Project Title: The Optimal Machine Learning Model for Automating Real Estate Pricing in Regina

Isaac Labrie-Boulay and Vaughn Geber

1.0 Intro

The global real estate market was valued at USD 3.69 trillion in 2021 and is expected to grow. Real estate estimation contributes a significant overhead cost to firms that operate in such markets. For our project, we will optimize a machine learning model to predict the price of properties on sale using real estate data from Regina.

We are attempting to build a model that can estimate the sale price of properties based on publicly available real estate data. The intent is to reduce the cost and time associated with estimating the cost of one's property. The study will compare and contrast the performance of different machine learning models. Current state-of-the-art approaches are discussed in Section 2.0. From there, we decided on specifically investigating the Random Forest method (RF)[11], the Gradient Boost Method (GB)[7], the K-Nearest Neighbors (KNN) algorithm [10], and finally a Deep Neural Network (DNN)[2]. For this project, each machine learning method will be optimized for the task and then the predictive power of these optimized models are going to be compared with one another.

Additionally, we understand that applications sometimes need to run on systems being more constrained on ressources. Our secondary objective would be to determine which model would be best suited for the task if ressources are more constrained. This means that this paper also considers model runtime and memory usage.

2.0 Preliminary Research

For this project, we did research to find what approaches have previously been used to tackle similar tasks. We specifically focused on the "Machine-Learning-Based Prediction of Land Prices in Seoul, South Korea" [5] paper and the "Identifying Real Estate Opportunities Using Machine Learning" paper [1]. The first paper attempts to build a model that estimates the unit price of a variety of different categories of land and the second paper specifically prices high-end real estate in the Salamanca district in Madrid, Spain.

2.1 Seoul Study

In the Seoul study, the researchers set out to find which model could make the best land unit price predictions of land for sale in Seoul [5]. They split their predictions into 4 different land categories: residential area, industrial areas, commercial areas, and green areas. They got their data from government websites. Their finalized dataset had the following features: individual appraised land value, standard lot status, geographical land information, and other land-use information [5]. The land-use data included administrative location, bearing, area size, topography, shape, and road interface, special district designation, planned facilities, and many others. Their final dataset consisted of 52900 data points [5]. They split their dataset into a training, validation, and test ratio of 7:1:2 respectively [5].

They examined the performance of two different modelling techniques, RF and GB, to predict the price of land. Both are ensemble methods [5]. For their RF models, the researchers found that having 10000 subtrees and using a maximum tree depth of 9 gave them the best results and the rest of the RF hyperparameters were kept to the scikit-learn default values. A similar fine tuning method was employed with the GB method, the optimal hyperparameters were determined to be 18385 estimators, a max depth of 6, and a learning rate of 0.005 [5].

In the end, the researchers determined that the gradient boost model on average performed better on all 4 land categories. For the test data, they reached an accuracy of 85.88 for industrial areas, 84.99 in residential and commercial areas, and 82.58 in green areas [5].

2.2 Salamanca Study

For our research, we also referred to "Identifying Real Estate Opportunities Using Machine Learning" which attempted to build a model for the same task this time for the high-end real estate market of the Salamanca district in Madrid, Spain [1].

They had three categories of data features: location, home characteristics, and 2 advertisement characteristics. This gave them a total of 23 features. They had 2266 real estate assets in their dataset [1].

This group compared 4 different techniques: support vector regression, k-nearest neighbour, ensemble of regression trees, and a multi-layer perceptron. Table 1 below shows the hyperparameters used for all 4 techniques [1].

ML algorithm	Parameter	Values
Support vector regression	Kernel type (kernel) Penalty (C) Kernel coefficient (gamma)	Radial basis function kernel (rbf) 1.0 Inverse of the number of features
K-nearest neighbors	Number of neighbors (n_neighbors) Distance metric (metric) Weight function (weights)	5, 10, 20, 50 Minkowski, cosine Uniform, inverse to distance
Ensembles of regression trees	Number of trees in the forest (n_estimators) Criterion of split quality (criterion) Whether bootstrap samples are used (bootstrap)	10, 20, 50 Mean absolute & mean squared error True, false
Multi-layer perceptron	Network architecture (hidden_layer_sizes) Activation function (activation) Learning rate (learning_rate_init) Optimizer (solver) Batch size (batch_size)	1024, 256-128, 128-64-32 Rectified linear unit (relu) 0.001 Adam 200

Table 1. Model Hyperparameters for the Salamanca Study. This table shows the configuration of the different machine-earning models used in the Salamanca study [1].

