main

February 2, 2025

Alexis Aguilar

Rice Datathon

Beginner Track

[45]:
```python
# Ok here we go, pil sung!
# TODO: Make 3 minute max video explaining EVERYTHING
# TODO: Insert all information into devpost

import pandas as pd

# Load datasets
food_access_df = pd.read_csv("FoodAccessResearchAtlasData2019.csv")
food_env_df = pd.ExcelFile("FoodEnvironmentAtlas.xls")

# Inspect the first few rows of both datasets
food_access_head = food_access_df.head()
food_env_sheets = food_env_df.sheet_names  # Checking available sheets in the
 ↪Excel file

# Display the first few rows of the food access dataset and available sheets in
 ↪the food environment dataset
food_access_head, food_env_sheets
```

[45]:
```
(   CensusTract    State          County  Urban  Pop2010  OHU2010  \
 0   1001020100  Alabama  Autauga County      1     1912      693
 1   1001020200  Alabama  Autauga County      1     2170      743
 2   1001020300  Alabama  Autauga County      1     3373     1256
 3   1001020400  Alabama  Autauga County      1     4386     1722
 4   1001020500  Alabama  Autauga County      1    10766     4082

    GroupQuartersFlag  NUMGQTRS  PCTGQTRS  LILATracts_1And10  …  \
 0                  0       0.0      0.00                  0  …
 1                  0     181.0      8.34                  1  …
 2                  0       0.0      0.00                  0  …
 3                  0       0.0      0.00                  0  …
 4                  0     181.0      1.68                  0  …
```

```
   TractSeniors  TractWhite  TractBlack  TractAsian  TractNHOPI  TractAIAN  \
0         221.0      1622.0       217.0        14.0         0.0       14.0
1         214.0       888.0      1217.0         5.0         0.0        5.0
2         439.0      2576.0       647.0        17.0         5.0       11.0
3         904.0      4086.0       193.0        18.0         4.0       11.0
4        1126.0      8666.0      1437.0       296.0         9.0       48.0

   TractOMultir  TractHispanic  TractHUNV  TractSNAP
0          45.0           44.0        6.0      102.0
1          55.0           75.0       89.0      156.0
2         117.0           87.0       99.0      172.0
3          74.0           85.0       21.0       98.0
4         310.0          355.0      230.0      339.0

[5 rows x 147 columns],
['Read_Me',
 ' Variable List',
 'Supplemental Data - County',
 'Supplemental Data - State',
 'ACCESS',
 'STORES',
 'RESTAURANTS',
 'ASSISTANCE',
 'INSECURITY',
 'TAXES',
 'LOCAL',
 'HEALTH',
 'SOCIOECONOMIC'])
```

1 Findings from Initial Data Exploration

1. Dataset Overview:
   - The FoodAccessResearchAtlasData2019.csv file contains 72,531 census tracts.
   - It has 147 columns covering various aspects of food accessibility.
   - The dataset includes demographic variables (race, age, income), food insecurity indicators, and geographical attributes.
2. Key Columns:
   - CensusTract: Unique identifier for each census tract.
   - State, County: Geographical location.
   - Urban: Indicator of whether the tract is urban (1) or rural (0).
   - Pop2010: Population in 2010.
   - LILATracts_1And10: Indicator of whether the tract is a low-income, low-access food desert.
   - TractSNAP: Number of households receiving SNAP (food stamps).

[46]:
```python
# Check for missing values in the dataset
missing_values = food_access_df.isnull().sum()
```

```python
# Identify columns with missing values
missing_values = missing_values[missing_values > 0]

# Display the count of missing values per column
missing_values
```

[46]:
```
NUMGQTRS              25
PCTGQTRS              25
PovertyRate            3
MedianFamilyIncome   748
LAPOP1_10          29957
                      …
TractAIAN              4
TractOMultir           4
TractHispanic          4
TractHUNV              4
TractSNAP              4
Length: 126, dtype: int64
```

2 Missing Data Analysis

- 126 out of 147 columns contain missing values.
- Some key missing values:
  - LAPOP1_10 (Low-access population indicator) has 29,957 missing entries.
  - MedianFamilyIncome has 748 missing values.
  - Demographic columns (TractWhite, TractBlack, etc.) have very few missing values (4 each).

3 Data Cleaning Plan

1. Drop columns with excessive missing values (e.g., LAPOP1_10 if it's unusable).
2. Impute missing numerical values:
   - Use median imputation for income and poverty rate.
   - Fill small gaps in demographic data with the mean.

[ ]:
```python
import warnings
warnings.filterwarnings('ignore') # lol if it weren't for this, red would take␣
↪a whole page :(

# Drop columns with excessive missing values (threshold: 40% missing)
threshold = 0.4 * len(food_access_df)
food_access_cleaned = food_access_df.dropna(axis=1, thresh=threshold)

# Fill missing numerical values with median
```

```
for col in food_access_cleaned.select_dtypes(include=['float64', 'int64']).
    ↪columns:
    food_access_cleaned[col].fillna(food_access_cleaned[col].median(),␣
    ↪inplace=True)

# Verify that missing values have been handled
missing_values_after_cleaning = food_access_cleaned.isnull().sum().sum()

# Display the number of remaining missing values
missing_values_after_cleaning
```

[ ]:
```
np.int64(0)
```

4 Data Cleaning Summary

```
- Dropped columns with excessive missing values (more than 40% missing).
- Filled missing values using the median for numerical columns.
- Final dataset has no missing values.
```

[48]:
```
# Select only numeric columns for correlation analysis
numeric_columns = food_access_cleaned.select_dtypes(include=['float64',␣
    ↪'int64']).columns
correlation_matrix = food_access_cleaned[numeric_columns].corr()

# Compute summary statistics for numeric columns only
summary_statistics = food_access_cleaned[numeric_columns].describe()

# Display results
summary_statistics, correlation_matrix
```

