Analysis of Homelessness by HUD CoC

Identifying key measures to moving our unsheltered neighbors off the streets and into a warm bed

Project Team

- Jonathan L.
- Chris B.
- Michal U.
- Scott S.
- Taylor W.

Project Background

- On a single night in January 2020
    - 580,466 people - about 18 of every 10,000 people in the United States - experienced homelessness across the United States.
    - Six in 10 people experiencing homelessness (61%), were staying in sheltered locations, and nearly four in 10 (39%) were unsheltered.
- The US government’s Department of Housing and Urban development (HUD) funds the continuum of care (CoC) in major cities and select rural areas across the United States.
- CoC - A program providing funding for efforts by nonprofit providers and state/local governments to rehouse homeless individuals/families swiftly while minimizing the trauma and dislocation caused by homelessness.

Website Overview

Data Sources – HUD Resources

- HUD requires CoCs to perform an annual Point In Time (PIT) count to develop an estimate of homeless individuals & families within the bounds of the CoC.[1]
- Each CoC must also report a Housing Inventory Count (HIC) that provides information on both the counts and classifications of available shelter beds.
- The PIP and the HIC files, with annual records from 2007–2020, provide the backbone of the dataset utilized in the project.

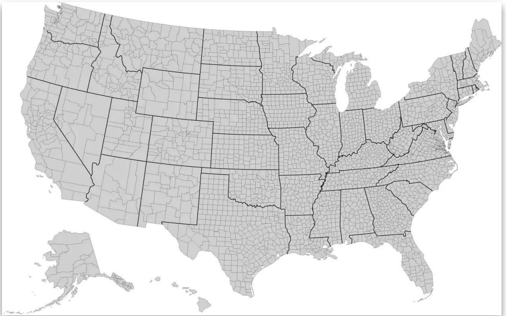
EXHIBIT 1.3: **Change in Number of People Experiencing Homelessness**
2007–2020

	Change 2019–2020		Change 2010–2020		Change 2007–2020	
	#	%	#	%	#	%
Total	**12,751**	**2.2%**	**-56,611**	**-8.9%**	**-66,792**	**-10.3%**
Sheltered	-2,036	-0.6%	-49,157	-12.2%	-37,015	-9.5%
Unsheltered	14,787	7.0%	-7,454	-3.2%	-29,777	-11.6%

Key pieces of data from the HIC and PIT files include counts of homeless and unsheltered individuals per CoC as well as counts and types of available beds per CoC.

[1] The 2020 Annual Homeless Assessment Report (AHAR) to Congress; Jan 2021. Available at: https://www.huduser.gov/portal/sites/default/files/pdf/2020-AHAR-Part-1.pdf

Data Sources – County-Level US Data



- Leveraging previous work[1], we identified a table that connected CoC numbers to each county in the US, along with the county- and city-specific Federal Information Processing Standards (FIPS) code.
- CoC information was merged with county-level population[2] and unemployment[3] data using the Pandas library in Jupyter Notebook, joined on the county-specific FIPS codes, and converted to a .csv file.

	FIPS	county	state	number	Geographic Area	County	2010_population	2011_population	2012_population	2013_population	...	Unemployment_rate_2013	Unemployment_rate_
0	1001	Autauga	AL	AL-507	.Autauga County, Alabama	Autauga	54,773	55,227	54,954	54,727	...	6.3	
1	1003	Baldwin	AL	AL-501	.Baldwin County, Alabama	Baldwin	183,112	186,558	190,145	194,885	...	6.7	
2	1005	Barbour	AL	AL-507	.Barbour County, Alabama	Barbour	27,327	27,341	27,169	26,937	...	10.4	
3	1007	Bibb	AL	AL-507	.Bibb County, Alabama	Bibb	22,870	22,745	22,667	22,521	...	8.0	

[1] https://github.com/windyseng/predicting_homelessness,
[2] https://www.census.gov/data/datasets/time-series/demo/popest/2010s-counties-total.html#par_textimage_739801612,
[3] https://www.ers.usda.gov/data-products/county-level-data-sets/download-data/

Data Cleaning

- Each data source had to be converted from a excel workbook to a workable pandas dataframe
- Unused data was removed from all DataFrames
- Population and unemployment data was transformed from county data into Coc data with basic table joining and grouping functions

	Total Beds (ES, TH, SH)									
CoC Number	Total Year-Round Beds (ES, TH, SH)	Total Non-DV Year-Round Beds (ES, TH, SH)	Total HMIS Year-Round Beds (ES, TH, SH)	HMIS Participation Rate for Year-Round Beds (ES, TH, SH)	Total Year-Round Beds (ES)	Total Year-Round Beds (TH)	Total Year-Round Beds (SH)	Total Units for Households with Children (ES, TH, SH)	Total Beds for Households with Children (ES, TH, SH)	Total Beds for Households without Children (ES, TH, SH)
AK-500	920	843	666	72.39%	720	200	0	74	311	598
AK-501	965	504	456	47.25%	690	275	0	120	411	543
AL-500	888	767	612	68.92%	545	309	34	101	282	594
AL-501	396	321	321	81.06%	274	122	0	66	214	182
AL-502	285	95	0	0.00%	118	167	0	77	109	160

2020 | 2019 | 2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2009 | 2008 | 2007 | Rev ...

