Applied Machine Learning Project 4
Prediction of real estate property prices in Montréal

Nissan Pow
McGill University
nissan.pow@mail.mcgill.ca

Emil Janulewicz
McGill University
emil.janulewicz@mail.mcgill.ca

Liu (Dave) Liu
McGill University
liu.liu2@mail.mcgill.ca

***Abstract*—In this machine learning paper, we analyzed the real estate property prices in Montréal. The information on the real estate listings was extracted from Centris.ca and duProprio.com. We predicted both asking and sold prices of real estate properties based on features such as geographical location, living area, and number of rooms, etc. Additional geographical features such as the nearest police station and fire station were extracted from the Montréal Open Data Portal. We used and compared regression methods such as linear regression, Support Vector Regression (SVR), k-Nearest Neighbours (kNN), and Regression Tree/Random Forest Regression. We predicted the asking price with an error of 0.0985 using an ensemble of kNN and Random Forest algorithms. In addition, where applicable, the final price sold was also predicted with an error of 0.023 using the Random Forest Regression. We will present the details of the prediction questions, the analysis of the real estate listings, and the testing and validation results for the different algorithms in this paper. In addition, we will also discuss the significances of our approach and methodology.**

I. Introduction

Prices of real estate properties is critically linked with our economy [1]. Despite this, we do not have accurate measures of housing prices based on the vast amount of data available. In the Montréal island alone, there are around **15,000** current listings at Centris.ca, and around **10,000** historical sales at duProprio.com. This dataset has close to a hundred features/attributes such as geographical location, living area, and number of rooms, etc. These features can be supplemented by sociodemographical data from the Montréal Open Data Portal and Statistics Canada. This rich dataset should be sufficient to establish a regression model to accurately predict the price of real estate properties in Montréal.

A property's appraised value is important in many real estate transactions such as sales, loans, and its marketability. Traditionally, estimates of property prices are often determined by professional appraisers. The disadvantage of this method is 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.

One didactic and heuristic dataset commonly used for regression analysis of housing prices is the Boston suburban housing dataset [2]. Previous analyses have found that the prices of houses in that dataset is most strongly dependent with its size and the geographical location [3], [4]. 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 (sociodemographical features such as average income, population density, etc.) [5], [6].

In the temporal domain, new machine learning algorithms were implemented by Yann LeCun's group to accurately predict the temporal patterns of housing prices in the Los Angeles area [7]. By taking into account geographical data, they were able to more accurately predict the temporal trends in housing prices using basic regression models (0.153 to 0.101).

In this machine learning paper, we predicted the selling prices of properties using regression methods such as linear regression, Support Vector Regression (SVR), k-Nearest Neighbours (kNN), and Regression Tree/Random Forest Regression. We predicted the asking price with an error of 0.0985 using an ensemble of kNN and Random Forest methods. We will present the details of the analysis, and the testing and validation results for the different algorithms below. In addition, we will also discuss the significances of our approach and methodology.

II. Prediction Question and Dataset

The prediction question is to predict the price of real estate properties using intrinsic features from the real estate listings themselves, and additional geographical features from the Montréal Open Data Portal and Statistics Canada. Although the features used for prediction are the same across the entire dataset, the targets to be predicted can have subtle differences depending on the intended usage of the prediction. For example, for the buyers to determine under/overpriced properties currently on the market, the predicted asking price may be the most useful. Since predictions that are significantly above the actual asking price may be underpriced properties, and the predictions that are significantly below the actual asking price may be overpriced properties (Fig. 5). In another scenario, for the sellers to accurately estimate the market value of their properties, the prediction of the final prices sold may

be more useful. For this reason, we collected data for current real estate listings from Centris.ca, as well as completed real estate sales in Montréal from duProprio.com.

The price sold is usually close to 0.030 of the asking price for real estate properties in Canada (Realtor.ca and Fig. 1). We also used the asking price with all features to predict the price sold, and achieved an error of 0.023. This is an additional advantage of our prediction system as buyers can use our predictions to more accurately estimate an appropriate offer price for the listings.

