MathWorks Math Modeling Challenge
A program of siam

SAMPLE PAPER: ABOVE AVERAGE

The team provided a very good executive summary. An overview of the problem is given, a good overview of the results is given, and the team provides good guidance about the general techniques used to obtain their results.

The team describes a discrete difference equation for their first model. The method is clearly defined. The formatting of their work makes it a little difficult to parse, but it is clear and easy to determine the details of their method. They do not provide citations within their text, but they do cite their use of the data provided to them by the event's organizers. The team discusses a good, basic sensitivity analysis. More importantly they give a good interpretation of their analysis and are able to discuss the relative impact of the different variables they explored.

The team made use of a Random Forest Regressor for the second question. Citations are provided for the extra data the team obtained, but the citations for the algorithm are lacking. A citation is given to indicate how the training data was used, and the team gives a good general description of the process. However, details about how the data was partitioned is not given.

The greatest weakness in the paper is the response to the third question. The team made use of a linear combination of different factors. The method is not well described, and the results are not as clearly described as those in the first two questions. The response to the third question is consistent with having run out of time. The team did provide a good start to a model, though.

M3 Challenge 2025 - Hot Button Issue: Staying Cool as the World Heats Up

Team 17***
Feb 28, 2025

Executive Summary

As global warming establishes a greater impact on the climate with each passing year, heat waves are a frightening reminder of the limits of air conditioning. The increasing intensity and length of heat waves can overwhelm even the recent improvements in cooling efficiency and housing infrastructure. Heat waves can also mean putting heavy strain on the power grid and increasing the risk of power outages, which can cut off access to both AC and critical health and communication systems. This is especially prevalent in low-income urban areas, which may have heavier reliance on what heat waves often compromise.

Our first model examined the way in which heat waves affect dwellings. We developed a model which takes information about a certain dwelling (location, structure, age, etc), and based on that data predicts how the internal temperature will perform during a given heat wave (with given temperatures and dew points each hour). Our model predicts that the dwellings will, overnight, experience roughly the same temperature on the inside as well as on the outside, however during the day will have an internal temperature of up to 7℉ or 8℉ higher than the outside temperature.

Our second model regards the power strain Memphis should expect during its worst heat, in the present day and the following decades. Although the Energy Information Administration provides daily figures on Tennessee's electricity use as far back as 2019, an accurate projection must incorporate how Memphis's climate and population are expected to evolve. So, we trained a Python-based ML model on past years of Tennessee's weather and power usage reports, then were able to extrapolate our predictions to focus on Memphis. We found that the peak power usage Memphis will experience in heat waves is actually predicted to decrease by about 10% over the next 20 years, though probably not uniformly.

Our third model deals with vulnerability to heat-related crises. We developed a composite score using percentile rank. We considered two subdomains which were social and physical variables. We also considered two categories within the social subdomain which considered socioeconomic and vulnerable populations. We found that census tracts in the center of the city were more likely to suffer from heat vulnerability. To address physical issues we considered a heat island index which took into account greenspaces.

Table of Contents

Q1: Hot to Go

1.1 Defining the Problem

The problem asks us to create a model which can predict the temperature of any non-air conditioned dwelling during a heat wave over a 24 hour period. We are then asked to apply this to four given dwellings, during a heat wave with given data, in either Memphis, Tennessee or Birmingham, England. We chose Memphis for all analyses in this paper. We used data on how various factors (outside temperature, solar radiation, and dew point), affect indoor temperatures during a heat wave to create our model.

1.2 Assumptions

At midnight, the indoor and outdoor temperatures are equal

- Justification: Traditionally, households without air conditioning would use nighttime in order to cool down the building by opening doors and windows[1]. In other words by equalizing temperatures and using the cooler nighttime temperatures outside as a heatsink for the house. Consequently, it is reasonable to assume that, for a non-air conditioned dwelling, temperatures inside and outside will be equal at midnight. Further, our model replicates this exact behavior towards the end of the 24 hour period approaching the next

Most dwellings will be brick

- Justification: In the dataset we used, almost every entry was brick. This means that the data for non-brick buildings will necessarily be skewed, as it is the result of very few dwellings. Further, since we don't have any construction material data for the homes we must test our model, it is rational to assume that they use the most common material.

1.3 Variables

Symbol	Definition	Units
T_O	Outside Temperature	°Celsius
T_I	Inside Temperature	°Celsius
R	Solar Radiation[3]	MJ/m^2
D	Dewpoint	°Celsius

Table 1.1: Variable definitions for Q1

Our data[2] provides a slew of constants, too many to mention here but they can be found in the code, essentially they fit into five main categories. Air Conditioning constants C_A, which weight the effect of a factor based on whether a household has AC, occupancy constants C_O which weight factors on occupancy and are further broken down into numbers for high rises and non high rises, construction type constants C_C , which are broken down by four types of construction (brick, asphalt, vinyl, wood), construction date constants C_D which weight factors by date of building construction, further broken down into numbers for pre 1940 buildings, 1940-1970 buildings, and post-1970 buildings, and neighborhood surroundings constants C_N which weight factors by their surroundings (four types, concrete, residential, yard/park, and urban). Each building has one constant from each category per factor, so 15 in total as explained in the next section.

1.4 The Model

The model is very simple. Essentially, it loops through each hour in the twenty four hour period, and for each one it adds degrees relative to the dew point, solar radiation, and temperature. Specifically, there is a list of coefficients which are multiplied by the differences between the temperature, dew points, and solar radiation of that hour and the previous hour, and then the results are added to the temperature. Further, we assume that all the homes are brick (because this is true of almost all the homes in our dataset), and add some temperature according to this. In order to get the exact function, we first must calculate the changes in outside temperature, solar radiation, and dew point since the previous hour:

$$\Delta T_O(t) = T_O(t) - T_O(t-1)$$

Equation 1 - Change in outside temperature

$$\Delta R(t) = R(t) - R(t-1)$$

Equation 2 - Change in solar radiation

$$\Delta D(t) = D(t) - D(t-1)$$

Equation 2 - Change in dew point

Further, we have to multiply each of these changes by the various constants which quantify the degree to which the changes in temperature, solar radiation, and dewpoint combine with the air conditioning, occupancy, age, etc of a house to produce changes in temperature. Since we are simply multiplying each constant by temperature, radiation, or dew point we can add them up together, and then multiply the sums by those three data points.

$$C_T = C_{AT} + C_{OT} + C_{CT} + C_{DT} + C_{NT}$$

Equation 4 - Total Weight for temperature

$$C_R = C_{AR} + C_{OR} + C_{CR} + C_{DR} + C_{NR}$$

Equation 5 - Total Weight for solar radiation

$$C_D = C_{AD} + C_{OD} + C_{CD} + C_{DD} + C_{ND}$$

Equation 6 - Total Weight for dew point

From these, we can multiply the change in outside temperature by constant for temperature, change in radiation by constant for radiation, change in dew point by constant for dew point, and add those all up along with the previous inside temperature to get the current inside temperature. Note that we assume that $T_I(0) = T_O(0)$, as previously mentioned.

$$T_I(t) = T_I(t - 1) + \Delta T_O(t) \cdot C_T + \Delta R(t) \cdot C_R + \Delta D(t) \cdot C_D$$