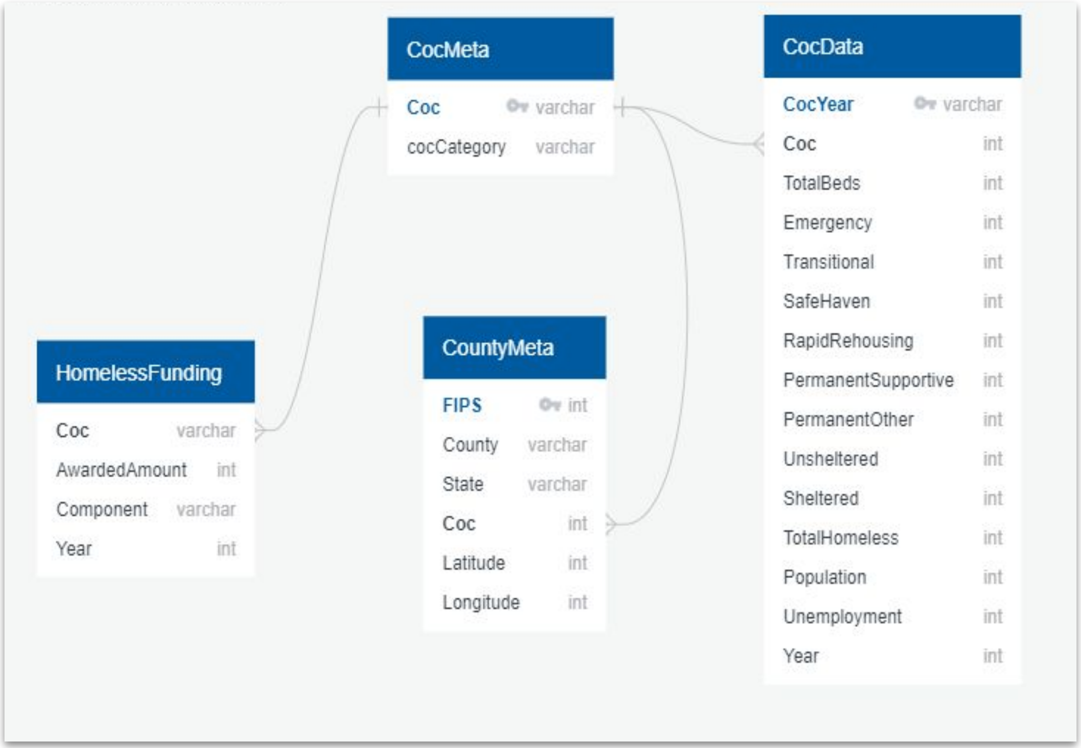
	Coc	TotalBeds	Emergency	Transitional	SafeHaven	RapidRehousing	PermanentSupportive	PermanentOther	Year	CocYear
0	AK-500	920.0	720.0	200.0	0.0	212.0	558.0	92.0	2020	AK-500 2020
1	AK-501	965.0	690.0	275.0	0.0	47.0	405.0	0.0	2020	AK-501 2020
2	AL-500	888.0	545.0	309.0	34.0	235.0	1823.0	32.0	2020	AL-500 2020