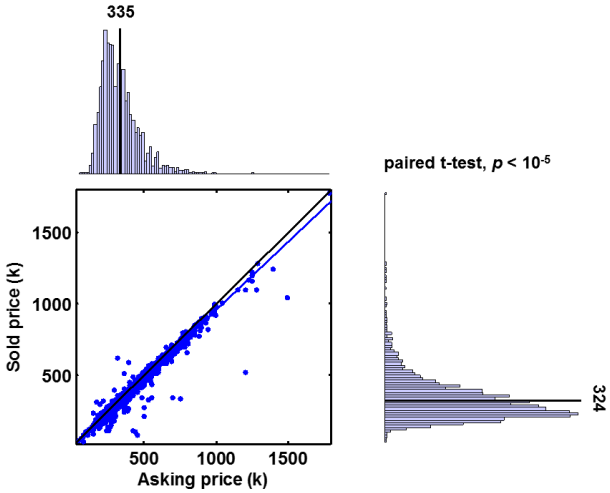

Fig. 1: The final sold price plotted against the asking price.

The dataset has approximately **25,000** examples (15,000 from Centris.ca and 10,000 from duProprio.com) and **130** features. Examples of features that accounted for the most variance in the target prices are listed in Table 1 of the Results section. The features (around **70**) from the real estate listings were mainly scraped from central listing websites, Centris.ca and duProprio.com. Some additional important features such as living area, municipal evaluation, and school tax need to be further scraped from individual real estate agencies such as, RE/MAX, Century 21, and Sutton, etc.

Since geographical location can account for some spatial and temporal trends in the prices [7], we incorporated additional sociodemographical features (around **60**) based on the Montréal borough where the property is located. The corresponding Montreal borough that a property belongs to was determined by inputting the GPS coordinate of its address to the bounding polygons that define the Montreal boroughs. The bounding polygons were obtained from the Montreal Open Data Portal[1] and the sociodemographical features are from the 2006 and 2011 census at Statistics Canada. Examples of the sociodemographical features are the population density, average income, and average family size, etc. for the borough. In addition, we incorporated the geographical distance to the nearest fire station and police station. These datasets were also obtained from the Montreal Open Data Portal[2].

Overall, we found these additional geographical features reduced the prediction error by approximately 0.02 (0.13 to 0.11 across the algorithms). This is comparable to the improvement found for the Boston suburban dataset (0.122 to 0.109) [3], but weaker than the improvement in the prediction of temporal trends (0.153 to 0.101) [7].

III. Methods

A. Data pre-processing

In general, the data parsed from raw HTML files may have mistakes both in the original record or initial processing. Outliers were determined by human inspection of the distribution of values, and subsequently removed/corrected. Fortunately, only a few of the examples had missing values, and these can be excluded from further analysis.

Properties with prices less than $10,000 were not included in the analysis since they were likely to be errors in original data recording or the retrieval process. In addition, we considered properties over 4 times the interquartile range to be outliers. This resulted in 186 out of 7,385 apartments in Montréal to be outliers (Fig. 2). Human inspection of some of these price outliers concluded that they require additional features to accurately predict. For example, some of these apartments included expensive interior decorations and/or furniture in the listing price.

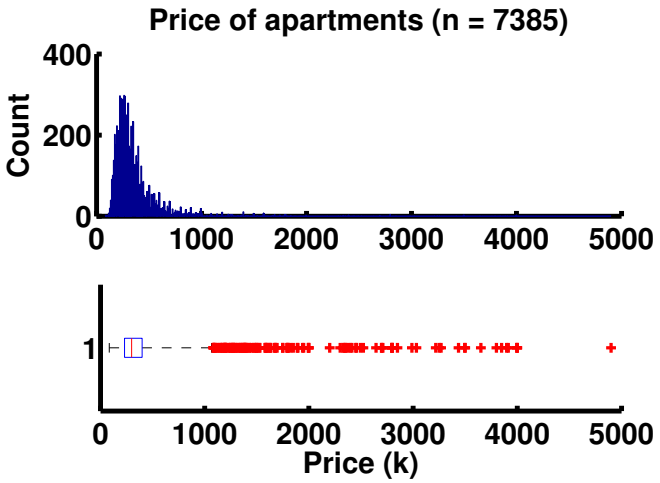

Fig. 2: Distribution of apartment prices in Montréal. Histogram with 500 bins shows a highly skewed distribution of apartment prices. Box-and-whisker plot using whisker length of 4 of interquartile range. The prices that fall outside of the whiskers are consider outliers and are not used in the prediction regression.