2.3 Thoughts on Research

Reading these papers gave us a good understanding of the methodologies required to pick a model best suited for our specific task of predicting the price of houses in Regina. It also gave us significant insight into how to properly segment our dataset and how to find the highest-performing hyperparameters.

3.0 Methods

This section discusses the methodologies used throughout the project at a high level. This includes notes on data, measuring model performance, and testing methodology.

3.1 Data

The real estate data from Regina was gathered from Realtor.ca [9]. We used Parsehub to automatically scrape the data [8]. The data was initially cleaned manually in Excel. This involved deleting data points that had missing features or were incorrectly formatted. Non-residential listings such as lands and lots were also erased from the dataset.

The 724 data points were divided into 80% for training and 20% for testing. Furthermore, the training set itself got split into 80% training and the remaining 20% for validation. This gave us an overal training/validation/test split of 4:1:1.25 respectively. Whenever we trained models many times in a row to calculate mean values, the validation split was always randomly selected from the total training data to add rigor to our statistics.

3.1.1. Correlating features. Before building any model, the first step was to first examine the data. Some features had obvious correlations to house price. The features that correlated most with price were the square footage of the house and the number of bathrooms (See Figure 1). Other features such as the year of build and the number of bedrooms had a littel bit of correlation but much less than we expected (See Figure 2).

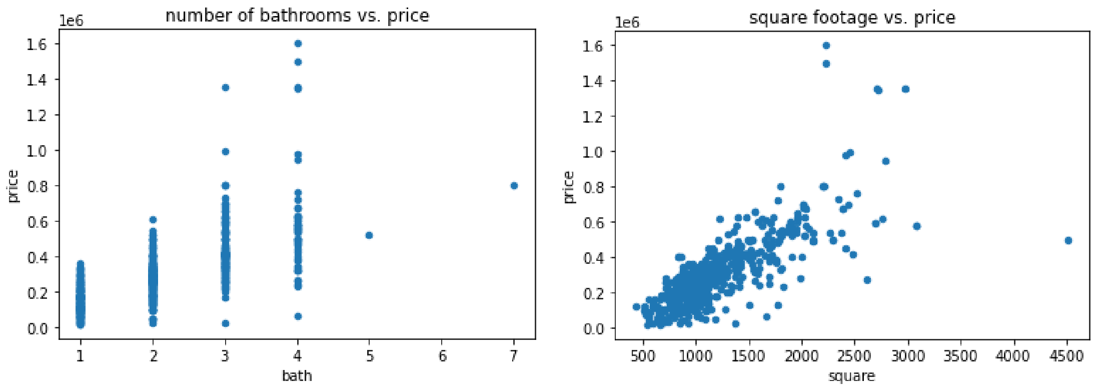


Figure 1. Strongly Correlating Features. The number of bathrooms per house and the square footage correlated most with house price.

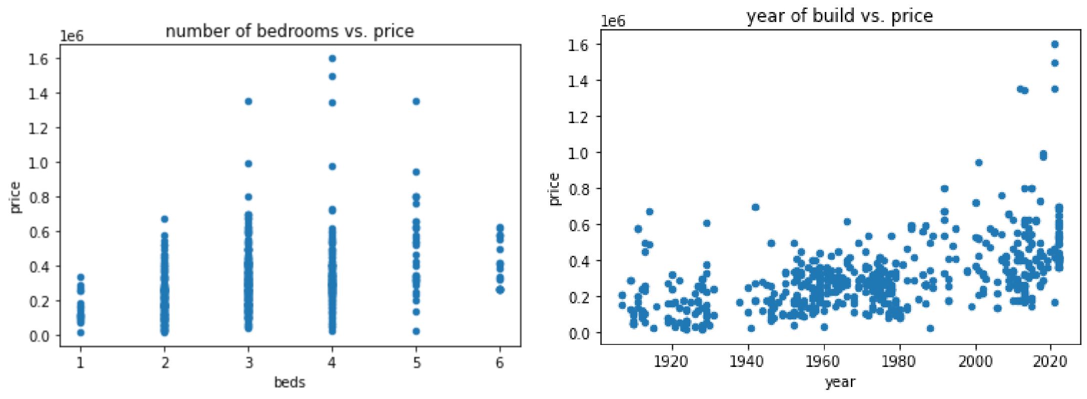


Figure 2. Weakly Correlating Features. The number of bedrooms and the year of build had correlated weakly with house price.