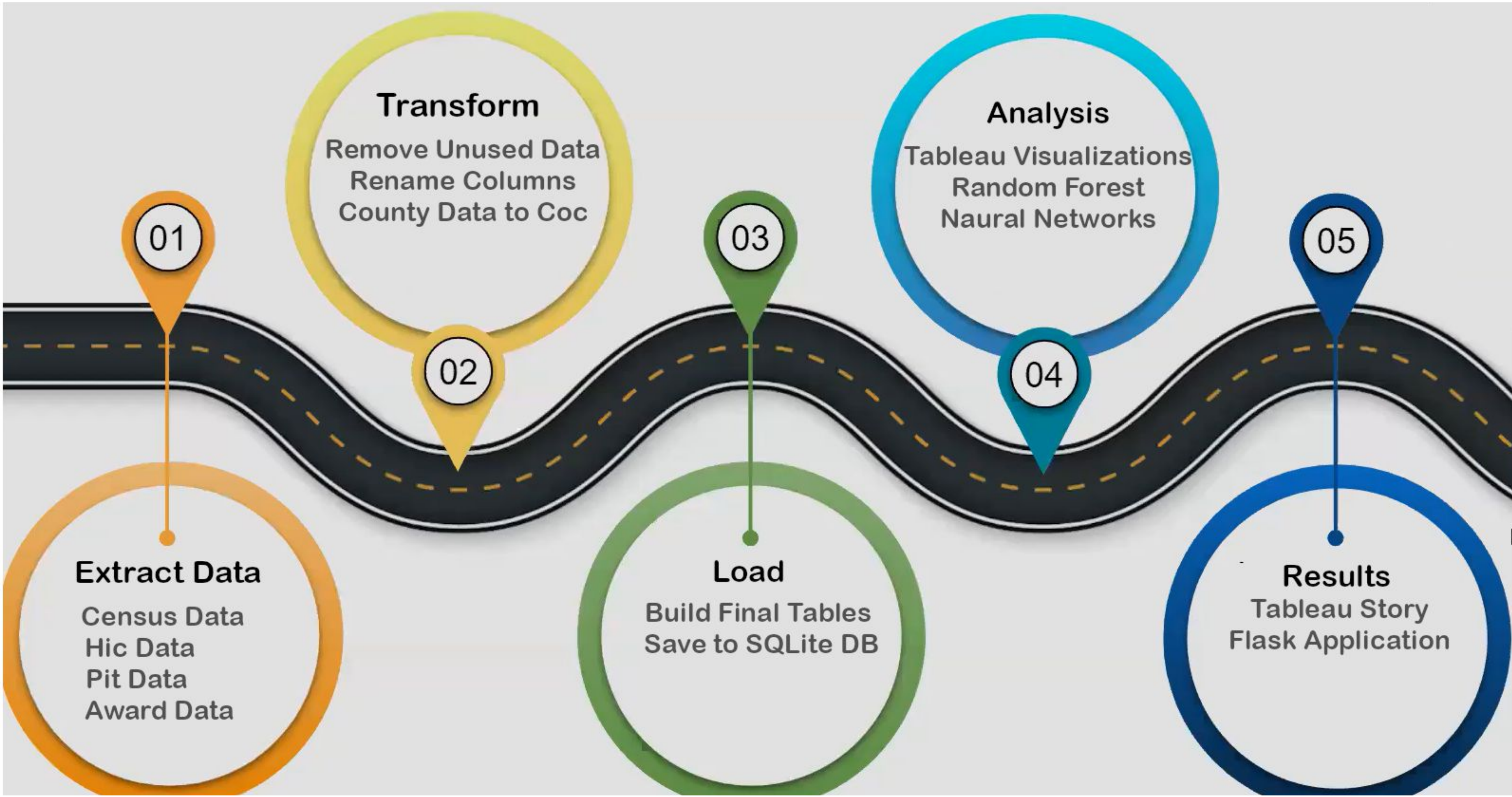
Database Structure

SQLite3 was used to create a database so that it could be easily accessed in python, and integrated into our ETL pipeline.

Tables

- HomelessFunding
- CocMeta
- CountyMeta
- CocData




01
Extract Data
Census Data
Hic Data
Pit Data
Award Data
02
Transform
Remove Unused Data
Rename Columns
County Data to Coc
03
Load
Build Final Tables
Save to SQLite DB
04
Analysis
Tableau Visualizations
Random Forest
Naural Networks
05
Results
Tableau Story
Flask Application

Machine Learning Goals

We would like to understand how to reduce the number of Unsheltered Individuals. What is the relationship between investments and outcomes? Can we predict outcomes based on investment (amount and type)? Are there other CoC specific data (total population, unemployment rate) that are important to consider?

Outcomes

Total Homeless Population = Sheltered Individuals + Unsheltered Individuals

Goal: Reduce the percentage of Unsheltered Individuals

Investments

Investments in shelter beds

Emergency, Transitional, Safe Haven

Investments in permanent housing options

Permanent Supportive, Rapid Rehousing, Permanent Other (housing vouchers)

Data Preprocessing for modeling

Challenge: Every CoC represents geographies of different sizes so we need a way to calibrate across all CoCs (otherwise our analysis will be dominated by LA, NYC, and other large cities).

Potential Solutions: Preprocess our data, converting the relevant data columns into either:

1. A percentage of each CoC’s total population.
2. A percentage of each CoC’s total homeless population.
3. Bucket the data by type of CoC (Urban, Rural)

After evaluating several models, we chose #2. (more detail in github readme)

Example transformation code:

```
In [32]:  df2['Unsheltered_perc_tot'] = (df2['Unsheltered'] / df2['TotalHomeless']) *100
```

Loading into Machine Learning Model

Starting with our Random Forest model, we connected to our SQLlite database to pull the relevant CoC, PIT, and HIC information into a dataframe. The data was then preprocessed using pandas.

Connecting to the SQL lite database

```
In [70]: # Initial imports.
         import pandas as pd
         import numpy as np
         from path import Path
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.datasets import make_regression
         from sklearn.preprocessing import StandardScaler
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
         from sklearn.tree import export_graphviz
         import sqlite3
```

```
In [71]: conn = sqlite3.connect('../HomelessData.db')
         c = conn.cursor()
```

```
In [72]: # Loading data
         df = pd.read_sql("SELECT * FROM CocData WHERE year < 2020", con = conn)

         df.head(20)
```

Out[72]:

	Coc	Total	Emergency	Transitional	SafeHaven	RapidRehousing	PermanentSupportive	PermanentOt
0	AK-500	1033	799	234	0	171	601	
1	AK-501	845	575	270	0	67	386	
2	AL-500	827	516	277	34	106	1740	
3	AL-501	394	261	133	0	31	240	
4	AL-502	189	106	83	0	0	42	