Equation 7 - Inside Temperature as a function of time

1.5 Results

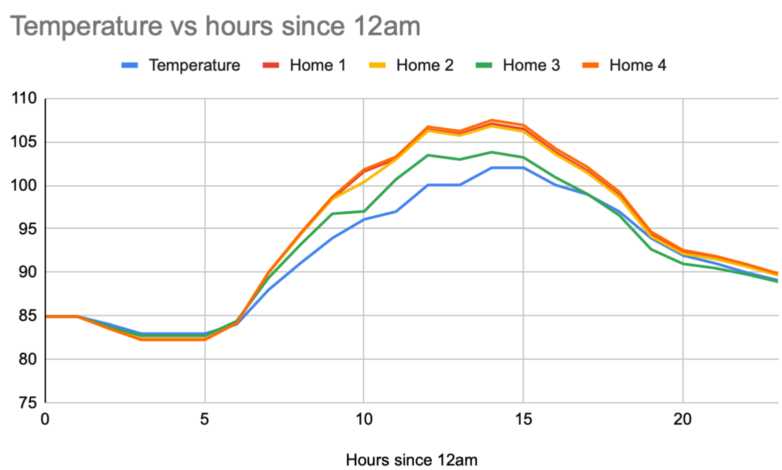


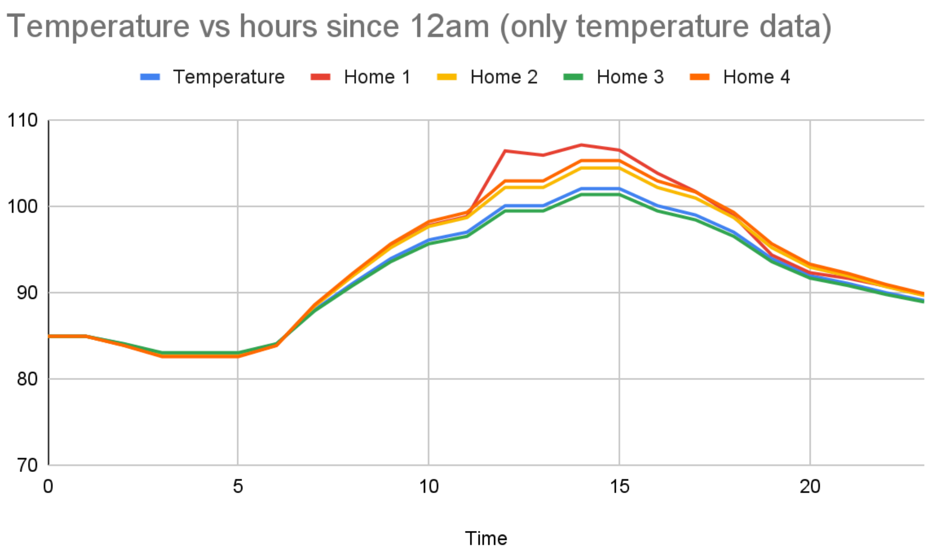
Table x:

1.6 Discussion

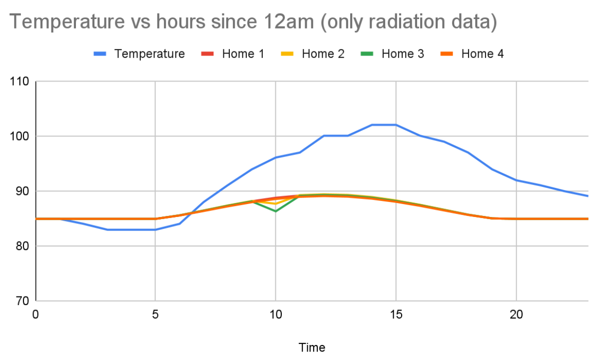
These results suggest that, during a heat wave, temperatures for homes will remain roughly the same as outside temperatures during the night, but then during the day will rise to peaks of up to 7 or 8 degrees higher than the outside temperature, returning to outside temperature at night once again.

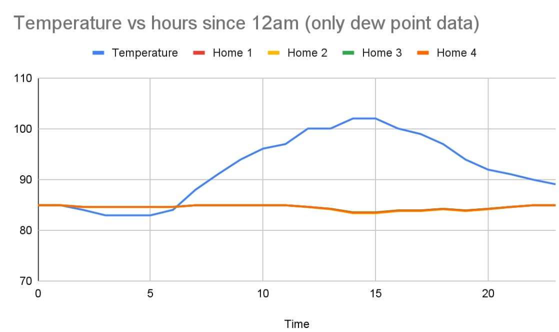
1.7 Sensitivity Analysis

Our inside temperature prediction uses three factors (outside temperature, solar radiation, dew point), weighted according to several factors. In order to analyze the sensitivity of our model, we ran three more times, each time giving sole weight to one of those factors. First, we gave sole weight to temperature.



As one can see, this is a very similar graph to the previous one which resulted from all variables being taken into account, suggesting that our model puts high weight on temperature. This theory is corroborated by the graphs only using solar radiation and dew point:





As one can see, these graphs essentially remain constant at $T_O(0)$. Together, these graphs indicate that our model places extremely heavy weight on temperature data, while placing less weight on solar radiation, and even less on dew point. This makes sense, as when attempting to calculate temperature in a certain area it is logical to rely primarily on temperature around that area. We are confident that the other factors have utility in our model, thus their inclusion, but nonetheless this sensitivity makes sense.

1.8 Strengths and Weaknesses

Strength

- Our model takes into account a variety of factors which could cause temperature to affect a building differently.

- Our model weights those factors in a complex manner which accounts for many possible different

Weaknesses

- We did not explicitly consider the size of a building, however this should not be the largest concern as a larger building will expose more surface area to temperature and solar radiation. Consequently, the increased size will likely be cancelled out due to the increased effect of the various factors on the building.
- We did not explicitly consider the effect of shade from trees upon the temperature of a building. However, we do consider solar radiation, and since the dataset we used did not account for shade from trees, it is likely that the numbers are naturally adjusted to account for some average tree coverage.
- We assumed that all buildings were constructed out of bricks. We did so because the vast majority of our dataset was composed of buildings made of bricks (likely indicating a prevalence of brick buildings in the US generally), meaning that data for any type of building which was not brick was likely to be unreliable. Further, we are not given the material out of which the buildings we must analyze are made, consequently we must would have had to assume some material anyway to use this model
- Our primary source of data was from a small sample size study of housing in Detroit, this was regrettably unavoidable due to a lack of thorough data available online.

1.9 Future Growth

Given more time, we would likely attempt to find more thorough data. Our central problem was that all data available online was either in such large quantities that gaining any useful insight from it in 14 hours would have been utterly impossible, or very small sample size studies (one of which we ended up using, since despite its flaws we were able to properly analyze it in the time given). If we had more time, we would have been able to use one of the larger census databases and have been able to establish better and more accurate models.

Q2: Power Hungry

2.1 Defining the Problem

The problem asks us to predict the peak power demand during the summer months for our chosen city, again Memphis. Further, we are asked what changes we foresee happening to that number in the next 20 years. Memphis' power grid is operated by the Tennessee Valley Authority (TVA) which serves the state of Tennessee as well as portions of surrounding states. The model we produced used a Python-based ML model to synthesize weather patterns with statewide energy usage, to estimate the maximum daily power demand in a year.

2.2 Assumptions

Weather is roughly constant throughout the TVA service area.

- All weather data are from stations in Memphis, which we determined through qualitative analysis was not consistently skewed against other Tennessee cities in the TVA area which contribute to the energy demand data we use.

