Isophilly: Predicting Philadelphia Real-Estate Values through Aggregating Indicators on an Isochrome Level

Benjamin Knight
University of Maryland
bknight5@umd.edu

Om Duggineni
University of Maryland
odo@terpmail.umd.edu

Jack Doggett
University of Maryland
jdogget1@terpmail.umd.edu

May 13, 2025

1 Abstract - Project Description

This project aims to introduce techniques for Isochrone-based Real Estate regression analysis through a novel visualization, providing analysis of transportation access - including public transportation, walkability, and driving - and the relationship it has on real estate and social impacts to census block groups in Philadelphia County, Pennsylvania. Our visualization runs a simple linear regression for each selected public transport type and isochrone distance using a diverse dataset including transportation access and economic indicators such as assessed property values, foreclosure rates, local business ratings, and other economic factors. Public census and geographic datasets are used to allow users to explore relationships between transportation access and key financial statistics, with an emphasis on showing interactions between metro and bus connections within an area and indicators related to economic activity in a neighborhood. Regressions are displayed in a table for easy comparison and features can be easily selected and visualized through our dashboard.

2 Intended Audience

Our intended audience comprises primarily of financial researchers interested in Real Estate and secondary people looking for broad-level insights on real estate trends in Philadelphia. Our tool is primarily designed to serve as an exploration on the impacts of regressing real estate features based on isochrone-level regions and to provide our insights and a starting point for future researchers in this

area to explore new ideas in Real Estate modeling. Our tool assumes basic knowledge of regressions and statistics and is most helpful to an audience that has experience in modeling and statistical analysis. However, we provide simple and flexible tools that are accessible to users of all types in interacting with our datasets. The features we use, while primarily designed for research, are also useful in the process of exploring the value and features of neighborhoods for potential renters and homebuyers.

3 Project Goals

Goal 1: Calculate isochrone regions for each census group region in Philadelphia. We decided for each census group to calculate 45 isochrones as follows:

1. 5 positions (Center, North, South, East, West).
2. 3 modes of transportation (Walking, Public Transit, Car)
3. 3 distances (10, 20, 30 minutes)

For each census block group, 45 isochrones would be calculated, a total of 61470 ischrones. GraphHopper, an open-source routing library and server, would be used to calculate the isochrones. OpenStreetMap and public transit data would be used with GraphHopper to calculate isochrones. A python script would make REST API calls to GraphHopper for all isochrones and output a GeoJSON file.

Goal 2: Build a custom dataset consisting of the below data points, each collected at the census block level, for all census block groups in the region surrounding Philadelphia, PA. We picked this region in order to maximize the amount of data we would have available for the project.

Data points:

1. The above isochrone data for each Census Block Group.
2. The below features for each isochrone combination and Census Block Group:
   (a) Average assessed property value.
   (b) Total foreclosure rate.
   (c) Total foreclosure rate divided by area.
   (d) Yelp business review ratings.
   (e) Neighborhood neglected status.
   (f) Region area.
   (g) Total mileage of streets prohibiting trucks.
   (h) Distance to nearest public pool.

This data is then sorted and aggregated for each geographic resolution that we are analyzing. (For each census block group and for each of the 9 isochrone resolutions.)

The specific datasets that we plan on using are described in the data section.

Goal 3: Run a simple linear regression with average assessed property value as the dependent variable for each isochrone group and on the original Census Block groups for easy comparisons between each grouping.

Goal 4: Design an interactive website to display the following information in an interactive leaflet choropleth:

1. Selectable census block groups on a map of the city of Philadelphia.
2. Toggles to switch between various isochrone modes and distances.
3. Toggles to display individual isochrone regions for orientation above a selected census block group.
4. Selectable choropleth shading and value displays of each feature we will incorporate in our analysis.
5. All regressions displayed in a grid to allow comparisons on which areas yielded the strongest regressions.

4 Datasets

4.1 OpenStreetMap

4.1.1 Source

OpenStreetMap is a collaborative mapping project where individuals and groups with data on a particular location can aggregate their data and combine it with other publicly available geographical data to create a mapping dataset containing the locations of roads, businesses, and other entities. The data in OpenStreetMap is considered open data and is free for commercial use under the Open Database License (OdBL). Notably, OpenStreetMap can be processed by freely available routing tools capable of calculating the travel time between two locations given data on the method of transportation used and the contents of OpenStreetMap's road network.

