Overview:

I did not get to create the two apps I wanted because the website would not let me upload/use the data I was giving it. I tried multiple ways to upload this data, but none of them worked. What is strange is that when I was experimenting previously in the semester, I had no issues. I didn't change the file format at all; the only change was that the data I was trying to use was a smaller subset. Furthermore, the file I had uploaded previously in the semester (that worked) now does not want to work either. I've checked my code multiple times, and nothing appears to have been corrupted in the writing process. Since my data doesn't want to load, I will summarize what both of my apps are supposed to do.

Data Summary:

Number of Observations: 1,499,214

Date Range: 2001-2024

Variable List:

[1] "APN..PARCEL.NUMBER.UNFORMATTED."

[2] "APN.SEQUENCE.NUMBER"

[3] "ONLINE.FORMATTED.PARCEL.ID"

[4] "PROPERTY.INDICATOR.CODE...STATIC"

[5] "DEED.SITUS.HOUSE.NUMBER...STATIC"

[6] "DEED.SITUS.HOUSE.NUMBER.SUFFIX...STATIC"

[7] "DEED.SITUS.HOUSE.NUMBER.2...STATIC"

[8] "DEED.SITUS.DIRECTION...STATIC"

[9] "DEED.SITUS.STREET.NAME...STATIC"

[10] "DEED.SITUS.MODE...STATIC"

[11] "DEED.SITUS.QUADRANT...STATIC"

[12] "DEED.SITUS.UNIT.NUMBER...STATIC"

[13] "DEED.SITUS.CITY...STATIC"

[14] "DEED.SITUS.STATE...STATIC"

[15] "DEED.SITUS.ZIP.CODE...STATIC"

[16] "DEED.SITUS.COUNTY...STATIC"

[17] "DEED.SITUS.STREET.ADDRESS...STATIC"

[18] "DEED.SITUS.UNIT.NUMBER...STATIC.1"

[19] "SALE.AMOUNT"

[20] "SALE.DERIVED.DATE"

[21] "PRIMARY.CATEGORY.CODE"

[22] "DEED.CATEGORY.TYPE.CODE"

[23] "SALE.DOCUMENT.TYPE.CODE"

[24] "CASH.PURCHASE.INDICATOR"

[25] "MORTGAGE.PURCHASE.INDICATOR"

[26] "INTERFAMILY.RELATED.INDICATOR"

[27] "INVESTOR.PURCHASE.INDICATOR"

[28] "RESALE.INDICATOR"

[29] "NEW.CONSTRUCTION.INDICATOR"

[30] "SHORT.SALE.INDICATOR"

[31] "FORECLOSURE.REO.INDICATOR"

[32] "FORECLOSURE.REO.SALE.INDICATOR"

[33] "PARCEL.LEVEL.LATITUDE"

[34] "PARCEL.LEVEL.LONGITUDE"

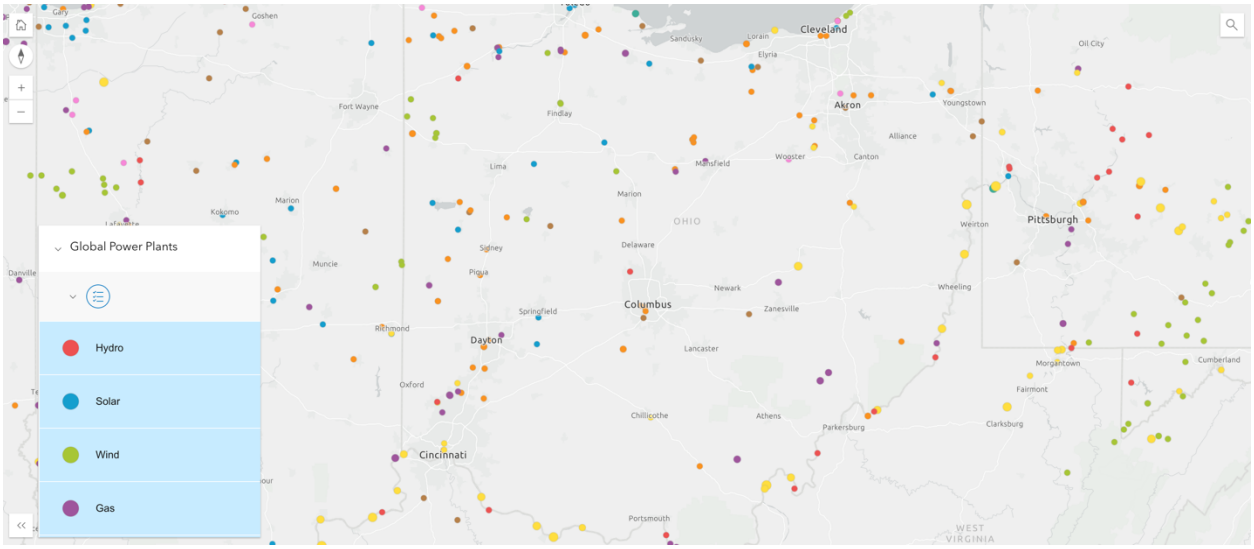
[35] "PREDICTED.SALE.AMOUNT"

