Neighborhood Matters

Clustering-Driven Regression for House Price Prediction

Shane Rhoads
University of Central Florida
Orlando, FL, USA
srhoads2@knights.ucf.edu

Connor Malley
University of Central Florida
Orlando, FL, USA
malleyconnor@knights.ucf.edu

Abstract

House prices are notoriously difficult to predict due to their dependence on an enormous number of factors that are difficult to express through features. Most of this challenge is due to the complex, yet massive impact of the location of the property. Simple regression models such as Linear Regression and K-Nearest Neighbors often fail utilize location data effectively, especially when considering a large geographical area. In this work, we propose a method for predicting house sale prices by extracting meaningful spatial clusters. We test our method on real property sales data from King County, USA [1], and we show that localizing the house data by clustering can significantly decrease the generalization error of the predicted house sale prices.

I. Introduction

For any prospective renter or homeowner, it is extremely difficult to analyze the true value of a property listing. Many properties are sold for far above or below the initial asking price, and it is a common frustration for potential buyers to question whether a property is over or under-valued on the market. A large part of this issue stems from how geographical location can significantly impact the value of similar houses, and in many cases there are not similar properties in an area to directly compare. Despite the complexity of determining the relationship between location and property value, a recent analysis on the impact of different factors influencing buyers decision to purchase residential properties found that location was the most important [2].

Classical machine learning models for predicting house sale prices either do not account for this, or simply include geographical location as a feature without considering the distribution of the properties in more detail. Several works have shown that housing prices have very strong spatial correlation [3], [4]. Furthermore, these correlations are not linear [3] and are nearly impossible to capture through simple feature engineering methods. We seek to improve upon house price prediction accuracy by capturing the geographical distribution of the houses with the use of spatial clustering.

Rather than training a singular regression model on the entire dataset [1], we propose to break it into multiple spatial sub-regions based on latitude and longitude information. We then preprocess and fit an independent regression model to each sub-region. This will allow each model to differentiate between the different types of properties and the relevance of features based on their location. For example, dense regions around major cities and sparser regions in rural areas will have notably different feature values, and using spatial sub-regions our model is able to more effectively capture this complex relationship between location and features. We further expand on this by showing how ensemble learning methods, such as Adaboosting and Gradient Boosting, can provide even greater performance on house price prediction through the use of spatial clustering.

II. Related Works

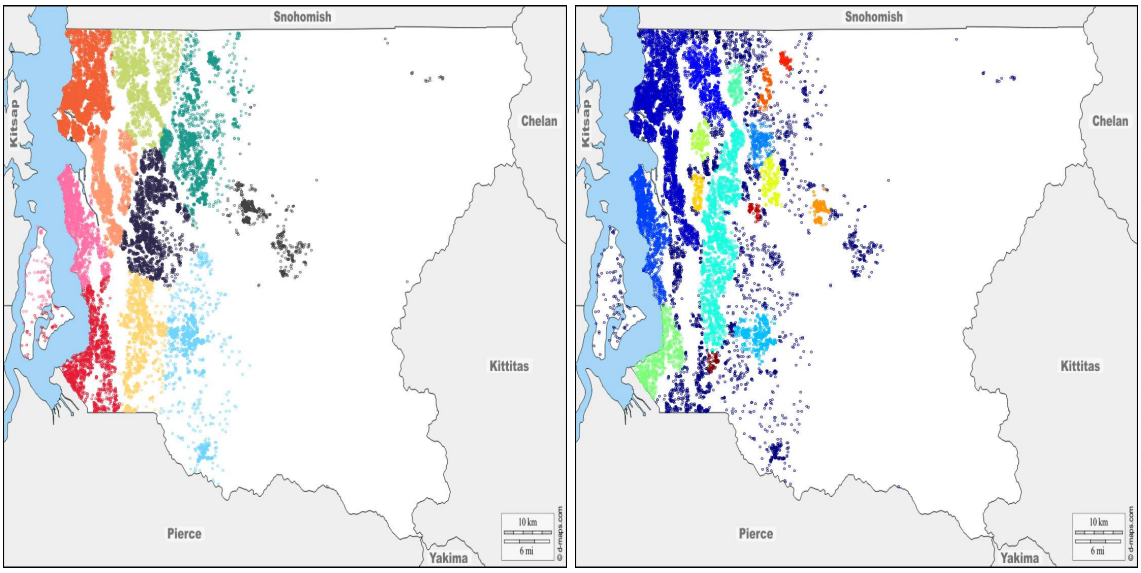
Several works in economics have shown the importance of spatial information on predicting house prices [3], [4]. The work of [3] has shown that there is strong evidence for the auto-correlation of prices by dividing larger regions of housing data into smaller submarkets for different metropolitan areas.