4.1.2 Data Points

Since OpenStreetMap is composed of mapping data consisting of locations and the roads between them, as well as information about things such as landmasses, countries, and territories, it isn't necessarily quite as easy to make a determination of what a "data point" is in OpenStreetMap, since it isn't a traditional tabular dataset.

OpenStreetMap is avaliable at the OpenStreetMap website (`https://planet.openstreetmap.org/`), from which a large "planet" file can be downloaded (using a custom "PBF" file format) which contains all of the mapping data stored in the OpenStreetMap database. "Extracts" of this file, which contain data for specific regions, are avaliable at the Geofabrik mirror (`https://download.geofabrik.de/`) of the database.

We use the software tool GraphHopper (`https://www.graphhopper.com/`) in order to calculate what areas are reachable from a particular point given a certain amount of time. Using the GraphHopper tool with this dataset and the next two mentioned datasets (GTFS, elevation data), this produces the "isochrones.geojson" dataset described below.

4.1.3 OpenStreetMap vs Other Alternatives

When we were initially exploring this idea, one issue we were discussing was the issue of which mapping system or service to use for isochrone calculation. Ultimately, we decided to use OpenStreetMap due to its freely avaliable nature and relatively standardized file format, which allows routing software to find the time taken to traverse a particular route given the structure of the road network.

One potential disadvantage of using OpenStreetMap over other services could be unreliability caused by the fact that OpenStreetMap is a community-run project. That said, our qualitative testing found it to be generally reliable, especially for the types of tasks we planned for this project.

4.2 GTFS Transit Data

4.2.1 Source

Almost all active publicly-funded transit departments share their current and historical transit schedules online using a protocol known as the General Transit Feed Specification (GTFS). This format, originally created by Google in 2006 for usage in Google Maps before being passed on to an open foundation, consists of, among other things, a link to a ZIP file containing machine-readable information on the historical schedules of a line, including what times a particular bus or train is expected to arrive at a particular stop. Since this information is not included in OpenStreetMap directly, we need to download this data as well in order to integrate it into our GraphHopper-based route planner and allow it to calculate public transit-based routes or isochrones.

We queried the following 10 transit agencies (listed below) for GTFS feed data that was used in isochrone calculations for our final project. This includes all transit providers with at least one transit station within a fairly wide area around Philadelphia.

1. NJ Transit:
    - Agency URL: `https://www.njtransit.com/`

- Feed URLs: Rail - https://www.njtransit.com/rail_data.zip, Bus - https://www.njtransit.com/bus_data.zip

2. Peter Pan Bus Lines:
   - Agency URL: https://peterpanbus.com/
   - Feed URLs: Bus - http://data.trilliumtransit.com/gtfs/peterpan-ma-us/peterpan-ma-us.zip
3. Adriondack Trailways:
   - Agency URL: https://trailways.com/
   - Feed URLs: Bus - https://s3.amazonaws.com/datatools-511ny/public/Adirondack_Trailways-FAB.zip
4. Amtrak:
   - Agency URL: https://www.amtrak.com/
   - Feed URLs: Bus & Rail - https://content.amtrak.com/content/gtfs/GTFS.zip
5. Amtrak (Vermonter Train):
   - Agency URL: https://www.amtrak.com/vermonter-train
   - Feed URLs: https://data.trilliumtransit.com/gtfs/amtrakvermonter-vt-us/amtrakvermonter-vt-us.zip
   - Note: While this is technically part of Amtrak, it has its own transit feed and is listed as a separate agency because it is a partnership between Amtrak and other state departments of transit.
6. Flixbus:
   - Agency URL: https://www.flixbus.com/
   - Feed URLs: Bus - http://data.ndovloket.nl/flixbus/flixbus-us.zip
7. Fullington Trailways:
   - Agency URL: https://trailways.com/
   - Feed URLs: Bus - https://s3.amazonaws.com/datatools-511ny/public/Fullington_Trailways.zip
8. Megabus / Coach USA:
   - Agency URL: https://us.megabus.com/
   - Feed URLs: Bus - https://data.trilliumtransit.com/gtfs/megabus-us/megabus-us.zip