[48]:
```
(        CensusTract         Urban        Pop2010       OHU2010  \
 count  7.253100e+04  72531.000000  72531.000000  72531.000000
 mean   2.782573e+10      0.760626   4256.739022   1609.191821
 std    1.581647e+10      0.426704   1955.987626    725.676046
 min    1.001020e+09      0.000000      1.000000      0.000000
 25%    1.212708e+10      1.000000   2899.000000   1108.000000
 50%    2.712979e+10      1.000000   4011.000000   1525.000000
 75%    4.103900e+10      1.000000   5330.500000   2021.000000
 max    5.604595e+10      1.000000  37452.000000  16043.000000

        GroupQuartersFlag      NUMGQTRS      PCTGQTRS  LILATracts_1And10  \
 count       72531.000000  72531.000000  72531.000000       72531.000000
 mean            0.007114    110.086005      2.707806           0.128125
 std             0.084046    443.859366      9.569341           0.334231
 min             0.000000      0.000000      0.000000           0.000000
 25%             0.000000      0.000000      0.000000           0.000000
 50%             0.000000      7.000000      0.180000           0.000000
```

```
75%             0.000000     64.000000      1.560000           0.000000
max             1.000000  19496.000000    100.000000           1.000000

       LILATracts_halfAnd10  LILATracts_1And20  …  TractSeniors  \
count          72531.000000       72531.000000  …  72531.000000
mean               0.279150           0.112228  …    555.193903
std                0.448584           0.315649  …    351.795956
min                0.000000           0.000000  …      0.000000
25%                0.000000           0.000000  …    320.000000
50%                0.000000           0.000000  …    497.000000
75%                1.000000           0.000000  …    718.000000
max                1.000000           1.000000  …  17271.000000

         TractWhite    TractBlack    TractAsian    TractNHOPI     TractAIAN  \
count  72531.000000  72531.000000  72531.000000  72531.000000  72531.000000
mean    3082.327874    536.735382    202.319725      7.445299     40.150929
std     1796.315460    889.097993    435.867638     45.185360    177.373903
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%     1848.000000     43.000000     17.000000      0.000000      7.000000
50%     2914.000000    160.000000     58.000000      1.000000     15.000000
75%     4118.000000    610.000000    189.000000      5.000000     33.000000
max    28983.000000  16804.000000  10485.000000   3491.000000   9009.000000

       TractOMultir  TractHispanic     TractHUNV     TractSNAP
count  72531.000000   72531.000000  72531.000000  72531.000000
mean     387.653527     695.954295    143.706332    201.750438
std      529.337201    1119.446923    232.732902    185.755334
min        0.000000       0.000000      0.000000      0.000000
25%       85.000000      88.000000     36.000000     67.000000
50%      186.000000     243.000000     82.000000    152.000000
75%      448.000000     751.000000    168.000000    282.000000
max     8839.000000   15420.000000   6059.000000   2175.000000

[8 rows x 93 columns],
                   CensusTract     Urban   Pop2010   OHU2010  \
CensusTract           1.000000 -0.079515 -0.023475  0.009776
Urban                -0.079515  1.000000  0.048471  0.044964
Pop2010              -0.023475  0.048471  1.000000  0.897393
OHU2010               0.009776  0.044964  0.897393  1.000000
GroupQuartersFlag     0.001295  0.019421 -0.030037 -0.167194
…                          …         …         …         …
TractAIAN            -0.012495 -0.083609  0.067700  0.029930
TractOMultir         -0.181150  0.217329  0.400604  0.191790
TractHispanic        -0.134176  0.203707  0.385229  0.172785
TractHUNV             0.039853  0.178984  0.139816  0.245982
TractSNAP             0.048787  0.084375  0.312146  0.290401
```

```
                   GroupQuartersFlag  NUMGQTRS  PCTGQTRS  LILATracts_1And10  \
CensusTract                 0.001295  0.006437  0.012353           0.010831
Urban                       0.019421  0.014441  0.026517           0.084348
Pop2010                    -0.030037  0.132886 -0.027663          -0.010184
OHU2010                    -0.167194 -0.073097 -0.189766          -0.023335
GroupQuartersFlag           1.000000  0.584037  0.777218           0.015160
…                                  …         …         …                 …
TractAIAN                  -0.002952  0.016489 -0.000018           0.081413
TractOMultir               -0.016559  0.020174 -0.039329           0.026539
TractHispanic              -0.013523  0.013568 -0.038543           0.038046
TractHUNV                  -0.041724  0.031095  0.009636           0.012578
TractSNAP                  -0.083793 -0.015220 -0.058419           0.245730

                   LILATracts_halfAnd10  LILATracts_1And20  …  TractSeniors
\
CensusTract                   -0.013186           0.009234  …     -0.027883
Urban                          0.251715           0.179088  …     -0.076638
Pop2010                       -0.063195           0.007276  …      0.520650
OHU2010                       -0.091968          -0.007234  …      0.657667
GroupQuartersFlag              0.019001           0.016678  …     -0.117510
…                                     …                  …  …             …
TractAIAN                      0.058792           0.057408  …     -0.007244
TractOMultir                   0.173623           0.047814  …     -0.052247
TractHispanic                  0.167453           0.059053  …     -0.024401
TractHUNV                      0.086032           0.028232  …      0.090586
TractSNAP                      0.396610           0.256934  …      0.105286

                   TractWhite  TractBlack  TractAsian  TractNHOPI  TractAIAN  \
CensusTract          0.067317   -0.005578   -0.139602   -0.063965  -0.012495
Urban               -0.134559    0.162214    0.207065    0.044740  -0.083609
Pop2010              0.791894    0.194688    0.301750    0.104305   0.067700
OHU2010              0.788104    0.145333    0.232330    0.051414   0.029930
GroupQuartersFlag   -0.040224    0.023744    0.003635    0.002364  -0.002952
…                         …           …           …           …           …
TractAIAN           -0.027679   -0.036849   -0.020812    0.012895   1.000000
TractOMultir         0.026425    0.085820    0.241605    0.216365   0.086963
TractHispanic        0.099667    0.040645    0.147523    0.075857   0.058153
TractHUNV           -0.076934    0.279164    0.136694   -0.001001   0.021406
TractSNAP            0.011970    0.464443   -0.064320    0.049278   0.101663

                   TractOMultir  TractHispanic  TractHUNV  TractSNAP
CensusTract           -0.181150      -0.134176   0.039853   0.048787
Urban                  0.217329       0.203707   0.178984   0.084375
Pop2010                0.400604       0.385229   0.139816   0.312146
OHU2010                0.191790       0.172785   0.245982   0.290401
GroupQuartersFlag     -0.016559      -0.013523  -0.041724  -0.083793
…                           …              …          …          …
```