Wether the house was a condominium or a house set a limit on the listing price. It should be noted that although the latitude and longitude don't individually correlate with house price, we still assumed that, as a pair, our machine learning model might be able to do some form of neighborhood subclassification within the fitting process which could improve the predictions.

3.2 Performance Metrics

The goal of a regression task is to minimize error when making predictions on a dataset. The two metrics we used to measure the performance of our models are the R-squared score (R2) [3] and the mean squared error (MSE) [4]. The R2 score seems to be the most used metric to determine the goodness of fit in regression papers. Its value ranges from 0 to 1. A value of 1 signifies that the fit perfectly goes through all data points. The MSE represents the average prediction error on the entire dataset. Depending on the scaling of the data, MSE can take any positive value and a perfect prediction results in an MSE of 0.

3.4 Models Investigated

As aforementioned, throughout the project we examined the performance of the Random Forest method (RF), the Gradient Boost Method (GB), the K-Nearest Neighbors (KNN) algorithm, and Deep Neural Networks (DNN).This section gives a brief description of how each machine learning method works

3.4.1 Random forest regressor. The RF is an algorithm that splits data up into many decision trees and runs an average of their estimates once they all have been completed [11]. Therefore this algorithm is an ensemble; it uses many sub problems to achieve a more robust result.

Each sub tree in the RF is called an estimator. Estimators each make their own prediction and once they have all made their own predictions, the average is compiled, forming the final estimate [11]. The number of estimators can be defined in the ScikitLearn *RandomForestRegressor* function.

When using the *RandomForestRegressor* in ScikitLearn, there is a plethora of parameters to chose from. For our model, we hypthosized that the most impactful parameters were, learning rate, number of estimators, as well as the estimator tree depth. Our parameter tuning will be discussed in the results section.

3.4.2 Gradient boost regressor. GB, like RF, is an ensemble method [6]. Except for running the sub trees in parallel, GB works in a forward-staging fashion [7].

The GB regressor has two estimators. The first estimator will take into account the input parameters by sorting them using a decision process. Once the decision aspect is completed, the second estimator is used to refine the estimate.

The second estimator's purpose is to collect the residue (*y - y_average*) of the leaf nodes from estimate one and apply another decision process on those values. Once we have received the mean of the second estimator, we then add both estimators to achieve the final prediction [7].

The objective with GB is to minimize error, so to analyze this algorithm, we use the Mean Squared Error. To calculate the MSE, we square all final residuals, sum them together and divide by the number of final residuals.

3.4.3 K-nearest neighbors. The KNN algorithm is a machine learning method which calculates the distance of a data point to its closest K neighbours and predicts these data points by averaging the neighbouring data [10]. The neighbour data points are picked based on features similarity which is given by the literal distance in the space of features. The hyperparameter *K* refers to the amount of neighbours that are averaged to give a prediction [10]. For regression problems, distance between data points are often calculated using either the euclidian or the manhattan distance [10].

We do not have important neighbourhood information such as proximity to school and proximity to parks in our data features but we hypothesized that this algorithm would be able to really leverage our dataset's longitude and lattitude features as good proxies for neighbourhood. Though this algorithm seems very simple compared to the ensemble methods, our guess is that this algorithm would perform very well on our data by literally making predictions based on neighbours. For this project, we used the scikit-learn *KNeighborsRegressor* library.

3.4.4 Deep supervised learning. Deep learning is a machine learning technique which attempts to mimic how neurons in our brains learn. DNNs consist of interconnected layers of neurons which can be non-linearly "activated" [2]. The input neurons map to the *X* features of the data points, and in the case of a regression task, its output layer consists of a single neuron who's activation level given the input data corresponds to a continuous values such as the price of a house.

DNNs are trained by passing data through the network (forward pass) and making a prediction based on the weight and biases of all its neurons [2]. The deviation from the real truth value y is calculated as a loss (Mean Squared Error in our case). From there, the gradient of the loss function is calculated and the weights are all adjusted to minimize the loss [2]. To be able to propagate the error from the output all the way back through the model to the input, an algorithm called Backpropagation is used [2].

The deep learning method was not used in any of the research papers that we refer to in this project. These types of model are known to be able to fit highly non-linear curves and we were simply curious to see if this method could maybe draw unexpecting high level features from features which seemingly do not correlate with house price. At the same time, we were cognizant that we cannot overfit the data and so we took precautions while training models. We used *Tensorflow.keras* to build these DNNs.