9. PATCO / Port Authority Transit Corporation:
    - Agency URL: `http://www.ridepatco.org/`
    - Feed URLs: Bus + Rail - `http://www.ridepatco.org/developers/PortAuthorityTransitCorporation.zip`
10. SEPTA:
    - Agency URL: `https://septa.org/`
    - Feed URLs: Bus - `https://www3.septa.org/developer/google_bus.zip`, Rail - `https://www3.septa.org/developer/google_rail.zip`
    - Note: The reason these files are named `google_bus.zip` and `google_rail.zip` is due to the fact that the General Transit Feed Specification was originally called the Google Transit Feed Specification while it was still a Google project. The filenames were kept when the rename occurred for consistency reasons. Notably, they don't imply any actual association with Google.

While not all of these were specifically used in routing calculations, we made sure to cast an especially wide net when looking for transit lines to avoid missing any in our calculations. Most of the stops we used were provided by SEPTA, Philadelphia's main public transit network.

While the license that GTFS data is released under depends on the transit provider, all of the above transit providers have released their data under a license that is, at minimum, free for research and academic use. Most feeds are also free for commercial use as well.

4.2.2 Data Points

GTFS data consists of the following data points:

- Basic information about the transit agency and the data
- A list of stops that the transit agency services
- A list of rail/bus lines servicing these stops
- A list of "trips" representing scheduled times when a bus is supposed to arrive at each station along a transit line. Generally represented on a weekly basis. For example, one potential "trip" would be one where Bus #1 travels to Station A at 10:00 AM, Station B at 10:08 AM, and Station C at 10:20 AM.

4.3 Elevation Data

4.3.1 Source

One option GraphHopper supports for slightly increased route accuracy is the inclusion of publicly avaliable elevation data. Elevation data informs GraphHopper's vehicle model by adjusting travel distances to account for elevation changes and by, potentially, adjusting vehicle speed based on typical vehicle responses to elevation changes.

To facilitate this, we downloaded elevation data from NASA's Shuttle Radar Topography Mission (SRTM) which is current as of the year 2000 and accurate to the nearest 30 meters. This data is avaliable at `https://srtm.kurviger.de/`.

4.3.2 Data Points

This data is in a custom binary format and, generally, consists of elevation values for each 30 meter x 30 meter square region in the area surveyed by the original SRTM mission.

4.4 isochrones.geojson

4.4.1 Source

The census block group GeoJSON of the City of Philadelphia was used to find points to calculate as the center of isochrones. The GraphHopper open-source server was used to calculate isochrones. OpenStreetMap and public transit timetable data for the City of Philidelphia provided routing data for GraphHopper.

4.4.2 Statistics

Entries: 60120, features:

1. geoid - Unique ID for each census block group.
2. point_label - center, north, south, east, or west.
3. profile - walk, pt, or car.
4. time_limit - 10, 20, or 30.
5. geometry - GeoJSON geometry.

4.4.3 Data Cleaning

OpenStreetMap and public transit timetable datasets were combined into larger datasets for GraphHopper to process.

4.4.4 Challenges

With 45 isochrones for each of the 1336 census block groups, a total of 60120 isochrones were calculated and stored in a GeoJSON dataset. Isochrones were calculated using the GraphHopper open-source routing server using OpenStreetMap and public transit data for the City of Philadelphia. Four separate servers were used to calculate the isochrones in parallel; their outputs were combined to form a single GeoJSON dataset. A python script was used to communicate with a locally running GraphHopper instance on each server. The centroid, north, south, east, and west points were calculated from each shape of the census block groups. These points, along with each mode of transportation and time distance, were used to make API calls to the GraphHopper instance to calculate and return isochrone shapes. For public transit, all isochrones were calculated using timetable data starting the journey on Monday, March 31 2025 at 9 A.M. The python script received and stored the resulting shapes into a GeoJSON.