In [5] the authors compared several different models for predicting house prices, specifically focusing on spatial and temporal patterns. The tested models include a simple least-squared model, a local regression model, a model based on clustering demographic information, and a model based on fitting a regressor from the nearest neighbors. The last of these is effectively a 2-step k-nearest neighbor algorithm, which first finds similar datapoints and then provides a unique regressor based on them. This is quite different to our method as it computes both a grouping and model separately for each datapoint prediction. The demographic clustering model is the most effective out of the models in [5]. Their approach is the most similar to ours. However, their model generates clusters from multiple features in addition to location, and it explicitly includes nearest-neighbor residuals in the model prediction.

III. Spatial Clustering

It has been concretely established that the geographical location of a house plays a massive role in determining its value [3], [4], [6], [7]. This is due to many crucial factors being highly sensitive to location such as: average income, population density, local school ratings, availability of public services, and many others [2]. However, it is impractical to collect and directly model the effect of these features as they differ significantly by both area and person. As a result, simple models such as K-Nearest Neighbors (KNN) or Linear Regression are unable to provide precise predictions for datasets covering a large geographical area.

To solve this issue, we propose a model based on spatial clustering to separate data into different sub-regions based on latitude and longitude. The principle of our model is to assign new data points to a spatial cluster, then predict the price using a regressor specifically tailored to that cluster. This approach is geographically localized and can indirectly capture the effect of these location-sensitive features.



(a) K-Means++ (b) DBSCAN

Fig. 1. Result of latitude and longitude clustering on the dataset. Each datapoint is colored based on the cluster it is inside. In this example, k=10 for K-Means and eps=0.0175, neighbors=100 for DBSCAN.

Here, we describe the details of our implementation of the full cluster-based model framework. The framework consists of 4 stages:

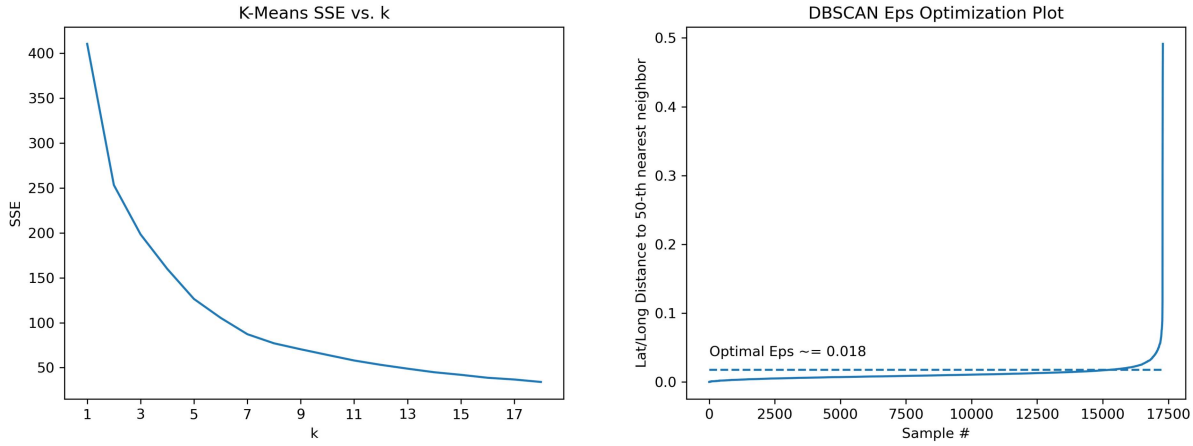
1) Generate spatial clusters
2) Preprocess each cluster's feature data and perform feature selection
3) Fit an independent regression model for each cluster
4) Predict future house prices by assigning them to a cluster and predicting with its regressor

We describe each of these stages in detail in the follow sections.