Converting to percent of total homeless population

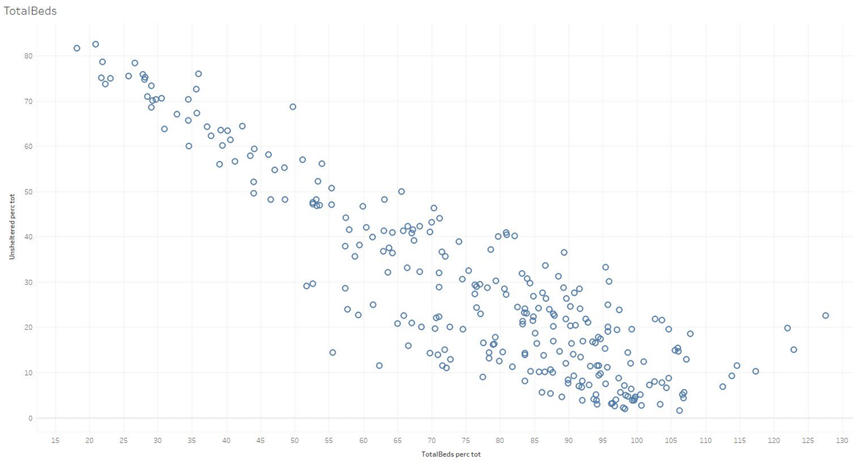
```
In [75]: df2['Unsheltered_perc_tot'] = (df2['Unsheltered'] / df2['Total:1']) *100

In [76]: df2['TotalBeds_perc_tot'] = (df2['Total'] / df2['Total:1']) *100

In [77]: df2['Emergency_perc_tot'] = (df2['Emergency'] / df2['Total:1']) *100

In [78]: df2['Transitional_perc_tot'] = (df2['Transitional'] / df2['Total:1']) *100

In [79]: df2['SafeHaven_perc_tot'] = (df2['SafeHaven'] / df2['Total:1']) *100

In [80]: df2['PermanentSupportive_perc_tot'] = (df2['PermanentSupportive'] / df2['Total:1']) *100

In [81]: df2['PermanentOther_perc_tot'] = (df2['PermanentOther'] / df2['Total:1']) *100

In [82]: df2['RapidRehousing_perc_tot'] = (df2['RapidRehousing'] / df2['Total:1']) *100
```

Dropping original columns

```
df3 = df2.drop(['Emergency', 'Transitional', 'SafeHaven', 'RapidRehousing', 'PermanentSupportive', 'PermanentOther', 'Unshelt
```

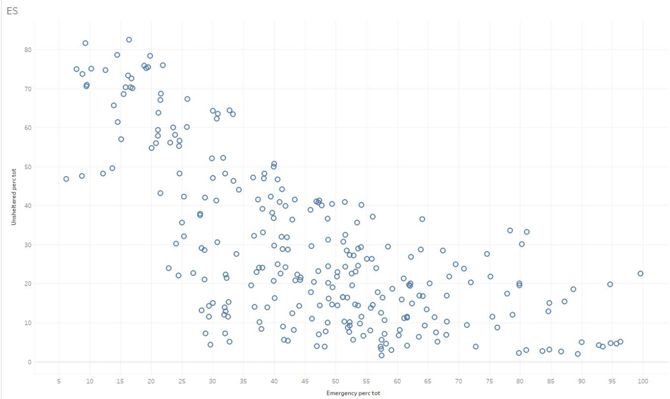
Visualizing the Data to add sanity check to ML models

Unsheltered % of total homeless pop



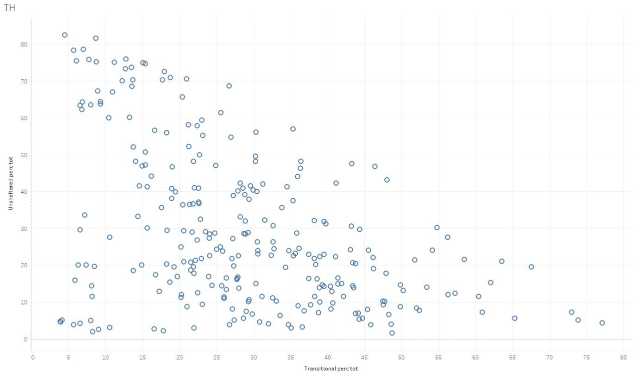
Total Shelter beds % of total homeless pop

Unsheltered % of total homeless pop



Emergency Shelter beds % of total homeless pop

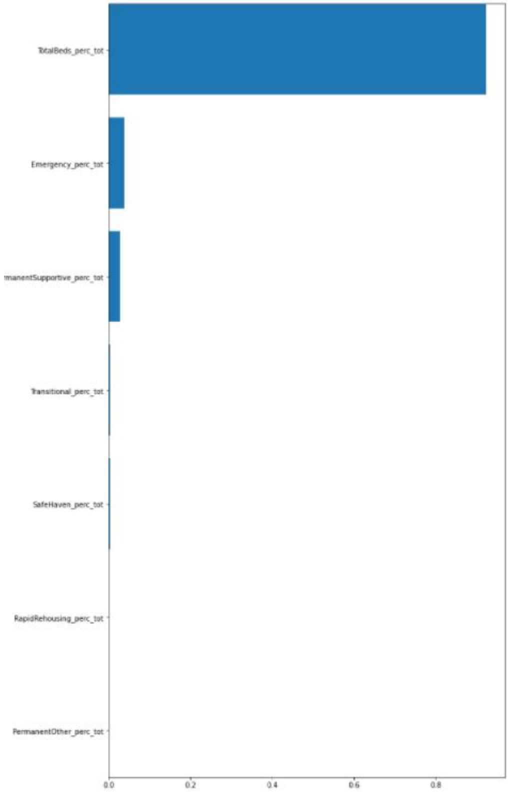
Unsheltered % of total homeless pop



Transitional beds % of total homeless pop

Random Forest

```
In [88]:  # Define the target set.
          y = df3["Unsheltered_perc_tot"].ravel()
          y[:5]

Out[88]: array([ 8.73087309, 22.11055276, 33.23139653, 40.        , 51.53846154])

In [89]:  # Splitting into Train and Test sets.
          X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=78)

In [90]:  regr = RandomForestRegressor(max_depth=3, random_state=0)

In [91]:  reg_model = regr.fit(X, y)

In [92]:  importances = reg_model.feature_importances_
          importances

Out[92]: array([9.23422410e-01, 3.91236446e-02, 4.24955130e-03, 4.17659831e-03,
                2.87525192e-02, 0.00000000e+00, 2.75276383e-04])

In [93]:  # We can sort the features by their importance.
          sorted(zip(reg_model.feature_importances_, X.columns), reverse=True)

Out[93]: [(0.9234224101984242, 'TotalBeds_perc_tot'),
          (0.03912364458868498, 'Emergency_perc_tot'),
          (0.028752519220807318, 'PermanentSupportive_perc_tot'),
          (0.00424955129586764, 'Transitional_perc_tot'),
          (0.004176598312966011, 'SafeHaven_perc_tot'),
          (0.0002752763832496561, 'RapidRehousing_perc_tot'),
          (0.0, 'PermanentOther_perc_tot')]
```