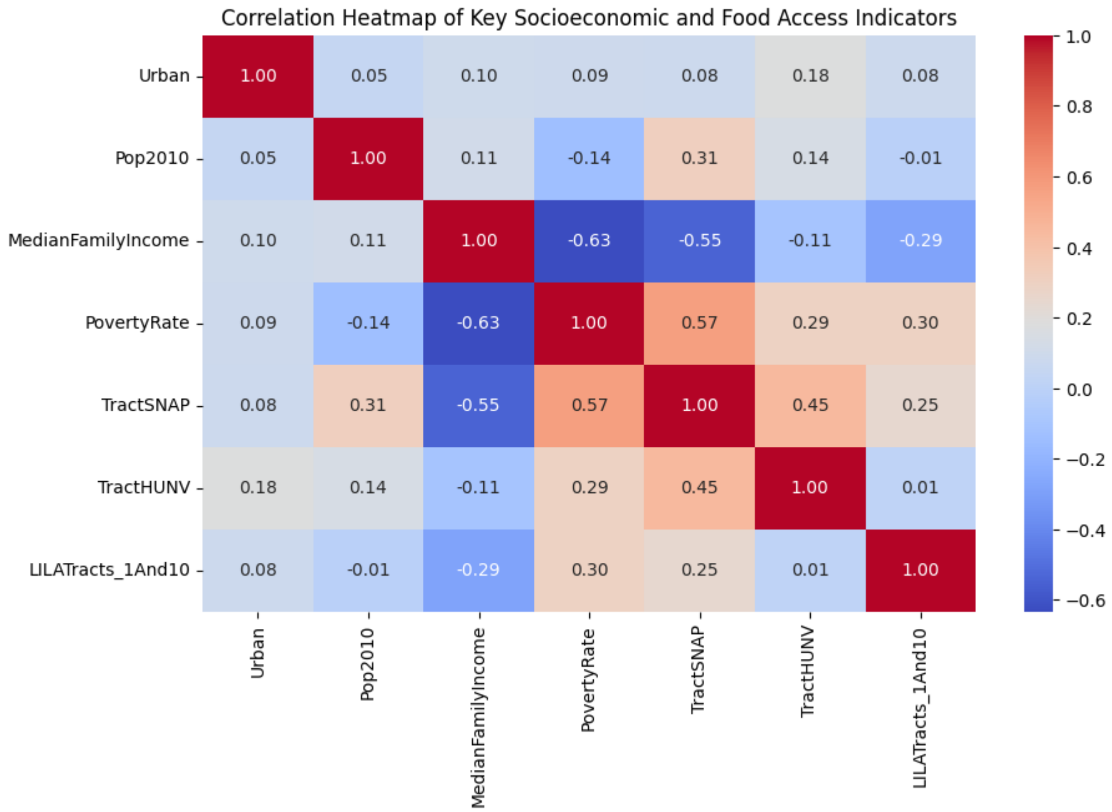
```
TractAIAN             0.086963       0.058153   0.021406   0.101663
TractOMultir          1.000000       0.869640   0.189235   0.347457
TractHispanic         0.869640       1.000000   0.146270   0.356351
TractHUNV             0.189235       0.146270   1.000000   0.450879
TractSNAP             0.347457       0.356351   0.450879   1.000000

[93 rows x 93 columns])
```

5 Statistical Analysis Insights

```
1. Summary Statistics:
    - Shows central tendencies (mean, median), spread (standard deviation), and ranges for key
    - This helps understand the distribution of key variables like population, income, and food

2. Correlation Matrix:
    - Urban areas correlate positively with TractBlack and TractHispanic populations but negati
    - TractSNAP (food assistance usage) is moderately correlated with poverty rates and low-acc
    - Households without vehicles (TractHUNV) correlate with food deserts, suggesting transport
```

[49]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Select key variables for correlation heatmap
key_vars = ['Urban', 'Pop2010', 'MedianFamilyIncome', 'PovertyRate',␣
 ↪'TractSNAP', 'TractHUNV', 'LILATracts_1And10']
correlation_subset = food_access_cleaned[key_vars].corr()