A. Spatial clustering model

For our implementation, we chose to focus on two clustering methods: K-Means++ [8] and DBSCAN. These methods are representative of center-based and density-based clustering approaches, respectively. K-Means was chosen because it is a very well-known, center-based clustering algorithm. In addition, K-Means explicitly accepts the number of clusters as an input. This allows fine control over the number of samples in each cluster and the creation of globular clusters of approximately equal size.

Alternatively, DBSCAN is a density-based clustering method and is able capture contiguous regions representing individual cites, towns, or neighborhoods. These may be missed in the globular clusters extracted by k-means. In both clustering methods, euclidean distance is used for similarity calculations. Examples of the clusters extracted by each method are shown in Figure 1.



(a) K-Means++ K Optimization

(b) DBSCAN Eps Optimization

Fig. 2. Optimization of clustering hyperparameters

For both clustering methods, it is crucial to optimize the hyperparameters to ensure consistent performance. If the clusters are not well-defined or if there are a small number of data points per cluster, insufficient information will be captured by the clustering process. This can hinder the classification performance later in the model.

1) K-means Hyperparameter tuning: For K-Means, we only need to define the number of clusters k. Intuitively, we want to select a number of clusters such that the maximum amount of relevant information is captured, while still having a significant amount of samples in each cluster. To select this parameter, we use the sum of squared errors (SSE). The SSE will inherently decrease as new clusters are added, and we iteratively test values of k and search for the clusterings where the SSE begins to decrease linearly with respect to k. In this case, we choose $k = 7$ for all experiments.

2) DBSCAN Hyperparameter tuning: Optimizing DBSCAN is more complicated as it contains multiple hyperparameters. The first parameter, eps, is the maximum distance between two points for them to be considered neighbors. The other, $min\_samples$, is the minimum number of neighboring samples that a core point must have. To select the optimal value for both parameters, we use the method described in [9]. For some value of $min\_samples$, we plot the sorted distances for every point to its k-th nearest neighbor, where $min\_samples$. Conceptually, noise points will have a much higher distance to their k-th nearest neighbor, and we choose the value of eps such that these noisy points are truncated. We define $min\_samples = 100$ and $eps = 0.0175$ for all experiments, which creates about 17 clusters. Hyperparameter optimization for both of these clustering methods is visualized in Figure 2.

B. Preprocessing and Feature Selection

Prior to feature selection, we preprocess the training data by normalizing each feature to within the range $[0, 1]$. This is performed on a per-cluster basis for each feature. This step helps to increase the variance for certain clusters.

For example, if one cluster consisted of homes with very similar square footage, normalizing the data independently allows the small variations in this feature to be highlighted within the cluster.

Another issue that may arise from the clustering is when some features become invariant. For example, the waterfront feature, which is a boolean indicating whether a house has a view of a waterfront or not, will always be false for clusters not located near a body of water. For situations such as this, the feature is removed from that cluster.

In our model, we utilize two feature selection methods, mRMR and Random Forest ranking. For both methods, we ranked all features in the dataset and selected the top features via thresholding of importance scores.

1) mRMR Feature Selection: Our mRMR implementation uses Pearson correlation for computing redundancy and F-Test for computing relevance. We take as input the number of features to select, then greedily pick them one at a time by maximizing the global optimization function. We tested both an additive and multiplicative global optimization function, and found the multiplicative approach to be slightly more effective. These correspond to the FCD and FCQ version of mRMR, respectively.

The performance of mRMR feature selection method was tested by training a KNN model using each number of features from 1 to 20 selected by mRMR. The best global features are then used for each individual cluster. In this work, the FCQ mRMR performance improved negligibly after the inclusion of 8 features.

2) Random Forest Ranking: As an alternative, we also used the Random Forest algorithm to rank the features by their impurity-based importance. This allows us to easily define a hard threshold, at which the less important features are then truncated. In this work, we define $threshold = 0.01$, which is about 11 features, though this changed slightly on each training split.

C. Model Training

As described before, the model is trained using the selected features by fitting an independent regressor to the training data from each cluster. The complete training process is elaborated by Algorithm 1.