4.5 blockgroup_areas.csv

4.5.1 Source

The isochrones GeoJSON was used to calculate areas of each isochrone.

4.5.2 Statistics

Entries: 12024, features:

1. geoid - Unique ID for each census block group.
2. profile - walk, pt, or car.
3. time_limit - 10, 20, or 30.
4. average_area – Average area of geometry in m^2.

4.5.3 Data Cleaning

Every census block had every isochrone calculated and all had valid areas.

4.5.4 Challenges

From the isochrone GeoJSON, a python script was used to calculate the area of each shape in the isochrone geoJSON. The area over each position (Center, North, South, East, West) for a specific mode of transit and time distance was averaged. This was written into a CSV dataset.

4.6 blockgroup_poi.csv

4.6.1 Source

The isochrones GeoJSON and Yelp POI JSON was used to calculate Points of Interest inside each isochrone.

4.6.2 Statistics

Entries: 12024, features:

1. geoid - Unique ID for each census block group.
2. profile - walk, pt, or car.
3. time_limit - 10, 20, or 30.
4. average_num_poi – Average number of Yelp Points of Interest in geometries.
5. average_rating_poi – Average rating of Yelp Points of Interest in geometries (out of 5 stars).
6. poi_list_center – List of unique IDs of Points of Interest inside center geometry.

4.6.3 Data Cleaning

Every census block had every isochrone calculated. Some isochrones did not contain any Points of Interest, so code was added to the python script to handle edge cases of empty POI lists.

4.6.4 Challenges

From the isochrone GeoJSON, a python script was used to calculate all the Yelp points of interest that were in each isochrone shape. The average number and average rating (out of 5 stars) for each position for a specific mode of transit and time distance was calculated. Additionally, the list of unique IDs for each POI in the center position was stored. This was written into a CSV dataset.

4.7 Census Block Groups

4.7.1 Source

Data Source: City of Philidelphia (https://opendataphilly.org/)

4.7.2 Statistics

Entries: 1336, Rows:

1. OBJECTID - Internal feature ID (sequential integer) used by the GIS software.
2. STATEFP10 - 2-digit FIPS code for the state (2010).
3. COUNTYFP10 - 3-digit FIPS code for the county within the state (2010).
4. TRACTCE10 - 6-digit census tract code (2010).

5. BLKGRPCE10 - 1-digit block-group code within the tract (2010).
6. GEOID10 - Full 12-digit concatenation: STATEFP10 + COUNTYFP10 + TRACTCE10 + BLKGRPCE10.
7. NAMELSAD10 - Naming Groups.
8. MTFCC10 - MAF/TIGER feature class code (identifies this as a census block group—G5030).
9. FUNCSTAT10 - Functional status code (e.g. "S" = "State-defined/statistical area").
10. ALAND10 - Land area in square meters.
11. AWATER10 - Water area in square meters.
12. INTPTLAT10 - Latitude of the centroid.
13. INTPTLON10 - Longitude of the centroid.
14. Shape__Area - Latitude and longitude of the "internal point" (centroid) of the polygon.
15. Shape__Length - Perimeter length of the polygon
16. geometry - GeoJSON Geometry.

4.7.3 Data Cleaning

We primarily used INTPTLAT10, INTPTLON10, Shape__Area, and Shape__Length for our analysis. To more quickly find the cooresponding isochrone for each census block, as the centroids did not match perfectly, we developed a script that for each Census block group, would append the coordinates for the first isochrone region that had its centroid within the census block group. This allowed us easy comparisons between groups for future tasks.

4.7.4 Challenges

Our greatest challenges with the Census Block dataset was finding corresponding isochrone regions. While our method worked generally well, there were some isochrones with centers on the edges. This is unlikely to be part of the correct census block group and it is unclear where these isochrones were calculated from. Additionally, there were some horseshoe or strangely shaped census block groups that did not contain the center of any isochrones. While these isochrones are likely in the dataset, for this initial analysis we omitted these areas from our data and regressions.