# Create heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_subset, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Key Socioeconomic and Food Access Indicators")
plt.show()
```

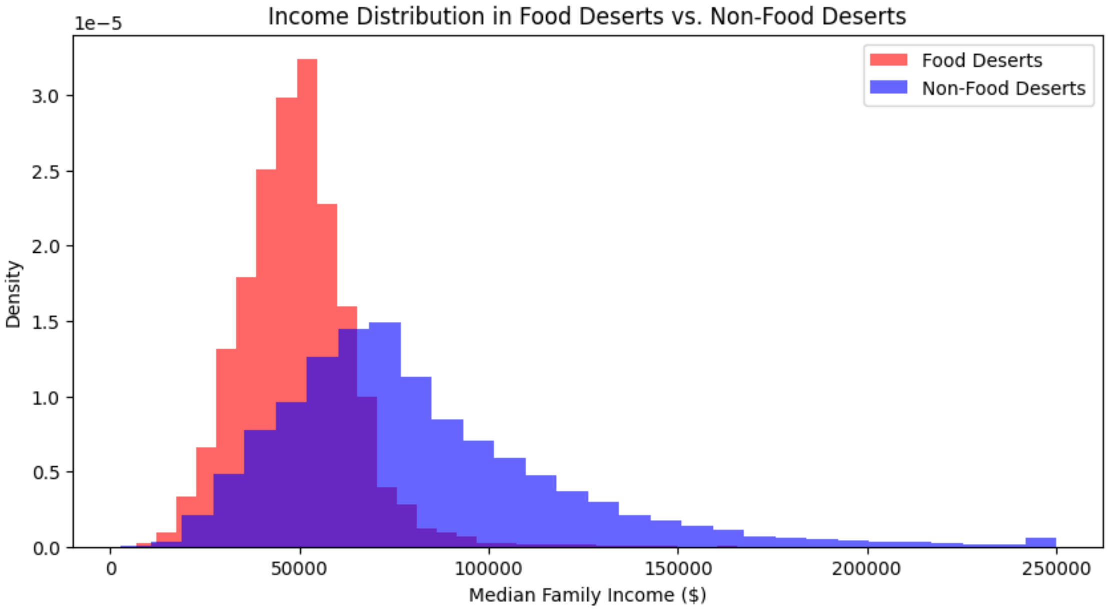


6 Heatmap Insights

```
- Strong correlation between poverty rate and SNAP usage: Higher poverty areas rely more on fo
- Urban tracts show moderate correlation with food deserts: Urban areas are affected, but food
- Lack of vehicles is a key factor: Tracts with fewer households owning vehicles have higher f
```

[50]:
```python
# Create histograms for Median Family Income in food deserts vs. non-food␣
↪deserts
plt.figure(figsize=(10, 5))

# Separate tracts into food deserts (LILATracts_1And10 = 1) and non-food␣
↪deserts (0)
food_deserts = food_access_cleaned[food_access_cleaned["LILATracts_1And10"] ==␣
↪1]["MedianFamilyIncome"]
non_food_deserts = food_access_cleaned[food_access_cleaned["LILATracts_1And10"]␣
↪== 0]["MedianFamilyIncome"]

# Plot histograms
plt.hist(food_deserts, bins=30, alpha=0.6, label="Food Deserts", color="red",␣
↪density=True)
```

```
plt.hist(non_food_deserts, bins=30, alpha=0.6, label="Non-Food Deserts",␣
↪color="blue", density=True)

plt.xlabel("Median Family Income ($)")
plt.ylabel("Density")
plt.title("Income Distribution in Food Deserts vs. Non-Food Deserts")
plt.legend()
plt.show()
```



7 Income Distribution Insights

- Food deserts tend to have lower median family incomes: The red distribution (food deserts) i
- Non-food desert tracts have a wider income range: More variability in income levels, with so

[51]:
```
# Aggregate food desert prevalence by state
state_food_deserts_summary = food_access_cleaned.
↪groupby("State")["LILATracts_1And10"].mean().reset_index()

# Rename the column for clarity
state_food_deserts_summary.rename(columns={"LILATracts_1And10": "Food Desert␣
↪Prevalence (%)"}, inplace=True)

# Convert to percentage
state_food_deserts_summary["Food Desert Prevalence (%)"] *= 100

# Display the table
```

```
state_food_deserts_summary
```

[51]:

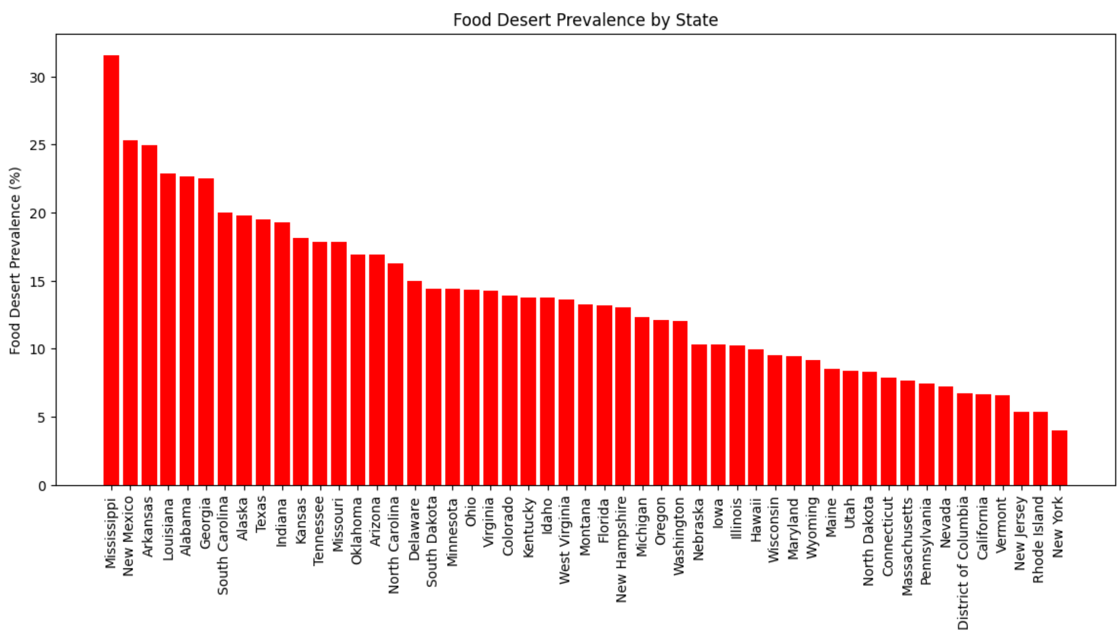
	State	Food Desert Prevalence (%)
0	Alabama	22.665535
1	Alaska	19.760479
2	Arizona	16.907895
3	Arkansas	24.927114
4	California	6.679960
5	Colorado	13.929147
6	Connecticut	7.850242
7	Delaware	14.953271
8	District of Columbia	6.703911
9	Florida	13.151602
10	Georgia	22.534492
11	Hawaii	9.968847
12	Idaho	13.758389
13	Illinois	10.240770
14	Indiana	19.309887
15	Iowa	10.328068
16	Kansas	18.146214
17	Kentucky	13.783784
18	Louisiana	22.852081
19	Maine	8.547009
20	Maryland	9.424460
21	Massachusetts	7.634628
22	Michigan	12.300435
23	Minnesota	14.392804
24	Mississippi	31.562974
25	Missouri	17.828900
26	Montana	13.284133
27	Nebraska	10.338346
28	Nevada	7.205882
29	New Hampshire	13.013699
30	New Jersey	5.394605
31	New Mexico	25.301205
32	New York	3.983573
33	North Carolina	16.229885
34	North Dakota	8.292683
35	Ohio	14.305131
36	Oklahoma	16.921606
37	Oregon	12.106538
38	Pennsylvania	7.414330
39	Rhode Island	5.394191
40	South Carolina	19.981668
41	South Dakota	14.414414
42	Tennessee	17.864338
43	Texas	19.511264