Random Forest with Population, Unemployment, and total number of homeless

In [35]:
```python
# Define the target set.
y = df3["Unsheltered_perc_tot"].ravel()
y[:5]
```
Out[35]:
```
array([ 8.73087309,  8.59232176, 13.74113475, 21.71945701, 14.81788079])
```

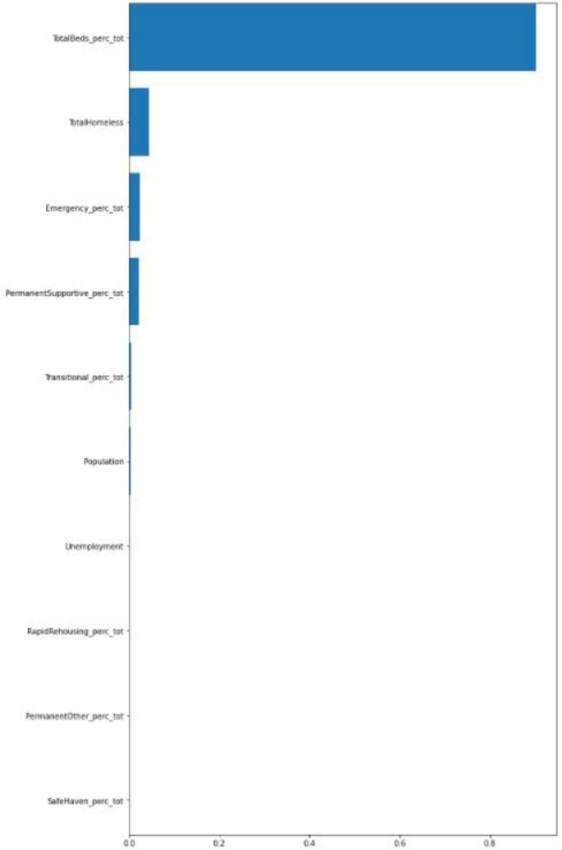
In [36]:
```python
# Splitting into Train and Test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=78)
```

In [37]:
```python
regr = RandomForestRegressor(max_depth=3, random_state=0)
```

In [38]:
```python
reg_model = regr.fit(X, y)
```

In [39]:
```python
importances = reg_model.feature_importances_
importances
```
Out[39]:
```
array([4.34197794e-02, 2.89082978e-03, 4.22231544e-04, 9.03091483e-01,
       2.34493219e-02, 5.30305761e-03, 0.00000000e+00, 2.14232964e-02,
       0.00000000e+00, 0.00000000e+00])
```

In [40]:
```python
# We can sort the features by their importance.
sorted(zip(reg_model.feature_importances_, X.columns), reverse=True)
```
Out[40]:
```
[(0.9030914833875665, 'TotalBeds_perc_tot'),
 (0.04341977937150892, 'TotalHomeless'),
 (0.023449321900400662, 'Emergency_perc_tot'),
 (0.021423296412410545, 'PermanentSupportive_perc_tot'),
 (0.0053030576071620966, 'Transitional_perc_tot'),
 (0.0028908297772433024, 'Population'),
 (0.00042223154370774096, 'Unemployment'),
 (0.0, 'SafeHaven_perc_tot'),
 (0.0, 'RapidRehousing_perc_tot'),
 (0.0, 'PermanentOther_perc_tot')]
```



Multiple Linear Regression analysis

```
> summary(lm(Unsheltered_perc_tot~Population+Unemployment+TotalBeds_perc_tot+Emergency_perc_tot+Trans
itional_perc_tot+SafeHaven_perc_tot+PermanentSupportive_perc_tot+PermanentOther_perc_tot+RapidRehousi
ng_perc_tot, data = all_years_perc_total_homeless))

Call:
lm(formula = Unsheltered_perc_tot ~ Population + Unemployment +
    TotalBeds_perc_tot + Emergency_perc_tot + Transitional_perc_tot +
    SafeHaven_perc_tot + PermanentSupportive_perc_tot + PermanentOther_perc_tot +
    RapidRehousing_perc_tot, data = all_years_perc_total_homeless)

Residuals:
    Min      1Q  Median      3Q     Max
-58.891 -10.714  -1.215  11.046  82.547

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  5.721e+01  1.416e+00  40.391  < 2e-16 ***
Population                   7.000e-07  2.697e-07   2.595 0.009510 **
Unemployment                 1.551e+00  1.943e-01   7.985 2.24e-15 ***
TotalBeds_perc_tot          -1.536e+00  2.204e-01  -6.971 4.16e-12 ***
Emergency_perc_tot           1.174e+00  2.206e-01   5.319 1.15e-07 ***
Transitional_perc_tot        1.046e+00  2.213e-01   4.725 2.44e-06 ***
SafeHaven_perc_tot                  NA         NA      NA       NA
PermanentSupportive_perc_tot -5.138e-02  5.431e-03  -9.461  < 2e-16 ***
PermanentOther_perc_tot     -3.272e-02  9.043e-03  -3.618 0.000304 ***
RapidRehousing_perc_tot     -2.444e-02  1.512e-02  -1.616 0.106191
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.01 on 2190 degrees of freedom
Multiple R-squared:  0.5155,	Adjusted R-squared:  0.5138
F-statistic: 291.3 on 8 and 2190 DF,  p-value: < 2.2e-16
```