4.8 Yelp Business Data

4.8.1 Source

Data Source: Yelp Open Business Dataset (https://business.yelp.com/data/resources/open-dataset/)

4.8.2 Statistics

Entries: 14569 (after filtering), Rows:

1. business_id - Unique Yelp-assigned identifier for the business.
2. name - Business name.
3. address
4. city
5. state
6. postal_code
7. latitude
8. longitude
9. stars - Average star rating (1.0–5.0).
10. review_count - Total number of reviews.
11. is_open - (1 = currently open (at time of data collection); 0 = closed.)
12. attributes - JSON/object of boolean or categorical attributes (e.g. "WiFi": "free", "Parking": "garage")
13. categories - Comma-separated list of business categories (e.g. "Restaurants, Italian").
14. hours - JSON/object mapping days ("Monday"–"Sunday") to opening and closing times.

4.8.3 Data Cleaning

Isochrone Features We first created a new POINT geometry by combining the longitude and latitude columns. We then mapped the dataset to an epsg of 4326. We then merged the dataset, grouping by the mean number of stars in each isochrone region.

Census Group Features We did the same as above with our census group features.

4.8.4 Challenges

] The yelp dataset was extremely large and limited to only 11 cities. We originally intended to study the DMV area, but pivoted to Philadelphia due to the lack of data for the DMV area.

4.9 RTT Real Estate Data

4.9.1 Source

Data Source: City of Philidelphia (https://opendataphilly.org/)

4.9.2 Statistics

Entries: 296448, Rows:

1. objectid - Feature ID.
2. document_id - Document ID.
3. record_id - Record ID.
4. document_type - Type of the legal document (e.g. "DEED", "SHERRIFS DEED").
5. display_date - Date/time of the transaction.
6. receipt_date - Date on the receipt issued when the document was filed.
7. recording_date - Date the document was officially recorded in the registry.
8. document_date - Date appearing on the document itself (e.g. signing date).
9. street_address
10. address_low - Numeric low house number.
11. address_low_suffix - Alphabetic suffix on the low number (e.g. "A" in 123A).
12. address_low_frac - Fractional portion of the low number (e.g. "1/2").
13. address_high - Numeric high house number (if the document covers a range).
14. street_predir - Directional prefix (e.g. N, S, E, W).
15. street_name - Street name (e.g. "Market").
16. street_suffix - Street type (e.g. "St", "Ave", "Blvd").
17. street_postdir - Directional suffix (e.g. "NW").

18. zip_code - 5-digit ZIP code (stored as float if there are missing/nulls).
19. ward - City political ward number.
20. condo_name - Name of the condominium complex (if applicable).
21. unit_num - Unit or apartment number within a condo or multi-unit building.
22. grantors - Name(s) of the seller(s) or party transferring interest.
23. grantees - Name(s) of the buyer(s) or party receiving interest.
24. cash_consideration - Cash portion of the purchase price.
25. other_consideration - Non-cash consideration (e.g. assumption of debt).
26. total_consideration - Sum of cash + other consideration.
27. assessed_value - Property value as assessed by the Office of Property Assessment (OPA).
28. common_level_ratio - Ratio used to adjust assessed values to market values
29. fair_market_value - Estimated market value of the property.
30. state_tax_amount - State transfer tax charged on the transaction.
31. state_tax_percent - Percentage rate of the state transfer tax.
32. local_tax_amount - Local (city/county) transfer tax amount.
33. local_tax_percent - Percentage rate of the local transfer tax.
34. (Next 8 lines are inflation adjuster parameters that are unused)
35. lat - Latitude.
36. lng - Longitude.

4.9.3 Data Cleaning

Isochrone Features We extracted three features out of this dataset. We first broke the dataset into two. (deed_df: document_type == 'DEED', assessed_value.notnull()), display_date greater than 2010) (foreclosure_df: document_type == 'DEED SHERIFF', assessed_value.notnull()), display_date greater than 2010). For each point we combined the lon and lat columns to create a new POINT geomtery. We then did an inner join and grouped by each region, aggregating the mean housing value. For the foreclosure dataset we instead found the total amount of foreclosures per region. We then extracted a third foreclosures_area feature by dividing this by the isochrone area.

Census Group Features We did the same as above with our census block dataset for our Census Block group baseline features.