2.3 The Model

Our main power demand model consists of two main data foundations: historical weather data and energy usage statistics. Energy usage statistics came from the US Energy Information Administration (EIA)[4] and reported daily power demand in the TVA region from the start of June to the end of September in the years 2019 through 2024. Historical and future weather data was obtained from Open-Meteo's Historical Weather API and Climate API[5] and reported several pieces of information for each day, including temperature highs and lows, humidity, precipitation, and solar radiation.

These two data sets were both used to train a machine learning model, the programming of which was guided by Gigi Dattardon's similar demonstration[6] of weather-informed electricity demand forecasting. Both data sets spanned years 2019-2024 during months June-September, with the meteorological measurements coming from Open-Meteo's Historical Weather API.

We adapted a Python Jupyter notebook with Dattardon's code frame to accept our weather and electricity datasets. Our initial weather data included Tennessee's historical measurements of daily maximum and minimum temperatures, humidities, precipitation, dew points measurements, and windspeeds every day of the year. Our electricity data contained the TVA area electricity demand in megawatt hours from the same date range. By combining these two data tables in the notebook, we were able to apply the combined dataset to three different predictive machine learning models: Histogram-based Gradient Boosting Classification Tree, a Random Forest Regressor, and an XGBoost Regressor.

In the Linear Regression model, a linear relationship between the input(s) and the output is estimated and refined until it best fits the data. The quality of a guess is judged by the conditional mean deviation of each data point, and the line is adjusted until the deviation is minimized. Linear regression is a member of a large family of regression models for other curve families, often used among which are quadratic and exponential regression. The equation of a regression model takes very little time to compute, although it is only viable if the nature of the relationship is known. If the data do not indicate a linear relationship, linear regression is useless.

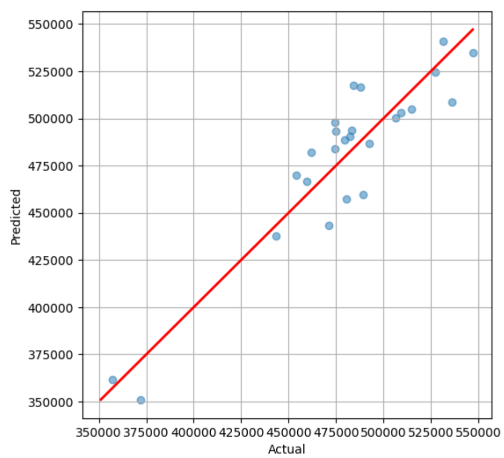
Random Forest Regressor, a popular ensemble learning technique, constructs multiple decision trees during training. Each tree is built using a random subset of the training data and features. By aggregating predictions from these individual trees, Random Forest Regressor effectively handles nonlinear relationships and mitigates overfitting. Its robustness and efficiency make it well-suited for datasets with high dimensionality and varying complexities.

In the Random Forest Regressor, multiple decision trees are utilized simultaneously during training, each being assigned a random assortment of categories from the training data. By evaluating the combined results of all the trees, this predictive model can handle nonlinear relationships better.

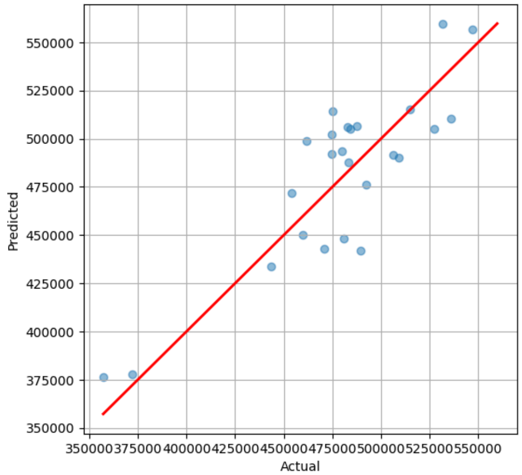
In the XGBoost Regressor, multiple decision trees are used similarly to the Random Forest Regressor. However, unlike the previous model, this algorithm builds the trees sequentially, with each tree acting as an upgrade that corrects the errors of the previous decision trees.

2.4 Results

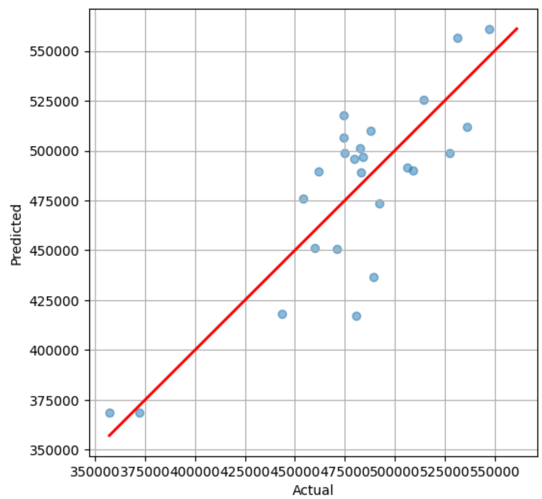
	Linear Regression	Random Forest Regressor	XGBoost Regressor
Mean Squared Error	309088540.87535405	536846495.7848759	692985214.1654687
Mean Absolute Percentage Error	3.14%	4.26%	4.68%
R Squared	0.83	0.70	0.62



Linear Regression



Random Forest Regressor



XGBoost Regressor

With these three trained models, we have three different extrapolated relationships between climate conditions and energy demand in the Memphis area. By plugging the future daily weather projections from 2025 to 2045 produced from Open-Meteo's Climate API into these models, we can derive the associated maximum power demand for each year.

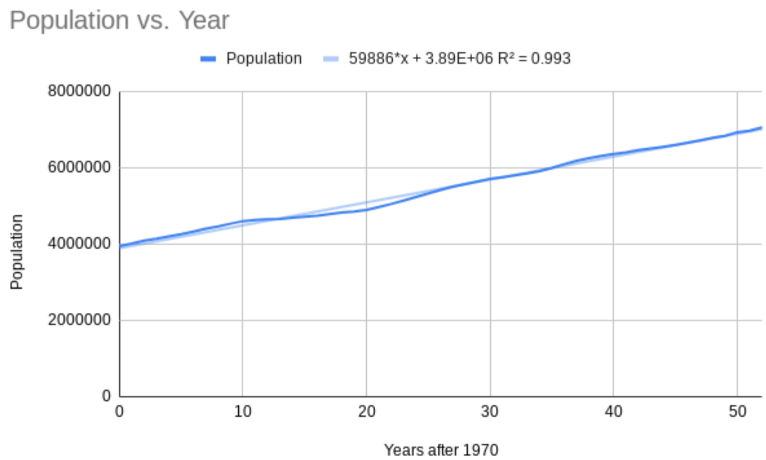
Year	Linear Regression (MWh)	Random Forest Regression (MWh)	XGBoost Regression (MWh)
2025	607033.577	555555.52	546944.375
2026	683346.1907	555555.52	546944.375
2027	708691.4973	554769.98	545941.8125
2028	638881.6062	555555.52	546944.375

2029	582069.0275	554769.98	545941.8125
2030	693297.147	555555.52	546944.375
2031	709904.3367	554672.75	549127.4375
2032	666776.2778	554769.98	548626.3125
2033	702895.51	554769.98	545941.8125
2034	682355.1413	554769.98	548136.5625
2035	664332.4311	554613.32	547480.1875
2036	663076.5057	555343.02	548626.3125
2037	672270.0406	555555.52	546944.375
2038	666380.6696	555343.02	548626.3125
2039	667297.5953	555555.52	546944.375
2040	626644.2405	555343.02	546944.375
2041	744928.326	554769.98	548626.3125
2042	667121.177	554613.32	544670.3125
2043	629685.2475	554769.98	549127.4375
2044	682306.0255	555343.02	548626.3125
2045	697402.3502	554613.32	545661.6875

2.5 The Model (cont.)

Our main model's output projected the TVA region's total daily energy demand, which we needed to translate to Memphis by itself. To do this, we used two scaling factors.