3.5 Testing Methodologies

Though all tuneable parameters in the *scikit* and *keras* implementations of these algorithms are important, finding the optimal tuning for each would have simply taken too much time. Therefore, the first step in optimizing the models for each method was to figure out which hyperparameters are the most impactful in each algorithm. For this, we simply referred to other research papers to see which parameters were the most commonly examined. From there, we proceeded to "sweep" through different combinations of parameters to find the best combinations.

There are many random processes in machine learning such as random model weight initialization, random batches, or stochastic optimizers to name a few. Therefore, we made sure to always calculate the mean and standard deviation for the model training/prediction procedure. We used a sample size of 20 throughout the project.

4.0 Results

The following section outlines which parameters and hyperparameters performed the best for all 4 machine learning techniques we studied. Finally, it explains which model we chose for our application.

4.1 Best Models by Method

After sweeping over the many different hyperparameters for each machine learning methods, sections 5.1.1 to 5.1.4 summarize our findings for each machine learning methods. In these sections, optimal parameters and other observations are discussed.

4.1.1 RF Results. When tuning the hyper-parameters, we decided to test 3 different parameters, number of estimators, depth of those estimators, and whether or not the model should be bootstrapped. After running different configurations we found an ideal number of estimators to be 100 with a depth of 10 for each. We also concluded that the best model should be bootstrapped (set to true).

Our average R2 score for this configuration was 0.8463 with a mean squared error of 0.0025. Considering this model only took ~1.2s (~10x faster than GB) to complete and is less than 1% away from our best model in Gradient Boost, this may be an ideal model if time is a contraining factor.

RF also used less memory than GB, but the difference is negligible. The peak memory usage was 163.7MB.

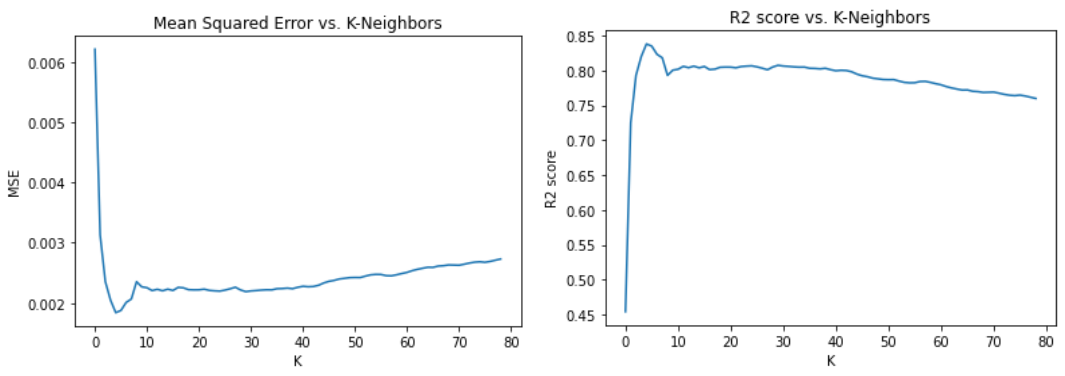
4.1.2 GB Results. After testing three different hyper-parameters (learning rate, number of estimators, and depth of estimators) for GB, we have determined some ideal parameters. We tested various learning rates we came out with the 0.000625 as the ideal. For the number of the estimators we have determined that 18,500 was the ideal. And for the depth of each estimator, we found that a depth of 3 was sufficient in producing the best result.

After testing these three hyper parameters, we came out with an R2 score of 0.8549 and a mean squared error of 0.0018. When compared to the results from the study from Seoul and Madrid, these seem to be very acceptable values.

The model did take much longer to run than RF, clocking in at ~13s per run. And with the number of estimators at 18,500, we began to run out of memory with some of these test trials. The peak memory usage for best hyper-paramters was 186.4MB. Although the model did take longer to run, it did produce the best result.

4.1.3 KNN Results. Given our training set, the KNN models gave the best performance when trained using the correlating features (square footage, number of bathrooms, and residential/condo categories) with the addition of the latitude and longitude values. The models did not perform as well when trained using all the data or when trained with only the correlating features. Our interpretation of these results is that the non-correlating features simply add unhelpful noise to the training and that latitude and longitude helps the model infer the price based on the literal neighbourhood in which the house is found in the city.

The models showed better performance when distance was calculated using the manhattan distance and with a K of around 5.



Figures 3. Performance plotted against K-neighbors. The plots above show the MSE and R2 score versus the chosen K value of the trained model using the manhattan distance to make predictions. The peak is at K = 5.