4.9.4 Challenges

A key difference was that our area column had to be modified as it was in the wrong format in the Census block dataset. This dataset was most challenging to interpret, as it has many columns with technical terms that we were not familiar with. The dataset was very large so e only selected 3 features from the dataset: assessed property value, number of foreclosures, and foreclosures divided by area. We understand that assessed property value has its flaw in predicting true value, but had inconsistent values for the other columns. For example, the property value column would often be set at 1 as this as the minimum value when changing ownership between people.

4.10 Choice Neighborhoods

4.10.1 Source

Data Source: City of Philidelphia (https://opendataphilly.org/)

4.10.2 Statistics

Entries: 4, Rows:

1. OBJECTID - Feature ID.
2. NAME - Name of the neighborhood (e.g. “Mantua”).
3. TYPE - Planning Status (e.g. “Planning”, “Implementation”).
4. Shape__Area - Polygon area (square meters).
5. Shape__Length - Polygon perimeter length (meters).
6. geometry - Polygon geometry defining the region.

4.10.3 Data Cleaning

Isochrone Features We did a one hot encoding for each isochrone to see if they intersected with a choice neighborhood in the dataset. This was done by creating a unary_union of all geometries in this dataset, and then for each isochrone calculating whether there was an intersection.

Census Group Features We did the same one hot encoding for each Census Block Group for our baseline.

4.10.4 Challenges

For Choice neighborhoods, we treated the feature as a binary. Either a geometry has overlap with a choice neighborhood, or it does not. We used this to identified areas of the city that overlapped with these neglected groups for our feature.

4.11 Public / Private Pools

4.11.1 Source

Data Source: City of Philidelphia (https://opendataphilly.org/)

4.11.2 Statistics

Entries: 72, Rows:

1. OBJECTID - Feature ID.
2. AMENITY_NAME - Official name of the pool facility.
3. PARK_NAME - Name of the park or recreation center housing the pool.
4. ADDRESS_911 - Street address formatted for 911/emergency services.
5. ZIP_CODE
6. POOL_TYPE - "INDOOR" vs. "OUTDOOR".
7. POOL_STATUS - Operational status (e.g. "ACTIVE", "INACTIVE").
8. DATE_INSTALLED - Date the pool was built or opened.
9. COMMENTS - Additional notes.
10. DATA_SOURCE - References to where/when the data were collected (e.g. "Programs 2023", "Nearmap 2023").
11. geometry - Point geometry of the pool.

4.11.3 Data Cleaning

Isochrone Features We first mapped the geometry column to an epsg of 4326. Then we created a new column in the isochrone dataset using a lambda function to the minimum distance from the geometry to the nearest pool.

Census Group Features We did the same mapping as above with the Census Group geometries for our baseline.

4.11.4 Challenges

As there were so few pools in the dataset, we found that it was not useful to have a simple binary parameter on whether there were pools in a specific isochrone or census group. Instead, for each geometry we calculated the distance to the nearest pool, giving a more useful feature.

4.12 No trucks

4.12.1 Source

Data Source: City of Philidelphia (https://opendataphilly.org/)

4.12.2 Statistics

Entries: 2728, Rows:

1. OBJECTID - Feature ID.
2. ON_ - Name of the street where the no-through-trucks restriction applies.
3. FROM_ - Starting street of the segment.
4. TO_ - Ending street of the segment.
5. YEAR - Year the restriction was enacted.
6. SEG_ID - Street segment identifier.
7. ST_CODE - State (or jurisdiction) code.
8. ONEWAY - Directionality indicator ("B" = both directions, "F"/"T" = one-way).
9. ST_NAME - Street name.
10. STNAME - Street name alternate.
11. Shape__length -Length of the road prohibiting trucks.
12. geometry - A LINESTRING geometry representing the segment on which trucks are prohibited.

4.12.3 Data Cleaning

Isochrone Features We first mapped the geometry column to an epsg of 4326. We then did an inner, intersection join with the isochrone dataset, followed bygrouping by the sum of the Shape__length per each group.

Census Group Features We did the same mapping as above, but joined it with the Census Group geometries to calculate our baseline.