B. Feature design and selection

The full set of numerical features can be readily used, and gives good prediction error in our results (Table 3) that is comparable with previous literature [5]–[7]. However, we did attempt some feature engineering as described below.

[1]http://donnees.ville.montreal.qc.ca/dataset/polygones-arrondissements

[2]http://donnees.ville.montreal.qc.ca/dataset/carte-postes-quartier

For the living area of the properties, we used a logarithmic scale since differences in size of smaller properties have a bigger influence on price than differences in size of larger properties [6]. However, this did not improve our prediction error.

To account for temporal factors in the price, we represented the year as a categorical variable as described in [1]. We also found that by incorporating the value of the Montreal Housing Price Index (HPI)[3] for the month when the listing was sold, we were able to reduce the error by 0.01.

To reduce the dimensionality, we used Principal Component Analysis (PCA) to project the examples onto a lower dimensional space. We selected the orthogonal principal components (PC) that represent the most variance in the data [8]. PCA can benefit algorithms such as kNN that relies heavily on the distance between examples in the feature space [8]. However, this did not improve the performance of our kNN algorithm. This is possibly due to that many of the features are noisy compared to the most informative ones (Table. 1). When we used the top 3 features with the highest coefficients from linear regression, we did observe an improvement in kNN performance (see kNN results for more detail), as the magnitude of the coefficients in linear regression are a good proxy for the importance of the feature [9].

Overall, we felt the number of examples in our dataset was sufficient for the training of regression models as the learning curves with one of our top regression model (Random Forest Regression) showed a plateau in prediction error with 90% of the dataset (Fig. 3 and 4).

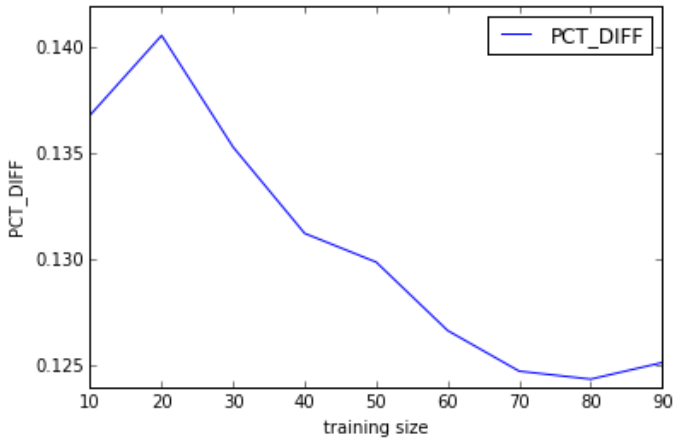

Fig. 3: Random Forest Regression performance as a function of the amount of data. Results are from 10-fold cross-validation from the subset of data.

[3]http://homepriceindex.ca/hpi_tool_en.html

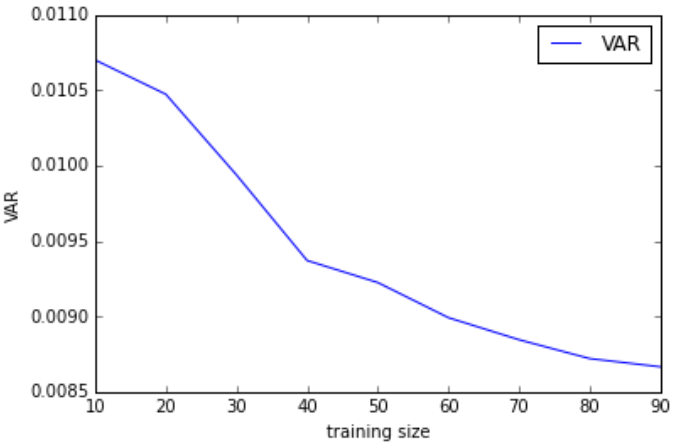

Fig. 4: The variance of the prediction errors in Fig. 3.

