurons in each layer showed better performance.

D. Random Forest Regression (RF)

For Random forest we changed no of trees in the forest and minimum sample leaf size to find the optimum combination.

Table 7. Parameter Tuning for RF (No. of tree & Minimum Sample leaf size)

		Minimum Sample leaf				
		1	**2**	**3**	**4**	**5**
n_estimators	**100**	0.138287	0.13826	0.138561	0.139595	0.140881
	150	0.137555	0.137579	0.138277	0.1393	0.140797
	200	0.137397	0.137501	0.138215	0.13939	0.140878
	300	0.137384	0.137386	0.138165	0.139421	0.140652
	400	0.137299	0.137322	0.138044	0.139229	0.140472
	500	0.137227	0.137243	0.137927	0.139111	0.140383

Result shows that increase in no. of trees were able to reduce the RMSE whereas increase in Minimum sample leaf size increased the RMSE.

E. K-Nearest Neighbor (K-NN)

For K-NN Number of neighbours were varied to get the optimum no. of neighbour which reduces the RMSE.

Table 8. Parameter Tuning for K-NN (No. Neighbour)

n= 1	**n=2**	**n=3**	**n=4**	**n=5**
0.249428	0.21196	0.206983	0.20522	0.206545
n=6	**n=7**	**n=8**	**n=9**	**n=10**
0.206778	0.204925	0.205186	0.204891	0.205207
n=11	**n=12**	**n=13**	**n=14**	**n=15**
0.205744	0.206142	0.207348	0.208376	0.209431
n=16	**n=17**	**n=18**	**n=19**	**n=20**
0.211034	0.211868	0.212967	0.213884	0.21508

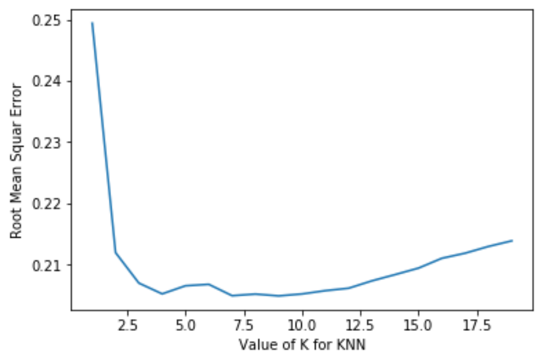


Figure 12. RMSE vs No. of Neighbour

Above graph shows that error initially decreases with the increase of no. of neighbours but increase again after sometime. Optimum no. of neighbour for this particular case is 9.

Below summary table has been provided for various algorithms we have tuned with the optimum result. RMSE has been calculated using 5 fold cross validation.

Table 9. RMSE Comparison between different models

Algorithm	**Parameter Selection**	**RMSE (5-Fold Cross Validation on Training set)**
Random Forest	Minimum Sample leaf = 1 n_estimators = 500	0.137227
XGBOOST	n_estimator = 10000 Max Depth = 3	0.115374

SVR	C = 100
	Gamma = 0.0001
	Kernel = "rbf"
		0.116266

Neural Network	layer 1 = 600 layer 1 = 600 "relu"	0.250434
K-NN	n=9	0.204891
Ensamble	SVR, XGBOOST	0.1146

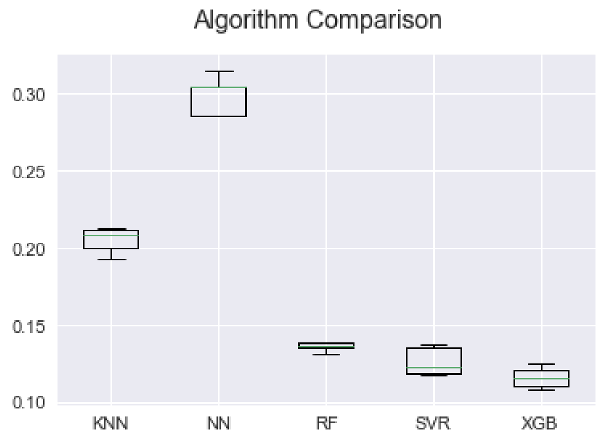


Figure 13. Algorithm Comparison

From the result, it is clear that XGBoost perform better than the other algorithms for the given data set.