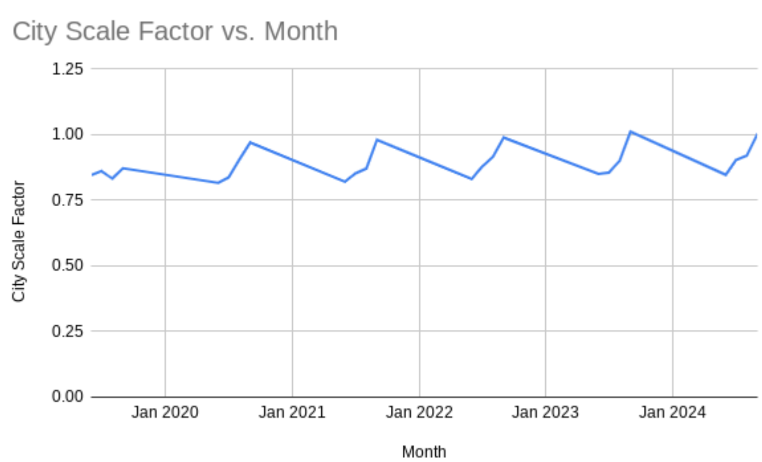
The first is the ratio between the populations of Memphis and TVA, which are both found via projections. We first used the last 50 years of Tennessee's population[7] to predict the next 20 years via linear regression.



Tennessee Population History with Linear Fit

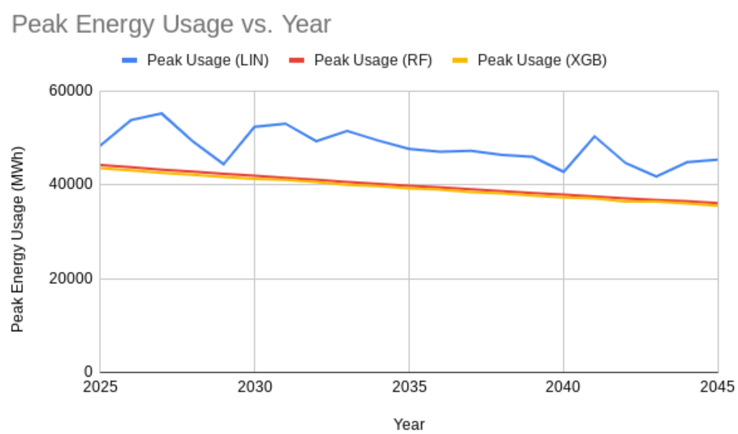
To account for the discrepancy between the state of Tennessee and the TVA region, we searched for TVA's population history. Since we could only find one data point, for 2023[8], we assumed proportionality between its population and Tennessee's multiplying the linear model by a constant. Combined with projections for Shelby County's population (the county containing Memphis and immediate surroundings) provided by the University of Knoxville[9], we calculated the ratio between TVA and Shelby populations from the present up to 2045.

The second scaling factor accounts for how much power is used per capita in Shelby County compared to the TVA region as a whole. Using the aforementioned population models and the TVA energy demand data, as well as monthly Shelby energy demand reports[10], we calculated the energy demand per capita for both regions during each considered summer month. When plotting the ratio, which we named the City Scale Factor, we found a consistent increasing pattern over each summer.



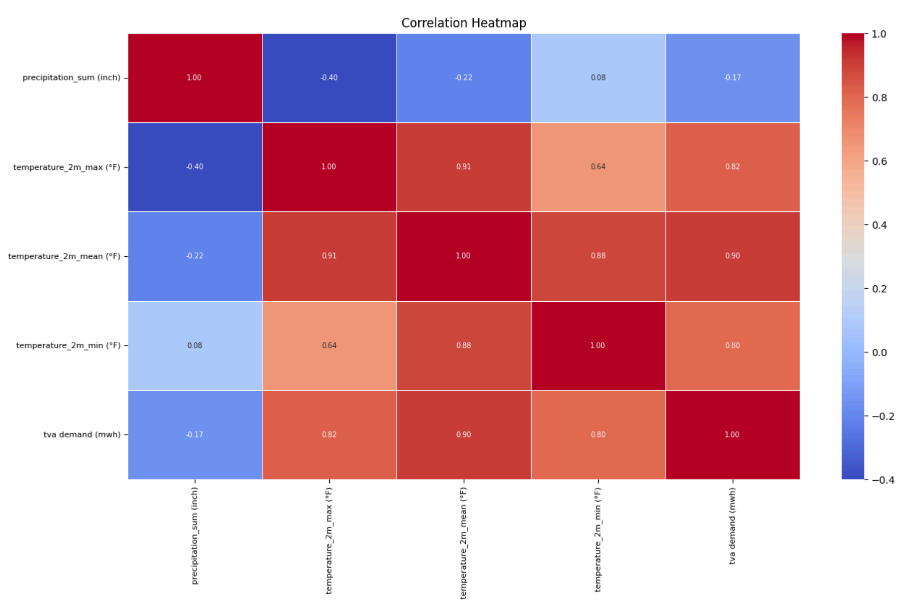
This pattern is likely due to using the same population figure for each month in a year, since January's population can be significantly far from the true value. So, we elected to simply take an average of the 24 plotted scale factors and incorporate it as a constant, equalling roughly 0.891. The are many reasons why this value may be less than 1, but it's plausible that industrial activity outside of the Memphis area may skew energy demand away from the city.

The projected Shelby County peak energy demand can be calculated by multiplying the predicted TVA peak by the population ratio and the constant City Scale Factor. Plotting this for each year from 2025 to 2045 yields a subtly decreasing relationship.



2.6 Discussion

All three models predict an overall decrease in Memphis summers' most power-intense days. However, it's unlikely that this change will happen smoothly, as the linear regression model's superior R^2 value indicates. Roughly, this summer's prediction stands at 45000 MWh, and the prediction in 2045 is around 40000 MWh.



Correlation Heatmap

Further analysis shows that the temperature fields affected the model much more strongly than rainfall. This demonstrates that our model correctly evaluates the direct cause of high energy usage - extreme heat.

2.7 Strengths and Weaknesses

Strengths

- Multiple predictive models corroborate trends and relationships.
- There exists a strong correlation between energy demand and temperature, and a weak correlation between energy demand and precipitation.
- Energy demands factor in both climate and population factors.
- The data is concise and relevant to Memphis, which makes up a significant portion of its state and happens to have a declining population.

Weaknesses:

- Possibly unreliable evaluation of Memphis's energy use compared to greater TVA area
- More effective predictive models could have been selected instead of ensemble learning methods.