IV. Algorithm Selection, Optimization, and Parameters

We selected the following algorithms for regression of prices: Linear Regression, Support Vector Regression (SVR), k-Nearest Neighbours (kNN), and Random Forest Regression. The algorithms were implemented using Python's scikit-learn library [10].

A. Linear regression

To establish baseline performance with a linear classifier, we used Linear Regression to model the price targets, Y, as a linear function of the data, X [8], [11].

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{X}) &= w_0 + w_1 x_1 + ... + w_m x_m \\ &= w_0 + \sum_{j=1:m} w_j x_j \end{aligned}$$

Where w_j are the weights of the features, m are the number of features and w_0 is the weight to the bias term, $x_0 = 1$.

The weights of the linear model can be found with the least-square solution method, where we find the $\mathbf{w}$ that minimizes:

$$Err(w) = \sum_{i=1:n} (y_i - \mathbf{w^T x_i})^2$$

Here $\mathbf{w}$ and $\mathbf{x}$ are column vectors of size $m + 1$.

Re-writing in matrix notation we have:

$$\begin{aligned} f_{\mathbf{w}}(\mathbf{X}) &= X\mathbf{w} \\ Err(\mathbf{w}) &= (Y - X\mathbf{w})^T (Y - X\mathbf{w}) \end{aligned}$$

where X is the $n \times m$ matrix of input data, Y is the $n \times 1$ vector of output data, and $\mathbf{w}$ is the $m \times 1$ vector of weights.

To minimize the error, take the derivative w.r.t. $\mathbf{w}$ to get a system of m equations with m unknowns:

$$\partial Err(\mathbf{w})/\partial \mathbf{w} = -2X^T(Y - X\mathbf{w})$$

Set the equation to 0 and and solve for $\mathbf{w}$:

$$\begin{aligned} X^T(Y - X\mathbf{w}) &= 0 \\ X^T Y &= X^T X\mathbf{w} \\ \hat{\mathbf{w}} &= (X^T X)^{-1} X^T Y \end{aligned}$$

where $\hat{\mathbf{w}}$ denotes the estimated weights from the closed-form solution.

To speed up computation, the weights can be fitted iteratively with a gradient descent approach.

Given an initial weight vector $\mathbf{w_0}$, for $k = 1, 2, ..., m$, $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \delta Err(\mathbf{w}_k)/\delta \mathbf{w}_k$, and end when $|\mathbf{w}_{k+1} - \mathbf{w}_k| < \epsilon$.

Here, parameter $\alpha_k > 0$ is the learning rate for iteration k.

The performance of linear regression is in Table 3.

B. Support Vector Regression (SVR)

We used the linear SVR and also the polynomial and Gaussian kernels for regression of target prices [8], [12].
The linear SVR estimates a function by maximizing the number of deviations from the actually obtained targets y_n within the normalized margin stripe, ϵ, while keeping the function as flat as possible [13]. In other word, the magnitude of the error does not matter as long as they are less than ϵ, and *flatness* in this case means minimize w. For a data set of N target prices with M features, there are feature vectors $x_n \in R^M$ where $n = 1, ..., N$ and the targets y_n corresponding to the price of real estate properties. The SVR algorithm is a convex minimization problem that finds the normal vector $\mathbf{w} \in R^M$ of the linear function as follows [14]:

$$\min_{\mathbf{w},\gamma} \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum_{n=1}^{N} \gamma_n + \gamma_n^\star \right)$$

subject to the constraints for each n:

$$y_n - (\mathbf{w} * \mathbf{x}_n) \leq \epsilon + \gamma_n,$$
$$(\mathbf{w} * \mathbf{x}_n) - y_n \leq \epsilon + \gamma_n^\star,$$
$$\gamma_n, \gamma_n^\star \geq 0$$

Where $\gamma_n, \gamma_n^\star$ are 'slack' variables allowing for errors to cross the margin. The constant $C > 0$ determines the trade off between the flatness of the function and the amount up to which deviations larger than ϵ are tolerated [15].

The results for the SVR can be found in Fig. 5 and Table 3.

C. k-Nearest Neighbours (kNN)

k-Nearest-Neighbour (kNN) is a non-parametric instance based learning method. In this case, training is not required. The first work on kNN was submitted by Fix & Hodges in 1951 for the United States Air-force [16].

