Applied Machine Lea
Prediction of real estate prop

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

Emil Janulev
McGill Unive
emil.janulewicz@ma

***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**

On
regre
housi
price
its si
basic
predi
prope
geogr
avera
In
were
predi
Ange
they
in ho
0.101
In
price
ear r
Neigh
gress
using

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.

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.

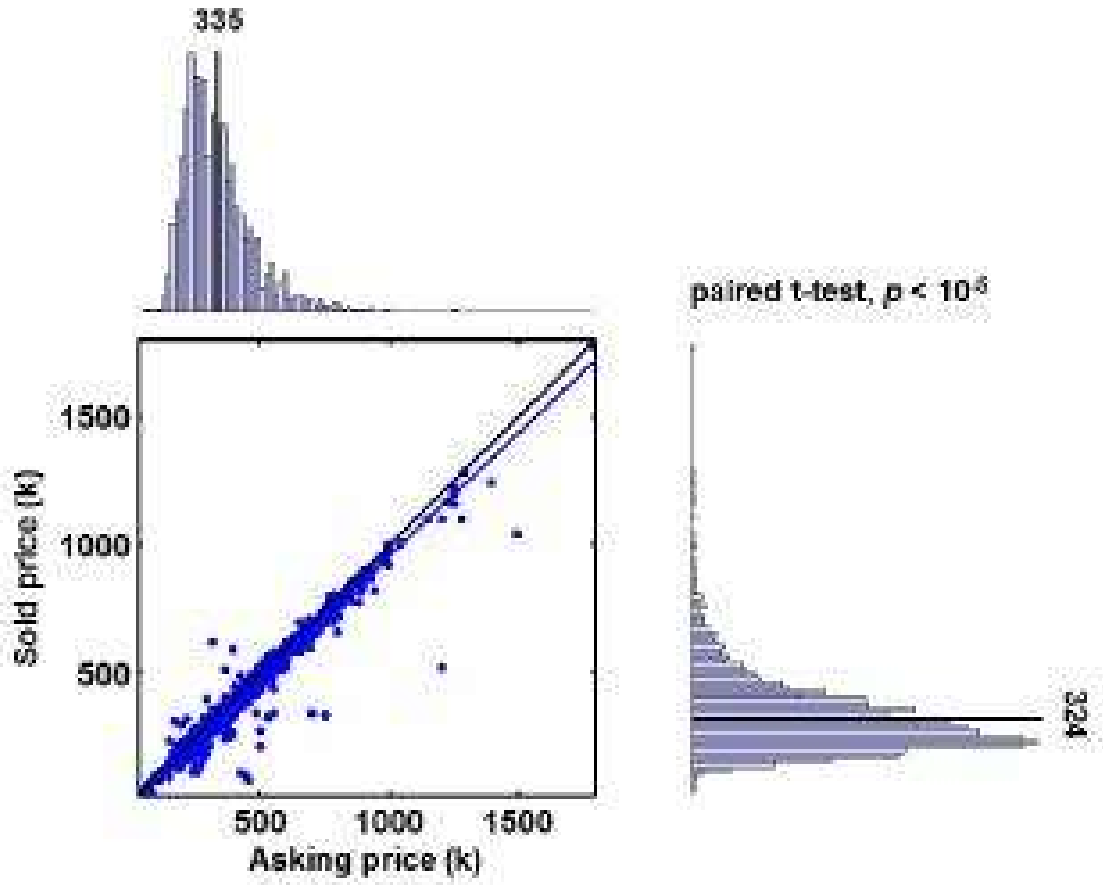

Fig. 1: The final sold price plotted against the asking price.

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

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

Fig. 2:
a highl
length
outliers

B. Fe

Th
gives
comp
attem

[2]htt

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).

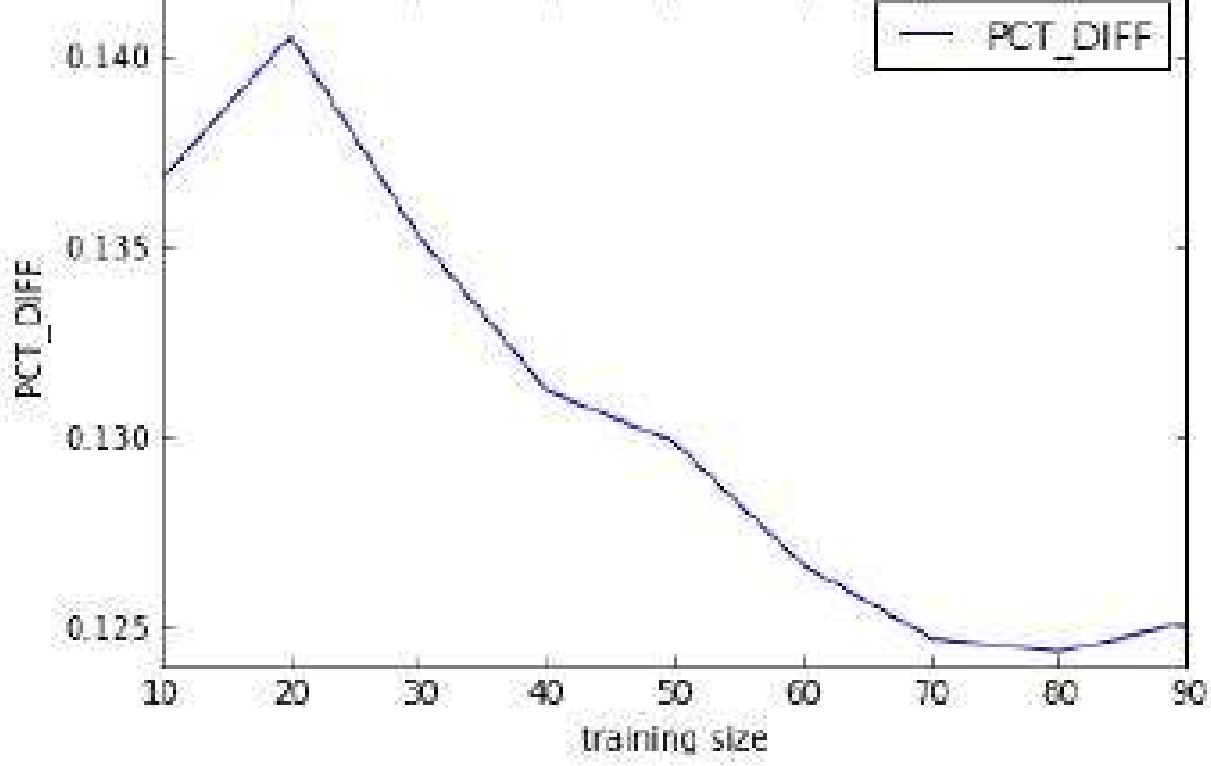

Fig. 3: Random Forest Regression performance as a function of the amount of data. Results are from 10-fold cross-validation from the subset of data.

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

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:

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

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:

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

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