Many of the features are statistically significant

Neural Network Structure

Can we build a neural network model that predicts percent of unsheltered homeless based on shelter, housing, and other features?

First make our target categorical by bucketing into thirds.

```
In [2]:  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2199 entries, 0 to 2198
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Year                        2199 non-null   int64
 1   TotalHomeless               2199 non-null   int64
 2   Population                  2199 non-null   int64
 3   Unemployment                2199 non-null   float64
 4   Unsheltered_perc_tot        2199 non-null   float64
 5   TotalBeds_perc_tot          2199 non-null   float64
 6   Emergency_perc_tot          2199 non-null   float64
 7   Transitional_perc_tot       2199 non-null   float64
 8   SafeHaven_perc_tot          2199 non-null   float64
 9   PermanentSupportive_perc_tot  2199 non-null   float64
 10  PermanentOther_perc_tot     2199 non-null   float64
 11  RapidRehousing_perc_tot     2199 non-null   float64
dtypes: float64(9), int64(3)
memory usage: 206.3 KB

In [3]:  pd.cut(df['Unsheltered_perc_tot'], bins=3).value_counts()

Out[3]: (-0.0961, 32.033]    1500
        (32.033, 64.065]      484
        (64.065, 96.098]      215
        Name: Unsheltered_perc_tot, dtype: int64

In [4]:  size_bins=[-1, 33, 66, 100]
         group_names = ["0", "1", "2"]

In [5]:  df['Unsheltered_thirds'] = pd.cut(df['Unsheltered_perc_tot'], size_bins, labels=group_names)

In [6]:  df['Unsheltered_thirds'] = df['Unsheltered_thirds'].astype(int)
```

Next drop unneeded columns, split the df into training and testing sets, and scale the data.

```
df=df.drop(['Unsheltered_perc_tot', 'Year'], axis=1)

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2199 entries, 0 to 2198
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   TotalHomeless               2199 non-null   int64
 1   Population                  2199 non-null   int64
 2   Unemployment                2199 non-null   float64
 3   TotalBeds_perc_tot          2199 non-null   float64
 4   Emergency_perc_tot          2199 non-null   float64
 5   Transitional_perc_tot       2199 non-null   float64
 6   SafeHaven_perc_tot          2199 non-null   float64
 7   PermanentSupportive_perc_tot  2199 non-null   float64
 8   PermanentOther_perc_tot     2199 non-null   float64
 9   RapidRehousing_perc_tot     2199 non-null   float64
 10  Unsheltered_thirds          2199 non-null   int32
dtypes: float64(8), int32(1), int64(2)
memory usage: 180.5 KB

# Remove outcome target from features data
y = df.Unsheltered_thirds.values
X = df.drop(columns="Unsheltered_thirds").values

# Split training/test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, stratify=y)

# Preprocess numerical data for neural network

# Create a StandardScaler instances
scaler = StandardScaler()

# Fit the StandardScaler
X_scaler = scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

Initial Modeling Attempt Results

Initial attempts to break the data into quintiles led to a low accuracy neural network model:

In [25]:
```
# Define the deep learning model
nn_model = tf.keras.models.Sequential()
nn_model.add(tf.keras.layers.Dense(units=64, activation="sigmoid", input_dim=11))
nn_model.add(tf.keras.layers.Dense(units=64, activation="relu"))
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Compile the Sequential model together and customize metrics
nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])

# Train the model
fit_model = nn_model.fit(X_train_scaled, y_train, epochs=100)

# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
52/52 [==============================] - 0s 1ms/step - loss: 0.5789 - accuracy: 0.0904
Epoch 93/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5799 - accuracy: 0.0885
Epoch 94/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5789 - accuracy: 0.0904
Epoch 95/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5788 - accuracy: 0.0891
Epoch 96/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5798 - accuracy: 0.0916
Epoch 97/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5779 - accuracy: 0.0898
Epoch 98/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5780 - accuracy: 0.0891
Epoch 99/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5786 - accuracy: 0.0898
Epoch 100/100
52/52 [==============================] - 0s 1ms/step - loss: 0.5787 - accuracy: 0.0891
18/18 - 0s - loss: 0.5815 - accuracy: 0.0945 - 108ms/epoch - 6ms/step
Loss: 0.5814591646194458, Accuracy: 0.09454545378684998
```

Neural Network Output

Run the model.