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. The notion of *nearest* uses Euclidean distance in the m-dimensional feature space. For two input vectors $\mathbf{x}$ and $\mathbf{w}$, their distance is defined by:

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

Once the k nearest neighbors are selected, the predicted value can either be the average of the k neighbouring outputs (uniform weighting), or a weighted sum defined by some function. We will propose a good choice of such a function based on geographic distance between the test point and its neighbours.

In our analysis we use leave-one-out cross-validation. In this case, if we start with n input vectors, each sample is predicted based on the other $n - 1$ input vectors.

Before passing our data through a kNN regressor, we first used a Standard Scaler (scikit-learn) on our input data. This transforms each feature column vector to have zero mean and unit variance. With all features normalized, each feature has a fair weight in estimating the Euclidean distance and there will be no dominating features.

It has been shown in previous studies that there is strong spatial dependency between house values [6], [7]. Thus we argue that geographical distance between houses will have a large impact on the predicted value. We use a Gaussian Kernel to weight the output of our neighbors:

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

where d is the geographical shortest distance between the two points (in km) and σ is a hyper-parameter to be optimized. We note that by making σ smaller, we are weighting our prediction 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. In fact, the same line of reasoning would be used by a broker to estimate the price of a house, i.e. looking at the value of nearby houses with similar features.

D. Random Forest Regression

The Random Forest Regression (RFR) is an ensemble algorithm that combines multiple Regression Trees (RTs). Each RT is trained using a random subset of the features, and the output is the average of the individual RTs.

The sum of squared errors for a tree T is:

$$S = \sum_{c \in leaves(T)} \sum_{i \in C} (y_i - m_c)^2$$

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

Each split in the RT is performed in order to minimize S. The basic RT growing algorithm is as follows:

1) Begin with a single node containing all points. Calculate m_c and S
2) If all the points in the node have the same value for all the independent variables, then stop. Otherwise, search over all binary splits of all variables for the one which will reduce S the most. If the largest decrease in S would be less than some threshold δ, or one of the resulting nodes would contain less than q points, then stop. Otherwise, take that split, creating two new nodes.
3) In each new node, go back to step 1.

One problem with the basic tree-growing algorithm is early termination. An approach that works better in practice is to

fully grow the tree (ie, set $q = 1$ and $\delta = 0$), then prune the tree using a holdout test set.

V. Results

The results below are reported in the order based on how they performed (worst to best, Table. 3) The prediction errors reported used 10-fold cross-validation unless otherwise noted. We looked at both the mean absolute percentage difference (Fig. 3) and also its variance (Fig. 4). Presumably, the mean and variance should be correlated, however, decrease of either one indicate an improvement in the performance of our model.

A. Linear regression and SVR

Linear regression using Lasso (L1) regularization and the SVR had similar performances (Table 3).

TABLE 1: Variance Accounted For (VAF) in the target price for different features using the linear regression.

Feature	Area	# Rm	# Bedroom	# Bathroom	Pool
VAF	0.472	0.141	0.158	0.329	0.110

We used the SVR with different kernels to predict the target prices (Fig. 5 and Table. 2). Interestingly, the linear kernel had the best performance (Table. 2).

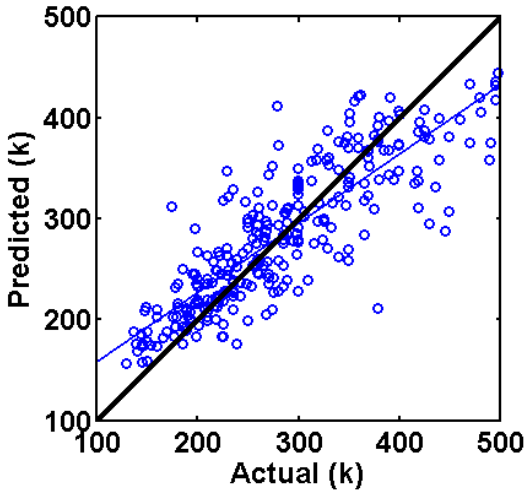

Fig. 5: SVR using features from the listings. Results are from 10-fold cross-validation. Plotting the predicted price vs. the actual price can be useful for the buyers, since the points that are largely above the unity slope line may be properties that are undervalued, and the points that are largely below the unity line may be properties that are overvalued.