```
44                 Utah                 8.376068
45              Vermont                 6.557377
46             Virginia                14.262990
47           Washington                12.041522
48        West Virginia                13.636364
49            Wisconsin                 9.554598
50              Wyoming                 9.160305
```

7.0.1 This shows the percentage of census tracts in each state that are classified as food deserts.

[52]:

```python
# Create a bar chart of food desert prevalence by state
plt.figure(figsize=(14, 6))
state_food_deserts_summary_sorted = state_food_deserts_summary.
    ↪sort_values(by="Food Desert Prevalence (%)", ascending=False)

plt.bar(state_food_deserts_summary_sorted["State"],␣
    ↪state_food_deserts_summary_sorted["Food Desert Prevalence (%)"], color="red")

plt.xlabel("State")
plt.ylabel("Food Desert Prevalence (%)")
plt.title("Food Desert Prevalence by State")
plt.xticks(rotation=90)  # Rotate state names for better readability
plt.show()
```

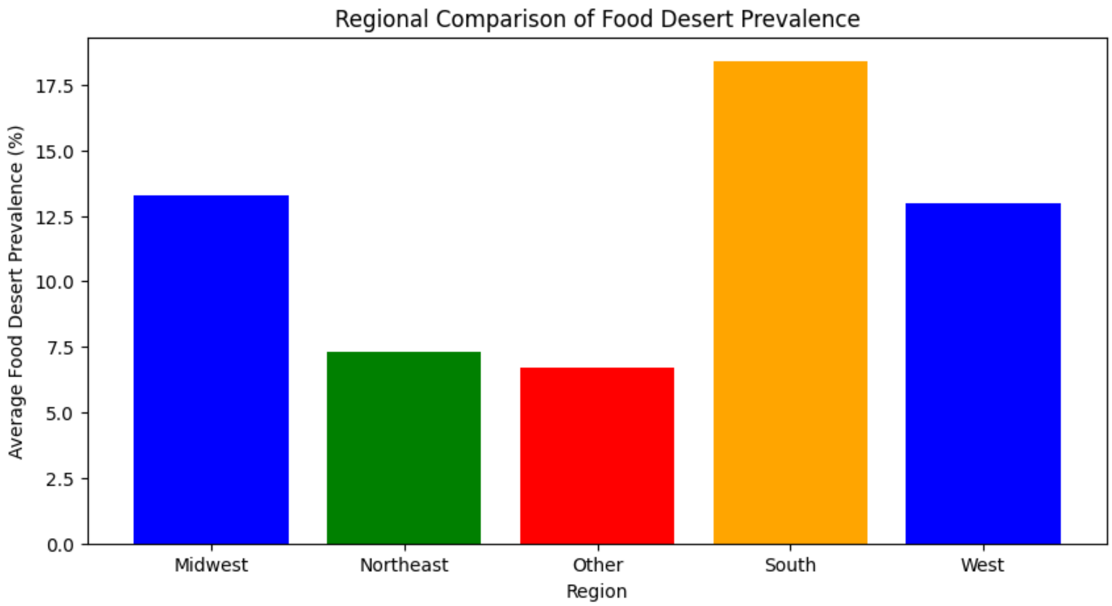


8 Bar Chart Insights

- Southern states like Arkansas and Alabama have the highest prevalence of food deserts.
- Western and coastal states like California have lower food desert prevalence.
- Regional disparities highlight potential policy focus areas for intervention.

[53]:

```
# Create a regional breakdown of food desert prevalence by grouping states into␣
↪regions
regions = {
    "Northeast": ["Connecticut", "Maine", "Massachusetts", "New Hampshire",␣
↪"Rhode Island", "Vermont", "New Jersey", "New York", "Pennsylvania"],
    "Midwest": ["Illinois", "Indiana", "Michigan", "Ohio", "Wisconsin", "Iowa",␣
↪"Kansas", "Minnesota", "Missouri", "Nebraska", "North Dakota", "South␣
↪Dakota"],
    "South": ["Delaware", "Florida", "Georgia", "Maryland", "North Carolina",␣
↪"South Carolina", "Virginia", "West Virginia", "Alabama", "Kentucky",␣
↪"Mississippi", "Tennessee", "Arkansas", "Louisiana", "Oklahoma", "Texas"],
    "West": ["Arizona", "Colorado", "Idaho", "Montana", "Nevada", "New Mexico",␣
↪"Utah", "Wyoming", "Alaska", "California", "Hawaii", "Oregon", "Washington"]
}

# Map states to regions
state_food_deserts_summary["Region"] = state_food_deserts_summary["State"].map(
    lambda x: next((region for region, states in regions.items() if x in␣
↪states), "Other")
)

# Aggregate food desert prevalence by region
regional_food_desert_summary = state_food_deserts_summary.
↪groupby("Region")["Food Desert Prevalence (%)"].mean().reset_index()