D. Model Testing

In order to provide a prediction for a new datapoint, we must first determine the cluster of the datapoint. To do this, we compute euclidean distance between the latitude and longitude of the new point and each point in the training set. We then assign the datapoint to the cluster of the closest point from the training set. Once we have determined the cluster, we predict the price using the regressor for that cluster. This is described by Algorithm 2.

IV. Dataset overview

The dataset consists of $n = 21,613$ property sale records, taken from King County, USA (Washington) from over a period of one year from May 2014 to May 2015. Each datapoint contains 20 features and the sale price. The exact content of each feature is explained in Fig 3.

Algorithm 1: Cluster-Based Model Training

Input: List of clusters C each containing a non-empty subset of the training data D_c
Output: List of regressors R, one for each cluster

for each cluster $c_i \in$ C:
    normalize($D_(c_i)$)
    features = select_features(D_{c_i})
    data = []
    for each feature f in features:
        data.append($D_{c_i}[f]$)

    initialize regressor r_i
    r_i.fit(data)
    R.append(r_i)

Algorithm 2: Cluster-Based Model Prediction

Input: Test Datapoint D, List of clusters C each containing a non-empty subset of the training data T
Output: Predicted price of the datapoint P

min_distance = 1e6
min_datapoint = None
for each training point $T_i \in$ T:
    distance = Euclidean_distance(T_i, D)
    If distance ¡ min_distance **then**
        min_distance = distance
        min_datapoint = T_i

test_cluster = min_datapoint.cluster
P = test_cluster.predict(D)

Most of these features represent specific aspects of the individual property. However, there are also some features which contain aggregated data from other features. One example is Sqft_living15, which is the average square foot of the property's 15 nearest neighbors.

Variable	Description
Id	Unique ID for each home sold
Date	Date of the home sale
Price	Price of each home sold
Bedrooms	Number of bedrooms
Bathrooms	Number of bathrooms, where .5 accounts for a room with a toilet but no shower
Sqft_living	Square footage of the apartments interior living space
Sqft_lot	Square footage of the land space
Floors	Number of floors
Waterfront	A dummy variable for whether the apartment was overlooking the waterfront or not
View	An index from 0 to 4 of how good the view of the property was
Condition	An index from 1 to 5 on the condition of the apartment,
Grade	An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design
Sqft_above	The square footage of the interior housing space that is above ground level
Sqft_basement	The square footage of the interior housing space that is below ground level
Yr_built	The year the house was initially built
Yr_renovated	The year of the house's last renovation
Zipcode	What zipcode area the house is in
Lat	Lattitude
Long	Longitude
Sqft_living15	The square footage of interior housing living space for the nearest 15 neighbors
Sqft_lot15	The square footage of the land lots of the nearest 15 neighbors

Fig. 3. Explanation of dataset features [1]

V. Experimentation

In order to examine the overall effectiveness of our clustering model, we tested the performance using a wide range of classical and ensemble regression algorithms: Linear Regression, 2nd-order Polynomial Regression, K-Nearest Neighbors, Random Forest Regression, Gradient Boosting, Adaboost, and XGBoost [10]. For each algorithm, we also tested each combination of feature selection and clustering method as described in Section III. Each of these are evaluated using 5-fold cross validation ($n_{train} = 17,290 \; and \; n_{test} = 4,323$).

A. Baseline methods

For our baseline, we placed the entire training dataset into a single cluster, which is equivalent to performing no clustering. This allows us to directly observe how our more localized, clustering models perform in comparison to a traditional global model, regardless of the underlying regressor that is used.

VI. Results

A summary of our results can be seen in Tables 1 through 6. Tables 1 to 3 show the results from using mRMR to perform feature selection and tables 4 to 6 show the results from using Random Forest feature selection. With either method, it is clear that the cluster-based model offers a notable improvement to every tested regression method.