TABLE 2: SVM performance with different kernels.

		Polynomial (Order)			
Linear	Radial Basis Function	2	3	4	5
0.1604	0.1618	0.1624	0.1681	0.1796	0.1966

A comparison of the performance of different algorithms are presented below (Table. 3). Details for the other methods will follow.

TABLE 3: Comparison of different regression algorithms.

	LR	SVR	kNN	Random Forest	Ensemble
Error	0.1725	0.1604	0.1103	0.1135	0.0985

B. kNN

All kNN results were computed using leave-one-out cross-validation. As stated earlier, the top 3 informative features were used: living area, number of bedrooms, and number of bathrooms (Table. 1). When using all the features, we get an average error of 0.1918. Thus we have significant improvement by reducing the dimensionality of the features, resulting in 0.1103 (Table 3). Our first optimization task required us to find the optimal σ and k. This corresponds to the variance for our Gaussian Kernel and the number of nearest neighbours. The error metric used is the average percent error: for prediction $\hat{y}$ and actual value y, percent error is defined as $\frac{|\hat{y}-y|}{y}$. For example, Figure 6 gives different average percent errors for varying k and different σ (1,1.5 and 2). After simulating for various values, a minimum was found at 0.1103 for $k = 100$ and $\sigma = 0.4$.

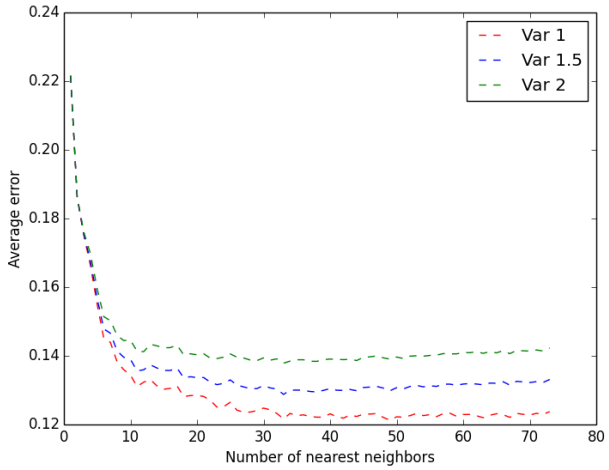

Fig. 6: The performance of kNN as a function of k for different σ.

Another interesting result was to measure the average percent error for inputs where its nearest neighbours in the feature space were at a particular average geographic distance. To measure the average weighted distance, we compute the actual distance between neighbours in kilometers and then take a weighted sum according to our Gaussian Kernel. The binned results are shown in Figure 7.

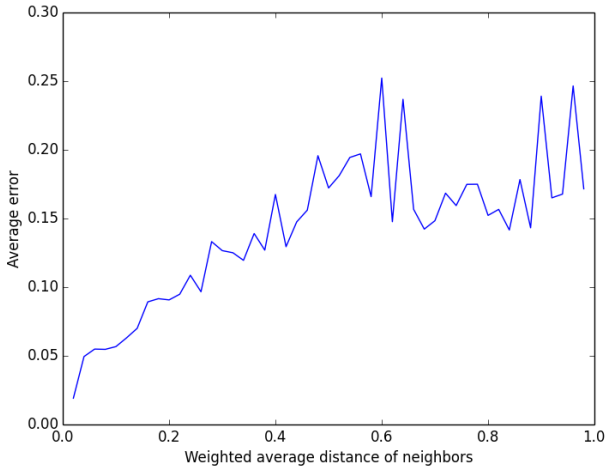

Fig. 7: Average percent error as a function of average weighted distance (km).

As expected, we see that using the nearest neighbours from the feature space which are also close geographically results in the lowest average error.

C. Random Forest Regression

We use 10-fold cross-validation to decide the optimal number of RTs. Figure 8 shows the effect of varying the number of trees with the average percentage error, as defined previously. In general, as we increase the number of trees, the average error decreases until it converges around 0.113. This is in line with other studies which have shown that there is a threshold beyond which there is no significant gain for increasing the number of trees[17].

