Real-estate price prediction for King County region

Bharat Chandramouli (SUNET ID bharatc)
Project Category: Generalized Machine Learning
Code link: https://drive.google.com/drive/folders/1jGGdRczsmr7aBiNScV348gg6V0VAhFkz

1. Introduction

Real-estate accounts for more than 1.13 trillion of the USA economy and purchasing/selling a house is among the biggest commitments for most people. Accurate prediction of prices based on other sales can be a critical tool to make sure the buyer/seller is making an informed decision. While different real-estate companies are working on custom algorithms to provide an estimate of the value of the property, they don't provide insight into the method used and have varying degrees of error in different locations[1] . The paper evaluates a combination of Stochastic Gradient Descent, Stochastic Dual Gradient Ascent, Gradient Tree Boosting and clustering using K-Means to predict real-estate prices.

The input to the algorithm is data from Kaggle[2] that has listing attributes and price for all the houses sold in King County from May 2014 to May 2015. The algorithm uses the attributes to train a model and predict the price.

2. Related work

There are several papers that use regression techniques to predict real-estate prices. Hujia Yu and Jiafu Wu [1] use regression and classification of houses into buckets based on the price of the house. Kalehbasti et. al.[3] use SVR, K-means with ridge regression, ridge regression, gradient boost and neural networks to predict Airbnb prices.

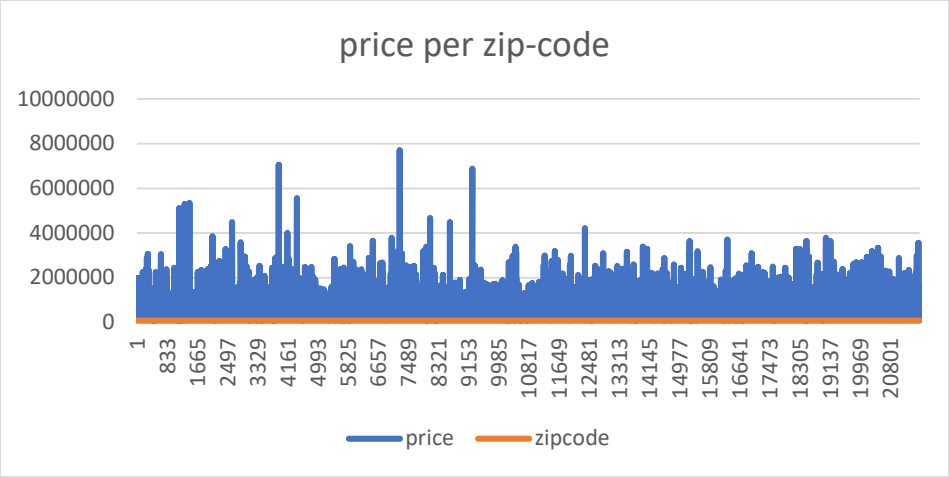
I use Stochastic Gradient decent to establish a baseline and use Stochastic Dual Gradient Ascent, Gradient Tree Boosting and Gradient Tree Boosting with K-Means clustering to determine which algorithm gives the best price prediction.

The implementation uses ML.Net libraries - Stochastic Gradient Descent[8] , Gradient Tree Boosting[5], K-Means[6] and SDCA regression[7].

3. Dataset and Features

The dataset used is from Kaggle[2] . It has 21,000 listings from May 2014 to May 2015. The attributes cover the price, attractiveness, size, location and comparison to 15 nearby homes.

I used manual selection to pick features that seemed relevant and used forward prediction to remove un-related feature. I used 80% for training, 10% for developer validation and 10% for cross-validation.



4. Method:

All the regression models were implemented using ML.Net libraries: Stochastic Gradient Descent[8], Gradient Tree Boosting[5], K-Means[6] and SDCA regression[7].

- Stochastic non-batch Gradient Decent (SGD)
  - Minimizes the least squares loss function.
  - Used learning rate of 0.1 and ran one iteration.

$$L(\theta) = \sum_{i=1}^{n} L_i(\theta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} (\theta_1 + \theta_2 x_i - y_i)^2$$

- Stochastic Dual Gradient Ascent (SDGA)

$$R_{emp}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i)$$

$$\hat{h} = \arg\min_{h \in H} R_{emp}(h)$$

  - Empirical risk minimization that determines the hypothesis that reduces the risk $R_{emp}(h)$.
  - The loss function used was squared loss function $(h(x_i) - y_i)^2$
  - Used a learning rate of 0.2
- Gradient tree boosting algorithm [5]
  - Ml.Net implementation of gradient tree boosting.
  - Learns an ensemble of regression trees.
  - Number of trees used were 100, number of leaves 20 and learning rate of 0.2
- Clustering by K-means and training each cluster independently. The test and validation were run on the cluster after predicting using K-means.
  - K-means assigns each data point to the nearest cluster.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \le \left\| x_p - m_j^{(t)} \right\|^2 \forall j, 1 \le j \le k \right\}$$

$$m_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{x_{i \in S_i^{(t)}}} x_j$$

  - Ran the Gradient tree boosting trainer which gave the best results.

5. Results and Discussion:

Trainer	R2	Normalized RMS
SDGA	0.77	0.0501
SGD	0.65	0.061
Gradient Tree boosting	0.8	0.046
Combined SDCA and Gradient tree boosting	0.8	0.046
K-means cluster 1 and Gradient tree boosting	0.78	0.09
K-means cluster 2 and Gradient tree boosting	0.56	0.13

The table above represents the cross-validation results between model prediction and actual sales price. It shows the R^2 and normalized Root Mean Square (RMS/(max price – min price)) to determine the performance of the different regression models.

Among the models, the Gradient tree boosting gave the best results with the highest R^2 and the lowest normalized root mean square error. This is likely because instead of determining parameters that minimize the loss function for all inputs, gradient tree boosting divides the input into regions and predicts and optimal value for each region. This should cause it to fit a better curve than SGD and SDGA.

6. Conclusion and future work

The objective was to determine an algorithm that best predicts home prices in a certain region. The project compares various regression techniques and measures the R^2 and normalized root mean square error to measure the performance. Future work can include [1] testing various neural network models, [2] testing against more datasets [3] including features like description, reviews and images that can help determine how attractive the property appears to the buyers.

7. Citations

[1] https://www.thebalance.com/how-accurate-are-zillow-home-estimates-1798268

[2] hYu, Hujia, and Jiafu Wu. "Real Estate Price Prediction with Regression and Classification." CS229 (Machine Learning) Final Project Reports (2016).

[3] Kalehbasti, Pouya Rezazadeh, Liubov Nikolenko, and Hoormazd Rezaei. "Airbnb Price Prediction Using Machine Learning and Sentiment Analysis." arXiv preprint arXiv:1907.12665 (2019).

[4] Rashmi, Korlakai Vinayak, and Ran Gilad-Bachrach. "DART: Dropouts meet Multiple Additive Regression Trees." AISTATS. 2015.

[5] https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.fasttree.fasttreeregressiontrainer?view=ml-dotnet

[6] https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.kmeanstrainer?view=ml-dotnet

[7] https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.sdcaregressiontrainer?view=ml-dotnet

[8] https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.onlinegradientdescenttrainer?view=ml-dotnet

[9] https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[10] https://en.wikipedia.org/wiki/Empirical_risk_minimization

[11] https://en.wikipedia.org/wiki/Gradient_boosting#Gradient_tree_boosting

[12] https://en.wikipedia.org/wiki/K-means_clustering