Ensamble averaging has also been tried to oimprove the result. Ensemble averaging is the process of creating multiple models and combining them to produce a desired output, as opposed to creating just one model. The result shows a RMSE value of 0.1146 for SVR and XGBOOST which is slightly better than the XGBoost performing alone.

IX. Result

This is a running “Kaggle” competition and to compare our model performance we finally compared our result with top kaggle scorers. Below a comparison has been shown of RMSE value on training data set after 5 fold cross validation between our model and one of the top “kaggle” score. It has been observed that some of the model performed better than Kaggle’s score.

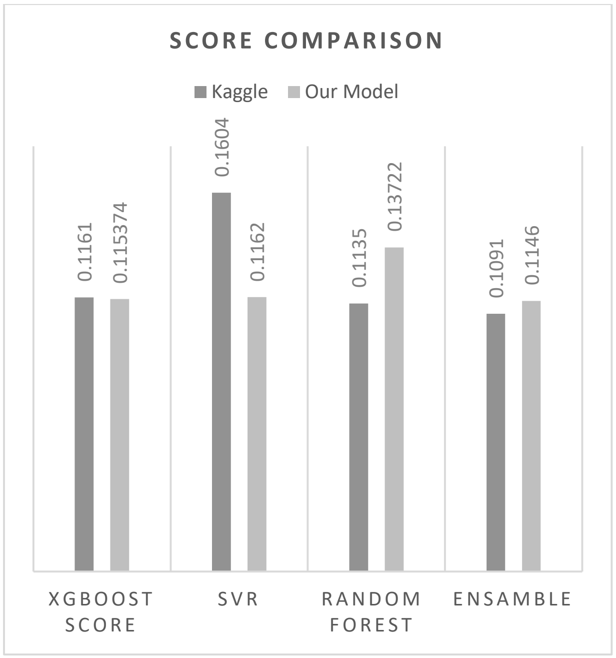


Figure 14. Score Comparison [9]

From the above graph it can be seen that XGBoost and SVR outperformed kaggle’s score but failed to beat the score obtained using Random Forest and Ensambling method.

As “SalesValue” are not available for test data we divided the training dataset in to 70:30 and tested our model on the 30% data. Results showed that parameter tuning obtained better results on the test set.

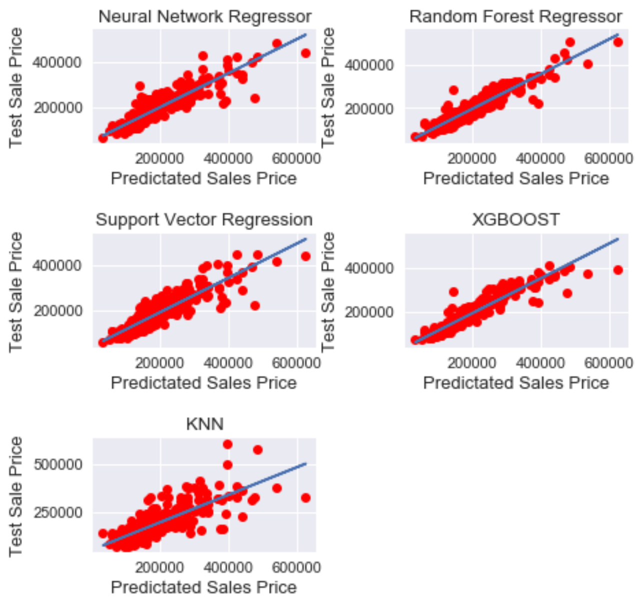


Figure 15. Output (Without Parameter Tuning)