The data I was going to use was a subset of Core Logic data purchased for a research project. The subset specifically comes from a set of data that details transfer information (this will be important later). It covers all of Franklin County and includes all property types. Importantly, the data has what is referred to (sometimes) as a “many-to-many” relationship. This means some parcels have multiple observations (i.e., sales). Variables 1-3 are three different parcel number formats. Why didn’t I just choose one? Because everyone formats data differently, and I had planned to use the parcel numbers to cross-reference other data sets. For those who may not know, your parcel number is unique. Variables 4-18 are all the chosen geospatial information about the property. Variables 19-20 are fairly straightforward; this is the observed sale price and date. Variables 21-32 are all the indicator variables I chose (more on this later). Variables 33-34 are self-explanatory. They’re just another way to identify property locations within ArcGIS. Finally, variable 35 is a variable I created that stores the predicted sale price (more on this later).

App 1: Property Transaction Indicator

The first app I was going to create would use the indicator variables (21-32). Note: variables 19-32 were all the variables that would be visible. It was going to use an interactive legend so users could quickly filter through information (see image below). Additionally, it was going to include an option to filter by price ranges. I wanted to create this app so people could gain multiple insights into home sales. For example, take Zillow; if you filter by sold properties, you cannot get more information beyond when it was sold and the sale price. In contrast, my app would give you much more information about the sale. This would allow users to see what types of sales are happening and where they are most predominant.

Interactive Legend Example:



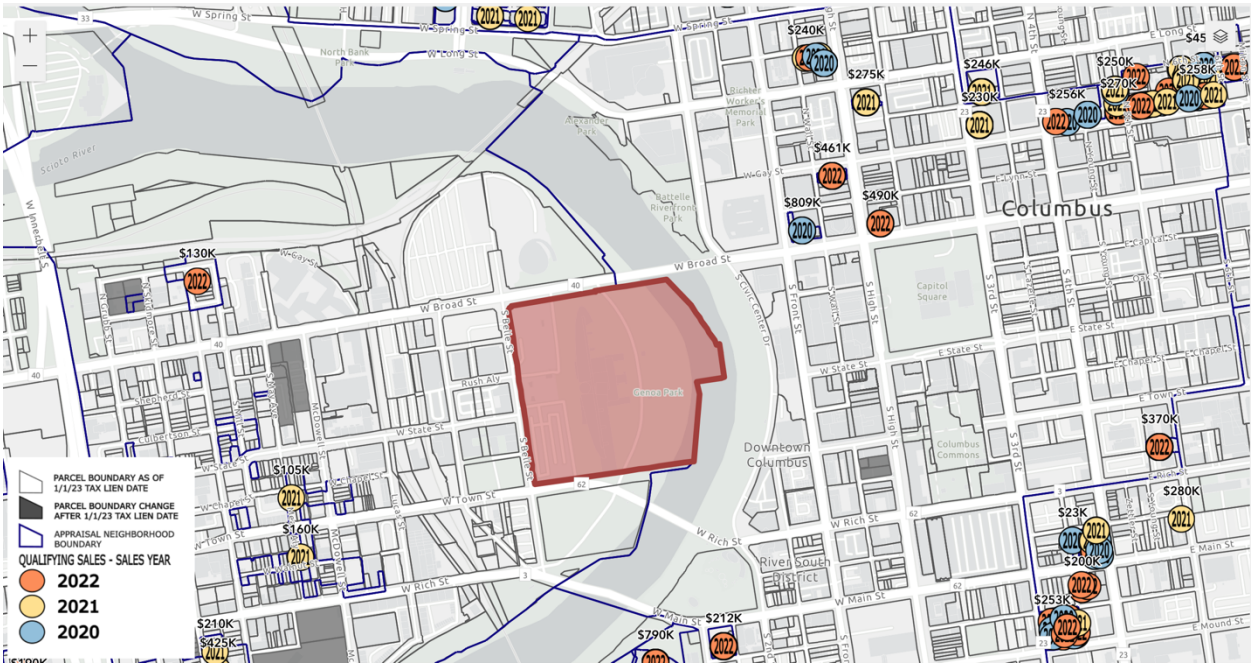
You may have noticed that the data above doesn't include property characteristics (e.g., beds, baths, square footage, etc.). This is on purpose because I had planned to use a data set from the Franklin County Auditor's open data repository (link below). It outlines all the parcel boundaries and includes basic property characteristics. Hence, I only needed to include this as my base layer for all the basic property information (the same method is used for app 2).