```python
# Define the deep learning model
nn_model = tf.keras.models.Sequential()
nn_model.add(tf.keras.layers.Dense(units=64, activation="sigmoid", input_dim=10))
nn_model.add(tf.keras.layers.Dense(units=32, activation="relu"))
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Compile the Sequential model together and customize metrics
nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])

# Train the model
fit_model = nn_model.fit(X_train_scaled, y_train, epochs=100)

# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
52/52 [==============================] - 0s 1ms/step - loss: -107.8725 - accuracy: 0.7635
Epoch 93/100
52/52 [==============================] - 0s 1ms/step - loss: -111.5524 - accuracy: 0.7586
Epoch 94/100
52/52 [==============================] - 0s 1ms/step - loss: -115.2923 - accuracy: 0.7550
Epoch 95/100
52/52 [==============================] - 0s 1ms/step - loss: -119.0671 - accuracy: 0.7471
Epoch 96/100
52/52 [==============================] - 0s 1ms/step - loss: -122.8830 - accuracy: 0.7562
Epoch 97/100
52/52 [==============================] - 0s 1ms/step - loss: -126.7909 - accuracy: 0.7562
Epoch 98/100
52/52 [==============================] - 0s 2ms/step - loss: -130.6378 - accuracy: 0.7635
Epoch 99/100
52/52 [==============================] - 0s 1ms/step - loss: -134.9138 - accuracy: 0.7580
Epoch 100/100
52/52 [==============================] - 0s 1ms/step - loss: -138.9732 - accuracy: 0.7532
18/18 - 0s - loss: -1.6396e+02 - accuracy: 0.7745 - 117ms/epoch - 7ms/step
Loss: -163.95704650878906, Accuracy: 0.774545431137085
```

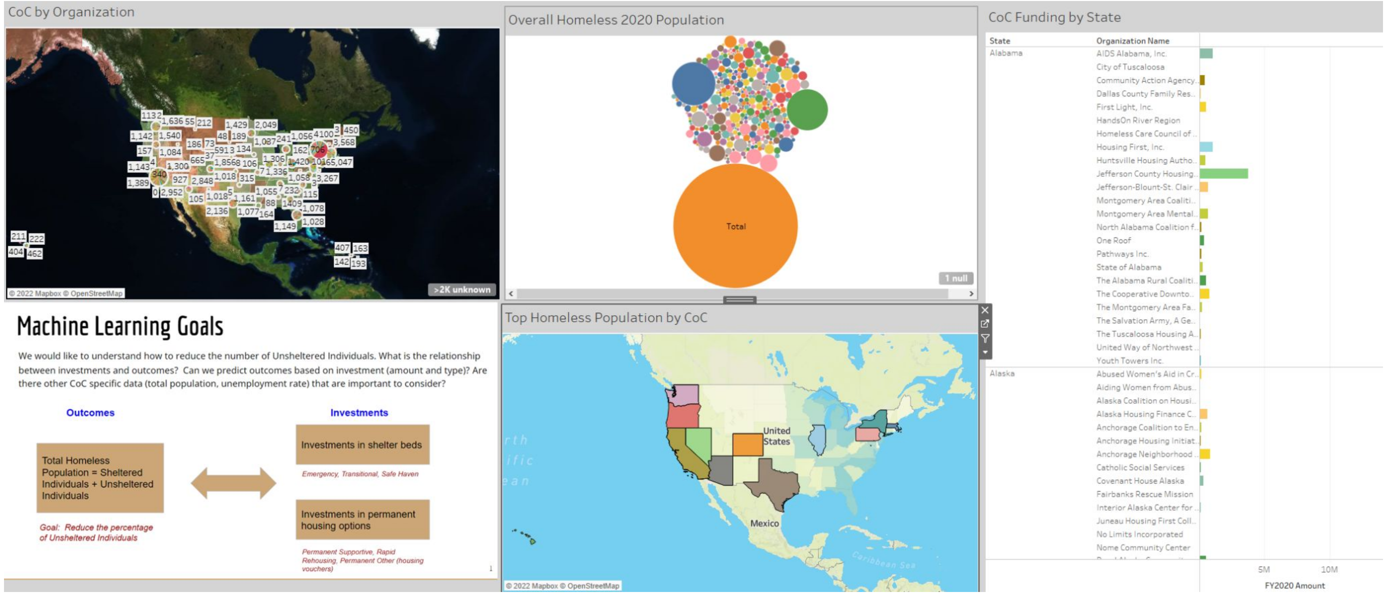
Machine Learning: Lessons Learned

- Things that didn't work so well
    - **Too Granular:** Splitting the data into deciles or quartiles
    - **Too Complicated of a Model:** Changing the number of Neurons per hidden layer
    - **Too Specific:** Using only individual years
    - **Also Too Specific:** Looking only at certain CoC types
    - **Including Messy Data:** Inclusion/exclusion of variables (e.g., unemployment, homelessness)

Results

Tableau Storyboard

Since the goals of the machine learning project were to view the correlation between investments and outcomes, a dashboard with bar graphs, packed bubbles and maps was created to illustrate the funding received per CoC, largest overall homeless populations and top homeless population per CoC.



Machine Learning: Key Takeaways

- The PIT data is very messy which is to be expected since each CoC conducts their count on a single night and utilize slightly different methodologies.
- The HIC data is confounded by the fact that we are collecting data over several years and some shelter/housing investment types (eg Safe Haven) were not as prevalent in the earlier years. Additionally, housing investments may not demonstrate impact on PIT numbers in the year of investment (PIT outcomes may be a trailing indicator of housing investment impact).
- We are able to achieve a usable prospective neural network model only when we are categorizing outcomes (unsheltered as a percent of the CoC total homeless population) into large buckets. Striating the outcome data into buckets greater than thirds results in a lower quality prediction.
- Initial inspection of the data shows that the high level trends (more beds and/or more housing units are correlated with lower percentage of unsheltered) are picked up by our models (Random Forest, Regression, Neural Network).
- ***Our predictive Neural Network model would be useful for CoCs who are in the worst 2 buckets (>33% unsheltered) and who are looking to model shelter/housing investments to see how they can move to an improved bucket.***

Let's test it out!

Making some predictions - part 1

The CA-502 Continuum of Care (Oakland, Berkeley/Alameda County) in 2019 recorded that 79% of the homeless population was unsheltered, placing them solidly in our worst performing bucket (bucket 2). We will create some hypothetical investment scenarios and predict which will move them into bucket 1 (better) or bucket 0 (best).

TotalHomeless	Population	Unemployment	TotalBeds_perc_tot	Emergency_perc_tot	Transitional_perc_tot	SafeHaven_perc_tot	PermanentSupportive_perc_tot	PermanentOther_perc_tot	RapidRehousing_perc_tot
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	32.44203441	25	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	57.44203441	50	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	82.44203441	75	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	107.4420344	100	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	157.4420344	150	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	207.4420344	200	7.043131389	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	39.85913737	14.46023436	25	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	64.85913737	14.46023436	50	0.398903017	41.79755672	0	7.554225879
8022	1671329	3	46.50336574	14.46023436	7.043131389	25	41.79755672	0	7.554225879
8022	1671329	3	71.50336574	14.46023436	7.043131389	50	41.79755672	0	7.554225879
8022	1671329	3	300	100	100	100	41.79755672	0	7.554225879
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	100	0	7.554225879
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	100	7.554225879
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	0	100

Making some predictions - part 2

The code to predict outcomes from our hypothetical data.