4.12.4 Challenges

Our main challenge with this data was figuring out which feature to extract. This data just givens coordinates and names of roads, which is not useful on its own. We came up with a meaningful measurement by summing the total length of roads with no trucks in a given area to get our feature.

5 Methods

The main steps taken in the completion of this project were:

1. **Defining Study Scope:** The geographical focus of the study was established as Philadelphia, Pennsylvania. We used census block groups as the fundamental unit of analysis for data aggregation, isochrone generation, and subsequent statistical modeling using these isochrone-based groups.

2. **Dataset Sourcing and Curation:** Acquired and processed a diverse range of datasets, including OpenStreetMap for road networks, GTFS feeds for public transit data, elevation data, and various City of Philadelphia public datasets. We then cleaned and filtered each dataset and combined them into one dataset that could be used for our final analysis.

3. **Isochrone Computation:** A comprehensive set of 60,120 isochrones was generated. For each of Philadelphia's 1336 census block groups, 45 distinct isochrones were calculated by using an open-source routing tool known as GraphHopper.

4. **Census Block Group-Level Feature Aggregation:** We then aggregated each feature at the Census Block Group level and calculated the average value of the feature within the isochrone area. This created the final dataset upon which we ran our analysis.

5. **Data Analysis:** We used a linear regression - based approach to analyze the correlations present in the data and trained a decision tree to attempt to predict the average price of homes within each isochrone area given other factors within the isochrone.

6. **Website Creation:** We uploaded a filtered version of our data and regression outputs designed to load on the web to a site hosted using GitHub Pages at `https://3benknight.github.io/Isophilly_site/`

6 Data Analysis

For each isochrone, we aggregated features from multiple sources within its bounds. Key features included: the *count of foreclosures* in the area (and density per area), the *average Yelp rating of businesses* within the area, the total length of roads with "no-truck" restrictions, and the distance to the nearest public pool or recreation center. We obtained these data from the OpenDataPhilly online database and the Yelp site. Each dataset was processed into a geospatial layer (point or line) with attributes. We then performed spatial joins to compute features for each isochrone. Pseudocode:

```
for each isochrone in dataset:
    subset_points = all_point_data within isochrone.geometry
    subset_lines = all_line_data intersecting isochrone.geometry
```

```
features = {
   foreclosure_count: count(subset_points["Foreclosure"]),
   foreclosure_over_area: foreclosure_count / isochrone.area,
   avg_stars: mean(subset_points["YelpRating"]),
   no_truck_length: sum(length of subset_lines[road_type="no_truck"]),
   distance_pool: distance(from isochrone center to nearest pool)
}
attach features to features csv
```

We did a variety of analyses on our final data, with the ultimate goal of quantifying the relationship between the average real estate prices within each isochrone and the other factors we measured. We first did a regression-based analysis where we calculated the correlation coefficient between each factor in our analysis and the average housing price in an area, then trained a random-forest based model to predict the average housing price in an area.

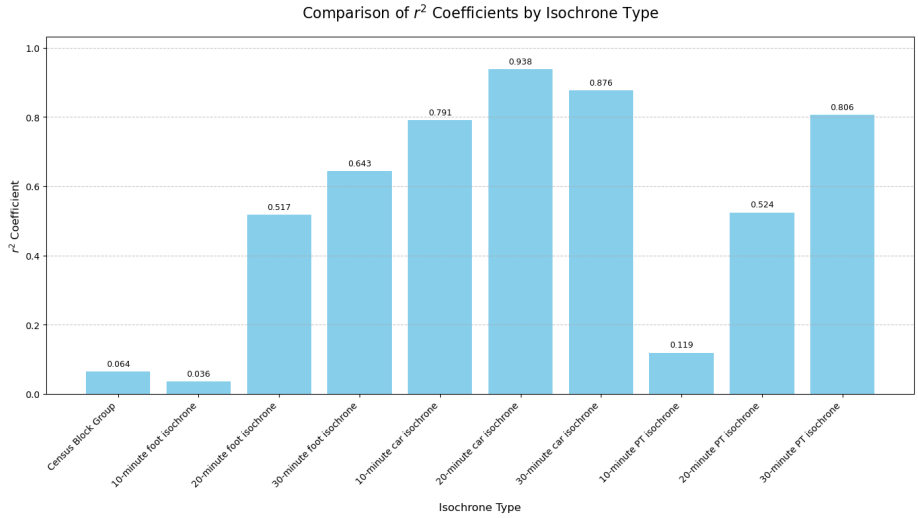
6.1 Linear Regression Based Analysis

Our first step was to, for each isochrone type, run a linear regression between the factors measured for all isochrones of that type and the resulting average housing price in this area.