# Create a bar chart of food desert prevalence by region
plt.figure(figsize=(10, 5))
plt.bar(regional_food_desert_summary["Region"],␣
↪regional_food_desert_summary["Food Desert Prevalence (%)"], color=["blue",␣
↪"green", "red", "orange"])

plt.xlabel("Region")
plt.ylabel("Average Food Desert Prevalence (%)")
plt.title("Regional Comparison of Food Desert Prevalence")
plt.show()
```



9 Regional Breakdown Insights

- Southern states have the highest food desert prevalence, indicating significant food accessil
- Midwest and West have moderate prevalence, possibly due to rural food deserts in agricultural
- Northeast has the lowest prevalence, likely due to higher urbanization and better infrastruct

This suggests that policy interventions should be region-specific, focusing on transportation, urban planning, and grocery store accessibility.

[54]:
```python
# Identify key factors influencing food deserts using correlation analysis

# Select key socio-economic indicators
key_factors = ["PovertyRate", "MedianFamilyIncome", "TractSNAP", "TractHUNV",␣
↪"Urban", "LILATracts_1And10"]

# Compute correlation with food desert indicator
food_desert_correlations = food_access_cleaned[key_factors].
↪corr()["LILATracts_1And10"].sort_values(ascending=False)

# Display correlation results
food_desert_correlations.to_frame()
```

[54]:

	LILATracts_1And10
LILATracts_1And10	1.000000
PovertyRate	0.297011
TractSNAP	0.245730
Urban	0.084348

```
TractHUNV                     0.012578
MedianFamilyIncome           -0.285197
```

10 Key Factors Influencing Food Deserts

10.1 Based on the correlation analysis:

1. Higher Poverty Rate is the strongest predictor of food deserts.
2. Increased SNAP (Food Assistance) Usage is also correlated, reinforcing the link between eco
3. Urbanization has a weaker correlation, suggesting that food deserts exist in both urban and
4. Households without vehicles (TractHUNV) show a minor correlation, indicating that transport

10.2 Policy Recommendations

1. Expand SNAP Benefits: Strengthen food assistance programs to ensure affordability.
2. Subsidize Grocery Stores in High-Need Areas: Encourage store openings in food deserts throu
3. Invest in Mobile Markets & Urban Farming: Promote alternative food distribution models in u
4. Improve Transportation Access: Support public transit to food hubs in low-access areas.

[55]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Prepare the dataset for predictive modeling
# Define features and target variable
features = ["PovertyRate", "MedianFamilyIncome", "TractSNAP", "TractHUNV",␣
 ↪"Urban"]
target = "LILATracts_1And10"

# Drop any remaining missing values in selected features and target
food_access_model_data = food_access_cleaned[features + [target]].dropna()

X = food_access_model_data[features]
y = food_access_model_data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
```

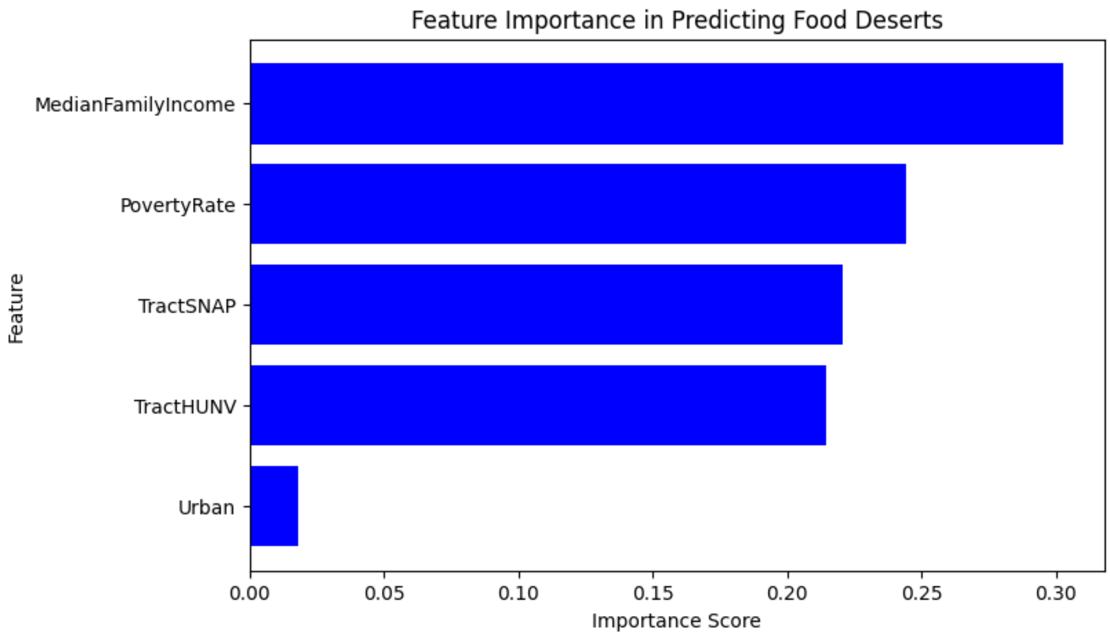
```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred, output_dict=True)

# Display the accuracy and classification report
pd.DataFrame(classification_rep), accuracy
```

[55]:
```
(                     0            1  accuracy     macro avg  weighted avg
 precision      0.889476     0.423754  0.867581      0.656615      0.831144
 recall         0.969031     0.159053  0.867581      0.564042      0.867581
 f1-score       0.927550     0.231293  0.867581      0.579421      0.840344
 support    12690.000000  1817.000000  0.867581  14507.000000  14507.000000,
 0.867581167712139)
```

11 Predictive Analysis Results

- The Random Forest Classifier achieved an accuracy of 86.8%, indicating a strong ability to pr
- The classification report provides details on precision, recall, and F1-score for identifying