Q3: Beat the Heat

3.1 Defining the Problem

The problem asks us to create scores to rank the vulnerabilities of various neighborhoods in Memphis to a power outage during a heat wave. Then, given our results, we are asked to determine a single approach for incorporating this data into a strategy for managing heat waves. We used climate, housing, and social vulnerability data for creating our model.

3.2 Assumptions

Vulnerability equally considers social and environmental factor

- We decided to weight social and environmental factors each at 50% because we determined that they are both significant factors and did not have a way to determine which was "more important".

3.3 Variables

To calculate the model, we use the following census tract level data:

Variable	Category	Subcategory	Type	Stat
P_e	Physical - Environmental[11] -	Environmental	Heat	Urban Heat Index
S_i	Social[12]	Socioeconomi c	Income	Income w/in 138% of poverty line
S_u	Social	Socioeconomi c	Income	Unemployment
S_d	Social	Pop Vulnerability	Disability	% of persons
S_e	Social	Pop Vulnerability	Elderly	# of persons
S_c	Social	Pop Vulnerability	Children	# of persons
S_m	Social	Pop Vulnerability	Uninsured	# of persons

3.4 The Model

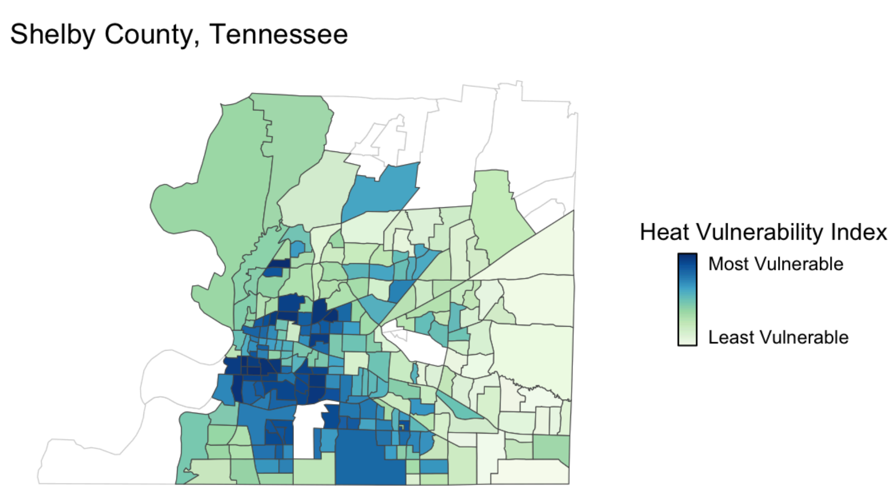
To create our heat vulnerability score, we use R Studio to create an average of the factors listed above. As explained in section 3.2, we evenly weight social and physical factors to create the index. First, we z-score each factor to mean center the values to allow for them to create an evenly weighted index because each variable uses a different scale. Then, we use the equation below (C(t), where t is the census tract) to create a weighted score:

$$C(t) = z(P_e)/2 + ((z(S_u)/2 + z(S_i)/2)/4 + (z(S_d)/4 + z(S_e)/4 + z(S_c)/4 + z(S_m)/4)/2$$

Equation 7: weighted score

Finally, we create a percentile based ranking of each Memphis census tract using the dplyr.

3.5 Results



Map of Shelby County Census Tracts by HVI

3.6 Discussion

Considering all that, the HVI scores should certainly be incorporated into decision making by the City of Memphis. Primarily, the best approach would be to allocate resources for measures to alleviate the effects of heat waves based on these scores. For example, the city could allocate funds for cooling centers to census tracts with higher HVI scores, and thereby reduce the HVI score of that area. This way, a high HVI score would serve as a warning label for municipal governments to demonstrate what the problem areas in a heat wave would be.

3.7 Strengths and Weaknesses

Percentile rank may fail to account for differences between tracts

- While percentile rank allows us to easily see the most advantaged and disadvantaged areas, it may ignore/oversimplify the magnitude of the difference between tracts.

This model uses a limited number of factors that may not fully reflect all factors that affect a community's vulnerability to a heat wave

- Due to time, we were forced to condense to a limited number of factors, but we believe that the factors we consider provide a relatively broad picture of the disadvantages communities face.

References

1. https://www.usatoday.com/story/news/nation/2024/07/12/how-to-stay-cool-without-air-conditioning/74336830007/
2. https://pmc.ncbi.nlm.nih.gov/articles/PMC4352572/#SM
3. https://nsrdb.nrel.gov/data-viewer
4. https://www.eia.gov/electricity/gridmonitor/dashboard/electric_overview/US48/US48
5. https://www.visualcrossing.com/weather-query-builder/Memphis
6. https://medium.com/@gigi.dattaradon/forecasting-electricity-demand-with-weather-data-a-machine-learning-approach-80f7270bfd38
7. https://usafacts.org/data/topics/people-society/population-and-demographics/our-changing-population/state/tennessee/?endDate=2022-01-01&startDate=2010-01-01
8. https://tva-azr-eastus-cdn-ep-tvawcm-prd.azureedge.net/cdn-tvawcma/docs/default-source/about-tva/fact-sheets/state-fact-sheet-tn-2023.pdf?sfvrsn=7e31336_1
9. https://tnsdc.utk.edu/estimates-and-projections/boyd-center-population-projections/
10. https://findenergy.com/tn/shelby-county-electricity/#production
11. https://www.climatecentral.org/climate-matters/urban-heat-islands-2023
12. https://data.census.gov/table/ACSST5Y2023.S2701?q=s2701&g=050XX00US47157$1400000

