Housing ML Model: Implementing K-Means Clustering To Improve Housing Affordability

By: Sanjana Srikanth

In this project, I wanted to focus on building a model to address the housing unaffordability crisis in current real estate markets. With real estate prices skyrocketing in states across America, I wanted to look at a dataset describing real estate statistics from a specific state and apply machine learning to gather information from that dataset to better understand from the data how we can address housing inequalities in the area. The dataset I chose (from data.gov) detailed many different towns in Connecticut mapped to critical housing statistics including:

Year: when the statistics were recorded; **Town code**: a unique code associated with every town for identification; **Town:** the name of the specific town in Connecticut; **Census units:** the number of census units within each town (geographically separated, smaller zones for the purposes of census recordings); **Government assisted**: the number of housing programs that receive government assistance; **Tenant rental assistance**: the number of tenants receiving rental assistance; **Single family/mortgages**: number of single family mortgages issued to make housing affordable for families; **Deed restricted units**: number of units that safeguard the long-term value of the home often by restricting the resale of houses to maintain price affordability; **Total assisted units**: the sum of total government, tenant, and deed restricted units; **Percent affordability**: a metric describing the percent of housing units in the town that are considered affordable

Therefore, based on these factors, the focus question I created for this model is: "How can towns in Connecticut be grouped based on the availability and types of affordable housing assistance, and what insights can this provide for housing policy?

To address this question, I decided to go with the K-Means Clustering algorithm, exploring how I can use the K-Means algorithm to create clusters of towns in Connecticut that describe varying zones of affordability. To implement this model, I first imported the data set and applied various data cleaning steps to ensure the data was primed for modeling. Such data cleaning steps included getting rid of duplicate values, displaying the number of missing values and converting them to NaN (for standardization), and getting rid of leading/trailing spaces in column titles (this was a problem in the original dataset for some of the columns that threw small, tedious errors).

To implement the actual K-Means clustering algorithm, I identified the relevant factors for clustering, which was every statistic excluding the total number of census units (which doesn't offer as much insight into housing affordability). This would also allow me to narrow down how I can plot my final clusters, which would be based on the number of total assisted units v. the percent affordability. After isolating the relevant factors, I utilized the Elbow Method to identify the optimal number of clusters I should use. The Elbow Method is a way of obtaining the optimal number of clusters (k) by plotting the number of clusters v. their respective wcss scores and identifying the "elbow" point on the graph (the point where the graph experiences a relatively sharp change in decrease). To do this, I iterated through a set number of clusters (1-10 clusters) and calculated the wcss scores for each. Plotting this, I identified that the "elbow" of my graph was 4, making 4 the optimal number of clusters I sought to create through the rest of the model. Before fitting the K-Means algorithm to my dataset, I standardized the mean (to be centered around 0) and standard deviation (to be centered around 1) for each factor to ensure that each data attribute contributed equally to the distance calculations (from each data point to the centroid) that would be used to create the clusters. If one factor contributed more to the creation of the clusters than the other, it would significantly skew the resulting clusters towards one attribute. Instead, we want the clusters at the end to form zones of affordability taking into account equal parts of all factors affecting housing affordability. After applying the K-Means

algorithm, I then finally created a column called ‘Cluster’ that would map each point in the DataFrame to its assigned cluster and display the four final clusters on a graph for visible results.

The final graph showed clusters that were formed nicely and were distinct from each other. This graph of the four resulting clusters from my model, however, answers only the first part of my focus question. To answer the second part of my question, what insights we can draw from these 4 clusters about housing policy, I created an inference report that is printed at the end of the model that calculates and displays some key insights. Namely, the model identified the cluster with the lowest percentage affordability for housing, therefore identifying this cluster as an area future housing policy and economic equality initiatives can be directed towards. Furthermore, the model identified the cluster with the least amount of government assisted units; therefore, government programs could ideally target the towns in that cluster to make a better impact. These insights reveal how applying K-Means clustering algorithms to real world problems can be critical in identifying areas of need through the creation of clusters to provide targeted assistance and create stronger positive social impacts. In this case, this K-means based machine learning model was able to create clusters of towns based on housing data to identify the strongest zones that can be targeted to effectively sustain more affordable housing in the state of Connecticut.

Finally, and most importantly, I applied two key evaluation metrics to assess the performance of my model. To assess the implementation of K-Means clustering, I looked at the final wcss score and the final silhouette score. The wcss score (or inertia) is the sum of the distances of all the data points within a cluster to its centroid. Essentially, it measures the compactness of a cluster. A lower wcss score means a better defined cluster where the points are closer to each other. The final wcss score for my model was around 2685, which although stands to be high on its own, was lower relative to the results I received from testing a different k number of clusters other than 4. A stronger metric of the performance of my model though was the silhouette score. The silhouette score evaluates the clustering quality and how distinctly the clusters were formed. A score of 1 is a perfect fit (which ideally shouldn’’ happen because these models are not perfect, while less than 0.26 means there is no identifiable structure. Most importantly, a score between 0.7-1 indicates a strong structure. The final silhouette score for my model was 0.71, indicating a strong clustering quality. This score combined with the wcss score speaks to the strength of this K-Means clustering model and the results it provided, ultimately establishing great effectiveness and performance.

Sources:

Affordable Housing in Connecticut dataset from data.gov:
https://catalog.data.gov/dataset/affordable-housing-by-town-2011-present

Understanding the K-means clustering algorithm and Evaluation Metrics:
https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/#:~:text=The%20silhouette%20score%20and%20plot,these%20scores%20for%20each%20sample.

What K-Means clustering is:
https://medium.com/@tiami.abiola/clustering-wcss-and-elbow-method-427db8968ba1

Scikit K-Means documentation:
https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.KMeans.html

What is the function of Random State in K-Means:
https://kishanmodasiya.medium.com/what-the-heck-is-random-state-24a7a8389f3d

What the Elbow Method is:
https://medium.com/@zalarushirajsinh07/the-elbow-method-finding-the-optimal-number-of-clusters-d297f5aeb189#:~:text=The%20Elbow%20Method%20is%20a,points%20and%20their%20cluster%20center.

How fit(), transform(), and fit_transform() work and their differences:
https://medium.com/@swarnpriyaswarn/fit-transform-fit-transform-predict-the-codes-which-does-it-all-c1f4cc24a4da

Scikit documentation for Standard Scalar:
https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html

Understanding steps to clean data:
https://towardsdatascience.com/so-youve-got-a-dataset-here-s-how-you-clean-it-5d0b04a2ed86