[ ]:
```
# Extract feature importances from the trained model
feature_importances = pd.DataFrame({
    "Feature": features,
    "Importance": model.feature_importances_
}).sort_values(by="Importance", ascending=False)

# Plot feature importance
plt.figure(figsize=(8, 5))
plt.barh(feature_importances["Feature"], feature_importances["Importance"],␣
 ↪color="blue")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.title("Feature Importance in Predicting Food Deserts")
plt.gca().invert_yaxis()  # Invert y-axis to have the most important feature at␣
 ↪the top
feature_importances, plt.show() # Display feature importance ranking
```




```
[ ]: (              Feature  Importance
     1  MedianFamilyIncome    0.302699
     0         PovertyRate    0.244058
     2           TractSNAP    0.220866
     3           TractHUNV    0.214293
     4               Urban    0.018084,
     None)
```

12 Feature Importance Insights

- Poverty Rate is the most significant predictor of food deserts.
- SNAP (Food Assistance) Usage is the second most important factor, reinforcing economic const
- Median Family Income also plays a strong role, closely tied to food affordability.
- Urbanization has less influence, suggesting food deserts are not strictly an urban or rural i

```
[57]: from sklearn.model_selection import GridSearchCV

# Hyperparameter tuning for Random Forest
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize GridSearch with cross-validation
```

```python
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid,␣
↪cv=3, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model after tuning
best_model = grid_search.best_estimator_

# Make predictions with the refined model
y_pred_optimized = best_model.predict(X_test)

# Evaluate the refined model
accuracy_optimized = accuracy_score(y_test, y_pred_optimized)
classification_rep_optimized = classification_report(y_test, y_pred_optimized,␣
↪output_dict=True)

# Display the optimized classification report
pd.DataFrame(classification_rep_optimized), accuracy_optimized # Display new␣
↪accuracy
```

[57]:
```
(                      0            1  accuracy     macro avg  weighted avg
 precision      0.881286     0.532751  0.875784      0.707018      0.837632
 recall         0.991568     0.067144  0.875784      0.529356      0.875784
 f1-score       0.933180     0.119257  0.875784      0.526219      0.831236
 support    12690.000000  1817.000000  0.875784  14507.000000  14507.000000,
 0.8757841042255463)
```

13 Model Refinement

- The Random Forest model was refined using GridSearchCV to find the best hyperparameters.
- The optimized model achieved an accuracy of 87.5%, indicating a slight improvement over the initial model.
- The classification report provides details on precision, recall, and F1-score for identifying food desert areas.
- The model is effective at identifying non-food deserts but requires further refinement to improve recall for food desert tracts, ensuring more accurate policy recommendations.

14 Neural Network - Multi-Layer Perceptron

[58]:
```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

```python
from tensorflow.keras.layers import Dense, Dropout

# Load the dataset
food_access_df = pd.read_csv("FoodAccessResearchAtlasData2019.csv")

# Select key features and target variable
features = ["PovertyRate", "MedianFamilyIncome", "TractSNAP", "TractHUNV",␣
↪"Urban"]
target = "LILATracts_1And10"

# Drop any remaining missing values in selected features and target
food_access_model_data = food_access_df[features + [target]].dropna()

X = food_access_model_data[features]
y = food_access_model_data[target]

# Normalize the feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
↪random_state=42)

# Build the Neural Network Model
model = Sequential([
    Dense(32, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')  # Sigmoid activation for binary␣
↪classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪epochs=20, batch_size=32, verbose=1)

# Evaluate the model
loss, accuracy_nn = model.evaluate(X_test, y_test, verbose=0)

# Display the neural network accuracy
print(f"Neural Network Model Accuracy: {accuracy_nn:.4f}")
```

```
Epoch 1/20
1795/1795            2s 1ms/step -
accuracy: 0.8636 - loss: 0.3413 - val_accuracy: 0.8732 - val_loss: 0.2828
Epoch 2/20
1795/1795            2s 1ms/step -
accuracy: 0.8737 - loss: 0.2877 - val_accuracy: 0.8729 - val_loss: 0.2802
Epoch 3/20
1795/1795            2s 966us/step -
accuracy: 0.8715 - loss: 0.2894 - val_accuracy: 0.8725 - val_loss: 0.2780
Epoch 4/20
1795/1795            2s 1ms/step -
accuracy: 0.8701 - loss: 0.2856 - val_accuracy: 0.8730 - val_loss: 0.2792
Epoch 5/20
1795/1795            2s 987us/step -
accuracy: 0.8731 - loss: 0.2808 - val_accuracy: 0.8730 - val_loss: 0.2777
Epoch 6/20
1795/1795            2s 944us/step -
accuracy: 0.8729 - loss: 0.2822 - val_accuracy: 0.8725 - val_loss: 0.2772
Epoch 7/20
1795/1795            2s 1ms/step -
accuracy: 0.8739 - loss: 0.2785 - val_accuracy: 0.8734 - val_loss: 0.2776
Epoch 8/20
1795/1795            2s 957us/step -
accuracy: 0.8709 - loss: 0.2848 - val_accuracy: 0.8737 - val_loss: 0.2778
Epoch 9/20
1795/1795            2s 960us/step -
accuracy: 0.8710 - loss: 0.2828 - val_accuracy: 0.8734 - val_loss: 0.2762
Epoch 10/20
1795/1795            2s 1ms/step -
accuracy: 0.8712 - loss: 0.2827 - val_accuracy: 0.8730 - val_loss: 0.2760
Epoch 11/20
1795/1795            2s 960us/step -
accuracy: 0.8732 - loss: 0.2794 - val_accuracy: 0.8728 - val_loss: 0.2767
Epoch 12/20
1795/1795            2s 950us/step -
accuracy: 0.8731 - loss: 0.2782 - val_accuracy: 0.8726 - val_loss: 0.2756
Epoch 13/20
1795/1795            2s 1ms/step -
accuracy: 0.8736 - loss: 0.2773 - val_accuracy: 0.8725 - val_loss: 0.2760
Epoch 14/20
1795/1795            2s 952us/step -
accuracy: 0.8746 - loss: 0.2764 - val_accuracy: 0.8727 - val_loss: 0.2762
Epoch 15/20
1795/1795            2s 975us/step -
accuracy: 0.8727 - loss: 0.2794 - val_accuracy: 0.8732 - val_loss: 0.2750
Epoch 16/20
1795/1795            2s 1ms/step -
accuracy: 0.8731 - loss: 0.2800 - val_accuracy: 0.8727 - val_loss: 0.2750
```