Appendix A: Code

Problem 1

```
// home types - 0 = Single-family, 1 = multi-family
let home1 = {"type": 0, "neighborhood": "East Memphis", "size": 950, "year": 1953,
"shade": "very"}
let home2 = {"type": 1, "neighborhood": "South Memphis", "size": 675, "year": 1967,
"shade": "not_very"}
let home3 = {"type": 1, "neighborhood": "Downtown", "size": 800, "year": 2003,
"shade": "not_at_all"}
let home4 = {"type": 0, "neighborhood": "Egypt", "size": 2993, "year": 1990, "shade":
"not_at_all"}

let hrly_temps =
[29.4,29.4,28.9,28.3,28.3,28.3,28.9,31.1,32.8,34.4,35.6,36.1,37.8,37.8,38.9,38.9,37.8,
37.2,36.1,34.4,33.3,32.8,32.2,31.7]
let hrly_dewpoints =
[24.4,24.4,23.9,23.9,23.9,23.9,23.9,24.4,24.4,24.4,24.4,24.4,23.9,23.3,22.2,22.2,22.8,
22.8,23.3,22.8,23.3,23.9,24.4,24.4]

let neighborhood_surrounding_mapping = {"East Memphis": 1,"South Memphis": 2,
"Downtown": 3, "Egypt": 2}
let neighborhood_radiation_mapping = {"East Memphis":
[0,0,0,0,0,0,0.504,1.1952,1.9044,2.5416,3.0492,3.3732,3.4956,3.4056,3.1068,2.6244,2.00
52,1.3032,0.6048,0.072,0,0,0,0],"South Memphis":
[0,0,0,0,0,0,0.5076,1.1988,1.9116,2.5452,2.1492,3.3804,3.4992,3.4128,3.1104,2.6244,2.0
052,1.3032,0.6012,0.0684,0,0,0,0], "Downtown":
[0,0,0,0,0,0,0.5148,1.206,1.9152,2.5524,1.1052,3.3768,3.4956,3.402,3.0996,2.6136,1.994
4,1.2924,0.5976,0.0684,0,0,0,0], "Egypt":
[0,0,0,0,0,0,0.5148,1.206,1.9116,2.5488,3.0492,3.3768,3.4956,3.402,3.096,2.6136,1.9944
,1.2924,0.5976,0.0684,0,0,0,0]}

let occupancy = [[.26, .21],[.03,.06],[.10,.15]] // [high rise, non high rise]
let construction = [[.20, .21, .45, .35],[.05,.06,.07,.02],[.14,.25,.01,.22]] //
[brick, asphalt, vinyl, wood]
let date = [[.26, .19, .12],[.06,.068,.04],[.18,.13,.10]] // [1912-1939, 1940-1970,
1970+]
let surroundings = [[.17, .23, .27, .09],[.05,.06,.07,.02],[.15,.16,.12,.10]] //
[concrete, res, yard, urban]
let no_ac = [.27,.06,.17]
```

```
let results = [[hrly_temps[0]],[hrly_temps[0]],[hrly_temps[0]],[hrly_temps[0]]]

for(let i = 1; i<hrly_temps.length; i++){
   results[0].push(homeCalculation(home1, i, 0))
   results[1].push(homeCalculation(home2, i, 1))
   results[2].push(homeCalculation(home3, i, 2))
   results[3].push(homeCalculation(home4, i, 3))
}
console.log(results)

function homeCalculation(home, hour, index){
   let temp_diff = hrly_temps[hour] - hrly_temps[hour-1]
   let dewpoint_diff = hrly_dewpoints[hour] - hrly_dewpoints[hour-1]
   let rad_diff = neighborhood_radiation_mapping[home["neighborhood"]][hour] -
neighborhood_radiation_mapping[home["neighborhood"]][hour-1]

   // temp
   let temp = results[index][hour-1] + temp_diff*no_ac[0]
   temp += occupancy[0][home["type"]]*temp_diff
   // most homes in dataset were brick, so we assume this is true here too. impact is
discussed in sensitivity analysis.
   temp += construction[0][0]*temp_diff
   if (1940 <= home["year"] <= 1970){
       temp += date[0][1] * temp_diff
   }
   else if (home["year"] > 1970){
       temp += date[0][2] * temp_diff
   }
   temp += surroundings[0][neighborhood_surrounding_mapping[home["neighborhood"]]] *
temp_diff

   // radiation
   temp += rad_diff*no_ac[2]
   temp += occupancy[2][home["type"]]*rad_diff
   // most homes in the dataset were brick, so we assume this is true here too. impact
is discussed in sensitivity analysis.
   temp += construction[2][0]*rad_diff
   if (1940 <= home["year"] <= 1970){
       temp += date[2][1] * rad_diff
   }
   else if (home["year"] > 1970){
       temp += date[2][2] * rad_diff
```

```
    }
    temp += surroundings[2][neighborhood_surrounding_mapping[home["neighborhood"]]] *
rad_diff


    // dewpoint
    temp += dewpoint_diff*no_ac[1]
    temp += occupancy[1][home["type"]]*dewpoint_diff
    // most homes in dataset were brick, so we assume this is true here too. impact is
discussed in sensitivity analysis.
    temp += construction[2][0]*dewpoint_diff
    if (1940 <= home["year"] <= 1970){
        temp += date[1][1] * dewpoint_diff
    }
    else if (home["year"] > 1970){
        temp += date[1][2] * dewpoint_diff
    }
    temp += surroundings[1][neighborhood_surrounding_mapping[home["neighborhood"]]] *
dewpoint_diff

    return temp;
}
```

Problem 2

M3 Rendition March 1, 2025 [ ]: import pandas as pd import numpy as np import os import datetime import seaborn as sns import matplotlib.pyplot as plt from sklearn.model_selection import train_test_split from sklearn.linear_model import LinearRegression from sklearn.ensemble import RandomForestRegressor from sklearn.ensemble import HistGradientBoostingClassifier from xgboost import XGBRegressor from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,␣ ↪r2_score from IPython.display import display pd.set_option('display.max_columns', 100) start_mth = datetime.datetime(2021, 1, 1) end_mth = datetime.datetime(2021, 12, 31) ## set the period of forcast in 2021 start_mth = datetime.datetime(2021, 1, 1) end_mth = datetime.datetime(2021, 12, 31) ## Read Weather Data weather = pd.read_csv('open-meteo-35.11N90.00W91m.csv', encoding='utf-8') print(weather.head()) # Pivot the weather dataframe # weather = weather.pivot(index='date', columns='index',␣ ↪values=['temperature_2m_max (°F)', # 'temperature_2m_min (°F)', # 'temperature_2m_mean (°F)', # 'precipitation_sum (inch)', # 'wind_speed_10m_max (m/s)']) # Flatten multi-index columns # weather.columns = ['_'.join(col).strip() for col in weather.columns.values] # Reset index 1 weather = weather.reset_index() # Set date column as date type weather.date = pd.to_datetime(weather.date, format='%Y-%m-%d') selected_col = ['date','temperature_2m_max (°F)', 'temperature_2m_min (°F)',␣ ↪'temperature_2m_mean (°F)', 'precipitation_sum (inch)'] weather = weather[selected_col] weather [ ]: ## Read Eletricity Demand Data usage = pd.read_csv('tva_data.csv', sep=',') usage.columns = usage.columns.str.lower() print(usage.head()) ## Filter only date and total demand columns selected_col = ['date','tva demand (mwh)'] usage = usage[selected_col] ## convert column type usage['date'] = pd.to_datetime(usage['date']) usage['tva demand (mwh)'] = usage['tva demand (mwh)'].astype(float) usage [ ]: ## Filter usage within start and end month usage = usage[(usage['date']>=start_mth) & (usage['date'] <= end_mth)] ## sum average daily demand and prepare data usage.index = usage.date usage_d = usage.resample('D').mean() usage_d.date = pd.to_datetime(usage_d.date).dt.date usage_d_column = ['date', 'tva demand (mwh)'] usage_d.columns = usage_d_column usage_d = usage_d.reset_index(drop=True) usage_d.date = pd.to_datetime(usage_d.date) ### merge usage and weather dataframe together merge = pd.merge(weather, usage_d, how = 'inner', on = 'date') merge = merge.reindex(sorted(merge.columns), axis=1) merge.head() [ ]: ## correlation heat map merge_wo_date = merge.drop(columns='date') corr_matrix = merge_wo_date.corr() ##plot the heat map 2 plt.figure(figsize=(15,8)) sns.heatmap(corr_matrix, annot = True, cmap = 'coolwarm', fmt = ".2f",␣ ↪linewidths=.5, annot_kws={'size':7}) plt.xticks(fontsize=8) plt.yticks(fontsize=8) plt.title('Correlation Heatmap') plt.show() [ ]: ## Split data into feature and target variable X = merge.drop(columns=['date','tva demand (mwh)']) y = merge['tva demand (mwh)'] ## Split data into train and test set X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣ ↪random_state=42) ## Set the model dictionary models = { 'Linear Regression': LinearRegression() , 'Random Forest Regressor': RandomForestRegressor(random_state = 42) , 'XGBoost Regressor' :