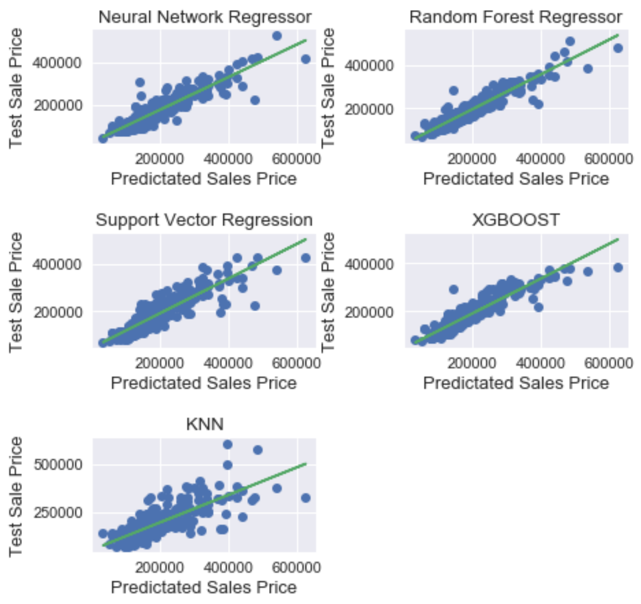


Figure 16. Output (With Parameter Tuning)

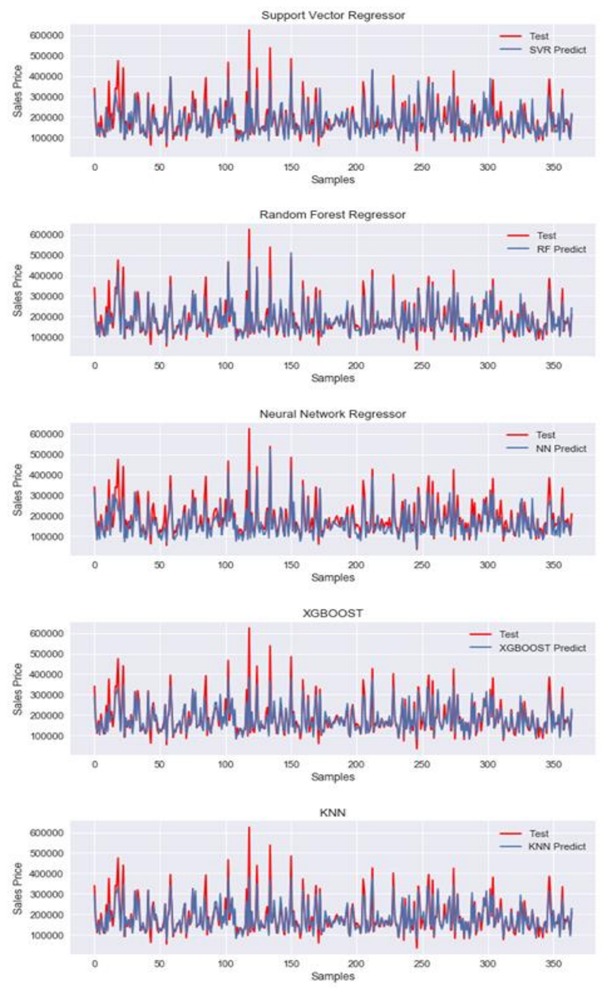


Figure 17. Comparison between test and prediction

Table 10. RMSE Comparison on test data set

Algorithm	**RMSE (Before Parameter Tunning)**	**RMSE (After Parameter Tunning)**
Random Forest	0.0467167	0.0435351
XGBOOST	0.0490879	0.0457886
SVR	0.0565561	0.0554984
Neural Network	0.0669836	0.0551696
K-NN	0.0836104	0.0696217

X. Discussion

In our work, we worked under the assumption that the housing dataset we used is indicative of future unseen examples. We employed a number of predictive models and evaluated the performance of each. The predicted and actual house price were in close agreement as the models have the ability to deduce and generalize the relationship between the input and output variables through learning. However, one of the possible limitations of the learning process is that the size of data set, which consists of only 1460 observations which may not be sufficiently large. We believe an increase in the amount of data will lead to better prediction error.