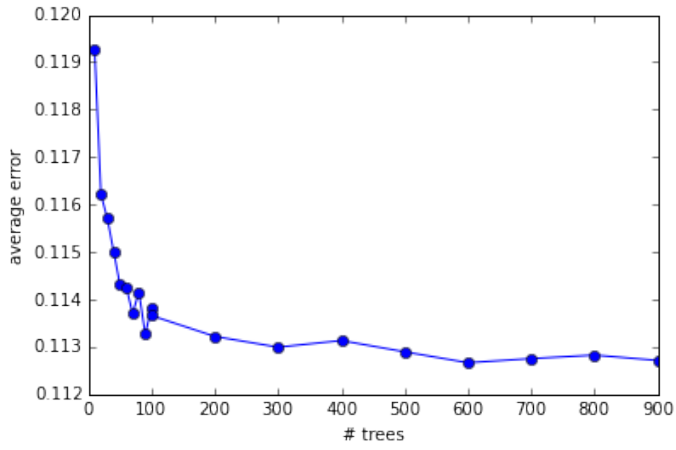

Fig. 8: Results from varying the number of trees vs the average error for the Random Forest Regression. Results are from 10-fold cross-validation.

Overall, the best prediction result comes from a careful ensemble of our two best regression models (kNN and Random Forest). Since our implementation of kNN strongly depended on the geographical distance between the neighbours, the performance is tightly coupled to the number of geographically close neighbours available. For the examples without any neighbours within 100m, we used the Random Forest Regressor to predict the price. This ensemble method gave us a prediction error of 0.0985 (Table. 3).

VI. Discussion

A. Algorithms comparison

kNN and Random Forest Regression performed significantly better than the baseline linear algorithm (linear regression and linear SVR) (Table. 3). This is possible due to their ability to account for nonlinear interactions between the numerical features and the price targets. Also, the version of kNN we implemented using the geographical distance most closely resembles the appraisal technique used by real estate agents. Therefore, the good performance of kNN in our case could be attributed to our attempt of mimicking human methods with machine learning. We attempted some feature engineering such as using the logarithms of the living area and PCA, however, these did not improve the prediction errors for the linear methods. The learning curves indicate that we may have a sufficient number of examples and informative numerical features for our prediction question (Fig. 3 and 4), therefore the prediction errors achieved in Table. 3 are as expected for these algorithms. As we saw in the kNN result, if we have a dense distribution of properties geographically, then we can perform fairly well at those examples (Fig. 7). Of course, this is not always the case in real estate properties, but at least the kNN algorithm gives us a strong intuition into the operation of real estate pricing.

We found that ensembling the kNN with Random Forest Regression improved prediction. Since examples that do not have sufficient number of geographical neighbours are unlikely to be well estimated by the kNN method, we hypothesize that these examples can greatly benefit from the estimate of another regressor. This is a useful strategy that is worthy of future investigation, and this will be further discussed below.

B. Other possible machine learning algorithms

Neural networks are most commonly used for classification tasks. However, since any arbitrary functions can be fitted with a multilayer perceptron [18], [19], theoretically, they should perform equally well in regression tasks. However, preliminary analysis using the PyBrain [20] implementation gave us an error of 0.200. This could be due to the limited amount of data used to train the neural network since we found the prediction performance did not significantly improve after 100's of epochs of training. Future efforts could be spent into applying neural network regression to richer datasets and/or hyperparameter tuning.

C. Contribution to knowledge

Our prediction of housing prices in Montréal using similar sets of features and linear regression methods performed on par with previous literature [3]–[6] (results in Table 3 compared to 0.113 in previous literature). However, using an ensemble of kNN with the Random Forest Regression, we were able to perform at 0.0985. Therefore, this approach

has the potential to be further applied. In another study, Yann LeCun's group used autoregressive models to predict the temporal trend in housing prices, and our performance roughly matched theirs in magnitude [7]. However, we could not predict the temporal trend in our historical data with same degree of accuracy as discussed below.

In the subset of historical data where we had both asking price and final price sold, we achieved a prediction error of 0.023 using the asking price as an additional feature. This is lower than the mean deviation (0.030) between the asking price and price sold (Fig. 1). This application can be useful for the buyers to accurately estimate an appropriate offer price for a particular listing.