XGBRegressor(random_state = 42) } X_test [ ]: ## Run on Actual Data pred_weather = pd.read_csv('open-meteo-35.10N90.00W91m.csv', encoding='utf-8') pred_weather['year'] = pd.DatetimeIndex(pred_weather["date"]).year pred_weather_f = pred_weather.drop(columns=['date','year','wind_speed_10m_mean␣ ↪(mp/h)']) pred_weather_f = pred_weather_f.reindex(sorted(pred_weather_f.columns), axis=1) pred_weather_f [ ]: ## Predict and Evaluation for model_name, model in models.items(): model.fit(X_train, y_train) y_pred = model.predict(X_test) pred_weather['predicted_' + model_name] = model.predict(pred_weather_f) mse = mean_squared_error(y_test, y_pred) mape = mean_absolute_percentage_error(y_test, y_pred) * 100 r_squared = r2_score(y_test, y_pred) ## print model accuracy print(f"{model_name}") 3 print(f"Mean Squared Error = {mse}") print(f"Mean Absolute Percentage Error = {mape: .2f}%") print(f"R-squared = {r_squared:.2f}") ## plot scatter, and line comparing test data and prediction plt.figure(figsize=(6,6)) plt.scatter(y_test, y_pred, alpha=0.5) plt.xlabel('Actual') plt.ylabel('Predicted') p1 = max(max(y_pred), max(y_test)) p2 = min(min(y_pred), min(y_test)) plt.plot([p1, p2], [p1, p2], color = 'r', linestyle = '-', lw=2) plt.grid(True) plt.show() display(pred_weather) [ ]: results = [] # Loop through all years for i in range(26): # Filter all entries in a given year of the loop temp = pred_weather[pred_weather['year'] == 2025+i] # Sort by value and assign greatest value to the results value_1 = temp.sort_values('predicted_Linear Regression', ascending=False) value_2 = temp.sort_values('predicted_Random Forest Regressor',␣ ↪ascending=False) value_3 = temp.sort_values('predicted_XGBoost Regressor', ascending=False) results.append({"year": 2025+i}) results[i]["lin_reg"] = value_1.iloc[0]["predicted_Linear Regression"] results[i]["rf"] = value_1.iloc[0]["predicted_Random Forest Regressor"] results[i]["xgb"] = value_1.iloc[0]["predicted_XGBoost Regressor"] # Display results results = pd.DataFrame(results, columns=['year', 'lin_reg','rf','xgb']) results 4

Problem 3

```
# Calculating Heat Vulnerability Index
# 2.28.25

library(tidyverse)

source(REDACTED)

# Read in Census Data for Vulnerability
census_vulnerable_data <-
read_csv("./Data/ACSST5Y2023.S2701_2025-02-28T193412/ACSST5Y2023.S2701-Data.csv")

# Renaming columns with first row
colnames(census_vulnerable_data) <- census_vulnerable_data[1,]
```

```
# Removing duplicate column name row
census_vulnerable_data <- census_vulnerable_data[-1,]

# Cleaning dataset to prepare for operations on data
census_vul_clean <- census_vulnerable_data %>%
  select(Geography, population = `Estimate!!Total!!Civilian noninstitutionalized
population`,
         child_pop = `Estimate!!Total!!Civilian noninstitutionalized
population!!AGE!!Under 6 years`,
         senior_pop = `Estimate!!Total!!Civilian noninstitutionalized population!!AGE!!65
years and older`,
         disabled_pop = `Estimate!!Total!!Civilian noninstitutionalized
population!!DISABILITY STATUS!!With a disability`,
         unemployed_pop = `Estimate!!Total!!Civilian noninstitutionalized
population!!EMPLOYMENT STATUS!!Civilian noninstitutionalized population 19 to 64
years!!In labor force!!Unemployed`,
         poverty_pop = `Estimate!!Total!!Civilian noninstitutionalized population!!RATIO
OF INCOME TO POVERTY LEVEL IN THE PAST 12 MONTHS!!Civilian
noninstitutionalized population for whom poverty status is determined!!Below 138 percent of the
poverty threshold`,
         uninsured_pop = `Estimate!!Uninsured!!Civilian noninstitutionalized
population`)%>%
  separate(Geography, c(NA, "geoid_tract"), "US")%>%
  mutate(geoid_tract = as.numeric(geoid_tract)) %>%
  mutate(population = as.numeric(population)) %>%
  mutate(child_pop = as.numeric(child_pop)) %>%
  mutate(senior_pop = as.numeric(senior_pop))%>%
  mutate(disabled_pop = as.numeric(disabled_pop)) %>%
  mutate(unemployed_pop = as.numeric(unemployed_pop))%>%
  mutate(poverty_pop = as.numeric(poverty_pop)) %>%
  mutate(uninsured_pop = as.numeric(uninsured_pop))

# Reading in heat data
heat_island_data <- read_csv("./Data/heat_island_memphis.csv")

# Merging dataset and converting populations to percentages before converting data to
z-scores
big_data <- left_join(heat_island_data, census_vul_clean, by = "geoid_tract")%>%
  filter(!is.na(population), population != 0) %>%
```

```
  mutate_at(vars(child_pop:uninsured_pop),list(percent=~./population))%>%
  mutate_at(vars(celsius_urban_heat, fahrenheit_urban_heat,
child_pop_percent:uninsured_pop_percent), scale)

# Creating index with percentile rank weights
hvi <- big_data %>%
  mutate(hvi = percent_rank(fahrenheit_urban_heat*.5 + (senior_pop_percent*.25 +
disabled_pop_percent*.25 + uninsured_pop_percent*.25 + child_pop_percent*.25)*.25 +
(poverty_pop_percent*.5 + unemployed_pop_percent*.5)*.25)) %>%
  select(geoid_tract, hvi)

write_csv(hvi, "./Data/hvi.csv")

# HVI Map

library(tidyverse)
library(tidycensus)
library(tidygeocoder)
library(RColorBrewer)

source(REDACTED)

hvi_data <- read_csv("./Data/hvi.csv")

census_api_key(key = api_key)

options(tigris_use_cache = TRUE)

library(tigris)

# Gets geometries for tracts in 2020 census
shelby_tract_geom <- get_acs(
  geography = "tract",
  variables = "B01002_001",
  state = "TN",
  county = "Shelby",
  year = 2020,
  geometry = TRUE
)%>%
  mutate(tract_num = as.numeric(gsub("[^0-9.]", "", NAME)))%>%
```

```
  mutate(geoid_tract = as.numeric(GEOID))

tract_geom <- left_join(hvi_data, shelby_tract_geom, by = c("geoid_tract"))

ggplot(data = tract_geom, aes(fill = hvi)) +
  geom_sf(aes(geometry = geometry))

hvi_color <- brewer.pal(n = 9, name = "GnBu")

bwidth <- 0.12
bheight <- 0.6

hvi_plot <- ggplot(data = shelby_tract_geom) +
  geom_sf(color = "lightgray", fill = "white") +
  geom_sf(data = tract_geom, aes(geometry = geometry, fill = hvi))+
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        rect = element_blank()) +
  labs(fill = "Heat Vulnerability Index")+
  scale_fill_gradientn(
    colours = hvi_color, # jet_colors(10)
    # may want to hard code these for comparison across groups
    limits = c(0, 1), # these should be determined from the uninterpolated data (i think)
    guide = "colourbar",
    breaks = c(0,.1,.9, 1),
    labels = c("", "Least Vulnerable" , "Most Vulnerable", "")
  )+
  guides(
    fill = guide_colourbar(
      title.position = "top",
      title.hjust = 0.5,
      frame.colour = "black",
      ticks.colour = NA,
      barwidth = unit(bwidth, "in"),
      barheight = unit(bheight, "in")
    )
  )+
  theme(legend.position.inside = c(1.1,0.75))+
  labs(title = "Shelby County, Tennessee")
```