Of all the models that we tested, we observed that Gradient boosting performed the best with the lowest RMSE. Gradient Boosting is a robust out of the box regressor that can learn non-linear decision boundaries via boosting decision trees. This indicates that the data has some degree of non-linearity that decision trees capture well, and the algorithm allows the decision trees to generalize well by generating a sequence of trees, each grown on the residuals of the previous tree and the sale price is accomplished by weighting the ensemble outputs of all the regression trees.

Support Vector Machine also performed comparably to Gradient Boosting. SVM models using a Gaussian or RBF kernel covers a very rich hypothesis space and using the dual trick, it can work with a very large number of dimensions, and even cases where the number of dimensions is greater than the number of observations. It is thus well suited for our data where we have limited number of observations. One downside for SVM is that it is a better classifier model than a regression model. It still performed quite well.

The next best model in our experiments was the Random Forest. Decision trees are prone to over-fitting. This issue can be reduced by Random forest as it builds lots of weak or shallow decision trees and using a sort of majority vote to result in complex decision boundaries. But Gradient Boosting try to learn a more complex decision boundary by successively fitting trees on the error with a low learning rate over hundreds of trees. Thus we have seen better results for Gradient boosting but Random Forest itself also performs well.

Neural networks are very powerful because it can learn complex representations from the underlying data by learning hierarchical concepts and compositions. It can learn arbitrarily complex functions or decision boundaries. But it can overfit very easily unless it deals with huge number of data points along with thousands of attributes. Moreover, to learn properly neural networks (particularly deep neural networks) require high number of data points which our data unfortunately lacks. We see this issue in our results where we got higher RMSE on 5 fold cross validation. It still performed quite well comparatively.

k-NN build a regression function that lies within the interval of the training data. The optimal value for K will depend on the bias-variance trade-off. Choosing the optimal value for K is done by cross-validation. The training time is low for k-NN compared to other algorithms but the test time is really high. Moreover it suffers greatly from the curse of dimensionality.

XI. Conclusion

We looked to find how different predictive models performed for housing price data. Our focus was on implementing various models to learn about the state-of-the-art machine learning techniques used in data science and compare their performances. However, it was beyond the scope of our work to achieve state-of-the-art performance from our models. In order to do that, we would have to perform highly rigorous data pre-processing, augmentation, feature selection and extraction. Moreover in our work, we worked with some reasonable values for the hyperparameters that are empirically shown to have good results in the real world. But different data requires different hyperparameters to get the best performance. Despite this, we achieved close to state-of-the-art performance from some of our models.

We restricted our work to a particular dataset, but we believe it is possible to extend the models to transfer learning from this dataset to different contexts (e.g.- geographic location) where data is limited. Online learning is another avenue of active research that can be quite useful in his domain. We identify this as further areas to build our work.

References

[1] W. T. Lim, L. Wang, Y. Wang and Q. Chang, "Housing price prediction using neural networks," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, 2016, pp. 518-522.

[2] L. Li and K. H. Chu, "Prediction of real estate price variation based on economic parameters," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, 2017, pp. 87-90. doi: 10.1109/ICASI.2017.7988353.

[3] H. Zhou and Y. Wei, "Stocks market modeling and forecasting based on HGA and wavelet neural networks," in 2010 Sixth International Conference on Natural Computation (ICNC), 2010, vol. 2, pp. 620–625.

[4] Y. Hamzaoui, J. Perez, "Application of Artificial Neural Networks to Predict the Selling Price in the Real Estate Valuation Process," 2011 10th Mexican International Conference on Artificial Intelligence (MICAI), pp.175 – 181, 2011.

[5] Nissan Pow, Emil Janulewicz, and Liu (Dave) Liu, "Prediction of real estate property prices in Montreal".

[6] Jingyi Mu, Fang Wu, and Aihua Zhang, "Housing Value Forecasting Based on Machine Learning Methods", Abstract and Applied Analysis, vol. 2014, Article ID 648047, 7 pages, 2014.

[7] A. Caplin, S. Chopra, J. Leahy, Y. Lecun, and T. Thampy, Machine learning and the spatial structure of house prices and housing returns, 2008.

[8] Kaggle web <https://www.kaggle.com/>

[9] Scikit Learn Web. <http://scikit-learn.org/stable/>