D. Open questions and Future directions

While most of our dataset and subsequent analysis focused on using intrinsic and geographical features to predict spatial trends in housing prices, we did not have access to sufficient amount of historical real estate transactions to predict temporal trends in our analysis. Our preliminary analysis predicting the temporal trend of housing prices using thousands of examples per year yields an error of 0.20 on average, while a previous study using close to 100,000 of examples per year was able achieve error of 0.101 [7]. This previous study used similar intrinsic and geographical features as ours with simple regression models, therefore, we believe an increase in the amount of data will lead to better prediction error. The temporal trend in housing prices is critically linked with our economy [1], and future work in predicting the temporal trend in housing prices, i.e. the Housing Price Index, can greatly benefit the City of Montréal.

In well-used datasets in machine learning, improving the error by 0.01 with a particular algorithm can be considered a significant breakthrough [21]–[23]. However, it is arguable whether these improvements will translate into any useful applications in everyday life [24]. Since real estate investments usually involve large monetary transactions (the median of real estate property price in Montreal is around $300,000 (Fig. 2)), improving the the prediction error by 0.01 can lead into the development of interesting future applications for the City of Montréal.

APPENDIX

The dataset used for this project and the code used for the analysis can be found at the link below.

https://github.com/npow/centris

✓ **We hereby state that all work presented in this report is that of the authors.**

REFERENCE

[1] R. J. Shiller, "Understanding recent trends in house prices and home ownership," National Bureau of Economic Research, Working Paper 13553, Oct. 2007. DOI: 10.3386/w13553. [Online]. Available: http://www.nber.org/papers/w13553.

[2] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of Environmental Economics and Management*, vol. 5, no. 1, pp. 81–102, 1978. [Online]. Available: http://EconPapers.repec.org/RePEc:eee:jeeman:v:5:y:1978:i:1:p:81-102.

[3] D. Belsley, E. Kuh, and R. Welsch, *Regression Diagnostics: Identifying Influential Data and Source of Collinearity*. New York: John Wiley, 1980.

[4] J. R. Quinlan, "Combining instance-based and model-based learning," Morgan Kaufmann, 1993, pp. 236–243.

[5] S. C. Bourassa, E. Cantoni, and M. Hoesli, "Predicting house prices with spatial dependence: a comparison of alternative methods," *Journal of Real Estate Research*, vol. 32, no. 2, pp. 139–160, 2010. [Online]. Available: http://EconPapers.repec.org/RePEc:jre:issued:v:32:n:2:2010:p:139-160.

[6] S. C. Bourassa, E. Cantoni, and M. E. Hoesli, "Spatial dependence, housing submarkets and house price prediction," eng, 330; 332/658, 2007, ID: unige:5737. [Online]. Available: http://archive-ouverte.unige.ch/unige:5737.

[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] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

[9] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389–422, 2002, ISSN: 08856125. DOI: 10.1023/A:1012487302797.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*, 1. Springer, 2009, vol. 2.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 0885-6125. DOI: 10.1023/A:1022627411411.

[13] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995, ISBN: 0-387-94559-8.

[14] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004, ISSN: 0960-3174. DOI: 10.1023/B:STCO.0000035301.49549.88. [Online]. Available: http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

[15] K. P. Bennett and O. L. Mangasarian, *Robust linear programming discrimination of two linearly inseparable sets*, 1992.

[16] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: consistency properties," DTIC Document, Tech. Rep., 1951.

[17] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7376 LNAI, 2012, pp. 154–168, ISBN: 9783642315367. DOI: 10.1007/978-3-642-31537-4_13.

[18] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[19] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies in Applied Mathematics*, vol. 52, no. 3, pp. 213–257, 1973.

[20] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.

[21] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS 2010*, vol. 9, Chia Laguna Resort, Sardinia, Italy, May 2010, pp. 249–256.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 3642–3649, ISBN: 978-1-4673-1226-4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2354409.2354694.

[24] K. Wagstaff, "Machine learning that matters," *CoRR*, vol. abs/1206.4656, 2012. [Online]. Available: http://arxiv.org/abs/1206.4656.