Appendix B: HVI Scores

geoid_tract	hvi
47157000100	0.4358974359
47157000200	0.3931623932
47157000300	0.2820512821
47157000400	0.3162393162
47157000600	0.9273504274
47157000700	0.9743589744
47157000800	0.4017094017
47157000900	0.8290598291
47157001100	0.9786324786
47157001200	0.7777777778
47157001300	0.8504273504
47157001400	0.7863247863
47157001500	0.811965812
47157001600	0.1709401709
47157001700	0.4700854701
47157001900	0.8803418803
47157002000	0.8461538462
47157002100	0.4273504274
47157002400	0.7905982906
47157002500	0.688034188
47157002600	0.4572649573
47157002700	0.6752136752
47157002800	0.9401709402
47157002900	0.4444444444
47157003000	0.7692307692
47157003100	0.4487179487
47157003200	0.5769230769
47157003300	0.5299145299
47157003400	0.594017094
47157003500	0.3974358974
47157003600	0.6239316239
47157003700	0.8333333333
47157003800	0.6367521368
47157003900	0.735042735
47157004200	0.5

47157004300	0.2521367521
47157004500	0.858974359
47157004600	0.7094017094
47157005000	0.9957264957
47157005300	0.7008547009
47157005500	0.9700854701
47157005600	0.8076923077
47157005700	0.9188034188
47157005800	0.9871794872
47157005900	1
47157006000	0.9316239316
47157006200	0.9615384615
47157006300	0.4743589744
47157006400	0.7735042735
47157006500	0.764957265
47157006600	0.5555555556
47157006700	0.9230769231
47157006800	0.8888888889
47157006900	0.9017094017
47157007000	0.9102564103
47157007100	0.4316239316
47157007200	0.452991453
47157007300	0.6153846154
47157007400	0.5854700855
47157007500	0.8376068376
47157007810	0.905982906
47157007821	0.8162393162
47157007822	0.7307692308
47157007900	0.9658119658
47157008000	0.6837606838
47157008110	0.9145299145
47157008120	0.952991453
47157008200	0.8717948718
47157008500	0.1367521368
47157008600	0.4914529915
47157008700	0.6324786325
47157008800	0.7948717949
47157008900	0.3034188034
47157009100	0.3760683761
47157009201	0.4230769231

47157009202	0.08974358974
47157009300	0.5726495726
47157009400	0.5256410256
47157009501	0.5341880342
47157009502	0.5598290598
47157009600	0.4871794872
47157009700	0.6965811966
47157009800	0.3418803419
47157009901	0.405982906
47157009902	0.4145299145
47157010001	0.6495726496
47157010002	0.3205128205
47157010120	0.3888888889
47157010121	0.9914529915
47157010122	0.8632478632
47157010210	0.311965812
47157010220	0.2991452991
47157010300	0.4786324786
47157010500	0.8931623932
47157010610	0.7393162393
47157010620	0.8205128205
47157010630	0.8418803419
47157010710	0.7478632479
47157010720	0.6923076923
47157010810	0.6111111111
47157010820	0.7179487179
47157011010	0.7564102564
47157011020	0.6452991453
47157011100	0.9572649573
47157011200	0.9487179487
47157011300	0.8034188034
47157011401	0.6709401709
47157011402	0.5811965812
47157011500	0.9829059829
47157011600	0.9444444444
47157011700	0.9358974359
47157011800	0.7435897436
47157020101	0.358974359
47157020102	0.3547008547
47157020221	0.1495726496

47157020222	0.6025641026
47157020511	0.2051282051
47157020521	0.0811965812
47157020523	0.3675213675
47157020524	0.188034188
47157020531	0.07264957265
47157020532	0.2094017094
47157020541	0.5427350427
47157020542	0.3376068376
47157020543	0.2863247863
47157020544	0.3290598291
47157020610	0.5683760684
47157020621	0.1452991453
47157020622	0.7222222222
47157020632	0.235042735
47157020633	0.5897435897
47157020634	0.5641025641
47157020635	0.5982905983
47157020651	0.6068376068
47157020652	0.5042735043
47157020653	0.09829059829
47157020654	0.1666666667
47157020655	0.5170940171
47157020656	0.0641025641
47157020657	0.1068376068
47157020658	0.2564102564
47157020833	0.1111111111
47157020834	0.1623931624
47157020836	0.2264957265
47157020837	0.01282051282
47157021020	0.03418803419
47157021021	0.05128205128
47157021022	0.1581196581
47157021023	0.01709401709
47157021111	0.2735042735
47157021112	0.1752136752
47157021113	0.2777777778
47157021121	0.2905982906
47157021122	0.264957265
47157021124	0.5128205128

47157021125	0.5085470085
47157021126	0.2307692308
47157021135	0.5213675214
47157021136	0.1196581197
47157021138	0.1239316239
47157021139	0.02991452991
47157021140	0.03846153846
47157021141	0.2435897436
47157021142	0.1923076923
47157021143	0.4829059829
47157021144	0.141025641
47157021311	0.2008547009
47157021312	0.547008547
47157021320	0.4188034188
47157021331	0.1794871795
47157021333	0.6538461538
47157021334	0.2692307692
47157021341	0.4957264957
47157021351	0.2393162393
47157021352	0.1025641026
47157021354	0.1324786325
47157021355	0.1538461538
47157021356	0.1282051282
47157021357	0.04700854701
47157021410	0.05982905983
47157021420	0.09401709402
47157021430	0.1965811966
47157021530	0.07692307692
47157021541	0.008547008547
47157021542	0.02136752137
47157021543	0
47157021544	0.06837606838
47157021545	0.2222222222
47157021546	0.004273504274
47157021547	0.1837606838
47157021548	0.02564102564
47157021611	0.04273504274
47157021612	0.05555555556
47157021613	0.08547008547
47157021620	0.3461538462

47157021710	0.3247863248
47157021721	0.5512820513
47157021724	0.5384615385
47157021725	0.641025641
47157021731	0.7606837607
47157021744	0.6666666667
47157021745	0.2136752137
47157021746	0.7051282051
47157021747	0.4658119658
47157021751	0.6581196581
47157021752	0.2606837607
47157021753	0.1153846154
47157021754	0.3333333333
47157021755	0.3803418803
47157021756	0.8760683761
47157021757	0.6196581197
47157021758	0.7820512821
47157021759	0.6282051282
47157021760	0.4102564103
47157021900	0.2478632479
47157022023	0.8547008547
47157022024	0.8974358974
47157022025	0.8675213675
47157022026	0.8846153846
47157022111	0.7521367521
47157022121	0.6623931624
47157022122	0.7136752137
47157022130	0.3504273504
47157022131	0.8247863248
47157022132	0.6794871795
47157022210	0.4615384615
47157022220	0.4401709402
47157022310	0.3717948718
47157022321	0.2948717949
47157022322	0.3076923077
47157022330	0.3632478632
47157022410	0.2179487179
47157022500	0.7264957265
47157022600	0.7991452991
47157022700	0.3846153846