The model training statistics were a mean R-squared score of 0.768 with a standard deviation of 0.0726. The standard deviation is very high. The model takes an average of 1.94 milliseconds to train and predict.

4.1.4 DNN Results. For the deep learning method, we experimented with many parameters in order to determine which model performed the best. These parameters included learning rate (LR), dropout percentages, the number of hidden layers, training batch size, and using a learning rate scheduler. It should be noted that the number of nodes per hidden layer stayed at a constant 512 nodes. See Table 2 for a summary of the best model's parameters.

Parameter/Hyperparameter	**Values**
Activation Function	ReLU
Node per layer	512
Dropout	10%
Batch Size	64 data points
LR Schedule	starting LR = 0.001 monitor = validation MSE patience = 7 epochs decrease factor = 0.4 minimum LR = 0.000001

Table 2. Best DNN Parameters. The table shows the values of the parameters and hyperparameters of the best-performing DNN model.

To prevent overfitting, we made sure to implement the TensorFlow *EarlyStopping* callback which was programmed to stop the training early if the validation MSE stayed constant for 7 epochs of training. The training could run for a maximum of 200 epochs. See Figure 4 for an illustration of the learning curves for the MSE.

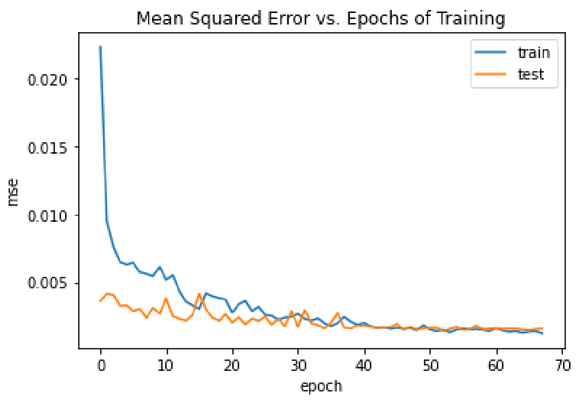


Figure 4. DNN MSE Learning Curves. This figure illustrates how the training stops when the validation set MSE finds a minimum with patience of 7 epochs. This coincides with its intersection with the training set MSE curve.

For the best parameters, the model training statistics were a mean R-squared score of 0.777 with a standard deviation of 0.023. The average MSE was 0.0031 with a standard deviation of 0.0003. The model takes a lot of time to train and predict (an average of 4.7 seconds).

5.1.5 Summary of Results. Overall, it was found that the GB model performed best on average. This being said, this model takes the longest time to train and make predictions (refer to Table 3). The RF model is almost as performant, it is much more consistent in training given its low standard deviation and it is more than 10 times faster than the GB model. Also, the model uses around 20 MB less RAM as GB. Given its run time and memory space advantages, the RF model is the best. It should be noted that, even though it had the lowest performance, the KNN model is orders of magnitude faster than the rest of the models (600 times faster than the second fastest model). The summary of our study can be found below in Table 3.

Model	R2 score	MSE	Average Runtime (ms)	Size (MB)
RF	$\mu = 0.8463$ $\sigma = 0.0025$	$\mu = 0.0030$ $\sigma = 0.0011$	1200	163.7
XGBoost	$\mu = 0.8549$ $\sigma = 0.057$	$\mu = 0.0026$ $\sigma = 0.0011$	13000	186.4
KNN	$\mu = 0.768$ $\sigma = 0.073$	$\mu = 0.0036$ $\sigma = 0.0016$	2	158.2
DNN	$\mu = 0.777$ $\sigma = 0.023$	$\mu = 0.0031$ $\sigma = 0.0003$	4700	301.4

Table 3. Final Model Statistics. The tables shows the mean and standard deviation for the R2 score, MSE, and runtime for the 4 optimized models. It also shows the peak memory usage while training. As one can see, the gradient boost method gave the best results followed closely by the RF method.

When put against the test data, the order of performance remains consistent, the GB model still has the best performance having an R2 score of 0.780 closely followed by the RF model (0.760), then the DNN (0.750) and KNN (0.707) models. We finally tried the average prediction of all the models against the test data. Specifically, we summed the price prediction of each house given by each model and divided this sum by 4. The average prediction R2 score rose to 0.791 which is even higher than the GB prediction. See Table 4 below for these results against the test data. An interesting thing to note is that the performance of the DNN barely decreased when put against the test data. Specifically it scored at 0.777 on the validation data and 0.750 on the test data. We can therefore infer that this model provides the best generalization capabilities and in theory would require to be trained at a lesser time interval.