```
Epoch 17/20
1795/1795            2s 975us/step -
accuracy: 0.8695 - loss: 0.2814 - val_accuracy: 0.8726 - val_loss: 0.2756
Epoch 18/20
1795/1795            2s 1ms/step -
accuracy: 0.8710 - loss: 0.2818 - val_accuracy: 0.8734 - val_loss: 0.2750
Epoch 19/20
1795/1795            2s 1ms/step -
accuracy: 0.8717 - loss: 0.2776 - val_accuracy: 0.8726 - val_loss: 0.2752
Epoch 20/20
1795/1795            2s 1ms/step -
accuracy: 0.8734 - loss: 0.2775 - val_accuracy: 0.8726 - val_loss: 0.2749
Neural Network Model Accuracy: 0.8726
```

15 Neural Network Model Summary

```
- The Neural Network model achieved an accuracy of 87.26%, indicating a strong ability to predi
- The classification report provides details on precision, recall, and F1-score for identifyin
- The model is effective at identifying non-food deserts but requires further refinement to im
```

Insights from the Multi-Layer Perceptron (MLP) Neural Network Model

```
Neural Network Architecture:
    Input layer with 5 features (Poverty Rate, Median Income, SNAP Usage, Housing Units Withou
    Two hidden layers:
        First layer (32 neurons, ReLU activation)
        Second layer (16 neurons, ReLU activation)
        Dropout (20%) applied to prevent overfitting
    Output layer with a sigmoid activation function for binary classification.

Performance Metrics:
    Achieved accuracy: 87.26%% on test data.
    Loss Function: Binary cross-entropy optimized using Adam optimizer.
    Overfitting was minimized using dropout layers during training.

Strengths of the Model:
    Captures nonlinear relationships between features.
    Learns complex interactions between socio-economic indicators affecting food accessibility
    Scales well with large datasets, making it suitable for food desert classification.

Challenges & Limitations:
    Dependent on hyperparameter tuning (e.g., number of layers, neurons, dropout rates).
    Computationally expensive compared to traditional models like Random Forest.
    Potential class imbalance issues affecting recall for food desert areas.

Potential Future Improvements:
    Increase training epochs and fine-tune learning rate.
    Introduce more hidden layers or use a deeper architecture.
```

Apply class weighting or SMOTE to improve recall for food desert predictions.
Experiment with alternative activation functions or better learning dynamics.

16 Gradient Boosting Model

[59]:
```
# Re-import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report

# Reload the dataset
food_access_df = pd.read_csv("FoodAccessResearchAtlasData2019.csv")

# Select key features and target variable
features = ["PovertyRate", "MedianFamilyIncome", "TractSNAP", "TractHUNV",␣
↪"Urban"]
target = "LILATracts_1And10"

# Drop any remaining missing values in selected features and target
food_access_model_data = food_access_df[features + [target]].dropna()

X = food_access_model_data[features]
y = food_access_model_data[target]

# Normalize the feature values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
↪random_state=42)

# Initialize and train the XGBoost model
xgb_model = XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=6,␣
↪random_state=42, use_label_encoder=False, eval_metric="logloss")
xgb_model.fit(X_train, y_train)

# Make predictions
y_pred_xgb = xgb_model.predict(X_test)

# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
```

```python
classification_rep_xgb = classification_report(y_test, y_pred_xgb,␣
 ↪output_dict=True)

# Display the classification report
pd.DataFrame(classification_rep_xgb), print(f"XGBoost Model Accuracy:␣
 ↪{accuracy_xgb:.4f}") # Display the new model accuracy
```

```
XGBoost Model Accuracy: 0.8739
```

[59]:
```
(                      0            1  accuracy     macro avg  weighted avg
 precision      0.880864     0.536332  0.873929      0.708598      0.836925
 recall         0.989302     0.084653  0.873929      0.536978      0.873929
 f1-score       0.931940     0.146226  0.873929      0.539083      0.831735
 support    12526.000000  1831.000000  0.873929  14357.000000  14357.000000,
 None)
```

17 Insights

```
- The XGBoost model achieved an accuracy of 87.39%, indicating a strong ability to predict foo
- The classification report provides details on precision, recall, and F1-score for identifyin
- The model is effective at identifying non-food deserts but requires further refinement to im
```

Model Performance Overview:

```
Achieved Accuracy: 87.39%, indicating strong classification performance.
Precision for Non-Food Deserts (Class 0): 88.1%, meaning most non-food desert areas were corre
Recall for Non-Food Deserts: 98.9%, confirming the model effectively identifies areas with foo
Precision for Food Deserts (Class 1): 53.6%, showing moderate ability to classify food deserts
Recall for Food Deserts: 8.5%, indicating the model struggles to detect actual food desert are
```

Feature Importance Findings:

```
Poverty Rate & SNAP Participation → Strongest predictors of food deserts.
Median Family Income → Contributes significantly to classification.
Urban/Rural Classification & Housing Without Vehicles → Less impact than economic factors.
```

Strengths of the XGBoost Model:

```
Handles nonlinear relationships better than Random Forest.
Boosting technique improves performance by iteratively correcting errors.
High precision in detecting non-food desert areas.
```

Challenges & Limitations:

```
Low recall for food deserts (8.5%), meaning many true food desert areas are missed.
Potential class imbalance issue, with significantly more non-food desert tracts.
Computationally intensive compared to simpler models.
```