Results with mRMR Feature Selection

Regressor	R^2	RMSE (USD)
KNN	0.775	$1.737 \cdot 10^5$
Linear Regression	0.651	$2.165 \cdot 10^5$
Polynomial Regression	0.722	$1.927 \cdot 10^5$
Adaboost	0.518	$2.536 \cdot 10^5$
Gradient Boosting	0.808	$1.601 \cdot 10^5$
Random Forest	0.808	$1.603 \cdot 10^5$
Decision Tree	0.617	$2.26 \cdot 10^5$
XGBoost	0.81	$1.596 \cdot 10^5$

TABLE 1. Baseline Results

Regressor	R^2	RMSE (USD)
KNN	0.812	$1.589 \cdot 10^5$
Linear Regression	0.772	$1.752 \cdot 10^5$
Polynomial Regression	0.808	$1.603 \cdot 10^5$
Adaboost	0.721	$1.933 \cdot 10^5$
Gradient Boosting	0.831	$1.504 \cdot 10^5$
Random Forest	0.83	$1.509 \cdot 10^5$
Decision Tree	0.688	$2.041 \cdot 10^5$
XGBoost	0.836	$1.483 \cdot 10^5$

TABLE 2. Kmeans Results

Regressor	R^2	RMSE (USD)
KNN	0.827	$1.525 \cdot 10^5$
Linear Regression	0.766	$1.771 \cdot 10^5$
Polynomial Regression	0.787	$1.649 \cdot 10^5$
Adaboost	0.753	$1.818 \cdot 10^5$
Gradient Boosting	0.844	$1.445 \cdot 10^5$
Random Forest	0.848	$1.428 \cdot 10^5$
Decision Tree	0.715	$1.95 \cdot 10^5$
XGBoost	0.856	$1.389 \cdot 10^5$

TABLE 3. DBSCAN Results

Results with Random Forest Feature Selection

Regressor	R^2	RMSE (USD)
KNN	0.833	$1.497 \cdot 10^5$
Linear Regression	0.682	$2.068 \cdot 10^5$
Polynomial Regression	0.752	$1.82 \cdot 10^5$
Adaboost	0.588	$2.346 \cdot 10^5$
Gradient Boosting	0.885	$1.242 \cdot 10^5$
Random Forest	0.878	$1.281 \cdot 10^5$
Decision Tree	0.74	$1.862 \cdot 10^5$
XGBoost	0.885	$1.242 \cdot 10^5$

TABLE 4. Baseline Results

Regressor	R^2	RMSE (USD)
KNN	0.851	$1.414 \cdot 10^5$
Linear Regression	0.8	$1.64 \cdot 10^5$
Polynomial Regression	0.841	$1.435 \cdot 10^5$
Adaboost	0.77	$1.757 \cdot 10^5$
Gradient Boosting	0.89	$1.213 \cdot 10^5$
Random Forest	0.879	$1.274 \cdot 10^5$
Decision Tree	0.751	$1.828 \cdot 10^5$
XGBoost	0.889	$1.221 \cdot 10^5$

TABLE 5. Kmeans Results

Regressor	R^2	RMSE (USD)
KNN	0.85	$1.419 \cdot 10^5$
Linear Regression	0.791	$1.674 \cdot 10^5$
Polynomial Regression	0.798	$1.67 \cdot 10^5$
Adaboost	0.771	$1.752 \cdot 10^5$
Gradient Boosting	0.874	$1.296 \cdot 10^5$
Random Forest	0.878	$1.321 \cdot 10^5$
Decision Tree	0.734	$1.884 \cdot 10^5$
XGBoost	0.881	$1.262 \cdot 10^5$

TABLE 6. DBSCAN Results

A. Impact of clustering method

We find that both DBSCAN and K-Means clustering improves the overall performance of the regression models. Interestingly, we find that DBSCAN clustering performs best with mRMR feature selection while K-Means clustering performs best with Random Forest feature selection.

B. Comparison with Baselines

From Tables 1 - 6, we see that Adaboost achieves the largest decrease in error between the baseline model and the cluster-based model, with $\Delta R^2 \approx 0.20$ in most cases. Linear Regression showed the second largest increase with $\Delta R^2 \approx 0.12$. The 2nd-order Polynomial Regression achieved $\Delta R^2 \approx 0.07$, though this fluctuated more than other methods due the higher dimensionality of the polynomial features and the lower sample sizes in some clusters. The KNN model showed surprisingly good baseline results, and still exhibited an improvement of $\Delta R^2 \approx 0.025$. A simple impurity-based Decision Tree also displayed improvement with $\Delta R^2 \approx 0.05$.