Parcel Boundaries Data:

https://auditor-fca.opendata.arcgis.com/maps/3b1c75018fca40c8b24083fd48197e74/about

App 2: Predicted House Values

The second app was going to use the predicted house price variable (35). Note: variables 19, 20, and 35 were all the variables that would be visible. For context, the Franklin County auditor runs a site where you can see your estimated home value (see image below). There is not much information on how they got to these estimates, but it appears it uses some sort of weighted comparative analysis. My evidence for this is limited, but if you look at the image, you can see the "Qualifying Sales" around the subject property (for those interested, that's COSI). I don't know why they'd include this if they're not using those to arrive at some estimate. Additionally, if you look at the surrounding sales values (see the link below if it is difficult to see), the estimate is clearly not derived from a simple average.



Property Link: https://auditor-fca.opendata.arcgis.com/maps/3b1c75018fca40c8b24083fd48197e74/about

Getting to my app, notice that the "Qualifying Sales" only range from 2020-2022, implying that the property's value comes from old sale data. I'm not here to critique the Franklin County Auditors' forecasting methods, but this raises a few issues. Firstly, it is 2024, and they claim these values represent 2023 home values (i.e., outdated). Secondly, if these are 2023 home values, why are they using only 2020-2022 home sales? My app was going to fix these issues by providing a much more accurate estimate of the home value. Additionally, it would provide the fair market value rather than the appraised value. Which no one actually cares about (I'm just joking).

My model forecasted home prices using a supervised machine-learning algorithm. The model I used is the hybrid particle swarm optimization (PSO) support vector machine (SVM) model or the hybrid PSO-SVM model. Without getting too complicated, SVM is a classification method that attempts to find a hyperplane that best partitions classes of points. However, SVM only works when data is linearly separable. An easy solution is to apply a non-linear transformation to the data before using SVM. For example, we could transform 1-dimensional data into 2 dimensions. This works in relatively low dimensions, but scaling into higher dimensions significantly increases computational requirements. Scaling into higher dimensions is almost always necessary since it affects how sophisticated the decision boundary is (i.e., the support vector classifier). Additionally, the transformation required is not always known. Therefore, a kernel function is used to solve these problems simultaneously. The kernel function calculates the high-dimensional relationship of data but without doing the transformation (I think that's pretty cool). Subsequently, the kernel function allows SVM to scale into higher dimensions without the computational strain of doing the transformations. Mathematically, the implied transformation is often denoted $\phi(\cdot)$ such that $\phi : \mathbb{R}^n \to \mathcal{D}$, $x_i \in \mathbb{R}^n$, where $\mathcal{D}$ is the dimension of the feature space and x_i is some feature vector. Different kernel functions can be chosen depending on the regression one wants to construct. I used the Gaussian kernel or radial basis function (RBF) since it can project into an infinitely high-dimensional feature space. The Gaussian kernel is defined as follows:

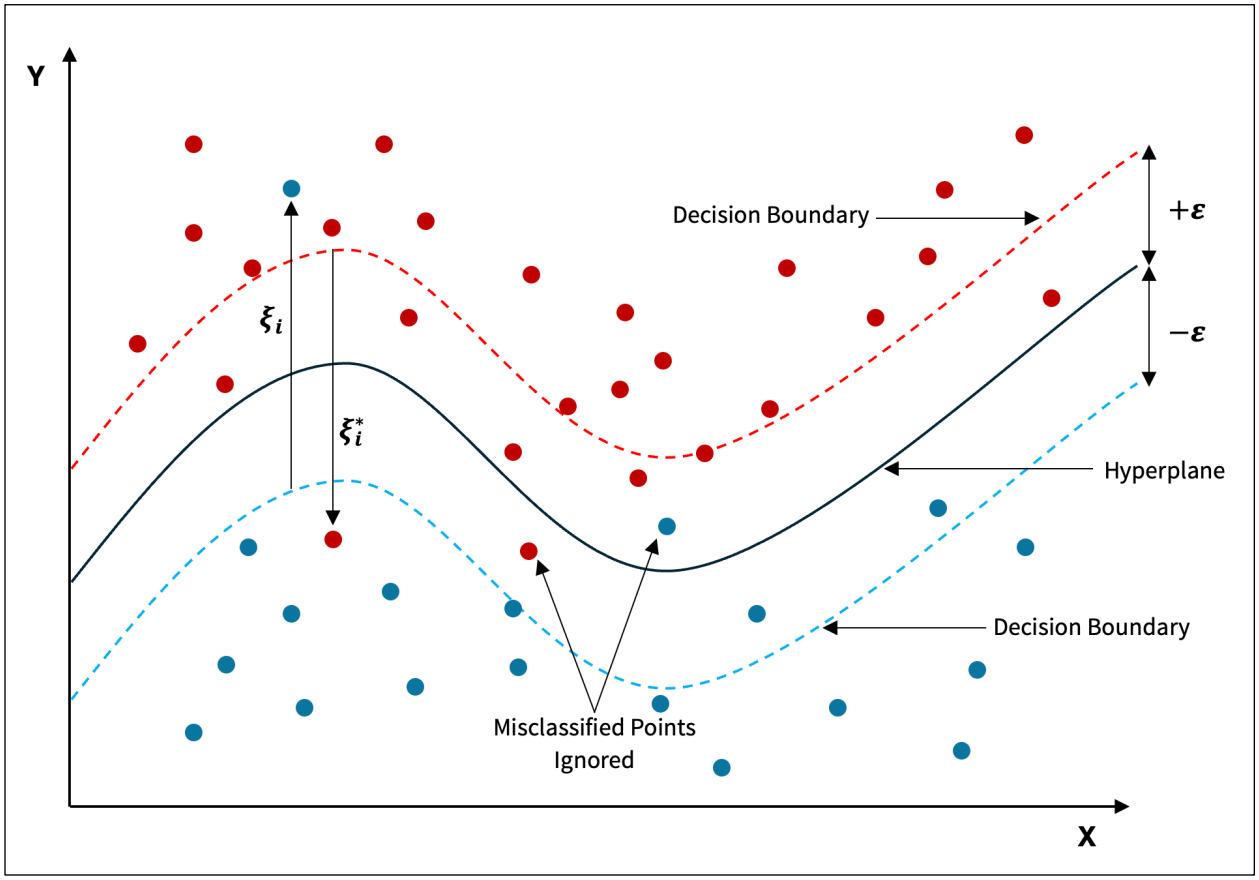
$$\mathcal{K}(x, x_i) = exp[-\gamma \|x - x_i\|^2] \ , \ \ \gamma = \frac{1}{2\sigma^2}$$

Furthermore, in the context of SVM regression, we define the following optimization problem:

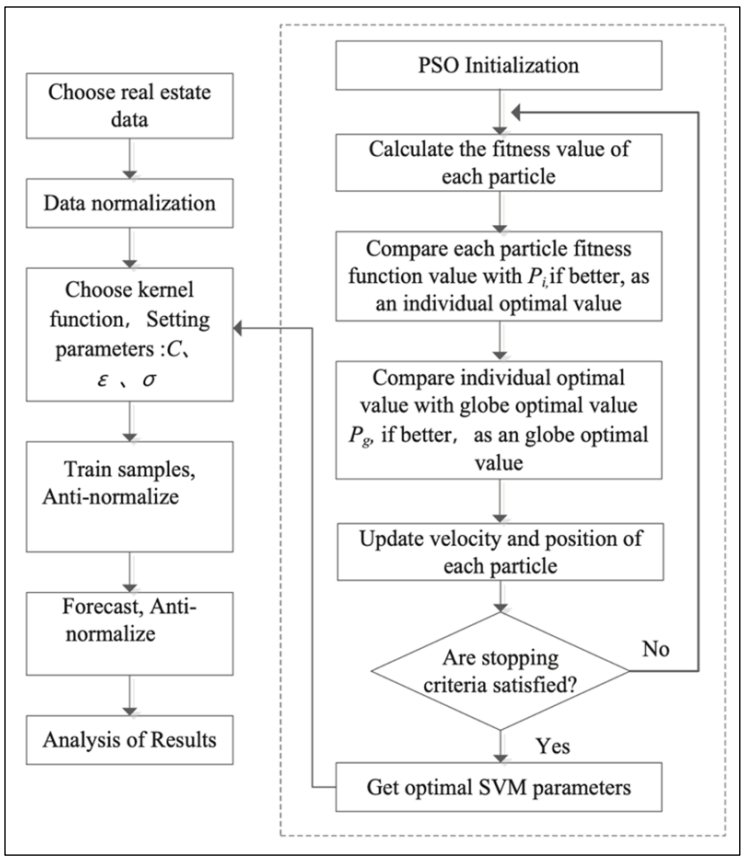
$$\min_{\omega, b, \xi^*} \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

$$\text{subject to} \begin{cases} (\omega \cdot x_i) + b - y_i \leq \varepsilon + \ \xi_i \ , & i = 1, \dots, m \\ y_i - (\omega \cdot x_i) - b \leq \varepsilon + \ \xi_i^* \ , & i = 1, \dots, m \\ \xi_i^*, \xi_i \geq 0 \ , & i = 1, \dots, m \end{cases}$$

where ξ_i and ξ_i^* are slack variables, C is the punishment coefficient, and ε is the error insensitive loss function (i.e., soft margin). The slack variables measure the degree of misclassification (error) and ensure that the soft margin constraint is satisfied for all points. The punishment coefficient controls the magnitude of the penalty placed on points larger than ε. The error-insensitive loss function defines the soft margin (tube) size where misclassification is allowed. This leads to better overall model performance. Finally, the parameter σ is the standard deviation of the Gaussian kernel and controls the complexity of the decision boundary. The objective is to maximize the margin while minimizing the amount of misclassification. Since the radial basis function (RBF) kernel can project into an infinite-dimensional space, visualization is impossible. Therefore, I made a diagram that shows a simplified example of what a support vector regression may look like using a polynomial kernel (see image below). The red and blue circles represent two arbitrary classes of points.



Lastly, particle swarm optimization (PSO) is an optimization method developed by Eberhart and Kennedy (1995). It is inspired by the social behavior patterns observed in bird flocking or fish schooling. A group of particles (a particle swarm) is initialized, where each particle's position represents a potential solution. Each particle's position is evaluated using a fitness function, which I used the mean absolute percentage error (MAPE). Particles adjust their positions based on their personal best solution (P_i) and the global best solution (P_g). The algorithm repeats this evaluation and updating process until the maximum number of iterations is reached or the fitness value is below the error threshold (see image below). I used the PSO algorithm to tune the SVM hyperparameters C, ε, and σ. My results yielded a MAPE of 0.096%. Therefore, the app was going to be far more accurate than the auditors' site. Also, I think this is obvious, but the data set I mentioned above is not the one I used to make these predictions. The set I used to make these predictions is much larger and has many more variables. For perspective, it took two weeks to train the model (I did have to stop/start it multiple times since I needed my computer for school) and a further 7-8 hours to calculate predictions (that was able to finish overnight).


Choose real estate data
Data normalization
Choose kernel function, Setting parameters :C、ε 、σ
Train samples, Anti-normalize
Forecast, Anti-normalize
Analysis of Results
PSO Initialization
Calculate the fitness value of each particle
Compare each particle fitness function value with P_i if better, as an individual optimal value
Compare individual optimal value with globe optimal value P_g, if better, as an globe optimal value
Update velocity and position of each particle
Are stopping criteria satisfied?
No
Yes
Get optimal SVM parameters