Model	R2 score	MSE
RF	0.760	0.00552
XGBoost	0.780	0.00506
KNN	0.707	0.00674
DNN	0.750	0.00576
Average*	**0.791**	**0.00480**

Table 4. Model Performance on the Test Set. This table shows the prediction R2 scores and MSE of each model and the average of all models on the never before seen test set. The average of all model predictions gives the highest score

These results indicate that if the application on which this algorithm runs is not heavily constrained on ressource, taking the average of all models could be used to improve the system (i.e. the application runs on the cloud). If the application were to run on a more constrained embedded system, the RF model would be used instead because of its speed and lower memory usage.

6.0 Conclusion

To conclude, we optimized four very different machine learning regression models for the task of predicting the price of houses as listed on Realtor.ca [9]. Since this was a real estate listing website and not a proper database, we were constrained to 7 data features: the latitude position, the longitude position, the number of bedrooms and bathrooms, the square footage, the year of build, and wether or not the house was a condo. It was found that the GB model performed the best on the validation and test data with an R2 score of 0.780. We then determined that the RF model was probably the better option as it was more time and memory space efficient. This being said, if we predicted the house prices with all four models and averaged the prediction, the R2 score actually increased to 0.791.

In terms of future work, we believe that we can certainly improve our predictive power if we were to get access to a Regina real estate database. This would allow us to have access to so much more data and data features which could hopefully allow us to reach R2 scores that are comparable to the ones in the Seoul study [1]. We think that having had access to more geographical and legal data such as proximity to school, proximity to park, and year of last renovations would have really improved our predicting power. After this, the next step would be to determine how our application would be hosted. Since these websites are updated at relatively low intervals, we would probably be training and running all four models on the cloud and taking the average prediction to price houses. Since some models such as the DNN are seemingly able to generalize to better, another next step would be to determine the best way to weight the predictions of each model before taking the average instead of simply diving their sums by 4.

<u>WORK STATEMENT</u>

This section outlines how the many tasks in this project were delegated. First, Vaughn collected all the data from the website. Together, we manually parsed through this raw data to delete bad data points. Isaac was responsible for writing code which formats the dataset. This includes converting the street address to geographical coordinates, turning the price into integers (getting rid of units), and one-hot encoding the categorical data.

We then both did research to find algorithms which would be suitable for our tasks. That's how we found the research papers that we refer to in this report. We then split the algorithms between the both of us. Vaughn implemented and optimized the Random Forest and Gradient Boost models and Isaac did the same for the K-Nearest Neighbors and Deep Neural Network. We both collected training and validation prediction statistics. Finally, we both worked on implementing the application which uses these models to make predictions and perform the in-class demo.

References

[1] A. Baldominos, I. Blanco, A. J. Moreno, R. Iturrarte, O. Bernadez, and C. Afonso. (2018). *Identifying Real Estate Opportunities Using Machine Learning*, *Applied Sciences (MDPI)*, vol 8, issue 11.

[2] H. Cai, ... S. Han (2022) *Efficient methods for deep learning.* ScienceDirect. https://www.sciencedirect.com/topics/computer-science/deep-neural-network

[3] J. Frost. (unknown date). *How To Interpret R-squared in Regression Analysis*. Statistics By Jim. https://statisticsbyjim.com/regression/interpret-r-squared-regression/

[4] J. Frost. (unknown date). *Mean Squared Error (MSE)*. Statistics By Jim. https://statisticsbyjim.com/regression/mean-squared-error-mse/

[5] K. Jungsun, W. Jaewoong, K. Hyeongsoon, and H. Joonghyeok. (2021) *Machine-Learning-Based Prediction of Land Prices in Seoul, South Korea*, *Sustainability (MDPI)*, vol 13, issue 23.

[6] E. Lutins. (2017). *Ensemble Methods in Machine Learning: What are They and Why Use Them?* Towards Data Science https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

[7] T. Masui. (2022). *All You Need to Know about Gradient Boosting Algorithm − Part 1. Regression.* Towards Data Science. https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502

[8] parsehub. (2022). https://www.parsehub.com/

[9] Realtor. (2022). *Listings in Regina.* https://www.realtor.ca/ Retrieved in October 2022.

[10] T. Timbers, T. Campbell, M. Lee. (2022). *7.5 K-nearest neighbors regression*. Data Science: A First Introduction. https://datasciencebook.ca/regression1.html#k-nearest-neighbors-regression

[11] T. Yio. (2019). *Understanding Random Forest*. Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2