Compared to the baseline, all of the tested ensemble methods showed slightly better performance of $\Delta R^2 \approx 0.02$ when using the mRMR selected features. Interestingly, these same models also displayed a much smaller increase in performance when using the Random Forest selected features. Nevertheless, Random Forest, Gradient Boosting, and XGBoost displayed the overall best performance on average. The Gradient Boosting model with Random Forest feature selection and K-Means clustering had the best performance in any case, with $R^2 \approx 0.890$ and $RMSE \approx \$121,300$. From table 5, it is clear that Random Forest and XGBoost performed on par Gradient Boosting.

C. Impact of feature selection

In general, the ensemble regressors performed significantly better when utilizing the Random Forest selected features. This is due to the impurity-based ensemble regressors being better suited for use with impurity-based feature selection. In fact, most of these ensemble methods showed a more significant improvement from the addition of impurity-based ranking than from the addition of the cluster-based model.

Interestingly, combining the two methods resulted in slightly higher errors in a few clusters. This is likely attributed to a significant decrease in the sample size as a result of the spatial clustering, and could likely be improved with further tuning. See below for the most common features selected, and ranked, by mRMR and Random Forest.

Selected Features

mRMR: *sqft_living, grade, lat, waterfront, yr_renovated, sqft_above, sqft_living15, condition*

Random Forest: *grade, sqft_living, lat, long, waterfront, sqft_living15, yr_built, sqf_above, sqft_lot15, view, sqft_lot*

VII. Conclusions and Future Work

In this work, we have shown that the spatial clustering of large housing datasets can be utilized to improve performance on a wide range of regression algorithms. This is accomplished by generating spatial sub-regions that provide greater feature variances and location-specific feature valuation.

We have also shown that the robust ensemble methods such as XGBoost, Gradient Boosting and Random Forest, which provide the best predictions in most cases, are the most challenging to improve with clustering.

For future work, our spatial clustering method can be extended to datasets with much larger sample size. This would enable denser clusters and a general increase in performance. Furthermore, covering a larger geographical region may would allow us to explore the use of nested clusters and regressors, which could further improve performance. It is also possible to apply these same clustering techniques to incorporate other, non-spatial feautres. This could lead to the creation of more informative clusters based on specific features, such as public services or demographic information.

References

[1] A. Alsaqri, S. Inturi, P. Shivhare, S. Singhania, and K. T. Vinayagam, "King county house prices prediction model," 2017. [Online]. Available: https://www2.slideshare.net/PawanShivhare1/predicting-king-county-house-prices

[2] D. Rachmawati, S. Shukri, S. Azam, and A. Khatibi, "Factors influencing customers' purchase decision of residential property in selangor, malaysia," *Management Science Letters*, vol. 9, pp. 1341–1348, 2019.

[3] S. Basu and T. Thibodeau, "Analysis of spatial autocorrelation in house prices," *The Journal of Real Estate Finance and Economics*, vol. 17, pp. 61—85, 1998.

[4] X. Liu, "Spatial and temporal dependence in house price prediction," *The Journal of Real Estate Finance and Economics*, vol. 47, pp. 341—-3695, 2013.

[5] B. Case, R. Clapp, J.and Dubin, and M. Rodriguez, "Modeling spatial and temporal house price patterns: A comparison of four models," *The Journal of Real Estate Finance and Economics*, vol. 29, pp. 167—191, 2004.

[6] y. Tu, H. Sun, and S. Yu, "Spatial autocorrelations and urban housing market segmentation," *The Journal of Real Estate Finance and Economics*, vol. 34, pp. 385–406, 2007.

[7] S. Bourassa, E. Cantoni, and M. Hoesli, "Predicting house prices with spatial dependence: A comparison of alternative methods," *Journal of Real Estate Research*, vol. 32, pp. 139–160, 2010.

[8] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," June 2006. [Online]. Available: http://ilpubs.stanford.edu:8090/778/

[9] M. Elbatta, R. Bolbol, and W. Ashour, "A vibration method for discovering density varied clusters," *ISRN Artificial Intelligence*, vol. 2012, 01 2012.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785