```
# Import our input dataset
new_data = pd.read_excel('CA502_2019_Simulation_vf.xlsx')
new_data.head()
```

```
# convert the dataframe to a numpy array
X_new_data = new_data.to_numpy()
```

```
# Don't forget to scale the data with the existing scaler!
X_new_data_scaled = scaler.transform(X_new_data)
```

```
# Use predict to apply the model to the new data
y_new_data = nn_model.predict(X_new_data_scaled)
```

```
# Add the result to the data frame
new_data['Prediction'] = y_new_data
```

Making some predictions - our results!

As you can see, the predictions put our outcomes in either bucket 1 or bucket 0 with the more extreme adjustments ending up in bucket 0, as we would hope. Unfortunately, the model mispredicted the actual data (first line) and placed it in bucket 1 instead of bucket 2. This is not surprising, given the accuracy of our model, but gives us hope that further tightening of the model (through adding more features or further refining existing features) will increase the utility.

TotalHomeless	Population	Unemployment	TotalBeds_perc_tot	Emergency_perc_tot	Transitional_perc_tot	SafeHaven_perc_tot	PermanentSupportive_perc_tot	PermanentOther_perc_tot	RapidRehousing_perc_tot	Prediction
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	0	7.554225879	1
8022	1671329	3	32.44203441	25	7.043131389	0.398903017	41.79755672	0	7.554225879	1
8022	1671329	3	57.44203441	50	7.043131389	0.398903017	41.79755672	0	7.554225879	1
8022	1671329	3	82.44203441	75	7.043131389	0.398903017	41.79755672	0	7.554225879	1.13396E-26
8022	1671329	3	107.4420344	100	7.043131389	0.398903017	41.79755672	0	7.554225879	1.49272E-18
8022	1671329	3	157.4420344	150	7.043131389	0.398903017	41.79755672	0	7.554225879	0.034948051
8022	1671329	3	207.4420344	200	7.043131389	0.398903017	41.79755672	0	7.554225879	0.05510512
8022	1671329	3	39.85913737	14.46023436	25	0.398903017	41.79755672	0	7.554225879	1
8022	1671329	3	64.85913737	14.46023436	50	0.398903017	41.79755672	0	7.554225879	1
8022	1671329	3	46.50336574	14.46023436	7.043131389	25	41.79755672	0	7.554225879	1
8022	1671329	3	71.50336574	14.46023436	7.043131389	50	41.79755672	0	7.554225879	1
8022	1671329	3	300	100	100	100	41.79755672	0	7.554225879	3.78765E-22
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	100	0	7.554225879	1
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	100	7.554225879	1
8022	1671329	3	21.90226876	14.46023436	7.043131389	0.398903017	41.79755672	0	100	1

Project Conclusions

- It is possible to develop machine learning models that can effectively predict the necessary investments that will improve the ratio of unsheltered homeless individuals vs. the total homeless population in a specific CoC when compared to other CoCs in the United States
- This model could be a powerful tool to assist CoCs with high unsheltered rates to improve the success rate of putting a roof over the heads of more homeless individuals.

Additional Data/Further Analyses to Improve Model

Additional Data:

- Incorporating HUD funding data
- Stratifying CoCs by climate/region
- Stratifying CoCs by Population Density
- Incorporating secondary/higher education data per CoC
- Evaluating mental health resources and funding per CoC
- All results are based on federal funding; what state resources were utilized?

Further Analytic Tools:

- Refine data cleaning to remove anomalous messy data
- Re-investigate random forest and other ML methods after enhanced data cleaning

Website, Github repo, and Tableau Links

- Website
- Github
- Tableau Public

Project Resources

Tableau Public

Python

Pandas

Flask

SQLite

R Code

GitHub

Questions?

Thank you!