In this particular analysis, more than the exact values itself, the most important factor to be measured is the dependence between the correlation observed and the type of isochrone used. For example, if 30-minute foot isochrones offered the highest correlation between housing prices and the other factors measured, this would mean that, for someone living in that area, the statistics within a 30-minute walking radius are most informative to the area's housing prices. This means that, when considering that area's housing market, it is most useful to consider the real estate factors present in the area surrounding this location - and specifically, on the area reachable by driving for 30 minutes.

The results we obtained from the analysis are visible below:

Isochrone Type	r^2 Coefficient
No Isochrone (Census Block Group Directly)	0.064
10-minute foot isochrone	0.036
20-minute foot isochrone	0.517
30-minute foot isochrone	0.643
10-minute car isochrone	0.791
20-minute car isochrone	0.938
30-minute car isochrone	0.876
10-minute public transit isochrone	0.119
20-minute public transit isochrone	0.524
30-minute public transit isochrone	0.806



Based

6.2 User Interface + Data Visualization

Our final product is an interactive visualization that users can interact with to determine how various factors across these datasets contribute to the statistics of census block groups. We show two main things:

1. A map view of all the isochrones measured in our project, which allows users to view color-coded maps of the statistics aggregated within each isochrone.
2. A table view of all the regressions conducted in the first stage of the data analysis and their coefficients for each isochrone.

The final website is avaliable at `https://3benknight.github.io/Isophilly_site/`.

7 Project Archival

We have archived all files and our github site for future access. Our files can be found in the following repos:

1. Reports/Data Cleaning: `https://github.com/bufn-403-project/transit-and-finance`
2. Website: `https://github.com/3benknight/Isophilly_site`.
3. Isochrone Scripts: `https://files.doggett.tech/umd/bufn-403/Transit-And-Finance/`
4. Early Website Test: `https://github.com/3benknight/IsoPhilly/tree/main`

8 Team Roles

Member	Role
Benjamin Knight	Web Development Data Collection/Cleaning/Analysis Regressions
Jack Doggett	Distributed Compute Isochrone Calculation
Om Duggineni	OpenStreetMap + GTFS Data Aggregation Isochrone Calculation Setup Writeup of Data Analysis

References

[1] Dorantes, L. M., Paez, A., & Vassallo, J. M. (2011). Analysis of House Prices to Assess Economic Impacts of New Public Transport Infrastructure: Madrid Metro Line 12. Transportation Research Record, 2245(1), 962-978. https://doi.org/10.1177/13634607221107827

[2] General Transit Feed Specification. (n.d.). Retrieved March 7, 2025, from https://gtfs.org/

[3] Lin, D., Broere, W., & Cui, J. (2022). Metro systems and urban development: Impacts and implications. Tunnelling and underground space technology, 125, 104509.

[4] TIWARI, G. (2013). Metro Rail and the City: Derailing Public Transport. Economic and Political Weekly, 48(48), 65–76. http://www.jstor.org/stable/23528925

[5] OpenStreetMap. (n.d.). OpenStreetMap. Retrieved March 7, 2025, from https://www.openstreetmap.org/

[6] Summary of deposits. (2023, June 1). FDIC. https://www.fdic.gov/bank-financial-reports/summary-deposits

[7] Truong, R., Gkountouna, O., Pfoser, D., & Züfle, A. (2018). Towards a better understanding of public transportation traffic: A case study of the Washington, DC metro. Urban Science, 2(3), 65.

[8] Who uses OpenStreetMap? (n.d.). Retrieved March 7, 2025, from https://welcome.openstreetmap.org/about-osm-community/consumers/